



CryptoAuthLib

v3.7.9

1 CryptoAuthLib - Microchip CryptoAuthentication Library	1
1.1 Introduction	1
1.2 Examples	2
1.3 Configuration	2
1.4 Release notes	3
1.5 Host Device Support	3
1.6 CryptoAuthLib Architecture	3
1.7 Directory Structure	3
1.8 Tests	3
1.9 Using CryptoAuthLib (Microchip CryptoAuth Library)	4
1.9.1 Incorporating CryptoAuthLib in a Linux project using USB HID devices	4
1.10 Updating X.509 compressed certificate format to support extended years	4
2 License	5
3 IP Protection with Symmetric Authentication	7
3.1 User Considerations	7
3.2 Examples	7
4 PKCS11 Application Information	9
4.1 Setting up cryptoauthlib as a PKCS11 Provider for your system (LINUX)	9
4.1.1 Update libp11 on the system. The version should be at minimum 0.4.10	9
4.1.2 Build and Install cryptoauthlib with PKCS11 support	10
4.1.3 Configuring the cryptoauthlib PKCS11 library	10
4.1.4 Using p11-kit-proxy	11
4.1.5 Without using p11-kit-proxy	11
4.1.6 Testing	12
5 Application Support	13
6 Secure boot using ATECC608	15
6.1 Implementation Considerations	16
6.2 Examples	16
7 Contribution Guidelines	17
8 Migrating to v3.7.6 to v3.7.7	19
8.1 atcacert	19
8.1.1 atcacert API Migration	19
9 openssl directory - Purpose	21
10 atcab	23
10.1 atcab API reference	23
11 Python CryptoAuthLib module	25

11.1 Introduction	25
11.1.1 Code Examples	25
11.2 Installation	25
11.2.1 CryptoAuthLib python module can be installed through Python's pip tool:	25
11.2.2 To upgrade your installation when new releases are made:	25
11.2.3 If you ever need to remove your installation:	25
11.3 What does python CryptoAuthLib package do?	25
11.4 Supported hardware	26
11.5 Supported devices	26
11.6 Using cryptoauthlib python module	26
11.7 In Summary	27
11.7.1 Step I: Import the module	27
11.7.2 Step II: Initilize the module	27
11.7.3 Step III: Use Cryptoauthlib APIs	27
11.8 Code portability	27
11.9 Cryptoauthlib module API documentation	27
11.9.1 help() command	27
11.9.2 dir() command	27
11.10 Code Examples	27
11.11 Tests	28
11.12 Release notes	28
12 Python CryptoAuthLib Module Testing	29
12.1 Introduction	29
12.1.1 Running	29
12.1.2 Test options	29
13 Microchip Cryptoauthlib Release Notes	31
13.1 Release v3.7.9 (09/05/2025)	31
13.1.1 New Features	31
13.1.2 Fixes	31
13.2 Release v3.7.8 (05/05/2025)	32
13.2.1 Fixes	32
13.3 Release v3.7.7 (02/07/2025)	33
13.3.1 New Features	33
13.3.2 Fixes	33
13.3.3 API CHANGES	33
13.4 Release v3.7.6 (09/26/2024)	33
13.4.1 New Features	33
13.4.2 Fixes	34
13.5 Release v3.7.5 (06/26/2024)	34
13.5.1 New Features	34
13.5.2 Fixes	34

13.5.3 API Changes	34
13.6 Release v3.7.4 (03/08/2024)	35
13.6.1 New Features	35
13.6.2 Fixes	35
13.7 Release v3.7.3 (01/31/2024)	35
13.7.1 New Features	35
13.7.2 Fixes	35
13.8 Release v3.7.2 (01/19/2024)	35
13.8.1 New Features	35
13.8.2 Fixes	36
13.8.3 API Changes	36
13.9 Release v3.7.1 (12/15/2023)	36
13.9.1 New Features	36
13.9.2 Fixes	36
13.9.3 API Changes	36
13.10 Release v3.7.0 (09/08/2023)	36
13.10.1 New Features	36
13.10.2 Fixes	37
13.10.3 API Changes	37
13.11 Release v3.6.1 (07/14/2023)	37
13.11.1 New Features	37
13.11.2 Fixes	37
13.12 Release v3.6.0 (04/04/2023)	37
13.12.1 New Features	37
13.12.2 Fixes	38
13.12.3 API Changes	38
13.13 Release v3.5.1 (03/26/2023)	38
13.13.1 New Features	38
13.14 Release v3.5.0 (03/14/2023)	38
13.14.1 New Features	38
13.15 Release v3.4.3 (12/23/2022)	38
13.15.1 New Features	38
13.15.2 Fixes	38
13.16 Release v3.4.2 (12/04/2022)	39
13.16.1 Fixes	39
13.17 Release v3.4.1 (11/11/2022)	39
13.17.1 Fixes	39
13.18 Release v3.4.0 (10/27/2022)	39
13.18.1 New Features	39
13.18.2 Fixes	40
13.19 Release v3.3.3 (10/06/2021)	40
13.19.1 New features	40

13.19.2 Fixes	40
13.20 Release v3.3.2 (06/20/2021)	40
13.20.1 New features	40
13.20.2 Fixes	41
13.21 Release v3.3.1 (04/23/2021)	41
13.21.1 New features	41
13.21.2 Fixes	41
13.22 Release v3.3.0 (01/22/2021)	41
13.22.1 API Updates	41
13.22.2 New features	42
13.22.3 Fixes	42
13.23 Release v3.2.5 (11/30/2020)	42
13.23.1 New features	42
13.23.2 Fixes	42
13.24 Release v3.2.4 (10/17/2020)	42
13.24.1 New features	42
13.24.2 Fixes	43
13.25 Release v3.2.3 (09/12/2020)	43
13.25.1 New features	43
13.25.2 Fixes	43
13.26 Release v3.2.2 (07/28/2020)	43
13.26.1 New Features	43
13.26.2 Fixes	43
13.27 Release v3.2.1 (06/29/2020)	43
13.27.1 Fixes	43
13.28 Release v3.2.0 (06/10/2020)	44
13.28.1 New features	44
13.28.2 Known issues	44
13.29 Release v3.1.1 (03/06/2020)	44
13.30 Release v3.1.0 (02/05/2020)	44
13.31 Release 11/22/2019	45
13.32 Release 08/30/2019	45
13.33 Release 05/17/2019	45
13.34 Release 03/04/2019	45
13.35 Release 01/25/2019	45
13.36 Release 01/04/2019	45
13.37 Release 10/25/2018	46
13.38 Release 08/17/2018	46
13.39 Release 07/25/2018	46
13.40 Release 07/18/2018	46
13.41 Release 03/29/2018	46
13.42 Release 01/15/2018	47

13.43 Release 11/22/2017	47
13.44 Release 11/17/2017	47
13.45 Release 07/01/2017	47
13.46 Release 01/08/2016	48
13.47 Release 9/19/2015	49
14 Security Policy	51
14.1 Supported Versions	51
14.2 Reporting a Vulnerability	51
15 Deprecated List	53
16 Module Index	55
16.1 Modules	55
17 Namespace Index	57
17.1 Namespace List	57
18 Hierarchical Index	59
18.1 Class Hierarchy	59
19 Data Structure Index	65
19.1 Data Structures	65
20 File Index	71
20.1 File List	71
21 Module Documentation	81
21.1 TNG API (tng_)	81
21.1.1 Detailed Description	82
21.1.2 Function Documentation	83
21.2 Basic Crypto API methods (atcab_)	88
21.2.1 Detailed Description	97
21.2.2 Function Documentation	97
21.3 Configuration (cfg_)	171
21.4 ATCADevice (atca_)	171
21.4.1 Detailed Description	172
21.4.2 Function Documentation	172
21.5 ATCAIface (atca_)	174
21.5.1 Detailed Description	176
21.5.2 Enumeration Type Documentation	176
21.5.3 Function Documentation	176
21.6 Certificate manipulation methods (atcacert_)	182
21.6.1 Detailed Description	186
21.6.2 Macro Definition Documentation	186

21.6.3 Typedef Documentation	188
21.6.4 Enumeration Type Documentation	189
21.6.5 Function Documentation	192
21.7 Basic Crypto API methods for CryptoAuth Devices (calib_)	202
21.7.1 Detailed Description	207
21.7.2 Function Documentation	207
21.8 Software crypto methods (atcac_)	214
21.8.1 Detailed Description	214
21.9 Hardware abstraction layer (hal_)	214
21.9.1 Cryptoauthlib HAL Architecture	214
21.9.2 CryptoAuthLib Supported HAL Layers	215
21.9.3 Detailed Description	221
21.9.4 Macro Definition Documentation	221
21.9.5 Function Documentation	222
21.10 Host side crypto methods (atcah_)	253
21.10.1 Detailed Description	258
21.11 JSON Web Token (JWT) methods (atca_jwt_)	258
21.12 mbedTLS Wrapper methods (atca_mbedtls_)	258
21.12.1 Detailed Description	259
21.12.2 Typedef Documentation	259
21.12.3 Function Documentation	259
21.13 Attributes (pkcs11_attrib_)	260
21.13.1 Detailed Description	269
21.13.2 Function Documentation	269
21.13.3 Variable Documentation	271
22 Namespace Documentation	275
22.1 cryptoauthlib Namespace Reference	275
22.1.1 Detailed Description	275
22.2 cryptoauthlib.atcab Namespace Reference	275
22.2.1 Detailed Description	278
22.2.2 Function Documentation	278
22.3 cryptoauthlib.atcacert Namespace Reference	325
22.3.1 Detailed Description	325
22.3.2 Function Documentation	326
22.4 cryptoauthlib.atcaenum Namespace Reference	330
22.4.1 Detailed Description	331
22.5 cryptoauthlib.atjwt Namespace Reference	331
22.5.1 Detailed Description	331
22.6 cryptoauthlib.device Namespace Reference	331
22.6.1 Detailed Description	331
22.7 cryptoauthlib.exceptions Namespace Reference	332

22.7.1 Detailed Description	332
22.8 cryptoauthlib.iface Namespace Reference	333
22.8.1 Detailed Description	333
22.8.2 Function Documentation	333
22.9 cryptoauthlib.library Namespace Reference	335
22.9.1 Detailed Description	336
22.9.2 Function Documentation	336
22.10 cryptoauthlib.sha206_api Namespace Reference	342
22.10.1 Detailed Description	342
22.10.2 Function Documentation	342
22.11 cryptoauthlib.status Namespace Reference	347
22.11.1 Detailed Description	347
22.11.2 Function Documentation	347
22.12 cryptoauthlib.tng Namespace Reference	348
22.12.1 Detailed Description	348
22.12.2 Function Documentation	348
22.13 test_device Namespace Reference	352
22.13.1 Detailed Description	352
22.13.2 Variable Documentation	352
22.14 test_iface Namespace Reference	354
22.14.1 Detailed Description	354
23 Data Structure Documentation	355
23.1 _ascii_kit_host_context Struct Reference	355
23.2 cryptoauthlib.iface._ATCACUSTOM Class Reference	355
23.2.1 Detailed Description	356
23.2.2 Field Documentation	356
23.3 cryptoauthlib.iface._ATCAHID Class Reference	356
23.3.1 Detailed Description	357
23.3.2 Field Documentation	357
23.4 cryptoauthlib.iface._ATCAI2C Class Reference	357
23.4.1 Detailed Description	358
23.4.2 Field Documentation	358
23.5 cryptoauthlib.iface._ATCAIfaceParams Class Reference	359
23.5.1 Detailed Description	359
23.5.2 Field Documentation	359
23.6 cryptoauthlib.iface._ATCAKIT Class Reference	360
23.6.1 Detailed Description	360
23.6.2 Field Documentation	360
23.7 cryptoauthlib.iface._ATCASPI Class Reference	361
23.7.1 Detailed Description	361
23.7.2 Field Documentation	361

23.8 cryptoauthlib.iface._ATCASWI Class Reference	362
23.8.1 Detailed Description	362
23.8.2 Field Documentation	362
23.9 cryptoauthlib.iface._ATCAUART Class Reference	363
23.9.1 Detailed Description	363
23.9.2 Field Documentation	363
23.10 cryptoauthlib.library._CtypeIterator Class Reference	364
23.10.1 Detailed Description	364
23.11 _kit_host_map_entry Struct Reference	364
23.11.1 Detailed Description	364
23.12 cryptoauthlib.iface._U_Address Class Reference	365
23.12.1 Detailed Description	365
23.12.2 Field Documentation	365
23.13 cryptoauthlib.device.AesEnable Class Reference	366
23.13.1 Detailed Description	366
23.13.2 Field Documentation	366
23.14 cryptoauthlib.exceptions.AssertionFailure Class Reference	366
23.14.1 Detailed Description	366
23.15 cryptoauthlib.atcab.atca_aes_cbc_ctx Class Reference	367
23.15.1 Detailed Description	367
23.15.2 Field Documentation	367
23.16 cryptoauthlib.atcab.atca_aes_cbcmac_ctx Class Reference	367
23.16.1 Detailed Description	368
23.16.2 Field Documentation	368
23.17 cryptoauthlib.atcab.atca_aes_ccm_ctx Class Reference	368
23.17.1 Detailed Description	368
23.17.2 Field Documentation	368
23.18 cryptoauthlib.atcab.atca_aes_cmac_ctx Class Reference	369
23.18.1 Detailed Description	369
23.18.2 Field Documentation	369
23.19 cryptoauthlib.atcab.atca_aes_ctr_ctx Class Reference	370
23.19.1 Detailed Description	370
23.19.2 Field Documentation	370
23.20 cryptoauthlib.atcab.atca_aes_gcm_ctx Class Reference	370
23.20.1 Detailed Description	371
23.20.2 Field Documentation	371
23.21 atca_check_mac_in_out Struct Reference	371
23.21.1 Detailed Description	372
23.21.2 Field Documentation	372
23.22 atca_decrypt_in_out Struct Reference	372
23.22.1 Detailed Description	372
23.23 atca_delete_in_out Struct Reference	372

23.23.1 Detailed Description	373
23.24 atca_derive_key_in_out Struct Reference	373
23.24.1 Detailed Description	373
23.25 atca_derive_key_mac_in_out Struct Reference	373
23.25.1 Detailed Description	374
23.26 atca_device Struct Reference	374
23.26.1 Detailed Description	374
23.26.2 Field Documentation	374
23.27 atca_diversified_key_in_out Struct Reference	375
23.27.1 Detailed Description	375
23.28 atca_evp_ctx Struct Reference	375
23.29 atca_gen_dig_in_out Struct Reference	375
23.29.1 Detailed Description	376
23.30 atca_gen_key_in_out Struct Reference	376
23.30.1 Detailed Description	377
23.31 atca_hal_kit_phy_t Struct Reference	377
23.31.1 Field Documentation	377
23.32 atca_hal_list_entry_t Struct Reference	378
23.32.1 Detailed Description	378
23.32.2 Field Documentation	378
23.33 atca_hal_shm_t Struct Reference	378
23.34 atca_hmac_in_out Struct Reference	378
23.34.1 Detailed Description	379
23.35 cryptoauthlib.atcab.atca_hmac_sha256_ctx Class Reference	379
23.35.1 Detailed Description	379
23.36 atca_i2c_host_s Struct Reference	380
23.37 atca_iface Struct Reference	380
23.37.1 Detailed Description	380
23.37.2 Field Documentation	380
23.38 atca_include_data_in_out Struct Reference	381
23.38.1 Detailed Description	381
23.39 atca_io_decrypt_in_out Struct Reference	381
23.40 atca_mac_in_out Struct Reference	381
23.40.1 Detailed Description	382
23.41 atca_mbedtls_eckey_s Struct Reference	382
23.41.1 Detailed Description	382
23.42 atca_nonce_in_out Struct Reference	382
23.42.1 Detailed Description	383
23.43 atca_resp_mac_in_out Struct Reference	383
23.43.1 Detailed Description	383
23.44 atca_secureboot_enc_in_out Struct Reference	384
23.45 atca_secureboot_mac_in_out Struct Reference	384

23.46 atca_session_key_in_out Struct Reference	384
23.46.1 Detailed Description	385
23.47 atca_sha256_ctx Struct Reference	385
23.48 cryptoauthlib.atcab.atca_sha256_ctx Class Reference	385
23.48.1 Detailed Description	385
23.48.2 Field Documentation	385
23.49 atca_sign_internal_in_out Struct Reference	386
23.49.1 Detailed Description	386
23.50 atca_spi_host_s Struct Reference	386
23.51 atca_temp_key Struct Reference	387
23.51.1 Detailed Description	387
23.52 atca_uart_host_s Struct Reference	387
23.53 atca_verify_in_out Struct Reference	387
23.53.1 Detailed Description	388
23.54 atca_verify_mac Struct Reference	388
23.55 atca_write_mac_in_out Struct Reference	388
23.55.1 Detailed Description	389
23.56 cryptoauthlib_mock.atcab_mock Class Reference	389
23.57 atcac_aes_cmac_ctx Struct Reference	393
23.58 atcac_aes_gcm_ctx Struct Reference	393
23.59 atcac_hmac_ctx Struct Reference	394
23.60 atcac_pk_ctx Struct Reference	394
23.61 atcac_sha1_ctx Struct Reference	394
23.62 atcac_sha2_256_ctx Struct Reference	394
23.63 atcac_sha2_384_ctx Struct Reference	394
23.64 atcac_sha2_512_ctx Struct Reference	394
23.65 atcac_x509_ctx Struct Reference	395
23.66 atcacert_build_state_s Struct Reference	395
23.66.1 Detailed Description	395
23.67 atcacert_cert_element_s Struct Reference	395
23.67.1 Detailed Description	396
23.68 cryptoauthlib.atcacert.atcacert_cert_element_t Class Reference	396
23.68.1 Detailed Description	397
23.68.2 Field Documentation	397
23.69 atcacert_cert_loc_s Struct Reference	397
23.69.1 Detailed Description	397
23.70 cryptoauthlib.atcacert.atcacert_cert_loc_t Class Reference	397
23.70.1 Detailed Description	398
23.71 cryptoauthlib.atcacert.atcacert_cert_sn_src_t Class Reference	398
23.71.1 Detailed Description	399
23.72 cryptoauthlib.atcacert.atcacert_cert_type_t Class Reference	399
23.72.1 Detailed Description	400

23.73	cryptoauthlib.atcacert.atcacert_comp_data_t Class Reference	400
23.73.1	Detailed Description	400
23.73.2	Field Documentation	401
23.74	cryptoauthlib.atcacert.atcacert_date_format_t Class Reference	401
23.74.1	Detailed Description	402
23.75	atcacert_def_s Struct Reference	402
23.75.1	Detailed Description	402
23.76	cryptoauthlib.atcacert.atcacert_def_t Class Reference	402
23.76.1	Detailed Description	403
23.77	atcacert_device_loc_s Struct Reference	403
23.77.1	Detailed Description	403
23.78	cryptoauthlib.atcacert.atcacert_device_loc_t Class Reference	404
23.78.1	Detailed Description	404
23.78.2	Field Documentation	404
23.79	cryptoauthlib.atcacert.atcacert_device_zone_t Class Reference	405
23.79.1	Detailed Description	405
23.80	cryptoauthlib.atcacert.atcacert_std_cert_element_t Class Reference	405
23.80.1	Detailed Description	406
23.81	atcacert_tm_utc_s Struct Reference	406
23.81.1	Detailed Description	406
23.82	cryptoauthlib.atcacert.atcacert_tm_utc_t Class Reference	407
23.82.1	Detailed Description	407
23.82.2	Field Documentation	407
23.83	cryptoauthlib.atcacert.atcacert_transform_t Class Reference	407
23.83.1	Detailed Description	408
23.84	cryptoauthlib.iface.ATCADeviceType Class Reference	408
23.84.1	Detailed Description	409
23.85	cryptoauthlib.atcaenum.AtcaEnum Class Reference	410
23.85.1	Detailed Description	410
23.86	ATCAHAL_t Struct Reference	411
23.86.1	Detailed Description	411
23.87	atcal2Cmaster Struct Reference	411
23.87.1	Detailed Description	411
23.88	ATCAIfaceCfg Struct Reference	412
23.88.1	Field Documentation	413
23.89	cryptoauthlib.iface.ATCAIfaceCfg Class Reference	413
23.89.1	Detailed Description	414
23.89.2	Field Documentation	414
23.90	cryptoauthlib.iface.ATCAIfaceType Class Reference	414
23.90.1	Detailed Description	415
23.91	cryptoauthlib.iface.ATCAKitType Class Reference	415
23.91.1	Detailed Description	416

23.92 ATCAPacket Struct Reference	416
23.93 cryptoauthlib.library.AtcaReference Class Reference	416
23.93.1 Detailed Description	417
23.94 cryptoauthlib.library.AtcaStructure Class Reference	417
23.94.1 Detailed Description	418
23.94.2 Member Function Documentation	418
23.95 atcaSWImaster Struct Reference	418
23.95.1 Detailed Description	418
23.96 cryptoauthlib.library.AtcaUnion Class Reference	419
23.96.1 Detailed Description	419
23.96.2 Member Function Documentation	419
23.97 atecc508a_config_s Struct Reference	420
23.98 cryptoauthlib.device.Atecc508aConfig Class Reference	420
23.98.1 Detailed Description	421
23.98.2 Field Documentation	421
23.99 atecc608_config_s Struct Reference	422
23.100 cryptoauthlib.device.Atecc608Config Class Reference	422
23.100.1 Detailed Description	423
23.100.2 Field Documentation	423
23.101 atsha204a_config_s Struct Reference	424
23.102 cryptoauthlib.device.Atsha204aConfig Class Reference	424
23.102.1 Detailed Description	425
23.102.2 Field Documentation	425
23.103 cryptoauthlib.exceptions.BadArgumentError Class Reference	425
23.103.1 Detailed Description	425
23.104 cryptoauthlib.exceptions.BadCrcError Class Reference	426
23.104.1 Detailed Description	426
23.105 cryptoauthlib.exceptions.BadOpcodeError Class Reference	426
23.105.1 Detailed Description	426
23.106 setup.BinaryDistribution Class Reference	426
23.107 cryptoauthlib.library.cal_buffer Class Reference	427
23.107.1 Field Documentation	427
23.108 cal_buffer_s Struct Reference	427
23.108.1 Field Documentation	428
23.109 calib_packet_cache_s Struct Reference	428
23.110 cryptoauthlib.atcacert.CertStatus Class Reference	428
23.110.1 Detailed Description	429
23.111 cryptoauthlib.exceptions.CheckmacVerifyFailedError Class Reference	429
23.111.1 Detailed Description	429
23.112 cryptoauthlib.device.ChipMode508 Class Reference	429
23.112.1 Detailed Description	430
23.112.2 Field Documentation	430

23.113 cryptoauthlib.device.ChipMode608 Class Reference	430
23.113.1 Detailed Description	430
23.113.2 Field Documentation	430
23.114 cryptoauthlib.device.ChipOptions Class Reference	431
23.114.1 Detailed Description	431
23.114.2 Field Documentation	431
23.115 CK_AES_CBC_ENCRYPT_DATA_PARAMS Struct Reference	432
23.116 CK_AES_CCM_PARAMS Struct Reference	432
23.117 CK_AES_CTR_PARAMS Struct Reference	432
23.118 CK_AES_GCM_PARAMS Struct Reference	432
23.119 CK_ARIA_CBC_ENCRYPT_DATA_PARAMS Struct Reference	432
23.120 CK_ATTRIBUTE Struct Reference	433
23.121 CK_C_INITIALIZE_ARGS Struct Reference	433
23.122 CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS Struct Reference	433
23.123 CK_CAMELLIA_CTR_PARAMS Struct Reference	433
23.124 CK_CCM_PARAMS Struct Reference	433
23.125 CK_CMS_SIG_PARAMS Struct Reference	434
23.126 CK_DATE Struct Reference	434
23.127 CK_DES_CBC_ENCRYPT_DATA_PARAMS Struct Reference	434
23.128 CK_DSA_PARAMETER_GEN_PARAM Struct Reference	434
23.129 CK_ECDH1_DERIVE_PARAMS Struct Reference	434
23.130 CK_ECDH2_DERIVE_PARAMS Struct Reference	435
23.131 CK_ECDH_AES_KEY_WRAP_PARAMS Struct Reference	435
23.132 CK_ECMQV_DERIVE_PARAMS Struct Reference	435
23.133 CK_FUNCTION_LIST Struct Reference	435
23.134 CK_GCM_PARAMS Struct Reference	436
23.135 CK_GOSTR3410_DERIVE_PARAMS Struct Reference	436
23.136 CK_GOSTR3410_KEY_WRAP_PARAMS Struct Reference	436
23.137 CK_INFO Struct Reference	436
23.138 CK_KEA_DERIVE_PARAMS Struct Reference	437
23.139 CK_KEY_DERIVATION_STRING_DATA Struct Reference	437
23.140 CK_KEY_WRAP_SET_OAEP_PARAMS Struct Reference	437
23.141 CK_KIP_PARAMS Struct Reference	437
23.142 CK_MECHANISM Struct Reference	437
23.143 CK_MECHANISM_INFO Struct Reference	438
23.144 CK_OTP_PARAM Struct Reference	438
23.145 CK_OTP_PARAMS Struct Reference	438
23.146 CK_OTP_SIGNATURE_INFO Struct Reference	438
23.147 CK_PBE_PARAMS Struct Reference	438
23.148 CK_PKCS5_PBKD2_PARAMS Struct Reference	439
23.149 CK_PKCS5_PBKD2_PARAMS2 Struct Reference	439
23.150 CK_RC2_CBC_PARAMS Struct Reference	439

23.151 CK_RC2_MAC_GENERAL_PARAMS Struct Reference	439
23.152 CK_RC5_CBC_PARAMS Struct Reference	440
23.153 CK_RC5_MAC_GENERAL_PARAMS Struct Reference	440
23.154 CK_RC5_PARAMS Struct Reference	440
23.155 CK_RSA_AES_KEY_WRAP_PARAMS Struct Reference	440
23.156 CK_RSA_PKCS_OAEP_PARAMS Struct Reference	440
23.157 CK_RSA_PKCS_PSS_PARAMS Struct Reference	441
23.158 CK_SEED_CBC_ENCRYPT_DATA_PARAMS Struct Reference	441
23.159 CK_SESSION_INFO Struct Reference	441
23.160 CK_SKIPJACK_PRIVATE_WRAP_PARAMS Struct Reference	441
23.161 CK_SKIPJACK_RELAYX_PARAMS Struct Reference	442
23.162 CK_SLOT_INFO Struct Reference	442
23.163 CK_SSL3_KEY_MAT_OUT Struct Reference	442
23.164 CK_SSL3_KEY_MAT_PARAMS Struct Reference	442
23.165 CK_SSL3_MASTER_KEY_DERIVE_PARAMS Struct Reference	443
23.166 CK_SSL3_RANDOM_DATA Struct Reference	443
23.167 CK_TLS12_KEY_MAT_PARAMS Struct Reference	443
23.168 CK_TLS12_MASTER_KEY_DERIVE_PARAMS Struct Reference	443
23.169 CK_TLS_KDF_PARAMS Struct Reference	443
23.170 CK_TLS_MAC_PARAMS Struct Reference	444
23.171 CK_TLS_PRF_PARAMS Struct Reference	444
23.172 CK_TOKEN_INFO Struct Reference	444
23.173 CK_VERSION Struct Reference	444
23.174 CK_WTLS_KEY_MAT_OUT Struct Reference	445
23.175 CK_WTLS_KEY_MAT_PARAMS Struct Reference	445
23.176 CK_WTLS_MASTER_KEY_DERIVE_PARAMS Struct Reference	445
23.177 CK_WTLS_PRF_PARAMS Struct Reference	445
23.178 CK_WTLS_RANDOM_DATA Struct Reference	445
23.179 CK_X9_42_DH1_DERIVE_PARAMS Struct Reference	446
23.180 CK_X9_42_DH2_DERIVE_PARAMS Struct Reference	446
23.181 CK_X9_42_MQV_DERIVE_PARAMS Struct Reference	446
23.182 CL_HashContext Struct Reference	446
23.183 cryptoauthlib.exceptions.CommunicationError Class Reference	447
23.183.1 Detailed Description	447
23.184 cryptoauthlib.exceptions.ConfigZoneLockedError Class Reference	447
23.184.1 Detailed Description	447
23.185 cryptoauthlib.device.Counter204 Class Reference	447
23.185.1 Detailed Description	448
23.185.2 Field Documentation	448
23.186 cryptoauthlib.device.CountMatch Class Reference	448
23.186.1 Detailed Description	448
23.186.2 Field Documentation	448

23.187 cryptoauthlib.exceptions.CrcError Class Reference	449
23.187.1 Detailed Description	449
23.188 setup.CryptoAuthCommandBuildExt Class Reference	449
23.189 setup.CryptoAuthCommandInstall Class Reference	449
23.190 cryptoauthlib.exceptions.CryptoError Class Reference	450
23.190.1 Detailed Description	450
23.191 cryptoauthlib.exceptions.DataZoneLockedError Class Reference	450
23.191.1 Detailed Description	451
23.192 device_execution_time_t Struct Reference	451
23.192.1 Detailed Description	451
23.193 devtype_names_t Struct Reference	451
23.194 cryptoauthlib.exceptions.EccFaultError Class Reference	451
23.194.1 Detailed Description	451
23.195 cryptoauthlib.exceptions.ExecutionError Class Reference	452
23.195.1 Detailed Description	452
23.196 cryptoauthlib.exceptions.FunctionError Class Reference	452
23.196.1 Detailed Description	452
23.197 cryptoauthlib.exceptions.GenericError Class Reference	452
23.197.1 Detailed Description	453
23.198 cryptoauthlib.exceptions.HealthTestError Class Reference	453
23.198.1 Detailed Description	453
23.199 cryptoauthlib.atjwt.HwEcAlgorithm Class Reference	453
23.199.1 Detailed Description	453
23.199.2 Member Function Documentation	454
23.200 cryptoauthlib.atjwt.HwHmacAlgorithm Class Reference	454
23.200.1 Detailed Description	454
23.200.2 Member Function Documentation	454
23.201 i2c_sam0_instance Struct Reference	455
23.202 i2c_sam_instance Struct Reference	455
23.203 i2c_start_instance Struct Reference	455
23.204 cryptoauthlib.device.I2cEnable Class Reference	456
23.204.1 Detailed Description	456
23.204.2 Field Documentation	456
23.205 cryptoauthlib.exceptions.InvalidIdentifierError Class Reference	456
23.205.1 Detailed Description	456
23.206 cryptoauthlib.exceptions.InvalidSizeError Class Reference	457
23.206.1 Detailed Description	457
23.207 cryptoauthlib.device.KeyConfig Class Reference	457
23.207.1 Detailed Description	457
23.207.2 Field Documentation	457
23.208 cryptoauthlib.exceptions.LibraryLoadError Class Reference	458
23.208.1 Detailed Description	458

23.209 cryptoauthlib.exceptions.LibraryMemoryError Class Reference	458
23.209.1 Detailed Description	458
23.210 cryptoauthlib.exceptions.LibraryNotInitialized Class Reference	459
23.210.1 Detailed Description	459
23.211 memory_parameters Struct Reference	459
23.212 cryptoauthlib.exceptions.NoDevicesFoundError Class Reference	459
23.212.1 Detailed Description	459
23.213 cryptoauthlib.exceptions.NoResponseError Class Reference	460
23.213.1 Detailed Description	460
23.214 cryptoauthlib.exceptions.NoUseFlagError Class Reference	460
23.214.1 Detailed Description	460
23.215 cryptoauthlib.exceptions.ParityError Class Reference	460
23.215.1 Detailed Description	461
23.216 cryptoauthlib.exceptions.ParseError Class Reference	461
23.216.1 Detailed Description	461
23.217 pkcs11_mech_table_e Struct Reference	461
23.218 pkcs11_attr_model_s Struct Reference	461
23.219 pkcs11_conf_filedata_s Struct Reference	461
23.220 pkcs11_dev_ctx Struct Reference	462
23.220.1 Detailed Description	462
23.221 pkcs11_dev_res Struct Reference	462
23.221.1 Detailed Description	462
23.222 pkcs11_dev_state Struct Reference	462
23.222.1 Detailed Description	462
23.222.2 Field Documentation	462
23.223 pkcs11_ecc_key_info_s Struct Reference	463
23.224 pkcs11_key_info_s Struct Reference	463
23.225 pkcs11_lib_ctx_s Struct Reference	463
23.225.1 Detailed Description	464
23.225.2 Field Documentation	464
23.226 pkcs11_object_cache_s Struct Reference	465
23.226.1 Field Documentation	465
23.227 pkcs11_object_s Struct Reference	465
23.227.1 Field Documentation	466
23.228 pkcs11_rsa_key_info_s Struct Reference	466
23.229 pkcs11_session_ctx_s Struct Reference	466
23.229.1 Detailed Description	467
23.230 pkcs11_session_mech_ctx_s Struct Reference	467
23.231 pkcs11_slot_ctx_s Struct Reference	467
23.231.1 Detailed Description	467
23.231.2 Field Documentation	468
23.232 cryptoauthlib.atjwt.PyJWT Class Reference	468

23.232.1 Detailed Description	468
23.233 cryptoauthlib.exceptions.ReceiveError Class Reference	468
23.233.1 Detailed Description	468
23.234 cryptoauthlib.exceptions.ReceiveTimeoutError Class Reference	469
23.234.1 Detailed Description	469
23.235 cryptoauthlib.exceptions.ResyncWithWakeupError Class Reference	469
23.235.1 Detailed Description	469
23.236 secure_boot_config_bits Struct Reference	469
23.237 secure_boot_parameters Struct Reference	470
23.238 cryptoauthlib.device.SecureBoot Class Reference	470
23.238.1 Detailed Description	470
23.238.2 Field Documentation	470
23.239 cryptoauthlib.device.SlotConfig Class Reference	471
23.239.1 Detailed Description	471
23.239.2 Field Documentation	471
23.240 cryptoauthlib.status.Status Class Reference	471
23.240.1 Detailed Description	473
23.241 cryptoauthlib.exceptions.StatusUnknownError Class Reference	473
23.241.1 Detailed Description	473
23.242 cryptoauthlib.exceptions.TimeOutError Class Reference	473
23.242.1 Detailed Description	473
23.243 cryptoauthlib.exceptions.TransmissionError Class Reference	473
23.243.1 Detailed Description	474
23.244 cryptoauthlib.exceptions.TransmissionTimeoutError Class Reference	474
23.244.1 Detailed Description	474
23.245 cryptoauthlib.exceptions.UnimplementedError Class Reference	474
23.245.1 Detailed Description	474
23.246 cryptoauthlib.exceptions.UnsupportedInterface Class Reference	474
23.246.1 Detailed Description	475
23.247 cryptoauthlib.device.UseLock Class Reference	475
23.247.1 Detailed Description	475
23.247.2 Field Documentation	475
23.248 cryptoauthlib.device.VolatileKeyPermission Class Reference	475
23.248.1 Detailed Description	476
23.248.2 Field Documentation	476
23.249 cryptoauthlib.exceptions.WakeFailedError Class Reference	476
23.249.1 Detailed Description	476
23.250 cryptoauthlib.device.X509Format Class Reference	476
23.250.1 Detailed Description	477
23.250.2 Field Documentation	477
23.251 cryptoauthlib.exceptions.ZoneNotLockedError Class Reference	477
23.251.1 Detailed Description	477

24 File Documentation	479
24.1 api_206a.c File Reference	479
24.1.1 Detailed Description	480
24.1.2 Function Documentation	480
24.2 api_206a.h File Reference	485
24.2.1 Detailed Description	486
24.2.2 Function Documentation	486
24.3 symmetric_authentication.c File Reference	492
24.3.1 Detailed Description	492
24.3.2 Function Documentation	492
24.4 symmetric_authentication.h File Reference	493
24.4.1 Detailed Description	493
24.4.2 Function Documentation	493
24.5 ascii_kit_host.c File Reference	494
24.5.1 Detailed Description	494
24.5.2 Function Documentation	494
24.6 ascii_kit_host.h File Reference	495
24.6.1 Detailed Description	496
24.6.2 Macro Definition Documentation	496
24.6.3 Typedef Documentation	497
24.6.4 Function Documentation	497
24.7 trust_pkcs11_config.c File Reference	498
24.7.1 Detailed Description	498
24.8 io_protection_key.h File Reference	498
24.8.1 Detailed Description	499
24.9 secure_boot.c File Reference	499
24.9.1 Detailed Description	499
24.9.2 Function Documentation	499
24.10 secure_boot.h File Reference	500
24.10.1 Detailed Description	501
24.10.2 Function Documentation	501
24.11 secure_boot_memory.h File Reference	502
24.11.1 Detailed Description	502
24.12 tflxtls_cert_def_4_device.c File Reference	502
24.12.1 Detailed Description	502
24.13 tflxtls_cert_def_4_device.h File Reference	503
24.13.1 Detailed Description	503
24.14 tng_atca.c File Reference	503
24.14.1 Detailed Description	503
24.15 tng_atca.h File Reference	504
24.15.1 Detailed Description	504
24.16 tng_atcacert_client.c File Reference	504

24.16.1 Detailed Description	504
24.17 tng_atcacert_client.h File Reference	505
24.17.1 Detailed Description	505
24.18 tng_root_cert.c File Reference	505
24.18.1 Detailed Description	506
24.19 tng_root_cert.h File Reference	506
24.19.1 Detailed Description	506
24.20 tnglora_cert_def_1_signer.c File Reference	506
24.20.1 Detailed Description	506
24.21 tnglora_cert_def_1_signer.h File Reference	507
24.21.1 Detailed Description	507
24.22 tnglora_cert_def_2_device.c File Reference	507
24.22.1 Detailed Description	507
24.23 tnglora_cert_def_2_device.h File Reference	507
24.23.1 Detailed Description	508
24.24 tnglora_cert_def_4_device.c File Reference	508
24.24.1 Detailed Description	508
24.25 tnglora_cert_def_4_device.h File Reference	508
24.25.1 Detailed Description	509
24.26 tngtls_cert_def_1_signer.c File Reference	509
24.26.1 Detailed Description	509
24.27 tngtls_cert_def_1_signer.h File Reference	509
24.27.1 Detailed Description	509
24.28 tngtls_cert_def_2_device.c File Reference	510
24.28.1 Detailed Description	510
24.29 tngtls_cert_def_2_device.h File Reference	510
24.29.1 Detailed Description	510
24.30 tngtls_cert_def_3_device.c File Reference	510
24.30.1 Detailed Description	511
24.31 tngtls_cert_def_3_device.h File Reference	511
24.31.1 Detailed Description	511
24.32 wpc_apis.c File Reference	511
24.32.1 Detailed Description	511
24.33 wpc_apis.h File Reference	512
24.33.1 Detailed Description	512
24.34 wpccert_client.c File Reference	513
24.34.1 Detailed Description	513
24.34.2 Function Documentation	513
24.35 wpccert_client.h File Reference	514
24.35.1 Detailed Description	514
24.35.2 Function Documentation	514
24.36 atca_basic.c File Reference	515

24.36.1 Detailed Description	523
24.37 atca_basic.h File Reference	523
24.37.1 Detailed Description	532
24.38 atca_cfgs.c File Reference	532
24.38.1 Detailed Description	532
24.39 atca_cfgs.h File Reference	532
24.39.1 Detailed Description	532
24.40 atca_compiler.h File Reference	533
24.40.1 Detailed Description	533
24.40.2 Macro Definition Documentation	533
24.41 atca_config_check.h File Reference	533
24.41.1 Detailed Description	535
24.41.2 Macro Definition Documentation	535
24.42 atca_debug.c File Reference	538
24.42.1 Detailed Description	538
24.43 atca_device.c File Reference	539
24.43.1 Detailed Description	539
24.44 atca_device.h File Reference	539
24.44.1 Detailed Description	540
24.45 atca_devtypes.h File Reference	540
24.45.1 Detailed Description	541
24.46 atca_helpers.c File Reference	541
24.46.1 Detailed Description	542
24.46.2 Function Documentation	542
24.47 atca_helpers.h File Reference	550
24.47.1 Detailed Description	551
24.48 atca_iface.c File Reference	551
24.48.1 Detailed Description	552
24.49 atca_iface.h File Reference	553
24.49.1 Detailed Description	556
24.49.2 Variable Documentation	556
24.50 atca_platform.h File Reference	556
24.50.1 Detailed Description	557
24.51 atca_status.h File Reference	557
24.51.1 Detailed Description	558
24.51.2 Macro Definition Documentation	558
24.52 atca_utils_sizes.c File Reference	563
24.52.1 Detailed Description	564
24.53 atca_version.h File Reference	564
24.53.1 Detailed Description	564
24.54 atcacert.h File Reference	564
24.54.1 Detailed Description	565

24.55 atcacert_check_config.h File Reference	565
24.55.1 Detailed Description	566
24.56 atcacert_client.c File Reference	566
24.56.1 Detailed Description	566
24.57 atcacert_client.h File Reference	566
24.57.1 Detailed Description	567
24.57.2 Function Documentation	567
24.58 atcacert_date.c File Reference	570
24.58.1 Detailed Description	571
24.59 atcacert_date.h File Reference	571
24.59.1 Detailed Description	573
24.60 atcacert_def.c File Reference	573
24.60.1 Detailed Description	573
24.61 atcacert_def.h File Reference	573
24.61.1 Detailed Description	575
24.62 atcacert_der.c File Reference	575
24.62.1 Detailed Description	576
24.63 atcacert_der.h File Reference	576
24.63.1 Detailed Description	576
24.64 atcacert_host_hw.c File Reference	577
24.64.1 Detailed Description	577
24.65 atcacert_host_hw.h File Reference	577
24.65.1 Detailed Description	577
24.66 atcacert_host_sw.c File Reference	577
24.66.1 Detailed Description	578
24.67 atcacert_host_sw.h File Reference	578
24.67.1 Detailed Description	578
24.68 atcacert_pem.c File Reference	578
24.68.1 Detailed Description	578
24.69 atcacert_pem.h File Reference	579
24.69.1 Detailed Description	579
24.69.2 Function Documentation	579
24.70 cal_buffer.c File Reference	582
24.70.1 Detailed Description	583
24.70.2 Function Documentation	583
24.71 cal_buffer.h File Reference	585
24.71.1 Detailed Description	586
24.71.2 Function Documentation	586
24.72 cal_internal.h File Reference	588
24.72.1 Detailed Description	588
24.73 calib_aes.c File Reference	588
24.73.1 Detailed Description	588

24.74 calib_aes_gcm.c File Reference	589
24.74.1 Detailed Description	589
24.75 calib_aes_gcm.h File Reference	589
24.75.1 Detailed Description	589
24.76 calib_basic.c File Reference	589
24.76.1 Detailed Description	590
24.77 calib_checkmac.c File Reference	590
24.77.1 Detailed Description	590
24.78 calib_command.c File Reference	591
24.78.1 Detailed Description	591
24.78.2 Function Documentation	591
24.79 calib_command.h File Reference	594
24.79.1 Detailed Description	612
24.79.2 Function Documentation	612
24.80 calib_config_check.h File Reference	615
24.80.1 Detailed Description	617
24.80.2 Macro Definition Documentation	617
24.81 calib_counter.c File Reference	620
24.81.1 Detailed Description	620
24.82 calib_delete.c File Reference	620
24.82.1 Detailed Description	620
24.83 calib_derivekey.c File Reference	621
24.83.1 Detailed Description	621
24.84 calib_device.h File Reference	621
24.84.1 Detailed Description	624
24.85 calib_ecdh.c File Reference	624
24.85.1 Detailed Description	625
24.86 calib_execution.c File Reference	625
24.86.1 Detailed Description	625
24.86.2 Function Documentation	625
24.87 calib_execution.h File Reference	626
24.87.1 Detailed Description	627
24.87.2 Function Documentation	627
24.88 calib_gendig.c File Reference	628
24.88.1 Detailed Description	628
24.89 calib_genkey.c File Reference	629
24.89.1 Detailed Description	629
24.90 calib_helpers.c File Reference	629
24.90.1 Detailed Description	629
24.91 calib_hmac.c File Reference	630
24.91.1 Detailed Description	630
24.92 calib_info.c File Reference	630

24.92.1 Detailed Description	631
24.93 calib_kdf.c File Reference	631
24.93.1 Detailed Description	631
24.94 calib_lock.c File Reference	631
24.94.1 Detailed Description	632
24.95 calib_mac.c File Reference	632
24.95.1 Detailed Description	632
24.96 calib_nonce.c File Reference	632
24.96.1 Detailed Description	633
24.97 calib_packet.c File Reference	633
24.97.1 Detailed Description	633
24.98 calib_packet.h File Reference	634
24.98.1 Detailed Description	634
24.99 calib_privwrite.c File Reference	634
24.99.1 Detailed Description	635
24.100 calib_random.c File Reference	635
24.100.1 Detailed Description	635
24.101 calib_read.c File Reference	635
24.101.1 Detailed Description	636
24.102 calib_secureboot.c File Reference	636
24.102.1 Detailed Description	636
24.103 calib_selftest.c File Reference	636
24.103.1 Detailed Description	637
24.104 calib_sha.c File Reference	637
24.104.1 Detailed Description	637
24.105 calib_sign.c File Reference	637
24.105.1 Detailed Description	638
24.106 calib_updateextra.c File Reference	638
24.106.1 Detailed Description	638
24.107 calib_verify.c File Reference	638
24.107.1 Detailed Description	639
24.108 calib_write.c File Reference	639
24.108.1 Detailed Description	639
24.109 atca_crypto_hw_aes.h File Reference	639
24.109.1 Detailed Description	640
24.110 atca_crypto_hw_aes_cbc.c File Reference	640
24.110.1 Detailed Description	640
24.111 atca_crypto_hw_aes_cbcmac.c File Reference	640
24.111.1 Detailed Description	640
24.112 atca_crypto_hw_aes_ccm.c File Reference	641
24.112.1 Detailed Description	641
24.113 atca_crypto_hw_aes_cmac.c File Reference	641

24.113.1 Detailed Description	641
24.114 atca_crypto_hw_aes_ctr.c File Reference	642
24.114.1 Detailed Description	642
24.115 atca_crypto_pad.c File Reference	642
24.115.1 Detailed Description	642
24.116 atca_crypto_pbkdf2.c File Reference	642
24.116.1 Detailed Description	643
24.117 atca_crypto_sw.h File Reference	643
24.117.1 Detailed Description	643
24.118 atca_crypto_sw_aes_cmac.c File Reference	643
24.118.1 Detailed Description	644
24.119 atca_crypto_sw_aes_gcm.c File Reference	644
24.119.1 Detailed Description	644
24.120 atca_crypto_sw_sha1.c File Reference	644
24.120.1 Detailed Description	644
24.121 atca_crypto_sw_sha1.h File Reference	645
24.121.1 Detailed Description	645
24.122 atca_crypto_sw_sha2.c File Reference	645
24.122.1 Detailed Description	645
24.123 atca_crypto_sw_sha2.h File Reference	645
24.123.1 Detailed Description	646
24.124 crypto_hw_config_check.h File Reference	646
24.124.1 Detailed Description	646
24.124.2 Macro Definition Documentation	647
24.125 crypto_sw_config_check.h File Reference	648
24.125.1 Detailed Description	649
24.125.2 Macro Definition Documentation	649
24.126 sha1_routines.c File Reference	654
24.126.1 Detailed Description	654
24.127 sha1_routines.h File Reference	654
24.127.1 Detailed Description	655
24.128 sha2_routines.c File Reference	655
24.128.1 Detailed Description	655
24.129 sha2_routines.h File Reference	655
24.129.1 Detailed Description	656
24.130 cryptoauthlib.h File Reference	656
24.130.1 Detailed Description	658
24.130.2 Macro Definition Documentation	658
24.131 atca_hal.c File Reference	659
24.131.1 Detailed Description	659
24.132 atca_hal.h File Reference	659
24.132.1 Detailed Description	661

24.133	hal_all_platforms_kit_hidapi.c File Reference	661
24.133.1	Detailed Description	662
24.134	hal_freertos.c File Reference	662
24.134.1	Detailed Description	662
24.135	hal_gpio_harmony.c File Reference	662
24.135.1	Detailed Description	663
24.135.2	Function Documentation	663
24.136	hal_i2c_harmony.c File Reference	665
24.136.1	Detailed Description	665
24.137	hal_i2c_start.c File Reference	666
24.137.1	Detailed Description	666
24.138	hal_i2c_start.h File Reference	667
24.138.1	Detailed Description	667
24.139	hal_kit_bridge.c File Reference	667
24.139.1	Detailed Description	668
24.140	hal_kit_bridge.h File Reference	668
24.140.1	Detailed Description	668
24.141	hal_linux.c File Reference	669
24.141.1	Detailed Description	669
24.142	hal_linux_i2c_userspace.c File Reference	669
24.142.1	Detailed Description	670
24.143	hal_linux_uart_userspace.c File Reference	670
24.143.1	Detailed Description	671
24.143.2	Function Documentation	671
24.144	hal_sam0_i2c_asf.c File Reference	674
24.144.1	Detailed Description	674
24.145	hal_sam0_i2c_asf.h File Reference	675
24.145.1	Detailed Description	675
24.146	hal_sam_i2c_asf.c File Reference	675
24.146.1	Detailed Description	676
24.147	hal_sam_i2c_asf.h File Reference	676
24.147.1	Detailed Description	677
24.148	hal_sam_timer_asf.c File Reference	677
24.148.1	Detailed Description	677
24.149	hal_spi_harmony.c File Reference	678
24.149.1	Detailed Description	678
24.150	hal_swi_gpio.c File Reference	679
24.150.1	Detailed Description	679
24.150.2	Function Documentation	679
24.151	hal_swi_gpio.h File Reference	682
24.151.1	Detailed Description	683
24.151.2	Macro Definition Documentation	684

24.152	hal_swi_uart.c File Reference	684
24.152.1	Detailed Description	685
24.153	hal_timer_start.c File Reference	685
24.153.1	Detailed Description	686
24.154	hal_uart_harmony.c File Reference	686
24.154.1	Detailed Description	686
24.154.2	Function Documentation	687
24.154.3	Variable Documentation	689
24.155	hal_uc3_i2c_asf.c File Reference	690
24.155.1	Detailed Description	691
24.156	hal_uc3_i2c_asf.h File Reference	691
24.156.1	Detailed Description	691
24.157	hal_uc3_timer_asf.c File Reference	692
24.157.1	Detailed Description	692
24.158	hal_windows.c File Reference	692
24.158.1	Detailed Description	693
24.159	hal_windows_kit_uart.c File Reference	693
24.159.1	Detailed Description	694
24.159.2	Function Documentation	694
24.160	kit_protocol.c File Reference	697
24.160.1	Detailed Description	698
24.161	kit_protocol.h File Reference	698
24.161.1	Detailed Description	699
24.162	swi_uart_samd21_asf.c File Reference	699
24.162.1	Detailed Description	699
24.163	swi_uart_samd21_asf.h File Reference	700
24.163.1	Detailed Description	701
24.164	swi_uart_start.c File Reference	701
24.164.1	Detailed Description	701
24.165	swi_uart_start.h File Reference	702
24.165.1	Detailed Description	702
24.166	atca_host.c File Reference	703
24.166.1	Detailed Description	703
24.167	atca_host.h File Reference	703
24.167.1	Detailed Description	706
24.168	atca_host_config_check.h File Reference	706
24.168.1	Detailed Description	707
24.168.2	Macro Definition Documentation	707
24.169	atca_jwt.c File Reference	712
24.169.1	Detailed Description	713
24.170	atca_jwt.h File Reference	713
24.170.1	Detailed Description	713

24.171 atca_mbedtlsls_interface.h File Reference	713
24.171.1 Detailed Description	714
24.171.2 Macro Definition Documentation	714
24.172 atca_mbedtlsls_wrap.c File Reference	715
24.172.1 Detailed Description	717
24.172.2 Function Documentation	717
24.173 atca_openssl_interface.c File Reference	727
24.173.1 Detailed Description	729
24.173.2 Function Documentation	729
24.174 atca_openssl_interface.h File Reference	739
24.174.1 Detailed Description	740
24.174.2 Macro Definition Documentation	740
24.175 pkcs11_attrib.c File Reference	742
24.175.1 Detailed Description	742
24.176 pkcs11_attrib.h File Reference	742
24.176.1 Detailed Description	743
24.176.2 Typedef Documentation	743
24.177 pkcs11_cert.c File Reference	743
24.177.1 Detailed Description	744
24.178 pkcs11_cert.h File Reference	744
24.178.1 Detailed Description	745
24.179 pkcs11_config.c File Reference	745
24.179.1 Detailed Description	746
24.180 pkcs11_debug.c File Reference	746
24.180.1 Detailed Description	746
24.181 pkcs11_debug.h File Reference	747
24.181.1 Detailed Description	747
24.182 pkcs11_digest.h File Reference	747
24.182.1 Detailed Description	747
24.183 pkcs11_encrypt.c File Reference	748
24.183.1 Detailed Description	748
24.184 pkcs11_encrypt.h File Reference	748
24.184.1 Detailed Description	749
24.185 pkcs11_find.c File Reference	749
24.185.1 Detailed Description	750
24.186 pkcs11_find.h File Reference	750
24.186.1 Detailed Description	750
24.187 pkcs11_info.c File Reference	750
24.187.1 Detailed Description	751
24.188 pkcs11_info.h File Reference	751
24.188.1 Detailed Description	751
24.189 pkcs11_init.c File Reference	752

24.189.1 Detailed Description	752
24.190 pkcs11_init.h File Reference	752
24.190.1 Detailed Description	753
24.190.2 Typedef Documentation	753
24.191 pkcs11_key.c File Reference	754
24.191.1 Detailed Description	755
24.192 pkcs11_key.h File Reference	755
24.192.1 Detailed Description	757
24.193 pkcs11_main.c File Reference	757
24.193.1 Detailed Description	761
24.194 pkcs11_mech.c File Reference	761
24.194.1 Detailed Description	762
24.195 pkcs11_mech.h File Reference	762
24.195.1 Detailed Description	762
24.196 pkcs11_object.c File Reference	763
24.196.1 Detailed Description	764
24.197 pkcs11_object.h File Reference	764
24.197.1 Detailed Description	765
24.198 pkcs11_os.c File Reference	765
24.198.1 Detailed Description	766
24.199 pkcs11_os.h File Reference	766
24.199.1 Detailed Description	766
24.200 pkcs11_session.c File Reference	767
24.200.1 Detailed Description	767
24.201 pkcs11_session.h File Reference	767
24.201.1 Detailed Description	768
24.201.2 Typedef Documentation	768
24.202 pkcs11_signature.c File Reference	769
24.202.1 Detailed Description	769
24.203 pkcs11_signature.h File Reference	770
24.203.1 Detailed Description	770
24.204 pkcs11_slot.c File Reference	770
24.204.1 Detailed Description	771
24.205 pkcs11_slot.h File Reference	771
24.205.1 Detailed Description	772
24.205.2 Typedef Documentation	772
24.206 pkcs11_token.c File Reference	772
24.206.1 Detailed Description	773
24.207 pkcs11_token.h File Reference	773
24.207.1 Detailed Description	774
24.208 pkcs11_util.c File Reference	774
24.208.1 Detailed Description	774

24.209 pkcs11_util.h File Reference	775
24.209.1 Detailed Description	775
24.210 atca_wolfssl_interface.c File Reference	775
24.210.1 Detailed Description	775
24.211 atca_wolfssl_interface.h File Reference	775
24.211.1 Detailed Description	776
24.212 atca_wolfssl_internal.h File Reference	776
24.212.1 Detailed Description	776
Index	777

Chapter 1

CryptoAuthLib - Microchip CryptoAuthentication Library

1.1 Introduction

This library implements the APIs required to communicate with Microchip Security devices. The family of devices supported currently are:

CryptoAuth	CryptoAuth2
ATECC608C	ECC204
ATECC608B	ECC206
ATECC608A	SHA104
ATECC508A	SHA105
ATECC108A	SHA106
ATSHA204A	RNG90
ATSHA206A	

The best place to start is with the [Microchip Trust Platform](#)

Online API documentation is at <https://microchiptech.github.io/cryptoauthlib/>

Latest software and examples can be found at:

- <https://www.microchip.com/design-centers/security-ics/trust-platform>
- <http://www.microchip.com/SWLibraryWeb/product.aspx?product=CryptoAuthLib>

Prerequisite hardware to run CryptoAuthLib examples:

- [CryptoAuth Trust Platform Development Kit](#)

Alternatively a Microchip MCU and Adapter Board:

- [ATSAMR21 Xplained Pro](#) or [ATSAMD21 Xplained Pro](#)

- [CryptoAuthentication SOIC Socket Board](#) to accept SOIC parts
- [ATECC608B mikroBUS evaluation board](#)
- [ECC204 mikroBUS evaluation board](#)
- [SHA104/SHA105 mikroBUS evaluation board](#)
- [TA010 mikroBUS evaluation board](#)

For most development, using socketed top-boards is preferable until your configuration is well tested, then you can commit it to a CryptoAuth Xplained Pro Extension, for example. Keep in mind that once you lock a device, it will not be changeable.

1.2 Examples

- Install the [Trust Platform Design Suite](#) to access Use Case examples for the different Security Solutions (ATECC608, SHA104/105, ECC204, TA010, TA100...)

1.3 Configuration

In order to properly configure the library there must be a header file in your project named `atca_config.h`.↵ at minimum this needs to contain defines for the hal and device types being used. Most integrations have a configuration mechanism for generating this file. See the [atca_config.h.in](#) template which is configured by CMake for Linux, MacOS, & Windows projects. This file also contains some specific options that are fully documented where the desired features can be selected.

An example of the configuration:

```
/* Cryptoauthlib Configuration File */
#ifndef ATCA_CONFIG_H
#define ATCA_CONFIG_H
/* Include HALS */
#define ATCA_HAL_I2C
/* Included device support */
#define ATCA_ATECC608_SUPPORT
/* \brief How long to wait after an initial wake failure for the POST to
 * complete.
 * If Power-on self test (POST) is enabled, the self test will run on waking
 * from sleep or during power-on, which delays the wake reply.
 */
#ifndef ATCA_POST_DELAY_MSEC
#define ATCA_POST_DELAY_MSEC 25
#endif
#endif // ATCA_CONFIG_H
```

There are two major compiler defines that affect the operation of the library.

- `ATCA_NO_POLL` can be used to revert to a non-polling mechanism for device responses. Normally responses are polled for after sending a command, giving quicker response times. However, if `ATCA_NO_POLL`↵ is defined, then the library will simply delay the max execution time of a command before reading the response.
- `ATCA_NO_HEAP` can be used to remove the use of malloc/free from the main library. This can be helpful for smaller MCUs that don't have a heap implemented. If just using the basic API, then there shouldn't be any code changes required. The lower-level API will no longer use the new/delete functions and the init/release functions should be used directly.

1.4 Release notes

See [Release Notes](#)

1.5 Host Device Support

CryptoAuthLib will run on a variety of platforms from small micro-controllers to desktop host systems. See [hal readme](#)

Porting requires a time delay function of millisecond resolution (`hal_delay_ms`) which can be implemented via loop, timer, or rtos sleep/wait and a communication interface.

1.6 CryptoAuthLib Architecture

Cryptoauthlib API documentation is at <https://microchiptech.github.io/cryptoauthlib/>

The library is structured to support portability to:

- multiple hardware/microcontroller platforms
- multiple environments including bare-metal, RTOS and Windows/Linux/macOS
- multiple chip communication protocols (I2C, SPI, and SWI)

All platform dependencies are contained within the HAL (hardware abstraction layer).

1.7 Directory Structure

```
lib - primary library source code
lib/atcacert - certificate data and i/o methods
lib/calib - the Basic Cryptoauth API
lib/crypto - Software crypto implementations external crypto libraries support (primarily SHA1 and SHA2)
lib/hal - hardware abstraction layer code for supporting specific platforms
lib/host - support functions for common host-side calculations
lib/jwt - json web token functions
test - Integration test and examples. See test/cmd-processor.c for main() implementation.
For production code, test directories should be excluded by not compiling it
into a project, so it is up to the developer to include or not as needed. Test
code adds significant bulk to an application - it's not intended to be included
in production code.
```

1.8 Tests

There is a set of integration tests found in the test directory which will at least partially demonstrate the use of the objects. Some tests may depend upon a certain device being configured in a certain way and may not work for all devices or specific configurations of the device. See test readme

1.9 Using CryptoAuthLib (Microchip CryptoAuth Library)

The best place to start is with the [Microchip Trust Platform](#)

Also application examples are included as part of the Harmony 3 framework and can be copied from the Harmony Content Manager or found with the Harmony 3 Framework [Cryptoauthlib_apps](#)

1.9.1 Incorporating CryptoAuthLib in a Linux project using USB HID devices

The Linux HID HAL files use the Linux udev development software package.

To install the udev development package under Ubuntu Linux, please type the following command at the terminal window:

```
sudo apt-get install libudev-dev
```

This adds the udev development development software package to the Ubuntu Linux installation.

The Linux HID HAL files also require a udev rule to be added to change the permissions of the USB HID Devices. Please add a new udev rule for the Microchip CryptoAuth USB devices.

```
cd /etc/udev/rules.d
sudo touch mchp-cryptoauth.rules
```

Edit the mchp-cryptoauth.rules file and add the following line to the file:

```
SUBSYSTEM=="hidraw", ATTRS{idVendor}=="03eb", ATTRS{idProduct}=="2312", MODE="0666"
```

1.10 Updating **X.509 compressed certificate format** to support extended years

The X.509 compressed certificate format now supports encoding issue and expiry years beyond 2031.

For implementation details, see: [Extending Certificate Years](#)

Chapter 2

License

Replace mbedTLS ECDH Functions with hardware acceleration & hardware key security.

mbedTLS Interface Functions that enable mbedtIs objects to use cryptoauthlib functions

Replace mbedTLS ECDSA Functions with hardware acceleration & hardware key security.

Subject to your compliance with these terms, you may use Microchip software and any derivatives exclusively with Microchip products. It is your responsibility to comply with third party license terms applicable to your use of third party software (including open source software) that may accompany Microchip software.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED WARRANTIES OF NON-↵ INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

(c) 2018 Microchip Technology Inc. and its subsidiaries. You may use this software and any derivatives exclusively with Microchip products.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED WARRANTIES OF NON-↵ INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIPS TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE TERMS.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

Subject to your compliance with these terms, you may use Microchip software and any derivatives exclusively with Microchip products. It is your responsibility to comply with third party license terms applicable to your use of third party software (including open source software) that may accompany Microchip software.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED WARRANTIES OF NON-~~IN~~FRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

Chapter 3

IP Protection with Symmetric Authentication

n

The IP protection can be easily integrated to the existing projects. The user project should include [symmetric_authentication.c](#) & [symmetric_authentication.h](#) files which contains the api

- [symmetric_authenticate\(\)](#) - For Performing the authentication between host & device.

3.1 User Considerations

- The user should take care on how the master key should be stored on the MCU side.
- The api's in the file doesn't do the provisioning of the chip and user should take care of the provisioning.

With the provisioned cryptoauthentication device and after doing the cryptoauthlib initialisation, user should only be calling the function [symmetric_authenticate\(\)](#) with its necessary parameters for the authentication. The returned authentication status should be used in the application.

3.2 Examples

For more information about IP protection and its example project refer [Microchip github](#)

Chapter 4

PKCS11 Application Information

n

4.1 Setting up cryptoauthlib as a PKCS11 Provider for your system (LINUX)

These instructions are for building, installing and configuring cryptoauthlib as a pkcs11 provider. These instructions are for commonly available Linux systems with package managers.

4.1.1 Update libp11 on the system. The version should be at minimum 0.4.10

- Install the build dependencies for the system:

```
# Debian like systems
$ sudo apt-get build-dep libengine-pkcs11-openssl1.1
# RPM based systems
$ yum-builddep engine-pkcs11
```
- Change to a sane directory

```
cd ~
```
- Get the latest version of libp11

```
$ git clone https://github.com/OpenSC/libp11.git
```
- Rerun the build configuration tools:

```
$ cd libp11
$ ./bootstrap
$ ./configure
```
- Build the library:

```
$ make
```
- Install the library:

```
$ sudo make install
```


4.1.2 Build and Install cryptoauthlib with PKCS11 support

- Install the build dependencies for the system:

```
# Debian like systems
$ sudo apt-get install cmake libudev-dev
# RPM based systems
$ yum install cmake
$ yum install libudev-devel
```
- Change to a sane directory

```
cd ~
```
- Get the latest version of cryptoauthlib with PKCS11 support

```
$ git clone https://github.com/MicrochipTech/cryptoauthlib
```
- Rerun the build configuration tools:

```
$ cd cryptoauthlib
$ cmake -DATCA_PKCS11=ON .
```
- Build the library:

```
$ make
```
- Install the library:

```
$ sudo make install
```

4.1.3 Configuring the cryptoauthlib PKCS11 library

By default the following files will be created.

- `/etc/cryptoauthlib/cryptoauthlib.conf`

```
# Cryptoauthlib Configuration File
filestore = /var/lib/cryptoauthlib
```
- `/var/lib/cryptoauthlib/slot.conf.tmpl`

```
# Reserved Configuration for a device
# The objects in this file will be created and marked as undeletable
# These are processed in order. Configuration parameters must be comma
# delimited and may not contain spaces
interface = i2c,0xB0
freeslots = 1,2,3
# Slot 0 is the primary private key
object = private,device,0
# Slot 10 is the certificate data for the device's public key
#object = certificate,device,10
# Slot 12 is the intermediate/signer certificate data
#object = certificate,signer,12
# Slot 15 is a public key
object = public,root,15
```

4.1.3.1 cryptoauthlib.conf

This file provides the basic configuration information for the library. The only variable is "filestore" which is where cryptoauthlib will find device specific configuration and where it will store object files from pkcs11 operations.

4.1.3.2 slot.conf.tmpl

This is a template for device configuration files that cryptoauthlib will use to map devices and their resources into pkcs11 tokens and objects.

A device file must be named `<pkcs11_slot_number>.conf`

For a single device:

```
$ cd /var/lib/cryptoauthlib
$ cp slot.conf.tmpl 0.conf
```

Then edit 0.conf to match the device configuration being used.

4.1 Setting up cryptoauthlib as a PKCS11 Provider for your system (LINUX)

4.1.3.2.1 interface Allows values: 'hid', 'i2c' If using i2c specify the address in hex for the device. This is in the device format (upper 7 bits define the address) so will not appear the same as the i2cdetect address (lower 7 bits)

4.1.3.2.2 freeslots This is a list of slots that may be used by the library when a pkcs11 operation that creates new objects is used. When the library is initialized it will scan for files of the form <pkcs11_slot_num>.<device_↵ slot_num>.conf which defines the object using that device resource.

4.1.4 Using p11-kit-proxy

This is an optional step but is very helpful for using multiple pkcs11 libraries in a system. Detailed setup can be found at [p11-glue](#)

```
# Debian like systems
$ sudo apt-get install p11-kit
# RPM based systems
$ yum install p11-kit
```

- Create or edit the global configuration file /etc/pkcs11/pkcs11.conf. The directory /etc/pkcs11 may require creation first.

```
# This setting controls whether to load user configuration from the
# ~/.config/pkcs11 directory. Possible values:
#   none: No user configuration
#   merge: Merge the user config over the system configuration (default)
#   only: Only user configuration, ignore system configuration
user-config: merge
```

- Create a module configuration file.

- User module name (only available for a single user): ~/.config/pkcs11/modules/cryptoauthlib.↵ module
- Global module name (available to the whole system): /usr/share/p11-kit/modules/cryptoauthlib.modu↵
module: /usr/lib/libcryptoauth.so
critical: yes
trust-policy: yes
managed: yes
log-calls: no

For more details on the configuration files see the [configuration documentation](#).

4.1.5 Without using p11-kit-proxy

OpenSSL (via the libp11 project above) and p11tool support p11-kit-proxy natively so do not require additional set up if it is being used. If p11-kit-proxy is not being used then OpenSSL will have to be manually configured to use libp11 and cryptoauthlib

This requires editing the default openssl.cnf file. To locate the file being used by the system run the following command:

```
$ openssl version -a | grep OPENSSLDIR:
OPENSSLDIR: "/usr/lib/ssl"
```

This gives the default path where openssl is compiled to find the openssl.cnf file

In this case the file to edit will be /usr/lib/ssl/openssl.cnf

This line must be placed at the top, before any sections are defined:

```
openssl_conf = openssl_init
```

This should be added to the bottom of the file:

```
[openssl_init]
engines=engine_section
[engine_section]
pkcs11 = pkcs11_section
[pkcs11_section]
engine_id = pkcs11
# Wherever the engine installed by libp11 is. For example it could be:
# /usr/lib/arm-linux-gnueabi/hf/engines-1.1/libpkcs11.so
dynamic_path = /usr/lib/ssl/engines/libpkcs11.so
MODULE_PATH = /usr/lib/libcryptoauth.so
init = 0
```

4.1.6 Testing

To use p11tool it has to be installed:

```
# Debian like systems
$ sudo apt-get install gnutls-bin
# RPM based systems
$ yum install gnutls-utils
```

Note: If not using p11-kit-proxy then the provider has to be specified in p11tool calls:

```
$ p11tool --provider=/usr/lib/libcryptoauth.so
```

- Get the public key for a private key (as defined by the 0.conf file cited above):

```
$ p11tool --export-pubkey "pkcs11:token=0123EE;object=device;type=private"
warning: --login was not specified and it may be required for this operation.
warning: no --outfile was specified and the public key will be printed on screen.
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE9wzUq1EUAoNrG01rXYjNd35mxKuA
Ojw/k1IrNEBciSLL0TLjs/gvFS7N8AFXDK18vpxxu6yKzF2LRd7RY8yEFw==
-----END PUBLIC KEY-----
```

- Get the public key and decode it using OpenSSL

```
$ p11tool --export-pubkey "pkcs11:token=0123EE;object=device;type=private" | openssl pkey -pubin -text
-noout
warning: --login was not specified and it may be required for this operation.
warning: no --outfile was specified and the public key will be printed on screen.
Public-Key: (256 bit)
pub:
    04:f7:0c:d4:ab:51:14:02:83:6b:1b:4d:6b:5d:88:
    cd:77:7e:66:c4:ab:80:3a:3c:3f:92:52:2b:34:40:
    5c:89:22:cb:39:32:e3:b3:f8:2f:15:2e:cd:f0:01:
    57:0c:ad:7c:be:9c:71:bb:ac:a4:cc:5d:8b:45:de:
    dl:63:cc:84:17
ASN1 OID: prime256v1
NIST CURVE: P-256
```

- Create a CSR for the private key

```
$ openssl req -engine pkcs11 -key "pkcs11:token=0123EE;object=device;type=private" -keyform engine
-new -out new_device.csr -subj "/CN=NEW CSR EXAMPLE"
engine "pkcs11" set.
$ cat new_device.csr
-----BEGIN CERTIFICATE REQUEST-----
MIHVMHwCAQAwGjEYMBYGA1UEAwPTkVXIEN0U1BFWEFNUExFMFkwEwYHKoZIzj0C
AQYIKoZIzj0DAQcDQgAE9wzUq1EUAoNrG01rXYjNd35mxKuAOjw/k1IrNEBciSLL
OTLjs/gvFS7N8AFXDK18vpxxu6yKzF2LRd7RY8yEF6AAMAoGCCqGSM49BAMCA0kA
MEYCIQDUPElFpCOWtZxYJDYXpdl2UhpReVn6kK2LKCCX6byM8QIhAIfqfnggtcCi
W21xLAzabr8A4mHyfIIQ1oFYBg8QO9jZ
-----END CERTIFICATE REQUEST-----
```

- Verify the newly created csr

```
$ openssl req -in new_device.csr -verify -text -noout
verify OK
Certificate Request:
Data:
  Version: 1 (0x0)
  Subject: CN = NEW CSR EXAMPLE
  Subject Public Key Info:
    Public Key Algorithm: id-ecPublicKey
    Public-Key: (256 bit)
    pub:
        04:f7:0c:d4:ab:51:14:02:83:6b:1b:4d:6b:5d:88:
        cd:77:7e:66:c4:ab:80:3a:3c:3f:92:52:2b:34:40:
        5c:89:22:cb:39:32:e3:b3:f8:2f:15:2e:cd:f0:01:
        57:0c:ad:7c:be:9c:71:bb:ac:a4:cc:5d:8b:45:de:
        dl:63:cc:84:17
    ASN1 OID: prime256v1
    NIST CURVE: P-256
  Attributes:
    a0:00
  Signature Algorithm: ecdsa-with-SHA256
    30:46:02:21:00:d4:3d:e2:df:3d:c3:b0:b5:9c:58:24:36:17:
    3d:d9:76:52:1a:51:79:59:fa:90:ad:a5:28:20:97:e9:bc:8c:
    f1:02:21:00:87:ea:7e:78:20:b5:c0:a2:5b:6d:71:2c:0c:da:
    6e:bf:00:e2:61:f2:7c:82:10:d6:87:d8:06:0f:10:3b:d8:d9
```

Chapter 5

Application Support

This directory is for application specific implementation of various use cases.

Methods in this directory provide a simple API to perform potentially complex combinations of calls to the main library or API.

[app_info_ip_prot](#)

[app_info_pkcs11](#)

[app_info_secure_boot](#)

Chapter 6

Secure boot using ATECC608

8

The SecureBoot command is a new feature on the [ATECC608A](#) device compared to earlier CryptoAuthentication devices from Microchip. This feature helps the MCU to identify fraudulent code installed on it. When this feature is implemented, the MCU can send a firmware digest and signature to the ATECC608. The ATECC608 validates this information (ECDSA verify) and responds to host with a yes or no answer.

The ATECC608 provides options to reduce the firmware verification time by storing the signature or digest after a good full verification (FullStore mode of the SecureBoot command).

- When the ATECC608 stores the digest (SecureBootMode is FullDig), the host only needs to send the firmware digest, which is compared to the stored copy. This skips the comparatively lengthy ECDSA verify, speeding up the secure boot process.
- When the ATECC608 stores the signature (SecureBootMode is FullSig), the host only needs to send the firmware digest, which is verified against the stored signature using ECDSA. This saves time by not needing to send the signature in the command over the bus.

The ATECC608 also provides wire protection features for the SecureBoot command, which can be used to encrypt the digest being sent from the host to the ATECC608 and add a MAC to the verify result coming back to the host so it can't be forced to a success state. This feature makes use of a shared secret between the host and ATECC608, called the IO protection key.

The secure boot feature can be easily integrated to an existing project. The project should include the following files from the secure_boot folder:

- [secure_boot.c](#)
- [secure_boot.h](#)
- [secure_boot_memory.h](#)
- [io_protection_key.h](#)

The project should also implement the following platform-specific APIs:

- `secure_boot_init_memory()`
- `secure_boot_read_memory()`

- `secure_boot_deinit_memory()`
- `secure_boot_mark_full_copy_completion()`
- `secure_boot_check_full_copy_completion()`
- `io_protection_get_key()`
- `io_protection_set_key()`

The project can set the secure boot configuration with the following defines:

- `SECURE_BOOT_CONFIGURATION`
- `SECURE_BOOT_DIGEST_ENCRYPT_ENABLED`
- `SECURE_BOOT_UPGRADE_SUPPORT`

The secure boot process is performed by initializing CryptoAuthLib and calling the [secure_boot_process\(\)](#) function.

6.1 Implementation Considerations

- Need to perform SHA256 calculations on the host. CryptoAuthLib provides a software implementation in [lib/crypto/atca_crypto_sw_sha2.c](#)
- When using the wire protection features:
 - The host needs to be able to generate a nonce (number used once). This is the NumIn parameter to the Nonce command that is sent before the SecureBoot command. The ATECC608 can not be used to generate NumIn, but it should come from a good random or non-repeating source in the host.
 - If the host has any protected internal memory, it should be used to store its copy of the IO protection key.
- Secure boot depends on proper protections of the boot loader code in the host. If the code can be easily changed, then the secure boot process can be easily skipped. Boot loader should ideally be stored in an immutable (unchangeable) location like a boot ROM or write-protected flash.
- Note that these APIs don't provision the ATECC608. They assume the ATECC608 has already been configured and provisioned with the necessary keys for secure boot.

6.2 Examples

For more information about secure boot, please see the example implementation project and documentation at: https://github.com/MicrochipTech/cryptoauth_usecase_secureboot

Chapter 7

Contribution Guidelines

While this is an open source project there are a few considerations that make it somewhat unique in how it is managed. The first issue is that the development workflow is a hybrid between internal development and CI/CD systems and external develop and associated CI/CD systems.

- This project contains a mixture of licenses depending on the section. The vast majority is under a Microchip proprietary license that is restrictive.
- Contributors must be aware of the specific license they are working under and must be aware that by submitting the patch that they agree to the terms of the license covering the target file.
- Sources contained in the `third_party` path are covered by true open source licenses and as such are not bound by Microchip's license restrictions.
- Third party contributions for HALs must be licensed under MIT, BSD (3 clause), or Apache 2.0 license and are placed in `third_party/hal/<platform>`
- Pull requests (PR) must attest to reviewing of these rules, that licensing terms have been reviewed, the submitter has approval to submit the changes under the target license terms.

Chapter 8

Migrating to v3.7.6 to v3.7.7

8.1 atcacert

8.1.1 atcacert API Migration

This release adds support for TA Compressed certs in atcacert module.

In order to support these, especially on resource constrained device, the atcacert APIs have been modified to accept array inputs as a new structure, `cal_buffer` which allows multipart buffers to be used to provide data to these APIs. This makes for a substantial number of changes to function signatures

8.1.1.0.1 Using the `cal_buffer` structure This structure for most circumstances simply ensures that the length of the data is included with the buffer provided. There are two convenience macros provided for initialization: `CAL_BUF_INIT` & `CAL_BUF_INIT_LINK`. The later macro is only used when multipart buffers are enabled (rare). A simple example of their usage is as follows:

```
uint8_t signer_ca_public_key[64] = { 0 };
cal_buffer signer_ca_public_key_buf = CAL_BUF_INIT(sizeof(signer_ca_public_key), signer_ca_public_key);

status = atcacert_read_cert(&cert_def, &signer_ca_public_key_buf, cert, &cert_size);
```

v3.7.6	v3.7.7	Parameter Changes
atcacert_get_response	[atcacert_get_response]	Buffers
atcacert_read_cert	[atcacert_read_cert]	Public key Buffer
atcacert_read_cert_ext	[atcacert_read_cert_ext]	Public key Buffer
atcacert_merge_device_loc	[atcacert_merge_device_loc]	Added device context
atcacert_cert_build_start	[atcacert_cert_build_start]	Buffers
atcacert_set_subj_public_key	[atcacert_set_subj_public_key]	Public key Buffer
atcacert_set_signature	[atcacert_set_signature]	Signature Buffer
atcacert_get_signature	[atcacert_get_signature]	Signature Buffer
atcacert_generate_sn	[atcacert_generate_sn]	Public key Buffer
atcacert_set_auth_key_id	[atcacert_set_auth_key_id]	Public key Buffer
atcacert_get_tbs_digest	[atcacert_get_tbs_digest]	Digest key Buffer
atcacert_get_key_id	[atcacert_get_key_id]	Public key Buffer
atcacert_der_enc_ecdsa_sig_value	[atcacert_der_enc_ecdsa_sig_value]	Signature Buffer
atcacert_der_dec_ecdsa_sig_value	[atcacert_der_dec_ecdsa_sig_value]	Signature Buffer

v3.7.6	v3.7.7	Parameter Changes
atcacert_verify_cert_hw	[atcacert_verify_cert_hw]	Public key Buffer
atcacert_gen_challenge_hw	[atcacert_gen_challenge_hw]	Buffers
atcacert_verify_response_hw	[atcacert_verify_response_hw]	Buffers
atcacert_gen_challenge_sw	[atcacert_gen_challenge_sw]	Buffers
atcacert_verify_cert_sw	[atcacert_verify_cert_sw]	Public key Buffer
atcacert_verify_response_sw	[atcacert_verify_response_sw]	Buffers

8.1.1.0.2 Function Migration

Chapter 9

openssl directory - Purpose

This directory contains the interfacing and wrapper functions to integrate openssl as the software crypto library.

Chapter 10

atcab

10.1 atcab API reference

Hardware Feature	atcab_
SHA256	[atcab_sha_start]
SHA256	[atcab_sha_update]
SHA256	[atcab_sha_end]

Chapter 11

Python CryptoAuthLib module

11.1 Introduction

This module provides a thin python ctypes layer to evaluate the cryptauthlib interface to Microchip Crypto↔ Authentication devices.

11.1.1 Code Examples

Code examples for python are available on github as part of [CryptoAuthTools](#) under the [python/examples](#) directory

11.2 Installation

11.2.1 CryptoAuthLib python module can be installed through Python's pip tool:

```
pip install cryptauthlib
```

11.2.2 To upgrade your installation when new releases are made:

```
pip install -U cryptauthlib
```

11.2.3 If you ever need to remove your installation:

```
pip uninstall cryptauthlib
```

11.3 What does python CryptoAuthLib package do?

CryptoAuthLib module gives access to most functions available as part of standard cryptauthlib (which is written in 'C'). These python functions for the most part are very similar to 'C' functions. The module in short acts as a wrapper over the 'C' cryptauth library functions.

Microchip cryptauthlib product page: [Link](#)

11.4 Supported hardware

- AT88CK101
- CryptoAuthentication SOIC XPRO Starter Kit (DM320109)

11.5 Supported devices

The family of devices supported currently are:

- ATSHA204A
- ATECC108A
- ATECC508A
- ATECC608A

11.6 Using cryptoauthlib python module

The following is a 'C' code made using cryptoauthlib 'C' library.

```
#include "cryptoauthlib.h"
void main()
{
    ATCA_STATUS status;
    uint8_t revision[4];
    uint8_t randomnum[32];
    status = atcab_init(cfg_ateccx08a_kitcdc_default);
    if (status != ATCA_SUCCESS)
    {
        printf("Error");
        exit();
    }
    status = atcab_info(revision);
    if (status != ATCA_SUCCESS)
    {
        printf("Error");
        exit();
    }
    status = atcab_random(randomnum);
    if (status != ATCA_SUCCESS)
    {
        printf("Error");
        exit();
    }
}
```

The same code in python would be:

```
from cryptoauthlib import *
ATCA_SUCCESS = 0x00
revision = bytearray(4)
randomnum = bytearray(32)
# Locate and load the compiled library
load_cryptoauthlib()
assert ATCA_SUCCESS == atcab_init(cfg_ateccx08a_kithid_default())
assert ATCA_SUCCESS == atcab_info(revision)
print("".join(['%02X ' % x for x in revision]))
assert ATCA_SUCCESS == atcab_random(randomnum)
print("".join(['%02X ' % x for x in randomnum]))
```

In the above python code, "import cryptoauthlib" imports the python module. load_cryptoauthlib() function loads the compiled library. The load_cryptoauthlib() is a function that you will not see in the 'C' library, this is a python specific utility function and is required for python scripts to locate and load the compiled library.

11.7 In Summary

11.7.1 Step I: Import the module

```
from cryptoauthlib import *
```

11.7.2 Step II: Initilize the module

```
load_cryptoauthlib()  
assert ATCA_SUCCESS == atcab_init(cfg_ateccx08a_kithid_default())
```

11.7.3 Step III: Use Cryptoauthlib APIs

Call library APIs of your choice

11.8 Code portability

Microchip's CryptoAuthentication products can now be evaluated with the power and flexibility of python. Once the evaluation stage is done the python code can be ported to 'C' code.

As seen above the python API maintains a 1 to 1 equivalence to the 'C' API in order to easy the transition between the two.

11.9 Cryptoauthlib module API documentation

11.9.1 help() command

All of the python function's documentation can be viewed through python's built in help() function.

For example, to get the documentation of [atcab_info\(\)](#) function:

```
>>> help(cryptoauthlib.atcab_info)  
Help on function atcab_info in module cryptoauthlib.atcab:  
atcab_info(revision)  
Used to get the device revision number. (DevRev)  
Args:  
    revision          4-byte bytearray receiving the revision number  
                      from the device. (Expects bytearray)  
Returns:  
    Status code
```

11.9.2 dir() command

The dir command without arguments, return the list of names in the current local scope. With an argument, attempt to return a list of valid attributes for that object. For example `dir(cryptoauthlib)` will return all the methods available in the cryptoauthlib module.

11.10 Code Examples

Code examples for python are available on github as part of [CryptoAuthTools](#) under the python/examples directory

11.11 Tests

Module tests can be located in the [python/tests](#) of the main cryptoauthlib repository. The [README.md](#) has details for how to run the tests. The module tests are not comprehensive for the entire functionality of cryptoauthlib but rather are meant to test the python module code only against the library to ensure the interfaces are correct and ctypes structures match the platform.

11.12 Release notes

See Release Notes

Chapter 12

Python CryptoAuthLib Module Testing

12.1 Introduction

These tests are designed to only test the python interface to the library and are not designed to test the library itself which is covered by the main cryptoauthlib tests

12.1.1 Running

The best way to run the test suite is to use `tox` which can be easily installed with pip:

```
$ pip install tox
```

From the python folder:

```
~/cryptoauthlib/python $ tox
```

It is possible to directly run tests but requires more setup

1) Install pytest

```
$ pip install pytest
```

2) Modify the PYTHONPATH environment variable

Windows:

```
cryptoauthlib/python> set PYTHONPATH=<path_to>/cryptoauthlib/python
```

Linux:

```
$ export PYTHONPATH=${PYTHONPATH}:<path_to>/cryptoauthlib/python
```

3) Run the tests

```
$ pytest -vv
```

12.1.2 Test options

There are additional options that can be invoked with the tests that define what tests will be run

1) `--with-lib` will attempt to run tests against the compiled c library. These tests are good for detecting possible platform incompatibilities between the C compiler and the expectations of python

2) `--with-device` will attempt to invoke some tests with a real attached device These tests are restricted to only the minimum required to verify the python to library connectivity and are only meant to detect situations can not be determined from the library tests alone.

Chapter 13

Microchip Cryptoauthlib Release Notes

13.1 Release v3.7.9 (09/05/2025)

13.1.1 New Features

- Upgraded MbedTLS CAL wrapper APIs for compatibility with Mbed TLS v3.6.3.1 (from v2.28.4)

13.1.2 Fixes

- Fixed `atcacert_verify_` APIs to use correct key size macros and to utilize `cal_buffer` members
- Enhanced error handling for initialization (`halinit`) failures by adding appropriate `hal_release` calls
- Fixed `atca_mbedtls_cert_add` API to read public key using correct `is_genkey` check
- Updated `hal_kit_hid_send()` API to flush any pending data from HID device input buffer (by reading its data) before writes
- Added validation of input parameters in `hal_free_shared()` and updated access permissions for the shared memory in `hal_linux.c`
- Replaced `memset` calls with `pkcs11_util_memset` when dealing with structures, in the PKCS#11 implementation
- Modified buffer size setting to use `P256 signature size` before calling `atcacert_set_signature()`, in order to correctly read WPC certificates from ECC608 devices
- Fixed `ATCA_MBEDTLS` guards to exclude MbedTLS code when disabled
- Replaced `CMAKE_INSTALL_FULL_<dir>` with `CMAKE_INSTALL_<dir>` to ensure proper handling of `--prefix` and `cpack`
- Updated `atcab_init_ext()` to read the ECC608 config zone only with `ATCA_NO_POLL` configuration
- MPLAB Harmony Configurator changes
 - Added `device_conf.json` file to contain the features pertinent to each of the secure elements and use this as base configuration options for the selected device
 - Updated the Configurator code generation for TNG devices selection to generate `Tester module` files as well
- MISRA C:2012 and CERT C checks

- Addressed more violations of 'REQUIRED' category for MISRA C:2012 and CERT C compliance standards
- Introduced file-based suppression lists to support maximum coverage for MISRA C:2012 rule checks while using `MISRA Check tool` from `MPLAB Analysis tool suite`
- Updated Zephyr I2C HAL wrapper
 - Integrated latest I2C driver configuration updates from Zephyr 3.6.0
 - Now utilizes the `word_address` parameter in `hal_i2c_send()` even when the transmit buffer is empty (supports cases like device wake-up)
- Updated `hal_esp32_i2c.c` to support latest ESP-IDF library
 - Maintained existing I2C initialization and command methods for backward compatibility
 - Note: The `i2c_config_t` fields and the `i2c_driver_install` API may be deprecated in future ESP-IDF releases. Refer to the ESP-IDF I2C official documentation for latest updates.

13.2 Release v3.7.8 (05/05/2025)

13.2.1 Fixes

- Updated minimum required version of CMake to v3.20 to support its new features and improvements
- Resolved `tng_atcacert_read_signer_cert` build failure when using MPLAB Harmony projects
- Fixed `tng_atcacert_max_device_cert_size` function to prevent overwriting `max_cert_size`
- Addressed the dependency issue where SHA512 is required for SHA384 in the SW Crypto module
- Fixed an issue in the Linux environment where the SPI file descriptor was inadvertently closed twice
- Incorporated additional `ATCA_CHECK_PARAMS_EN` checks in sections of the code where they were previously absent
- Eliminated all compiler warnings in MPLAB Melody related to the `atcacert` module
- Eliminated all compiler warnings in MPLAB Harmony projects when adding the `CAL` library as a project component
- Addressed all MISRA violations categorized as "Required" in MPLAB Harmony. Note that "Advisory" issues remain unaddressed
- Resolved build issues in the `PyCAL` library to ensure its compatibility `CAL` library counterpart in C
- Resolved build errors on arm64 MacOS platforms when utilizing the USB library for `hidapi`; kick-start transition to `libusb-maintained` version of `hidapi`, moving away from the previous `signal11` repository
- Addressed incorrect header file inclusions in `atca_mbedtls_wrap.h` related to the `ATCA_MBEDTLS` configuration
- PKCS11 layer fixes/updates
 - Corrected return value in `C_FindObjectsInit` API to no longer return `CKR_OK` when no objects are found
 - Fixed an issue causing certificate export failures when using `ATECCC608` `TNGTLS` devices
 - Resolved a race condition that occurred during the creation of a mutex

13.3 Release v3.7.7 (02/07/2025)

13.3.1 New Features

- Extended atcacert module to support compressed certificate usage for TA devices
- Enhanced WPC application to support TA devices
- Updated PKCS#11 and Openssl wiki documentation to include steps for using Openssl 3.0+ versions
- Updated PKCS#11 module to add compatibility for higher SHA-2 (SHA384 and SHA512) functions
- Added NIST vector tests to cover AES CCM module validation
- Modified calib packet allocation to use memory from either heap or data segment based on user configuration instead of always using data segment.

13.3.2 Fixes

- Resolved SWI 1-wire communication failure occurring in hal_swi_gpio while using MPLABx Harmony projects
 - delay routines in hal_cortex_m_delay are optimized to generate accurate delays for SAM cortex-m device family
 - removed call stack overhead in hal_swi_gpio to meet required SWI bit timing
- Minimum required version for CMake is changed from 3.0.1 to 3.10.0
- Fixed compilation issues with atcac_get_subj_public_key when WolfSSL configuration is enabled

13.3.3 API CHANGES

- Replaced I/O buffers in atcacert with cal_buffer at few instances to support resource-constrained PIC18 devices
 - Refer [lib/atcacert/MIGRATION.md] for details on atcacert API changes

13.4 Release v3.7.6 (09/26/2024)

13.4.1 New Features

- In PKCS11 module, added support for RSA key types, certificates and algorithms
- Added SHA384 and SHA512 support for host side software crypto (lib/crypto/) operations
- Modified WPC application to support ECC204 and TA010 devices
- See [talib/CHANGES.md] for details on talib module changes

13.4.2 Fixes

- Shared library build (libcryptoauth.so) sets ABI version number (libcryptoauth.so.x)
- Fixed [atcacert_read_cert\(\)](#) API failure when certificate elements read from config zone for ECC204 and TA010 devices
- Resolved kit protocol compilation failure for PIC18 device (XC8) builds
- PKCS11 layer fixes/updates
 - Fixed C_DestroyObject failure when deleting a key pair
 - Fixed C_DeriveKey API usage sequence

13.5 Release v3.7.5 (06/26/2024)

13.5.1 New Features

- In PKCS11 module, added ECCP384,ECCP521,ECCP224 elliptic curves support for ECC key operations, in addition to the existing ECCP256 support
- Enhanced certificate related tests to include coverage for ECC204 and TA010 devices
- Added a new ATCA_HEAP internal macro check in place of ATCA_NO_HEAP for dynamic memory usages
- Added an additional test to validate AES-CBC encrypt/decrypt APIs using CAVP's AES multiblock message test (MMT) sample vectors
- See [talib/CHANGES.md] for details on talib module changes

13.5.2 Fixes

- Fixed [atcacert_get_comp_cert\(\)](#) API to support certificates with expiry dates beyond year 2031
- Fixed [atcacert_read_cert\(\)](#) API to consider serial number as source while processing extracted certificates
- Fixed [atcacert_write_cert\(\)](#) API to support X509 certificates with an odd byte length, without any additional padding
- Fixed [calib_execute_send\(\)](#) to consider correct data buffer when ATCA_HAL_LEGACY_API is used
- PKCS11 layer fixes/updates
 - Fixed certificate chain/key export failures in ECC608 Trust devices
 - Fixed memory leak during C_Finalize API call usage in a multi-slot configuration

13.5.3 API Changes

- Added [atcacert_generate_sn\(\)](#) API in atcacert module to generate certificate serial number from a valid serial number source

13.6 Release v3.7.4 (03/08/2024)

13.6.1 New Features

- Updated wolfSSL interface `atcac` wrapper APIs usage for AES GCM encrypt/decrypt similar to MbedTLS and openssl library wrapper APIs
- Added package.yml file to support MPLAB Harmony metadata package format

13.6.2 Fixes

- Fixed `calib_wakeup_i2c` API to follow specified i2c wakeup sequence for ECC608 devices
- PKCS11 layer fixes/updates
 - Lock usage optimization in `pkcs11_find_continue` API
 - `pkcs11_digest` API updates for SHA context memory allocation
 - `pkcs11_token_set_pin` API updates to write data based on generated GCM key size
- Fixed `atcacert_get_comp_cert` API to remove a redundant `atcacert_date_enc_compcert` call
- Resolved build warnings/issues in Windows, Linux and 8-bit (XC8) platforms
- wolfSSL's `atcac_pk_init_pem` wrapper API updates to use `wc_PEM` to DER functions
- Fixed broken links in README.md files

13.7 Release v3.7.3 (01/31/2024)

13.7.1 New Features

- In PKCS11 module, added cache support to store `Key id` attribute of key type objects into stack memory and use it for subsequent accesses

13.7.2 Fixes

- Fixed `calib_sha_hmac_finish` api to set mode value correctly for ECC204, TA010 and ECC608 devices
- Fixed memory leak in MbedTLS configuration
- Fixed build errors when a project is generated with PKCS11 Component enabled in MPLAB Harmony Configurator (MHC)

13.8 Release v3.7.2 (01/19/2024)

13.8.1 New Features

- See [talib/CHANGES.md] for details on talib module changes

13.8.2 Fixes

- Updated PKCS11 token info to list TA101 device details
- Fixed compilation errors when ECC508 device is enabled
- See [talib/CHANGES.md] for details on talib module fixes

13.8.3 API Changes

- Added sign and verify API in talib module to support 1024 bytes ED25519 mode

13.9 Release v3.7.1 (12/15/2023)

13.9.1 New Features

- PKCS11 module enhancements for x509 public key certificates
 - Added more certificate attributes to x509 public key certificates. These attributes include certificate start date, certificate end date, subject, subject key, DER encoded certificate issuer name, DER encoded certificate serial number and hash of the issuer public key.
 - Added cache support to store these certificates into stack memory and utilize it for parsing the above specified certificate attributes.
- See [talib/CHANGES.md] for details on talib module changes

13.9.2 Fixes

- Updated atcab_read_config_zone to support SHA106
- For Linux platforms, i2c baud rate is always set to 100 khz as the default configuration
- Resolved build errors when ATCA_USE_SHARED_MUTEX is disabled
- Resolved build error with ATCA_JWT_EN

13.9.3 API Changes

- Added atcacert_get_subject api to get the subject name from public x509 certificates
- Added atcacert_get_issuer api to get the issuer name from public x509 certificates
- Updated the [atcacert_def_s](#) structure to include x509 full certificates support

13.10 Release v3.7.0 (09/08/2023)

13.10.1 New Features

- Added unified buffer implementation to enable multipart buffer use with APIs that support them.
- See [talib/CHANGES.md] for details on talib module changes

13.10.2 Fixes

- Made `atcac` structures referencing third party libraries opaque to the user so installed header files are usable by applications without also including the third party headers.

13.10.3 API Changes

- The software crypto structures are generally no longer typedef'd so they must be declared with the `struct` keyword. New typedefs were added by appending the suffix `_t` which allows for the same mechanism for declaring these structure in code if building a standalone application (such as in embedded projects). If dynamically linking with the library and using a third party crypto library one will need to use the `_new` & `_free` APIs to allocate these structures for use with the `atcac` interfaces.

13.11 Release v3.6.1 (07/14/2023)

13.11.1 New Features

- Added support for PIC18 memory model with a `MAX_PACKET_SIZE` setting.
- PKCS11 Improvement to support context reservation automatically for operations that span multiple `pkcs11` calls such as login/logout, encrypt/decrypt, etc. This prevents concurrent processes from interrupting init-update-finish operations in PKCS11
- Added support for data element transfers between trust anchor devices

13.11.2 Fixes

- PKCS11: resolved issues with configuration directory parsing to ensure configurations parse in the correct order and any extraneous files get properly rejected.
- PKCS11: improved public key loading logic for trust anchor handles to use the most appropriate mechanism based on handle configuration.
- Fixed minimal kit host implementation in support bridging to SPI by using select and deselect control commands

13.12 Release v3.6.0 (04/04/2023)

13.12.1 New Features

- Compliance certified to CERT-C Level 2 & MISRA 2012. Compliance reports can be requested from your FAE or account manager
- Added `talib_handle` helper functions to determine if a handle access type is allowed in the given auth session

13.12.2 Fixes

- pkcs11 public key for private keys requiring the token to be logged in will make a best effort to return a value by detecting various storage methods.
- pkcs11 encrypt/decrypt update calls return the maximum possible bytes per the selected algorithm.
- pkcs7 would return the wrong padding for `length % 16 == 0`
- hmac counter kdf method will default to digest length specified in bits

13.12.3 API Changes

- ATCA_STATUS enum is now an integer and all APIs return type ATCA_STATUS
- atcacert API return type is now ATCA_STATUS rather than `int`
- atcac_sw_sha... API return type is now ATCA_STATUS rather than `int`
- `_atcab_exit` has been removed (includes `_calib_exit` and `_talib_exit`)
- `_gDevice` has been renamed to `g_atcab_device_ptr` (one should be using `atcab_get_device()`)

13.13 Release v3.5.1 (03/26/2023)

13.13.1 New Features

- Add support for SHA104, SHA105, & SHA106

13.14 Release v3.5.0 (03/14/2023)

13.14.1 New Features

- Add support for ECC204, TA010 and framework for future devices

13.15 Release v3.4.3 (12/23/2022)

13.15.1 New Features

- Add key load mode flags for FCE config command

13.15.2 Fixes

- WPC certificate reconstruction buffer length was too short
- ECC204 block Read/Write did not write remaining bytes if the provided buffer was not padded to a 32 byte boundary
- TA100 lock CRC was being passed with the native endianness.
- ECC204 nonce command was missing the mode bit to emit a random number when called with the intention of producing random bytes

13.16 Release v3.4.2 (12/04/2022)

13.16.1 Fixes

- PKCS11: Correct init/deinit failures from initialization mutex options. These would manifest as a segmentation fault on deinit, unterminated authorization sessions, or library already initialized return codes based on the configuration and initialization data.
- PKCS11: Added configuration option to always terminate authorization sessions on library initialization to work around applications that may fail to call `C_CloseSession` or `C_Finalize` before exiting.
- PKCS11: Fix failures in `C_DigestInit` resulting from failing to check the session state before checking the requested digest mechanism type.
- PKCS11: Modify how the library returns public key information based on access levels of the private key (generate from the private key if allowed, read from a linked public key, and finally return data unavailable). For the vast majority of situations this prevents openssl & libp11 from crashing with segmentation faults if the user fails to provide a pkcs11 URI with pin value specified. These segmentation faults were confirmed to also exist with other PKCS11 libraries - the fundamental problem should be taken up with the maintainers of openssl, libp11, and pkcs11-provider (experimental OpenSSL 3.0 PKCS11 support).
- Modified CBC update/finish APIs (added as an experimental API in v3.4.0) to match standard expectations of how the APIs would function. Updated algorithm tests to reflect this usage.
- PKCS11: Updated encrypt/decrypt in cbc/cbcpad modes to use the updated algorithm implementations
- talib full element read & write functions now account for the maximum packet size based on session state.

13.17 Release v3.4.1 (11/11/2022)

13.17.1 Fixes

- `test_atcacert_build_start_signer` modified to verify the structure fields since the structure is no longer packed
- Python `ctypes_to_bytes` routine to work for all python versions
- Pkcs11 signature rules to match section 5.2 of the specification
- Compilation error when PKCS11 monotonic counter is enabled
- Compilation error when no HALs are specified during configuration
- Align ECC204 and cryptoauth counter APIs

13.18 Release v3.4.0 (10/27/2022)

13.18.1 New Features

- Added framework for fine grain library configuration including configuration check header files `<api>_↔ config_check.h` see [lib/atca_config_check.h](#) for the top level header
- Added WPC application files with reference message generation/parsing and library configuration file to optimize to the smallest footprint
- TA100 read/write apis updated to segment incoming buffer into partial read/write operations if it exceeds the maximum supported packet size
- Added PKCS7 padding algorithm for use with AES-CBC
- Expose PKCS11 configuration options to CMake configuration

13.18.2 Fixes

- Improve ECC204 apis to match cryptoauthlib apis and abstract the device differences
- Support for strict C99 compliance and clean up warnings from -Wall and pedantic levels
- Add rsa2048 key size support to talib_rsaenc command
- Fix for ta100 devupdate to set the proper auth session exit flags so the library will properly reconnect when the ta100 reboots
- Fix ECC608 verify failure when ReqRandom bit is set for a stored public key by using tempkey in this situation rather than the message digest buffer. See the ECC608 datasheet for more details of this special condition
- Improve ta100 auth session handling of long messages by reporting the message size exceeds the wrapped message limit earlier in the packet creation process
- Fixes and Improvements for PKCS11 interface based on compliance testing

13.19 Release v3.3.3 (10/06/2021)

13.19.1 New features

- Added Zephyr support and zephyr driver api HALs for I2C & SPI. Adding cryptoauthlib to a zephyr project CMakeLists.txt is now possible - use subdirectory(cryptoauthlib/lib). One can also include the repo in the west manifest
- Added SWI device support for linux platforms using hardware uarts
- Added contributing guidelines and PR process documentation
- SWI bitbang driver for harmony - supports Atmel SWI and ECC204 protocols

13.19.2 Fixes

- Wolfssl build errors when generating MHC projects containing wolfssl
- Removed zero length aad limitation in CCM implementation
- Changed ECC204 zone identifiers and slot types to align with cryptoauthlib standard forms
- XC8/XC16 build warnings
- Several pkcs11 fixes - token_init deadlock, null num_in for private key writes, fsecret key length parsing, object_create failing, etc
- Null pointer access violation in atcab_release when using a native hal and double free in openssl implementation of atcac_pk_verify

13.20 Release v3.3.2 (06/20/2021)

13.20.1 New features

- All memory allocations now go through the hal_platform definitions. In harmony these are the OSAL_fuctions which work with any of the supported RTOS'.
- Enable multiple interfaces in the Harmony 3 test project through the user interface.
- Kit protocol over UART has been added. This can be paired with the included hosting application
- Simple kit protocol hosting application has been added. It is available in app/kit_host and through Harmony 3. This is a preview release of the application.

13.20.2 Fixes

- Enable ATSHA206A api in the python extension
- Made the linux i2c configuration default to 100khz so they should work again without having to make modifications to the baud rate field.
- Fix pkcs11 static configuration option when used with the trust platform configuration file
- Fix PKCS11 ec_point return value when pValue is null (libp11 checks the size in this manner before requesting it for real).
- Fix warnings generated by missing end of file newlines.
- Removed legacy (empty) START header references.

13.21 Release v3.3.1 (04/23/2021)

13.21.1 New features

- Core support for kit protocol over serial ports (i.e. tty/COM ports)
- PKCS11 support for TA100 auth sessions

13.21.2 Fixes

- Fix mbedtls integration combinations that would produce unexpected behavior. All variations of sign/verify _ALT now work as expected given a configured key (for example if a key is configured as a stored public and VERIFY_ALT is enabled then library will perform a stored key verify rather than an external public key load and verify)
- Added mbedtls integration tests to confirm that integrations are working on a target platform as expected. These generally bootstrap using NIST example vectors before using the validated functions/algorithms to test the remaining integration.
- Clean up warnings when run with very strict settings (-Wall -Wextra -pedantic -Werror)
- Fix false wake errors when baud rate switching for I2C
- Fix for I2C errors that could be created on the bus when there are devices on the bus that support general calls - this fix should also correct linux zero length kernel messages when enabled.
- Fix ESP32 HAL to work with the updated HAL structure.

13.22 Release v3.3.0 (01/22/2021)

13.22.1 API Updates

- HAL API has been significantly revised to improve portability. This update simplifies the requirements of each HAL to only the physical transport mechanisms. Please see the hal porting and library upgrading notes: <https://github.com/MicrochipTech/cryptoauthlib/wiki/Upgrading-to-v3.3>
- Internal structures have been updated by removing obsolete elements and combining mandatory fields. This saves significant memory in both program and data regions.
- Inclusive language update: all remaining legacy language elements have been updated. Where this impacts the external API there is the option ATCA_ENABLE_DEPRECATED to use the previous names.

13.22.2 New features

- ECC204 support has been added with one wire HAL support.
- ECC204, SHA206, one wire and single wire (uart and gpio) hals have been added to the Harmony 3 configurator.
- PKCS11 support for symmetric (AES & HMAC) keys has been added and enabled for additional mechanisms such as HMAC signing and AES encrypt/decrypt

13.22.3 Fixes

- pkcs11_token_init had several conditions that were corrected
- fix to detect differences in i2c clock rate specifications between flexcom and sercom configurators in Harmony 3 and the emit the correct value for the cryptoauthlib interface config structure.

13.23 Release v3.2.5 (11/30/2020)

13.23.1 New features

- TA100 ShareKey API to drive the sharekey process (requires NDA, consult with your FAE or submit a request through your myMicrochip account)
- Additional software crypto library interface functions for asymmetric cryptography (sign, verify, ecdh, etc)
- XC8 & XC16 compiler support
- AES CCM & CBC-MAC upper layer API using AES-ECB primitives

13.23.2 Fixes

- TA100 AES-GCM auth session tx packet length when command data is included
- PKCS11 Pin length check rejecting valid pin lengths
- aes-gcm nist vector test failed with mbedtls crypto backend due to aad update not being executed when aad length was zero

13.24 Release v3.2.4 (10/17/2020)

13.24.1 New features

- Additional TA100 command support (requires NDA, consult with your FAE or submit a request through your myMicrochip account)
- Library build and install on linux now also installs the headers that were used to build the library including all configuration files like atca_config.h - customer applications building against the library will need to add the include/cryptoauthlib to their include search paths

13.24.2 Fixes

- Fixed errors produced when -fno-common was used during build of the library by resolving the variable declaration and exporting macros (tested with static/dynamic linkage on linux & windows platforms)
- Added a timeout during i2c plib commands in the Harmony3 hals to prevent system lockups from failed peripheral transfers that don't return errors.

13.25 Release v3.2.3 (09/12/2020)

13.25.1 New features

- Additional TA100 command support (requires NDA, consult with your FAE or submit a request through your myMicrochip account)

13.25.2 Fixes

- Security patch for USB HALs. Removed deprecated HALs and removed enumeration from the hidapi HAL.
- Fix device matching logic to support older kits when using "auto detect" settings in the interface configuration
- Fix SPI HAL generation errors for SAMG55 & SAM71 (flexcom) devices
- Added a timeout for Harmony I2C calls to prevent infinite loops on peripheral failures. If a loop exists inside the peripheral library then it may still cause processor spins until a watchdog reset.

13.26 Release v3.2.2 (07/28/2020)

13.26.1 New Features

- ATECC608B support added

13.26.2 Fixes

- Consistent null pointer checks between calib & talib apis. Tracing enabled for most all status changes
- Fix for pkcs11 ecdh with the legacy slot write mode and encrypted read to pull the read key id from the correct slot (private key slot | 0x01)
- call the proper api from atcab_init_ext so it works with device structures that are not the global instance

13.27 Release v3.2.1 (06/29/2020)

13.27.1 Fixes

- PKCS11 configuration option to set token label to the device serial number
- Fix OSX CLANG macro error
- Add missing c++ wrapper macros to calib_basic.h
- Ensure atcab_init_ext calls atcab_release_ext rather than atcab_release

13.28 Release v3.2.0 (06/10/2020)

13.28.1 New features

- TA100 device support (requires NDA, consult with your FAE or submit a request through your myMicrochip account)
- Extension of the existing API to support device context retention to allow multiple independent contexts to be maintained. The application still needs to ensure concurrency protections are used in the application to guard bus communication.
- PKCS11 support has been moved into the main library and will be maintained together.
- TNG/TFLEX support has been added to PKCS11 so enabling a TNG part in pkcs11 can be done by specifying the part number: `device = ATECC608A-TNGTLS`
- Several cryptographic library integrations have been added to enable additional host/mcu side functionality. This includes replacing cryptoauthlib software implementations of sha1 & sha256 with your preferred library. For example using WolfSSL in Harmony 3 will also enable hardware acceleration of those cryptographic functions. Cryptographic libraries enabled: WolfSSL, mbedTLS, & OpenSSL
- Changes to atcacert ("compressed" certificate processing) to enable exact certificate size retrieval which will help with some use cases that had issues with the max possible size answers.
- Consolidation of HALs into device families rather than exact processor model This should reduce the amount of effort required to port the library to a specific platform if the framework is one that is already known.

13.28.2 Known issues

- Power modes/states for the TA100 are not automatically controlled by the library so the application has to manually change the power state when lower power modes are required. A command such as the info command will wake the TA100 from sleep but will produce an error. Try another command after the specified time to ensure communication is restored. This behavior is detailed in the datasheet.
- Several TA100 commands and features are planned for the next released of the library such as import/export, transfer, and devupdate.

13.29 Release v3.1.1 (03/06/2020)

- Update Trust Flex certificates. Add compile time options to reduce code space by selectively including the trust certificates that are required
- Python updates: add sha206 apis. Fix atcab_kdf parameters
- Fix compiler warnings in test application files and sha206 api

13.30 Release v3.1.0 (02/05/2020)

- The library is now semantic versioned along with the legacy date versioning. Python will continue to be released with the date version. Version APIs have been updated.
- Configuration is done via a configuration file `atca_config.h` rather than global compiler options. You have to add this file to your project to support this version of the library.
- Harmony 3 support has been added. Update harmony configurator (and content loader) or manually clone cryptoauthlib into your harmony directory.
- Additional Compiler support has been added for IAR-ARM and ARMCC

13.31 Release 11/22/2019

- Patches for CVE-2019-16128 & CVE-2019-16129: Ensure reported packet length is valid for the packet being processed.
- Improvement to encrypted read operations to allow supply of a host nonce (prevent replay of a read sequence to the host). Default API is changed but can be reverted by setting the option `ATCA_USE_CONSTANT_HOST_NONCE`
- Added Azure compatible TNGTLS and TNGLORA certificates. Use the TNG client API to retrieve the proper certificate based on the device.
- Misc Python updates (updated APIs for encrypted reads to match the C-API change) `atcacert_cert_element_t` now initializes properly

13.32 Release 08/30/2019

- Added big-endian architecture support
- Fixes to `atcah_gen_dig()` and `atcah_nonce()`

13.33 Release 05/17/2019

- Added support for TNG devices (cert transforms, new API)
- `atcab_write_pub_key()` now works when the data zone is unlocked

13.34 Release 03/04/2019

- mbed TLS wrapper added
- Minor bug fixes

13.35 Release 01/25/2019

- Python JWT support
- Python configuration structures added
- Restructure of secure boot app

13.36 Release 01/04/2019

- Added GCM functions
- Split AES modes into separate files
- Bug fix in SWI START driver

13.37 Release 10/25/2018

- Added basic certificate functions to the python wrapper.
- Added Espressif ESP32 I2C driver.
- Made generic Atmel START drivers to support most MCUs in START.
- Added AES-CTR mode functions.
- Python wrapper functions now return single values with AtcaReference.
- Added mutex support to HAL and better support for freeRTOS.

13.38 Release 08/17/2018

- Better support for multiple kit protocol devices

13.39 Release 07/25/2018

- Clean up python wrapper

13.40 Release 07/18/2018

- Added ATCA_NO_HEAP define to remove use of malloc/free.
- Moved PEM functions to their own file in atcacert.
- Added wake retry to accomodate power on self test delay.
- Added ca_cert_def member to [atcacert_def_s](#) so cert chains can be traversed as a linked list.

13.41 Release 03/29/2018

- Added support for response polling by default, which will make commands return faster (define ATCA_NO←_POLL to use old delay method).
- Removed atcatls related files as they were of limited value.
- Test framework generates a prompt before locking test configuration.
- Test framework puts device to sleep between tests.
- Fixed mode parameter issue in atcah_gen_key_msg().
- ATECC608A health test error code added.

13.42 Release 01/15/2018

- Added AES-128 CBC implementation using AES command
- Added AES-128 CMAC implementation using AES command

13.43 Release 11/22/2017

- Added support for FLEXCOM6 on SAMG55 driver

13.44 Release 11/17/2017

- Added library support for the ATECC608A device
- Added support for Counter command
- `atca_basic` functions and tests now split into multiple files based on command
- Added support for multiple base64 encoding rules
- Added support for JSON Web Tokens (jwt)
- Fixed `atcab_write_enc()` function to encrypt the data even when the device is unlocked
- Fixed `atcab_base64encode_()` for the extra newline
- Updated `atcab_ecdh_enc()` to work more consistently

13.45 Release 07/01/2017

- Removed assumption of `SN[0:1]=0123`, `SN[8]=EE`. SN now needs to be passed in for functions in `atca_host` and `atca_basic` functions will now read the config zone for the SN if needed.
- Renamed `atcab_gendig_host()` to `atcab_gendig()` since it's not a host function. Removed original `atcab_gendig()`, which had limited scope.
- Fixed `atcah_hmac()` for host side HMAC calculations. Added `atcab_hmac()`.
- Removed unnecessary `ATCADeviceType` parameters from some `atca_basic` functions.
- Added `atcacert_create_csr()` to create a signed CSR.
- New HAL implementation for Kit protocol over HID on Linux. Please see the Incorporating CryptoAuthLib in a Linux project using USB HID devices section in this file for more information.
- Added `atcacert_write_cert()` for writing certificates to the device.
- Added support for dynamic length certificate serial numbers in `atcacert`.
- Added `atcab_write()` for lower level write commands.
- Fixed `atcah_write_auth_mac()`, which had wrong OpCode.
- Added `atcab_verify()` command for lower level verify commands.
- Added `atcab_verify_stored()` for verifying data with a stored public key.

- Removed `atcab_write_bytes_slot()`. Use `atcab_write_bytes_zone()` instead.
- Modified `atcab_write_bytes_zone()` and `atcab_read_bytes_zone()` to specify a slot
- Added `atcab_verify_validate()` and `atcab_verify_invalidate()`
- Improvements to host functions to handle more cases.
- Added `atcab_updateextra()`, `atcab_derive_key()`
- Added support for more certificate formats.
- Added general purpose hardware SHA256 functions. See `atcab_hw_sha2_256()`.
- Removed device specific config read/write. Generic now handles both.
- Removed unnecessary response parameter from lock commands.
- Enhanced and added unit tests.
- Encrypted read and write functions now handle keys with `SlotConfig.NoMac` set
- `atcab_cmp_config_zone()` handles all devices now.
- Fixed some edge cases in `atcab_read_bytes_zone()`.
- Updated `atSHA()` to work with all devices.
- Fixed `atcacert_get_device_locs()` when using stored sn.

13.46 Release 01/08/2016

- New HAL implementations for
 - Single Wire interface for SAMD21 / SAMR21
 - SAMV71 I2C HAL implementation
 - XMega A3Bu HAL implementation
- Added `atcab_version()` method to return current version string of library to application
- New Bus and Discovery API
 - returns a list of ATCA device configurations for each CryptoAuth device found
 - currently implemented on SAMD21/R21 I2C, SAMV71
 - additional discovery implementations to come
- TLS APIs solidified and documented
- Added missing doxygen documentation for some CryptoAuthLib methods
- Stubs for HAL SPI removed as they are unused for SHA204A and ECC508A support
- bug fixes
- updated `atcab_sha()` to accept a variable length message that is > 64 bytes and not a multiple of 64 bytes (the SHA block size).
- refactored Cert I/O and Cert Data tests to be smaller
- 'uncrustify' source formatting
- published on GitHub

13.47 Release 9/19/2015

- Kit protocol over HID on Windows
- Kit protocol over CDC on Linux
- TLS integration with ATECC508A
- Certificate I/O and reconstruction
- New SHA2 implementation
- Major update to API docs, Doxygen files found in `cryptoauthlib/docs`
- load `cryptoauthlib/docs/index.html` with your browser

Chapter 14

Security Policy

We take the security of cryptauthlib very seriously. Please submit security vulnerabilities to the Microchip Product Security Incident Response Team (PSIRT) which is responsible for receiving and responding to reports of potential security vulnerabilities in our products, as well as in any related hardware, software, firmware, and tools. Please see below for instructions on how to submit your report.

14.1 Supported Versions

The previous API version is maintained for a year after a new version is released.

Version	Supported	Notes
3.7.x	:heavy_check_↔mark:	
3.6.x	:heavy_check_↔mark:	Support Ends September 8 2024
3.5.x	:heavy_check_↔mark:	Support Ends April 4 2024
3.4.x	:heavy_check_↔mark:	Support Ends March 14 2024
3.3.x	:x:	
3.2.x	:x:	
< 3.2	:x:	

14.2 Reporting a Vulnerability

How to Report Potential Product Security Vulnerabilities

Once a report is received, the PSIRT will take the necessary steps to review the issue and determine what actions might be required to address any potential impacts to our products. Microchip PSIRT follows a coordinated vulnerability responsible disclosure policy that is available for review.

Please use the above instructions to securely submit your findings - We ask that you refrain from reporting vulnerabilities through the public github issues system.

Chapter 15

Deprecated List

Global `atcab_init_device` (ATCADevice `ca_device`)

This function is not recommended for use generally. Use of `_ext` is recommended instead. You can use `atcab↵_init_ext` to obtain an initialized instance and associated it with the global structure - but this shouldn't be a required process except in extremely unusual circumstances.

Global `atidle` (ATCAIface `ca_iface`)

This function does not have defined behavior when `ATCA_HAL_LEGACY_API` is undefined.

Global `atsleep` (ATCAIface `ca_iface`)

This function does not have defined behavior when `ATCA_HAL_LEGACY_API` is undefined.

Global `atwake` (ATCAIface `ca_iface`)

This function does not have defined behavior when `ATCA_HAL_LEGACY_API` is undefined.

Chapter 16

Module Index

16.1 Modules

Here is a list of all modules:

TNG API (tng_)	81
Basic Crypto API methods (atcab_)	88
Configuration (cfg_)	171
ATCADevice (atca_)	171
ATCAIface (atca_)	174
Certificate manipulation methods (atcacert_)	182
Basic Crypto API methods for CryptoAuth Devices (calib_)	202
Software crypto methods (atcac_)	214
Hardware abstraction layer (hal_)	214
Host side crypto methods (atcah_)	253
JSON Web Token (JWT) methods (atca_jwt_)	258
mbedTLS Wrapper methods (atca_mbedtls_)	258
Attributes (pkcs11_attr_)	260

Chapter 17

Namespace Index

17.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

cryptoauthlib	275
cryptoauthlib.atcab	275
cryptoauthlib.atcacert	325
cryptoauthlib.atcaenum	330
cryptoauthlib.atjwt	331
cryptoauthlib.device	331
cryptoauthlib.exceptions	332
cryptoauthlib.iface	333
cryptoauthlib.library	335
cryptoauthlib.sha206_api	342
cryptoauthlib.status	347
cryptoauthlib.tng	348
test_device	352
test_iface	354

Chapter 18

Hierarchical Index

18.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

_ascii_kit_host_context	355
cryptoauthlib.library._CtypeIterator	364
_kit_host_map_entry	364
atca_check_mac_in_out	371
atca_decrypt_in_out	372
atca_delete_in_out	372
atca_derive_key_in_out	373
atca_derive_key_mac_in_out	373
atca_device	374
atca_diversified_key_in_out	375
atca_evp_ctx	375
atca_gen_dig_in_out	375
atca_gen_key_in_out	376
atca_hal_kit_phy_t	377
atca_hal_list_entry_t	378
atca_hal_shm_t	378
atca_hmac_in_out	378
atca_i2c_host_s	380
atca_iface	380
atca_include_data_in_out	381
atca_io_decrypt_in_out	381
atca_mac_in_out	381
atca_mbedtls_eckey_s	382
atca_nonce_in_out	382
atca_resp_mac_in_out	383
atca_secureboot_enc_in_out	384
atca_secureboot_mac_in_out	384
atca_session_key_in_out	384
atca_sha256_ctx	385
atca_sign_internal_in_out	386
atca_spi_host_s	386
atca_temp_key	387
atca_uart_host_s	387
atca_verify_in_out	387
atca_verify_mac	388

atca_write_mac_in_out	388
atcac_aes_cmac_ctx	393
atcac_aes_gcm_ctx	393
atcac_hmac_ctx	394
atcac_pk_ctx	394
atcac_sha1_ctx	394
atcac_sha2_256_ctx	394
atcac_sha2_384_ctx	394
atcac_sha2_512_ctx	394
atcac_x509_ctx	395
atcacert_build_state_s	395
atcacert_cert_element_s	395
atcacert_cert_loc_s	397
atcacert_def_s	402
atcacert_device_loc_s	403
atcacert_tm_utc_s	406
ATCAHAL_t	411
atcal2Cmaster	411
ATCAIfaceCfg	412
ATCAPacket	416
cryptoauthlib.library.AtcaReference	416
atcaSWImaster	418
atecc508a_config_s	420
atecc608_config_s	422
atsha204a_config_s	424
cal_buffer_s	427
calib_packet_cache_s	428
CK_AES_CBC_ENCRYPT_DATA_PARAMS	432
CK_AES_CCM_PARAMS	432
CK_AES_CTR_PARAMS	432
CK_AES_GCM_PARAMS	432
CK_ARIA_CBC_ENCRYPT_DATA_PARAMS	432
CK_ATTRIBUTE	433
CK_C_INITIALIZE_ARGS	433
CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS	433
CK_CAMELLIA_CTR_PARAMS	433
CK_CCM_PARAMS	433
CK_CMS_SIG_PARAMS	434
CK_DATE	434
CK_DES_CBC_ENCRYPT_DATA_PARAMS	434
CK_DSA_PARAMETER_GEN_PARAM	434
CK_ECDH1_DERIVE_PARAMS	434
CK_ECDH2_DERIVE_PARAMS	435
CK_ECDH_AES_KEY_WRAP_PARAMS	435
CK_ECMQV_DERIVE_PARAMS	435
CK_FUNCTION_LIST	435
CK_GCM_PARAMS	436
CK_GOSTR3410_DERIVE_PARAMS	436
CK_GOSTR3410_KEY_WRAP_PARAMS	436
CK_INFO	436
CK_KEA_DERIVE_PARAMS	437
CK_KEY_DERIVATION_STRING_DATA	437
CK_KEY_WRAP_SET_OAEP_PARAMS	437
CK_KIP_PARAMS	437
CK_MECHANISM	437
CK_MECHANISM_INFO	438
CK_OTP_PARAM	438
CK_OTP_PARAMS	438

CK_OTP_SIGNATURE_INFO	438
CK_PBE_PARAMS	438
CK_PKCS5_PBKD2_PARAMS	439
CK_PKCS5_PBKD2_PARAMS2	439
CK_RC2_CBC_PARAMS	439
CK_RC2_MAC_GENERAL_PARAMS	439
CK_RC5_CBC_PARAMS	440
CK_RC5_MAC_GENERAL_PARAMS	440
CK_RC5_PARAMS	440
CK_RSA_AES_KEY_WRAP_PARAMS	440
CK_RSA_PKCS_OAEP_PARAMS	440
CK_RSA_PKCS_PSS_PARAMS	441
CK_SEED_CBC_ENCRYPT_DATA_PARAMS	441
CK_SESSION_INFO	441
CK_SKIPJACK_PRIVATE_WRAP_PARAMS	441
CK_SKIPJACK_RELAYX_PARAMS	442
CK_SLOT_INFO	442
CK_SSL3_KEY_MAT_OUT	442
CK_SSL3_KEY_MAT_PARAMS	442
CK_SSL3_MASTER_KEY_DERIVE_PARAMS	443
CK_SSL3_RANDOM_DATA	443
CK_TLS12_KEY_MAT_PARAMS	443
CK_TLS12_MASTER_KEY_DERIVE_PARAMS	443
CK_TLS_KDF_PARAMS	443
CK_TLS_MAC_PARAMS	444
CK_TLS_PRF_PARAMS	444
CK_TOKEN_INFO	444
CK_VERSION	444
CK_WTLS_KEY_MAT_OUT	445
CK_WTLS_KEY_MAT_PARAMS	445
CK_WTLS_MASTER_KEY_DERIVE_PARAMS	445
CK_WTLS_PRF_PARAMS	445
CK_WTLS_RANDOM_DATA	445
CK_X9_42_DH1_DERIVE_PARAMS	446
CK_X9_42_DH2_DERIVE_PARAMS	446
CK_X9_42_MQV_DERIVE_PARAMS	446
CL_HashContext	446
device_execution_time_t	451
devtype_names_t	451
Exception	
cryptoauthlib.exceptions.CryptoError	450
cryptoauthlib.exceptions.AssertionFailure	366
cryptoauthlib.exceptions.BadArgumentError	425
cryptoauthlib.exceptions.BadCrcError	426
cryptoauthlib.exceptions.BadOpcodeError	426
cryptoauthlib.exceptions.CheckmacVerifyFailedError	429
cryptoauthlib.exceptions.CommunicationError	447
cryptoauthlib.exceptions.ConfigZoneLockedError	447
cryptoauthlib.exceptions.CrcError	449
cryptoauthlib.exceptions.DataZoneLockedError	450
cryptoauthlib.exceptions.EccFaultError	451
cryptoauthlib.exceptions.ExecutionError	452
cryptoauthlib.exceptions.FunctionError	452
cryptoauthlib.exceptions.GenericError	452
cryptoauthlib.exceptions.HealthTestError	453
cryptoauthlib.exceptions.InvalidIdentifierError	456
cryptoauthlib.exceptions.InvalidSizeError	457
cryptoauthlib.exceptions.LibraryLoadError	458

cryptoauthlib.exceptions.LibraryMemoryError	458
cryptoauthlib.exceptions.LibraryNotInitialized	459
cryptoauthlib.exceptions.NoDevicesFoundError	459
cryptoauthlib.exceptions.NoResponseError	460
cryptoauthlib.exceptions.NoUseFlagError	460
cryptoauthlib.exceptions.ParityError	460
cryptoauthlib.exceptions.ParseError	461
cryptoauthlib.exceptions.ReceiveError	468
cryptoauthlib.exceptions.ReceiveTimeoutError	469
cryptoauthlib.exceptions.ResyncWithWakeupError	469
cryptoauthlib.exceptions.StatusUnknownError	473
cryptoauthlib.exceptions.TimeOutError	473
cryptoauthlib.exceptions.TransmissionError	473
cryptoauthlib.exceptions.TransmissionTimeoutError	474
cryptoauthlib.exceptions.UnimplementedError	474
cryptoauthlib.exceptions.UnsupportedInterface	474
cryptoauthlib.exceptions.WakeFailedError	476
cryptoauthlib.exceptions.ZoneNotLockedError	477
i2c_sam0_instance	455
i2c_sam_instance	455
i2c_start_instance	455
Jwt	
cryptoauthlib.atjwt.PyJWT	468
memory_parameters	459
object	
cryptoauthlib_mock.atcab_mock	389
pkcs11_mech_table_e	461
pkcs11_attr_model_s	461
pkcs11_conf_filedata_s	461
pkcs11_dev_ctx	462
pkcs11_dev_res	462
pkcs11_dev_state	462
pkcs11_ecc_key_info_s	463
pkcs11_key_info_s	463
pkcs11_lib_ctx_s	463
pkcs11_object_cache_s	465
pkcs11_object_s	465
pkcs11_rsa_key_info_s	466
pkcs11_session_ctx_s	466
pkcs11_session_mech_ctx_s	467
pkcs11_slot_ctx_s	467
secure_boot_config_bits	469
secure_boot_parameters	470
Union	
cryptoauthlib.library.AtcaUnion	419
cryptoauthlib.iface._ATCAIfaceParams	359
cryptoauthlib.iface._U_Address	365
build_ext	
setup.CryptoAuthCommandBuildExt	449
Distribution	
setup.BinaryDistribution	426
ECAAlgorithm	
cryptoauthlib.atjwt.HwEcAlgorithm	453
Enum	
cryptoauthlib.atcaenum.AtcaEnum	410
cryptoauthlib.atcacert.CertStatus	428
cryptoauthlib.atcacert.atcacert_cert_sn_src_t	398
cryptoauthlib.atcacert.atcacert_cert_type_t	399

cryptoauthlib.atcacert.atcacert_date_format_t	401
cryptoauthlib.atcacert.atcacert_device_zone_t	405
cryptoauthlib.atcacert.atcacert_std_cert_element_t	405
cryptoauthlib.atcacert.atcacert_transform_t	407
cryptoauthlib.iface.ATCADeviceType	408
cryptoauthlib.iface.ATCAIfaceType	414
cryptoauthlib.iface.ATCAKitType	415
cryptoauthlib.status.Status	471
HMACAAlgorithm	
cryptoauthlib.atjwt.HwHmacAlgorithm	454
install	
setup.CryptoAuthCommandInstall	449
Structure	
cryptoauthlib.atcab.atca_aes_cbc_ctx	367
cryptoauthlib.atcab.atca_aes_cbcmac_ctx	367
cryptoauthlib.atcab.atca_aes_ccm_ctx	368
cryptoauthlib.atcab.atca_aes_cmac_ctx	369
cryptoauthlib.atcab.atca_aes_ctr_ctx	370
cryptoauthlib.atcab.atca_aes_gcm_ctx	370
cryptoauthlib.atcab.atca_sha256_ctx	385
cryptoauthlib.atcab.atca_hmac_sha256_ctx	379
cryptoauthlib.atcacert.atcacert_tm_utc_t	407
cryptoauthlib.device.AesEnable	366
cryptoauthlib.device.ChipMode508	429
cryptoauthlib.device.ChipMode608	430
cryptoauthlib.device.ChipOptions	431
cryptoauthlib.device.CountMatch	448
cryptoauthlib.device.Counter204	447
cryptoauthlib.device.I2cEnable	456
cryptoauthlib.device.KeyConfig	457
cryptoauthlib.device.SecureBoot	470
cryptoauthlib.device.SlotConfig	471
cryptoauthlib.device.UseLock	475
cryptoauthlib.device.VolatileKeyPermission	475
cryptoauthlib.device.X509Format	476
cryptoauthlib.library.AtcaStructure	417
cryptoauthlib.atcacert.atcacert_cert_element_t	396
cryptoauthlib.atcacert.atcacert_cert_loc_t	397
cryptoauthlib.atcacert.atcacert_comp_data_t	400
cryptoauthlib.atcacert.atcacert_def_t	402
cryptoauthlib.atcacert.atcacert_device_loc_t	404
cryptoauthlib.device.Atecc508aConfig	420
cryptoauthlib.device.Atecc608Config	422
cryptoauthlib.device.Atsha204aConfig	424
cryptoauthlib.iface.ATCAIfaceCfg	413
cryptoauthlib.iface._ATCACUSTOM	355
cryptoauthlib.iface._ATCAHID	356
cryptoauthlib.iface._ATCAI2C	357
cryptoauthlib.iface._ATCAKIT	360
cryptoauthlib.iface._ATCASPI	361
cryptoauthlib.iface._ATCASWI	362
cryptoauthlib.iface._ATCAUART	363
cryptoauthlib.library.cal_buffer	427

Chapter 19

Data Structure Index

19.1 Data Structures

Here are the data structures with brief descriptions:

_ascii_kit_host_context	355
cryptoauthlib.iface._ATCACUSTOM	355
cryptoauthlib.iface._ATCAHID	356
cryptoauthlib.iface._ATCAI2C	357
cryptoauthlib.iface._ATCAIfaceParams	359
cryptoauthlib.iface._ATCAKIT	360
cryptoauthlib.iface._ATCASPI	361
cryptoauthlib.iface._ATCASWI	362
cryptoauthlib.iface._ATCAUART	363
cryptoauthlib.library._Ctypeliterator	364
_kit_host_map_entry	364
cryptoauthlib.iface._U_Address	365
cryptoauthlib.device.AesEnable	366
cryptoauthlib.exceptions.AssertionFailure	366
cryptoauthlib.atcab.atca_aes_cbc_ctx	367
cryptoauthlib.atcab.atca_aes_cbcmac_ctx	367
cryptoauthlib.atcab.atca_aes_ccm_ctx	368
cryptoauthlib.atcab.atca_aes_cmac_ctx	369
cryptoauthlib.atcab.atca_aes_ctr_ctx	370
cryptoauthlib.atcab.atca_aes_gcm_ctx	370
atca_check_mac_in_out	
Input/output parameters for function atcah_check_mac()	371
atca_decrypt_in_out	
Input/output parameters for function atca_decrypt()	372
atca_delete_in_out	
Input/Output paramters for calculating the mac.Used with Delete command	372
atca_derive_key_in_out	
Input/output parameters for function atcah_derive_key()	373
atca_derive_key_mac_in_out	
Input/output parameters for function atcah_derive_key_mac()	373
atca_device	
Atca_device is the C object backing ATCADevice. See the atca_device.h file for details on the ATCADevice methods	374
atca_diversified_key_in_out	
Input/output parameters for function atcah_gendivkey()	375

atca_evp_ctx	375
atca_gen_dig_in_out	
Input/output parameters for function atcah_gen_dig()	375
atca_gen_key_in_out	
Input/output parameters for calculating the PubKey digest put into TempKey by the GenKey command with the atcah_gen_key_msg() function	376
atca_hal_kit_phy_t	377
atca_hal_list_entry_t	
Structure that holds the hal/phy mapping for different interface types	378
atca_hal_shm_t	378
atca_hmac_in_out	
Input/output parameters for function atca_hmac()	378
cryptoauthlib.atcab.atca_hmac_sha256_ctx	379
atca_i2c_host_s	380
atca_iface	
Atca_iface is the context structure for a configured interface	380
atca_include_data_in_out	
Input / output parameters for function atca_include_data()	381
atca_io_decrypt_in_out	381
atca_mac_in_out	
Input/output parameters for function atca_mac()	381
atca_mbedtls_eckey_s	382
atca_nonce_in_out	
Input/output parameters for function atca_nonce()	382
atca_resp_mac_in_out	
Input/Output parameters for calculating the output response mac in SHA105 device. Used with the atcah_gen_output_resp_mac() function	383
atca_secureboot_enc_in_out	384
atca_secureboot_mac_in_out	384
atca_session_key_in_out	
Input/Output paramters for calculating the session key by the nonce command. Used with the atcah_gen_session_key() function	384
atca_sha256_ctx	385
cryptoauthlib.atcab.atca_sha256_ctx	385
atca_sign_internal_in_out	
Input/output parameters for calculating the message and digest used by the Sign(internal) command. Used with the atcah_sign_internal_msg() function	386
atca_spi_host_s	386
atca_temp_key	
Structure to hold TempKey fields	387
atca_uart_host_s	387
atca_verify_in_out	
Input/output parameters for function atcah_verify()	387
atca_verify_mac	388
atca_write_mac_in_out	
Input/output parameters for function atcah_write_auth_mac() and atcah_privwrite_auth_mac()	388
cryptoauthlib_mock.atcab_mock	389
atcac_aes_cmac_ctx	393
atcac_aes_gcm_ctx	393
atcac_hmac_ctx	394
atcac_pk_ctx	394
atcac_sha1_ctx	394
atcac_sha2_256_ctx	394
atcac_sha2_384_ctx	394
atcac_sha2_512_ctx	394
atcac_x509_ctx	395
atcacert_build_state_s	395
atcacert_cert_element_s	395

cryptoauthlib.atcacert.atcacert_cert_element_t	396
atcacert_cert_loc_s	397
cryptoauthlib.atcacert.atcacert_cert_loc_t	397
cryptoauthlib.atcacert.atcacert_cert_sn_src_t	398
cryptoauthlib.atcacert.atcacert_cert_type_t	399
cryptoauthlib.atcacert.atcacert_comp_data_t	400
cryptoauthlib.atcacert.atcacert_date_format_t	401
atcacert_def_s	402
cryptoauthlib.atcacert.atcacert_def_t	402
atcacert_device_loc_s	403
cryptoauthlib.atcacert.atcacert_device_loc_t	404
cryptoauthlib.atcacert.atcacert_device_zone_t	405
cryptoauthlib.atcacert.atcacert_std_cert_element_t	405
atcacert_tm_utc_s	406
cryptoauthlib.atcacert.atcacert_tm_utc_t	407
cryptoauthlib.atcacert.atcacert_transform_t	407
cryptoauthlib.iface.ATCADeviceType	408
cryptoauthlib.atcaenum.AtcaEnum	410
ATCAHAL_t	
HAL Driver Structure	411
atcal2Cmaster	
This is the hal_data for ATCA HAL for ASF SERCOM	411
ATCAIfaceCfg	412
cryptoauthlib.iface.ATCAIfaceCfg	413
cryptoauthlib.iface.ATCAIfaceType	414
cryptoauthlib.iface.ATCAKitType	415
ATCAPacket	416
cryptoauthlib.library.AtcaReference	416
cryptoauthlib.library.AtcaStructure	417
atcaSWImaster	
This is the hal_data for ATCA HAL for ASF SERCOM	418
cryptoauthlib.library.AtcaUnion	419
atecc508a_config_s	420
cryptoauthlib.device.Atecc508aConfig	420
atecc608_config_s	422
cryptoauthlib.device.Atecc608Config	422
atsha204a_config_s	424
cryptoauthlib.device.Atsha204aConfig	424
cryptoauthlib.exceptions.BadArgumentError	425
cryptoauthlib.exceptions.BadCrcError	426
cryptoauthlib.exceptions.BadOpcodeError	426
setup.BinaryDistribution	426
cryptoauthlib.library.cal_buffer	427
cal_buffer_s	427
calib_packet_cache_s	428
cryptoauthlib.atcacert.CertStatus	428
cryptoauthlib.exceptions.CheckmacVerifyFailedError	429
cryptoauthlib.device.ChipMode508	429
cryptoauthlib.device.ChipMode608	430
cryptoauthlib.device.ChipOptions	431
CK_AES_CBC_ENCRYPT_DATA_PARAMS	432
CK_AES_CCM_PARAMS	432
CK_AES_CTR_PARAMS	432
CK_AES_GCM_PARAMS	432
CK_ARIA_CBC_ENCRYPT_DATA_PARAMS	432
CK_ATTRIBUTE	433
CK_C_INITIALIZE_ARGS	433
CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS	433

CK_CAMELLIA_CTR_PARAMS	433
CK_CCM_PARAMS	433
CK_CMS_SIG_PARAMS	434
CK_DATE	434
CK_DES_CBC_ENCRYPT_DATA_PARAMS	434
CK_DSA_PARAMETER_GEN_PARAM	434
CK_ECDH1_DERIVE_PARAMS	434
CK_ECDH2_DERIVE_PARAMS	435
CK_ECDH_AES_KEY_WRAP_PARAMS	435
CK_ECMQV_DERIVE_PARAMS	435
CK_FUNCTION_LIST	435
CK_GCM_PARAMS	436
CK_GOSTR3410_DERIVE_PARAMS	436
CK_GOSTR3410_KEY_WRAP_PARAMS	436
CK_INFO	436
CK_KEA_DERIVE_PARAMS	437
CK_KEY_DERIVATION_STRING_DATA	437
CK_KEY_WRAP_SET_OAEP_PARAMS	437
CK_KIP_PARAMS	437
CK_MECHANISM	437
CK_MECHANISM_INFO	438
CK_OTP_PARAM	438
CK_OTP_PARAMS	438
CK_OTP_SIGNATURE_INFO	438
CK_PBE_PARAMS	438
CK_PKCS5_PBKD2_PARAMS	439
CK_PKCS5_PBKD2_PARAMS2	439
CK_RC2_CBC_PARAMS	439
CK_RC2_MAC_GENERAL_PARAMS	439
CK_RC5_CBC_PARAMS	440
CK_RC5_MAC_GENERAL_PARAMS	440
CK_RC5_PARAMS	440
CK_RSA_AES_KEY_WRAP_PARAMS	440
CK_RSA_PKCS_OAEP_PARAMS	440
CK_RSA_PKCS_PSS_PARAMS	441
CK_SEED_CBC_ENCRYPT_DATA_PARAMS	441
CK_SESSION_INFO	441
CK_SKIPJACK_PRIVATE_WRAP_PARAMS	441
CK_SKIPJACK_RELAYX_PARAMS	442
CK_SLOT_INFO	442
CK_SSL3_KEY_MAT_OUT	442
CK_SSL3_KEY_MAT_PARAMS	442
CK_SSL3_MASTER_KEY_DERIVE_PARAMS	443
CK_SSL3_RANDOM_DATA	443
CK_TLS12_KEY_MAT_PARAMS	443
CK_TLS12_MASTER_KEY_DERIVE_PARAMS	443
CK_TLS_KDF_PARAMS	443
CK_TLS_MAC_PARAMS	444
CK_TLS_PRF_PARAMS	444
CK_TOKEN_INFO	444
CK_VERSION	444
CK_WTLS_KEY_MAT_OUT	445
CK_WTLS_KEY_MAT_PARAMS	445
CK_WTLS_MASTER_KEY_DERIVE_PARAMS	445
CK_WTLS_PRF_PARAMS	445
CK_WTLS_RANDOM_DATA	445
CK_X9_42_DH1_DERIVE_PARAMS	446
CK_X9_42_DH2_DERIVE_PARAMS	446

CK_X9_42_MQV_DERIVE_PARAMS	446
CL_HashContext	446
cryptoauthlib.exceptions.CommunicationError	447
cryptoauthlib.exceptions.ConfigZoneLockedError	447
cryptoauthlib.device.Counter204	447
cryptoauthlib.device.CountMatch	448
cryptoauthlib.exceptions.CrcError	449
setup.CryptoAuthCommandBuildExt	449
setup.CryptoAuthCommandInstall	449
cryptoauthlib.exceptions.CryptoError	450
cryptoauthlib.exceptions.DataZoneLockedError	450
device_execution_time_t	
Structure to hold the device execution time and the opcode for the corresponding command	451
devtype_names_t	451
cryptoauthlib.exceptions.EccFaultError	451
cryptoauthlib.exceptions.ExecutionError	452
cryptoauthlib.exceptions.FunctionError	452
cryptoauthlib.exceptions.GenericError	452
cryptoauthlib.exceptions.HealthTestError	453
cryptoauthlib.atjwt.HwEcAlgorithm	453
cryptoauthlib.atjwt.HwHmacAlgorithm	454
i2c_sam0_instance	455
i2c_sam_instance	455
i2c_start_instance	455
cryptoauthlib.device.I2cEnable	456
cryptoauthlib.exceptions.InvalidIdentifierError	456
cryptoauthlib.exceptions.InvalidSizeError	457
cryptoauthlib.device.KeyConfig	457
cryptoauthlib.exceptions.LibraryLoadError	458
cryptoauthlib.exceptions.LibraryMemoryError	458
cryptoauthlib.exceptions.LibraryNotInitialized	459
memory_parameters	459
cryptoauthlib.exceptions.NoDevicesFoundError	459
cryptoauthlib.exceptions.NoResponseError	460
cryptoauthlib.exceptions.NoUseFlagError	460
cryptoauthlib.exceptions.ParityError	460
cryptoauthlib.exceptions.ParseError	461
pkcs11_mech_table_e	461
pkcs11_attrib_model_s	461
pkcs11_conf_filedata_s	461
pkcs11_dev_ctx	462
pkcs11_dev_res	462
pkcs11_dev_state	462
pkcs11_ecc_key_info_s	463
pkcs11_key_info_s	463
pkcs11_lib_ctx_s	463
pkcs11_object_cache_s	465
pkcs11_object_s	465
pkcs11_rsa_key_info_s	466
pkcs11_session_ctx_s	466
pkcs11_session_mech_ctx_s	467
pkcs11_slot_ctx_s	467
cryptoauthlib.atjwt.PyJWT	468
cryptoauthlib.exceptions.ReceiveError	468
cryptoauthlib.exceptions.ReceiveTimeoutError	469
cryptoauthlib.exceptions.ResyncWithWakeupError	469
secure_boot_config_bits	469
secure_boot_parameters	470

cryptoauthlib.device.SecureBoot	470
cryptoauthlib.device.SlotConfig	471
cryptoauthlib.status.Status	471
cryptoauthlib.exceptions.StatusUnknownError	473
cryptoauthlib.exceptions.TimeOutError	473
cryptoauthlib.exceptions.TransmissionError	473
cryptoauthlib.exceptions.TransmissionTimeoutError	474
cryptoauthlib.exceptions.UnimplementedError	474
cryptoauthlib.exceptions.UnsupportedInterface	474
cryptoauthlib.device.UseLock	475
cryptoauthlib.device.VolatileKeyPermission	475
cryptoauthlib.exceptions.WakeFailedError	476
cryptoauthlib.device.X509Format	476
cryptoauthlib.exceptions.ZoneNotLockedError	477

Chapter 20

File Index

20.1 File List

Here is a list of all documented files with brief descriptions:

api_206a.c	Provides APIs to use with ATSHA206A device	479
api_206a.h	Provides api interfaces to use with ATSHA206A device	485
symmetric_authentication.c	Contains API for performing the symmetric Authentication between the Host and the device . .	492
symmetric_authentication.h	Contains API for performing the symmetric Authentication between the Host and the device . .	493
ascii_kit_host.c	KIT protocol interpreter	494
ascii_kit_host.h	KIT protocol interpreter	495
trust_pkcs11_config.c	PKCS11 Trust Platform Configuration	498
io_protection_key.h	Provides required interface to access IO protection key	498
secure_boot.c	Provides required APIs to manage secure boot under various scenarios	499
secure_boot.h	Provides required APIs to manage secure boot under various scenarios	500
secure_boot_memory.h	Provides interface to memory component for the secure boot	502
tflxtls_cert_def_4_device.c	TNG TLS device certificate definition	502
tflxtls_cert_def_4_device.h	TNG TLS device certificate definition	503
tng_atca.c	TNG Helper Functions	503
tng_atca.h	TNG Helper Functions	504
tng_atcacert_client.c	Client side certificate I/O functions for TNG devices	504
tng_atcacert_client.h	Client side certificate I/O functions for TNG devices	505
tng_root_cert.c	TNG root certificate (DER)	505

tng_root_cert.h	
TNG root certificate (DER)	506
tnglora_cert_def_1_signer.c	
TNG LORA signer certificate definition	506
tnglora_cert_def_1_signer.h	
TNG LORA signer certificate definition	507
tnglora_cert_def_2_device.c	
TNG LORA device certificate definition	507
tnglora_cert_def_2_device.h	
TNG LORA device certificate definition	507
tnglora_cert_def_4_device.c	
TNG LORA device certificate definition	508
tnglora_cert_def_4_device.h	
TNG LORA device certificate definition	508
tngtls_cert_def_1_signer.c	
TNG TLS signer certificate definition	509
tngtls_cert_def_1_signer.h	
TNG TLS signer certificate definition	509
tngtls_cert_def_2_device.c	
TNG TLS device certificate definition	510
tngtls_cert_def_2_device.h	
TNG TLS device certificate definition	510
tngtls_cert_def_3_device.c	
TNG TLS device certificate definition	510
tngtls_cert_def_3_device.h	
TNG TLS device certificate definition	511
wpc_apis.c	
Provides api interfaces for WPC authentication	511
wpc_apis.h	
Provides api interfaces for WPC authentication	512
wpccert_client.c	
Provides api interfaces for accessing WPC certificates from device	513
wpccert_client.h	
Provides api interfaces for accessing WPC certificates from device	514
atca_basic.c	
CryptoAuthLib Basic API methods. These methods provide a simpler way to access the core crypto methods	515
atca_basic.h	
CryptoAuthLib Basic API methods - a simple crypto authentication API. These methods manage a global ATCADevice object behind the scenes. They also manage the wake/idle state transitions so callers don't need to	523
atca_cfgs.c	
Set of default configurations for various ATCA devices and interfaces	532
atca_cfgs.h	
Set of default configurations for various ATCA devices and interfaces	532
atca_compiler.h	
CryptoAuthLib is meant to be portable across architectures, even non-Microchip architectures and compiler environments. This file is for isolating compiler specific macros	533
atca_config_check.h	
Consistency checks for configuration options	533
atca_debug.c	
Debug/Trace for CryptoAuthLib calls	538
atca_device.c	
Microchip CryptoAuth device object	539
atca_device.h	
Microchip Crypto Auth device object	539
atca_devtypes.h	
Microchip Crypto Auth	540

atca_helpers.c	Helpers to support the CryptoAuthLib Basic API methods	541
atca_helpers.h	Helpers to support the CryptoAuthLib Basic API methods	550
atca_iface.c	Microchip CryptoAuthLib hardware interface object	551
atca_iface.h	Microchip Crypto Auth hardware interface object	553
atca_platform.h	Configure the platform interfaces for cryptoauthlib	556
atca_status.h	Microchip Crypto Auth status codes	557
atca_utils_sizes.c	API to Return structure sizes of cryptoauthlib structures	563
atca_version.h	Microchip CryptoAuth Library Version	564
atcacert.h	Declarations common to all atcacert code	564
atcacert_check_config.h	Configuration check and defaults for the atcacert module	565
atcacert_client.c	Client side cert i/o methods. These declarations deal with the client-side, the node being authenticated, of the authentication process. It is assumed the client has an ECC CryptoAuthentication device (e.g. ATECC508A) and the certificates are stored on that device	566
atcacert_client.h	Client side cert i/o methods. These declarations deal with the client-side, the node being authenticated, of the authentication process. It is assumed the client has an ECC CryptoAuthentication device (e.g. ATECC508A) and the certificates are stored on that device	566
atcacert_date.c	Date handling with regard to certificates	570
atcacert_date.h	Declarations for date handling with regard to certificates	571
atcacert_def.c	Main certificate definition implementation	573
atcacert_def.h	Declarations for certificates related to ECC CryptoAuthentication devices. These are the definitions required to define a certificate and its various elements with regards to the CryptoAuthentication ECC devices	573
atcacert_der.c	Functions required to work with DER encoded data related to X.509 certificates	575
atcacert_der.h	Function declarations required to work with DER encoded data related to X.509 certificates	576
atcacert_host_hw.c	Host side methods using CryptoAuth hardware	577
atcacert_host_hw.h	Host side methods using CryptoAuth hardware	577
atcacert_host_sw.c	Host side methods using software implementations	577
atcacert_host_sw.h	Host side methods using software implementations. host-side, the one authenticating a client, of the authentication process. Crypto functions are performed using a software library	578
atcacert_pem.c	Functions required to work with PEM encoded data related to X.509 certificates	578
atcacert_pem.h	Functions for converting between DER and PEM formats	579
cal_buffer.c	Cryptoauthlib buffer management system	582

cal_buffer.h	CryptoAuthLib buffer management system	585
cal_internal.h	Internal CryptoAuthLib Interfaces	588
calib_aes.c	CryptoAuthLib Basic API methods for AES command	588
calib_aes_gcm.c	CryptoAuthLib Basic API methods for AES GCM mode	589
calib_aes_gcm.h	Unity tests for the cryptoauthlib AES GCM functions	589
calib_basic.c	CryptoAuthLib Basic API methods. These methods provide a simpler way to access the core crypto methods	589
calib_checkmac.c	CryptoAuthLib Basic API methods for CheckMAC command	590
calib_command.c	Microchip CryptoAuthentication device command builder - this is the main object that builds the command byte strings for the given device. It does not execute the command. The basic flow is to call a command method to build the command you want given the parameters and then send that byte string through the device interface	591
calib_command.h	Microchip Crypto Auth device command object - this is a command builder only, it does not send the command. The result of a command method is a fully formed packet, ready to send to the ATCAIFace object to dispatch	594
calib_config_check.h	Consistency checks for configuration options	615
calib_counter.c	CryptoAuthLib Basic API methods for Counter command	620
calib_delete.c	CryptoAuthLib Basic API methods for Delete command	620
calib_derivekey.c	CryptoAuthLib Basic API methods for DeriveKey command	621
calib_device.h	Microchip Crypto Auth Device Data	621
calib_ecdh.c	CryptoAuthLib Basic API methods for ECDH command	624
calib_execution.c	Implements an execution handler that executes a given command on a device and returns the results	625
calib_execution.h	Defines an execution handler that executes a given command on a device and returns the results	626
calib_gendig.c	CryptoAuthLib Basic API methods for GenDig command	628
calib_genkey.c	CryptoAuthLib Basic API methods for GenKey command	629
calib_helpers.c	CryptoAuthLib Basic API - Helper Functions to	629
calib_hmac.c	CryptoAuthLib Basic API methods for HMAC command	630
calib_info.c	CryptoAuthLib Basic API methods for Info command	630
calib_kdf.c	CryptoAuthLib Basic API methods for KDF command	631
calib_lock.c	CryptoAuthLib Basic API methods for Lock command	631
calib_mac.c	CryptoAuthLib Basic API methods for MAC command	632

calib_nonce.c	CryptoAuthLib Basic API methods for Nonce command	632
calib_packet.c	CryptoAuthLib API for packet allocation	633
calib_packet.h	Defines packet allocation functions	634
calib_privwrite.c	CryptoAuthLib Basic API methods for PrivWrite command	634
calib_random.c	CryptoAuthLib Basic API methods for Random command	635
calib_read.c	CryptoAuthLib Basic API methods for Read command	635
calib_secureboot.c	CryptoAuthLib Basic API methods for SecureBoot command	636
calib_selftest.c	CryptoAuthLib Basic API methods for SelfTest command	636
calib_sha.c	CryptoAuthLib Basic API methods for SHA command	637
calib_sign.c	CryptoAuthLib Basic API methods for Sign command	637
calib_updateextra.c	CryptoAuthLib Basic API methods for UpdateExtra command	638
calib_verify.c	CryptoAuthLib Basic API methods for Verify command	638
calib_write.c	CryptoAuthLib Basic API methods for Write command	639
atca_crypto_hw_aes.h	AES CTR, CBC & CMAC structure definitions	639
atca_crypto_hw_aes_cbc.c	CryptoAuthLib Basic API methods for AES CBC mode	640
atca_crypto_hw_aes_cbcmac.c	CryptoAuthLib Basic API methods for AES CBC_MAC mode	640
atca_crypto_hw_aes_ccm.c	CryptoAuthLib Basic API methods for AES CCM mode	641
atca_crypto_hw_aes_cmac.c	CryptoAuthLib Basic API methods for AES CBC_MAC mode	641
atca_crypto_hw_aes_ctr.c	CryptoAuthLib Basic API methods for AES CTR mode	642
atca_crypto_pad.c	Implementation of PKCS7 Padding for block encryption	642
atca_crypto_pbkdf2.c	Implementation of the PBKDF2 algorithm for use in generating password hashes	642
atca_crypto_sw.h	Common defines for CryptoAuthLib software crypto wrappers	643
atca_crypto_sw_aes_cmac.c	Common Wrapper for host side AES-CMAC implementations that feature update APIs rather than an all at once implementation	643
atca_crypto_sw_aes_gcm.c	Common Wrapper for host side AES-GCM implementations that feature update APIs rather than an all at once implementation	644
atca_crypto_sw_sha1.c	Wrapper API for SHA 1 routines	644
atca_crypto_sw_sha1.h	Wrapper API for SHA 1 routines	645
atca_crypto_sw_sha2.c	Wrapper API for software SHA 256 routines	645
atca_crypto_sw_sha2.h	Wrapper API for software SHA 256 routines	645

crypto_hw_config_check.h	Consistency checks for configuration options	646
crypto_sw_config_check.h	Consistency checks for configuration options	648
sha1_routines.c	Software implementation of the SHA1 algorithm	654
sha1_routines.h	Software implementation of the SHA1 algorithm	654
sha2_routines.c	Software implementation of the SHA256, SHA384 and SHA512 algorithm	655
sha2_routines.h	Software implementation of the SHA256, SHA384 and SHA512 algorithm	655
cryptoauthlib.h	Single aggregation point for all CryptoAuthLib header files	656
atca_hal.c	Low-level HAL - methods used to setup indirection to physical layer interface. this level does the dirty work of abstracting the higher level ATCAIFace methods from the low-level physical interfaces. Its main goal is to keep low-level details from bleeding into the logical interface implementation	659
atca_hal.h	Low-level HAL - methods used to setup indirection to physical layer interface	659
hal_all_platforms_kit_hidapi.c	HAL for kit protocol over HID for any platform	661
hal_freertos.c	FreeRTOS Hardware/OS Abstraction Layer	662
hal_gpio_harmony.c	ATCA Hardware abstraction layer for GPIO	662
hal_i2c_harmony.c	ATCA Hardware abstraction layer for SAMD21 I2C over Harmony PLIB	665
hal_i2c_start.c	ATCA Hardware abstraction layer for SAMD21 I2C over START drivers	666
hal_i2c_start.h	ATCA Hardware abstraction layer for SAMD21 I2C over START drivers	667
hal_kit_bridge.c	Kit Bridging HAL for cryptoauthlib. This is not intended to be a zero copy driver. It should work with any interface that confirms to a few basic requirements: a) will accept an arbitrary number of bytes and packetize it if necessary for transmission, b) will block for the duration of the transmit	667
hal_kit_bridge.h	Kit Bridging HAL for cryptoauthlib. This is not intended to be a zero copy driver. It should work with any interface that confirms to a few basic requirements: a) will accept an arbitrary number of bytes and packetize it if necessary for transmission, b) will block for the duration of the transmit	668
hal_linux.c	Timer Utility Functions for Linux	669
hal_linux_i2c_userspace.c	ATCA Hardware abstraction layer for Linux using I2C	669
hal_linux_uart_userspace.c	ATCA Hardware abstraction layer for Linux using UART	670
hal_sam0_i2c_asf.c	ATCA Hardware abstraction layer for SAMD21 I2C over ASF drivers	674
hal_sam0_i2c_asf.h	ATCA Hardware abstraction layer for SAMD21 I2C over ASF drivers	675
hal_sam_i2c_asf.c	ATCA Hardware abstraction layer for SAM flexcom & twi I2C over ASF drivers	675
hal_sam_i2c_asf.h	ATCA Hardware abstraction layer for SAMG55 I2C over ASF drivers	676
hal_sam_timer_asf.c	ATCA Hardware abstraction layer for SAMD21 timer/delay over ASF drivers	677

hal_spi_harmony.c	ATCA Hardware abstraction layer for SPI over Harmony PLIB	678
hal_swi_gpio.c	ATCA Hardware abstraction layer for 1WIRE or SWI over GPIO	679
hal_swi_gpio.h	ATCA Hardware abstraction layer for SWI over GPIO drivers	682
hal_swi_uart.c	ATCA Hardware abstraction layer for SWI over UART drivers	684
hal_timer_start.c	ATCA Hardware abstraction layer for SAMD21 I2C over START drivers	685
hal_uart_harmony.c	ATCA Hardware abstraction layer for SWI uart over Harmony PLIB	686
hal_uc3_i2c_asf.c	ATCA Hardware abstraction layer for SAMV71 I2C over ASF drivers	690
hal_uc3_i2c_asf.h	ATCA Hardware abstraction layer for SAMV71 I2C over ASF drivers	691
hal_uc3_timer_asf.c	ATCA Hardware abstraction layer for SAM4S I2C over ASF drivers	692
hal_windows.c	ATCA Hardware abstraction layer for windows timer functions	692
hal_windows_kit_uart.c	ATCA Hardware abstraction layer for Windows using UART	693
kit_protocol.c	Microchip Crypto Auth hardware interface object	697
kit_protocol.h	698
swi_uart_samd21_asf.c	ATXMEGA's ATCA Hardware abstraction layer for SWI interface over UART drivers	699
swi_uart_samd21_asf.h	ATXMEGA's ATCA Hardware abstraction layer for SWI interface over UART drivers	700
swi_uart_start.c	701
swi_uart_start.h	702
atca_host.c	Host side methods to support CryptoAuth computations	703
atca_host.h	Definitions and Prototypes for ATCA Utility Functions	703
atca_host_config_check.h	Consistency checks for configuration options	706
atca_jwt.c	Utilities to create and verify a JSON Web Token (JWT)	712
atca_jwt.h	Utilities to create and verify a JSON Web Token (JWT)	713
atca_mbedtls_interface.h	Configuration Check for MbedTLS Integration Support	713
atca_mbedtls_wrap.c	Wrapper functions to replace cryptoauthlib software crypto functions with the mbedtls equivalent	715
atca_openssl_interface.c	Crypto abstraction functions for external host side cryptography	727
atca_openssl_interface.h	OpenSSL Integration Support	739
pkcs11_attr.c	PKCS11 Library Object Attributes Handling	742
pkcs11_attr.h	PKCS11 Library Object Attribute Handling	742
pkcs11_cert.c	PKCS11 Library Certificate Handling	743
pkcs11_cert.h	PKCS11 Library Certificate Handling	744

pkcs11_config.c	PKCS11 Library Configuration	745
pkcs11_debug.c	PKCS11 Library Debugging	746
pkcs11_debug.h	PKCS11 Library Debugging	747
pkcs11_digest.h	PKCS11 Library Digest (SHA256) Handling	747
pkcs11_encrypt.c	PKCS11 Library Encrypt Support	748
pkcs11_encrypt.h	PKCS11 Library AES Support	748
pkcs11_find.c	PKCS11 Library Object Find/Searching	749
pkcs11_find.h	PKCS11 Library Object Find/Searching	750
pkcs11_info.c	PKCS11 Library Information Functions	750
pkcs11_info.h	PKCS11 Library Information Functions	751
pkcs11_init.c	PKCS11 Library Init/Deinit	752
pkcs11_init.h	PKCS11 Library Initialization & Context	752
pkcs11_key.c	PKCS11 Library Key Object Handling	754
pkcs11_key.h	PKCS11 Library Object Handling	755
pkcs11_main.c	PKCS11 Basic library redirects based on the 2.40 specification docs.oasis-open.↵ org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.html	757
pkcs11_mech.c	PKCS11 Library Mechanism Handling	761
pkcs11_mech.h	PKCS11 Library Mechanism Handling	762
pkcs11_object.c	PKCS11 Library Object Handling Base	763
pkcs11_object.h	PKCS11 Library Object Handling	764
pkcs11_os.c	PKCS11 Library Operating System Abstraction Functions	765
pkcs11_os.h	PKCS11 Library Operating System Abstraction	766
pkcs11_session.c	PKCS11 Library Session Handling	767
pkcs11_session.h	PKCS11 Library Session Management & Context	767
pkcs11_signature.c	PKCS11 Library Sign/Verify Handling	769
pkcs11_signature.h	PKCS11 Library Sign/Verify Handling	770
pkcs11_slot.c	PKCS11 Library Slot Handling	770
pkcs11_slot.h	PKCS11 Library Slot Handling & Context	771
pkcs11_token.c	PKCS11 Library Token Handling	772

20.1 File List

pkcs11_token.h	
PKCS11 Library Token Management & Context	773
pkcs11_util.c	
PKCS11 Library Utility Functions	774
pkcs11_util.h	
PKCS11 Library Utilities	775
atca_wolfssl_interface.c	
Crypto abstraction functions for external host side cryptography	775
atca_wolfssl_interface.h	
Configuration Check for WolfSSL Integration Support	775
atca_wolfssl_internal.h	
WolfSSL Integration Support	776

Chapter 21

Module Documentation

21.1 TNG API (tng_)

These methods provide some convenience functions (mostly around certificates) for TNG devices, which currently include ATECC608A-MAHTN-T.

21.1.0.1 TNG Functions

This folder has a number of convenience functions for working with TNG devices (currently ATECC608A-MAHTN-T).

These devices have standard certificates that can be easily read using the functions in [tng_atcacert_client.h](#)

Functions

- const [atcacert_def_t](#) * [tng_map_get_device_cert_def](#) (int index)
Helper function to iterate through all trust cert definitions.
- ATCA_STATUS [tng_get_device_cert_def](#) (const [atcacert_def_t](#) **cert_def)
Get the TNG device certificate definition.
- ATCA_STATUS [tng_get_device_cert_def_ext](#) (ATCADevice device, const [atcacert_def_t](#) **cert_def)
Get the TNG device certificate definition.
- ATCA_STATUS [tng_get_device_pubkey](#) (uint8_t *public_key)
Uses GenKey command to calculate the public key from the primary device public key.

- const uint8_t [g_tflxtls_cert_template_4_device](#) [500]
- const [atcacert_def_t](#) [g_tflxtls_cert_def_4_device](#)
- const [atcacert_cert_element_t](#) [g_tflxtls_cert_elements_4_device](#) []

- ATCA_DLL const [atcacert_def_t](#) [g_tnqlora_cert_def_1_signer](#)

- ATCA_DLL const [atcacert_def_t](#) [g_tnqlora_cert_def_2_device](#)

- const uint8_t [g_cryptoauth_root_ca_002_cert](#) []
- const size_t [g_cryptoauth_root_ca_002_cert_size](#)

- #define **CRYPTOAUTH_ROOT_CA_002_PUBLIC_KEY_OFFSET** 266
- ATCA_DLL const [atcacert_def_t](#) **g_tnglora_cert_def_4_device**
- SHARED_LIB_EXPORT const uint8_t **g_tnglora_cert_template_4_device** []
- SHARED_LIB_EXPORT const [atcacert_cert_element_t](#) **g_tnglora_cert_elements_4_device** []
- #define **TNGLORA_CERT_TEMPLATE_4_DEVICE_SIZE** 552
- ATCA_DLL const [atcacert_def_t](#) **g_tngtls_cert_def_1_signer**
- SHARED_LIB_EXPORT const uint8_t **g_tngtls_cert_template_1_signer** []
- SHARED_LIB_EXPORT const [atcacert_cert_element_t](#) **g_tngtls_cert_elements_1_signer** []
- #define **TNGTLS_CERT_TEMPLATE_1_SIGNER_SIZE** 520
- ATCA_DLL const [atcacert_def_t](#) **g_tngtls_cert_def_2_device**
- SHARED_LIB_EXPORT const uint8_t **g_tngtls_cert_template_2_device** []
- SHARED_LIB_EXPORT const [atcacert_cert_element_t](#) **g_tngtls_cert_elements_2_device** []
- #define **TNGTLS_CERT_TEMPLATE_2_DEVICE_SIZE** 505
- #define **TNGTLS_CERT_ELEMENTS_2_DEVICE_COUNT** 2
- ATCA_DLL const [atcacert_def_t](#) **g_tngtls_cert_def_3_device**
- ATCA_DLL const uint8_t **g_tngtls_cert_template_3_device** []
- ATCA_DLL const [atcacert_cert_element_t](#) **g_tngtls_cert_elements_3_device** []
- #define **TNGTLS_CERT_TEMPLATE_3_DEVICE_SIZE** 546
- int [tng_atcacert_max_device_cert_size](#) (size_t *max_cert_size)
Return the maximum possible certificate size in bytes for a TNG device certificate. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificate.
- int [tng_atcacert_read_device_cert](#) (uint8_t *cert, size_t *cert_size, const uint8_t *signer_cert)
Reads the device certificate for a TNG device.
- int [tng_atcacert_device_public_key](#) (uint8_t *public_key, uint8_t *cert)
Reads the device public key.
- int [tng_atcacert_max_signer_cert_size](#) (size_t *max_cert_size)
Return the maximum possible certificate size in bytes for a TNG signer certificate. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificate.
- int [tng_atcacert_read_signer_cert](#) (uint8_t *cert, size_t *cert_size)
Reads the signer certificate for a TNG device.
- int [tng_atcacert_signer_public_key](#) (uint8_t *public_key, uint8_t *cert)
Reads the signer public key.
- int [tng_atcacert_root_cert_size](#) (size_t *cert_size)
Get the size of the TNG root cert.
- int [tng_atcacert_root_cert](#) (uint8_t *cert, size_t *cert_size)
Get the TNG root cert.
- int [tng_atcacert_root_public_key](#) (uint8_t *public_key)
Gets the root public key.

21.1.1 Detailed Description

These methods provide some convenience functions (mostly around certificates) for TNG devices, which currently include ATECC608A-MAHTN-T.

21.1.2 Function Documentation

21.1.2.1 tng_atcacert_device_public_key()

```
int tng_atcacert_device_public_key (
    uint8_t * public_key,
    uint8_t * cert )
```

Reads the device public key.

Parameters

out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve.
in	<i>cert</i>	If supplied, the device public key is used from this certificate. If set to NULL, the device public key is read from the device.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.1.2.2 tng_atcacert_max_device_cert_size()

```
int tng_atcacert_max_device_cert_size (
    size_t * max_cert_size )
```

Return the maximum possible certificate size in bytes for a TNG device certificate. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificate.

Parameters

out	<i>max_cert_size</i>	Maximum certificate size will be returned here in bytes.
-----	----------------------	--

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.1.2.3 tng_atcacert_max_signer_cert_size()

```
int tng_atcacert_max_signer_cert_size (
    size_t * max_cert_size )
```

Return the maximum possible certificate size in bytes for a TNG signer certificate. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificate.

Parameters

out	<i>max_cert_size</i>	Maximum certificate size will be returned here in bytes.
-----	----------------------	--

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.1.2.4 tng_atcacert_read_device_cert()

```
int tng_atcacert_read_device_cert (
    uint8_t * cert,
    size_t * cert_size,
    const uint8_t * signer_cert )
```

Reads the device certificate for a TNG device.

Parameters

out	<i>cert</i>	Buffer to received the certificate (DER format).
in, out	<i>cert_size</i>	As input, the size of the cert buffer in bytes. As output, the size of the certificate returned in cert in bytes.
in	<i>signer_cert</i>	If supplied, the signer public key is used from this certificate. If set to NULL, the signer public key is read from the device.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.1.2.5 tng_atcacert_read_signer_cert()

```
int tng_atcacert_read_signer_cert (
    uint8_t * cert,
    size_t * cert_size )
```

Reads the signer certificate for a TNG device.

Parameters

out	<i>cert</i>	Buffer to received the certificate (DER format).
in, out	<i>cert_size</i>	As input, the size of the cert buffer in bytes. As output, the size of the certificate returned in cert in bytes.

21.1 TNG API (tng_)

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.1.2.6 tng_atcacert_root_cert()

```
int tng_atcacert_root_cert (
    uint8_t * cert,
    size_t * cert_size )
```

Get the TNG root cert.

Parameters

out	<i>cert</i>	Buffer to received the certificate (DER format).
in, out	<i>cert_size</i>	As input, the size of the cert buffer in bytes. As output, the size of the certificate returned in cert in bytes.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.1.2.7 tng_atcacert_root_cert_size()

```
int tng_atcacert_root_cert_size (
    size_t * cert_size )
```

Get the size of the TNG root cert.

Parameters

out	<i>cert_size</i>	Certificate size will be returned here in bytes.
-----	------------------	--

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.1.2.8 tng_atcacert_root_public_key()

```
int tng_atcacert_root_public_key (
    uint8_t * public_key )
```

Gets the root public key.

Parameters

out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve.
-----	-------------------	--

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.1.2.9 tng_atcacert_signer_public_key()

```
int tng_atcacert_signer_public_key (
    uint8_t * public_key,
    uint8_t * cert )
```

Reads the signer public key.

Parameters

out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve.
in	<i>cert</i>	If supplied, the signer public key is used from this certificate. If set to NULL, the signer public key is read from the device.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.1.2.10 tng_get_device_cert_def()

```
ATCA_STATUS tng_get_device_cert_def (
    const atcacert_def_t ** cert_def )
```

Get the TNG device certificate definition.

Parameters

out	<i>cert_def</i>	TNG device certificate definition is returned here.
-----	-----------------	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.1 TNG API (tng_)

21.1.2.11 tng_get_device_cert_def_ext()

```
ATCA_STATUS tng_get_device_cert_def_ext (
    ATCADevice device,
    const atcacert_def_t ** cert_def )
```

Get the TNG device certificate definition.

Parameters

in	<i>device</i>	Pointer to the device context pointer
out	<i>cert_def</i>	TNG device certificate definition is returned here.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.1.2.12 tng_get_device_pubkey()

```
ATCA_STATUS tng_get_device_pubkey (
    uint8_t * public_key )
```

Uses GenKey command to calculate the public key from the primary device public key.

Parameters

out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve.
-----	-------------------	--

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.1.2.13 tng_map_get_device_cert_def()

```
const atcacert_def_t * tng_map_get_device_cert_def (
    int index )
```

Helper function to iterate through all trust cert definitions.

Parameters

in	<i>index</i>	Map index
----	--------------	-----------

Returns

non-null value if success, otherwise NULL

21.2 Basic Crypto API methods (atcab_)

These methods provide the most convenient, simple API to CryptoAuth chips.

Macros

- #define **atcab_get_addr**(...) **calib_get_addr**(__VA_ARGS__)
- #define **atca_execute_command**(...) **calib_execute_command**(__VA_ARGS__)
- #define **SHA_CONTEXT_MAX_SIZE** (109)

Functions

- ATCA_STATUS **atcab_version** (char *ver_str)
basic API methods are all prefixed with atcab_ (CryptoAuthLib Basic) the fundamental premise of the basic API is it is based on a single interface instance and that instance is global, so all basic API commands assume that one global device is the one to operate on.
- ATCA_STATUS **atcab_init_ext** (ATCADevice *device, ATCAIfaceCfg *cfg)
Creates and initializes a ATCADevice context.
- ATCA_STATUS **atcab_init** (ATCAIfaceCfg *cfg)
Creates a global ATCADevice object used by Basic API.
- ATCA_STATUS **atcab_init_device** (ATCADevice ca_device)
Initialize the global ATCADevice object to point to one of your choosing for use with all the atcab_ basic API.
- ATCA_STATUS **atcab_release_ext** (ATCADevice *device)
release (free) the an ATCADevice instance.
- ATCA_STATUS **atcab_release** (void)
release (free) the global ATCADevice instance. This must be called in order to release or free up the interface.
- ATCADevice **atcab_get_device** (void)
Get the global device object.
- ATCADeviceType **atcab_get_device_type_ext** (ATCADevice device)
Get the selected device type of rthe device context.
- ATCADeviceType **atcab_get_device_type** (void)
Get the current device type configured for the global ATCADevice.
- uint8_t **atcab_get_device_address** (ATCADevice device)
Get the current device address based on the configured device and interface.
- bool **atcab_is_ca_device** (ATCADeviceType dev_type)
Check whether the device is cryptoauth device.
- bool **atcab_is_ca2_device** (ATCADeviceType dev_type)
Check whether the device is cryptoauth device.
- bool **atcab_is_ta_device** (ATCADeviceType dev_type)
Check whether the device is Trust Anchor device.
- ATCA_STATUS **atcab_wakeup** (void)
wakeup the CryptoAuth device
- ATCA_STATUS **atcab_idle** (void)
idle the CryptoAuth device
- ATCA_STATUS **atcab_sleep** (void)

- invoke sleep on the CryptoAuth device*
- ATCA_STATUS [atcab_get_zone_size](#) (uint8_t zone, uint16_t slot, size_t *size)
Gets the size of the specified zone in bytes.
 - ATCA_STATUS [atcab_get_zone_size_ext](#) (ATCADevice device, uint8_t zone, uint16_t slot, size_t *size)
Gets the size of the specified zone in bytes.
 - ATCA_STATUS [atcab_aes](#) (uint8_t mode, uint16_t key_id, const uint8_t *aes_in, uint8_t *aes_out)
Compute the AES-128 encrypt, decrypt, or GFM calculation.
 - ATCA_STATUS [atcab_aes_encrypt](#) (uint16_t key_id, uint8_t key_block, const uint8_t *plaintext, uint8_t *ciphertext)
Perform an AES-128 encrypt operation with a key in the device.
 - ATCA_STATUS [atcab_aes_encrypt_ext](#) (ATCADevice device, uint16_t key_id, uint8_t key_block, const uint8_t *plaintext, uint8_t *ciphertext)
Perform an AES-128 encrypt operation with a key in the device.
 - ATCA_STATUS [atcab_aes_decrypt](#) (uint16_t key_id, uint8_t key_block, const uint8_t *ciphertext, uint8_t *plaintext)
Perform an AES-128 decrypt operation with a key in the device.
 - ATCA_STATUS [atcab_aes_decrypt_ext](#) (ATCADevice device, uint16_t key_id, uint8_t key_block, const uint8_t *ciphertext, uint8_t *plaintext)
Perform an AES-128 decrypt operation with a key in the device.
 - ATCA_STATUS [atcab_aes_gfm](#) (const uint8_t *h, const uint8_t *input, uint8_t *output)
Perform a Galois Field Multiply (GFM) operation.
 - ATCA_STATUS [atcab_aes_gcm_init](#) (atca_aes_gcm_ctx_t *ctx, uint16_t key_id, uint8_t key_block, const uint8_t *iv, size_t iv_size)
Initialize context for AES GCM operation with an existing IV, which is common when starting a decrypt operation.
 - ATCA_STATUS [atcab_aes_gcm_init_ext](#) (ATCADevice device, atca_aes_gcm_ctx_t *ctx, uint16_t key_id, uint8_t key_block, const uint8_t *iv, size_t iv_size)
Initialize context for AES GCM operation with an existing IV, which is common when starting a decrypt operation.
 - ATCA_STATUS [atcab_aes_gcm_init_rand](#) (atca_aes_gcm_ctx_t *ctx, uint16_t key_id, uint8_t key_block, size_t rand_size, const uint8_t *free_field, size_t free_field_size, uint8_t *iv)
Initialize context for AES GCM operation with a IV composed of a random and optional fixed(free) field, which is common when starting an encrypt operation.
 - ATCA_STATUS [atcab_aes_gcm_aad_update](#) (atca_aes_gcm_ctx_t *ctx, const uint8_t *aad, uint32_t aad_size)
Process Additional Authenticated Data (AAD) using GCM mode and a key within the ATECC608 device.
 - ATCA_STATUS [atcab_aes_gcm_aad_update_ext](#) (ATCADevice device, atca_aes_gcm_ctx_t *ctx, const uint8_t *aad, uint32_t aad_size)
Process Additional Authenticated Data (AAD) using GCM mode and a key within the ATECC608 device.
 - ATCA_STATUS [atcab_aes_gcm_encrypt_update](#) (atca_aes_gcm_ctx_t *ctx, const uint8_t *plaintext, uint32_t plaintext_size, uint8_t *ciphertext)
Encrypt data using GCM mode and a key within the ATECC608 device. [atcab_aes_gcm_init\(\)](#) or [atcab_aes_gcm_init_rand\(\)](#) should be called before the first use of this function.
 - ATCA_STATUS [atcab_aes_gcm_encrypt_update_ext](#) (ATCADevice device, atca_aes_gcm_ctx_t *ctx, const uint8_t *plaintext, uint32_t plaintext_size, uint8_t *ciphertext)
Encrypt data using GCM mode and a key within the ATECC608 device. [atcab_aes_gcm_init\(\)](#) or [atcab_aes_gcm_init_rand\(\)](#) should be called before the first use of this function.
 - ATCA_STATUS [atcab_aes_gcm_encrypt_finish](#) (atca_aes_gcm_ctx_t *ctx, uint8_t *tag, size_t tag_size)
Complete a GCM encrypt operation returning the authentication tag.
 - ATCA_STATUS [atcab_aes_gcm_encrypt_finish_ext](#) (ATCADevice device, atca_aes_gcm_ctx_t *ctx, uint8_t *tag, size_t tag_size)
Complete a GCM encrypt operation returning the authentication tag.
 - ATCA_STATUS [atcab_aes_gcm_decrypt_update](#) (atca_aes_gcm_ctx_t *ctx, const uint8_t *ciphertext, uint32_t ciphertext_size, uint8_t *plaintext)

Decrypt data using GCM mode and a key within the ATECC608 device. [atcab_aes_gcm_init\(\)](#) or [atcab_aes_gcm_init_rand\(\)](#) should be called before the first use of this function.

- ATCA_STATUS [atcab_aes_gcm_decrypt_update_ext](#) (ATCADevice device, atca_aes_gcm_ctx_t *ctx, const uint8_t *ciphertext, uint32_t ciphertext_size, uint8_t *plaintext)

Decrypt data using GCM mode and a key within the ATECC608 device. [atcab_aes_gcm_init\(\)](#) or [atcab_aes_gcm_init_rand\(\)](#) should be called before the first use of this function.

- ATCA_STATUS [atcab_aes_gcm_decrypt_finish](#) (atca_aes_gcm_ctx_t *ctx, const uint8_t *tag, size_t tag_size, bool *is_verified)

Complete a GCM decrypt operation verifying the authentication tag.

- ATCA_STATUS [atcab_aes_gcm_decrypt_finish_ext](#) (ATCADevice device, atca_aes_gcm_ctx_t *ctx, const uint8_t *tag, size_t tag_size, bool *is_verified)

Complete a GCM decrypt operation verifying the authentication tag.

- ATCA_STATUS [atcab_checkmac](#) (uint8_t mode, uint16_t key_id, const uint8_t *challenge, const uint8_t *response, const uint8_t *other_data)

Compares a MAC response with input values.

- ATCA_STATUS [atcab_checkmac_with_response_mac](#) (uint8_t mode, const uint8_t *challenge, const uint8_t *response, const uint8_t *other_data, uint8_t *mac)

Compares a MAC response with input values. SHA105 device can generate optional mac Output response mac mode only supports in SHA105 device.

- ATCA_STATUS [atcab_counter](#) (uint8_t mode, uint16_t counter_id, uint32_t *counter_value)

Compute the Counter functions.

- ATCA_STATUS [atcab_counter_increment](#) (uint16_t counter_id, uint32_t *counter_value)

Increments one of the device's monotonic counters.

- ATCA_STATUS [atcab_counter_read](#) (uint16_t counter_id, uint32_t *counter_value)

Read one of the device's monotonic counters.

- ATCA_STATUS [atcab_derivekey](#) (uint8_t mode, uint16_t key_id, const uint8_t *mac)

Executes the DeviveKey command for deriving a new key from a nonce (TempKey) and an existing key.

- ATCA_STATUS [atcab_derivekey_ext](#) (ATCADevice device, uint8_t mode, uint16_t key_id, const uint8_t *mac)

Executes the DeviveKey command for deriving a new key from a nonce (TempKey) and an existing key.

- ATCA_STATUS [atcab_ecdh_base](#) (uint8_t mode, uint16_t key_id, const uint8_t *public_key, uint8_t *pms, uint8_t *out_nonce)

Base function for generating premaster secret key using ECDH.

- ATCA_STATUS [atcab_ecdh](#) (uint16_t key_id, const uint8_t *public_key, uint8_t *pms)

ECDH command with a private key in a slot and the premaster secret is returned in the clear.

- ATCA_STATUS [atcab_ecdh_enc](#) (uint16_t key_id, const uint8_t *public_key, uint8_t *pms, const uint8_t *read_key, uint16_t read_key_id, const uint8_t num_in[(20)])

ECDH command with a private key in a slot and the premaster secret is read from the next slot.

- ATCA_STATUS [atcab_ecdh_ioenc](#) (uint16_t key_id, const uint8_t *public_key, uint8_t *pms, const uint8_t *io_key)

ECDH command with a private key in a slot and the premaster secret is returned encrypted using the IO protection key.

- ATCA_STATUS [atcab_ecdh_tempkey](#) (const uint8_t *public_key, uint8_t *pms)

ECDH command with a private key in TempKey and the premaster secret is returned in the clear.

- ATCA_STATUS [atcab_ecdh_tempkey_ioenc](#) (const uint8_t *public_key, uint8_t *pms, const uint8_t *io_key)

ECDH command with a private key in TempKey and the premaster secret is returned encrypted using the IO protection key.

- ATCA_STATUS [atcab_gendig](#) (uint8_t zone, uint16_t key_id, const uint8_t *other_data, uint8_t other_data_size)

Issues a GenDig command, which performs a SHA256 hash on the source data indicated by zone with the contents of TempKey. See the CryptoAuth datasheet for your chip to see what the values of zone correspond to.

- ATCA_STATUS [atcab_gendivkey](#) (const uint8_t *other_data)

Issues a GenDivKey command to generate the equivalent diversified key as that programmed into the client side device.

- ATCA_STATUS [atcab_genkey_base](#) (uint8_t mode, uint16_t key_id, const uint8_t *other_data, uint8_t *public_key)
Issues GenKey command, which can generate a private key, compute a public key, and/or compute a digest of a public key.
- ATCA_STATUS [atcab_genkey](#) (uint16_t key_id, uint8_t *public_key)
Issues GenKey command, which generates a new random private key in slot/handle and returns the public key.
- ATCA_STATUS [atcab_genkey_ext](#) (ATCADevice device, uint16_t key_id, uint8_t *public_key)
Issues GenKey command, which generates a new random private key in slot/handle and returns the public key.
- ATCA_STATUS [atcab_get_pubkey](#) (uint16_t key_id, uint8_t *public_key)
Uses GenKey command to calculate the public key from an existing private key in a slot.
- ATCA_STATUS [atcab_get_pubkey_ext](#) (ATCADevice device, uint16_t key_id, uint8_t *public_key)
Uses GenKey command to calculate the public key from an existing private key in a slot.
- ATCA_STATUS [atcab_hmac](#) (uint8_t mode, uint16_t key_id, uint8_t *digest)
Issues a HMAC command, which computes an HMAC/SHA-256 digest of a key stored in the device, a challenge, and other information on the device.
- ATCA_STATUS [atcab_info_base](#) (uint8_t mode, uint16_t param2, uint8_t *out_data)
Issues an Info command, which return internal device information and can control GPIO and the persistent latch.
- ATCA_STATUS [atcab_info](#) (uint8_t *revision)
Use the Info command to get the device revision (DevRev).
- ATCA_STATUS [atcab_info_ext](#) (ATCADevice device, uint8_t *revision)
Use the Info command to get the device revision (DevRev).
- ATCA_STATUS [atcab_info_lock_status](#) (uint16_t param2, uint8_t *is_locked)
Use the Info command to get the lock status.
- ATCA_STATUS [atcab_info_chip_status](#) (uint8_t *chip_status)
Use the Info command to get the chip status.
- ATCA_STATUS [atcab_info_set_latch](#) (bool state)
Use the Info command to set the persistent latch state for an ATECC608 device.
- ATCA_STATUS [atcab_info_get_latch](#) (bool *state)
Use the Info command to get the persistent latch current state for an ATECC608 device.
- ATCA_STATUS [atcab_kdf](#) (uint8_t mode, uint16_t key_id, const uint32_t details, const uint8_t *message, uint8_t *out_data, uint8_t *out_nonce)
Executes the KDF command, which derives a new key in PRF, AES, or HKDF modes.
- ATCA_STATUS [atcab_lock](#) (uint8_t mode, uint16_t summary_crc)
The Lock command prevents future modifications of the Configuration and/or Data and OTP zones. If the device is so configured, then this command can be used to lock individual data slots. This command fails if the designated area is already locked.
- ATCA_STATUS [atcab_lock_config_zone](#) (void)
Unconditionally (no CRC required) lock the config zone.
- ATCA_STATUS [atcab_lock_config_zone_ext](#) (ATCADevice device)
Unconditionally (no CRC required) lock the config zone.
- ATCA_STATUS [atcab_lock_config_zone_crc](#) (uint16_t summary_crc)
Lock the config zone with summary CRC.
- ATCA_STATUS [atcab_lock_data_zone](#) (void)
Unconditionally (no CRC required) lock the data zone (slots and OTP). for CryptoAuth devices and lock the setup for Trust Anchor device.
- ATCA_STATUS [atcab_lock_data_zone_ext](#) (ATCADevice device)
Unconditionally (no CRC required) lock the data zone (slots and OTP). for CryptoAuth devices and lock the setup for Trust Anchor device.
- ATCA_STATUS [atcab_lock_data_zone_crc](#) (uint16_t summary_crc)
Lock the data zone (slots and OTP) with summary CRC.
- ATCA_STATUS [atcab_lock_data_slot](#) (uint16_t slot)

Lock an individual slot in the data zone on an ATECC device. Not available for ATSHA devices. Slot must be configured to be slot lockable (`KeyConfig.Lockable=1`) (for cryptoauth devices) or Lock an individual handle in shared data element on an Trust Anchor device (for Trust Anchor devices).

- ATCA_STATUS `atcab_lock_data_slot_ext` (ATCADevice device, uint16_t slot)
 Lock an individual slot in the data zone on an ATECC device. Not available for ATSHA devices. Slot must be configured to be slot lockable (`KeyConfig.Lockable=1`) (for cryptoauth devices) or Lock an individual handle in shared data element on an Trust Anchor device (for Trust Anchor devices).
- ATCA_STATUS `atcab_mac` (uint8_t mode, uint16_t key_id, const uint8_t *challenge, uint8_t *digest)
 Executes MAC command, which computes a SHA-256 digest of a key stored in the device, a challenge, and other information on the device.
- ATCA_STATUS `atcab_nonce_base` (uint8_t mode, uint16_t zero, const uint8_t *num_in, uint8_t *rand_out)
 Executes Nonce command, which loads a random or fixed nonce/data into the device for use by subsequent commands.
- ATCA_STATUS `atcab_nonce` (const uint8_t *num_in)
 Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.
- ATCA_STATUS `atcab_nonce_load` (uint8_t target, const uint8_t *num_in, uint16_t num_in_size)
 Execute a Nonce command in pass-through mode to load one of the device's internal buffers with a fixed value.
- ATCA_STATUS `atcab_nonce_rand` (const uint8_t *num_in, uint8_t *rand_out)
 Execute a Nonce command to generate a random nonce combining a host nonce (`num_in`) and a device random number.
- ATCA_STATUS `atcab_nonce_rand_ext` (ATCADevice device, const uint8_t *num_in, uint8_t *rand_out)
 Execute a Nonce command to generate a random nonce combining a host nonce (`num_in`) and a device random number.
- ATCA_STATUS `atcab_challenge` (const uint8_t *num_in)
 Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.
- ATCA_STATUS `atcab_challenge_seed_update` (const uint8_t *num_in, uint8_t *rand_out)
 Execute a Nonce command to generate a random challenge combining a host nonce (`num_in`) and a device random number.
- ATCA_STATUS `atcab_priv_write` (uint16_t key_id, const uint8_t priv_key[36], uint16_t write_key_id, const uint8_t write_key[32], const uint8_t num_in[(20)])
 Executes PrivWrite command, to write externally generated ECC private keys into the device.
- ATCA_STATUS `atcab_random` (uint8_t *rand_out)
 Executes Random command, which generates a 32 byte random number from the device.
- ATCA_STATUS `atcab_random_ext` (ATCADevice device, uint8_t *rand_out)
 Executes Random command, which generates a 32 byte random number from the device.
- ATCA_STATUS `atcab_read_zone` (uint8_t zone, uint16_t slot, uint8_t block, uint8_t offset, uint8_t *data, uint8_t len)
 Executes Read command, which reads either 4 or 32 bytes of data from a given slot, configuration zone, or the OTP zone.
- ATCA_STATUS `atcab_read_zone_ext` (ATCADevice device, uint8_t zone, uint16_t slot, uint8_t block, uint8_t offset, uint8_t *data, uint8_t len)
 Executes Read command, which reads either 4 or 32 bytes of data from a given slot, configuration zone, or the OTP zone.
- ATCA_STATUS `atcab_is_locked` (uint8_t zone, bool *is_locked)
 Executes Read command, which reads the configuration zone to see if the specified zone is locked.
- ATCA_STATUS `atcab_is_config_locked` (bool *is_locked)
 This function check whether configuration zone is locked or not.
- ATCA_STATUS `atcab_is_config_locked_ext` (ATCADevice device, bool *is_locked)
 This function check whether configuration zone is locked or not.
- ATCA_STATUS `atcab_is_data_locked` (bool *is_locked)
 This function check whether data/setup zone is locked or not.
- ATCA_STATUS `atcab_is_data_locked_ext` (ATCADevice device, bool *is_locked)
 This function check whether data/setup zone is locked or not.

- ATCA_STATUS [atcab_is_slot_locked](#) (uint16_t slot, bool *is_locked)
This function check whether slot/handle is locked or not.
- ATCA_STATUS [atcab_is_slot_locked_ext](#) (ATCADevice device, uint16_t slot, bool *is_locked)
This function check whether slot/handle is locked or not.
- ATCA_STATUS [atcab_is_private_ext](#) (ATCADevice device, uint16_t slot, bool *is_private)
Check to see if the key is a private key or not.
- ATCA_STATUS [atcab_is_private](#) (uint16_t slot, bool *is_private)
- ATCA_STATUS [atcab_read_bytes_zone_ext](#) (ATCADevice device, uint8_t zone, uint16_t slot, size_t offset, uint8_t *data, size_t length)
- ATCA_STATUS [atcab_read_bytes_zone](#) (uint8_t zone, uint16_t slot, size_t offset, uint8_t *data, size_t length)
Used to read an arbitrary number of bytes from any zone configured for clear reads.
- ATCA_STATUS [atcab_read_serial_number](#) (uint8_t *serial_number)
This function returns serial number of the device.
- ATCA_STATUS [atcab_read_serial_number_ext](#) (ATCADevice device, uint8_t *serial_number)
This function returns serial number of the device.
- ATCA_STATUS [atcab_read_pubkey](#) (uint16_t slot, uint8_t *public_key)
Executes Read command to read an ECC P256 public key from a slot configured for clear reads.
- ATCA_STATUS [atcab_read_pubkey_ext](#) (ATCADevice device, uint16_t slot, uint8_t *public_key)
Executes Read command to read an ECC P256 public key from a slot configured for clear reads.
- ATCA_STATUS [atcab_read_sig](#) (uint16_t slot, uint8_t *sig)
Executes Read command to read a 64 byte ECDSA P256 signature from a slot configured for clear reads.
- ATCA_STATUS [atcab_read_config_zone](#) (uint8_t *config_data)
Executes Read command to read the complete device configuration zone.
- ATCA_STATUS [atcab_read_config_zone_ext](#) (ATCADevice device, uint8_t *config_data)
Executes Read command to read the complete device configuration zone.
- ATCA_STATUS [atcab_cmp_config_zone](#) (uint8_t *config_data, bool *same_config)
Compares a specified configuration zone with the configuration zone currently on the device.
- ATCA_STATUS [atcab_read_enc](#) (uint16_t key_id, uint8_t block, uint8_t *data, const uint8_t *enc_key, const uint16_t enc_key_id, const uint8_t num_in[(20)])
Executes Read command on a slot configured for encrypted reads and decrypts the data to return it as plaintext.
- ATCA_STATUS [atcab_secureboot](#) (uint8_t mode, uint16_t param2, const uint8_t *digest, const uint8_t *signature, uint8_t *mac)
Executes Secure Boot command, which provides support for secure boot of an external MCU or MPU.
- ATCA_STATUS [atcab_secureboot_mac](#) (uint8_t mode, const uint8_t *digest, const uint8_t *signature, const uint8_t *num_in, const uint8_t *io_key, bool *is_verified)
Executes Secure Boot command with encrypted digest and validated MAC response using the IO protection key.
- ATCA_STATUS [atcab_selftest](#) (uint8_t mode, uint16_t param2, uint8_t *result)
Executes the SelfTest command, which performs a test of one or more of the cryptographic engines within the ATECC608 chip.
- ATCA_STATUS [atcab_sha_base](#) (uint8_t mode, uint16_t length, const uint8_t *data_in, uint8_t *data_out, uint16_t *data_out_size)
Executes SHA command, which computes a SHA-256 or HMAC/SHA-256 digest for general purpose use by the host system.
- ATCA_STATUS [atcab_sha_start](#) (void)
Executes SHA command to initialize SHA-256 calculation engine.
- ATCA_STATUS [atcab_sha_update](#) (const uint8_t *message)
Executes SHA command to add 64 bytes of message data to the current context.
- ATCA_STATUS [atcab_sha_end](#) (uint8_t *digest, uint16_t length, const uint8_t *message)
Executes SHA command to complete SHA-256 or HMAC/SHA-256 operation.
- ATCA_STATUS [atcab_sha_read_context](#) (uint8_t *context, uint16_t *context_size)
Executes SHA command to read the SHA-256 context back. Only for ATECC608 with SHA-256 contexts. HMAC not supported.

- ATCA_STATUS [atcab_sha_write_context](#) (const uint8_t *context, uint16_t context_size)
Executes SHA command to write (restore) a SHA-256 context into the the device. Only supported for ATECC608 with SHA-256 contexts.
- ATCA_STATUS [atcab_sha](#) (uint16_t length, const uint8_t *message, uint8_t *digest)
Use the SHA command to compute a SHA-256 digest.
- ATCA_STATUS [atcab_hw_sha2_256](#) (const uint8_t *data, size_t data_size, uint8_t *digest)
Use the SHA command to compute a SHA-256 digest.
- ATCA_STATUS [atcab_hw_sha2_256_init](#) (atca_sha256_ctx_t *ctx)
Initialize a SHA context for performing a hardware SHA-256 operation on a device. Note that only one SHA operation can be run at a time.
- ATCA_STATUS [atcab_hw_sha2_256_update](#) (atca_sha256_ctx_t *ctx, const uint8_t *data, size_t data_size)
Add message data to a SHA context for performing a hardware SHA-256 operation on a device.
- ATCA_STATUS [atcab_hw_sha2_256_finish](#) (atca_sha256_ctx_t *ctx, uint8_t *digest)
Finish SHA-256 digest for a SHA context for performing a hardware SHA-256 operation on a device.
- ATCA_STATUS [atcab_sha_hmac_init](#) (atca_hmac_sha256_ctx_t *ctx, uint16_t key_slot)
Executes SHA command to start an HMAC/SHA-256 operation.
- ATCA_STATUS [atcab_sha_hmac_update](#) (atca_hmac_sha256_ctx_t *ctx, const uint8_t *data, size_t data_size)
Executes SHA command to add an arbitrary amount of message data to a HMAC/SHA-256 operation.
- ATCA_STATUS [atcab_sha_hmac_finish](#) (atca_hmac_sha256_ctx_t *ctx, uint8_t *digest, uint8_t target)
Executes SHA command to complete a HMAC/SHA-256 operation.
- ATCA_STATUS [atcab_sha_hmac](#) (const uint8_t *data, size_t data_size, uint16_t key_slot, uint8_t *digest, uint8_t target)
Use the SHA command to compute an HMAC/SHA-256 operation.
- ATCA_STATUS [atcab_sha_hmac_ext](#) (ATCADevice device, const uint8_t *data, size_t data_size, uint16_t key_slot, uint8_t *digest, uint8_t target)
Use the SHA command to compute an HMAC/SHA-256 operation.
- ATCA_STATUS [atcab_sign_base](#) (uint8_t mode, uint16_t key_id, uint8_t *signature)
Executes the Sign command, which generates a signature using the ECDSA algorithm.
- ATCA_STATUS [atcab_sign](#) (uint16_t key_id, const uint8_t *msg, uint8_t *signature)
Executes Sign command, to sign a 32-byte external message using the private key in the specified slot. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.
- ATCA_STATUS [atcab_sign_ext](#) (ATCADevice device, uint16_t key_id, const uint8_t *msg, uint8_t *signature)
Executes Sign command, to sign a 32-byte external message using the private key in the specified slot. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.
- ATCA_STATUS [atcab_sign_internal](#) (uint16_t key_id, bool is_invalidate, bool is_full_sn, uint8_t *signature)
Executes Sign command to sign an internally generated message.
- ATCA_STATUS [atcab_updateextra](#) (uint8_t mode, uint16_t new_value)
Executes UpdateExtra command to update the values of the two extra bytes within the Configuration zone (bytes 84 and 85).
- ATCA_STATUS [atcab_verify](#) (uint8_t mode, uint16_t key_id, const uint8_t *signature, const uint8_t *public_key, const uint8_t *other_data, uint8_t *mac)
Executes the Verify command, which takes an ECDSA [R,S] signature and verifies that it is correctly generated from a given message and public key. In all cases, the signature is an input to the command.
- ATCA_STATUS [atcab_verify_extern](#) (const uint8_t *message, const uint8_t *signature, const uint8_t *public_key, bool *is_verified)
Executes the Verify command, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.
- ATCA_STATUS [atcab_verify_extern_ext](#) (ATCADevice device, const uint8_t *message, const uint8_t *signature, const uint8_t *public_key, bool *is_verified)
Executes the Verify command, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

- ATCA_STATUS [atcab_verify_extern_mac](#) (const uint8_t *message, const uint8_t *signature, const uint8_t *public_key, const uint8_t *num_in, const uint8_t *io_key, bool *is_verified)
Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. This function is only available on the ATECC608.
- ATCA_STATUS [atcab_verify_stored](#) (const uint8_t *message, const uint8_t *signature, uint16_t key_id, bool *is_verified)
Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.
- ATCA_STATUS [atcab_verify_stored_ext](#) (ATCADevice device, const uint8_t *message, const uint8_t *signature, uint16_t key_id, bool *is_verified)
Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.
- ATCA_STATUS [atcab_verify_stored_with_tempkey](#) (const uint8_t *signature, uint16_t key_id, bool *is_verified)
Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. keyConfig.regrandom bit should be set and the message to be signed should be already loaded into TempKey for all devices.
- ATCA_STATUS [atcab_verify_stored_mac](#) (const uint8_t *message, const uint8_t *signature, uint16_t key_id, const uint8_t *num_in, const uint8_t *io_key, bool *is_verified)
Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with a public key stored in the device. This function is only available on the ATECC608.
- ATCA_STATUS [atcab_verify_validate](#) (uint16_t key_id, const uint8_t *signature, const uint8_t *other_data, bool *is_verified)
Executes the Verify command in Validate mode to validate a public key stored in a slot.
- ATCA_STATUS [atcab_verify_invalidate](#) (uint16_t key_id, const uint8_t *signature, const uint8_t *other_data, bool *is_verified)
Executes the Verify command in Invalidate mode which invalidates a previously validated public key stored in a slot.
- ATCA_STATUS [atcab_write](#) (uint8_t zone, uint16_t address, const uint8_t *value, const uint8_t *mac)
Executes the Write command, which writes either one four byte word or a 32-byte block to one of the EEPROM zones on the device. Depending upon the value of the WriteConfig byte for this slot, the data may be required to be encrypted by the system prior to being sent to the device. This command cannot be used to write slots configured as ECC private keys.
- ATCA_STATUS [atcab_write_zone](#) (uint8_t zone, uint16_t slot, uint8_t block, uint8_t offset, const uint8_t *data, uint8_t len)
Executes the Write command, which writes either 4 or 32 bytes of data into a device zone.
- ATCA_STATUS [atcab_write_zone_ext](#) (ATCADevice device, uint8_t zone, uint16_t slot, uint8_t block, uint8_t offset, const uint8_t *data, uint8_t len)
Executes the Write command, which writes either 4 or 32 bytes of data into a device zone.
- ATCA_STATUS [atcab_write_bytes_zone_ext](#) (ATCADevice device, uint8_t zone, uint16_t slot, size_t offset_bytes, const uint8_t *data, size_t length)
Executes the Write command, which writes data into the configuration, otp, or data zones with a given byte offset and length. Offset and length must be multiples of a word (4 bytes).
- ATCA_STATUS [atcab_write_bytes_zone](#) (uint8_t zone, uint16_t slot, size_t offset_bytes, const uint8_t *data, size_t length)
Executes the Write command, which writes data into the configuration, otp, or data zones with a given byte offset and length. Offset and length must be multiples of a word (4 bytes).
- ATCA_STATUS [atcab_write_pubkey](#) (uint16_t slot, const uint8_t *public_key)
Uses the write command to write a public key to a slot in the proper format.
- ATCA_STATUS [atcab_write_pubkey_ext](#) (ATCADevice device, uint16_t slot, const uint8_t *public_key)
Uses the write command to write a public key to a slot in the proper format.
- ATCA_STATUS [atcab_write_config_zone](#) (const uint8_t *config_data)
Executes the Write command, which writes the configuration zone.
- ATCA_STATUS [atcab_write_config_zone_ext](#) (ATCADevice device, const uint8_t *config_data)
Executes the Write command, which writes the configuration zone.

- ATCA_STATUS [atcab_write_enc](#) (uint16_t key_id, uint8_t block, const uint8_t *data, const uint8_t *enc_key, const uint16_t enc_key_id, const uint8_t num_in[(20)])

Executes the Write command, which performs an encrypted write of a 32 byte block into given slot.

- ATCA_STATUS [atcab_write_config_counter](#) (uint16_t counter_id, uint32_t counter_value)

Initialize one of the monotonic counters in device with a specific value.

Variables

- [ATCADevice](#) [g_atcab_device_ptr](#)

- ATCA_STATUS [atcab_bin2hex](#) (const uint8_t *bin, size_t bin_size, char *hex, size_t *hex_size)

Convert a binary buffer to a hex string for easy reading.

- ATCA_STATUS [atcab_bin2hex](#) (const uint8_t *bin, size_t bin_size, char *hex, size_t *hex_size, bool is_↵ pretty, bool is_space, bool is_upper)

Function that converts a binary buffer to a hex string suitable for easy reading.

- ATCA_STATUS [atcab_hex2bin](#) (const char *ascii_hex, size_t ascii_hex_len, uint8_t *binary, size_t *bin_len)

Function that converts a hex string to binary buffer.

- ATCA_STATUS [atcab_hex2bin](#) (const char *hex, size_t hex_size, uint8_t *bin, size_t *bin_size, bool is_↵ _space)

- ATCA_STATUS [packHex](#) (const char *ascii_hex, size_t ascii_hex_len, char *packed_hex, size_t *packed_↵ _len)

Remove spaces from a ASCII hex string.

- bool [isDigit](#) (char c)

Checks to see if a character is an ASCII representation of a digit ((c ge '0') and (c le '9'))

- bool [isBlankSpace](#) (char c)

Checks to see if a character is blank space.

- bool [isAlpha](#) (char c)

Checks to see if a character is an ASCII representation of hex ((c >= 'A') and (c <= 'F')) || ((c >= 'a') and (c <= 'f'))

- bool [isHexAlpha](#) (char c)

Checks to see if a character is an ASCII representation of hex ((c >= 'A') and (c <= 'F')) || ((c >= 'a') and (c <= 'f'))

- bool [isHex](#) (char c)

Returns true if this character is a valid hex character or if this is blankspace (The character can be included in a valid hexstring).

- bool [isHexDigit](#) (char c)

Returns true if this character is a valid hex character.

- bool [isBase64](#) (char c, const uint8_t *rules)

Returns true if this character is a valid base 64 character or if this is space (A character can be included in a valid base 64 string).

- bool [isBase64Digit](#) (char c, const uint8_t *rules)

Returns true if this character is a valid base 64 character.

- const uint8_t * [atcab_b64rules_default](#) (void)

- const uint8_t * [atcab_b64rules_mime](#) (void)

- const uint8_t * [atcab_b64rules_urllsafe](#) (void)

- ATCA_STATUS [atcab_base64decode](#) (const char *encoded, size_t encoded_size, uint8_t *data, size_↵ t *data_size, const uint8_t *rules)

Decode base64 string to data with ruleset option.

- ATCA_STATUS [atcab_base64encode](#) (const uint8_t *byte_array, size_t array_len, char *encoded, size_↵ t *encoded_len)

Encode data as base64 string.

- ATCA_STATUS [atcab_base64encode](#) (const uint8_t *data, size_t data_size, char *encoded, size_↵ t *encoded_size, const uint8_t *rules)

Encode data as base64 string with ruleset option.

- ATCA_STATUS [atcab_base64decode](#) (const char *encoded, size_t encoded_len, uint8_t *byte_array, size_t *array_len)

Decode base64 string to data.

- ATCA_STATUS [atcab_reversal](#) (const uint8_t *bin, size_t bin_size, uint8_t *dest, size_t *dest_size)

To reverse the input data.

- int [atcab_memset_s](#) (void *dest, size_t destsz, int ch, size_t count)

Guaranteed to perform memory writes regardless of optimization level. Matches memset_s signature.

- size_t [atcab_pointer_delta](#) (const void *start, const void *end)

Helper function to calculate the number of bytes between two pointers.

- char [lib_toupper](#) (char c)

Converts a character to uppercase.

- char [lib_tolower](#) (char c)

Converts a character to lowercase.

- #define [IS_ADD_SAFE_UINT16_T](#)(a, b) (((UINT16_MAX - (a)) >= (b)) ? true : false)
- #define [IS_ADD_SAFE_UINT32_T](#)(a, b) (((UINT32_MAX - (a)) >= (b)) ? true : false)
- #define [IS_ADD_SAFE_UINT64_T](#)(a, b) (((UINT64_MAX - (a)) >= (b)) ? true : false)
- #define [IS_ADD_SAFE_SIZE_T](#)(a, b) (((SIZE_MAX - (a)) >= (b)) ? true : false)
- #define [IS_MUL_SAFE_UINT16_T](#)(a, b) (((a) <= UINT16_MAX / (b))) ? true : false)
- #define [IS_MUL_SAFE_UINT32_T](#)(a, b) (((a) <= UINT32_MAX / (b))) ? true : false)
- #define [IS_MUL_SAFE_UINT64_T](#)(a, b) (((a) <= UINT64_MAX / (b))) ? true : false)
- #define [IS_MUL_SAFE_SIZE_T](#)(a, b) (((a) <= SIZE_MAX / (b))) ? true : false)
- #define [ABS_VAL](#)(x) (((x) < 0) ? -(x) : (x))

21.2.1 Detailed Description

These methods provide the most convenient, simple API to CryptoAuth chips.

21.2.2 Function Documentation

21.2.2.1 atcab_aes()

```
ATCA_STATUS atcab_aes (
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * aes_in,
    uint8_t * aes_out )
```

Compute the AES-128 encrypt, decrypt, or GFM calculation.

Parameters

in	<i>mode</i>	The mode for the AES command.
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>aes_in</i>	Input data to the AES command (16 bytes).
out	<i>aes_out</i>	Output data from the AES command is returned here (16 bytes).

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.2 atcab_aes_decrypt()

```
ATCA_STATUS atcab_aes_decrypt (
    uint16_t key_id,
    uint8_t key_block,
    const uint8_t * ciphertext,
    uint8_t * plaintext )
```

Perform an AES-128 decrypt operation with a key in the device.

Parameters

in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>ciphertext</i>	Input ciphertext to be decrypted (16 bytes).
out	<i>plaintext</i>	Output plaintext is returned here (16 bytes).

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.3 atcab_aes_decrypt_ext()

```
ATCA_STATUS atcab_aes_decrypt_ext (
    ATCADevice device,
    uint16_t key_id,
    uint8_t key_block,
    const uint8_t * ciphertext,
    uint8_t * plaintext )
```

Perform an AES-128 decrypt operation with a key in the device.

Parameters

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>ciphertext</i>	Input ciphertext to be decrypted (16 bytes).
out	<i>plaintext</i>	Output plaintext is returned here (16 bytes).

21.2 Basic Crypto API methods (atcab_)

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.4 atcab_aes_encrypt()

```
ATCA_STATUS atcab_aes_encrypt (
    uint16_t key_id,
    uint8_t key_block,
    const uint8_t * plaintext,
    uint8_t * ciphertext )
```

Perform an AES-128 encrypt operation with a key in the device.

Parameters

in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>plaintext</i>	Input plaintext to be encrypted (16 bytes).
out	<i>ciphertext</i>	Output ciphertext is returned here (16 bytes).

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.5 atcab_aes_encrypt_ext()

```
ATCA_STATUS atcab_aes_encrypt_ext (
    ATCADevice device,
    uint16_t key_id,
    uint8_t key_block,
    const uint8_t * plaintext,
    uint8_t * ciphertext )
```

Perform an AES-128 encrypt operation with a key in the device.

Parameters

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>plaintext</i>	Input plaintext to be encrypted (16 bytes).
out	<i>ciphertext</i>	Output ciphertext is returned here (16 bytes).

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.6 atcab_aes_gcm_aad_update()

```
ATCA_STATUS atcab_aes_gcm_aad_update (
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * aad,
    uint32_t aad_size )
```

Process Additional Authenticated Data (AAD) using GCM mode and a key within the ATECC608 device.

This can be called multiple times. [atcab_aes_gcm_init\(\)](#) or [atcab_aes_gcm_init_rand\(\)](#) should be called before the first use of this function. When there is AAD to include, this should be called before [atcab_aes_gcm_encrypt_update\(\)](#) or [atcab_aes_gcm_decrypt_update\(\)](#).

Parameters

in	<i>ctx</i>	AES GCM context
in	<i>aad</i>	Additional authenticated data to be added
in	<i>aad_size</i>	Size of aad in bytes

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.7 atcab_aes_gcm_aad_update_ext()

```
ATCA_STATUS atcab_aes_gcm_aad_update_ext (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * aad,
    uint32_t aad_size )
```

Process Additional Authenticated Data (AAD) using GCM mode and a key within the ATECC608 device.

This can be called multiple times. [atcab_aes_gcm_init\(\)](#) or [atcab_aes_gcm_init_rand\(\)](#) should be called before the first use of this function. When there is AAD to include, this should be called before [atcab_aes_gcm_encrypt_update\(\)](#) or [atcab_aes_gcm_decrypt_update\(\)](#).

Parameters

in	<i>device</i>	Device context
in	<i>ctx</i>	AES GCM context
in	<i>aad</i>	Additional authenticated data to be added
in	<i>aad_size</i>	Size of aad in bytes

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.8 atcab_aes_gcm_decrypt_finish()

```
ATCA_STATUS atcab_aes_gcm_decrypt_finish (
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * tag,
    size_t tag_size,
    bool * is_verified )
```

Complete a GCM decrypt operation verifying the authentication tag.

Parameters

in	<i>ctx</i>	AES GCM context structure.
in	<i>tag</i>	Expected authentication tag.
in	<i>tag_size</i>	Size of tag in bytes (12 to 16 bytes).
out	<i>is_verified</i>	Returns whether or not the tag verified.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.9 atcab_aes_gcm_decrypt_finish_ext()

```
ATCA_STATUS atcab_aes_gcm_decrypt_finish_ext (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * tag,
    size_t tag_size,
    bool * is_verified )
```

Complete a GCM decrypt operation verifying the authentication tag.

Parameters

in	<i>device</i>	Device context
in	<i>ctx</i>	AES GCM context structure.
in	<i>tag</i>	Expected authentication tag.
in	<i>tag_size</i>	Size of tag in bytes (12 to 16 bytes).
out	<i>is_verified</i>	Returns whether or not the tag verified.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.10 atcab_aes_gcm_decrypt_update()

```
ATCA_STATUS atcab_aes_gcm_decrypt_update (
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * ciphertext,
    uint32_t ciphertext_size,
    uint8_t * plaintext )
```

Decrypt data using GCM mode and a key within the ATECC608 device. [atcab_aes_gcm_init\(\)](#) or [atcab_aes_gcm_init_rand\(\)](#) should be called before the first use of this function.

Parameters

in	<i>ctx</i>	AES GCM context structure.
in	<i>ciphertext</i>	Ciphertext to be decrypted.
in	<i>ciphertext_size</i>	Size of ciphertext in bytes.
out	<i>plaintext</i>	Decrypted data is returned here.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.11 atcab_aes_gcm_decrypt_update_ext()

```
ATCA_STATUS atcab_aes_gcm_decrypt_update_ext (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * ciphertext,
    uint32_t ciphertext_size,
    uint8_t * plaintext )
```

Decrypt data using GCM mode and a key within the ATECC608 device. [atcab_aes_gcm_init\(\)](#) or [atcab_aes_gcm_init_rand\(\)](#) should be called before the first use of this function.

Parameters

in	<i>device</i>	Device context
in	<i>ctx</i>	AES GCM context structure.
in	<i>ciphertext</i>	Ciphertext to be decrypted.
in	<i>ciphertext_size</i>	Size of ciphertext in bytes.
out	<i>plaintext</i>	Decrypted data is returned here.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.12 atcab_aes_gcm_encrypt_finish()

```
ATCA_STATUS atcab_aes_gcm_encrypt_finish (
    atca_aes_gcm_ctx_t * ctx,
    uint8_t * tag,
    size_t tag_size )
```

Complete a GCM encrypt operation returning the authentication tag.

Parameters

in	<i>ctx</i>	AES GCM context structure.
out	<i>tag</i>	Authentication tag is returned here.
in	<i>tag_size</i>	Tag size in bytes (12 to 16 bytes).

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.13 atcab_aes_gcm_encrypt_finish_ext()

```
ATCA_STATUS atcab_aes_gcm_encrypt_finish_ext (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    uint8_t * tag,
    size_t tag_size )
```

Complete a GCM encrypt operation returning the authentication tag.

Parameters

in	<i>device</i>	Device context
in	<i>ctx</i>	AES GCM context structure.
out	<i>tag</i>	Authentication tag is returned here.
in	<i>tag_size</i>	Tag size in bytes (12 to 16 bytes).

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.14 atcab_aes_gcm_encrypt_update()

```
ATCA_STATUS atcab_aes_gcm_encrypt_update (
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * plaintext,
    uint32_t plaintext_size,
    uint8_t * ciphertext )
```

Encrypt data using GCM mode and a key within the ATECC608 device. [atcab_aes_gcm_init\(\)](#) or [atcab_aes_gcm_init_rand\(\)](#) should be called before the first use of this function.

Parameters

in	<i>ctx</i>	AES GCM context structure.
in	<i>plaintext</i>	Plaintext to be encrypted (16 bytes).
in	<i>plaintext_size</i>	Size of plaintext in bytes.
out	<i>ciphertext</i>	Encrypted data is returned here.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.15 atcab_aes_gcm_encrypt_update_ext()

```
ATCA_STATUS atcab_aes_gcm_encrypt_update_ext (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * plaintext,
    uint32_t plaintext_size,
    uint8_t * ciphertext )
```

Encrypt data using GCM mode and a key within the ATECC608 device. [atcab_aes_gcm_init\(\)](#) or [atcab_aes_gcm_init_rand\(\)](#) should be called before the first use of this function.

Parameters

in	<i>device</i>	Device context
in	<i>ctx</i>	AES GCM context structure.
in	<i>plaintext</i>	Plaintext to be encrypted (16 bytes).
in	<i>plaintext_size</i>	Size of plaintext in bytes.
out	<i>ciphertext</i>	Encrypted data is returned here.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.16 atcab_aes_gcm_init()

```
ATCA_STATUS atcab_aes_gcm_init (
    atca_aes_gcm_ctx_t * ctx,
    uint16_t key_id,
    uint8_t key_block,
    const uint8_t * iv,
    size_t iv_size )
```

Initialize context for AES GCM operation with an existing IV, which is common when starting a decrypt operation.

Parameters

in	<i>ctx</i>	AES GCM context to be initialized.
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>iv</i>	Initialization vector.
in	<i>iv_size</i>	Size of IV in bytes. Standard is 12 bytes.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.17 atcab_aes_gcm_init_ext()

```
ATCA_STATUS atcab_aes_gcm_init_ext (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    uint16_t key_id,
    uint8_t key_block,
    const uint8_t * iv,
    size_t iv_size )
```

Initialize context for AES GCM operation with an existing IV, which is common when starting a decrypt operation.

Parameters

in	<i>device</i>	Device context
in	<i>ctx</i>	AES GCM context to be initialized.
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>iv</i>	Initialization vector.
in	<i>iv_size</i>	Size of IV in bytes. Standard is 12 bytes.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.18 atcab_aes_gcm_init_rand()

```

ATCA_STATUS atcab_aes_gcm_init_rand (
    atca_aes_gcm_ctx_t * ctx,
    uint16_t key_id,
    uint8_t key_block,
    size_t rand_size,
    const uint8_t * free_field,
    size_t free_field_size,
    uint8_t * iv )

```

Initialize context for AES GCM operation with a IV composed of a random and optional fixed(free) field, which is common when starting an encrypt operation.

Parameters

in	<i>ctx</i>	AES CTR context to be initialized.
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>rand_size</i>	Size of the random field in bytes. Minimum and recommended size is 12 bytes. Max is 32 bytes.
in	<i>free_field</i>	Fixed data to include in the IV after the random field. Can be NULL if not used.
in	<i>free_field_size</i>	Size of the free field in bytes.
out	<i>iv</i>	Initialization vector is returned here. Its size will be <i>rand_size</i> and <i>free_field_size</i> combined.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.19 atcab_aes_gfm()

```

ATCA_STATUS atcab_aes_gfm (
    const uint8_t * h,
    const uint8_t * input,
    uint8_t * output )

```

Perform a Galois Field Multiply (GFM) operation.

Parameters

in	<i>h</i>	First input value (16 bytes).
in	<i>input</i>	Second input value (16 bytes).
out	<i>output</i>	GFM result is returned here (16 bytes).

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2 Basic Crypto API methods (atcab_)

21.2.2.20 atcab_base64decode()

```
ATCA_STATUS atcab_base64decode (
    const char * encoded,
    size_t encoded_len,
    uint8_t * byte_array,
    size_t * array_len )
```

Decode base64 string to data.

Parameters

in	<i>encoded</i>	Base64 string to be decoded.
in	<i>encoded_len</i>	Size of the base64 string in bytes.
out	<i>byte_array</i>	Decoded data will be returned here.
in, out	<i>array_len</i>	As input, the size of the byte_array buffer. As output, the length of the decoded data.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.21 atcab_base64decode_()

```
ATCA_STATUS atcab_base64decode_ (
    const char * encoded,
    size_t encoded_size,
    uint8_t * data,
    size_t * data_size,
    const uint8_t * rules )
```

Decode base64 string to data with ruleset option.

Parameters

in	<i>encoded</i>	Base64 string to be decoded.
in	<i>encoded_size</i>	Size of the base64 string in bytes.
out	<i>data</i>	Decoded data will be returned here.
in, out	<i>data_size</i>	As input, the size of the byte_array buffer. As output, the length of the decoded data.
in	<i>rules</i>	base64 ruleset to use

21.2.2.22 atcab_base64encode()

```
ATCA_STATUS atcab_base64encode (
    const uint8_t * byte_array,
    size_t array_len,
```

```
char * encoded,
size_t * encoded_len )
```

Encode data as base64 string.

Parameters

in	<i>byte_array</i>	Data to be encode in base64.
in	<i>array_len</i>	Size of byte_array in bytes.
in	<i>encoded</i>	Base64 output is returned here.
in, out	<i>encoded_len</i>	As input, the size of the encoded buffer. As output, the length of the encoded base64 character string.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.23 atcab_base64encode_()

```
ATCA_STATUS atcab_base64encode_ (
    const uint8_t * data,
    size_t data_size,
    char * encoded,
    size_t * encoded_size,
    const uint8_t * rules )
```

Encode data as base64 string with ruleset option.

Parameters

in	<i>data</i>	The input byte array that will be converted to base 64 encoded characters
in	<i>data_size</i>	The length of the byte array
in	<i>encoded</i>	The output converted to base 64 encoded characters.
in, out	<i>encoded_size</i>	Input: The size of the encoded buffer, Output: The length of the encoded base 64 character string
in	<i>rules</i>	ruleset to use during encoding

21.2.2.24 atcab_bin2hex()

```
ATCA_STATUS atcab_bin2hex (
    const uint8_t * bin,
    size_t bin_size,
    char * hex,
    size_t * hex_size )
```

Convert a binary buffer to a hex string for easy reading.

21.2 Basic Crypto API methods (atcab_)

Parameters

in	<i>bin</i>	Input data to convert.
in	<i>bin_size</i>	Size of data to convert.
out	<i>hex</i>	Buffer that receives hex string.
in, out	<i>hex_size</i>	As input, the size of the hex buffer. As output, the size of the output hex.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.25 atcab_bin2hex_()

```
ATCA_STATUS atcab_bin2hex_ (
    const uint8_t * bin,
    size_t bin_size,
    char * hex,
    size_t * hex_size,
    bool is_pretty,
    bool is_space,
    bool is_upper )
```

Function that converts a binary buffer to a hex string suitable for easy reading.

Parameters

in	<i>bin</i>	Input data to convert.
in	<i>bin_size</i>	Size of data to convert.
out	<i>hex</i>	Buffer that receives hex string.
in, out	<i>hex_size</i>	As input, the size of the hex buffer. As output, the size of the output hex.
in	<i>is_pretty</i>	Indicates whether new lines should be added for pretty printing.
in	<i>is_space</i>	Convert the output hex with space between it.
in	<i>is_upper</i>	Convert the output hex to upper case.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.26 atcab_challenge()

```
ATCA_STATUS atcab_challenge (
    const uint8_t * num_in )
```

Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.

Parameters

in	<i>num_in</i>	Data to be loaded into TempKey (32 bytes).
----	---------------	--

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.27 atcab_challenge_seed_update()

```
ATCA_STATUS atcab_challenge_seed_update (
    const uint8_t * num_in,
    uint8_t * rand_out )
```

Execute a Nonce command to generate a random challenge combining a host nonce (num_in) and a device random number.

Parameters

in	<i>num_in</i>	Host nonce to be combined with the device random number (20 bytes).
out	<i>rand_out</i>	Internally generated 32-byte random number that was used in the nonce/challenge calculation is returned here. Can be NULL if not needed.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.28 atcab_checkmac()

```
ATCA_STATUS atcab_checkmac (
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * challenge,
    const uint8_t * response,
    const uint8_t * other_data )
```

Compares a MAC response with input values.

Parameters

in	<i>mode</i>	Controls which fields within the device are used in the message
in	<i>key_id</i>	Key location in the CryptoAuth device to use for the MAC
in	<i>challenge</i>	Challenge data (32 bytes)
in	<i>response</i>	MAC response data (32 bytes)
in	<i>other_data</i>	OtherData parameter (13 bytes)

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.29 atcab_checkmac_with_response_mac()

```
ATCA_STATUS atcab_checkmac_with_response_mac (
    uint8_t mode,
    const uint8_t * challenge,
    const uint8_t * response,
    const uint8_t * other_data,
    uint8_t * mac )
```

Compares a MAC response with input values. SHA105 device can generate optional mac Output response mac mode only supports in SHA105 device.

Parameters

in	<i>mode</i>	Controls which fields within the device are used in the message
in	<i>challenge</i>	Challenge data (32 bytes)
in	<i>response</i>	MAC response data (32 bytes)
in	<i>other_data</i>	OtherData parameter (13 bytes)
out	<i>mac</i>	MAC response (32 bytes)

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.30 atcab_cmp_config_zone()

```
ATCA_STATUS atcab_cmp_config_zone (
    uint8_t * config_data,
    bool * same_config )
```

Compares a specified configuration zone with the configuration zone currently on the device.

This only compares the static portions of the configuration zone and skips those that are unique per device (first 16 bytes) and areas that can change after the configuration zone has been locked (e.g. LastKeyUse).

Parameters

in	<i>config_data</i>	Full configuration data to compare the device against.
out	<i>same_config</i>	Result is returned here. True if the static portions on the configuration zones are the same.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.31 atcab_counter()

```
ATCA_STATUS atcab_counter (
    uint8_t mode,
    uint16_t counter_id,
    uint32_t * counter_value )
```

Compute the Counter functions.

Parameters

in	<i>mode</i>	the mode used for the counter
in	<i>counter_id</i>	The counter to be used
out	<i>counter_value</i>	pointer to the counter value returned from device

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.32 atcab_counter_increment()

```
ATCA_STATUS atcab_counter_increment (
    uint16_t counter_id,
    uint32_t * counter_value )
```

Increments one of the device's monotonic counters.

Parameters

in	<i>counter_id</i>	Counter to be incremented
out	<i>counter_value</i>	New value of the counter is returned here. Can be NULL if not needed.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.33 atcab_counter_read()

```
ATCA_STATUS atcab_counter_read (
    uint16_t counter_id,
    uint32_t * counter_value )
```

21.2 Basic Crypto API methods (atcab_)

Read one of the device's monotonic counters.

Parameters

in	<i>counter_id</i>	Counter to be read
out	<i>counter_value</i>	Counter value is returned here.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.34 atcab_derivekey()

```
ATCA_STATUS atcab_derivekey (
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * mac )
```

Executes the DeviveKey command for deriving a new key from a nonce (TempKey) and an existing key.

Parameters

in	<i>mode</i>	Bit 2 must match the value in TempKey.SourceFlag
in	<i>key_id</i>	Key slot to be written
in	<i>mac</i>	Optional 32 byte MAC used to validate operation. NULL if not required.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.35 atcab_derivekey_ext()

```
ATCA_STATUS atcab_derivekey_ext (
    ATCADevice device,
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * mac )
```

Executes the DeviveKey command for deriving a new key from a nonce (TempKey) and an existing key.

Parameters

in	<i>device</i>	Device context
in	<i>mode</i>	Bit 2 must match the value in TempKey.SourceFlag
in	<i>key_id</i>	Key slot to be written
in	<i>mac</i>	Optional 32 byte MAC used to validate operation. NULL if not required.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.36 atcab_ecdh()

```
ATCA_STATUS atcab_ecdh (
    uint16_t key_id,
    const uint8_t * public_key,
    uint8_t * pms )
```

ECDH command with a private key in a slot and the premaster secret is returned in the clear.

Parameters

in	<i>key_id</i>	Slot of private key for ECDH computation
in	<i>public_key</i>	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key.
out	<i>pms</i>	Computed ECDH premaster secret is returned here. 32 bytes.

Returns

ATCA_SUCCESS on success

21.2.2.37 atcab_ecdh_base()

```
ATCA_STATUS atcab_ecdh_base (
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * public_key,
    uint8_t * pms,
    uint8_t * out_nonce )
```

Base function for generating premaster secret key using ECDH.

Parameters

in	<i>mode</i>	Mode to be used for ECDH computation
in	<i>key_id</i>	Slot of key for ECDH computation
in	<i>public_key</i>	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key.
out	<i>pms</i>	Computed ECDH pre-master secret is returned here (32 bytes) if returned directly. Otherwise NULL.
out	<i>out_nonce</i>	Nonce used to encrypt pre-master secret. NULL if output encryption not used.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.38 atcab_ecdh_enc()

```
ATCA_STATUS atcab_ecdh_enc (
    uint16_t key_id,
    const uint8_t * public_key,
    uint8_t * pms,
    const uint8_t * read_key,
    uint16_t read_key_id,
    const uint8_t num_in[ (20) ] )
```

ECDH command with a private key in a slot and the premaster secret is read from the next slot.

This function only works for even numbered slots with the proper configuration.

Parameters

in	key_id	Slot of key for ECDH computation
in	public_key	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key.
out	pms	Computed ECDH premaster secret is returned here (32 bytes).
in	read_key	Read key for the premaster secret slot (key_id 1).
in	read_key_id	Read key slot for read_key.
in	num_in	20 byte host nonce to inject into Nonce calculation

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.39 atcab_ecdh_ioenc()

```
ATCA_STATUS atcab_ecdh_ioenc (
    uint16_t key_id,
    const uint8_t * public_key,
    uint8_t * pms,
    const uint8_t * io_key )
```

ECDH command with a private key in a slot and the premaster secret is returned encrypted using the IO protection key.

Parameters

in	key_id	Slot of key for ECDH computation
in	public_key	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key.
out	pms	Computed ECDH premaster secret is returned here (32 bytes).
in	io_key	IO protection key.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.40 atcab_ecdh_tempkey()

```
ATCA_STATUS atcab_ecdh_tempkey (
    const uint8_t * public_key,
    uint8_t * pms )
```

ECDH command with a private key in TempKey and the premaster secret is returned in the clear.

Parameters

in	<i>public_key</i>	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key.
out	<i>pms</i>	Computed ECDH premaster secret is returned here (32 bytes).

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.41 atcab_ecdh_tempkey_ioenc()

```
ATCA_STATUS atcab_ecdh_tempkey_ioenc (
    const uint8_t * public_key,
    uint8_t * pms,
    const uint8_t * io_key )
```

ECDH command with a private key in TempKey and the premaster secret is returned encrypted using the IO protection key.

Parameters

in	<i>public_key</i>	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key.
out	<i>pms</i>	Computed ECDH premaster secret is returned here (32 bytes).
in	<i>io_key</i>	IO protection key.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.42 atcab_gendig()

```
ATCA_STATUS atcab_gendig (
    uint8_t zone,
    uint16_t key_id,
    const uint8_t * other_data,
    uint8_t other_data_size )
```

Issues a GenDig command, which performs a SHA256 hash on the source data indicated by zone with the contents of TempKey. See the CryptoAuth datasheet for your chip to see what the values of zone correspond to.

Parameters

in	<i>zone</i>	Designates the source of the data to hash with TempKey.
in	<i>key_id</i>	Indicates the key, OTP block, or message order for shared nonce mode.
in	<i>other_data</i>	Four bytes of data for SHA calculation when using a NoMac key, 32 bytes for "Shared Nonce" mode, otherwise ignored (can be NULL).
in	<i>other_data_size</i>	Size of other_data in bytes.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.43 atcab_gendivkey()

```
ATCA_STATUS atcab_gendivkey (
    const uint8_t * other_data )
```

Issues a GenDivKey command to generate the equivalent diversified key as that programmed into the client side device.

Parameters

in	<i>device</i>	Device context pointer
in	<i>other_data</i>	Must match data used when generating the diversified key in the client device

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.44 atcab_genkey()

```
ATCA_STATUS atcab_genkey (
    uint16_t key_id,
    uint8_t * public_key )
```

Issues GenKey command, which generates a new random private key in slot/handle and returns the public key.

Parameters

in	<i>key_id</i>	Slot number where an ECC private key is configured. Can also be ATCA_TEMPKEY_KEYID to generate a private key in TempKey.
out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve. Set to NULL if public key isn't required.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.45 atcab_genkey_base()

```
ATCA_STATUS atcab_genkey_base (
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * other_data,
    uint8_t * public_key )
```

Issues GenKey command, which can generate a private key, compute a public key, nd/or compute a digest of a public key.

Parameters

in	<i>mode</i>	Mode determines what operations the GenKey command performs.
in	<i>key_id</i>	Slot to perform the GenKey command on.
in	<i>other_data</i>	OtherData for PubKey digest calculation. Can be set to NULL otherwise.
out	<i>public_key</i>	If the mode indicates a public key will be calculated, it will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve. Set to NULL if public key isn't required.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.46 atcab_genkey_ext()

```
ATCA_STATUS atcab_genkey_ext (
    ATCADevice device,
    uint16_t key_id,
    uint8_t * public_key )
```

Issues GenKey command, which generates a new random private key in slot/handle and returns the public key.

21.2 Basic Crypto API methods (atcab_)

Parameters

in	<i>device</i>	Device context
in	<i>key_id</i>	Slot number where an ECC private key is configured. Can also be ATCA_TEMPKEY_KEYID to generate a private key in TempKey.
out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve. Set to NULL if public key isn't required.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.47 atcab_get_device()

```
ATCADevice atcab_get_device (
    void )
```

Get the global device object.

Returns

instance of global ATCADevice

21.2.2.48 atcab_get_device_address()

```
uint8_t atcab_get_device_address (
    ATCADevice device )
```

Get the current device address based on the configured device and interface.

Returns

the device address if applicable else 0xFF

21.2.2.49 atcab_get_device_type()

```
ATCADeviceType atcab_get_device_type (
    void )
```

Get the current device type configured for the global ATCADevice.

Returns

Device type if basic api is initialized or ATCA_DEV_UNKNOWN.

21.2.2.50 atcab_get_device_type_ext()

```
ATCADeviceType atcab_get_device_type_ext (
    ATCADevice device )
```

Get the selected device type of the device context.

Parameters

in	<i>device</i>	Device context pointer
----	---------------	------------------------

Returns

Device type if basic api is initialized or ATCA_DEV_UNKNOWN.

21.2.2.51 atcab_get_pubkey()

```
ATCA_STATUS atcab_get_pubkey (
    uint16_t key_id,
    uint8_t * public_key )
```

Uses GenKey command to calculate the public key from an existing private key in a slot.

Parameters

in	<i>key_id</i>	Slot number of the private key.
out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve. Set to NULL if public key isn't required.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.52 atcab_get_pubkey_ext()

```
ATCA_STATUS atcab_get_pubkey_ext (
    ATCADevice device,
    uint16_t key_id,
    uint8_t * public_key )
```

Uses GenKey command to calculate the public key from an existing private key in a slot.

Parameters

in	<i>key_id</i>	Slot number of the private key.
out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve. Set to NULL if public key isn't required.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.53 atcab_get_zone_size()

```
ATCA_STATUS atcab_get_zone_size (
    uint8_t zone,
    uint16_t slot,
    size_t * size )
```

Gets the size of the specified zone in bytes.

Parameters

in	<i>zone</i>	Zone to get size information from. Config(0), OTP(1), or Data(2) which requires a slot.
in	<i>slot</i>	If zone is Data(2), the slot to query for size.
out	<i>size</i>	Zone size is returned here.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.54 atcab_get_zone_size_ext()

```
ATCA_STATUS atcab_get_zone_size_ext (
    ATCADevice device,
    uint8_t zone,
    uint16_t slot,
    size_t * size )
```

Gets the size of the specified zone in bytes.

Parameters

in	<i>device</i>	Device context
in	<i>zone</i>	Zone to get size information from. Config(0), OTP(1), or Data(2) which requires a slot.
in	<i>slot</i>	If zone is Data(2), the slot to query for size.
out	<i>size</i>	Zone size is returned here.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.55 atcab_hex2bin()

```
ATCA_STATUS atcab_hex2bin (
    const char * ascii_hex,
```



```
size_t  ascii_hex_len,
uint8_t * binary,
size_t * bin_len )
```

Function that converts a hex string to binary buffer.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ascii_hex</i>	Input buffer to convert
in	<i>ascii_hex_len</i>	Length of buffer to convert
out	<i>binary</i>	Buffer that receives binary
in, out	<i>bin_len</i>	As input, the size of the bin buffer. As output, the size of the bin data.

21.2.2.56 atcab_hmac()

```
ATCA_STATUS atcab_hmac (
    uint8_t mode,
    uint16_t key_id,
    uint8_t * digest )
```

Issues a HMAC command, which computes an HMAC/SHA-256 digest of a key stored in the device, a challenge, and other information on the device.

Parameters

in	<i>mode</i>	Controls which fields within the device are used in the message.
in	<i>key_id</i>	Which key is to be used to generate the response. Bits 0:3 only are used to select a slot but all 16 bits are used in the HMAC message.
out	<i>digest</i>	HMAC digest is returned in this buffer (32 bytes).

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.57 atcab_hw_sha2_256()

```
ATCA_STATUS atcab_hw_sha2_256 (
    const uint8_t * data,
    size_t data_size,
    uint8_t * digest )
```

Use the SHA command to compute a SHA-256 digest.

21.2 Basic Crypto API methods (atcab_)

Parameters

in	<i>data</i>	Message data to be hashed.
in	<i>data_size</i>	Size of data in bytes.
out	<i>digest</i>	Digest is returned here (32 bytes).

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.58 atcab_hw_sha2_256_finish()

```
ATCA_STATUS atcab_hw_sha2_256_finish (
    atca_sha256_ctx_t * ctx,
    uint8_t * digest )
```

Finish SHA-256 digest for a SHA context for performing a hardware SHA-256 operation on a device.

Parameters

in	<i>ctx</i>	SHA256 context
out	<i>digest</i>	SHA256 digest is returned here (32 bytes)

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.59 atcab_hw_sha2_256_init()

```
ATCA_STATUS atcab_hw_sha2_256_init (
    atca_sha256_ctx_t * ctx )
```

Initialize a SHA context for performing a hardware SHA-256 operation on a device. Note that only one SHA operation can be run at a time.

Parameters

in	<i>ctx</i>	SHA256 context
----	------------	----------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.60 atcab_hw_sha2_256_update()

```
ATCA_STATUS atcab_hw_sha2_256_update (
    atca_sha256_ctx_t * ctx,
    const uint8_t * data,
    size_t data_size )
```

Add message data to a SHA context for performing a hardware SHA-256 operation on a device.

Parameters

in	<i>ctx</i>	SHA256 context
in	<i>data</i>	Message data to be added to hash.
in	<i>data_size</i>	Size of data in bytes.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.61 atcab_idle()

```
ATCA_STATUS atcab_idle (
    void )
```

idle the CryptoAuth device

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.62 atcab_info()

```
ATCA_STATUS atcab_info (
    uint8_t * revision )
```

Use the Info command to get the device revision (DevRev).

Parameters

out	<i>revision</i>	Device revision is returned here (4 bytes).
-----	-----------------	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.63 atcab_info_base()

```
ATCA_STATUS atcab_info_base (
    uint8_t mode,
    uint16_t param2,
    uint8_t * out_data )
```

Issues an Info command, which return internal device information and can control GPIO and the persistent latch.

Parameters

in	<i>mode</i>	Selects which mode to be used for info command.
in	<i>param2</i>	Selects the particular fields for the mode.
out	<i>out_data</i>	Response from info command (4 bytes). Can be set to NULL if not required.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.64 atcab_info_chip_status()

```
ATCA_STATUS atcab_info_chip_status (
    uint8_t * chip_status )
```

Use the Info command to get the chip status.

Parameters

out	<i>chip_status</i>	returns chip status here
-----	--------------------	--------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.65 atcab_info_ext()

```
ATCA_STATUS atcab_info_ext (
    ATCADevice device,
    uint8_t * revision )
```

Use the Info command to get the device revision (DevRev).

Parameters

in	<i>device</i>	Device context
out	<i>revision</i>	Device revision is returned here (4 bytes).

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.66 atcab_info_get_latch()

```
ATCA_STATUS atcab_info_get_latch (
    bool * state )
```

Use the Info command to get the persistent latch current state for an ATECC608 device.

Parameters

out	state	The state is returned here. Set (true) or Cier (false).
-----	-------	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.67 atcab_info_lock_status()

```
ATCA_STATUS atcab_info_lock_status (
    uint16_t param2,
    uint8_t * is_locked )
```

Use the Info command to get the lock status.

Parameters

in	param2	selects the zone and slot
out	is_locked	returns lock status here

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.68 atcab_info_set_latch()

```
ATCA_STATUS atcab_info_set_latch (
    bool state )
```

Use the Info command to set the persistent latch state for an ATECC608 device.

Parameters

out	state	Persistent latch state. Set (true) or clear (false).
-----	-------	--

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.69 atcab_init()

```
ATCA_STATUS atcab_init (
    ATCAIfaceCfg * cfg )
```

Creates a global ATCADevice object used by Basic API.

Parameters

in	cfg	Logical interface configuration. Some predefined configurations can be found in atca_cfgs.h
----	-----	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.70 atcab_init_device()

```
ATCA_STATUS atcab_init_device (
    ATCADevice ca_device )
```

Initialize the global ATCADevice object to point to one of your choosing for use with all the atcab_ basic API.

Deprecated This function is not recommended for use generally. Use of `_ext` is recommended instead. You can use `atcab_init_ext` to obtain an initialized instance and associated it with the global structure - but this shouldn't be a required process except in extremely unusual circumstances.

Parameters

in	ca_device	ATCADevice instance to use as the global Basic API crypto device instance
----	-----------	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.71 atcab_init_ext()

```
ATCA_STATUS atcab_init_ext (
    ATCADevice * device,
    ATCAIfaceCfg * cfg )
```

Creates and initializes a ATCADevice context.

Parameters

out	<i>device</i>	Pointer to the device context pointer
in	<i>cfg</i>	Logical interface configuration. Some predefined configurations can be found in atca_cfgs.h

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.72 atcab_is_ca2_device()

```
bool atcab_is_ca2_device (
    ATCADeviceType dev_type )
```

Check whether the device is cryptoauth device.

Returns

True if device is cryptoauth device or False.

21.2.2.73 atcab_is_ca_device()

```
bool atcab_is_ca_device (
    ATCADeviceType dev_type )
```

Check whether the device is cryptoauth device.

Returns

True if device is cryptoauth device or False.

21.2.2.74 atcab_is_config_locked()

```
ATCA_STATUS atcab_is_config_locked (
    bool * is_locked )
```

This function check whether configuration zone is locked or not.

Parameters

out	<i>is_locked</i>	Lock state returned here. True if locked.
-----	------------------	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.75 atcab_is_config_locked_ext()

```
ATCA_STATUS atcab_is_config_locked_ext (
    ATCADevice device,
    bool * is_locked )
```

This function check whether configuration zone is locked or not.

Parameters

in	<i>device</i>	Device context
out	<i>is_locked</i>	Lock state returned here. True if locked.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.76 atcab_is_data_locked()

```
ATCA_STATUS atcab_is_data_locked (
    bool * is_locked )
```

This function check whether data/setup zone is locked or not.

Parameters

out	<i>is_locked</i>	Lock state returned here. True if locked.
-----	------------------	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.77 atcab_is_data_locked_ext()

```
ATCA_STATUS atcab_is_data_locked_ext (
    ATCADevice device,
    bool * is_locked )
```

This function check whether data/setup zone is locked or not.

Parameters

in	<i>device</i>	Device context
out	<i>is_locked</i>	Lock state returned here. True if locked.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.78 atcab_is_locked()

```
ATCA_STATUS atcab_is_locked (
    uint8_t zone,
    bool * is_locked )
```

Executes Read command, which reads the configuration zone to see if the specified zone is locked.

Parameters

in	<i>zone</i>	The zone to query for locked (use LOCK_ZONE_CONFIG or LOCK_ZONE_DATA).
out	<i>is_locked</i>	Lock state returned here. True if locked.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.79 atcab_is_private_ext()

```
ATCA_STATUS atcab_is_private_ext (
    ATCADevice device,
    uint16_t slot,
    bool * is_private )
```

Check to see if the key is a private key or not.

This function will issue the Read command as many times as is required to read the requested data.

21.2 Basic Crypto API methods (atcab_)

Parameters

in	<i>slot</i>	Slot number to read from if zone is ATCA_ZONE_DATA(2). Ignored for all other zones.
out	<i>is_private</i>	Returned valud if successful. True if key is private.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.80 atcab_is_slot_locked()

```
ATCA_STATUS atcab_is_slot_locked (
    uint16_t slot,
    bool * is_locked )
```

This function check whether slot/handle is locked or not.

Parameters

in	<i>slot</i>	Slot to query for locked
out	<i>is_locked</i>	Lock state returned here. True if locked.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.81 atcab_is_slot_locked_ext()

```
ATCA_STATUS atcab_is_slot_locked_ext (
    ATCADevice device,
    uint16_t slot,
    bool * is_locked )
```

This function check whether slot/handle is locked or not.

Parameters

in	<i>device</i>	Device context
in	<i>slot</i>	Slot to query for locked
out	<i>is_locked</i>	Lock state returned here. True if locked.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.82 atcab_is_ta_device()

```
bool atcab_is_ta_device (
    ATCADeviceType dev_type )
```

Check whether the device is Trust Anchor device.

Returns

True if device is Trust Anchor device or False.

21.2.2.83 atcab_kdf()

```
ATCA_STATUS atcab_kdf (
    uint8_t mode,
    uint16_t key_id,
    const uint32_t details,
    const uint8_t * message,
    uint8_t * out_data,
    uint8_t * out_nonce )
```

Executes the KDF command, which derives a new key in PRF, AES, or HKDF modes.

Generally this function combines a source key with an input string and creates a result key/digest/array.

Parameters

in	<i>mode</i>	Mode determines KDF algorithm (PRF,AES,HKDF), source key location, and target key locations.
in	<i>key_id</i>	Source and target key slots if locations are in the EEPROM. Source key slot is the LSB and target key slot is the MSB.
in	<i>details</i>	Further information about the computation, depending on the algorithm (4 bytes).
in	<i>message</i>	Input value from system (up to 128 bytes). Actual size of message is 16 bytes for AES algorithm or is encoded in the MSB of the details parameter for other algorithms.
out	<i>out_data</i>	Output of the KDF function is returned here. If the result remains in the device, this can be NULL.
out	<i>out_nonce</i>	If the output is encrypted, a 32 byte random nonce generated by the device is returned here. If output encryption is not used, this can be NULL.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.84 atcab_lock()

```
ATCA_STATUS atcab_lock (
    uint8_t mode,
    uint16_t summary_crc )
```

The Lock command prevents future modifications of the Configuration and/or Data and OTP zones. If the device is so configured, then this command can be used to lock individual data slots. This command fails if the designated area is already locked.

Parameters

in	<i>mode</i>	Zone, and/or slot, and summary check (bit 7).
in	<i>summary_crc</i>	CRC of the config or data zones. Ignored for slot locks or when mode bit 7 is set.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.85 atcab_lock_config_zone()

```
ATCA_STATUS atcab_lock_config_zone (
    void )
```

Unconditionally (no CRC required) lock the config zone.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.86 atcab_lock_config_zone_crc()

```
ATCA_STATUS atcab_lock_config_zone_crc (
    uint16_t summary_crc )
```

Lock the config zone with summary CRC.

The CRC is calculated over the entire config zone contents. 48 bytes for TA100, 88 bytes for ATSHA devices, 128 bytes for ATECC devices. Lock will fail if the provided CRC doesn't match the internally calculated one.

Parameters

in	<i>summary_crc</i>	Expected CRC over the config zone.
----	--------------------	------------------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.87 atcab_lock_config_zone_ext()

```
ATCA_STATUS atcab_lock_config_zone_ext (
    ATCADevice device )
```

Unconditionally (no CRC required) lock the config zone.

Parameters

in	<i>device</i>	Device context
----	---------------	----------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.88 atcab_lock_data_slot()

```
ATCA_STATUS atcab_lock_data_slot (
    uint16_t slot )
```

Lock an individual slot in the data zone on an ATECC device. Not available for ATSHA devices. Slot must be configured to be slot lockable (KeyConfig.Lockable=1) (for cryptoauth devices) or Lock an individual handle in shared data element on an Trust Anchor device (for Trust Anchor devices).

Parameters

in	<i>slot</i>	Slot to be locked in data zone.
----	-------------	---------------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.89 atcab_lock_data_slot_ext()

```
ATCA_STATUS atcab_lock_data_slot_ext (
    ATCADevice device,
    uint16_t slot )
```

Lock an individual slot in the data zone on an ATECC device. Not available for ATSHA devices. Slot must be configured to be slot lockable (KeyConfig.Lockable=1) (for cryptoauth devices) or Lock an individual handle in shared data element on an Trust Anchor device (for Trust Anchor devices).

21.2 Basic Crypto API methods (atcab_)

Parameters

in	<i>device</i>	Device context
in	<i>slot</i>	Slot to be locked in data zone.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.90 atcab_lock_data_zone()

```
ATCA_STATUS atcab_lock_data_zone (
    void )
```

Unconditionally (no CRC required) lock the data zone (slots and OTP). for CryptoAuth devices and lock the setup for Trust Anchor device.

ConfigZone must be locked and DataZone must be unlocked for the zone to be successfully locked.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.91 atcab_lock_data_zone_crc()

```
ATCA_STATUS atcab_lock_data_zone_crc (
    uint16_t summary_crc )
```

Lock the data zone (slots and OTP) with summary CRC.

The CRC is calculated over the concatenated contents of all the slots and OTP at the end. Private keys (Key↔Config.Private=1) are skipped. Lock will fail if the provided CRC doesn't match the internally calculated one.

Parameters

in	<i>summary_crc</i>	Expected CRC over the data zone.
----	--------------------	----------------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.92 atcab_lock_data_zone_ext()

```
ATCA_STATUS atcab_lock_data_zone_ext (
    ATCADevice device )
```

Unconditionally (no CRC required) lock the data zone (slots and OTP). for CryptoAuth devices and lock the setup for Trust Anchor device.

Parameters

in	device	Device context ConfigZone must be locked and DataZone must be unlocked for the zone to be successfully locked.
----	--------	--

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.93 atcab_mac()

```
ATCA_STATUS atcab_mac (
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * challenge,
    uint8_t * digest )
```

Executes MAC command, which computes a SHA-256 digest of a key stored in the device, a challenge, and other information on the device.

Parameters

in	mode	Controls which fields within the device are used in the message
in	key_id	Key in the CryptoAuth device to use for the MAC
in	challenge	Challenge message (32 bytes). May be NULL if mode indicates a challenge isn't required.
out	digest	MAC response is returned here (32 bytes).

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.94 atcab_nonce()

```
ATCA_STATUS atcab_nonce (
    const uint8_t * num_in )
```

Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.

21.2 Basic Crypto API methods (atcab_)

Parameters

in	<i>num_in</i>	Data to be loaded into TempKey (32 bytes).
----	---------------	--

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.95 atcab_nonce_base()

```
ATCA_STATUS atcab_nonce_base (
    uint8_t mode,
    uint16_t zero,
    const uint8_t * num_in,
    uint8_t * rand_out )
```

Executes Nonce command, which loads a random or fixed nonce/data into the device for use by subsequent commands.

Parameters

in	<i>mode</i>	Controls the mechanism of the internal RNG or fixed write.
in	<i>zero</i>	Param2, normally 0, but can be used to indicate a nonce calculation mode (bit 15).
in	<i>num_in</i>	Input value to either be included in the nonce calculation in random modes (20 bytes) or to be written directly (32 bytes or 64 bytes(ATECC608)) in pass-through mode.
out	<i>rand_out</i>	If using a random mode, the internally generated 32-byte random number that was used in the nonce calculation is returned here. Can be NULL if not needed.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.96 atcab_nonce_load()

```
ATCA_STATUS atcab_nonce_load (
    uint8_t target,
    const uint8_t * num_in,
    uint16_t num_in_size )
```

Execute a Nonce command in pass-through mode to load one of the device's internal buffers with a fixed value.

For the ATECC608, available targets are TempKey (32 or 64 bytes), Message Digest Buffer (32 or 64 bytes), or the Alternate Key Buffer (32 bytes). For all other devices, only TempKey (32 bytes) is available.

Parameters

in	<i>target</i>	Target device buffer to load. Can be NONCE_MODE_TARGET_TEMPKEY, NONCE_MODE_TARGET_MSGDIGBUF, or NONCE_MODE_TARGET_ALTKEYBUF.
in	<i>num_in</i>	Data to load into the buffer.
in	<i>num_in_size</i>	Size of num_in in bytes. Can be 32 or 64 bytes depending on device and target.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.97 atcab_nonce_rand()

```
ATCA_STATUS atcab_nonce_rand (
    const uint8_t * num_in,
    uint8_t * rand_out )
```

Execute a Nonce command to generate a random nonce combining a host nonce (num_in) and a device random number.

Parameters

in	<i>num_in</i>	Host nonce to be combined with the device random number (20 bytes).
out	<i>rand_out</i>	Internally generated 32-byte random number that was used in the nonce/challenge calculation is returned here. Can be NULL if not needed.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.98 atcab_nonce_rand_ext()

```
ATCA_STATUS atcab_nonce_rand_ext (
    ATCADevice device,
    const uint8_t * num_in,
    uint8_t * rand_out )
```

Execute a Nonce command to generate a random nonce combining a host nonce (num_in) and a device random number.

Parameters

in	<i>device</i>	Device context
in	<i>num_in</i>	Host nonce to be combined with the device random number (20 bytes).
out	<i>rand_out</i>	Internally generated 32-byte random number that was used in the nonce/challenge calculation is returned here. Can be NULL if not needed.

21.2 Basic Crypto API methods (atcab_)

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.99 atcab_priv_write()

```
ATCA_STATUS atcab_priv_write (
    uint16_t key_id,
    const uint8_t priv_key[36],
    uint16_t write_key_id,
    const uint8_t write_key[32],
    const uint8_t num_in[ (20) ] )
```

Executes PrivWrite command, to write externally generated ECC private keys into the device.

Parameters

in	<i>key_id</i>	Slot to write the external private key into.
in	<i>priv_key</i>	External private key (36 bytes) to be written. The first 4 bytes should be zero for P256 curve.
in	<i>write_key_id</i>	Write key slot. Ignored if write_key is NULL.
in	<i>write_key</i>	Write key (32 bytes). If NULL, perform an unencrypted PrivWrite, which is only available when the data zone is unlocked.
in	<i>num_in</i>	20 byte host nonce to inject into Nonce calculation

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.100 atcab_random()

```
ATCA_STATUS atcab_random (
    uint8_t * rand_out )
```

Executes Random command, which generates a 32 byte random number from the device.

Parameters

out	<i>rand_out</i>	32 bytes of random data is returned here.
-----	-----------------	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.101 atcab_random_ext()

```
ATCA_STATUS atcab_random_ext (
    ATCADevice device,
    uint8_t * rand_out )
```

Executes Random command, which generates a 32 byte random number from the device.

Parameters

in	<i>device</i>	Device context pointer
out	<i>rand_out</i>	32 bytes of random data is returned here.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.102 atcab_read_bytes_zone()

```
ATCA_STATUS atcab_read_bytes_zone (
    uint8_t zone,
    uint16_t slot,
    size_t offset,
    uint8_t * data,
    size_t length )
```

Used to read an arbitrary number of bytes from any zone configured for clear reads.

This function will issue the Read command as many times as is required to read the requested data.

Parameters

in	<i>zone</i>	Zone to read data from. Option are ATCA_ZONE_CONFIG(0), ATCA_ZONE_OTP(1), or ATCA_ZONE_DATA(2).
in	<i>slot</i>	Slot number to read from if zone is ATCA_ZONE_DATA(2). Ignored for all other zones.
in	<i>offset</i>	Byte offset within the zone to read from.
out	<i>data</i>	Read data is returned here.
in	<i>length</i>	Number of bytes to read starting from the offset.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.103 atcab_read_config_zone()

```
ATCA_STATUS atcab_read_config_zone (
    uint8_t * config_data )
```

21.2 Basic Crypto API methods (atcab_)

Executes Read command to read the complete device configuration zone.

Parameters

out	<i>config_data</i>	Configuration zone data is returned here. 88 bytes for ATSHA devices, 128 bytes for ATECC devices and 48 bytes for Trust Anchor devices.
-----	--------------------	--

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.104 atcab_read_config_zone_ext()

```
ATCA_STATUS atcab_read_config_zone_ext (
    ATCADevice device,
    uint8_t * config_data )
```

Executes Read command to read the complete device configuration zone.

Parameters

in	<i>device</i>	device context
out	<i>config_data</i>	Configuration zone data is returned here. 88 bytes for ATSHA devices, 128 bytes for ATECC devices and 48 bytes for Trust Anchor devices.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.105 atcab_read_enc()

```
ATCA_STATUS atcab_read_enc (
    uint16_t key_id,
    uint8_t block,
    uint8_t * data,
    const uint8_t * enc_key,
    const uint16_t enc_key_id,
    const uint8_t num_in[ (20) ] )
```

Executes Read command on a slot configured for encrypted reads and decrypts the data to return it as plaintext.

Data zone must be locked for this command to succeed. Can only read 32 byte blocks.

Parameters

in	<i>key_id</i>	The slot ID to read from.
----	---------------	---------------------------

Parameters

in	<i>block</i>	Index of the 32 byte block within the slot to read.
out	<i>data</i>	Decrypted (plaintext) data from the read is returned here (32 bytes).
in	<i>enc_key</i>	32 byte ReadKey for the slot being read.
in	<i>enc_key_id</i>	KeyID of the ReadKey being used.
in	<i>num_in</i>	20 byte host nonce to inject into Nonce calculation

returns ATCA_SUCCESS on success, otherwise an error code.

21.2.2.106 atcab_read_pubkey()

```
ATCA_STATUS atcab_read_pubkey (
    uint16_t slot,
    uint8_t * public_key )
```

Executes Read command to read an ECC P256 public key from a slot configured for clear reads.

This function assumes the public key is stored using the ECC public key format specified in the datasheet.

Parameters

in	<i>slot</i>	Slot number to read from. Only slots 8 to 15 are large enough for a public key.
out	<i>public_key</i>	Public key is returned here (64 bytes). Format will be the 32 byte X and Y big-endian integers concatenated.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.107 atcab_read_pubkey_ext()

```
ATCA_STATUS atcab_read_pubkey_ext (
    ATCADevice device,
    uint16_t slot,
    uint8_t * public_key )
```

Executes Read command to read an ECC P256 public key from a slot configured for clear reads.

This function assumes the public key is stored using the ECC public key format specified in the datasheet.

Parameters

in	<i>device</i>	Device context pointer
in	<i>slot</i>	Slot number to read from. Only slots 8 to 15 are large enough for a public key.
out	<i>public_key</i>	Public key is returned here (64 bytes). Format will be the 32 byte X and Y big-endian integers concatenated.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.108 atcab_read_serial_number()

```
ATCA_STATUS atcab_read_serial_number (
    uint8_t * serial_number )
```

This function returns serial number of the device.

Parameters

out	<i>serial_number</i>	9 byte serial number is returned here.
-----	----------------------	--

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.109 atcab_read_serial_number_ext()

```
ATCA_STATUS atcab_read_serial_number_ext (
    ATCADevice device,
    uint8_t * serial_number )
```

This function returns serial number of the device.

Parameters

in	<i>device</i>	Device context
out	<i>serial_number</i>	9 byte serial number is returned here.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.110 atcab_read_sig()

```
ATCA_STATUS atcab_read_sig (
    uint16_t slot,
    uint8_t * sig )
```

Executes Read command to read a 64 byte ECDSA P256 signature from a slot configured for clear reads.

Parameters

in	<i>slot</i>	Slot number to read from. Only slots 8 to 15 are large enough for a signature.
out	<i>sig</i>	Signature will be returned here (64 bytes). Format will be the 32 byte R and S big-endian integers concatenated.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.111 atcab_read_zone()

```
ATCA_STATUS atcab_read_zone (
    uint8_t zone,
    uint16_t slot,
    uint8_t block,
    uint8_t offset,
    uint8_t * data,
    uint8_t len )
```

Executes Read command, which reads either 4 or 32 bytes of data from a given slot, configuration zone, or the OTP zone.

When reading a slot or OTP, data zone must be locked and the slot configuration must not be secret for a slot to be successfully read.

Parameters

in	<i>zone</i>	Zone to be read from device. Options are ATCA_ZONE_CONFIG, ATCA_ZONE_OTP, or ATCA_ZONE_DATA.
in	<i>slot</i>	Slot number for data zone and ignored for other zones.
in	<i>block</i>	32 byte block index within the zone.
in	<i>offset</i>	4 byte work index within the block. Ignored for 32 byte reads.
out	<i>data</i>	Read data is returned here.
in	<i>len</i>	Length of the data to be read. Must be either 4 or 32.

returns ATCA_SUCCESS on success, otherwise an error code.

21.2.2.112 atcab_read_zone_ext()

```
ATCA_STATUS atcab_read_zone_ext (
    ATCADevice device,
    uint8_t zone,
    uint16_t slot,
    uint8_t block,
    uint8_t offset,
    uint8_t * data,
    uint8_t len )
```

21.2 Basic Crypto API methods (atcab_)

Executes Read command, which reads either 4 or 32 bytes of data from a given slot, configuration zone, or the OTP zone.

When reading a slot or OTP, data zone must be locked and the slot configuration must not be secret for a slot to be successfully read.

Parameters

in	<i>device</i>	Device context
in	<i>zone</i>	Zone to be read from device. Options are ATCA_ZONE_CONFIG, ATCA_ZONE_OTP, or ATCA_ZONE_DATA.
in	<i>slot</i>	Slot number for data zone and ignored for other zones.
in	<i>block</i>	32 byte block index within the zone.
in	<i>offset</i>	4 byte work index within the block. Ignored for 32 byte reads.
out	<i>data</i>	Read data is returned here.
in	<i>len</i>	Length of the data to be read. Must be either 4 or 32.

returns ATCA_SUCCESS on success, otherwise an error code.

21.2.2.113 atcab_release()

```
ATCA_STATUS atcab_release (
    void )
```

release (free) the global ATCADevice instance. This must be called in order to release or free up the interface.

Returns

Returns ATCA_SUCCESS .

21.2.2.114 atcab_release_ext()

```
ATCA_STATUS atcab_release_ext (
    ATCADevice * device )
```

release (free) the an ATCADevice instance.

Parameters

in	<i>device</i>	Pointer to the device context pointer
----	---------------	---------------------------------------

Returns

Returns ATCA_SUCCESS .

21.2.2.115 atcab_reversal()

```
ATCA_STATUS atcab_reversal (
    const uint8_t * bin,
    size_t bin_size,
    uint8_t * dest,
    size_t * dest_size )
```

To reverse the input data.

Parameters

in	<i>bin</i>	Input data to reverse.
in	<i>bin_size</i>	Size of data to reverse.
out	<i>dest</i>	Buffer to store reversed binary data.
in	<i>dest_size</i>	The size of the dest buffer.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.116 atcab_secureboot()

```
ATCA_STATUS atcab_secureboot (
    uint8_t mode,
    uint16_t param2,
    const uint8_t * digest,
    const uint8_t * signature,
    uint8_t * mac )
```

Executes Secure Boot command, which provides support for secure boot of an external MCU or MPU.

Parameters

in	<i>mode</i>	Mode determines what operations the SecureBoot command performs.
in	<i>param2</i>	Not used, must be 0.
in	<i>digest</i>	Digest of the code to be verified (32 bytes).
in	<i>signature</i>	Signature of the code to be verified (64 bytes). Can be NULL when using the FullStore mode.
out	<i>mac</i>	Validating MAC will be returned here (32 bytes). Can be NULL if not required.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.117 atcab_secureboot_mac()

```
ATCA_STATUS atcab_secureboot_mac (
    uint8_t mode,
    const uint8_t * digest,
    const uint8_t * signature,
    const uint8_t * num_in,
    const uint8_t * io_key,
    bool * is_verified )
```

Executes Secure Boot command with encrypted digest and validated MAC response using the IO protection key.

Parameters

in	<i>mode</i>	Mode determines what operations the SecureBoot command performs.
in	<i>digest</i>	Digest of the code to be verified (32 bytes). This is the plaintext digest (not encrypted).
in	<i>signature</i>	Signature of the code to be verified (64 bytes). Can be NULL when using the FullStore mode.
in	<i>num_in</i>	Host nonce (20 bytes).
in	<i>io_key</i>	IO protection key (32 bytes).
out	<i>is_verified</i>	Verify result is returned here.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.118 atcab_selftest()

```
ATCA_STATUS atcab_selftest (
    uint8_t mode,
    uint16_t param2,
    uint8_t * result )
```

Executes the SelfTest command, which performs a test of one or more of the cryptographic engines within the ATECC608 chip.

Parameters

in	<i>mode</i>	Functions to test. Can be a bit field combining any of the following: SELFTEST_MODE_RNG, SELFTEST_MODE_ECDSA_VERIFY, SELFTEST_MODE_ECDSA_SIGN, SELFTEST_MODE_ECDH, SELFTEST_MODE_AES, SELFTEST_MODE_SHA, SELFTEST_MODE_ALL.
in	<i>param2</i>	Currently unused, should be 0.
out	<i>result</i>	Results are returned here as a bit field.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.119 atcab_sha()

```
ATCA_STATUS atcab_sha (
    uint16_t length,
    const uint8_t * message,
    uint8_t * digest )
```

Use the SHA command to compute a SHA-256 digest.

Parameters

in	<i>length</i>	Size of message parameter in bytes.
in	<i>message</i>	Message data to be hashed.
out	<i>digest</i>	Digest is returned here (32 bytes).

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.120 atcab_sha_base()

```
ATCA_STATUS atcab_sha_base (
    uint8_t mode,
    uint16_t length,
    const uint8_t * data_in,
    uint8_t * data_out,
    uint16_t * data_out_size )
```

Executes SHA command, which computes a SHA-256 or HMAC/SHA-256 digest for general purpose use by the host system.

Only the Start(0) and Compute(1) modes are available for ATSHA devices.

Parameters

in	<i>mode</i>	SHA command mode Start(0), Update/Compute(1), End(2), Public(3), HMACstart(4), HMACend(5), Read_Context(6), or Write_Context(7). Also message digest target location for the ATECC608.
in	<i>length</i>	Number of bytes in the message parameter or KeySlot for the HMAC key if Mode is HMACstart(4) or Public(3).
in	<i>data_in</i>	Message bytes to be hashed or Write_Context if restoring a context on the ATECC608. Can be NULL if not required by the mode.
out	<i>data_out</i>	Data returned by the command (digest or context).
in, out	<i>data_out_size</i>	As input, the size of the data_out buffer. As output, the number of bytes returned in data_out.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.121 atcab_sha_end()

```
ATCA_STATUS atcab_sha_end (
    uint8_t * digest,
    uint16_t length,
    const uint8_t * message )
```

Executes SHA command to complete SHA-256 or HMAC/SHA-256 operation.

Parameters

out	<i>digest</i>	Digest from SHA-256 or HMAC/SHA-256 will be returned here (32 bytes).
in	<i>length</i>	Length of any remaining data to include in hash. Max 64 bytes.
in	<i>message</i>	Remaining data to include in hash. NULL if length is 0.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.122 atcab_sha_hmac()

```
ATCA_STATUS atcab_sha_hmac (
    const uint8_t * data,
    size_t data_size,
    uint16_t key_slot,
    uint8_t * digest,
    uint8_t target )
```

Use the SHA command to compute an HMAC/SHA-256 operation.

Parameters

in	<i>data</i>	Message data to be hashed.
in	<i>data_size</i>	Size of data in bytes.
in	<i>key_slot</i>	Slot key id to use for the HMAC calculation
out	<i>digest</i>	Digest is returned here (32 bytes).
in	<i>target</i>	Where to save the digest internal to the device. For ATECC608, can be SHA_MODE_TARGET_TEMPKEY, SHA_MODE_TARGET_MSGDIGBUF, or SHA_MODE_TARGET_OUT_ONLY. For all other devices, SHA_MODE_TARGET_TEMPKEY is the only option.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.123 atcab_sha_hmac_ext()

```

ATCA_STATUS atcab_sha_hmac_ext (
    ATCADevice device,
    const uint8_t * data,
    size_t data_size,
    uint16_t key_slot,
    uint8_t * digest,
    uint8_t target )

```

Use the SHA command to compute an HMAC/SHA-256 operation.

Parameters

in	<i>device</i>	Device context pointer
in	<i>data</i>	Message data to be hashed.
in	<i>data_size</i>	Size of data in bytes.
in	<i>key_slot</i>	Slot key id to use for the HMAC calculation
out	<i>digest</i>	Digest is returned here (32 bytes).
in	<i>target</i>	Where to save the digest internal to the device. For ATECC608, can be SHA_MODE_TARGET_TEMPKEY, SHA_MODE_TARGET_MSGDIGBUF, or SHA_MODE_TARGET_OUT_ONLY. For all other devices, SHA_MODE_TARGET_TEMPKEY is the only option.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.124 atcab_sha_hmac_finish()

```

ATCA_STATUS atcab_sha_hmac_finish (
    atca_hmac_sha256_ctx_t * ctx,
    uint8_t * digest,
    uint8_t target )

```

Executes SHA command to complete a HMAC/SHA-256 operation.

Parameters

in	<i>ctx</i>	HMAC/SHA-256 context
out	<i>digest</i>	HMAC/SHA-256 result is returned here (32 bytes).
in	<i>target</i>	Where to save the digest internal to the device. For ATECC608, can be SHA_MODE_TARGET_TEMPKEY, SHA_MODE_TARGET_MSGDIGBUF, or SHA_MODE_TARGET_OUT_ONLY. For all other devices, SHA_MODE_TARGET_TEMPKEY is the only option.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.125 atcab_sha_hmac_init()

```
ATCA_STATUS atcab_sha_hmac_init (
    atca_hmac_sha256_ctx_t * ctx,
    uint16_t key_slot )
```

Executes SHA command to start an HMAC/SHA-256 operation.

Parameters

in	<i>ctx</i>	HMAC/SHA-256 context
in	<i>key_slot</i>	Slot key id to use for the HMAC calculation

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.126 atcab_sha_hmac_update()

```
ATCA_STATUS atcab_sha_hmac_update (
    atca_hmac_sha256_ctx_t * ctx,
    const uint8_t * data,
    size_t data_size )
```

Executes SHA command to add an arbitrary amount of message data to a HMAC/SHA-256 operation.

Parameters

in	<i>ctx</i>	HMAC/SHA-256 context
in	<i>data</i>	Message data to add
in	<i>data_size</i>	Size of message data in bytes

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.127 atcab_sha_read_context()

```
ATCA_STATUS atcab_sha_read_context (
    uint8_t * context,
    uint16_t * context_size )
```

Executes SHA command to read the SHA-256 context back. Only for ATECC608 with SHA-256 contexts. HMAC not supported.

Parameters

out	<i>context</i>	Context data is returned here.
in, out	<i>context_size</i>	As input, the size of the context buffer in bytes. As output, the size of the returned context data.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.128 atcab_sha_start()

```
ATCA_STATUS atcab_sha_start (
    void )
```

Executes SHA command to initialize SHA-256 calculation engine.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.129 atcab_sha_update()

```
ATCA_STATUS atcab_sha_update (
    const uint8_t * message )
```

Executes SHA command to add 64 bytes of message data to the current context.

Parameters

in	<i>message</i>	64 bytes of message data to add to add to operation.
----	----------------	--

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.130 atcab_sha_write_context()

```
ATCA_STATUS atcab_sha_write_context (
    const uint8_t * context,
    uint16_t context_size )
```

Executes SHA command to write (restore) a SHA-256 context into the the device. Only supported for ATECC608 with SHA-256 contexts.

21.2 Basic Crypto API methods (atcab_)

Parameters

in	<i>context</i>	Context data to be restored.
in	<i>context_size</i>	Size of the context data in bytes.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.131 atcab_sign()

```
ATCA_STATUS atcab_sign (
    uint16_t key_id,
    const uint8_t * msg,
    uint8_t * signature )
```

Executes Sign command, to sign a 32-byte external message using the private key in the specified slot. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

Parameters

in	<i>key_id</i>	Slot of the private key to be used to sign the message.
in	<i>msg</i>	32-byte message to be signed. Typically the SHA256 hash of the full message.
out	<i>signature</i>	Signature will be returned here. Format is R and S integers in big-endian format. 64 bytes for P256 curve.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.132 atcab_sign_base()

```
ATCA_STATUS atcab_sign_base (
    uint8_t mode,
    uint16_t key_id,
    uint8_t * signature )
```

Executes the Sign command, which generates a signature using the ECDSA algorithm.

Parameters

in	<i>mode</i>	Mode determines what the source of the message to be signed.
in	<i>key_id</i>	Private key slot used to sign the message.
out	<i>signature</i>	Signature is returned here. Format is R and S integers in big-endian format. 64 bytes for P256 curve.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.133 atcab_sign_ext()

```
ATCA_STATUS atcab_sign_ext (
    ATCADevice device,
    uint16_t key_id,
    const uint8_t * msg,
    uint8_t * signature )
```

Executes Sign command, to sign a 32-byte external message using the private key in the specified slot. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

Parameters

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	Slot of the private key to be used to sign the message.
in	<i>msg</i>	32-byte message to be signed. Typically the SHA256 hash of the full message.
out	<i>signature</i>	Signature will be returned here. Format is R and S integers in big-endian format. 64 bytes for P256 curve.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.134 atcab_sign_internal()

```
ATCA_STATUS atcab_sign_internal (
    uint16_t key_id,
    bool is_invalidate,
    bool is_full_sn,
    uint8_t * signature )
```

Executes Sign command to sign an internally generated message.

Parameters

in	<i>key_id</i>	Slot of the private key to be used to sign the message.
in	<i>is_invalidate</i>	Set to true if the signature will be used with the Verify(Invalid) command. false for all other cases.
in	<i>is_full_sn</i>	Set to true if the message should incorporate the device's full serial number.
out	<i>signature</i>	Signature is returned here. Format is R and S integers in big-endian format. 64 bytes for P256 curve.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.135 atcab_sleep()

```
ATCA_STATUS atcab_sleep (
    void )
```

invoke sleep on the CryptoAuth device

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.136 atcab_updateextra()

```
ATCA_STATUS atcab_updateextra (
    uint8_t mode,
    uint16_t new_value )
```

Executes UpdateExtra command to update the values of the two extra bytes within the Configuration zone (bytes 84 and 85).

Can also be used to decrement the limited use counter associated with the key in slot NewValue.

Parameters

in	<i>mode</i>	Mode determines what operations the UpdateExtra command performs.
in	<i>new_value</i>	Value to be written.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.137 atcab_verify()

```
ATCA_STATUS atcab_verify (
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * signature,
    const uint8_t * public_key,
```

```
const uint8_t * other_data,
uint8_t * mac )
```

Executes the Verify command, which takes an ECDSA [R,S] signature and verifies that it is correctly generated from a given message and public key. In all cases, the signature is an input to the command.

For the Stored, External, and ValidateExternal Modes, the contents of TempKey (or Message Digest Buffer in some cases for the ATECC608) should contain the 32 byte message.

Parameters

in	<i>mode</i>	Verify command mode and options
in	<i>key_id</i>	Stored mode, the slot containing the public key to be used for the verification. ValidateExternal mode, the slot containing the public key to be validated. External mode, KeyID contains the curve type to be used to Verify the signature. Validate or Invalidate mode, the slot containing the public key to be (in)validated.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>public_key</i>	If mode is External, the public key to be used for verification. X and Y integers in big-endian format. 64 bytes for P256 curve. NULL for all other modes.
in	<i>other_data</i>	If mode is Validate, the bytes used to generate the message for the validation (19 bytes). NULL for all other modes.
out	<i>mac</i>	If mode indicates a validating MAC, then the MAC will be returned here. Can be NULL otherwise.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.138 atcab_verify_extern()

```
ATCA_STATUS atcab_verify_extern (
    const uint8_t * message,
    const uint8_t * signature,
    const uint8_t * public_key,
    bool * is_verified )
```

Executes the Verify command, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

Parameters

in	<i>message</i>	32 byte message to be verified. Typically the SHA256 hash of the full message.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>public_key</i>	The public key to be used for verification. X and Y integers in big-endian format. 64 bytes for P256 curve.
out	<i>is_verified</i>	Boolean whether or not the message, signature, public key verified.

Returns

ATCA_SUCCESS on verification success or failure, because the command still completed successfully.

21.2.2.139 atcab_verify_extern_ext()

```
ATCA_STATUS atcab_verify_extern_ext (
    ATCADevice device,
    const uint8_t * message,
    const uint8_t * signature,
    const uint8_t * public_key,
    bool * is_verified )
```

Executes the Verify command, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

Parameters

in	<i>device</i>	Device context pointer
in	<i>message</i>	32 byte message to be verified. Typically the SHA256 hash of the full message.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>public_key</i>	The public key to be used for verification. X and Y integers in big-endian format. 64 bytes for P256 curve.
out	<i>is_verified</i>	Boolean whether or not the message, signature, public key verified.

Returns

ATCA_SUCCESS on verification success or failure, because the command still completed successfully.

21.2.2.140 atcab_verify_extern_mac()

```
ATCA_STATUS atcab_verify_extern_mac (
    const uint8_t * message,
    const uint8_t * signature,
    const uint8_t * public_key,
    const uint8_t * num_in,
    const uint8_t * io_key,
    bool * is_verified )
```

Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. This function is only available on the ATECC608.

Parameters

in	<i>message</i>	32 byte message to be verified. Typically the SHA256 hash of the full message.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.

Parameters

in	<i>public_key</i>	The public key to be used for verification. X and Y integers in big-endian format. 64 bytes for P256 curve.
in	<i>num_in</i>	System nonce (32 byte) used for the verification MAC.
in	<i>io_key</i>	IO protection key for verifying the validation MAC.
out	<i>is_verified</i>	Boolean whether or not the message, signature, public key verified.

Returns

ATCA_SUCCESS on verification success or failure, because the command still completed successfully.

21.2.2.141 atcab_verify_invalidate()

```
ATCA_STATUS atcab_verify_invalidate (
    uint16_t key_id,
    const uint8_t * signature,
    const uint8_t * other_data,
    bool * is_verified )
```

Executes the Verify command in Invalidate mode which invalidates a previously validated public key stored in a slot.

This command can only be run after GenKey has been used to create a PubKey digest of the public key to be invalidated in TempKey (mode=0x10).

Parameters

in	<i>key_id</i>	Slot containing the public key to be invalidated.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>other_data</i>	19 bytes of data used to build the verification message.
out	<i>is_verified</i>	Boolean whether or not the message, signature, validation public key verified.

Returns

ATCA_SUCCESS on verification success or failure, because the command still completed successfully.

21.2.2.142 atcab_verify_stored()

```
ATCA_STATUS atcab_verify_stored (
    const uint8_t * message,
    const uint8_t * signature,
    uint16_t key_id,
    bool * is_verified )
```

Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

21.2 Basic Crypto API methods (atcab_)

Parameters

in	<i>message</i>	32 byte message to be verified. Typically the SHA256 hash of the full message.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>key_id</i>	Slot containing the public key to be used in the verification.
out	<i>is_verified</i>	Boolean whether or not the message, signature, public key verified.

Returns

ATCA_SUCCESS on verification success or failure, because the command still completed successfully.

21.2.2.143 atcab_verify_stored_ext()

```
ATCA_STATUS atcab_verify_stored_ext (
    ATCADevice device,
    const uint8_t * message,
    const uint8_t * signature,
    uint16_t key_id,
    bool * is_verified )
```

Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

Parameters

in	<i>device</i>	Device context pointer
in	<i>message</i>	32 byte message to be verified. Typically the SHA256 hash of the full message.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>key_id</i>	Slot containing the public key to be used in the verification.
out	<i>is_verified</i>	Boolean whether or not the message, signature, public key verified.

Returns

ATCA_SUCCESS on verification success or failure, because the command still completed successfully.

21.2.2.144 atcab_verify_stored_mac()

```
ATCA_STATUS atcab_verify_stored_mac (
    const uint8_t * message,
    const uint8_t * signature,
    uint16_t key_id,
    const uint8_t * num_in,
    const uint8_t * io_key,
    bool * is_verified )
```

Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with a public key stored in the device. This function is only available on the ATECC608.

Parameters

in	<i>message</i>	32 byte message to be verified. Typically the SHA256 hash of the full message.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>key_id</i>	Slot containing the public key to be used in the verification.
in	<i>num_in</i>	System nonce (32 byte) used for the verification MAC.
in	<i>io_key</i>	IO protection key for verifying the validation MAC.
out	<i>is_verified</i>	Boolean whether or not the message, signature, public key verified.

Returns

ATCA_SUCCESS on verification success or failure, because the command still completed successfully.

21.2.2.145 atcab_verify_stored_with_tempkey()

```
ATCA_STATUS atcab_verify_stored_with_tempkey (
    const uint8_t * signature,
    uint16_t key_id,
    bool * is_verified )
```

Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. keyConfig.reqrandom bit should be set and the message to be signed should be already loaded into TempKey for all devices.

Please refer to TEST(atca_cmd_basic_test, verify_stored_on_reqrandom_set) in atca_tests_verify.c for proper use of this api

Parameters

in	<i>device</i>	Device context pointer
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>key_id</i>	Slot containing the public key to be used in the verification.
out	<i>is_verified</i>	Boolean whether or not the message, signature, public key verified.

Returns

ATCA_SUCCESS on verification success or failure, because the command still completed successfully.

21.2.2.146 atcab_verify_validate()

```
ATCA_STATUS atcab_verify_validate (
    uint16_t key_id,
    const uint8_t * signature,
    const uint8_t * other_data,
    bool * is_verified )
```

21.2 Basic Crypto API methods (atcab_)

Executes the Verify command in Validate mode to validate a public key stored in a slot.

This command can only be run after GenKey has been used to create a PubKey digest of the public key to be validated in TempKey (mode=0x10).

Parameters

in	<i>key_id</i>	Slot containing the public key to be validated.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>other_data</i>	19 bytes of data used to build the verification message.
out	<i>is_verified</i>	Boolean whether or not the message, signature, validation public key verified.

Returns

ATCA_SUCCESS on verification success or failure, because the command still completed successfully.

21.2.2.147 atcab_version()

```
ATCA_STATUS atcab_version (
    char * ver_str )
```

basic API methods are all prefixed with atcab_ (CryptoAuthLib Basic) the fundamental premise of the basic API is it is based on a single interface instance and that instance is global, so all basic API commands assume that one global device is the one to operate on.

returns a version string for the CryptoAuthLib release. The format of the version string returned is "yyyymmdd"

Parameters

out	<i>ver_str</i>	ptr to space to receive version string
-----	----------------	--

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.148 atcab_wakeup()

```
ATCA_STATUS atcab_wakeup (
    void )
```

wakeup the CryptoAuth device

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.149 atcab_write()

```
ATCA_STATUS atcab_write (
    uint8_t zone,
    uint16_t address,
    const uint8_t * value,
    const uint8_t * mac )
```

Executes the Write command, which writes either one four byte word or a 32-byte block to one of the EEPROM zones on the device. Depending upon the value of the WriteConfig byte for this slot, the data may be required to be encrypted by the system prior to being sent to the device. This command cannot be used to write slots configured as ECC private keys.

Parameters

in	<i>zone</i>	Zone/Param1 for the write command.
in	<i>address</i>	Address/Param2 for the write command.
in	<i>value</i>	Plain-text data to be written or cipher-text for encrypted writes. 32 or 4 bytes depending on bit 7 in the zone.
in	<i>mac</i>	MAC required for encrypted writes (32 bytes). Set to NULL if not required.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.150 atcab_write_bytes_zone()

```
ATCA_STATUS atcab_write_bytes_zone (
    uint8_t zone,
    uint16_t slot,
    size_t offset_bytes,
    const uint8_t * data,
    size_t length )
```

Executes the Write command, which writes data into the configuration, otp, or data zones with a given byte offset and length. Offset and length must be multiples of a word (4 bytes).

Config zone must be unlocked for writes to that zone. If data zone is unlocked, only 32-byte writes are allowed to slots and OTP and the offset and length must be multiples of 32 or the write will fail.

Parameters

in	<i>zone</i>	Zone to write data to: ATCA_ZONE_CONFIG(0), ATCA_ZONE_OTP(1), or ATCA_ZONE_DATA(2).
in	<i>slot</i>	If zone is ATCA_ZONE_DATA(2), the slot number to write to. Ignored for all other zones.
in	<i>offset_bytes</i>	Byte offset within the zone to write to. Must be a multiple of a word (4 bytes).
in	<i>data</i>	Data to be written.
in	<i>length</i>	Number of bytes to be written. Must be a multiple of a word (4 bytes).

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.151 atcab_write_config_counter()

```
ATCA_STATUS atcab_write_config_counter (
    uint16_t counter_id,
    uint32_t counter_value )
```

Initialize one of the monotonic counters in device with a specific value.

The monotonic counters are stored in the configuration zone using a special format. This encodes a binary count value into the 8 byte encoded value required. Can only be set while the configuration zone is unlocked.

Parameters

in	<i>counter_id</i>	Counter to be written.
in	<i>counter_value</i>	Counter value to set.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.152 atcab_write_config_zone()

```
ATCA_STATUS atcab_write_config_zone (
    const uint8_t * config_data )
```

Executes the Write command, which writes the configuration zone.

First 16 bytes are skipped as they are not writable. LockValue and LockConfig are also skipped and can only be changed via the Lock command.

This command may fail if UserExtra and/or Selector bytes have already been set to non-zero values.

Parameters

in	<i>config_data</i>	Data to the config zone data. This should be 88 bytes for SHA devices and 128 bytes for ECC devices.
----	--------------------	--

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.153 atcab_write_config_zone_ext()

```
ATCA_STATUS atcab_write_config_zone_ext (
    ATCADevice device,
    const uint8_t * config_data )
```

Executes the Write command, which writes the configuration zone.

First 16 bytes are skipped as they are not writable. LockValue and LockConfig are also skipped and can only be changed via the Lock command.

This command may fail if UserExtra and/or Selector bytes have already been set to non-zero values.

Parameters

in	<i>device</i>	Device context
in	<i>config_data</i>	Data to the config zone data. This should be 88 bytes for SHA devices and 128 bytes for ECC devices.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.154 atcab_write_enc()

```
ATCA_STATUS atcab_write_enc (
    uint16_t key_id,
    uint8_t block,
    const uint8_t * data,
    const uint8_t * enc_key,
    const uint16_t enc_key_id,
    const uint8_t num_in[ (20) ] )
```

Executes the Write command, which performs an encrypted write of a 32 byte block into given slot.

The function takes clear text bytes and encrypts them for writing over the wire. Data zone must be locked and the slot configuration must be set to encrypted write for the block to be successfully written.

Parameters

in	<i>key_id</i>	Slot ID to write to.
in	<i>block</i>	Index of the 32 byte block to write in the slot.
in	<i>data</i>	32 bytes of clear text data to be written to the slot
in	<i>enc_key</i>	WriteKey to encrypt with for writing
in	<i>enc_key_id</i>	The KeyID of the WriteKey
in	<i>num_in</i>	20 byte host nonce to inject into Nonce calculation

returns ATCA_SUCCESS on success, otherwise an error code.

21.2 Basic Crypto API methods (atcab_)

21.2.2.155 atcab_write_pubkey()

```
ATCA_STATUS atcab_write_pubkey (
    uint16_t slot,
    const uint8_t * public_key )
```

Uses the write command to write a public key to a slot in the proper format.

Parameters

in	<i>slot</i>	Slot number to write. Only slots 8 to 15 are large enough to store a public key.
in	<i>public_key</i>	Public key to write into the slot specified. X and Y integers in big-endian format. 64 bytes for P256 curve.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.156 atcab_write_pubkey_ext()

```
ATCA_STATUS atcab_write_pubkey_ext (
    ATCADevice device,
    uint16_t slot,
    const uint8_t * public_key )
```

Uses the write command to write a public key to a slot in the proper format.

Parameters

in	<i>device</i>	Device context
in	<i>slot</i>	Slot number to write. Only slots 8 to 15 are large enough to store a public key.
in	<i>public_key</i>	Public key to write into the slot specified. X and Y integers in big-endian format. 64 bytes for P256 curve.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.157 atcab_write_zone()

```
ATCA_STATUS atcab_write_zone (
    uint8_t zone,
    uint16_t slot,
    uint8_t block,
    uint8_t offset,
```

```
const uint8_t * data,  
uint8_t len )
```

Executes the Write command, which writes either 4 or 32 bytes of data into a device zone.

21.2 Basic Crypto API methods (atcab_)

Parameters

in	<i>zone</i>	Device zone to write to (0=config, 1=OTP, 2=data).
in	<i>slot</i>	If writing to the data zone, it is the slot to write to, otherwise it should be 0.
in	<i>block</i>	32-byte block to write to.
in	<i>offset</i>	4-byte word within the specified block to write to. If performing a 32-byte write, this should be 0.
in	<i>data</i>	Data to be written.
in	<i>len</i>	Number of bytes to be written. Must be either 4 or 32.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.158 atcab_write_zone_ext()

```
ATCA_STATUS atcab_write_zone_ext (
    ATCADevice device,
    uint8_t zone,
    uint16_t slot,
    uint8_t block,
    uint8_t offset,
    const uint8_t * data,
    uint8_t len )
```

Executes the Write command, which writes either 4 or 32 bytes of data into a device zone.

Parameters

in	<i>device</i>	Device context
in	<i>zone</i>	Device zone to write to (0=config, 1=OTP, 2=data).
in	<i>slot</i>	If writing to the data zone, it is the slot to write to, otherwise it should be 0.
in	<i>block</i>	32-byte block to write to.
in	<i>offset</i>	4-byte word within the specified block to write to. If performing a 32-byte write, this should be 0.
in	<i>data</i>	Data to be written.
in	<i>len</i>	Number of bytes to be written. Must be either 4 or 32.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.2.2.159 isAlpha()

```
bool isAlpha (
    char c )
```

Checks to see if a character is an ASCII representation of hex ((c >= 'A') and (c <= 'F')) || ((c >= 'a') and (c <= 'f'))

Parameters

in	<i>c</i>	character to check
----	----------	--------------------

Returns

True if the character is a hex

21.2.2.160 isBase64()

```
bool isBase64 (
    char c,
    const uint8_t * rules )
```

Returns true if this character is a valid base 64 character or if this is space (A character can be included in a valid base 64 string).

Parameters

in	<i>c</i>	character to check
in	<i>rules</i>	base64 ruleset to use

Returns

True if the character can be included in a valid base 64 string

21.2.2.161 isBase64Digit()

```
bool isBase64Digit (
    char c,
    const uint8_t * rules )
```

Returns true if this character is a valid base 64 character.

Parameters

in	<i>c</i>	character to check
in	<i>rules</i>	base64 ruleset to use

Returns

True if the character can be included in a valid base 64 string

21.2.2.162 isBlankSpace()

```
bool isBlankSpace (  
    char c )
```

Checks to see if a character is blank space.

Parameters

in	c	character to check
----	---	--------------------

Returns

True if the character is blankspace

21.2.2.163 isDigit()

```
bool isDigit (  
    char c )
```

Checks to see if a character is an ASCII representation of a digit ((c ge '0') and (c le '9'))

Parameters

in	c	character to check
----	---	--------------------

Returns

True if the character is a digit

21.2.2.164 isHex()

```
bool isHex (  
    char c )
```

Returns true if this character is a valid hex character or if this is blankspace (The character can be included in a valid hexstring).

Parameters

in	c	character to check
----	---	--------------------

Returns

True if the character can be included in a valid hexstring

21.2.2.165 isHexAlpha()

```
bool isHexAlpha (
    char c )
```

Checks to see if a character is an ASCII representation of hex ((c >= 'A') and (c <= 'F')) || ((c >= 'a') and (c <= 'f'))

Parameters

in	c	character to check
----	---	--------------------

Returns

True if the character is a hex

21.2.2.166 isHexDigit()

```
bool isHexDigit (
    char c )
```

Returns true if this character is a valid hex character.

Parameters

in	c	character to check
----	---	--------------------

Returns

True if the character can be included in a valid hexstring

21.2.2.167 packHex()

```
ATCA_STATUS packHex (
    const char * ascii_hex,
    size_t ascii_hex_len,
    char * packed_hex,
    size_t * packed_len )
```

Remove spaces from a ASCII hex string.

21.3 Configuration (cfg_)

Parameters

in	<i>ascii_hex</i>	Initial hex string to remove blankspace from
in	<i>ascii_hex_len</i>	Length of the initial hex string
in	<i>packed_hex</i>	Resulting hex string without blankspace
in, out	<i>packed_len</i>	In: Size to packed_hex buffer Out: Number of bytes in the packed hex string

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.3 Configuration (cfg_)

Logical device configurations describe the CryptoAuth device type and logical interface.

Logical device configurations describe the CryptoAuth device type and logical interface.

21.4 ATCADevice (atca_)

ATCADevice object - composite of command and interface objects.

Data Structures

- struct [atca_device](#)
atca_device is the C object backing ATCADevice. See the [atca_device.h](#) file for details on the ATCADevice methods

Macros

- #define **ATSHA204A** (0U)
The supported Device type in Cryptoauthlib library.
- #define **ATECC108A** (1U)
- #define **ATECC508A** (2U)
- #define **ATECC608A** (3U)
- #define **ATECC608B** (3U)
- #define **ATECC608** (3U)
- #define **ATSHA206A** (4U)
- #define **TA100** (0x10U)
- #define **TA101** (0x11U)
- #define **ECC204** (0x20U)
- #define **TA010** (0x21U)
- #define **ECC206** (0x22U)
- #define **RNG90** (0x23U)
- #define **SHA104** (0x24U)
- #define **SHA105** (0x25U)
- #define **SHA106** (0x26U)
- #define **ATCA_DEV_UNKNOWN** (0x7EU)
- #define **ATCA_DEV_INVALID** (0x7FU)

Typedefs

- typedef void(* **ctx_cb**) (void *ctx)
Callback function to clean up the session context.
- typedef struct **atca_device** * **ATCADevice**
- typedef uint8_t **ATCADeviceType**

Enumerations

- enum **ATCADeviceState** { **ATCA_DEVICE_STATE_UNKNOWN** = 0 , **ATCA_DEVICE_STATE_SLEEP** , **ATCA_DEVICE_STATE_IDLE** , **ATCA_DEVICE_STATE_ACTIVE** }
ATCADeviceState says about device state.

Functions

- **ATCADevice** **newATCADevice** (**ATCAIfaceCfg** *cfg)
constructor for a Microchip CryptoAuth device
- void **deleteATCADevice** (**ATCADevice** *ca_dev)
destructor for a device NULLs reference after object is freed
- **ATCA_STATUS** **initATCADevice** (**ATCAIfaceCfg** *cfg, **ATCADevice** ca_dev)
Initializer for an Microchip CryptoAuth device.
- **ATCAIface** **atGetIFace** (**ATCADevice** dev)
returns a reference to the ATCAIface interface object for the device
- **ATCA_STATUS** **releaseATCADevice** (**ATCADevice** ca_dev)
Release any resources associated with the device.

21.4.1 Detailed Description

ATCADevice object - composite of command and interface objects.

21.4.2 Function Documentation

21.4.2.1 atGetIFace()

```
ATCAIface atGetIFace (
    ATCADevice dev )
```

returns a reference to the ATCAIface interface object for the device

Parameters

in	dev	reference to a device
----	-----	-----------------------

21.4 ATCADevice (atca_)

Returns

reference to the ATCAIface object for the device

21.4.2.2 deleteATCADevice()

```
void deleteATCADevice (
    ATCADevice * ca_dev )
```

destructor for a device NULLs reference after object is freed

Parameters

in	ca_dev	pointer to a reference to a device
----	--------	------------------------------------

21.4.2.3 initATCADevice()

```
ATCA_STATUS initATCADevice (
    ATCAIfaceCfg * cfg,
    ATCADevice ca_dev )
```

Initializer for an Microchip CryptoAuth device.

Parameters

in	cfg	pointer to an interface configuration object
in, out	ca_dev	As input, pre-allocated structure to be initialized. mCommands and mIface members should point to existing structures to be initialized.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.4.2.4 newATCADevice()

```
ATCADevice newATCADevice (
    ATCAIfaceCfg * cfg )
```

constructor for a Microchip CryptoAuth device

Parameters

in	cfg	Interface configuration object
----	-----	--------------------------------

Returns

Reference to a new ATCADevice on success. NULL on failure.

21.4.2.5 releaseATCADevice()

```
ATCA_STATUS releaseATCADevice (
    ATCADevice ca_dev )
```

Release any resources associated with the device.

Parameters

in	ca_dev	Device to release
----	--------	-------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.5 ATCAIface (atca_)

Abstract interface to all CryptoAuth device types. This interface connects to the HAL implementation and abstracts the physical details of the device communication from all the upper layers of CryptoAuthLib.

Data Structures

- struct [devtype_names_t](#)
- struct [ATCAIfaceCfg](#)
- struct [ATCAHAL_t](#)
HAL Driver Structure.
- struct [atca_iface](#)
atca_iface is the context structure for a configured interface

Macros

- #define **ATCA_IFACECFG_NAME**(x) (x)
- #define **ATCA_IFACECFG_I2C_ADDRESS**(c) (c)->cfg.atcai2c.address
- #define **ATCA_IFACECFG_I2C_BAUD**(c) (c)->cfg.atcai2c.baud
- #define **ATCA_IFACECFG_VALUE**(c, v) (c)->cfg.v

Typedefs

- typedef struct [atca_iface](#) * **ATCAIface**
- typedef struct [atca_iface](#) **atca_iface_t**
atca_iface is the context structure for a configured interface

Enumerations

- enum [ATCAIfaceType](#) {
[ATCA_I2C_IFACE](#) = 0 , [ATCA_SWI_IFACE](#) = 1 , [ATCA_UART_IFACE](#) = 2 , [ATCA_SPI_IFACE](#) = 3 ,
[ATCA_HID_IFACE](#) = 4 , [ATCA_KIT_IFACE](#) = 5 , [ATCA_CUSTOM_IFACE](#) = 6 , [ATCA_I2C_GPIO_IFACE](#) = 7 ,
[ATCA_SWI_GPIO_IFACE](#) = 8 , [ATCA_SPI_GPIO_IFACE](#) = 9 , [ATCA_UNKNOWN_IFACE](#) = 0xFE }
- enum [ATCAKitType](#) {
[ATCA_KIT_AUTO_IFACE](#) , [ATCA_KIT_I2C_IFACE](#) , [ATCA_KIT_SWI_IFACE](#) , [ATCA_KIT_SPI_IFACE](#) ,
[ATCA_KIT_UNKNOWN_IFACE](#) }

Functions

- ATCA_STATUS [initATCAIface](#) ([ATCAIfaceCfg](#) *cfg, [ATCAIface](#) ca_iface)
Initializer for ATCAIface objects.
- ATCA_STATUS [atinit](#) ([ATCAIface](#) ca_iface)
Performs the HAL initialization by calling intermediate HAL wrapper function. If using the basic API, the [atcab_init\(\)](#) function should be called instead.
- ATCA_STATUS [atsend](#) ([ATCAIface](#) ca_iface, uint8_t word_address, uint8_t *txdata, int txlength)
Sends the data to the device by calling intermediate HAL wrapper function.
- ATCA_STATUS [atreceive](#) ([ATCAIface](#) ca_iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxlength)
Receives data from the device by calling intermediate HAL wrapper function.
- ATCA_STATUS [atcontrol](#) ([ATCAIface](#) ca_iface, uint8_t option, void *param, size_t paramlen)
Perform control operations with the underlying hal driver.
- ATCA_STATUS [atwake](#) ([ATCAIface](#) ca_iface)
Wakes up the device by calling intermediate HAL wrapper function. The [atcab_wakeup\(\)](#) function should be used instead.
- ATCA_STATUS [atidle](#) ([ATCAIface](#) ca_iface)
Puts the device into idle state by calling intermediate HAL wrapper function. The [atcab_idle\(\)](#) function should be used instead.
- ATCA_STATUS [atsleep](#) ([ATCAIface](#) ca_iface)
Puts the device into sleep state by calling intermediate HAL wrapper function. The [atcab_sleep\(\)](#) function should be used instead.
- [ATCAIfaceCfg](#) * [atgetifacecfg](#) ([ATCAIface](#) ca_iface)
Returns the logical interface configuration for the device.
- void * [atgetifacehaldat](#) ([ATCAIface](#) ca_iface)
Returns the HAL data pointer for the device.
- bool [iface_type_is_kit](#) ([ATCAIfaceType](#) iface_type)
Check if the given interface is a "kit protocol" one.
- bool [atca_iface_is_kit](#) ([ATCAIface](#) ca_iface)
Check if the given interface is configured as a "kit protocol" one where transactions are atomic.
- bool [atca_iface_is_swi](#) ([ATCAIface](#) ca_iface)
Check if the given interface is configured as a SWI.
- int [atca_iface_get_retries](#) ([ATCAIface](#) ca_iface)
Retrive the number of retries for a configured interface.
- uint16_t [atca_iface_get_wake_delay](#) ([ATCAIface](#) ca_iface)
Retrive the wake/retry delay for a configured interface/device.
- uint8_t [ifacecfg_get_address](#) ([ATCAIfaceCfg](#) *cfg)
Retrieves the device address given an interface configuration.
- ATCA_STATUS [ifacecfg_set_address](#) ([ATCAIfaceCfg](#) *cfg, uint8_t address, [ATCAKitType](#) kitiface)
Change the address of the selected device.
- ATCA_STATUS [releaseATCAIface](#) ([ATCAIface](#) ca_iface)
Instruct the HAL driver to release any resources associated with this interface.
- void [deleteATCAIface](#) ([ATCAIface](#) *ca_iface)
Instruct the HAL driver to release any resources associated with this interface, then delete the object.
- ATCADeviceType [iface_get_device_type_by_name](#) (const char *name)
Get the ATCADeviceType for a string that looks like a part number.

21.5.1 Detailed Description

Abstract interface to all CryptoAuth device types. This interface connects to the HAL implementation and abstracts the physical details of the device communication from all the upper layers of CryptoAuthLib.

21.5.2 Enumeration Type Documentation

21.5.2.1 ATCAIfaceType

enum `ATCAIfaceType`

Enumerator

<code>ATCA_I2C_IFACE</code>	Native I2C Driver
<code>ATCA_SWI_IFACE</code>	SWI or 1-Wire over UART/USART
<code>ATCA_UART_IFACE</code>	Kit v1 over UART/USART
<code>ATCA_SPI_IFACE</code>	Native SPI Driver
<code>ATCA_HID_IFACE</code>	Kit v1 over HID
<code>ATCA_KIT_IFACE</code>	Kit v2 (Binary/Bridging)
<code>ATCA_CUSTOM_IFACE</code>	Custom HAL functions provided during interface init
<code>ATCA_I2C_GPIO_IFACE</code>	I2C "Bitbang" Driver
<code>ATCA_SWI_GPIO_IFACE</code>	SWI or 1-Wire using a GPIO
<code>ATCA_SPI_GPIO_IFACE</code>	SWI or 1-Wire using a GPIO

21.5.3 Function Documentation

21.5.3.1 `atca_iface_is_kit()`

```
bool atca_iface_is_kit (
    ATCAIface ca_iface )
```

Check if the given interface is configured as a "kit protocol" one where transactions are atomic.

Returns

true if the interface is considered a kit

21.5 ATCAIface (atca_)

21.5.3.2 atca_iface_is_swi()

```
bool atca_iface_is_swi (
    ATCAIface ca_iface )
```

Check if the given interface is configured as a SWI.

Returns

true if the interface is considered a kit

21.5.3.3 atcontrol()

```
ATCA_STATUS atcontrol (
    ATCAIface ca_iface,
    uint8_t option,
    void * param,
    size_t paramlen )
```

Perform control operations with the underlying hal driver.

Parameters

in	<i>ca_iface</i>	Device to interact with.
in	<i>option</i>	Control parameter identifier
in	<i>param</i>	Optional pointer to parameter value
in	<i>paramlen</i>	Length of the parameter

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.5.3.4 atgetifacecfg()

```
ATCAIfaceCfg * atgetifacecfg (
    ATCAIface ca_iface )
```

Returns the logical interface configuration for the device.

Parameters

in	<i>ca_iface</i>	Device interface.
----	-----------------	-------------------

Returns

Logical interface configuration.

21.5.3.5 atgetifacehaldat()

```
void * atgetifacehaldat (
    ATCAIface ca_iface )
```

Returns the HAL data pointer for the device.

Parameters

in	<i>ca_iface</i>	Device interface.
----	-----------------	-------------------

Returns

HAL data pointer.

21.5.3.6 atidle()

```
ATCA_STATUS atidle (
    ATCAIface ca_iface )
```

Puts the device into idle state by calling intermediate HAL wrapper function. The [atcab_idle\(\)](#) function should be used instead.

Deprecated This function does not have defined behavior when ATCA_HAL_LEGACY_API is undefined.

Parameters

in	<i>ca_iface</i>	Device to interact with.
----	-----------------	--------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.5.3.7 atinit()

```
ATCA_STATUS atinit (
    ATCAIface ca_iface )
```

Performs the HAL initialization by calling intermediate HAL wrapper function. If using the basic API, the [atcab_init\(\)](#) function should be called instead.

21.5 ATCAIface (atca_)

Parameters

in	<i>ca_iface</i>	Device to interact with.
----	-----------------	--------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.5.3.8 atreceive()

```
ATCA_STATUS atreceive (
    ATCAIface ca_iface,
    uint8_t word_address,
    uint8_t * rxdata,
    uint16_t * rxlength )
```

Receives data from the device by calling intermediate HAL wrapper function.

Parameters

in	<i>ca_iface</i>	Device to interact with.
in	<i>word_address</i>	device transaction type
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.5.3.9 atsend()

```
ATCA_STATUS atsend (
    ATCAIface ca_iface,
    uint8_t word_address,
    uint8_t * txdata,
    int txlength )
```

Sends the data to the device by calling intermediate HAL wrapper function.

Parameters

in	<i>ca_iface</i>	Device to interact with.
in	<i>word_address</i>	device transaction type
in	<i>txdata</i>	Data to be transmitted to the device.
in	<i>txlength</i>	Number of bytes to be transmitted to the device.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.5.3.10 atsleep()

```
ATCA_STATUS atsleep (
    ATCAIface ca_iface )
```

Puts the device into sleep state by calling intermediate HAL wrapper function. The [atcab_sleep\(\)](#) function should be used instead.

Deprecated This function does not have defined behavior when ATCA_HAL_LEGACY_API is undefined.

Parameters

in	ca_iface	Device to interact with.
----	----------	--------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.5.3.11 atwake()

```
ATCA_STATUS atwake (
    ATCAIface ca_iface )
```

Wakes up the device by calling intermediate HAL wrapper function. The [atcab_wakeup\(\)](#) function should be used instead.

Deprecated This function does not have defined behavior when ATCA_HAL_LEGACY_API is undefined.

Parameters

in	ca_iface	Device to interact with.
----	----------	--------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.5 ATCAIface (atca_)

21.5.3.12 deleteATCAIface()

```
void deleteATCAIface (
    ATCAIface * ca_iface )
```

Instruct the HAL driver to release any resources associated with this interface, then delete the object.

Parameters

in	<i>ca_iface</i>	Device interface.
----	-----------------	-------------------

21.5.3.13 ifacecfg_set_address()

```
ATCA_STATUS ifacecfg_set_address (
    ATCAIfaceCfg * cfg,
    uint8_t address,
    ATCAKitType kitiface )
```

Change the address of the selected device.

Parameters

in	<i>cfg</i>	Interface configuration structure to update
in	<i>address</i>	Desired address
in	<i>kitiface</i>	Optional parameter to set the kit iface type

21.5.3.14 ifacetype_is_kit()

```
bool ifacetype_is_kit (
    ATCAIfaceType iface_type )
```

Check if the given interface is a "kit protocol" one.

Returns

true if the interface type is considered a kit

21.5.3.15 initATCAIface()

```
ATCA_STATUS initATCAIface (
    ATCAIfaceCfg * cfg,
    ATCAIface ca_iface )
```

Initializer for ATCAIface objects.

Parameters

in	<i>cfg</i>	Logical configuration for the interface
in	<i>ca_iface</i>	Interface structure to initialize.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.5.3.16 releaseATCAIface()

```
ATCA_STATUS releaseATCAIface (
    ATCAIface ca_iface )
```

Instruct the HAL driver to release any resources associated with this interface.

Parameters

in	<i>ca_iface</i>	Device interface.
----	-----------------	-------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.6 Certificate manipulation methods (atcacert_)

These methods provide convenient ways to perform certification I/O with CryptoAuth chips and perform certificate manipulation in memory.

Data Structures

- struct [atcacert_tm_utc_s](#)
- struct [atcacert_device_loc_s](#)
- struct [atcacert_cert_loc_s](#)
- struct [atcacert_cert_element_s](#)
- struct [atcacert_def_s](#)
- struct [atcacert_build_state_s](#)

Macros

- #define **FALSE** (0)
- #define **TRUE** (1)
- #define [ATCACERT_E_SUCCESS](#) ATCA_SUCCESS
- #define [ATCACERT_E_ERROR](#) ATCA_GEN_FAIL
- #define [ATCACERT_E_BAD_PARAMS](#) ATCA_BAD_PARAM

- #define `ATCACERT_E_BUFFER_TOO_SMALL` `ATCA_SMALL_BUFFER`
- #define `ATCACERT_E_UNIMPLEMENTED` `ATCA_UNIMPLEMENTED`
- #define `ATCACERT_E_DECODING_ERROR` 4
- #define `ATCACERT_E_INVALID_DATE` 5
- #define `ATCACERT_E_UNEXPECTED_ELEM_SIZE` 7
- #define `ATCACERT_E_ELEM_MISSING` 8
- #define `ATCACERT_E_ELEM_OUT_OF_BOUNDS` 9
- #define `ATCACERT_E_BAD_CERT` 10
- #define `ATCACERT_E_WRONG_CERT_DEF` 11
- #define `ATCACERT_E_VERIFY_FAILED` 12
- #define `ATCACERT_E_INVALID_TRANSFORM` 13
- #define `DATEFMT_ISO8601_SEP` (0U)
ISO8601 full date YYYY-MM-DDThh:mm:ssZ.
- #define `DATEFMT_RFC5280.UTC` (1U)
RFC 5280 (X.509) 4.1.2.5.1 UTCTime format YYMMDDhhmmssZ.
- #define `DATEFMT_POSIX_UINT32_BE` (2U)
POSIX (aka UNIX) date format. Seconds since Jan 1, 1970. 32 bit unsigned integer, big endian.
- #define `DATEFMT_POSIX_UINT32_LE` (3U)
POSIX (aka UNIX) date format. Seconds since Jan 1, 1970. 32 bit unsigned integer, little endian.
- #define `DATEFMT_RFC5280_GEN` (4U)
RFC 5280 (X.509) 4.1.2.5.2 GeneralizedTime format YYYYMMDDhhmmssZ.
- #define `DATEFMT_INVALID` (0xFFU)
- #define `DATEFMT_ISO8601_SEP_SIZE` (20)
- #define `DATEFMT_RFC5280.UTC_SIZE` (13)
- #define `DATEFMT_POSIX_UINT32_BE_SIZE` (4)
- #define `DATEFMT_POSIX_UINT32_LE_SIZE` (4)
- #define `DATEFMT_RFC5280_GEN_SIZE` (15)
- #define `DATEFMT_MAX_SIZE` `DATEFMT_ISO8601_SEP_SIZE`
- #define `ATCACERT_DATE_FORMAT_SIZES_COUNT` 5
- #define `atcacert_date_enc_posix_uint32_be` `atcacert_date_enc_posix_be`
- #define `atcacert_date_dec_posix_uint32_be` `atcacert_date_dec_posix_be`
- #define `atcacert_date_enc_posix_uint32_le` `atcacert_date_enc_posix_le`
- #define `atcacert_date_dec_posix_uint32_le` `atcacert_date_dec_posix_le`

Typedefs

- typedef struct `atcacert_tm_utc_s` `atcacert_tm_utc_t`
- typedef uint8_t `atcacert_date_format_t`
- typedef enum `atcacert_cert_type_e` `atcacert_cert_type_t`
- typedef enum `atcacert_cert_sn_src_e` `atcacert_cert_sn_src_t`
- typedef enum `atcacert_device_zone_e` `atcacert_device_zone_t`
- typedef enum `atcacert_transform_e` `atcacert_transform_t`
How to transform the data from the device to the certificate.
- typedef enum `atcacert_std_cert_element_e` `atcacert_std_cert_element_t`
- typedef struct `ATCA_PACKED atcacert_device_loc_s` `atcacert_device_loc_t`
- typedef struct `ATCA_PACKED atcacert_cert_loc_s` `atcacert_cert_loc_t`
- typedef struct `ATCA_PACKED atcacert_cert_element_s` `atcacert_cert_element_t`
- typedef struct `atcacert_def_s` `atcacert_def_t`
- typedef struct `atcacert_build_state_s` `atcacert_build_state_t`

Enumerations

- enum `atcacert_cert_type_e` { `CERTTYPE_X509`, `CERTTYPE_CUSTOM`, `CERTTYPE_X509_FULL_STORED` }
 - enum `atcacert_cert_sn_src_e` {
`SNSRC_STORED` = 0x0 , `SNSRC_STORED_DYNAMIC` = 0x7 , `SNSRC_DEVICE_SN` = 0x8 ,
`SNSRC_SIGNER_ID` = 0x9 ,
`SNSRC_PUB_KEY_HASH` = 0xA , `SNSRC_DEVICE_SN_HASH` = 0xB , `SNSRC_PUB_KEY_HASH_POS`
= 0xC , `SNSRC_DEVICE_SN_HASH_POS` = 0xD ,
`SNSRC_PUB_KEY_HASH_RAW` = 0xE , `SNSRC_DEVICE_SN_HASH_RAW` = 0xF }
 - enum `atcacert_device_zone_e` {
`DEVZONE_CONFIG` = 0x00 , `DEVZONE_OTP` = 0x01 , `DEVZONE_DATA` = 0x02 , `DEVZONE_GENKEY` =
0x03 ,
`DEVZONE_DEDICATED_DATA` = 0x04 , `DEVZONE_NONE` = 0x07 }
 - enum `atcacert_transform_e` {
`TF_NONE` , `TF_REVERSE` , `TF_BIN2HEX_UC` , `TF_BIN2HEX_LC` ,
`TF_HEX2BIN_UC` , `TF_HEX2BIN_LC` , `TF_BIN2HEX_SPACE_UC` , `TF_BIN2HEX_SPACE_LC` ,
`TF_HEX2BIN_SPACE_UC` , `TF_HEX2BIN_SPACE_LC` }
- How to transform the data from the device to the certificate.*
- enum `atcacert_std_cert_element_e` {
`STDCERT_PUBLIC_KEY` , `STDCERT_SIGNATURE` , `STDCERT_ISSUE_DATE` , `STDCERT_EXPIRE_↔`
`DATE` ,
`STDCERT_SIGNER_ID` , `STDCERT_CERT_SN` , `STDCERT_AUTH_KEY_ID` , `STDCERT_SUBJ_KEY_ID` ,
`STDCERT_NUM_ELEMENTS` }

Functions

- ATCA_STATUS `atcacert_date_enc` (`atcacert_date_format_t` format, const `atcacert_tm_utc_t` *timestamp, `uint8_t` *formatted_date, `size_t` *formatted_date_size)
Format a timestamp according to the format type.
- ATCA_STATUS `atcacert_date_dec` (`atcacert_date_format_t` format, const `uint8_t` *formatted_date, `size_t` ↔
formatted_date_size, `atcacert_tm_utc_t` *timestamp)
Parse a formatted timestamp according to the specified format.
- ATCA_STATUS `atcacert_date_enc_compcert` (const `atcacert_tm_utc_t` *issue_date, `uint8_t` expire_years,
`uint8_t` enc_dates[3])
Encode the issue and expire dates in the format used by the compressed certificate.
- ATCA_STATUS `atcacert_date_enc_compcert_ext` (const `atcacert_tm_utc_t` *issue_date, `uint8_t` expire_↔
years, `uint8_t` comp_cert[ATCACERT_COMP_CERT_MAX_SIZE])
Encode the issue and expire dates in the format used by the compressed certificate.
- ATCA_STATUS `atcacert_date_dec_compcert` (const `uint8_t` enc_dates[3], `atcacert_date_format_t` expire_↔
date_format, `atcacert_tm_utc_t` *issue_date, `atcacert_tm_utc_t` *expire_date)
Decode the issue and expire dates from the format used by the compressed certificate.
- ATCA_STATUS `atcacert_date_dec_compcert_ext` (const `uint8_t` comp_cert[ATCACERT_COMP_CERT_↔
MAX_SIZE], `atcacert_date_format_t` expire_date_format, `atcacert_tm_utc_t` *issue_date, `atcacert_tm_utc_t`
*expire_date)
Decode the issue and expire dates from the format used by the compressed certificate.
- `atcacert_date_format_t` `atcacert_date_from_asn1_tag` (const `uint8_t` tag)
Convert the asn1 tag for the supported time formats into the local time format.
- ATCA_STATUS `atcacert_date_get_max_date` (`atcacert_date_format_t` format, `atcacert_tm_utc_t` *timestamp)
Return the maximum date available for the given format.
- ATCA_STATUS `atcacert_date_enc_iso8601_sep` (const `atcacert_tm_utc_t` *timestamp, `uint8_t` ↔
formatted_date[(20)])
- ATCA_STATUS `atcacert_date_dec_iso8601_sep` (const `uint8_t` formatted_date[(20)], `atcacert_tm_utc_t`
*timestamp)

- ATCA_STATUS **atcacert_date_enc_rfc5280_utc** (const [atcacert_tm_utc_t](#) *timestamp, uint8_t formatted_date[(13)])
- ATCA_STATUS **atcacert_date_dec_rfc5280_utc** (const uint8_t formatted_date[(13)], [atcacert_tm_utc_t](#) *timestamp)
- ATCA_STATUS **atcacert_date_enc_rfc5280_gen** (const [atcacert_tm_utc_t](#) *timestamp, uint8_t formatted_date[(15)])
- ATCA_STATUS **atcacert_date_dec_rfc5280_gen** (const uint8_t formatted_date[(15)], [atcacert_tm_utc_t](#) *timestamp)
- ATCA_STATUS **atcacert_date_enc_posix_be** (const [atcacert_tm_utc_t](#) *timestamp, uint8_t formatted_date[(4)])
- ATCA_STATUS **atcacert_date_dec_posix_be** (const uint8_t formatted_date[(4)], [atcacert_tm_utc_t](#) *timestamp)
- ATCA_STATUS **atcacert_date_enc_posix_le** (const [atcacert_tm_utc_t](#) *timestamp, uint8_t formatted_date[(4)])
- ATCA_STATUS **atcacert_date_dec_posix_le** (const uint8_t formatted_date[(4)], [atcacert_tm_utc_t](#) *timestamp)
- int **atcacert_date_cmp** (const [atcacert_tm_utc_t](#) *timestamp1, const [atcacert_tm_utc_t](#) *timestamp2)
Compare two dates.
- ATCA_STATUS **atcacert_get_subject** (const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size, [cal_buffer](#) *cert_subj_buf)
Gets the subject name from a certificate.
- ATCA_STATUS **atcacert_get_subj_public_key** (const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size, [cal_buffer](#) *subj_public_key)
Gets the subject public key from a certificate.
- ATCA_STATUS **atcacert_get_subj_key_id** (const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size, uint8_t subj_key_id[20])
Gets the subject key ID from a certificate.
- ATCA_STATUS **atcacert_get_issuer** (const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size, uint8_t cert_issuer[128])
Gets the issuer name of a certificate.
- ATCA_STATUS **atcacert_get_issue_date** (const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size, [atcacert_tm_utc_t](#) *timestamp)
Gets the issue date from a certificate. Will be parsed according to the date format specified in the certificate definition.
- ATCA_STATUS **atcacert_get_expire_date** (const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size, [atcacert_tm_utc_t](#) *timestamp)
Gets the expire date from a certificate. Will be parsed according to the date format specified in the certificate definition.
- ATCA_STATUS **atcacert_get_cert_sn** (const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size, uint8_t *cert_sn, size_t *cert_sn_size)
Gets the certificate serial number from a certificate.
- ATCA_STATUS **atcacert_get_auth_key_id** (const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size, uint8_t auth_key_id[20])
Gets the authority key ID from a certificate.
- ATCA_STATUS **atcacert_der_enc_length** (size_t length, uint8_t *der_length, size_t *der_length_size)
Encode a length in DER format.
- ATCA_STATUS **atcacert_der_dec_length** (const uint8_t *der_length, size_t *der_length_size, size_t *length)
Decode a DER format length.
- ATCA_STATUS **atcacert_der_adjust_length** (uint8_t *der_length, size_t *der_length_size, int delta_length, size_t *new_length)
- ATCA_STATUS **atcacert_der_enc_integer** (const uint8_t *int_data, size_t int_data_size, uint8_t is_unsigned, uint8_t *der_int, size_t *der_int_size)
Encode an ASN.1 integer in DER format, including tag and length fields.
- ATCA_STATUS **atcacert_der_dec_integer** (const uint8_t *der_int, size_t *der_int_size, uint8_t *int_data, size_t *int_data_size)
Decode an ASN.1 DER encoded integer.

- ATCA_STATUS [atcacert_der_enc_ecdsa_sig_value](#) (const [cal_buffer](#) *raw_sig, uint8_t *der_sig, size_t *der_sig_size)

Formats a raw ECDSA P256 signature in the DER encoding found in X.509 certificates.

- ATCA_STATUS [atcacert_der_dec_ecdsa_sig_value](#) (const uint8_t *der_sig, size_t *der_sig_size, [cal_buffer](#) *raw_sig)

Parses an ECDSA P256/P384/P521 signature in the DER encoding as found in X.509 certificates.

Variables

- const size_t **ATCACERT_DATE_FORMAT_SIZES** [5]

21.6.1 Detailed Description

These methods provide convenient ways to perform certification I/O with CryptoAuth chips and perform certificate manipulation in memory.

21.6.2 Macro Definition Documentation

21.6.2.1 ATCACERT_E_BAD_CERT

```
#define ATCACERT_E_BAD_CERT 10
```

Certificate structure is bad in some way.

21.6.2.2 ATCACERT_E_BAD_PARAMS

```
#define ATCACERT_E_BAD_PARAMS ATCA\_BAD\_PARAM
```

Invalid/bad parameter passed to function.

21.6.2.3 ATCACERT_E_BUFFER_TOO_SMALL

```
#define ATCACERT_E_BUFFER_TOO_SMALL ATCA\_SMALL\_BUFFER
```

Supplied buffer for output is too small to hold the result.

21.6.2.4 ATCACERT_E_DECODING_ERROR

```
#define ATCACERT_E_DECODING_ERROR 4
```

Data being decoded/parsed has an invalid format.

21.6.2.5 ATCACERT_E_ELEM_MISSING

```
#define ATCACERT_E_ELEM_MISSING 8
```

The certificate element isn't defined for the certificate definition.

21.6.2.6 ATCACERT_E_ELEM_OUT_OF_BOUNDS

```
#define ATCACERT_E_ELEM_OUT_OF_BOUNDS 9
```

Certificate element is out of bounds for the given certificate.

21.6.2.7 ATCACERT_E_ERROR

```
#define ATCACERT_E_ERROR ATCA_GEN_FAIL
```

General error.

21.6.2.8 ATCACERT_E_INVALID_DATE

```
#define ATCACERT_E_INVALID_DATE 5
```

Date is invalid.

21.6.2.9 ATCACERT_E_INVALID_TRANSFORM

```
#define ATCACERT_E_INVALID_TRANSFORM 13
```

Invalid transform passed to function.

21.6.2.10 ATCACERT_E_SUCCESS

```
#define ATCACERT_E_SUCCESS ATCA_SUCCESS
```

Operation completed successfully.

21.6.2.11 ATCACERT_E_UNEXPECTED_ELEM_SIZE

```
#define ATCACERT_E_UNEXPECTED_ELEM_SIZE 7
```

A certificate element size was not what was expected.

21.6.2.12 ATCACERT_E_UNIMPLEMENTED

```
#define ATCACERT_E_UNIMPLEMENTED ATCA_UNIMPLEMENTED
```

Function is unimplemented for the current configuration.

21.6.2.13 ATCACERT_E_VERIFY_FAILED

```
#define ATCACERT_E_VERIFY_FAILED 12
```

Certificate or challenge/response verification failed.

21.6.2.14 DATEFMT_ISO8601_SEP

```
#define DATEFMT_ISO8601_SEP (0U)
```

ISO8601 full date YYYY-MM-DDThh:mm:ssZ.

Date formats.

21.6.3 Typedef Documentation

21.6.3.1 atccert_build_state_t

```
typedef struct atccert_build_state_s atccert_build_state_t
```

Tracks the state of a certificate as it's being rebuilt from device information.

21.6.3.2 atccert_cert_element_t

```
typedef struct ATCA_PACKED atccert_cert_element_s atccert_cert_element_t
```

Defines a generic dynamic element for a certificate including the device and template locations.

21.6.3.3 atccert_cert_loc_t

```
typedef struct ATCA_PACKED atccert_cert_loc_s atccert_cert_loc_t
```

Defines a chunk of data in a certificate template.

21.6.3.4 atccert_cert_sn_src_t

```
typedef enum atccert_cert_sn_src_e atccert_cert_sn_src_t
```

Sources for the certificate serial number.

21.6.3.5 atccert_cert_type_t

```
typedef enum atccert_cert_type_e atccert_cert_type_t
```

Types of certificates.

21.6 Certificate manipulation methods (atcacert_)

21.6.3.6 atcacert_def_t

```
typedef struct atcacert_def_s atcacert_def_t
```

Defines a certificate and all the pieces to work with it.

If any of the standard certificate elements (std_cert_elements) are not a part of the certificate definition, set their count to 0 to indicate their absence.

21.6.3.7 atcacert_device_loc_t

```
typedef struct ATCA_PACKED atcacert_device_loc_s atcacert_device_loc_t
```

Defines a chunk of data in an ATECC device.

21.6.3.8 atcacert_device_zone_t

```
typedef enum atcacert_device_zone_e atcacert_device_zone_t
```

ATECC device zones. The values match the Zone Encodings as specified in the datasheet.

21.6.3.9 atcacert_std_cert_element_t

```
typedef enum atcacert_std_cert_element_e atcacert_std_cert_element_t
```

Standard dynamic certificate elements.

21.6.3.10 atcacert_tm_utc_t

```
typedef struct atcacert_tm_utc_s atcacert_tm_utc_t
```

Holds a broken-down date in UTC. Mimics atcacert_tm_utc_t from time.h.

21.6.4 Enumeration Type Documentation

Enumerator

21.6.4.1 atcacert_cert_sn_src_e

```
enum atcacert_cert_sn_src_e
```

Sources for the certificate serial number.

Enumerator

SNSRC_STORED	Cert serial is stored on the device.
SNSRC_STORED_DYNAMIC	Cert serial is stored on the device with the first byte being the DER size (X509 certs only).
SNSRC_DEVICE_SN	Cert serial number is 0x40(MSB) + 9-byte device serial number. Only applies to device certificates.
SNSRC_SIGNER_ID	Cert serial number is 0x40(MSB) + 2-byte signer ID. Only applies to signer certificates.
SNSRC_PUB_KEY_HASH	Cert serial number is the SHA256(Subject public key + Encoded dates), with uppermost 2 bits set to 01.
SNSRC_DEVICE_SN_HASH	Cert serial number is the SHA256(Device SN + Encoded dates), with uppermost 2 bits set to 01. Only applies to device certificates.
SNSRC_PUB_KEY_HASH_POS	Deprecated, don't use. Cert serial number is the SHA256(Subject public key + Encoded dates), with MSBit set to 0 to ensure it's positive.
SNSRC_DEVICE_SN_HASH_POS	Deprecated, don't use. Cert serial number is the SHA256(Device SN + Encoded dates), with MSBit set to 0 to ensure it's positive. Only applies to device certificates.
SNSRC_PUB_KEY_HASH_RAW	Deprecated, don't use. Cert serial number is the SHA256(Subject public key + Encoded dates).
SNSRC_DEVICE_SN_HASH_RAW	Deprecated, don't use. Cert serial number is the SHA256(Device SN + Encoded dates). Only applies to device certificates.

21.6.4.2 atcacert_cert_type_e

```
enum atcacert_cert_type_e
```

Types of certificates.

Enumerator

CERTTYPE_X509	Standard X509 certificate.
CERTTYPE_CUSTOM	Custom format.
CERTTYPE_X509_FULL_STORED	Full Stored X509 Certificate.

21.6 Certificate manipulation methods (atcacert_)

21.6.4.3 atcacert_device_zone_e

enum `atcacert_device_zone_e`

ATECC device zones. The values match the Zone Encodings as specified in the datasheet.

Enumerator

DEVZONE_CONFIG	Configuration zone.
DEVZONE_OTP	One Time Programmable zone.
DEVZONE_DATA	Data zone (slots).
DEVZONE_GENKEY	Data zone - Generate Pubkey (slots).
DEVZONE_DEDICATED_DATA	Dedicated data zone.
DEVZONE_NONE	Special value used to indicate there is no device location.

21.6.4.4 atcacert_std_cert_element_e

enum `atcacert_std_cert_element_e`

Standard dynamic certificate elements.

Enumerator

STDCERT_NUM_ELEMENTS	Special item to give the number of elements in this enum.
----------------------	---

21.6.4.5 atcacert_transform_e

enum `atcacert_transform_e`

How to transform the data from the device to the certificate.

Enumerator

TF_NONE	No transform, data is used byte for byte.
TF_REVERSE	Reverse the bytes (e.g. change endianness)
TF_BIN2HEX_UC	Convert raw binary into ASCII hex, uppercase.
TF_BIN2HEX_LC	Convert raw binary into ASCII hex, lowercase.
TF_HEX2BIN_UC	Convert ASCII hex, uppercase to binary.
TF_HEX2BIN_LC	Convert ASCII hex, lowercase to binary.
TF_BIN2HEX_SPACE_UC	Convert raw binary into ASCII hex, uppercase space between bytes.
TF_BIN2HEX_SPACE_LC	Convert raw binary into ASCII hex, lowercase space between bytes.
TF_HEX2BIN_SPACE_UC	Convert ASCII hex, uppercase with spaces between bytes to binary.
TF_HEX2BIN_SPACE_LC	Convert ASCII hex, lowercase with spaces between bytes to binary.

21.6.5 Function Documentation

21.6.5.1 atcacert_date_cmp()

```
int atcacert_date_cmp (
    const atcacert_tm_utc_t * timestamp1,
    const atcacert_tm_utc_t * timestamp2 )
```

Compare two dates.

Dates are not checked for validity before comparing.

Parameters

in	<i>timestamp1</i>	First date to compare.
in	<i>timestamp2</i>	Second date to compare.

Returns

-1 if timestamp1 is before timestamp2, 0 if they are equal, 1 if they are timestamp1 is after timestamp2.
 ATCACERT_E_BAD_PARAMS if either input is NULL.

21.6.5.2 atcacert_date_dec()

```
ATCA_STATUS atcacert_date_dec (
    atcacert_date_format_t format,
    const uint8_t * formatted_date,
    size_t formatted_date_size,
    atcacert_tm_utc_t * timestamp )
```

Parse a formatted timestamp according to the specified format.

Parameters

in	<i>format</i>	Format to parse the formatted date as.
in	<i>formatted_date</i>	Formatted date to be parsed.
in	<i>formatted_date_size</i>	Size of the formatted date in bytes.
out	<i>timestamp</i>	Parsed timestamp is returned here.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.6 Certificate manipulation methods (atcacert_)

21.6.5.3 atcacert_date_dec_compcert()

```
ATCA_STATUS atcacert_date_dec_compcert (
    const uint8_t enc_dates[3],
    atcacert_date_format_t expire_date_format,
    atcacert_tm_utc_t * issue_date,
    atcacert_tm_utc_t * expire_date )
```

Decode the issue and expire dates from the format used by the compressed certificate.

Parameters

in	<i>enc_dates</i>	Encoded date from the compressed certificate. 3 bytes.
in	<i>expire_date_format</i>	Expire date format. Only used to determine max date when no expiration date is specified by the encoded date.
out	<i>issue_date</i>	Decoded issue date is returned here.
out	<i>expire_date</i>	Decoded expire date is returned here. If there is no expiration date, the expire date will be set to a maximum value for the given <i>expire_date_format</i> .

Returns

0 on success

21.6.5.4 atcacert_date_dec_compcert_ext()

```
ATCA_STATUS atcacert_date_dec_compcert_ext (
    const uint8_t comp_cert[ATCACERT_COMP_CERT_MAX_SIZE],
    atcacert_date_format_t expire_date_format,
    atcacert_tm_utc_t * issue_date,
    atcacert_tm_utc_t * expire_date )
```

Decode the issue and expire dates from the format used by the compressed certificate.

Supports extended dates if the format version field is 1

Parameters

in, out	<i>comp_cert</i>	Compressed certificate (72 bytes) where the encoded dates will be set. Format version (In <i>comp_cert</i> byte 70([3:0]) must be set to 1 to use extended dates.
in	<i>expire_date_format</i>	Expire date format. Only used to determine max date when no expiration date is specified by the encoded date.
out	<i>issue_date</i>	Decoded issue date is returned here.
out	<i>expire_date</i>	Decoded expire date is returned here. If there is no expiration date, the expire date will be set to a maximum value for the given <i>expire_date_format</i> .

Returns

0 on success

21.6.5.5 atcacert_date_enc()

```
ATCA_STATUS atcacert_date_enc (
    atcacert_date_format_t format,
    const atcacert_tm_utc_t * timestamp,
    uint8_t * formatted_date,
    size_t * formatted_date_size )
```

Format a timestamp according to the format type.

Parameters

in	<i>format</i>	Format to use.
in	<i>timestamp</i>	Timestamp to format.
out	<i>formatted_date</i>	Formatted date will be returned in this buffer.
in, out	<i>formatted_date_size</i>	As input, the size of the formatted_date buffer. As output, the size of the returned formatted_date.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.6.5.6 atcacert_date_enc_compcert()

```
ATCA_STATUS atcacert_date_enc_compcert (
    const atcacert_tm_utc_t * issue_date,
    uint8_t expire_years,
    uint8_t enc_dates[3] )
```

Encode the issue and expire dates in the format used by the compressed certificate.

Parameters

in	<i>issue_date</i>	Issue date to encode. Note that minutes and seconds will be ignored.
in	<i>expire_years</i>	Expire date is expressed as a number of years past the issue date. 0 should be used if there is no expire date.
out	<i>enc_dates</i>	Encoded dates for use in the compressed certificate is returned here. 3 bytes.

Returns

0 on success

21.6 Certificate manipulation methods (atcacert_)

21.6.5.7 atcacert_date_enc_compcert_ext()

```
ATCA_STATUS atcacert_date_enc_compcert_ext (
    const atcacert_tm_utc_t * issue_date,
    uint8_t expire_years,
    uint8_t comp_cert[ATCACERT_COMP_CERT_MAX_SIZE] )
```

Encode the issue and expire dates in the format used by the compressed certificate.

Supports extended dates if the format version field is set appropriately (currently 1).

Parameters

in	<i>issue_date</i>	Issue date to encode. Note that minutes and seconds will be ignored.
in	<i>expire_years</i>	Expire date is expressed as a number of years past the issue date. 0 should be used if there is no expire date.
in, out	<i>comp_cert</i>	Compressed certificate (72 bytes) where the encoded dates will be set. Format version must be set appropriately.

Returns

0 on success

21.6.5.8 atcacert_date_from_asn1_tag()

```
atcacert_date_format_t atcacert_date_from_asn1_tag (
    const uint8_t tag )
```

Convert the asn1 tag for the supported time formats into the local time format.

Returns

DATEFMT_RFC5280_UTC, DATEFMT_RFC5280_GEN, or DATEFMT_INVALID

21.6.5.9 atcacert_date_get_max_date()

```
ATCA_STATUS atcacert_date_get_max_date (
    atcacert_date_format_t format,
    atcacert_tm_utc_t * timestamp )
```

Return the maximum date available for the given format.

Parameters

in	<i>format</i>	Format to get the max date for.
out	<i>timestamp</i>	Max date is returned here.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.6.5.10 atcacert_der_dec_ecdsa_sig_value()

```
ATCA_STATUS atcacert_der_dec_ecdsa_sig_value (
    const uint8_t * der_sig,
    size_t * der_sig_size,
    cal_buffer * raw_sig )
```

Parses an ECDSA P256/P384/P521 signature in the DER encoding as found in X.509 certificates.

This will parse the DER encoding of the signatureValue field as found in an X.509 certificate (RFC 5280). x509_sig should include the tag, length, and value. The value of the signatureValue is the DER encoding of the ECDSA-Sig-Value as specified by RFC 5480 and SECG SEC1.

Parameters

in	der_sig	X.509 format signature (TLV of signatureValue) to be parsed.
in, out	der_sig_size	As input, size of the der_sig buffer in bytes. As output, size of the DER x.509 signature parsed from the buffer.
out	raw_sig	Buffer pointing to the parsed P256/P384/P521 ECDSA signature will be returned. Formatted as R and S integers concatenated together.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.6.5.11 atcacert_der_dec_integer()

```
ATCA_STATUS atcacert_der_dec_integer (
    const uint8_t * der_int,
    size_t * der_int_size,
    uint8_t * int_data,
    size_t * int_data_size )
```

Decode an ASN.1 DER encoded integer.

X.680 (<http://www.itu.int/rec/T-REC-X.680/en>) section 19.8, for tag value X.690 (<http://www.itu.int/rec/T-REC-X.690/en>) section 8.3, for encoding

Parameters

in	der_int	DER encoded ASN.1 integer, including the tag and length fields.
in, out	der_int_size	As input, the size of the der_int buffer in bytes. As output, the size of the DER integer decoded in bytes.
out	int_data	Decode integer is returned in this buffer in a signed big-endian format.
in, out	int_data_size	As input, the size of int_data in bytes. As output, the size of the decoded integer in bytes.

21.6 Certificate manipulation methods (atcacert_)

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.6.5.12 atcacert_der_dec_length()

```
ATCA_STATUS atcacert_der_dec_length (
    const uint8_t * der_length,
    size_t * der_length_size,
    size_t * length )
```

Decode a DER format length.

X.690 (<http://www.itu.int/rec/T-REC-X.690/en>) section 8.1.3, for encoding

Parameters

in	<i>der_length</i>	DER encoded length.
in, out	<i>der_length_size</i>	As input, the size of the der_length buffer in bytes. As output, the size of the DER encoded length that was decoded.
out	<i>length</i>	Decoded length is returned here.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.6.5.13 atcacert_der_enc_ecdsa_sig_value()

```
ATCA_STATUS atcacert_der_enc_ecdsa_sig_value (
    const cal_buffer * raw_sig,
    uint8_t * der_sig,
    size_t * der_sig_size )
```

Formats a raw ECDSA P256 signature in the DER encoding found in X.509 certificates.

This will return the DER encoding of the signatureValue field as found in an X.509 certificate (RFC 5280). This include the tag, length, and value. The value of the signatureValue is the DER encoding of the ECDSA-Sig-Value as specified by RFC 5480 and SECG SEC1.

Parameters

in	<i>raw_sig</i>	Buffer pointing to the P256/P384/P521 ECDSA signature to be formatted. Input format is R and S integers concatenated together.
out	<i>der_sig</i>	X.509 format signature (TLV of signatureValue) will be returned in this buffer.
in, out	<i>der_sig_size</i>	As input, the size of the x509_sig buffer in bytes. As output, the size of the returned X.509 signature in bytes.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.6.5.14 atcacert_der_enc_integer()

```
ATCA_STATUS atcacert_der_enc_integer (
    const uint8_t * int_data,
    size_t int_data_size,
    uint8_t is_unsigned,
    uint8_t * der_int,
    size_t * der_int_size )
```

Encode an ASN.1 integer in DER format, including tag and length fields.

X.680 (<http://www.itu.int/rec/T-REC-X.680/en>) section 19.8, for tag value X.690 (<http://www.itu.int/rec/T-REC-X.690/en>) section 8.3, for encoding

Parameters

in	<i>int_data</i>	Raw integer in big-endian format.
in	<i>int_data_size</i>	Size of the raw integer in bytes.
in	<i>is_unsigned</i>	Indicate whether the input integer should be treated as unsigned.
out	<i>der_int</i>	DER encoded integer is returned in this buffer.
in, out	<i>der_int_size</i>	As input, the size of the der_int buffer in bytes. As output, the size of the DER integer returned in bytes.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.6.5.15 atcacert_der_enc_length()

```
ATCA_STATUS atcacert_der_enc_length (
    size_t length,
    uint8_t * der_length,
    size_t * der_length_size )
```

Encode a length in DER format.

X.690 (<http://www.itu.int/rec/T-REC-X.690/en>) section 8.1.3, for encoding

Parameters

in	<i>length</i>	Length to be encoded.
out	<i>der_length</i>	DER encoded length will returned in this buffer.
in, out	<i>der_length_size</i>	As input, size of der_length buffer in bytes. As output, the size of the DER length encoding in bytes.

21.6 Certificate manipulation methods (atcacert_)

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.6.5.16 atcacert_get_auth_key_id()

```
ATCA_STATUS atcacert_get_auth_key_id (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    uint8_t auth_key_id[20] )
```

Gets the authority key ID from a certificate.

Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to get element from.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>auth_key_id</i>	Authority key ID is returned in this buffer. 20 bytes.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.6.5.17 atcacert_get_cert_sn()

```
ATCA_STATUS atcacert_get_cert_sn (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    uint8_t * cert_sn,
    size_t * cert_sn_size )
```

Gets the certificate serial number from a certificate.

Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to get element from.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>cert_sn</i>	Certificate SN will be returned in this buffer.
in, out	<i>cert_sn_size</i>	As input, the size of the cert_sn buffer. As output, the size of the certificate SN (cert_sn) in bytes.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.6.5.18 atcacert_get_expire_date()

```
ATCA_STATUS atcacert_get_expire_date (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    atcacert_tm_utc_t * timestamp )
```

Gets the expire date from a certificate. Will be parsed according to the date format specified in the certificate definition.

Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to get element from.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>timestamp</i>	Expire date is returned in this structure.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.6.5.19 atcacert_get_issue_date()

```
ATCA_STATUS atcacert_get_issue_date (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    atcacert_tm_utc_t * timestamp )
```

Gets the issue date from a certificate. Will be parsed according to the date format specified in the certificate definition.

Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to get element from.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>timestamp</i>	Issue date is returned in this structure.

21.6 Certificate manipulation methods (atcacert_)

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.6.5.20 atcacert_get_issuer()

```
ATCA_STATUS atcacert_get_issuer (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    uint8_t cert_issuer[128] )
```

Gets the issuer name of a certificate.

Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to get element from.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>cert_issuer</i>	Certificate's issuer is returned in this buffer.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.6.5.21 atcacert_get_subj_key_id()

```
ATCA_STATUS atcacert_get_subj_key_id (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    uint8_t subj_key_id[20] )
```

Gets the subject key ID from a certificate.

Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to get element from.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>subj_key_id</i>	Subject key ID is returned in this buffer. 20 bytes.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.6.5.22 atcacert_get_subj_public_key()

```
ATCA_STATUS atcacert_get_subj_public_key (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    cal_buffer * subj_public_key )
```

Gets the subject public key from a certificate.

Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to get element from.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>subj_public_key</i>	Subject public key is returned in the buffer pointed by subj_public_key

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.6.5.23 atcacert_get_subject()

```
ATCA_STATUS atcacert_get_subject (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    cal_buffer * cert_subj_buf )
```

Gets the subject name from a certificate.

Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to get element from.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>subject</i>	Subject name is returned in this buffer.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

21.7 Basic Crypto API methods for CryptoAuth Devices (calib_)

These methods provide a simple API to CryptoAuth chips.

21.7.0.1 calib directory - Purpose

The purpose of this directory is to contain the files implementing the APIs for a basic interface to the core CryptoAuthLib library.

High-level functions like these make it very convenient to use the library when standard configurations and defaults are in play. They are the easiest to use when developing examples or trying to understand the "flow" of an authentication operation without getting overwhelmed by the details.

This makes simple jobs easy and if you need more sophistication and power, you can employ the full power of the CryptoAuthLib object model.

See the Doxygen documentation in `cryptoauthlib/docs` for details on the API of the calib commands.

Data Structures

- struct [atca_sha256_ctx](#)
- struct [atsha204a_config_s](#)
- struct [atecc508a_config_s](#)
- struct [atecc608_config_s](#)

Macros

- `#define ATCA_AES_ENABLE_EN_SHIFT (0)`
- `#define ATCA_AES_ENABLE_EN_MASK (0x01u << ATCA_AES_ENABLE_EN_SHIFT)`
- `#define ATCA_I2C_ENABLE_EN_SHIFT (0)`
- `#define ATCA_I2C_ENABLE_EN_MASK (0x01u << ATCA_I2C_ENABLE_EN_SHIFT)`
- `#define ATCA_COUNTER_MATCH_EN_SHIFT (0)`
- `#define ATCA_COUNTER_MATCH_EN_MASK (0x01u << ATCA_COUNTER_MATCH_EN_SHIFT)`
- `#define ATCA_COUNTER_MATCH_KEY_SHIFT (4)`
- `#define ATCA_COUNTER_MATCH_KEY_MASK (0x0Fu << ATCA_COUNTER_MATCH_KEY_SHIFT)`
- `#define ATCA_COUNTER_MATCH_KEY(v) (ATCA_COUNTER_MATCH_KEY_MASK & ((v) << ATCA_COUNTER_MATCH_KEY_SHIFT))`
- `#define ATCA_CHIP_MODE_I2C_EXTRA_SHIFT (0)`
- `#define ATCA_CHIP_MODE_I2C_EXTRA_MASK (0x01u << ATCA_CHIP_MODE_I2C_EXTRA_SHIFT)`
- `#define ATCA_CHIP_MODE_TTL_EN_SHIFT (1)`
- `#define ATCA_CHIP_MODE_TTL_EN_MASK (0x01u << ATCA_CHIP_MODE_TTL_EN_SHIFT)`
- `#define ATCA_CHIP_MODE_WDG_LONG_SHIFT (2)`
- `#define ATCA_CHIP_MODE_WDG_LONG_MASK (0x01u << ATCA_CHIP_MODE_WDG_LONG_SHIFT)`
- `#define ATCA_CHIP_MODE_CLK_DIV_SHIFT (3)`
- `#define ATCA_CHIP_MODE_CLK_DIV_MASK (0x1Fu << ATCA_CHIP_MODE_CLK_DIV_SHIFT)`
- `#define ATCA_CHIP_MODE_CLK_DIV(v) (ATCA_CHIP_MODE_CLK_DIV_MASK & ((v) << ATCA_CHIP_MODE_CLK_DIV_SHIFT))`
- `#define ATCA_SLOT_CONFIG_READKEY_SHIFT (0)`
- `#define ATCA_SLOT_CONFIG_READKEY_MASK (0x0Fu << ATCA_SLOT_CONFIG_READKEY_SHIFT)`
- `#define ATCA_SLOT_CONFIG_READKEY(v) (ATCA_SLOT_CONFIG_READKEY_MASK & ((v) << ATCA_SLOT_CONFIG_READKEY_SHIFT))`
- `#define ATCA_SLOT_CONFIG_NOMAC_SHIFT (4)`
- `#define ATCA_SLOT_CONFIG_NOMAC_MASK (0x01u << ATCA_SLOT_CONFIG_NOMAC_SHIFT)`
- `#define ATCA_SLOT_CONFIG_LIMITED_USE_SHIFT (5)`
- `#define ATCA_SLOT_CONFIG_LIMITED_USE_MASK (0x01u << ATCA_SLOT_CONFIG_LIMITED_USE_SHIFT)`
- `#define ATCA_SLOT_CONFIG_ENC_READ_SHIFT (6)`

- `#define ATCA_SLOT_CONFIG_ENC_READ_MASK (0x01u << ATCA_SLOT_CONFIG_ENC_READ_↵SHIFT)`
- `#define ATCA_SLOT_CONFIG_IS_SECRET_SHIFT (7)`
- `#define ATCA_SLOT_CONFIG_IS_SECRET_MASK (0x01u << ATCA_SLOT_CONFIG_IS_SECRET_↵SHIFT)`
- `#define ATCA_SLOT_CONFIG_WRITE_KEY_SHIFT (8)`
- `#define ATCA_SLOT_CONFIG_WRITE_KEY_MASK ((uint32_t)0x0Fu << ATCA_SLOT_CONFIG_↵WRITE_KEY_SHIFT)`
- `#define ATCA_SLOT_CONFIG_WRITE_KEY(v) (ATCA_SLOT_CONFIG_WRITE_KEY_MASK & ((v) << ATCA_SLOT_CONFIG_WRITE_KEY_SHIFT))`
- `#define ATCA_SLOT_CONFIG_WRITE_CONFIG_SHIFT (12)`
- `#define ATCA_SLOT_CONFIG_WRITE_CONFIG_MASK (((uint32_t)0x0Fu << ATCA_SLOT_CONFIG_↵WRITE_CONFIG_SHIFT))`
- `#define ATCA_SLOT_CONFIG_WRITE_CONFIG(v) ((ATCA_SLOT_CONFIG_WRITE_CONFIG_MASK & ((uint32_t)(v) << ATCA_SLOT_CONFIG_WRITE_CONFIG_SHIFT)))`
- `#define ATCA_SLOT_CONFIG_EXT_SIG_SHIFT (0)`
- `#define ATCA_SLOT_CONFIG_EXT_SIG_MASK (0x01u << ATCA_SLOT_CONFIG_EXT_SIG_SHIFT)`
- `#define ATCA_SLOT_CONFIG_INT_SIG_SHIFT (1)`
- `#define ATCA_SLOT_CONFIG_INT_SIG_MASK (0x01u << ATCA_SLOT_CONFIG_INT_SIG_SHIFT)`
- `#define ATCA_SLOT_CONFIG_ECDH_SHIFT (2)`
- `#define ATCA_SLOT_CONFIG_ECDH_MASK (0x01u << ATCA_SLOT_CONFIG_ECDH_SHIFT)`
- `#define ATCA_SLOT_CONFIG_WRITE_ECDH_SHIFT (3)`
- `#define ATCA_SLOT_CONFIG_WRITE_ECDH_MASK (0x01u << ATCA_SLOT_CONFIG_WRITE_↵ECDH_SHIFT)`
- `#define ATCA_SLOT_CONFIG_GEN_KEY_SHIFT (8)`
- `#define ATCA_SLOT_CONFIG_GEN_KEY_MASK (0x01u << ATCA_SLOT_CONFIG_GEN_KEY_SHIFT)`
- `#define ATCA_SLOT_CONFIG_PRIV_WRITE_SHIFT (9)`
- `#define ATCA_SLOT_CONFIG_PRIV_WRITE_MASK (0x01u << ATCA_SLOT_CONFIG_PRIV_WRITE_↵SHIFT)`
- `#define ATCA_USE_LOCK_ENABLE_SHIFT (0)`
- `#define ATCA_USE_LOCK_ENABLE_MASK (0x0Fu << ATCA_USE_LOCK_ENABLE_SHIFT)`
- `#define ATCA_USE_LOCK_KEY_SHIFT (4)`
- `#define ATCA_USE_LOCK_KEY_MASK (0x0Fu << ATCA_USE_LOCK_KEY_SHIFT)`
- `#define ATCA_VOL_KEY_PERM_SLOT_SHIFT (0)`
- `#define ATCA_VOL_KEY_PERM_SLOT_MASK (0x0Fu << ATCA_VOL_KEY_PERM_SLOT_SHIFT)`
- `#define ATCA_VOL_KEY_PERM_SLOT(v) (ATCA_VOL_KEY_PERM_SLOT_MASK & ((v) << ATCA_↵VOL_KEY_PERM_SLOT_SHIFT))`
- `#define ATCA_VOL_KEY_PERM_EN_SHIFT (7)`
- `#define ATCA_VOL_KEY_PERM_EN_MASK (0x01u << ATCA_VOL_KEY_PERM_EN_SHIFT)`
- `#define ATCA_SECURE_BOOT_MODE_SHIFT (0)`
- `#define ATCA_SECURE_BOOT_MODE_MASK (0x03u << ATCA_SECURE_BOOT_MODE_SHIFT)`
- `#define ATCA_SECURE_BOOT_MODE(v) (ATCA_SECURE_BOOT_MODE_MASK & ((v) << ATCA_↵SECURE_BOOT_MODE_SHIFT))`
- `#define ATCA_SECURE_BOOT_PERSIST_EN_SHIFT (3)`
- `#define ATCA_SECURE_BOOT_PERSIST_EN_MASK (0x01u << ATCA_SECURE_BOOT_PERSIST_↵EN_SHIFT)`
- `#define ATCA_SECURE_BOOT_RAND_NONCE_SHIFT (4)`
- `#define ATCA_SECURE_BOOT_RAND_NONCE_MASK (0x01u << ATCA_SECURE_BOOT_RAND_↵NONCE_SHIFT)`
- `#define ATCA_SECURE_BOOT_DIGEST_SHIFT (8)`
- `#define ATCA_SECURE_BOOT_DIGEST_MASK (0x0Fu << ATCA_SECURE_BOOT_DIGEST_SHIFT)`
- `#define ATCA_SECURE_BOOT_DIGEST(v) (ATCA_SECURE_BOOT_DIGEST_MASK & ((v) << ATCA_↵SECURE_BOOT_DIGEST_SHIFT))`
- `#define ATCA_SECURE_BOOT_PUB_KEY_SHIFT (12)`
- `#define ATCA_SECURE_BOOT_PUB_KEY_MASK (0x0Fu << ATCA_SECURE_BOOT_PUB_KEY_↵SHIFT)`

- `#define ATCA_SECURE_BOOT_PUB_KEY(v) (ATCA_SECURE_BOOT_PUB_KEY_MASK & ((v) << ATCA_SECURE_BOOT_PUB_KEY_SHIFT))`
- `#define ATCA_SLOT_LOCKED(v) ((0x01 << (v)) & 0xFFFFu)`
- `#define ATCA_CHIP_OPT_POST_EN_SHIFT (0)`
- `#define ATCA_CHIP_OPT_POST_EN_MASK (0x01u << ATCA_CHIP_OPT_POST_EN_SHIFT)`
- `#define ATCA_CHIP_OPT_IO_PROT_EN_SHIFT (1)`
- `#define ATCA_CHIP_OPT_IO_PROT_EN_MASK (0x01u << ATCA_CHIP_OPT_IO_PROT_EN_SHIFT)`
- `#define ATCA_CHIP_OPT_KDF_AES_EN_SHIFT (2)`
- `#define ATCA_CHIP_OPT_KDF_AES_EN_MASK (0x01u << ATCA_CHIP_OPT_KDF_AES_EN_SHIFT)`
- `#define ATCA_CHIP_OPT_ECDH_PROT_SHIFT (8)`
- `#define ATCA_CHIP_OPT_ECDH_PROT_MASK (0x03u << ATCA_CHIP_OPT_ECDH_PROT_SHIFT)`
- `#define ATCA_CHIP_OPT_ECDH_PROT(v) (ATCA_CHIP_OPT_ECDH_PROT_MASK & ((v) << ATCA_↵CHIP_OPT_ECDH_PROT_SHIFT))`
- `#define ATCA_CHIP_OPT_KDF_PROT_SHIFT (10)`
- `#define ATCA_CHIP_OPT_KDF_PROT_MASK (0x03u << ATCA_CHIP_OPT_KDF_PROT_SHIFT)`
- `#define ATCA_CHIP_OPT_KDF_PROT(v) (ATCA_CHIP_OPT_KDF_PROT_MASK & ((v) << ATCA_↵CHIP_OPT_KDF_PROT_SHIFT))`
- `#define ATCA_CHIP_OPT_IO_PROT_KEY_SHIFT (12)`
- `#define ATCA_CHIP_OPT_IO_PROT_KEY_MASK ((uint16_t)0x0Fu << ATCA_CHIP_OPT_IO_PROT_↵KEY_SHIFT)`
- `#define ATCA_CHIP_OPT_IO_PROT_KEY(v) (ATCA_CHIP_OPT_IO_PROT_KEY_MASK & ((v) << ATCA_CHIP_OPT_IO_PROT_KEY_SHIFT))`
- `#define ATCA_KEY_CONFIG_OFFSET(x) (96UL + (x) * 2u)`
- `#define ATCA_KEY_CONFIG_PRIVATE_SHIFT (0)`
- `#define ATCA_KEY_CONFIG_PRIVATE_MASK (0x01u << ATCA_KEY_CONFIG_PRIVATE_SHIFT)`
- `#define ATCA_KEY_CONFIG_PUB_INFO_SHIFT (1)`
- `#define ATCA_KEY_CONFIG_PUB_INFO_MASK (0x01u << ATCA_KEY_CONFIG_PUB_INFO_SHIFT)`
- `#define ATCA_KEY_CONFIG_KEY_TYPE_SHIFT (2)`
- `#define ATCA_KEY_CONFIG_KEY_TYPE_MASK ((0x07u << ATCA_KEY_CONFIG_KEY_TYPE_↵SHIFT))`
- `#define ATCA_KEY_CONFIG_KEY_TYPE(v) ((ATCA_KEY_CONFIG_KEY_TYPE_MASK & ((v) << ATCA_KEY_CONFIG_KEY_TYPE_SHIFT)))`
- `#define ATCA_KEY_CONFIG_LOCKABLE_SHIFT (5)`
- `#define ATCA_KEY_CONFIG_LOCKABLE_MASK (0x01u << ATCA_KEY_CONFIG_LOCKABLE_SHIFT)`
- `#define ATCA_KEY_CONFIG_REQ_RANDOM_SHIFT (6)`
- `#define ATCA_KEY_CONFIG_REQ_RANDOM_MASK (0x01u << ATCA_KEY_CONFIG_REQ_↵RANDOM_SHIFT)`
- `#define ATCA_KEY_CONFIG_REQ_AUTH_SHIFT (7)`
- `#define ATCA_KEY_CONFIG_REQ_AUTH_MASK (0x01u << ATCA_KEY_CONFIG_REQ_AUTH_SHIFT)`
- `#define ATCA_KEY_CONFIG_AUTH_KEY_SHIFT (8)`
- `#define ATCA_KEY_CONFIG_AUTH_KEY_MASK (0x0Fu << ATCA_KEY_CONFIG_AUTH_KEY_SHIFT)`
- `#define ATCA_KEY_CONFIG_AUTH_KEY(v) (ATCA_KEY_CONFIG_AUTH_KEY_MASK & ((v) << ATCA_KEY_CONFIG_AUTH_KEY_SHIFT))`
- `#define ATCA_KEY_CONFIG_PERSIST_DIS_SHIFT (12)`
- `#define ATCA_KEY_CONFIG_PERSIST_DIS_MASK (0x01u << ATCA_KEY_CONFIG_PERSIST_DIS_↵SHIFT)`
- `#define ATCA_KEY_CONFIG_RFU_SHIFT (13)`
- `#define ATCA_KEY_CONFIG_RFU_MASK (0x01u << ATCA_KEY_CONFIG_RFU_SHIFT)`
- `#define ATCA_KEY_CONFIG_X509_ID_SHIFT (14)`
- `#define ATCA_KEY_CONFIG_X509_ID_MASK (0x03u << ATCA_KEY_CONFIG_X509_ID_SHIFT)`
- `#define ATCA_KEY_CONFIG_X509_ID(v) (ATCA_KEY_CONFIG_X509_ID_MASK & ((v) << ATCA_↵KEY_CONFIG_X509_ID_SHIFT))`

Typedefs

- typedef struct [atca_sha256_ctx](#) [atca_sha256_ctx_t](#)
- typedef [atca_sha256_ctx_t](#) [atca_hmac_sha256_ctx_t](#)
- typedef struct ATCA_PACKED [atsha204a_config_s](#) [atsha204a_config_t](#)
- typedef struct ATCA_PACKED [atecc508a_config_s](#) [atecc508a_config_t](#)
- typedef struct ATCA_PACKED [atecc608_config_s](#) [atecc608_config_t](#)

Functions

- ATCA_STATUS [calib_wakeup_i2c](#) (ATCADevice device)
basic API methods are all prefixed with atcab_ (CryptoAuthLib Basic) the fundamental premise of the basic API is it is based on a single interface instance and that instance is global, so all basic API commands assume that one global device is the one to operate on.
- ATCA_STATUS [calib_wakeup](#) (ATCADevice device)
wakeup the CryptoAuth device
- ATCA_STATUS [calib_idle](#) (ATCADevice device)
idle the CryptoAuth device
- ATCA_STATUS [calib_sleep](#) (ATCADevice device)
invoke sleep on the CryptoAuth device
- ATCA_STATUS [calib_exit](#) (ATCADevice device)
common cleanup code which idles the device after any operation
- ATCA_STATUS [calib_get_addr](#) (uint8_t zone, uint16_t slot, uint8_t block, uint8_t offset, uint16_t *addr)
Compute the address given the zone, slot, block, and offset.
- ATCA_STATUS [calib_get_zone_size](#) (ATCADevice device, uint8_t zone, uint16_t slot, size_t *size)
Gets the size of the specified zone in bytes.
- ATCA_STATUS [calib_is_locked](#) (ATCADevice device, uint8_t zone, bool *is_locked)
- ATCA_STATUS [calib_is_slot_locked](#) (ATCADevice device, uint16_t slot, bool *is_locked)
- ATCA_STATUS [calib_ca2_get_addr](#) (uint8_t zone, uint16_t slot, uint8_t block, uint8_t offset, uint16_t *addr)
Compute the address given the zone, slot, block, and offset for the device.
- ATCA_STATUS [calib_ca2_is_locked](#) (ATCADevice device, uint8_t zone, bool *is_locked)
Use Info command to check config/data is locked or not.
- ATCA_STATUS [calib_ca2_is_data_locked](#) (ATCADevice device, bool *is_locked)
Use Info command to check ECC204 Data zone lock status.
- ATCA_STATUS [calib_ca2_is_config_locked](#) (ATCADevice device, bool *is_locked)
Executes Read command, which reads the configuration zone to see if the specified slot is locked.
- ATCADeviceType [calib_get_devicetype_with_device_id](#) (uint8_t device_id, uint8_t device_revision)
- ATCADeviceType [calib_get_devicetype](#) (uint8_t revision[4])
Parse the revision field to get the device type.
- ATCA_STATUS [calib_info_base](#) (ATCADevice device, uint8_t mode, uint16_t param2, uint8_t *out_data)
Issues an Info command, which return internal device information and can control GPIO and the persistent latch.
- ATCA_STATUS [calib_info](#) (ATCADevice device, uint8_t *revision)
Use the Info command to get the device revision (DevRev).
- ATCA_STATUS [calib_info_privkey_valid](#) (ATCADevice device, uint16_t key_id, uint8_t *is_valid)
Use Info command to check ECC Private key stored in key slot is valid or not.
- ATCA_STATUS [calib_info_lock_status](#) (ATCADevice device, uint16_t param2, uint8_t *is_locked)
Use Info command to ECC204,TA010 config/data zone lock status.
- ATCA_STATUS [calib_info_chip_status](#) (ATCADevice device, uint8_t *chip_status)
Use Info command to get ECC204,TA010,SHA10x chip status.

21.7.1 Detailed Description

These methods provide a simple API to CryptoAuth chips.

21.7.2 Function Documentation

21.7.2.1 calib_ca2_get_addr()

```
ATCA_STATUS calib_ca2_get_addr (
    uint8_t zone,
    uint16_t slot,
    uint8_t block,
    uint8_t offset,
    uint16_t * addr )
```

Compute the address given the zone, slot, block, and offset for the device.

Parameters

in	<i>zone</i>	Zone to get address from. Config(1) or Data(0) which requires a slot.
in	<i>slot</i>	Slot Id number for data zone and zero for other zones.
in	<i>block</i>	Block number within the data zone .
in	<i>offset</i>	Always zero.
out	<i>addr</i>	Pointer to the address of data or configuration zone.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.7.2.2 calib_ca2_is_config_locked()

```
ATCA_STATUS calib_ca2_is_config_locked (
    ATCADevice device,
    bool * is_locked )
```

Executes Read command, which reads the configuration zone to see if the specified slot is locked.

Parameters

in	<i>device</i>	Device context pointer
in	<i>slot</i>	Slot to query for locked (slot 0-15)
out	<i>is_locked</i>	Lock state returned here. True if locked.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Use Info command to check ECC204 Config zone lock status

Parameters

in	<i>device</i>	Device context pointer
out	<i>is_locked</i>	return lock status

Returns

ATCA_SUCCESS on success, otherwise an error code

21.7.2.3 **calib_ca2_is_data_locked()**

```
ATCA_STATUS calib_ca2_is_data_locked (
    ATCADevice device,
    bool * is_locked )
```

Use Info command to check ECC204 Data zone lock status.

Parameters

in	<i>device</i>	Device context pointer
out	<i>is_locked</i>	return lock status

Returns

ATCA_SUCCESS on success, otherwise an error code

21.7.2.4 **calib_ca2_is_locked()**

```
ATCA_STATUS calib_ca2_is_locked (
    ATCADevice device,
    uint8_t zone,
    bool * is_locked )
```

Use Info command to check config/data is locked or not.

Parameters

in	<i>device</i>	Device context pointer
in	<i>zone</i>	Config/Data zone
out	<i>is_locked</i>	return lock status here

Returns

ATCA_SUCCESS on success, otherwise an error code

21.7.2.5 calib_exit()

```
ATCA_STATUS calib_exit (
    ATCADevice device )
```

common cleanup code which idles the device after any operation

Parameters

in	<i>device</i>	Device context pointer
----	---------------	------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.7.2.6 calib_get_addr()

```
ATCA_STATUS calib_get_addr (
    uint8_t zone,
    uint16_t slot,
    uint8_t block,
    uint8_t offset,
    uint16_t * addr )
```

Compute the address given the zone, slot, block, and offset.

Parameters

in	<i>zone</i>	Zone to get address from. Config(0), OTP(1), or Data(2) which requires a slot.
in	<i>slot</i>	Slot Id number for data zone and zero for other zones.
in	<i>block</i>	Block number within the data or configuration or OTP zone .
in	<i>offset</i>	Offset Number within the block of data or configuration or OTP zone.
out	<i>addr</i>	Pointer to the address of data or configuration or OTP zone.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.7.2.7 calib_get_zone_size()

```
ATCA_STATUS calib_get_zone_size (
    ATCADevice device,
    uint8_t zone,
    uint16_t slot,
    size_t * size )
```

Gets the size of the specified zone in bytes.

Parameters

in	<i>device</i>	Device context pointer
in	<i>zone</i>	Zone to get size information from. Config(0), OTP(1), or Data(2) which requires a slot.
in	<i>slot</i>	If zone is Data(2), the slot to query for size.
out	<i>size</i>	Zone size is returned here.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.7.2.8 calib_idle()

```
ATCA_STATUS calib_idle (
    ATCADevice device )
```

idle the CryptoAuth device

Parameters

in	<i>device</i>	Device context pointer
----	---------------	------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.7.2.9 calib_info()

```
ATCA_STATUS calib_info (
    ATCADevice device,
    uint8_t * revision )
```

Use the Info command to get the device revision (DevRev).

Parameters

in	<i>device</i>	Device context pointer
out	<i>revision</i>	Device revision is returned here (4 bytes).

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.7.2.10 calib_info_base()

```
ATCA_STATUS calib_info_base (
    ATCADevice device,
    uint8_t mode,
    uint16_t param2,
    uint8_t * out_data )
```

Issues an Info command, which return internal device information and can control GPIO and the persistent latch.

Parameters

in	<i>device</i>	Device context pointer
in	<i>mode</i>	Selects which mode to be used for info command.
in	<i>param2</i>	Selects the particular fields for the mode.
out	<i>out_data</i>	Response from info command (4 bytes). Can be set to NULL if not required.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.7.2.11 calib_info_chip_status()

```
ATCA_STATUS calib_info_chip_status (
    ATCADevice device,
    uint8_t * chip_status )
```

Use Info command to get ECC204,TA010,SHA10x chip status.

Parameters

in	<i>device</i>	Device context pointer
out	<i>chip_status</i>	return chip status here

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.7.2.12 calib_info_lock_status()

```
ATCA_STATUS calib_info_lock_status (
    ATCADevice device,
    uint16_t param2,
    uint8_t * is_locked )
```

Use Info command to ECC204,TA010 config/data zone lock status.

Parameters

in	<i>device</i>	Device context pointer
in	<i>param2</i>	selects the zone and slot
out	<i>is_locked</i>	return lock status here

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.7.2.13 calib_info_privkey_valid()

```
ATCA_STATUS calib_info_privkey_valid (
    ATCADevice device,
    uint16_t key_id,
    uint8_t * is_valid )
```

Use Info command to check ECC Private key stored in key slot is valid or not.

Parameters

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	ECC private key slot id For ECC204,TA010 key_id is 0x00
out	<i>is_valid</i>	return private key is valid or invalid

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.7.2.14 calib_sleep()

```
ATCA_STATUS calib_sleep (
    ATCADevice device )
```

invoke sleep on the CryptoAuth device

Parameters

in	<i>device</i>	Device context pointer
----	---------------	------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.7.2.15 calib_wakeup()

```
ATCA_STATUS calib_wakeup (
    ATCADevice device )
```

wakeup the CryptoAuth device

Parameters

in	<i>device</i>	Device context pointer
----	---------------	------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.7.2.16 calib_wakeup_i2c()

```
ATCA_STATUS calib_wakeup_i2c (
    ATCADevice device )
```

basic API methods are all prefixed with atcab_ (CryptoAuthLib Basic) the fundamental premise of the basic API is it is based on a single interface instance and that instance is global, so all basic API commands assume that one global device is the one to operate on.

Drive the SDA pin low for wake up Set i2c device addr as 0U to drive SDA low

I2C general call should not interpreted as an addr write

Set the i2c device address

21.8 Software crypto methods (atcac_)

These methods provide a software implementation of various crypto algorithms.

21.8.0.1 crypto directory - Purpose

This directory contains software implementations of cryptographic functions. The functions at the base level are wrappers that will point to the final implementations of the software crypto functions.

Functions

- ATCA_STATUS **atcac_sw_sha1** (const uint8_t *data, size_t data_size, uint8_t digest[(20U)])
- ATCA_STATUS **atcac_sha256_hmac_ctr_iteration** (struct [atcac_hmac_ctx](#) *ctx, uint8_t iteration, uint16_t length, const uint8_t *label, size_t label_len, const uint8_t *data, size_t data_len, uint8_t digest[(32U)])
- ATCA_STATUS **atcac_sha256_hmac_counter** (uint8_t *key, size_t key_len, const uint8_t *label, size_t label_len, const uint8_t *data, size_t data_len, uint8_t *digest, size_t diglen)

21.8.1 Detailed Description

These methods provide a software implementation of various crypto algorithms.

21.9 Hardware abstraction layer (hal_)

These methods define the hardware abstraction layer for communicating with a CryptoAuth device.

21.9.0.1 HAL Directory - Purpose

This directory contains all the Hardware Abstraction Layer (HAL) files used to adapt the upper levels of atca-ng and abstractions to physical hardware.

HAL contains physical implementations for I2C, SWI, SPI, UART and timers for specific hardware platforms.

Include just those HAL files you require based on platform type.

21.9.1 Cryptoauthlib HAL Architecture

Cryptoauthlib has several intermediate conceptual layers

1. The highest layer of cryptoauthlib (outside of integration APIS) that may be used with an application is the `atcab_` api functions. These are general purpose functions that present a simple and consistent crypto interface to the application regardless of the device being used.
2. `calib_`, `talib_` APIs are the library functions behind `atcab_` ones that generate the correct command packets and process the received responses. Device specific logic is handled by the library here
3. `hal_` these functions perform the transmit/recieve of data for a given interface. These are split into sublayers
 - The HAL layer is the first hal layer that presents the interface expected by the higher level library. When using a native driver and no further interpretation is required this layer is all that is required.
 - The PHY layer if for hals that perform an interpretation or additional protocol logic. In this situation the HAL performs protocol interpretation while the phy performs the physical communication

21.9 Hardware abstraction layer (hal_)

21.9.1.0.1 HAL and PHY Requirements The hal and phy layers have the same construction. A hal or phy must have the following functions and their signatures

- `ATCA_STATUS hal_<name>init(ATCAIface iface, ATCAIfaceCfg *cfg);`
- `ATCA_STATUS hal_<name>post_init(ATCAIface iface);`
- `ATCA_STATUS hal_<name>send(ATCAIface iface, uint8_t address, uint8_t *txdata, int txlength);`
- `ATCA_STATUS hal_<name>receive(ATCAIface iface, uint8_t address, uint8_t *rxdata, uint16_t *rxlength);`
- `ATCA_STATUS hal_<name>control(ATCAIface iface, uint8_t option, void* param, size_t paramlen);`
- `ATCA_STATUS hal_<name>_release(void *hal_data);`

If the hal is a native driver no phy is required. See the tables below for which hal is required to be ported based on a configured interface

21.9.2 CryptoAuthLib Supported HAL Layers

Device Interface	Physical Interface	HAL	PHY
i2c	i2c	hal_i2c	
	gpio	hal_i2c_gpio	hal_gpio
spi	spi	hal_spi	
swi	uart	hal_swi	hal_uart
	gpio	hal_swi_gpio	hal_gpio
any	uart	kit	hal_uart
	hid	kit	hal_hid
	any (user provided)	kit_bridge	

21.9.2.1 Microchip Harmony 3 for all PIC32 & ARM products - Use the Harmony 3 Configurator to generate and configure projects

Obtain library and configure using [Harmony 3](#)

Interface	Files	API	Notes
I2C	hal_i2c_harmony.c	plib.↔ h	For all Harmony 3 based projects
SPI	hal_spi_harmony.c	plib.↔ h	
UART	hal_uart_harmony.c	plib.↔ h	

21.9.2.2 Microchip 8 & 16 bit products - AVR, PIC16/18, PIC24/DSPIC

Obtain library and integration through [Microchip Code Configurator](#)

21.9.2.3 OS & RTOS integrations

Use [CMake](#) to configure the library in Linux, Windows, and MacOS environments

OS	Interface	Files	API	Notes
Linux	I2C	hal_linux_i2c_userspace.c/h	i2c-dev	
Linux	SPI	hal_linux_spi_userspace.c/h	spidev	
Linux/Mac		hal_linux.c		For all Linux/Mac projects
Windows		hal_windows.c		For all Windows projects
All	kit-hid	hal_all_platforms_kit_hidapi.c/h	hidapi	Works for Windows, Linux, and Mac
freeRTOS		hal_freertos.c		freeRTOS common routines

21.9.2.4 Legacy Support - [Atmel START](https://www.microchip.com/start) for AVR, ARM based processors (SAM)

Interface	Files	API	Notes
	hal_timer_start.c	START	Timer implementation
I2C	hal_i2c_start.c/h	START	
SWI	swi_uart_start.c/h	START	SWI using UART

21.9.2.5 Legacy Support - ASF3 for ARM Cortex-m0 & Cortex-m based processors (SAM)

SAM Micros	Interface	Files	API	Notes
cortex-m0	I2C	hal_sam0_i2c_asf.c/h	ASF3	SAMD21, SAMB11, etc
cortex-m3/4/7	I2C	hal_sam_i2c_asf.c/h	ASF3	SAM4S, SAMG55, SAMV71, etc
all		hal_sam_timer_asf.c	ASF3	Common timer hal for all platforms

Data Structures

- struct [atca_hal_kit_phy_t](#)
- struct [atca_hal_shm_t](#)
- struct [i2c_start_instance](#)
- struct [atca_i2c_host_s](#)
- struct [i2c_sam_instance](#)
- struct [atcal2Cmaster](#)

this is the hal_data for ATCA HAL for ASF SERCOM

- struct [atcaSWImaster](#)

this is the hal_data for ATCA HAL for ASF SERCOM

Macros

- #define **ATCA_POLLING_INIT_TIME_MSEC** 1
- #define **ATCA_POLLING_FREQUENCY_TIME_MSEC** 2
- #define **ATCA_POLLING_MAX_TIME_MSEC** 2500
- #define **ATCA_HAL_CONTROL_WAKE** (0U)
Execute the hardware specific wake - generally only for kits.
- #define **ATCA_HAL_CONTROL_IDLE** (1U)
Execute the hardware specific idle - generally only for kits.
- #define **ATCA_HAL_CONTROL_SLEEP** (2U)
Execute the hardware specific sleep - generally only for kits.
- #define **ATCA_HAL_CONTROL_RESET** (3U)

- *Execute the hardware specific reset - generally only for kits.*
- #define **ATCA_HAL_CONTROL_SELECT** (4U)
Select the device - assert CS, open device, etc.
- #define **ATCA_HAL_CONTROL_DESELECT** (5U)
Select the device - de-assert CS, release device, etc.
- #define **ATCA_HAL_CHANGE_BAUD** (6U)
Change the datarate of the phy.
- #define **ATCA_HAL_FLUSH_BUFFER** (7U)
If the phy has a buffer make sure all bytes are transmitted.
- #define **ATCA_HAL_CONTROL_DIRECTION** (8U)
Set the PIN mode (in vs out)
- #define **MAX_I2C_BUSES** 3
- #define **KIT_MAX_SCAN_COUNT** 8
- #define **KIT_MAX_TX_BUF** 32
- #define **KIT_TA_WORDADD_SEND_LEN** 20
- #define **KIT_TX_WRAP_SIZE** (10)
- #define **KIT_MSG_SIZE** (32u)
- #define **KIT_RX_WRAP_SIZE** (KIT_MSG_SIZE + 6u)
- #define **MAX_SWI_BUSES** 6
- #define **RECEIVE_MODE** 0
- #define **TRANSMIT_MODE** 1
- #define **RX_DELAY** 10
- #define **TX_DELAY** 90
- #define **DEBUG_PIN_1** EXT2_PIN_5
- #define **DEBUG_PIN_2** EXT2_PIN_6
- #define **MAX_SWI_BUSES** 6
- #define **RECEIVE_MODE** 0
- #define **TRANSMIT_MODE** 1
- #define **RX_DELAY** 10
- #define **TX_DELAY** 93

Typedefs

- typedef void * **hal_mutex_t**
Generic mutex type definition for most systems.
- typedef void(* **start_change_baudrate**) ([ATCAIface](#) iface, uint32_t speed)
- typedef struct [i2c_start_instance](#) **i2c_start_instance_t**
- typedef struct [atca_i2c_host_s](#) **atca_i2c_host_t**
- typedef void(* **sam_change_baudrate**) ([ATCAIface](#) iface, uint32_t speed)
- typedef struct [i2c_sam_instance](#) **i2c_sam_instance_t**
- typedef struct [atcal2Cmaster](#) **ATCAI2CMaster_t**
this is the hal_data for ATCA HAL for ASF SERCOM
- typedef struct [atcaSWImaster](#) **ATCASWIMaster_t**
this is the hal_data for ATCA HAL for ASF SERCOM
- typedef struct [atcaSWImaster](#) **ATCASWIMaster_t**
this is the hal_data for ATCA HAL for ASF SERCOM

Functions

- ATCA_STATUS [hal_iface_init](#) (ATCAIfaceCfg *cfg, ATCAHAL_t **hal, ATCAHAL_t **phy)
Standard HAL API for ATCA to initialize a physical interface.
- ATCA_STATUS [hal_iface_release](#) (ATCAIfaceType iface_type, void *hal_data)
releases a physical interface, HAL knows how to interpret hal_data
- ATCA_STATUS [hal_check_wake](#) (const uint8_t *response, int response_size)
Utility function for hal_wake to check the reply.
- void [atca_delay_ms](#) (uint32_t ms)
Timer API for legacy implementations.
- void [atca_delay_us](#) (uint32_t delay)
This function delays for a number of microseconds.
- void [hal_delay_ms](#) (uint32_t delay)
Timer API implemented at the HAL level.
- void [hal_delay_us](#) (uint32_t delay)
This function delays for a number of microseconds.
- ATCA_STATUS [hal_create_mutex](#) (void **ppMutex, const char *pName)
Optional hal interfaces.
- ATCA_STATUS [hal_init_mutex](#) (void *pMutex, bool shared)
- ATCA_STATUS [hal_destroy_mutex](#) (void *pMutex)
- ATCA_STATUS [hal_lock_mutex](#) (void *pMutex)
- ATCA_STATUS [hal_unlock_mutex](#) (void *pMutex)
- ATCA_STATUS [hal_alloc_shared](#) (void **pShared, size_t size, const char *pName, bool *initialized)
- ATCA_STATUS [hal_free_shared](#) (void *pShared, size_t size)
- ATCA_STATUS [hal_iface_register_hal](#) (ATCAIfaceType iface_type, ATCAHAL_t *hal, ATCAHAL_t **old_hal, ATCAHAL_t *phy, ATCAHAL_t **old_phy)
Register/Replace a HAL with a.
- uint8_t [hal_is_command_word](#) (uint8_t word_address)
Utility function for hal_wake to check the reply.
- ATCA_STATUS [hal_kit_hid_init](#) (ATCAIface iface, ATCAIfaceCfg *cfg)
HAL implementation of Kit USB HID init.
- ATCA_STATUS [hal_kit_hid_post_init](#) (ATCAIface iface)
HAL implementation of Kit HID post init.
- ATCA_STATUS [hal_kit_hid_send](#) (ATCAIface iface, uint8_t word_address, uint8_t *txdata, int txlength)
HAL implementation of kit protocol send over USB HID.
- ATCA_STATUS [hal_kit_hid_receive](#) (ATCAIface iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxlength)
HAL implementation of send over USB HID.
- ATCA_STATUS [hal_kit_hid_control](#) (ATCAIface iface, uint8_t option, void *param, size_t paramlen)
Perform control operations for the kit protocol.
- ATCA_STATUS [hal_kit_hid_release](#) (void *hal_data)
Close the physical port for HID.
- void * [hal_malloc](#) (size_t size)
- void [hal_free](#) (void *ptr)
- void [hal_rtos_delay_ms](#) (uint32_t delay)
This function delays for a number of milliseconds.
- ATCA_STATUS [hal_i2c_discover_buses](#) (int i2c_buses[], int max_buses)
discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge
- ATCA_STATUS [hal_i2c_discover_devices](#) (int bus_num, ATCAIfaceCfg cfg[], int *found)
discover any CryptoAuth devices on a given logical bus number
- ATCA_STATUS [hal_i2c_init](#) (ATCAIface iface, ATCAIfaceCfg *cfg)

hal_i2c_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIFace instances using the same bus, and you can have multiple ATCAIFace instances on multiple i2c buses, so *hal_i2c_init* manages these things and ATCAIFace is abstracted from the physical details.

- ATCA_STATUS [hal_i2c_post_init](#) (ATCAIFace iface)
HAL implementation of I2C post init.
- ATCA_STATUS [hal_i2c_send](#) (ATCAIFace iface, uint8_t word_address, uint8_t *txdata, int txlength)
HAL implementation of I2C send over START.
- ATCA_STATUS [hal_i2c_receive](#) (ATCAIFace iface, uint8_t address, uint8_t *rxdata, uint16_t *rxlength)
HAL implementation of I2C receive function for START I2C.
- ATCA_STATUS [change_i2c_speed](#) (ATCAIFace iface, uint32_t speed)
method to change the bus speed of I2C
- ATCA_STATUS [hal_i2c_control](#) (ATCAIFace iface, uint8_t option, void *param, size_t paramlen)
Perform control operations for the kit protocol.
- ATCA_STATUS [hal_i2c_release](#) (void *hal_data)
manages reference count on given bus and releases resource if no more references exist
- ATCA_STATUS [hal_i2c_init](#) (void *hal, ATCAIFaceCfg *cfg)
hal_i2c_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIFace instances using the same bus, and you can have multiple ATCAIFace instances on multiple i2c buses, so *hal_i2c_init* manages these things and ATCAIFace is abstracted from the physical details.
- ATCA_STATUS [hal_i2c_wake](#) (ATCAIFace iface)
wake up CryptoAuth device using I2C bus
- ATCA_STATUS [hal_i2c_idle](#) (ATCAIFace iface)
idle CryptoAuth device using I2C bus
- ATCA_STATUS [hal_i2c_sleep](#) (ATCAIFace iface)
sleep CryptoAuth device using I2C bus
- ATCA_STATUS [hal_kit_attach_phy](#) (ATCAIFaceCfg *cfg, atca_hal_kit_phy_t *phy)
Helper function that connects a physical layer context structure that will be used by the kit protocol bridge.
- ATCA_STATUS [hal_kit_init](#) (ATCAIFace iface, ATCAIFaceCfg *cfg)
HAL implementation of Kit USB HID init.
- ATCA_STATUS [hal_kit_post_init](#) (ATCAIFace iface)
HAL implementation of Kit HID post init.
- ATCA_STATUS [hal_kit_send](#) (ATCAIFace iface, uint8_t word_address, uint8_t *txdata, int txlength)
HAL implementation of kit protocol send over USB HID.
- ATCA_STATUS [hal_kit_receive](#) (ATCAIFace iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxsize)
HAL implementation of send over USB HID.
- ATCA_STATUS [hal_kit_control](#) (ATCAIFace iface, uint8_t option, void *param, size_t paramlen)
Kit Protocol Control.
- ATCA_STATUS [hal_kit_release](#) (void *hal_data)
Close the physical port for HID.
- ATCA_STATUS [hal_check_pid](#) (hal_pid_t pid)
Check if the pid exists in the system.
- void [atca_delay_10us](#) (uint32_t delay)
This function delays for a number of tens of microseconds.
- ATCA_STATUS [hal_spi_discover_buses](#) (int spi_buses[], int max_buses)
discover spi buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge
- ATCA_STATUS [hal_spi_discover_devices](#) (int bus_num, ATCAIFaceCfg cfg[], int *found)
discover any TA10x devices on a given logical bus number
- ATCA_STATUS [hal_spi_init](#) (ATCAIFace iface, ATCAIFaceCfg *cfg)
initialize an SPI interface using given config

- ATCA_STATUS [hal_spi_post_init](#) (ATCAIface iface)
HAL implementation of SPI post init.
- ATCA_STATUS [hal_spi_select](#) (ATCAIface iface)
HAL implementation to assert the device chip select.
- ATCA_STATUS [hal_spi_deselect](#) (ATCAIface iface)
HAL implementation to deassert the device chip select.
- ATCA_STATUS [hal_spi_send](#) (ATCAIface iface, uint8_t word_address, uint8_t *txdata, int txlength)
HAL implementation of SPI send over Harmony.
- ATCA_STATUS [hal_spi_receive](#) (ATCAIface iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxlength)
HAL implementation of SPI receive function for HARMONY SPI.
- ATCA_STATUS [hal_spi_control](#) (ATCAIface iface, uint8_t option, void *param, size_t paramlen)
Perform control operations for the kit protocol.
- ATCA_STATUS [hal_spi_release](#) (void *hal_data)
manages reference count on given bus and releases resource if no more references exist
- ATCA_STATUS [hal_swi_init](#) (ATCAIface iface, ATCAIfaceCfg *cfg)
initialize an SWI interface using given config
- ATCA_STATUS [hal_swi_post_init](#) (ATCAIface iface)
HAL implementation of SWI post init.
- ATCA_STATUS [hal_swi_send](#) (ATCAIface iface, uint8_t word_address, uint8_t *txdata, int txlength)
HAL implementation of SWI send command over UART.
- ATCA_STATUS [hal_swi_receive](#) (ATCAIface iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxlength)
HAL implementation of SWI receive function over UART.
- ATCA_STATUS [hal_swi_wake](#) (ATCAIface iface)
Send Wake flag via SWI.
- ATCA_STATUS [hal_swi_sleep](#) (ATCAIface iface)
Send Sleep flag via SWI.
- ATCA_STATUS [hal_swi_idle](#) (ATCAIface iface)
Send Idle flag via SWI.
- ATCA_STATUS [hal_swi_control](#) (ATCAIface iface, uint8_t option, void *param, size_t paramlen)
Perform control operations for the kit protocol.
- ATCA_STATUS [hal_swi_release](#) (void *hal_data)
manages reference count on given bus and releases resource if no more references exist
- const char * [kit_id_from_devtype](#) (ATCADeviceType devtype)
- const char * [kit_interface_from_kittype](#) (ATCAKitType kittype)
- const char * [kit_interface](#) (ATCAKitType kittype)
- ATCA_STATUS [kit_init](#) (ATCAIface iface, ATCAIfaceCfg *cfg)
- ATCA_STATUS [kit_post_init](#) (ATCAIface iface)
- ATCA_STATUS [kit_send](#) (ATCAIface iface, uint8_t word_address, uint8_t *txdata, int txlength)
- ATCA_STATUS [kit_receive](#) (ATCAIface iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxsize)
- ATCA_STATUS [kit_control](#) (ATCAIface iface, uint8_t option, void *param, size_t paramlen)
- ATCA_STATUS [kit_release](#) (void *hal_data)
- ATCA_STATUS [kit_wrap_cmd](#) (ATCAIface iface, uint8_t word_address, const uint8_t *txdata, int txlen, char *pkitcmd, int *nkitcmd)
- ATCA_STATUS [kit_parse_rsp](#) (const char *pkitbuf, int nkitbuf, uint8_t *kitstatus, uint8_t *rxdata, int *datasize)
- ATCA_STATUS [kit_wake](#) (ATCAIface iface)
- ATCA_STATUS [kit_idle](#) (ATCAIface iface)
- ATCA_STATUS [kit_sleep](#) (ATCAIface iface)
- ATCA_STATUS [kit_phy_send](#) (ATCAIface iface, uint8_t *txdata, int txlength)
- ATCA_STATUS [kit_phy_receive](#) (ATCAIface iface, uint8_t *rxdata, int *rxsize)
- ATCA_STATUS [swi_uart_init](#) (ATCASWIMaster_t *instance)
Implementation of SWI UART init.

- ATCA_STATUS [swi_uart_deinit](#) ([ATCASWIMaster_t](#) *instance)
Implementation of SWI UART deinit.
- void [swi_uart_setbaud](#) ([ATCASWIMaster_t](#) *instance, uint32_t baudrate)
implementation of SWI UART change baudrate.
- void [swi_uart_mode](#) ([ATCASWIMaster_t](#) *instance, uint8_t mode)
implementation of SWI UART change mode.
- void [swi_uart_discover_buses](#) (int swi_uart_buses[], int max_buses)
discover UART buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge
- ATCA_STATUS [swi_uart_send_byte](#) ([ATCASWIMaster_t](#) *instance, uint8_t data)
HAL implementation of SWI UART send byte over ASF. This function send one byte over UART.
- ATCA_STATUS [swi_uart_receive_byte](#) ([ATCASWIMaster_t](#) *instance, uint8_t *data)
HAL implementation of SWI UART receive bytes over ASF. This function receive one byte over UART.

Variables

- struct port_config **pin_conf**

21.9.3 Detailed Description

These methods define the hardware abstraction layer for communicating with a CryptoAuth device.

These methods define the hardware abstraction layer for communicating with a CryptoAuth device using SWI Interface.

These methods define the hardware abstraction layer for communicating with a TA10x device.

< Uncomment when debugging

These methods define the hardware abstraction layer for communicating with a CryptoAuth device using I2C driver of ASF.

21.9.4 Macro Definition Documentation

21.9.4.1 MAX_SWI_BUSES [1/2]

```
#define MAX_SWI_BUSES 6
```

- this HAL implementation assumes you've included the ASF SERCOM UART libraries in your project, otherwise, the HAL layer will not compile because the ASF UART drivers are a dependency *

21.9.4.2 MAX_SWI_BUSES [2/2]

```
#define MAX_SWI_BUSES 6
```

- this HAL implementation assumes you've included the ASF SERCOM UART libraries in your project, otherwise, the HAL layer will not compile because the ASF UART drivers are a dependency *

21.9.5 Function Documentation

21.9.5.1 atca_delay_10us()

```
void atca_delay_10us (
    uint32_t delay )
```

This function delays for a number of tens of microseconds.

Parameters

in	<i>delay</i>	number of 0.01 milliseconds to delay
----	--------------	--------------------------------------

Parameters

in	<i>delay</i>	number of 0.01 milliseconds to delay
----	--------------	--------------------------------------

21.9.5.2 atca_delay_ms()

```
void atca_delay_ms (
    uint32_t delay )
```

Timer API for legacy implementations.

This function delays for a number of milliseconds.

```
You can override this function if you like to do
something else in your system while delaying.
```

Parameters

in	<i>delay</i>	number of milliseconds to delay
----	--------------	---------------------------------

```
You can override this function if you like to do
something else in your system while delaying.
```

Parameters

in	<i>delay</i>	number of milliseconds to delay
----	--------------	---------------------------------

21.9.5.3 atca_delay_us()

```
void atca_delay_us (
    uint32_t delay )
```

This function delays for a number of microseconds.

Parameters

in	<i>delay</i>	number of 0.001 milliseconds to delay
----	--------------	---------------------------------------

Parameters

in	<i>delay</i>	number of microseconds to delay
----	--------------	---------------------------------

Parameters

in	<i>delay</i>	number of 0.001 milliseconds to delay
----	--------------	---------------------------------------

21.9.5.4 change_i2c_speed()

```
ATCA_STATUS change_i2c_speed (
    ATCAIface iface,
    uint32_t speed )
```

method to change the bus speed of I2C

method to change the bus speed of I2C

Parameters

in	<i>iface</i>	interface on which to change bus speed
in	<i>speed</i>	baud rate (typically 100000 or 400000)
in	<i>iface</i>	interface on which to change bus speed
in	<i>speed</i>	baud rate (typically 100000 or 400000)

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.5 hal_check_wake()

```
ATCA_STATUS hal_check_wake (
    const uint8_t * response,
    int response_size )
```

Utility function for hal_wake to check the reply.

Parameters

in	<i>response</i>	Wake response to be checked.
in	<i>response_size</i>	Size of the response to check.

Returns

ATCA_SUCCESS for expected wake, ATCA_STATUS_SELFTEST_ERROR if the power on self test failed, ATCA_WAKE_FAILED for other failures.

21.9.5.6 hal_create_mutex()

```
ATCA_STATUS hal_create_mutex (
    void ** ppMutex,
    const char * pName )
```

Optional hal interfaces.

Application callback for creating a mutex object.

Parameters

in, out	<i>ppMutex</i>	location to receive ptr to mutex
in, out	<i>pName</i>	String used to identify the mutex
	<i>[IN/OUT]</i>	ppMutex location to receive ptr to mutex
	<i>[IN]</i>	pName Name of the mutex for systems using named objects

21.9.5.7 hal_delay_ms()

```
void hal_delay_ms (
    uint32_t delay )
```

Timer API implemented at the HAL level.

This function delays for a number of milliseconds.

21.9 Hardware abstraction layer (hal_)

Parameters

in	<i>delay</i>	number of milliseconds to delay
----	--------------	---------------------------------

You can override this function if you like to do something else in your system while delaying.

Parameters

in	<i>delay</i>	number of milliseconds to delay
----	--------------	---------------------------------

21.9.5.8 hal_delay_us()

```
void hal_delay_us (
    uint32_t delay )
```

This function delays for a number of microseconds.

Parameters

in	<i>delay</i>	number of microseconds to delay
----	--------------	---------------------------------

Parameters

in	<i>delay</i>	number of microseconds to delay
----	--------------	---------------------------------

21.9.5.9 hal_i2c_control()

```
ATCA_STATUS hal_i2c_control (
    ATCAIface iface,
    uint8_t option,
    void * param,
    size_t paramlen )
```

Perform control operations for the kit protocol.

Parameters

in	<i>iface</i>	Interface to interact with.
in	<i>option</i>	Control parameter identifier
in	<i>param</i>	Optional pointer to parameter value
in	<i>paramlen</i>	Length of the parameter

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.10 hal_i2c_discover_buses()

```
ATCA_STATUS hal_i2c_discover_buses (
    int i2c_buses[],
    int max_buses )
```

discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge

This HAL implementation assumes you've included the ASF TWI libraries in your project, otherwise, the HAL layer will not compile because the ASF TWI drivers are a dependency.

logical to physical bus mapping structure

discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge

Parameters

in	<i>i2c_buses</i>	- an array of logical bus numbers
in	<i>max_buses</i>	- maximum number of buses the app wants to attempt to discover

Returns

ATCA_SUCCESS

discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge

Parameters

in	<i>i2c_buses</i>	- an array of logical bus numbers
in	<i>max_buses</i>	- maximum number of buses the app wants to attempt to discover

Returns

ATCA_SUCCESS

discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge

Parameters

in	<i>i2c_buses</i>	- an array of logical bus numbers
in	<i>max_buses</i>	- maximum number of buses the app wants to attempt to discover return ATCA_SUCCESS

21.9.5.11 hal_i2c_discover_devices()

```
ATCA_STATUS hal_i2c_discover_devices (
    int bus_num,
    ATCAIfaceCfg cfg[],
    int * found )
```

discover any CryptoAuth devices on a given logical bus number

Parameters

in	<i>bus_num</i>	logical bus number on which to look for CryptoAuth devices
out	<i>cfg</i>	pointer to head of an array of interface config structures which get filled in by this method
out	<i>found</i>	number of devices found on this bus

Returns

ATCA_SUCCESS

Parameters

in	<i>bus_num</i>	- logical bus number on which to look for CryptoAuth devices
out	<i>cfg[]</i>	- pointer to head of an array of interface config structures which get filled in by this method
out	<i>*found</i>	- number of devices found on this bus

Returns

ATCA_SUCCESS

Parameters

in	<i>bus_num</i>	Logical bus number on which to look for CryptoAuth devices
out	<i>cfg</i>	Pointer to head of an array of interface config structures which get filled in by this method
out	<i>found</i>	Number of devices found on this bus

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.12 hal_i2c_idle()

```
ATCA_STATUS hal_i2c_idle (
    ATCAIface iface )
```

idle CryptoAuth device using I2C bus

Parameters

in	<i>iface</i>	interface to logical device to idle
----	--------------	-------------------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>iface</i>	interface to logical device to idle
----	--------------	-------------------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.13 hal_i2c_init() [1/2]

```
ATCA_STATUS hal_i2c_init (
    ATCAIFace iface,
    ATCAIFaceCfg * cfg )
```

hal_i2c_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIFace instances using the same bus, and you can have multiple ATCAIFace instances on multiple i2c buses, so hal_i2c_init manages these things and ATCAIFace is abstracted from the physical details.

HAL implementation of I2C init.

- this HAL implementation assumes you've included the START Twi libraries in your project, otherwise, the HAL layer will not compile because the START TWI drivers are a dependency *

initialize an I2C interface using given config

Parameters

in	<i>hal</i>	- opaque ptr to HAL data
in	<i>cfg</i>	- interface configuration

Returns

ATCA_SUCCESS on success, otherwise an error code.

this implementation assumes I2C peripheral has been enabled by user. It only initialize an I2C interface using given config.

Parameters

in	<i>hal</i>	pointer to HAL specific data that is maintained by this HAL
in	<i>cfg</i>	pointer to HAL specific configuration data that is used to initialize this HAL

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.14 hal_i2c_init() [2/2]

```
ATCA_STATUS hal_i2c_init (
    void * hal,
    ATCAIFaceCfg * cfg )
```

hal_i2c_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIFace instances using the same bus, and you can have multiple ATCAIFace instances on multiple i2c buses, so hal_i2c_init manages these things and ATCAIFace is abstracted from the physical details.

hal_i2c_init manages requests to initialize a physical interface. It manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIFace instances using the same bus, and you can have multiple ATCAIFace instances on multiple i2c buses, so hal_i2c_init manages these things and ATCAIFace is abstracted from the physical details.

initialize an I2C interface using given config

- this HAL implementation assumes you've included the START Twi libraries in your project, otherwise, the HAL layer will not compile because the START TWI drivers are a dependency *

initialize an I2C interface using given config

Parameters

in	<i>hal</i>	- opaque ptr to HAL data
in	<i>cfg</i>	- interface configuration

Returns

ATCA_SUCCESS on success, otherwise an error code.

- this HAL implementation assumes you've included the ASF SERCOM I2C libraries in your project, otherwise, the HAL layer will not compile because the ASF I2C drivers are a dependency *

Parameters

in	<i>hal</i>	- opaque ptr to HAL data
in	<i>cfg</i>	- interface configuration

Returns

ATCA_SUCCESS on success, otherwise an error code.

initialize an I2C interface using given config

Parameters

in	<i>hal</i>	- opaque ptr to HAL data
in	<i>cfg</i>	- interface configuration

Returns

ATCA_SUCCESS on success, otherwise an error code.

- this HAL implementation assumes you've included the ASF Twi libraries in your project, otherwise, the HAL layer will not compile because the ASF TWI drivers are a dependency *

initialize an I2C interface using given config

Parameters

in	<i>hal</i>	- opaque ptr to HAL data
in	<i>cfg</i>	- interface configuration

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.15 hal_i2c_post_init()

```
ATCA_STATUS hal_i2c_post_init (
    ATCAIface iface )
```

HAL implementation of I2C post init.

Parameters

in	<i>iface</i>	instance
----	--------------	----------

Returns

ATCA_SUCCESS

Parameters

in	<i>iface</i>	instance
----	--------------	----------

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>iface</i>	instance
----	--------------	----------

Returns

ATCA_SUCCESS

21.9.5.16 hal_i2c_receive()

```
ATCA_STATUS hal_i2c_receive (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * rxdata,
    uint16_t * rxlength )
```

HAL implementation of I2C receive function for START I2C.

HAL implementation of I2C receive function for ASF I2C.

HAL implementation of I2C receive function.

Parameters

in	<i>iface</i>	Device to interact with.
in	<i>word_address</i>	device transaction type
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>iface</i>	Device to interact with.
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>iface</i>	Device to interact with.
in	<i>address</i>	device address
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>iface</i>	Device to interact with.
in	<i>word_address</i>	device word address
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.17 hal_i2c_release()

```
ATCA_STATUS hal_i2c_release (
    void * hal_data )
```

manages reference count on given bus and releases resource if no more refences exist

manages reference count on given bus and releases resource if no more refernces exist

Parameters

in	<i>hal_data</i>	- opaque pointer to hal data structure - known only to the HAL implementation
----	-----------------	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>hal_data</i>	- opaque pointer to hal data structure - known only to the HAL implementation return ATCA_SUCCESS
----	-----------------	---

21.9 Hardware abstraction layer (hal_)

Parameters

in	<i>hal_data</i>	- opaque pointer to hal data structure - known only to the HAL implementation
----	-----------------	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>hal_data</i>	- opaque pointer to hal data structure - known only to the HAL implementation
----	-----------------	---

Returns

ATCA_SUCCESS

21.9.5.18 hal_i2c_send()

```
ATCA_STATUS hal_i2c_send (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * txdata,
    int txlength )
```

HAL implementation of I2C send over START.

HAL implementation of I2C send over ASF.

HAL implementation of I2C send.

Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	device transaction type
in	<i>txdata</i>	pointer to space to bytes to send
in	<i>txlength</i>	number of bytes to send

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>iface</i>	instance
in	<i>txdata</i>	pointer to space to bytes to send
in	<i>txlength</i>	number of bytes to send

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	device word address
in	<i>txdata</i>	pointer to space to bytes to send
in	<i>txlength</i>	number of bytes to send

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	device word address
in	<i>txdata</i>	pointer to space to bytes to send
in	<i>txlength</i>	number of bytes to send

Returns

ATCA_SUCCESS on success, otherwise an error code.

Add 1 byte for word address

Add 1 byte for word address

21.9.5.19 hal_i2c_sleep()

```
ATCA_STATUS hal_i2c_sleep (
    ATCAIface iface )
```

sleep CryptoAuth device using I2C bus

Parameters

in	<i>iface</i>	interface to logical device to sleep
----	--------------	--------------------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>iface</i>	interface to logical device to sleep
----	--------------	--------------------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.20 hal_i2c_wake()

```
ATCA_STATUS hal_i2c_wake (
    ATCAIface iface )
```

wake up CryptoAuth device using I2C bus

Parameters

in	<i>iface</i>	interface to logical device to wakeup
----	--------------	---------------------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>iface</i>	interface to logical device to wakeup
----	--------------	---------------------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.21 hal_iface_init()

```
ATCA_STATUS hal_iface_init (
    ATCAIfaceCfg * cfg,
    ATCAHAL_t ** hal,
    ATCAHAL_t ** phy )
```

Standard HAL API for ATCA to initialize a physical interface.

Parameters

in	<i>cfg</i>	pointer to ATCAIfaceCfg object
in	<i>hal</i>	pointer to ATCAHAL_t intermediate data structure

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.22 hal_iface_register_hal()

```
ATCA_STATUS hal_iface_register_hal (
    ATCAIfaceType iface_type,
    ATCAHAL_t * hal,
    ATCAHAL_t ** old_hal,
    ATCAHAL_t * phy,
    ATCAHAL_t ** old_phy )
```

Register/Replace a HAL with a.

Parameters

in	<i>iface_type</i>	- the type of physical interface to register
in	<i>hal</i>	pointer to the new ATCAHAL_t structure to register
out	<i>old</i>	pointer to the existing ATCAHAL_t structure

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.23 hal_iface_release()

```
ATCA_STATUS hal_iface_release (
    ATCAIfaceType iface_type,
    void * hal_data )
```

releases a physical interface, HAL knows how to interpret hal_data

Parameters

in	<i>iface_type</i>	- the type of physical interface to release
in	<i>hal_data</i>	- pointer to opaque hal data maintained by HAL implementation for this interface type

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.24 hal_is_command_word()

```
uint8_t hal_is_command_word (
    uint8_t word_address )
```

Utility function for hal_wake to check the reply.

21.9 Hardware abstraction layer (hal_)

Parameters

in	<i>word_address</i>	Command to check
----	---------------------	------------------

Returns

true if the *word_address* is considered a command

21.9.5.25 hal_kit_attach_phy()

```
ATCA_STATUS hal_kit_attach_phy (
    ATCAIfaceCfg * cfg,
    atca_hal_kit_phy_t * phy )
```

Helper function that connects a physical layer context structure that will be used by the kit protocol bridge.

Returns

ATCA_STATUS

Parameters

<i>cfg</i>	[IN] Interface configuration structure
<i>phy</i>	[IN] Structure with physical layer interface functions and context

21.9.5.26 hal_kit_control()

```
ATCA_STATUS hal_kit_control (
    ATCAIface iface,
    uint8_t option,
    void * param,
    size_t paramlen )
```

Kit Protocol Control.

Parameters

in	<i>iface</i>	ATCAIface instance that is the interface object to send the bytes over
in	<i>option</i>	Control option to use

Returns

ATCA_STATUS

21.9.5.27 `hal_kit_hid_control()`

```
ATCA_STATUS hal_kit_hid_control (
    ATCAIface iface,
    uint8_t option,
    void * param,
    size_t paramlen )
```

Perform control operations for the kit protocol.

Parameters

in	<i>iface</i>	Interface to interact with.
in	<i>option</i>	Control parameter identifier
in	<i>param</i>	Optional pointer to parameter value
in	<i>paramlen</i>	Length of the parameter

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.28 `hal_kit_hid_init()`

```
ATCA_STATUS hal_kit_hid_init (
    ATCAIface iface,
    ATCAIfaceCfg * cfg )
```

HAL implementation of Kit USB HID init.

Parameters

in	<i>hal</i>	pointer to HAL specific data that is maintained by this HAL
in	<i>cfg</i>	pointer to HAL specific configuration data that is used to initialize this HAL

Returns

ATCA_STATUS

21.9.5.29 `hal_kit_hid_post_init()`

```
ATCA_STATUS hal_kit_hid_post_init (
    ATCAIface iface )
```

HAL implementation of Kit HID post init.

Parameters

in	<i>iface</i>	instance
----	--------------	----------

Returns

ATCA_STATUS

21.9.5.30 hal_kit_hid_receive()

```
ATCA_STATUS hal_kit_hid_receive (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * rxdata,
    uint16_t * rxlength )
```

HAL implementation of send over USB HID.

Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	determine device transaction type
in	<i>rxdata</i>	pointer to space to receive the data
in, out	<i>rxsize</i>	ptr to expected number of receive bytes to request

Returns

ATCA_STATUS

21.9.5.31 hal_kit_hid_release()

```
ATCA_STATUS hal_kit_hid_release (
    void * hal_data )
```

Close the physical port for HID.

Parameters

in	<i>hal_data</i>	The hardware abstraction data specific to this HAL
----	-----------------	--

Returns

ATCA_STATUS

21.9.5.32 hal_kit_hid_send()

```
ATCA_STATUS hal_kit_hid_send (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * txdata,
    int txlength )
```

HAL implementation of kit protocol send over USB HID.

Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	determine device transaction type
in	<i>txdata</i>	pointer to bytes to send
in	<i>txlength</i>	number of bytes to send

Returns

ATCA_STATUS

21.9.5.33 hal_kit_init()

```
ATCA_STATUS hal_kit_init (
    ATCAIface iface,
    ATCAIfaceCfg * cfg )
```

HAL implementation of Kit USB HID init.

Parameters

in	<i>iface</i>	instance
in	<i>cfg</i>	pointer to HAL specific configuration data that is used to initialize this HAL

Returns

ATCA_STATUS

21.9.5.34 hal_kit_post_init()

```
ATCA_STATUS hal_kit_post_init (
    ATCAIface iface )
```

HAL implementation of Kit HID post init.

Parameters

in	<i>iface</i>	instance
----	--------------	----------

Returns

ATCA_STATUS

21.9.5.35 hal_kit_receive()

```
ATCA_STATUS hal_kit_receive (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * rxdata,
    uint16_t * rxsize )
```

HAL implementation of send over USB HID.

Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	determine device transaction type
in	<i>rxdata</i>	pointer to space to receive the data
in, out	<i>rxsize</i>	ptr to expected number of receive bytes to request

Returns

ATCA_STATUS

21.9.5.36 hal_kit_release()

```
ATCA_STATUS hal_kit_release (
    void * hal_data )
```

Close the physical port for HID.

Parameters

in	<i>hal_data</i>	The hardware abstraction data specific to this HAL
----	-----------------	--

Returns

ATCA_STATUS

21.9.5.37 hal_kit_send()

```
ATCA_STATUS hal_kit_send (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * txdata,
    int txlength )
```

HAL implementation of kit protocol send over USB HID.

Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	determine device transaction type
in	<i>txdata</i>	pointer to bytes to send
in	<i>txlength</i>	number of bytes to send

Returns

ATCA_STATUS

Add 1 byte to txlength for word address

21.9.5.38 hal_rtos_delay_ms()

```
void hal_rtos_delay_ms (
    uint32_t delay )
```

This function delays for a number of milliseconds.

You can override this function if you like to do something else in your system while delaying.

Parameters

in	<i>delay</i>	Number of milliseconds to delay
----	--------------	---------------------------------

21.9.5.39 hal_spi_control()

```
ATCA_STATUS hal_spi_control (
    ATCAIface iface,
    uint8_t option,
    void * param,
    size_t paramlen )
```

Perform control operations for the kit protocol.

Parameters

in	<i>iface</i>	Interface to interact with.
in	<i>option</i>	Control parameter identifier
in	<i>param</i>	Optional pointer to parameter value
in	<i>paramlen</i>	Length of the parameter

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.40 hal_spi_deselect()

```
ATCA_STATUS hal_spi_deselect (
    ATCAIface iface )
```

HAL implementation to deassert the device chip select.

Parameters

in	<i>iface</i>	Device to interact with.
----	--------------	--------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.41 hal_spi_discover_buses()

```
ATCA_STATUS hal_spi_discover_buses (
    int spi_buses[],
    int max_buses )
```

discover spi buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge

Parameters

in	<i>spi_buses</i>	- an array of logical bus numbers
in	<i>max_buses</i>	- maximum number of buses the app wants to attempt to discover

Returns

ATCA_SUCCESS

21.9.5.42 hal_spi_discover_devices()

```
ATCA_STATUS hal_spi_discover_devices (
    int bus_num,
    ATCAInterfaceCfg cfg[],
    int * found )
```

discover any TA10x devices on a given logical bus number

Parameters

in	bus_num	logical bus number on which to look for TA10x devices
out	cfg	pointer to head of an array of interface config structures which get filled in by this method
out	found	number of devices found on this bus

Returns

ATCA_SUCCESS

21.9.5.43 hal_spi_init()

```
ATCA_STATUS hal_spi_init (
    ATCAInterface iface,
    ATCAInterfaceCfg * cfg )
```

initialize an SPI interface using given config

Parameters

in	hal	- opaque ptr to HAL data
in	cfg	- interface configuration

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.44 hal_spi_post_init()

```
ATCA_STATUS hal_spi_post_init (
    ATCAInterface iface )
```

HAL implementation of SPI post init.

Parameters

in	<i>iface</i>	instance
----	--------------	----------

Returns

ATCA_SUCCESS

21.9.5.45 hal_spi_receive()

```
ATCA_STATUS hal_spi_receive (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * rxdata,
    uint16_t * rxlength )
```

HAL implementation of SPI receive function for HARMONY SPI.

Parameters

in	<i>iface</i>	Device to interact with.
in	<i>word_address</i>	device transaction type
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.46 hal_spi_release()

```
ATCA_STATUS hal_spi_release (
    void * hal_data )
```

manages reference count on given bus and releases resource if no more refences exist

Parameters

in	<i>hal_data</i>	- opaque pointer to hal data structure - known only to the HAL implementation
----	-----------------	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.47 hal_spi_select()

```
ATCA_STATUS hal_spi_select (
    ATCAIface iface )
```

HAL implementation to assert the device chip select.

Parameters

in	<i>iface</i>	Device to interact with.
----	--------------	--------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.48 hal_spi_send()

```
ATCA_STATUS hal_spi_send (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * txdata,
    int txlength )
```

HAL implementation of SPI send over Harmony.

Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	device transaction type
in	<i>txdata</i>	pointer to space to bytes to send
in	<i>txlength</i>	number of bytes to send

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.49 hal_swi_control()

```
ATCA_STATUS hal_swi_control (
    ATCAIface iface,
    uint8_t option,
    void * param,
    size_t paramlen )
```

Perform control operations for the kit protocol.

Parameters

in	<i>iface</i>	Interface to interact with.
in	<i>option</i>	Control parameter identifier
in	<i>param</i>	Optional pointer to parameter value
in	<i>paramlen</i>	Length of the parameter

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.50 hal_swi_idle()

```
ATCA_STATUS hal_swi_idle (  
    ATCAIface iface )
```

Send Idle flag via SWI.

Parameters

in	<i>iface</i>	interface of the logical device to idle
----	--------------	---

Returns

ATCA_SUCCESS

21.9.5.51 hal_swi_init()

```
ATCA_STATUS hal_swi_init (  
    ATCAIface iface,  
    ATCAIfaceCfg * cfg )
```

initialize an SWI interface using given config

Parameters

in	<i>hal</i>	- opaque ptr to HAL data
in	<i>cfg</i>	- interface configuration

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.52 hal_swi_post_init()

```
ATCA_STATUS hal_swi_post_init (
    ATCAIface iface )
```

HAL implementation of SWI post init.

Parameters

in	iface	instance
----	-------	----------

Returns

ATCA_SUCCESS

21.9.5.53 hal_swi_receive()

```
ATCA_STATUS hal_swi_receive (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * rxdata,
    uint16_t * rxlength )
```

HAL implementation of SWI receive function over UART.

Parameters

in	iface	Device to interact with.
in	word_address	device transaction type
out	rxdata	Data received will be returned here.
in, out	rxlength	As input, the size of the rxdata buffer. As output, the number of bytes received.

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.54 hal_swi_release()

```
ATCA_STATUS hal_swi_release (
    void * hal_data )
```

manages reference count on given bus and releases resource if no more refences exist

21.9 Hardware abstraction layer (hal_)

Parameters

in	<i>hal_data</i>	- opaque pointer to hal data structure - known only to the HAL implementation
----	-----------------	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.55 hal_swi_send()

```
ATCA_STATUS hal_swi_send (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * txdata,
    int txlength )
```

HAL implementation of SWI send command over UART.

Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	device transaction type
in	<i>txdata</i>	pointer to space to bytes to send
in	<i>txlength</i>	number of bytes to send

Returns

ATCA_SUCCESS on success, otherwise an error code.

Send word address

Send data

21.9.5.56 hal_swi_sleep()

```
ATCA_STATUS hal_swi_sleep (
    ATCAIface iface )
```

Send Sleep flag via SWI.

Parameters

in	<i>iface</i>	interface of the logical device to sleep
----	--------------	--

Returns

ATCA_SUCCESS

21.9.5.57 hal_swi_wake()

```
ATCA_STATUS hal_swi_wake (
    ATCAIface iface )
```

Send Wake flag via SWI.

Parameters

in	<i>iface</i>	interface of the logical device to wake up
----	--------------	--

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.58 kit_id_from_devtype()

```
const char * kit_id_from_devtype (
    ATCADeviceType devtype )
```

Kit Protocol is key

21.9.5.59 kit_interface()

```
const char * kit_interface (
    ATCAKitType kittype )
```

Kit parser physical interface string

21.9.5.60 kit_interface_from_kittype()

```
const char * kit_interface_from_kittype (
    ATCAKitType kittype )
```

Kit interface from device

21.9.5.61 swi_uart_deinit()

```
ATCA_STATUS swi_uart_deinit (
    ATCASWIMaster_t * instance )
```

Implementation of SWI UART deinit.

HAL implementation of SWI UART deinit.

Parameters

in	<i>instance</i>	instance
----	-----------------	----------

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>instance</i>	instance
----	-----------------	----------

Returns

ATCA_SUCCESS

21.9.5.62 swi_uart_discover_buses()

```
void swi_uart_discover_buses (
    int swi_uart_buses[],
    int max_buses )
```

discover UART buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge

Parameters

in	<i>swi_uart_buses</i>	- an array of logical bus numbers
in	<i>max_buses</i>	- maximum number of buses the app wants to attempt to discover

21.9.5.63 swi_uart_init()

```
ATCA_STATUS swi_uart_init (
    ATCASWIMaster_t * instance )
```

Implementation of SWI UART init.

HAL implementation of SWI UART init.

- this HAL implementation assumes you've included the ASF SERCOM UART libraries in your project, otherwise, the HAL layer will not compile because the ASF UART drivers are a dependency *

Parameters

in	<i>instance</i>	instance
----	-----------------	----------

Returns

ATCA_SUCCESS on success, otherwise an error code.

- this HAL implementation assumes you've included the START SERCOM UART libraries in your project, otherwise, the HAL layer will not compile because the START UART drivers are a dependency *

Parameters

in	<i>instance</i>	instance
----	-----------------	----------

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.64 swi_uart_mode()

```
void swi_uart_mode (
    ATCASWIMaster_t * instance,
    uint8_t mode )
```

implementation of SWI UART change mode.

HAL implementation of SWI UART change mode.

Parameters

in	<i>instance</i>	instance
in	<i>mode</i>	(TRANSMIT_MODE or RECEIVE_MODE)

21.9.5.65 swi_uart_receive_byte()

```
ATCA_STATUS swi_uart_receive_byte (
    ATCASWIMaster_t * instance,
    uint8_t * data )
```

HAL implementation of SWI UART receive bytes over ASF. This function receive one byte over UART.

Parameters

in	<i>instance</i>	instance
out	<i>data</i>	pointer to space to receive the data

21.10 Host side crypto methods (atcah_)

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.66 swi_uart_send_byte()

```
ATCA_STATUS swi_uart_send_byte (
    ATCASWIMaster_t * instance,
    uint8_t data )
```

HAL implementation of SWI UART send byte over ASF. This function send one byte over UART.

Parameters

in	<i>instance</i>	instance
in	<i>data</i>	number of byte to send

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.9.5.67 swi_uart_setbaud()

```
void swi_uart_setbaud (
    ATCASWIMaster_t * instance,
    uint32_t baudrate )
```

implementation of SWI UART change baudrate.

HAL implementation of SWI UART change baudrate.

Parameters

in	<i>instance</i>	instance
in	<i>baudrate</i>	(typically 230400 , 160000 or 115200)
in	<i>instance</i>	instance
in	<i>baudrate</i>	(typically 230400 or 115200)

21.10 Host side crypto methods (atcah_)

Use these functions if your system does not use an ATCADevice as a host but implements the host in firmware. The functions provide host-side cryptographic functionality for an ATECC client device. They are intended to accompany the CryptoAuthLib functions. They can be called directly from an application, or integrated into an API.

Data Structures

- struct [atca_temp_key](#)
Structure to hold TempKey fields.
- struct [atca_include_data_in_out](#)
Input / output parameters for function `atca_include_data()`.
- struct [atca_nonce_in_out](#)
Input/output parameters for function `atca_nonce()`.
- struct [atca_io_decrypt_in_out](#)
- struct [atca_verify_mac](#)
- struct [atca_secureboot_enc_in_out](#)
- struct [atca_secureboot_mac_in_out](#)
- struct [atca_mac_in_out](#)
Input/output parameters for function `atca_mac()`.
- struct [atca_hmac_in_out](#)
Input/output parameters for function `atca_hmac()`.
- struct [atca_gen_dig_in_out](#)
Input/output parameters for function `atcah_gen_dig()`.
- struct [atca_diversified_key_in_out](#)
Input/output parameters for function `atcah_gendivkey()`.
- struct [atca_write_mac_in_out](#)
Input/output parameters for function `atcah_write_auth_mac()` and `atcah_privwrite_auth_mac()`.
- struct [atca_derive_key_in_out](#)
Input/output parameters for function `atcah_derive_key()`.
- struct [atca_derive_key_mac_in_out](#)
Input/output parameters for function `atcah_derive_key_mac()`.
- struct [atca_decrypt_in_out](#)
Input/output parameters for function `atca_decrypt()`.
- struct [atca_check_mac_in_out](#)
Input/output parameters for function `atcah_check_mac()`.
- struct [atca_resp_mac_in_out](#)
*Input/Output parameters for calculating the output response mac in SHA105 device. Used with the `atcah_gen_↔
output_resp_mac()` function.*
- struct [atca_verify_in_out](#)
Input/output parameters for function `atcah_verify()`.
- struct [atca_gen_key_in_out](#)
*Input/output parameters for calculating the PubKey digest put into TempKey by the GenKey command with the `atcah_↔
_gen_key_msg()` function.*
- struct [atca_sign_internal_in_out](#)
Input/output parameters for calculating the message and digest used by the Sign(internal) command. Used with the `atcah_sign_internal_msg()` function.
- struct [atca_session_key_in_out](#)
*Input/Output paramters for calculating the session key by the nonce command. Used with the `atcah_gen_session_↔
_key()` function.*
- struct [atca_delete_in_out](#)
Input/Output paramters for calculating the mac.Used with Delete command.

Typedefs

- typedef struct [atca_temp_key](#) **atca_temp_key_t**
Structure to hold TempKey fields.
- typedef struct [atca_nonce_in_out](#) **atca_nonce_in_out_t**
- typedef struct [atca_io_decrypt_in_out](#) **atca_io_decrypt_in_out_t**
- typedef struct [atca_verify_mac](#) **atca_verify_mac_in_out_t**
- typedef struct [atca_secureboot_enc_in_out](#) **atca_secureboot_enc_in_out_t**
- typedef struct [atca_secureboot_mac_in_out](#) **atca_secureboot_mac_in_out_t**
- typedef struct [atca_mac_in_out](#) **atca_mac_in_out_t**
- typedef struct [atca_gen_dig_in_out](#) **atca_gen_dig_in_out_t**
Input/output parameters for function `atcah_gen_dig()`.
- typedef struct [atca_diversified_key_in_out](#) **atca_diversified_key_in_out_t**
Input/output parameters for function `atcah_gendivkey()`.
- typedef struct [atca_write_mac_in_out](#) **atca_write_mac_in_out_t**
Input/output parameters for function `atcah_write_auth_mac()` and `atcah_privwrite_auth_mac()`.
- typedef struct [atca_check_mac_in_out](#) **atca_check_mac_in_out_t**
Input/output parameters for function `atcah_check_mac()`.
- typedef struct [atca_resp_mac_in_out](#) **atca_resp_mac_in_out_t**
Input/Output parameters for calculating the output response mac in SHA105 device. Used with the `atcah_gen_↔output_resp_mac()` function.
- typedef struct [atca_verify_in_out](#) **atca_verify_in_out_t**
- typedef struct [atca_gen_key_in_out](#) **atca_gen_key_in_out_t**
Input/output parameters for calculating the PubKey digest put into TempKey by the GenKey command with the `atcah_↔gen_key_msg()` function.
- typedef struct [atca_sign_internal_in_out](#) **atca_sign_internal_in_out_t**
Input/output parameters for calculating the message and digest used by the Sign(internal) command. Used with the `atcah_sign_internal_msg()` function.
- typedef struct [atca_session_key_in_out](#) **atca_session_key_in_out_t**
Input/Output paramters for calculating the session key by the nonce command. Used with the `atcah_gen_session_↔key()` function.
- typedef struct [atca_delete_in_out](#) **atca_delete_in_out_t**
Input/Output paramters for calculating the mac.Used with Delete command.

Functions

- ATCA_STATUS **atcah_nonce** (struct [atca_nonce_in_out](#) *param)
- ATCA_STATUS **atcah_mac** (struct [atca_mac_in_out](#) *param)
- ATCA_STATUS **atcah_check_mac** (struct [atca_check_mac_in_out](#) *param)
- ATCA_STATUS **atcah_hmac** (struct [atca_hmac_in_out](#) *param)
- ATCA_STATUS **atcah_gen_dig** (struct [atca_gen_dig_in_out](#) *param)
- ATCA_STATUS **atcah_gendivkey** (struct [atca_diversified_key_in_out](#) *param)
- ATCA_STATUS **atcah_gen_mac** (struct [atca_gen_dig_in_out](#) *param)
- ATCA_STATUS **atcah_write_auth_mac** (struct [atca_write_mac_in_out](#) *param)
- ATCA_STATUS **atcah_privwrite_auth_mac** (struct [atca_write_mac_in_out](#) *param)
- ATCA_STATUS **atcah_derive_key** (struct [atca_derive_key_in_out](#) *param)
- ATCA_STATUS **atcah_derive_key_mac** (struct [atca_derive_key_mac_in_out](#) *param)
- ATCA_STATUS **atcah_decrypt** (struct [atca_decrypt_in_out](#) *param)
- ATCA_STATUS **atcah_sha256** (uint32_t len, const uint8_t *message, uint8_t *digest)
- uint8_t * **atcah_include_data** (struct [atca_include_data_in_out](#) *param)
- ATCA_STATUS **atcah_gen_key_msg** (struct [atca_gen_key_in_out](#) *param)
- ATCA_STATUS **atcah_config_to_sign_internal** (ATCADeviceType device_type, struct [atca_sign_internal_in_out](#) *param, const uint8_t *config)

- ATCA_STATUS **atcah_sign_internal_msg** (ATCADeviceType device_type, struct [atca_sign_internal_in_out](#) *param)
- ATCA_STATUS **atcah_verify_mac** ([atca_verify_mac_in_out_t](#) *param)
- ATCA_STATUS **atcah_secureboot_enc** ([atca_secureboot_enc_in_out_t](#) *param)
- ATCA_STATUS **atcah_secureboot_mac** ([atca_secureboot_mac_in_out_t](#) *param)
- ATCA_STATUS **atcah_encode_counter_match** (uint32_t counter_value, uint8_t *counter_match_value)
- ATCA_STATUS **atcah_io_decrypt** (struct [atca_io_decrypt_in_out](#) *param)
- ATCA_STATUS **atcah_ecc204_write_auth_mac** (struct [atca_write_mac_in_out](#) *param)
- ATCA_STATUS **atcah_gen_session_key** ([atca_session_key_in_out_t](#) *param)
- ATCA_STATUS **atcah_gen_output_resp_mac** (struct [atca_resp_mac_in_out](#) *param)

Variables

- uint8_t * **atca_include_data_in_out::p_temp**
[out] pointer to output buffer
- const uint8_t * **atca_include_data_in_out::otp**
[in] pointer to one-time-programming data
- const uint8_t * **atca_include_data_in_out::sn**
[in] pointer to serial number data
- uint8_t **atca_nonce_in_out::mode**
[in] Mode parameter used in Nonce command (Param1).
- uint16_t **atca_nonce_in_out::zero**
[in] Zero parameter used in Nonce command (Param2).
- const uint8_t * **atca_nonce_in_out::num_in**
[in] Pointer to 20-byte NumIn data used in Nonce command.
- const uint8_t * **atca_nonce_in_out::rand_out**
[in] Pointer to 32-byte RandOut data from Nonce command.
- struct [atca_temp_key](#) * **atca_nonce_in_out::temp_key**
[in,out] Pointer to TempKey structure.
- uint8_t **atca_mac_in_out::mode**
[in] Mode parameter used in MAC command (Param1).
- uint16_t **atca_mac_in_out::key_id**
[in] KeyID parameter used in MAC command (Param2).
- const uint8_t * **atca_mac_in_out::challenge**
[in] Pointer to 32-byte Challenge data used in MAC command, depending on mode.
- const uint8_t * **atca_mac_in_out::key**
[in] Pointer to 32-byte key used to generate MAC digest.
- const uint8_t * **atca_mac_in_out::otp**
[in] Pointer to 11-byte OTP, optionally included in MAC digest, depending on mode.
- const uint8_t * **atca_mac_in_out::sn**
[in] Pointer to 9-byte SN, optionally included in MAC digest, depending on mode.
- uint8_t * **atca_mac_in_out::response**
[out] Pointer to 32-byte SHA-256 digest (MAC).
- struct [atca_temp_key](#) * **atca_mac_in_out::temp_key**
[in,out] Pointer to TempKey structure.
- uint8_t **atca_hmac_in_out::mode**
[in] Mode parameter used in HMAC command (Param1).
- uint16_t **atca_hmac_in_out::key_id**
[in] KeyID parameter used in HMAC command (Param2).
- const uint8_t * **atca_hmac_in_out::key**
[in] Pointer to 32-byte key used to generate HMAC digest.

- `const uint8_t * atca_hmac_in_out::otp`
[in] Pointer to 11-byte OTP, optionally included in HMAC digest, depending on mode.
- `const uint8_t * atca_hmac_in_out::sn`
[in] Pointer to 9-byte SN, optionally included in HMAC digest, depending on mode.
- `uint8_t * atca_hmac_in_out::response`
[out] Pointer to 32-byte SHA-256 HMAC digest.
- `struct atca_temp_key * atca_hmac_in_out::temp_key`
[in,out] Pointer to TempKey structure.
- `uint8_t * atca_decrypt_in_out::crypto_data`
[in,out] Pointer to 32-byte data. Input encrypted data from Read command (Contents field), output decrypted.
- `struct atca_temp_key * atca_decrypt_in_out::temp_key`
[in,out] Pointer to TempKey structure.
- `uint16_t atca_verify_in_out::curve_type`
[in] Curve type used in Verify command (Param2).
- `const uint8_t * atca_verify_in_out::signature`
[in] Pointer to ECDSA signature to be verified
- `const uint8_t * atca_verify_in_out::public_key`
[in] Pointer to the public key to be used for verification
- `struct atca_temp_key * atca_verify_in_out::temp_key`
[in,out] Pointer to TempKey structure.

Definitions for ATECC Message Sizes to Calculate a SHA256 Hash

"||" is the concatenation operator. The number in braces is the length of the hash input value in bytes.

- `#define ATCA_MSG_SIZE_NONCE (55)`
`RandOut{32} || NumIn{20} || OpCode{1} || Mode{1} || LSB of Param2{1}.`
- `#define ATCA_MSG_SIZE_MAC (88)`
`(Key or TempKey){32} || (Challenge or TempKey){32} || OpCode{1} || Mode{1} || Param2{2} || (OTP0_7 or 0){8} || (OTP8_10 or 0){3} || SN8{1} || (SN4_7 or 0){4} || SN0_1{2} || (SN2_3 or 0){2}`
- `#define ATCA_MSG_SIZE_HMAC (88u)`
- `#define ATCA_MSG_SIZE_GEN_DIG (96)`
`KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0_1{2} || 0{25} || TempKey{32}.`
- `#define ATCA_MSG_SIZE_DIVERSIFIED_KEY (96)`
`ParentKey{32} || OtherData{4} || SN8{1} || SN0_1{2} || 0{25} || InputData{32}.`
- `#define ATCA_MSG_SIZE_DERIVE_KEY (96)`
`KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0_1{2} || 0{25} || TempKey{32}.`
- `#define ATCA_MSG_SIZE_DERIVE_KEY_MAC (39)`
`KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0_1{2}.`
- `#define ATCA_MSG_SIZE_ENCRYPT_MAC (96)`
`KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0_1{2} || 0{25} || TempKey{32}.`
- `#define ATCA_MSG_SIZE_SESSION_KEY (96)`
`TransportKey{32} || 0x15{1} || 0x00{1} || KeyId{2} || SN8{1} || SN0_1{2} || 0{25} || Nonce{32}.`
- `#define ATCA_MSG_SIZE_DELETE_MAC (96)`
`Hmac/SecretKey{32} || 0x13{1} || 0x00{1} || 0x0000{2} || SN8{1} || SN0_1{2} || 0{25} || Nonce{32}.`
- `#define ATCA_MSG_SIZE_RESPONSE_MAC (97)`
`SlotKey{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0_1{2} || 0{25} || client_Resp{32} || checkmac←_result{1}.`
- `#define ATCA_MSG_SIZE_PRIVWRITE_MAC (96)`
`KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0_1{2} || 0{21} || PlainText{36}.`

- `#define ATCA_COMMAND_HEADER_SIZE (4)`
- `#define ATCA_GENDIG_ZEROS_SIZE (25)`
- `#define ATCA_GENDIVKEY_ZEROS_SIZE (25)`
- `#define ATCA_WRITE_MAC_ZEROS_SIZE (25)`
- `#define ATCA_DELETE_MAC_ZEROS_SIZE (25)`
- `#define ATCA_RESP_MAC_ZEROS_SIZE (25)`
- `#define ATCA_PRIVWRITE_MAC_ZEROS_SIZE (21)`
- `#define ATCA_PRIVWRITE_PLAIN_TEXT_SIZE (36)`
- `#define ATCA_DERIVE_KEY_ZEROS_SIZE (25)`
- `#define ATCA_HMAC_BLOCK_SIZE (64u)`
- `#define ATCA_ENCRYPTION_KEY_SIZE (64)`

Definition for TempKey Mode

- `#define MAC_MODE_USE_TEMPKEY_MASK ((uint8_t)0x03)`
mode mask for MAC command when using TempKey

21.10.1 Detailed Description

Use these functions if your system does not use an ATCADevice as a host but implements the host in firmware. The functions provide host-side cryptographic functionality for an ATECC client device. They are intended to accompany the CryptoAuthLib functions. They can be called directly from an application, or integrated into an API.

Modern compilers can garbage-collect unused functions. If your compiler does not support this feature, you can just discard this module from your project if you do use an ATECC as a host. Or, if you don't, delete the functions you do not use.

21.11 JSON Web Token (JWT) methods (atca_jwt_)

Methods for signing and verifying JSON Web Token (JWT) tokens.

Methods for signing and verifying JSON Web Token (JWT) tokens.

21.12 mbedTLS Wrapper methods (atca_mbedtls_)

These methods are for interfacing cryptoauthlib to mbedtls.

21.12.0.1 mbedtls directory - Purpose

This directory contains the interfacing and wrapper functions to integrate mbedtls as the software crypto library as well as provide elliptic curve cryptography (ECC) hardware acceleration.

Data Structures

- struct [atca_mbedtls_eckey_s](#)

Typedefs

- typedef struct [atca_mbedtls_eckey_s](#) [atca_mbedtls_eckey_t](#)

Functions

- int [atca_mbedtls_ecdsa_sign](#) (const mbedtls_mpi *data, mbedtls_mpi *r, mbedtls_mpi *s, const unsigned char *msg, size_t msg_len)
- int [atca_mbedtls_pk_init_ext](#) ([ATCADevice](#) device, mbedtls_pk_context *pkey, const uint16_t slotid)
Initializes an mbedtls pk context for use with EC operations.
- int [atca_mbedtls_pk_init](#) (mbedtls_pk_context *pkey, const uint16_t slotid)
Initializes an mbedtls pk context for use with EC operations.
- int [atca_mbedtls_cert_add](#) (struct mbedtls_x509_crt *cert, const struct [atcacert_def_s](#) *cert_def)
- int [atca_mbedtls_random_ctx](#) (void *ctx, unsigned char *data, size_t data_size)
mbedtls f_rng callback interface to fill a buffer with random bytes
- struct mbedtls_x509_crt * [atcac_mbedtls_new](#) (void)
- struct [atcac_x509_ctx](#) * [atcac_x509_ctx_new](#) (void)
- void [atcac_x509_ctx_free](#) (struct [atcac_x509_ctx](#) *ctx)

Variables

- const mbedtls_pk_info_t [atca_mbedtls_eckey_info](#)

21.12.1 Detailed Description

These methods are for interfacing cryptauthlib to mbedtls.

21.12.2 Typedef Documentation

21.12.2.1 [atca_mbedtls_eckey_t](#)

```
typedef struct atca\_mbedtls\_eckey\_s atca\_mbedtls\_eckey\_t
```

Structure to hold metadata - is written into the mbedtls pk structure as the private key bignum value 'd' which otherwise would be unused. Bignums can be any arbitrary length of bytes

21.12.3 Function Documentation

21.12.3.1 [atca_mbedtls_pk_init\(\)](#)

```
int atca_mbedtls_pk_init (  
    mbedtls_pk_context * pkey,  
    const uint16_t slotid )
```

Initializes an mbedtls pk context for use with EC operations.

Parameters

<code>in, out</code>	<code>pkey</code>	ptr to space to receive version string
<code>in</code>	<code>slotid</code>	Associated with this key

Returns

0 on success, otherwise an error code.

21.12.3.2 `atca_mbedtls_pk_init_ext()`

```
int atca_mbedtls_pk_init_ext (
    ATCADevice device,
    mbedtls_pk_context * pkey,
    const uint16_t slotid )
```

Initializes an mbedtls pk context for use with EC operations.

Parameters

<code>in, out</code>	<code>pkey</code>	ptr to space to receive version string
<code>in</code>	<code>slotid</code>	Associated with this key

Returns

0 on success, otherwise an error code.

21.12.3.3 `atca_mbedtls_random_ctx()`

```
int atca_mbedtls_random_ctx (
    void * ctx,
    unsigned char * data,
    size_t data_size )
```

mbedtls f_rng callback interface to fill a buffer with random bytes

Returns

ATCA_SUCCESS on success, otherwise an error code.

21.13 Attributes (`pkcs11_attrib_`)

Data Structures

- struct `pkcs11_conf_filedata_s`
- struct `pkcs11_mech_table_e`

Macros

- #define **PKCS11_CONFIG_U8_MAX** 0xFFL
- #define **PKCS11_CONFIG_U16_MAX** 0xFFFFL
- #define **PKCS11_CONFIG_U32_MAX** 0xFFFFFFFFL
- #define **PKCS11_MECH_ECC508_EC_CAPABILITY** (CKF_EC_F_P | CKF_EC_NAMEDCURVE | CKF_↵
EC_UNCOMPRESS)
- #define **TABLE_SIZE**(x) sizeof(x) / sizeof(x[0])

Typedefs

- typedef struct **pkcs11_conf_filedata_s** **pkcs11_conf_filedata**
- typedef struct **pkcs11_conf_filedata_s** * **pkcs11_conf_filedata_ptr**
- typedef struct **pkcs11_mech_table_e** **pkcs11_mech_table_e**
- typedef struct **pkcs11_mech_table_e** * **pkcs11_mech_table_ptr**

Functions

- CK_RV **pkcs11_attrib_fill** (CK_ATTRIBUTE_PTR pAttribute, const void *pData, const CK_ULONG ulSize)
Perform the nessasary checks and copy data into an attribute structure.
- CK_RV **pkcs11_attrib_value** (CK_ATTRIBUTE_PTR pAttribute, const CK_ULONG ulValue, const CK_↵
ULONG ulSize)
Helper function to write a numerical value to an attribute buffer.
- CK_RV **pkcs11_attrib_false** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, **pkcs11_session_ctx_ptr**
pSession)
- CK_RV **pkcs11_attrib_true** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, **pkcs11_session_ctx_ptr**
pSession)
- CK_RV **pkcs11_attrib_empty** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, **pkcs11_session_ctx_ptr**
pSession)
- CK_RV **pkcs11_cert_load** (pkcs11_object_ptr pObject, CK_ATTRIBUTE_PTR pAttribute, **ATCADevice** de-
vice)
- CK_RV **pkcs11_cert_x509_write** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, **pkcs11_session_ctx_ptr**
pSession)
- CK_RV **pkcs11_cert_clear_session_cache** (**pkcs11_session_ctx_ptr** session_ctx)
- CK_RV **pkcs11_cert_clear_object_cache** (pkcs11_object_ptr pObject)
- void **pkcs11_config_set_key_size** (pkcs11_object_ptr pObject)
- void **pkcs11_config_init_private** (pkcs11_object_ptr pObject, const char *label, size_t len)
- void **pkcs11_config_init_public** (pkcs11_object_ptr pObject, const char *label, size_t len)
- void **pkcs11_config_init_secret** (pkcs11_object_ptr pObject, const char *label, size_t len, size_t keylen)
- void **pkcs11_config_init_cert** (pkcs11_object_ptr pObject, const char *label, size_t len)
- void **pkcs11_config_split_string** (char *s, char splitter, int *argc, char *argv[])
- CK_RV **pkcs11_config_cert** (pkcs11_lib_ctx_ptr pLibCtx, pkcs11_slot_ctx_ptr pSlot, pkcs11_object_ptr p↵
Object, CK_ATTRIBUTE_PTR pLabel)
- CK_RV **pkcs11_config_key** (pkcs11_lib_ctx_ptr pLibCtx, pkcs11_slot_ctx_ptr pSlot, pkcs11_object_ptr p↵
Object, CK_ATTRIBUTE_PTR pLabel)
- CK_RV **pkcs11_config_remove_object** (pkcs11_lib_ctx_ptr pLibCtx, pkcs11_slot_ctx_ptr pSlot, pkcs11↵
_object_ptr pObject)
- CK_RV **pkcs11_config_load_objects** (pkcs11_slot_ctx_ptr slot_ctx)
- CK_RV **pkcs11_config_load** (pkcs11_slot_ctx_ptr slot_ctx)
- CK_RV **pkcs11_encrypt_init** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism,
CK_OBJECT_HANDLE hObject)
- CK_RV **pkcs11_encrypt** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulData↵
Len, CK_BYTE_PTR pEncryptedData, CK_ULONG_PTR pulEncryptedDataLen)

- CK_RV **pkcs11_encrypt_update** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pEncryptedData, CK_ULONG_PTR pulEncryptedDataLen)
- CK_RV **pkcs11_encrypt_final** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedData, CK_ULONG_PTR pulEncryptedDataLen)
Finishes a multiple-part encryption operation.
- CK_RV **pkcs11_decrypt_init** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hObject)
- CK_RV **pkcs11_decrypt** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedData, CK_ULONG ulEncryptedDataLen, CK_BYTE_PTR pData, CK_ULONG_PTR pulDataLen)
- CK_RV **pkcs11_decrypt_update** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedData, CK_ULONG ulEncryptedDataLen, CK_BYTE_PTR pData, CK_ULONG_PTR pulDataLen)
- CK_RV **pkcs11_decrypt_final** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG_PTR pulDataLen)
Finishes a multiple-part decryption operation.
- CK_RV **pkcs11_find_init** (CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount)
- CK_RV **pkcs11_find_continue** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE_PTR phObject, CK_ULONG ulMaxObjectCount, CK_ULONG_PTR pulObjectCount)
- CK_RV **pkcs11_find_finish** (CK_SESSION_HANDLE hSession)
- CK_RV **pkcs11_find_get_attribute** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount)
- CK_RV **pkcs11_get_lib_info** (CK_INFO_PTR pInfo)
Obtains general information about Cryptoki.
- **pkcs11_lib_ctx_ptr** **pkcs11_get_context** (void)
Retrieve the current library context.
- CK_RV **pkcs11_lock_context** (pkcs11_lib_ctx_ptr pContext)
- CK_RV **pkcs11_unlock_context** (pkcs11_lib_ctx_ptr pContext)
- CK_RV **pkcs11_lock_device** (pkcs11_lib_ctx_ptr pContext)
- CK_RV **pkcs11_unlock_device** (pkcs11_lib_ctx_ptr pContext)
- CK_RV **pkcs11_lock_both** (pkcs11_lib_ctx_ptr pContext)
- CK_RV **pkcs11_unlock_both** (pkcs11_lib_ctx_ptr pContext)
- CK_RV **pkcs11_init_check** (pkcs11_lib_ctx_ptr *ppContext, CK_BBOOL lock)
Check if the library is initialized properly.
- CK_RV **pkcs11_init** (CK_C_INITIALIZE_ARGS const *pInitArgs)
Initializes the PKCS11 API Library for Cryptoauthlib.
- CK_RV **pkcs11_deinit** (CK_VOID_PTR pReserved)
- const **pkcs11_key_info_t** * **pkcs11_get_object_key_type** (ATCADevice device_ctx, pkcs11_object_ptr obj_ptr)
- CK_RV **pkcs11_ta_get_pubkey** (CK_VOID_PTR pObject, cal_buffer *key_buffer, pkcs11_session_ctx_ptr session_ctx)
- CK_RV **pkcs11_key_write** (CK_VOID_PTR pSession, CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute)
- CK_RV **pkcs11_key_generate** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phKey)
- CK_RV **pkcs11_key_generate_pair** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_ATTRIBUTE_PTR pPublicKeyTemplate, CK_ULONG ulPublicKeyAttributeCount, CK_ATTRIBUTE_PTR pPrivateKeyTemplate, CK_ULONG ulPrivateKeyAttributeCount, CK_OBJECT_HANDLE_PTR phPublicKey, CK_OBJECT_HANDLE_PTR phPrivateKey)
- CK_RV **pkcs11_key_derive** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hBaseKey, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phKey)
- CK_RV **pkcs11_key_clear_session_cache** (pkcs11_session_ctx_ptr session_ctx)
- CK_RV **pkcs11_key_clear_object_cache** (pkcs11_object_ptr pObject)
- CK_RV **C_Initialize** (CK_VOID_PTR pInitArgs)

Initializes Cryptoki library NOTES: If pInitArgs is a non-NULL_PTR is must dereference to a [CK_C_INITIALIZE_ARGS](#) structure.

- **CK_RV C_Finalize** (CK_VOID_PTR pReserved)
Clean up miscellaneous Cryptoki-associated resources.
- **CK_RV C_GetInfo** (CK_INFO_PTR pInfo)
Obtains general information about Cryptoki.
- **CK_RV C_GetFunctionList** (CK_FUNCTION_LIST_PTR_PTR ppFunctionList)
Obtains entry points of Cryptoki library functions.
- **CK_RV C_GetSlotList** (CK_BBOOL tokenPresent, CK_SLOT_ID_PTR pSlotList, CK_ULONG_PTR pulCount)
Obtains a list of slots in the system.
- **CK_RV C_GetSlotInfo** (CK_SLOT_ID slotID, CK_SLOT_INFO_PTR pInfo)
Obtains information about a particular slot.
- **CK_RV C_GetTokenInfo** (CK_SLOT_ID slotID, CK_TOKEN_INFO_PTR pInfo)
Obtains information about a particular token.
- **CK_RV C_GetMechanismList** (CK_SLOT_ID slotID, CK_MECHANISM_TYPE_PTR pMechanismList, CK_ULONG_PTR pulCount)
Obtains a list of mechanisms supported by a token (in a slot)
- **CK_RV C_GetMechanismInfo** (CK_SLOT_ID slotID, CK_MECHANISM_TYPE type, CK_MECHANISM_INFO_PTR pInfo)
Obtains information about a particular mechanism of a token (in a slot)
- **CK_RV C_InitToken** (CK_SLOT_ID slotID, CK_UTF8CHAR_PTR pPin, CK_ULONG ulPinLen, CK_UTF8CHAR_PTR pLabel)
Initializes a token (in a slot)
- **CK_RV C_InitPIN** (CK_SESSION_HANDLE hSession, CK_UTF8CHAR_PTR pPin, CK_ULONG ulPinLen)
Initializes the normal user's PIN.
- **CK_RV C_SetPIN** (CK_SESSION_HANDLE hSession, CK_UTF8CHAR_PTR pOldPin, CK_ULONG ulOldLen, CK_UTF8CHAR_PTR pNewPin, CK_ULONG ulNewLen)
Modifies the PIN of the current user.
- **CK_RV C_OpenSession** (CK_SLOT_ID slotID, CK_FLAGS flags, CK_VOID_PTR pApplication, CK_NOTIFY Notify, CK_SESSION_HANDLE_PTR phSession)
Opens a connection between an application and a particular token or sets up an application callback for token insertion.
- **CK_RV C_CloseSession** (CK_SESSION_HANDLE hSession)
Close the given session.
- **CK_RV C_CloseAllSessions** (CK_SLOT_ID slotID)
Close all open sessions.
- **CK_RV C_GetSessionInfo** (CK_SESSION_HANDLE hSession, CK_SESSION_INFO_PTR pInfo)
Retrieve information about the specified session.
- **CK_RV C_GetOperationState** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pOperationState, CK_ULONG_PTR pulOperationStateLen)
Obtains the cryptographic operations state of a session.
- **CK_RV C_SetOperationState** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pOperationState, CK_ULONG ulOperationStateLen, CK_OBJECT_HANDLE hEncryptionKey, CK_OBJECT_HANDLE hAuthenticationKey)
Sets the cryptographic operations state of a session.
- **CK_RV C_Login** (CK_SESSION_HANDLE hSession, CK_USER_TYPE userType, CK_UTF8CHAR_PTR pPin, CK_ULONG ulPinLen)
Login on the token in the specified session.
- **CK_RV C_Logout** (CK_SESSION_HANDLE hSession)
Log out of the token in the specified session.
- **CK_RV C_CreateObject** (CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phObject)

Create a new object on the token in the specified session using the given attribute template.

- CK_RV **C_CopyObject** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_↔
ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phNewObject)

Create a copy of the object with the specified handle.

- CK_RV **C_DestroyObject** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject)

Destroy the specified object.

- CK_RV **C_GetObjectSize** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_↔
ULONG_PTR pulSize)

Obtains the size of an object in bytes.

- CK_RV **C_GetAttributeValue** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_↔
ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount)

Obtains an attribute value of an object.

- CK_RV **C_SetAttributeValue** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_↔
ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount)

Change or set the value of the specified attributes on the specified object.

- CK_RV **C_FindObjectsInit** (CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_↔
ULONG ulCount)

Initializes an object search in the specified session using the specified attribute template as search parameters.

- CK_RV **C_FindObjects** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE_PTR phObject, CK_↔
ULONG ulMaxObjectCount, CK_ULONG_PTR pulObjectCount)

Continue the search for objects in the specified session.

- CK_RV **C_FindObjectsFinal** (CK_SESSION_HANDLE hSession)

Finishes an object search operation (and cleans up)

- CK_RV **C_EncryptInit** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_↔
OBJECT_HANDLE hKey)

Initializes an encryption operation using the specified mechanism and session.

- CK_RV **C_Encrypt** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen,
CK_BYTE_PTR pEncryptedData, CK_ULONG_PTR pulEncryptedDataLen)

Perform a single operation encryption operation in the specified session.

- CK_RV **C_EncryptUpdate** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ul↔
PartLen, CK_BYTE_PTR pEncryptedPart, CK_ULONG_PTR pulEncryptedPartLen)

Continues a multiple-part encryption operation.

- CK_RV **C_EncryptFinal** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pLastEncryptedPart, CK_↔
ULONG_PTR pulLastEncryptedPartLen)

Finishes a multiple-part encryption operation.

- CK_RV **C_DecryptInit** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_↔
OBJECT_HANDLE hKey)

Initialize decryption using the specified object.

- CK_RV **C_Decrypt** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedData, CK_ULONG ul↔
EncryptedDataLen, CK_BYTE_PTR pData, CK_ULONG_PTR pulDataLen)

Perform a single operation decryption in the given session.

- CK_RV **C_DecryptUpdate** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedPart, CK_↔
ULONG ulEncryptedPartLen, CK_BYTE_PTR pPart, CK_ULONG_PTR pulPartLen)

Continues a multiple-part decryption operation.

- CK_RV **C_DecryptFinal** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pLastPart, CK_ULONG_PTR
pulLastPartLen)

Finishes a multiple-part decryption operation.

- CK_RV **C_DigestInit** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism)

Initializes a message-digesting operation using the specified mechanism in the specified session.

- CK_RV **C_Digest** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen,
CK_BYTE_PTR pDigest, CK_ULONG_PTR pulDigestLen)

Digest the specified data in a one-pass operation and return the resulting digest.

- **CK_RV C_DigestUpdate** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen)

Continues a multiple-part digesting operation.
- **CK_RV C_DigestKey** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hKey)

Update a running digest operation by digesting a secret key with the specified handle.
- **CK_RV C_DigestFinal** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pDigest, CK_ULONG_PTR pulDigestLen)

Finishes a multiple-part digesting operation.
- **CK_RV C_SignInit** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey)

Initialize a signing operation using the specified key and mechanism.
- **CK_RV C_Sign** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG_PTR pulSignatureLen)

Sign the data in a single pass operation.
- **CK_RV C_SignUpdate** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen)

Continues a multiple-part signature operation.
- **CK_RV C_SignFinal** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pSignature, CK_ULONG_PTR pulSignatureLen)

Finishes a multiple-part signature operation.
- **CK_RV C_SignRecoverInit** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey)

Initializes a signature operation, where the data can be recovered from the signature.
- **CK_RV C_SignRecover** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG_PTR pulSignatureLen)

Signs single-part data, where the data can be recovered from the signature.
- **CK_RV C_VerifyInit** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey)

Initializes a verification operation using the specified key and mechanism.
- **CK_RV C_Verify** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG ulSignatureLen)

Verifies a signature on single-part data.
- **CK_RV C_VerifyUpdate** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen)

Continues a multiple-part verification operation.
- **CK_RV C_VerifyFinal** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pSignature, CK_ULONG ulSignatureLen)

Finishes a multiple-part verification operation.
- **CK_RV C_VerifyRecoverInit** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey)

Initializes a verification operation where the data is recovered from the signature.
- **CK_RV C_VerifyRecover** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pSignature, CK_ULONG ulSignatureLen, CK_BYTE_PTR pData, CK_ULONG_PTR pulDataLen)

Verifies a signature on single-part data, where the data is recovered from the signature.
- **CK_RV C_DigestEncryptUpdate** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen, CK_BYTE_PTR pEncryptedPart, CK_ULONG_PTR pulEncryptedPartLen)

Continues simultaneous multiple-part digesting and encryption operations.
- **CK_RV C_DecryptDigestUpdate** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedPart, CK_ULONG ulEncryptedPartLen, CK_BYTE_PTR pPart, CK_ULONG_PTR pulPartLen)

Continues simultaneous multiple-part decryption and digesting operations.
- **CK_RV C_SignEncryptUpdate** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen, CK_BYTE_PTR pEncryptedPart, CK_ULONG_PTR pulEncryptedPartLen)

Continues simultaneous multiple-part signature and encryption operations.

- **CK_RV C_DecryptVerifyUpdate** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedPart, CK_ULONG ulEncryptedPartLen, CK_BYTE_PTR pPart, CK_ULONG_PTR pulPartLen)
Continues simultaneous multiple-part decryption and verification operations.
- **CK_RV C_GenerateKey** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phKey)
Generates a secret key using the specified mechanism.
- **CK_RV C_GenerateKeyPair** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_ATTRIBUTE_PTR pPublicKeyTemplate, CK_ULONG ulPublicKeyAttributeCount, CK_ATTRIBUTE_PTR pPrivateKeyTemplate, CK_ULONG ulPrivateKeyAttributeCount, CK_OBJECT_HANDLE_PTR phPublicKey, CK_OBJECT_HANDLE_PTR phPrivateKey)
Generates a public-key/private-key pair using the specified mechanism.
- **CK_RV C_WrapKey** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hWrappingKey, CK_OBJECT_HANDLE hKey, CK_BYTE_PTR pWrappedKey, CK_ULONG_PTR pulWrappedKeyLen)
Wraps (encrypts) the specified key using the specified wrapping key and mechanism.
- **CK_RV C_UnwrapKey** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hUnwrappingKey, CK_BYTE_PTR pWrappedKey, CK_ULONG ulWrappedKeyLen, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulAttributeCount, CK_OBJECT_HANDLE_PTR phKey)
Unwraps (decrypts) the specified key using the specified unwrapping key.
- **CK_RV C_DeriveKey** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hBaseKey, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulAttributeCount, CK_OBJECT_HANDLE_PTR phKey)
Derive a key from the specified base key.
- **CK_RV C_SeedRandom** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pSeed, CK_ULONG ulSeedLen)
Mixes in additional seed material to the random number generator.
- **CK_RV C_GenerateRandom** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR RandomData, CK_ULONG ulRandomLen)
Generate the specified amount of random data.
- **CK_RV C_GetFunctionStatus** (CK_SESSION_HANDLE hSession)
Legacy function - see PKCS#11 v2.40.
- **CK_RV C_CancelFunction** (CK_SESSION_HANDLE hSession)
Legacy function.
- **CK_RV C_WaitForSlotEvent** (CK_FLAGS flags, CK_SLOT_ID_PTR pSlot, CK_VOID_PTR pRsrvd)
Wait for a slot event (token insertion, removal, etc) on the specified slot to occur.
- **CK_RV pkcs11_mech_get_list** (CK_SLOT_ID slotID, CK_MECHANISM_TYPE_PTR pMechanismList, CK_ULONG_PTR pulCount)
- **CK_RV pkcs11_mech_get_info** (CK_SLOT_ID slotID, CK_MECHANISM_TYPE type, CK_MECHANISM_INFO_PTR pInfo)
- **CK_RV pkcs11_object_alloc** (CK_SLOT_ID slotId, pkcs11_object_ptr *ppObject)
- **CK_RV pkcs11_object_free** (pkcs11_object_ptr pObject)
- **CK_RV pkcs11_object_check** (pkcs11_object_ptr *ppObject, CK_OBJECT_HANDLE hObject)
- **CK_RV pkcs11_object_get_handle** (pkcs11_object_ptr pObject, CK_OBJECT_HANDLE_PTR phObject)
- **CK_RV pkcs11_object_get_owner** (pkcs11_object_ptr pObject, CK_SLOT_ID_PTR pSlotId)
- **CK_RV pkcs11_object_get_name** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- **CK_RV pkcs11_object_get_class** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- **CK_RV pkcs11_object_get_type** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- **CK_RV pkcs11_object_get_destroyable** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- **CK_RV pkcs11_object_get_size** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_ULONG_PTR pulSize)

- CK_RV **pkcs11_object_find** (CK_SLOT_ID slotId, pkcs11_object_ptr *ppObject, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount)
- CK_RV **pkcs11_object_create** (CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phObject)
Create a new object on the token in the specified session using the given attribute template.
- CK_RV **pkcs11_object_destroy** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject)
Destroy the specified object.
- CK_RV **pkcs11_object_deinit** (pkcs11_lib_ctx_ptr pContext)
- ATCA_STATUS **pkcs11_object_load_handle_info** (ATCADevice device, pkcs11_lib_ctx_ptr pContext)
- CK_RV **pkcs11_object_is_private** (pkcs11_object_ptr pObject, CK_BBOOL *is_private, pkcs11_session_ctx_ptr pSession)
Checks the attributes of the underlying cryptographic asset to determine if it is a private key - this changes the way the associated public key is referenced.
- CK_RV **pkcs11_os_create_mutex** (CK_VOID_PTR_PTR ppMutex)
Application callback for creating a mutex object.
- CK_RV **pkcs11_os_destroy_mutex** (CK_VOID_PTR pMutex)
- CK_RV **pkcs11_os_lock_mutex** (CK_VOID_PTR pMutex)
- CK_RV **pkcs11_os_unlock_mutex** (CK_VOID_PTR pMutex)
- CK_RV **pkcs11_os_alloc_shared_ctx** (void **ppShared, size_t size)
- CK_RV **pkcs11_os_free_shared_ctx** (void *pShared, size_t size)
- **pkcs11_session_ctx_ptr pkcs11_get_session_context** (CK_SESSION_HANDLE hSession)
- CK_RV **pkcs11_session_check** (pkcs11_session_ctx_ptr *pSession, CK_SESSION_HANDLE hSession)
Check if the session is initialized properly.
- CK_RV **pkcs11_reserve_resource** (pkcs11_lib_ctx_ptr pContext, pkcs11_session_ctx_ptr pSession, uint8_t resource)
- CK_RV **pkcs11_release_resource** (pkcs11_lib_ctx_ptr pContext, pkcs11_session_ctx_ptr pSession, uint8_t resource)
- CK_RV **pkcs11_session_open** (CK_SLOT_ID slotID, CK_FLAGS flags, CK_VOID_PTR pApplication, CK_NOTIFY notify, CK_SESSION_HANDLE_PTR phSession)
- CK_RV **pkcs11_session_close** (CK_SESSION_HANDLE hSession)
- CK_RV **pkcs11_session_closeall** (CK_SLOT_ID slotID)
Close all sessions for a given slot - not actually all open sessions.
- CK_RV **pkcs11_session_get_info** (CK_SESSION_HANDLE hSession, CK_SESSION_INFO_PTR pInfo)
Obtains information about a particular session.
- CK_RV **pkcs11_session_login** (CK_SESSION_HANDLE hSession, CK_USER_TYPE userType, CK_UTF8CHAR_PTR pPin, CK_ULONG ulPinLen)
- CK_RV **pkcs11_session_logout** (CK_SESSION_HANDLE hSession)
- CK_RV **pkcs11_signature_sign_init** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey)
Initialize a signing operation using the specified key and mechanism.
- CK_RV **pkcs11_signature_sign** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG_PTR pulSignatureLen)
Sign the data in a single pass operation.
- CK_RV **pkcs11_signature_sign_continue** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen)
Continues a multiple-part signature operation.
- CK_RV **pkcs11_signature_sign_finish** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pSignature, CK_ULONG_PTR pulSignatureLen)
Finishes a multiple-part signature operation.
- CK_RV **pkcs11_signature_verify_init** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey)
Initializes a verification operation using the specified key and mechanism.
- CK_RV **pkcs11_signature_verify** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG ulSignatureLen)

Verifies a signature on single-part data.

- CK_RV **pkcs11_signature_verify_continue** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen)

Continues a multiple-part verification operation.

- CK_RV **pkcs11_signature_verify_finish** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pSignature, CK_ULONG ulSignatureLen)

Finishes a multiple-part verification operation.

- pkcs11_slot_ctx_ptr **pkcs11_slot_get_context** (pkcs11_lib_ctx_ptr lib_ctx, CK_SLOT_ID slotID)

Retrieve the current slot context.

- pkcs11_slot_ctx_ptr **pkcs11_slot_get_new_context** (pkcs11_lib_ctx_ptr lib_ctx)
- CK_VOID_PTR **pkcs11_slot_initslots** (CK_ULONG pulCount)
- CK_RV **pkcs11_slot_deinitslots** (pkcs11_lib_ctx_ptr lib_ctx)
- CK_RV **pkcs11_slot_config** (CK_SLOT_ID slotID)
- CK_RV **pkcs11_slot_init** (CK_SLOT_ID slotID)

This is an internal function that initializes a pkcs11 slot - it must already have the locks in place before being called.

- CK_RV **pkcs11_slot_get_list** (CK_BBOOL tokenPresent, CK_SLOT_ID_PTR pSlotList, CK_ULONG_PTR pulCount)
- CK_RV **pkcs11_slot_get_info** (CK_SLOT_ID slotID, CK_SLOT_INFO_PTR pInfo)

Obtains information about a particular slot.

- CK_RV **pkcs11_token_init** (CK_SLOT_ID slotID, CK_UTF8CHAR_PTR pPin, CK_ULONG ulPinLen, CK_UTF8CHAR_PTR pLabel)
- CK_RV **pkcs11_token_get_access_type** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, pkcs11_session_ctx_ptr pSession)
- CK_RV **pkcs11_token_get_writable** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, pkcs11_session_ctx_ptr pSession)
- CK_RV **pkcs11_token_get_storage** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, pkcs11_session_ctx_ptr pSession)
- CK_RV **pkcs11_token_get_info** (CK_SLOT_ID slotID, CK_TOKEN_INFO_PTR pInfo)

Obtains information about a particular token.

- CK_RV **pkcs11_token_random** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pRandomData, CK_ULONG ulRandomLen)

Generate the specified amount of random data.

- CK_RV **pkcs11_token_convert_pin_to_key** (const CK_UTF8CHAR_PTR pPin, const CK_ULONG ulPinLen, const CK_UTF8CHAR_PTR pSalt, const CK_ULONG ulSaltLen, CK_BYTE_PTR pKey, CK_ULONG ulKeyLen, pkcs11_slot_ctx_ptr slot_ctx)
- CK_RV **pkcs11_token_set_pin** (CK_SESSION_HANDLE hSession, CK_UTF8CHAR_PTR pOldPin, CK_ULONG ulOldLen, CK_UTF8CHAR_PTR pNewPin, CK_ULONG ulNewLen)
- void **pkcs11_util_escape_string** (CK_UTF8CHAR_PTR buf, CK_ULONG buf_len)
- CK_RV **pkcs11_util_convert_rv** (ATCA_STATUS status)
- int **pkcs11_util_memset** (void *dest, size_t destsz, int ch, size_t count)

Variables

- const pkcs11_attr_model pkcs11_cert_x509public_attributes []
- const CK_ULONG pkcs11_cert_x509public_attributes_count = (CK_ULONG)(sizeof(pkcs11_cert_x509public_attributes) / sizeof(pkcs11_cert_x509public_attributes[0]))
- const pkcs11_attr_model pkcs11_cert_wtlspublic_attributes []
- const CK_ULONG pkcs11_cert_wtlspublic_attributes_count = (CK_ULONG)(sizeof(pkcs11_cert_wtlspublic_attributes) / sizeof(pkcs11_cert_wtlspublic_attributes[0]))
- const pkcs11_attr_model pkcs11_cert_x509_attributes []
- const CK_ULONG pkcs11_cert_x509_attributes_count = (CK_ULONG)(sizeof(pkcs11_cert_x509_attributes) / sizeof(pkcs11_cert_x509_attributes[0]))
- const char pkcs11_lib_manufacturer_id [] = "Microchip Technology Inc"

- const char **pkcs11_lib_description** [] = "Cryptoauthlib PKCS11 Interface"
- CK_BYTE **pkcs11_ec_pbkey_asn1_hdr_p256** []
- CK_BYTE **pkcs11_x962_asn1_hdr_ec256** []
- CK_BYTE **pkcs11_key_ec_params_p256** [] = { 0x06, 0x08, 0x2a, 0x86, 0x48, 0xce, 0x3d, 0x03, 0x01, 0x07 }
- CK_BYTE **pkcs11_ec_pbkey_asn1_hdr_p224** []
- CK_BYTE **pkcs11_x962_asn1_hdr_ec224** []
- CK_BYTE **pkcs11_key_ec_params_p224** [] = { 0x06, 0x05, 0x2B, 0x81, 0x04, 0x00, 0x21 }
- CK_BYTE **pkcs11_ec_pbkey_asn1_hdr_p384** []
- CK_BYTE **pkcs11_key_ec_params_p384** [] = { 0x06, 0x05, 0x2B, 0x81, 0x04, 0x00, 0x22 }
- CK_BYTE **pkcs11_x962_asn1_hdr_ec384** []
- CK_BYTE **pkcs11_ec_pbkey_asn1_hdr_p521** []
- CK_BYTE **pkcs11_x962_asn1_hdr_ec521** []
- CK_BYTE **pkcs11_key_ec_params_p521** [] = { 0x06, 0x05, 0x2B, 0x81, 0x04, 0x00, 0x23 }
- const **pkcs11_ecc_key_info_t** **ec_key_data_table** [4]
- const **pkcs11_rsa_key_info_t** **rsa_key_data_table** [4]
- const **pkcs11_key_info_t** **key_data_table** []
- const **pkcs11_attrib_model** **pkcs11_key_public_attributes** []
- const CK_ULONG **pkcs11_key_public_attributes_count** = (CK_ULONG)(sizeof(**pkcs11_key_public_attributes**) / sizeof(**pkcs11_key_public_attributes** [0]))
- const **pkcs11_attrib_model** **pkcs11_key_private_attributes** []
- const CK_ULONG **pkcs11_key_private_attributes_count** = (CK_ULONG)(sizeof(**pkcs11_key_private_attributes**) / sizeof(**pkcs11_key_private_attributes** [0]))
- const **pkcs11_attrib_model** **pkcs11_key_secret_attributes** []
- const CK_ULONG **pkcs11_key_secret_attributes_count** = (CK_ULONG)(sizeof(**pkcs11_key_secret_attributes**) / sizeof(**pkcs11_key_secret_attributes** [0]))
- **pkcs11_object_cache_t** **pkcs11_object_cache** [PKCS11_MAX_OBJECTS_ALLOWED]
- const **pkcs11_attrib_model** **pkcs11_object_monotonic_attributes** []
- const CK_ULONG **pkcs11_object_monotonic_attributes_count** = (CK_ULONG)(sizeof(**pkcs11_object_monotonic_attributes**) / sizeof(**pkcs11_object_monotonic_attributes** [0]))

21.13.1 Detailed Description

21.13.2 Function Documentation

21.13.2.1 pkcs11_attrib_fill()

```
CK_RV pkcs11_attrib_fill (
    CK_ATTRIBUTE_PTR pAttribute,
    const void * pData,
    const CK_ULONG ulSize )
```

Perform the nessasary checks and copy data into an attribute structure.

The ulValueLen field is modified to hold the exact length of the specified attribute for the object. In the special case of an attribute whose value is an array of attributes, for example CKA_WRAP_TEMPLATE, where it is passed in with pValue not NULL, then if the pValue of elements within the array is NULL_PTR then the ulValueLen of elements within the array will be set to the required length. If the pValue of elements within the array is not NULL_PTR, then the ulValueLen element of attributes within the array MUST reflect the space that the corresponding pValue points to, and pValue is filled in if there is sufficient room. Therefore it is important to initialize the contents of a buffer before calling C_GetAttributeValue to get such an array value. If any ulValueLen within the array isn't large enough, it will be set to CK_UNAVAILABLE_INFORMATION and the function will return CKR_BUFFER_TOO_SMALL, as it does if an attribute in the pTemplate argument has ulValueLen too small. Note that any attribute whose value is an array of attributes is identifiable by virtue of the attribute type having the CKF_ARRAY_ATTRIBUTE bit set.

21.13.2.2 pkcs11_deinit()

```
CK_RV pkcs11_deinit (
    CK_VOID_PTR pReserved )
```

21.13.2.3 pkcs11_init()

```
CK_RV pkcs11_init (
    CK_C_INITIALIZE_ARGS const * pInitArgs )
```

Initializes the PKCS11 API Library for Cryptoauthlib.

21.13.2.4 pkcs11_os_create_mutex()

```
CK_RV pkcs11_os_create_mutex (
    CK_VOID_PTR_PTR ppMutex )
```

Application callback for creating a mutex object.

Parameters

in, out	<i>ppMutex</i>	location to receive ptr to mutex
---------	----------------	----------------------------------

21.13.2.5 pkcs11_session_closeall()

```
CK_RV pkcs11_session_closeall (
    CK_SLOT_ID slotID )
```

Close all sessions for a given slot - not actually all open sessions.

for specified slotid close all sessions related with it.

21.13.2.6 pkcs11_session_login()

```
CK_RV pkcs11_session_login (
    CK_SESSION_HANDLE hSession,
    CK_USER_TYPE userType,
    CK_UTF8CHAR_PTR pPin,
    CK_ULONG ulPinLen )
```

Reserve the PKCS11_AUTH_OP_0 / PKCS11_AUTH_OP_1 based on availability

Auth operation unavailable return error

21.13.2.7 pkcs11_token_init()

```
CK_RV pkcs11_token_init (
    CK_SLOT_ID slotID,
    CK_UTF8CHAR_PTR pPin,
    CK_ULONG ulPinLen,
    CK_UTF8CHAR_PTR pLabel )
```

Write the configuration into the device and generate new keys

21.13.3 Variable Documentation

21.13.3.1 ec_key_data_table

```
const pkcs11_ec_key_info_t ec_key_data_table[4]
```

Initial value:

```
= {
    { ATCA_KEY_TYPE_ECCP256, (CK_BYTE)ATCA_ECCP256_OID_SIZE, pkcs11_key_ec_params_p256,
      pkcs11_ec_pbkey_asn1_hdr_p256,
      pkcs11_x962_asn1_hdr_ec256, (uint16_t)ATCA_ECCP256_ASN1_HDR_SIZE, ATCA_ECCP256_PUBKEY_SIZE,
      ATCA_ECCP256_MSG_SIZE, ATCA_ECCP256_SIG_SIZE },
    { TA_KEY_TYPE_ECCP224, (CK_BYTE)TA_ECC224_OID_SIZE, pkcs11_key_ec_params_p224,
      pkcs11_ec_pbkey_asn1_hdr_p224,
      pkcs11_x962_asn1_hdr_ec224, (uint16_t)TA_ECC224_ASN1_HDR_SIZE, TA_ECC224_PUB_KEY_SIZE,
      TA_SIGN_P224_MSG_SIZE, TA_SIGN_P224_SIG_SIZE },
    { TA_KEY_TYPE_ECCP384, (CK_BYTE)TA_ECC384_OID_SIZE, pkcs11_key_ec_params_p384,
      pkcs11_ec_pbkey_asn1_hdr_p384,
      pkcs11_x962_asn1_hdr_ec384, (uint16_t)TA_ECC384_ASN1_HDR_SIZE, TA_ECC384_PUB_KEY_SIZE,
      TA_SIGN_P384_MSG_SIZE, TA_SIGN_P384_SIG_SIZE },
    { TA_KEY_TYPE_ECCP521, (CK_BYTE)TA_ECC521_OID_SIZE, pkcs11_key_ec_params_p521,
      pkcs11_ec_pbkey_asn1_hdr_p521,
      pkcs11_x962_asn1_hdr_ec521, (uint16_t)TA_ECC521_ASN1_HDR_SIZE, TA_ECC521_PUB_KEY_SIZE,
      TA_SIGN_P521_MSG_SIZE, TA_SIGN_P521_SIG_SIZE },
}
```

21.13.3.2 key_data_table

```
const pkcs11_key_info_t key_data_table[]
```

Initial value:

```
= {
    { &ec_key_data_table[0], NULL },
    { &ec_key_data_table[1], NULL },
    { &ec_key_data_table[2], NULL },
    { &ec_key_data_table[3], NULL },
}
```

21.13.3.3 pkcs11_cert_wtlspublic_attributes

```
const pkcs11_attrib_model pkcs11_cert_wtlspublic_attributes[]
```

CKO_CERTIFICATE (Type: CKC_WTLS) - TLS Public Key Certificate Model

21.13.3.4 pkcs11_cert_x509_attributes

```
const pkcs11_attr_model pkcs11_cert_x509_attributes[]
```

CKO_CERTIFICATE (Type: CKC_X_509_ATTR_CERT) - X509 Attribute Certificate Model

21.13.3.5 pkcs11_cert_x509public_attributes

```
const pkcs11_attr_model pkcs11_cert_x509public_attributes[]
```

CKO_CERTIFICATE (Type: CKC_X_509) - X509 Public Key Certificate Model

21.13.3.6 pkcs11_ec_pbkey_asn1_hdr_p224

```
CK_BYTE pkcs11_ec_pbkey_asn1_hdr_p224[]
```

Initial value:

```
= {
    0x30, 0x4e,
    0x30, 0x10,
    0x06, 0x07,
    0x2a, 0x86, 0x48, 0xce, 0x3d, 0x02, 0x01,
    0x06, 0x05, 0x2b, 0x81, 0x04, 0x00, 0x21,
    0x03, 0x3a, 0x00,
    0x04
}
```

ASN.1 Header for SECP224R1 public keys

21.13.3.7 pkcs11_ec_pbkey_asn1_hdr_p256

```
CK_BYTE pkcs11_ec_pbkey_asn1_hdr_p256[]
```

Initial value:

```
= {
    0x30, 0x59,
    0x30, 0x13,
    0x06, 0x07,
    0x2a, 0x86, 0x48, 0xce, 0x3d, 0x02, 0x01,
    0x06, 0x08, 0x2a, 0x86, 0x48, 0xce, 0x3d, 0x03, 0x01, 0x07,
    0x03, 0x42, 0x00,
    0x04
}
```

ASN.1 Header for SECP256R1 public keys

21.13.3.8 pkcs11_ec_pbkey_asn1_hdr_p384

```
CK_BYTE pkcs11_ec_pbkey_asn1_hdr_p384[]
```

Initial value:

```
= {
    0x30, 0x76,
    0x30, 0x10,
    0x06, 0x07,
    0x2a, 0x86, 0x48, 0xce, 0x3d, 0x02, 0x01,
    0x06, 0x05,
    0x2b, 0x81, 0x04, 0x00, 0x22,
    0x03, 0x62, 0x00,
    0x04
}
```

ASN.1 Header for SECP384R1 public keys

21.13.3.9 pkcs11_ec_pbkey_asn1_hdr_p521

CK_BYTE pkcs11_ec_pbkey_asn1_hdr_p521[]

Initial value:

```
= {  
    0x30, 0x81, 0x9b,  
    0x30, 0x10,  
    0x06, 0x07,  
    0x2a, 0x86, 0x48, 0xce, 0x3d, 0x02, 0x01,  
    0x06, 0x05,  
    0x2b, 0x81, 0x04, 0x00, 0x23,  
    0x03, 0x81, 0x86, 0x00,  
    0x04  
}
```

ASN.1 Header for SECP521R1 public keys

21.13.3.10 pkcs11_key_private_attributes

const [pkcs11_attrib_model](#) pkcs11_key_private_attributes[]

CKO_PRIVATE_KEY - Private Key Object Base Model

21.13.3.11 pkcs11_key_public_attributes

const [pkcs11_attrib_model](#) pkcs11_key_public_attributes[]

CKO_PUBLIC_KEY - Public Key Object Model

21.13.3.12 pkcs11_key_secret_attributes

const [pkcs11_attrib_model](#) pkcs11_key_secret_attributes[]

CKO_SECRET_KEY - Secret Key Object Base Model

21.13.3.13 pkcs11_object_monotonic_attributes

const [pkcs11_attrib_model](#) pkcs11_object_monotonic_attributes[]

Initial value:

```
= {  
    { 0x00000000UL ,          pkcs11_object_get_class },  
    { 0x00000300UL , pkcs11_object_get_type },  
    { 0x00000301UL ,          pkcs11_attrib_false },  
    { 0x00000302UL ,          pkcs11_attrib_false },  
    { 0x00000011UL ,          0 },  
}
```

CKA_CLASS == CKO_HW_FEATURE_TYPE CKA_HW_FEATURE_TYPE == CKH_MONOTONIC_COUNTER

21.13.3.14 pkcs11_x962_asn1_hdr_ec224

```
CK_BYTE pkcs11_x962_asn1_hdr_ec224[ ]
```

Initial value:

```
= {  
    0x04, 0x39, 0x04  
}
```

X.962 ASN.1 Header for EC224 public keys

21.13.3.15 pkcs11_x962_asn1_hdr_ec256

```
CK_BYTE pkcs11_x962_asn1_hdr_ec256[ ]
```

Initial value:

```
= {  
    0x04, 0x41, 0x04  
}
```

X.962 ASN.1 Header for EC256 public keys

21.13.3.16 pkcs11_x962_asn1_hdr_ec384

```
CK_BYTE pkcs11_x962_asn1_hdr_ec384[ ]
```

Initial value:

```
= {  
    0x04, 0x61, 0x04  
}
```

X.962 ASN.1 Header for EC384 public keys

21.13.3.17 pkcs11_x962_asn1_hdr_ec521

```
CK_BYTE pkcs11_x962_asn1_hdr_ec521[ ]
```

Initial value:

```
= {  
    0x04, 0x85, 0x04  
}
```

X.962 ASN.1 Header for EC521 public keys

21.13.3.18 rsa_key_data_table

```
const pkcs11_rsa_key_info_t rsa_key_data_table[4]
```

Initial value:

```
= {  
}
```

Chapter 22

Namespace Documentation

22.1 cryptoauthlib Namespace Reference

Namespaces

- namespace [atcab](#)
- namespace [atcacert](#)
- namespace [atcaenum](#)
- namespace [atjwt](#)
- namespace [device](#)
- namespace [exceptions](#)
- namespace [iface](#)
- namespace [library](#)
- namespace [sha206_api](#)
- namespace [status](#)
- namespace [tng](#)

Variables

- **try :**
- `os_lib_definition_file = os.path.join(os.path.dirname(__file__), 'cryptoauth.json')`

22.1.1 Detailed Description

Package Definition

22.2 cryptoauthlib.atcab Namespace Reference

Data Structures

- class [atca_aes_cbc_ctx](#)
- class [atca_aes_cbcmac_ctx](#)
- class [atca_aes_ccm_ctx](#)
- class [atca_aes_cmac_ctx](#)
- class [atca_aes_ctr_ctx](#)
- class [atca_aes_gcm_ctx](#)
- class [atca_hmac_sha256_ctx](#)
- class [atca_sha256_ctx](#)

Functions

- def [atcab_init](#) (iface_cfg)
- def [atcab_release](#) ()
- def [atcab_get_device](#) ()
- def [atcab_get_device_type](#) ()
- def [atcab_aes](#) (mode, key_id, aes_in, aes_out)
- def [atcab_aes_encrypt](#) (key_id, key_block, plaintext, ciphertext)
- def [atcab_aes_decrypt](#) (key_id, key_block, ciphertext, plaintext)
- def [atcab_aes_gfm](#) (hash_key, inp, output)
- def [atcab_aes_cbc_init](#) (ctx, key_id, key_block, iv)
- def [atcab_aes_cbc_encrypt_block](#) (ctx, plaintext, ciphertext)
- def [atcab_aes_cbc_decrypt_block](#) (ctx, ciphertext, plaintext)
- def [atcab_aes_cmac_init](#) (ctx, key_id, key_block)
- def [atcab_aes_cmac_update](#) (ctx, data, data_size)
- def [atcab_aes_cmac_finish](#) (ctx, cmac, size)
- def [atcab_aes_ctr_init](#) (ctx, key_id, key_block, counter_size, iv)
- def [atcab_aes_ctr_init_rand](#) (ctx, key_id, key_block, counter_size, iv)
- def [atcab_aes_ctr_encrypt_block](#) (ctx, plaintext, ciphertext)
- def [atcab_aes_ctr_decrypt_block](#) (ctx, ciphertext, plaintext)
- def [atcab_aes_gcm_init](#) (ctx, key_id, key_block, iv, iv_size)
- def [atcab_aes_gcm_init_rand](#) (ctx, key_id, key_block, rand_size, free_field, free_field_size, iv)
- def [atcab_aes_gcm_aad_update](#) (ctx, aad, aad_size)
- def [atcab_aes_gcm_encrypt_update](#) (ctx, plaintext, plaintext_size, ciphertext)
- def [atcab_aes_gcm_encrypt_finish](#) (ctx, tag, tag_size)
- def [atcab_aes_gcm_decrypt_update](#) (ctx, ciphertext, ciphertext_size, plaintext)
- def [atcab_aes_gcm_decrypt_finish](#) (ctx, tag, tag_size, is_verified)
- def [atcab_aes_cbcmac_init](#) (ctx, key_id, key_block)
- def [atcab_aes_cbcmac_update](#) (ctx, data, data_size)
- def [atcab_aes_cbcmac_finish](#) (ctx, mac, mac_size)
- def [atcab_aes_ccm_init](#) (ctx, key_id, key_block, iv, iv_size, aad_size, text_size, tag_size)
- def [atcab_aes_ccm_init_rand](#) (ctx, key_id, key_block, iv, iv_size, aad_size, text_size, tag_size)
- def [atcab_aes_ccm_aad_update](#) (ctx, aad, aad_size)
- def [atcab_aes_ccm_aad_finish](#) (ctx)
- def [atcab_aes_ccm_encrypt_update](#) (ctx, plaintext, plaintext_size, ciphertext)
- def [atcab_aes_ccm_decrypt_update](#) (ctx, ciphertext, ciphertext_size, plaintext)
- def [atcab_aes_ccm_encrypt_finish](#) (ctx, tag, tag_size)
- def [atcab_aes_ccm_decrypt_finish](#) (ctx, tag, is_verified)
- def [atcab_checkmac](#) (mode, key_id, challenge, response, other_data)
- def [atcab_counter](#) (mode, counter_id, counter_value)
- def [atcab_counter_increment](#) (counter_id, counter_value)
- def [atcab_counter_read](#) (counter_id, counter_value)
- def [atcab_derivekey](#) (mode, target_key, mac)
- def [atcab_ecdh_base](#) (mode, key_id, public_key, pms, out_nonce)
- def [atcab_ecdh](#) (key_id, public_key, pms)
- def [atcab_ecdh_enc](#) (key_id, public_key, pms, read_key, read_key_id, num_in=None)
- def [atcab_ecdh_ioenc](#) (key_id, public_key, pms, io_key)
- def [atcab_ecdh_tempkey](#) (public_key, pms)
- def [atcab_ecdh_tempkey_ioenc](#) (public_key, pms, io_key)
- def [atcab_gendig](#) (zone, key_id, other_data, other_data_size)
- def [atcab_genkey_base](#) (mode, key_id, other_data, public_key=None)
- def [atcab_genkey](#) (key_id, public_key)
- def [atcab_get_pubkey](#) (key_id, public_key)
- def [atcab_hmac](#) (mode, key_id, digest)
- def [atcab_info_base](#) (mode, param2, out_data)

- def [atcab_info](#) (revision)
- def [atcab_info_get_latch](#) (state)
- def [atcab_info_set_latch](#) (state)
- def [atcab_kdf](#) (mode, key_id, details, message, out_data, out_nonce)
- def [atcab_lock](#) (mode, summary_crc)
- def [atcab_lock_config_zone](#) ()
- def [atcab_lock_config_zone_crc](#) (summary_crc)
- def [atcab_lock_data_zone](#) ()
- def [atcab_lock_data_zone_crc](#) (summary_crc)
- def [atcab_lock_data_slot](#) (slot)
- def [atcab_mac](#) (mode, key_id, challenge, digest)
- def [atcab_nonce_base](#) (mode, zero, num_in, rand_out)
- def [atcab_nonce](#) (num_in)
- def [atcab_nonce_load](#) (target, num_in, num_in_size)
- def [atcab_nonce_rand](#) (num_in, rand_out)
- def [atcab_challenge](#) (num_in)
- def [atcab_challenge_seed_update](#) (num_in, rand_out)
- def [atcab_priv_write](#) (key_id, priv_key, write_key_id, write_key, num_in=None)
- def [atcab_random](#) (random_number)
- def [atcab_read_zone](#) (zone, slot, block, offset, data, length)
- def [atcab_read_serial_number](#) (serial_number)
- def [atcab_is_slot_locked](#) (slot, is_locked)
- def [atcab_is_locked](#) (zone, is_locked)
- def [atcab_read_enc](#) (key_id, block, data, enc_key, enc_key_id, num_in=None)
- def [atcab_read_config_zone](#) (config_data)
- def [atcab_cmp_config_zone](#) (config_data, same_config)
- def [atcab_read_sig](#) (slot, sig)
- def [atcab_read_pubkey](#) (slot, public_key)
- def [atcab_read_bytes_zone](#) (zone, slot, offset, data, length)
- def [atcab_secureboot](#) (mode, param2, digest, signature, mac)
- def [atcab_secureboot_mac](#) (mode, digest, signature, num_in, io_keys, is_verified)
- def [atcab_selftest](#) (mode, param2, result)
- def [atcab_sha_base](#) (mode, length, message, data_out, data_out_size)
- def [atcab_sha_start](#) ()
- def [atcab_sha_update](#) (message)
- def [atcab_sha_end](#) (digest, length, message)
- def [atcab_sha_read_context](#) (context, context_size)
- def [atcab_sha_write_context](#) (context, context_size)
- def [atcab_sha](#) (length, message, digest)
- def [atcab_hw_sha2_256_init](#) (ctx)
- def [atcab_hw_sha2_256_update](#) (ctx, data, data_size)
- def [atcab_hw_sha2_256_finish](#) (ctx, digest)
- def [atcab_hw_sha2_256](#) (data, data_size, digest)
- def [atcab_sha_hmac_init](#) (ctx, key_slot)
- def [atcab_sha_hmac_update](#) (ctx, data, data_size)
- def [atcab_sha_hmac_finish](#) (ctx, digest, target)
- def [atcab_sha_hmac](#) (data, data_size, key_slot, digest, target)
- def [atcab_sign_base](#) (mode, key_id, signature)
- def [atcab_sign](#) (key_id, msg, signature)
- def [atcab_sign_internal](#) (key_id, is_invalidate, is_full_sn, signature)
- def [atcab_updateextra](#) (mode, new_value)
- def [atcab_verify](#) (mode, key_id, signature, public_key, other_data, mac)
- def [atcab_verify_extern_stored_mac](#) (mode, key_id, message, signature, public_key, num_in, io_key, is_↵
verified)
- def [atcab_verify_extern](#) (message, signature, public_key, is_verified)

- def [atcab_verify_extern_mac](#) (message, signature, public_key, num_in, io_key, is_verified)
- def [atcab_verify_stored](#) (message, signature, key_id, is_verified)
- def [atcab_verify_stored_mac](#) (message, signature, key_id, num_in, io_key, is_verified)
- def [atcab_verify_validate](#) (key_id, signature, other_data, is_verified)
- def [atcab_verify_invalidate](#) (key_id, signature, other_data, is_verified)
- def [atcab_write](#) (zone, [address](#), value, mac)
- def [atcab_write_zone](#) (zone, slot, block, offset, data, length)
- def [atcab_write_enc](#) (key_id, block, data, enc_key, enc_key_id, num_in=None)
- def [atcab_write_config_zone](#) (conf)
- def [atcab_write_pubkey](#) (slot, public_key)
- def [atcab_write_bytes_zone](#) (zone, slot, offset_bytes, data, length)
- def [atcab_write_config_counter](#) (counter_id, counter_value)

22.2.1 Detailed Description

Dynamic link library loading under ctypes and HAL initialization/release functions

22.2.2 Function Documentation

22.2.2.1 atcab_aes()

```
def cryptauthlib.atcab.atcab_aes (
    mode,
    key_id,
    aes_in,
    aes_out )
```

Compute the AES-128 encrypt, decrypt, or GFM calculation.

Args:

mode	The mode for the AES command. (int)
key_id	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey. (int)
aes_in	Input data to the AES command (16 bytes). (Can be of type bytearray or bytes)
aes_out	Output data from the AES command is returned here (16 bytes). (Expects bytearray of size 16)

Returns:

Status Code

22.2.2.2 atcab_aes_cbc_decrypt_block()

```
def cryptoauthlib.atcab.atcab_aes_cbc_decrypt_block (
    ctx,
    ciphertext,
    plaintext )
```

Decrypt a block of data using CBC mode and a key within the ATECC608. `atcab_aes_cbc_init()` should be called before the first use of this function.

Args:

<code>ctx</code>	AES CBC context.
<code>ciphertext</code>	Ciphertext to be decrypted (16 bytes). (Bytearray or bytes)
<code>plaintext</code>	Decrypted data is returned here (16 bytes). (Bytearray or bytes)

Returns:

Status code

22.2.2.3 atcab_aes_cbc_encrypt_block()

```
def cryptoauthlib.atcab.atcab_aes_cbc_encrypt_block (
    ctx,
    plaintext,
    ciphertext )
```

Encrypt a block of data using CBC mode and a key within the ATECC608. `atcab_aes_cbc_init()` should be called before the first use of this function.

Args:

<code>ctx</code>	AES CBC context.
<code>plaintext</code>	Plaintext to be encrypted (16 bytes). (Bytearray or bytes)
<code>ciphertext</code>	Encrypted data is returned here (16 bytes). (Bytearray or bytes)

Returns:

Status code

22.2.2.4 atcab_aes_cbc_init()

```
def cryptoauthlib.atcab.atcab_aes_cbc_init (
    ctx,
    key_id,
    key_block,
    iv )
```

Initialize context for AES CBC operation.

Args:

<code>ctx</code>	AES CBC context to be initialized
<code>key_id</code>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
<code>key_block</code>	Index of the 16-byte block to use within the key location for the actual key.
<code>iv</code>	Initialization vector (16 bytes). Bytearray format

Returns:

Status Code

22.2.2.5 atcab_aes_cbcmac_finish()

```
def cryptoauthlib.atcab.atcab_aes_cbcmac_finish (
    ctx,
    mac,
    mac_size )
```

Finish a CBC-MAC operation returning the CBC-MAC value. If the data provided to the `atcab_aes_cbcmac_update()` function has incomplete block this function will return an error code.

Args:

<code>ctx</code>	AES-128 CBC-MAC context.
<code>mac</code>	CBC-MAC is returned here.
<code>mac_size</code>	Size of CBC-MAC requested in bytes (max 16 bytes).

Returns:

ATCA_SUCCESS on success, otherwise an error code.

22.2.2.6 atcab_aes_cbcmac_init()

```
def cryptoauthlib.atcab.atcab_aes_cbcmac_init (
    ctx,
    key_id,
    key_block )
```

Initialize context for AES CBC-MAC operation.

Args:

<code>ctx</code>	AES CBC-MAC context to be initialized
<code>key_id</code>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
<code>key_block</code>	Index of the 16-byte block to use within the key location for the actual key.

Returns:

ATCA_SUCCESS on success, otherwise an error code.

22.2.2.7 atcab_aes_cbcmac_update()

```
def cryptoauthlib.atcab.atcab_aes_cbcmac_update (
    ctx,
    data,
    data_size )
```

Calculate AES CBC-MAC with key stored within ECC608A device.
atcab_aes_cbcmac_init() should be called before the first use of
this function.

Args:

ctx	AES CBC-MAC context structure.
data	Data to be added for AES CBC-MAC calculation. Can be bytearray or bytes.
data_size	Data length in bytes.

Returns:

ATCA_SUCCESS on success, otherwise an error code.

22.2.2.8 atcab_aes_ccm_aad_finish()

```
def cryptoauthlib.atcab.atcab_aes_ccm_aad_finish (
    ctx )
```

Finish processing Additional Authenticated Data (AAD) using CCM mode.

Args:

ctx	AES CCM context
-----	-----------------

22.2.2.9 atcab_aes_ccm_aad_update()

```
def cryptoauthlib.atcab.atcab_aes_ccm_aad_update (
    ctx,
    aad,
    aad_size )
```

Process Additional Authenticated Data (AAD) using CCM mode and a
key within the ATECC608A device

Args:

ctx	AES CCM context
aad	Additional authenticated data to be added
aad_size	Size of aad in bytes.

22.2.2.10 atcab_aes_ccm_decrypt_finish()

```
def cryptoauthlib.atcab.atcab_aes_ccm_decrypt_finish (
    ctx,
    tag,
    is_verified )
```

Complete a CCM decrypt operation authenticating provided tag.

Args:

ctx	AES CCM context structure.
tag	Authentication tag is returned here.
is_verified	Value is set to true if the tag is authenticated else the value is set to false.

22.2.2.11 atcab_aes_ccm_decrypt_update()

```
def cryptoauthlib.atcab.atcab_aes_ccm_decrypt_update (
    ctx,
    ciphertext,
    ciphertext_size,
    plaintext )
```

Process data using CCM mode and a key within the ATECC608A device. atcab_aes_ccm_init() or atcab_aes_ccm_init_rand() should be called before the first use of this function.

Args:

ctx	AES CCM context structure.
ciphertext	Data to be processed.
ciphertext_size	Size of the data to be processed.
plaintext	Output data is returned here.

22.2.2.12 atcab_aes_ccm_encrypt_finish()

```
def cryptoauthlib.atcab.atcab_aes_ccm_encrypt_finish (
    ctx,
    tag,
    tag_size )
```

Complete a CCM encrypt operation returning the authentication tag.

Args:

ctx	AES CCM context structure.
tag	Authentication tag is returned here.
tag_size	Tag size in bytes.

22.2.2.13 atcab_aes_ccm_encrypt_update()

```
def cryptoauthlib.atcab.atcab_aes_ccm_encrypt_update (
    ctx,
    plaintext,
    plaintext_size,
    ciphertext )
```

Process data using CCM mode and a key within the ATECC608A device. `atcab_aes_ccm_init()` or `atcab_aes_ccm_init_rand()` should be called before the first use of this function.

Args:

<code>ctx</code>	AES CCM context structure.
<code>plaintext</code>	Data to be processed.
<code>plaintext_size</code>	Size of the data to be processed.
<code>ciphertext</code>	Output data is returned here.

22.2.2.14 atcab_aes_ccm_init()

```
def cryptoauthlib.atcab.atcab_aes_ccm_init (
    ctx,
    key_id,
    key_block,
    iv,
    iv_size,
    aad_size,
    text_size,
    tag_size )
```

Initialize context for AES CCM operation with an existing IV, which is common when starting a decrypt operation.

Args:

<code>ctx</code>	AES CCM context to be initialized
<code>key_id</code>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
<code>key_block</code>	Index of the 16-byte block to use within the key location for the actual key.
<code>iv</code>	Nonce to be fed into the AES CCM calculation.
<code>iv_size</code>	Size of iv.
<code>aad_size</code>	Size of Additional authentication data.
<code>text_size</code>	Size of plaintext/ciphertext to be processed.
<code>tag_size</code>	Preferred size of tag.

22.2.2.15 atcab_aes_ccm_init_rand()

```
def cryptoauthlib.atcab.atcab_aes_ccm_init_rand (
    ctx,
    key_id,
    key_block,
    iv,
    iv_size,
    aad_size,
    text_size,
    tag_size )
```

Initialize context for AES CCM operation with a random nonce

Args:

<code>ctx</code>	AES CCM context to be initialized
<code>key_id</code>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
<code>key_block</code>	Index of the 16-byte block to use within the key location for the actual key.
<code>iv</code>	Nonce to be fed into the AES CCM calculation.
<code>iv_size</code>	Size of iv.
<code>aad_size</code>	Size of Additional authentication data.
<code>text_size</code>	Size of plaintext/ciphertext to be processed.
<code>tag_size</code>	Preferred size of tag.

22.2.2.16 atcab_aes_cmac_finish()

```
def cryptoauthlib.atcab.atcab_aes_cmac_finish (
    ctx,
    cmac,
    size )
```

Finish a CMAC operation returning the CMAC value.

Args:

<code>ctx</code>	AES-128 CMAC context.
<code>cmac</code>	CMAC is returned here.
<code>cmac_size</code>	Size of CMAC requested in bytes (max 16 bytes).

Returns:

Status code

22.2.2.17 atcab_aes_cmac_init()

```
def cryptoauthlib.atcab.atcab_aes_cmac_init (
    ctx,
    key_id,
    key_block )
```

Initialize a CMAC calculation using an AES-128 key in the ATECC608.

Args:

<code>ctx</code>	AES-128 CMAC context.
<code>key_id</code>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
<code>key_block</code>	Index of the 16-byte block to use within the key location for the actual key.

Returns:

Status code

22.2.2.18 atcab_aes_cmac_update()

```
def cryptoauthlib.atcab.atcab_aes_cmac_update (
    ctx,
    data,
    data_size )
```

Add data to an initialized CMAC calculation.

Args:

ctx	AES-128 CMAC context.
data	Data to be added.
data_size	Size of the data to be added in bytes.

Returns:

Status code

22.2.2.19 atcab_aes_ctr_decrypt_block()

```
def cryptoauthlib.atcab.atcab_aes_ctr_decrypt_block (
    ctx,
    ciphertext,
    plaintext )
```

Decrypt a block of data using CTR mode and a key within the ATECC608 device. `atcab_aes_ctr_init()` or `atcab_aes_ctr_init_rand()` should be called before the first use of this function.

Args:

ctx	AES CTR context structure.
ciphertext	Ciphertext to be decrypted (16 bytes).
plaintext	Decrypted data is returned here (16 bytes).

Returns:

Status code

22.2.2.20 atcab_aes_ctr_encrypt_block()

```
def cryptoauthlib.atcab.atcab_aes_ctr_encrypt_block (
    ctx,
    plaintext,
    ciphertext )
```

Encrypt a block of data using CTR mode and a key within the ATECC608 device. `atcab_aes_ctr_init()` or `atcab_aes_ctr_init_rand()` should be called before the first use of this function.

Args:

ctx	AES CTR context structure.
plaintext	Plaintext to be encrypted (16 bytes).
ciphertext	Encrypted data is returned here (16 bytes).

Returns:

Status code

22.2.2.21 atcab_aes_ctr_init()

```
def cryptoauthlib.atcab.atcab_aes_ctr_init (
    ctx,
    key_id,
    key_block,
    counter_size,
    iv )
```

Initialize context for AES CTR operation with an existing IV, which is common when start a decrypt operation.

The IV is a combination of nonce (left-field) and big-endian counter (right-field). The counter_size field sets the size of the counter and the remaining bytes are assumed to be the nonce.

Args:

ctx	AES CTR context to be initialized.
key_id	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
key_block	Index of the 16-byte block to use within the key location for the actual key.
counter_size	Size of counter in IV in bytes. 4 bytes is a common size.
iv	Initialization vector (concatenation of nonce and counter) 16 bytes.

Returns:

ATCA_SUCCESS on success, otherwise an error code.

22.2.2.22 atcab_aes_ctr_init_rand()

```
def cryptoauthlib.atcab.atcab_aes_ctr_init_rand (
    ctx,
    key_id,
    key_block,
    counter_size,
    iv )
```

Initialize context for AES CTR operation with a random nonce and counter set to 0 as the IV, which is common when starting an encrypt operation.

The IV is a combination of nonce (left-field) and big-endian counter (right-field). The counter_size field sets the size of the counter and the remaining bytes are assumed to be the nonce.

Args:

ctx	AES CTR context to be initialized.
key_id	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
key_block	Index of the 16-byte block to use within the key location for the actual key.
counter_size	Size of counter in IV in bytes. 4 bytes is a common size.
iv	Initialization vector (concatenation of nonce and counter) is returned here (16 bytes).

Returns:

ATCA_SUCCESS on success, otherwise an error code.

22.2.2.23 atcab_aes_decrypt()

```
def cryptoauthlib.atcab.atcab_aes_decrypt (
    key_id,
    key_block,
    ciphertext,
    plaintext )
```

Perform an AES-128 decrypt operation with a key in the device.

Args:

key_id	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey. (int)
key_block	Index of the 16-byte block to use within the key location for the actual key. (int)
ciphertext	Input ciphertext to be decrypted (16 bytes). (bytearray or bytes)
plaintext	Output plaintext is returned here (16 bytes). (Expects bytearray of size 16)s

Returns:

Status Code

22.2.2.24 atcab_aes_encrypt()

```
def cryptoauthlib.atcab.atcab_aes_encrypt (
    key_id,
    key_block,
    plaintext,
    ciphertext )
```

Perform an AES-128 encrypt operation with a key in the device.

Args:

key_id	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey. (int)
key_block	Index of the 16-byte block to use within the key location for the actual key. (int)
plaintext	Input plaintext to be encrypted (16 bytes). (Can be of type bytearray or bytes)
ciphertext	Output ciphertext is returned here (16 bytes). (Expects bytearray of size 16)

Returns:

Status Code

22.2.2.25 atcab_aes_gcm_aad_update()

```
def cryptoauthlib.atcab.atcab_aes_gcm_aad_update (
    ctx,
    aad,
    aad_size )
```

Process Additional Authenticated Data (AAD) using GCM mode and a key within the ATECC608 device.

This can be called multiple times. `atcab_aes_gcm_init()` or `atcab_aes_gcm_init_rand()` should be called before the first use of this function. When there is AAD to include, this should be called before `atcab_aes_gcm_encrypt_update()` or `atcab_aes_gcm_decrypt_update()`.

Args:

<code>ctx</code>	AES GCM context
<code>aad</code>	Additional authenticated data to be added
<code>aad_size</code>	Size of aad in bytes

Returns:

ATCA_SUCCESS on success, otherwise an error code.

22.2.2.26 `atcab_aes_gcm_decrypt_finish()`

```
def cryptoauthlib.atcab.atcab_aes_gcm_decrypt_finish (
    ctx,
    tag,
    tag_size,
    is_verified )
```

Complete a GCM decrypt operation verifying the authentication tag.

Args:

<code>ctx</code>	AES GCM context structure.
<code>tag</code>	Expected authentication tag.
<code>tag_size</code>	Size of tag in bytes (12 to 16 bytes).
<code>is_verified</code>	Returns whether or not the tag verified.

Returns:

ATCA_SUCCESS on success, otherwise an error code.

22.2.2.27 `atcab_aes_gcm_decrypt_update()`

```
def cryptoauthlib.atcab.atcab_aes_gcm_decrypt_update (
    ctx,
    ciphertext,
    ciphertext_size,
    plaintext )
```

Decrypt data using GCM mode and a key within the ATECC608 device. `atcab_aes_gcm_init()` or `atcab_aes_gcm_init_rand()` should be called before the first use of this function.

Args:

<code>ctx</code>	AES GCM context structure.
<code>ciphertext</code>	Ciphertext to be decrypted.
<code>ciphertext_size</code>	Size of ciphertext in bytes.
<code>plaintext</code>	Decrypted data is returned here.

Returns:

ATCA_SUCCESS on success, otherwise an error code.

22.2.2.28 atcab_aes_gcm_encrypt_finish()

```
def cryptoauthlib.atcab.atcab_aes_gcm_encrypt_finish (
    ctx,
    tag,
    tag_size )
```

Complete a GCM encrypt operation returning the authentication tag.

Args:

ctx	AES GCM context structure.
tag	Authentication tag is returned here.
tag_size	Tag size in bytes (12 to 16 bytes).

Returns:

ATCA_SUCCESS on success, otherwise an error code.

22.2.2.29 atcab_aes_gcm_encrypt_update()

```
def cryptoauthlib.atcab.atcab_aes_gcm_encrypt_update (
    ctx,
    plaintext,
    plaintext_size,
    ciphertext )
```

Encrypt data using GCM mode and a key within the ATECC608 device. atcab_aes_gcm_init() or atcab_aes_gcm_init_rand() should be called before the first use of this function.

Args:

ctx	AES GCM context structure.
plaintext	Plaintext to be encrypted (16 bytes).
plaintext_size	Size of plaintext in bytes.
ciphertext	Encrypted data is returned here.

Returns:

ATCA_SUCCESS on success, otherwise an error code.

22.2.2.30 atcab_aes_gcm_init()

```
def cryptoauthlib.atcab.atcab_aes_gcm_init (
    ctx,
    key_id,
    key_block,
    iv,
    iv_size )
```

Initialize context for AES GCM operation with an existing IV, which is common when starting a decrypt operation.

Args:

ctx	AES GCM context to be initialized.
key_id	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
key_block	Index of the 16-byte block to use within the key location for the actual key.
iv	Initialization vector.
iv_size	Size of IV in bytes. Standard is 12 bytes.

Returns:

ATCA_SUCCESS on success, otherwise an error code.

22.2.2.31 atcab_aes_gcm_init_rand()

```
def cryptoauthlib.atcab.atcab_aes_gcm_init_rand (
    ctx,
    key_id,
    key_block,
    rand_size,
    free_field,
    free_field_size,
    iv )
```

Initialize context for AES GCM operation with a IV composed of a random and optional fixed(free) field, which is common when starting an encrypt operation.

Args:

ctx	AES CTR context to be initialized.
key_id	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
key_block	Index of the 16-byte block to use within the key location for the actual key.
rand_size	Size of the random field in bytes. Minimum and recommended size is 12 bytes. Max is 32 bytes.
free_field	Fixed data to include in the IV after the random field. Can be NULL if not used.
free_field_size	Size of the free field in bytes.
iv	Initialization vector is returned here. Its size will be rand_size and free_field_size combined.

Returns:

ATCA_SUCCESS on success, otherwise an error code.

22.2.2.32 atcab_aes_gfm()

```
def cryptoauthlib.atcab.atcab_aes_gfm (
    hash_key,
    inp,
    output )
```

Perform a Galois Field Multiply (GFM) operation.

Args:

hash_key	First input value (16 bytes). (bytearray or bytes)
inp	Second input value (16 bytes). (bytearray or bytes)
output	GFM result is returned here (16 bytes). (Expects bytearray of size 16)

Returns:

Status Code

22.2.2.33 atcab_challenge()

```
def cryptoauthlib.atcab.atcab_challenge (
    num_in )
```

Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.

Args:

num_in	Data to be loaded into TempKey (32 bytes). (bytearray or bytes)
--------	--

Returns:

Status Code

22.2.2.34 atcab_challenge_seed_update()

```
def cryptoauthlib.atcab.atcab_challenge_seed_update (
    num_in,
    rand_out )
```

Execute a Nonce command to generate a random challenge combining a host nonce (num_in) and a device random number.

Args:

num_in	Host nonce to be combined with the device random number (20 bytes). (bytearray or bytes)
rand_out	Internally generated 32-byte random number that was used in the nonce/challenge calculation is returned here. Can be NULL if not needed. (Expects bytearray)

Returns:

Status code

22.2.2.35 atcab_checkmac()

```
def cryptoauthlib.atcab.atcab_checkmac (
    mode,
    key_id,
    challenge,
    response,
    other_data )
```

Compares a MAC response with input values

Args:

mode	Controls which fields within the device are used in the message (int)
key_id	Key location in the CryptoAuth device to use for the MAC (int)
challenge	Challenge data (32 bytes) (bytearray or bytes)
response	MAC response data (32 bytes) (bytearray or bytes)
other_data	OtherData parameter (13 bytes) (bytearray or bytes)

Returns:

Status code

22.2.2.36 atcab_cmp_config_zone()

```
def cryptoauthlib.atcab.atcab_cmp_config_zone (
    config_data,
    same_config )
```

Compares a specified configuration zone with the configuration zone currently on the device.

This only compares the static portions of the configuration zone and skips those that are unique per device (first 16 bytes) and areas that can change after the configuration zone has been locked (e.g. LastKeyUse).

Args:

config_data	Full configuration data to compare the device against. (bytearray or bytes)
same_config	Result is returned here. True if the static portions on the configuration zones are the same. (Expects AtcaReference)

Returns:

Status code

22.2.2.37 atcab_counter()

```
def cryptoauthlib.atcab.atcab_counter (
    mode,
    counter_id,
    counter_value )
```

Compute the Counter functions

Args:

mode	The mode used for the counter (int)
counter_id	The counter to be used (int)
counter_value	Counter value returned from device (AtcaReference expected)

Returns:

Status code

22.2.2.38 atcab_counter_increment()

```
def cryptoauthlib.atcab.atcab_counter_increment (
    counter_id,
    counter_value )
```

Increments one of the device's monotonic counters

Args:

counter_id	Counter to be incremented (int)
counter_value	New value of the counter is returned here (AtcaReference expected)

Returns:

Status code

22.2.2.39 atcab_counter_read()

```
def cryptoauthlib.atcab.atcab_counter_read (
    counter_id,
    counter_value )
```

Reads one of the device's monotonic counters

Args:

counter_id	Counter to be read (int)
counter_value	Counter value is returned here (AtcaReference expected)

Returns:

Status code

22.2.2.40 atcab_derivekey()

```
def cryptoauthlib.atcab.atcab_derivekey (
    mode,
    target_key,
    mac )
```

Executes the DeviveKey command for deriving a new key from a nonce (TempKey) and an existing key.

Args:

mode	Bit 2 must match the value in TempKey.SourceFlag (int)
target_key	Key slot to be written (int)
mac	Optional 32 byte MAC used to validate operation. (bytearray or bytes)

Returns:

Status code

22.2.2.41 atcab_ecdh()

```
def cryptoauthlib.atcab.atcab_ecdh (
    key_id,
    public_key,
    pms )
```

ECDH command with a private key in a slot and the premaster secret is returned in the clear.

Args:

key_id	Slot of key for ECDH computation (int)
public_key	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key. (bytearray or bytes)
pms	ByteArray - Computed ECDH premaster secret is returned here (32 bytes). (Expects bytearray of size 32)

Returns:

Status code

22.2.2.42 atcab_ecdh_base()

```
def cryptoauthlib.atcab.atcab_ecdh_base (
    mode,
    key_id,
    public_key,
    pms,
    out_nonce )
```

Base function for generating premaster secret key using ECDH.

Args:

mode	Mode to be used for ECDH computation (int)
key_id	Slot of key for ECDH computation (int)
public_key	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key. (bytearray or bytes)
pms	ByteArray - Computed ECDH pre-master secret is returned here (32 bytes) if returned directly. Otherwise NULL.
out_nonce	ByteArray - Nonce used to encrypt pre-master secret. NULL if output encryption not used.

Returns:

Status code

22.2.2.43 atcab_ecdh_enc()

```
def cryptoauthlib.atcab.atcab_ecdh_enc (
    key_id,
    public_key,
    pms,
    read_key,
    read_key_id,
    num_in = None )
```

ECDH command with a private key in a slot and the premaster secret is read from the next slot. This function only works for even numbered slots with the proper configuration.

Args:

key_id	Slot of key for ECDH computation (int)
public_key	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key. (bytearray or bytes)
read_key	Read key for the premaster secret slot (key_id 1) (32 bytes). (bytearray or bytes)
read_key_id	Read key slot for read_key. (int)
pms	ByteArray - Computed ECDH premaster secret is returned here (32 bytes). (Expects bytearray of size 32)
num_in	Bytearray - Host nonce used to calculate nonce (20 bytes)

Returns:

Status code

22.2.2.44 atcab_ecdh_ioenc()

```
def cryptoauthlib.atcab.atcab_ecdh_ioenc (
    key_id,
    public_key,
    pms,
    io_key )
```

ECDH command with a private key in a slot and the premaster secret is returned encrypted using the IO protection key.

Args:

key_id	Slot of key for ECDH computation (int)
public_key	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key. (bytearray or bytes)
io_key	IO protection key (32 bytes). (bytearray or bytes)
pms	Computed ECDH premaster secret is returned here (32 bytes). (Expects bytearray of size 32)

Returns:

Status code

22.2.2.45 atcab_ecdh_tempkey()

```
def cryptoauthlib.atcab.atcab_ecdh_tempkey (
    public_key,
    pms )
```

ECDH command with a private key in TempKey and the premaster secret is returned in the clear.

Args:

public_key	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key. (bytearray or bytes)
pms	Computed ECDH premaster secret is returned here (32 bytes). (Expects bytearray of size 32)

Retuns:

Status code

22.2.2.46 atcab_ecdh_tempkey_ioenc()

```
def cryptoauthlib.atcab.atcab_ecdh_tempkey_ioenc (
    public_key,
    pms,
    io_key )
```

ECDH command with a private key in TempKey and the premaster secret is returned encrypted using the IO protection key.

Args:

<code>public_key</code>	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key. (bytearray or bytes)
<code>io_key</code>	IO protection key (32 bytes). (bytearray or bytes)
<code>pms</code>	Computed ECDH premaster secret is returned here (32 bytes). (Expects bytearray of size 32)

Returns:

Status code

22.2.2.47 atcab_gendig()

```
def cryptoauthlib.atcab.atcab_gendig (
    zone,
    key_id,
    other_data,
    other_data_size )
```

Issues a GenDig command, which performs a SHA256 hash on the source data indicated by zone with the contents of TempKey. See the CryptoAuth datasheet for your chip to see what the values of zone correspond to.

Args:

<code>zone</code>	Designates the source of the data to hash with TempKey. (int)
<code>key_id</code>	Indicates the key, OTP block, or message order for shared nonce mode. (int)
<code>other_data</code>	Four bytes of data for SHA calculation when using a NoMac key, 32 bytes for "Shared Nonce" mode, otherwise ignored (can be NULL). (bytearray or bytes)
<code>other_data_size</code>	Size of other_data in bytes. (int)

Returns:

Status code

22.2.2.48 atcab_genkey()

```
def cryptoauthlib.atcab.atcab_genkey (
    key_id,
    public_key )
```

Issues GenKey command, which generates a new random private key in slot and returns the public key.

Args:

<code>key_id</code>	Slot number where an ECC private key is configured. Can also be ATCA_TEMPKEY_KEYID to generate a private key in TempKey. (int)
<code>public_key</code>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve. Set to NULL if public key isn't required. (Expects bytearray)

Returns:

Status code

22.2.2.49 atcab_genkey_base()

```
def cryptoauthlib.atcab.atcab_genkey_base (
    mode,
    key_id,
    other_data,
    public_key = None )
```

Issues GenKey command, which can generate a private key, compute a public key, nd/or compute a digest of a public key.

Args:

mode	Mode determines what operations the GenKey command performs. (int)
key_id	Slot to perform the GenKey command on. (int)
other_data	OtherData for PubKey digest calculation. Can be set to NULL otherwise. (bytearray or bytes)
public_key	If the mode indicates a public key will be calculated, it will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve. Set to NULL if public key isn't required. (Expects bytearray of size 64 bytes)

Returns:

Status code

22.2.2.50 atcab_get_device()

```
def cryptoauthlib.atcab.atcab_get_device (
    void )
```

Return the global device instance

22.2.2.51 atcab_get_device_type()

```
def cryptoauthlib.atcab.atcab_get_device_type (
    void )
```

Return the device type of the currently initialized device.

22.2.2.52 atcab_get_pubkey()

```
def cryptoauthlib.atcab.atcab_get_pubkey (
    key_id,
    public_key )
```

Uses GenKey command to calculate the public key from an existing private key in a slot.

Args:

key_id	Slot number of the private key. (int)
public_key	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve. Set to NULL if public key isn't required. (Expects bytearray)

Returns:

Status code

22.2.2.53 atcab_hmac()

```
def cryptoauthlib.atcab.atcab_hmac (
    mode,
    key_id,
    digest )
```

Issues a HMAC command, which computes an HMAC/SHA-256 digest of a key stored in the device, a challenge, and other information on the device.

Args:

mode	Controls which fields within the device are used in the message. (int)
key_id	Which key is to be used to generate the response. Bits 0:3 only are used to select a slot but all 16 bits are used in the HMAC message. (int)
digest	HMAC digest is returned in this buffer (32 bytes). (Expects bytearray)

Returns:

Status code

22.2.2.54 atcab_hw_sha2_256()

```
def cryptoauthlib.atcab.atcab_hw_sha2_256 (
    data,
    data_size,
    digest )
```

Use the SHA command to compute a SHA-256 digest.

Args:

data	Message data to be hashed. (bytearray or bytes)
data_size	Size of data in bytes. (int)
digest	Digest is returned here (32 bytes). (Expects bytearray)

Returns:

Status code

22.2.2.55 atcab_hw_sha2_256_finish()

```
def cryptoauthlib.atcab.atcab_hw_sha2_256_finish (
    ctx,
    digest )
```

Finish SHA-256 digest for a SHA context for performing a hardware SHA-256 operation on a device.

Args:

ctx	SHA256 context (atca_sha256_ctx)
digest	SHA256 digest is returned here (32 bytes) (Expects bytearray)

Returns:

Status code

22.2.2.56 atcab_hw_sha2_256_init()

```
def cryptoauthlib.atcab.atcab_hw_sha2_256_init (
    ctx )
```

Initialize a SHA context for performing a hardware SHA-256 operation on a device. Note that only one SHA operation can be run at a time.

Args:

ctx	SHA256 context (atca_sha256_ctx)
-----	----------------------------------

Returns:

Status code

22.2.2.57 atcab_hw_sha2_256_update()

```
def cryptoauthlib.atcab.atcab_hw_sha2_256_update (
    ctx,
    data,
    data_size )
```

--> Add message data to a SHA context for performing a hardware SHA-256 operation on a device.

Args:

ctx	SHA256 context (atca_sha256_ctx)
data	Message data to be added to hash. (bytearray or bytes)
data_size	Size of data in bytes. (int)

Returns:

Status code

22.2.2.58 atcab_info()

```
def cryptauthlib.atcab.atcab_info (
    revision )
```

Used to get the device revision number. (DevRev)

Args:

revision	4-byte bytearray receiving the revision number from the device. (Expects bytearray)
----------	---

Returns:

Status code

22.2.2.59 atcab_info_base()

```
def cryptauthlib.atcab.atcab_info_base (
    mode,
    param2,
    out_data )
```

Issues an Info command, which return internal device information and can control GPIO and the persistent latch.

Args:

mode	Selects which mode to be used for info command.(int)
param2	Selects the particular fields for the mode.(int)
out_data	Response from info command (4 bytes). Can be set to NULL if not required.(Expects bytearray)

Returns:

Status

22.2.2.60 atcab_info_get_latch()

```
def cryptauthlib.atcab.atcab_info_get_latch (
    state )
```

Using the Info command to get the persistent latch current state for an ATECC608 device.

Args:

state	The state is returned here. Set (True) or clear (False). Expects AtcaReference.
-------	---

Returns:

Status code

22.2.2.61 atcab_info_set_latch()

```
def cryptoauthlib.atcab.atcab_info_set_latch (
    state )
```

Use the Info command to set the persistent latch state for an ATECC608 device.

Args:

state	Persistent latch state. Set (True) or clear (False).
-------	--

Returns:

Status code

22.2.2.62 atcab_init()

```
def cryptoauthlib.atcab.atcab_init (
    iface_cfg )
```

Initialize the communication stack and initializes the ATCK590 kit
Communication over USB HID and Kit Protocol by default
raise CryptoException

22.2.2.63 atcab_is_locked()

```
def cryptoauthlib.atcab.atcab_is_locked (
    zone,
    is_locked )
```

Executes Read command, which reads the configuration zone to see if the specified slot is locked.

Args:

zone	The zone to query for locked (use LOCK_ZONE_CONFIG(0x00) or LOCK_ZONE_DATA(0x01)). (int)
is_locked	Lock state returned here. True if locked. (Expects AtcaReference)

Returns:

Status code

22.2.2.64 atcab_is_slot_locked()

```
def cryptoauthlib.atcab.atcab_is_slot_locked (
    slot,
    is_locked )
```

Executes Read command, which reads the configuration zone to see if the specified slot is locked.

Args:

slot	Slot to query for locked (slot 0-15) (int)
is_locked	Lock state returned here. True if locked. (Expects AtcaReference)

Returns:

Status code

22.2.2.65 atcab_kdf()

```
def cryptoauthlib.atcab.atcab_kdf (
    mode,
    key_id,
    details,
    message,
    out_data,
    out_nonce )
```

Executes the KDF command, which derives a new key in PRF, AES, or HKDF modes. Generally this function combines a source key with an input string and creates a result key/digest/array.

Args:

mode	Mode determines KDF algorithm (PRF,AES,HKDF), source key location, and target key locations. (int)
key_id	Source and target key slots if locations are in the EEPROM. Source key slot is the LSB and target key slot is the MSB. (int)
details	Further information about the computation, depending on the algorithm. (int)
message	Input value from system (up to 128 bytes). Actual size of message is 16 bytes for AES algorithm or is encoded in the MSB of the details parameter for other algorithms. (bytearray or bytes)
out_data	Output of the KDF function is returned here. If the result remains in the device, this can be NULL. (Expects bytearray)
out_nonce	If the output is encrypted, a 32 byte random nonce generated by the device is returned here. If output encryption is not used, this can be NULL. (Expects bytearray)

Returns:

Status code

22.2.2.66 atcab_lock()

```
def cryptoauthlib.atcab.atcab_lock (
    mode,
    summary_crc )
```

The Lock command prevents future modifications of the Configuration and/or Data and OTP zones. If the device is so configured, then this command can be used to lock individual data slots. This command fails if the designated area is already locked.

Args:

mode	Zone, and/or slot, and summary check (bit 7).(int)
summary_crc	CRC of the config or data zones. Ignored for slot locks or when mode bit 7 is set. (int)

Returns:

Status code

22.2.2.67 atcab_lock_config_zone()

```
def cryptoauthlib.atcab.atcab_lock_config_zone (
    void )
```

Unconditionally (no CRC required) lock the config zone.

Args:

None

Returns:

Status code

22.2.2.68 atcab_lock_config_zone_crc()

```
def cryptoauthlib.atcab.atcab_lock_config_zone_crc (
    summary_crc )
```

Lock the config zone with summary CRC.

The CRC is calculated over the entire config zone contents. 88 bytes for ATSHA devices, 128 bytes for ATECC devices. Lock will fail if the provided CRC doesn't match the internally calculated one.

Args:

summary_crc	Expected CRC over the config zone. (int)
-------------	--

Returns:

Status code

22.2.2.69 atcab_lock_data_slot()

```
def cryptoauthlib.atcab.atcab_lock_data_slot (  
    slot )
```

Lock an individual slot in the data zone on an ATECC device. Not available for ATSHA devices. Slot must be configured to be slot lockable (KeyConfig.Lockable=1).

Args:
slot Slot to be locked in data zone. (int)

Returns:
Status code

22.2.2.70 atcab_lock_data_zone()

```
def cryptoauthlib.atcab.atcab_lock_data_zone (  
    void )
```

Unconditionally (no CRC required) lock the data zone (slots and OTP).

ConfigZone must be locked and DataZone must be unlocked for the zone to be successfully locked.

Args:
None

Returns:
Status code

22.2.2.71 atcab_lock_data_zone_crc()

```
def cryptoauthlib.atcab.atcab_lock_data_zone_crc (  
    summary_crc )
```

Lock the data zone (slots and OTP) with summary CRC.

The CRC is calculated over the concatenated contents of all the slots and OTP at the end. Private keys (KeyConfig.Private=1) are skipped. Lock will fail if the provided CRC doesn't match the internally calculated one.

Args:
summary_crc Expected CRC over the config zone. (int)

Returns:
Status code

22.2.2.72 atcab_mac()

```
def cryptoauthlib.atcab.atcab_mac (
    mode,
    key_id,
    challenge,
    digest )
```

Executes MAC command, which computes a SHA-256 digest of a key stored in the device, a challenge, and other information on the device.

Args:

mode	Controls which fields within the device are used in the message (int)
key_id	Key in the CryptoAuth device to use for the MAC (int)
challenge	Challenge message (32 bytes). May be NULL if mode indicates a challenge isn't required. (bytearray or bytes)
digest	MAC response is returned here (32 bytes). (Expects bytearray)

Returns:

Status code

22.2.2.73 atcab_nonce()

```
def cryptoauthlib.atcab.atcab_nonce (
    num_in )
```

Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.

Args:

num_in	Data to be loaded into TempKey (32 bytes). (bytearray or bytes)
--------	---

Returns:

None

22.2.2.74 atcab_nonce_base()

```
def cryptoauthlib.atcab.atcab_nonce_base (
    mode,
    zero,
    num_in,
    rand_out )
```


Executes Nonce command, which loads a random or fixed nonce/data into the device for use by subsequent commands.

Args:

mode	Controls the mechanism of the internal RNG or fixed write. (int)
zero	Param2, normally 0, but can be used to indicate a nonce calculation mode (bit 15). (int)
num_in	Input value to either be included in the nonce calculation in random modes (20 bytes) or to be written directly (32 bytes or 64 bytes (ATECC608)) in pass-through mode. (bytearray or bytes)
rand_out	If using a random mode, the internally generated 32-byte random number that was used in the nonce calculation is returned here. Can be NULL if not needed. (Expects bytearray)

Returns:

Status code

22.2.2.75 atcab_nonce_load()

```
def cryptoauthlib.atcab.atcab_nonce_load (
    target,
    num_in,
    num_in_size )
```

Execute a Nonce command in pass-through mode to load one of the device's internal buffers with a fixed value.

For the ATECC608, available targets are TempKey (32 or 64 bytes), Message Digest Buffer (32 or 64 bytes), or the Alternate Key Buffer (32 bytes). For all other devices, only TempKey (32 bytes) is available.

Args:

target	Target device buffer to load. Can be NONCE_MODE_TARGET_TEMPKEY, NONCE_MODE_TARGET_MSGDIGBUF, or NONCE_MODE_TARGET_ALTKEYBUF. (int)
num_in	Data to load into the buffer. (bytearray or bytes)
num_in_size	Size of num_in in bytes. Can be 32 or 64 bytes depending on device and target. (int)

Returns:

Status code

22.2.2.76 atcab_nonce_rand()

```
def cryptoauthlib.atcab.atcab_nonce_rand (
    num_in,
    rand_out )
```

22.2 cryptoauthlib.atcab Namespace Reference

Execute a Nonce command to generate a random nonce combining a host nonce (num_in) and a device random number.

Args:

num_in	Host nonce to be combined with the device random number (20 bytes). (bytearray or bytes)
rand_out	Internally generated 32-byte random number that was used in the nonce/challenge calculation is returned here. Can be NULL if not needed. (Expects bytearray)

Returns:

Status code

22.2.2.77 atcab_priv_write()

```
def cryptoauthlib.atcab.atcab_priv_write (
    key_id,
    priv_key,
    write_key_id,
    write_key,
    num_in = None )
```

Executes PrivWrite command, to write externally generated ECC private keys into the device.

Args:

key_id	Slot to write the external private key into. (int)
priv_key	External private key (36 bytes) to be written. The first 4 bytes should be zero for P256 curve. (bytearray or bytes)
write_key_id	Write key slot. Ignored if write_key is NULL. (int)
write_key	Write key (32 bytes). If NULL, perform an unencrypted PrivWrite, which is only available when the data zone is unlocked. (bytearray or bytes)
num_in	Bytearray - Host nonce used to calculate nonce (20 bytes)

Returns:

Status code

22.2.2.78 atcab_random()

```
def cryptoauthlib.atcab.atcab_random (
    random_number )
```

Generates a 32 byte random number. Note that if the configuration zone isn't locked yet (LockConfig) then it will return a 0xFFFF0000 repeating pattern instead.

Args:

random_number	Random number is returned here (expects bytearray)
---------------	--

Returns:

Status code

22.2.2.79 atcab_read_bytes_zone()

```
def cryptoauthlib.atcab.atcab_read_bytes_zone (
    zone,
    slot,
    offset,
    data,
    length )
```

Used to read an arbitrary number of bytes from any zone configured for clear reads.

This function will issue the Read command as many times as is required to read the requested data.

Args:

zone	Zone to read data from. Option are ATCA_ZONE_CONFIG(0), ATCA_ZONE_OTP(1), or ATCA_ZONE_DATA(2). (int)
slot	Slot number to read from if zone is ATCA_ZONE_DATA(2). Ignored for all other zones. (int)
offset	Byte offset within the zone to read from. (int)
length	Number of bytes to read starting from the offset. (int)
data	Read data is returned here. (Expects bytearray)

Returns:

Status code

22.2.2.80 atcab_read_config_zone()

```
def cryptoauthlib.atcab.atcab_read_config_zone (
    config_data )
```

Executes Read command to read the complete device configuration zone.

Args:

config_data	Configuration zone data is returned here. 88 bytes for ATSHA devices, 128 bytes for ATECC devices. (Expects bytearray)
-------------	--

Returns:

Status code

22.2.2.81 atcab_read_enc()

```
def cryptoauthlib.atcab.atcab_read_enc (
    key_id,
    block,
    data,
    enc_key,
    enc_key_id,
    num_in = None )
```

22.2 cryptoauthlib.atcab Namespace Reference

Executes Read command on a slot configured for encrypted reads and decrypts the data to return it as plaintext.

Data zone must be locked for this command to succeed. Can only read 32 byte blocks.

Args:

key_id	The slot ID to read from. (int)
block	Index of the 32 byte block within the slot to read. (int)
enc_key	32 byte ReadKey for the slot being read. (bytearray or bytes)
enc_key_id	KeyID of the ReadKey being used. (int)
data	Decrypted (plaintext) data from the read is returned here (32 bytes). (Expects bytearray)
num_in	Bytearray - Host nonce used to calculate nonce (20 byte)

Returns:

Status code

22.2.2.82 atcab_read_pubkey()

```
def cryptoauthlib.atcab.atcab_read_pubkey (
    slot,
    public_key )
```

Executes Read command to read an ECC P256 public key from a slot configured for clear reads.

This function assumes the public key is stored using the ECC public key format specified in the datasheet.

Args:

slot	Slot number to read from. Only slots 8 to 15 are large enough for a public key. (int)
public_key	Public key is returned here (64 bytes). Format will be the 32 byte X and Y big-endian integers concatenated. (Expects bytearray)

Returns:

Status code

22.2.2.83 atcab_read_serial_number()

```
def cryptoauthlib.atcab.atcab_read_serial_number (
    serial_number )
```

Executes Read command, which reads the 9 byte serial number of the device from the config zone.

Args:

serial_number	9 byte serial number is returned here. (Expects bytearray)
---------------	--

Returns:

Status code

22.2.2.84 atcab_read_sig()

```
def cryptoauthlib.atcab.atcab_read_sig (
    slot,
    sig )
```

Executes Read command to read a 64 byte ECDSA P256 signature from a slot configured for clear reads.

Args:

slot	Slot number to read from. Only slots 8 to 15 are large enough for a signature. (int)
sig	Signature will be returned here (64 bytes). Format will be the 32 byte R and S big-endian integers concatenated. (Expects bytearray)

Returns:

Status code

22.2.2.85 atcab_read_zone()

```
def cryptoauthlib.atcab.atcab_read_zone (
    zone,
    slot,
    block,
    offset,
    data,
    length )
```

Executes Read command, which reads either 4 or 32 bytes of data from a given slot, configuration zone, or the OTP zone.

When reading a slot or OTP, data zone must be locked and the slot configuration must not be secret for a slot to be successfully read.

Args:

zone	Zone to be read from device. Options are ATCA_ZONE_CONFIG, ATCA_ZONE_OTP, or ATCA_ZONE_DATA. (int)
slot	Slot number for data zone and ignored for other zones. (int)
block	32 byte block index within the zone. (int)
offset	4 byte work index within the block. Ignored for 32 byte reads. (Expects bytearray)
length	Length of the data to be read. Must be either 4 or 32.
data	Read data is returned here. (Expects bytearray)

Returns:

Status code

22.2.2.86 atcab_release()

```
def cryptoauthlib.atcab.atcab_release (
    void )
```

Release the kit and the communication stack
raise CryptoException

22.2.2.87 atcab_secureboot()

```
def cryptoauthlib.atcab.atcab_secureboot (
    mode,
    param2,
    digest,
    signature,
    mac )
```

Executes Secure Boot command, which provides support for secure boot of an external MCU or MPU.

Args:

mode	Mode determines what operations the SecureBoot command performs. (int)
param2	Not used, must be 0. (int)
digest	Digest of the code to be verified (32 bytes). (bytearray or bytes)
signature	Signature of the code to be verified (64 bytes). Can be NULL when using the FullStore mode. (bytearray or bytes)
mac	Validating MAC will be returned here (32 bytes). Can be NULL if not required. (Expects bytearray)

Return:

Status code

22.2.2.88 atcab_secureboot_mac()

```
def cryptoauthlib.atcab.atcab_secureboot_mac (
    mode,
    digest,
    signature,
    num_in,
    io_keys,
    is_verified )
```

Executes Secure Boot command with encrypted digest and validated MAC response using the IO protection key.

Args:

mode	Mode determines what operations the SecureBoot command performs. (int)
digest	Digest of the code to be verified (32 bytes). This is the plaintext digest (not encrypted). (bytearray or bytes)
signature	Signature of the code to be verified (64 bytes). Can be NULL when using the FullStore mode. (bytearray or bytes)
num_in	Host nonce (20 bytes). (bytearray or bytes)
io_key	IO protection key (32 bytes). (bytearray or bytes)
is_verified	Verify result is returned here. (Expects AtcaReference)

Returns:

Status code

22.2.2.89 atcab_selftest()

```
def cryptoauthlib.atcab.atcab_selftest (
    mode,
    param2,
    result )
```

Executes the SelfTest command, which performs a test of one or more of the cryptographic engines within the ATECC608 chip.

Args:

mode	Functions to test. Can be a bit field combining any of the following: SELFTEST_MODE_RNG, SELFTEST_MODE_ECDSA_VERIFY, SELFTEST_MODE_ECDSA_SIGN, SELFTEST_MODE_ECDH, SELFTEST_MODE_AES, SELFTEST_MODE_SHA, SELFTEST_MODE_ALL. (int)
param2	Currently unused, should be 0. (int)
result	Results are returned here as a bit field. (Expects AtcaReference)

Returns:

Status code

22.2.2.90 atcab_sha()

```
def cryptoauthlib.atcab.atcab_sha (
    length,
    message,
    digest )
```

Use the SHA command to compute a SHA-256 digest.

Args:

length	Size of message parameter in bytes. (int)
message	Message data to be hashed. (bytearray or bytes)
digest	Digest is returned here (32 bytes). (Expects bytearray)

Returns:

Status code

22.2.2.91 atcab_sha_base()

```
def cryptoauthlib.atcab.atcab_sha_base (
    mode,
    length,
    message,
    data_out,
    data_out_size )
```

22.2 cryptoauthlib.atcab Namespace Reference

Executes SHA command, which computes a SHA-256 or HMAC/SHA-256 digest for general purpose use by the host system.

Only the Start(0) and Compute(1) modes are available for ATSHA devices.

Args:

mode	SHA command mode Start(0), Update/Compute(1), End(2), Public(3), HMACstart(4), HMACend(5), Read_Context(6), or Write_Context(7). Also message digest target location for the ATECC608. (int)
length	Number of bytes in the message parameter or KeySlot for the HMAC key if Mode is HMACstart(4) or Public(3). (int)
message	Message bytes to be hashed or Write_Context if restoring a context on the ATECC608. Can be NULL if not required by the mode. (bytearray or bytes)
data_out	Data returned by the command (digest or context). (Expects bytearray)
data_out_size	As input, the size of the data_out buffer. As output, the number of bytes returned in data_out. (Expects AtcaReference)

Returns:

Status code

22.2.2.92 atcab_sha_end()

```
def cryptoauthlib.atcab.atcab_sha_end (  
    digest,  
    length,  
    message )
```

Executes SHA command to complete SHA-256 or HMAC/SHA-256 operation.

Args:

length	Length of any remaining data to include in hash. Max 64 bytes. (int)
message	Remaining data to include in hash. NULL if length is 0. (bytearray or bytes)
digest	Digest from SHA-256 or HMAC/SHA-256 will be returned here (32 bytes). (Expects bytearray)

Returns:

Status code

22.2.2.93 atcab_sha_hmac()

```
def cryptoauthlib.atcab.atcab_sha_hmac (  
    data,  
    data_size,  
    key_slot,  
    digest,  
    target )
```


Use the SHA command to compute an HMAC/SHA-256 operation.

Args:

<code>data</code>	Message data to be hashed. (bytearray or bytes)
<code>data_size</code>	Size of data in bytes. (int)
<code>key_slot</code>	Slot key id to use for the HMAC calculation (int)
<code>target</code>	Where to save the digest internal to the device. For ATECC608, can be <code>SHA_MODE_TARGET_TEMPKEY</code> , <code>SHA_MODE_TARGET_MSGDIGBUF</code> , or <code>SHA_MODE_TARGET_OUT_ONLY</code> . For all other devices, <code>SHA_MODE_TARGET_TEMPKEY</code> is the only option. (int)
<code>digest</code>	Digest is returned here (32 bytes). (Expects bytearray)

Return:

<code>Status code</code>

22.2.2.94 `atcab_sha_hmac_finish()`

```
def cryptoauthlib.atcab.atcab_sha_hmac_finish (
    ctx,
    digest,
    target )
```

Executes SHA command to complete a HMAC/SHA-256 operation.

Args:

<code>ctx</code>	HMAC/SHA-256 context (<code>atca_hmac_sha256_ctx_t</code>)
<code>target</code>	Where to save the digest internal to the device. For ATECC608, can be <code>SHA_MODE_TARGET_TEMPKEY</code> , <code>SHA_MODE_TARGET_MSGDIGBUF</code> , or <code>SHA_MODE_TARGET_OUT_ONLY</code> . For all other devices, <code>SHA_MODE_TARGET_TEMPKEY</code> is the only option. (int)
<code>digest</code>	HMAC/SHA-256 result is returned here (32 bytes). (Expects bytearray)

Returns:

<code>Status code</code>

22.2.2.95 `atcab_sha_hmac_init()`

```
def cryptoauthlib.atcab.atcab_sha_hmac_init (
    ctx,
    key_slot )
```

Executes SHA command to start an HMAC/SHA-256 operation

Args:

<code>ctx</code>	HMAC/SHA-256 context (<code>atca_hmac_sha256_ctx_t</code>)
<code>key_slot</code>	Slot key id to use for the HMAC calculation (int)

Returns:

<code>Status code</code>

22.2.2.96 atcab_sha_hmac_update()

```
def cryptoauthlib.atcab.atcab_sha_hmac_update (
    ctx,
    data,
    data_size )
```

Executes SHA command to add an arbitrary amount of message data to a HMAC/SHA-256 operation.

Args:

ctx	HMAC/SHA-256 context (atca_hmac_sha256_ctx_t)
data	Message data to add (bytearray or bytes)
data_size	Size of message data in bytes (int)

Returns:

Status code

22.2.2.97 atcab_sha_read_context()

```
def cryptoauthlib.atcab.atcab_sha_read_context (
    context,
    context_size )
```

Executes SHA command to read the SHA-256 context back. Only for ATECC608 with SHA-256 contexts. HMAC not supported.

Args:

context	Context data is returned here. (Expects bytearray)
context_size	As input, the size of the context buffer in bytes. As output, the size of the returned context data. (Expects AtcaReference)

Returns:

Status code

22.2.2.98 atcab_sha_start()

```
def cryptoauthlib.atcab.atcab_sha_start (
    void )
```

Executes SHA command to initialize SHA-256 calculation engine

Args:

None

Returns;

Status code

22.2.2.99 atcab_sha_update()

```
def cryptoauthlib.atcab.atcab_sha_update (
    message )
```

Executes SHA command to add 64 bytes of message data to the current context.

Args:

message	64 bytes of message data to add to add to operation. (Expects bytearray)
---------	---

Returns:

Status code

22.2.2.100 atcab_sha_write_context()

```
def cryptoauthlib.atcab.atcab_sha_write_context (
    context,
    context_size )
```

Executes SHA command to write (restore) a SHA-256 context into the the device. Only supported for ATECC608 with SHA-256 contexts.

Args:

context	Context data to be restored. (bytearray or bytes)
context_size	Size of the context data in bytes. (int)

Returns:

Status code

22.2.2.101 atcab_sign()

```
def cryptoauthlib.atcab.atcab_sign (
    key_id,
    msg,
    signature )
```

Executes Sign command, to sign a 32-byte external message using the private key in the specified slot. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

Args:

key_id	Slot of the private key to be used to sign the message (int)
msg	32-byte message to be signed. Typically the SHA256 hash of the full message. (bytearray or bytes)
signature	Signature will be returned here. Format is R and S integers in big-endian format. 64 bytes for P256 curve. (Expects bytearray)

Returns:

Status code

22.2.2.102 atcab_sign_base()

```
def cryptoauthlib.atcab.atcab_sign_base (
    mode,
    key_id,
    signature )
```

Executes the Sign command, which generates a signature using the ECDSA algorithm.

Args:

mode	Mode determines what the source of the message to be signed (int)
key_id	Private key slot used to sign the message. (int)
signature	Signature is returned here. Format is R and S integers in big-endian format. 64 bytes for P256 curve (Expects bytearray)

Returns:

Stauts code

22.2.2.103 atcab_sign_internal()

```
def cryptoauthlib.atcab.atcab_sign_internal (
    key_id,
    is_invalidate,
    is_full_sn,
    signature )
```

Executes Sign command to sign an internally generated message.

Args:

key_id	Slot of the private key to be used to sign the message (int)
is_invalidate	Set to true if the signature will be used with the Verify(Invalidate) command. false for all other cases.
is_full_sn	Set to true if the message should incorporate the device's full serial number.
signature	Signature is returned here. Format is R and S integers in big-endian format. 64 bytes for P256 curve (Expects bytearray)

Returns:

Status code

22.2.2.104 atcab_updateextra()

```
def cryptoauthlib.atcab.atcab_updateextra (
    mode,
    new_value )
```

Executes UpdateExtra command to update the values of the two extra bytes within the Configuration zone (bytes 84 and 85). an also be used to decrement the limited use counter associated with the key in slot NewValue.

Args:

mode	Mode determines what operations the UpdateExtra command performs. (int)
new_value	Value to be written. (int)

Returns:

Status code

22.2.2.105 atcab_verify()

```
def cryptoauthlib.atcab.atcab_verify (
    mode,
    key_id,
    signature,
    public_key,
    other_data,
    mac )
```

Executes the Verify command, which takes an ECDSA [R,S] signature and verifies that it is correctly generated from a given message and public key. In all cases, the signature is an input to the command. For the Stored, External, and ValidateExternal Modes, the contents of TempKey (or Message Digest Buffer in some cases for the ATECC608) should contain the 32 byte message.

Args:

mode	Verify command mode and options (int)
key_id	Stored mode, the slot containing the public key to be used for the verification. ValidateExternal mode, the slot containing the public key to be validated. External mode, KeyID contains the curve type to be used to Verify the signature. Validate or Invalidate mode, the slot containing the public key to be (in)validated. (int)
signature	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve. (bytearray or bytes)
public_key	If mode is External, the public key to be used for verification. X and Y integers in big-endian format. 64 bytes for P256 curve. NULL for all other modes. (bytearray or bytes)
other_data	If mode is Validate, the bytes used to generate the message for the validation (19 bytes). NULL for all other modes. (bytearray or bytes)
mac	If mode indicates a validating MAC, then the MAC will be returned here. Can be NULL otherwise. (Expects bytearray)

Returns:

Status code

22.2.2.106 atcab_verify_extern()

```
def cryptoauthlib.atcab.atcab_verify_extern (
    message,
    signature,
    public_key,
    is_verified )
```

Executes the Verify command, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

Args:

message	32 byte message to be verified. Typically the SHA256 hash of the full message. (Expects bytes)
signature	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve. (Expects bytes)
public_key	The public key to be used for verification. X and Y integers in big-endian format. 64 bytes for P256 curve. (Expects bytes)

22.2 cryptoauthlib.atcab Namespace Reference

`is_verified` Boolean whether or not the message, signature, public key verified.
(Expects AtcaReference)

Returns:
Status code

22.2.2.107 atcab_verify_extern_mac()

```
def cryptoauthlib.atcab.atcab_verify_extern_mac (
    message,
    signature,
    public_key,
    num_in,
    io_key,
    is_verified )
```

Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. This function is only available on the ATECC608.

Args:

<code>message</code>	32 byte message to be verified. Typically the SHA256 hash of the full message. (bytearray or bytes)
<code>signature</code>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve. (bytearray or bytes)
<code>public_key</code>	The public key to be used for verification. X and Y integers in big-endian format. 64 bytes for P256 curve. (bytearray or bytes)
<code>num_in</code>	System nonce (32 byte) used for the verification MAC. (bytearray or bytes)
<code>io_key</code>	IO protection key for verifying the validation MAC. (bytearray or bytes)
<code>is_verified</code>	Boolean whether or not the message, signature, public key verified. (Expects AtcaReference)

Returns:
Stats code

22.2.2.108 atcab_verify_extern_stored_mac()

```
def cryptoauthlib.atcab.atcab_verify_extern_stored_mac (
    mode,
    key_id,
    message,
    signature,
    public_key,
    num_in,
    io_key,
    is_verified )
```

Executes the Verify command with verification MAC for the External or Stored Verify modes..

Args:

<code>mode</code>	Verify command mode. Can be VERIFY_MODE_EXTERNAL or VERIFY_MODE_STORED. (int)
<code>key_id</code>	For VERIFY_MODE_STORED mode, the slot containing the public key

	to be used for the verification. For VERIFY_MODE_EXTERNAL mode, KeyID contains the curve type to be used to Verify the signature. Only VERIFY_KEY_P256 supported. (int)
message	32 byte message to be verified. Typically the SHA256 hash of the full message. (bytearray or bytes)
signature	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve. (bytearray or bytes)
public_key	For VERIFY_MODE_EXTERNAL mode, the public key to be used for verification. X and Y integers in big-endian format. 64 bytes for P256 curve. Null for VERIFY_MODE_STORED mode. (bytearray or bytes)
num_in	System nonce (32 byte) used for the verification MAC. (bytearray or bytes)
io_key	IO protection key for verifying the validation MAC. (bytearray or bytes)
is_verified	Boolean whether or not the message, signature, public key verified. (Expects AtcaReference)

Returns:

Status code

22.2.2.109 atcab_verify_invalidate()

```
def cryptoauthlib.atcab.atcab_verify_invalidate (
    key_id,
    signature,
    other_data,
    is_verified )
```

Executes the Verify command in Invalidate mode which invalidates a previously validated public key stored in a slot. This command can only be run after GenKey has been used to create a PubKey digest of the public key to be invalidated in TempKey (mode=0x10).

Args:

key_id	Slot containing the public key to be invalidated. (int)
signature	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve. (bytearray or bytes)
other_data	19 bytes of data used to build the verification message (bytearray or bytes)
is_verified	Boolean whether or not the message, signature, public key verified. (Expects AtcaReference)

Returns:

Status code

22.2.2.110 atcab_verify_stored()

```
def cryptoauthlib.atcab.atcab_verify_stored (
    message,
    signature,
    key_id,
    is_verified )
```

Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

Args:

message	32 byte message to be verified. Typically the SHA256 hash of
---------	--

22.2 cryptoauthlib.atcab Namespace Reference

signature	the full message. (bytearray or bytes) Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve. (bytearray or bytes)
key_id	Slot containing the public key to be used in the verification.(int)
is_verified	Boolean whether or not the message, signature, public key verified. (Expects AtcaReference)

Returns:
Status code

22.2.2.111 atcab_verify_stored_mac()

```
def cryptoauthlib.atcab.atcab_verify_stored_mac (  
    message,  
    signature,  
    key_id,  
    num_in,  
    io_key,  
    is_verified )
```

Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with a public key stored in the device. This function is only available on the ATECC608.

Args:

message	32 byte message to be verified. Typically the SHA256 hash of the full message. (bytearray or bytes)
signature	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve. (bytearray or bytes)
key_id	Slot containing the public key to be used in the verification. (int)
num_in	System nonce (32 byte) used for the verification MAC. (bytearray or bytes)
io_key	IO protection key for verifying the validation MAC. (bytearray or bytes)
is_verified	Boolean whether or not the message, signature, public key verified. (Expects AtcaReference)

Returns:
Status code

22.2.2.112 atcab_verify_validate()

```
def cryptoauthlib.atcab.atcab_verify_validate (  
    key_id,  
    signature,  
    other_data,  
    is_verified )
```

Executes the Verify command in Validate mode to validate a public key stored in a slot. This command can only be run after GenKey has been used to create a PubKey digest of the public key to be validated in TempKey (mode=0x10).

Args:

key_id	Slot containing the public key to be validated.(int)
--------	--

<code>signature</code>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve. (bytearray or bytes)
<code>other_data</code>	19 bytes of data used to build the verification message (bytearray or bytes)
<code>is_verified</code>	Boolean whether or not the message, signature, public key verified. (Expects AtcaReference)

Returns:

 Status code

22.2.2.113 atcab_write()

```
def cryptoauthlib.atcab.atcab_write (
    zone,
    address,
    value,
    mac )
```

Executes the Write command, which writes either one four byte word or a 32-byte block to one of the EEPROM zones on the device. Depending upon the value of the WriteConfig byte for this slot, the data may be required to be encrypted by the system prior to being sent to the device. This command cannot be used to write slots configured as ECC private keys.

Args:

<code>zone</code>	Zone/Param1 for the write command. (int)
<code>address</code>	Address/Param2 for the write command. (int)
<code>value</code>	Plain-text data to be written or cipher-text for encrypted writes. 32 or 4 bytes depending on bit 7 in the zone. (bytearray or bytes)
<code>data</code>	Data to be written. (bytearray or bytes)
<code>mac</code>	MAC required for encrypted writes (32 bytes). (bytearray or bytes)

Returns:

 Status code

22.2.2.114 atcab_write_bytes_zone()

```
def cryptoauthlib.atcab.atcab_write_bytes_zone (
    zone,
    slot,
    offset_bytes,
    data,
    length )
```

Executes the Write command, which writes data into config, otp, or data zone with a given byte offset and length. Offset and length must be multiples of a word (4 bytes).

Config zone must be unlocked for writes to that zone. If data zone is unlocked, only 32-byte writes are allowed to slots and OTP and the offset and length must be multiples of 32 or the write will fail.

Args:

<code>zone</code>	Zone to write data to: Zones.ATCA_ZONE_CONFIG, Zones.ATCA_ZONE_OTP, or Zones.ATCA_ZONE_DATA. (int)
<code>slot</code>	If zone is Zones.ATCA_ZONE_DATA, the slot number to write to.

	Ignored for all other zones. (int)
offset_bytes	Byte offset within the zone to write to. Must be a multiple of a word (4 bytes). (int)
data	bytearray containing Data to be written. (bytearray or bytes)
length	Number of bytes to be written. Must be a multiple of a word (4 bytes). (int)

Returns:
None

22.2.2.115 atcab_write_config_counter()

```
def cryptoauthlib.atcab.atcab_write_config_counter (
    counter_id,
    counter_value )
```

Initialize one of the monotonic counters in device with a specific value. The monotonic counters are stored in the configuration zone using a special format. This encodes a binary count value into the 8 byte encoded value required. This can only be set while the configuration zone is unlocked.

Args:

counter_id	Counter to be written (int)
counter_value	Counter value to set (int)

22.2.2.116 atcab_write_config_zone()

```
def cryptoauthlib.atcab.atcab_write_config_zone (
    conf )
```

Executes the Write command, which writes the configuration zone. First 16 bytes are skipped as they are not writable. LockValue and LockConfig are also skipped and can only be changed via the Lock command.

This command may fail if UserExtra and/or Selector bytes have already been set to non-zero values.

Args:

conf	Data to the config zone data. This should be a 88 byte bytearray for SHA devices and 128 byte bytearray for ECC devices. (bytearray or bytes)
------	---

Returns:
Status code

22.2.2.117 atcab_write_enc()

```
def cryptoauthlib.atcab.atcab_write_enc (
    key_id,
    block,
    data,
    enc_key,
    enc_key_id,
    num_in = None )
```

Executes the Write command, which performs an encrypted write of a 32 byte block into given slot. The function takes clear text bytes and encrypts them for writing over the wire. Data zone must be locked and the slot configuration must be set to encrypted write for the block to be successfully written.

Args:

key_id	Slot ID to write to. (int)
block	Index of the 32 byte block to write in the slot. (int)
data	32 bytes of clear text data to be written to the slot. (bytearray or bytes)
enc_key	WriteKey to encrypt with for writing (bytearray or bytes)
enc_key_id	The KeyID of the WriteKey (int)
num_in	Bytearray - Host nonce used to calculate nonce (20 bytes)

Returns:

Status code

22.2.2.118 atcab_write_pubkey()

```
def cryptoauthlib.atcab.atcab_write_pubkey (
    slot,
    public_key )
```

Executes the Write command, which writes a public key to a data slot in the device format.

Args:

slot	Slot number to write. Only slots 8 to 15 are large enough to store a public key. (int)
public_key	Public key to write into the slot specified. X and Y integers in big-endian format. 64 bytes for P256 curve. (bytearray or bytes)

Returns:

Status code

22.2.2.119 atcab_write_zone()

```
def cryptoauthlib.atcab.atcab_write_zone (
    zone,
    slot,
    block,
    offset,
    data,
    length )
```

22.3 cryptauthlib.atcacert Namespace Reference

Executes the Write command, which writes either 4 or 32 bytes of data into a device zone.

Args:

zone	Device zone to write to (0=config, 1=OTP, 2=data). (int)
slot	If writing to the data zone, it is the slot to write to, otherwise it should be 0. (int)
block	32-byte block to write to. (int)
offset	4-byte word within the specified block to write to. If performing a 32-byte write, this should be 0. (int)
data	Data to be written. (bytearray or bytes)
len	Number of bytes to be written. Must be either 4 or 32. (int)

Returns:

Status code

22.3 cryptauthlib.atcacert Namespace Reference

Data Structures

- class [atcacert_cert_element_t](#)
- class [atcacert_cert_loc_t](#)
- class [atcacert_cert_sn_src_t](#)
- class [atcacert_cert_type_t](#)
- class [atcacert_comp_data_t](#)
- class [atcacert_date_format_t](#)
- class [atcacert_def_t](#)
- class [atcacert_device_loc_t](#)
- class [atcacert_device_zone_t](#)
- class [atcacert_std_cert_element_t](#)
- class [atcacert_tm_utc_t](#)
- class [atcacert_transform_t](#)
- class [CertStatus](#)

Functions

- def [_atcacert_convert_bytes](#) (kwargs, name, pointer)
- def [_atcacert_convert_enum](#) (kwargs, name, enum)
- def [atcacert_max_cert_size](#) (cert_def, max_cert_size)
- def [atcacert_get_response](#) (device_private_key_slot, challenge, response)
- def [atcacert_read_cert](#) (cert_def, ca_public_key, cert, cert_size)
- def [atcacert_write_cert](#) (cert_def, cert, cert_size)
- def [atcacert_create_csr](#) (csr_def, csr, csr_size)
- def [atcacert_create_csr_pem](#) (csr_def, csr, csr_size)
- def [atcacert_date_enc](#) (date_format, timestamp, formatted_date, formatted_date_size)
- def [atcacert_date_dec](#) (date_format, formatted_date, formatted_date_size, timestamp)
- def [atcacert_date_enc_compcert](#) (issue_date, expire_years, enc_dates)
- def [atcacert_date_dec_compcert](#) (enc_dates, expire_date_format, issue_date, expire_date)
- def [atcacert_date_get_max_date](#) (date_format, timestamp)

22.3.1 Detailed Description

ATCACERT: classes and functions for interacting with compressed certificates

22.3.2 Function Documentation

22.3.2.1 `_atcacert_convert_bytes()`

```
def cryptoauthlib.atcacert._atcacert_convert_bytes (
    kwargs,
    name,
    pointer ) [protected]
```

Internal Helper Function: Convert python 'bytes' into memory pointer for ctypes structure
:param kwargs: kwargs dictionary
:param name: `_field_` name that will be converted
:param pointer: Conversion Class (resulting type - pointer of type x)
:return:

22.3.2.2 `_atcacert_convert_enum()`

```
def cryptoauthlib.atcacert._atcacert_convert_enum (
    kwargs,
    name,
    enum ) [protected]
```

Internal Helper Function: Convert python enum into ctypes integer
:param kwargs: kwargs dictionary
:param name: `_field_` name that will be converted
:param enum: Conversion Class (resulting type)
:return:

22.3.2.3 `atcacert_create_csr()`

```
def cryptoauthlib.atcacert.atcacert_create_csr (
    csr_def,
    csr,
    csr_size )
```

Creates a CSR specified by the CSR definition from the ATECC508A device.
This process involves reading the dynamic CSR data from the device and combining it
with the template found in the CSR definition, then signing it. Return the CSR int der format

Args:

<code>csr_def</code>	CSR definition describing where to find the dynamic CSR information on the device and how to incorporate it into the template. Expects <code>atcacert_def_t</code> .
<code>csr</code>	Buffer to receive the CSR. Expects bytearray.
<code>csr_size</code>	As input, the size of the CSR buffer in bytes. As output, the size of the CSR as PEM returned in cert in bytes. Expects <code>AtcaReference</code> .

Returns:

ATCACERT_E_SUCCESS on success, otherwise an error code.

22.3.2.4 atcacert_create_csr_pem()

```
def cryptoauthlib.atcacert.atcacert_create_csr_pem (
    csr_def,
    csr,
    csr_size )
```

Creates a CSR specified by the CSR definition from the ATECC508A device. This process involves reading the dynamic CSR data from the device and combining it with the template found in the CSR definition, then signing it. Return the CSR in der format

Args:

csr_def	CSR definition describing where to find the dynamic CSR information on the device and how to incorporate it into the template. Expects atcacert_def_t.
csr	Buffer to receive the CSR. Expects bytearray.
csr_size	As input, the size of the CSR buffer in bytes. As output, the size of the CSR as PEM returned in cert in bytes. Expects AtcaReference.

Returns:

ATCACERT_E_SUCCESS on success, otherwise an error code.

22.3.2.5 atcacert_date_dec()

```
def cryptoauthlib.atcacert.atcacert_date_dec (
    date_format,
    formatted_date,
    formatted_date_size,
    timestamp )
```

Parse a formatted timestamp according to the specified format.

Args:

date_format	Format to parse the formatted date as.
formatted_date	Formatted date to be parsed.
formatted_date_size	Size of the formatted date in bytes.
timestamp	Parsed timestamp is returned here. Expects atcacert_tm_utc_t.

Returns:

ATCACERT_E_SUCCESS on success, otherwise an error code.

22.3.2.6 atcacert_date_dec_compcert()

```
def cryptoauthlib.atcacert.atcacert_date_dec_compcert (
    enc_dates,
    expire_date_format,
    issue_date,
    expire_date )
```

Decode the issue and expire dates from the format used by the compressed certificate.

Args:

<code>enc_dates</code>	Encoded date from the compressed certificate. 3 bytes.
<code>expire_date_format</code>	Expire date format. Only used to determine max date when no expiration date is specified by the encoded date.
<code>issue_date</code>	Decoded issue date is returned here. Expects <code>atcacert_tm_utc_t</code> .
<code>expire_date</code>	Decoded expire date is returned here. If there is no expiration date, the expire date will be set to a maximum value for the given <code>expire_date_format</code> . Expects <code>atcacert_tm_utc_t</code> .

Returns:

`ATCACERT_E_SUCCESS` on success

22.3.2.7 atcacert_date_enc()

```
def cryptoauthlib.atcacert.atcacert_date_enc (
    date_format,
    timestamp,
    formatted_date,
    formatted_date_size )
```

Format a timestamp according to the format type.

Args:

<code>date_format</code>	Format to use.
<code>timestamp</code>	Timestamp to format. Expects <code>atcacert_tm_utc_t</code> .
<code>formatted_date</code>	Formatted date will be returned in this buffer. Expects bytearray.
<code>formatted_date_size</code>	As input, the size of the <code>formatted_date</code> buffer. As output, the size of the returned <code>formatted_date</code> . Expects <code>AtcaReference</code> .

Returns:

`ATCACERT_E_SUCCESS` on success, otherwise an error code.

22.3.2.8 atcacert_date_enc_compcert()

```
def cryptoauthlib.atcacert.atcacert_date_enc_compcert (
    issue_date,
    expire_years,
    enc_dates )
```

Encode the issue and expire dates in the format used by the compressed certificate.

Args:

<code>issue_date</code>	Issue date to encode. Note that minutes and seconds will be ignored. Expects <code>atcacert_tm_utc_t</code> .
<code>expire_years</code>	Expire date is expressed as a number of years past the issue date. 0 should be used if there is no expire date.
<code>enc_dates</code>	Encoded dates for use in the compressed certificate is returned here. 3 bytes. Expects bytearray.

Returns:

`ATCACERT_E_SUCCESS` on success

22.3.2.9 atcacert_date_get_max_date()

```
def cryptoauthlib.atcacert.atcacert_date_get_max_date (
    date_format,
    timestamp )
```

Return the maximum date available for the given format.

Args:

format	Format to get the max date for.
timestamp	Max date is returned here. Expects atcacert_tm_utc_t.

Returns:

ATCACERT_E_SUCCESS on success, otherwise an error code.

22.3.2.10 atcacert_get_response()

```
def cryptoauthlib.atcacert.atcacert_get_response (
    device_private_key_slot,
    challenge,
    response )
```

Calculates the response to a challenge sent from the host.

The challenge-response protocol is an ECDSA Sign and Verify. This performs the ECDSA Sign on the challenge and returns the signature as the response.

Args:

device_private_key_slot	Slot number for the device's private key. This must be the same slot used to generate the public key included in the device's certificate.
challenge	Challenge to generate the response for. Must be 32 bytes.
response	Response will be returned in this buffer. 64 bytes.

Returns:

ATCACERT_E_SUCCESS on success, otherwise an error code.

22.3.2.11 atcacert_max_cert_size()

```
def cryptoauthlib.atcacert.atcacert_max_cert_size (
    cert_def,
    max_cert_size )
```

Return the maximum possible certificate size in bytes for a given cert def. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificates.

Args:

cert_def	Certificate definition to find a max size for. Expects atcacert_def_t.
max_cert_size	Maximum certificate size will be returned here in bytes. Expects AtcaReference.

Returns:

ATCACERT_E_SUCCESS on success, otherwise an error code.

22.3.2.12 atcacert_read_cert()

```
def cryptoauthlib.atcacert.atcacert_read_cert (
    cert_def,
    ca_public_key,
    cert,
    cert_size )
```

Reads the certificate specified by the certificate definition from the ATECC508A device.

This process involves reading the dynamic cert data from the device and combining it with the template found in the certificate definition.

Args:

<code>cert_def</code>	Certificate definition describing where to find the dynamic certificate information on the device and how to incorporate it into the template. Expects <code>atcacert_def_t</code> .
<code>ca_public_key</code>	The ECC P256 public key of the certificate authority that signed this certificate. Formatted as the 32 byte X and Y integers concatenated together (64 bytes total). Set to NULL if the authority key id is not needed, set properly in the <code>cert_def</code> template, or stored on the device as specified in the <code>cert_def</code> <code>cert_elements</code> .
<code>cert</code>	Buffer to received the certificate. Expects bytearray.
<code>cert_size</code>	As input, the size of the cert buffer in bytes. As output, the size of the certificate returned in <code>cert</code> in bytes. Expects <code>AtcaReference</code> .

Returns:

ATCACERT_E_SUCCESS on success, otherwise an error code.

22.3.2.13 atcacert_write_cert()

```
def cryptoauthlib.atcacert.atcacert_write_cert (
    cert_def,
    cert,
    cert_size )
```

Take a full certificate and write it to the ATECC508A device according to the certificate definition.

Args:

<code>cert_def</code>	Certificate definition describing where the dynamic certificate information is and how to store it on the device. Expects <code>atcacert_def_t</code> .
<code>cert</code>	Full certificate to be stored.
<code>cert_size</code>	Size of the full certificate in bytes.

Returns:

ATCACERT_E_SUCCESS on success, otherwise an error code.

22.4 cryptoauthlib.atcaenum Namespace Reference

Data Structures

- class [AtcaEnum](#)

22.4.1 Detailed Description

Enum Extension for improved comparisons

22.5 cryptoauthlib.atjwt Namespace Reference

Data Structures

- class [HwEcAlgorithm](#)
- class [HwHmacAlgorithm](#)
- class [PyJWT](#)

Variables

- `try` :

22.5.1 Detailed Description

JWT: Extension to the jwt module with hardware based security

22.6 cryptoauthlib.device Namespace Reference

Data Structures

- class [AesEnable](#)
- class [Atecc508aConfig](#)
- class [Atecc608Config](#)
- class [Atsha204aConfig](#)
- class [ChipMode508](#)
- class [ChipMode608](#)
- class [ChipOptions](#)
- class [Counter204](#)
- class [CountMatch](#)
- class [I2cEnable](#)
- class [KeyConfig](#)
- class [SecureBoot](#)
- class [SlotConfig](#)
- class [UseLock](#)
- class [VolatileKeyPermission](#)
- class [X509Format](#)

22.6.1 Detailed Description

Cryptoauthlib Device Configuration

22.7 cryptoauthlib.exceptions Namespace Reference

Data Structures

- class [AssertionFailure](#)
- class [BadArgumentError](#)
- class [BadCrcError](#)
- class [BadOpcodeError](#)
- class [CheckmacVerifyFailedError](#)
- class [CommunicationError](#)
- class [ConfigZoneLockedError](#)
- class [CrcError](#)
- class [CryptoError](#)
- class [DataZoneLockedError](#)
- class [EccFaultError](#)
- class [ExecutionError](#)
- class [FunctionError](#)
- class [GenericError](#)
- class [HealthTestError](#)
- class [InvalidIdentifierError](#)
- class [InvalidSizeError](#)
- class [LibraryLoadError](#)
- class [LibraryMemoryError](#)
- class [LibraryNotInitialized](#)
- class [NoDevicesFoundError](#)
- class [NoResponseError](#)
- class [NoUseFlagError](#)
- class [ParityError](#)
- class [ParseError](#)
- class [ReceiveError](#)
- class [ReceiveTimeoutError](#)
- class [ResyncWithWakeupError](#)
- class [StatusUnknownError](#)
- class [TimeOutError](#)
- class [TransmissionError](#)
- class [TransmissionTimeoutError](#)
- class [UnimplementedError](#)
- class [UnsupportedInterface](#)
- class [WakeFailedError](#)
- class [ZoneNotLockedError](#)

22.7.1 Detailed Description

Cryptoauthlib Exceptions

22.8 cryptauthlib.iface Namespace Reference

Data Structures

- class [_ATCACUSTOM](#)
- class [_ATCAHID](#)
- class [_ATCAI2C](#)
- class [_ATCAIfaceParams](#)
- class [_ATCAKIT](#)
- class [_ATCASPI](#)
- class [_ATCASWI](#)
- class [_ATCAUART](#)
- class [_U_Address](#)
- class [ATCADeviceType](#)
- class [ATCAIfaceCfg](#)
- class [ATCAIfaceType](#)
- class [ATCAKitType](#)

Functions

- def [_iface_load_default_config](#) (name)
- def [cfg_ateccx08a_i2c_default](#) ()
- def [cfg_ateccx08a_swi_default](#) ()
- def [cfg_ateccx08a_kithid_default](#) ()
- def [cfg_atsha20xa_i2c_default](#) ()
- def [cfg_atsha20xa_swi_default](#) ()
- def [cfg_atsha20xa_kithid_default](#) ()

22.8.1 Detailed Description

Interface Configuration

22.8.2 Function Documentation

22.8.2.1 _iface_load_default_config()

```
def cryptauthlib.iface._iface_load_default_config (  
    name ) [protected]
```

"Attempt to load the default configuration structure from the library by name

22.8.2.2 `cfg_ateccx08a_i2c_default()`

```
def cryptoauthlib.iface.cfg_ateccx08a_i2c_default ( )
```

Default configuration for an ECCx08A device on the first logical I2C bus

22.8.2.3 `cfg_ateccx08a_kithid_default()`

```
def cryptoauthlib.iface.cfg_ateccx08a_kithid_default ( )
```

Default configuration for Kit protocol over a HID interface

22.8.2.4 `cfg_ateccx08a_swi_default()`

```
def cryptoauthlib.iface.cfg_ateccx08a_swi_default ( )
```

Default configuration for an ECCx08A device on the logical SWI bus over UART

22.8.2.5 `cfg_atsha20xa_i2c_default()`

```
def cryptoauthlib.iface.cfg_atsha20xa_i2c_default ( )
```

Default configuration for a SHA204A device on the first logical I2C bus

22.8.2.6 `cfg_atsha20xa_kithid_default()`

```
def cryptoauthlib.iface.cfg_atsha20xa_kithid_default ( )
```

Default configuration for Kit protocol over a HID interface for SHA204

22.8.2.7 `cfg_atsha20xa_swi_default()`

```
def cryptoauthlib.iface.cfg_atsha20xa_swi_default ( )
```

Default configuration for an SHA204A device on the logical SWI bus over UART

22.9 cryptoauthlib.library Namespace Reference

Data Structures

- class [_CtypeIterator](#)
- class [AtcaReference](#)
- class [AtcaStructure](#)
- class [AtcaUnion](#)
- class [cal_buffer](#)

Functions

- def **indent** (lines, insert)
- def [_force_local_library](#) ()
- def [load_cryptoauthlib](#) (lib=None)
- def [get_cryptoauthlib](#) ()
- def [get_device_name](#) (revision)
- def [get_device_name_with_device_id](#) (revision)
- def [get_device_type_id](#) (name)
- def [get_size_by_name](#) (name)
- def [get_ctype_by_name](#) (name)
- def [get_ctype_structure_instance](#) (structure, value)
- def [get_ctype_array_instance](#) (array, value)
- def [_get_field_definition](#) (obj, name)
- def [_def_to_field](#) (f_type, f_size=None)
- def [_convert_pointer_to_list](#) (p, length)
- def [_get_attribute_from_ctypes](#) (obj, obj_type, length=None, *args)
- def [_check_type_rationality](#) (cls)
- def [_array_to_code](#) (obj, name=None, parent=None, **kwargs)
- def [_object_definition_code](#) (obj, name=None, parent=None, parent_name=None, anon=None, type_info=None, check_names={}, **kwargs)
- def [_union_to_code](#) (obj, name=None, parent=None, anon=None, entry=None, parent_name=None, type_info=None, **kwargs)
- def [_structure_to_code](#) (obj, name=None, parent=None, type_info=None, parent_name=None, **kwargs)
- def [_obj_to_code](#) (obj, name, parent=None, anon=None, parent_name=None, **kwargs)
- def [_pointer_to_code](#) (obj, name=None, parent=None, parent_name=None, check_names={}, skip_references=[], **kwargs)
- def [_is_pointer](#) (obj, type_info=None, **kwargs)
- def [_to_code](#) (obj, name=None, **kwargs)
- def [_structure_to_string](#) (item, int level=0)
- def [_ctype_from_definition](#) (cls)
- def [ctypes_to_bytes](#) (obj)
- def **create_byte_buffer** (init_or_size)

Variables

- **try :**
- dict **ATCA_NAMES** = {'i2c': 'i2c', 'hid': 'kithid', 'sha': 'sha204', 'ecc': 'eccx08'}
- None **_CRYPTO_LIB** = None
- dict **_CTYPES_BY_SIZE** = {1: c_uint8, 2: c_uint16, 4:c_uint32}
- **_fields_**

22.9.1 Detailed Description

Cryptoauthlib Library Management

22.9.2 Function Documentation

22.9.2.1 `_array_to_code()`

```
def cryptoauthlib.library._array_to_code (
    obj,
    name = None,
    parent = None,
    ** kwargs ) [protected]
```

Convert an array like item from a ctypes structure into a C language formatted string

22.9.2.2 `_check_type_rationality()`

```
def cryptoauthlib.library._check_type_rationality (
    cls ) [protected]
```

This checks the structure or union size against the constants that are stored in the library during compilation. This is not an absolute guarentee that alignment is completely correct but it will catch most cases of incompatibility between the compiled library that is installed and the python module

22.9.2.3 `_convert_pointer_to_list()`

```
def cryptoauthlib.library._convert_pointer_to_list (
    p,
    length ) [protected]
```

Pointer types can be frustrating to interact with generally when processing data in python so this converts them into types that are iterable and bounded

22.9.2.4 `_ctype_from_definition()`

```
def cryptoauthlib.library._ctype_from_definition (
    cls ) [protected]
```

Extends the ctypes structure and union types to add a new attribute `_def_` which is a dictionary of field attributes. This extends functionality by quite a bit by supporting additional types and field linkages

22.9.2.5 `_def_to_field()`

```
def cryptoauthlib.library._def_to_field (
    f_type,
    f_size = None ) [protected]
```

Helper function to convert an entry in the `_def_` dictionary to the tuple required for a `_field_` entry

22.9.2.6 `_force_local_library()`

```
def cryptoauthlib.library._force_local_library ( ) [protected]
```

In some environments loading seems to fail under all circumstances unless brute forcing it.

22.9.2.7 `_get_attribute_from_ctypes()`

```
def cryptoauthlib.library._get_attribute_from_ctypes (
    obj,
    obj_type,
    length = None,
    * args ) [protected]
```

Helper function that is used by `AtcaStructure` and `AtcaUnion` to intercept attribute access to those objects and convert the resulting values into easier to use python objects based on the configuration of the structure/union

22.9.2.8 `_get_field_definition()`

```
def cryptoauthlib.library._get_field_definition (
    obj,
    name ) [protected]
```

Get meta information about the ctypes structure/union by accessing the field description attributes of the class that were provided as part of the ctypes structure/union definition

22.9.2.9 `_is_pointer()`

```
def cryptoauthlib.library._is_pointer (
    obj,
    type_info = None,
    ** kwargs ) [protected]
```

Checks to see if object looks like a pointer

22.9.2.10 `_obj_to_code()`

```
def cryptoauthlib.library._obj_to_code (
    obj,
    name,
    parent = None,
    anon = None,
    parent_name = None,
    ** kwargs ) [protected]
```

Convert python/ctypes object into a C language representation

22.9.2.11 `_object_definition_code()`

```
def cryptoauthlib.library._object_definition_code (
    obj,
    name = None,
    parent = None,
    parent_name = None,
    anon = None,
    type_info = None,
    check_names = {},
    ** kwargs ) [protected]
```

Emits the first half of the assignment of this object

22.9.2.12 `_pointer_to_code()`

```
def cryptoauthlib.library._pointer_to_code (
    obj,
    name = None,
    parent = None,
    parent_name = None,
    check_names = {},
    skip_references = [],
    ** kwargs ) [protected]
```

Convert the pointer into a representative object by creating a definition in the prepend area

22.9.2.13 `_structure_to_code()`

```
def cryptoauthlib.library._structure_to_code (
    obj,
    name = None,
    parent = None,
    type_info = None,
    parent_name = None,
    ** kwargs ) [protected]
```

Emits a string with a C language representation of the structure(s) following pointers the best that is can

22.9.2.14 `_structure_to_string()`

```
def cryptoauthlib.library._structure_to_string (
    item,
    int level = 0 ) [protected]
```

Emits a readable string of the structure elements coverting types and following pointers and arrays the best that is can

22.9.2.15 `_to_code()`

```
def cryptoauthlib.library._to_code (
    obj,
    name = None,
    ** kwargs ) [protected]
```

Map object types to the proper renderer function by catching pointer like objects first

Returns: (append, prepend)

22.9.2.16 ctypes_to_bytes()

```
def cryptoauthlib.library.ctypes_to_bytes (
    obj )
```

Convert a ctypes structure/array into bytes. This is for python2 compatibility

22.9.2.17 get_cryptoauthlib()

```
def cryptoauthlib.library.get_cryptoauthlib ( )
```

This is a helper function for the other python files in this module to use the loaded library

22.9.2.18 get_ctype_array_instance()

```
def cryptoauthlib.library.get_ctype_array_instance (
    array,
    value )
```

Internal Helper Function: Convert python list into ctype array
:param value: Value to convert
:param array: Conversion Class (resulting type)
:return:

22.9.2.19 get_ctype_by_name()

```
def cryptoauthlib.library.get_ctype_by_name (
    name )
```

For known (atca_utils_sizes.c) types that are custom to the library retrieve the size

22.9.2.20 get_ctype_structure_instance()

```
def cryptoauthlib.library.get_ctype_structure_instance (
    structure,
    value )
```

Internal Helper Function: Convert a value into the correct ctypes structure for a given field
:param value: Value to convert
:param structure: Conversion Class (resulting type)
:return:

22.9.2.21 `get_device_name()`

```
def cryptoauthlib.library.get_device_name (
    revision )
```

Returns the device name based on the info byte array values returned by `atcab_info`

22.9.2.22 `get_device_name_with_device_id()`

```
def cryptoauthlib.library.get_device_name_with_device_id (
    revision )
```

Returns the device name based on the info byte array values returned by `atcab_info` for ECC204 family

22.9.2.23 `get_device_type_id()`

```
def cryptoauthlib.library.get_device_type_id (
    name )
```

Returns the `ATCADeviceType` value based on the device name

22.9.2.24 `get_size_by_name()`

```
def cryptoauthlib.library.get_size_by_name (
    name )
```

Get the size of an object in the library using the `name_size` api from `atca_utils_sizes.c`

22.9.2.25 `load_cryptoauthlib()`

```
def cryptoauthlib.library.load_cryptoauthlib (
    lib = None )
```

Load `CryptoAuthLib` into Python environment
raise `LibraryLoadError` if `cryptoauthlib` library can't be loaded

22.10 cryptoauthlib.sha206_api Namespace Reference

Functions

- def [sha206a_generate_derive_key](#) (parent_key, derived_key, param1, param2)
- def [sha206a_generate_challenge_response_pair](#) (key, challenge, response)
- def [sha206a_authenticate](#) (challenge, expected_response, is_verified)
- def [sha206a_write_data_store](#) (slot, data, block, offset, length, lock_after_write)
- def [sha206a_read_data_store](#) (slot, data, offset, length)
- def [sha206a_get_data_store_lock_status](#) (slot, is_locked)
- def [sha206a_get_dk_update_count](#) (dk_update_count)
- def [sha206a_get_pk_useflag_count](#) (pk_avail_count)
- def [sha206a_get_dk_useflag_count](#) (dk_avail_count)
- def [sha206a_check_pk_useflag_validity](#) (is_consumed)
- def [sha206a_check_dk_useflag_validity](#) (is_consumed)
- def [sha206a_verify_device_consumption](#) (is_consumed)
- def [sha206a_diversify_parent_key](#) (parent_key, diversified_key)

22.10.1 Detailed Description

SHA206 API: classes and functions for interacting with SHA206A device

22.10.2 Function Documentation

22.10.2.1 sha206a_authenticate()

```
def cryptoauthlib.sha206_api.sha206a_authenticate (
    challenge,
    expected_response,
    is_verified )
```

verifies the challenge and provided response using key in device

Args:

challenge	Challenge to be used in the response calculations (Expects bytearray of size 32)
expected_response	Expected response from the device (Expects bytearray of size 32)
is_authenticated	result of expected of response and calcualted response (AtcaReference expected)

Returns:

Status Code

22.10.2.2 sha206a_check_dk_useflag_validity()

```
def cryptoauthlib.sha206_api.sha206a_check_dk_useflag_validity (  
    is_consumed )
```

verifies Derived Key use flags for consumption

Args:

<code>is_consumed</code>	indicates if derived key is available for consumption (Expected AtcaReference)
--------------------------	---

Returns:

Status Code

22.10.2.3 sha206a_check_pk_useflag_validity()

```
def cryptoauthlib.sha206_api.sha206a_check_pk_useflag_validity (  
    is_consumed )
```

verifies Parent Key use flags for consumption

Args:

<code>is_consumed</code>	indicates if parent key is available for consumption (Expected AtcaReference)
--------------------------	--

Returns:

Status Code

22.10.2.4 sha206a_diversify_parent_key()

```
def cryptoauthlib.sha206_api.sha206a_diversify_parent_key (  
    parent_key,  
    diversified_key )
```

Computes the diversified key based on the parent key provided and device serial number

Args:

<code>parent_key</code>	parent key to be diversified (Expected bytearray of size 32)
<code>diversified_key</code>	output diversified key is returned here (Expected bytearray of size 32)

Returns:

Status Code

22.10.2.5 sha206a_generate_challenge_response_pair()

```
def cryptoauthlib.sha206_api.sha206a_generate_challenge_response_pair (
    key,
    challenge,
    response )
```

Generates the response based on Key and Challenge provided

Args:

key	input data contains device's key (Expects bytearray of size 32)
challenge	input data to be used in challenge response calculation (Expects bytearray of size 32)
response	output response is returned here (Expects bytearray of size 32)

Returns:

Status Code

22.10.2.6 sha206a_generate_derive_key()

```
def cryptoauthlib.sha206_api.sha206a_generate_derive_key (
    parent_key,
    derived_key,
    param1,
    param2 )
```

Generates the derived key based on the parent key and other parameters provided

Args:

parent_key	input data contains device's parent key (Expects bytearray of size 32)
derived key	output derived key is returned here (Expects bytearray of size 32)
param1	input data to be used in derive key calculation (int)
param2	input data to be used in derive key calculation (int)

Returns:

Status Code

22.10.2.7 sha206a_get_data_store_lock_status()

```
def cryptoauthlib.sha206_api.sha206a_get_data_store_lock_status (
    slot,
    is_locked )
```

22.10 cryptoauthlib.sha206_api Namespace Reference

Returns the lock status of the given data store

Args:

slot	Slot number of the data store (int)
is_locked	lock status of the data store slot (Expected AtcaReference)

Returns:

Status Code

22.10.2.8 sha206a_get_dk_update_count()

```
def cryptoauthlib.sha206_api.sha206a_get_dk_update_count (
    dk_update_count )
```

Read Derived Key slot update count. It will be wraps around 256

Args:

dk_update_count	returns number of times the slot has been updated with derived key (Expected AtcaReference)
-----------------	--

Returns:

Status Code

22.10.2.9 sha206a_get_dk_useflag_count()

```
def cryptoauthlib.sha206_api.sha206a_get_dk_useflag_count (
    dk_avail_count )
```

calculates available Derived Key use counts

Args:

dk_avail_count	counts available bit's as 1 (int)
----------------	-----------------------------------

Returns:

Status Code

22.10.2.10 sha206a_get_pk_useflag_count()

```
def cryptoauthlib.sha206_api.sha206a_get_pk_useflag_count (
    pk_avail_count )
```

calculates available Parent Key use counts

Args:

pk_avail_count	counts available bit's as 1 (int)
----------------	-----------------------------------

Returns:

Status Code

22.10.2.11 sha206a_read_data_store()

```
def cryptoauthlib.sha206_api.sha206a_read_data_store (
    slot,
    data,
    offset,
    length )
```

Read the data stored in Data store

Args:

slot	Slot number to read from (int)
data	Pointer that holds the data (Expected bytearray of size 32)
offset	Byte offset within the zone to read from. (int)
length	data length (int)

Returns:

Status Code

22.10.2.12 sha206a_verify_device_consumption()

```
def cryptoauthlib.sha206_api.sha206a_verify_device_consumption (
    is_consumed )
```

verifies the device is fully consumed or not based on Parent and Derived Key use flags.

Args:

is_consumed	result of device consumption is returned here (Expected AtcaReference)
-------------	---

Returns:

Status Code

22.10.2.13 sha206a_write_data_store()

```
def cryptoauthlib.sha206_api.sha206a_write_data_store (
    slot,
    data,
    block,
    offset,
    length,
    lock_after_write )
```

22.11 cryptoauthlib.status Namespace Reference

Update the data store slot with user data and lock it if necessary

Args:

slot	Slot number to be written with data (int)
data	Pointer that holds the data (Expected bytearray of size 32)
block	32-byte block to write (int)
offset	4-byte word within the specified block to write to. If performing a 32-byte write, this should be 0. (int)
length	data length (int)
lock_after_write	set 1 to lock slot after write, otherwise 0 (Expected bool/int)

Returns:

Status Code

22.11 cryptoauthlib.status Namespace Reference

Data Structures

- class [Status](#)

Functions

- def [check_status](#) (status, *args, **kwargs)

Variables

- dict **STATUS_EXCEPTION_MAP**

22.11.1 Detailed Description

Status codes and status to exception conversions.

22.11.2 Function Documentation

22.11.2.1 check_status()

```
def cryptoauthlib.status.check_status (  
    status,  
    * args,  
    ** kwargs )
```

Look up the status return code from an API call and raise the exception that matches

22.12 cryptauthlib.tng Namespace Reference

Functions

- def [tng_get_device_pubkey](#) (public_key)
- def [tng_atcacert_max_device_cert_size](#) (max_cert_size)
- def [tng_atcacert_read_device_cert](#) (cert, cert_size, signer_cert=None)
- def [tng_atcacert_device_public_key](#) (public_key, cert=None)
- def [tng_atcacert_max_signer_cert_size](#) (max_cert_size)
- def [tng_atcacert_read_signer_cert](#) (cert, cert_size)
- def [tng_atcacert_signer_public_key](#) (public_key, cert=None)
- def [tng_atcacert_root_cert_size](#) (cert_size)
- def [tng_atcacert_root_cert](#) (cert, cert_size)
- def [tng_atcacert_root_public_key](#) (public_key)

22.12.1 Detailed Description

TNG: classes and functions for interacting with TNG devices

22.12.2 Function Documentation

22.12.2.1 tng_atcacert_device_public_key()

```
def cryptauthlib.tng.tng_atcacert_device_public_key (
    public_key,
    cert = None )
```

Reads the device public key.

Args:

public_key	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve. Expects bytearray.
cert	If supplied, the device public key is used from this certificate. If set to None, the device public key is read from the device. Expects bytes or None.

Returns:

ATCACERT_E_SUCCESS on success, otherwise an error code.

22.12.2.2 tng_atcacert_max_device_cert_size()

```
def cryptoauthlib.tng.tng_atcacert_max_device_cert_size (
    max_cert_size )
```

Return the maximum possible certificate size in bytes for a TNG device certificate. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificate.

Args:

max_cert_size Maximum certificate size will be returned here in bytes. Expects AtcaReference.

Returns:

ATCACERT_E_SUCCESS on success, otherwise an error code.

22.12.2.3 tng_atcacert_max_signer_cert_size()

```
def cryptoauthlib.tng.tng_atcacert_max_signer_cert_size (
    max_cert_size )
```

Return the maximum possible certificate size in bytes for a TNG signer certificate. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificate.

Args:

max_cert_size Maximum certificate size will be returned here in bytes. Expects AtcaReference.

Returns:

ATCACERT_E_SUCCESS on success, otherwise an error code.

22.12.2.4 tng_atcacert_read_device_cert()

```
def cryptoauthlib.tng.tng_atcacert_read_device_cert (
    cert,
    cert_size,
    signer_cert = None )
```

Reads the device certificate for a TNG device.

Args:

cert Buffer to receive the certificate (DER format). Expects bytearray.
cert_size As input, the size of the cert buffer in bytes. As output, the size of the certificate returned in cert in bytes. Expects AtcaReference.
signer_cert If supplied, the signer public key is used from this certificate. If set to None, the signer public key is read from the device. Expects bytes or None.

Returns:

ATCACERT_E_SUCCESS on success, otherwise an error code.

22.12.2.5 tng_atcacert_read_signer_cert()

```
def cryptoauthlib.tng.tng_atcacert_read_signer_cert (
    cert,
    cert_size )
```

Reads the signer certificate for a TNG device.

Args:

cert	Buffer to received the certificate (DER format). Expects bytearray.
cert_size	As input, the size of the cert buffer in bytes. As output, the size of the certificate returned in cert in bytes. Expects AtcaReference.

Returns:

ATCACERT_E_SUCCESS on success, otherwise an error code.

22.12.2.6 tng_atcacert_root_cert()

```
def cryptoauthlib.tng.tng_atcacert_root_cert (
    cert,
    cert_size )
```

Get the TNG root cert.

Args:

cert	Buffer to received the certificate (DER format). Expects bytearray.
cert_size	As input, the size of the cert buffer in bytes. As output, the size of the certificate returned in cert in bytes. Expects AtcaReference.

Returns:

ATCACERT_E_SUCCESS on success, otherwise an error code.

22.12.2.7 tng_atcacert_root_cert_size()

```
def cryptoauthlib.tng.tng_atcacert_root_cert_size (
    cert_size )
```

Get the size of the TNG root cert.

Args:

cert_size	Certificate size will be returned here in bytes. Expects AtcaReference.
-----------	--

Returns:

ATCACERT_E_SUCCESS on success, otherwise an error code.

22.12.2.8 tng_atcacert_root_public_key()

```
def cryptoauthlib.tng.tng_atcacert_root_public_key (
    public_key )
```

Gets the root public key.

Args:

public_key Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve. Expects bytearray.

Returns:

ATCACERT_E_SUCCESS on success, otherwise an error code.

22.12.2.9 tng_atcacert_signer_public_key()

```
def cryptoauthlib.tng.tng_atcacert_signer_public_key (
    public_key,
    cert = None )
```

Reads the signer public key.

Args:

public_key Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve. Expects bytearray.
cert If supplied, the signer public key is used from this certificate. If set to None, the signer public key is read from the device. Expects bytes or None.

Returns:

ATCACERT_E_SUCCESS on success, otherwise an error code.

22.12.2.10 tng_get_device_pubkey()

```
def cryptoauthlib.tng.tng_get_device_pubkey (
    public_key )
```

Uses GenKey command to calculate the public key from the primary device public key.

Args:

public_key Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve. Expects bytearray.

Returns:

ATCA_SUCCESS on success, otherwise an error code.

22.13 test_device Namespace Reference

Functions

- def **test_device_config_size** (config, size)
- def **test_device_config_from_def** (config, definition, vector)
- def **test_device_config_from_vector** (config, vector)
- def **test_device_serial_number_from_def** (config, definition, vector)

Variables

- bytearray **ATSHA204A_SER_NUM_VECTOR** = bytearray.fromhex('01 23 6E AA CE FE 0B 8D EE')
- bytearray **ATSHA204A_DEVICE_CONFIG_VECTOR**
- dict **ATSHA204A_DEVICE_CONFIG**
- bytearray **ATECC508A_SER_NUM_VECTOR** = bytearray.fromhex('01 23 72 E8 B9 63 B2 D3 EE')
- bytearray **ATECC508A_DEVICE_CONFIG_VECTOR**
- dict **ATECC508A_DEVICE_CONFIG**
- bytearray **ATECC608_SER_NUM_VECTOR** = bytearray.fromhex('01 23 72 E8 B9 63 B2 D3 EE')
- bytearray **ATECC608_DEVICE_CONFIG_VECTOR**
- dict **ATECC608_DEVICE_CONFIG**
- **id**

22.13.1 Detailed Description

Device.py tests. Covers the configuration structures

22.13.2 Variable Documentation

22.13.2.1 ATECC508A_DEVICE_CONFIG

dict test_device.ATECC508A_DEVICE_CONFIG

Initial value:

```
00001 = {
00002     'SN03': [0x01, 0x23, 0x72, 0xE8],
00003     'RevNum': [0x00, 0x00, 0x60, 0x02],
00004     'SN48': [0xB9, 0x63, 0xB2, 0xD3, 0xEE],
00005     'I2C_Enable': 0x2D,
00006     'I2C_Address': 0xB0,
00007     'OTPmode': 0x55,
00008     'SlotConfig': [0x208F, 0x44C4, 0x2087, 0x2087,
00009                   0x0F8F, 0x36C4, 0x0F9F, 0x2082,
00010                   0x0F0F, 0x44C4, 0x0F0F, 0x0F0F,
00011                   0x0F0F, 0x0F0F, 0x0F0F, 0x0F0F],
00012     'Counter0': [0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00],
00013     'Counter1': [0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00],
00014     'LastKeyUse': [0xFF, 0xFF, 0xFF, 0xFF,
00015                  0xFF, 0xFF, 0xFF, 0xFF,
00016                  0xFF, 0xFF, 0xFF, 0xFF,
00017                  0xFF, 0xFF, 0xFF, 0xFF],
00018     'LockValue': 0x55,
00019     'LockConfig': 0x55,
00020     'SlotLocked': 0xFFFF,
00021     'KeyConfig': [0x0033, 0x001C, 0x0013, 0x0013,
00022                  0x007C, 0x001C, 0x003C, 0x0033,
00023                  0x003C, 0x003C, 0x003C, 0x0030,
00024                  0x003C, 0x003C, 0x003C, 0x0030]
00025 }
```

22.13.2.2 ATECC508A_DEVICE_CONFIG_VECTOR

bytearray test_device.ATECC508A_DEVICE_CONFIG_VECTOR

Initial value:

```
00001 = bytearray.fromhex(
00002     '01 23 72 E8 00 00 60 02 B9 63 B2 D3 EE 00 2D 00'
00003     'B0 00 55 00 8F 20 C4 44 87 20 87 20 8F 0F C4 36'
00004     '9F 0F 82 20 0F 0F C4 44 0F 0F 0F 0F 0F 0F 0F 0F'
00005     '0F 0F 0F 0F FF FF FF FF 00 00 00 00 FF FF FF FF'
00006     '00 00 00 00 FF FF FF FF FF FF FF FF FF FF FF FF'
00007     'FF FF FF FF 00 00 55 55 FF FF 00 00 00 00 00 00'
00008     '33 00 1C 00 13 00 13 00 7C 00 1C 00 3C 00 33 00'
00009     '3C 00 3C 00 3C 00 30 00 3C 00 3C 00 3C 00 30 00')
```

22.13.2.3 ATECC608_DEVICE_CONFIG

dict test_device.ATECC608_DEVICE_CONFIG

Initial value:

```
00001 = {
00002     'SN03': [0x01, 0x23, 0x72, 0xE8],
00003     'RevNum': [0x00, 0x00, 0x60, 0x02],
00004     'SN48': [0xB9, 0x63, 0xB2, 0xD3, 0xEE],
00005     'AES_Enable': {'Enable': 1},
00006     'I2C_Enable': 0x2D,
00007     'I2C_Address': 0xB0,
00008     'ChipMode': 1,
00009     'CountMatch': 0x55,
00010     'SlotConfig': [0x208F, 0x44C4, 0x2087, 0x2087,
00011                    0x0F8F, 0x36C4, 0x0F9F, 0x2082,
00012                    0x0F0F, 0x44C4, 0x0F0F, 0x0F0F,
00013                    0x0F0F, 0x0F0F, 0x0F0F, 0x0F0F],
00014     'Counter0': [0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00],
00015     'Counter1': [0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00],
00016     'SlotLocked': 0xFFFF,
00017     'ChipOptions': {
00018         'IoProtectionKeyEnable': 1,
00019         'KdfAesEnable': 1,
00020         'IoProtectionKey': 4
00021     },
00022     'KeyConfig': [0x0033, 0x001C, 0x0013, 0x0013,
00023                  0x007C, 0x001C, 0x003C, 0x0033,
00024                  0x003C, 0x003C, 0x003C, 0x0030,
00025                  0x003C, 0x003C, 0x003C, 0x0030]
00026 }
```

22.13.2.4 ATECC608_DEVICE_CONFIG_VECTOR

bytearray test_device.ATECC608_DEVICE_CONFIG_VECTOR

Initial value:

```
00001 = bytearray.fromhex(
00002     '01 23 72 E8 00 00 60 02 B9 63 B2 D3 EE 01 2D 00'
00003     'B0 00 55 01 8F 20 C4 44 87 20 87 20 8F 0F C4 36'
00004     '9F 0F 82 20 0F 0F C4 44 0F 0F 0F 0F 0F 0F 0F 0F'
00005     '0F 0F 0F 0F FF FF FF FF 00 00 00 00 FF FF FF FF'
00006     '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
00007     '00 00 00 00 00 00 00 00 FF FF 06 40 00 00 00 00'
00008     '33 00 1C 00 13 00 13 00 7C 00 1C 00 3C 00 33 00'
00009     '3C 00 3C 00 3C 00 30 00 3C 00 3C 00 3C 00 30 00')
```


22.13.2.5 ATSHA204A_DEVICE_CONFIG

```
dict test_device.ATSHA204A_DEVICE_CONFIG
```

Initial value:

```
00001 = {
00002     'SN03': [0x01, 0x23, 0x6E, 0xAA],
00003     'RevNum': [0x00, 0x09, 0x04, 0x00],
00004     'SN48': [0xCE, 0xFE, 0x0B, 0x8D, 0xEE],
00005     'I2C_Enable': 0x01,
00006     'I2C_Address': 0xC8,
00007     'OTPmode': 0x55,
00008     'SlotConfig': [0x808F, 0xA180, 0xE082, 0xF4C4,
00009                     0x0084, 0x85A0, 0x4086, 0x0787,
00010                     0x000F, 0x64C4, 0x7A8A, 0x8B0B,
00011                     0x4C0C, 0x4DDD, 0x42C2, 0x8FAF],
00012     'Counter': [0xFF, 0xFF, 0xFF, 0xFF,
00013                 0xFF, 0xFF, 0xFF, 0xFF],
00014     'LastKeyUse': [0xFF, 0xFF, 0xFF, 0xFF,
00015                    0xFF, 0xFF, 0xFF, 0xFF,
00016                    0xFF, 0xFF, 0xFF, 0xFF,
00017                    0xFF, 0xFF, 0xFF, 0xFF],
00018     'LockValue': 0x55,
00019     'LockConfig': 0x55
00020 }
```

22.13.2.6 ATSHA204A_DEVICE_CONFIG_VECTOR

```
bytearray test_device.ATSHA204A_DEVICE_CONFIG_VECTOR
```

Initial value:

```
00001 = bytearray.fromhex(
00002     '01 23 6E AA 00 09 04 00 CE FE 0B 8D EE 00 01 00'
00003     'C8 00 55 00 8F 80 80 A1 82 E0 C4 F4 84 00 A0 85'
00004     '86 40 87 07 0F 00 C4 64 8A 7A 0B 8B 0C 4C DD 4D'
00005     'C2 42 AF 8F FF 00 FF 00 FF 00 FF 00 FF 00 FF 00'
00006     'FF 00 FF 00 FF FF FF FF FF FF FF FF FF FF FF'
00007     'FF FF FF FF 00 00 55 55')
```

22.14 test_iface Namespace Reference

Functions

- def **test_iface_init** (test_init_with_lib)
- def **test_iface_cfg_size** (test_iface_init)
- def **test_iface_cfg_ateccx08a_i2c** (test_iface_init)
- def **test_iface_cfg_ateccx08a_swi** (test_iface_init)
- def **test_iface_cfg_ateccx08a_kithid** (test_iface_init)
- def **test_iface_cfg_atsha20xa_i2c** (test_iface_init)
- def **test_iface_cfg_atsha20xa_swi** (test_iface_init)
- def **test_iface_cfg_atsha20xa_kithid** (test_iface_init)

22.14.1 Detailed Description

These tests verify the structures match the expectation from what is in atca_cfs.c. If that file has been modified then the tests will fail. If the file has not been modified then we can reasonably expect that there is a problem with the ctypes definition or assumptions of the platform build and memory alignment is wrong.

Chapter 23

Data Structure Documentation

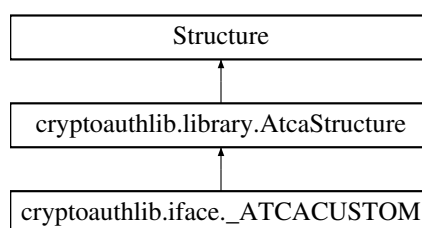
23.1 `_ascii_kit_host_context` Struct Reference

Data Fields

- const [atca_hal_kit_phy_t](#) * **phy**
- `uint8_t` **buffer** [(2500)]
- [ATCADevice](#) **device**
- [ATCAIfaceCfg](#) ** **iface**
- `size_t` **iface_count**
- `uint32_t` **flags**

23.2 `cryptoauthlib.iface._ATCACUSTOM` Class Reference

Inheritance diagram for `cryptoauthlib.iface._ATCACUSTOM`:



Static Protected Attributes

- list [_fields_](#)

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.library.AtcaStructure](#)

- None **__init__** (self, *args, **kwargs)
- def **from_definition** (cls)
- def **check_rationality** (cls)
- def **get_field_definition** (cls, str name)
- Any **__getattr__** (self, str name)
- def **__iter__** (self)
- def **__str__** (self)
- def **to_c_code** (self, name=None, **kwargs)
- def **update_from_buffer** (self, buffer)

23.2.1 Detailed Description

Custom HAL configuration

23.2.2 Field Documentation

23.2.2.1 `_fields_`

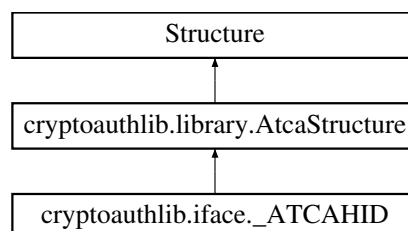
```
list cryptoauthlib.iface._ATCACUSTOM._fields_ [static], [protected]
```

Initial value:

```
= [('halinit', c_void_p),
    ('halpostinit', c_void_p),
    ('halsend', c_void_p),
    ('halreceive', c_void_p),
    ('halwake', c_void_p),
    ('halidle', c_void_p),
    ('halsleep', c_void_p),
    ('halrelease', c_void_p)]
```

23.3 `cryptoauthlib.iface._ATCAHID` Class Reference

Inheritance diagram for `cryptoauthlib.iface._ATCAHID`:



Static Protected Attributes

- dict `_def_`

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.library.AtcaStructure](#)

- None `__init__` (self, *args, **kwargs)
- def `from_definition` (cls)
- def `check_rationality` (cls)
- def `get_field_definition` (cls, str name)
- Any `__getattr__` (self, str name)
- def `__iter__` (self)
- def `__str__` (self)
- def `to_c_code` (self, name=None, **kwargs)
- def `update_from_buffer` (self, buffer)

23.3.1 Detailed Description

USB (HID) HAL configuration

23.3.2 Field Documentation

23.3.2.1 `_def_`

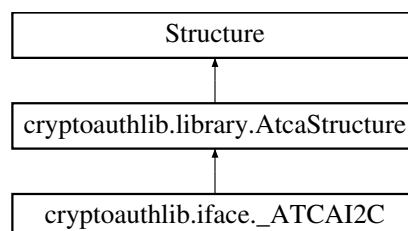
dict `cryptoauthlib.iface._ATCAHID._def_` [static], [protected]

Initial value:

```
= {
    'idx': (c_int,),
    'dev_interface': (ATCAKitType,),
    'dev_identity': (c_uint8,),
    'vid': (c_uint32,),
    'pid': (c_uint32,),
    'packet_size': (c_uint32,)
}
```

23.4 cryptoauthlib.iface._ATCAI2C Class Reference

Inheritance diagram for `cryptoauthlib.iface._ATCAI2C`:



Static Protected Attributes

- tuple `_anonymous_` = ('u',)
- dict `_map_`
- list `_fields_`

Additional Inherited Members

Public Member Functions inherited from `cryptoauthlib.library.AtcaStructure`

- None `__init__` (self, *args, **kwargs)
- def `from_definition` (cls)
- def `check_rationality` (cls)
- def `get_field_definition` (cls, str name)
- Any `__getattr__` (self, str name)
- def `__iter__` (self)
- def `__str__` (self)
- def `to_c_code` (self, name=None, **kwargs)
- def `update_from_buffer` (self, buffer)

23.4.1 Detailed Description

I2C/TWI HAL configuration

23.4.2 Field Documentation

23.4.2.1 `_fields_`

```
list cryptoauthlib.iface._ATCAI2C._fields_ [static], [protected]
```

Initial value:

```
= [ ('u', _U_Address),
    ('bus', c_uint8),
    ('baud', c_uint32)]
```

23.4.2.2 `_map_`

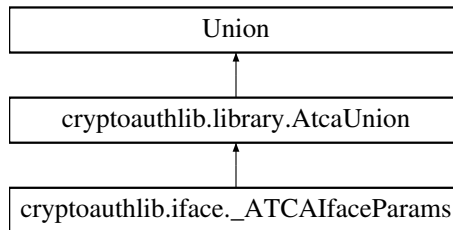
```
dict cryptoauthlib.iface._ATCAI2C._map_ [static], [protected]
```

Initial value:

```
= {
    'u': (1,)
}
```

23.5 cryptoauthlib.iface._ATCAIfaceParams Class Reference

Inheritance diagram for cryptoauthlib.iface._ATCAIfaceParams:



Static Protected Attributes

- list [_fields_](#)

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.library.AtcaUnion](#)

- def `__init__` (self, *args, **kwargs)
- def [from_definition](#) (cls)
- def [check_rationality](#) (cls)
- def [get_field_definition](#) (cls, str name)
- Any `__getattr__` (self, str name)
- def `__iter__` (self)
- def `__str__` (self)
- def [to_c_code](#) (self, name=None, **kwargs)
- def [update_from_buffer](#) (self, buffer)

Protected Attributes inherited from [cryptoauthlib.library.AtcaUnion](#)

- `_selected`

23.5.1 Detailed Description

HAL Configurations supported by the library (this is a union)

23.5.2 Field Documentation

23.5.2.1 `_fields_`

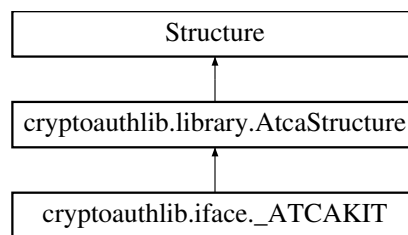
```
list cryptoauthlib.iface._ATCAIfaceParams._fields_ [static], [protected]
```

Initial value:

```
= [ ('atcai2c', _ATCAI2C),
    ('atcaswi', _ATCASWI),
    ('atcaspi', _ATCASPI),
    ('atcauart', _ATCAUART),
    ('atcahid', _ATCAHID),
    ('atcakit', _ATCAKIT),
    ('atcacustom', _ATCACUSTOM)]
```

23.6 `cryptoauthlib.iface._ATCAKIT` Class Reference

Inheritance diagram for `cryptoauthlib.iface._ATCAKIT`:



Static Protected Attributes

- dict `_def_`

Additional Inherited Members

Public Member Functions inherited from `cryptoauthlib.library.AtcaStructure`

- None `__init__` (self, *args, **kwargs)
- def `from_definition` (cls)
- def `check_rationality` (cls)
- def `get_field_definition` (cls, str name)
- Any `__getattr__` (self, str name)
- def `__iter__` (self)
- def `__str__` (self)
- def `to_c_code` (self, name=None, **kwargs)
- def `update_from_buffer` (self, buffer)

23.6.1 Detailed Description

Kit (Bridge) HAL Configuration

23.6.2 Field Documentation

23.6.2.1 `_def_`

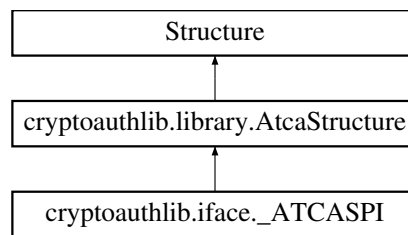
```
dict cryptoauthlib.iface._ATCAKIT._def_ [static], [protected]
```

Initial value:

```
= {  
    'dev_interface': (ATCAKitType,),  
    'dev_identity': (c_uint8,),  
    'flags': (c_uint32,)  
}
```

23.7 cryptoauthlib.iface._ATCASPI Class Reference

Inheritance diagram for `cryptoauthlib.iface._ATCASPI`:



Static Protected Attributes

- list `_fields_`

Additional Inherited Members

Public Member Functions inherited from `cryptoauthlib.library.AtcaStructure`

- None `__init__` (self, *args, **kwargs)
- def `from_definition` (cls)
- def `check_rationality` (cls)
- def `get_field_definition` (cls, str name)
- Any `__getattr__` (self, str name)
- def `__iter__` (self)
- def `__str__` (self)
- def `to_c_code` (self, name=None, **kwargs)
- def `update_from_buffer` (self, buffer)

23.7.1 Detailed Description

SPI HAL configuration

23.7.2 Field Documentation

23.7.2.1 `_fields_`

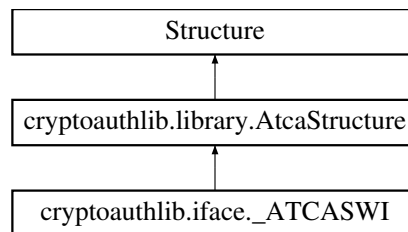
```
list cryptoauthlib.iface._ATCASPI._fields_ [static], [protected]
```

Initial value:

```
= [('bus', c_uint8),
    ('select_pin', c_uint8),
    ('baud', c_uint32)]
```

23.8 `cryptoauthlib.iface._ATCASWI` Class Reference

Inheritance diagram for `cryptoauthlib.iface._ATCASWI`:



Static Protected Attributes

- list [_fields_](#)

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.library.AtcaStructure](#)

- None `__init__` (self, *args, **kwargs)
- def [from_definition](#) (cls)
- def [check_rationality](#) (cls)
- def [get_field_definition](#) (cls, str name)
- Any `__getattr__` (self, str name)
- def `__iter__` (self)
- def `__str__` (self)
- def [to_c_code](#) (self, name=None, **kwargs)
- def [update_from_buffer](#) (self, buffer)

23.8.1 Detailed Description

SWI (Atmel Single Wire Interface) HAL configuration

23.8.2 Field Documentation

23.8.2.1 `_fields_`

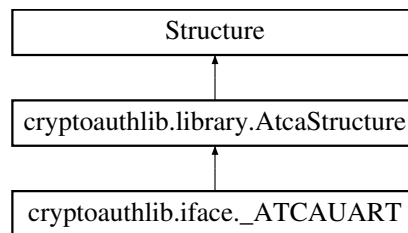
```
list cryptoauthlib.iface._ATCASWI._fields_ [static], [protected]
```

Initial value:

```
= [('address', c_uint8),  
    ('bus', c_uint8)]
```

23.9 cryptoauthlib.iface._ATCAUART Class Reference

Inheritance diagram for cryptoauthlib.iface._ATCAUART:



Static Protected Attributes

- dict `_def_`

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.library.AtcaStructure](#)

- None `__init__` (self, *args, **kwargs)
- def [from_definition](#) (cls)
- def [check_rationality](#) (cls)
- def [get_field_definition](#) (cls, str name)
- Any `__getattr__` (self, str name)
- def `__iter__` (self)
- def `__str__` (self)
- def [to_c_code](#) (self, name=None, **kwargs)
- def [update_from_buffer](#) (self, buffer)

23.9.1 Detailed Description

Generic UART HAL configuration

23.9.2 Field Documentation

23.9.2.1 `_def_`

```
dict cryptoauthlib.iface._ATCAUART._def_ [static], [protected]
```

Initial value:

```
= {
    'dev_interface': (ATCAKitType,),
    'dev_identity': (c_uint8,),
    'port': (c_uint8,),
    'baud': (c_uint32,),
    'wordsize': (c_uint8,),
    'parity': (c_uint8,),
    'stopbits': (c_uint8,)
}
```

23.10 `cryptoauthlib.library._CtpeIterator` Class Reference

Public Member Functions

- None `__init__` (self, obj)
- `def __iter__` (self)
- `def __next__` (self)

Protected Attributes

- `_obj`
- `_index`
- `_end`

23.10.1 Detailed Description

Used to iterate through a ctypes structure or union. This iterator returns a tuple of three elements:

```
<field_name>, <field_contents>, <field_info>
```

Of course `field_info` is a tuple of varying size depending on how the field was defined (arrays, bitfields, etc)

23.11 `_kit_host_map_entry` Struct Reference

```
#include <app/kit_host/ascii_kit_host.h>
```

Data Fields

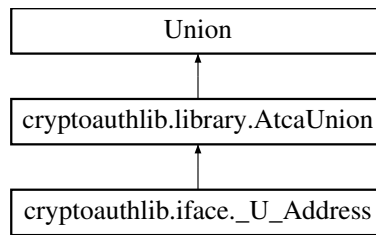
- `const char * id`
- `ATCA_STATUS(* fp_command)(ascii_kit_host_context_t *ctx, int argc, char *argv[], uint8_t *response, size_t *rlen)`

23.11.1 Detailed Description

Used to create command tables for the kit host parser

23.12 cryptoauthlib.iface._U_Address Class Reference

Inheritance diagram for cryptoauthlib.iface._U_Address:



Static Protected Attributes

- list [_fields_](#)

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.library.AtcaUnion](#)

- def [__init__](#) (self, *args, **kwargs)
- def [from_definition](#) (cls)
- def [check_rationality](#) (cls)
- def [get_field_definition](#) (cls, str name)
- Any [__getattr__](#) (self, str name)
- def [__iter__](#) (self)
- def [__str__](#) (self)
- def [to_c_code](#) (self, name=None, **kwargs)
- def [update_from_buffer](#) (self, buffer)

Protected Attributes inherited from [cryptoauthlib.library.AtcaUnion](#)

- [_selected](#)

23.12.1 Detailed Description

Hidden union to provide backward compatibility with the api change

23.12.2 Field Documentation

23.12.2.1 [_fields_](#)

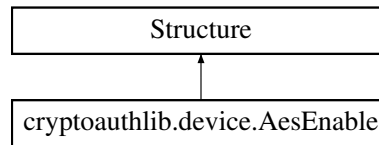
```
list cryptoauthlib.iface._U_Address._fields_  [static], [protected]
```

Initial value:

```
= [ ('slave_address', c_uint8),
    ('address', c_uint8)]
```

23.13 cryptauthlib.device.AesEnable Class Reference

Inheritance diagram for cryptauthlib.device.AesEnable:



Static Protected Attributes

- list [_fields_](#)
- int [_pack_](#) = 1

23.13.1 Detailed Description

AES Enable (608) Field Definition

23.13.2 Field Documentation

23.13.2.1 [_fields_](#)

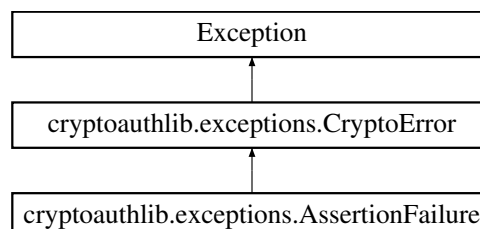
```
list cryptauthlib.device.AesEnable._fields_ [static], [protected]
```

Initial value:

```
= [('Enable', c_uint8, 1),
    ('Reserved', c_uint8, 6)]
```

23.14 cryptauthlib.exceptions.AssertionFailure Class Reference

Inheritance diagram for cryptauthlib.exceptions.AssertionFailure:

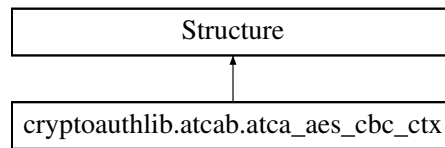


23.14.1 Detailed Description

Code failed run-time consistency check

23.15 cryptoauthlib.atcab.atca_aes_cbc_ctx Class Reference

Inheritance diagram for cryptoauthlib.atcab.atca_aes_cbc_ctx:



Static Protected Attributes

- list [_fields_](#)

23.15.1 Detailed Description

AES CBC Context

23.15.2 Field Documentation

23.15.2.1 [_fields_](#)

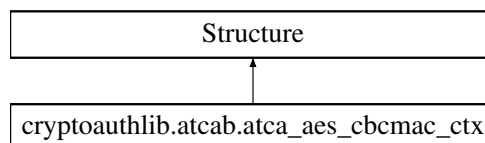
```
list cryptoauthlib.atcab.atca_aes_cbc_ctx._fields_ [static], [protected]
```

Initial value:

```
= [{"device", c_void_p),  
    ("key_id", c_uint16),  
    ("key_block", c_uint8),  
    ("ciphertext", c_char*16)]
```

23.16 cryptoauthlib.atcab.atca_aes_cbcmac_ctx Class Reference

Inheritance diagram for cryptoauthlib.atcab.atca_aes_cbcmac_ctx:



Static Protected Attributes

- list [_fields_](#)

23.16.1 Detailed Description

AES CBCMAC Context

23.16.2 Field Documentation

23.16.2.1 `_fields_`

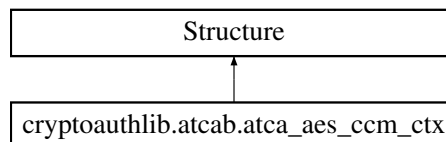
```
list cryptoauthlib.atcab.atca_aes_cbcmac_ctx._fields_ [static], [protected]
```

Initial value:

```
= [{"cbc_ctx", atca_aes_cbc_ctx),
    ("block_size", c_uint8),
    ("block", c_char*16)]
```

23.17 cryptoauthlib.atcab.atca_aes_ccm_ctx Class Reference

Inheritance diagram for cryptoauthlib.atcab.atca_aes_ccm_ctx:



Static Protected Attributes

- [list `_fields_`](#)

23.17.1 Detailed Description

AES CCM Context

23.17.2 Field Documentation

23.17.2.1 `_fields_`

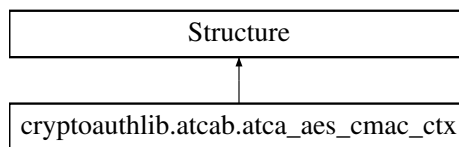
```
list cryptoauthlib.atcab.atca_aes_ccm_ctx._fields_ [static], [protected]
```

Initial value:

```
= [{"cbc_mac_ctx", atca_aes_cbc_ctx),
    ("ctr_ctx", atca_aes_ctr_ctx),
    ("iv_size", c_uint8),
    ("M", c_uint8),
    ("counter", c_char*16),
    ("partial_aad", c_char*16),
    ("partial_aad_size", c_uint32),
    ("text_size", c_uint32),
    ("enc_cb", c_char*16),
    ("data_size", c_uint32),
    ("ciphertext_block", c_char*16)]
```

23.18 cryptoauthlib.atcab.atca_aes_cmac_ctx Class Reference

Inheritance diagram for cryptoauthlib.atcab.atca_aes_cmac_ctx:



Static Protected Attributes

- list [_fields_](#)

23.18.1 Detailed Description

AES CMAC Context

23.18.2 Field Documentation

23.18.2.1 `_fields_`

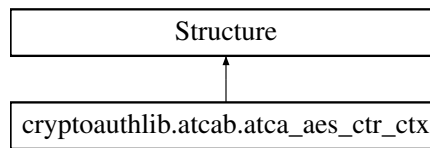
```
list cryptoauthlib.atcab.atca_aes_cmac_ctx._fields_ [static], [protected]
```

Initial value:

```
= [{"cbc_ctx", atca_aes_cbc_ctx),
    ("block_size", c_uint32),
    ("block", c_char*16)]
```


23.19 cryptauthlib.atcab.atca_aes_ctr_ctx Class Reference

Inheritance diagram for cryptauthlib.atcab.atca_aes_ctr_ctx:



Static Protected Attributes

- list [_fields_](#)

23.19.1 Detailed Description

AES CTR Context

23.19.2 Field Documentation

23.19.2.1 [_fields_](#)

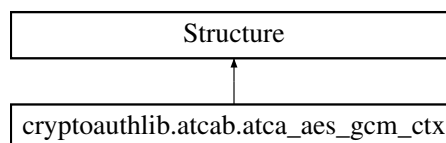
```
list cryptauthlib.atcab.atca_aes_ctr_ctx._fields_ [static], [protected]
```

Initial value:

```
= [{"key_id", c_uint16},
    {"key_block", c_uint8},
    {"iv", c_char*16},
    {"counter_size", c_uint8}]
```

23.20 cryptauthlib.atcab.atca_aes_gcm_ctx Class Reference

Inheritance diagram for cryptauthlib.atcab.atca_aes_gcm_ctx:



Static Protected Attributes

- list [_fields_](#)

23.20.1 Detailed Description

Context structure for AES GCM operations

23.20.2 Field Documentation

23.20.2.1 _fields_

```
list cryptoauthlib.atcab.atca_aes_gcm_ctx._fields_ [static], [protected]
```

Initial value:

```
= [{"key_id", c_uint16},
    ("key_block", c_uint8),
    ("cb", c_char*16),
    ("data_size", c_uint32),
    ("aad_size", c_uint32),
    ("h", c_char*16),
    ("j0", c_char*16),
    ("y", c_char*16),
    ("partial_aad", c_char*16),
    ("partial_aad_size", c_uint32),
    ("enc_cb", c_char*16),
    ("ciphertext_block", c_char*16)]
```

23.21 atca_check_mac_in_out Struct Reference

Input/output parameters for function atcah_check_mac().

```
#include <lib/host/atca_host.h>
```

Data Fields

- **uint8_t mode**
[in] CheckMac command Mode
- **uint16_t key_id**
[in] CheckMac command KeyID
- **const uint8_t * sn**
[in] Device serial number SN[0:8]. Only SN[0:1] and SN[8] are required though.
- **const uint8_t * client_chal**
[in] ClientChal data, 32 bytes. Can be NULL if mode[0] is 1.
- **uint8_t * client_resp**
[out] Calculated ClientResp will be returned here.
- **const uint8_t * other_data**
[in] OtherData, 13 bytes
- **const uint8_t * otp**
[in] First 8 bytes of the OTP zone data. Can be NULL is mode[5] is 0.
- **const uint8_t * slot_key**
- **const uint8_t * target_key**
- **struct atca_temp_key * temp_key**
[in,out] Current state of TempKey. Required if mode[0] or mode[1] are 1.

23.21.1 Detailed Description

Input/output parameters for function `atcah_check_mac()`.

23.21.2 Field Documentation

23.21.2.1 slot_key

```
const uint8_t* atca_check_mac_in_out::slot_key
```

[in] 32 byte key value in the slot specified by `slot_id`. Can be NULL if `mode[1]` is 1.

23.21.2.2 target_key

```
const uint8_t* atca_check_mac_in_out::target_key
```

[in] If this is not NULL, it assumes CheckMac copy is enabled for the specified `key_id` (`ReadKey=0`). If `key_id` is even, this should be the 32-byte key value for the slot `key_id+1`, otherwise this should be set to `slot_key`.

23.22 atca_decrypt_in_out Struct Reference

Input/output parameters for function `atca_decrypt()`.

```
#include <lib/host/atca_host.h>
```

Data Fields

- `uint8_t * crypto_data`
[in,out] Pointer to 32-byte data. Input encrypted data from Read command (Contents field), output decrypted.
- struct `atca_temp_key * temp_key`
[in,out] Pointer to TempKey structure.

23.22.1 Detailed Description

Input/output parameters for function `atca_decrypt()`.

23.23 atca_delete_in_out Struct Reference

Input/Output paramters for calculating the mac.Used with Delete command.

```
#include <lib/host/atca_host.h>
```

Data Fields

- uint16_t **key_id**
- const uint8_t * **sn**
- uint8_t * **nonce**
- const uint8_t * **key**
- uint8_t * **mac**

23.23.1 Detailed Description

Input/Output paramters for calculating the mac.Used with Delete command.

23.24 atca_derive_key_in_out Struct Reference

Input/output parameters for function atcah_derive_key().

```
#include <lib/host/atca_host.h>
```

Data Fields

- uint8_t **mode**
Mode (param 1) of the derive key command.
- uint16_t **target_key_id**
Key ID (param 2) of the target slot to run the command on.
- const uint8_t * **sn**
Device serial number SN[0:8]. Only SN[0:1] and SN[8] are required though.
- const uint8_t * **parent_key**
Parent key to be used in the derive key calculation (32 bytes).
- uint8_t * **target_key**
Derived key will be returned here (32 bytes).
- struct [atca_temp_key](#) * **temp_key**
Current state of TempKey.

23.24.1 Detailed Description

Input/output parameters for function atcah_derive_key().

23.25 atca_derive_key_mac_in_out Struct Reference

Input/output parameters for function atcah_derive_key_mac().

```
#include <lib/host/atca_host.h>
```

Data Fields

- `uint8_t mode`
Mode (param 1) of the derive key command.
- `uint16_t target_key_id`
Key ID (param 2) of the target slot to run the command on.
- `const uint8_t * sn`
Device serial number SN[0:8]. Only SN[0:1] and SN[8] are required though.
- `const uint8_t * parent_key`
Parent key to be used in the derive key calculation (32 bytes).
- `uint8_t * mac`
DeriveKey MAC will be returned here.

23.25.1 Detailed Description

Input/output parameters for function `atcah_derive_key_mac()`.

23.26 atca_device Struct Reference

`atca_device` is the C object backing ATCADevice. See the [atca_device.h](#) file for details on the ATCADevice methods

```
#include <lib/atca_device.h>
```

Data Fields

- `atca_iface_t miface`
- `uint8_t device_state`
- `uint8_t clock_divider`
- `uint16_t execution_time_msec`
- `void * session_ctx`
- `ctx_cb session_cb`

23.26.1 Detailed Description

`atca_device` is the C object backing ATCADevice. See the [atca_device.h](#) file for details on the ATCADevice methods

23.26.2 Field Documentation

23.26.2.1 device_state

```
uint8_t atca_device::device_state
```

Device Power State

23.26.2.2 miface

`atca_iface_t` `atca_device::mIface`

Physical interface

23.27 atca_diversified_key_in_out Struct Reference

Input/output parameters for function `atcah_gendivkey()`.

```
#include <lib/host/atca_host.h>
```

Data Fields

- `const uint8_t * parent_key`
- `const uint8_t * other_data`
- `const uint8_t * sn`
[in] Device serial number SN[0:8]. Only SN[0:1] and SN[8] are required though.
- `const uint8_t * input_data`
- `struct atca_temp_key * temp_key`
[inout] Current state of TempKey

23.27.1 Detailed Description

Input/output parameters for function `atcah_gendivkey()`.

23.28 atca_evp_ctx Struct Reference

Data Fields

- `void * ptr`

23.29 atca_gen_dig_in_out Struct Reference

Input/output parameters for function `atcah_gen_dig()`.

```
#include <lib/host/atca_host.h>
```

Data Fields

- `uint8_t zone`
[in] Zone/Param1 for the GenDig command
- `uint16_t key_id`
[in] KeyId/Param2 for the GenDig command
- `uint16_t slot_conf`
[in] Slot config for the GenDig command
- `uint16_t key_conf`
[in] Key config for the GenDig command
- `uint8_t slot_locked`
[in] slot locked for the GenDig command
- `uint32_t counter`
[in] counter for the GenDig command
- `bool is_key_nomac`
[in] Set to true if the slot pointed to be key_id has the SotConfig.NoMac bit set
- `const uint8_t * sn`
[in] Device serial number SN[0:8]. Only SN[0:1] and SN[8] are required though.
- `const uint8_t * stored_value`
[in] 32-byte slot value, config block, OTP block as specified by the Zone/KeyId parameters
- `const uint8_t * other_data`
[in] 32-byte value for shared nonce zone, 4-byte value if is_key_nomac is true, ignored and/or NULL otherwise
- `struct atca_temp_key * temp_key`
[inout] Current state of TempKey

23.29.1 Detailed Description

Input/output parameters for function `atcah_gen_dig()`.

23.30 atca_gen_key_in_out Struct Reference

Input/output parameters for calculating the PubKey digest put into TempKey by the GenKey command with the `atcah_gen_key_msg()` function.

```
#include <lib/host/atca_host.h>
```

Data Fields

- `uint8_t mode`
[in] GenKey Mode
- `uint16_t key_id`
[in] GenKey KeyID
- `const uint8_t * public_key`
[in] Public key to be used in the PubKey digest. X and Y integers in big-endian format. 64 bytes for P256 curve.
- `size_t public_key_size`
[in] Total number of bytes in the public key. 64 bytes for P256 curve.
- `const uint8_t * other_data`
[in] 3 bytes required when bit 4 of the mode is set. Can be NULL otherwise.
- `const uint8_t * sn`
[in] Device serial number SN[0:8] (9 bytes). Only SN[0:1] and SN[8] are required though.
- `struct atca_temp_key * temp_key`
[in,out] As input the current state of TempKey. As output, the resulting PubKey digest.

23.30.1 Detailed Description

Input/output parameters for calculating the PubKey digest put into TempKey by the GenKey command with the atcah_gen_key_msg() function.

23.31 atca_hal_kit_phy_t Struct Reference

Data Fields

- ATCA_STATUS(* [send](#))(void *ctx, uint8_t *txdata, uint16_t txlen)
- ATCA_STATUS(* [recv](#))(void *ctx, uint8_t *rxdata, uint16_t *rxlen)
- void (* [packet_alloc](#))(size_t bytes)
- void(* [packet_free](#))(void *packet)
- void * [hal_data](#)

23.31.1 Field Documentation

23.31.1.1 hal_data

```
void* atca_hal_kit_phy_t::hal_data
```

Physical layer context

23.31.1.2 packet_alloc

```
void (* atca_hal_kit_phy_t::packet_alloc) (size_t bytes)
```

Allocate a phy packet

23.31.1.3 packet_free

```
void(* atca_hal_kit_phy_t::packet_free) (void *packet)
```

Free a phy packet

23.31.1.4 recv

```
ATCA_STATUS(* atca_hal_kit_phy_t::recv) (void *ctx, uint8_t *rxdata, uint16_t *rxlen)
```

Must be a blocking receive

23.31.1.5 send

```
ATCA_STATUS(* atca_hal_kit_phy_t::send) (void *ctx, uint8_t *txdata, uint16_t txlen)
```

Must be a blocking send

23.32 atca_hal_list_entry_t Struct Reference

Structure that holds the hal/phy mapping for different interface types.

Data Fields

- `uint8_t iface_type`
- `ATCAHAL_t * hal`
- `ATCAHAL_t * phy`

23.32.1 Detailed Description

Structure that holds the hal/phy mapping for different interface types.

23.32.2 Field Documentation

23.32.2.1 phy

```
ATCAHAL_t* atca_hal_list_entry_t::phy
```

Physical interface for the specific HAL

23.33 atca_hal_shm_t Struct Reference

Data Fields

- `int recordedPID`
- `uint8_t sessionID`
- `uint8_t index`

23.34 atca_hmac_in_out Struct Reference

Input/output parameters for function `atca_hmac()`.

```
#include <lib/host/atca_host.h>
```

Data Fields

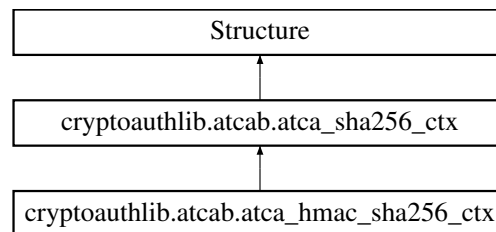
- `uint8_t mode`
[in] Mode parameter used in HMAC command (Param1).
- `uint16_t key_id`
[in] KeyID parameter used in HMAC command (Param2).
- `const uint8_t * key`
[in] Pointer to 32-byte key used to generate HMAC digest.
- `const uint8_t * otp`
[in] Pointer to 11-byte OTP, optionally included in HMAC digest, depending on mode.
- `const uint8_t * sn`
[in] Pointer to 9-byte SN, optionally included in HMAC digest, depending on mode.
- `uint8_t * response`
[out] Pointer to 32-byte SHA-256 HMAC digest.
- `struct atca_temp_key * temp_key`
[in,out] Pointer to TempKey structure.

23.34.1 Detailed Description

Input/output parameters for function `atca_hmac()`.

23.35 cryptoauthlib.atcab.atca_hmac_sha256_ctx Class Reference

Inheritance diagram for `cryptoauthlib.atcab.atca_hmac_sha256_ctx`:



Additional Inherited Members

Static Protected Attributes inherited from `cryptoauthlib.atcab.atca_sha256_ctx`

- `list _fields_`

23.35.1 Detailed Description

HMAC-SHA256 context

23.36 atca_i2c_host_s Struct Reference

Data Fields

- char **i2c_file** [16]
- int **ref_ct**

23.37 atca_iface Struct Reference

[atca_iface](#) is the context structure for a configured interface

```
#include <lib/atca_iface.h>
```

Data Fields

- [ATCAIfaceCfg](#) * **mIfaceCFG**
- [ATCAHAL_t](#) * **hal**
- [ATCAHAL_t](#) * **phy**
- void * **hal_data**

23.37.1 Detailed Description

[atca_iface](#) is the context structure for a configured interface

23.37.2 Field Documentation

23.37.2.1 hal

```
ATCAHAL\_t* atca_iface::hal
```

The configured HAL for the interface

23.37.2.2 hal_data

```
void* atca_iface::hal_data
```

Pointer to HAL specific context/data

23.37.2.3 mIfaceCFG

```
ATCAIfaceCfg* atca_iface::mIfaceCFG
```

Points to previous defined/given Cfg object, the caller manages this

23.37.2.4 phy

`ATCAHAL_t* atca_iface::phy`

When a HAL is not a "native" hal it needs a physical layer to be associated with it

23.38 atca_include_data_in_out Struct Reference

Input / output parameters for function `atca_include_data()`.

```
#include <lib/host/atca_host.h>
```

Data Fields

- `uint8_t * p_temp`
[out] pointer to output buffer
- `const uint8_t * otp`
[in] pointer to one-time-programming data
- `const uint8_t * sn`
[in] pointer to serial number data
- `uint8_t mode`

23.38.1 Detailed Description

Input / output parameters for function `atca_include_data()`.

23.39 atca_io_decrypt_in_out Struct Reference

Data Fields

- `const uint8_t * io_key`
IO protection key (32 bytes).
- `const uint8_t * out_nonce`
OutNonce returned from command (32 bytes).
- `uint8_t * data`
As input, encrypted data. As output, decrypted data.
- `size_t data_size`
Size of data in bytes (32 or 64).

23.40 atca_mac_in_out Struct Reference

Input/output parameters for function `atca_mac()`.

```
#include <lib/host/atca_host.h>
```

Data Fields

- `uint8_t mode`
[in] Mode parameter used in MAC command (Param1).
- `uint16_t key_id`
[in] KeyID parameter used in MAC command (Param2).
- `const uint8_t * challenge`
[in] Pointer to 32-byte Challenge data used in MAC command, depending on mode.
- `const uint8_t * key`
[in] Pointer to 32-byte key used to generate MAC digest.
- `const uint8_t * otp`
[in] Pointer to 11-byte OTP, optionally included in MAC digest, depending on mode.
- `const uint8_t * sn`
[in] Pointer to 9-byte SN, optionally included in MAC digest, depending on mode.
- `uint8_t * response`
[out] Pointer to 32-byte SHA-256 digest (MAC).
- `struct atca_temp_key * temp_key`
[in,out] Pointer to TempKey structure.

23.40.1 Detailed Description

Input/output parameters for function `atca_mac()`.

23.41 `atca_mbedtls_eckey_s` Struct Reference

```
#include <lib/mbedtls/atca_mbedtls_wrap.h>
```

Data Fields

- `ATCADevice device`
- `uint16_t handle`

23.41.1 Detailed Description

Structure to hold metadata - is written into the mbedtls pk structure as the private key bignum value 'd' which otherwise would be unused. Bignums can be any arbitrary length of bytes

23.42 `atca_nonce_in_out` Struct Reference

Input/output parameters for function `atca_nonce()`.

```
#include <lib/host/atca_host.h>
```

Data Fields

- `uint8_t mode`
[in] Mode parameter used in Nonce command (Param1).
- `uint16_t zero`
[in] Zero parameter used in Nonce command (Param2).
- `const uint8_t * num_in`
[in] Pointer to 20-byte NumIn data used in Nonce command.
- `const uint8_t * rand_out`
[in] Pointer to 32-byte RandOut data from Nonce command.
- `struct atca_temp_key * temp_key`
[in,out] Pointer to TempKey structure.

23.42.1 Detailed Description

Input/output parameters for function `atca_nonce()`.

23.43 atca_resp_mac_in_out Struct Reference

Input/Output parameters for calculating the output response mac in SHA105 device. Used with the `atcah_gen_↔` `output_resp_mac()` function.

```
#include <lib/host/atca_host.h>
```

Data Fields

- `const uint8_t * slot_key`
- `uint8_t mode`
- `uint16_t key_id`
- `const uint8_t * sn`
- `uint8_t * client_resp`
- `uint8_t checkmac_result`
- `uint8_t * mac_output`

23.43.1 Detailed Description

Input/Output parameters for calculating the output response mac in SHA105 device. Used with the `atcah_gen_↔` `output_resp_mac()` function.

23.44 atca_secureboot_enc_in_out Struct Reference

Data Fields

- `const uint8_t * io_key`
IO protection key value (32 bytes)
- `const struct atca_temp_key * temp_key`
Current value of TempKey.
- `const uint8_t * digest`
Plaintext digest as input.
- `uint8_t * hashed_key`
Calculated key is returned here (32 bytes)
- `uint8_t * digest_enc`
Encrypted (ciphertext) digest is return here (32 bytes)

23.45 atca_secureboot_mac_in_out Struct Reference

Data Fields

- `uint8_t mode`
SecureBoot mode (param1)
- `uint16_t param2`
SecureBoot param2.
- `uint16_t secure_boot_config`
SecureBootConfig value from configuration zone.
- `const uint8_t * hashed_key`
Hashed key. SHA256(IO Protection Key | TempKey)
- `const uint8_t * digest`
Digest (unencrypted)
- `const uint8_t * signature`
Signature (can be NULL if not required)
- `uint8_t * mac`
MAC is returned here.

23.46 atca_session_key_in_out Struct Reference

Input/Output paramters for calculating the session key by the nonce command. Used with the `atcah_gen_session_key()` function.

```
#include <lib/host/atca_host.h>
```

Data Fields

- `uint8_t * transport_key`
- `uint16_t transport_key_id`
- `const uint8_t * sn`
- `uint8_t * nonce`
- `uint8_t * session_key`

23.46.1 Detailed Description

Input/Output paramters for calculating the session key by the nonce command. Used with the `atcah_gen_session_key()` function.

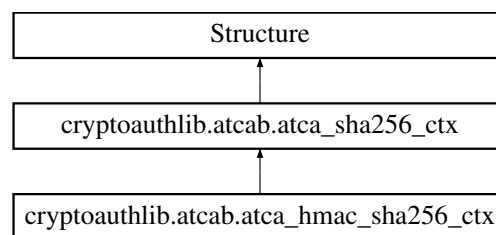
23.47 atca_sha256_ctx Struct Reference

Data Fields

- `uint32_t total_msg_size`
Total number of message bytes processed.
- `uint32_t block_size`
Number of bytes in current block.
- `uint8_t block [ATCA_SHA256_BLOCK_SIZE * 2u]`
Unprocessed message storage.

23.48 cryptoauthlib.atcab.atca_sha256_ctx Class Reference

Inheritance diagram for `cryptoauthlib.atcab.atca_sha256_ctx`:



Static Protected Attributes

- `list_fields_`

23.48.1 Detailed Description

SHA256 context

23.48.2 Field Documentation

23.48.2.1 fields_

```
list cryptoauthlib.atcab.atca_sha256_ctx.fields_ [static], [protected]
```

Initial value:

```
= [{"total_msg_size", c_uint32},
    {"block_size", c_uint32},
    {"block", c_char*64*2}]
```


23.49 atca_sign_internal_in_out Struct Reference

Input/output parameters for calculating the message and digest used by the Sign(internal) command. Used with the atcah_sign_internal_msg() function.

```
#include <lib/host/atca_host.h>
```

Data Fields

- **uint8_t mode**
[in] Sign Mode
- **uint16_t key_id**
[in] Sign KeyID
- **uint16_t slot_config**
[in] SlotConfig[TempKeyFlags.keyId]
- **uint16_t key_config**
[in] KeyConfig[TempKeyFlags.keyId]
- **uint8_t use_flag**
[in] UseFlag[TempKeyFlags.keyId], 0x00 for slots 8 and above and for ATECC508A
- **uint8_t update_count**
[in] UpdateCount[TempKeyFlags.keyId], 0x00 for slots 8 and above and for ATECC508A
- **bool is_slot_locked**
[in] Is TempKeyFlags.keyId slot locked.
- **bool for_invalidate**
[in] Set to true if this will be used for the Verify(Invalidate) command.
- **const uint8_t * sn**
[in] Device serial number SN[0:8] (9 bytes)
- **const struct atca_temp_key * temp_key**
[in] The current state of TempKey.
- **uint8_t * message**
[out] Full 55 byte message the Sign(internal) command will build. Can be NULL if not required.
- **uint8_t * verify_other_data**
[out] The 19 byte OtherData bytes to be used with the Verify(In/Validate) command. Can be NULL if not required.
- **uint8_t * digest**
[out] SHA256 digest of the full 55 byte message. Can be NULL if not required.

23.49.1 Detailed Description

Input/output parameters for calculating the message and digest used by the Sign(internal) command. Used with the atcah_sign_internal_msg() function.

23.50 atca_spi_host_s Struct Reference

Data Fields

- **char spi_file** [20]
- **int f_spi**

23.51 atca_temp_key Struct Reference

Structure to hold TempKey fields.

```
#include <lib/host/atca_host.h>
```

Data Fields

- `uint8_t value` [ATCA_KEY_SIZE *2u]
Value of TempKey (64 bytes for ATECC608 only)
- unsigned `key_id`: 4
If TempKey was derived from a slot or transport key (GenDig or GenKey), that key ID is saved here.
- unsigned `source_flag`: 1
Indicates id TempKey started from a random nonce (0) or not (1).
- unsigned `gen_dig_data`: 1
TempKey was derived from the GenDig command.
- unsigned `gen_key_data`: 1
TempKey was derived from the GenKey command (ATECC devices only).
- unsigned `no_mac_flag`: 1
TempKey was derived from a key that has the NoMac bit set preventing the use of the MAC command. Known as CheckFlag in ATSHA devices).
- unsigned `valid`: 1
TempKey is valid.
- `uint8_t is_64`
TempKey has 64 bytes of valid data.

23.51.1 Detailed Description

Structure to hold TempKey fields.

23.52 atca_uart_host_s Struct Reference

Data Fields

- `char uart_file` [20]
- `int fd_uart`
- `int ref_ct`
- `HANDLE hSerial`

23.53 atca_verify_in_out Struct Reference

Input/output parameters for function atcah_verify().

```
#include <lib/host/atca_host.h>
```

Data Fields

- `uint16_t curve_type`
[in] Curve type used in Verify command (Param2).
- `const uint8_t * signature`
[in] Pointer to ECDSA signature to be verified
- `const uint8_t * public_key`
[in] Pointer to the public key to be used for verification
- `struct atca_temp_key * temp_key`
[in,out] Pointer to TempKey structure.

23.53.1 Detailed Description

Input/output parameters for function `atcah_verify()`.

23.54 atca_verify_mac Struct Reference

Data Fields

- `uint8_t mode`
Mode (Param1) parameter used in Verify command.
- `uint16_t key_id`
KeyID (Param2) used in Verify command.
- `const uint8_t * signature`
Signature used in Verify command (64 bytes).
- `const uint8_t * other_data`
OtherData used in Verify command (19 bytes).
- `const uint8_t * msg_dig_buf`
Message digest buffer (64 bytes).
- `const uint8_t * io_key`
IO protection key value (32 bytes).
- `const uint8_t * sn`
Serial number (9 bytes).
- `const atca_temp_key_t * temp_key`
TempKey.
- `uint8_t * mac`
Calculated verification MAC is returned here (32 bytes).

23.55 atca_write_mac_in_out Struct Reference

Input/output parameters for function `atcah_write_auth_mac()` and `atcah_privwrite_auth_mac()`.

```
#include <lib/host/atca_host.h>
```

Data Fields

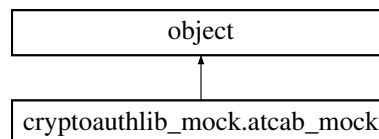
- `uint8_t zone`
Zone/Param1 for the Write or PrivWrite command.
- `uint16_t key_id`
KeyID/Param2 for the Write or PrivWrite command.
- `const uint8_t * sn`
Device serial number SN[0:8]. Only SN[0:1] and SN[8] are required though.
- `const uint8_t * input_data`
Data to be encrypted. 32 bytes for Write command, 36 bytes for PrivWrite command.
- `uint8_t * encrypted_data`
Encrypted version of input_data will be returned here. 32 bytes for Write command, 36 bytes for PrivWrite command.
- `uint8_t * auth_mac`
Write MAC will be returned here. 32 bytes.
- `struct atca_temp_key * temp_key`
Current state of TempKey.

23.55.1 Detailed Description

Input/output parameters for function `atcah_write_auth_mac()` and `atcah_privwrite_auth_mac()`.

23.56 cryptoauthlib_mock.atcab_mock Class Reference

Inheritance diagram for `cryptoauthlib_mock.atcab_mock`:



Public Member Functions

- `def atcab_init (self)`
- `def atcab_release (self)`
- `def atcab_get_device_type (self)`
- `def atcab_aes (self, mode, key_id, aes_in, aes_out)`
- `def atcab_aes_encrypt (self, key_id, key_block, plaintext, ciphertext)`
- `def atcab_aes_decrypt (self, key_id, key_block, ciphertext, plaintext)`
- `def atcab_aes_gfm (self, hash_key, inp, output)`
- `def atcab_aes_cbc_init (self, ctx, key_id, key_block, iv)`
- `def atcab_aes_cbc_encrypt_block (self, ctx, plaintext, ciphertext)`
- `def atcab_aes_cbc_decrypt_block (self, ctx, ciphertext, plaintext)`
- `def atcab_aes_cmac_init (self, ctx, key_id, key_block)`
- `def atcab_aes_cmac_update (self, ctx, data, data_size)`
- `def atcab_aes_cmac_finish (self, ctx, cmac, size)`
- `def atcab_aes_ctr_init (self, ctx, key_id, key_block, counter_size, iv)`
- `def atcab_aes_ctr_init_rand (self, ctx, key_id, key_block, counter_size, iv)`

- def **atcab_aes_ctr_encrypt_block** (self, ctx, plaintext, ciphertext)
- def **atcab_aes_ctr_decrypt_block** (self, ctx, ciphertext, plaintext)
- def **atcab_aes_gcm_init** (self, ctx, key_id, key_block, iv, iv_size)
- def **atcab_aes_gcm_init_rand** (self, ctx, key_id, key_block, rand_size, free_field, free_field_size, iv)
- def **atcab_aes_gcm_aad_update** (self, ctx, aad, aad_size)
- def **atcab_aes_gcm_encrypt_update** (self, ctx, plaintext, plaintext_size, ciphertext)
- def **atcab_aes_gcm_encrypt_finish** (self, ctx, tag, tag_size)
- def **atcab_aes_gcm_decrypt_update** (self, ctx, ciphertext, ciphertext_size, plaintext)
- def **atcab_aes_gcm_decrypt_finish** (self, ctx, tag, tag_size, is_verified)
- def **atcab_aes_cbcmac_init** (self, ctx, key_id, key_block)
- def **atcab_aes_cbcmac_update** (self, ctx, data, data_size)
- def **atcab_aes_cbcmac_finish** (self, ctx, mac, mac_size)
- def **atcab_aes_ccm_init** (self, ctx, key_id, key_block, iv, iv_size, aad_size, text_size, tag_size)
- def **atcab_aes_ccm_init_rand** (self, ctx, key_id, key_block, iv, iv_size, aad_size, text_size, tag_size)
- def **atcab_aes_ccm_aad_update** (self, ctx, aad, aad_size)
- def **atcab_aes_ccm_aad_finish** (self, ctx)
- def **atcab_aes_ccm_encrypt_update** (self, ctx, plaintext, plaintext_size, ciphertext)
- def **atcab_aes_ccm_decrypt_update** (self, ctx, ciphertext, ciphertext_size, plaintext)
- def **atcab_aes_ccm_encrypt_finish** (self, ctx, tag, tag_size)
- def **atcab_aes_ccm_decrypt_finish** (self, ctx, tag, is_verified)
- def **atcab_checkmac** (self, mode, key_id, challenge, response, other_data)
- def **atcab_counter** (self, mode, counter_id, counter_value)
- def **atcab_counter_increment** (self, counter_id, counter_value)
- def **atcab_counter_read** (self, counter_id, counter_value)
- def **atcab_derivekey** (self, mode, target_key, mac)
- def **atcab_ecdh_base** (self, mode, key_id, public_key, pms, out_nonce)
- def **atcab_ecdh** (self, key_id, public_key, pms)
- def **atcab_ecdh_enc** (self, key_id, public_key, pms, read_key, read_key_id, num_in)
- def **atcab_ecdh_ioenc** (self, key_id, public_key, pms, io_key)
- def **atcab_ecdh_tempkey** (self, public_key, pms)
- def **atcab_ecdh_tempkey_ioenc** (self, public_key, pms, io_key)
- def **atcab_gendig** (self, zone, key_id, other_data, other_data_size)
- def **atcab_genkey_base** (self, mode, key_id, other_data, public_key)
- def **atcab_genkey** (self, key_id, public_key)
- def **atcab_get_pubkey** (self, key_id, public_key)
- def **atcab_hmac** (self, mode, key_id, digest)
- def **atcab_info_base** (self, mode, param2, out_data)
- def **atcab_info** (self, revision)
- def **atcab_info_get_latch** (self, state)
- def **atcab_info_set_latch** (self, state)
- def **atcab_kdf** (self, mode, key_id, details, message, out_data, out_nonce)
- def **atcab_lock** (self, mode, summary_crc)
- def **atcab_lock_config_zone** (self)
- def **atcab_lock_config_zone_crc** (self, summary_crc)
- def **atcab_lock_data_zone** (self)
- def **atcab_lock_data_zone_crc** (self, summary_crc)
- def **atcab_lock_data_slot** (self, slot)
- def **atcab_mac** (self, mode, key_id, challenge, digest)
- def **atcab_nonce_base** (self, mode, zero, num_in, rand_out)
- def **atcab_nonce** (self, num_in)
- def **atcab_nonce_load** (self, target, num_in, num_in_size)
- def **atcab_nonce_rand** (self, num_in, rand_out)
- def **atcab_challenge** (self, num_in)
- def **atcab_challenge_seed_update** (self, num_in, rand_out)
- def **atcab_priv_write** (self, key_id, priv_key, write_key_id, write_key, num_in)

- def **atcab_random** (self, random_number)
- def **atcab_read_zone** (self, zone, slot, block, offset, data, length)
- def **atcab_read_serial_number** (self, serial_number)
- def **atcab_is_slot_locked** (self, slot, is_locked)
- def **atcab_is_locked** (self, zone, is_locked)
- def **atcab_read_enc** (self, key_id, block, data, enc_key, enc_key_id, num_in)
- def **atcab_read_config_zone** (self, config_data)
- def **atcab_cmp_config_zone** (self, config_data, same_config)
- def **atcab_read_sig** (self, slot, sig)
- def **atcab_read_pubkey** (self, slot, public_key)
- def **atcab_read_bytes_zone** (self, zone, slot, offset, data, length)
- def **atcab_secureboot** (self, mode, param2, digest, signature, mac)
- def **atcab_secureboot_mac** (self, mode, digest, signature, num_in, io_keys, is_verified)
- def **atcab_selftest** (self, mode, param2, result)
- def **atcab_sha_base** (self, mode, length, message, data_out, data_out_size)
- def **atcab_sha_start** (self)
- def **atcab_sha_update** (self, message)
- def **atcab_sha_end** (self, digest, length, message)
- def **atcab_sha_read_context** (self, context, context_size)
- def **atcab_sha_write_context** (self, context, context_size)
- def **atcab_sha** (self, length, message, digest)
- def **atcab_hw_sha2_256_init** (self, ctx)
- def **atcab_hw_sha2_256_update** (self, ctx, data, data_size)
- def **atcab_hw_sha2_256_finish** (self, ctx, digest)
- def **atcab_hw_sha2_256** (self, data, data_size, digest)
- def **atcab_sha_hmac_init** (self, ctx, key_slot)
- def **atcab_sha_hmac_update** (self, ctx, data, data_size)
- def **atcab_sha_hmac_finish** (self, ctx, digest, target)
- def **atcab_sha_hmac** (self, data, data_size, key_slot, digest, target)
- def **atcab_sign_base** (self, mode, key_id, signature)
- def **atcab_sign** (self, key_id, msg, signature)
- def **atcab_sign_internal** (self, key_id, is_invalidate, is_full_sn, signature)
- def **atcab_updateextra** (self, mode, new_value)
- def **atcab_verify** (self, mode, key_id, signature, public_key, other_data, mac)
- def **atcab_verify_extern_stored_mac** (self, mode, key_id, message, signature, public_key, num_in, io_key, is_verified)
- def **atcab_verify_extern** (self, message, signature, public_key, is_verified)
- def **atcab_verify_extern_mac** (self, message, signature, public_key, num_in, io_key, is_verified)
- def **atcab_verify_stored** (self, message, signature, key_id, is_verified)
- def **atcab_verify_stored_mac** (self, message, signature, key_id, num_in, io_key, is_verified)
- def **atcab_verify_validate** (self, key_id, signature, other_data, is_verified)
- def **atcab_verify_invalidate** (self, key_id, signature, other_data, is_verified)
- def **atcab_write** (self, zone, [address](#), value, mac)
- def **atcab_write_zone** (self, zone, slot, block, offset, data, length)
- def **atcab_write_enc** (self, key_id, block, data, enc_key, enc_key_id, num_in)
- def **atcab_write_config_zone** (self, conf)
- def **atcab_write_pubkey** (self, slot, public_key)
- def **atcab_write_bytes_zone** (self, zone, slot, offset_bytes, data, length)
- def **atcab_write_config_counter** (self, counter_id, counter_value)
- def **atcacert_get_response** (self, device_private_key_slot, challenge, response)
- def **atcacert_read_cert** (self, cert_def, ca_public_key, cert, cert_size)
- def **atcacert_write_cert** (self, cert_def, cert, cert_size)
- def **atcacert_create_csr** (self, csr_def, csr, csr_size)
- def **atcacert_create_csr_pem** (self, csr_def, csr, csr_size)
- def **atcacert_date_enc** (self, format, timestamp, formatted_date, formatted_date_size)

- def **atcacert_date_dec** (self, format, formatted_date, formatted_date_size, timestamp)
- def **atcacert_date_enc_compcert** (self, issue_date, expire_years, enc_dates)
- def **atcacert_date_dec_compcert** (self, enc_dates, expire_date_format, issue_date, expire_date)
- def **atcacert_date_get_max_date** (self, date_format, timestamp)
- def **atcacert_max_cert_size** (self, cert_def, max_cert_size)
- def **tng_get_device_pubkey** (self, public_key)
- def **tng_atcacert_max_device_cert_size** (self, max_cert_size)
- def **tng_atcacert_read_device_cert** (self, cert, cert_size, signer_cert)
- def **tng_atcacert_device_public_key** (self, public_key, cert)
- def **tng_atcacert_max_signer_cert_size** (self, max_cert_size)
- def **tng_atcacert_read_signer_cert** (self, cert, cert_size)
- def **tng_atcacert_signer_public_key** (self, public_key, cert)
- def **tng_atcacert_root_cert_size** (self, cert_size)
- def **tng_atcacert_root_cert** (self, cert, cert_size)
- def **tng_atcacert_root_public_key** (self, public_key)
- def **sha206a_generate_derive_key** (self, parent_key, derived_key, param1, param2)
- def **sha206a_diversify_parent_key** (self, parent_key, diversified_key)
- def **sha206a_generate_challenge_response_pair** (self, key, challenge, response)
- def **sha206a_authenticate** (self, challenge, expected_response, is_verified)
- def **sha206a_write_data_store** (self, slot, data, block, offset, length, lock_after_write)
- def **sha206a_read_data_store** (self, slot, data, offset, length)
- def **sha206a_get_data_store_lock_status** (self, slot, is_locked)
- def **sha206a_get_dk_update_count** (self, dk_update_count)
- def **sha206a_get_pk_useflag_count** (self, pk_avail_count)
- def **sha206a_get_dk_useflag_count** (self, dk_avail_count)
- def **sha206a_check_pk_useflag_validity** (self, is_consumed)
- def **sha206a_check_dk_useflag_validity** (self, is_consumed)
- def **sha206a_verify_device_consumption** (self, is_consumed)

Static Public Attributes

- int **r_devtype** = 3
- create_string_buffer **r_aes_out** = create_string_buffer(16)
- **value**
- create_string_buffer **r_ciphertext** = create_string_buffer(16)
- create_string_buffer **r_plaintext** = create_string_buffer(16)
- create_string_buffer **r_aes_gfm_output** = create_string_buffer(16)
- create_string_buffer **r_aes_cmac_output** = create_string_buffer(16)
- create_string_buffer **r_aes_ctr_output** = create_string_buffer(16)
- create_string_buffer **r_iv** = create_string_buffer(16)
- create_string_buffer **r_tag** = create_string_buffer(16)
- c_uint8 **r_is_verified** = c_uint8()
- create_string_buffer **r_aes CBCMAC_output** = create_string_buffer(16)
- c_uint8 **r_tag_size** = c_uint8()
- c_uint32 **r_counter_value** = c_uint32()
- create_string_buffer **r_ecdh_pms** = create_string_buffer(32)
- create_string_buffer **r_ecdh_out_nonce** = create_string_buffer(32)
- create_string_buffer **r_genkey_pubkey** = create_string_buffer(64)
- create_string_buffer **r_hmac_digest** = create_string_buffer(32)
- create_string_buffer **r_revision** = create_string_buffer(4)
- c_uint8 **r_latch_state** = c_uint8()
- create_string_buffer **r_kdf_out_data** = create_string_buffer(64)
- create_string_buffer **r_kdf_out_nonce** = create_string_buffer(32)
- create_string_buffer **r_mac_digest** = create_string_buffer(32)

- create_string_buffer **r_nonce_rand_out** = create_string_buffer(32)
- create_string_buffer **r_rand_out** = create_string_buffer(32)
- create_string_buffer **r_read_zone_data** = create_string_buffer(32)
- create_string_buffer **r_ser_num** = create_string_buffer(9)
- c_uint8 **r_is_locked** = c_uint8()
- create_string_buffer **r_read_enc_data** = create_string_buffer(32)
- create_string_buffer **r_read_config_data** = create_string_buffer(128)
- c_uint8 **r_same_config** = c_uint8()
- create_string_buffer **r_read_sig** = create_string_buffer(64)
- create_string_buffer **r_read_pubkey** = create_string_buffer(64)
- create_string_buffer **r_read_bytes_zone_data** = create_string_buffer(64)
- create_string_buffer **r_sboot_mac** = create_string_buffer(32)
- c_uint8 **r_sboot_is_verified** = c_uint8()
- c_uint8 **r_stest_res** = c_uint8()
- create_string_buffer **r_sha_base_data** = create_string_buffer(130)
- c_uint8 **r_sha_base_data_size** = c_uint8()
- create_string_buffer **r_sha_digest** = create_string_buffer(32)
- create_string_buffer **r_sha_context_data** = create_string_buffer(130)
- c_uint8 **r_sha_context_size** = c_uint8()
- create_string_buffer **r_signature** = create_string_buffer(64)
- create_string_buffer **r_mac** = create_string_buffer(64)
- c_uint8 **r_verify_is_verified** = c_uint8()
- create_string_buffer **r_response** = create_string_buffer(64)
- c_size_t **r_cert_size** = c_size_t(64)
- create_string_buffer **r_cert** = create_string_buffer(r_cert_size.value)
- c_uint8 **r_csr_size** = c_uint8()
- create_string_buffer **r_csr** = create_string_buffer(64)
- create_string_buffer **r_formatted_date** = create_string_buffer(3)
- c_uint8 **r_formatted_date_size** = c_uint8()
- create_string_buffer **r_enc_dates** = create_string_buffer(3)
- c_size_t **r_max_cert_size** = c_size_t(123)
- c_int **r_tng_type** = c_int(1)
- create_string_buffer **r_derived_key** = create_string_buffer(32)
- create_string_buffer **r_diversified_key** = create_string_buffer(32)
- create_string_buffer **r_challenge_response** = create_string_buffer(32)
- c_uint8 **r_verify_is_locked** = c_uint8()
- c_uint8 **r_dk_update_count** = c_uint8()
- c_uint8 **r_pk_avail_count** = c_uint8()
- c_uint8 **r_dk_avail_count** = c_uint8()
- c_uint8 **r_verify_is_consumed** = c_uint8()

23.57 atcac_aes_cmac_ctx Struct Reference

Data Fields

- mbedtls_cipher_context_t **mctx**
- void * **ptr**

23.58 atcac_aes_gcm_ctx Struct Reference

Data Fields

- mbedtls_cipher_context_t **mctx**

23.59 atcac_hmac_ctx Struct Reference

Data Fields

- mbedtls_md_context_t * **mctx**
- void * **ptr**

23.60 atcac_pk_ctx Struct Reference

Data Fields

- mbedtls_pk_context **mctx**
- void * **ptr**

23.61 atcac_sha1_ctx Struct Reference

Data Fields

- mbedtls_md_context_t **mctx**
- void * **ptr**

23.62 atcac_sha2_256_ctx Struct Reference

Data Fields

- void * **ptr**

23.63 atcac_sha2_384_ctx Struct Reference

Data Fields

- void * **ptr**

23.64 atcac_sha2_512_ctx Struct Reference

Data Fields

- void * **ptr**

23.65 atcac_x509_ctx Struct Reference

Data Fields

- void * **ptr**

23.66 atcacert_build_state_s Struct Reference

```
#include <lib/atcacert/atcacert_def.h>
```

Data Fields

- const [atcacert_def_t](#) * **cert_def**
Certificate definition for the certificate being rebuilt.
- uint8_t * **cert**
Buffer to contain the rebuilt certificate.
- size_t * **cert_size**
Current size of the certificate in bytes.
- size_t **max_cert_size**
Max size of the cert buffer in bytes.
- uint8_t **is_device_sn**
Indicates the structure contains the device SN.
- ATCADeviceType **devtype**
Device type info for the certificate being rebuilt.
- uint8_t **device_sn** [9]
Storage for the device SN, when it's found.
- uint8_t **is_comp_cert**
Indicates the structure contains the compressed certificate.
- uint8_t **comp_cert** [ATCACERT_COMP_CERT_MAX_SIZE]
Storage for the compressed certificate when it's found.

23.66.1 Detailed Description

Tracks the state of a certificate as it's being rebuilt from device information.

23.67 atcacert_cert_element_s Struct Reference

```
#include <lib/atcacert/atcacert_def.h>
```

Data Fields

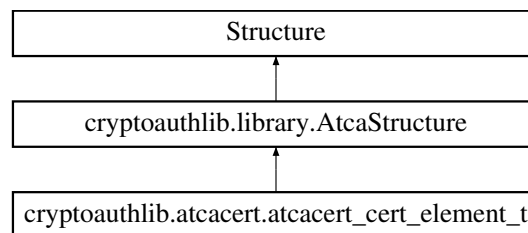
- char **id** [25]
ID identifying this element.
- [atcacert_device_loc_t](#) **device_loc**
Location in the device for the element.
- [atcacert_cert_loc_t](#) **cert_loc**
Location in the certificate template for the element.
- [atcacert_transform_t](#) **transforms** [2]
List of transforms from device to cert for this element.

23.67.1 Detailed Description

Defines a generic dynamic element for a certificate including the device and template locations.

23.68 cryptauthlib.atcacert.atcacert_cert_element_t Class Reference

Inheritance diagram for cryptauthlib.atcacert.atcacert_cert_element_t:



Static Protected Attributes

- int **__pack__** = 1
- dict **__def__**

Additional Inherited Members

Public Member Functions inherited from [cryptauthlib.library.AtcaStructure](#)

- None **__init__** (self, *args, **kwargs)
- def **from_definition** (cls)
- def **check_rationality** (cls)
- def **get_field_definition** (cls, str name)
- Any **__getattr__** (self, str name)
- def **__iter__** (self)
- def **__str__** (self)
- def **to_c_code** (self, name=None, **kwargs)
- def **update_from_buffer** (self, buffer)

23.68.1 Detailed Description

CTypes mirror of atcacert_cert_element_t from atcacert_def.h

23.68.2 Field Documentation

23.68.2.1 _def_

```
dict cryptoauthlib.atcacert.atcacert_cert_element_t._def_ [static], [protected]
```

Initial value:

```
= {
    'id': (c_char, 25), # ID identifying this element.
    'device_loc': (atcacert_device_loc_t,), # Location in the device for the element.
    'cert_loc': (atcacert_cert_loc_t,), # Location in the certificate template for the element.
    'transforms': (atcacert_transform_t, 2) # Transforms for converting the device data.
}
```

23.69 atcacert_cert_loc_s Struct Reference

```
#include <lib/atcacert/atcacert_def.h>
```

Data Fields

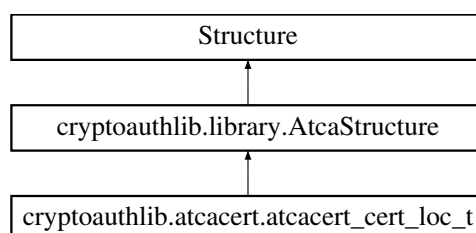
- **uint16_t offset**
Byte offset in the certificate template.
- **uint16_t count**
Byte count. Set to 0 if it doesn't exist.

23.69.1 Detailed Description

Defines a chunk of data in a certificate template.

23.70 cryptoauthlib.atcacert.atcacert_cert_loc_t Class Reference

Inheritance diagram for cryptoauthlib.atcacert.atcacert_cert_loc_t:



Static Protected Attributes

- `int __pack__ = 1`
- `list __fields__ = [('offset', c_uint16), ('count', c_uint16)]`

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.library.AtcaStructure](#)

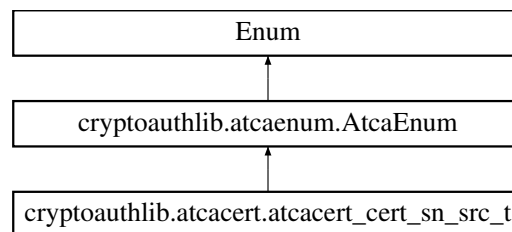
- `None __init__ (self, *args, **kwargs)`
- `def from_definition (cls)`
- `def check_rationality (cls)`
- `def get_field_definition (cls, str name)`
- `Any __getattr__ (self, str name)`
- `def __iter__ (self)`
- `def __str__ (self)`
- `def to_c_code (self, name=None, **kwargs)`
- `def update_from_buffer (self, buffer)`

23.70.1 Detailed Description

CTypes mirror of `atcacert_cert_loc_t` from `atcacert_def.h`

23.71 cryptoauthlib.atcacert.atcacert_cert_sn_src_t Class Reference

Inheritance diagram for `cryptoauthlib.atcacert.atcacert_cert_sn_src_t`:



Static Public Attributes

- `int SNSRC_STORED = 0x0`
- `int SNSRC_STORED_DYNAMIC = 0x7`
- `int SNSRC_DEVICE_SN = 0x8`
- `int SNSRC_SIGNER_ID = 0x9`
- `int SNSRC_PUB_KEY_HASH = 0xA`
- `int SNSRC_DEVICE_SN_HASH = 0xB`
- `int SNSRC_PUB_KEY_HASH_POS = 0xC`
- `int SNSRC_DEVICE_SN_HASH_POS = 0xD`
- `int SNSRC_PUB_KEY_HASH_RAW = 0xE`
- `int SNSRC_DEVICE_SN_HASH_RAW = 0xF`

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

- `def __str__ (self)`
- `def __eq__ (self, other)`
- `def __ne__ (self, other)`
- `def __int__ (self)`
- `def __hash__ (self)`

Data Fields inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

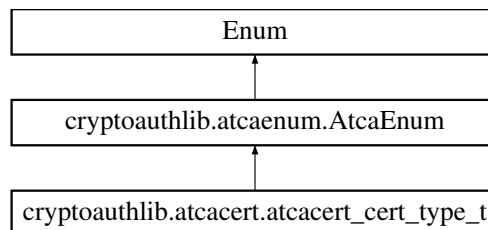
- `name`
- `value`

23.71.1 Detailed Description

Sources for the certificate serial number

23.72 cryptoauthlib.atcacert.atcacert_cert_type_t Class Reference

Inheritance diagram for `cryptoauthlib.atcacert.atcacert_cert_type_t`:



Static Public Attributes

- `int CERTTYPE_X509 = 0`
- `int CERTTYPE_CUSTOM = 1`
- `int CERTTYPE_X509_FULL_STORED = 2`

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

- `def __str__ (self)`
- `def __eq__ (self, other)`
- `def __ne__ (self, other)`
- `def __int__ (self)`
- `def __hash__ (self)`

Data Fields inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

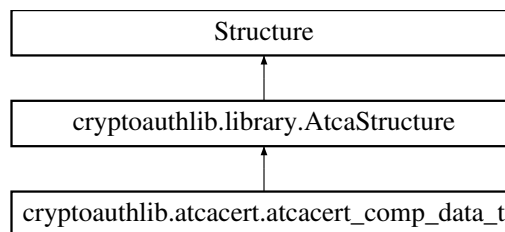
- **name**
- **value**

23.72.1 Detailed Description

Types of certificates

23.73 [cryptoauthlib.atcacert.atcacert_comp_data_t](#) Class Reference

Inheritance diagram for [cryptoauthlib.atcacert.atcacert_comp_data_t](#):



Static Protected Attributes

- `int __pack_ = 1`
- `int __size_ = 72`
- `list __fields_`

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.library.AtcaStructure](#)

- `None __init__ (self, *args, **kwargs)`
- `def from_definition (cls)`
- `def check_rationality (cls)`
- `def get_field_definition (cls, str name)`
- `Any __getattr__ (self, str name)`
- `def __iter__ (self)`
- `def __str__ (self)`
- `def to_c_code (self, name=None, **kwargs)`
- `def update_from_buffer (self, buffer)`

23.73.1 Detailed Description

CTypes definition of certificate signature storage which includes other certificate metadata which is why it's often identified as "compresessed cert" for the slot in configurators

23.73.2 Field Documentation

23.73.2.1 _fields_

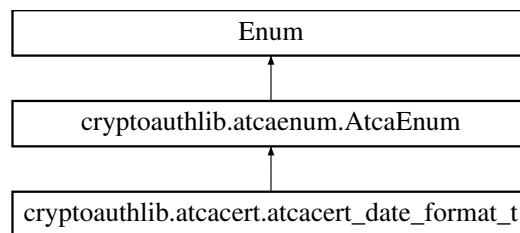
list cryptoauthlib.atcacert.atcacert_comp_data_t._fields_ [static], [protected]

Initial value:

```
= [
    ('r', c_uint8*32),          # P256 signature 'r' value - big endian
    ('s', c_uint8*32),          # P256 signature 's' value - big endian
    ('year', c_uint64, 5),      # Years after 2000
    ('month', c_uint64, 4),     # Month (0 - 11), see atcacert_tm_utc_t
    ('day', c_uint64, 5),       # Day (1 - 31), see atcacert_tm_utc_t
    ('hour', c_uint64, 5),      # Hour (0 - 23), see atcacert_tm_utc_t
    ('expire', c_uint64, 5),     # Expire years (<=31)
    ('signer_id', c_uint64, 16), # Value used in the signing cert subject name
    ('chain_id', c_uint64, 4),   # Revision identifier
    ('template_id', c_uint64, 4), # Location in a chain
    ('reserved_70_4', c_uint64, 4), # Reserved - lower four bits of byte 70
    ('sn_source', c_uint64, 4),  # Serial number format, see atcacert_cert_sn_src_t
    ('reserved_71_8', c_uint64, 8) # Reserved - byte 71
]
```

23.74 cryptoauthlib.atcacert.atcacert_date_format_t Class Reference

Inheritance diagram for cryptoauthlib.atcacert.atcacert_date_format_t:



Static Public Attributes

- int `DATEFMT_ISO8601_SEP` = 0
- int `DATEFMT_RFC5280.UTC` = 1
- int `DATEFMT_POSIX_UINT32_BE` = 2
- int `DATEFMT_POSIX_UINT32_LE` = 3
- int `DATEFMT_RFC5280_GEN` = 4

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

- def `__str__` (self)
- def `__eq__` (self, other)
- def `__ne__` (self, other)
- def `__int__` (self)
- def `__hash__` (self)

Data Fields inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

- **name**
- **value**

23.74.1 Detailed Description

Support Date formats by the atcacert

23.75 atcacert_def_s Struct Reference

```
#include <lib/atcacert/atcacert_def.h>
```

Data Fields

- [atcacert_cert_type_t](#) **type**
Certificate type.
- [atcacert_device_loc_t](#) **comp_cert_dev_loc**
Where on the device the compressed cert can be found.
- [uint16_t](#) **private_key_slot**
If this is a device certificate template, this is the device slot for the device private key.
- [const uint8_t *](#) **cert_template**
Pointer to the actual certificate template data.
- [uint16_t](#) **cert_template_size**
Size of the certificate template in cert_template in bytes.
- [const struct \[atcacert_def_s\]\(#\) *](#) **ca_cert_def**
Certificate definition of the CA certificate.

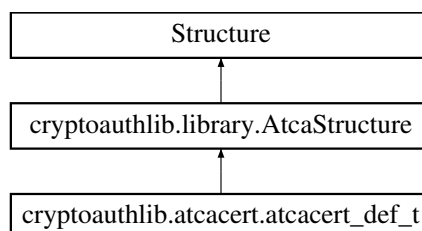
23.75.1 Detailed Description

Defines a certificate and all the pieces to work with it.

If any of the standard certificate elements (`std_cert_elements`) are not a part of the certificate definition, set their count to 0 to indicate their absence.

23.76 cryptoauthlib.atcacert.atcacert_def_t Class Reference

Inheritance diagram for `cryptoauthlib.atcacert.atcacert_def_t`:



Static Protected Attributes

- `_def_`

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.library.AtcaStructure](#)

- None `__init__` (self, *args, **kwargs)
- def `from_definition` (cls)
- def `check_rationality` (cls)
- def `get_field_definition` (cls, str name)
- Any `__getattr__` (self, str name)
- def `__iter__` (self)
- def `__str__` (self)
- def `to_c_code` (self, name=None, **kwargs)
- def `update_from_buffer` (self, buffer)

23.76.1 Detailed Description

CTypes mirror of `atcacert_def_t` from `atcacert_def.h`

23.77 atcacert_device_loc_s Struct Reference

```
#include <lib/atcacert/atcacert_def.h>
```

Data Fields

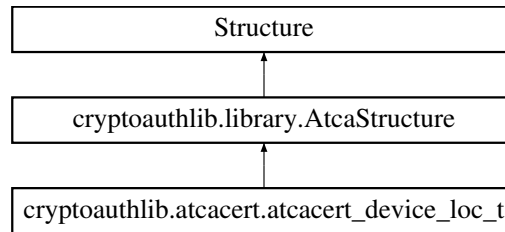
- [atcacert_device_zone_t](#) **zone**
Zone in the device.
- `uint16_t` **slot**
Slot within the data zone. Only applies if zone is DEVZONE_DATA.
- `uint8_t` **is_genkey**
If true, use GenKey command to get the contents instead of Read.
- `uint16_t` **offset**
Byte offset in the zone.
- `uint16_t` **count**
Byte count.

23.77.1 Detailed Description

Defines a chunk of data in an ATECC device.

23.78 cryptoauthlib.atcacert.atcacert_device_loc_t Class Reference

Inheritance diagram for cryptoauthlib.atcacert.atcacert_device_loc_t:



Static Protected Attributes

- int `__pack__` = 1
- dict `__def__`

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.library.AtcaStructure](#)

- None `__init__` (self, *args, **kwargs)
- def `from_definition` (cls)
- def `check_rationality` (cls)
- def `get_field_definition` (cls, str name)
- Any `__getattr__` (self, str name)
- def `__iter__` (self)
- def `__str__` (self)
- def `to_c_code` (self, name=None, **kwargs)
- def `update_from_buffer` (self, buffer)

23.78.1 Detailed Description

CTypes mirror of atcacert_device_loc_t from atcacert_def.h

23.78.2 Field Documentation

23.78.2.1 `__def__`

dict cryptoauthlib.atcacert.atcacert_device_loc_t.`__def__` [static], [protected]

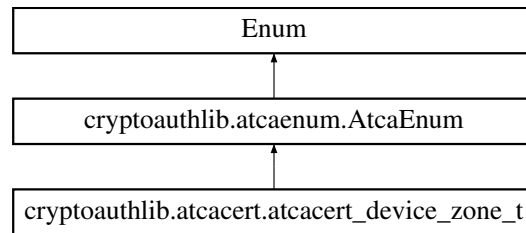
Initial value:

```

= {
    'zone': (atcacert_device_zone_t,), # Zone in the device.
    'slot': (c_uint16,), # Slot within the data zone. Only applies if zone is DEVZONE_DATA.
    'is_genkey': (c_uint8,), # If true, use GenKey command to get the contents instead of Read.
    'offset': (c_uint16,), # Byte offset in the zone.
    'count': (c_uint16,) # Byte count.
}
  
```

23.79 cryptoauthlib.atcacert.atcacert_device_zone_t Class Reference

Inheritance diagram for cryptoauthlib.atcacert.atcacert_device_zone_t:



Static Public Attributes

- int **DEVZONE_CONFIG** = 0x00
- int **DEVZONE_OTP** = 0x01
- int **DEVZONE_DATA** = 0x02
- int **DEVZONE_GENKEY** = 0x03
- int **DEVZONE_DEDICATED_DATA** = 0x04
- int **DEVZONE_NONE** = 0x07

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

- def **__str__** (self)
- def **__eq__** (self, other)
- def **__ne__** (self, other)
- def **__int__** (self)
- def **__hash__** (self)

Data Fields inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

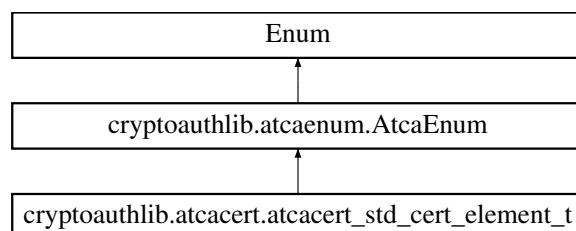
- **name**
- **value**

23.79.1 Detailed Description

ATECC device zones. The values match the Zone Encodings as specified in the datasheet

23.80 cryptoauthlib.atcacert.atcacert_std_cert_element_t Class Reference

Inheritance diagram for cryptoauthlib.atcacert.atcacert_std_cert_element_t:



Static Public Attributes

- int **STDCERT_PUBLIC_KEY** = 0
- int **STDCERT_SIGNATURE** = 1
- int **STDCERT_ISSUE_DATE** = 2
- int **STDCERT_EXPIRE_DATE** = 3
- int **STDCERT_SIGNER_ID** = 4
- int **STDCERT_CERT_SN** = 5
- int **STDCERT_AUTH_KEY_ID** = 6
- int **STDCERT_SUBJ_KEY_ID** = 7

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

- def **__str__** (self)
- def **__eq__** (self, other)
- def **__ne__** (self, other)
- def **__int__** (self)
- def **__hash__** (self)

Data Fields inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

- **name**
- **value**

23.80.1 Detailed Description

Standard dynamic certificate elements

23.81 atcacert_tm_utc_s Struct Reference

```
#include <lib/atcacert/atcacert_date.h>
```

Data Fields

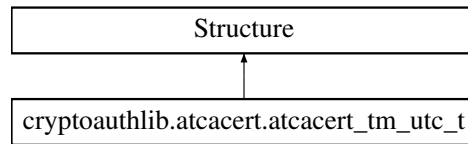
- int **tm_sec**
- int **tm_min**
- int **tm_hour**
- int **tm_mday**
- int **tm_mon**
- int **tm_year**

23.81.1 Detailed Description

Holds a broken-down date in UTC. Mimics atcacert_tm_utc_t from time.h.

23.82 cryptoauthlib.atcacert.atcacert_tm_utc_t Class Reference

Inheritance diagram for cryptoauthlib.atcacert.atcacert_tm_utc_t:



Public Member Functions

- `def __init__(self, *args, **kwargs)`

Static Protected Attributes

- `list _fields_`

23.82.1 Detailed Description

CTypes mirror of atcacert_tm_utc_t from atcacert_date.h which mimics the posix time structure

23.82.2 Field Documentation

23.82.2.1 `_fields_`

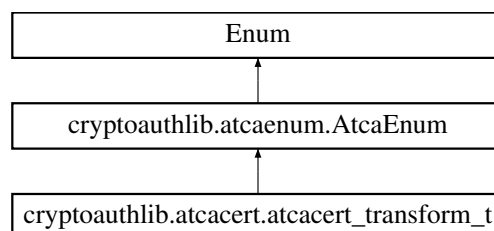
```
list cryptoauthlib.atcacert.atcacert_tm_utc_t._fields_ [static], [protected]
```

Initial value:

```
= [
    ('tm_sec', c_int),    # 0 to 59
    ('tm_min', c_int),    # 0 to 59
    ('tm_hour', c_int),   # 0 to 23
    ('tm_mday', c_int),   # 1 to 31
    ('tm_mon', c_int),    # 0 to 11
    ('tm_year', c_int),   # years since 1900
]
```

23.83 cryptoauthlib.atcacert.atcacert_transform_t Class Reference

Inheritance diagram for cryptoauthlib.atcacert.atcacert_transform_t:



Static Public Attributes

- `int TF_NONE = 0x00`
- `int TF_REVERSE = 0x01`
- `int TF_BIN2HEX_UC = 0x02`
- `int TF_BIN2HEX_LC = 0x03`
- `int TF_HEX2BIN_UC = 0x04`
- `int TF_HEX2BIN_LC = 0x05`
- `int TF_BIN2HEX_SPACE_UC = 0x06`
- `int TF_BIN2HEX_SPACE_LC = 0x07`
- `int TF_HEX2BIN_SPACE_UC = 0x08`
- `int TF_HEX2BIN_SPACE_LC = 0x09`

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

- `def __str__(self)`
- `def __eq__(self, other)`
- `def __ne__(self, other)`
- `def __int__(self)`
- `def __hash__(self)`

Data Fields inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

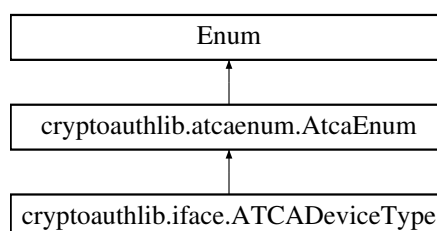
- `name`
- `value`

23.83.1 Detailed Description

Transforms for converting the device data.

23.84 cryptoauthlib.iface.ATCADeviceType Class Reference

Inheritance diagram for `cryptoauthlib.iface.ATCADeviceType`:



Static Public Attributes

- int **ATSHA204A** = 0
- int **ATECC108A** = 1
- int **ATECC508A** = 2
- int **ATECC608A** = 3
- int **ATECC608B** = 3
- int **ATECC608** = 3
- int **ATSHA206A** = 4
- int **TA100** = 0x10
- int **TA101** = 0x11
- int **ECC204** = 0x20
- int **TA010** = 0x21
- int **ECC206** = 0x22
- int **RNG90** = 0x23
- int **SHA104** = 0x24
- int **SHA105** = 0x25
- int **SHA106** = 0x26
- int **ATCA_DEV_UNKNOWN** = 0x7E
- int **ATCA_DEV_INVALID** = 0x7F

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

- def **__str__** (self)
- def **__eq__** (self, other)
- def **__ne__** (self, other)
- def **__int__** (self)
- def **__hash__** (self)

Data Fields inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

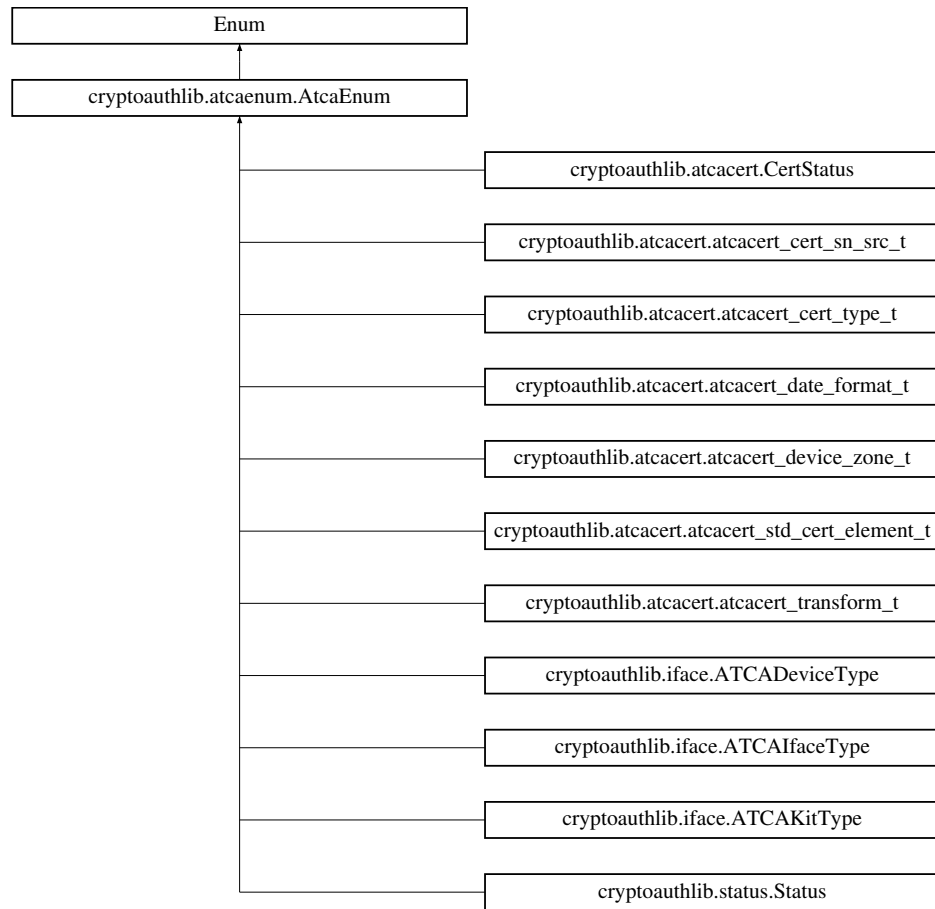
- **name**
- **value**

23.84.1 Detailed Description

Device Type Enumeration from atca_devtypes.h

23.85 cryptoauthlib.atcaenum.AtcaEnum Class Reference

Inheritance diagram for cryptoauthlib.atcaenum.AtcaEnum:



Public Member Functions

- def **__str__** (self)
- def **__eq__** (self, other)
- def **__ne__** (self, other)
- def **__int__** (self)
- def **__hash__** (self)

Data Fields

- **name**
- **value**

23.85.1 Detailed Description

Overload of standard python enum for some additional convenience features. Assumes closer alignment to C style where the value is always an integer

23.86 ATCAHAL_t Struct Reference

HAL Driver Structure.

```
#include <lib/atca_iface.h>
```

Data Fields

- ATCA_STATUS(* **halinit**)(ATCAIface iface, ATCAIfaceCfg *cfg)
- ATCA_STATUS(* **halpostinit**)(ATCAIface iface)
- ATCA_STATUS(* **halsend**)(ATCAIface iface, uint8_t word_address, uint8_t *txdata, int txlength)
- ATCA_STATUS(* **halreceive**)(ATCAIface iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxlength)
- ATCA_STATUS(* **halcontrol**)(ATCAIface iface, uint8_t option, void *param, size_t paramlen)
- ATCA_STATUS(* **halrelease**)(void *hal_data)

23.86.1 Detailed Description

HAL Driver Structure.

23.87 atcal2Cmaster Struct Reference

this is the hal_data for ATCA HAL for ASF SERCOM

```
#include <lib/hal/hal_uc3_i2c_asf.h>
```

Data Fields

- int **id**
- i2c_master_bus_config_t **conf**
- i2c_device_config_t **dev_conf**
- i2c_master_bus_handle_t **bus_handle**
- i2c_master_dev_handle_t **dev_handle**
- int **ref_ct**
- uint8_t **twi_id**
- avr32_twi_t * **twi_master_instance**
- int **bus_index**

23.87.1 Detailed Description

this is the hal_data for ATCA HAL for ASF SERCOM

23.88 ATCAIfaceCfg Struct Reference

Data Fields

- [ATCAIfaceType](#) **iface_type**
- ATCADeviceType **devtype**
-

```
union {
    struct {
        uint8_t address
        uint8_t bus
        uint32_t baud
    } atcai2c
    struct {
        uint8_t address
        uint8_t bus
    } atcaswi
    struct {
        uint8_t bus
        uint8_t select_pin
        uint32_t baud
    } atcaspi
    struct {
        ATCAKitType dev_interface
        uint8_t dev_identity
        uint8_t port
        uint32_t baud
        uint8_t wordsize
        uint8_t parity
        uint8_t stopbits
    } atcauart
    struct {
        int idx
        ATCAKitType dev_interface
        uint8_t dev_identity
        uint32_t vid
        uint32_t pid
        uint32_t packetsize
    } atcahid
    struct {
        ATCAKitType dev_interface
        uint8_t dev_identity
        uint32_t flags
    } atcakit
    struct {
        ATCA_STATUS(* halinit )(void *hal, void *cfg)
        ATCA_STATUS(* halpostinit )(void *iface)
        ATCA_STATUS(* halsend )(void *iface, uint8_t
            word_address, uint8_t *txdata,
            int txlength)
        ATCA_STATUS(* halreceive )(void *iface, uint8_t
            word_address, uint8_t *rxdata,
            uint16_t *rxlength)
        ATCA_STATUS(* halwake )(void *iface)
        ATCA_STATUS(* halidle )(void *iface)
        ATCA_STATUS(* halsleep )(void *iface)
        ATCA_STATUS(* halrelease )(void *hal_data)
```

```
    } atcacustom  
} cfg
```

- uint16_t **wake_delay**
- int **rx_retries**
- void * **cfg_data**

23.88.1 Field Documentation

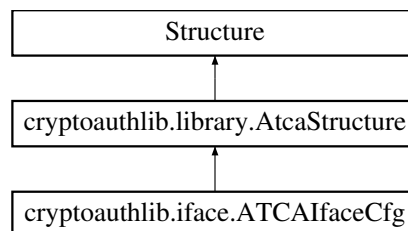
23.88.1.1 address

```
uint8_t ATCAIfaceCfg::address
```

Device address - the upper 7 bits are the I2c address bits

23.89 cryptoauthlib.iface.ATCAIfaceCfg Class Reference

Inheritance diagram for cryptoauthlib.iface.ATCAIfaceCfg:



Static Protected Attributes

- tuple **_anonymous_** = ('cfg',)
- dict **_map_**
- dict **_def_**

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.library.AtcaStructure](#)

- None **__init__** (self, *args, **kwargs)
- def **from_definition** (cls)
- def **check_rationality** (cls)
- def **get_field_definition** (cls, str name)
- Any **__getattr__** (self, str name)
- def **__iter__** (self)
- def **__str__** (self)
- def **to_c_code** (self, name=None, **kwargs)
- def **update_from_buffer** (self, buffer)

23.89.1 Detailed Description

Interface configuration structure used by `atcab_init()`

23.89.2 Field Documentation

23.89.2.1 `_def_`

dict `cryptoauthlib.iface.ATCAIfaceCfg._def_` [static], [protected]

Initial value:

```
= {
    'iface_type': (ATCAIfaceType,),
    'devtype': (ATCADeviceType,),
    'cfg': (_ATCAIfaceParams,),
    'wake_delay': (c_uint16,),
    'rx_retries': (c_int,),
    'cfg_data': (c_void_p,)
```

23.89.2.2 `_map_`

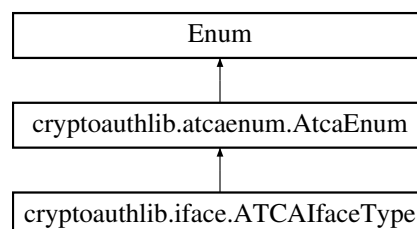
dict `cryptoauthlib.iface.ATCAIfaceCfg._map_` [static], [protected]

Initial value:

```
= {
    'cfg': ('iface_type', {
        ATCAIfaceType.ATCA_I2C_IFACE: 'atcai2c',
        ATCAIfaceType.ATCA_SWI_IFACE: 'atcaswi',
        ATCAIfaceType.ATCA_UART_IFACE: 'atcauart',
        ATCAIfaceType.ATCA_SPI_IFACE: 'atcaspi',
        ATCAIfaceType.ATCA_HID_IFACE: 'atcahid',
        ATCAIfaceType.ATCA_KIT_IFACE: 'atcakit',
        ATCAIfaceType.ATCA_CUSTOM_IFACE: 'atcacustom'
    })
}
```

23.90 `cryptoauthlib.iface.ATCAIfaceType` Class Reference

Inheritance diagram for `cryptoauthlib.iface.ATCAIfaceType`:



Static Public Attributes

- int **ATCA_I2C_IFACE** = 0
- int **ATCA_SWI_IFACE** = 1
- int **ATCA_UART_IFACE** = 2
- int **ATCA_SPI_IFACE** = 3
- int **ATCA_HID_IFACE** = 4
- int **ATCA_KIT_IFACE** = 5
- int **ATCA_CUSTOM_IFACE** = 6
- int **ATCA_I2C_GPIO_IFACE** = 7
- int **ATCA_SWI_GPIO_IFACE** = 8
- int **ATCA_SPI_GPIO_IFACE** = 9
- int **ATCA_UNKNOWN_IFACE** = 0xFE

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

- def **__str__** (self)
- def **__eq__** (self, other)
- def **__ne__** (self, other)
- def **__int__** (self)
- def **__hash__** (self)

Data Fields inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

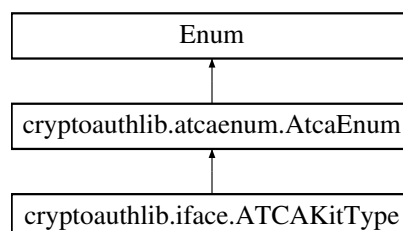
- **name**
- **value**

23.90.1 Detailed Description

Interface Type Enumerations from atca_iface.h

23.91 cryptoauthlib.iface.ATCAKitType Class Reference

Inheritance diagram for cryptoauthlib.iface.ATCAKitType:



Static Public Attributes

- int **ATCA_KIT_AUTO_IFACE** = 0
- int **ATCA_KIT_I2C_IFACE** = 1
- int **ATCA_KIT_SWI_IFACE** = 2
- int **ATCA_KIT_SPI_IFACE** = 3
- int **ATCA_KIT_UNKNOWN_IFACE** = 4

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

- def **__str__** (self)
- def **__eq__** (self, other)
- def **__ne__** (self, other)
- def **__int__** (self)
- def **__hash__** (self)

Data Fields inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

- **name**
- **value**

23.91.1 Detailed Description

Interface Type Enumerations for Kit devices

23.92 ATCAPacket Struct Reference

Data Fields

- uint8_t **reserved**
- uint8_t **txsize**
- uint8_t **opcode**
- uint8_t **param1**
- uint16_t **param2**
- uint8_t **data** [((198u)) - 6u]
- uint8_t **execTime**

23.93 cryptoauthlib.library.AtcaReference Class Reference

Public Member Functions

- def **__init__** (self, value)
- def **__eq__** (self, other)
- def **__ne__** (self, other)
- def **__lt__** (self, other)
- def **__le__** (self, other)
- def **__gt__** (self, other)
- def **__ge__** (self, other)
- def **__int__** (self)
- def **__str__** (self)

Data Fields

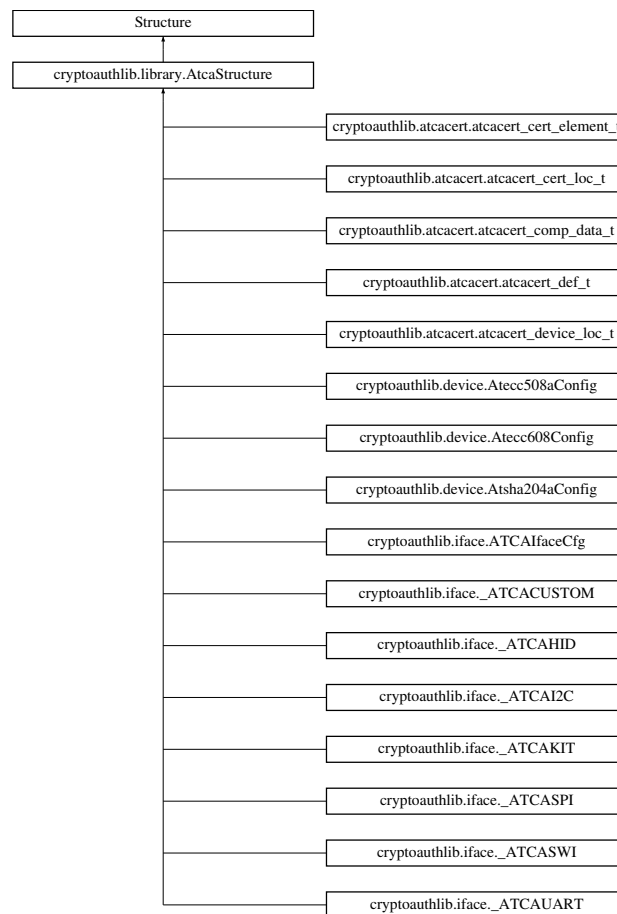
- `value`

23.93.1 Detailed Description

A simple wrapper to pass an immutable type to a function for return

23.94 cryptoauthlib.library.AtcaStructure Class Reference

Inheritance diagram for `cryptoauthlib.library.AtcaStructure`:



Public Member Functions

- `None __init__(self, *args, **kwargs)`
- `def from_definition(cls)`
- `def check_rationality(cls)`
- `def get_field_definition(cls, str name)`
- `Any __getattr__(self, str name)`
- `def __iter__(self)`
- `def __str__(self)`
- `def to_c_code(self, name=None, **kwargs)`
- `def update_from_buffer(self, buffer)`

23.94.1 Detailed Description

An extended ctypes structure to accept complex inputs

23.94.2 Member Function Documentation

23.94.2.1 `check_rationality()`

```
def cryptoauthlib.library.AtcaStructure.check_rationality (
    cls )
```

Perform a rationality check on the structure definition against the expected definition by checking structure sizes between the compiled library and the python library

23.94.2.2 `from_definition()`

```
def cryptoauthlib.library.AtcaStructure.from_definition (
    cls )
```

Trigger `_field_` creation from the values provided in `_def_` - must be run before the class is instantiated

23.95 atcaSWImaster Struct Reference

this is the `hal_data` for ATCA HAL for ASF SERCOM

```
#include <lib/hal/swi_uart_start.h>
```

Data Fields

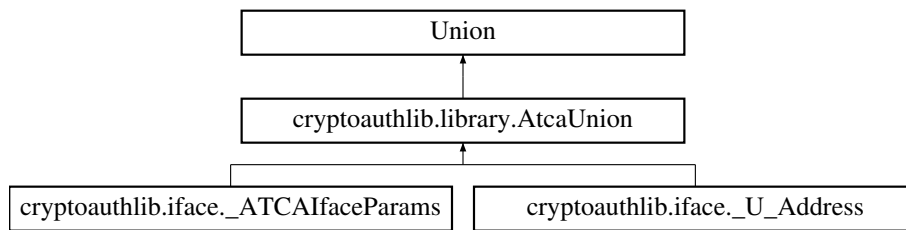
- struct `usart_module` **usart_instance**
- int **ref_ct**
- int **bus_index**
- struct `usart_sync_descriptor` **USART_SWI**
- `uint32_t` **sercom_core_freq**

23.95.1 Detailed Description

this is the `hal_data` for ATCA HAL for ASF SERCOM

23.96 cryptoauthlib.library.AtcaUnion Class Reference

Inheritance diagram for cryptoauthlib.library.AtcaUnion:



Public Member Functions

- def `__init__` (self, *args, **kwargs)
- def `from_definition` (cls)
- def `check_rationality` (cls)
- def `get_field_definition` (cls, str name)
- Any `__getattr__` (self, str name)
- def `__iter__` (self)
- def `__str__` (self)
- def `to_c_code` (self, name=None, **kwargs)
- def `update_from_buffer` (self, buffer)

Protected Attributes

- `_selected`

23.96.1 Detailed Description

An extended ctypes structure to accept complex inputs

23.96.2 Member Function Documentation

23.96.2.1 `check_rationality()`

```
def cryptoauthlib.library.AtcaUnion.check_rationality (  
    cls )
```

Perform a rationality check on the structure definition against the expected definition by checking structure sizes between the compiled library and the python library

23.96.2.2 from_definition()

```
def cryptoauthlib.library.AtcaUnion.from_definition (
    cls )
```

Trigger `_field_` creation from the values provided in `_def_` - must be run before the class is instantiated

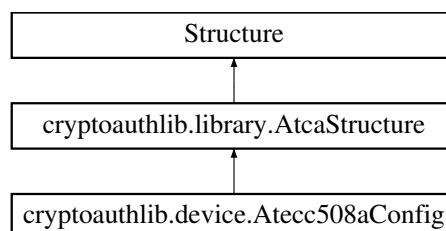
23.97 atecc508a_config_s Struct Reference

Data Fields

- uint32_t **SN03**
- uint32_t **RevNum**
- uint32_t **SN47**
- uint8_t **SN8**
- uint8_t **Reserved0**
- uint8_t **I2C_Enable**
- uint8_t **Reserved1**
- uint8_t **I2C_Address**
- uint8_t **Reserved2**
- uint8_t **OTPmode**
- uint8_t **ChipMode**
- uint16_t **SlotConfig** [16]
- uint8_t **Counter0** [8]
- uint8_t **Counter1** [8]
- uint8_t **LastKeyUse** [16]
- uint8_t **UserExtra**
- uint8_t **Selector**
- uint8_t **LockValue**
- uint8_t **LockConfig**
- uint16_t **SlotLocked**
- uint16_t **RFU**
- uint32_t **X509format**
- uint16_t **KeyConfig** [16]

23.98 cryptoauthlib.device.Atecc508aConfig Class Reference

Inheritance diagram for `cryptoauthlib.device.Atecc508aConfig`:



Static Protected Attributes

- list [_fields_](#)
- int [_pack_](#) = 1

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.library.AtcaStructure](#)

- None [__init__](#) (self, *args, **kwargs)
- def [from_definition](#) (cls)
- def [check_rationality](#) (cls)
- def [get_field_definition](#) (cls, str name)
- Any [__getattr__](#) (self, str name)
- def [__iter__](#) (self)
- def [__str__](#) (self)
- def [to_c_code](#) (self, name=None, **kwargs)
- def [update_from_buffer](#) (self, buffer)

23.98.1 Detailed Description

ATECC508A Config Zone Definition

23.98.2 Field Documentation

23.98.2.1 [_fields_](#)

```
list cryptoauthlib.device.Atecc508aConfig._fields_ [static], [protected]
```

Initial value:

```
= [('SN03', c_uint8*4),
    ('RevNum', c_uint8*4),
    ('SN48', c_uint8*5),
    ('Reserved13', c_uint8),
    ('I2C_Enable', I2cEnable),
    ('Reserved15', c_uint8),
    ('I2C_Address', c_uint8),
    ('Reserved17', c_uint8),
    ('OTPmode', c_uint8),
    ('ChipMode', ChipMode508),
    ('SlotConfig', SlotConfig*16),
    ('Counter0', c_uint8*8),
    ('Counter1', c_uint8*8),
    ('LastKeyUse', c_uint8*16),
    ('UserExtra', c_uint8),
    ('Selector', c_uint8),
    ('LockValue', c_uint8),
    ('LockConfig', c_uint8),
    ('SlotLocked', c_uint16),
    ('RFU', c_uint16),
    ('X509format', X509Format*4),
    ('KeyConfig', KeyConfig*16)]
```

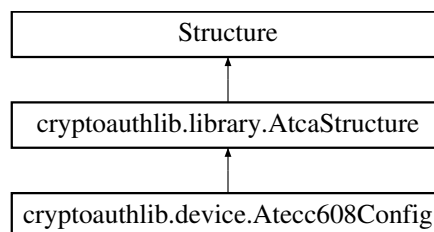
23.99 atecc608_config_s Struct Reference

Data Fields

- uint32_t **SN03**
- uint32_t **RevNum**
- uint32_t **SN47**
- uint8_t **SN8**
- uint8_t **AES_Enable**
- uint8_t **I2C_Enable**
- uint8_t **Reserved1**
- uint8_t **I2C_Address**
- uint8_t **Reserved2**
- uint8_t **CountMatch**
- uint8_t **ChipMode**
- uint16_t **SlotConfig** [16]
- uint8_t **Counter0** [8]
- uint8_t **Counter1** [8]
- uint8_t **UseLock**
- uint8_t **VolatileKeyPermission**
- uint16_t **SecureBoot**
- uint8_t **KdfIvLoc**
- uint16_t **KdfIvStr**
- uint8_t **Reserved3** [9]
- uint8_t **UserExtra**
- uint8_t **UserExtraAdd**
- uint8_t **LockValue**
- uint8_t **LockConfig**
- uint16_t **SlotLocked**
- uint16_t **ChipOptions**
- uint32_t **X509format**
- uint16_t **KeyConfig** [16]

23.100 cryptoauthlib.device.Atecc608Config Class Reference

Inheritance diagram for cryptoauthlib.device.Atecc608Config:



Static Protected Attributes

- list [_fields_](#)
- int **_pack_** = 1

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.library.AtcaStructure](#)

- None `__init__` (self, *args, **kwargs)
- def `from_definition` (cls)
- def `check_rationality` (cls)
- def `get_field_definition` (cls, str name)
- Any `__getattr__` (self, str name)
- def `__iter__` (self)
- def `__str__` (self)
- def `to_c_code` (self, name=None, **kwargs)
- def `update_from_buffer` (self, buffer)

23.100.1 Detailed Description

ATECC608 Config Zone Definition

23.100.2 Field Documentation

23.100.2.1 `_fields_`

```
list cryptoauthlib.device.Atecc608Config._fields_ [static], [protected]
```

Initial value:

```
= [ ('SN03', c_uint8*4),  
    ('RevNum', c_uint8*4),  
    ('SN48', c_uint8*5),  
    ('AES_Enable', AesEnable),  
    ('I2C_Enable', I2cEnable),  
    ('Reserved15', c_uint8),  
    ('I2C_Address', c_uint8),  
    ('Reserved17', c_uint8),  
    ('CountMatch', CountMatch),  
    ('ChipMode', ChipMode608),  
    ('SlotConfig', SlotConfig*16),  
    ('Counter0', c_uint8*8),  
    ('Counter1', c_uint8*8),  
    ('UseLock', UseLock),  
    ('VolatileKeyPermission', VolatileKeyPermission),  
    ('SecureBoot', SecureBoot),  
    ('KdfIvLoc', c_uint8),  
    ('KdfIvStr', c_uint8*2),  
    ('Reserved68', c_uint8*9),  
    ('UserExtra', c_uint8),  
    ('UserExtraAdd', c_uint8),  
    ('LockValue', c_uint8),  
    ('LockConfig', c_uint8),  
    ('SlotLocked', c_uint16),  
    ('ChipOptions', ChipOptions),  
    ('X509format', X509Format*4),  
    ('KeyConfig', KeyConfig*16)]
```

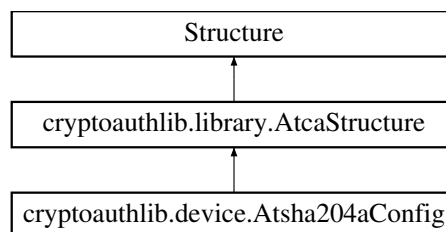
23.101 atsha204a_config_s Struct Reference

Data Fields

- uint32_t **SN03**
- uint32_t **RevNum**
- uint32_t **SN47**
- uint8_t **SN8**
- uint8_t **Reserved0**
- uint8_t **I2C_Enable**
- uint8_t **Reserved1**
- uint8_t **I2C_Address**
- uint8_t **Reserved2**
- uint8_t **OTPmode**
- uint8_t **ChipMode**
- uint16_t **SlotConfig** [16]
- uint16_t **Counter** [8]
- uint8_t **LastKeyUse** [16]
- uint8_t **UserExtra**
- uint8_t **Selector**
- uint8_t **LockValue**
- uint8_t **LockConfig**

23.102 cryptoauthlib.device.Atsha204aConfig Class Reference

Inheritance diagram for cryptoauthlib.device.Atsha204aConfig:



Static Protected Attributes

- list [_fields_](#)
- int [_pack_](#) = 1

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.library.AtcaStructure](#)

- None [__init__](#) (self, *args, **kwargs)
- def [from_definition](#) (cls)
- def [check_rationality](#) (cls)
- def [get_field_definition](#) (cls, str name)
- Any [__getattr__](#) (self, str name)
- def [__iter__](#) (self)
- def [__str__](#) (self)
- def [to_c_code](#) (self, name=None, **kwargs)
- def [update_from_buffer](#) (self, buffer)

23.102.1 Detailed Description

ATSHA204A Config Zone Definition

23.102.2 Field Documentation

23.102.2.1 `_fields_`

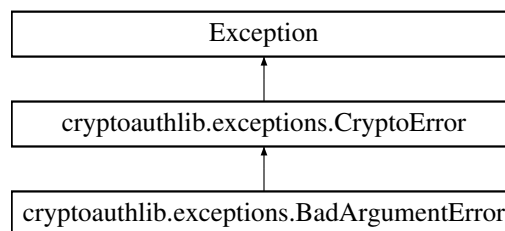
```
list cryptoauthlib.device.Atsha204aConfig._fields_  [static], [protected]
```

Initial value:

```
= [ ('SN03', c_uint8*4),  
    ('RevNum', c_uint8*4),  
    ('SN48', c_uint8*5),  
    ('Reserved13', c_uint8),  
    ('I2C_Enable', I2cEnable),  
    ('Reserved15', c_uint8),  
    ('I2C_Address', c_uint8),  
    ('CheckMacConfig', c_uint8),  
    ('OTPmode', c_uint8),  
    ('SelectorMode', c_uint8),  
    ('SlotConfig', SlotConfig*16),  
    ('Counter', Counter204*8),  
    ('LastKeyUse', c_uint8*16),  
    ('UserExtra', c_uint8),  
    ('Selector', c_uint8),  
    ('LockValue', c_uint8),  
    ('LockConfig', c_uint8)]
```

23.103 cryptoauthlib.exceptions.BadArgumentError Class Reference

Inheritance diagram for cryptoauthlib.exceptions.BadArgumentError:

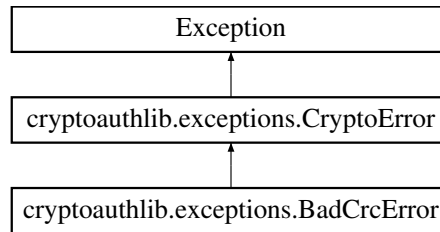


23.103.1 Detailed Description

bad argument (out of range, null pointer, etc.)

23.104 cryptoauthlib.exceptions.BadCrcError Class Reference

Inheritance diagram for cryptoauthlib.exceptions.BadCrcError:

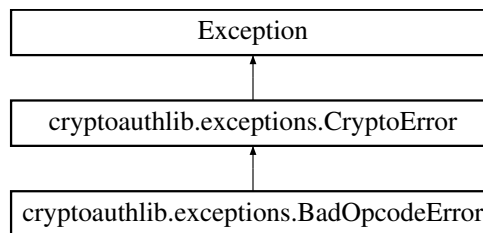


23.104.1 Detailed Description

`incorrect CRC received`

23.105 cryptoauthlib.exceptions.BadOpcodeError Class Reference

Inheritance diagram for cryptoauthlib.exceptions.BadOpcodeError:

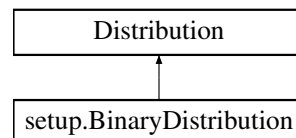


23.105.1 Detailed Description

`Opcode is not supported by the device`

23.106 setup.BinaryDistribution Class Reference

Inheritance diagram for setup.BinaryDistribution:

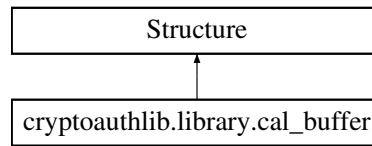


Public Member Functions

- `def has_ext_modules (self)`

23.107 cryptauthlib.library.cal_buffer Class Reference

Inheritance diagram for cryptauthlib.library.cal_buffer:



Public Member Functions

- `def __init__(self, length, data)`

Data Fields

- `len`
- `buf`

Static Protected Attributes

- `list _fields_`
- `int _pack_ = 1`

23.107.1 Field Documentation

23.107.1.1 `_fields_`

```
list cryptauthlib.library.cal_buffer._fields_ [static], [protected]
```

Initial value:

```
= [
    ('len', c_size_t),
    ('buf', POINTER(c_uint8))
]
```

23.108 cal_buffer_s Struct Reference

Data Fields

- `size_t len`
- `uint8_t * buf`

23.108.1 Field Documentation

23.108.1.1 buf

```
uint8_t* cal_buffer_s::buf
```

Pointer to the actual buffer

23.108.1.2 len

```
size_t cal_buffer_s::len
```

Length of the provided buffer

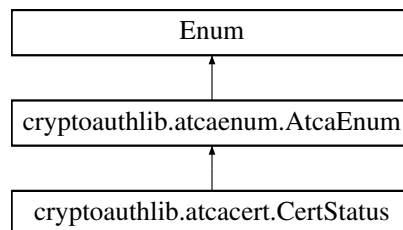
23.109 calib_packet_cache_s Struct Reference

Data Fields

- [ATCAPacket](#) packet_pool
- bool used

23.110 cryptoauthlib.atcacert.CertStatus Class Reference

Inheritance diagram for cryptoauthlib.atcacert.CertStatus:



Static Public Attributes

- int ATCACERT_E_SUCCESS = 0
- int ATCACERT_E_ERROR = 1
- int ATCACERT_E_BAD_PARAMS = 2
- int ATCACERT_E_BUFFER_TOO_SMALL = 3
- int ATCACERT_E_DECODING_ERROR = 4
- int ATCACERT_E_INVALID_DATE = 5
- int ATCACERT_E_UNIMPLEMENTED = 6
- int ATCACERT_E_UNEXPECTED_ELEM_SIZE = 7
- int ATCACERT_E_ELEM_MISSING = 8
- int ATCACERT_E_ELEM_OUT_OF_BOUNDS = 9
- int ATCACERT_E_BAD_CERT = 10
- int ATCACERT_E_WRONG_CERT_DEF = 11
- int ATCACERT_E_VERIFY_FAILED = 12

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

- `def __str__(self)`
- `def __eq__(self, other)`
- `def __ne__(self, other)`
- `def __int__(self)`
- `def __hash__(self)`

Data Fields inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

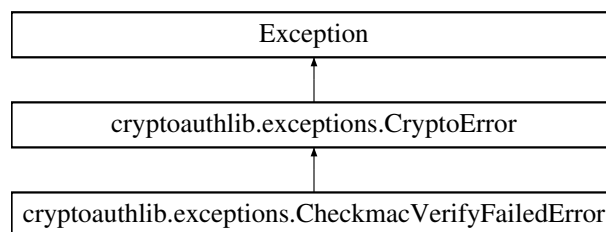
- `name`
- `value`

23.110.1 Detailed Description

Status codes returned from `atcacert` commands and their meanings. From `atcacert.h`

23.111 `cryptoauthlib.exceptions.CheckmacVerifyFailedError` Class Reference

Inheritance diagram for `cryptoauthlib.exceptions.CheckmacVerifyFailedError`:

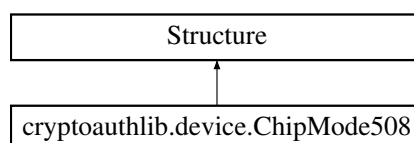


23.111.1 Detailed Description

`response status byte indicates CheckMac failure (status byte = 0x01)`

23.112 `cryptoauthlib.device.ChipMode508` Class Reference

Inheritance diagram for `cryptoauthlib.device.ChipMode508`:



Static Protected Attributes

- list [_fields_](#)
- int [_pack_](#) = 1

23.112.1 Detailed Description

ChipMode for 508 Field Definition

23.112.2 Field Documentation

23.112.2.1 [_fields_](#)

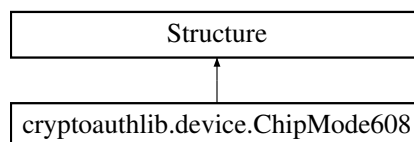
```
list cryptoauthlib.device.ChipMode508._fields_ [static], [protected]
```

Initial value:

```
= [('UserExtraAdd', c_uint8, 1),
    ('TTLenable', c_uint8, 1),
    ('WatchdogDuration', c_uint8, 1)]
```

23.113 cryptoauthlib.device.ChipMode608 Class Reference

Inheritance diagram for cryptoauthlib.device.ChipMode608:



Static Protected Attributes

- list [_fields_](#)
- int [_pack_](#) = 1

23.113.1 Detailed Description

ChipMode for 608 Field Definition

23.113.2 Field Documentation

23.113.2.1 `_fields_`

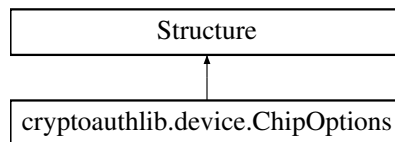
```
list cryptoauthlib.device.ChipMode608._fields_ [static], [protected]
```

Initial value:

```
= [('UserExtraAdd', c_uint8, 1),  
    ('TTLenable', c_uint8, 1),  
    ('WatchdogDuration', c_uint8, 1),  
    ('ClockDivider', c_uint8, 5)]
```

23.114 cryptoauthlib.device.ChipOptions Class Reference

Inheritance diagram for `cryptoauthlib.device.ChipOptions`:



Static Protected Attributes

- list `_fields_`
- int `_pack_` = 1

23.114.1 Detailed Description

ChipOptions Field Definition

23.114.2 Field Documentation

23.114.2.1 `_fields_`

```
list cryptoauthlib.device.ChipOptions._fields_ [static], [protected]
```

Initial value:

```
= [('PowerOnSelfTest', c_uint16, 1),  
    ('IoProtectionKeyEnable', c_uint16, 1),  
    ('KdfAesEnable', c_uint16, 1),  
    ('AutoClearFirstFail', c_uint16, 1),  
    ('Reserved', c_uint16, 4),  
    ('EcdhProtectionBits', c_uint16, 2),  
    ('KdfProtectionBits', c_uint16, 2),  
    ('IoProtectionKey', c_uint16, 4)]
```

23.115 CK_AES_CBC_ENCRYPT_DATA_PARAMS Struct Reference

Data Fields

- CK_BYTE iv [16]
- CK_BYTE_PTR pData
- CK_ULONG length

23.116 CK_AES_CCM_PARAMS Struct Reference

Data Fields

- CK_ULONG ulDataLen
- CK_BYTE_PTR pNonce
- CK_ULONG ulNonceLen
- CK_BYTE_PTR pAAD
- CK_ULONG ulAADLen
- CK_ULONG ulMACLen

23.117 CK_AES_CTR_PARAMS Struct Reference

Data Fields

- CK_ULONG ulCounterBits
- CK_BYTE cb [16]

23.118 CK_AES_GCM_PARAMS Struct Reference

Data Fields

- CK_BYTE_PTR pIv
- CK_ULONG ulIvLen
- CK_ULONG ulIvBits
- CK_BYTE_PTR pAAD
- CK_ULONG ulAADLen
- CK_ULONG ulTagBits

23.119 CK_ARIA_CBC_ENCRYPT_DATA_PARAMS Struct Reference

Data Fields

- CK_BYTE iv [16]
- CK_BYTE_PTR pData
- CK_ULONG length

23.120 CK_ATTRIBUTE Struct Reference

Data Fields

- CK_ATTRIBUTE_TYPE **type**
- CK_VOID_PTR **pValue**
- CK_ULONG **ulValueLen**

23.121 CK_C_INITIALIZE_ARGS Struct Reference

Data Fields

- CK_CREATEMUTEX **CreateMutex**
- CK_DESTROYMUTEX **DestroyMutex**
- CK_LOCKMUTEX **LockMutex**
- CK_UNLOCKMUTEX **UnlockMutex**
- CK_FLAGS **flags**
- CK_VOID_PTR **pReserved**

23.122 CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS Struct Reference

Data Fields

- CK_BYTE **iv** [16]
- CK_BYTE_PTR **pData**
- CK_ULONG **length**

23.123 CK_CAMELLIA_CTR_PARAMS Struct Reference

Data Fields

- CK_ULONG **ulCounterBits**
- CK_BYTE **cb** [16]

23.124 CK_CCM_PARAMS Struct Reference

Data Fields

- CK_ULONG **ulDataLen**
- CK_BYTE_PTR **pNonce**
- CK_ULONG **ulNonceLen**
- CK_BYTE_PTR **pAAD**
- CK_ULONG **ulAADLen**
- CK_ULONG **ulMACLen**

23.125 CK_CMS_SIG_PARAMS Struct Reference

Data Fields

- CK_OBJECT_HANDLE **certificateHandle**
- CK_MECHANISM_PTR **pSigningMechanism**
- CK_MECHANISM_PTR **pDigestMechanism**
- CK_UTF8CHAR_PTR **pContentType**
- CK_BYTE_PTR **pRequestedAttributes**
- CK_ULONG **ulRequestedAttributesLen**
- CK_BYTE_PTR **pRequiredAttributes**
- CK_ULONG **ulRequiredAttributesLen**

23.126 CK_DATE Struct Reference

Data Fields

- CK_CHAR **year** [4]
- CK_CHAR **month** [2]
- CK_CHAR **day** [2]

23.127 CK_DES_CBC_ENCRYPT_DATA_PARAMS Struct Reference

Data Fields

- CK_BYTE **iv** [8]
- CK_BYTE_PTR **pData**
- CK_ULONG **length**

23.128 CK_DSA_PARAMETER_GEN_PARAM Struct Reference

Data Fields

- CK_MECHANISM_TYPE **hash**
- CK_BYTE_PTR **pSeed**
- CK_ULONG **ulSeedLen**
- CK_ULONG **ulIndex**

23.129 CK_ECDH1_DERIVE_PARAMS Struct Reference

Data Fields

- CK_EC_KDF_TYPE **kdf**
- CK_ULONG **ulSharedDataLen**
- CK_BYTE_PTR **pSharedData**
- CK_ULONG **ulPublicDataLen**
- CK_BYTE_PTR **pPublicData**

23.130 CK_ECDH2_DERIVE_PARAMS Struct Reference

Data Fields

- CK_EC_KDF_TYPE **kdf**
- CK_ULONG **ulSharedDataLen**
- CK_BYTE_PTR **pSharedData**
- CK_ULONG **ulPublicDataLen**
- CK_BYTE_PTR **pPublicData**
- CK_ULONG **ulPrivateDataLen**
- CK_OBJECT_HANDLE **hPrivateData**
- CK_ULONG **ulPublicDataLen2**
- CK_BYTE_PTR **pPublicData2**

23.131 CK_ECDH_AES_KEY_WRAP_PARAMS Struct Reference

Data Fields

- CK_ULONG **ulAESKeyBits**
- CK_EC_KDF_TYPE **kdf**
- CK_ULONG **ulSharedDataLen**
- CK_BYTE_PTR **pSharedData**

23.132 CK_ECMQV_DERIVE_PARAMS Struct Reference

Data Fields

- CK_EC_KDF_TYPE **kdf**
- CK_ULONG **ulSharedDataLen**
- CK_BYTE_PTR **pSharedData**
- CK_ULONG **ulPublicDataLen**
- CK_BYTE_PTR **pPublicData**
- CK_ULONG **ulPrivateDataLen**
- CK_OBJECT_HANDLE **hPrivateData**
- CK_ULONG **ulPublicDataLen2**
- CK_BYTE_PTR **pPublicData2**
- CK_OBJECT_HANDLE **publicKey**

23.133 CK_FUNCTION_LIST Struct Reference

Data Fields

- [CK_VERSION](#) **version**

23.134 CK_GCM_PARAMS Struct Reference

Data Fields

- CK_BYTE_PTR **plv**
- CK_ULONG **ulIvLen**
- CK_ULONG **ulIvBits**
- CK_BYTE_PTR **pAAD**
- CK_ULONG **ulAADLen**
- CK_ULONG **ulTagBits**

23.135 CK_GOSTR3410_DERIVE_PARAMS Struct Reference

Data Fields

- CK_EC_KDF_TYPE **kdf**
- CK_BYTE_PTR **pPublicData**
- CK_ULONG **ulPublicDataLen**
- CK_BYTE_PTR **pUKM**
- CK_ULONG **ulUKMLen**

23.136 CK_GOSTR3410_KEY_WRAP_PARAMS Struct Reference

Data Fields

- CK_BYTE_PTR **pWrapOID**
- CK_ULONG **ulWrapOIDLen**
- CK_BYTE_PTR **pUKM**
- CK_ULONG **ulUKMLen**
- CK_OBJECT_HANDLE **hKey**

23.137 CK_INFO Struct Reference

Data Fields

- [CK_VERSION](#) **cryptokiVersion**
- CK_UTF8CHAR **manufacturerID** [32]
- CK_FLAGS **flags**
- CK_UTF8CHAR **libraryDescription** [32]
- [CK_VERSION](#) **libraryVersion**

23.138 CK_KEA_DERIVE_PARAMS Struct Reference

Data Fields

- CK_BBOOL **isSender**
- CK_ULONG **ulRandomLen**
- CK_BYTE_PTR **pRandomA**
- CK_BYTE_PTR **pRandomB**
- CK_ULONG **ulPublicDataLen**
- CK_BYTE_PTR **pPublicData**

23.139 CK_KEY_DERIVATION_STRING_DATA Struct Reference

Data Fields

- CK_BYTE_PTR **pData**
- CK_ULONG **ulLen**

23.140 CK_KEY_WRAP_SET_OAEP_PARAMS Struct Reference

Data Fields

- CK_BYTE **bBC**
- CK_BYTE_PTR **pX**
- CK_ULONG **ulXLen**

23.141 CK_KIP_PARAMS Struct Reference

Data Fields

- CK_MECHANISM_PTR **pMechanism**
- CK_OBJECT_HANDLE **hKey**
- CK_BYTE_PTR **pSeed**
- CK_ULONG **ulSeedLen**

23.142 CK_MECHANISM Struct Reference

Data Fields

- CK_MECHANISM_TYPE **mechanism**
- CK_VOID_PTR **pParameter**
- CK_ULONG **ulParameterLen**

23.143 CK_MECHANISM_INFO Struct Reference

Data Fields

- CK_ULONG **ulMinKeySize**
- CK_ULONG **ulMaxKeySize**
- CK_FLAGS **flags**

23.144 CK_OTP_PARAM Struct Reference

Data Fields

- CK_OTP_PARAM_TYPE **type**
- CK_VOID_PTR **pValue**
- CK_ULONG **ulValueLen**

23.145 CK_OTP_PARAMS Struct Reference

Data Fields

- CK_OTP_PARAM_PTR **pParams**
- CK_ULONG **ulCount**

23.146 CK_OTP_SIGNATURE_INFO Struct Reference

Data Fields

- CK_OTP_PARAM_PTR **pParams**
- CK_ULONG **ulCount**

23.147 CK_PBE_PARAMS Struct Reference

Data Fields

- CK_BYTE_PTR **pInitVector**
- CK_UTF8CHAR_PTR **pPassword**
- CK_ULONG **ulPasswordLen**
- CK_BYTE_PTR **pSalt**
- CK_ULONG **ulSaltLen**
- CK_ULONG **ulIteration**

23.148 CK_PKCS5_PBKD2_PARAMS Struct Reference

Data Fields

- CK_PKCS5_PBKDF2_SALT_SOURCE_TYPE **saltSource**
- CK_VOID_PTR **pSaltSourceData**
- CK_ULONG **ulSaltSourceDataLen**
- CK_ULONG **iterations**
- CK_PKCS5_PBKD2_PSEUDO_RANDOM_FUNCTION_TYPE **prf**
- CK_VOID_PTR **pPrfData**
- CK_ULONG **ulPrfDataLen**
- CK_UTF8CHAR_PTR **pPassword**
- CK_ULONG_PTR **ulPasswordLen**

23.149 CK_PKCS5_PBKD2_PARAMS2 Struct Reference

Data Fields

- CK_PKCS5_PBKDF2_SALT_SOURCE_TYPE **saltSource**
- CK_VOID_PTR **pSaltSourceData**
- CK_ULONG **ulSaltSourceDataLen**
- CK_ULONG **iterations**
- CK_PKCS5_PBKD2_PSEUDO_RANDOM_FUNCTION_TYPE **prf**
- CK_VOID_PTR **pPrfData**
- CK_ULONG **ulPrfDataLen**
- CK_UTF8CHAR_PTR **pPassword**
- CK_ULONG **ulPasswordLen**

23.150 CK_RC2_CBC_PARAMS Struct Reference

Data Fields

- CK_ULONG **ulEffectiveBits**
- CK_BYTE **iv** [8]

23.151 CK_RC2_MAC_GENERAL_PARAMS Struct Reference

Data Fields

- CK_ULONG **ulEffectiveBits**
- CK_ULONG **ulMacLength**

23.152 CK_RC5_CBC_PARAMS Struct Reference

Data Fields

- CK_ULONG **ulWordsize**
- CK_ULONG **ulRounds**
- CK_BYTE_PTR **plv**
- CK_ULONG **ullvLen**

23.153 CK_RC5_MAC_GENERAL_PARAMS Struct Reference

Data Fields

- CK_ULONG **ulWordsize**
- CK_ULONG **ulRounds**
- CK_ULONG **ulMacLength**

23.154 CK_RC5_PARAMS Struct Reference

Data Fields

- CK_ULONG **ulWordsize**
- CK_ULONG **ulRounds**

23.155 CK_RSA_AES_KEY_WRAP_PARAMS Struct Reference

Data Fields

- CK_ULONG **ulAESKeyBits**
- CK_RSA_PKCS_OAEP_PARAMS_PTR **pOAEPParams**

23.156 CK_RSA_PKCS_OAEP_PARAMS Struct Reference

Data Fields

- CK_MECHANISM_TYPE **hashAlg**
- CK_RSA_PKCS_MGF_TYPE **mgf**
- CK_RSA_PKCS_OAEP_SOURCE_TYPE **source**
- CK_VOID_PTR **pSourceData**
- CK_ULONG **ulSourceDataLen**

23.157 CK_RSA_PKCS_PSS_PARAMS Struct Reference

Data Fields

- CK_MECHANISM_TYPE **hashAlg**
- CK_RSA_PKCS_MGF_TYPE **mgf**
- CK_ULONG **sLen**

23.158 CK_SEED_CBC_ENCRYPT_DATA_PARAMS Struct Reference

Data Fields

- CK_BYTE **iv** [16]
- CK_BYTE_PTR **pData**
- CK_ULONG **length**

23.159 CK_SESSION_INFO Struct Reference

Data Fields

- CK_SLOT_ID **slotID**
- CK_STATE **state**
- CK_FLAGS **flags**
- CK_ULONG **ulDeviceError**

23.160 CK_SKIPJACK_PRIVATE_WRAP_PARAMS Struct Reference

Data Fields

- CK_ULONG **ulPasswordLen**
- CK_BYTE_PTR **pPassword**
- CK_ULONG **ulPublicDataLen**
- CK_BYTE_PTR **pPublicData**
- CK_ULONG **ulPAndGLen**
- CK_ULONG **ulQLen**
- CK_ULONG **ulRandomLen**
- CK_BYTE_PTR **pRandomA**
- CK_BYTE_PTR **pPrimeP**
- CK_BYTE_PTR **pBaseG**
- CK_BYTE_PTR **pSubprimeQ**

23.161 CK_SKIPJACK_RELAYX_PARAMS Struct Reference

Data Fields

- CK_ULONG **ulOldWrappedXLen**
- CK_BYTE_PTR **pOldWrappedX**
- CK_ULONG **ulOldPasswordLen**
- CK_BYTE_PTR **pOldPassword**
- CK_ULONG **ulOldPublicDataLen**
- CK_BYTE_PTR **pOldPublicData**
- CK_ULONG **ulOldRandomLen**
- CK_BYTE_PTR **pOldRandomA**
- CK_ULONG **ulNewPasswordLen**
- CK_BYTE_PTR **pNewPassword**
- CK_ULONG **ulNewPublicDataLen**
- CK_BYTE_PTR **pNewPublicData**
- CK_ULONG **ulNewRandomLen**
- CK_BYTE_PTR **pNewRandomA**

23.162 CK_SLOT_INFO Struct Reference

Data Fields

- CK_UTF8CHAR **slotDescription** [64]
- CK_UTF8CHAR **manufacturerID** [32]
- CK_FLAGS **flags**
- [CK_VERSION](#) **hardwareVersion**
- [CK_VERSION](#) **firmwareVersion**

23.163 CK_SSL3_KEY_MAT_OUT Struct Reference

Data Fields

- CK_OBJECT_HANDLE **hClientMacSecret**
- CK_OBJECT_HANDLE **hServerMacSecret**
- CK_OBJECT_HANDLE **hClientKey**
- CK_OBJECT_HANDLE **hServerKey**
- CK_BYTE_PTR **pIVClient**
- CK_BYTE_PTR **pIVServer**

23.164 CK_SSL3_KEY_MAT_PARAMS Struct Reference

Data Fields

- CK_ULONG **ulMacSizeInBits**
- CK_ULONG **ulKeySizeInBits**
- CK_ULONG **ulIVSizeInBits**
- CK_BBOOL **blsExport**
- [CK_SSL3_RANDOM_DATA](#) **RandomInfo**
- CK_SSL3_KEY_MAT_OUT_PTR **pReturnedKeyMaterial**

23.165 CK_SSL3_MASTER_KEY_DERIVE_PARAMS Struct Reference

Data Fields

- [CK_SSL3_RANDOM_DATA](#) RandomInfo
- CK_VERSION_PTR pVersion

23.166 CK_SSL3_RANDOM_DATA Struct Reference

Data Fields

- CK_BYTE_PTR pClientRandom
- CK_ULONG ulClientRandomLen
- CK_BYTE_PTR pServerRandom
- CK_ULONG ulServerRandomLen

23.167 CK_TLS12_KEY_MAT_PARAMS Struct Reference

Data Fields

- CK_ULONG ulMacSizeInBits
- CK_ULONG ulKeySizeInBits
- CK_ULONG ulIVSizeInBits
- CK_BBOOL bIsExport
- [CK_SSL3_RANDOM_DATA](#) RandomInfo
- CK_SSL3_KEY_MAT_OUT_PTR pReturnedKeyMaterial
- CK_MECHANISM_TYPE prfHashMechanism

23.168 CK_TLS12_MASTER_KEY_DERIVE_PARAMS Struct Reference

Data Fields

- [CK_SSL3_RANDOM_DATA](#) RandomInfo
- CK_VERSION_PTR pVersion
- CK_MECHANISM_TYPE prfHashMechanism

23.169 CK_TLS_KDF_PARAMS Struct Reference

Data Fields

- CK_MECHANISM_TYPE prfMechanism
- CK_BYTE_PTR pLabel
- CK_ULONG ulLabelLength
- [CK_SSL3_RANDOM_DATA](#) RandomInfo
- CK_BYTE_PTR pContextData
- CK_ULONG ulContextDataLength

23.170 CK_TLS_MAC_PARAMS Struct Reference

Data Fields

- CK_MECHANISM_TYPE **prfHashMechanism**
- CK_ULONG **ulMacLength**
- CK_ULONG **ulServerOrClient**

23.171 CK_TLS_PRF_PARAMS Struct Reference

Data Fields

- CK_BYTE_PTR **pSeed**
- CK_ULONG **ulSeedLen**
- CK_BYTE_PTR **pLabel**
- CK_ULONG **ulLabelLen**
- CK_BYTE_PTR **pOutput**
- CK_ULONG_PTR **pulOutputLen**

23.172 CK_TOKEN_INFO Struct Reference

Data Fields

- CK_UTF8CHAR **label** [32]
- CK_UTF8CHAR **manufacturerID** [32]
- CK_UTF8CHAR **model** [16]
- CK_CHAR **serialNumber** [16]
- CK_FLAGS **flags**
- CK_ULONG **ulMaxSessionCount**
- CK_ULONG **ulSessionCount**
- CK_ULONG **ulMaxRwSessionCount**
- CK_ULONG **ulRwSessionCount**
- CK_ULONG **ulMaxPinLen**
- CK_ULONG **ulMinPinLen**
- CK_ULONG **ulTotalPublicMemory**
- CK_ULONG **ulFreePublicMemory**
- CK_ULONG **ulTotalPrivateMemory**
- CK_ULONG **ulFreePrivateMemory**
- [CK_VERSION](#) **hardwareVersion**
- [CK_VERSION](#) **firmwareVersion**
- CK_CHAR **utcTime** [16]

23.173 CK_VERSION Struct Reference

Data Fields

- CK_BYTE **major**
- CK_BYTE **minor**

23.174 CK_WTLS_KEY_MAT_OUT Struct Reference

Data Fields

- CK_OBJECT_HANDLE **hMacSecret**
- CK_OBJECT_HANDLE **hKey**
- CK_BYTE_PTR **pIV**

23.175 CK_WTLS_KEY_MAT_PARAMS Struct Reference

Data Fields

- CK_MECHANISM_TYPE **DigestMechanism**
- CK_ULONG **ulMacSizeInBits**
- CK_ULONG **ulKeySizeInBits**
- CK_ULONG **ulIVSizeInBits**
- CK_ULONG **ulSequenceNumber**
- CK_BBOOL **blsExport**
- [CK_WTLS_RANDOM_DATA](#) **RandomInfo**
- CK_WTLS_KEY_MAT_OUT_PTR **pReturnedKeyMaterial**

23.176 CK_WTLS_MASTER_KEY_DERIVE_PARAMS Struct Reference

Data Fields

- CK_MECHANISM_TYPE **DigestMechanism**
- [CK_WTLS_RANDOM_DATA](#) **RandomInfo**
- CK_BYTE_PTR **pVersion**

23.177 CK_WTLS_PRF_PARAMS Struct Reference

Data Fields

- CK_MECHANISM_TYPE **DigestMechanism**
- CK_BYTE_PTR **pSeed**
- CK_ULONG **ulSeedLen**
- CK_BYTE_PTR **pLabel**
- CK_ULONG **ulLabelLen**
- CK_BYTE_PTR **pOutput**
- CK_ULONG_PTR **pulOutputLen**

23.178 CK_WTLS_RANDOM_DATA Struct Reference

Data Fields

- CK_BYTE_PTR **pClientRandom**
- CK_ULONG **ulClientRandomLen**
- CK_BYTE_PTR **pServerRandom**
- CK_ULONG **ulServerRandomLen**

23.179 CK_X9_42_DH1_DERIVE_PARAMS Struct Reference

Data Fields

- CK_X9_42_DH_KDF_TYPE **kdf**
- CK_ULONG **ulOtherInfoLen**
- CK_BYTE_PTR **pOtherInfo**
- CK_ULONG **ulPublicDataLen**
- CK_BYTE_PTR **pPublicData**

23.180 CK_X9_42_DH2_DERIVE_PARAMS Struct Reference

Data Fields

- CK_X9_42_DH_KDF_TYPE **kdf**
- CK_ULONG **ulOtherInfoLen**
- CK_BYTE_PTR **pOtherInfo**
- CK_ULONG **ulPublicDataLen**
- CK_BYTE_PTR **pPublicData**
- CK_ULONG **ulPrivateDataLen**
- CK_OBJECT_HANDLE **hPrivateData**
- CK_ULONG **ulPublicDataLen2**
- CK_BYTE_PTR **pPublicData2**

23.181 CK_X9_42_MQV_DERIVE_PARAMS Struct Reference

Data Fields

- CK_X9_42_DH_KDF_TYPE **kdf**
- CK_ULONG **ulOtherInfoLen**
- CK_BYTE_PTR **pOtherInfo**
- CK_ULONG **ulPublicDataLen**
- CK_BYTE_PTR **pPublicData**
- CK_ULONG **ulPrivateDataLen**
- CK_OBJECT_HANDLE **hPrivateData**
- CK_ULONG **ulPublicDataLen2**
- CK_BYTE_PTR **pPublicData2**
- CK_OBJECT_HANDLE **publicKey**

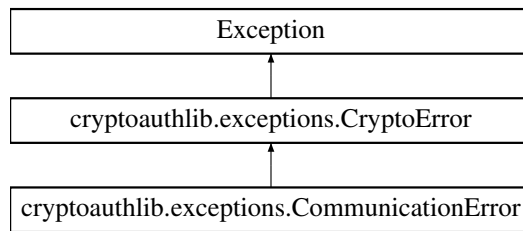
23.182 CL_HashContext Struct Reference

Data Fields

- uint32_t **h** [20/4]
- uint32_t **buf** [64/4]
- uint32_t **byteCount**
- uint32_t **byteCountHi**

23.183 cryptolib.exceptions.CommunicationError Class Reference

Inheritance diagram for cryptolib.exceptions.CommunicationError:

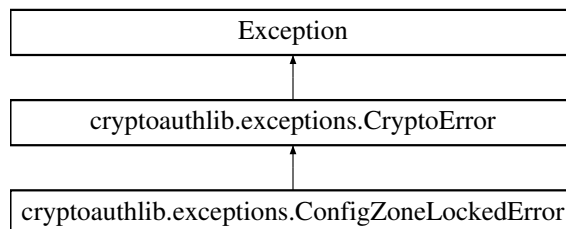


23.183.1 Detailed Description

Communication with device failed. Same as in hardware dependent modules.

23.184 cryptolib.exceptions.ConfigZoneLockedError Class Reference

Inheritance diagram for cryptolib.exceptions.ConfigZoneLockedError:

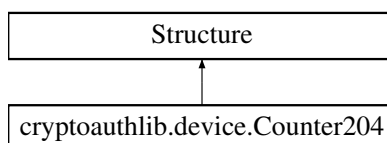


23.184.1 Detailed Description

Config Zone Locked

23.185 cryptolib.device.Counter204 Class Reference

Inheritance diagram for cryptolib.device.Counter204:



Static Protected Attributes

- list [_fields_](#)
- int [_pack_](#) = 1

23.185.1 Detailed Description

Counter Definition for SHA204

23.185.2 Field Documentation

23.185.2.1 [_fields_](#)

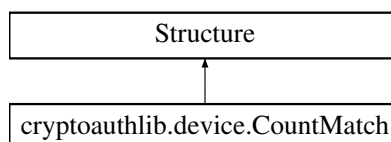
```
list cryptoauthlib.device.Counter204._fields_ [static], [protected]
```

Initial value:

```
= [ ('UseFlag', c_uint8),  
    ('UpdateCount', c_uint8)]
```

23.186 cryptoauthlib.device.CountMatch Class Reference

Inheritance diagram for cryptoauthlib.device.CountMatch:



Static Protected Attributes

- list [_fields_](#)
- int [_pack_](#) = 1

23.186.1 Detailed Description

CountMatch (608) Field Definition

23.186.2 Field Documentation

23.186.2.1 `_fields_`

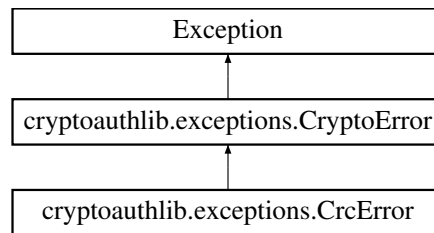
```
list cryptolib.device.CountMatch._fields_ [static], [protected]
```

Initial value:

```
= [('Enable', c_uint8, 1),  
    ('Reserved', c_uint8, 3),  
    ('CountMatchKey', c_uint8, 4)]
```

23.187 cryptolib.exceptions.CrcError Class Reference

Inheritance diagram for cryptolib.exceptions.CrcError:

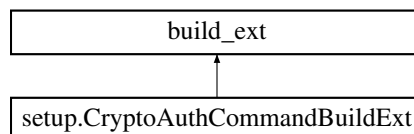


23.187.1 Detailed Description

response status byte indicates CRC error (status byte = 0xFF)

23.188 setup.CryptoAuthCommandBuildExt Class Reference

Inheritance diagram for setup.CryptoAuthCommandBuildExt:

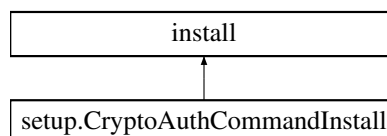


Public Member Functions

- def **build_extension** (self, ext)

23.189 setup.CryptoAuthCommandInstall Class Reference

Inheritance diagram for setup.CryptoAuthCommandInstall:

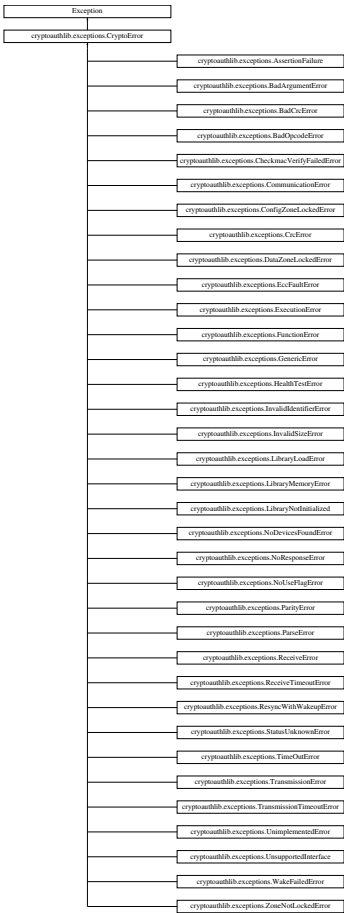


Public Member Functions

- def run (self)

23.190 cryptoauthlib.exceptions.CryptoError Class Reference

Inheritance diagram for cryptoauthlib.exceptions.CryptoError:

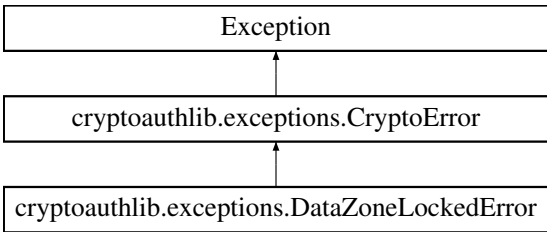


23.190.1 Detailed Description

Standard CryptoAuthLib Exceptions

23.191 cryptoauthlib.exceptions.DataZoneLockedError Class Reference

Inheritance diagram for cryptoauthlib.exceptions.DataZoneLockedError:



23.191.1 Detailed Description

Configuration Enabled

23.192 device_execution_time_t Struct Reference

Structure to hold the device execution time and the opcode for the corresponding command.

```
#include <lib/calib/calib_execution.h>
```

Data Fields

- uint8_t **opcode**
- uint16_t **execution_time_msec**

23.192.1 Detailed Description

Structure to hold the device execution time and the opcode for the corresponding command.

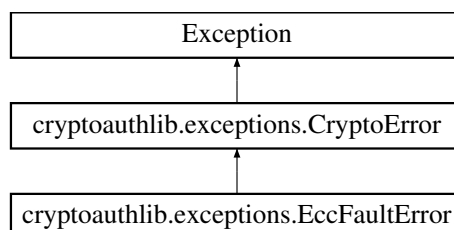
23.193 devtype_names_t Struct Reference

Data Fields

- ATCADeviceType **devtype**
- const char * **name**

23.194 cryptoauthlib.exceptions.EccFaultError Class Reference

Inheritance diagram for cryptoauthlib.exceptions.EccFaultError:

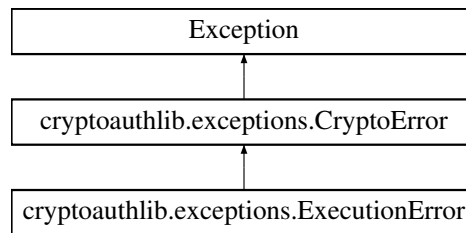


23.194.1 Detailed Description

response status byte is ECC fault (status byte = 0x05)

23.195 cryptoauthlib.exceptions.ExecutionError Class Reference

Inheritance diagram for cryptoauthlib.exceptions.ExecutionError:

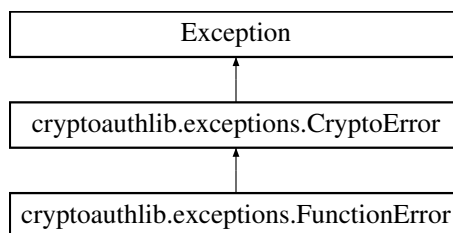


23.195.1 Detailed Description

chip was in a state where it could not execute the command, response status byte indicates command execution error (status byte = 0x0F)

23.196 cryptoauthlib.exceptions.FunctionError Class Reference

Inheritance diagram for cryptoauthlib.exceptions.FunctionError:

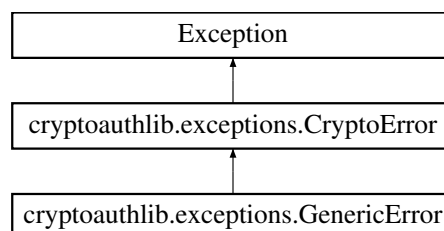


23.196.1 Detailed Description

Function could not execute due to incorrect condition / state.

23.197 cryptoauthlib.exceptions.GenericError Class Reference

Inheritance diagram for cryptoauthlib.exceptions.GenericError:

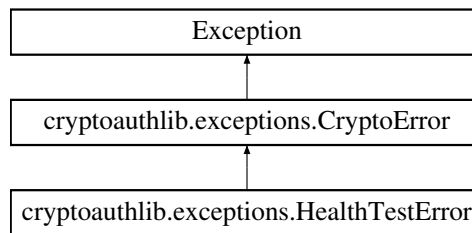


23.197.1 Detailed Description

unspecified error

23.198 cryptauthlib.exceptions.HealthTestError Class Reference

Inheritance diagram for cryptauthlib.exceptions.HealthTestError:

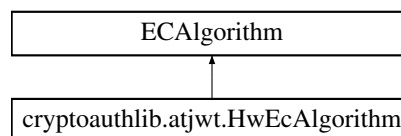


23.198.1 Detailed Description

Random number generator health test error

23.199 cryptauthlib.atjwt.HwEcAlgorithm Class Reference

Inheritance diagram for cryptauthlib.atjwt.HwEcAlgorithm:



Public Member Functions

- def `__init__` (self, hash_alg, slot, iface_cfg)
- def `sign` (self, msg, _)

Protected Attributes

- `_cfg`
- `_slot`

23.199.1 Detailed Description

Extended Algorithm with hardware based elliptic curve support

23.199.2 Member Function Documentation

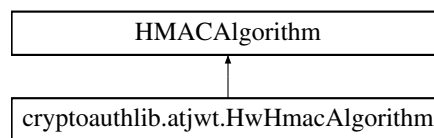
23.199.2.1 sign()

```
def cryptoauthlib.atjwt.HwEcAlgorithm.sign (
    self,
    msg,
    _ )
```

Return a signature of the JWT with hardware ECDSA

23.200 cryptoauthlib.atjwt.HwHmacAlgorithm Class Reference

Inheritance diagram for cryptoauthlib.atjwt.HwHmacAlgorithm:



Public Member Functions

- def `__init__` (self, hash_alg, slot, iface_cfg)
- def `sign` (self, msg, _)
- def `verify` (self, msg, key, sig)

Protected Attributes

- `_cfg`
- `_slot`

23.200.1 Detailed Description

Extended Algorithm with hardware based HMAC support

23.200.2 Member Function Documentation

23.200.2.1 sign()

```
def cryptoauthlib.atjwt.HwHmacAlgorithm.sign (
    self,
    msg,
    _ )
```

Return a signature of the JWT with hardware SHA256 HMAC and stored key

23.200.2.2 verify()

```
def cryptoauthlib.atjwt.HwHmacAlgorithm.verify (
    self,
    msg,
    key,
    sig )
```

Verify a signature using the software HMAC module

23.201 i2c_sam0_instance Struct Reference

Data Fields

- struct i2c_master_module * **i2c_instance**
- sam0_change_baudrate **change_baudrate**

23.202 i2c_sam_instance Struct Reference

Data Fields

- Twi * **i2c_instance**
- sam_change_baudrate **change_baudrate**

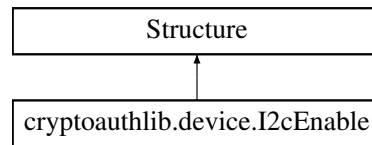
23.203 i2c_start_instance Struct Reference

Data Fields

- struct i2c_m_sync_desc * **i2c_descriptor**
- start_change_baudrate **change_baudrate**

23.204 cryptoauthlib.device.I2cEnable Class Reference

Inheritance diagram for cryptoauthlib.device.I2cEnable:



Static Protected Attributes

- list [_fields_](#)
- int [_pack_](#) = 1

23.204.1 Detailed Description

I2C Enable Field Definition

23.204.2 Field Documentation

23.204.2.1 [_fields_](#)

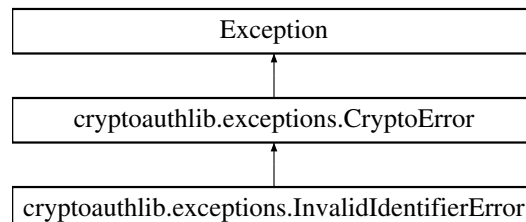
```
list cryptoauthlib.device.I2cEnable._fields_ [static], [protected]
```

Initial value:

```
= [('Enable', c_uint8, 1),  
   ('Reserved', c_uint8, 6)]
```

23.205 cryptoauthlib.exceptions.InvalidIdentifierError Class Reference

Inheritance diagram for cryptoauthlib.exceptions.InvalidIdentifierError:

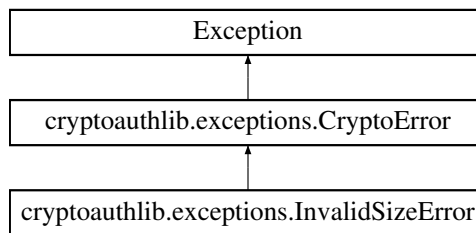


23.205.1 Detailed Description

invalid device id, id not set

23.206 cryptauthlib.exceptions.InvalidSizeError Class Reference

Inheritance diagram for cryptauthlib.exceptions.InvalidSizeError:

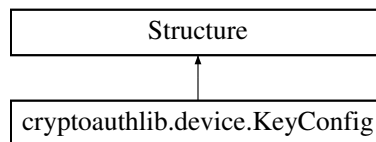


23.206.1 Detailed Description

Count value is out of range or greater than buffer size.

23.207 cryptauthlib.device.KeyConfig Class Reference

Inheritance diagram for cryptauthlib.device.KeyConfig:



Static Protected Attributes

- list [_fields_](#)
- int [_pack_](#) = 1

23.207.1 Detailed Description

KeyConfig Field Definition

23.207.2 Field Documentation

23.207.2.1 `_fields_`

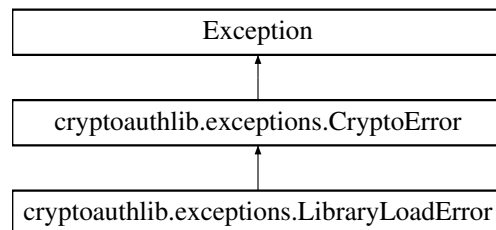
```
list cryptoauthlib.device.KeyConfig._fields_  [static], [protected]
```

Initial value:

```
= [ ('Private', c_uint16, 1),
    ('PubInfo', c_uint16, 1),
    ('KeyType', c_uint16, 3),
    ('Lockable', c_uint16, 1),
    ('ReqRandom', c_uint16, 1),
    ('ReqAuth', c_uint16, 1),
    ('AuthKey', c_uint16, 4),
    ('PersistentDisable', c_uint16, 1),
    ('RFU', c_uint16, 1),
    ('X509id', c_uint16, 2)]
```

23.208 `cryptoauthlib.exceptions.LibraryLoadError` Class Reference

Inheritance diagram for `cryptoauthlib.exceptions.LibraryLoadError`:

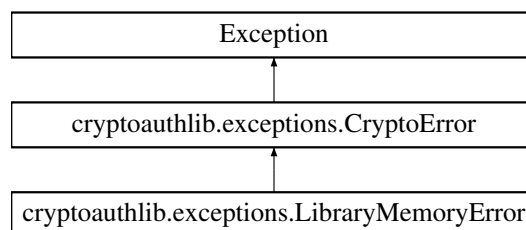


23.208.1 Detailed Description

```
CryptpAuthLib failed to Load
```

23.209 `cryptoauthlib.exceptions.LibraryMemoryError` Class Reference

Inheritance diagram for `cryptoauthlib.exceptions.LibraryMemoryError`:

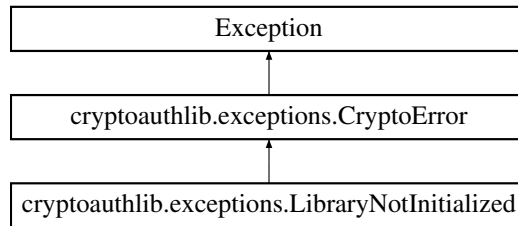


23.209.1 Detailed Description

```
CryptoAuthLib was unable to allocate memory
```

23.210 cryptauthlib.exceptions.LibraryNotInitialized Class Reference

Inheritance diagram for cryptauthlib.exceptions.LibraryNotInitialized:



23.210.1 Detailed Description

Indication that library or context was not initialized prior to an API call

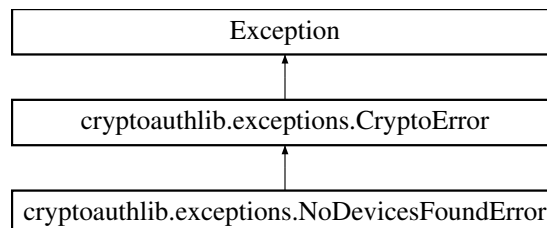
23.211 memory_parameters Struct Reference

Data Fields

- uint32_t **start_address**
- uint32_t **memory_size**
- uint32_t **version_info**
- uint8_t **reserved** [52]
- uint8_t **signature** [[ATCA_SIG_SIZE](#)]

23.212 cryptauthlib.exceptions.NoDevicesFoundError Class Reference

Inheritance diagram for cryptauthlib.exceptions.NoDevicesFoundError:

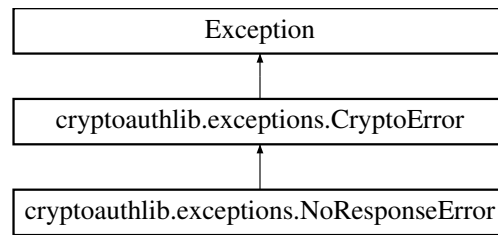


23.212.1 Detailed Description

For protocols that support device discovery (kit protocol), no devices were found

23.213 cryptoauthlib.exceptions.NoResponseError Class Reference

Inheritance diagram for cryptoauthlib.exceptions.NoResponseError:

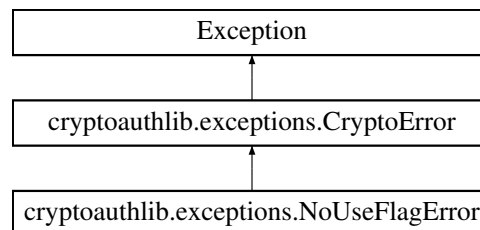


23.213.1 Detailed Description

error while the Command layer is polling for a command response.

23.214 cryptoauthlib.exceptions.NoUseFlagError Class Reference

Inheritance diagram for cryptoauthlib.exceptions.NoUseFlagError:

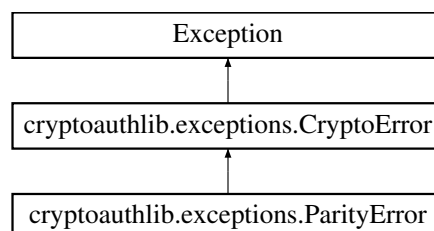


23.214.1 Detailed Description

Indication that no dk pk flag is available to perform

23.215 cryptoauthlib.exceptions.ParityError Class Reference

Inheritance diagram for cryptoauthlib.exceptions.ParityError:

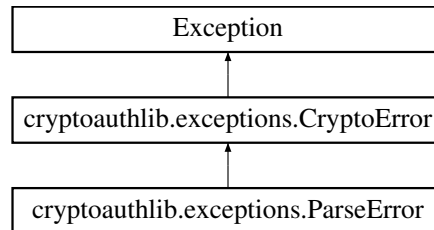


23.215.1 Detailed Description

`for protocols needing parity`

23.216 cryptoauthlib.exceptions.ParseError Class Reference

Inheritance diagram for `cryptoauthlib.exceptions.ParseError`:



23.216.1 Detailed Description

`response status byte indicates parsing error (status byte = 0x03)`

23.217 pkcs11_mech_table_e Struct Reference

Data Fields

- `CK_MECHANISM_TYPE` **type**
- `CK_MECHANISM_INFO` **info**

23.218 pkcs11_attrib_model_s Struct Reference

Data Fields

- `const CK_ATTRIBUTE_TYPE` **type**
- `const attrib_f` **func**

23.219 pkcs11_conf_filedata_s Struct Reference

Data Fields

- `bool` **initialized**
- `char` **filename** [MAX_CONF_FILE_NAME_SIZE]

23.220 `pkcs11_dev_ctx` Struct Reference

```
#include <lib/pkcs11/pkcs11_init.h>
```

Data Fields

- `CK_SESSION_HANDLE session`

23.220.1 Detailed Description

Context Tracking Info

23.221 `pkcs11_dev_res` Struct Reference

```
#include <lib/pkcs11/pkcs11_init.h>
```

Data Fields

- [pkcs11_dev_ctx contexts](#) [(5u)]

23.221.1 Detailed Description

Reservable Device Resources

23.222 `pkcs11_dev_state` Struct Reference

```
#include <lib/pkcs11/pkcs11_init.h>
```

Data Fields

- [hal_mutex_t dev_lock](#)
- [pkcs11_dev_res resources](#) [PKCS11_MAX_SLOTS_ALLOWED]

23.222.1 Detailed Description

Device state tracker structure

23.222.2 Field Documentation

23.223 pkcs11_ecc_key_info_s Struct Reference

23.222.2.1 dev_lock

`hal_mutex_t` `pkcs11_dev_state::dev_lock`

Lock to protect concurrent access to the device

23.222.2.2 resources

`pkcs11_dev_res` `pkcs11_dev_state::resources[PKCS11_MAX_SLOTS_ALLOWED]`

Track the usage of device resources

23.223 pkcs11_ecc_key_info_s Struct Reference

Data Fields

- CK_BYTE `ec_key_type`
- CK_BYTE `oid_size`
- CK_BYTE_PTR `curve_oid`
- CK_BYTE_PTR `ec_asn1_header`
- CK_BYTE_PTR `ec_x962_asn1_header`
- uint16_t `asn1_header_sz`
- CK_ULONG `pubkey_sz`
- CK_ULONG `min_msg_sz`
- CK_ULONG `sig_sz`

23.224 pkcs11_key_info_s Struct Reference

Data Fields

- const `pkcs11_ecc_key_info_t` * `ecc_key_info`
- const `pkcs11_rsa_key_info_t` * `rsa_key_info`

23.225 pkcs11_lib_ctx_s Struct Reference

```
#include <lib/pkcs11/pkcs11_init.h>
```

Data Fields

- CK_BBOOL `initialized`
- CK_C_INITIALIZE_ARGS `init_args`
- CK_VOID_PTR `lib_lock`
- `pkcs11_dev_state` * `dev_state`
- CK_BBOOL `dev_lock_enabled`
- CK_VOID_PTR `slots`
- CK_ULONG `slot_cnt`
- CK_CHAR `config_path` [200]

23.225.1 Detailed Description

Library Context

23.225.2 Field Documentation

23.225.2.1 config_path

```
CK_CHAR pkcs11_lib_ctx_s::config_path[200]
```

Filesystem path where the base config is located

23.225.2.2 dev_lock_enabled

```
CK_BBOOL pkcs11_lib_ctx_s::dev_lock_enabled
```

Flag to indicate if a device lock is enabled and configured

23.225.2.3 dev_state

```
pkcs11_dev_state* pkcs11_lib_ctx_s::dev_state
```

Device State state and Lock (if configured)

23.225.2.4 init_args

```
CK_C_INITIALIZE_ARGS pkcs11_lib_ctx_s::init_args
```

Arguments provided by the app for C_Initialize

23.225.2.5 initialized

```
CK_BBOOL pkcs11_lib_ctx_s::initialized
```

Indicates that the library has been initialized

23.225.2.6 lib_lock

```
CK_VOID_PTR pkcs11_lib_ctx_s::lib_lock
```

Application Lock for concurrent access to the library if the application will be using threads

23.225.2.7 slot_cnt

CK_ULONG pkcs11_lib_ctx_s::slot_cnt

Number of configured slots

23.225.2.8 slots

CK_VOID_PTR pkcs11_lib_ctx_s::slots

Configured slots in the library

23.226 pkcs11_object_cache_s Struct Reference

Data Fields

- CK_OBJECT_HANDLE [handle](#)
- CK_SLOT_ID **slotid**
- pkcs11_object_ptr [object](#)

23.226.1 Field Documentation

23.226.1.1 handle

CK_OBJECT_HANDLE pkcs11_object_cache_s::handle

Arbitrary (but unique) non-null identifier for an object

23.226.1.2 object

pkcs11_object_ptr pkcs11_object_cache_s::object

The actual object

23.227 pkcs11_object_s Struct Reference

Data Fields

- CK_OBJECT_CLASS [class_id](#)
- CK_ULONG [class_type](#)
- [pkcs11_attr_model](#) const * [attributes](#)
- CK_ULONG [count](#)
- CK_ULONG **size**
- uint16_t **slot**
- CK_FLAGS **flags**
- CK_UTF8CHAR **name** [PKCS11_MAX_LABEL_SIZE+1]
- CK_VOID_PTR **config**
- CK_VOID_PTR **data**
- ta_element_attributes_t **handle_info**

23.227.1 Field Documentation

23.227.1.1 attributes

```
pkcs11_attrib_model const* pkcs11_object_s::attributes
```

List of attribute models this object possesses

23.227.1.2 class_id

```
CK_OBJECT_CLASS pkcs11_object_s::class_id
```

The Class Identifier

23.227.1.3 class_type

```
CK_ULONG pkcs11_object_s::class_type
```

The Class Type

23.227.1.4 count

```
CK_ULONG pkcs11_object_s::count
```

Count of attribute models

23.228 pkcs11_rsa_key_info_s Struct Reference

23.229 pkcs11_session_ctx_s Struct Reference

```
#include <lib/pkcs11/pkcs11_session.h>
```

Data Fields

- CK_BBOOL **initialized**
- pkcs11_slot_ctx_ptr **slot**
- CK_SESSION_HANDLE **handle**
- CK_STATE **state**
- CK_ULONG **error**
- CK_ATTRIBUTE_PTR **attrib_list**
- CK_ULONG **attrib_count**
- CK_ULONG **object_index**
- CK_ULONG **object_count**
- CK_OBJECT_HANDLE **active_object**
- CK_MECHANISM_TYPE **active_mech**
- [pkcs11_session_mech_ctx](#) **active_mech_data**

23.229.1 Detailed Description

Session Context

23.230 pkcs11_session_mech_ctx_s Struct Reference

Data Fields

- [atcac_hmac_ctx_t](#) **hmac**
- [atca_aes_cmac_ctx_t](#) **cmac**
- [atca_aes_cbc_ctx_t](#) **cbc**
- struct {
 [atca_aes_gcm_ctx_t](#) **context**
 CK_BYTE **tag_len**
 } **gcm**
- struct {
 uint8_t **iv** [TA_AES_GCM_IV_LENGTH]
 uint8_t **aad** [ATCA_AES128_BLOCK_SIZE]
 CK_BYTE **aad_len**
 } **gcm_single**

23.231 pkcs11_slot_ctx_s Struct Reference

```
#include <lib/pkcs11/pkcs11_slot.h>
```

Data Fields

- CK_BYTE **slot_state**
- CK_SLOT_ID **slot_id**
- [ATCADevice](#) **device_ctx**
- [ATCAIfaceCfg](#) **interface_config**
- CK_SESSION_HANDLE **session**
- [atecc608_config_t](#) **cfg_zone**
- CK_FLAGS **flags**
- uint16_t **user_pin_handle**
- uint16_t **so_pin_handle**
- CK_UTF8CHAR **label** [PKCS11_MAX_LABEL_SIZE+1]
- CK_BBOOL **logged_in**
- CK_BYTE [read_key](#) [32]

23.231.1 Detailed Description

Slot Context

23.231.2 Field Documentation

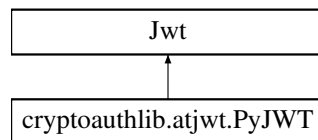
23.231.2.1 read_key

```
CK_BYTE pkcs11_slot_ctx_s::read_key[32]
```

Accepted through C_Login as the user pin

23.232 cryptoauthlib.atjwt.PyJWT Class Reference

Inheritance diagram for cryptoauthlib.atjwt.PyJWT:



Public Member Functions

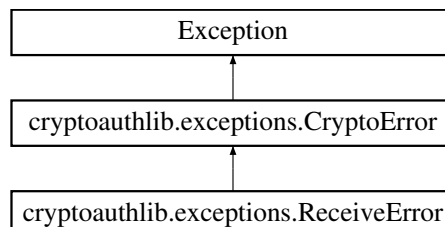
- def **__init__** (self, slot=0, iface_cfg=None, options=None)
- def **register_algorithm** (self, alg_id, algorithm)

23.232.1 Detailed Description

Extended PyJWT class from the pyjwt module

23.233 cryptoauthlib.exceptions.ReceiveError Class Reference

Inheritance diagram for cryptoauthlib.exceptions.ReceiveError:

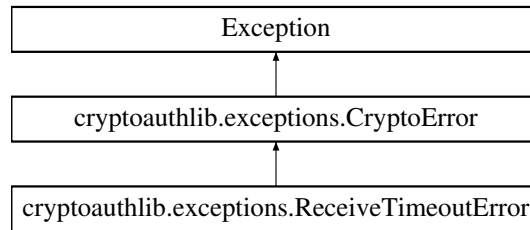


23.233.1 Detailed Description

Timed out while waiting for response. Number of bytes received is > 0.

23.234 cryptoauthlib.exceptions.ReceiveTimeoutError Class Reference

Inheritance diagram for cryptoauthlib.exceptions.ReceiveTimeoutError:

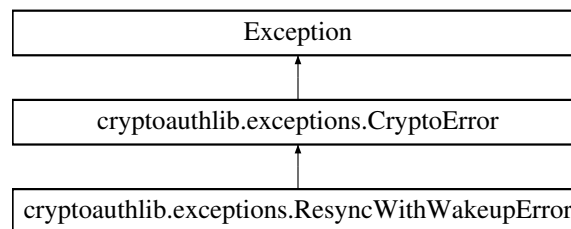


23.234.1 Detailed Description

for Microchip PHY protocol, timeout on receipt waiting for master

23.235 cryptoauthlib.exceptions.ResyncWithWakeupError Class Reference

Inheritance diagram for cryptoauthlib.exceptions.ResyncWithWakeupError:



23.235.1 Detailed Description

Re-synchronization succeeded, but only after generating a Wake-up

23.236 secure_boot_config_bits Struct Reference

Data Fields

- uint16_t **secure_boot_mode**: 2
- uint16_t **secure_boot_reserved1**: 1
- uint16_t **secure_boot_persistent_enable**: 1
- uint16_t **secure_boot_rand_nonce**: 1
- uint16_t **secure_boot_reserved2**: 3
- uint16_t **secure_boot_sig_dig**: 4
- uint16_t **secure_boot_pub_key**: 4

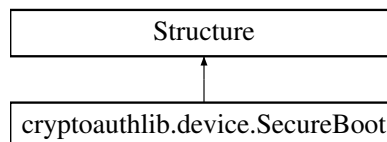
23.237 secure_boot_parameters Struct Reference

Data Fields

- [memory_parameters](#) `memory_params`
- [atcac_sha2_256_ctx](#) `s_sha_context`
- `uint8_t app_digest` [ATCA_SHA_DIGEST_SIZE]

23.238 cryptoauthlib.device.SecureBoot Class Reference

Inheritance diagram for `cryptoauthlib.device.SecureBoot`:



Static Protected Attributes

- `list _fields_`
- `int _pack_ = 1`

23.238.1 Detailed Description

SecureBoot Field Definition

23.238.2 Field Documentation

23.238.2.1 _fields_

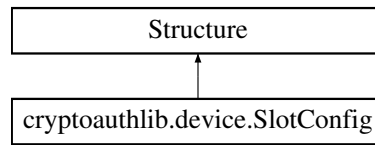
```
list cryptoauthlib.device.SecureBoot._fields_ [static], [protected]
```

Initial value:

```
= [ ('SecureBootMode', c_uint16, 2),
    ('Reserved0', c_uint16, 1),
    ('SecureBootPersistentEnable', c_uint16, 1),
    ('SecureBootRandNonce', c_uint16, 1),
    ('Reserved1', c_uint16, 3),
    ('SecureBootSigDig', c_uint16, 4),
    ('SecureBootPubKey', c_uint16, 4)]
```

23.239 cryptolib.device.SlotConfig Class Reference

Inheritance diagram for cryptolib.device.SlotConfig:



Static Protected Attributes

- list `_fields_`
- int `_pack_` = 1

23.239.1 Detailed Description

Slot Configuration Field Definition

23.239.2 Field Documentation

23.239.2.1 `_fields_`

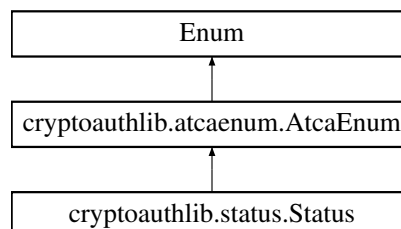
```
list cryptolib.device.SlotConfig._fields_ [static], [protected]
```

Initial value:

```
= [('ReadKey', c_uint16, 4),  
   ('NoMac', c_uint16, 1),  
   ('LimitedUse', c_uint16, 1),  
   ('EncryptRead', c_uint16, 1),  
   ('IsSecret', c_uint16, 1),  
   ('WriteKey', c_uint16, 4),  
   ('WriteConfig', c_uint16, 4)]
```

23.240 cryptolib.status.Status Class Reference

Inheritance diagram for cryptolib.status.Status:



Static Public Attributes

- int **ATCA_SUCCESS** = 0
- int **ATCA_CONFIG_ZONE_LOCKED** = 0x01
- int **ATCA_DATA_ZONE_LOCKED** = 0x02
- int **ATCA_WAKE_FAILED** = -48
- int **ATCA_CHECKMAC_VERIFY_FAILED** = -47
- int **ATCA_PARSE_ERROR** = -46
- int **ATCA_STATUS_CRC** = -44
- int **ATCA_STATUS_UNKNOWN** = -43
- int **ATCA_STATUS_ECC** = -42
- int **ATCA_STATUS_SELFTEST_ERROR** = -41
- int **ATCA_FUNC_FAIL** = -32
- int **ATCA_GEN_FAIL** = -31
- int **ATCA_BAD_PARAM** = -30
- int **ATCA_INVALID_ID** = -29
- int **ATCA_INVALID_SIZE** = -28
- int **ATCA_BAD_CRC** = -27
- int **ATCA_RX_FAIL** = -26
- int **ATCA_RX_NO_RESPONSE** = -25
- int **ATCA_RESYNC_WITH_WAKEUP** = -24
- int **ATCA_PARITY_ERROR** = -23
- int **ATCA_TX_TIMEOUT** = -22
- int **ATCA_RX_TIMEOUT** = -21
- int **ATCA_COMM_FAIL** = -16
- int **ATCA_TIMEOUT** = -15
- int **ATCA_BAD_OPCODE** = -14
- int **ATCA_WAKE_SUCCESS** = -13
- int **ATCA_EXECUTION_ERROR** = -12
- int **ATCA_UNIMPLEMENTED** = -11
- int **ATCA_ASSERT_FAILURE** = -10
- int **ATCA_TX_FAIL** = -9
- int **ATCA_NOT_LOCKED** = -8
- int **ATCA_NO_DEVICES** = -7
- int **ATCA_HEALTH_TEST_ERROR** = -6
- int **ATCA_ALLOC_FAILURE** = -5
- int **ATCA_USE_FLAGS_CONSUMED** = -4
- int **ATCA_NOT_INITIALIZED** = -3

Additional Inherited Members

Public Member Functions inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

- def **__str__** (self)
- def **__eq__** (self, other)
- def **__ne__** (self, other)
- def **__int__** (self)
- def **__hash__** (self)

Data Fields inherited from [cryptoauthlib.atcaenum.AtcaEnum](#)

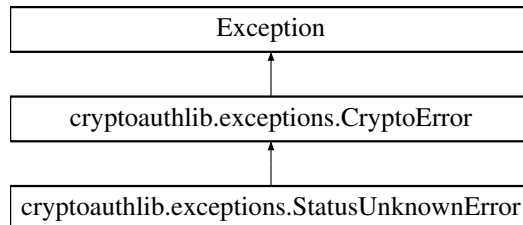
- **name**
- **value**

23.240.1 Detailed Description

Status codes returned from `cryptoauthlib` commands and their meanings. See `atca_status.h`

23.241 `cryptoauthlib.exceptions.StatusUnknownError` Class Reference

Inheritance diagram for `cryptoauthlib.exceptions.StatusUnknownError`:

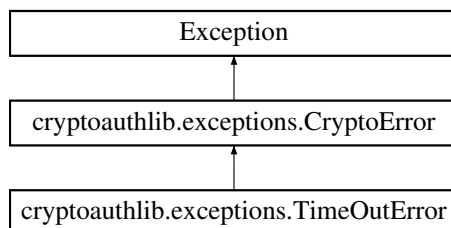


23.241.1 Detailed Description

Response status byte is unknown

23.242 `cryptoauthlib.exceptions.TimeoutError` Class Reference

Inheritance diagram for `cryptoauthlib.exceptions.TimeoutError`:

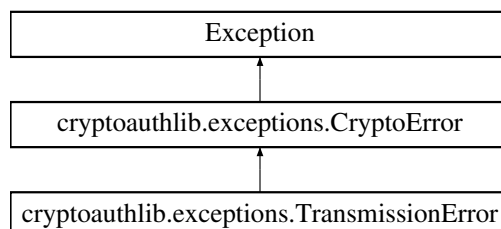


23.242.1 Detailed Description

Timed out while waiting for response. Number of bytes received is 0.

23.243 `cryptoauthlib.exceptions.TransmissionError` Class Reference

Inheritance diagram for `cryptoauthlib.exceptions.TransmissionError`:

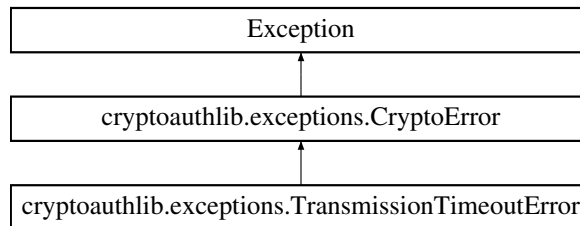


23.243.1 Detailed Description

Failed to write

23.244 `cryptoauthlib.exceptions.TransmissionTimeoutError` Class Reference

Inheritance diagram for `cryptoauthlib.exceptions.TransmissionTimeoutError`:

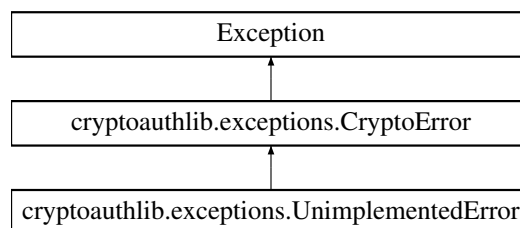


23.244.1 Detailed Description

for Microchip PHY protocol, timeout on transmission waiting for master

23.245 `cryptoauthlib.exceptions.UnimplementedError` Class Reference

Inheritance diagram for `cryptoauthlib.exceptions.UnimplementedError`:

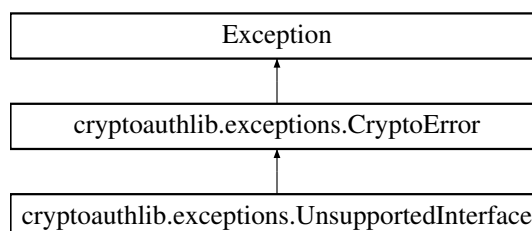


23.245.1 Detailed Description

Function or some element of it hasn't been implemented yet

23.246 `cryptoauthlib.exceptions.UnsupportedInterface` Class Reference

Inheritance diagram for `cryptoauthlib.exceptions.UnsupportedInterface`:

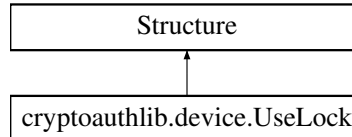


23.246.1 Detailed Description

"The selected interface is not supported by the library

23.247 cryptoauthlib.device.UseLock Class Reference

Inheritance diagram for cryptoauthlib.device.UseLock:



Static Protected Attributes

- list [_fields_](#)
- int [_pack_](#) = 1

23.247.1 Detailed Description

UseLock Field Definition

23.247.2 Field Documentation

23.247.2.1 [_fields_](#)

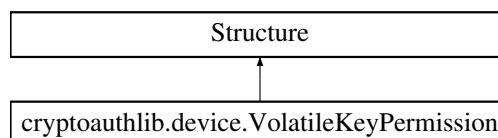
```
list cryptoauthlib.device.UseLock._fields_ [static], [protected]
```

Initial value:

```
= [ ('UseLockEnable', c_uint8, 4),  
    ('UseLockKey', c_uint8, 4)]
```

23.248 cryptoauthlib.device.VolatileKeyPermission Class Reference

Inheritance diagram for cryptoauthlib.device.VolatileKeyPermission:



Static Protected Attributes

- list `_fields_`
- int `_pack_` = 1

23.248.1 Detailed Description

VolatileKeyPermission Field Definition

23.248.2 Field Documentation

23.248.2.1 `_fields_`

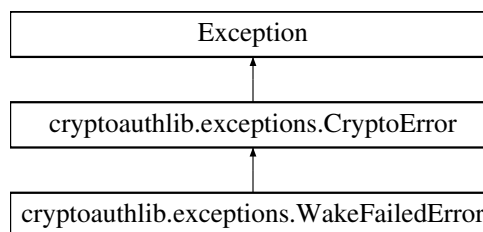
```
list cryptoauthlib.device.VolatileKeyPermission._fields_ [static], [protected]
```

Initial value:

```
= [('VolatileKeyPermitSlot', c_uint8, 4),
    ('Reserved', c_uint8, 3),
    ('VolatileKeyPermitEnable', c_uint8, 1)]
```

23.249 cryptoauthlib.exceptions.WakeFailedError Class Reference

Inheritance diagram for `cryptoauthlib.exceptions.WakeFailedError`:

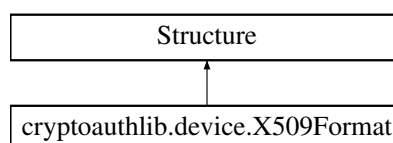


23.249.1 Detailed Description

Device Wake failed

23.250 cryptoauthlib.device.X509Format Class Reference

Inheritance diagram for `cryptoauthlib.device.X509Format`:



Static Protected Attributes

- list [_fields_](#)
- int [_pack_](#) = 1

23.250.1 Detailed Description

X509Format Field Definition

23.250.2 Field Documentation

23.250.2.1 [_fields_](#)

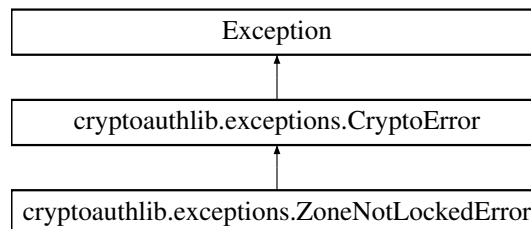
```
list cryptauthlib.device.X509Format._fields_  [static], [protected]
```

Initial value:

```
=  [('PublicPosition', c_uint8, 4),  
    ('TemplateLength', c_uint8, 4)]
```

23.251 cryptauthlib.exceptions.ZoneNotLockedError Class Reference

Inheritance diagram for cryptauthlib.exceptions.ZoneNotLockedError:



23.251.1 Detailed Description

required zone was not locked

Chapter 24

File Documentation

24.1 api_206a.c File Reference

Provides APIs to use with ATSHA206A device.

```
#include <stdlib.h>
#include <stdio.h>
#include "cryptoauthlib.h"
#include "api_206a.h"
```

Functions

- ATCA_STATUS [sha206a_diversify_parent_key](#) (uint8_t *parent_key, uint8_t *diversified_key)
Computes the diversified key based on the parent key provided and device serial number.
- ATCA_STATUS [sha206a_generate_derive_key](#) (uint8_t *parent_key, uint8_t *derived_key, uint8_t param1, uint16_t param2)
Generates the derived key based on the parent key and other parameters provided.
- ATCA_STATUS [sha206a_generate_challenge_response_pair](#) (uint8_t *key, uint8_t *challenge, uint8_t *response)
Generates the response based on Key and Challenge provided.
- ATCA_STATUS [sha206a_authenticate](#) (uint8_t *challenge, uint8_t *expected_response, uint8_t *is_authenticated)
verifies the challenge and provided response using key in device
- ATCA_STATUS [sha206a_verify_device_consumption](#) (uint8_t *is_consumed)
verifies the device is fully consumed or not based on Parent and Derived Key use flags.
- ATCA_STATUS [sha206a_check_dk_useflag_validity](#) (uint8_t *is_consumed)
verifies Derived Key use flags for consumption
- ATCA_STATUS [sha206a_check_pk_useflag_validity](#) (uint8_t *is_consumed)
verifies Parent Key use flags for consumption
- ATCA_STATUS [sha206a_get_dk_useflag_count](#) (uint8_t *dk_available_count)
calculates available Derived Key use counts
- ATCA_STATUS [sha206a_get_pk_useflag_count](#) (uint8_t *pk_available_count)
calculates available Parent Key use counts
- ATCA_STATUS [sha206a_get_dk_update_count](#) (uint8_t *dk_update_count)
Read Derived Key slot update count. It will be wraps around 256.

- ATCA_STATUS [sha206a_write_data_store](#) (uint8_t slot, uint8_t *data, uint8_t block, uint8_t offset, uint8_t len, bool lock_after_write)
Update the data store slot with user data and lock it if necessary.
- ATCA_STATUS [sha206a_read_data_store](#) (uint8_t slot, uint8_t *data, uint8_t offset, uint8_t len)
Read the data stored in Data store.
- ATCA_STATUS [sha206a_get_data_store_lock_status](#) (uint8_t slot, uint8_t *is_locked)
Returns the lock status of the given data store.

24.1.1 Detailed Description

Provides APIs to use with ATSHA206A device.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.1.2 Function Documentation

24.1.2.1 sha206a_authenticate()

```
ATCA_STATUS sha206a_authenticate (
    uint8_t * challenge,
    uint8_t * expected_response,
    uint8_t * is_authenticated )
```

verifies the challenge and provided response using key in device

Parameters

in	<i>challenge</i>	Challenge to be used in the response calculations
in	<i>expected_response</i>	Expected response from the device.
out	<i>is_authenticated</i>	result of expected of response and calcualted response

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.1.2.2 sha206a_check_dk_useflag_validity()

```
ATCA_STATUS sha206a_check_dk_useflag_validity (
    uint8_t * is_consumed )
```

verifies Derived Key use flags for consumption

Parameters

out	<i>is_consumed</i>	indicates if DK is available for consumption.
-----	--------------------	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.1.2.3 sha206a_check_pk_useflag_validity()

```
ATCA_STATUS sha206a_check_pk_useflag_validity (
    uint8_t * is_consumed )
```

verifies Parent Key use flags for consumption

Parameters

out	<i>is_consumed</i>	indicates if PK is available for consumption
-----	--------------------	--

Returns

ATCA_SUCCESS on success, otherwise an error code

24.1.2.4 sha206a_diversify_parent_key()

```
ATCA_STATUS sha206a_diversify_parent_key (
    uint8_t * parent_key,
    uint8_t * diversified_key )
```

Computes the diversified key based on the parent key provided and device serial number.

Parameters

in	<i>parent_key</i>	parent key to be diversified
out	<i>diversified_key</i>	diversified parent key

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.1.2.5 sha206a_generate_challenge_response_pair()

```
ATCA_STATUS sha206a_generate_challenge_response_pair (
    uint8_t * key,
    uint8_t * challenge,
    uint8_t * response )
```

Generates the response based on Key and Challenge provided.

Parameters

in	<i>key</i>	Input data contains device's key
in	<i>challenge</i>	Input data to be used in challenge response calculation
out	<i>response</i>	response derived from key and challenge

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.1.2.6 sha206a_generate_derive_key()

```
ATCA_STATUS sha206a_generate_derive_key (
    uint8_t * parent_key,
    uint8_t * derived_key,
    uint8_t param1,
    uint16_t param2 )
```

Generates the derived key based on the parent key and other parameters provided.

Parameters

in	<i>parent_key</i>	Input data contains device's parent key
out	<i>derived_key</i>	Output data derived from parent key
in	<i>param1</i>	Input data to be used in derive key calculation
in	<i>param2</i>	Input data to be used in derive key calculation

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.1.2.7 sha206a_get_data_store_lock_status()

```
ATCA_STATUS sha206a_get_data_store_lock_status (
    uint8_t slot,
    uint8_t * is_locked )
```

Returns the lock status of the given data store.

Parameters

in	<i>slot</i>	Slot number of the data store
out	<i>is_locked</i>	lock status of the data store

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.1.2.8 sha206a_get_dk_update_count()

```
ATCA_STATUS sha206a_get_dk_update_count (
    uint8_t * dk_update_count )
```

Read Derived Key slot update count. It will be wraps around 256.

Parameters

out	<i>dk_update_count</i>	returns number of times the slot has been updated with derived key
-----	------------------------	--

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.1.2.9 sha206a_get_dk_useflag_count()

```
ATCA_STATUS sha206a_get_dk_useflag_count (
    uint8_t * dk_available_count )
```

calculates available Derived Key use counts

Parameters

out	<i>dk_available_count</i>	counts available bit's as 1
-----	---------------------------	-----------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.1.2.10 sha206a_get_pk_useflag_count()

```
ATCA_STATUS sha206a_get_pk_useflag_count (
    uint8_t * pk_available_count )
```

calculates available Parent Key use counts

Parameters

out	<i>pk_available_count</i>	counts available bit's as 1
-----	---------------------------	-----------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.1.2.11 sha206a_read_data_store()

```
ATCA_STATUS sha206a_read_data_store (
    uint8_t slot,
    uint8_t * data,
    uint8_t offset,
    uint8_t len )
```

Read the data stored in Data store.

Parameters

in	<i>slot</i>	Slot number to read from
in	<i>data</i>	Pointer to hold slot data data
in	<i>offset</i>	Byte offset within the zone to read from.
in	<i>len</i>	data length

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.1.2.12 sha206a_verify_device_consumption()

```
ATCA_STATUS sha206a_verify_device_consumption (
    uint8_t * is_consumed )
```

verifies the device is fully consumed or not based on Parent and Derived Key use flags.

Parameters

out	<i>is_consumed</i>	result of device consumption
-----	--------------------	------------------------------

24.2 api_206a.h File Reference

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.1.2.13 sha206a_write_data_store()

```
ATCA_STATUS sha206a_write_data_store (
    uint8_t slot,
    uint8_t * data,
    uint8_t block,
    uint8_t offset,
    uint8_t len,
    bool lock_after_write )
```

Update the data store slot with user data and lock it if necessary.

Parameters

in	<i>slot</i>	Slot number to be written with data
in	<i>data</i>	Pointer that holds the data
in	<i>block</i>	32-byte block to write to.
in	<i>offset</i>	4-byte word within the specified block to write to. If performing a 32-byte write, this should be 0.
in	<i>len</i>	data length
in	<i>lock_after_write</i>	set 1 to lock slot after write, otherwise 0

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.2 api_206a.h File Reference

Provides api interfaces to use with ATSHA206A device.

```
#include "atca_status.h"
```

Macros

- #define ATCA_SHA206A_ZONE_WRITE_LOCK 0x20u
- #define ATCA_SHA206A_DKEY_CONSUMPTION_MASK 0x01u
- #define ATCA_SHA206A_PKEY_CONSUMPTION_MASK 0x02u
- #define ATCA_SHA206A_SYMMETRIC_KEY_ID_SLOT 0x07u

Enumerations

- enum { SHA206A_DATA_STORE0 =8 , SHA206A_DATA_STORE1 , SHA206A_DATA_STORE2 }

Functions

- ATCA_STATUS [sha206a_diversify_parent_key](#) (uint8_t *parent_key, uint8_t *diversified_key)
Computes the diversified key based on the parent key provided and device serial number.
- ATCA_STATUS [sha206a_generate_derive_key](#) (uint8_t *parent_key, uint8_t *derived_key, uint8_t param1, uint16_t param2)
Generates the derived key based on the parent key and other parameters provided.
- ATCA_STATUS [sha206a_generate_challenge_response_pair](#) (uint8_t *key, uint8_t *challenge, uint8_t *response)
Generates the response based on Key and Challenge provided.
- ATCA_STATUS [sha206a_authenticate](#) (uint8_t *challenge, uint8_t *expected_response, uint8_t *is_authenticated)
verifies the challenge and provided response using key in device
- ATCA_STATUS [sha206a_verify_device_consumption](#) (uint8_t *is_consumed)
verifies the device is fully consumed or not based on Parent and Derived Key use flags.
- ATCA_STATUS [sha206a_check_dk_useflag_validity](#) (uint8_t *is_consumed)
verifies Derived Key use flags for consumption
- ATCA_STATUS [sha206a_check_pk_useflag_validity](#) (uint8_t *is_consumed)
verifies Parent Key use flags for consumption
- ATCA_STATUS [sha206a_get_dk_useflag_count](#) (uint8_t *dk_available_count)
calculates available Derived Key use counts
- ATCA_STATUS [sha206a_get_pk_useflag_count](#) (uint8_t *pk_available_count)
calculates available Parent Key use counts
- ATCA_STATUS [sha206a_get_dk_update_count](#) (uint8_t *dk_update_count)
Read Derived Key slot update count. It will be wraps around 256.
- ATCA_STATUS [sha206a_write_data_store](#) (uint8_t slot, uint8_t *data, uint8_t block, uint8_t offset, uint8_t len, bool lock_after_write)
Update the data store slot with user data and lock it if necessary.
- ATCA_STATUS [sha206a_read_data_store](#) (uint8_t slot, uint8_t *data, uint8_t offset, uint8_t len)
Read the data stored in Data store.
- ATCA_STATUS [sha206a_get_data_store_lock_status](#) (uint8_t slot, uint8_t *is_locked)
Returns the lock status of the given data store.

24.2.1 Detailed Description

Provides api interfaces to use with ATSHA206A device.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.2.2 Function Documentation

24.2.2.1 sha206a_authenticate()

```
ATCA_STATUS sha206a_authenticate (
    uint8_t * challenge,
    uint8_t * expected_response,
    uint8_t * is_authenticated )
```

verifies the challenge and provided response using key in device

Parameters

in	<i>challenge</i>	Challenge to be used in the response calculations
in	<i>expected_response</i>	Expected response from the device.
out	<i>is_authenticated</i>	result of expected of response and calcaulted response

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.2.2.2 sha206a_check_dk_useflag_validity()

```
ATCA_STATUS sha206a_check_dk_useflag_validity (
    uint8_t * is_consumed )
```

verifies Derived Key use flags for consumption

Parameters

out	<i>is_consumed</i>	indicates if DK is available for consumption.
-----	--------------------	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.2.2.3 sha206a_check_pk_useflag_validity()

```
ATCA_STATUS sha206a_check_pk_useflag_validity (
    uint8_t * is_consumed )
```

verifies Parent Key use flags for consumption

Parameters

out	<i>is_consumed</i>	indicates if PK is available for consumption
-----	--------------------	--

Returns

ATCA_SUCCESS on success, otherwise an error code

24.2.2.4 sha206a_diversify_parent_key()

```
ATCA_STATUS sha206a_diversify_parent_key (
    uint8_t * parent_key,
    uint8_t * diversified_key )
```

Computes the diversified key based on the parent key provided and device serial number.

Parameters

in	<i>parent_key</i>	parent key to be diversified
out	<i>diversified_key</i>	diversified parent key

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.2.2.5 sha206a_generate_challenge_response_pair()

```
ATCA_STATUS sha206a_generate_challenge_response_pair (
    uint8_t * key,
    uint8_t * challenge,
    uint8_t * response )
```

Generates the response based on Key and Challenge provided.

Parameters

in	<i>key</i>	Input data contains device's key
in	<i>challenge</i>	Input data to be used in challenge response calculation
out	<i>response</i>	response derived from key and challenge

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.2.2.6 sha206a_generate_derive_key()

```
ATCA_STATUS sha206a_generate_derive_key (
    uint8_t * parent_key,
    uint8_t * derived_key,
    uint8_t param1,
    uint16_t param2 )
```

Generates the derived key based on the parent key and other parameters provided.

Parameters

in	<i>parent_key</i>	Input data contains device's parent key
out	<i>derived_key</i>	Output data derived from parent key
in	<i>param1</i>	Input data to be used in derive key calculation
in	<i>param2</i>	Input data to be used in derive key calculation

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.2.2.7 sha206a_get_data_store_lock_status()

```
ATCA_STATUS sha206a_get_data_store_lock_status (
    uint8_t slot,
    uint8_t * is_locked )
```

Returns the lock status of the given data store.

Parameters

in	<i>slot</i>	Slot number of the data store
out	<i>is_locked</i>	lock status of the data store

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.2.2.8 sha206a_get_dk_update_count()

```
ATCA_STATUS sha206a_get_dk_update_count (
    uint8_t * dk_update_count )
```

Read Derived Key slot update count. It will be wraps around 256.

Parameters

out	<i>dk_update_count</i>	returns number of times the slot has been updated with derived key
-----	------------------------	--

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.2.2.9 sha206a_get_dk_useflag_count()

```
ATCA_STATUS sha206a_get_dk_useflag_count (
    uint8_t * dk_available_count )
```

calculates available Derived Key use counts

Parameters

out	<i>dk_available_count</i>	counts available bit's as 1
-----	---------------------------	-----------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.2.2.10 sha206a_get_pk_useflag_count()

```
ATCA_STATUS sha206a_get_pk_useflag_count (
    uint8_t * pk_available_count )
```

calculates available Parent Key use counts

Parameters

out	<i>pk_available_count</i>	counts available bit's as 1
-----	---------------------------	-----------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.2.2.11 sha206a_read_data_store()

```
ATCA_STATUS sha206a_read_data_store (
    uint8_t slot,
    uint8_t * data,
    uint8_t offset,
    uint8_t len )
```

Read the data stored in Data store.

Parameters

in	<i>slot</i>	Slot number to read from
in	<i>data</i>	Pointer to hold slot data data
in	<i>offset</i>	Byte offset within the zone to read from.
in	<i>len</i>	data length

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.2.2.12 sha206a_verify_device_consumption()

```
ATCA_STATUS sha206a_verify_device_consumption (
    uint8_t * is_consumed )
```

verifies the device is fully consumed or not based on Parent and Derived Key use flags.

Parameters

out	<i>is_consumed</i>	result of device consumption
-----	--------------------	------------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.2.2.13 sha206a_write_data_store()

```
ATCA_STATUS sha206a_write_data_store (
    uint8_t slot,
    uint8_t * data,
    uint8_t block,
    uint8_t offset,
    uint8_t len,
    bool lock_after_write )
```

Update the data store slot with user data and lock it if necessary.

Parameters

in	<i>slot</i>	Slot number to be written with data
in	<i>data</i>	Pointer that holds the data
in	<i>block</i>	32-byte block to write to.
in	<i>offset</i>	4-byte word within the specified block to write to. If performing a 32-byte write, this should be 0.
in	<i>len</i>	data length
in	<i>lock_after_write</i>	set 1 to lock slot after write, otherwise 0

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.3 symmetric_authentication.c File Reference

Contains API for performing the symmetric Authentication between the Host and the device.

```
#include "cryptoauthlib.h"
#include "host/atca_host.h"
#include "symmetric_authentication.h"
```

Functions

- ATCA_STATUS [symmetric_authenticate](#) (uint8_t slot, const uint8_t *master_key, const uint8_t *rand_↵ number)

Function which does the authentication between the host and device.

24.3.1 Detailed Description

Contains API for performing the symmetric Authentication between the Host and the device.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.3.2 Function Documentation

24.3.2.1 symmetric_authenticate()

```
ATCA_STATUS symmetric_authenticate (
    uint8_t slot,
    const uint8_t * master_key,
    const uint8_t * rand_number )
```

Function which does the authentication between the host and device.

Parameters

in	<i>slot</i>	The slot number used for the symmetric authentication.
in	<i>master_key</i>	The master key used for the calculating the symmetric key.
in	<i>rand_number</i>	The 20 byte rand_number from the host.

Returns

ATCA_SUCCESS on successful authentication, otherwise an error code.

24.4 symmetric_authentication.h File Reference

Contains API for performing the symmetric Authentication between the Host and the device.

```
#include "cryptoauthlib.h"
```

Functions

- ATCA_STATUS [symmetric_authenticate](#) (uint8_t slot, const uint8_t *master_key, const uint8_t *rand_number)

Function which does the authentication between the host and device.

24.4.1 Detailed Description

Contains API for performing the symmetric Authentication between the Host and the device.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.4.2 Function Documentation

24.4.2.1 symmetric_authenticate()

```
ATCA_STATUS symmetric_authenticate (  
    uint8_t slot,  
    const uint8_t * master_key,  
    const uint8_t * rand_number )
```

Function which does the authentication between the host and device.

Parameters

in	<i>slot</i>	The slot number used for the symmetric authentication.
in	<i>master_key</i>	The master key used for the calculating the symmetric key.
in	<i>rand_number</i>	The 20 byte rand_number from the host.

Returns

ATCA_SUCCESS on successful authentication, otherwise an error code.

24.5 ascii_kit_host.c File Reference

KIT protocol interpreter.

```
#include <ctype.h>
#include "ascii_kit_host.h"
#include "hal/kit_protocol.h"
#include "talib/talib_fce.h"
```

Functions

- ATCA_STATUS [kit_host_init_phy](#) ([atca_hal_kit_phy_t](#) *phy, [ATCAIface](#) iface)
Initializes a phy structure with a cryptoauthlib hal adapter.
- ATCA_STATUS [kit_host_init](#) ([ascii_kit_host_context_t](#) *ctx, [ATCAIfaceCfg](#) *iface[], const size_t iface_count, const [atca_hal_kit_phy_t](#) *phy, const uint32_t flags)
Initializes the kit protocol parser context.
- size_t [kit_host_format_response](#) (uint8_t *response, size_t rlen, ATCA_STATUS status, uint8_t *data, size_t dlen)
Format the status and data into the kit protocol response format.
- ATCA_STATUS [kit_host_process_cmd](#) ([ascii_kit_host_context_t](#) *ctx, const [kit_host_map_entry_t](#) *cmd↵_list, int argc, char *argv[], uint8_t *response, size_t *rlen)
Iterate through a command list to match the given command and then will execute it.
- ATCA_STATUS [kit_host_process_ta](#) ([ascii_kit_host_context_t](#) *ctx, int argc, char *argv[], uint8_t↵ *response, size_t *rlen)
- ATCA_STATUS [kit_host_process_line](#) ([ascii_kit_host_context_t](#) *ctx, uint8_t *input_line, size_t ilen, uint8_t *response, size_t *rlen)
Parse a line as a kit protocol command. The kit protocol is printable ascii and each line ends with a newline character.
- void [kit_host_task](#) ([ascii_kit_host_context_t](#) *ctx)
Non returning kit protocol runner using the configured physical interface that was provided when the context was initialized.

24.5.1 Detailed Description

KIT protocol interpreter.

Copyright

(c) 2018 Microchip Technology Inc. and its subsidiaries. You may use this software and any derivatives exclusively with Microchip products.

24.5.2 Function Documentation

24.5.2.1 kit_host_init()

```
ATCA_STATUS kit_host_init (
    ascii_kit_host_context_t * ctx,
    ATCAIfaceCfg * iface[],
    const size_t iface_count,
    const atca_hal_kit_phy_t * phy,
    const uint32_t flags )
```

Initializes the kit protocol parser context.

Returns

ATCA_SUCCESS on success, otherwise an error code

Parameters

<i>ctx</i>	Kit protocol parser context
<i>iface</i>	List of device configurations which will be used
<i>iface_count</i>	Number of configurations provided
<i>phy</i>	Kit protocol physical adapter
<i>flags</i>	Option Flags

24.5.2.2 kit_host_init_phy()

```
ATCA_STATUS kit_host_init_phy (
    atca_hal_kit_phy_t * phy,
    ATCAIface iface )
```

Initializes a phy structure with a cryptoauthlib hal adapter.

Returns

ATCA_SUCCESS on success, otherwise an error code

24.6 ascii_kit_host.h File Reference

KIT protocol interpreter.

```
#include "cryptoauthlib.h"
```

Data Structures

- struct [_ascii_kit_host_context](#)
- struct [_kit_host_map_entry](#)

Macros

- #define **KIT_LAYER_DELIMITER** ':'
- #define **KIT_DATA_BEGIN_DELIMITER** '('
- #define **KIT_DATA_END_DELIMITER** ')'
- #define **KIT_MESSAGE_DELIMITER** '\n'
- #define **KIT_MESSAGE_SIZE_MAX** (2500)
The Kit Protocol maximum message size.
- #define **KIT_SECTION_NAME_SIZE_MAX** **KIT_MESSAGE_SIZE_MAX**
- #define **KIT_VERSION_SIZE_MAX** (32)
- #define **KIT_FIRMWARE_SIZE_MAX** (32)

Typedefs

- typedef struct **_ascii_kit_host_context** **ascii_kit_host_context_t**
- typedef struct **_kit_host_map_entry** **kit_host_map_entry_t**

Functions

- ATCA_STATUS **kit_host_init_phy** (**atca_hal_kit_phy_t** *phy, **ATCAIface** iface)
Initializes a phy structure with a cryptoauthlib hal adapter.
- ATCA_STATUS **kit_host_init** (**ascii_kit_host_context_t** *ctx, **ATCAIfaceCfg** *iface[], const size_t iface_count, const **atca_hal_kit_phy_t** *phy, const uint32_t flags)
Initializes the kit protocol parser context.
- size_t **kit_host_format_response** (uint8_t *response, size_t rlen, ATCA_STATUS status, uint8_t *data, size_t dlen)
Format the status and data into the kit protocol response format.
- ATCA_STATUS **kit_host_process_cmd** (**ascii_kit_host_context_t** *ctx, const **kit_host_map_entry_t** *cmd↵_list, int argc, char *argv[], uint8_t *response, size_t *rlen)
Iterate through a command list to match the given command and then will execute it.
- ATCA_STATUS **kit_host_process_line** (**ascii_kit_host_context_t** *ctx, uint8_t *input_line, size_t ilen, uint8_t *response, size_t *rlen)
Parse a line as a kit protocol command. The kit protocol is printable ascii and each line ends with a newline character.
- void **kit_host_task** (**ascii_kit_host_context_t** *ctx)
Non returning kit protocol runner using the configured physical interface that was provided when the context was initialized.

24.6.1 Detailed Description

KIT protocol interpreter.

Copyright

(c) 2018 Microchip Technology Inc. and its subsidiaries. You may use this software and any derivatives exclusively with Microchip products.

24.6.2 Macro Definition Documentation

24.6 ascii_kit_host.h File Reference

24.6.2.1 KIT_MESSAGE_SIZE_MAX

```
#define KIT_MESSAGE_SIZE_MAX (2500)
```

The Kit Protocol maximum message size.

Note

Send: <target>:<command>(optional hex bytes to send)
Receive: <status hex byte>(optional hex bytes of response)

24.6.3 Typedef Documentation

24.6.3.1 kit_host_map_entry_t

```
typedef struct _kit_host_map_entry kit_host_map_entry_t
```

Used to create command tables for the kit host parser

24.6.4 Function Documentation

24.6.4.1 kit_host_init()

```
ATCA_STATUS kit_host_init (  
    ascii_kit_host_context_t * ctx,  
    ATCAIfaceCfg * iface[],  
    const size_t iface_count,  
    const atca_hal_kit_phy_t * phy,  
    const uint32_t flags )
```

Initializes the kit protocol parser context.

Returns

ATCA_SUCCESS on success, otherwise an error code

Parameters

<i>ctx</i>	Kit protocol parser context
<i>iface</i>	List of device configurations which will be used
<i>iface_count</i>	Number of configurations provided
<i>phy</i>	Kit protocol physical adapter
<i>flags</i>	Option Flags

24.6.4.2 kit_host_init_phy()

```
ATCA_STATUS kit_host_init_phy (
    atca_hal_kit_phy_t * phy,
    ATCAIface iface )
```

Initializes a phy structure with a cryptoauthlib hal adapter.

Returns

ATCA_SUCCESS on success, otherwise an error code

24.7 trust_pkcs11_config.c File Reference

PKCS11 Trust Platform Configuration.

```
#include "cryptoauthlib.h"
#include "pkcs11_config.h"
#include "pkcs11/pkcs11_object.h"
#include "pkcs11/pkcs11_slot.h"
```

24.7.1 Detailed Description

PKCS11 Trust Platform Configuration.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.8 io_protection_key.h File Reference

Provides required interface to access IO protection key.

```
#include "atca_status.h"
```

Functions

- ATCA_STATUS **io_protection_get_key** (uint8_t *io_key)
- ATCA_STATUS **io_protection_set_key** (uint8_t *io_key)

24.8.1 Detailed Description

Provides required interface to access IO protection key.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.9 secure_boot.c File Reference

Provides required APIs to manage secure boot under various scenarios.

```
#include <string.h>
#include "secure_boot.h"
#include "io_protection_key.h"
#include "basic/atca_basic.h"
```

Functions

- ATCA_STATUS [secure_boot_process](#) (void)
Handles secure boot functionality through initialization, execution, and de-initialization.
- ATCA_STATUS [bind_host_and_secure_element_with_io_protection](#) (uint16_t slot)
Binds host MCU and Secure element with IO protection key.

24.9.1 Detailed Description

Provides required APIs to manage secure boot under various scenarios.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.9.2 Function Documentation

24.9.2.1 bind_host_and_secure_element_with_io_protection()

```
ATCA_STATUS bind_host_and_secure_element_with_io_protection (
    uint16_t slot )
```

Binds host MCU and Secure element with IO protection key.

Parameters

<code>in</code>	<code>slot</code>	The slot number of IO protection Key.
-----------------	-------------------	---------------------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.9.2.2 secure_boot_process()

```
ATCA_STATUS secure_boot_process (
    void )
```

Handles secure boot functionality through initialization, execution, and de-initialization.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.10 secure_boot.h File Reference

Provides required APIs to manage secure boot under various scenarios.

```
#include "atca_status.h"
#include "secure_boot_memory.h"
#include "atca_command.h"
#include "crypto/atca_crypto_sw_sha2.h"
```

Data Structures

- struct [secure_boot_config_bits](#)
- struct [secure_boot_parameters](#)

Macros

- #define **SECURE_BOOT_CONFIG_DISABLE** 0
- #define **SECURE_BOOT_CONFIG_FULL_BOTH** 1
- #define **SECURE_BOOT_CONFIG_FULL_SIGN** 2
- #define **SECURE_BOOT_CONFIG_FULL_DIG** 3
- #define **SECURE_BOOT_CONFIGURATION** SECURE_BOOT_CONFIG_FULL_DIG
- #define **SECURE_BOOT_DIGEST_ENCRYPT_ENABLED** true
- #define **SECURE_BOOT_UPGRADE_SUPPORT** true

Functions

- ATCA_STATUS [secure_boot_process](#) (void)
Handles secure boot functionality through initialization, execution, and de-initialization.
- ATCA_STATUS [bind_host_and_secure_element_with_io_protection](#) (uint16_t slot)
Binds host MCU and Secure element with IO protection key.
- ATCA_STATUS [host_generate_random_number](#) (uint8_t *rand)

24.10.1 Detailed Description

Provides required APIs to manage secure boot under various scenarios.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.10.2 Function Documentation

24.10.2.1 [bind_host_and_secure_element_with_io_protection\(\)](#)

```
ATCA_STATUS bind_host_and_secure_element_with_io_protection (
    uint16_t slot )
```

Binds host MCU and Secure element with IO protection key.

Parameters

in	slot	The slot number of IO protection Key.
----	------	---------------------------------------

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.10.2.2 [secure_boot_process\(\)](#)

```
ATCA_STATUS secure_boot_process (
    void )
```

Handles secure boot functionality through initialization, execution, and de-initialization.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.11 secure_boot_memory.h File Reference

Provides interface to memory component for the secure boot.

```
#include "atca_status.h"  
#include "atca_command.h"
```

Data Structures

- struct [memory_parameters](#)

Functions

- ATCA_STATUS **secure_boot_init_memory** ([memory_parameters](#) *memory_params)
- ATCA_STATUS **secure_boot_read_memory** (uint8_t *pu8_data, uint32_t *pu32_target_length)
- ATCA_STATUS **secure_boot_write_memory** (uint8_t *pu8_data, uint32_t *pu32_target_length)
- void **secure_boot_deinit_memory** ([memory_parameters](#) *memory_params)
- ATCA_STATUS **secure_boot_mark_full_copy_completion** (void)
- bool **secure_boot_check_full_copy_completion** (void)

24.11.1 Detailed Description

Provides interface to memory component for the secure boot.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.12 tflxtls_cert_def_4_device.c File Reference

TNG TLS device certificate definition.

```
#include "atcacert/atcacert_def.h"  
#include "tngtls_cert_def_1_signer.h"  
#include "tflxtls_cert_def_4_device.h"
```

24.12.1 Detailed Description

TNG TLS device certificate definition.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.13 tflxtls_cert_def_4_device.h File Reference

TNG TLS device certificate definition.

```
#include "atcacert/atcacert_def.h"
```

Variables

- const uint8_t **g_tflxtls_cert_template_4_device** [500]
- const [atcacert_def_t](#) **g_tflxtls_cert_def_4_device**
- const [atcacert_cert_element_t](#) **g_tflxtls_cert_elements_4_device** []

24.13.1 Detailed Description

TNG TLS device certificate definition.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.14 tng_atca.c File Reference

TNG Helper Functions.

```
#include <string.h>
#include "cryptoauthlib.h"
#include "tng_atca.h"
#include "tnglora_cert_def_2_device.h"
#include "tnglora_cert_def_4_device.h"
#include "tngtls_cert_def_2_device.h"
#include "tngtls_cert_def_3_device.h"
#include "tflxtls_cert_def_4_device.h"
#include "atcacert/atcacert_def.h"
```

24.14.1 Detailed Description

TNG Helper Functions.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.15 tng_atca.h File Reference

TNG Helper Functions.

```
#include "atca_basic.h"
#include "atcacert/atcacert_def.h"
```

Macros

- `#define ATCA_OTP_CODE_SIZE (8u)`

Functions

- `const atcacert_def_t * tng_map_get_device_cert_def (int index)`
Helper function to iterate through all trust cert definitions.
- `ATCA_STATUS tng_get_device_cert_def (const atcacert_def_t **cert_def)`
Get the TNG device certificate definition.
- `ATCA_STATUS tng_get_device_cert_def_ext (ATCADevice device, const atcacert_def_t **cert_def)`
Get the TNG device certificate definition.
- `ATCA_STATUS tng_get_device_pubkey (uint8_t *public_key)`
Uses GenKey command to calculate the public key from the primary device public key.

24.15.1 Detailed Description

TNG Helper Functions.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.16 tng_atcacert_client.c File Reference

Client side certificate I/O functions for TNG devices.

```
#include "tng_atca.h"
#include "atcacert/atcacert_client.h"
#include "tng_atcacert_client.h"
#include "tngtls_cert_def_1_signer.h"
#include "tng_root_cert.h"
#include <limits.h>
```

24.16.1 Detailed Description

Client side certificate I/O functions for TNG devices.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.17 tng_atcacert_client.h File Reference

Client side certificate I/O functions for TNG devices.

```
#include <stdint.h>
#include "atcacert/atcacert.h"
```

Functions

- int [tng_atcacert_max_device_cert_size](#) (size_t *max_cert_size)
Return the maximum possible certificate size in bytes for a TNG device certificate. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificate.
- int [tng_atcacert_read_device_cert](#) (uint8_t *cert, size_t *cert_size, const uint8_t *signer_cert)
Reads the device certificate for a TNG device.
- int [tng_atcacert_device_public_key](#) (uint8_t *public_key, uint8_t *cert)
Reads the device public key.
- int [tng_atcacert_max_signer_cert_size](#) (size_t *max_cert_size)
Return the maximum possible certificate size in bytes for a TNG signer certificate. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificate.
- int [tng_atcacert_read_signer_cert](#) (uint8_t *cert, size_t *cert_size)
Reads the signer certificate for a TNG device.
- int [tng_atcacert_signer_public_key](#) (uint8_t *public_key, uint8_t *cert)
Reads the signer public key.
- int [tng_atcacert_root_cert_size](#) (size_t *cert_size)
Get the size of the TNG root cert.
- int [tng_atcacert_root_cert](#) (uint8_t *cert, size_t *cert_size)
Get the TNG root cert.
- int [tng_atcacert_root_public_key](#) (uint8_t *public_key)
Gets the root public key.

24.17.1 Detailed Description

Client side certificate I/O functions for TNG devices.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.18 tng_root_cert.c File Reference

TNG root certificate (DER)

```
#include <stdint.h>
#include <stddef.h>
#include "tng_root_cert.h"
```


Variables

- `const uint8_t g_cryptoauth_root_ca_002_cert` [501]
- `const size_t g_cryptoauth_root_ca_002_cert_size` = `sizeof(g_cryptoauth_root_ca_002_cert)`

24.18.1 Detailed Description

TNG root certificate (DER)

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.19 tng_root_cert.h File Reference

TNG root certificate (DER)

```
#include <stdint.h>
```

- `#define CRYPTOAUTH_ROOT_CA_002_PUBLIC_KEY_OFFSET` 266
- `const uint8_t g_cryptoauth_root_ca_002_cert` []
- `const size_t g_cryptoauth_root_ca_002_cert_size`

24.19.1 Detailed Description

TNG root certificate (DER)

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.20 tnglora_cert_def_1_signer.c File Reference

TNG LORA signer certificate definition.

```
#include "atcacert/atcacert_def.h"
#include "tngtls_cert_def_1_signer.h"
#include "tnglora_cert_def_1_signer.h"
```

24.20.1 Detailed Description

TNG LORA signer certificate definition.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.21 tnglora_cert_def_1_signer.h File Reference

TNG LORA signer certificate definition.

```
#include "atcacert/atcacert_def.h"
```

Variables

- ATCA_DLL const [atcacert_def_t g_tnglora_cert_def_1_signer](#)

24.21.1 Detailed Description

TNG LORA signer certificate definition.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.22 tnglora_cert_def_2_device.c File Reference

TNG LORA device certificate definition.

```
#include "atcacert/atcacert_def.h"  
#include "tngtls_cert_def_2_device.h"  
#include "tngtls_cert_def_1_signer.h"  
#include "tnglora_cert_def_1_signer.h"  
#include "tnglora_cert_def_2_device.h"
```

24.22.1 Detailed Description

TNG LORA device certificate definition.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.23 tnglora_cert_def_2_device.h File Reference

TNG LORA device certificate definition.

```
#include "atcacert/atcacert_def.h"
```

Variables

- ATCA_DLL const [atcacert_def_t](#) **g_tnqlora_cert_def_2_device**

24.23.1 Detailed Description

TNG LORA device certificate definition.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.24 tnqlora_cert_def_4_device.c File Reference

TNG LORA device certificate definition.

```
#include "atcacert/atcacert_def.h"
#include "tnqlora_cert_def_4_device.h"
#include "tnqlora_cert_def_1_signer.h"
```

24.24.1 Detailed Description

TNG LORA device certificate definition.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.25 tnqlora_cert_def_4_device.h File Reference

TNG LORA device certificate definition.

```
#include "atcacert/atcacert_def.h"
```

- #define **TNQLORA_CERT_TEMPLATE_4_DEVICE_SIZE** 552
- ATCA_DLL const [atcacert_def_t](#) **g_tnqlora_cert_def_4_device**
- SHARED_LIB_EXPORT const uint8_t **g_tnqlora_cert_template_4_device** []
- SHARED_LIB_EXPORT const [atcacert_cert_element_t](#) **g_tnqlora_cert_elements_4_device** []

24.25.1 Detailed Description

TNG LORA device certificate definition.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.26 tngtls_cert_def_1_signer.c File Reference

TNG TLS signer certificate definition.

```
#include "atcacert/atcacert_def.h"  
#include "tngtls_cert_def_1_signer.h"
```

24.26.1 Detailed Description

TNG TLS signer certificate definition.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.27 tngtls_cert_def_1_signer.h File Reference

TNG TLS signer certificate definition.

```
#include "atcacert/atcacert_def.h"
```

- #define **TNGTLS_CERT_TEMPLATE_1_SIGNER_SIZE** 520
- ATCA_DLL const [atcacert_def_t](#) **g_tngtls_cert_def_1_signer**
- SHARED_LIB_EXPORT const uint8_t **g_tngtls_cert_template_1_signer** []
- SHARED_LIB_EXPORT const [atcacert_cert_element_t](#) **g_tngtls_cert_elements_1_signer** []

24.27.1 Detailed Description

TNG TLS signer certificate definition.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.28 tngtls_cert_def_2_device.c File Reference

TNG TLS device certificate definition.

```
#include "atcacert/atcacert_def.h"  
#include "tngtls_cert_def_2_device.h"  
#include "tngtls_cert_def_1_signer.h"
```

24.28.1 Detailed Description

TNG TLS device certificate definition.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.29 tngtls_cert_def_2_device.h File Reference

TNG TLS device certificate definition.

```
#include "atcacert/atcacert_def.h"
```

- #define **TNGTLS_CERT_TEMPLATE_2_DEVICE_SIZE** 505
- #define **TNGTLS_CERT_ELEMENTS_2_DEVICE_COUNT** 2
- ATCA_DLL const [atcacert_def_t](#) **g_tngtls_cert_def_2_device**
- SHARED_LIB_EXPORT const uint8_t **g_tngtls_cert_template_2_device** []
- SHARED_LIB_EXPORT const [atcacert_cert_element_t](#) **g_tngtls_cert_elements_2_device** []

24.29.1 Detailed Description

TNG TLS device certificate definition.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.30 tngtls_cert_def_3_device.c File Reference

TNG TLS device certificate definition.

```
#include "atcacert/atcacert_def.h"  
#include "tngtls_cert_def_3_device.h"  
#include "tngtls_cert_def_1_signer.h"
```

24.30.1 Detailed Description

TNG TLS device certificate definition.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.31 tngtls_cert_def_3_device.h File Reference

TNG TLS device certificate definition.

```
#include "atcacert/atcacert_def.h"
```

- #define **TNGTLS_CERT_TEMPLATE_3_DEVICE_SIZE** 546
- ATCA_DLL const [atcacert_def_t g_tngtls_cert_def_3_device](#)
- ATCA_DLL const uint8_t [g_tngtls_cert_template_3_device](#) []
- ATCA_DLL const [atcacert_cert_element_t g_tngtls_cert_elements_3_device](#) []

24.31.1 Detailed Description

TNG TLS device certificate definition.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.32 wpc_apis.c File Reference

Provides api interfaces for WPC authentication.

```
#include "cryptoauthlib.h"  
#include "wpc_apis.h"  
#include "wpccert_client.h"  
#include "atcacert/atcacert_client.h"
```

24.32.1 Detailed Description

Provides api interfaces for WPC authentication.

Copyright

(c) 2015-2021 Microchip Technology Inc. and its subsidiaries.

24.33 wpc_apis.h File Reference

Provides api interfaces for WPC authentication.

```
#include "wpc_check_config.h"
```

Macros

- #define **WPC_PROTOCOL_VERSION** 0x01
- #define **WPC_PROTOCOL_MAX_VERSION** 0x01
- #define **WPC_TBS_AUTH_PREFIX** 0x41
- #define **WPC_CONST_N_RH** ATCA_SHA256_DIGEST_SIZE
- #define **WPC_CONST_OS_MC** (2 + WPC_CONST_N_RH)
- #define **WPC_HEADER(x)** ((WPC_PROTOCOL_VERSION << 4) | x)
- #define **WPC_GET_DIGESTS_TYPE** 0x09
- #define **WPC_GET_DIGESTS_HEADER** WPC_HEADER(WPC_GET_DIGESTS_TYPE)
- #define **WPC_GET_DIGESTS_LENGTH** (2)
- #define **WPC_GET_CERTIFICATE_TYPE** 0x0A
- #define **WPC_GET_CERTIFICATE_HEADER** WPC_HEADER(WPC_GET_CERTIFICATE_TYPE)
- #define **WPC_GET_CERTIFICATE_LENGTH** (4)
- #define **WPC_CHALLENGE_TYPE** 0x0B
- #define **WPC_CHALLENGE_HEADER** WPC_HEADER(WPC_CHALLENGE_TYPE)
- #define **WPC_CHALLENGE_NONCE_LENGTH** (16)
- #define **WPC_CHALLENGE_LENGTH** (2 + WPC_CHALLENGE_NONCE_LENGTH)
- #define **WPC_DIGESTS_TYPE** 0x01
- #define **WPC_DIGESTS_HEADER** WPC_HEADER(WPC_DIGESTS_TYPE)
- #define **WPC_DIGESTS_LENGTH(x)** (2 + (ATCA_SHA256_DIGEST_SIZE * x))
- #define **WPC_CERTIFICATE_TYPE** 0x02
- #define **WPC_CERTIFICATE_HEADER** WPC_HEADER(WPC_CERTIFICATE_TYPE)
- #define **WPC_CERTIFICATE_LENGTH(x)** (1 + x)
- #define **WPC_CHALLENGE_AUTH_TYPE** 0x03
- #define **WPC_CHALLENGE_AUTH_HEADER** WPC_HEADER(WPC_CHALLENGE_AUTH_TYPE)
- #define **WPC_CHALLENGE_AUTH_LENGTH** (67)
- #define **WPC_ERROR_TYPE** 0x07
- #define **WPC_ERROR_HEADER** WPC_HEADER(WPC_ERROR_TYPE)
- #define **WPC_ERROR_LENGTH** (3)
- #define **WPC_ERROR_INVALID_REQUEST** (0x01)
- #define **WPC_ERROR_UNSUPPORTED_PROTOCOL** (0x02)
- #define **WPC_ERROR_BUSY** (0x03)
- #define **WPC_ERROR_UNSPECIFIED** (0x04)

Variables

- const uint8_t **g_root_ca_digest** []

24.33.1 Detailed Description

Provides api interfaces for WPC authentication.

Copyright

(c) 2015-2021 Microchip Technology Inc. and its subsidiaries.

24.34 wpccert_client.c File Reference

Provides api interfaces for accessing WPC certificates from device.

```
#include "wpc_check_config.h"
#include "wpccert_client.h"
#include "atcacert/atcacert_def.h"
#include "atcacert/atcacert_der.h"
#include "atcacert/atcacert_client.h"
#include "atca_basic.h"
```

Functions

- ATCA_STATUS **wpccert_read_cert_size** ([ATCADevice](#) device, const [atcacert_def_t](#) *cert_def, size_t *cert_size)
- ATCA_STATUS **wpccert_read_cert** ([ATCADevice](#) device, const [atcacert_def_t](#) *cert_def, uint8_t *cert, size_t *cert_size)
WPC API -.
- ATCA_STATUS **wpccert_read_pdu_cert** ([ATCADevice](#) device, uint8_t *cert, size_t *cert_size, uint8_t slot)
- ATCA_STATUS **wpccert_read_mfg_cert** ([ATCADevice](#) device, uint8_t *cert, size_t *cert_size, uint8_t slot)
- ATCA_STATUS **wpccert_public_key** (const [atcacert_def_t](#) *cert_def, uint8_t *public_key, uint8_t *cert)

24.34.1 Detailed Description

Provides api interfaces for accessing WPC certificates from device.

Copyright

(c) 2015-2021 Microchip Technology Inc. and its subsidiaries.

24.34.2 Function Documentation

24.34.2.1 wpccert_read_cert()

```
ATCA_STATUS wpccert_read_cert (
    ATCADevice device,
    const atcacert\_def\_t * cert_def,
    uint8_t * cert,
    size_t * cert_size )
```

WPC API -.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.34.2.2 wpccert_read_mfg_cert()

```
ATCA_STATUS wpccert_read_mfg_cert (
    ATCADevice device,
    uint8_t * cert,
    size_t * cert_size,
    uint8_t slot )
```

CA or TA device MFG Cert

CA2 device MFG Cert

24.35 wpccert_client.h File Reference

Provides api interfaces for accessing WPC certificates from device.

```
#include "cryptoauthlib.h"
#include "atcacert/atcacert_def.h"
```

Functions

- uint8_t **wpccert_get_slots_populated** (void)
- uint8_t **wpccert_get_slot_count** (void)
- ATCA_STATUS **wpccert_get_slot_info** (uint16_t *dig_handle, const [atcacert_def_t](#) **def, uint8_t **mfg, uint8_t *root_dgst, uint16_t *root_dgst_handle, uint8_t slot)
- ATCA_STATUS **wpccert_read_cert_size** ([ATCADevice](#) device, const [atcacert_def_t](#) *cert_def, size_t *cert_size)
- ATCA_STATUS **wpccert_read_cert** ([ATCADevice](#) device, const [atcacert_def_t](#) *cert_def, uint8_t *cert, size_t *cert_size)
- *WPC API -.*
- ATCA_STATUS **wpccert_write_cert** ([ATCADevice](#) device, const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size)
- ATCA_STATUS **wpccert_read_pdu_cert** ([ATCADevice](#) device, uint8_t *cert, size_t *cert_size, uint8_t slot)
- ATCA_STATUS **wpccert_read_mfg_cert** ([ATCADevice](#) device, uint8_t *cert, size_t *cert_size, uint8_t slot)
- ATCA_STATUS **wpccert_public_key** (const [atcacert_def_t](#) *cert_def, uint8_t *public_key, uint8_t *cert)

24.35.1 Detailed Description

Provides api interfaces for accessing WPC certificates from device.

Copyright

(c) 2015-2021 Microchip Technology Inc. and its subsidiaries.

24.35.2 Function Documentation

24.35.2.1 wpccert_read_cert()

```
ATCA_STATUS wpccert_read_cert (
    ATCADevice device,
    const atcacert_def_t * cert_def,
    uint8_t * cert,
    size_t * cert_size )
```

WPC API -.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.35.2.2 wpccert_read_mfg_cert()

```
ATCA_STATUS wpccert_read_mfg_cert (
    ATCADevice device,
    uint8_t * cert,
    size_t * cert_size,
    uint8_t slot )
```

CA or TA device MFG Cert

CA2 device MFG Cert

24.36 atca_basic.c File Reference

CryptoAuthLib Basic API methods. These methods provide a simpler way to access the core crypto methods.

```
#include "atca_basic.h"
#include "atca_version.h"
```

Functions

- ATCA_STATUS [atcab_version](#) (char *ver_str)
basic API methods are all prefixed with atcab_ (CryptoAuthLib Basic) the fundamental premise of the basic API is it is based on a single interface instance and that instance is global, so all basic API commands assume that one global device is the one to operate on.
- ATCA_STATUS [atcab_init_ext](#) (ATCADevice *device, ATCAIfaceCfg *cfg)
Creates and initializes a ATCADevice context.
- ATCA_STATUS [atcab_init](#) (ATCAIfaceCfg *cfg)
Creates a global ATCADevice object used by Basic API.
- ATCA_STATUS [atcab_init_device](#) (ATCADevice ca_device)
Initialize the global ATCADevice object to point to one of your choosing for use with all the atcab_ basic API.
- ATCA_STATUS [atcab_release_ext](#) (ATCADevice *device)
release (free) the an ATCADevice instance.

- ATCA_STATUS [atcab_release](#) (void)
release (free) the global ATCADevice instance. This must be called in order to release or free up the interface.
- ATCADevice [atcab_get_device](#) (void)
Get the global device object.
- ATCADeviceType [atcab_get_device_type_ext](#) (ATCADevice device)
Get the selected device type of the device context.
- ATCADeviceType [atcab_get_device_type](#) (void)
Get the current device type configured for the global ATCADevice.
- uint8_t [atcab_get_device_address](#) (ATCADevice device)
Get the current device address based on the configured device and interface.
- bool [atcab_is_ca_device](#) (ATCADeviceType dev_type)
Check whether the device is cryptoauth device.
- bool [atcab_is_ca2_device](#) (ATCADeviceType dev_type)
Check whether the device is cryptoauth device.
- bool [atcab_is_ta_device](#) (ATCADeviceType dev_type)
Check whether the device is Trust Anchor device.
- ATCA_STATUS [atcab_wakeup](#) (void)
wakeup the CryptoAuth device
- ATCA_STATUS [atcab_idle](#) (void)
idle the CryptoAuth device
- ATCA_STATUS [atcab_sleep](#) (void)
invoke sleep on the CryptoAuth device
- ATCA_STATUS [atcab_get_zone_size_ext](#) (ATCADevice device, uint8_t zone, uint16_t slot, size_t *size)
Gets the size of the specified zone in bytes.
- ATCA_STATUS [atcab_get_zone_size](#) (uint8_t zone, uint16_t slot, size_t *size)
Gets the size of the specified zone in bytes.
- ATCA_STATUS [atcab_aes](#) (uint8_t mode, uint16_t key_id, const uint8_t *aes_in, uint8_t *aes_out)
Compute the AES-128 encrypt, decrypt, or GFM calculation.
- ATCA_STATUS [atcab_aes_encrypt_ext](#) (ATCADevice device, uint16_t key_id, uint8_t key_block, const uint8_t *plaintext, uint8_t *ciphertext)
Perform an AES-128 encrypt operation with a key in the device.
- ATCA_STATUS [atcab_aes_encrypt](#) (uint16_t key_id, uint8_t key_block, const uint8_t *plaintext, uint8_t *ciphertext)
Perform an AES-128 encrypt operation with a key in the device.
- ATCA_STATUS [atcab_aes_decrypt_ext](#) (ATCADevice device, uint16_t key_id, uint8_t key_block, const uint8_t *ciphertext, uint8_t *plaintext)
Perform an AES-128 decrypt operation with a key in the device.
- ATCA_STATUS [atcab_aes_decrypt](#) (uint16_t key_id, uint8_t key_block, const uint8_t *ciphertext, uint8_t *plaintext)
Perform an AES-128 decrypt operation with a key in the device.
- ATCA_STATUS [atcab_aes_gfm](#) (const uint8_t *h, const uint8_t *input, uint8_t *output)
Perform a Galois Field Multiply (GFM) operation.
- ATCA_STATUS [atcab_aes_gcm_init_ext](#) (ATCADevice device, atca_aes_gcm_ctx_t *ctx, uint16_t key_id, uint8_t key_block, const uint8_t *iv, size_t iv_size)
Initialize context for AES GCM operation with an existing IV, which is common when starting a decrypt operation.
- ATCA_STATUS [atcab_aes_gcm_init](#) (atca_aes_gcm_ctx_t *ctx, uint16_t key_id, uint8_t key_block, const uint8_t *iv, size_t iv_size)
Initialize context for AES GCM operation with an existing IV, which is common when starting a decrypt operation.
- ATCA_STATUS [atcab_aes_gcm_init_rand](#) (atca_aes_gcm_ctx_t *ctx, uint16_t key_id, uint8_t key_block, size_t rand_size, const uint8_t *free_field, size_t free_field_size, uint8_t *iv)
Initialize context for AES GCM operation with a IV composed of a random and optional fixed(free) field, which is common when starting an encrypt operation.

- ATCA_STATUS [atcab_aes_gcm_aad_update_ext](#) (ATCADevice device, atca_aes_gcm_ctx_t *ctx, const uint8_t *aad, uint32_t aad_size)
Process Additional Authenticated Data (AAD) using GCM mode and a key within the ATECC608 device.
- ATCA_STATUS [atcab_aes_gcm_aad_update](#) (atca_aes_gcm_ctx_t *ctx, const uint8_t *aad, uint32_t aad_size)
Process Additional Authenticated Data (AAD) using GCM mode and a key within the ATECC608 device.
- ATCA_STATUS [atcab_aes_gcm_encrypt_update_ext](#) (ATCADevice device, atca_aes_gcm_ctx_t *ctx, const uint8_t *plaintext, uint32_t plaintext_size, uint8_t *ciphertext)
Encrypt data using GCM mode and a key within the ATECC608 device. [atcab_aes_gcm_init\(\)](#) or [atcab_aes_gcm_init_rand\(\)](#) should be called before the first use of this function.
- ATCA_STATUS [atcab_aes_gcm_encrypt_update](#) (atca_aes_gcm_ctx_t *ctx, const uint8_t *plaintext, uint32_t plaintext_size, uint8_t *ciphertext)
Encrypt data using GCM mode and a key within the ATECC608 device. [atcab_aes_gcm_init\(\)](#) or [atcab_aes_gcm_init_rand\(\)](#) should be called before the first use of this function.
- ATCA_STATUS [atcab_aes_gcm_encrypt_finish_ext](#) (ATCADevice device, atca_aes_gcm_ctx_t *ctx, uint8_t *tag, size_t tag_size)
Complete a GCM encrypt operation returning the authentication tag.
- ATCA_STATUS [atcab_aes_gcm_encrypt_finish](#) (atca_aes_gcm_ctx_t *ctx, uint8_t *tag, size_t tag_size)
Complete a GCM encrypt operation returning the authentication tag.
- ATCA_STATUS [atcab_aes_gcm_decrypt_update_ext](#) (ATCADevice device, atca_aes_gcm_ctx_t *ctx, const uint8_t *ciphertext, uint32_t ciphertext_size, uint8_t *plaintext)
Decrypt data using GCM mode and a key within the ATECC608 device. [atcab_aes_gcm_init\(\)](#) or [atcab_aes_gcm_init_rand\(\)](#) should be called before the first use of this function.
- ATCA_STATUS [atcab_aes_gcm_decrypt_update](#) (atca_aes_gcm_ctx_t *ctx, const uint8_t *ciphertext, uint32_t ciphertext_size, uint8_t *plaintext)
Decrypt data using GCM mode and a key within the ATECC608 device. [atcab_aes_gcm_init\(\)](#) or [atcab_aes_gcm_init_rand\(\)](#) should be called before the first use of this function.
- ATCA_STATUS [atcab_aes_gcm_decrypt_finish_ext](#) (ATCADevice device, atca_aes_gcm_ctx_t *ctx, const uint8_t *tag, size_t tag_size, bool *is_verified)
Complete a GCM decrypt operation verifying the authentication tag.
- ATCA_STATUS [atcab_aes_gcm_decrypt_finish](#) (atca_aes_gcm_ctx_t *ctx, const uint8_t *tag, size_t tag_size, bool *is_verified)
Complete a GCM decrypt operation verifying the authentication tag.
- ATCA_STATUS [atcab_checkmac](#) (uint8_t mode, uint16_t key_id, const uint8_t *challenge, const uint8_t *response, const uint8_t *other_data)
Compares a MAC response with input values.
- ATCA_STATUS [atcab_checkmac_with_response_mac](#) (uint8_t mode, const uint8_t *challenge, const uint8_t *response, const uint8_t *other_data, uint8_t *mac)
Compares a MAC response with input values. SHA105 device can generate optional mac Output response mac mode only supports in SHA105 device.
- ATCA_STATUS [atcab_counter](#) (uint8_t mode, uint16_t counter_id, uint32_t *counter_value)
Compute the Counter functions.
- ATCA_STATUS [atcab_counter_increment](#) (uint16_t counter_id, uint32_t *counter_value)
Increments one of the device's monotonic counters.
- ATCA_STATUS [atcab_counter_read](#) (uint16_t counter_id, uint32_t *counter_value)
Read one of the device's monotonic counters.
- ATCA_STATUS [atcab_derivekey_ext](#) (ATCADevice device, uint8_t mode, uint16_t key_id, const uint8_t *mac)
Executes the DeviveKey command for deriving a new key from a nonce (TempKey) and an existing key.
- ATCA_STATUS [atcab_derivekey](#) (uint8_t mode, uint16_t key_id, const uint8_t *mac)
Executes the DeviveKey command for deriving a new key from a nonce (TempKey) and an existing key.
- ATCA_STATUS [atcab_ecdh_base](#) (uint8_t mode, uint16_t key_id, const uint8_t *public_key, uint8_t *pms, uint8_t *out_nonce)
Base function for generating premaster secret key using ECDH.

- ATCA_STATUS [atcab_ecdh](#) (uint16_t key_id, const uint8_t *public_key, uint8_t *pms)
ECDH command with a private key in a slot and the premaster secret is returned in the clear.
- ATCA_STATUS [atcab_ecdh_enc](#) (uint16_t key_id, const uint8_t *public_key, uint8_t *pms, const uint8_t *read_key, uint16_t read_key_id, const uint8_t num_in[(20)])
ECDH command with a private key in a slot and the premaster secret is read from the next slot.
- ATCA_STATUS [atcab_ecdh_ioenc](#) (uint16_t key_id, const uint8_t *public_key, uint8_t *pms, const uint8_t *io_key)
ECDH command with a private key in a slot and the premaster secret is returned encrypted using the IO protection key.
- ATCA_STATUS [atcab_ecdh_tempkey](#) (const uint8_t *public_key, uint8_t *pms)
ECDH command with a private key in TempKey and the premaster secret is returned in the clear.
- ATCA_STATUS [atcab_ecdh_tempkey_ioenc](#) (const uint8_t *public_key, uint8_t *pms, const uint8_t *io_key)
ECDH command with a private key in TempKey and the premaster secret is returned encrypted using the IO protection key.
- ATCA_STATUS [atcab_gendig](#) (uint8_t zone, uint16_t key_id, const uint8_t *other_data, uint8_t other_data_size)
Issues a GenDig command, which performs a SHA256 hash on the source data indicated by zone with the contents of TempKey. See the CryptoAuth datasheet for your chip to see what the values of zone correspond to.
- ATCA_STATUS [atcab_gendivkey](#) (const uint8_t *other_data)
Issues a GenDivKey command to generate the equivalent diversified key as that programmed into the client side device.
- ATCA_STATUS [atcab_genkey_base](#) (uint8_t mode, uint16_t key_id, const uint8_t *other_data, uint8_t *public_key)
Issues GenKey command, which can generate a private key, compute a public key, nd/or compute a digest of a public key.
- ATCA_STATUS [atcab_genkey_ext](#) (ATCADevice device, uint16_t key_id, uint8_t *public_key)
Issues GenKey command, which generates a new random private key in slot/handle and returns the public key.
- ATCA_STATUS [atcab_genkey](#) (uint16_t key_id, uint8_t *public_key)
Issues GenKey command, which generates a new random private key in slot/handle and returns the public key.
- ATCA_STATUS [atcab_get_pubkey_ext](#) (ATCADevice device, uint16_t key_id, uint8_t *public_key)
Uses GenKey command to calculate the public key from an existing private key in a slot.
- ATCA_STATUS [atcab_get_pubkey](#) (uint16_t key_id, uint8_t *public_key)
Uses GenKey command to calculate the public key from an existing private key in a slot.
- ATCA_STATUS [atcab_hmac](#) (uint8_t mode, uint16_t key_id, uint8_t *digest)
Issues a HMAC command, which computes an HMAC/SHA-256 digest of a key stored in the device, a challenge, and other information on the device.
- ATCA_STATUS [atcab_info_base](#) (uint8_t mode, uint16_t param2, uint8_t *out_data)
Issues an Info command, which return internal device information and can control GPIO and the persistent latch.
- ATCA_STATUS [atcab_info_ext](#) (ATCADevice device, uint8_t *revision)
Use the Info command to get the device revision (DevRev).
- ATCA_STATUS [atcab_info](#) (uint8_t *revision)
Use the Info command to get the device revision (DevRev).
- ATCA_STATUS [atcab_info_lock_status](#) (uint16_t param2, uint8_t *is_locked)
Use the Info command to get the lock status.
- ATCA_STATUS [atcab_info_chip_status](#) (uint8_t *chip_status)
Use the Info command to get the chip status.
- ATCA_STATUS [atcab_info_set_latch](#) (bool state)
Use the Info command to set the persistent latch state for an ATECC608 device.
- ATCA_STATUS [atcab_info_get_latch](#) (bool *state)
Use the Info command to get the persistent latch current state for an ATECC608 device.
- ATCA_STATUS [atcab_kdf](#) (uint8_t mode, uint16_t key_id, const uint32_t details, const uint8_t *message, uint8_t *out_data, uint8_t *out_nonce)

- Executes the KDF command, which derives a new key in PRF, AES, or HKDF modes.*

 - ATCA_STATUS [atcab_lock](#) (uint8_t mode, uint16_t summary_crc)

The Lock command prevents future modifications of the Configuration and/or Data and OTP zones. If the device is so configured, then this command can be used to lock individual data slots. This command fails if the designated area is already locked.
 - ATCA_STATUS [atcab_lock_config_zone_ext](#) (ATCADevice device)

Unconditionally (no CRC required) lock the config zone.
 - ATCA_STATUS [atcab_lock_config_zone](#) (void)

Unconditionally (no CRC required) lock the config zone.
 - ATCA_STATUS [atcab_lock_config_zone_crc](#) (uint16_t summary_crc)

Lock the config zone with summary CRC.
 - ATCA_STATUS [atcab_lock_data_zone_ext](#) (ATCADevice device)

Unconditionally (no CRC required) lock the data zone (slots and OTP). for CryptoAuth devices and lock the setup for Trust Anchor device.
 - ATCA_STATUS [atcab_lock_data_zone](#) (void)

Unconditionally (no CRC required) lock the data zone (slots and OTP). for CryptoAuth devices and lock the setup for Trust Anchor device.
 - ATCA_STATUS [atcab_lock_data_zone_crc](#) (uint16_t summary_crc)

Lock the data zone (slots and OTP) with summary CRC.
 - ATCA_STATUS [atcab_lock_data_slot_ext](#) (ATCADevice device, uint16_t slot)

Lock an individual slot in the data zone on an ATECC device. Not available for ATSHA devices. Slot must be configured to be slot lockable (KeyConfig.Lockable=1) (for cryptoauth devices) or Lock an individual handle in shared data element on an Trust Anchor device (for Trust Anchor devices).
 - ATCA_STATUS [atcab_lock_data_slot](#) (uint16_t slot)

Lock an individual slot in the data zone on an ATECC device. Not available for ATSHA devices. Slot must be configured to be slot lockable (KeyConfig.Lockable=1) (for cryptoauth devices) or Lock an individual handle in shared data element on an Trust Anchor device (for Trust Anchor devices).
 - ATCA_STATUS [atcab_mac](#) (uint8_t mode, uint16_t key_id, const uint8_t *challenge, uint8_t *digest)

Executes MAC command, which computes a SHA-256 digest of a key stored in the device, a challenge, and other information on the device.
 - ATCA_STATUS [atcab_nonce_base](#) (uint8_t mode, uint16_t zero, const uint8_t *num_in, uint8_t *rand_out)

Executes Nonce command, which loads a random or fixed nonce/data into the device for use by subsequent commands.
 - ATCA_STATUS [atcab_nonce](#) (const uint8_t *num_in)

Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.
 - ATCA_STATUS [atcab_nonce_load](#) (uint8_t target, const uint8_t *num_in, uint16_t num_in_size)

Execute a Nonce command in pass-through mode to load one of the device's internal buffers with a fixed value.
 - ATCA_STATUS [atcab_nonce_rand_ext](#) (ATCADevice device, const uint8_t *num_in, uint8_t *rand_out)

Execute a Nonce command to generate a random nonce combining a host nonce (num_in) and a device random number.
 - ATCA_STATUS [atcab_nonce_rand](#) (const uint8_t *num_in, uint8_t *rand_out)

Execute a Nonce command to generate a random nonce combining a host nonce (num_in) and a device random number.
 - ATCA_STATUS [atcab_challenge](#) (const uint8_t *num_in)

Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.
 - ATCA_STATUS [atcab_challenge_seed_update](#) (const uint8_t *num_in, uint8_t *rand_out)

Execute a Nonce command to generate a random challenge combining a host nonce (num_in) and a device random number.
 - ATCA_STATUS [atcab_priv_write](#) (uint16_t key_id, const uint8_t priv_key[36], uint16_t write_key_id, const uint8_t write_key[32], const uint8_t num_in[(20)])

Executes PrivWrite command, to write externally generated ECC private keys into the device.
 - ATCA_STATUS [atcab_random_ext](#) (ATCADevice device, uint8_t *rand_out)

Executes Random command, which generates a 32 byte random number from the device.

- ATCA_STATUS [atcab_random](#) (uint8_t *rand_out)
Executes Random command, which generates a 32 byte random number from the device.
- ATCA_STATUS [atcab_read_zone_ext](#) (ATCADevice device, uint8_t zone, uint16_t slot, uint8_t block, uint8_t offset, uint8_t *data, uint8_t len)
Executes Read command, which reads either 4 or 32 bytes of data from a given slot, configuration zone, or the OTP zone.
- ATCA_STATUS [atcab_read_zone](#) (uint8_t zone, uint16_t slot, uint8_t block, uint8_t offset, uint8_t *data, uint8_t len)
Executes Read command, which reads either 4 or 32 bytes of data from a given slot, configuration zone, or the OTP zone.
- ATCA_STATUS [atcab_is_locked](#) (uint8_t zone, bool *is_locked)
Executes Read command, which reads the configuration zone to see if the specified zone is locked.
- ATCA_STATUS [atcab_is_config_locked_ext](#) (ATCADevice device, bool *is_locked)
This function check whether configuration zone is locked or not.
- ATCA_STATUS [atcab_is_config_locked](#) (bool *is_locked)
This function check whether configuration zone is locked or not.
- ATCA_STATUS [atcab_is_data_locked_ext](#) (ATCADevice device, bool *is_locked)
This function check whether data/setup zone is locked or not.
- ATCA_STATUS [atcab_is_data_locked](#) (bool *is_locked)
This function check whether data/setup zone is locked or not.
- ATCA_STATUS [atcab_is_slot_locked_ext](#) (ATCADevice device, uint16_t slot, bool *is_locked)
This function check whether slot/handle is locked or not.
- ATCA_STATUS [atcab_is_slot_locked](#) (uint16_t slot, bool *is_locked)
This function check whether slot/handle is locked or not.
- ATCA_STATUS [atcab_is_private_ext](#) (ATCADevice device, uint16_t slot, bool *is_private)
Check to see if the key is a private key or not.
- ATCA_STATUS [atcab_is_private](#) (uint16_t slot, bool *is_private)
- ATCA_STATUS [atcab_read_bytes_zone_ext](#) (ATCADevice device, uint8_t zone, uint16_t slot, size_t offset, uint8_t *data, size_t length)
- ATCA_STATUS [atcab_read_bytes_zone](#) (uint8_t zone, uint16_t slot, size_t offset, uint8_t *data, size_t length)
Used to read an arbitrary number of bytes from any zone configured for clear reads.
- ATCA_STATUS [atcab_read_serial_number_ext](#) (ATCADevice device, uint8_t *serial_number)
This function returns serial number of the device.
- ATCA_STATUS [atcab_read_serial_number](#) (uint8_t *serial_number)
This function returns serial number of the device.
- ATCA_STATUS [atcab_read_pubkey_ext](#) (ATCADevice device, uint16_t slot, uint8_t *public_key)
Executes Read command to read an ECC P256 public key from a slot configured for clear reads.
- ATCA_STATUS [atcab_read_pubkey](#) (uint16_t slot, uint8_t *public_key)
Executes Read command to read an ECC P256 public key from a slot configured for clear reads.
- ATCA_STATUS [atcab_read_sig](#) (uint16_t slot, uint8_t *sig)
Executes Read command to read a 64 byte ECDSA P256 signature from a slot configured for clear reads.
- ATCA_STATUS [atcab_read_config_zone_ext](#) (ATCADevice device, uint8_t *config_data)
Executes Read command to read the complete device configuration zone.
- ATCA_STATUS [atcab_read_config_zone](#) (uint8_t *config_data)
Executes Read command to read the complete device configuration zone.
- ATCA_STATUS [atcab_cmp_config_zone](#) (uint8_t *config_data, bool *same_config)
Compares a specified configuration zone with the configuration zone currently on the device.
- ATCA_STATUS [atcab_read_enc](#) (uint16_t key_id, uint8_t block, uint8_t *data, const uint8_t *enc_key, const uint16_t enc_key_id, const uint8_t num_in[(20)])
Executes Read command on a slot configured for encrypted reads and decrypts the data to return it as plaintext.
- ATCA_STATUS [atcab_secureboot](#) (uint8_t mode, uint16_t param2, const uint8_t *digest, const uint8_t *signature, uint8_t *mac)

- Executes Secure Boot command, which provides support for secure boot of an external MCU or MPU.*
- ATCA_STATUS [atcab_secureboot_mac](#) (uint8_t mode, const uint8_t *digest, const uint8_t *signature, const uint8_t *num_in, const uint8_t *io_key, bool *is_verified)
Executes Secure Boot command with encrypted digest and validated MAC response using the IO protection key.
 - ATCA_STATUS [atcab_selftest](#) (uint8_t mode, uint16_t param2, uint8_t *result)
Executes the SelfTest command, which performs a test of one or more of the cryptographic engines within the ATECC608 chip.
 - ATCA_STATUS [atcab_sha_base](#) (uint8_t mode, uint16_t length, const uint8_t *data_in, uint8_t *data_out, uint16_t *data_out_size)
Executes SHA command, which computes a SHA-256 or HMAC/SHA-256 digest for general purpose use by the host system.
 - ATCA_STATUS [atcab_sha_start](#) (void)
Executes SHA command to initialize SHA-256 calculation engine.
 - ATCA_STATUS [atcab_sha_update](#) (const uint8_t *message)
Executes SHA command to add 64 bytes of message data to the current context.
 - ATCA_STATUS [atcab_sha_end](#) (uint8_t *digest, uint16_t length, const uint8_t *message)
Executes SHA command to complete SHA-256 or HMAC/SHA-256 operation.
 - ATCA_STATUS [atcab_sha_read_context](#) (uint8_t *context, uint16_t *context_size)
Executes SHA command to read the SHA-256 context back. Only for ATECC608 with SHA-256 contexts. HMAC not supported.
 - ATCA_STATUS [atcab_sha_write_context](#) (const uint8_t *context, uint16_t context_size)
Executes SHA command to write (restore) a SHA-256 context into the the device. Only supported for ATECC608 with SHA-256 contexts.
 - ATCA_STATUS [atcab_sha](#) (uint16_t length, const uint8_t *message, uint8_t *digest)
Use the SHA command to compute a SHA-256 digest.
 - ATCA_STATUS [atcab_hw_sha2_256](#) (const uint8_t *data, size_t data_size, uint8_t *digest)
Use the SHA command to compute a SHA-256 digest.
 - ATCA_STATUS [atcab_hw_sha2_256_init](#) (atca_sha256_ctx_t *ctx)
Initialize a SHA context for performing a hardware SHA-256 operation on a device. Note that only one SHA operation can be run at a time.
 - ATCA_STATUS [atcab_hw_sha2_256_update](#) (atca_sha256_ctx_t *ctx, const uint8_t *data, size_t data_size)
Add message data to a SHA context for performing a hardware SHA-256 operation on a device.
 - ATCA_STATUS [atcab_hw_sha2_256_finish](#) (atca_sha256_ctx_t *ctx, uint8_t *digest)
Finish SHA-256 digest for a SHA context for performing a hardware SHA-256 operation on a device.
 - ATCA_STATUS [atcab_sha_hmac_init](#) (atca_hmac_sha256_ctx_t *ctx, uint16_t key_slot)
Executes SHA command to start an HMAC/SHA-256 operation.
 - ATCA_STATUS [atcab_sha_hmac_update](#) (atca_hmac_sha256_ctx_t *ctx, const uint8_t *data, size_t data_size)
Executes SHA command to add an arbitrary amount of message data to a HMAC/SHA-256 operation.
 - ATCA_STATUS [atcab_sha_hmac_finish](#) (atca_hmac_sha256_ctx_t *ctx, uint8_t *digest, uint8_t target)
Executes SHA command to complete a HMAC/SHA-256 operation.
 - ATCA_STATUS [atcab_sha_hmac_ext](#) (ATCADevice device, const uint8_t *data, size_t data_size, uint16_t key_slot, uint8_t *digest, uint8_t target)
Use the SHA command to compute an HMAC/SHA-256 operation.
 - ATCA_STATUS [atcab_sha_hmac](#) (const uint8_t *data, size_t data_size, uint16_t key_slot, uint8_t *digest, uint8_t target)
Use the SHA command to compute an HMAC/SHA-256 operation.
 - ATCA_STATUS [atcab_sign_base](#) (uint8_t mode, uint16_t key_id, uint8_t *signature)
Executes the Sign command, which generates a signature using the ECDSA algorithm.
 - ATCA_STATUS [atcab_sign_ext](#) (ATCADevice device, uint16_t key_id, const uint8_t *msg, uint8_t *signature)
Executes Sign command, to sign a 32-byte external message using the private key in the specified slot. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.
 - ATCA_STATUS [atcab_sign](#) (uint16_t key_id, const uint8_t *msg, uint8_t *signature)

- Executes Sign command, to sign a 32-byte external message using the private key in the specified slot. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.*
- ATCA_STATUS [atcab_sign_internal](#) (uint16_t key_id, bool is_invalidate, bool is_full_sn, uint8_t *signature)
Executes Sign command to sign an internally generated message.
 - ATCA_STATUS [atcab_updateextra](#) (uint8_t mode, uint16_t new_value)
Executes UpdateExtra command to update the values of the two extra bytes within the Configuration zone (bytes 84 and 85).
 - ATCA_STATUS [atcab_verify](#) (uint8_t mode, uint16_t key_id, const uint8_t *signature, const uint8_t *public_key, const uint8_t *other_data, uint8_t *mac)
Executes the Verify command, which takes an ECDSA [R,S] signature and verifies that it is correctly generated from a given message and public key. In all cases, the signature is an input to the command.
 - ATCA_STATUS [atcab_verify_extern_ext](#) (ATCADevice device, const uint8_t *message, const uint8_t *signature, const uint8_t *public_key, bool *is_verified)
Executes the Verify command, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.
 - ATCA_STATUS [atcab_verify_extern](#) (const uint8_t *message, const uint8_t *signature, const uint8_t *public_key, bool *is_verified)
Executes the Verify command, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.
 - ATCA_STATUS [atcab_verify_extern_mac](#) (const uint8_t *message, const uint8_t *signature, const uint8_t *public_key, const uint8_t *num_in, const uint8_t *io_key, bool *is_verified)
Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. This function is only available on the ATECC608.
 - ATCA_STATUS [atcab_verify_stored_ext](#) (ATCADevice device, const uint8_t *message, const uint8_t *signature, uint16_t key_id, bool *is_verified)
Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.
 - ATCA_STATUS [atcab_verify_stored](#) (const uint8_t *message, const uint8_t *signature, uint16_t key_id, bool *is_verified)
Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.
 - ATCA_STATUS [atcab_verify_stored_with_tempkey](#) (const uint8_t *signature, uint16_t key_id, bool *is_verified)
Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. keyConfig.reqrndom bit should be set and the message to be signed should be already loaded into TempKey for all devices.
 - ATCA_STATUS [atcab_verify_stored_mac](#) (const uint8_t *message, const uint8_t *signature, uint16_t key_id, const uint8_t *num_in, const uint8_t *io_key, bool *is_verified)
Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with a public key stored in the device. This function is only available on the ATECC608.
 - ATCA_STATUS [atcab_verify_validate](#) (uint16_t key_id, const uint8_t *signature, const uint8_t *other_data, bool *is_verified)
Executes the Verify command in Validate mode to validate a public key stored in a slot.
 - ATCA_STATUS [atcab_verify_invalidate](#) (uint16_t key_id, const uint8_t *signature, const uint8_t *other_data, bool *is_verified)
Executes the Verify command in Invalidate mode which invalidates a previously validated public key stored in a slot.
 - ATCA_STATUS [atcab_write](#) (uint8_t zone, uint16_t address, const uint8_t *value, const uint8_t *mac)
Executes the Write command, which writes either one four byte word or a 32-byte block to one of the EEPROM zones on the device. Depending upon the value of the WriteConfig byte for this slot, the data may be required to be encrypted by the system prior to being sent to the device. This command cannot be used to write slots configured as ECC private keys.

- ATCA_STATUS [atcab_write_zone_ext](#) (ATCADevice device, uint8_t zone, uint16_t slot, uint8_t block, uint8_t offset, const uint8_t *data, uint8_t len)
Executes the Write command, which writes either 4 or 32 bytes of data into a device zone.
- ATCA_STATUS [atcab_write_zone](#) (uint8_t zone, uint16_t slot, uint8_t block, uint8_t offset, const uint8_t *data, uint8_t len)
Executes the Write command, which writes either 4 or 32 bytes of data into a device zone.
- ATCA_STATUS [atcab_write_bytes_zone_ext](#) (ATCADevice device, uint8_t zone, uint16_t slot, size_t offset_bytes, const uint8_t *data, size_t length)
Executes the Write command, which writes data into the configuration, otp, or data zones with a given byte offset and length. Offset and length must be multiples of a word (4 bytes).
- ATCA_STATUS [atcab_write_bytes_zone](#) (uint8_t zone, uint16_t slot, size_t offset_bytes, const uint8_t *data, size_t length)
Executes the Write command, which writes data into the configuration, otp, or data zones with a given byte offset and length. Offset and length must be multiples of a word (4 bytes).
- ATCA_STATUS [atcab_write_pubkey_ext](#) (ATCADevice device, uint16_t slot, const uint8_t *public_key)
Uses the write command to write a public key to a slot in the proper format.
- ATCA_STATUS [atcab_write_pubkey](#) (uint16_t slot, const uint8_t *public_key)
Uses the write command to write a public key to a slot in the proper format.
- ATCA_STATUS [atcab_write_config_zone_ext](#) (ATCADevice device, const uint8_t *config_data)
Executes the Write command, which writes the configuration zone.
- ATCA_STATUS [atcab_write_config_zone](#) (const uint8_t *config_data)
Executes the Write command, which writes the configuration zone.
- ATCA_STATUS [atcab_write_enc](#) (uint16_t key_id, uint8_t block, const uint8_t *data, const uint8_t *enc_key, const uint16_t enc_key_id, const uint8_t num_in[(20)])
Executes the Write command, which performs an encrypted write of a 32 byte block into given slot.
- ATCA_STATUS [atcab_write_config_counter](#) (uint16_t counter_id, uint32_t counter_value)
Initialize one of the monotonic counters in device with a specific value.

Variables

- ATCADevice [g_atcab_device_ptr](#) = NULL

24.36.1 Detailed Description

CryptoAuthLib Basic API methods. These methods provide a simpler way to access the core crypto methods.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.37 atca_basic.h File Reference

CryptoAuthLib Basic API methods - a simple crypto authentication API. These methods manage a global ATCADevice object behind the scenes. They also manage the wake/idle state transitions so callers don't need to.

```
#include "cryptoauthlib.h"
#include "crypto/atca_crypto_sw_sha2.h"
#include "crypto/atca_crypto_hw_aes.h"
```

Macros

- #define **atcab_get_addr**(...) **calib_get_addr**(__VA_ARGS__)
- #define **atca_execute_command**(...) **calib_execute_command**(__VA_ARGS__)
- #define **SHA_CONTEXT_MAX_SIZE** (109)

Functions

- ATCA_STATUS **atcab_version** (char *ver_str)
basic API methods are all prefixed with atcab_ (CryptoAuthLib Basic) the fundamental premise of the basic API is it is based on a single interface instance and that instance is global, so all basic API commands assume that one global device is the one to operate on.
- ATCA_STATUS **atcab_init_ext** (ATCADevice *device, ATCAIfaceCfg *cfg)
Creates and initializes a ATCADevice context.
- ATCA_STATUS **atcab_init** (ATCAIfaceCfg *cfg)
Creates a global ATCADevice object used by Basic API.
- ATCA_STATUS **atcab_init_device** (ATCADevice ca_device)
Initialize the global ATCADevice object to point to one of your choosing for use with all the atcab_ basic API.
- ATCA_STATUS **atcab_release_ext** (ATCADevice *device)
release (free) the an ATCADevice instance.
- ATCA_STATUS **atcab_release** (void)
release (free) the global ATCADevice instance. This must be called in order to release or free up the interface.
- ATCADevice **atcab_get_device** (void)
Get the global device object.
- ATCADeviceType **atcab_get_device_type_ext** (ATCADevice device)
Get the selected device type of rthe device context.
- ATCADeviceType **atcab_get_device_type** (void)
Get the current device type configured for the global ATCADevice.
- uint8_t **atcab_get_device_address** (ATCADevice device)
Get the current device address based on the configured device and interface.
- bool **atcab_is_ca_device** (ATCADeviceType dev_type)
Check whether the device is cryptoauth device.
- bool **atcab_is_ca2_device** (ATCADeviceType dev_type)
Check whether the device is cryptoauth device.
- bool **atcab_is_ta_device** (ATCADeviceType dev_type)
Check whether the device is Trust Anchor device.
- ATCA_STATUS **atcab_wakeup** (void)
wakeup the CryptoAuth device
- ATCA_STATUS **atcab_idle** (void)
idle the CryptoAuth device
- ATCA_STATUS **atcab_sleep** (void)
invoke sleep on the CryptoAuth device
- ATCA_STATUS **atcab_get_zone_size** (uint8_t zone, uint16_t slot, size_t *size)
Gets the size of the specified zone in bytes.
- ATCA_STATUS **atcab_get_zone_size_ext** (ATCADevice device, uint8_t zone, uint16_t slot, size_t *size)
Gets the size of the specified zone in bytes.
- ATCA_STATUS **atcab_aes** (uint8_t mode, uint16_t key_id, const uint8_t *aes_in, uint8_t *aes_out)
Compute the AES-128 encrypt, decrypt, or GFM calculation.
- ATCA_STATUS **atcab_aes_encrypt** (uint16_t key_id, uint8_t key_block, const uint8_t *plaintext, uint8_t *ciphertext)
Perform an AES-128 encrypt operation with a key in the device.

- ATCA_STATUS [atcab_aes_encrypt_ext](#) (ATCADevice device, uint16_t key_id, uint8_t key_block, const uint8_t *plaintext, uint8_t *ciphertext)
Perform an AES-128 encrypt operation with a key in the device.
- ATCA_STATUS [atcab_aes_decrypt](#) (uint16_t key_id, uint8_t key_block, const uint8_t *ciphertext, uint8_t *plaintext)
Perform an AES-128 decrypt operation with a key in the device.
- ATCA_STATUS [atcab_aes_decrypt_ext](#) (ATCADevice device, uint16_t key_id, uint8_t key_block, const uint8_t *ciphertext, uint8_t *plaintext)
Perform an AES-128 decrypt operation with a key in the device.
- ATCA_STATUS [atcab_aes_gfm](#) (const uint8_t *h, const uint8_t *input, uint8_t *output)
Perform a Galois Field Multiply (GFM) operation.
- ATCA_STATUS [atcab_aes_gcm_init](#) (atca_aes_gcm_ctx_t *ctx, uint16_t key_id, uint8_t key_block, const uint8_t *iv, size_t iv_size)
Initialize context for AES GCM operation with an existing IV, which is common when starting a decrypt operation.
- ATCA_STATUS [atcab_aes_gcm_init_ext](#) (ATCADevice device, atca_aes_gcm_ctx_t *ctx, uint16_t key_id, uint8_t key_block, const uint8_t *iv, size_t iv_size)
Initialize context for AES GCM operation with an existing IV, which is common when starting a decrypt operation.
- ATCA_STATUS [atcab_aes_gcm_init_rand](#) (atca_aes_gcm_ctx_t *ctx, uint16_t key_id, uint8_t key_block, size_t rand_size, const uint8_t *free_field, size_t free_field_size, uint8_t *iv)
Initialize context for AES GCM operation with a IV composed of a random and optional fixed(free) field, which is common when starting an encrypt operation.
- ATCA_STATUS [atcab_aes_gcm_aad_update](#) (atca_aes_gcm_ctx_t *ctx, const uint8_t *aad, uint32_t aad_size)
Process Additional Authenticated Data (AAD) using GCM mode and a key within the ATECC608 device.
- ATCA_STATUS [atcab_aes_gcm_aad_update_ext](#) (ATCADevice device, atca_aes_gcm_ctx_t *ctx, const uint8_t *aad, uint32_t aad_size)
Process Additional Authenticated Data (AAD) using GCM mode and a key within the ATECC608 device.
- ATCA_STATUS [atcab_aes_gcm_encrypt_update](#) (atca_aes_gcm_ctx_t *ctx, const uint8_t *plaintext, uint32_t plaintext_size, uint8_t *ciphertext)
Encrypt data using GCM mode and a key within the ATECC608 device. [atcab_aes_gcm_init\(\)](#) or [atcab_aes_gcm_init_rand\(\)](#) should be called before the first use of this function.
- ATCA_STATUS [atcab_aes_gcm_encrypt_update_ext](#) (ATCADevice device, atca_aes_gcm_ctx_t *ctx, const uint8_t *plaintext, uint32_t plaintext_size, uint8_t *ciphertext)
Encrypt data using GCM mode and a key within the ATECC608 device. [atcab_aes_gcm_init\(\)](#) or [atcab_aes_gcm_init_rand\(\)](#) should be called before the first use of this function.
- ATCA_STATUS [atcab_aes_gcm_encrypt_finish](#) (atca_aes_gcm_ctx_t *ctx, uint8_t *tag, size_t tag_size)
Complete a GCM encrypt operation returning the authentication tag.
- ATCA_STATUS [atcab_aes_gcm_encrypt_finish_ext](#) (ATCADevice device, atca_aes_gcm_ctx_t *ctx, uint8_t *tag, size_t tag_size)
Complete a GCM encrypt operation returning the authentication tag.
- ATCA_STATUS [atcab_aes_gcm_decrypt_update](#) (atca_aes_gcm_ctx_t *ctx, const uint8_t *ciphertext, uint32_t ciphertext_size, uint8_t *plaintext)
Decrypt data using GCM mode and a key within the ATECC608 device. [atcab_aes_gcm_init\(\)](#) or [atcab_aes_gcm_init_rand\(\)](#) should be called before the first use of this function.
- ATCA_STATUS [atcab_aes_gcm_decrypt_update_ext](#) (ATCADevice device, atca_aes_gcm_ctx_t *ctx, const uint8_t *ciphertext, uint32_t ciphertext_size, uint8_t *plaintext)
Decrypt data using GCM mode and a key within the ATECC608 device. [atcab_aes_gcm_init\(\)](#) or [atcab_aes_gcm_init_rand\(\)](#) should be called before the first use of this function.
- ATCA_STATUS [atcab_aes_gcm_decrypt_finish](#) (atca_aes_gcm_ctx_t *ctx, const uint8_t *tag, size_t tag_size, bool *is_verified)
Complete a GCM decrypt operation verifying the authentication tag.
- ATCA_STATUS [atcab_aes_gcm_decrypt_finish_ext](#) (ATCADevice device, atca_aes_gcm_ctx_t *ctx, const uint8_t *tag, size_t tag_size, bool *is_verified)
Complete a GCM decrypt operation verifying the authentication tag.

- ATCA_STATUS [atcab_checkmac](#) (uint8_t mode, uint16_t key_id, const uint8_t *challenge, const uint8_t *response, const uint8_t *other_data)
Compares a MAC response with input values.
- ATCA_STATUS [atcab_checkmac_with_response_mac](#) (uint8_t mode, const uint8_t *challenge, const uint8_t *response, const uint8_t *other_data, uint8_t *mac)
Compares a MAC response with input values. SHA105 device can generate optional mac Output response mac mode only supports in SHA105 device.
- ATCA_STATUS [atcab_counter](#) (uint8_t mode, uint16_t counter_id, uint32_t *counter_value)
Compute the Counter functions.
- ATCA_STATUS [atcab_counter_increment](#) (uint16_t counter_id, uint32_t *counter_value)
Increments one of the device's monotonic counters.
- ATCA_STATUS [atcab_counter_read](#) (uint16_t counter_id, uint32_t *counter_value)
Read one of the device's monotonic counters.
- ATCA_STATUS [atcab_derivekey](#) (uint8_t mode, uint16_t key_id, const uint8_t *mac)
Executes the DeviveKey command for deriving a new key from a nonce (TempKey) and an existing key.
- ATCA_STATUS [atcab_derivekey_ext](#) (ATCADevice device, uint8_t mode, uint16_t key_id, const uint8_t *mac)
Executes the DeviveKey command for deriving a new key from a nonce (TempKey) and an existing key.
- ATCA_STATUS [atcab_ecdh_base](#) (uint8_t mode, uint16_t key_id, const uint8_t *public_key, uint8_t *pms, uint8_t *out_nonce)
Base function for generating premaster secret key using ECDH.
- ATCA_STATUS [atcab_ecdh](#) (uint16_t key_id, const uint8_t *public_key, uint8_t *pms)
ECDH command with a private key in a slot and the premaster secret is returned in the clear.
- ATCA_STATUS [atcab_ecdh_enc](#) (uint16_t key_id, const uint8_t *public_key, uint8_t *pms, const uint8_t *read_key, uint16_t read_key_id, const uint8_t num_in[(20)])
ECDH command with a private key in a slot and the premaster secret is read from the next slot.
- ATCA_STATUS [atcab_ecdh_ioenc](#) (uint16_t key_id, const uint8_t *public_key, uint8_t *pms, const uint8_t *io_key)
ECDH command with a private key in a slot and the premaster secret is returned encrypted using the IO protection key.
- ATCA_STATUS [atcab_ecdh_tempkey](#) (const uint8_t *public_key, uint8_t *pms)
ECDH command with a private key in TempKey and the premaster secret is returned in the clear.
- ATCA_STATUS [atcab_ecdh_tempkey_ioenc](#) (const uint8_t *public_key, uint8_t *pms, const uint8_t *io_key)
ECDH command with a private key in TempKey and the premaster secret is returned encrypted using the IO protection key.
- ATCA_STATUS [atcab_gendig](#) (uint8_t zone, uint16_t key_id, const uint8_t *other_data, uint8_t other_data_size)
Issues a GenDig command, which performs a SHA256 hash on the source data indicated by zone with the contents of TempKey. See the CryptoAuth datasheet for your chip to see what the values of zone correspond to.
- ATCA_STATUS [atcab_gendivkey](#) (const uint8_t *other_data)
Issues a GenDivKey command to generate the equivalent diversified key as that programmed into the client side device.
- ATCA_STATUS [atcab_genkey_base](#) (uint8_t mode, uint16_t key_id, const uint8_t *other_data, uint8_t *public_key)
Issues GenKey command, which can generate a private key, compute a public key, nd/or compute a digest of a public key.
- ATCA_STATUS [atcab_genkey](#) (uint16_t key_id, uint8_t *public_key)
Issues GenKey command, which generates a new random private key in slot/handle and returns the public key.
- ATCA_STATUS [atcab_genkey_ext](#) (ATCADevice device, uint16_t key_id, uint8_t *public_key)
Issues GenKey command, which generates a new random private key in slot/handle and returns the public key.
- ATCA_STATUS [atcab_get_pubkey](#) (uint16_t key_id, uint8_t *public_key)
Uses GenKey command to calculate the public key from an existing private key in a slot.
- ATCA_STATUS [atcab_get_pubkey_ext](#) (ATCADevice device, uint16_t key_id, uint8_t *public_key)
Uses GenKey command to calculate the public key from an existing private key in a slot.

- ATCA_STATUS [atcab_hmac](#) (uint8_t mode, uint16_t key_id, uint8_t *digest)
Issues a HMAC command, which computes an HMAC/SHA-256 digest of a key stored in the device, a challenge, and other information on the device.
- ATCA_STATUS [atcab_info_base](#) (uint8_t mode, uint16_t param2, uint8_t *out_data)
Issues an Info command, which return internal device information and can control GPIO and the persistent latch.
- ATCA_STATUS [atcab_info](#) (uint8_t *revision)
Use the Info command to get the device revision (DevRev).
- ATCA_STATUS [atcab_info_ext](#) (ATCADevice device, uint8_t *revision)
Use the Info command to get the device revision (DevRev).
- ATCA_STATUS [atcab_info_lock_status](#) (uint16_t param2, uint8_t *is_locked)
Use the Info command to get the lock status.
- ATCA_STATUS [atcab_info_chip_status](#) (uint8_t *chip_status)
Use the Info command to get the chip status.
- ATCA_STATUS [atcab_info_set_latch](#) (bool state)
Use the Info command to set the persistent latch state for an ATECC608 device.
- ATCA_STATUS [atcab_info_get_latch](#) (bool *state)
Use the Info command to get the persistent latch current state for an ATECC608 device.
- ATCA_STATUS [atcab_kdf](#) (uint8_t mode, uint16_t key_id, const uint32_t details, const uint8_t *message, uint8_t *out_data, uint8_t *out_nonce)
Executes the KDF command, which derives a new key in PRF, AES, or HKDF modes.
- ATCA_STATUS [atcab_lock](#) (uint8_t mode, uint16_t summary_crc)
The Lock command prevents future modifications of the Configuration and/or Data and OTP zones. If the device is so configured, then this command can be used to lock individual data slots. This command fails if the designated area is already locked.
- ATCA_STATUS [atcab_lock_config_zone](#) (void)
Unconditionally (no CRC required) lock the config zone.
- ATCA_STATUS [atcab_lock_config_zone_ext](#) (ATCADevice device)
Unconditionally (no CRC required) lock the config zone.
- ATCA_STATUS [atcab_lock_config_zone_crc](#) (uint16_t summary_crc)
Lock the config zone with summary CRC.
- ATCA_STATUS [atcab_lock_data_zone](#) (void)
Unconditionally (no CRC required) lock the data zone (slots and OTP). for CryptoAuth devices and lock the setup for Trust Anchor device.
- ATCA_STATUS [atcab_lock_data_zone_ext](#) (ATCADevice device)
Unconditionally (no CRC required) lock the data zone (slots and OTP). for CryptoAuth devices and lock the setup for Trust Anchor device.
- ATCA_STATUS [atcab_lock_data_zone_crc](#) (uint16_t summary_crc)
Lock the data zone (slots and OTP) with summary CRC.
- ATCA_STATUS [atcab_lock_data_slot](#) (uint16_t slot)
Lock an individual slot in the data zone on an ATECC device. Not available for ATSHA devices. Slot must be configured to be slot lockable (KeyConfig.Lockable=1) (for cryptoauth devices) or Lock an individual handle in shared data element on an Trust Anchor device (for Trust Anchor devices).
- ATCA_STATUS [atcab_lock_data_slot_ext](#) (ATCADevice device, uint16_t slot)
Lock an individual slot in the data zone on an ATECC device. Not available for ATSHA devices. Slot must be configured to be slot lockable (KeyConfig.Lockable=1) (for cryptoauth devices) or Lock an individual handle in shared data element on an Trust Anchor device (for Trust Anchor devices).
- ATCA_STATUS [atcab_mac](#) (uint8_t mode, uint16_t key_id, const uint8_t *challenge, uint8_t *digest)
Executes MAC command, which computes a SHA-256 digest of a key stored in the device, a challenge, and other information on the device.
- ATCA_STATUS [atcab_nonce_base](#) (uint8_t mode, uint16_t zero, const uint8_t *num_in, uint8_t *rand_out)
Executes Nonce command, which loads a random or fixed nonce/data into the device for use by subsequent commands.
- ATCA_STATUS [atcab_nonce](#) (const uint8_t *num_in)

- Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.*
- ATCA_STATUS [atcab_nonce_load](#) (uint8_t target, const uint8_t *num_in, uint16_t num_in_size)
Execute a Nonce command in pass-through mode to load one of the device's internal buffers with a fixed value.
 - ATCA_STATUS [atcab_nonce_rand](#) (const uint8_t *num_in, uint8_t *rand_out)
Execute a Nonce command to generate a random nonce combining a host nonce (num_in) and a device random number.
 - ATCA_STATUS [atcab_nonce_rand_ext](#) (ATCADevice device, const uint8_t *num_in, uint8_t *rand_out)
Execute a Nonce command to generate a random nonce combining a host nonce (num_in) and a device random number.
 - ATCA_STATUS [atcab_challenge](#) (const uint8_t *num_in)
Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.
 - ATCA_STATUS [atcab_challenge_seed_update](#) (const uint8_t *num_in, uint8_t *rand_out)
Execute a Nonce command to generate a random challenge combining a host nonce (num_in) and a device random number.
 - ATCA_STATUS [atcab_priv_write](#) (uint16_t key_id, const uint8_t priv_key[36], uint16_t write_key_id, const uint8_t write_key[32], const uint8_t num_in[(20)])
Executes PrivWrite command, to write externally generated ECC private keys into the device.
 - ATCA_STATUS [atcab_random](#) (uint8_t *rand_out)
Executes Random command, which generates a 32 byte random number from the device.
 - ATCA_STATUS [atcab_random_ext](#) (ATCADevice device, uint8_t *rand_out)
Executes Random command, which generates a 32 byte random number from the device.
 - ATCA_STATUS [atcab_read_zone](#) (uint8_t zone, uint16_t slot, uint8_t block, uint8_t offset, uint8_t *data, uint8_t len)
Executes Read command, which reads either 4 or 32 bytes of data from a given slot, configuration zone, or the OTP zone.
 - ATCA_STATUS [atcab_read_zone_ext](#) (ATCADevice device, uint8_t zone, uint16_t slot, uint8_t block, uint8_t offset, uint8_t *data, uint8_t len)
Executes Read command, which reads either 4 or 32 bytes of data from a given slot, configuration zone, or the OTP zone.
 - ATCA_STATUS [atcab_is_locked](#) (uint8_t zone, bool *is_locked)
Executes Read command, which reads the configuration zone to see if the specified zone is locked.
 - ATCA_STATUS [atcab_is_config_locked](#) (bool *is_locked)
This function check whether configuration zone is locked or not.
 - ATCA_STATUS [atcab_is_config_locked_ext](#) (ATCADevice device, bool *is_locked)
This function check whether configuration zone is locked or not.
 - ATCA_STATUS [atcab_is_data_locked](#) (bool *is_locked)
This function check whether data/setup zone is locked or not.
 - ATCA_STATUS [atcab_is_data_locked_ext](#) (ATCADevice device, bool *is_locked)
This function check whether data/setup zone is locked or not.
 - ATCA_STATUS [atcab_is_slot_locked](#) (uint16_t slot, bool *is_locked)
This function check whether slot/handle is locked or not.
 - ATCA_STATUS [atcab_is_slot_locked_ext](#) (ATCADevice device, uint16_t slot, bool *is_locked)
This function check whether slot/handle is locked or not.
 - ATCA_STATUS [atcab_is_private_ext](#) (ATCADevice device, uint16_t slot, bool *is_private)
Check to see if the key is a private key or not.
 - ATCA_STATUS [atcab_is_private](#) (uint16_t slot, bool *is_private)
 - ATCA_STATUS [atcab_read_bytes_zone_ext](#) (ATCADevice device, uint8_t zone, uint16_t slot, size_t offset, uint8_t *data, size_t length)
 - ATCA_STATUS [atcab_read_bytes_zone](#) (uint8_t zone, uint16_t slot, size_t offset, uint8_t *data, size_t length)
Used to read an arbitrary number of bytes from any zone configured for clear reads.
 - ATCA_STATUS [atcab_read_serial_number](#) (uint8_t *serial_number)
This function returns serial number of the device.

- ATCA_STATUS [atcab_read_serial_number_ext](#) (ATCADevice device, uint8_t *serial_number)
This function returns serial number of the device.
- ATCA_STATUS [atcab_read_pubkey](#) (uint16_t slot, uint8_t *public_key)
Executes Read command to read an ECC P256 public key from a slot configured for clear reads.
- ATCA_STATUS [atcab_read_pubkey_ext](#) (ATCADevice device, uint16_t slot, uint8_t *public_key)
Executes Read command to read an ECC P256 public key from a slot configured for clear reads.
- ATCA_STATUS [atcab_read_sig](#) (uint16_t slot, uint8_t *sig)
Executes Read command to read a 64 byte ECDSA P256 signature from a slot configured for clear reads.
- ATCA_STATUS [atcab_read_config_zone](#) (uint8_t *config_data)
Executes Read command to read the complete device configuration zone.
- ATCA_STATUS [atcab_read_config_zone_ext](#) (ATCADevice device, uint8_t *config_data)
Executes Read command to read the complete device configuration zone.
- ATCA_STATUS [atcab_cmp_config_zone](#) (uint8_t *config_data, bool *same_config)
Compares a specified configuration zone with the configuration zone currently on the device.
- ATCA_STATUS [atcab_read_enc](#) (uint16_t key_id, uint8_t block, uint8_t *data, const uint8_t *enc_key, const uint16_t enc_key_id, const uint8_t num_in[(20)])
Executes Read command on a slot configured for encrypted reads and decrypts the data to return it as plaintext.
- ATCA_STATUS [atcab_secureboot](#) (uint8_t mode, uint16_t param2, const uint8_t *digest, const uint8_t *signature, uint8_t *mac)
Executes Secure Boot command, which provides support for secure boot of an external MCU or MPU.
- ATCA_STATUS [atcab_secureboot_mac](#) (uint8_t mode, const uint8_t *digest, const uint8_t *signature, const uint8_t *num_in, const uint8_t *io_key, bool *is_verified)
Executes Secure Boot command with encrypted digest and validated MAC response using the IO protection key.
- ATCA_STATUS [atcab_selftest](#) (uint8_t mode, uint16_t param2, uint8_t *result)
Executes the SelfTest command, which performs a test of one or more of the cryptographic engines within the ATECC608 chip.
- ATCA_STATUS [atcab_sha_base](#) (uint8_t mode, uint16_t length, const uint8_t *data_in, uint8_t *data_out, uint16_t *data_out_size)
Executes SHA command, which computes a SHA-256 or HMAC/SHA-256 digest for general purpose use by the host system.
- ATCA_STATUS [atcab_sha_start](#) (void)
Executes SHA command to initialize SHA-256 calculation engine.
- ATCA_STATUS [atcab_sha_update](#) (const uint8_t *message)
Executes SHA command to add 64 bytes of message data to the current context.
- ATCA_STATUS [atcab_sha_end](#) (uint8_t *digest, uint16_t length, const uint8_t *message)
Executes SHA command to complete SHA-256 or HMAC/SHA-256 operation.
- ATCA_STATUS [atcab_sha_read_context](#) (uint8_t *context, uint16_t *context_size)
Executes SHA command to read the SHA-256 context back. Only for ATECC608 with SHA-256 contexts. HMAC not supported.
- ATCA_STATUS [atcab_sha_write_context](#) (const uint8_t *context, uint16_t context_size)
Executes SHA command to write (restore) a SHA-256 context into the the device. Only supported for ATECC608 with SHA-256 contexts.
- ATCA_STATUS [atcab_sha](#) (uint16_t length, const uint8_t *message, uint8_t *digest)
Use the SHA command to compute a SHA-256 digest.
- ATCA_STATUS [atcab_hw_sha2_256](#) (const uint8_t *data, size_t data_size, uint8_t *digest)
Use the SHA command to compute a SHA-256 digest.
- ATCA_STATUS [atcab_hw_sha2_256_init](#) (atca_sha256_ctx_t *ctx)
Initialize a SHA context for performing a hardware SHA-256 operation on a device. Note that only one SHA operation can be run at a time.
- ATCA_STATUS [atcab_hw_sha2_256_update](#) (atca_sha256_ctx_t *ctx, const uint8_t *data, size_t data_size)
Add message data to a SHA context for performing a hardware SHA-256 operation on a device.
- ATCA_STATUS [atcab_hw_sha2_256_finish](#) (atca_sha256_ctx_t *ctx, uint8_t *digest)

- Finish SHA-256 digest for a SHA context for performing a hardware SHA-256 operation on a device.*
- ATCA_STATUS [atcab_sha_hmac_init](#) ([atca_hmac_sha256_ctx_t](#) *ctx, uint16_t key_slot)
Executes SHA command to start an HMAC/SHA-256 operation.
 - ATCA_STATUS [atcab_sha_hmac_update](#) ([atca_hmac_sha256_ctx_t](#) *ctx, const uint8_t *data, size_t data_size)
Executes SHA command to add an arbitrary amount of message data to a HMAC/SHA-256 operation.
 - ATCA_STATUS [atcab_sha_hmac_finish](#) ([atca_hmac_sha256_ctx_t](#) *ctx, uint8_t *digest, uint8_t target)
Executes SHA command to complete a HMAC/SHA-256 operation.
 - ATCA_STATUS [atcab_sha_hmac](#) (const uint8_t *data, size_t data_size, uint16_t key_slot, uint8_t *digest, uint8_t target)
Use the SHA command to compute an HMAC/SHA-256 operation.
 - ATCA_STATUS [atcab_sha_hmac_ext](#) ([ATCADevice](#) device, const uint8_t *data, size_t data_size, uint16_t key_slot, uint8_t *digest, uint8_t target)
Use the SHA command to compute an HMAC/SHA-256 operation.
 - ATCA_STATUS [atcab_sign_base](#) (uint8_t mode, uint16_t key_id, uint8_t *signature)
Executes the Sign command, which generates a signature using the ECDSA algorithm.
 - ATCA_STATUS [atcab_sign](#) (uint16_t key_id, const uint8_t *msg, uint8_t *signature)
Executes Sign command, to sign a 32-byte external message using the private key in the specified slot. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.
 - ATCA_STATUS [atcab_sign_ext](#) ([ATCADevice](#) device, uint16_t key_id, const uint8_t *msg, uint8_t *signature)
Executes Sign command, to sign a 32-byte external message using the private key in the specified slot. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.
 - ATCA_STATUS [atcab_sign_internal](#) (uint16_t key_id, bool is_invalidate, bool is_full_sn, uint8_t *signature)
Executes Sign command to sign an internally generated message.
 - ATCA_STATUS [atcab_updateextra](#) (uint8_t mode, uint16_t new_value)
Executes UpdateExtra command to update the values of the two extra bytes within the Configuration zone (bytes 84 and 85).
 - ATCA_STATUS [atcab_verify](#) (uint8_t mode, uint16_t key_id, const uint8_t *signature, const uint8_t *public_key, const uint8_t *other_data, uint8_t *mac)
Executes the Verify command, which takes an ECDSA [R,S] signature and verifies that it is correctly generated from a given message and public key. In all cases, the signature is an input to the command.
 - ATCA_STATUS [atcab_verify_extern](#) (const uint8_t *message, const uint8_t *signature, const uint8_t *public_key, bool *is_verified)
Executes the Verify command, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.
 - ATCA_STATUS [atcab_verify_extern_ext](#) ([ATCADevice](#) device, const uint8_t *message, const uint8_t *signature, const uint8_t *public_key, bool *is_verified)
Executes the Verify command, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.
 - ATCA_STATUS [atcab_verify_extern_mac](#) (const uint8_t *message, const uint8_t *signature, const uint8_t *public_key, const uint8_t *num_in, const uint8_t *io_key, bool *is_verified)
Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. This function is only available on the ATECC608.
 - ATCA_STATUS [atcab_verify_stored](#) (const uint8_t *message, const uint8_t *signature, uint16_t key_id, bool *is_verified)
Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.
 - ATCA_STATUS [atcab_verify_stored_ext](#) ([ATCADevice](#) device, const uint8_t *message, const uint8_t *signature, uint16_t key_id, bool *is_verified)
Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

- ATCA_STATUS [atcab_verify_stored_with_tempkey](#) (const uint8_t *signature, uint16_t key_id, bool *is_verified)
Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. keyConfig.reqrandom bit should be set and the message to be signed should be already loaded into TempKey for all devices.
- ATCA_STATUS [atcab_verify_stored_mac](#) (const uint8_t *message, const uint8_t *signature, uint16_t key_id, const uint8_t *num_in, const uint8_t *io_key, bool *is_verified)
Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with a public key stored in the device. This function is only available on the ATECC608.
- ATCA_STATUS [atcab_verify_validate](#) (uint16_t key_id, const uint8_t *signature, const uint8_t *other_data, bool *is_verified)
Executes the Verify command in Validate mode to validate a public key stored in a slot.
- ATCA_STATUS [atcab_verify_invalidate](#) (uint16_t key_id, const uint8_t *signature, const uint8_t *other_data, bool *is_verified)
Executes the Verify command in Invalidate mode which invalidates a previously validated public key stored in a slot.
- ATCA_STATUS [atcab_write](#) (uint8_t zone, uint16_t address, const uint8_t *value, const uint8_t *mac)
Executes the Write command, which writes either one four byte word or a 32-byte block to one of the EEPROM zones on the device. Depending upon the value of the WriteConfig byte for this slot, the data may be required to be encrypted by the system prior to being sent to the device. This command cannot be used to write slots configured as ECC private keys.
- ATCA_STATUS [atcab_write_zone](#) (uint8_t zone, uint16_t slot, uint8_t block, uint8_t offset, const uint8_t *data, uint8_t len)
Executes the Write command, which writes either 4 or 32 bytes of data into a device zone.
- ATCA_STATUS [atcab_write_zone_ext](#) (ATCADevice device, uint8_t zone, uint16_t slot, uint8_t block, uint8_t offset, const uint8_t *data, uint8_t len)
Executes the Write command, which writes either 4 or 32 bytes of data into a device zone.
- ATCA_STATUS [atcab_write_bytes_zone_ext](#) (ATCADevice device, uint8_t zone, uint16_t slot, size_t offset_bytes, const uint8_t *data, size_t length)
Executes the Write command, which writes data into the configuration, otp, or data zones with a given byte offset and length. Offset and length must be multiples of a word (4 bytes).
- ATCA_STATUS [atcab_write_bytes_zone](#) (uint8_t zone, uint16_t slot, size_t offset_bytes, const uint8_t *data, size_t length)
Executes the Write command, which writes data into the configuration, otp, or data zones with a given byte offset and length. Offset and length must be multiples of a word (4 bytes).
- ATCA_STATUS [atcab_write_pubkey](#) (uint16_t slot, const uint8_t *public_key)
Uses the write command to write a public key to a slot in the proper format.
- ATCA_STATUS [atcab_write_pubkey_ext](#) (ATCADevice device, uint16_t slot, const uint8_t *public_key)
Uses the write command to write a public key to a slot in the proper format.
- ATCA_STATUS [atcab_write_config_zone](#) (const uint8_t *config_data)
Executes the Write command, which writes the configuration zone.
- ATCA_STATUS [atcab_write_config_zone_ext](#) (ATCADevice device, const uint8_t *config_data)
Executes the Write command, which writes the configuration zone.
- ATCA_STATUS [atcab_write_enc](#) (uint16_t key_id, uint8_t block, const uint8_t *data, const uint8_t *enc_key, const uint16_t enc_key_id, const uint8_t num_in[(20)])
Executes the Write command, which performs an encrypted write of a 32 byte block into given slot.
- ATCA_STATUS [atcab_write_config_counter](#) (uint16_t counter_id, uint32_t counter_value)
Initialize one of the monotonic counters in device with a specific value.

Variables

- ATCADevice [g_atcab_device_ptr](#)

24.37.1 Detailed Description

CryptoAuthLib Basic API methods - a simple crypto authentication API. These methods manage a global ATCADevice object behind the scenes. They also manage the wake/idle state transitions so callers don't need to.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.38 atca_cfgs.c File Reference

a set of default configurations for various ATCA devices and interfaces

```
#include <stddef.h>
#include "cryptoauthlib.h"
#include "atca_cfgs.h"
#include "atca_iface.h"
#include "atca_device.h"
```

24.38.1 Detailed Description

a set of default configurations for various ATCA devices and interfaces

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.39 atca_cfgs.h File Reference

a set of default configurations for various ATCA devices and interfaces

```
#include "atca_iface.h"
```

24.39.1 Detailed Description

a set of default configurations for various ATCA devices and interfaces

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.40 atca_compiler.h File Reference

CryptoAuthLib is meant to be portable across architectures, even non-Microchip architectures and compiler environments. This file is for isolating compiler specific macros.

```
#include <stdbool.h>
```

Macros

- #define **SHARED_LIB_EXPORT**
- #define **ATCA_DLL** extern
- #define **ATCA_PACKED**
- #define **UNUSED_VAR**(x)

24.40.1 Detailed Description

CryptoAuthLib is meant to be portable across architectures, even non-Microchip architectures and compiler environments. This file is for isolating compiler specific macros.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.40.2 Macro Definition Documentation

24.40.2.1 UNUSED_VAR

```
#define UNUSED_VAR(  
    x )
```

Enables removal of compiler warning due to unused variables

24.41 atca_config_check.h File Reference

Consistency checks for configuration options.

```
#include "atca_config.h"
```

Macros

- #define **FEATURE_ENABLED** (1)
- #define **FEATURE_DISABLED** (0)
- #define **DEFAULT_ENABLED** FEATURE_ENABLED
- #define **DEFAULT_DISABLED** FEATURE_DISABLED
- #define **LIBRARY_BUILD_EN_CHECK** 0
- #define **LIBRARY_USAGE_EN_CHECK** 0
- #define **ATCA_SHA_SUPPORT** 1
- #define **ATCA_ECC_SUPPORT** DEFAULT_ENABLED
- #define **ATCA_CA2_SUPPORT** DEFAULT_ENABLED
- #define **ATCA_CA2_CERT_SUPPORT** DEFAULT_ENABLED
- #define **ATCA_CA_SUPPORT** DEFAULT_ENABLED
- #define **ATCA_HOSTLIB_EN** DEFAULT_ENABLED
- #define **ATCA_USE_ATCAB_FUNCTIONS**
- #define **ATCA_CHECK_PARAMS_EN** DEFAULT_ENABLED
- #define **ATCA_CHECK_INVALID_MSG**(c, s, m) if (c) { return ATCA_TRACE(s, m); }
- #define **ATCA_CHECK_VALID_MSG**(c, m) if (!ATCA_TRACE(!c, m))
- #define **ATCA_CHECK_INVALID**(c, s) **ATCA_CHECK_INVALID_MSG**(c, s, "")
- #define **ATCA_CHECK_VALID**(c) **ATCA_CHECK_VALID_MSG**(c, "")
- #define **MULTIPART_BUF_EN** (DEFAULT_DISABLED)
- #define **ATCA_HEAP**
- #define **ATCA_UNUSED_VAR_CHECK** (DEFAULT_ENABLED)
- #define **ATCAB_AES_EN** (DEFAULT_ENABLED)
- #define **ATCAB_AES_GFM_EN** (DEFAULT_ENABLED)
- #define **ATCAB_AES_GCM_EN** (DEFAULT_ENABLED)
- #define **ATCAB_CHECKMAC_EN** (DEFAULT_ENABLED)
- #define **ATCAB_COUNTER_EN** (DEFAULT_ENABLED)
- #define **ATCAB_DERIVEKEY_EN** (DEFAULT_ENABLED)
- #define **ATCAB_ECDH_EN** (DEFAULT_ENABLED)
- #define **ATCAB_ECDH_ENC_EN** (DEFAULT_ENABLED)
- #define **ATCAB_GENDIG_EN** (DEFAULT_ENABLED)
- #define **ATCAB_GENKEY_EN** (DEFAULT_ENABLED)
- #define **ATCAB_GENKEY_MAC_EN** ATCAB_GENKEY_EN
- #define **ATCAB_HMAC_EN** (DEFAULT_ENABLED)
- #define **ATCAB_INFO_LATCH_EN** (DEFAULT_ENABLED)
- #define **ATCAB_KDF_EN** (DEFAULT_ENABLED)
- #define **ATCAB_LOCK_EN** (DEFAULT_ENABLED)
- #define **ATCAB_MAC_EN** (DEFAULT_ENABLED)
- #define **ATCAB_NONCE_EN** (DEFAULT_ENABLED)
- #define **ATCAB_PRIVWRITE_EN** (DEFAULT_ENABLED)
- #define **ATCAB_RANDOM_EN** (DEFAULT_ENABLED)
- #define **ATCAB_READ_EN** (DEFAULT_ENABLED)
- #define **ATCAB_READ_ENC_EN** ATCAB_READ_EN
- #define **ATCAB_SECUREBOOT_EN** (DEFAULT_ENABLED)
- #define **ATCAB_SECUREBOOT_MAC_EN** ATCAB_SECUREBOOT_EN
- #define **ATCAB_SELFTEST_EN** (DEFAULT_ENABLED)
- #define **ATCAB_SHA_EN** (DEFAULT_ENABLED)
- #define **ATCAB_SHA_HMAC_EN** ATCAB_SHA_EN
- #define **ATCAB_SHA_CONTEXT_EN** ATCAB_SHA_EN
- #define **ATCAB_SIGN_EN** (DEFAULT_ENABLED)
- #define **ATCAB_SIGN_INTERNAL_EN** ATCAB_SIGN_EN
- #define **ATCAB_UPDATEEXTRA_EN** (DEFAULT_ENABLED)
- #define **ATCAB_VERIFY_EN** (DEFAULT_ENABLED)
- #define **ATCAB_VERIFY_EXTERN_EN** ATCAB_VERIFY_EN

- `#define ATCAB_VERIFY_MAC_EN ATCAB_VERIFY_EN`
- `#define ATCAB_VERIFY_STORED_EN ATCAB_VERIFY_EN`
- `#define ATCAB_VERIFY_VALIDATE_EN ATCAB_VERIFY_EN`
- `#define ATCAB_WRITE_EN (DEFAULT_ENABLED)`
- `#define ATCAB_WRITE_ENC_EN ATCAB_WRITE_EN`
- `#define ATCAC_SHA1_EN (DEFAULT_ENABLED)`
- `#define ATCAC_SHA256_EN (FEATURE_ENABLED)`
- `#define ATCAC_SHA384_EN (FEATURE_DISABLED)`
- `#define ATCAC_SHA512_EN (FEATURE_DISABLED)`
- `#define ATCAC_SHA256_HMAC_EN ATCAC_SHA256_EN`
- `#define ATCAC_SHA256_HMAC_CTR_EN ATCAC_SHA256_HMAC_EN`
- `#define ATCAC_RANDOM_EN ATCA_HOSTLIB_EN`
- `#define ATCAC_VERIFY_EN ATCA_HOSTLIB_EN`
- `#define ATCAC_SIGN_EN ATCA_HOSTLIB_EN`
- `#define ATCAC_CERT_ADD_EN ATCA_HOSTLIB_EN`

24.41.1 Detailed Description

Consistency checks for configuration options.

Copyright

(c) 2015-2021 Microchip Technology Inc. and its subsidiaries.

24.41.2 Macro Definition Documentation

24.41.2.1 ATCA_CHECK_INVALID_MSG

```
#define ATCA_CHECK_INVALID_MSG(  
    c,  
    s,  
    m ) if (c) { return ATCA_TRACE(s, m); }
```

Emits message and returns the status code when the condition is true

24.41.2.2 ATCA_UNUSED_VAR_CHECK

```
#define ATCA_UNUSED_VAR_CHECK (DEFAULT_ENABLED)
```

Enables removal of compiler warning due to unused variables

24.41.2.3 ATCA_USE_ATCAB_FUNCTIONS

```
#define ATCA_USE_ATCAB_FUNCTIONS
```

Does the atcab_ API layer need to be instantiated (adds a layer of abstraction)

24.41.2.4 ATCAB_AES_GFM_EN

```
#define ATCAB_AES_GFM_EN (DEFAULT_ENABLED)
```

Enable ATCAB_AES_GFM_EN to enabled Galois Field Multiply

Supported API's: atcab_aes

24.41.2.5 ATCAB_GENKEY_MAC_EN

```
#define ATCAB_GENKEY_MAC_EN ATCAB_GENKEY_EN
```

Requires: ATCAB_GENKEY_EN

Enable ATCAB_GENKEY_MAC_EN which provides for a mac with the genkey command

Supported API's: atcab_genkey_base

24.41.2.6 ATCAB_INFO_LATCH_EN

```
#define ATCAB_INFO_LATCH_EN (DEFAULT_ENABLED)
```

Enable ATCAB_INFO_LATCH_EN which enables control of GPIOs and the persistent latch

Supported API's: atcab_info_base

24.41.2.7 ATCAB_VERIFY_MAC_EN

```
#define ATCAB_VERIFY_MAC_EN ATCAB_VERIFY_EN
```

Requires: ATCAB_VERIFY

Executes verification command with verification MAC for the External or Stored Verify modes

Supported API's: atcab_verify_extern_mac, atcab_verify_stored_mac

24.41.2.8 ATCAB_WRITE_EN

```
#define ATCAB_WRITE_EN (DEFAULT_ENABLED)
```

Enable CALIB_WRITE which writes either one four byte word or a 32-byte block to one of the EEPROM zones on the device

Supported API's: calib_write

Supported ECC204 specific API's: calib_ca2_write

24.41.2.9 ATCAC_CERT_ADD_EN

```
#define ATCAC_CERT_ADD_EN ATCA_HOSTLIB_EN
```

Requires: ATCA_HOSTLIB_EN

Enable ATCAC_CERT_ADD_EN to use the host's cert add functions. Generally assumed to come from the host's cryptographic library or peripheral driver.

24.41.2.10 ATCAC_RANDOM_EN

```
#define ATCAC_RANDOM_EN ATCA_HOSTLIB_EN
```

Requires: ATCA_HOSTLIB_EN

Enable ATCAC_RANDOM_EN get random numbers from the host's implementation - generally assumed to come from the host's cryptographic library or peripheral driver

24.41.2.11 ATCAC_SHA1_EN

```
#define ATCAC_SHA1_EN (DEFAULT_ENABLED)
```

Enable ATCAC_SHA1_EN to enable sha1 host side api

Supported API's: atcab_write

24.41.2.12 ATCAC_SHA256_EN

```
#define ATCAC_SHA256_EN (FEATURE_ENABLED)
```

Enable ATCAC_SHA256_EN to enable sha256 host side api

24.41.2.13 ATCAC_SHA384_EN

```
#define ATCAC_SHA384_EN (FEATURE_DISABLED)
```

Enable ATCAC_SHA384_EN to enable sha384 host side api

Disabled by default. Enable ATCAC_SHA512_EN to use SHA384

24.41.2.14 ATCAC_SHA512_EN

```
#define ATCAC_SHA512_EN (FEATURE_DISABLED)
```

Enable ATCAC_SHA512_EN to enable sha512 host side api

Disabled by default. Use FEATURE_ENABLED to enable this feature

24.41.2.15 ATCAC_SIGN_EN

```
#define ATCAC_SIGN_EN ATCA_HOSTLIB_EN
```

Requires: ATCA_HOSTLIB_EN

Enable ATCAC_SIGN_EN to use the host's sign functions. Generally assumed to come from the host's cryptographic library or peripheral driver.

24.41.2.16 ATCAC_VERIFY_EN

```
#define ATCAC_VERIFY_EN ATCA_HOSTLIB_EN
```

Requires: ATCA_HOSTLIB_EN

Enable ATCAC_VERIFY_EN to use the host's verify functions. Generally assumed to come from the host's cryptographic library or peripheral driver.

24.41.2.17 LIBRARY_BUILD_EN_CHECK

```
#define LIBRARY_BUILD_EN_CHECK 0
```

Library Configuration File - All build attributes should be included in atca_config.h

24.41.2.18 MULTIPART_BUF_EN

```
#define MULTIPART_BUF_EN (DEFAULT_DISABLED)
```

Enables multipart buffer handling (generally for small memory model platforms)

24.42 atca_debug.c File Reference

Debug/Trace for CryptoAuthLib calls.

```
#include "cryptoauthlib.h"
```

Functions

- ATCA_STATUS **atca_trace** (ATCA_STATUS status)

24.42.1 Detailed Description

Debug/Trace for CryptoAuthLib calls.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.43 atca_device.c File Reference

Microchip CryptoAuth device object.

```
#include "cryptoauthlib.h"
```

Functions

- [ATCADevice newATCADevice](#) ([ATCAIfaceCfg](#) *cfg)
constructor for a Microchip CryptoAuth device
- void [deleteATCADevice](#) ([ATCADevice](#) *ca_dev)
destructor for a device NULLs reference after object is freed
- ATCA_STATUS [initATCADevice](#) ([ATCAIfaceCfg](#) *cfg, [ATCADevice](#) ca_dev)
Initializer for an Microchip CryptoAuth device.
- [ATCAIface atGetIFace](#) ([ATCADevice](#) dev)
returns a reference to the ATCAIface interface object for the device
- ATCA_STATUS [releaseATCADevice](#) ([ATCADevice](#) ca_dev)
Release any resources associated with the device.

24.43.1 Detailed Description

Microchip CryptoAuth device object.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.44 atca_device.h File Reference

Microchip Crypto Auth device object.

```
#include "atca_iface.h"
```

Data Structures

- struct [atca_device](#)
[atca_device](#) is the C object backing ATCADevice. See the [atca_device.h](#) file for details on the ATCADevice methods

Typedefs

- typedef void(* [ctx_cb](#)) (void *ctx)
Callback function to clean up the session context.
- typedef struct [atca_device](#) * [ATCADevice](#)

Enumerations

- enum [ATCADeviceState](#) { [ATCA_DEVICE_STATE_UNKNOWN](#) = 0 , [ATCA_DEVICE_STATE_SLEEP](#) , [ATCA_DEVICE_STATE_IDLE](#) , [ATCA_DEVICE_STATE_ACTIVE](#) }
ATCADeviceState says about device state.

Functions

- [ATCA_STATUS](#) [initATCADevice](#) ([ATCAIfaceCfg](#) *cfg, [ATCADevice](#) ca_dev)
Initializer for an Microchip CryptoAuth device.
- [ATCA_STATUS](#) [releaseATCADevice](#) ([ATCADevice](#) ca_dev)
Release any resources associated with the device.
- [ATCAIface](#) [atGetIFace](#) ([ATCADevice](#) dev)
returns a reference to the ATCAIface interface object for the device

24.44.1 Detailed Description

Microchip Crypto Auth device object.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.45 atca_devtypes.h File Reference

Microchip Crypto Auth.

```
#include <stdint.h>
```

Macros

- #define **ATSHA204A** (0U)
The supported Device type in Cryptoauthlib library.
- #define **ATECC108A** (1U)
- #define **ATECC508A** (2U)
- #define **ATECC608A** (3U)
- #define **ATECC608B** (3U)
- #define **ATECC608** (3U)
- #define **ATSHA206A** (4U)
- #define **TA100** (0x10U)
- #define **TA101** (0x11U)
- #define **ECC204** (0x20U)
- #define **TA010** (0x21U)
- #define **ECC206** (0x22U)
- #define **RNG90** (0x23U)
- #define **SHA104** (0x24U)
- #define **SHA105** (0x25U)
- #define **SHA106** (0x26U)
- #define **ATCA_DEV_UNKNOWN** (0x7EU)
- #define **ATCA_DEV_INVALID** (0x7FU)

Typedefs

- typedef uint8_t **ATCADeviceType**

24.45.1 Detailed Description

Microchip Crypto Auth.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.46 atca_helpers.c File Reference

Helpers to support the CryptoAuthLib Basic API methods.

```
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include "cryptoauthlib.h"
#include "atca_helpers.h"
```

Macros

- #define **B64_IS_EQUAL** (64u)
- #define **B64_IS_INVALID** (-1)

Functions

- const uint8_t * **atcab_b64rules_default** (void)
- const uint8_t * **atcab_b64rules_mime** (void)
- const uint8_t * **atcab_b64rules_urlsafe** (void)
- ATCA_STATUS **atcab_bin2hex** (const uint8_t *bin, size_t bin_size, char *hex, size_t *hex_size)
Convert a binary buffer to a hex string for easy reading.
- ATCA_STATUS **atcab_reversal** (const uint8_t *bin, size_t bin_size, uint8_t *dest, size_t *dest_size)
To reverse the input data.
- ATCA_STATUS **atcab_bin2hex_** (const uint8_t *bin, size_t bin_size, char *hex, size_t *hex_size, bool is_↵
pretty, bool is_space, bool is_upper)
Function that converts a binary buffer to a hex string suitable for easy reading.
- ATCA_STATUS **atcab_hex2bin_** (const char *hex, size_t hex_size, uint8_t *bin, size_t *bin_size, bool is_↵
_space)
- ATCA_STATUS **atcab_hex2bin** (const char *ascii_hex, size_t ascii_hex_len, uint8_t *binary, size_t *bin_len)
Function that converts a hex string to binary buffer.
- bool **isDigit** (char c)
Checks to see if a character is an ASCII representation of a digit ((c ge '0') and (c le '9'))
- bool **isBlankSpace** (char c)
Checks to see if a character is blank space.

- bool **isAlpha** (char c)
Checks to see if a character is an ASCII representation of hex ((c >= 'A') and (c <= 'F')) || ((c >= 'a') and (c <= 'f'))
- bool **isHexAlpha** (char c)
Checks to see if a character is an ASCII representation of hex ((c >= 'A') and (c <= 'F')) || ((c >= 'a') and (c <= 'f'))
- bool **isHex** (char c)
Returns true if this character is a valid hex character or if this is blankspace (The character can be included in a valid hexstring).
- bool **isHexDigit** (char c)
Returns true if this character is a valid hex character.
- ATCA_STATUS **packHex** (const char *ascii_hex, size_t ascii_hex_len, char *packed_hex, size_t *packed_hex_len)
Remove spaces from a ASCII hex string.
- bool **isBase64** (char c, const uint8_t *rules)
Returns true if this character is a valid base 64 character or if this is space (A character can be included in a valid base 64 string).
- bool **isBase64Digit** (char c, const uint8_t *rules)
Returns true if this character is a valid base 64 character.
- ATCA_STATUS **atcab_base64decode** (const char *encoded, size_t encoded_size, uint8_t *data, size_t *data_size, const uint8_t *rules)
Decode base64 string to data with ruleset option.
- ATCA_STATUS **atcab_base64encode** (const uint8_t *data, size_t data_size, char *encoded, size_t *encoded_size, const uint8_t *rules)
Encode data as base64 string with ruleset option.
- ATCA_STATUS **atcab_base64encode** (const uint8_t *byte_array, size_t array_len, char *encoded, size_t *encoded_len)
Encode data as base64 string.
- ATCA_STATUS **atcab_base64decode** (const char *encoded, size_t encoded_len, uint8_t *byte_array, size_t *array_len)
Decode base64 string to data.
- size_t **atcab_pointer_delta** (const void *start, const void *end)
Helper function to calculate the number of bytes between two pointers.
- int **atcab_memset_s** (void *dest, size_t destsz, int ch, size_t count)
Guaranteed to perform memory writes regardless of optimization level. Matches memset_s signature.
- char **lib_toupper** (char c)
Converts a character to uppercase.
- char **lib_tolower** (char c)
Converts a character to lowercase.
- const char * **lib_strcasestr** (const char *haystack, const char *needle)
Search for a substring in a case insensitive format.

24.46.1 Detailed Description

Helpers to support the CryptoAuthLib Basic API methods.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.46.2 Function Documentation

24.46.2.1 atcab_base64decode()

```
ATCA_STATUS atcab_base64decode (
    const char * encoded,
    size_t encoded_len,
    uint8_t * byte_array,
    size_t * array_len )
```

Decode base64 string to data.

Parameters

in	<i>encoded</i>	Base64 string to be decoded.
in	<i>encoded_len</i>	Size of the base64 string in bytes.
out	<i>byte_array</i>	Decoded data will be returned here.
in, out	<i>array_len</i>	As input, the size of the byte_array buffer. As output, the length of the decoded data.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.46.2.2 atcab_base64decode_()

```
ATCA_STATUS atcab_base64decode_ (
    const char * encoded,
    size_t encoded_size,
    uint8_t * data,
    size_t * data_size,
    const uint8_t * rules )
```

Decode base64 string to data with ruleset option.

Parameters

in	<i>encoded</i>	Base64 string to be decoded.
in	<i>encoded_size</i>	Size of the base64 string in bytes.
out	<i>data</i>	Decoded data will be returned here.
in, out	<i>data_size</i>	As input, the size of the byte_array buffer. As output, the length of the decoded data.
in	<i>rules</i>	base64 ruleset to use

24.46.2.3 atcab_base64encode()

```
ATCA_STATUS atcab_base64encode (
    const uint8_t * byte_array,
    size_t array_len,
```

```
char * encoded,
size_t * encoded_len )
```

Encode data as base64 string.

Parameters

in	<i>byte_array</i>	Data to be encode in base64.
in	<i>array_len</i>	Size of <i>byte_array</i> in bytes.
in	<i>encoded</i>	Base64 output is returned here.
in, out	<i>encoded_len</i>	As input, the size of the encoded buffer. As output, the length of the encoded base64 character string.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.46.2.4 atcab_base64encode_()

```
ATCA_STATUS atcab_base64encode_ (
    const uint8_t * data,
    size_t data_size,
    char * encoded,
    size_t * encoded_size,
    const uint8_t * rules )
```

Encode data as base64 string with ruleset option.

Parameters

in	<i>data</i>	The input byte array that will be converted to base 64 encoded characters
in	<i>data_size</i>	The length of the byte array
in	<i>encoded</i>	The output converted to base 64 encoded characters.
in, out	<i>encoded_size</i>	Input: The size of the encoded buffer, Output: The length of the encoded base 64 character string
in	<i>rules</i>	ruleset to use during encoding

24.46.2.5 atcab_bin2hex()

```
ATCA_STATUS atcab_bin2hex (
    const uint8_t * bin,
    size_t bin_size,
    char * hex,
    size_t * hex_size )
```

Convert a binary buffer to a hex string for easy reading.

Parameters

in	<i>bin</i>	Input data to convert.
in	<i>bin_size</i>	Size of data to convert.
out	<i>hex</i>	Buffer that receives hex string.
in, out	<i>hex_size</i>	As input, the size of the hex buffer. As output, the size of the output hex.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.46.2.6 atcab_bin2hex()

```
ATCA_STATUS atcab_bin2hex_ (
    const uint8_t * bin,
    size_t bin_size,
    char * hex,
    size_t * hex_size,
    bool is_pretty,
    bool is_space,
    bool is_upper )
```

Function that converts a binary buffer to a hex string suitable for easy reading.

Parameters

in	<i>bin</i>	Input data to convert.
in	<i>bin_size</i>	Size of data to convert.
out	<i>hex</i>	Buffer that receives hex string.
in, out	<i>hex_size</i>	As input, the size of the hex buffer. As output, the size of the output hex.
in	<i>is_pretty</i>	Indicates whether new lines should be added for pretty printing.
in	<i>is_space</i>	Convert the output hex with space between it.
in	<i>is_upper</i>	Convert the output hex to upper case.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.46.2.7 atcab_hex2bin()

```
ATCA_STATUS atcab_hex2bin (
    const char * ascii_hex,
    size_t ascii_hex_len,
    uint8_t * binary,
    size_t * bin_len )
```

Function that converts a hex string to binary buffer.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ascii_hex</i>	Input buffer to convert
in	<i>ascii_hex_len</i>	Length of buffer to convert
out	<i>binary</i>	Buffer that receives binary
in, out	<i>bin_len</i>	As input, the size of the bin buffer. As output, the size of the bin data.

24.46.2.8 atcab_reversal()

```
ATCA_STATUS atcab_reversal (
    const uint8_t * bin,
    size_t bin_size,
    uint8_t * dest,
    size_t * dest_size )
```

To reverse the input data.

Parameters

in	<i>bin</i>	Input data to reverse.
in	<i>bin_size</i>	Size of data to reverse.
out	<i>dest</i>	Buffer to store reversed binary data.
in	<i>dest_size</i>	The size of the dest buffer.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.46.2.9 isAlpha()

```
bool isAlpha (
    char c )
```

Checks to see if a character is an ASCII representation of hex ((c >= 'A') and (c <= 'F')) || ((c >= 'a') and (c <= 'f'))

Parameters

in	<i>c</i>	character to check
----	----------	--------------------

Returns

True if the character is a hex

24.46.2.10 isBase64()

```
bool isBase64 (
    char c,
    const uint8_t * rules )
```

Returns true if this character is a valid base 64 character or if this is space (A character can be included in a valid base 64 string).

Parameters

in	<i>c</i>	character to check
in	<i>rules</i>	base64 ruleset to use

Returns

True if the character can be included in a valid base 64 string

24.46.2.11 isBase64Digit()

```
bool isBase64Digit (
    char c,
    const uint8_t * rules )
```

Returns true if this character is a valid base 64 character.

Parameters

in	<i>c</i>	character to check
in	<i>rules</i>	base64 ruleset to use

Returns

True if the character can be included in a valid base 64 string

24.46.2.12 isBlankSpace()

```
bool isBlankSpace (
    char c )
```

Checks to see if a character is blank space.

Parameters

in	c	character to check
----	---	--------------------

Returns

True if the character is blankspace

24.46.2.13 isDigit()

```
bool isDigit (  
    char c )
```

Checks to see if a character is an ASCII representation of a digit ((c ge '0') and (c le '9'))

Parameters

in	c	character to check
----	---	--------------------

Returns

True if the character is a digit

24.46.2.14 isHex()

```
bool isHex (  
    char c )
```

Returns true if this character is a valid hex character or if this is blankspace (The character can be included in a valid hexstring).

Parameters

in	c	character to check
----	---	--------------------

Returns

True if the character can be included in a valid hexstring

24.46.2.15 isHexAlpha()

```
bool isHexAlpha (  
    char c )
```

24.46 atca_helpers.c File Reference

Checks to see if a character is an ASCII representation of hex ((c >= 'A') and (c <= 'F')) || ((c >= 'a') and (c <= 'f'))

Parameters

in	<i>c</i>	character to check
----	----------	--------------------

Returns

True if the character is a hex

24.46.2.16 isHexDigit()

```
bool isHexDigit (  
    char c )
```

Returns true if this character is a valid hex character.

Parameters

in	<i>c</i>	character to check
----	----------	--------------------

Returns

True if the character can be included in a valid hexstring

24.46.2.17 packHex()

```
ATCA_STATUS packHex (  
    const char * ascii_hex,  
    size_t ascii_hex_len,  
    char * packed_hex,  
    size_t * packed_len )
```

Remove spaces from a ASCII hex string.

Parameters

in	<i>ascii_hex</i>	Initial hex string to remove blankspace from
in	<i>ascii_hex_len</i>	Length of the initial hex string
in	<i>packed_hex</i>	Resulting hex string without blankspace
in, out	<i>packed_len</i>	In: Size to packed_hex buffer Out: Number of bytes in the packed hex string

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.47 atca_helpers.h File Reference

Helpers to support the CryptoAuthLib Basic API methods.

```
#include "cryptoauthlib.h"
```

- #define **IS_ADD_SAFE_UINT16_T**(a, b) (((UINT16_MAX - (a)) >= (b)) ? true : false)
- #define **IS_ADD_SAFE_UINT32_T**(a, b) (((UINT32_MAX - (a)) >= (b)) ? true : false)
- #define **IS_ADD_SAFE_UINT64_T**(a, b) (((UINT64_MAX - (a)) >= (b)) ? true : false)
- #define **IS_ADD_SAFE_SIZE_T**(a, b) (((SIZE_MAX - (a)) >= (b)) ? true : false)
- #define **IS_MUL_SAFE_UINT16_T**(a, b) (((a) <= UINT16_MAX / (b))) ? true : false)
- #define **IS_MUL_SAFE_UINT32_T**(a, b) (((a) <= UINT32_MAX / (b))) ? true : false)
- #define **IS_MUL_SAFE_UINT64_T**(a, b) (((a) <= UINT64_MAX / (b))) ? true : false)
- #define **IS_MUL_SAFE_SIZE_T**(a, b) (((a) <= SIZE_MAX / (b))) ? true : false)
- #define **ABS_VAL**(x) (((x) < 0) ? -(x) : (x))
- ATCA_STATUS **atcab_bin2hex** (const uint8_t *bin, size_t bin_size, char *hex, size_t *hex_size)
Convert a binary buffer to a hex string for easy reading.
- ATCA_STATUS **atcab_bin2hex** (const uint8_t *bin, size_t bin_size, char *hex, size_t *hex_size, bool is_↵
pretty, bool is_space, bool is_upper)
Function that converts a binary buffer to a hex string suitable for easy reading.
- ATCA_STATUS **atcab_hex2bin** (const char *ascii_hex, size_t ascii_hex_len, uint8_t *binary, size_t *bin_len)
Function that converts a hex string to binary buffer.
- ATCA_STATUS **atcab_hex2bin** (const char *hex, size_t hex_size, uint8_t *bin, size_t *bin_size, bool is_↵
space)
- ATCA_STATUS **packHex** (const char *ascii_hex, size_t ascii_hex_len, char *packed_hex, size_t *packed_↵
_len)
Remove spaces from a ASCII hex string.
- bool **isDigit** (char c)
Checks to see if a character is an ASCII representation of a digit ((c ge '0') and (c le '9'))
- bool **isBlankSpace** (char c)
Checks to see if a character is blank space.
- bool **isAlpha** (char c)
Checks to see if a character is an ASCII representation of hex ((c >= 'A') and (c <= 'F')) || ((c >= 'a') and (c <= 'f'))
- bool **isHexAlpha** (char c)
Checks to see if a character is an ASCII representation of hex ((c >= 'A') and (c <= 'F')) || ((c >= 'a') and (c <= 'f'))
- bool **isHex** (char c)
Returns true if this character is a valid hex character or if this is blankspace (The character can be included in a valid hexstring).
- bool **isHexDigit** (char c)
Returns true if this character is a valid hex character.
- bool **isBase64** (char c, const uint8_t *rules)
Returns true if this character is a valid base 64 character or if this is space (A character can be included in a valid base 64 string).
- bool **isBase64Digit** (char c, const uint8_t *rules)
Returns true if this character is a valid base 64 character.
- const uint8_t * **atcab_b64rules_default** (void)
- const uint8_t * **atcab_b64rules_mime** (void)

- `const uint8_t * atcab_b64rules_urlsafe` (void)
- ATCA_STATUS `atcab_base64decode_` (const char *encoded, size_t encoded_size, uint8_t *data, size_t *data_size, const uint8_t *rules)
Decode base64 string to data with ruleset option.
- ATCA_STATUS `atcab_base64encode` (const uint8_t *byte_array, size_t array_len, char *encoded, size_t *encoded_len)
Encode data as base64 string.
- ATCA_STATUS `atcab_base64encode_` (const uint8_t *data, size_t data_size, char *encoded, size_t *encoded_size, const uint8_t *rules)
Encode data as base64 string with ruleset option.
- ATCA_STATUS `atcab_base64decode` (const char *encoded, size_t encoded_len, uint8_t *byte_array, size_t *array_len)
Decode base64 string to data.
- ATCA_STATUS `atcab_reversal` (const uint8_t *bin, size_t bin_size, uint8_t *dest, size_t *dest_size)
To reverse the input data.
- `int atcab_memset_s` (void *dest, size_t destsz, int ch, size_t count)
Guaranteed to perform memory writes regardless of optimization level. Matches memset_s signature.
- `size_t atcab_pointer_delta` (const void *start, const void *end)
Helper function to calculate the number of bytes between two pointers.
- `char lib_toupper` (char c)
Converts a character to uppercase.
- `char lib_tolower` (char c)
Converts a character to lowercase.

24.47.1 Detailed Description

Helpers to support the CryptoAuthLib Basic API methods.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.48 atca_iface.c File Reference

Microchip CryptoAuthLib hardware interface object.

```
#include "cryptoauthlib.h"
#include <ctype.h>
```

Data Structures

- struct `devtype_names_t`

Functions

- ATCA_STATUS [initATCAIface](#) ([ATCAIfaceCfg](#) *cfg, [ATCAIface](#) ca_iface)
Initializer for ATCAIface objects.
- ATCA_STATUS [atinit](#) ([ATCAIface](#) ca_iface)
Performs the HAL initialization by calling intermediate HAL wrapper function. If using the basic API, the [atcab_init\(\)](#) function should be called instead.
- ATCA_STATUS [atsend](#) ([ATCAIface](#) ca_iface, uint8_t word_address, uint8_t *txdata, int txlength)
Sends the data to the device by calling intermediate HAL wrapper function.
- ATCA_STATUS [atreceive](#) ([ATCAIface](#) ca_iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxlength)
Receives data from the device by calling intermediate HAL wrapper function.
- ATCA_STATUS [atcontrol](#) ([ATCAIface](#) ca_iface, uint8_t option, void *param, size_t paramlen)
Perform control operations with the underlying hal driver.
- ATCA_STATUS [atwake](#) ([ATCAIface](#) ca_iface)
Wakes up the device by calling intermediate HAL wrapper function. The [atcab_wakeup\(\)](#) function should be used instead.
- ATCA_STATUS [atidle](#) ([ATCAIface](#) ca_iface)
Puts the device into idle state by calling intermediate HAL wrapper function. The [atcab_idle\(\)](#) function should be used instead.
- ATCA_STATUS [atsleep](#) ([ATCAIface](#) ca_iface)
Puts the device into sleep state by calling intermediate HAL wrapper function. The [atcab_sleep\(\)](#) function should be used instead.
- [ATCAIfaceCfg](#) * [atgetifacecfg](#) ([ATCAIface](#) ca_iface)
Returns the logical interface configuration for the device.
- void * [atgetifacehaldat](#) ([ATCAIface](#) ca_iface)
Returns the HAL data pointer for the device.
- bool [iface_type_is_kit](#) ([ATCAIfaceType](#) iface_type)
Check if the given interface is a "kit protocol" one.
- bool [atca_iface_is_kit](#) ([ATCAIface](#) ca_iface)
Check if the given interface is configured as a "kit protocol" one where transactions are atomic.
- bool [atca_iface_is_swi](#) ([ATCAIface](#) ca_iface)
Check if the given interface is configured as a SWI.
- int [atca_iface_get_retries](#) ([ATCAIface](#) ca_iface)
Retrieve the number of retries for a configured interface.
- uint16_t [atca_iface_get_wake_delay](#) ([ATCAIface](#) ca_iface)
Retrieve the wake/retry delay for a configured interface/device.
- uint8_t [ifacecfg_get_address](#) ([ATCAIfaceCfg](#) *cfg)
Retrieves the device address given an interface configuration.
- ATCA_STATUS [ifacecfg_set_address](#) ([ATCAIfaceCfg](#) *cfg, uint8_t address, [ATCAKitType](#) kitiface)
Change the address of the selected device.
- ATCA_STATUS [releaseATCAIface](#) ([ATCAIface](#) ca_iface)
Instruct the HAL driver to release any resources associated with this interface.
- void [deleteATCAIface](#) ([ATCAIface](#) *ca_iface)
Instruct the HAL driver to release any resources associated with this interface, then delete the object.
- [ATCADeviceType](#) [iface_get_device_type_by_name](#) (const char *name)
Get the ATCADeviceType for a string that looks like a part number.

24.48.1 Detailed Description

Microchip CryptoAuthLib hardware interface object.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.49 atca_iface.h File Reference

Microchip Crypto Auth hardware interface object.

```
#include <stdint.h>
#include <stddef.h>
#include "atca_devtypes.h"
#include "atca_status.h"
#include "atca_config.h"
```

Data Structures

- struct [ATCAIfaceCfg](#)
- struct [ATCAHAL_t](#)
HAL Driver Structure.
- struct [atca_iface](#)
[atca_iface](#) is the context structure for a configured interface

Macros

- #define [ATCA_IFACECFG_NAME\(x\)](#) (x)
- #define [ATCA_IFACECFG_I2C_ADDRESS\(c\)](#) (c)->[cfg.atcai2c.address](#)
- #define [ATCA_IFACECFG_I2C_BAUD\(c\)](#) (c)->[cfg.atcai2c.baud](#)
- #define [ATCA_IFACECFG_VALUE\(c, v\)](#) (c)->[cfg.v](#)

Typedefs

- typedef struct [atca_iface](#) * [ATCAIface](#)
- typedef struct [atca_iface](#) [atca_iface_t](#)
[atca_iface](#) is the context structure for a configured interface

Enumerations

- enum [ATCAIfaceType](#) {
 [ATCA_I2C_IFACE](#) = 0 , [ATCA_SWI_IFACE](#) = 1 , [ATCA_UART_IFACE](#) = 2 , [ATCA_SPI_IFACE](#) = 3 ,
 [ATCA_HID_IFACE](#) = 4 , [ATCA_KIT_IFACE](#) = 5 , [ATCA_CUSTOM_IFACE](#) = 6 , [ATCA_I2C_GPIO_IFACE](#) = 7 ,
 [ATCA_SWI_GPIO_IFACE](#) = 8 , [ATCA_SPI_GPIO_IFACE](#) = 9 , [ATCA_UNKNOWN_IFACE](#) = 0xFE }
- enum [ATCAKitType](#) {
 [ATCA_KIT_AUTO_IFACE](#) , [ATCA_KIT_I2C_IFACE](#) , [ATCA_KIT_SWI_IFACE](#) , [ATCA_KIT_SPI_IFACE](#) ,
 [ATCA_KIT_UNKNOWN_IFACE](#) }

Functions

- ATCA_STATUS [initATCAIface](#) ([ATCAIfaceCfg](#) *cfg, [ATCAIface](#) ca_iface)
Initializer for ATCAIface objects.
- ATCA_STATUS [releaseATCAIface](#) ([ATCAIface](#) ca_iface)
Instruct the HAL driver to release any resources associated with this interface.
- ATCA_STATUS [atinit](#) ([ATCAIface](#) ca_iface)
Performs the HAL initialization by calling intermediate HAL wrapper function. If using the basic API, the [atcab_init\(\)](#) function should be called instead.
- ATCA_STATUS [atsend](#) ([ATCAIface](#) ca_iface, uint8_t word_address, uint8_t *txdata, int txlength)
Sends the data to the device by calling intermediate HAL wrapper function.
- ATCA_STATUS [atreceive](#) ([ATCAIface](#) ca_iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxlength)
Receives data from the device by calling intermediate HAL wrapper function.
- ATCA_STATUS [atcontrol](#) ([ATCAIface](#) ca_iface, uint8_t option, void *param, size_t paramlen)
Perform control operations with the underlying hal driver.
- ATCA_STATUS [atwake](#) ([ATCAIface](#) ca_iface)
Wakes up the device by calling intermediate HAL wrapper function. The [atcab_wakeup\(\)](#) function should be used instead.
- ATCA_STATUS [atidle](#) ([ATCAIface](#) ca_iface)
Puts the device into idle state by calling intermediate HAL wrapper function. The [atcab_idle\(\)](#) function should be used instead.
- ATCA_STATUS [atsleep](#) ([ATCAIface](#) ca_iface)
Puts the device into sleep state by calling intermediate HAL wrapper function. The [atcab_sleep\(\)](#) function should be used instead.
- [ATCAIfaceCfg](#) * [atgetifacecfg](#) ([ATCAIface](#) ca_iface)
Returns the logical interface configuration for the device.
- void * [atgetifacehaldat](#) ([ATCAIface](#) ca_iface)
Returns the HAL data pointer for the device.
- ATCA_STATUS [ifacecfg_set_address](#) ([ATCAIfaceCfg](#) *cfg, uint8_t address, ATCAKitType kitiface)
Change the address of the selected device.
- uint8_t [ifacecfg_get_address](#) ([ATCAIfaceCfg](#) *cfg)
Retrieves the device address given an interface configuration.
- bool [ifacetype_is_kit](#) ([ATCAIfaceType](#) iface_type)
Check if the given interface is a "kit protocol" one.
- bool [atca_iface_is_kit](#) ([ATCAIface](#) ca_iface)
Check if the given interface is configured as a "kit protocol" one where transactions are atomic.
- bool [atca_iface_is_swi](#) ([ATCAIface](#) ca_iface)
Check if the given interface is configured as a SWI.
- int [atca_iface_get_retries](#) ([ATCAIface](#) ca_iface)
Retrive the number of retries for a configured interface.
- uint16_t [atca_iface_get_wake_delay](#) ([ATCAIface](#) ca_iface)
Retrive the wake/retry delay for a configured interface/device.
- ATCADeviceType [iface_get_device_type_by_name](#) (const char *name)
Get the ATCADeviceType for a string that looks like a part number.

Variables

-

```
struct {  
    uint8_t address  
    uint8_t bus  
    uint32_t baud  
} atcai2c
```

-

```
struct {  
    uint8_t address  
    uint8_t bus  
} atcaswi
```

-

```
struct {  
    uint8_t bus  
    uint8_t select_pin  
    uint32_t baud  
} atcaspi
```

-

```
struct {  
    ATCAKitType dev_interface  
    uint8_t dev_identity  
    uint8_t port  
    uint32_t baud  
    uint8_t wordsize  
    uint8_t parity  
    uint8_t stopbits  
} atcauart
```

-

```
struct {  
    int idx  
    ATCAKitType dev_interface  
    uint8_t dev_identity  
    uint32_t vid  
    uint32_t pid  
    uint32_t packetsize  
} atcahid
```

-

```
struct {  
    ATCAKitType dev_interface  
    uint8_t dev_identity  
    uint32_t flags  
} atcakit
```

-

```
struct {  
    ATCA_STATUS(* halinit)(void *hal, void *cfg)  
    ATCA_STATUS(* halpostinit)(void *iface)  
    ATCA_STATUS(* halsend)(void *iface, uint8_t  
        word_address, uint8_t *txdata,
```

```

    int txlength)
    ATCA_STATUS(* halreceive )(void *iface, uint8_t
        word_address, uint8_t *rxdata,
            uint16_t *rxlength)
        ATCA_STATUS(* halwake )(void *iface)
        ATCA_STATUS(* halidle )(void *iface)
        ATCA_STATUS(* halsleep )(void *iface)
        ATCA_STATUS(* halrelease )(void *hal_data)
    } atcacustom

```

24.49.1 Detailed Description

Microchip Crypto Auth hardware interface object.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.49.2 Variable Documentation

24.49.2.1 address

```
uint8_t address
```

Device address - the upper 7 bits are the I2c address bits

24.50 atca_platform.h File Reference

Configure the platform interfaces for cryptoauthlib.

```

#include <stddef.h>
#include <string.h>
#include "atca_config.h"

```

Macros

- #define **hal_memset_s** [atcab_memset_s](#)

Functions

- void * **hal_malloc** (size_t size)
- void **hal_free** (void *ptr)
- const char * **lib_strcasestr** (const char *haystack, const char *needle)
Search for a substring in a case insensitive format.

24.50.1 Detailed Description

Configure the platform interfaces for cryptoauthlib.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.51 atca_status.h File Reference

Microchip Crypto Auth status codes.

```
#include <stdint.h>
#include "atca_compiler.h"
```

Macros

- #define [ATCA_SUCCESS](#) (0)
- #define [ATCA_CONFIG_ZONE_LOCKED](#) (0x01)
- #define [ATCA_DATA_ZONE_LOCKED](#) (0x02)
- #define [ATCA_WAKE_FAILED](#) (-48)
- #define [ATCA_CHECKMAC_VERIFY_FAILED](#) (-47)
- #define [ATCA_PARSE_ERROR](#) (-46)
- #define [ATCA_STATUS_CRC](#) (-44)
- #define [ATCA_STATUS_UNKNOWN](#) (-43)
- #define [ATCA_STATUS_ECC](#) (-42)
- #define [ATCA_STATUS_SELFTEST_ERROR](#) (-41)
- #define [ATCA_FUNC_FAIL](#) (-32)
- #define [ATCA_GEN_FAIL](#) (-31)
- #define [ATCA_BAD_PARAM](#) (-30)
- #define [ATCA_INVALID_ID](#) (-29)
- #define [ATCA_INVALID_SIZE](#) (-28)
- #define [ATCA_RX_CRC_ERROR](#) (-27)
- #define [ATCA_RX_FAIL](#) (-26)
- #define [ATCA_RX_NO_RESPONSE](#) (-25)
- #define [ATCA_RESYNC_WITH_WAKEUP](#) (-24)
- #define [ATCA_PARITY_ERROR](#) (-23)
- #define [ATCA_TX_TIMEOUT](#) (-22)
- #define [ATCA_RX_TIMEOUT](#) (-21)
- #define [ATCA_TOO_MANY_COMM_RETRIES](#) (-20)
- #define [ATCA_SMALL_BUFFER](#) (-19)
- #define [ATCA_COMM_FAIL](#) (-16)
- #define [ATCA_TIMEOUT](#) (-15)
- #define [ATCA_BAD_OPCODE](#) (-14)
- #define [ATCA_WAKE_SUCCESS](#) (-13)
- #define [ATCA_EXECUTION_ERROR](#) (-12)
- #define [ATCA_UNIMPLEMENTED](#) (-11)
- #define [ATCA_ASSERT_FAILURE](#) (-10)
- #define [ATCA_TX_FAIL](#) (-9)
- #define [ATCA_NOT_LOCKED](#) (-8)
- #define [ATCA_NO_DEVICES](#) (-7)
- #define [ATCA_HEALTH_TEST_ERROR](#) (-6)
- #define [ATCA_ALLOC_FAILURE](#) (-5)
- #define [ATCA_USE_FLAGS_CONSUMED](#) (-4)
- #define [ATCA_NOT_INITIALIZED](#) (-3)
- #define [ATCA_STATUS_AUTH_BIT](#) 0x40u
- #define [ATCA_STATUS_AUTH_BIT_COMPLEMENT](#) ~(ATCA_STATUS_AUTH_BIT & 0xffu)

Typedefs

- typedef int **ATCA_STATUS**

24.51.1 Detailed Description

Microchip Crypto Auth status codes.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.51.2 Macro Definition Documentation

24.51.2.1 ATCA_ALLOC_FAILURE

```
#define ATCA_ALLOC_FAILURE (-5)
```

STATUS (0xFB): Couldn't allocate required memory

24.51.2.2 ATCA_ASSERT_FAILURE

```
#define ATCA_ASSERT_FAILURE (-10)
```

STATUS (0xF6): Code failed run-time consistency check

24.51.2.3 ATCA_BAD_OPCODE

```
#define ATCA_BAD_OPCODE (-14)
```

STATUS (0xF2): opcode is not supported by the device

24.51.2.4 ATCA_BAD_PARAM

```
#define ATCA_BAD_PARAM (-30)
```

STATUS (0xE2): bad argument (out of range, null pointer, etc.)

24.51.2.5 ATCA_CHECKMAC_VERIFY_FAILED

```
#define ATCA_CHECKMAC_VERIFY_FAILED (-47)
```

STATUS (0xD1): response status byte indicates CheckMac failure(status byte = 0x01)

24.51.2.6 ATCA_COMM_FAIL

```
#define ATCA_COMM_FAIL (-16)
```

STATUS (0xF0): Communication with device failed. Same as in hardware dependent modules.

24.51.2.7 ATCA_EXECUTION_ERROR

```
#define ATCA_EXECUTION_ERROR (-12)
```

STATUS (0xF4): chip was in a state where it could not execute the command, response status byte indicates command execution error (status byte = 0x0F)

24.51.2.8 ATCA_FUNC_FAIL

```
#define ATCA_FUNC_FAIL (-32)
```

STATUS (0xE0): Function could not execute due to incorrect condition / state.

24.51.2.9 ATCA_GEN_FAIL

```
#define ATCA_GEN_FAIL (-31)
```

STATUS (0xE1): unspecified error

24.51.2.10 ATCA_HEALTH_TEST_ERROR

```
#define ATCA_HEALTH_TEST_ERROR (-6)
```

STATUS (0xFA): random number generator health test error

24.51.2.11 ATCA_INVALID_ID

```
#define ATCA_INVALID_ID (-29)
```

STATUS (0xE3): invalid device id, id not set

24.51.2.12 ATCA_INVALID_SIZE

```
#define ATCA_INVALID_SIZE (-28)
```

STATUS (0xE4): Count value is out of range or greater than buffer size.

24.51.2.13 ATCA_NO_DEVICES

```
#define ATCA_NO_DEVICES (-7)
```

STATUS (0xF9): For protocols that support device discovery (kit protocol), no devices were found

24.51.2.14 ATCA_NOT_INITIALIZED

```
#define ATCA_NOT_INITIALIZED (-3)
```

STATUS (0xFD): The library has not been initialized so the command could not be executed

24.51.2.15 ATCA_NOT_LOCKED

```
#define ATCA_NOT_LOCKED (-8)
```

STATUS (0xF8): required zone was not locked

24.51.2.16 ATCA_PARITY_ERROR

```
#define ATCA_PARITY_ERROR (-23)
```

STATUS (0xE9): for protocols needing parity

24.51.2.17 ATCA_PARSE_ERROR

```
#define ATCA_PARSE_ERROR (-46)
```

STATUS (0xD2): response status byte indicates parsing error(status byte = 0x03)

24.51.2.18 ATCA_RESYNC_WITH_WAKEUP

```
#define ATCA_RESYNC_WITH_WAKEUP (-24)
```

STATUS (0xE8): Re-synchronization succeeded, but only after generating a Wake-up

24.51.2.19 ATCA_RX_CRC_ERROR

```
#define ATCA_RX_CRC_ERROR (-27)
```

STATUS (0xE5): CRC error in data received from device

24.51.2.20 ATCA_RX_FAIL

```
#define ATCA_RX_FAIL (-26)
```

STATUS (0xE6): Timed out while waiting for response. Number of bytes received is > 0.

24.51.2.21 ATCA_RX_NO_RESPONSE

```
#define ATCA_RX_NO_RESPONSE (-25)
```

STATUS (0xE7): Not an error while the Command layer is polling for a command response.

24.51.2.22 ATCA_RX_TIMEOUT

```
#define ATCA_RX_TIMEOUT (-21)
```

STATUS (0xEB): for Microchip PHY protocol, timeout on receipt waiting for master

24.51.2.23 ATCA_SMALL_BUFFER

```
#define ATCA_SMALL_BUFFER (-19)
```

STATUS (0xED): Supplied buffer is too small for data required

24.51.2.24 ATCA_STATUS_CRC

```
#define ATCA_STATUS_CRC (-44)
```

STATUS (0xD4): response status byte indicates DEVICE did not receive data properly(status byte = 0xFF)

24.51.2.25 ATCA_STATUS_ECC

```
#define ATCA_STATUS_ECC (-42)
```

STATUS (0xD6): response status byte is ECC fault(status byte = 0x05)

24.51.2.26 ATCA_STATUS_SELFTEST_ERROR

```
#define ATCA_STATUS_SELFTEST_ERROR (-41)
```

STATUS (0xD7): response status byte is Self Test Error, chip in failure mode (status byte = 0x07)

24.51.2.27 ATCA_STATUS_UNKNOWN

```
#define ATCA_STATUS_UNKNOWN (-43)
```

STATUS (0xD5): response status byte is unknown

24.51.2.28 ATCA_SUCCESS

```
#define ATCA_SUCCESS (0)
```

STATUS (0x00): Function Successful

24.51.2.29 ATCA_TIMEOUT

```
#define ATCA_TIMEOUT (-15)
```

STATUS (0xF1): Timed out while waiting for response. Number of bytes received is 0.

24.51.2.30 ATCA_TOO_MANY_COMM_RETRIES

```
#define ATCA_TOO_MANY_COMM_RETRIES (-20)
```

STATUS (0xEC): Device did not respond too many times during a transmission. Could indicate no device present.

24.51.2.31 ATCA_TX_FAIL

```
#define ATCA_TX_FAIL (-9)
```

STATUS (0xF7): Failed to write

24.51.2.32 ATCA_TX_TIMEOUT

```
#define ATCA_TX_TIMEOUT (-22)
```

STATUS (0xEA): for Microchip PHY protocol, timeout on transmission waiting for master

24.51.2.33 ATCA_UNIMPLEMENTED

```
#define ATCA_UNIMPLEMENTED (-11)
```

STATUS (0xF5): Function or some element of it hasn't been implemented yet

24.51.2.34 ATCA_USE_FLAGS_CONSUMED

```
#define ATCA_USE_FLAGS_CONSUMED (-4)
```

STATUS (0xFC): Use flags on the device indicates its consumed fully

24.51.2.35 ATCA_WAKE_FAILED

```
#define ATCA_WAKE_FAILED (-48)
```

STATUS (0xD0): response status byte indicates CheckMac failure(status byte = 0x01)

24.51.2.36 ATCA_WAKE_SUCCESS

```
#define ATCA_WAKE_SUCCESS (-13)
```

STATUS (0xF3): received proper wake token

24.52 atca_utils_sizes.c File Reference

API to Return structure sizes of cryptoauthlib structures.

```
#include "cryptoauthlib.h"
#include "cal_internal.h"
#include "atcacert/atcacert_check_config.h"
#include "atcacert/atcacert_date.h"
#include "atcacert/atcacert_def.h"
#include "host/atca_host.h"
```

Macros

- `#define SIZE_OF_API_T(x) size_t x ## _size(void); size_t x ## _size(void) { return sizeof(x); }`
- `#define SIZE_OF_API_S(x) size_t x ## _size(void); size_t x ## _size(void) { return sizeof(struct x); }`

Functions

- `size_t atcacert_tm_utc_t_size (void)`
- `size_t atcacert_date_format_t_size (void)`
- `size_t atcacert_cert_type_t_size (void)`
- `size_t atcacert_cert_sn_src_t_size (void)`
- `size_t atcacert_device_zone_t_size (void)`
- `size_t atcacert_std_cert_element_t_size (void)`
- `size_t atcacert_device_loc_t_size (void)`
- `size_t atcacert_cert_loc_t_size (void)`
- `size_t atcacert_cert_element_t_size (void)`
- `size_t atcacert_def_t_size (void)`
- `size_t atcacert_build_state_t_size (void)`
- `size_t atca_temp_key_t_size (void)`
- `size_t atca_include_data_in_out_size (void)`
- `size_t atca_nonce_in_out_t_size (void)`
- `size_t atca_io_decrypt_in_out_t_size (void)`
- `size_t atca_verify_mac_in_out_t_size (void)`
- `size_t atca_secureboot_enc_in_out_t_size (void)`
- `size_t atca_secureboot_mac_in_out_t_size (void)`
- `size_t atca_mac_in_out_t_size (void)`
- `size_t atca_hmac_in_out_size (void)`
- `size_t atca_gen_dig_in_out_t_size (void)`
- `size_t atca_write_mac_in_out_t_size (void)`
- `size_t atca_derive_key_in_out_size (void)`
- `size_t atca_derive_key_mac_in_out_size (void)`
- `size_t atca_decrypt_in_out_size (void)`
- `size_t atca_check_mac_in_out_t_size (void)`
- `size_t atca_verify_in_out_t_size (void)`
- `size_t atca_gen_key_in_out_t_size (void)`
- `size_t atca_sign_internal_in_out_t_size (void)`
- `size_t bool_size (void)`
- `size_t ATCAPacket_size (void)`
- `size_t atca_device_size (void)`
- `size_t ATCADeviceType_size (void)`

- `size_t ATCAIfaceType_size` (void)
- `size_t ATCAIfaceCfg_size` (void)
- `size_t atca_iface_size` (void)
- `size_t ATCA_STATUS_size` (void)
- `size_t atcac_sha1_ctx_size` (void)
- `size_t atcac_sha1_ctx_t_size` (void)
- `size_t atcac_sha2_256_ctx_size` (void)
- `size_t atcac_sha2_256_ctx_t_size` (void)
- `size_t atcac_hmac_ctx_size` (void)
- `size_t atcac_hmac_ctx_t_size` (void)

24.52.1 Detailed Description

API to Return structure sizes of cryptoauthlib structures.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.53 atca_version.h File Reference

Microchip CryptoAuth Library Version.

Macros

- `#define ATCA_LIBRARY_VERSION_DATE` "20250905"
- `#define ATCA_LIBRARY_VERSION_MAJOR` 3
- `#define ATCA_LIBRARY_VERSION_MINOR` 7
- `#define ATCA_LIBRARY_VERSION_BUILD` 9

24.53.1 Detailed Description

Microchip CryptoAuth Library Version.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.54 atcacert.h File Reference

Declarations common to all atcacert code.

```
#include <stddef.h>
#include <stdint.h>
#include "atcacert_check_config.h"
#include "atca_status.h"
```

Macros

- #define **FORMAT_VERSION_0** 0u
- #define **FORMAT_VERSION_1** 1u
- #define **FORMAT_VERSION_2** 2u
- #define **FALSE** (0)
- #define **TRUE** (1)
- #define **ATCACERT_E_SUCCESS** ATCA_SUCCESS
- #define **ATCACERT_E_ERROR** ATCA_GEN_FAIL
- #define **ATCACERT_E_BAD_PARAMS** ATCA_BAD_PARAM
- #define **ATCACERT_E_BUFFER_TOO_SMALL** ATCA_SMALL_BUFFER
- #define **ATCACERT_E_UNIMPLEMENTED** ATCA_UNIMPLEMENTED
- #define **ATCACERT_E_DECODING_ERROR** 4
- #define **ATCACERT_E_INVALID_DATE** 5
- #define **ATCACERT_E_UNEXPECTED_ELEM_SIZE** 7
- #define **ATCACERT_E_ELEM_MISSING** 8
- #define **ATCACERT_E_ELEM_OUT_OF_BOUNDS** 9
- #define **ATCACERT_E_BAD_CERT** 10
- #define **ATCACERT_E_WRONG_CERT_DEF** 11
- #define **ATCACERT_E_VERIFY_FAILED** 12
- #define **ATCACERT_E_INVALID_TRANSFORM** 13

24.54.1 Detailed Description

Declarations common to all atcacert code.

These are common definitions used by all the atcacert code.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.55 atcacert_check_config.h File Reference

Configuration check and defaults for the atcacert module.

```
#include "cryptoauthlib.h"
#include "crypto/atca_crypto_sw.h"
```

Macros

- #define **HOSTLIB_CERT_EN** DEFAULT_DISABLED
- #define **ATCACERT_INTEGRATION_EN** HOSTLIB_CERT_EN
- #define **ATCACERT_FULLSTOREDCERT_EN** DEFAULT_ENABLED
- #define **ATCACERT_COMPCERT_EN** DEFAULT_ENABLED
- #define **ATCACERT_EN** (ATCACERT_FULLSTOREDCERT_EN || ATCACERT_COMPCERT_EN)
- #define **ATCACERT_HW_CHALLENGE_EN** (ATCAB_RANDOM_EN && (ATCA_ECC_SUPPORT || ATCA_TA_SUPPORT))
- #define **ATCACERT_HW_VERIFY_EN** (ATCAB_VERIFY_EXTERN_EN && (ATCA_ECC_SUPPORT || ATCA_TA_SUPPORT))
- #define **ATCACERT_DATEFMT_ISO_EN** DEFAULT_ENABLED
- #define **ATCACERT_DATEFMT_UTC_EN** DEFAULT_ENABLED
- #define **ATCACERT_DATEFMT_POSIX_EN** DEFAULT_ENABLED
- #define **ATCACERT_DATEFMT_GEN_EN** DEFAULT_ENABLED

24.55.1 Detailed Description

Configuration check and defaults for the atcacert module.

Copyright

(c) 2015-2022 Microchip Technology Inc. and its subsidiaries.

24.56 atcacert_client.c File Reference

Client side cert i/o methods. These declarations deal with the client-side, the node being authenticated, of the authentication process. It is assumed the client has an ECC CryptoAuthentication device (e.g. ATECC508A) and the certificates are stored on that device.

```
#include <limits.h>
#include <stdlib.h>
#include "atcacert_client.h"
#include "atcacert_der.h"
#include "atcacert_pem.h"
#include "cryptoauthlib.h"
#include "calib/calib_basic.h"
#include "talib/talib_basic.h"
#include "talib/talib_internal.h"
```

24.56.1 Detailed Description

Client side cert i/o methods. These declarations deal with the client-side, the node being authenticated, of the authentication process. It is assumed the client has an ECC CryptoAuthentication device (e.g. ATECC508A) and the certificates are stored on that device.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.57 atcacert_client.h File Reference

Client side cert i/o methods. These declarations deal with the client-side, the node being authenticated, of the authentication process. It is assumed the client has an ECC CryptoAuthentication device (e.g. ATECC508A) and the certificates are stored on that device.

```
#include <stddef.h>
#include <stdint.h>
#include "atcacert_def.h"
```

Functions

- ATCA_STATUS [atcacert_read_cert](#) (const [atcacert_def_t](#) *cert_def, const [cal_buffer](#) *ca_public_key, uint8_t *cert, size_t *cert_size)
Reads the certificate specified by the certificate definition from the ATECC508A device.
- ATCA_STATUS [atcacert_read_cert_ext](#) (ATCADevice device, const [atcacert_def_t](#) *cert_def, const [cal_buffer](#) *ca_public_key, uint8_t *cert, size_t *cert_size)
Reads the certificate specified by the certificate definition from the ATECC508A device.
- ATCA_STATUS [atcacert_write_cert](#) (const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size)
Take a full certificate and write it to the ATECC508A device according to the certificate definition.
- ATCA_STATUS [atcacert_write_cert_ext](#) (ATCADevice device, const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size)
Take a full certificate and write it to the ATECC508A device according to the certificate definition.
- ATCA_STATUS [atcacert_read_cert_size](#) (const [atcacert_def_t](#) *cert_def, size_t *cert_size)
Return the actual certificate size in bytes for a given cert def. Certificate can be variable size, so this gives the absolute buffer size when reading the certificates.
- ATCA_STATUS [atcacert_read_cert_size_ext](#) (ATCADevice device, const [atcacert_def_t](#) *cert_def, size_t *cert_size)
Return the actual certificate size in bytes for a given cert def. Certificate can be variable size, so this gives the absolute buffer size when reading the certificates.

24.57.1 Detailed Description

Client side cert i/o methods. These declarations deal with the client-side, the node being authenticated, of the authentication process. It is assumed the client has an ECC CryptoAuthentication device (e.g. ATECC508A) and the certificates are stored on that device.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.57.2 Function Documentation

24.57.2.1 atcacert_read_cert()

```
ATCA_STATUS atcacert_read_cert (  
    const atcacert\_def\_t * cert_def,  
    const cal\_buffer * ca_public_key,  
    uint8_t * cert,  
    size_t * cert_size )
```

Reads the certificate specified by the certificate definition from the ATECC508A device.

This process involves reading the dynamic cert data from the device and combining it with the template found in the certificate definition.

Parameters

in	<i>cert_def</i>	Certificate definition describing where to find the dynamic certificate information on the device and how to incorporate it into the template.
in	<i>ca_public_key</i>	Buffer pointing to the ECC P256/P384/P521 public key of the certificate authority that signed this certificate. Formatted as X and Y integers concatenated together. Set to NULL if the authority key id is not needed, set properly in the cert_def template, or stored on the device as specified in the cert_def cert_elements.
out	<i>cert</i>	Buffer to received the certificate.
in, out	<i>cert_size</i>	As input, the size of the cert buffer in bytes. As output, the size of the certificate returned in cert in bytes.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

24.57.2.2 atcacert_read_cert_ext()

```
ATCA_STATUS atcacert_read_cert_ext (
    ATCADevice device,
    const atcacert_def_t * cert_def,
    const cal_buffer * ca_public_key,
    uint8_t * cert,
    size_t * cert_size )
```

Reads the certificate specified by the certificate definition from the ATECC508A device.

This process involves reading the dynamic cert data from the device and combining it with the template found in the certificate definition.

Parameters

in	<i>device</i>	Device context pointer
in	<i>cert_def</i>	Certificate definition describing where to find the dynamic certificate information on the device and how to incorporate it into the template.
in	<i>ca_public_key</i>	Buffer pointing to the public key of the certificate authority that signed this certificate. Formatted as X and Y integers concatenated together. Set to NULL if the authority key id is not needed, set properly in the cert_def template, or stored on the device as specified in the cert_def cert_elements.
out	<i>cert</i>	Buffer to received the certificate.
in, out	<i>cert_size</i>	As input, the size of the cert buffer in bytes. As output, the size of the certificate returned in cert in bytes.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

24.57.2.3 atcacert_read_cert_size()

```
ATCA_STATUS atcacert_read_cert_size (
    const atcacert_def_t * cert_def,
    size_t * cert_size )
```

Return the actual certificate size in bytes for a given cert def. Certificate can be variable size, so this gives the absolute buffer size when reading the certificates.

Parameters

in	<i>cert_def</i>	Certificate definition to find a max size for.
out	<i>cert_size</i>	Certificate size will be returned here in bytes.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

24.57.2.4 atcacert_read_cert_size_ext()

```
ATCA_STATUS atcacert_read_cert_size_ext (
    ATCADevice device,
    const atcacert_def_t * cert_def,
    size_t * cert_size )
```

Return the actual certificate size in bytes for a given cert def. Certificate can be variable size, so this gives the absolute buffer size when reading the certificates.

Parameters

in	<i>device</i>	Device context
in	<i>cert_def</i>	Certificate definition to find a max size for.
out	<i>cert_size</i>	Certificate size will be returned here in bytes.

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

24.57.2.5 atcacert_write_cert()

```
ATCA_STATUS atcacert_write_cert (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size )
```

Take a full certificate and write it to the ATECC508A device according to the certificate definition.

Parameters

in	<i>cert_def</i>	Certificate definition describing where the dynamic certificate information is and how to store it on the device.
in	<i>cert</i>	Full certificate to be stored.
in	<i>cert_size</i>	Size of the full certificate in bytes.
in	<i>device</i>	Device context

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

24.57.2.6 atcacert_write_cert_ext()

```
ATCA_STATUS atcacert_write_cert_ext (
    ATCADevice device,
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size )
```

Take a full certificate and write it to the ATECC508A device according to the certificate definition.

Parameters

in	<i>device</i>	Device context
in	<i>cert_def</i>	Certificate definition describing where the dynamic certificate information is and how to store it on the device.
in	<i>cert</i>	Full certificate to be stored.
in	<i>cert_size</i>	Size of the full certificate in bytes.
in	<i>device</i>	Device context

Returns

ATCACERT_E_SUCCESS on success, otherwise an error code.

24.58 atcacert_date.c File Reference

Date handling with regard to certificates.

```
#include <string.h>
#include <limits.h>
#include "atcacert_date.h"
#include "atca_compiler.h"
```

24.58.1 Detailed Description

Date handling with regard to certificates.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.59 atcacert_date.h File Reference

Declarations for date handling with regard to certificates.

```
#include <stddef.h>
#include "atcacert.h"
```

Data Structures

- struct [atcacert_tm_utc_s](#)

Macros

- #define [DATEFMT_ISO8601_SEP](#) (0U)
ISO8601 full date YYYY-MM-DDThh:mm:ssZ.
- #define [DATEFMT_RFC5280.UTC](#) (1U)
RFC 5280 (X.509) 4.1.2.5.1 UTCTime format YYMMDDhhmmssZ.
- #define [DATEFMT_POSIX_UINT32_BE](#) (2U)
POSIX (aka UNIX) date format. Seconds since Jan 1, 1970. 32 bit unsigned integer, big endian.
- #define [DATEFMT_POSIX_UINT32_LE](#) (3U)
POSIX (aka UNIX) date format. Seconds since Jan 1, 1970. 32 bit unsigned integer, little endian.
- #define [DATEFMT_RFC5280_GEN](#) (4U)
RFC 5280 (X.509) 4.1.2.5.2 GeneralizedTime format YYYYMMDDhhmmssZ.
- #define [DATEFMT_INVALID](#) (0xFFU)
- #define [DATEFMT_ISO8601_SEP_SIZE](#) (20)
- #define [DATEFMT_RFC5280.UTC_SIZE](#) (13)
- #define [DATEFMT_POSIX_UINT32_BE_SIZE](#) (4)
- #define [DATEFMT_POSIX_UINT32_LE_SIZE](#) (4)
- #define [DATEFMT_RFC5280_GEN_SIZE](#) (15)
- #define [DATEFMT_MAX_SIZE](#) DATEFMT_ISO8601_SEP_SIZE
- #define [ATCACERT_DATE_FORMAT_SIZES_COUNT](#) 5
- #define [atcacert_date_enc_posix_uint32_be](#) atcacert_date_enc_posix_be
- #define [atcacert_date_dec_posix_uint32_be](#) atcacert_date_dec_posix_be
- #define [atcacert_date_enc_posix_uint32_le](#) atcacert_date_enc_posix_le
- #define [atcacert_date_dec_posix_uint32_le](#) atcacert_date_dec_posix_le

Typedefs

- typedef struct [atcacert_tm_utc_s](#) [atcacert_tm_utc_t](#)
- typedef uint8_t [atcacert_date_format_t](#)

Functions

- ATCA_STATUS [atcacert_date_enc](#) (atcacert_date_format_t format, const [atcacert_tm_utc_t](#) *timestamp, uint8_t *formatted_date, size_t *formatted_date_size)
Format a timestamp according to the format type.
- ATCA_STATUS [atcacert_date_dec](#) (atcacert_date_format_t format, const uint8_t *formatted_date, size_t formatted_date_size, [atcacert_tm_utc_t](#) *timestamp)
Parse a formatted timestamp according to the specified format.
- ATCA_STATUS [atcacert_date_enc_compcert](#) (const [atcacert_tm_utc_t](#) *issue_date, uint8_t expire_years, uint8_t enc_dates[3])
Encode the issue and expire dates in the format used by the compressed certificate.
- ATCA_STATUS [atcacert_date_enc_compcert_ext](#) (const [atcacert_tm_utc_t](#) *issue_date, uint8_t expire_years, uint8_t comp_cert[ATCACERT_COMP_CERT_MAX_SIZE])
Encode the issue and expire dates in the format used by the compressed certificate.
- ATCA_STATUS [atcacert_date_dec_compcert](#) (const uint8_t enc_dates[3], atcacert_date_format_t expire_date_format, [atcacert_tm_utc_t](#) *issue_date, [atcacert_tm_utc_t](#) *expire_date)
Decode the issue and expire dates from the format used by the compressed certificate.
- ATCA_STATUS [atcacert_date_dec_compcert_ext](#) (const uint8_t comp_cert[ATCACERT_COMP_CERT_MAX_SIZE], atcacert_date_format_t expire_date_format, [atcacert_tm_utc_t](#) *issue_date, [atcacert_tm_utc_t](#) *expire_date)
Decode the issue and expire dates from the format used by the compressed certificate.
- atcacert_date_format_t [atcacert_date_from_asn1_tag](#) (const uint8_t tag)
Convert the asn1 tag for the supported time formats into the local time format.
- ATCA_STATUS [atcacert_date_get_max_date](#) (atcacert_date_format_t format, [atcacert_tm_utc_t](#) *timestamp)
Return the maximum date available for the given format.
- ATCA_STATUS [atcacert_date_enc_iso8601_sep](#) (const [atcacert_tm_utc_t](#) *timestamp, uint8_t formatted_date[(20)])
- ATCA_STATUS [atcacert_date_dec_iso8601_sep](#) (const uint8_t formatted_date[(20)], [atcacert_tm_utc_t](#) *timestamp)
- ATCA_STATUS [atcacert_date_enc_rfc5280_utc](#) (const [atcacert_tm_utc_t](#) *timestamp, uint8_t formatted_date[(13)])
- ATCA_STATUS [atcacert_date_dec_rfc5280_utc](#) (const uint8_t formatted_date[(13)], [atcacert_tm_utc_t](#) *timestamp)
- ATCA_STATUS [atcacert_date_enc_rfc5280_gen](#) (const [atcacert_tm_utc_t](#) *timestamp, uint8_t formatted_date[(15)])
- ATCA_STATUS [atcacert_date_dec_rfc5280_gen](#) (const uint8_t formatted_date[(15)], [atcacert_tm_utc_t](#) *timestamp)
- ATCA_STATUS [atcacert_date_enc_posix_be](#) (const [atcacert_tm_utc_t](#) *timestamp, uint8_t formatted_date[(4)])
- ATCA_STATUS [atcacert_date_dec_posix_be](#) (const uint8_t formatted_date[(4)], [atcacert_tm_utc_t](#) *timestamp)
- ATCA_STATUS [atcacert_date_enc_posix_le](#) (const [atcacert_tm_utc_t](#) *timestamp, uint8_t formatted_date[(4)])
- ATCA_STATUS [atcacert_date_dec_posix_le](#) (const uint8_t formatted_date[(4)], [atcacert_tm_utc_t](#) *timestamp)
- int [atcacert_date_cmp](#) (const [atcacert_tm_utc_t](#) *timestamp1, const [atcacert_tm_utc_t](#) *timestamp2)
Compare two dates.

Variables

- const size_t **ATCACERT_DATE_FORMAT_SIZES** [5]

24.59.1 Detailed Description

Declarations for date handling with regard to certificates.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.60 atcacert_def.c File Reference

Main certificate definition implementation.

```
#include "atcacert_def.h"
#include "crypto/atca_crypto_sw.h"
#include "crypto/atca_crypto_sw_sha1.h"
#include "crypto/atca_crypto_sw_sha2.h"
#include "atcacert_der.h"
#include "atcacert_date.h"
#include <string.h>
#include "atca_helpers.h"
#include "cal_buffer.h"
```

24.60.1 Detailed Description

Main certificate definition implementation.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.61 atcacert_def.h File Reference

Declarations for certificates related to ECC CryptoAuthentication devices. These are the definitions required to define a certificate and its various elements with regards to the CryptoAuthentication ECC devices.

```
#include <stddef.h>
#include <stdint.h>
#include "atca_compiler.h"
#include "atcacert.h"
#include "atcacert_date.h"
#include "atca_helpers.h"
#include "crypto/atca_crypto_sw.h"
#include "cal_buffer.h"
```

Data Structures

- struct [atcacert_device_loc_s](#)
- struct [atcacert_cert_loc_s](#)
- struct [atcacert_cert_element_s](#)
- struct [atcacert_def_s](#)
- struct [atcacert_build_state_s](#)

Macros

- #define **ATCA_MAX_TRANSFORMS** 2
- #define **CA_DEV_SN_SIZE** 9u
- #define **CA2_DEV_SN_SIZE_PART_1** 4u
- #define **CA2_DEV_SN_SIZE_PART_2** 5u
- #define **CA_DEV_SN_CONFIG_ZONE_OFFSET** 0u
- #define **CA2_DEV_SN_CONFIG_ZONE_OFFSET_PART_1** 0u
- #define **CA2_DEV_SN_CONFIG_ZONE_OFFSET_PART_2** 8u
- #define **TA_DEV_SN_DEDICATED_DATA_ZONE_OFFSET** 0u

Typedefs

- typedef enum [atcacert_cert_type_e](#) [atcacert_cert_type_t](#)
- typedef enum [atcacert_cert_sn_src_e](#) [atcacert_cert_sn_src_t](#)
- typedef enum [atcacert_device_zone_e](#) [atcacert_device_zone_t](#)
- typedef enum [atcacert_transform_e](#) [atcacert_transform_t](#)
How to transform the data from the device to the certificate.
- typedef enum [atcacert_std_cert_element_e](#) [atcacert_std_cert_element_t](#)
- typedef struct ATCA_PACKED [atcacert_device_loc_s](#) [atcacert_device_loc_t](#)
- typedef struct ATCA_PACKED [atcacert_cert_loc_s](#) [atcacert_cert_loc_t](#)
- typedef struct ATCA_PACKED [atcacert_cert_element_s](#) [atcacert_cert_element_t](#)
- typedef struct [atcacert_def_s](#) [atcacert_def_t](#)
- typedef struct [atcacert_build_state_s](#) [atcacert_build_state_t](#)

Enumerations

- enum [atcacert_cert_type_e](#) { CERTTYPE_X509 , CERTTYPE_CUSTOM , CERTTYPE_X509_FULL_STORED }
- enum [atcacert_cert_sn_src_e](#) {
SNSRC_STORED = 0x0 , **SNSRC_STORED_DYNAMIC** = 0x7 , **SNSRC_DEVICE_SN** = 0x8 ,
SNSRC_SIGNER_ID = 0x9 ,
SNSRC_PUB_KEY_HASH = 0xA , **SNSRC_DEVICE_SN_HASH** = 0xB , **SNSRC_PUB_KEY_HASH_POS**
= 0xC , **SNSRC_DEVICE_SN_HASH_POS** = 0xD ,
SNSRC_PUB_KEY_HASH_RAW = 0xE , **SNSRC_DEVICE_SN_HASH_RAW** = 0xF }
- enum [atcacert_device_zone_e](#) {
DEVZONE_CONFIG = 0x00 , **DEVZONE_OTP** = 0x01 , **DEVZONE_DATA** = 0x02 , **DEVZONE_GENKEY** =
0x03 ,
DEVZONE_DEDICATED_DATA = 0x04 , **DEVZONE_NONE** = 0x07 }
- enum [atcacert_transform_e](#) {
TF_NONE , **TF_REVERSE** , **TF_BIN2HEX_UC** , **TF_BIN2HEX_LC** ,
TF_HEX2BIN_UC , **TF_HEX2BIN_LC** , **TF_BIN2HEX_SPACE_UC** , **TF_BIN2HEX_SPACE_LC** ,
TF_HEX2BIN_SPACE_UC , **TF_HEX2BIN_SPACE_LC** }
- *How to transform the data from the device to the certificate.*
- enum [atcacert_std_cert_element_e](#) {
STDCERT_PUBLIC_KEY , **STDCERT_SIGNATURE** , **STDCERT_ISSUE_DATE** , **STDCERT_EXPIRE_↵**
DATE ,
STDCERT_SIGNER_ID , **STDCERT_CERT_SN** , **STDCERT_AUTH_KEY_ID** , **STDCERT_SUBJ_KEY_ID** ,
STDCERT_NUM_ELEMENTS }

Functions

- ATCA_STATUS [atcacert_get_subject](#) (const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size, [cal_buffer](#) *cert_subj_buf)
Gets the subject name from a certificate.
- ATCA_STATUS [atcacert_get_subj_public_key](#) (const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size, [cal_buffer](#) *subj_public_key)
Gets the subject public key from a certificate.
- ATCA_STATUS [atcacert_get_subj_key_id](#) (const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size, uint8_t subj_key_id[20])
Gets the subject key ID from a certificate.
- ATCA_STATUS [atcacert_get_issuer](#) (const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size, uint8_t cert_issuer[128])
Gets the issuer name of a certificate.
- ATCA_STATUS [atcacert_get_issue_date](#) (const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size, [atcacert_tm_utc_t](#) *timestamp)
Gets the issue date from a certificate. Will be parsed according to the date format specified in the certificate definition.
- ATCA_STATUS [atcacert_get_expire_date](#) (const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size, [atcacert_tm_utc_t](#) *timestamp)
Gets the expire date from a certificate. Will be parsed according to the date format specified in the certificate definition.
- ATCA_STATUS [atcacert_get_cert_sn](#) (const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size, uint8_t *cert_sn, size_t *cert_sn_size)
Gets the certificate serial number from a certificate.
- ATCA_STATUS [atcacert_get_auth_key_id](#) (const [atcacert_def_t](#) *cert_def, const uint8_t *cert, size_t cert_size, uint8_t auth_key_id[20])
Gets the authority key ID from a certificate.

24.61.1 Detailed Description

Declarations for certificates related to ECC CryptoAuthentication devices. These are the definitions required to define a certificate and its various elements with regards to the CryptoAuthentication ECC devices.

Only the dynamic elements of a certificate (the parts of the certificate that change from device to device) are stored on the ATECC device. The definitions here describe the form of the certificate, and where the dynamic elements can be found both on the ATECC device itself and in the certificate template.

This also defines utility functions for working with the certificates and their definitions.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.62 atcacert_der.c File Reference

functions required to work with DER encoded data related to X.509 certificates.

```
#include "cryptoauthlib.h"
#include "atcacert_der.h"
#include <string.h>
```

24.62.1 Detailed Description

functions required to work with DER encoded data related to X.509 certificates.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.63 atcacert_der.h File Reference

function declarations required to work with DER encoded data related to X.509 certificates.

```
#include <stddef.h>
#include <stdint.h>
#include "atcacert.h"
```

Functions

- ATCA_STATUS [atcacert_der_enc_length](#) (size_t length, uint8_t *der_length, size_t *der_length_size)
Encode a length in DER format.
- ATCA_STATUS [atcacert_der_dec_length](#) (const uint8_t *der_length, size_t *der_length_size, size_t *length)
Decode a DER format length.
- ATCA_STATUS [atcacert_der_adjust_length](#) (uint8_t *der_length, size_t *der_length_size, int delta_length, size_t *new_length)
- ATCA_STATUS [atcacert_der_enc_integer](#) (const uint8_t *int_data, size_t int_data_size, uint8_t is_unsigned, uint8_t *der_int, size_t *der_int_size)
Encode an ASN.1 integer in DER format, including tag and length fields.
- ATCA_STATUS [atcacert_der_dec_integer](#) (const uint8_t *der_int, size_t *der_int_size, uint8_t *int_data, size_t *int_data_size)
Decode an ASN.1 DER encoded integer.
- ATCA_STATUS [atcacert_der_enc_ecdsa_sig_value](#) (const [cal_buffer](#) *raw_sig, uint8_t *der_sig, size_t *der_sig_size)
Formats a raw ECDSA P256 signature in the DER encoding found in X.509 certificates.
- ATCA_STATUS [atcacert_der_dec_ecdsa_sig_value](#) (const uint8_t *der_sig, size_t *der_sig_size, [cal_buffer](#) *raw_sig)
Parses an ECDSA P256/P384/P521 signature in the DER encoding as found in X.509 certificates.

24.63.1 Detailed Description

function declarations required to work with DER encoded data related to X.509 certificates.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.64 atcacert_host_hw.c File Reference

host side methods using CryptoAuth hardware

```
#include "atcacert_host_hw.h"  
#include "atca_basic.h"  
#include "crypto/atca_crypto_sw_sha2.h"
```

24.64.1 Detailed Description

host side methods using CryptoAuth hardware

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.65 atcacert_host_hw.h File Reference

host side methods using CryptoAuth hardware

```
#include <stddef.h>  
#include <stdint.h>  
#include "atcacert_def.h"
```

24.65.1 Detailed Description

host side methods using CryptoAuth hardware

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.66 atcacert_host_sw.c File Reference

host side methods using software implementations

```
#include "atcacert_host_sw.h"  
#include "crypto/atca_crypto_sw.h"  
#include "cal_internal.h"
```


24.66.1 Detailed Description

host side methods using software implementations

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.67 atcacert_host_sw.h File Reference

Host side methods using software implementations. host-side, the one authenticating a client, of the authentication process. Crypto functions are performed using a software library.

```
#include <stddef.h>
#include <stdint.h>
#include "atcacert_def.h"
```

24.67.1 Detailed Description

Host side methods using software implementations. host-side, the one authenticating a client, of the authentication process. Crypto functions are performed using a software library.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.68 atcacert_pem.c File Reference

Functions required to work with PEM encoded data related to X.509 certificates.

```
#include <string.h>
#include "atcacert.h"
#include "atcacert_pem.h"
#include "atca_helpers.h"
```

24.68.1 Detailed Description

Functions required to work with PEM encoded data related to X.509 certificates.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.69 atcacert_pem.h File Reference

Functions for converting between DER and PEM formats.

```
#include <stdint.h>
```

Macros

- `#define PEM_CERT_BEGIN "-----BEGIN CERTIFICATE-----"`
- `#define PEM_CERT_END "-----END CERTIFICATE-----"`
- `#define PEM_CSR_BEGIN "-----BEGIN CERTIFICATE REQUEST-----"`
- `#define PEM_CSR_END "-----END CERTIFICATE REQUEST-----"`

Functions

- ATCA_STATUS [atcacert_encode_pem](#) (const uint8_t *der, size_t der_size, char *pem, size_t *pem_size, const char *header, const char *footer)
Encode a DER data in PEM format.
- ATCA_STATUS [atcacert_decode_pem](#) (const char *pem, size_t pem_size, uint8_t *der, size_t *der_size, const char *header, const char *footer)
Decode PEM data into DER format.
- ATCA_STATUS [atcacert_encode_pem_cert](#) (const uint8_t *der_cert, size_t der_cert_size, char *pem_cert, size_t *pem_cert_size)
Encode a DER certificate in PEM format.
- ATCA_STATUS [atcacert_decode_pem_cert](#) (const char *pem_cert, size_t pem_cert_size, uint8_t *der_cert, size_t *der_cert_size)
Decode a PEM certificate into DER format.
- ATCA_STATUS [atcacert_encode_pem_csr](#) (const uint8_t *der_csr, size_t der_csr_size, char *pem_csr, size_t *pem_csr_size)
Encode a DER CSR in PEM format.
- ATCA_STATUS [atcacert_decode_pem_csr](#) (const char *pem_csr, size_t pem_csr_size, uint8_t *der_csr, size_t *der_csr_size)
Extract the CSR certificate bytes from a PEM encoded CSR certificate.

24.69.1 Detailed Description

Functions for converting between DER and PEM formats.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.69.2 Function Documentation

24.69.2.1 atcacert_decode_pem()

```
ATCA_STATUS atcacert_decode_pem (
    const char * pem,
    size_t pem_size,
    uint8_t * der,
    size_t * der_size,
    const char * header,
    const char * footer )
```

Decode PEM data into DER format.

Parameters

in	<i>pem</i>	PEM data to decode to DER.
in	<i>pem_size</i>	PEM data size in bytes.
out	<i>der</i>	DER data is returned here.
in, out	<i>der_size</i>	As input, the size of the der buffer. As output, the size of the DER data.
in	<i>header</i>	Header to find the beginning of the PEM data.
in	<i>footer</i>	Footer to find the end of the PEM data.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.69.2.2 atcacert_decode_pem_cert()

```
ATCA_STATUS atcacert_decode_pem_cert (
    const char * pem_cert,
    size_t pem_cert_size,
    uint8_t * der_cert,
    size_t * der_cert_size )
```

Decode a PEM certificate into DER format.

Parameters

in	<i>pem_cert</i>	PEM certificate to decode to DER.
in	<i>pem_cert_size</i>	PEM certificate size in bytes.
out	<i>der_cert</i>	DER certificate is returned here.
in, out	<i>der_cert_size</i>	As input, the size of the der_cert buffer. As output, the size of the DER certificate.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.69.2.3 atcacert_decode_pem_csr()

```
ATCA_STATUS atcacert_decode_pem_csr (
    const char * pem_csr,
    size_t pem_csr_size,
    uint8_t * der_csr,
    size_t * der_csr_size )
```

Extract the CSR certificate bytes from a PEM encoded CSR certificate.

Parameters

in	<i>pem_csr</i>	PEM CSR to decode to DER.
in	<i>pem_csr_size</i>	PEM CSR size in bytes.
out	<i>der_csr</i>	DER CSR is returned here.
in, out	<i>der_csr_size</i>	As input, the size of the der_csr buffer. As output, the size of the DER CSR.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.69.2.4 atcacert_encode_pem()

```
ATCA_STATUS atcacert_encode_pem (
    const uint8_t * der,
    size_t der_size,
    char * pem,
    size_t * pem_size,
    const char * header,
    const char * footer )
```

Encode a DER data in PEM format.

Parameters

in	<i>der</i>	DER data to be encoded as PEM.
out	<i>der_size</i>	DER data size in bytes.
out	<i>pem</i>	PEM encoded data is returned here.
in, out	<i>pem_size</i>	As input, the size of the pem buffer. As output, the size of the PEM data.
in	<i>header</i>	Header to place at the beginning of the PEM data.
in	<i>footer</i>	Footer to place at the end of the PEM data.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.69.2.5 atcacert_encode_pem_cert()

```
ATCA_STATUS atcacert_encode_pem_cert (
    const uint8_t * der_cert,
    size_t der_cert_size,
    char * pem_cert,
    size_t * pem_cert_size )
```

Encode a DER certificate in PEM format.

Parameters

in	<i>der_cert</i>	DER certificate to be encoded as PEM.
out	<i>der_cert_size</i>	DER certificate size in bytes.
out	<i>pem_cert</i>	PEM encoded certificate is returned here.
in, out	<i>pem_cert_size</i>	As input, the size of the pem_cert buffer. As output, the size of the PEM certificate.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.69.2.6 atcacert_encode_pem_csr()

```
ATCA_STATUS atcacert_encode_pem_csr (
    const uint8_t * der_csr,
    size_t der_csr_size,
    char * pem_csr,
    size_t * pem_csr_size )
```

Encode a DER CSR in PEM format.

Parameters

in	<i>der_csr</i>	DER CSR to be encoded as PEM.
out	<i>der_csr_size</i>	DER CSR size in bytes.
out	<i>pem_csr</i>	PEM encoded CSR is returned here.
in, out	<i>pem_csr_size</i>	As input, the size of the pem_csr buffer. As output, the size of the PEM CSR.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.70 cal_buffer.c File Reference

Cryptoauthlib buffer management system.

```
#include <string.h>
#include "cal_buffer.h"
```

Functions

- ATCA_STATUS [cal_buf_read_bytes](#) ([cal_buffer](#) *cab, size_t offset, void *dest, size_t length)
Read bytes from a cal_buffer or cal_buffer linked list.
- ATCA_STATUS [cal_buf_read_byte](#) ([cal_buffer](#) *cab, size_t offset, uint8_t *value)
- ATCA_STATUS [cal_buf_write_byte](#) ([cal_buffer](#) *cab, size_t offset, uint8_t value)
- ATCA_STATUS [cal_buf_write_bytes](#) ([cal_buffer](#) *cab, size_t offset, const void *source, size_t length)
Write bytes into a single cal_buffer structure or cal_buffer linked list.
- ATCA_STATUS [cal_buf_read_number](#) ([cal_buffer](#) *cab, size_t offset, void *dest, size_t num_size, bool buf←_big_endian)
Read a number from a cal_buffer or cal_buffer linked list This function does not reinterpret the number and signedness is only preserved if the destination is the same size as the representation in the buffer.
- ATCA_STATUS [cal_buf_write_number](#) ([cal_buffer](#) *cab, size_t offset, const void *source, size_t num_size, bool buf_big_endian)
Write a number into a cal_buffer or cal_buffer linked list This function does not reinterpret the number and signedness is only preserved if the destination is the same size as the source.
- ATCA_STATUS [cal_buf_set_used](#) ([cal_buffer](#) *buf, size_t used)
- size_t [cal_buf_get_used](#) ([cal_buffer](#) *buf)
- ATCA_STATUS [cal_buf_copy](#) ([cal_buffer](#) *dst, size_t dst_offset, [cal_buffer](#) *src, size_t src_offset, size_t length)
- ATCA_STATUS [cal_buf_set](#) ([cal_buffer](#) *dst, size_t dst_offset, uint8_t value, size_t length)
- [cal_buffer](#) [cal_buf_init_const_ptr](#) (size_t len, const uint8_t *message)
Initialize a cal buffer with constant pointer Returns the initialized cal buffer.

24.70.1 Detailed Description

Cryptoauthlib buffer management system.

Copyright

(c) 2023 Microchip Technology Inc. and its subsidiaries.

24.70.2 Function Documentation

24.70.2.1 cal_buf_read_bytes()

```
ATCA_STATUS cal_buf_read_bytes (  
    cal\_buffer * cab,  
    size_t offset,  
    void * dest,  
    size_t length )
```

Read bytes from a cal_buffer or cal_buffer linked list.

Parameters

in	<i>cab</i>	Pointer to a buffer structure or the head of a buffer structure linked list
in	<i>offset</i>	Offset to start the read from
in	<i>dest</i>	Pointer to a destination buffer
in	<i>length</i>	Length of the read - assumes dest has sufficient memory to accept the bytes being read

24.70.2.2 cal_buf_read_number()

```
ATCA_STATUS cal_buf_read_number (
    cal_buffer * cab,
    size_t offset,
    void * dest,
    size_t num_size,
    bool buf_big_endian )
```

Read a number from a cal_buffer or cal_buffer linked list. This function does not reinterpret the number and signedness is only preserved if the destination is the same size as the representation in the buffer.

Parameters

in	<i>cab</i>	Pointer to a buffer structure or the head of a buffer structure linked list
in	<i>offset</i>	Offset to start the read from
in	<i>dest</i>	Pointer to a destination number
in	<i>num_size</i>	Size of the number in bytes
in	<i>buf_big_endian</i>	Specifies the expected endianness representation within the buffer

24.70.2.3 cal_buf_write_bytes()

```
ATCA_STATUS cal_buf_write_bytes (
    cal_buffer * cab,
    size_t offset,
    const void * source,
    size_t length )
```

Write bytes into a single cal_buffer structure or cal_buffer linked list.

Parameters

in	<i>cab</i>	Pointer to a buffer structure or the head of a buffer structure linked list
in	<i>offset</i>	Target offset to start the write at
in	<i>source</i>	Pointer to a source buffer
in	<i>length</i>	Length of the write - assumes source is sufficiently large to support this operation

24.70.2.4 cal_buf_write_number()

```
ATCA_STATUS cal_buf_write_number (
    cal_buffer * cab,
    size_t offset,
    const void * source,
    size_t num_size,
    bool buf_big_endian )
```

Write a number into a cal_buffer or cal_buffer linked list This function does not reinterpret the number and signedness is only preserved if the destination is the same size as the source.

Parameters

in	<i>cab</i>	Pointer to a buffer structure or the head of a buffer structure linked list
in	<i>offset</i>	Offset to start the write at
in	<i>source</i>	Pointer to a number to be written
in	<i>num_size</i>	Size of the number in bytes
in	<i>buf_big_endian</i>	Specifies the expected endianness representation within the buffer

24.71 cal_buffer.h File Reference

Cryptoauthlib buffer management system.

```
#include <stdint.h>
#include <stdlib.h>
#include <stdbool.h>
#include "atca_config_check.h"
#include "atca_status.h"
```

Data Structures

- struct [cal_buffer_s](#)
- #define **CAL_BUF_INIT**(s, b) { (size_t)(s), (uint8_t*)(b) }
- typedef struct [cal_buffer_s](#) **cal_buffer**
- ATCA_STATUS **cal_buf_read_byte** ([cal_buffer](#) *cab, size_t offset, uint8_t *value)
- ATCA_STATUS **cal_buf_write_byte** ([cal_buffer](#) *cab, size_t offset, uint8_t value)
- ATCA_STATUS **cal_buf_read_bytes** ([cal_buffer](#) *cab, size_t offset, void *dest, size_t length)
Read bytes from a cal_buffer or cal_buffer linked list.
- ATCA_STATUS **cal_buf_write_bytes** ([cal_buffer](#) *cab, size_t offset, const void *source, size_t length)
Write bytes into a single cal_buffer structure or cal_buffer linked list.
- ATCA_STATUS **cal_buf_read_number** ([cal_buffer](#) *cab, size_t offset, void *dest, size_t num_size, bool buf_big_endian)
Read a number from a cal_buffer or cal_buffer linked list This function does not reinterpret the number and signedness is only preserved if the destination is the same size as the representation in the buffer.
- ATCA_STATUS **cal_buf_write_number** ([cal_buffer](#) *cab, size_t offset, const void *source, size_t num_size, bool buf_big_endian)

Write a number into a `cal_buffer` or `cal_buffer` linked list This function does not reinterpret the number and signedness is only preserved if the destination is the same size as the source.

- ATCA_STATUS **cal_buf_copy** (`cal_buffer` *dst, size_t dst_offset, `cal_buffer` *src, size_t src_offset, size_t length)
- ATCA_STATUS **cal_buf_set** (`cal_buffer` *dst, size_t dst_offset, uint8_t value, size_t length)
- ATCA_STATUS **cal_buf_set_used** (`cal_buffer` *buf, size_t used)
- size_t **cal_buf_get_used** (`cal_buffer` *buf)
- `cal_buffer` **cal_buf_init_const_ptr** (size_t len, const uint8_t *message)

Initialize a cal buffer with constant pointer Returns the initialized cal buffer.

24.71.1 Detailed Description

Cryptoauthlib buffer management system.

Copyright

(c) 2023 Microchip Technology Inc. and its subsidiaries.

24.71.2 Function Documentation

24.71.2.1 cal_buf_read_bytes()

```
ATCA_STATUS cal_buf_read_bytes (
    cal_buffer * cab,
    size_t offset,
    void * dest,
    size_t length )
```

Read bytes from a `cal_buffer` or `cal_buffer` linked list.

Parameters

in	<i>cab</i>	Pointer to a buffer structure or the head of a buffer structure linked list
in	<i>offset</i>	Offset to start the read from
in	<i>dest</i>	Pointer to a destination buffer
in	<i>length</i>	Length of the read - assumes dest has sufficient memory to accept the bytes being read

24.71.2.2 cal_buf_read_number()

```
ATCA_STATUS cal_buf_read_number (
    cal_buffer * cab,
    size_t offset,
    void * dest,
```

24.71 cal_buffer.h File Reference

```
size_t num_size,  
bool buf_big_endian )
```

Read a number from a `cal_buffer` or `cal_buffer` linked list. This function does not reinterpret the number and signedness is only preserved if the destination is the same size as the representation in the buffer.

Parameters

in	<i>cab</i>	Pointer to a buffer structure or the head of a buffer structure linked list
in	<i>offset</i>	Offset to start the read from
in	<i>dest</i>	Pointer to a destination number
in	<i>num_size</i>	Size of the number in bytes
in	<i>buf_big_endian</i>	Specifies the expected endianness representation within the buffer

24.71.2.3 cal_buf_write_bytes()

```
ATCA_STATUS cal_buf_write_bytes (  
    cal_buffer * cab,  
    size_t offset,  
    const void * source,  
    size_t length )
```

Write bytes into a single `cal_buffer` structure or `cal_buffer` linked list.

Parameters

in	<i>cab</i>	Pointer to a buffer structure or the head of a buffer structure linked list
in	<i>offset</i>	Target offset to start the write at
in	<i>source</i>	Pointer to a source buffer
in	<i>length</i>	Length of the write - assumes source is sufficiently large to support this operation

24.71.2.4 cal_buf_write_number()

```
ATCA_STATUS cal_buf_write_number (  
    cal_buffer * cab,  
    size_t offset,  
    const void * source,  
    size_t num_size,  
    bool buf_big_endian )
```

Write a number into a `cal_buffer` or `cal_buffer` linked list. This function does not reinterpret the number and signedness is only preserved if the destination is the same size as the source.

Parameters

in	<i>cab</i>	Pointer to a buffer structure or the head of a buffer structure linked list
in	<i>offset</i>	Offset to start the write at

Parameters

in	<i>source</i>	Pointer to a number to be written
in	<i>num_size</i>	Size of the number in bytes
in	<i>buf_big_endian</i>	Specifies the expected endianness representation within the buffer

24.72 cal_internal.h File Reference

Internal CryptoAuthLib Interfaces.

```
#include "atca_config_check.h"
#include "crypto/atca_crypto_sw.h"
#include "mbedtls/atca_mbedtls_wrap.h"
```

24.72.1 Detailed Description

Internal CryptoAuthLib Interfaces.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.73 calib_aes.c File Reference

CryptoAuthLib Basic API methods for AES command.

```
#include "cryptoauthlib.h"
```

24.73.1 Detailed Description

CryptoAuthLib Basic API methods for AES command.

The AES command supports 128-bit AES encryption or decryption of small messages or data packets in ECB mode. Also can perform GFM (Galois Field Multiply) calculation in support of AES-GCM.

Note

List of devices that support this command - ATECC608A/B. Refer to device edatasheet for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.74 calib_aes_gcm.c File Reference

CryptoAuthLib Basic API methods for AES GCM mode.

```
#include "cryptoauthlib.h"
```

24.74.1 Detailed Description

CryptoAuthLib Basic API methods for AES GCM mode.

The AES command supports 128-bit AES encryption or decryption of small messages or data packets in ECB mode. Also can perform GFM (Galois Field Multiply) calculation in support of AES-GCM.

Note

List of devices that support this command - ATECC608A/B. Refer to device datasheet for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.75 calib_aes_gcm.h File Reference

Unity tests for the cryptoauthlib AES GCM functions.

```
#include "calib_config_check.h"
```

24.75.1 Detailed Description

Unity tests for the cryptoauthlib AES GCM functions.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.76 calib_basic.c File Reference

CryptoAuthLib Basic API methods. These methods provide a simpler way to access the core crypto methods.

```
#include "cryptoauthlib.h"
```

Functions

- ATCA_STATUS [calib_wakeup_i2c](#) (ATCADevice device)
basic API methods are all prefixed with atcab_ (CryptoAuthLib Basic) the fundamental premise of the basic API is it is based on a single interface instance and that instance is global, so all basic API commands assume that one global device is the one to operate on.
- ATCA_STATUS [calib_wakeup](#) (ATCADevice device)
wakeup the CryptoAuth device
- ATCA_STATUS [calib_idle](#) (ATCADevice device)
idle the CryptoAuth device
- ATCA_STATUS [calib_sleep](#) (ATCADevice device)
invoke sleep on the CryptoAuth device
- ATCA_STATUS [calib_exit](#) (ATCADevice device)
common cleanup code which idles the device after any operation
- ATCA_STATUS [calib_get_addr](#) (uint8_t zone, uint16_t slot, uint8_t block, uint8_t offset, uint16_t *addr)
Compute the address given the zone, slot, block, and offset.
- ATCA_STATUS [calib_ca2_get_addr](#) (uint8_t zone, uint16_t slot, uint8_t block, uint8_t offset, uint16_t *addr)
Compute the address given the zone, slot, block, and offset for the device.
- ATCA_STATUS [calib_get_zone_size](#) (ATCADevice device, uint8_t zone, uint16_t slot, size_t *size)
Gets the size of the specified zone in bytes.

24.76.1 Detailed Description

CryptoAuthLib Basic API methods. These methods provide a simpler way to access the core crypto methods.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.77 calib_checkmac.c File Reference

CryptoAuthLib Basic API methods for CheckMAC command.

```
#include "cryptoauthlib.h"
```

24.77.1 Detailed Description

CryptoAuthLib Basic API methods for CheckMAC command.

The CheckMac command calculates a MAC response that would have been generated on a different CryptoAuth[↔] Authentication device and then compares the result with input value.

Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.78 calib_command.c File Reference

Microchip CryptoAuthentication device command builder - this is the main object that builds the command byte strings for the given device. It does not execute the command. The basic flow is to call a command method to build the command you want given the parameters and then send that byte string through the device interface.

```
#include "cryptoauthlib.h"
```

Functions

- ATCA_STATUS [atInfo](#) (ATCADeviceType device_type, [ATCAPacket](#) *packet)
ATCACommand Info method.
- ATCA_STATUS [atPause](#) (ATCADeviceType device_type, [ATCAPacket](#) *packet)
ATCACommand Pause method.
- void [atCRC](#) (size_t length, const uint8_t *data, uint8_t *crc_le)
Calculates CRC over the given raw data and returns the CRC in little-endian byte order.
- void [atCalcCrc](#) ([ATCAPacket](#) *packet)
This function calculates CRC and adds it to the correct offset in the packet data.
- ATCA_STATUS [atCheckCrc](#) (const uint8_t *response)
This function checks the consistency of a response.
- bool [atIsSHAFamily](#) (ATCADeviceType device_type)
determines if a given device type is a SHA device or a superset of a SHA device
- bool [atIsECCFamily](#) (ATCADeviceType device_type)
determines if a given device type is an ECC device or a superset of a ECC device
- ATCA_STATUS [isATCAError](#) (uint8_t *data)
checks for basic error frame in data

24.78.1 Detailed Description

Microchip CryptoAuthentication device command builder - this is the main object that builds the command byte strings for the given device. It does not execute the command. The basic flow is to call a command method to build the command you want given the parameters and then send that byte string through the device interface.

The primary goal of the command builder is to wrap the given parameters with the correct packet size and CRC. The caller should first fill in the parameters required in the [ATCAPacket](#) parameter given to the command. The command builder will deal with the mechanics of creating a valid packet using the parameter information.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.78.2 Function Documentation

24.78.2.1 atCalcCrc()

```
void atCalcCrc (  
    ATCAPacket * packet )
```

This function calculates CRC and adds it to the correct offset in the packet data.

Parameters

in	<i>packet</i>	Packet to calculate CRC data for
----	---------------	----------------------------------

24.78.2.2 atCheckCrc()

```
ATCA_STATUS atCheckCrc (
    const uint8_t * response )
```

This function checks the consistency of a response.

Parameters

in	<i>response</i>	pointer to response
----	-----------------	---------------------

Returns

ATCA_SUCCESS on success, otherwise ATCA_RX_CRC_ERROR

24.78.2.3 atCRC()

```
void atCRC (
    size_t length,
    const uint8_t * data,
    uint8_t * crc_le )
```

Calculates CRC over the given raw data and returns the CRC in little-endian byte order.

Parameters

in	<i>length</i>	Size of data not including the CRC byte positions
in	<i>data</i>	Pointer to the data over which to compute the CRC
out	<i>crc↔ _le</i>	Pointer to the place where the two-bytes of CRC will be returned in little-endian byte order.

24.78.2.4 atInfo()

```
ATCA_STATUS atInfo (
    ATCADeviceType device_type,
    ATCAPacket * packet )
```

ATCACommand Info method.

Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

Returns

ATCA_SUCCESS

24.78.2.5 atIsECCFamily()

```
bool atIsECCFamily (  
    ATCADeviceType device_type )
```

determines if a given device type is an ECC device or a superset of a ECC device

Parameters

in	<i>device_type</i>	Type of device to check for family type
----	--------------------	---

Returns

boolean indicating whether the given device is an ECC family device.

24.78.2.6 atIsSHAFamily()

```
bool atIsSHAFamily (  
    ATCADeviceType device_type )
```

determines if a given device type is a SHA device or a superset of a SHA device

Parameters

in	<i>device_type</i>	Type of device to check for family type
----	--------------------	---

Returns

boolean indicating whether the given device is a SHA family device.

24.78.2.7 atPause()

```
ATCA_STATUS atPause (
    ATCADeviceType device_type,
    ATCAPacket * packet )
```

ATCACommand Pause method.

Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

Returns

ATCA_SUCCESS

24.78.2.8 isATCAError()

```
ATCA_STATUS isATCAError (
    uint8_t * data )
```

checks for basic error frame in data

Parameters

in	<i>data</i>	pointer to received data - expected to be in the form of a CA device response frame
----	-------------	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.79 calib_command.h File Reference

Microchip Crypto Auth device command object - this is a command builder only, it does not send the command. The result of a command method is a fully formed packet, ready to send to the ATCAIFace object to dispatch.

```
#include <stddef.h>
#include "calib_config_check.h"
```

Data Structures

- struct [ATCAPacket](#)

Macros

- **#define ATCA_CMD_SIZE_MIN** (7u)
minimum number of bytes in command (from count byte to second CRC byte)
- **#define ATCA_CMD_SIZE_MAX** ((uint8_t)4 * 36 + 7)
maximum size of command packet (Verify)
- **#define CMD_STATUS_SUCCESS** ((uint8_t)0x00)
status byte for success
- **#define CMD_STATUS_WAKEUP** ((uint8_t)0x11)
status byte after wake-up
- **#define CMD_STATUS_BYTE_PARSE** ((uint8_t)0x03)
command parse error
- **#define CMD_STATUS_BYTE_ECC** ((uint8_t)0x05)
command ECC error
- **#define CMD_STATUS_BYTE_EXEC** ((uint8_t)0x0F)
command execution error
- **#define CMD_STATUS_BYTE_COMM** ((uint8_t)0xFF)
communication error

Opcodes for Crypto Authentication device commands

- **#define ATCA_CHECKMAC** ((uint8_t)0x28)
CheckMac command op-code.
- **#define ATCA_DERIVE_KEY** ((uint8_t)0x1C)
DeriveKey command op-code.
- **#define ATCA_INFO** ((uint8_t)0x30)
Info command op-code.
- **#define ATCA_GENDIG** ((uint8_t)0x15)
GenDig command op-code.
- **#define ATCA_GENKEY** ((uint8_t)0x40)
GenKey command op-code.
- **#define ATCA_HMAC** ((uint8_t)0x11)
HMAC command op-code.
- **#define ATCA_LOCK** ((uint8_t)0x17)
Lock command op-code.
- **#define ATCA_MAC** ((uint8_t)0x08)
MAC command op-code.
- **#define ATCA_NONCE** ((uint8_t)0x16)
Nonce command op-code.
- **#define ATCA_PAUSE** ((uint8_t)0x01)
Pause command op-code.
- **#define ATCA_PRIVWRITE** ((uint8_t)0x46)
PrivWrite command op-code.
- **#define ATCA_RANDOM** ((uint8_t)0x1B)
Random command op-code.
- **#define ATCA_READ** ((uint8_t)0x02)
Read command op-code.
- **#define ATCA_SIGN** ((uint8_t)0x41)
Sign command op-code.
- **#define ATCA_UPDATE_EXTRA** ((uint8_t)0x20)
UpdateExtra command op-code.
- **#define ATCA_VERIFY** ((uint8_t)0x45)
GenKey command op-code.
- **#define ATCA_WRITE** ((uint8_t)0x12)
Write command op-code.
- **#define ATCA_ECDH** ((uint8_t)0x43)

- ECDH command op-code.*
- #define **ATCA_COUNTER** ((uint8_t)0x24)
- Counter command op-code.*
- #define **ATCA_DELETE** ((uint8_t)0x13)
- Delete command op-code.*
- #define **ATCA_SHA** ((uint8_t)0x47)
- SHA command op-code.*
- #define **ATCA_AES** ((uint8_t)0x51)
- AES command op-code.*
- #define **ATCA_KDF** ((uint8_t)0x56)
- KDF command op-code.*
- #define **ATCA_SECUREBOOT** ((uint8_t)0x80)
- Secure Boot command op-code.*
- #define **ATCA_SELFTEST** ((uint8_t)0x77)
- Self test command op-code.*

Definitions of Data and Packet Sizes

- #define **ATCA_BLOCK_SIZE** (32u)
- size of a block*
- #define **ATCA_WORD_SIZE** (4u)
- size of a word*
- #define **ATCA_PUB_KEY_PAD** (4u)
- size of the public key pad*
- #define **ATCA_SERIAL_NUM_SIZE** (9u)
- number of bytes in the device serial number*
- #define **ATCA_RSP_SIZE_VAL** ((uint8_t)7)
- size of response packet containing four bytes of data*
- #define **ATCA_KEY_COUNT** (16u)
- number of keys*
- #define **ATCA_ECC_CONFIG_SIZE** (128u)
- size of configuration zone*
- #define **ATCA_SHA_CONFIG_SIZE** (88u)
- size of configuration zone*
- #define **ATCA_CA2_CONFIG_SIZE** (64u)
- size of ECC204 configuration zone*
- #define **ATCA_CA2_CONFIG_SLOT_SIZE** (16u)
- size of ECC204 configuration slot size*
- #define **ATCA_OTP_SIZE** (64u)
- size of OTP zone*
- #define **ATCA_DATA_SIZE** (ATCA_KEY_COUNT * ATCA_KEY_SIZE)
- size of data zone*
- #define **ATCA_AES_GFM_SIZE** ATCA_BLOCK_SIZE
- size of GFM data*
- #define **ATCA_CHIPMODE_OFFSET** (19u)
- ChipMode byte offset within the configuration zone.*
- #define **ATCA_CHIPMODE_I2C_ADDRESS_FLAG** ((uint8_t)0x01)
- ChipMode I2C Address in UserExtraAdd flag.*
- #define **ATCA_CHIPMODE_TTL_ENABLE_FLAG** ((uint8_t)0x02)
- ChipMode TTLEnable flag.*
- #define **ATCA_CHIPMODE_WATCHDOG_MASK** ((uint8_t)0x04)
- ChipMode watchdog duration mask.*
- #define **ATCA_CHIPMODE_WATCHDOG_SHORT** ((uint8_t)0x00)
- ChipMode short watchdog (~1.3s)*
- #define **ATCA_CHIPMODE_WATCHDOG_LONG** ((uint8_t)0x04)
- ChipMode long watchdog (~13s)*
- #define **ATCA_CHIPMODE_CLOCK_DIV_MASK** ((uint8_t)0xF8)
- ChipMode clock divider mask.*

- #define **ATCA_CHIPMODE_CLOCK_DIV_M0** ((uint8_t)0x00)
ChipMode clock divider M0.
- #define **ATCA_CHIPMODE_CLOCK_DIV_M1** ((uint8_t)0x28)
ChipMode clock divider M1.
- #define **ATCA_CHIPMODE_CLOCK_DIV_M2** ((uint8_t)0x68)
ChipMode clock divider M2.
- #define **ATCA_COUNT_SIZE** (1u)
Number of bytes in the command packet Count.
- #define **ATCA_CRC_SIZE** (2u)
Number of bytes in the command packet CRC.
- #define **ATCA_PACKET_OVERHEAD** (**ATCA_COUNT_SIZE** + **ATCA_CRC_SIZE**)
Number of bytes in the command packet.
- #define **ATCA_PUB_KEY_SIZE** (64u)
size of a p256 public key
- #define **ATCA_PRIV_KEY_SIZE** (32u)
size of a p256 private key
- #define **ATCA_SIG_SIZE** (64u)
size of a p256 signature
- #define **ATCA_KEY_SIZE** (32u)
size of a symmetric SHA key
- #define **RSA2048_KEY_SIZE** (256u)
size of a RSA private key
- #define **ATCA_RSP_SIZE_MIN** ((uint8_t)4)
minimum number of bytes in response
- #define **ATCA_RSP_SIZE_4** ((uint8_t)7)
size of response packet containing 4 bytes data
- #define **ATCA_RSP_SIZE_72** ((uint8_t)75)
size of response packet containing 64 bytes data
- #define **ATCA_RSP_SIZE_64** ((uint8_t)67)
size of response packet containing 64 bytes data
- #define **ATCA_RSP_SIZE_32** (35u)
size of response packet containing 32 bytes data
- #define **ATCA_RSP_SIZE_16** ((uint8_t)19)
size of response packet containing 16 bytes data
- #define **ATCA_RSP_SIZE_MAX** ((uint8_t)75)
maximum size of response packet (GenKey and Verify command)
- #define **OUTNONCE_SIZE** (32u)
Size of the OutNonce response expected from several commands.

Definitions for Command Parameter Ranges

- #define **ATCA_KEY_ID_MAX** ((uint8_t)15)
maximum value for key id
- #define **ATCA_OTP_BLOCK_MAX** ((uint8_t)1)
maximum value for OTP block

Definitions for Indexes Common to All Commands

- #define **ATCA_COUNT_IDX** (0)
command packet index for count
- #define **ATCA_OPCODE_IDX** (1)
command packet index for op-code
- #define **ATCA_PARAM1_IDX** (2)
command packet index for first parameter
- #define **ATCA_PARAM2_IDX** (3)
command packet index for second parameter
- #define **ATCA_DATA_IDX** (5)
command packet index for data load

- #define **ATCA_RSP_DATA_IDX** (1u)
buffer index of data in response

Definitions for Zone and Address Parameters

- #define **ATCA_ZONE_MASK** ((uint8_t)0x03)
Zone mask.
- #define **ATCA_ZONE_ENCRYPTED** ((uint8_t)0x40)
Zone bit 6 set: Write is encrypted with an unlocked data zone.
- #define **ATCA_ZONE_READWRITE_32** ((uint8_t)0x80)
Zone bit 7 set: Access 32 bytes, otherwise 4 bytes.
- #define **ATCA_ADDRESS_MASK_CONFIG** ((uint16_t)0x001F)
Address bits 5 to 7 are 0 for Configuration zone.
- #define **ATCA_ADDRESS_MASK_OTP** ((uint16_t)0x000F)
Address bits 4 to 7 are 0 for OTP zone.
- #define **ATCA_ADDRESS_MASK** ((uint16_t)0x007F)
Address bit 7 to 15 are always 0.
- #define **ATCA_TEMPKEY_KEYID** ((uint16_t)0xFFFF)
KeyID when referencing TempKey.

Definitions for Key types

- #define **ATCA_B283_KEY_TYPE** 0
B283 NIST ECC key.
- #define **ATCA_K283_KEY_TYPE** 1
K283 NIST ECC key.
- #define **ATCA_P256_KEY_TYPE** 4
P256 NIST ECC key.
- #define **ATCA_AES_KEY_TYPE** 6
AES-128 Key.
- #define **ATCA_SHA_KEY_TYPE** 7
SHA key or other data.

Definitions for the AES Command

- #define **AES_MODE_IDX** [ATCA_PARAM1_IDX](#)
AES command index for mode.
- #define **AES_KEYID_IDX** [ATCA_PARAM2_IDX](#)
AES command index for key id.
- #define **AES_INPUT_IDX** [ATCA_DATA_IDX](#)
AES command index for input data.
- #define **AES_COUNT** (23u)
AES command packet size.
- #define **AES_MODE_MASK** ((uint8_t)0xC7)
AES mode bits 3 to 5 are 0.
- #define **AES_MODE_KEY_BLOCK_MASK** ((uint8_t)0xC0)
AES mode mask for key block field.
- #define **AES_MODE_OP_MASK** ((uint8_t)0x07)
AES mode operation mask.
- #define **AES_MODE_ENCRYPT** ((uint8_t)0x00)
AES mode: Encrypt.
- #define **AES_MODE_DECRYPT** ((uint8_t)0x01)
AES mode: Decrypt.
- #define **AES_MODE_GFM** ((uint8_t)0x03)
AES mode: GFM calculation.
- #define **AES_MODE_KEY_BLOCK_POS** (6u)
Bit shift for key block in mode.

- #define **AES_DATA_SIZE** (16u)
size of AES encrypt/decrypt data
- #define **AES_RSP_SIZE** [ATCA_RSP_SIZE_16](#)
AES command response packet size.

Definitions for the CheckMac Command

- #define **CHECKMAC_MODE_IDX** [ATCA_PARAM1_IDX](#)
CheckMAC command index for mode.
- #define **CHECKMAC_KEYID_IDX** [ATCA_PARAM2_IDX](#)
CheckMAC command index for key identifier.
- #define **CHECKMAC_CLIENT_CHALLENGE_IDX** [ATCA_DATA_IDX](#)
CheckMAC command index for client challenge.
- #define **CHECKMAC_CLIENT_RESPONSE_IDX** (37u)
CheckMAC command index for client response.
- #define **CHECKMAC_DATA_IDX** (69u)
CheckMAC command index for other data.
- #define **CHECKMAC_COUNT** (84u)
CheckMAC command packet size.
- #define **CHECKMAC_MODE_CHALLENGE** ((uint8_t)0x00)
CheckMAC mode 0: first SHA block from key id.
- #define **CHECKMAC_MODE_BLOCK2_TEMPKEY** ((uint8_t)0x01)
CheckMAC mode bit 0: second SHA block from TempKey.
- #define **CHECKMAC_MODE_BLOCK1_TEMPKEY** ((uint8_t)0x02)
CheckMAC mode bit 1: first SHA block from TempKey.
- #define **CHECKMAC_MODE_SOURCE_FLAG_MATCH** ((uint8_t)0x04)
CheckMAC mode bit 2: match TempKey.SourceFlag.
- #define **CHECKMAC_MODE_INCLUDE_OTP_64** ((uint8_t)0x20)
CheckMAC mode bit 5: include first 64 OTP bits.
- #define **CHECKMAC_MODE_MASK** ((uint8_t)0x27)
CheckMAC mode bits 3, 4, 6, and 7 are 0.
- #define **CHECKMAC_MODE_OUTPUT_MAC_RESPONSE** ((uint8_t)0x08)
CheckMAC mode bit 3: Single byte boolean response + 32 bytes mac in SHA105 device.
- #define **CHECKMAC_CLIENT_CHALLENGE_SIZE** (32u)
CheckMAC size of client challenge.
- #define **CHECKMAC_CLIENT_RESPONSE_SIZE** (32u)
CheckMAC size of client response.
- #define **CHECKMAC_OTHER_DATA_SIZE** (13u)
CheckMAC size of "other data".
- #define **CHECKMAC_CLIENT_COMMAND_SIZE** (4u)
CheckMAC size of client command header size inside "other data".
- #define **CHECKMAC_CMD_MATCH** (0u)
CheckMAC return value when there is a match.
- #define **CHECKMAC_CMD_MISMATCH** (1u)
CheckMAC return value when there is a mismatch.
- #define **CHECKMAC_RSP_SIZE** [ATCA_RSP_SIZE_MIN](#)
CheckMAC response packet size.
- #define **CHECKMAC_SINGLE_BYTE_BOOL_RESP** (1u)
- #define **CHECKMAC_SHA105_DEFAULT_KEYID** ((uint16_t)0x0003)

Definitions for the Counter command

- #define **COUNTER_COUNT** [ATCA_CMD_SIZE_MIN](#)
- #define **COUNTER_MODE_IDX** [ATCA_PARAM1_IDX](#)
Counter command index for mode.
- #define **COUNTER_KEYID_IDX** [ATCA_PARAM2_IDX](#)
Counter command index for key id.
- #define **COUNTER_MODE_MASK** ((uint8_t)0x01)

- Counter mode bits 1 to 7 are 0.
- #define **COUNTER_MAX_VALUE** ((uint32_t)2097151)
Counter maximum value of the counter.
- #define **COUNTER_MODE_READ** ((uint8_t)0x00)
Counter command mode for reading.
- #define **COUNTER_MODE_INCREMENT** ((uint8_t)0x01)
Counter command mode for incrementing.
- #define **COUNTER_RSP_SIZE** [ATCA_RSP_SIZE_4](#)
Counter command response packet size.
- #define **COUNTER_SIZE** [ATCA_RSP_SIZE_MIN](#)
Counter size in binary.
- #define **COUNTER_MAX_VALUE_CA2** ((uint16_t)10000)
Counter maximum value of the counter for ECC204.

Definitions for the Delete command

- #define **DELETE_COUNT** (39u)
- #define **DELETE_MODE** ((uint8_t)0x00)
- #define **DELETE_MAC_SIZE** (32u)
- #define **DELETE_NONCE_KEY_ID** ((uint16_t)0x8000)

Definitions for the DeriveKey Command

- #define **DERIVE_KEY_RANDOM_IDX** [ATCA_PARAM1_IDX](#)
DeriveKey command index for random bit.
- #define **DERIVE_KEY_TARGETKEY_IDX** [ATCA_PARAM2_IDX](#)
DeriveKey command index for target slot.
- #define **DERIVE_KEY_MAC_IDX** [ATCA_DATA_IDX](#)
DeriveKey command index for optional MAC.
- #define **DERIVE_KEY_COUNT_SMALL** [ATCA_CMD_SIZE_MIN](#)
DeriveKey command packet size without MAC.
- #define **DERIVE_KEY_MODE** ((uint8_t)0x04)
DeriveKey command mode set to 4 as in datasheet.
- #define **DERIVE_KEY_COUNT_LARGE** (39u)
DeriveKey command packet size with MAC.
- #define **DERIVE_KEY_RANDOM_FLAG** ((uint8_t)4)
DeriveKey 1. parameter; has to match TempKey.SourceFlag.
- #define **DERIVE_KEY_MAC_SIZE** (32u)
DeriveKey MAC size.
- #define **DERIVE_KEY_RSP_SIZE** [ATCA_RSP_SIZE_MIN](#)
DeriveKey response packet size.

Definitions for the ECDH Command

- #define **ECDH_PREFIX_MODE** ((uint8_t)0x00)
- #define **ECDH_COUNT** ([ATCA_CMD_SIZE_MIN](#) + ATCA_PUB_KEY_SIZE)
- #define **ECDH_MODE_SOURCE_MASK** ((uint8_t)0x01)
- #define **ECDH_MODE_SOURCE_EEPROM_SLOT** ((uint8_t)0x00)
- #define **ECDH_MODE_SOURCE_TEMPKEY** ((uint8_t)0x01)
- #define **ECDH_MODE_OUTPUT_MASK** ((uint8_t)0x02)
- #define **ECDH_MODE_OUTPUT_CLEAR** ((uint8_t)0x00)
- #define **ECDH_MODE_OUTPUT_ENC** ((uint8_t)0x02)
- #define **ECDH_MODE_COPY_MASK** ((uint8_t)0x0C)
- #define **ECDH_MODE_COPY_COMPATIBLE** ((uint8_t)0x00)
- #define **ECDH_MODE_COPY_EEPROM_SLOT** ((uint8_t)0x04)
- #define **ECDH_MODE_COPY_TEMP_KEY** ((uint8_t)0x08)
- #define **ECDH_MODE_COPY_OUTPUT_BUFFER** ((uint8_t)0x0C)
- #define **ECDH_KEY_SIZE** [ATCA_BLOCK_SIZE](#)
ECDH output data size.

- #define **ECDH_RSP_SIZE** [ATCA_RSP_SIZE_64](#)
ECDH command packet size.

Definitions for the GenDig Command

- #define **GENDIG_ZONE_IDX** [ATCA_PARAM1_IDX](#)
GenDig command index for zone.
- #define **GENDIG_KEYID_IDX** [ATCA_PARAM2_IDX](#)
GenDig command index for key id.
- #define **GENDIG_DATA_IDX** [ATCA_DATA_IDX](#)
GenDig command index for optional data.
- #define **GENDIG_COUNT** [ATCA_CMD_SIZE_MIN](#)
GenDig command packet size without "other data".
- #define **GENDIG_ZONE_CONFIG** ((uint8_t)0)
GenDig zone id config. Use KeyID to specify any of the four 256-bit blocks of the Configuration zone.
- #define **GENDIG_ZONE_OTP** ((uint8_t)1)
GenDig zone id OTP. Use KeyID to specify either the first or second 256-bit block of the OTP zone.
- #define **GENDIG_ZONE_DATA** ((uint8_t)2)
GenDig zone id data. Use KeyID to specify a slot in the Data zone or a transport key in the hardware array.
- #define **GENDIG_ZONE_SHARED_NONCE** ((uint8_t)3)
GenDig zone id shared nonce. KeyID specifies the location of the input value in the message generation.
- #define **GENDIG_ZONE_COUNTER** ((uint8_t)4)
GenDig zone id counter. KeyID specifies the monotonic counter ID to be included in the message generation.
- #define **GENDIG_ZONE_KEY_CONFIG** ((uint8_t)5)
GenDig zone id key config. KeyID specifies the slot for which the configuration information is to be included in the message generation.
- #define **GENDIG_RSP_SIZE** [ATCA_RSP_SIZE_MIN](#)
GenDig command response packet size.
- #define **GENDIG_USE_TEMPKEY_BIT** ((uint16_t)0x8000)
Use temp key for GenDig command if bit 15 is 1.

Definitions for the GenDivKey Command

- #define **GENDIVKEY_MODE** ((uint8_t)2)
- #define **GENDIVKEY_OTHER_DATA_SIZE** ((uint8_t)4)
- #define **GENDIVKEY_DEFAULT_KEYID** ((uint16_t)0x0003)

Definitions for the GenKey Command

- #define **GENKEY_MODE_IDX** [ATCA_PARAM1_IDX](#)
GenKey command index for mode.
- #define **GENKEY_KEYID_IDX** [ATCA_PARAM2_IDX](#)
GenKey command index for key id.
- #define **GENKEY_DATA_IDX** (5u)
GenKey command index for other data.
- #define **GENKEY_COUNT** [ATCA_CMD_SIZE_MIN](#)
GenKey command packet size without "other data".
- #define **GENKEY_COUNT_DATA** (10u)
GenKey command packet size with "other data".
- #define **GENKEY_OTHER_DATA_SIZE** (3u)
GenKey size of "other data".
- #define **GENKEY_MODE_MASK** ((uint8_t)0x1C)
GenKey mode bits 0 to 1 and 5 to 7 are 0.
- #define **GENKEY_MODE_PRIVATE** ((uint8_t)0x04)
GenKey mode: private key generation.
- #define **GENKEY_MODE_PUBLIC** ((uint8_t)0x00)
GenKey mode: public key calculation.

- #define **GENKEY_MODE_DIGEST** ((uint8_t)0x08)
GenKey mode: PubKey digest will be created after the public key is calculated.
- #define **GENKEY_MODE_PUBKEY_DIGEST** ((uint8_t)0x10)
GenKey mode: Calculate PubKey digest on the public key in KeyId.
- #define **GENKEY_MODE_MAC** ((uint8_t)0x20)
Genkey mode: Calculate MAC of public key + session key.
- #define **GENKEY_PRIVATE_TO_TEMPKEY** ((uint16_t)0xFFFF)
GenKey Create private key and store to tempkey (608 only)
- #define **GENKEY_RSP_SIZE_SHORT** [ATCA_RSP_SIZE_MIN](#)
GenKey response packet size in Digest mode.
- #define **GENKEY_RSP_SIZE_LONG** [ATCA_RSP_SIZE_64](#)
GenKey response packet size when returning a public key.

Definitions for the HMAC Command

- #define **HMAC_MODE_IDX** [ATCA_PARAM1_IDX](#)
HMAC command index for mode.
- #define **HMAC_KEYID_IDX** [ATCA_PARAM2_IDX](#)
HMAC command index for key id.
- #define **HMAC_COUNT** [ATCA_CMD_SIZE_MIN](#)
HMAC command packet size.
- #define **HMAC_MODE_FLAG_TK_RAND** ((uint8_t)0x00)
HMAC mode bit 2: The value of this bit must match the value in TempKey.SourceFlag or the command will return an error.
- #define **HMAC_MODE_FLAG_TK_NORAND** ((uint8_t)0x04)
HMAC mode bit 2: The value of this bit must match the value in TempKey.SourceFlag or the command will return an error.
- #define **HMAC_MODE_FLAG_OTP88** ((uint8_t)0x10)
HMAC mode bit 4: Include the first 88 OTP bits (OTP[0] through OTP[10]) in the message.; otherwise, the corresponding message bits are set to zero. Not applicable for ATECC508A.
- #define **HMAC_MODE_FLAG_OTP64** ((uint8_t)0x20)
HMAC mode bit 5: Include the first 64 OTP bits (OTP[0] through OTP[7]) in the message.; otherwise, the corresponding message bits are set to zero. If Mode[4] is set, the value of this mode bit is ignored. Not applicable for ATECC508A.
- #define **HMAC_MODE_FLAG_FULLSN** ((uint8_t)0x40)
HMAC mode bit 6: If set, include the 48 bits SN[2:3] and SN[4:7] in the message.; otherwise, the corresponding message bits are set to zero.
- #define **HMAC_MODE_MASK** ((uint8_t)0x74)
HMAC mode bits 0, 1, 3, and 7 are 0.
- #define **HMAC_DIGEST_SIZE** (32u)
HMAC size of digest response.
- #define **HMAC_RSP_SIZE** [ATCA_RSP_SIZE_32](#)
HMAC command response packet size.

Definitions for the Info Command

- #define **INFO_PARAM1_IDX** [ATCA_PARAM1_IDX](#)
Info command index for 1. parameter.
- #define **INFO_PARAM2_IDX** [ATCA_PARAM2_IDX](#)
Info command index for 2. parameter.
- #define **INFO_COUNT** [ATCA_CMD_SIZE_MIN](#)
Info command packet size.
- #define **INFO_MODE_REVISION** ((uint8_t)0x00)
Info mode Revision.
- #define **INFO_MODE_KEY_VALID** ((uint8_t)0x01)
Info mode KeyValid.
- #define **INFO_MODE_STATE** ((uint8_t)0x02)
Info mode State.

- **#define INFO_MODE_LOCK_STATUS** ((uint8_t)0x02)
Info mode Lock status for ECC204,TA010,SHA10x devices.
- **#define INFO_MODE_CHIP_STATUS** ((uint8_t)0xC5)
Info mode Chip status for ECC204,TA010,SHA10x devices.
- **#define INFO_MODE_GPIO** ((uint8_t)0x03)
Info mode GPIO.
- **#define INFO_MODE_VOL_KEY_PERMIT** ((uint8_t)0x04)
Info mode GPIO.
- **#define INFO_MODE_MAX** ((uint8_t)0x03)
Info mode maximum value.
- **#define INFO_NO_STATE** ((uint8_t)0x00)
Info mode is not the state mode.
- **#define INFO_OUTPUT_STATE_MASK** ((uint8_t)0x01)
Info output state mask.
- **#define INFO_DRIVER_STATE_MASK** ((uint8_t)0x02)
Info driver state mask.
- **#define INFO_PARAM2_SET_LATCH_STATE** ((uint16_t)0x0002)
Info param2 to set the persistent latch state.
- **#define INFO_PARAM2_LATCH_SET** ((uint16_t)0x0001)
Info param2 to set the persistent latch.
- **#define INFO_PARAM2_LATCH_CLEAR** ((uint16_t)0x0000)
Info param2 to clear the persistent latch.
- **#define INFO_SIZE** ((uint8_t)0x04)
Info return size.
- **#define INFO_RSP_SIZE ATCA_RSP_SIZE_VAL**
Info command response packet size.

Definitions for the KDF Command

- **#define KDF_MODE_IDX ATCA_PARAM1_IDX**
KDF command index for mode.
- **#define KDF_KEYID_IDX ATCA_PARAM2_IDX**
KDF command index for key id.
- **#define KDF_DETAILS_IDX ATCA_DATA_IDX**
KDF command index for details.
- **#define KDF_DETAILS_SIZE** (4u)
KDF details (param3) size.
- **#define KDF_MESSAGE_IDX (ATCA_DATA_IDX + KDF_DETAILS_SIZE)**
- **#define KDF_MODE_SOURCE_MASK** ((uint8_t)0x03)
KDF mode source key mask.
- **#define KDF_MODE_SOURCE_TEMPKEY** ((uint8_t)0x00)
KDF mode source key in TempKey.
- **#define KDF_MODE_SOURCE_TEMPKEY_UP** ((uint8_t)0x01)
KDF mode source key in upper TempKey.
- **#define KDF_MODE_SOURCE_SLOT** ((uint8_t)0x02)
KDF mode source key in a slot.
- **#define KDF_MODE_SOURCE_ALTKEYBUF** ((uint8_t)0x03)
KDF mode source key in alternate key buffer.
- **#define KDF_MODE_TARGET_MASK** ((uint8_t)0x1C)
KDF mode target key mask.
- **#define KDF_MODE_TARGET_TEMPKEY** ((uint8_t)0x00)
KDF mode target key in TempKey.
- **#define KDF_MODE_TARGET_TEMPKEY_UP** ((uint8_t)0x04)
KDF mode target key in upper TempKey.
- **#define KDF_MODE_TARGET_SLOT** ((uint8_t)0x08)
KDF mode target key in slot.
- **#define KDF_MODE_TARGET_ALTKEYBUF** ((uint8_t)0x0C)
KDF mode target key in alternate key buffer.

- **#define KDF_MODE_TARGET_OUTPUT** ((uint8_t)0x10)
KDF mode target key in output buffer.
- **#define KDF_MODE_TARGET_OUTPUT_ENC** ((uint8_t)0x14)
KDF mode target key encrypted in output buffer.
- **#define KDF_MODE_ALG_MASK** ((uint8_t)0x60)
KDF mode algorithm mask.
- **#define KDF_MODE_ALG_PRF** ((uint8_t)0x00)
KDF mode PRF algorithm.
- **#define KDF_MODE_ALG_AES** ((uint8_t)0x20)
KDF mode AES algorithm.
- **#define KDF_MODE_ALG_HKDF** ((uint8_t)0x40)
KDF mode HKDF algorithm.
- **#define KDF_DETAILS_PRF_KEY_LEN_MASK** ((uint32_t)0x00000003)
KDF details for PRF, source key length mask.
- **#define KDF_DETAILS_PRF_KEY_LEN_16** ((uint32_t)0x00000000)
KDF details for PRF, source key length is 16 bytes.
- **#define KDF_DETAILS_PRF_KEY_LEN_32** ((uint32_t)0x00000001)
KDF details for PRF, source key length is 32 bytes.
- **#define KDF_DETAILS_PRF_KEY_LEN_48** ((uint32_t)0x00000002)
KDF details for PRF, source key length is 48 bytes.
- **#define KDF_DETAILS_PRF_KEY_LEN_64** ((uint32_t)0x00000003)
KDF details for PRF, source key length is 64 bytes.
- **#define KDF_DETAILS_PRF_TARGET_LEN_MASK** ((uint32_t)0x00000100)
KDF details for PRF, target length mask.
- **#define KDF_DETAILS_PRF_TARGET_LEN_32** ((uint32_t)0x00000000)
KDF details for PRF, target length is 32 bytes.
- **#define KDF_DETAILS_PRF_TARGET_LEN_64** ((uint32_t)0x00000100)
KDF details for PRF, target length is 64 bytes.
- **#define KDF_DETAILS_PRF_AEAD_MASK** ((uint32_t)0x00000600)
KDF details for PRF, AEAD processing mask.
- **#define KDF_DETAILS_PRF_AEAD_MODE0** ((uint32_t)0x00000000)
KDF details for PRF, AEAD no processing.
- **#define KDF_DETAILS_PRF_AEAD_MODE1** ((uint32_t)0x00000200)
KDF details for PRF, AEAD First 32 go to target, second 32 go to output buffer.
- **#define KDF_DETAILS_AES_KEY_LOC_MASK** ((uint32_t)0x00000003)
KDF details for AES, key location mask.
- **#define KDF_DETAILS_HKDF_MSG_LOC_MASK** ((uint32_t)0x00000003)
KDF details for HKDF, message location mask.
- **#define KDF_DETAILS_HKDF_MSG_LOC_SLOT** ((uint32_t)0x00000000)
KDF details for HKDF, message location in slot.
- **#define KDF_DETAILS_HKDF_MSG_LOC_TEMPKEY** ((uint32_t)0x00000001)
KDF details for HKDF, message location in TempKey.
- **#define KDF_DETAILS_HKDF_MSG_LOC_INPUT** ((uint32_t)0x00000002)
KDF details for HKDF, message location in input parameter.
- **#define KDF_DETAILS_HKDF_MSG_LOC_IV** ((uint32_t)0x00000003)
KDF details for HKDF, message location is a special IV function.
- **#define KDF_DETAILS_HKDF_ZERO_KEY** ((uint32_t)0x00000004)
KDF details for HKDF, key is 32 bytes of zero.

Definitions for the Lock Command

- **#define LOCK_ZONE_IDX** [ATCA_PARAM1_IDX](#)
Lock command index for zone.
- **#define LOCK_SUMMARY_IDX** [ATCA_PARAM2_IDX](#)
Lock command index for summary.
- **#define LOCK_COUNT** [ATCA_CMD_SIZE_MIN](#)
Lock command packet size.
- **#define LOCK_ZONE_CONFIG** ((uint8_t)0x00)

- *Lock zone is Config.*
• #define **LOCK_ZONE_DATA** ((uint8_t)0x01)
- *Lock zone is OTP or Data.*
• #define **LOCK_ZONE_DATA_SLOT** ((uint8_t)0x02)
- *Lock slot of Data.*
• #define **LOCK_ZONE_CA2_DATA** ((uint8_t)0x00)
- *Lock second gen Data zone by slot.*
• #define **LOCK_ZONE_CA2_CONFIG** ((uint8_t)0x01)
- *Lock second gen configuration zone by slot.*
• #define **LOCK_ZONE_NO_CRC** ((uint8_t)0x80)
- *Lock command: Ignore summary.*
• #define **LOCK_ZONE_MASK** ((uint8_t)0xBF)
- *Lock parameter 1 bits 6 are 0.*
• #define **ATCA_UNLOCKED** ((uint8_t)0x55)
- *Value indicating an unlocked zone.*
• #define **ATCA_LOCKED** ((uint8_t)0x00)
- *Value indicating a locked zone.*
• #define **LOCK_RSP_SIZE** [ATCA_RSP_SIZE_MIN](#)
- *Lock command response packet size.*

Definitions for the MAC Command

- #define **MAC_MODE_IDX** [ATCA_PARAM1_IDX](#)
MAC command index for mode.
- #define **MAC_KEYID_IDX** [ATCA_PARAM2_IDX](#)
MAC command index for key id.
- #define **MAC_CHALLENGE_IDX** [ATCA_DATA_IDX](#)
MAC command index for optional challenge.
- #define **MAC_COUNT_SHORT** [ATCA_CMD_SIZE_MIN](#)
MAC command packet size without challenge.
- #define **MAC_COUNT_LONG** (39u)
MAC command packet size with challenge.
- #define **MAC_MODE_CHALLENGE** ((uint8_t)0x00)
MAC mode 0: first SHA block from data slot.
- #define **MAC_MODE_BLOCK2_TEMPKEY** ((uint8_t)0x01)
MAC mode bit 0: second SHA block from TempKey.
- #define **MAC_MODE_BLOCK1_TEMPKEY** ((uint8_t)0x02)
MAC mode bit 1: first SHA block from TempKey.
- #define **MAC_MODE_SOURCE_FLAG_MATCH** ((uint8_t)0x04)
MAC mode bit 2: match TempKey.SourceFlag.
- #define **MAC_MODE_PTNONCE_TEMPKEY** ((uint8_t)0x06)
MAC mode bit 0: second SHA block from TempKey.
- #define **MAC_MODE_PASSTHROUGH** ((uint8_t)0x07)
MAC mode bit 0-2: pass-through mode.
- #define **MAC_MODE_INCLUDE_OTP_88** ((uint8_t)0x10)
MAC mode bit 4: include first 88 OTP bits.
- #define **MAC_MODE_INCLUDE_OTP_64** ((uint8_t)0x20)
MAC mode bit 5: include first 64 OTP bits.
- #define **MAC_MODE_INCLUDE_SN** ((uint8_t)0x40)
MAC mode bit 6: include serial number.
- #define **MAC_CHALLENGE_SIZE** (32u)
MAC size of challenge.
- #define **MAC_SIZE** (32u)
MAC size of response.
- #define **MAC_MODE_MASK** ((uint8_t)0x77)
MAC mode bits 3 and 7 are 0.
- #define **MAC_RSP_SIZE** [ATCA_RSP_SIZE_32](#)
MAC command response packet size.

- #define **MAC_SHA104_DEFAULT_KEYID** ((uint16_t)0x0003)

Definitions for the Nonce Command

- #define **NONCE_MODE_IDX** [ATCA_PARAM1_IDX](#)
Nonce command index for mode.
- #define **NONCE_PARAM2_IDX** [ATCA_PARAM2_IDX](#)
Nonce command index for 2. parameter.
- #define **NONCE_INPUT_IDX** [ATCA_DATA_IDX](#)
Nonce command index for input data.
- #define **NONCE_COUNT_SHORT** ([ATCA_CMD_SIZE_MIN](#) + 20u)
Nonce command packet size for 20 bytes of NumIn.
- #define **NONCE_COUNT_LONG** ([ATCA_CMD_SIZE_MIN](#) + 32u)
Nonce command packet size for 32 bytes of NumIn.
- #define **NONCE_COUNT_LONG_64** ([ATCA_CMD_SIZE_MIN](#) + 64u)
Nonce command packet size for 64 bytes of NumIn.
- #define **NONCE_MODE_MASK** ((uint8_t)0x03)
Nonce mode bits 2 to 7 are 0.
- #define **NONCE_MODE_SEED_UPDATE** ((uint8_t)0x00)
Nonce mode: update seed.
- #define **NONCE_MODE_NO_SEED_UPDATE** ((uint8_t)0x01)
Nonce mode: do not update seed.
- #define **NONCE_MODE_INVALID** ((uint8_t)0x02)
Nonce mode 2 is invalid.
- #define **NONCE_MODE_PASSTHROUGH** ((uint8_t)0x03)
Nonce mode: pass-through.
- #define **NONCE_MODE_GEN_SESSION_KEY** ((uint8_t)0x02)
Nonce mode: Generate session key in ECC204 device.
- #define **NONCE_MODE_INPUT_LEN_MASK** ((uint8_t)0x20)
Nonce mode: input size mask.
- #define **NONCE_MODE_INPUT_LEN_32** ((uint8_t)0x00)
Nonce mode: input size is 32 bytes.
- #define **NONCE_MODE_INPUT_LEN_64** ((uint8_t)0x20)
Nonce mode: input size is 64 bytes.
- #define **NONCE_MODE_TARGET_MASK** ((uint8_t)0xC0)
Nonce mode: target mask.
- #define **NONCE_MODE_TARGET_TEMPKEY** ((uint8_t)0x00)
Nonce mode: target is TempKey.
- #define **NONCE_MODE_TARGET_MSGDIGBUF** ((uint8_t)0x40)
Nonce mode: target is Message Digest Buffer.
- #define **NONCE_MODE_TARGET_ALTKEYBUF** ((uint8_t)0x80)
Nonce mode: target is Alternate Key Buffer.
- #define **NONCE_ZERO_CALC_MASK** ((uint16_t)0x8000)
Nonce zero (param2): calculation mode mask.
- #define **NONCE_ZERO_CALC_RANDOM** ((uint16_t)0x0000)
Nonce zero (param2): calculation mode random, use RNG in calculation and return RNG output.
- #define **NONCE_ZERO_CALC_TEMPKEY** ((uint16_t)0x8000)
Nonce zero (param2): calculation mode TempKey, use TempKey in calculation and return new TempKey value.
- #define **NONCE_NUMIN_SIZE** (20)
Nonce NumIn size for random modes.
- #define **NONCE_NUMIN_SIZE_PASSTHROUGH** (32)
Nonce NumIn size for 32-byte pass-through mode.
- #define **NONCE_RSP_SIZE_SHORT** [ATCA_RSP_SIZE_MIN](#)
Nonce command response packet size with no output.
- #define **NONCE_RSP_SIZE_LONG** [ATCA_RSP_SIZE_32](#)
Nonce command response packet size with output.

Definitions for the Pause Command

- #define **PAUSE_SELECT_IDX** [ATCA_PARAM1_IDX](#)
Pause command index for Selector.
- #define **PAUSE_PARAM2_IDX** [ATCA_PARAM2_IDX](#)
Pause command index for 2. parameter.
- #define **PAUSE_COUNT** [ATCA_CMD_SIZE_MIN](#)
Pause command packet size.
- #define **PAUSE_RSP_SIZE** [ATCA_RSP_SIZE_MIN](#)
Pause command response packet size.

Definitions for the PrivWrite Command

- #define **PRIVWRITE_ZONE_IDX** [ATCA_PARAM1_IDX](#)
PrivWrite command index for zone.
- #define **PRIVWRITE_KEYID_IDX** [ATCA_PARAM2_IDX](#)
PrivWrite command index for KeyID.
- #define **PRIVWRITE_VALUE_IDX** (5)
PrivWrite command index for value.
- #define **PRIVWRITE_MAC_IDX** (41)
PrivWrite command index for MAC.
- #define **PRIVWRITE_COUNT** (75)
PrivWrite command packet size.
- #define **PRIVWRITE_ZONE_MASK** ((uint8_t)0x40)
PrivWrite zone bits 0 to 5 and 7 are 0.
- #define **PRIVWRITE_MODE_ENCRYPT** ((uint8_t)0x40)
PrivWrite mode: encrypted.
- #define **PRIVWRITE_RSP_SIZE** [ATCA_RSP_SIZE_MIN](#)
PrivWrite command response packet size.

Definitions for the Random Command

- #define **RANDOM_MODE_IDX** [ATCA_PARAM1_IDX](#)
Random command index for mode.
- #define **RANDOM_PARAM2_IDX** [ATCA_PARAM2_IDX](#)
Random command index for 2. parameter.
- #define **RANDOM_COUNT** [ATCA_CMD_SIZE_MIN](#)
Random command packet size.
- #define **RANDOM_SEED_UPDATE** ((uint8_t)0x00)
Random mode for automatic seed update.
- #define **RANDOM_NO_SEED_UPDATE** ((uint8_t)0x01)
Random mode for no seed update.
- #define **RANDOM_NUM_SIZE** ((uint8_t)32)
Number of bytes in the data packet of a random command.
- #define **RANDOM_RSP_SIZE** [ATCA_RSP_SIZE_32](#)
Random command response packet size.

Definitions for the Read Command

- #define **READ_ZONE_IDX** [ATCA_PARAM1_IDX](#)
Read command index for zone.
- #define **READ_ADDR_IDX** [ATCA_PARAM2_IDX](#)
Read command index for address.
- #define **READ_COUNT** [ATCA_CMD_SIZE_MIN](#)
Read command packet size.
- #define **READ_ZONE_MASK** ((uint8_t)0x83)
Read zone bits 2 to 6 are 0.
- #define **READ_4_RSP_SIZE** [ATCA_RSP_SIZE_VAL](#)
Read command response packet size when reading 4 bytes.

- #define **READ_32_RSP_SIZE** [ATCA_RSP_SIZE_32](#)
Read command response packet size when reading 32 bytes.

Definitions for the SecureBoot Command

- #define **SECUREBOOT_MODE_IDX** [ATCA_PARAM1_IDX](#)
SecureBoot command index for mode.
- #define **SECUREBOOT_DIGEST_SIZE** (32u)
SecureBoot digest input size.
- #define **SECUREBOOT_SIGNATURE_SIZE** (64u)
SecureBoot signature input size.
- #define **SECUREBOOT_COUNT_DIG** ([ATCA_CMD_SIZE_MIN](#) + [SECUREBOOT_DIGEST_SIZE](#))
SecureBoot command packet size for just a digest.
- #define **SECUREBOOT_COUNT_DIG_SIG** ([ATCA_CMD_SIZE_MIN](#) + [SECUREBOOT_DIGEST_SIZE](#) + [SECUREBOOT_SIGNATURE_SIZE](#))
SecureBoot command packet size for a digest and signature.
- #define **SECUREBOOT_MAC_SIZE** (32u)
SecureBoot MAC output size.
- #define **SECUREBOOT_RSP_SIZE_NO_MAC** [ATCA_RSP_SIZE_MIN](#)
SecureBoot response packet size for no MAC.
- #define **SECUREBOOT_RSP_SIZE_MAC** ([ATCA_PACKET_OVERHEAD](#) + [SECUREBOOT_MAC_SIZE](#))
SecureBoot response packet size with MAC.
- #define **SECUREBOOT_MODE_MASK** ((uint8_t)0x07)
SecureBoot mode mask.
- #define **SECUREBOOT_MODE_FULL** ((uint8_t)0x05)
SecureBoot mode Full.
- #define **SECUREBOOT_MODE_FULL_STORE** ((uint8_t)0x06)
SecureBoot mode FullStore.
- #define **SECUREBOOT_MODE_FULL_COPY** ((uint8_t)0x07)
SecureBoot mode FullCopy.
- #define **SECUREBOOT_MODE_PROHIBIT_FLAG** ((uint8_t)0x40)
SecureBoot mode flag to prohibit SecureBoot until next power cycle.
- #define **SECUREBOOT_MODE_ENC_MAC_FLAG** ((uint8_t)0x80)
SecureBoot mode flag for encrypted digest and returning validating MAC.
- #define **SECUREBOOTCONFIG_OFFSET** (70)
SecureBootConfig byte offset into the configuration zone.
- #define **SECUREBOOTCONFIG_MODE_MASK** ((uint16_t)0x0003)
Mask for SecureBootMode field in SecureBootConfig value.
- #define **SECUREBOOTCONFIG_MODE_DISABLED** ((uint16_t)0x0000)
Disabled SecureBootMode in SecureBootConfig value.
- #define **SECUREBOOTCONFIG_MODE_FULL_BOTH** ((uint16_t)0x0001)
Both digest and signature always required SecureBootMode in SecureBootConfig value.
- #define **SECUREBOOTCONFIG_MODE_FULL_SIG** ((uint16_t)0x0002)
Signature stored SecureBootMode in SecureBootConfig value.
- #define **SECUREBOOTCONFIG_MODE_FULL_DIG** ((uint16_t)0x0003)
Digest stored SecureBootMode in SecureBootConfig value.

Definitions for the SelfTest Command

- #define **SELFTEST_MODE_IDX** [ATCA_PARAM1_IDX](#)
SelfTest command index for mode.
- #define **SELFTEST_COUNT** [ATCA_CMD_SIZE_MIN](#)
SelfTest command packet size.
- #define **SELFTEST_MODE_RNG** ((uint8_t)0x01)
SelfTest mode RNG DRBG function.
- #define **SELFTEST_MODE_ECDSA_SIGN_VERIFY** ((uint8_t)0x04)
SelfTest mode ECDSA verify function.
- #define **SELFTEST_MODE_ECDH** ((uint8_t)0x08)

- SelfTest mode ECDH function.*
- #define **SELFTEST_MODE_AES** ((uint8_t)0x10)
SelfTest mode AES encrypt function.
- #define **SELFTEST_MODE_SHA** ((uint8_t)0x20)
SelfTest mode SHA function.
- #define **SELFTEST_MODE_ALL** ((uint8_t)0x3B)
SelfTest mode all algorithms.
- #define **SELFTEST_RSP_SIZE** [ATCA_RSP_SIZE_MIN](#)
SelfTest command response packet size.

Definitions for the SHA Command

- #define **SHA_COUNT_SHORT** [ATCA_CMD_SIZE_MIN](#)
- #define **SHA_COUNT_LONG** [ATCA_CMD_SIZE_MIN](#)
Just a starting size.
- #define **ATCA_SHA_DIGEST_SIZE** (32u)
- #define **SHA_DATA_MAX** (64)
- #define **SHA_MODE_MASK** ((uint8_t)0x07)
Mask the bit 0-2.
- #define **SHA_MODE_SHA256_START** ((uint8_t)0x00)
Initialization, does not accept a message.
- #define **SHA_MODE_SHA256_UPDATE** ((uint8_t)0x01)
Add 64 bytes in the message to the SHA context.
- #define **SHA_MODE_SHA256_END** ((uint8_t)0x02)
Complete the calculation and return the digest.
- #define **SHA_MODE_SHA256_PUBLIC** ((uint8_t)0x03)
Add 64 byte ECC public key in the slot to the SHA context.
- #define **SHA_MODE_HMAC_START** ((uint8_t)0x04)
Initialization, HMAC calculation.
- #define **SHA_MODE_ECC204_HMAC_START** ((uint8_t)0x03)
Initialization, HMAC calculation for ECC204.
- #define **SHA_MODE_HMAC_UPDATE** ((uint8_t)0x01)
Add 64 bytes in the message to the SHA context.
- #define **SHA_MODE_HMAC_END** ((uint8_t)0x05)
Complete the HMAC computation and return digest.
- #define **SHA_MODE_608_HMAC_END** ((uint8_t)0x02)
Complete the HMAC computation and return digest... Different command on 608.
- #define **SHA_MODE_ECC204_HMAC_END** ((uint8_t)0x02)
Complete the HMAC computation and return digest... Different mode on ECC204.
- #define **SHA_MODE_READ_CONTEXT** ((uint8_t)0x06)
Read current SHA-256 context out of the device.
- #define **SHA_MODE_WRITE_CONTEXT** ((uint8_t)0x07)
Restore a SHA-256 context into the device.
- #define **SHA_MODE_TARGET_MASK** ((uint8_t)0xC0)
Resulting digest target location mask.
- #define **SHA_RSP_SIZE** [ATCA_RSP_SIZE_32](#)
SHA command response packet size.
- #define **SHA_RSP_SIZE_SHORT** [ATCA_RSP_SIZE_MIN](#)
SHA command response packet size only status code.
- #define **SHA_RSP_SIZE_LONG** [ATCA_RSP_SIZE_32](#)
SHA command response packet size.

Definitions for the Sign Command

- #define **SIGN_MODE_IDX** [ATCA_PARAM1_IDX](#)
Sign command index for mode.
- #define **SIGN_KEYID_IDX** [ATCA_PARAM2_IDX](#)
Sign command index for key id.

- #define **SIGN_COUNT** [ATCA_CMD_SIZE_MIN](#)
Sign command packet size.
- #define **SIGN_MODE_MASK** ((uint8_t)0xE1)
Sign mode bits 1 to 4 are 0.
- #define **SIGN_MODE_INTERNAL** ((uint8_t)0x00)
Sign mode 0: internal.
- #define **SIGN_MODE_INVALIDATE** ((uint8_t)0x01)
Sign mode bit 1: Signature will be used for Verify(Invalidate)
- #define **SIGN_MODE_INCLUDE_SN** ((uint8_t)0x40)
Sign mode bit 6: include serial number.
- #define **SIGN_MODE_EXTERNAL** ((uint8_t)0x80)
Sign mode bit 7: external.
- #define **SIGN_MODE_SOURCE_MASK** ((uint8_t)0x20)
Sign mode message source mask.
- #define **SIGN_MODE_SOURCE_TEMPKEY** ((uint8_t)0x00)
Sign mode message source is TempKey.
- #define **SIGN_MODE_SOURCE_MSGDIGBUF** ((uint8_t)0x20)
Sign mode message source is the Message Digest Buffer.
- #define **SIGN_RSP_SIZE** [ATCA_RSP_SIZE_MAX](#)
Sign command response packet size.

Definitions for the UpdateExtra Command

- #define **UPDATE_MODE_IDX** [ATCA_PARAM1_IDX](#)
UpdateExtra command index for mode.
- #define **UPDATE_VALUE_IDX** [ATCA_PARAM2_IDX](#)
UpdateExtra command index for new value.
- #define **UPDATE_COUNT** [ATCA_CMD_SIZE_MIN](#)
UpdateExtra command packet size.
- #define **UPDATE_MODE_USER_EXTRA** ((uint8_t)0x00)
UpdateExtra mode update UserExtra (config byte 84)
- #define **UPDATE_MODE_SELECTOR** ((uint8_t)0x01)
UpdateExtra mode update Selector (config byte 85)
- #define **UPDATE_MODE_USER_EXTRA_ADD** [UPDATE_MODE_SELECTOR](#)
UpdateExtra mode update UserExtraAdd (config byte 85)
- #define **UPDATE_MODE_DEC_COUNTER** ((uint8_t)0x02)
UpdateExtra mode: decrement counter.
- #define **UPDATE_RSP_SIZE** [ATCA_RSP_SIZE_MIN](#)
UpdateExtra command response packet size.

Definitions for the Verify Command

- #define **VERIFY_MODE_IDX** [ATCA_PARAM1_IDX](#)
Verify command index for mode.
- #define **VERIFY_KEYID_IDX** [ATCA_PARAM2_IDX](#)
Verify command index for key id.
- #define **VERIFY_DATA_IDX** (5)
Verify command index for data.
- #define **VERIFY_256_STORED_COUNT** (71)
Verify command packet size for 256-bit key in stored mode.
- #define **VERIFY_283_STORED_COUNT** (79)
Verify command packet size for 283-bit key in stored mode.
- #define **VERIFY_256_VALIDATE_COUNT** (90)
Verify command packet size for 256-bit key in validate mode.
- #define **VERIFY_283_VALIDATE_COUNT** (98)
Verify command packet size for 283-bit key in validate mode.
- #define **VERIFY_256_EXTERNAL_COUNT** (135)
Verify command packet size for 256-bit key in external mode.

- **#define VERIFY_283_EXTERNAL_COUNT** (151)
Verify command packet size for 283-bit key in external mode.
- **#define VERIFY_256_KEY_SIZE** (64)
Verify key size for 256-bit key.
- **#define VERIFY_283_KEY_SIZE** (72)
Verify key size for 283-bit key.
- **#define VERIFY_256_SIGNATURE_SIZE** (64)
Verify signature size for 256-bit key.
- **#define VERIFY_283_SIGNATURE_SIZE** (72)
Verify signature size for 283-bit key.
- **#define VERIFY_OTHER_DATA_SIZE** (19u)
Verify size of "other data".
- **#define VERIFY_MODE_MASK** ((uint8_t)0x07)
Verify mode bits 3 to 7 are 0.
- **#define VERIFY_MODE_STORED** ((uint8_t)0x00)
Verify mode: stored.
- **#define VERIFY_MODE_VALIDATE_EXTERNAL** ((uint8_t)0x01)
Verify mode: validate external.
- **#define VERIFY_MODE_EXTERNAL** ((uint8_t)0x02)
Verify mode: external.
- **#define VERIFY_MODE_VALIDATE** ((uint8_t)0x03)
Verify mode: validate.
- **#define VERIFY_MODE_INVALIDATE** ((uint8_t)0x07)
Verify mode: invalidate.
- **#define VERIFY_MODE_SOURCE_MASK** ((uint8_t)0x20)
Verify mode message source mask.
- **#define VERIFY_MODE_SOURCE_TEMPKEY** ((uint8_t)0x00)
Verify mode message source is TempKey.
- **#define VERIFY_MODE_SOURCE_MSGDIGBUF** ((uint8_t)0x20)
Verify mode message source is the Message Digest Buffer.
- **#define VERIFY_MODE_MAC_FLAG** ((uint8_t)0x80)
Verify mode: MAC.
- **#define VERIFY_KEY_B283** ((uint16_t)0x0000)
Verify key type: B283.
- **#define VERIFY_KEY_K283** ((uint16_t)0x0001)
Verify key type: K283.
- **#define VERIFY_KEY_P256** ((uint16_t)0x0004)
Verify key type: P256.
- **#define VERIFY_RSP_SIZE** [ATCA_RSP_SIZE_MIN](#)
Verify command response packet size.
- **#define VERIFY_RSP_SIZE_MAC** [ATCA_RSP_SIZE_32](#)
Verify command response packet size with validating MAC.

Definitions for the Write Command

- **#define WRITE_ZONE_IDX** [ATCA_PARAM1_IDX](#)
Write command index for zone.
- **#define WRITE_ADDR_IDX** [ATCA_PARAM2_IDX](#)
Write command index for address.
- **#define WRITE_VALUE_IDX** [ATCA_DATA_IDX](#)
Write command index for data.
- **#define WRITE_MAC_VS_IDX** (9)
Write command index for MAC following short data.
- **#define WRITE_MAC_VL_IDX** (37)
Write command index for MAC following long data.
- **#define WRITE_MAC_SIZE** (32u)
Write MAC size.
- **#define WRITE_ZONE_MASK** ((uint8_t)0xC3)

- *Write zone bits 2 to 5 are 0.*
- `#define WRITE_ZONE_WITH_MAC ((uint8_t)0x40)`
Write zone bit 6: write encrypted with MAC.
- `#define WRITE_ZONE_OTP ((uint8_t)1)`
Write zone id OTP.
- `#define WRITE_ZONE_DATA ((uint8_t)2)`
Write zone id data.
- `#define WRITE_RSP_SIZE ATCA_RSP_SIZE_MIN`
Write command response packet size.

Functions

- `ATCA_STATUS atInfo` (ATCADeviceType device_type, ATCAPacket *packet)
ATCACommand Info method.
- `ATCA_STATUS atPause` (ATCADeviceType device_type, ATCAPacket *packet)
ATCACommand Pause method.
- `bool atIsSHAFamily` (ATCADeviceType device_type)
determines if a given device type is a SHA device or a superset of a SHA device
- `bool atIsECCFamily` (ATCADeviceType device_type)
determines if a given device type is an ECC device or a superset of a ECC device
- `ATCA_STATUS isATCAError` (uint8_t *data)
checks for basic error frame in data
- `void atCRC` (size_t length, const uint8_t *data, uint8_t *crc_le)
Calculates CRC over the given raw data and returns the CRC in little-endian byte order.
- `void atCalcCrc` (ATCAPacket *packet)
This function calculates CRC and adds it to the correct offset in the packet data.
- `ATCA_STATUS atCheckCrc` (const uint8_t *response)
This function checks the consistency of a response.

24.79.1 Detailed Description

Microchip Crypto Auth device command object - this is a command builder only, it does not send the command. The result of a command method is a fully formed packet, ready to send to the ATCAIFace object to dispatch.

This command object supports the ATSHA and ATECC device family. The command list is a superset of all device commands for this family. The command object differentiates the packet contents based on specific device type within the family.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.79.2 Function Documentation

24.79.2.1 atCalcCrc()

```
void atCalcCrc (
    ATCAPacket * packet )
```

This function calculates CRC and adds it to the correct offset in the packet data.

Parameters

in	<i>packet</i>	Packet to calculate CRC data for
----	---------------	----------------------------------

24.79.2.2 atCheckCrc()

```
ATCA_STATUS atCheckCrc (
    const uint8_t * response )
```

This function checks the consistency of a response.

Parameters

in	<i>response</i>	pointer to response
----	-----------------	---------------------

Returns

ATCA_SUCCESS on success, otherwise ATCA_RX_CRC_ERROR

24.79.2.3 atCRC()

```
void atCRC (
    size_t length,
    const uint8_t * data,
    uint8_t * crc_le )
```

Calculates CRC over the given raw data and returns the CRC in little-endian byte order.

Parameters

in	<i>length</i>	Size of data not including the CRC byte positions
in	<i>data</i>	Pointer to the data over which to compute the CRC
out	<i>crc_le</i>	Pointer to the place where the two-bytes of CRC will be returned in little-endian byte order.

24.79.2.4 atInfo()

```
ATCA_STATUS atInfo (
    ATCADeviceType device_type,
    ATCAPacket * packet )
```

ATCACommand Info method.

Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

Returns

ATCA_SUCCESS

24.79.2.5 atIsECCFamily()

```
bool atIsECCFamily (
    ATCADeviceType device_type )
```

determines if a given device type is an ECC device or a superset of a ECC device

Parameters

in	<i>device_type</i>	Type of device to check for family type
----	--------------------	---

Returns

boolean indicating whether the given device is an ECC family device.

24.79.2.6 atIsSHAFamily()

```
bool atIsSHAFamily (
    ATCADeviceType device_type )
```

determines if a given device type is a SHA device or a superset of a SHA device

Parameters

in	<i>device_type</i>	Type of device to check for family type
----	--------------------	---

Returns

boolean indicating whether the given device is a SHA family device.

24.79.2.7 atPause()

```
ATCA_STATUS atPause (
    ATCADeviceType device_type,
    ATCAPacket * packet )
```

ATCACommand Pause method.

Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

Returns

ATCA_SUCCESS

24.79.2.8 isATCAError()

```
ATCA_STATUS isATCAError (
    uint8_t * data )
```

checks for basic error frame in data

Parameters

in	<i>data</i>	pointer to received data - expected to be in the form of a CA device response frame
----	-------------	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.80 calib_config_check.h File Reference

Consistency checks for configuration options.

```
#include "atca_config_check.h"
#include "crypto/crypto_sw_config_check.h"
```

Macros

- `#define CALIB_SHA204_EN DEFAULT_ENABLED`
- `#define CALIB_SHA206_EN DEFAULT_ENABLED`
- `#define CALIB_ECC108_EN DEFAULT_DISABLED`
- `#define CALIB_ECC508_EN DEFAULT_ENABLED`

- #define **CALIB_ECC608_EN** DEFAULT_ENABLED
- #define **CALIB_ECC204_EN** DEFAULT_ENABLED
- #define **CALIB_TA010_EN** DEFAULT_ENABLED
- #define **CALIB_SHA104_EN** DEFAULT_ENABLED
- #define **CALIB_SHA105_EN** DEFAULT_ENABLED
- #define **CALIB_FULL_FEATURE** (CALIB_SHA204_EN || CALIB_ECC108_EN || CALIB_ECC508_EN || CALIB_ECC608_EN)
- #define **CALIB_ECC_SUPPORT** (CALIB_ECC108_EN || CALIB_ECC508_EN || CALIB_ECC608_EN || CALIB_ECC204_EN || CALIB_TA010_EN)
- #define **CALIB_CA2_SUPPORT** (CALIB_ECC204_EN || CALIB_TA010_EN || CALIB_SHA104_EN || CALIB_SHA105_EN)
- #define **CALIB_CA2_CERT_SUPPORT** (CALIB_ECC204_EN || CALIB_TA010_EN)
- #define **CALIB_SHA206_ONLY** (CALIB_SHA206_EN && !(CALIB_FULL_FEATURE || ATCA_CA2_SUPPORT))
- #define **DEFAULT_CA_MAX_PACKET_SIZE** (198u)
- #define **CA_MAX_PACKET_SIZE** (DEFAULT_CA_MAX_PACKET_SIZE)
- #define **CALIB_AES_EN** (ATCAB_AES_EN && CALIB_ECC608_EN)
- #define **CALIB_AES_GCM_EN** (ATCAB_AES_GCM_EN && CALIB_AES_EN && CALIB_ECC608_EN)
- #define **CALIB_CHECKMAC_EN** (ATCAB_CHECKMAC_EN && (CALIB_FULL_FEATURE || CALIB_SHA105_EN))
- #define **CALIB_COUNTER_EN** (ATCAB_COUNTER_EN && (CALIB_ECC_SUPPORT || CALIB_SHA104_EN || CALIB_SHA105_EN))
- #define **CALIB_DELETE_EN** (DEFAULT_DISABLED)
- #define **CALIB_DERIVEKEY_EN** (ATCAB_DERIVEKEY_EN && (CALIB_FULL_FEATURE || CALIB_SHA206_EN))
- #define **CALIB_ECDH_EN** (ATCAB_ECDH_EN && (CALIB_ECC508_EN || CALIB_ECC608_EN))
- #define **CALIB_ECDH_ENC_EN** (ATCAB_ECDH_ENC_EN && (CALIB_ECC508_EN || CALIB_ECC608_EN))
- #define **CALIB_GENDIG_EN** (ATCAB_GENDIG_EN && (CALIB_FULL_FEATURE || CALIB_SHA105_EN))
- #define **CALIB_GENDIVKEY_EN** (ATCAB_GENDIG_EN && CALIB_SHA105_EN)
- #define **CALIB_GENKEY_EN** (ATCAB_GENKEY_EN && CALIB_ECC_SUPPORT)
- #define **CALIB_GENKEY_MAC_EN** (ATCAB_GENKEY_MAC_EN && CALIB_ECC_SUPPORT)
- #define **CALIB_HMAC_EN** (ATCAB_HMAC_EN && (CALIB_SHA204_EN || CALIB_ECC108_EN || CALIB_ECC508_EN))
- #define **CALIB_INFO_LATCH_EN** ATCAB_INFO_LATCH_EN
- #define **CALIB_KDF_EN** (ATCAB_KDF_EN && CALIB_ECC608_EN)
- #define **CALIB_LOCK_EN** (ATCAB_LOCK_EN && CALIB_FULL_FEATURE)
- #define **CALIB_LOCK_CA2_EN** (ATCAB_LOCK_EN && ATCA_CA2_SUPPORT)
- #define **CALIB_MAC_EN** (ATCAB_MAC_EN && (CALIB_FULL_FEATURE || CALIB_SHA206_EN || CALIB_SHA104_EN))
- #define **CALIB_NONCE_EN** (ATCAB_NONCE_EN && (CALIB_FULL_FEATURE || CALIB_CA2_SUPPORT))
- #define **CALIB_PRIVWRITE_EN** (ATCAB_PRIVWRITE_EN && (CALIB_ECC108_EN || CALIB_ECC508_EN || CALIB_ECC608_EN))
- #define **CALIB_RANDOM_EN** (ATCAB_RANDOM_EN && CALIB_FULL_FEATURE)
- #define **CALIB_READ_EN** (ATCAB_READ_EN && (CALIB_FULL_FEATURE || CALIB_SHA206_EN))
- #define **CALIB_READ_CA2_EN** (ATCAB_READ_EN && CALIB_CA2_SUPPORT)
- #define **CALIB_READ_ENC_EN** (ATCAB_READ_ENC_EN && CALIB_FULL_FEATURE)
- #define **CALIB_SECUREBOOT_EN** (ATCAB_SECUREBOOT_EN && CALIB_ECC608_EN)
- #define **CALIB_SECUREBOOT_MAC_EN** (ATCAB_SECUREBOOT_MAC_EN && CALIB_ECC608_EN)
- #define **CALIB_SELFTEST_EN** (ATCAB_SELFTEST_EN && (CALIB_ECC608_EN || CALIB_CA2_SUPPORT))
- #define **CALIB_SHA_EN** (ATCAB_SHA_EN && (CALIB_FULL_FEATURE || CALIB_CA2_SUPPORT))
- #define **CALIB_SHA_HMAC_EN** (ATCAB_SHA_HMAC_EN && CALIB_ECC_SUPPORT)
- #define **CALIB_SHA_CONTEXT_EN** (ATCAB_SHA_CONTEXT_EN && CALIB_ECC608_EN)

- `#define CALIB_SIGN_EN (ATCAB_SIGN_EN && (CALIB_ECC108_EN || CALIB_ECC508_EN || CALIB_ECC608_EN))`
- `#define CALIB_SIGN_CA2_EN (ATCAB_SIGN_EN && (CALIB_ECC204_EN || CALIB_TA010_EN))`
- `#define CALIB_SIGN_INTERNAL_EN (ATCAB_SIGN_INTERNAL_EN && CALIB_SIGN_EN)`
- `#define CALIB_UPDATEEXTRA_EN (ATCAB_UPDATEEXTRA_EN && CALIB_FULL_FEATURE)`
- `#define CALIB_VERIFY_EN (ATCAB_VERIFY_EN && (CALIB_ECC108_EN || CALIB_ECC508_EN || CALIB_ECC608_EN))`
- `#define CALIB_VERIFY_MAC_EN (ATCAB_VERIFY_MAC_EN && CALIB_ECC608_EN)`
- `#define CALIB_VERIFY_EXTERN_EN (ATCAB_VERIFY_EXTERN_EN && CALIB_VERIFY_EN)`
- `#define CALIB_VERIFY_STORED_EN (ATCAB_VERIFY_STORED_EN && CALIB_VERIFY_EN)`
- `#define CALIB_VERIFY_VALIDATE_EN (ATCAB_VERIFY_VALIDATE_EN && CALIB_VERIFY_EN)`
- `#define CALIB_WRITE_EN (ATCAB_WRITE_EN && (CALIB_FULL_FEATURE || CALIB_SHA206_EN))`
- `#define CALIB_WRITE_ENC_EN (ATCAB_WRITE_ENC_EN && CALIB_FULL_FEATURE)`
- `#define CALIB_WRITE_CA2_EN (ATCAB_WRITE_EN && CALIB_CA2_SUPPORT)`

24.80.1 Detailed Description

Consistency checks for configuration options.

Copyright

(c) 2015-2021 Microchip Technology Inc. and its subsidiaries.

24.80.2 Macro Definition Documentation

24.80.2.1 CALIB_INFO_LATCH_EN

```
#define CALIB_INFO_LATCH_EN ATCAB_INFO_LATCH_EN
```

Supported API's: `calib_info_get_latch` `calib_info_set_latch`

ECC204 specific api: `calib_info_lock_status`

24.80.2.2 CALIB_LOCK_CA2_EN

```
#define CALIB_LOCK_CA2_EN (ATCAB_LOCK_EN && ATCA_CA2_SUPPORT)
```

Enable `CALIB_LOCK_CA2_EN` which enables the lock command for the ecc204 and ta010 devices

Supported API's: `calib_lock`

24.80.2.3 CALIB_LOCK_EN

```
#define CALIB_LOCK_EN (ATCAB_LOCK_EN && CALIB_FULL_FEATURE)
```

Enable `CALIB_LOCK_EN` to enable the lock commands for the classic cryptoauth parts

Supported API's: `calib_lock`

24.80.2.4 CALIB_READ_EN

```
#define CALIB_READ_EN (ATCAB_READ_EN && (CALIB_FULL_FEATURE || CALIB_SHA206_EN))
```

Enable CALIB_READ_EN which enables the read commands

Supported API's: calib_read_zone

24.80.2.5 CALIB_SHA_CONTEXT_EN

```
#define CALIB_SHA_CONTEXT_EN (ATCAB_SHA_CONTEXT_EN && CALIB_ECC608_EN)
```

Requires: CALIB_SHA_BASE

Use the SHA command to compute an HMAC/SHA-256 operation

Supported API's: calib_sha_read_context

24.80.2.6 CALIB_SHA_EN

```
#define CALIB_SHA_EN (ATCAB_SHA_EN && (CALIB_FULL_FEATURE || CALIB_CA2_SUPPORT))
```

Enable CALIB_SHA_EN to compute a SHA-256 or HMAC/SHA-256 digest for general purpose use by the host system

Supported API's: calib_sha_base

24.80.2.7 CALIB_SHA_HMAC_EN

```
#define CALIB_SHA_HMAC_EN (ATCAB_SHA_HMAC_EN && CALIB_ECC_SUPPORT)
```

Requires: CALIB_SHA_HMAC CALIB_SHA_BASE

Use the SHA command to compute an HMAC/SHA-256 operation

Supported API's: calib_sha_hmac, calib_sha_hmac_init, calib_sha_hmac_update, calib_sha_hmac_finish

24.80.2.8 CALIB_SIGN_CA2_EN

```
#define CALIB_SIGN_CA2_EN (ATCAB_SIGN_EN && (CALIB_ECC204_EN || CALIB_TA010_EN))
```

Enable CALIB_SIGN_CA2_EN to generate a signature using the ECDSA algorithm

Supported API's: calib_sign_base

24.80.2.9 CALIB_SIGN_EN

```
#define CALIB_SIGN_EN (ATCAB_SIGN_EN && (CALIB_ECC108_EN || CALIB_ECC508_EN || CALIB_ECC608_EN))
```

Enable CALIB_SIGN_EN to generate a signature using the ECDSA algorithm

Supported API's: calib_sign

24.80.2.10 CALIB_UPDATEEXTRA_EN

```
#define CALIB_UPDATEEXTRA_EN (ATCAB_UPDATEEXTRA_EN && CALIB_FULL_FEATURE)
```

Enable CALIB_UPDATEEXTRA_EN to update the values of the two extra bytes within the configuration zone (bytes 84 and 85)

Supported API's: calib_updateextra

24.80.2.11 CALIB_VERIFY_EN

```
#define CALIB_VERIFY_EN (ATCAB_VERIFY_EN && (CALIB_ECC108_EN || CALIB_ECC508_EN || CALIB_ECC608_↵_EN))
```

Enable CALIB_VERIFY_EN which takes an ECDSA [R,S] signature and verifies that it is correctly generated from a given message and public key. In all cases, the signature is an input to the command

Supported API's: calib_verify

24.80.2.12 CALIB_VERIFY_MAC_EN

```
#define CALIB_VERIFY_MAC_EN (ATCAB_VERIFY_MAC_EN && CALIB_ECC608_EN)
```

Requires: CALIB_NONCE_MODE_ENCODING CALIB_NONCE_BASE ATCAH_VERIFY_MAC ATCAC_SW_↵SHA2_256 CALIB_VERIFY

Executes verification command with verification MAC for the External or Stored Verify modes

Supported API's: calib_verify_extern_stored_mac, calib_verify_extern_mac, calib_verify_stored_mac

24.80.2.13 CALIB_VERIFY_STORED_EN

```
#define CALIB_VERIFY_STORED_EN (ATCAB_VERIFY_STORED_EN && CALIB_VERIFY_EN)
```

Requires: CALIB_NONCE_MODE_ENCODING CALIB_NONCE_BASE CALIB_VERIFY

Verifies a signature (ECDSA verify operation) with a public key stored in the device

Supported API's: calib_verify_stored

24.80.2.14 CALIB_WRITE_ENC_EN

```
#define CALIB_WRITE_ENC_EN (ATCAB_WRITE_ENC_EN && CALIB_FULL_FEATURE)
```

Requires: CALIB_NONCE_MODE_ENCODING CALIB_NONCE_BASE CALIB_READ_ZONE CALIB_GENDIG ATCAH_GENDIG ATCAH_WRITE_AUTH_MAC ATCAH_NONCE ATCAC_SW_SHA2_256 CALIB_WRITE ATCAH_GEN_SESSION_KEY

Performs an encrypted write of a 32 byte block into given slot

Supported API's: calib_write_enc

24.81 calib_counter.c File Reference

CryptoAuthLib Basic API methods for Counter command.

```
#include "cryptoauthlib.h"
```

24.81.1 Detailed Description

CryptoAuthLib Basic API methods for Counter command.

The Counter command reads or increments the binary count value for one of the two monotonic counters

Note

List of devices that support this command - ATECC508A and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.82 calib_delete.c File Reference

CryptoAuthLib Basic API methods for Delete command.

```
#include "cryptoauthlib.h"  
#include "host/atca_host.h"
```

24.82.1 Detailed Description

CryptoAuthLib Basic API methods for Delete command.

The Delete command, when executed, will clear all of the Data zone slots and set all bytes of each slot to 0xFF. The Configuration zone will be untouched, except for the value of the Primary_Deleted byte.

Note

List of devices that support this command - ECC204, TA010, SHA10x. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.83 calib_derivekey.c File Reference

CryptoAuthLib Basic API methods for DeriveKey command.

```
#include "cryptoauthlib.h"
```

24.83.1 Detailed Description

CryptoAuthLib Basic API methods for DeriveKey command.

The DeriveKey command combines the current value of a key with the nonce stored in TempKey using SHA-256 and derives a new key.

Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.84 calib_device.h File Reference

Microchip Crypto Auth Device Data.

```
#include <stdint.h>
#include "atca_compiler.h"
```

Data Structures

- struct [atsha204a_config_s](#)
- struct [atecc508a_config_s](#)
- struct [atecc608_config_s](#)

Macros

- `#define ATCA_AES_ENABLE_EN_SHIFT (0)`
- `#define ATCA_AES_ENABLE_EN_MASK (0x01u << ATCA_AES_ENABLE_EN_SHIFT)`
- `#define ATCA_I2C_ENABLE_EN_SHIFT (0)`
- `#define ATCA_I2C_ENABLE_EN_MASK (0x01u << ATCA_I2C_ENABLE_EN_SHIFT)`
- `#define ATCA_COUNTER_MATCH_EN_SHIFT (0)`
- `#define ATCA_COUNTER_MATCH_EN_MASK (0x01u << ATCA_COUNTER_MATCH_EN_SHIFT)`
- `#define ATCA_COUNTER_MATCH_KEY_SHIFT (4)`
- `#define ATCA_COUNTER_MATCH_KEY_MASK (0x0Fu << ATCA_COUNTER_MATCH_KEY_SHIFT)`
- `#define ATCA_COUNTER_MATCH_KEY(v) (ATCA_COUNTER_MATCH_KEY_MASK & ((v) << ATCA_COUNTER_MATCH_KEY_SHIFT))`
- `#define ATCA_CHIP_MODE_I2C_EXTRA_SHIFT (0)`
- `#define ATCA_CHIP_MODE_I2C_EXTRA_MASK (0x01u << ATCA_CHIP_MODE_I2C_EXTRA_SHIFT)`
- `#define ATCA_CHIP_MODE_TTL_EN_SHIFT (1)`
- `#define ATCA_CHIP_MODE_TTL_EN_MASK (0x01u << ATCA_CHIP_MODE_TTL_EN_SHIFT)`
- `#define ATCA_CHIP_MODE_WDG_LONG_SHIFT (2)`
- `#define ATCA_CHIP_MODE_WDG_LONG_MASK (0x01u << ATCA_CHIP_MODE_WDG_LONG_SHIFT)`
- `#define ATCA_CHIP_MODE_CLK_DIV_SHIFT (3)`
- `#define ATCA_CHIP_MODE_CLK_DIV_MASK (0x1Fu << ATCA_CHIP_MODE_CLK_DIV_SHIFT)`
- `#define ATCA_CHIP_MODE_CLK_DIV(v) (ATCA_CHIP_MODE_CLK_DIV_MASK & ((v) << ATCA_CHIP_MODE_CLK_DIV_SHIFT))`
- `#define ATCA_SLOT_CONFIG_READKEY_SHIFT (0)`
- `#define ATCA_SLOT_CONFIG_READKEY_MASK (0x0Fu << ATCA_SLOT_CONFIG_READKEY_SHIFT)`
- `#define ATCA_SLOT_CONFIG_READKEY(v) (ATCA_SLOT_CONFIG_READKEY_MASK & ((v) << ATCA_SLOT_CONFIG_READKEY_SHIFT))`
- `#define ATCA_SLOT_CONFIG_NOMAC_SHIFT (4)`
- `#define ATCA_SLOT_CONFIG_NOMAC_MASK (0x01u << ATCA_SLOT_CONFIG_NOMAC_SHIFT)`
- `#define ATCA_SLOT_CONFIG_LIMITED_USE_SHIFT (5)`
- `#define ATCA_SLOT_CONFIG_LIMITED_USE_MASK (0x01u << ATCA_SLOT_CONFIG_LIMITED_USE_SHIFT)`
- `#define ATCA_SLOT_CONFIG_ENC_READ_SHIFT (6)`
- `#define ATCA_SLOT_CONFIG_ENC_READ_MASK (0x01u << ATCA_SLOT_CONFIG_ENC_READ_SHIFT)`
- `#define ATCA_SLOT_CONFIG_IS_SECRET_SHIFT (7)`
- `#define ATCA_SLOT_CONFIG_IS_SECRET_MASK (0x01u << ATCA_SLOT_CONFIG_IS_SECRET_SHIFT)`
- `#define ATCA_SLOT_CONFIG_WRITE_KEY_SHIFT (8)`
- `#define ATCA_SLOT_CONFIG_WRITE_KEY_MASK (((uint32_t)0x0Fu << ATCA_SLOT_CONFIG_WRITE_KEY_SHIFT))`
- `#define ATCA_SLOT_CONFIG_WRITE_KEY(v) (ATCA_SLOT_CONFIG_WRITE_KEY_MASK & ((v) << ATCA_SLOT_CONFIG_WRITE_KEY_SHIFT))`
- `#define ATCA_SLOT_CONFIG_WRITE_CONFIG_SHIFT (12)`
- `#define ATCA_SLOT_CONFIG_WRITE_CONFIG_MASK (((uint32_t)0x0Fu << ATCA_SLOT_CONFIG_WRITE_CONFIG_SHIFT))`
- `#define ATCA_SLOT_CONFIG_WRITE_CONFIG(v) ((ATCA_SLOT_CONFIG_WRITE_CONFIG_MASK & ((uint32_t)v) << ATCA_SLOT_CONFIG_WRITE_CONFIG_SHIFT))`
- `#define ATCA_SLOT_CONFIG_EXT_SIG_SHIFT (0)`
- `#define ATCA_SLOT_CONFIG_EXT_SIG_MASK (0x01u << ATCA_SLOT_CONFIG_EXT_SIG_SHIFT)`
- `#define ATCA_SLOT_CONFIG_INT_SIG_SHIFT (1)`
- `#define ATCA_SLOT_CONFIG_INT_SIG_MASK (0x01u << ATCA_SLOT_CONFIG_INT_SIG_SHIFT)`
- `#define ATCA_SLOT_CONFIG_ECDH_SHIFT (2)`
- `#define ATCA_SLOT_CONFIG_ECDH_MASK (0x01u << ATCA_SLOT_CONFIG_ECDH_SHIFT)`
- `#define ATCA_SLOT_CONFIG_WRITE_ECDH_SHIFT (3)`

- `#define ATCA_SLOT_CONFIG_WRITE_ECDH_MASK (0x01u << ATCA_SLOT_CONFIG_WRITE_ECDH_SHIFT)`
- `#define ATCA_SLOT_CONFIG_GEN_KEY_SHIFT (8)`
- `#define ATCA_SLOT_CONFIG_GEN_KEY_MASK (0x01u << ATCA_SLOT_CONFIG_GEN_KEY_SHIFT)`
- `#define ATCA_SLOT_CONFIG_PRIV_WRITE_SHIFT (9)`
- `#define ATCA_SLOT_CONFIG_PRIV_WRITE_MASK (0x01u << ATCA_SLOT_CONFIG_PRIV_WRITE_SHIFT)`
- `#define ATCA_USE_LOCK_ENABLE_SHIFT (0)`
- `#define ATCA_USE_LOCK_ENABLE_MASK (0x0Fu << ATCA_USE_LOCK_ENABLE_SHIFT)`
- `#define ATCA_USE_LOCK_KEY_SHIFT (4)`
- `#define ATCA_USE_LOCK_KEY_MASK (0x0Fu << ATCA_USE_LOCK_KEY_SHIFT)`
- `#define ATCA_VOL_KEY_PERM_SLOT_SHIFT (0)`
- `#define ATCA_VOL_KEY_PERM_SLOT_MASK (0x0Fu << ATCA_VOL_KEY_PERM_SLOT_SHIFT)`
- `#define ATCA_VOL_KEY_PERM_SLOT(v) (ATCA_VOL_KEY_PERM_SLOT_MASK & ((v) << ATCA_VOL_KEY_PERM_SLOT_SHIFT))`
- `#define ATCA_VOL_KEY_PERM_EN_SHIFT (7)`
- `#define ATCA_VOL_KEY_PERM_EN_MASK (0x01u << ATCA_VOL_KEY_PERM_EN_SHIFT)`
- `#define ATCA_SECURE_BOOT_MODE_SHIFT (0)`
- `#define ATCA_SECURE_BOOT_MODE_MASK (0x03u << ATCA_SECURE_BOOT_MODE_SHIFT)`
- `#define ATCA_SECURE_BOOT_MODE(v) (ATCA_SECURE_BOOT_MODE_MASK & ((v) << ATCA_SECURE_BOOT_MODE_SHIFT))`
- `#define ATCA_SECURE_BOOT_PERSIST_EN_SHIFT (3)`
- `#define ATCA_SECURE_BOOT_PERSIST_EN_MASK (0x01u << ATCA_SECURE_BOOT_PERSIST_EN_SHIFT)`
- `#define ATCA_SECURE_BOOT_RAND_NONCE_SHIFT (4)`
- `#define ATCA_SECURE_BOOT_RAND_NONCE_MASK (0x01u << ATCA_SECURE_BOOT_RAND_NONCE_SHIFT)`
- `#define ATCA_SECURE_BOOT_DIGEST_SHIFT (8)`
- `#define ATCA_SECURE_BOOT_DIGEST_MASK (0x0Fu << ATCA_SECURE_BOOT_DIGEST_SHIFT)`
- `#define ATCA_SECURE_BOOT_DIGEST(v) (ATCA_SECURE_BOOT_DIGEST_MASK & ((v) << ATCA_SECURE_BOOT_DIGEST_SHIFT))`
- `#define ATCA_SECURE_BOOT_PUB_KEY_SHIFT (12)`
- `#define ATCA_SECURE_BOOT_PUB_KEY_MASK (0x0Fu << ATCA_SECURE_BOOT_PUB_KEY_SHIFT)`
- `#define ATCA_SECURE_BOOT_PUB_KEY(v) (ATCA_SECURE_BOOT_PUB_KEY_MASK & ((v) << ATCA_SECURE_BOOT_PUB_KEY_SHIFT))`
- `#define ATCA_SLOT_LOCKED(v) ((0x01 << (v)) & 0xFFFFu)`
- `#define ATCA_CHIP_OPT_POST_EN_SHIFT (0)`
- `#define ATCA_CHIP_OPT_POST_EN_MASK (0x01u << ATCA_CHIP_OPT_POST_EN_SHIFT)`
- `#define ATCA_CHIP_OPT_IO_PROT_EN_SHIFT (1)`
- `#define ATCA_CHIP_OPT_IO_PROT_EN_MASK (0x01u << ATCA_CHIP_OPT_IO_PROT_EN_SHIFT)`
- `#define ATCA_CHIP_OPT_KDF_AES_EN_SHIFT (2)`
- `#define ATCA_CHIP_OPT_KDF_AES_EN_MASK (0x01u << ATCA_CHIP_OPT_KDF_AES_EN_SHIFT)`
- `#define ATCA_CHIP_OPT_ECDH_PROT_SHIFT (8)`
- `#define ATCA_CHIP_OPT_ECDH_PROT_MASK (0x03u << ATCA_CHIP_OPT_ECDH_PROT_SHIFT)`
- `#define ATCA_CHIP_OPT_ECDH_PROT(v) (ATCA_CHIP_OPT_ECDH_PROT_MASK & ((v) << ATCA_CHIP_OPT_ECDH_PROT_SHIFT))`
- `#define ATCA_CHIP_OPT_KDF_PROT_SHIFT (10)`
- `#define ATCA_CHIP_OPT_KDF_PROT_MASK (0x03u << ATCA_CHIP_OPT_KDF_PROT_SHIFT)`
- `#define ATCA_CHIP_OPT_KDF_PROT(v) (ATCA_CHIP_OPT_KDF_PROT_MASK & ((v) << ATCA_CHIP_OPT_KDF_PROT_SHIFT))`
- `#define ATCA_CHIP_OPT_IO_PROT_KEY_SHIFT (12)`
- `#define ATCA_CHIP_OPT_IO_PROT_KEY_MASK ((uint16_t)0x0Fu << ATCA_CHIP_OPT_IO_PROT_KEY_SHIFT)`
- `#define ATCA_CHIP_OPT_IO_PROT_KEY(v) (ATCA_CHIP_OPT_IO_PROT_KEY_MASK & ((v) << ATCA_CHIP_OPT_IO_PROT_KEY_SHIFT))`

- #define **ATCA_KEY_CONFIG_OFFSET**(x) (96UL + (x) * 2u)
- #define **ATCA_KEY_CONFIG_PRIVATE_SHIFT** (0)
- #define **ATCA_KEY_CONFIG_PRIVATE_MASK** (0x01u << ATCA_KEY_CONFIG_PRIVATE_SHIFT)
- #define **ATCA_KEY_CONFIG_PUB_INFO_SHIFT** (1)
- #define **ATCA_KEY_CONFIG_PUB_INFO_MASK** (0x01u << ATCA_KEY_CONFIG_PUB_INFO_SHIFT)
- #define **ATCA_KEY_CONFIG_KEY_TYPE_SHIFT** (2)
- #define **ATCA_KEY_CONFIG_KEY_TYPE_MASK** ((0x07u << ATCA_KEY_CONFIG_KEY_TYPE_SHIFT))
- #define **ATCA_KEY_CONFIG_KEY_TYPE**(v) ((ATCA_KEY_CONFIG_KEY_TYPE_MASK & ((v) << ATCA_KEY_CONFIG_KEY_TYPE_SHIFT)))
- #define **ATCA_KEY_CONFIG_LOCKABLE_SHIFT** (5)
- #define **ATCA_KEY_CONFIG_LOCKABLE_MASK** (0x01u << ATCA_KEY_CONFIG_LOCKABLE_SHIFT)
- #define **ATCA_KEY_CONFIG_REQ_RANDOM_SHIFT** (6)
- #define **ATCA_KEY_CONFIG_REQ_RANDOM_MASK** (0x01u << ATCA_KEY_CONFIG_REQ_RANDOM_SHIFT)
- #define **ATCA_KEY_CONFIG_REQ_AUTH_SHIFT** (7)
- #define **ATCA_KEY_CONFIG_REQ_AUTH_MASK** (0x01u << ATCA_KEY_CONFIG_REQ_AUTH_SHIFT)
- #define **ATCA_KEY_CONFIG_AUTH_KEY_SHIFT** (8)
- #define **ATCA_KEY_CONFIG_AUTH_KEY_MASK** (0x0Fu << ATCA_KEY_CONFIG_AUTH_KEY_SHIFT)
- #define **ATCA_KEY_CONFIG_AUTH_KEY**(v) (ATCA_KEY_CONFIG_AUTH_KEY_MASK & ((v) << ATCA_KEY_CONFIG_AUTH_KEY_SHIFT))
- #define **ATCA_KEY_CONFIG_PERSIST_DIS_SHIFT** (12)
- #define **ATCA_KEY_CONFIG_PERSIST_DIS_MASK** (0x01u << ATCA_KEY_CONFIG_PERSIST_DIS_SHIFT)
- #define **ATCA_KEY_CONFIG_RFU_SHIFT** (13)
- #define **ATCA_KEY_CONFIG_RFU_MASK** (0x01u << ATCA_KEY_CONFIG_RFU_SHIFT)
- #define **ATCA_KEY_CONFIG_X509_ID_SHIFT** (14)
- #define **ATCA_KEY_CONFIG_X509_ID_MASK** (0x03u << ATCA_KEY_CONFIG_X509_ID_SHIFT)
- #define **ATCA_KEY_CONFIG_X509_ID**(v) (ATCA_KEY_CONFIG_X509_ID_MASK & ((v) << ATCA_KEY_CONFIG_X509_ID_SHIFT))

Typedefs

- typedef struct ATCA_PACKED [atsha204a_config_s](#) **atsha204a_config_t**
- typedef struct ATCA_PACKED [atecc508a_config_s](#) **atecc508a_config_t**
- typedef struct ATCA_PACKED [atecc608_config_s](#) **atecc608_config_t**

24.84.1 Detailed Description

Microchip Crypto Auth Device Data.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.85 calib_ecdh.c File Reference

CryptoAuthLib Basic API methods for ECDH command.

```
#include "cryptoauthlib.h"
#include "host/atca_host.h"
```

24.85.1 Detailed Description

CryptoAuthLib Basic API methods for ECDH command.

The ECDH command implements the Elliptic Curve Diffie-Hellman algorithm to combine an internal private key with an external public key to calculate a shared secret.

Note

List of devices that support this command - ATECC508A, ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.86 calib_execution.c File Reference

Implements an execution handler that executes a given command on a device and returns the results.

```
#include "cryptoauthlib.h"
```

Functions

- ATCA_STATUS [calib_get_execution_time](#) (uint8_t opcode, [ATCADevice](#) device)
return the typical execution time for the given command
- ATCA_STATUS [calib_execute_send](#) ([ATCADevice](#) device, uint8_t word_address, uint8_t *txdata, uint16_t txlength)
- ATCA_STATUS [calib_execute_receive](#) ([ATCADevice](#) device, uint8_t device_address, uint8_t *rxdata, uint16_t *rxlength)
- ATCA_STATUS [calib_execute_command](#) ([ATCAPacket](#) *packet, [ATCADevice](#) device)
Wakes up device, sends the packet, waits for command completion, receives response, and puts the device into the idle state.

24.86.1 Detailed Description

Implements an execution handler that executes a given command on a device and returns the results.

This implementation wraps Polling and No polling (simple wait) schemes into a single method and use it across the library. Polling is used by default, however, by defining the ATCA_NO_POLL symbol the code will instead wait an estimated max execution time before requesting the result.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.86.2 Function Documentation

24.86.2.1 calib_execute_command()

```
ATCA_STATUS calib_execute_command (  
    ATCAPacket * packet,  
    ATCADevice device )
```

Wakes up device, sends the packet, waits for command completion, receives response, and puts the device into the idle state.

Parameters

<code>in, out</code>	<i>packet</i>	As input, the packet to be sent. As output, the data buffer in the packet structure will contain the response.
<code>in</code>	<i>device</i>	CryptoAuthentication device to send the command to.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.86.2.2 calib_get_execution_time()

```
ATCA_STATUS calib_get_execution_time (
    uint8_t opcode,
    ATCADevice device )
```

return the typical execution time for the given command

Parameters

<code>in</code>	<i>opcode</i>	Opcode value of the command
<code>in</code>	<i>ca_cmd</i>	Command object for which the execution times are associated

Returns

ATCA_SUCCESS

24.87 calib_execution.h File Reference

Defines an execution handler that executes a given command on a device and returns the results.

```
#include "atca_status.h"
#include "calib_command.h"
#include "atca_device.h"
#include "atca_config.h"
```

Data Structures

- struct `device_execution_time_t`

Structure to hold the device execution time and the opcode for the corresponding command.

Macros

- #define **ATCA_UNSUPPORTED_CMD** ((uint16_t)0xFFFF)
- #define **CALIB_SWI_FLAG_WAKE** 0x00
flag preceding a command
- #define **CALIB_SWI_FLAG_CMD** 0x77
flag preceding a command
- #define **CALIB_SWI_FLAG_TX** 0x88
flag requesting a response
- #define **CALIB_SWI_FLAG_IDLE** 0xBB
flag requesting to go into Idle mode
- #define **CALIB_SWI_FLAG_SLEEP** 0xCC
flag requesting to go into Sleep mode

Functions

- ATCA_STATUS [calib_get_execution_time](#) (uint8_t opcode, [ATCADevice](#) device)
return the typical execution time for the given command
- ATCA_STATUS **calib_execute_send** ([ATCADevice](#) device, uint8_t word_address, uint8_t *txdata, uint16_t txlength)
- ATCA_STATUS **calib_execute_receive** ([ATCADevice](#) device, uint8_t device_address, uint8_t *rxdata, uint16_t *rxlength)
- ATCA_STATUS [calib_execute_command](#) ([ATCAPacket](#) *packet, [ATCADevice](#) device)
Wakes up device, sends the packet, waits for command completion, receives response, and puts the device into the idle state.

24.87.1 Detailed Description

Defines an execution handler that executes a given command on a device and returns the results.

The basic flow is to wake the device, send the command, wait/poll for completion, and finally receives the response from the device and does basic checks before returning to caller.

This handler supports the ATSHA and ATECC device family.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.87.2 Function Documentation

24.87.2.1 calib_execute_command()

```
ATCA_STATUS calib_execute_command (  
    ATCAPacket * packet,  
    ATCADevice device )
```

Wakes up device, sends the packet, waits for command completion, receives response, and puts the device into the idle state.

Parameters

<code>in, out</code>	<i>packet</i>	As input, the packet to be sent. As output, the data buffer in the packet structure will contain the response.
<code>in</code>	<i>device</i>	CryptoAuthentication device to send the command to.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.87.2.2 calib_get_execution_time()

```
ATCA_STATUS calib_get_execution_time (
    uint8_t opcode,
    ATCADevice device )
```

return the typical execution time for the given command

Parameters

<code>in</code>	<i>opcode</i>	Opcode value of the command
<code>in</code>	<i>ca_cmd</i>	Command object for which the execution times are associated

Returns

ATCA_SUCCESS

24.88 calib_gendig.c File Reference

CryptoAuthLib Basic API methods for GenDig command.

```
#include "cryptoauthlib.h"
```

24.88.1 Detailed Description

CryptoAuthLib Basic API methods for GenDig command.

The GenDig command uses SHA-256 to combine a stored value with the contents of TempKey, which must have been valid prior to the execution of this command.

Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.89 calib_genkey.c File Reference

CryptoAuthLib Basic API methods for GenKey command.

```
#include "cryptoauthlib.h"
```

24.89.1 Detailed Description

CryptoAuthLib Basic API methods for GenKey command.

The GenKey command is used for creating ECC private keys, generating ECC public keys, and for digest calculations involving public keys.

Note

List of devices that support this command - ATECC108A, ATECC508A, ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.90 calib_helpers.c File Reference

CryptoAuthLib Basic API - Helper Functions to.

```
#include "cryptoauthlib.h"
```

Functions

- ATCA_STATUS [calib_ca2_is_config_locked](#) ([ATCADevice](#) device, bool *is_locked)
Executes Read command, which reads the configuration zone to see if the specified slot is locked.
- ATCA_STATUS [calib_ca2_is_data_locked](#) ([ATCADevice](#) device, bool *is_locked)
Use Info command to check ECC204 Data zone lock status.
- ATCA_STATUS [calib_ca2_is_locked](#) ([ATCADevice](#) device, uint8_t zone, bool *is_locked)
Use Info command to check config/data is locked or not.
- ATCADeviceType [calib_get_devicetype](#) (uint8_t revision[4])
Parse the revision field to get the device type.
- ATCADeviceType [calib_get_devicetype_with_device_id](#) (uint8_t device_id, uint8_t device_revision)

24.90.1 Detailed Description

CryptoAuthLib Basic API - Helper Functions to.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.91 calib_hmac.c File Reference

CryptoAuthLib Basic API methods for HMAC command.

```
#include "cryptoauthlib.h"
```

24.91.1 Detailed Description

CryptoAuthLib Basic API methods for HMAC command.

The HMAC command computes an HMAC/SHA-256 digest using a key stored in the device over a challenge stored in the TempKey register, and/or other information stored within the device.

Note

List of devices that support this command - ATSHA204A, ATECC108A, and ATECC508A . There are differences in the modes that they support. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.92 calib_info.c File Reference

CryptoAuthLib Basic API methods for Info command.

```
#include "cryptoauthlib.h"
```

Functions

- ATCA_STATUS [calib_info_base](#) (ATCADevice device, uint8_t mode, uint16_t param2, uint8_t *out_data)
Issues an Info command, which return internal device information and can control GPIO and the persistent latch.
- ATCA_STATUS [calib_info](#) (ATCADevice device, uint8_t *revision)
Use the Info command to get the device revision (DevRev).
- ATCA_STATUS [calib_info_privkey_valid](#) (ATCADevice device, uint16_t key_id, uint8_t *is_valid)
Use Info command to check ECC Private key stored in key slot is valid or not.
- ATCA_STATUS [calib_info_lock_status](#) (ATCADevice device, uint16_t param2, uint8_t *is_locked)
Use Info command to ECC204,TA010 config/data zone lock status.
- ATCA_STATUS [calib_info_chip_status](#) (ATCADevice device, uint8_t *chip_status)
Use Info command to get ECC204,TA010,SHA10x chip status.

24.92.1 Detailed Description

CryptoAuthLib Basic API methods for Info command.

Info command returns a variety of static and dynamic information about the device and its state. Also is used to control the GPIO pin and the persistent latch.

Note

The ATSHA204A refers to this command as DevRev instead of Info, however, the OpCode and operation is the same.

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A & ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.93 calib_kdf.c File Reference

CryptoAuthLib Basic API methods for KDF command.

```
#include "cryptoauthlib.h"
```

24.93.1 Detailed Description

CryptoAuthLib Basic API methods for KDF command.

The KDF command implements one of a number of Key Derivation Functions (KDF). Generally this function combines a source key with an input string and creates a result key/digest/array. Three algorithms are currently supported: PRF, HKDF and AES.

Note

List of devices that support this command - ATECC608A/B. Refer to device datasheet for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.94 calib_lock.c File Reference

CryptoAuthLib Basic API methods for Lock command.

```
#include "cryptoauthlib.h"
```

24.94.1 Detailed Description

CryptoAuthLib Basic API methods for Lock command.

The Lock command prevents future modifications of the Configuration zone, enables configured policies for Data and OTP zones, and can render individual slots read-only regardless of configuration.

Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.95 calib_mac.c File Reference

CryptoAuthLib Basic API methods for MAC command.

```
#include "cryptoauthlib.h"
```

24.95.1 Detailed Description

CryptoAuthLib Basic API methods for MAC command.

The MAC command computes a SHA-256 digest of a key stored in the device, a challenge, and other information on the device. The output of this command is the digest of this message.

Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.96 calib_nonce.c File Reference

CryptoAuthLib Basic API methods for Nonce command.

```
#include "cryptoauthlib.h"
```

24.96.1 Detailed Description

CryptoAuthLib Basic API methods for Nonce command.

The Nonce command generates a nonce for use by a subsequent commands of the device by combining an internally generated random number with an input value from the system.

Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.97 calib_packet.c File Reference

CryptoAuthLib API for packet allocation.

```
#include "cryptoauthlib.h"
#include "calib_packet.h"
```

Functions

- [ATCAPacket](#) * **calib_packet_alloc** (void)
- void **calib_packet_free** ([ATCAPacket](#) *packet)

24.97.1 Detailed Description

CryptoAuthLib API for packet allocation.

The APIs are used for allocating packets in heap or bss according to atcab heap availability. Corresponding memory free is done

Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, ATECC608A/B

Copyright

(c) 2024 Microchip Technology Inc. and its subsidiaries.

24.98 calib_packet.h File Reference

Defines packet allocation functions.

```
#include "calib_command.h"
#include "atca_device.h"
#include "atca_config.h"
```

Data Structures

- struct [calib_packet_cache_s](#)

Typedefs

- typedef struct [calib_packet_cache_s](#) [calib_packet_cache_t](#)

Functions

- [ATCAPacket](#) * [calib_packet_alloc](#) (void)
- void [calib_packet_free](#) ([ATCAPacket](#) *packet)

24.98.1 Detailed Description

Defines packet allocation functions.

The APIs are used for allocating packets in heap or bss according to atcab heap availability. Corresponding memory free is done

This supports the ATECC device family.

Copyright

(c) 2024 Microchip Technology Inc. and its subsidiaries.

24.99 calib_privwrite.c File Reference

CryptoAuthLib Basic API methods for PrivWrite command.

```
#include "cryptoauthlib.h"
```

24.99.1 Detailed Description

CryptoAuthLib Basic API methods for PrivWrite command.

The PrivWrite command is used to write externally generated ECC private keys into the device.

Note

List of devices that support this command - ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.100 calib_random.c File Reference

CryptoAuthLib Basic API methods for Random command.

```
#include "cryptoauthlib.h"
```

24.100.1 Detailed Description

CryptoAuthLib Basic API methods for Random command.

The Random command generates a random number for use by the system.

Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.101 calib_read.c File Reference

CryptoAuthLib Basic API methods for Read command.

```
#include "cryptoauthlib.h"
```

24.101.1 Detailed Description

CryptoAuthLib Basic API methods for Read command.

The Read command reads words either 4-byte words or 32-byte blocks from one of the memory zones of the device. The data may optionally be encrypted before being returned to the system.

Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.102 calib_secureboot.c File Reference

CryptoAuthLib Basic API methods for SecureBoot command.

```
#include "cryptoauthlib.h"
```

24.102.1 Detailed Description

CryptoAuthLib Basic API methods for SecureBoot command.

The SecureBoot command provides support for secure boot of an external MCU or MPU.

Note

List of devices that support this command - ATECC608A/B. Refer to device datasheet for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.103 calib_selftest.c File Reference

CryptoAuthLib Basic API methods for SelfTest command.

```
#include "cryptoauthlib.h"
```

24.103.1 Detailed Description

CryptoAuthLib Basic API methods for SelfTest command.

The SelfTest command performs a test of one or more of the cryptographic engines within the device.

Note

List of devices that support this command - ATECC608A/B. Refer to device datasheet for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.104 calib_sha.c File Reference

CryptoAuthLib Basic API methods for SHA command.

```
#include "cryptoauthlib.h"
```

24.104.1 Detailed Description

CryptoAuthLib Basic API methods for SHA command.

The SHA command Computes a SHA-256 or HMAC/SHA digest for general purpose use by the host system.

Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.105 calib_sign.c File Reference

CryptoAuthLib Basic API methods for Sign command.

```
#include "cryptoauthlib.h"
```

24.105.1 Detailed Description

CryptoAuthLib Basic API methods for Sign command.

The Sign command generates a signature using the private key in slot with ECDSA algorithm.

Note

List of devices that support this command - ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.106 calib_updateextra.c File Reference

CryptoAuthLib Basic API methods for UpdateExtra command.

```
#include "cryptoauthlib.h"
```

24.106.1 Detailed Description

CryptoAuthLib Basic API methods for UpdateExtra command.

The UpdateExtra command is used to update the values of the two extra bytes within the Configuration zone after the Configuration zone has been locked.

Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.107 calib_verify.c File Reference

CryptoAuthLib Basic API methods for Verify command.

```
#include "cryptoauthlib.h"  
#include "host/atca_host.h"
```

24.107.1 Detailed Description

CryptoAuthLib Basic API methods for Verify command.

The Verify command takes an ECDSA [R,S] signature and verifies that it is correctly generated given an input message digest and public key.

Note

List of devices that support this command - ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheet for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.108 calib_write.c File Reference

CryptoAuthLib Basic API methods for Write command.

```
#include "cryptoauthlib.h"
#include "host/atca_host.h"
```

24.108.1 Detailed Description

CryptoAuthLib Basic API methods for Write command.

The Write command writes either one 4-byte word or a 32-byte block to one of the EEPROM zones on the device. Depending upon the value of the WriteConfig byte for a slot, the data may be required to be encrypted by the system prior to being sent to the device

Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.109 atca_crypto_hw_aes.h File Reference

AES CTR, CBC & CMAC structure definitions.

```
#include "cryptoauthlib.h"
#include "crypto_hw_config_check.h"
```

24.109.1 Detailed Description

AES CTR, CBC & CMAC structure definitions.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.110 atca_crypto_hw_aes_cbc.c File Reference

CryptoAuthLib Basic API methods for AES CBC mode.

```
#include "cryptoauthlib.h"
#include "atca_crypto_hw_aes.h"
```

24.110.1 Detailed Description

CryptoAuthLib Basic API methods for AES CBC mode.

The AES command supports 128-bit AES encryption or decryption of small messages or data packets in ECB mode.

Note

List of devices that support this command - ATECC608A, ATECC608B, & TA10x. Refer to device datasheet for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.111 atca_crypto_hw_aes_cbcmac.c File Reference

CryptoAuthLib Basic API methods for AES CBC_MAC mode.

```
#include "cryptoauthlib.h"
#include "crypto_hw_config_check.h"
```

24.111.1 Detailed Description

CryptoAuthLib Basic API methods for AES CBC_MAC mode.

The AES command supports 128-bit AES encryption or decryption of small messages or data packets in ECB mode. Also can perform GFM (Galois Field Multiply) calculation in support of AES-GCM.

Note

List of devices that support this command - ATECC608A. Refer to device datasheet for full details.

Copyright

(c) 2015-2018 Microchip Technology Inc. and its subsidiaries.

24.112 atca_crypto_hw_aes_ccm.c File Reference

CryptoAuthLib Basic API methods for AES CCM mode.

```
#include "cryptoauthlib.h"
```

24.112.1 Detailed Description

CryptoAuthLib Basic API methods for AES CCM mode.

The AES command supports 128-bit AES encryption or decryption of small messages or data packets in ECB mode. CCM mode provides security and authenticity to the message being processed.

Note

List of devices that support this command - ATECC608A. Refer to device datasheet for full details.

Copyright

(c) 2015-2018 Microchip Technology Inc. and its subsidiaries.

24.113 atca_crypto_hw_aes_cmac.c File Reference

CryptoAuthLib Basic API methods for AES CBC_MAC mode.

```
#include "cryptoauthlib.h"  
#include "atca_crypto_hw_aes.h"
```

24.113.1 Detailed Description

CryptoAuthLib Basic API methods for AES CBC_MAC mode.

The AES command supports 128-bit AES encryption or decryption of small messages or data packets in ECB mode.

Note

List of devices that support this command - ATECC608A, ATECC608B, & TA10x. Refer to device datasheet for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.114 atca_crypto_hw_aes_ctr.c File Reference

CryptoAuthLib Basic API methods for AES CTR mode.

```
#include "cryptoauthlib.h"  
#include "atca_crypto_hw_aes.h"
```

24.114.1 Detailed Description

CryptoAuthLib Basic API methods for AES CTR mode.

The AES command supports 128-bit AES encryption or decryption of small messages or data packets in ECB mode.

Note

List of devices that support this command - ATECC608A, ATECC608B, & TA100. Refer to device datasheet for full details.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.115 atca_crypto_pad.c File Reference

Implementation of PKCS7 Padding for block encryption.

```
#include "cryptoauthlib.h"  
#include "atca_crypto_sw.h"
```

24.115.1 Detailed Description

Implementation of PKCS7 Padding for block encryption.

Copyright

(c) 2022 Microchip Technology Inc. and its subsidiaries.

24.116 atca_crypto_pbkdf2.c File Reference

Implementation of the PBKDF2 algorithm for use in generating password hashes.

```
#include "cryptoauthlib.h"  
#include "cal_internal.h"
```

24.116.1 Detailed Description

Implementation of the PBKDF2 algorithm for use in generating password hashes.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.117 atca_crypto_sw.h File Reference

Common defines for CryptoAuthLib software crypto wrappers.

```
#include <stdint.h>
#include <stdlib.h>
#include "crypto/crypto_sw_config_check.h"
#include "atca_status.h"
```

Macros

- #define **ATCA_SHA1_DIGEST_SIZE** (20U)
- #define **ATCA_SHA2_256_DIGEST_SIZE** (32U)
- #define **ATCA_SHA2_256_BLOCK_SIZE** (64U)
- #define **ATCA_SHA2_384_DIGEST_SIZE** (48U)
- #define **ATCA_SHA2_384_BLOCK_SIZE** (128U)
- #define **ATCA_SHA2_512_DIGEST_SIZE** (64U)
- #define **ATCA_SHA2_512_BLOCK_SIZE** (128U)

24.117.1 Detailed Description

Common defines for CryptoAuthLib software crypto wrappers.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.118 atca_crypto_sw_aes_cmac.c File Reference

Common Wrapper for host side AES-CMAC implementations that feature update APIs rather than an all at once implementation.

```
#include "atca_crypto_sw.h"
```

24.118.1 Detailed Description

Common Wrapper for host side AES-CMAC implementations that feature update APIs rather than an all at once implementation.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.119 atca_crypto_sw_aes_gcm.c File Reference

Common Wrapper for host side AES-GCM implementations that feature update APIs rather than an all at once implementation.

```
#include "atca_crypto_sw.h"
```

24.119.1 Detailed Description

Common Wrapper for host side AES-GCM implementations that feature update APIs rather than an all at once implementation.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.120 atca_crypto_sw_sha1.c File Reference

Wrapper API for SHA 1 routines.

```
#include "atca_crypto_sw_sha1.h"  
#include "hashes/sha1_routines.h"  
#include "cryptoauthlib.h"  
#include "cal_internal.h"
```

24.120.1 Detailed Description

Wrapper API for SHA 1 routines.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.121 atca_crypto_sw_sha1.h File Reference

Wrapper API for SHA 1 routines.

```
#include "atca_crypto_sw.h"
#include <stddef.h>
#include <stdint.h>
```

Functions

- ATCA_STATUS **atcac_sw_sha1** (const uint8_t *data, size_t data_size, uint8_t digest[(20U)])

24.121.1 Detailed Description

Wrapper API for SHA 1 routines.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.122 atca_crypto_sw_sha2.c File Reference

Wrapper API for software SHA 256 routines.

```
#include "cryptoauthlib.h"
#include "atca_crypto_sw_sha2.h"
#include "cal_internal.h"
```

24.122.1 Detailed Description

Wrapper API for software SHA 256 routines.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.123 atca_crypto_sw_sha2.h File Reference

Wrapper API for software SHA 256 routines.

```
#include "atca_crypto_sw.h"
#include <stddef.h>
#include <stdint.h>
```

Functions

- ATCA_STATUS **atcac_sha256_hmac_ctr_iteration** (struct [atcac_hmac_ctx](#) *ctx, uint8_t iteration, uint16_t length, const uint8_t *label, size_t label_len, const uint8_t *data, size_t data_len, uint8_t digest[(32U)])
- ATCA_STATUS **atcac_sha256_hmac_counter** (uint8_t *key, size_t key_len, const uint8_t *label, size_t label_len, const uint8_t *data, size_t data_len, uint8_t *digest, size_t diglen)

24.123.1 Detailed Description

Wrapper API for software SHA 256 routines.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.124 crypto_hw_config_check.h File Reference

Consistency checks for configuration options.

```
#include "atca_config_check.h"
#include "calib/calib_config_check.h"
#include "talib/talib_config_check.h"
```

Macros

- #define [ATCAB_AES_EXTRAS_EN](#) (CALIB_AES_EN || TALIB_AES_EN || LIBRARY_USAGE_EN_CHECK)
- #define **ATCAB_AES_RANDOM_IV_EN** (ATCA_HOSTLIB_EN || CALIB_RANDOM_EN || TALIB_RANDOM_EN)
- #define [ATCAB_AES_UPDATE_EN](#) [ATCAB_AES_EXTRAS_EN](#)
- #define [ATCAB_AES_CBC_ENCRYPT_EN](#) [ATCAB_AES_EXTRAS_EN](#)
- #define [ATCAB_AES_CBC_DECRYPT_EN](#) [ATCAB_AES_EXTRAS_EN](#)
- #define **ATCAB_AES_CBC_UPDATE_EN** [ATCAB_AES_UPDATE_EN](#)
- #define [ATCAB_AES_CBCMAC_EN](#) [ATCAB_AES_CBC_ENCRYPT_EN](#)
- #define [ATCAB_AES_CTR_EN](#) [ATCAB_AES_EXTRAS_EN](#)
- #define [ATCAB_AES_CTR_RAND_IV_EN](#) ([ATCAB_AES_CTR_EN](#) && [ATCAB_AES_RANDOM_IV_EN](#))
- #define [ATCAB_AES_CCM_EN](#) ([ATCAB_AES_CBCMAC_EN](#) && [ATCAB_AES_CTR_EN](#))
- #define **ATCAB_AES_CCM_RAND_IV_EN** ([ATCAB_AES_CCM_EN](#) && [ATCAB_AES_RANDOM_IV_EN](#))
- #define **ATCAB_AES_CMAC_EN** [ATCAB_AES_CBC_ENCRYPT_EN](#)
- #define **ATCAC_PKCS7_PAD_EN** [ATCAB_AES_EXTRAS_EN](#)

24.124.1 Detailed Description

Consistency checks for configuration options.

Copyright

(c) 2015-2021 Microchip Technology Inc. and its subsidiaries.

24.124.2 Macro Definition Documentation

24.124.2.1 ATCAB_AES_CBC_DECRYPT_EN

```
#define ATCAB_AES_CBC_DECRYPT_EN ATCAB_AES_EXTRAS_EN
```

Requires: ATCAB_AES_EN

Enable ATCAB_AES_CBC_DECRYPT to decrypt a block of data using CBC mode and a key within the device. atcab_aes_cbc_init() should be called before the first use of this function

Supported API's: atcab_aes_cbc_decrypt_block, atcab_aes_cbc_init_ext, atcab_aes_cbc_init

24.124.2.2 ATCAB_AES_CBC_ENCRYPT_EN

```
#define ATCAB_AES_CBC_ENCRYPT_EN ATCAB_AES_EXTRAS_EN
```

Requires: ATCAB_AES_EN

Enable ATCAB_AES_CBC_ENCRYPT_EN to encrypt a block of data using CBC mode and a key within the device. atcab_aes_cbc_init() should be called before the first use of this function

Supported API's: atcab_aes_cbc_encrypt_block , atcab_aes_cbc_init_ext, atcab_aes_cbc_init

24.124.2.3 ATCAB_AES_CBCMAC_EN

```
#define ATCAB_AES_CBCMAC_EN ATCAB_AES_CBC_ENCRYPT_EN
```

Requires: ATCAB_AES_CBCMAC ATCAB_AES_CBC_ENCRYPT ATCAB_AES_MODE_ENCODING CALIB_↔ AES_MODE_ENCODING CALIB_AES

Enable ATCAB_AES_CBCMAC to initialize context for AES CBC-MAC operation Enable ATCAB_AES_CBCMAC to calculate AES CBC-MAC with key stored within ECC608 device Enable ATCAB_AES_CBCMAC to finish a CBC-↔ MAC operation returning the CBC-MAC value

Supported API's: atcab_aes_cbcmac_init_ext atcab_aes_cbcmac_init, atcab_aes_cbcmac_init_update, atcab_↔ aes_cbcmac_finish

24.124.2.4 ATCAB_AES_CCM_EN

```
#define ATCAB_AES_CCM_EN (ATCAB_AES_CBCMAC_EN && ATCAB_AES_CTR_EN)
```

Requires: ATCAB_AES_EN ATCAB_AES_CTR_EN

Enable ATCAB_AES_CCM_EN to enable AES CCM operation

24.124.2.5 ATCAB_AES_CTR_EN

```
#define ATCAB_AES_CTR_EN ATCAB_AES_EXTRAS_EN
```

Requires: ATCAB_AES_EN

Enable ATCAB_AES_CTR_EN to support AES-CTR mode

24.124.2.6 ATCAB_AES_CTR_RANDOM_IV_EN

```
#define ATCAB_AES_CTR_RANDOM_IV_EN (ATCAB_AES_CTR_EN && ATCAB_AES_RANDOM_IV_EN)
```

Requires: ATCAB_AES_CTR_EN ATCAB_RANDOM_EN

Enable ATCAB_AES_CTR_RANDOM_IV_EN to initialize context for AES CTR operation with a random nonce and counter set to 0 as the IV, which is common when starting an encrypt operation

Supported API's: atcab_aes_ctr_init_rand_ext, atcab_aes_ctr_init_rand

24.124.2.7 ATCAB_AES_EXTRAS_EN

```
#define ATCAB_AES_EXTRAS_EN (CALIB_AES_EN || TALIB_AES_EN || LIBRARY_USAGE_EN_CHECK)
```

Automatically set base on other configuration options but can be overridden to disable all CBC, CBCMAC, CTR, & CCM modes at once rather than individually

24.124.2.8 ATCAB_AES_UPDATE_EN

```
#define ATCAB_AES_UPDATE_EN ATCAB_AES_EXTRAS_EN
```

Enable update/finalize APIs for block ciphers

24.125 crypto_sw_config_check.h File Reference

Consistency checks for configuration options.

```
#include "atca_config_check.h"
```

Macros

- #define [ATCAC_SHA1_EN](#) (DEFAULT_ENABLED)
- #define [ATCAC_SHA256_EN](#) (FEATURE_ENABLED)
- #define [ATCAC_SHA384_EN](#) (FEATURE_DISABLED)
- #define [ATCAC_SHA512_EN](#) (FEATURE_DISABLED)
- #define [ATCAC_SHA256_HMAC_EN](#) [ATCAC_SHA256_EN](#)
- #define [ATCAC_SHA256_HMAC_CTR_EN](#) [ATCAC_SHA256_HMAC_EN](#)
- #define [ATCAC_PKEY_EN](#) [ATCA_HOSTLIB_EN](#)
- #define [ATCAC_RANDOM_EN](#) [ATCA_HOSTLIB_EN](#)
- #define [ATCAC_VERIFY_EN](#) [ATCA_HOSTLIB_EN](#)
- #define [ATCAC_SIGN_EN](#) [ATCA_HOSTLIB_EN](#)
- #define [ATCAC_CERT_ADD_EN](#) [ATCA_HOSTLIB_EN](#)
- #define [ATCA_CRYPT_SHA1_EN](#) ([ATCAC_SHA1_EN](#) && ![ATCA_HOSTLIB_EN](#))
- #define [ATCA_CRYPT_SHA256_EN](#) (([ATCAC_SHA256_EN](#)) && ![ATCA_HOSTLIB_EN](#))
- #define [ATCA_CRYPT_SHA384_EN](#) (([ATCAC_SHA384_EN](#)) && ![ATCA_HOSTLIB_EN](#))
- #define [ATCA_CRYPT_SHA512_EN](#) (([ATCAC_SHA512_EN](#)) && ![ATCA_HOSTLIB_EN](#))
- #define [ATCA_CRYPT_SHA2_EN](#) ([ATCA_CRYPT_SHA256_EN](#) || [ATCA_CRYPT_SHA384_EN](#) || [ATCA_CRYPT_SHA512_EN](#))
- #define [ATCA_CRYPT_SHA2_HMAC_EN](#) ([ATCAC_SHA256_HMAC_EN](#) && ![ATCA_HOSTLIB_EN](#) && ![LIBRARY_BUILD_EN_CHECK](#))
- #define [ATCA_CRYPT_SHA2_HMAC_CTR_EN](#) [ATCAC_SHA256_HMAC_CTR_EN](#)
- #define [ATCAC_PBKDF2_SHA256_EN](#) [ATCAC_SHA256_HMAC_EN](#)
- #define [ATCAB_PBKDF2_SHA256_EN](#) ([CALIB_SHA_HMAC_EN](#) || [TALIB_SHA_HMAC_EN](#))
- #define [ATCAC_AES_GCM_EN](#) ([ATCA_HOSTLIB_EN](#))
- #define [ATCA_CRYPT_AES_GCM_EN](#) (![ATCA_HOSTLIB_EN](#) && ([LIBRARY_BUILD_EN_CHECK](#) || [LIBRARY_USAGE_EN_CHECK](#)))
- #define [ATCAC_AES_CMAC_EN](#) ([ATCA_HOSTLIB_EN](#))
- #define [ATCA_CRYPT_AES_CMAC_EN](#) (![ATCA_HOSTLIB_EN](#) && ([LIBRARY_BUILD_EN_CHECK](#) || [LIBRARY_USAGE_EN_CHECK](#)))
- #define [MAX_HMAC_CTX_SIZE](#) (648)
- #define [MAX_AES_CMAC_CTX_SIZE](#) (600)
- #define [MAX_AES_GCM_CTX_SIZE](#) (540)

24.125.1 Detailed Description

Consistency checks for configuration options.

Copyright

(c) 2015-2021 Microchip Technology Inc. and its subsidiaries.

24.125.2 Macro Definition Documentation

24.125.2.1 ATCA_CRYPT_AES_CMAC_EN

```
#define ATCA_CRYPT_AES_CMAC_EN (!ATCA_HOSTLIB_EN && (LIBRARY\_BUILD\_EN\_CHECK || LIBRARY\_USAGE←  
_EN_CHECK))
```

Enable [ATCA_CRYPT_AES_CMAC_EN](#) to enable AES CMAC host side api

24.125.2.2 ATCA_CRYPT0_AES_GCM_EN

```
#define ATCA_CRYPT0_AES_GCM_EN (!ATCA_HOSTLIB_EN && (LIBRARY_BUILD_EN_CHECK || LIBRARY_USAGE_↵
EN_CHECK))
```

Enable ATCA_CRYPT0_AES_GCM_EN to enable AES GCM host side api

24.125.2.3 ATCA_CRYPT0_SHA1_EN

```
#define ATCA_CRYPT0_SHA1_EN (ATCAC_SHA1_EN && !ATCA_HOSTLIB_EN)
```

Enable ATCAC_SHA1_EN to enable sha1 host side api

Supported API's: atcab_write

24.125.2.4 ATCA_CRYPT0_SHA256_EN

```
#define ATCA_CRYPT0_SHA256_EN ((ATCAC_SHA256_EN) && !ATCA_HOSTLIB_EN)
```

Enable ATCA_CRYPT0_SHA256_EN to enable SHA2 host side api

24.125.2.5 ATCA_CRYPT0_SHA2_EN

```
#define ATCA_CRYPT0_SHA2_EN (ATCA_CRYPT0_SHA256_EN || ATCA_CRYPT0_SHA384_EN || ATCA_CRYPT0_SHA512_EN)
```

Enable ATCAC_SHA2_EN to enable sha2 host side api

24.125.2.6 ATCA_CRYPT0_SHA2_HMAC_CTR_EN

```
#define ATCA_CRYPT0_SHA2_HMAC_CTR_EN ATCAC_SHA256_HMAC_CTR_EN
```

Requires: ATCAC_SHA256_HMAC_EN

Enable ATCAC_SHA256_HMAC_COUNTER to implement SHA256 HMAC-Counter per NIST SP 800-108 used for KDF like operations

Supported API's: atcac_sha256_hmac_counter

24.125.2.7 ATCA_CRYPT0_SHA2_HMAC_EN

```
#define ATCA_CRYPT0_SHA2_HMAC_EN (ATCAC_SHA256_HMAC_EN && !ATCA_HOSTLIB_EN && !LIBRARY_BUILD_EN_CHECK)
```

Requires: ATCAC_SHA256_EN

Enable ATCAC_SHA256_HMAC to initialize context for performing HMAC (sha256) in software

Supported API's: atcac_sha256_hmac_init, atcac_sha256_hmac_update, atcac_sha256_hmac_finish

24.125.2.8 ATCA_CRYPT_SHA384_EN

```
#define ATCA_CRYPT_SHA384_EN ((ATCAC_SHA384_EN) && !ATCA_HOSTLIB_EN)
```

Enable ATCA_CRYPT_SHA384_EN to enable SHA384 host side api

24.125.2.9 ATCA_CRYPT_SHA512_EN

```
#define ATCA_CRYPT_SHA512_EN ((ATCAC_SHA512_EN) && !ATCA_HOSTLIB_EN)
```

Enable ATCA_CRYPT_SHA512_EN to enable SHA2512 host side api

24.125.2.10 ATCAB_PBKDF2_SHA256_EN

```
#define ATCAB_PBKDF2_SHA256_EN (CALIB_SHA_HMAC_EN || TALIB_SHA_HMAC_EN)
```

Requires: CALIB_SHA_HMAC_EN

Enable ATCAB_PBKDF2_SHA256_EN to calculate a PBKDF2 password hash using a stored key inside a device. The key length is determined by the device being used. ECCx08: 32 bytes, TA100: 16-64 bytes

Supported API's: atcab_pbkdf2_256, atcab_pbkdf2_256_ext

24.125.2.11 ATCAC_AES_CMAC_EN

```
#define ATCAC_AES_CMAC_EN (ATCA_HOSTLIB_EN)
```

Indicates if this module is a provider of an AES-CMAC implementation

24.125.2.12 ATCAC_AES_GCM_EN

```
#define ATCAC_AES_GCM_EN (ATCA_HOSTLIB_EN)
```

Indicates if this module is a provider of an AES-GCM implementation

24.125.2.13 ATCAC_CERT_ADD_EN

```
#define ATCAC_CERT_ADD_EN ATCA_HOSTLIB_EN
```

Requires: ATCA_HOSTLIB_EN

Enable ATCAC_CERT_ADD_EN to use the host's cert add functions. Generally assumed to come from the host's cryptographic library or peripheral driver.

24.125.2.14 ATCAC_PBKDF2_SHA256_EN

```
#define ATCAC_PBKDF2_SHA256_EN ATCAC_SHA256_HMAC_EN
```

Requires: ATCAC_SHA256_EN ATCAC_SHA256_HMAC_EN

Enable ATCAC_PBKDF2_SHA256_EN to calculate a PBKDF2 hash of a given password and salt

Supported API's: atcac_pbkdf2_256

24.125.2.15 ATCAC_PKEY_EN

```
#define ATCAC_PKEY_EN ATCA_HOSTLIB_EN
```

Requires: ATCA_HOSTLIB_EN

Enable ATCAC_PKEY_EN to use the host's asymmetric cryptography functions - generally assumed to come from the host's cryptographic library or peripheral driver.

24.125.2.16 ATCAC_RANDOM_EN

```
#define ATCAC_RANDOM_EN ATCA_HOSTLIB_EN
```

Requires: ATCA_HOSTLIB_EN

Enable ATCAC_RANDOM_EN get random numbers from the host's implementation - generally assumed to come from the host's cryptographic library or peripheral driver

24.125.2.17 ATCAC_SHA1_EN

```
#define ATCAC_SHA1_EN (DEFAULT_ENABLED)
```

Enable ATCAC_SHA1_EN to enable sha1 host side api

Supported API's: atcab_write

24.125.2.18 ATCAC_SHA256_EN

```
#define ATCAC_SHA256_EN (FEATURE_ENABLED)
```

Enable ATCAC_SHA256_EN to enable sha256 host side api

24.125.2.19 ATCAC_SHA384_EN

```
#define ATCAC_SHA384_EN (FEATURE_DISABLED)
```

Enable ATCAC_SHA384_EN to enable sha384 host side api

Disabled by default. Enable ATCAC_SHA512_EN to use SHA384

24.125.2.20 ATCAC_SHA512_EN

```
#define ATCAC_SHA512_EN (FEATURE_DISABLED)
```

Enable ATCAC_SHA512_EN to enable sha512 host side api

Disabled by default. Use FEATURE_ENABLED to enable this feature

24.125.2.21 ATCAC_SIGN_EN

```
#define ATCAC_SIGN_EN ATCA_HOSTLIB_EN
```

Requires: ATCA_HOSTLIB_EN

Enable ATCAC_SIGN_EN to use the host's sign functions. Generally assumed to come from the host's cryptographic library or peripheral driver.

24.125.2.22 ATCAC_VERIFY_EN

```
#define ATCAC_VERIFY_EN ATCA_HOSTLIB_EN
```

Requires: ATCA_HOSTLIB_EN

Enable ATCAC_VERIFY_EN to use the host's verify functions. Generally assumed to come from the host's cryptographic library or peripheral driver.

24.125.2.23 MAX_AES_CMAC_CTX_SIZE

```
#define MAX_AES_CMAC_CTX_SIZE (600)
```

Set to Maximum AES CMAC context size

24.125.2.24 MAX_AES_GCM_CTX_SIZE

```
#define MAX_AES_GCM_CTX_SIZE (540)
```

Set to Maximum AES GCM context size

24.125.2.25 MAX_HMAC_CTX_SIZE

```
#define MAX_HMAC_CTX_SIZE (648)
```

Set to Maximum HMAC context size

24.126 sha1_routines.c File Reference

Software implementation of the SHA1 algorithm.

```
#include "sha1_routines.h"
#include <string.h>
#include "atca_compiler.h"
#include "cryptoauthlib.h"
```

24.126.1 Detailed Description

Software implementation of the SHA1 algorithm.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.127 sha1_routines.h File Reference

Software implementation of the SHA1 algorithm.

```
#include "atca_compiler.h"
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <stdint.h>
```

Data Structures

- struct [CL_HashContext](#)

Macros

- #define **U8** uint8_t
- #define **U16** uint16_t
- #define **U32** uint32_t
- #define **U64** uint64_t
- #define **memcpy_P** memmove
- #define **strcpy_P** strcpy
- #define **_WDRESET()**
- #define **_NOP()**
- #define **leftRotate**(x, n) (x) = (((x) << (n)) | ((x) >> (32 - (n))))

Functions

- void **shaEngine** (uint32_t *buf, uint32_t *h)
- void **CL_hashInit** (CL_HashContext *ctx)
- void **CL_hashUpdate** (CL_HashContext *ctx, const uint8_t *src, int nbytes)
- void **CL_hashFinal** (CL_HashContext *ctx, uint8_t *dest)
- void **CL_hash** (uint8_t *msg, int msgBytes, uint8_t *dest)

24.127.1 Detailed Description

Software implementation of the SHA1 algorithm.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.128 sha2_routines.c File Reference

Software implementation of the SHA256, SHA384 and SHA512 algorithm.

```
#include "cryptoauthlib.h"
#include "sha2_routines.h"
```

Macros

- #define **rotate_right**(value, places) (((value) >> (places)) | ((value) << (32U - (places))))
- #define **rotate_right_64bit**(value, places) (((value) >> (places)) | ((value) << (64U - (places))))

24.128.1 Detailed Description

Software implementation of the SHA256, SHA384 and SHA512 algorithm.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.129 sha2_routines.h File Reference

Software implementation of the SHA256, SHA384 and SHA512 algorithm.

```
#include <stdint.h>
```

Macros

- #define **SHA256_DIGEST_SIZE** (32U)
- #define **SHA512_DIGEST_SIZE** (64U)
- #define **SHA384_DIGEST_SIZE** (48U)
- #define **SHA256_BLOCK_SIZE** (64U)
- #define **SHA384_BLOCK_SIZE** (128U)
- #define **SHA512_BLOCK_SIZE** (128U)

24.129.1 Detailed Description

Software implementation of the SHA256, SHA384 and SHA512 algorithm.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.130 cryptoauthlib.h File Reference

Single aggregation point for all CryptoAuthLib header files.

```
#include <stdio.h>
#include <stdint.h>
#include <stddef.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include <stdbool.h>
#include "atca_config_check.h"
#include "atca_compiler.h"
#include "atca_version.h"
#include "atca_platform.h"
#include "atca_status.h"
#include "atca_debug.h"
#include "cal_buffer.h"
#include "atca_iface.h"
#include "atca_device.h"
#include "atca_helpers.h"
#include "hal/atca_hal.h"
#include "atca_cfgs.h"
#include "calib/calib_basic.h"
#include "calib/calib_command.h"
#include "calib/calib_aes_gcm.h"
#include "calib/calib_packet.h"
#include "talib/talib_status.h"
#include "talib/talib_basic.h"
#include "atca_basic.h"
```

Macros

- #define [ATCA_SHA256_BLOCK_SIZE](#) (64u)
- #define [ATCA_SHA256_DIGEST_SIZE](#) (32u)
- #define [ATCA_SHA384_BLOCK_SIZE](#) (128u)
- #define [ATCA_SHA384_DIGEST_SIZE](#) (48u)
- #define [ATCA_SHA512_BLOCK_SIZE](#) (128u)
- #define [ATCA_SHA512_DIGEST_SIZE](#) (64u)
- #define [ATCA_SHA224_DIGEST_SIZE](#) (28u)
- #define [ATCA_AES128_BLOCK_SIZE](#) (16u)
- #define [ATCA_AES128_KEY_SIZE](#) (16)
- #define [ATCA_AES256_BLOCK_SIZE](#) (16u)
- #define [ATCA_AES256_KEY_SIZE](#) (32u)
- #define [ATCA_ECCP256_MSG_SIZE](#) (32u)
- #define [ATCA_KEY_TYPE_ECCP256](#) (0u)
- #define [ATCA_ECCP256_KEY_SIZE](#) (32u)
- #define [ATCA_ECCP224_PUBKEY_SIZE](#) (56u)
- #define [ATCA_ECCP256_PUBKEY_SIZE](#) (64u)
- #define [ATCA_ECCP384_PUBKEY_SIZE](#) (96u)
- #define [ATCA_ECCP521_PUBKEY_SIZE](#) (132u)
- #define [ATCA_ECCP256_PVTKEY_SIZE](#) (32u)
- #define [ATCA_ECCP256_SIG_SIZE](#) (64u)
- #define [ATCA_ECCP384_SIG_SIZE](#) (96u)
- #define [ATCA_ECCP521_SIG_SIZE](#) (132u)
- #define [ATCA_ECCP256_OID_SIZE](#) (10u)
- #define [ATCA_ECCP256_ASN1_HDR_SIZE](#) (27u)
- #define [ATCA_MAX_ECC_RSA_PB_KEY_SIZE](#) (512u)
- #define [ATCA_RSA4K_ASN1_HDR_SIZE](#) (33u)
- #define [ATCA_ECC_SIG_OVERHEAD_SIZE](#) (8u)
- #define [ATCA_MAX_SLOT_NUM](#) (16)
- #define [ATCA_MAX_ECC_PB_KEY_SIZE](#) (132u)
- #define [ATCA_MAX_ECC_SIG_SIZE](#) (132u)
- #define [ATCA_MAX_DATA_SIZE](#) (512u)
- #define [ATCA_KEY_TYPE_ECC_COUNT](#) (4u)
- #define [ATCA_SERNUM_OFFSET](#) (0u)
- #define [ATCACERT_COMP_CERT_MAX_SIZE](#) (140u)
- #define [ATCACERT_MAX_SIG_OVERHEAD](#) (19u)
- #define [ATCACERT_MAX_R_SIG_OFFSET](#) (14u)
- #define [ATCACERT_COMPCERT_OVERHEAD](#) (9u)
- #define [R_S_LEN](#) (ATCA_MAX_ECC_SIG_SIZE / 2u)
- #define [ATCA_ECC_UNCOMPRESSED_TYPE](#) ((uint8_t)0x04)
- #define [ATCA_ECC_UNCOMPRESSED_TYPE_OFFSET](#) (1u)
- #define [ATCA_DEDICATED_DATA_SIZE](#) (16u)
- #define [ATCA_CA_SERNUM_SIZE](#) (9u)
- #define [ATCA_TA_SERNUM_SIZE](#) (8u)
- #define [ATCA_MAX_SERNUM_SIZE](#) (9u)
- #define [ATCA_CA_MAX_SIG_OVERHEAD](#) (11u)
- #define [ATCA_TA_MAX_SIG_OVERHEAD](#) (19u)
- #define [ATCA_ZONE_CONFIG](#) ((uint8_t)0x00)
- #define [ATCA_ZONE_OTP](#) ((uint8_t)0x01)
- #define [ATCA_ZONE_DATA](#) ((uint8_t)0x02)
- #define [DEVICE_PRODUCT_ID_LOCATION](#) 0
- #define [DEVICE_IDENTIFIER_LOCATION](#) 1
- #define [DEVICE_PART_LOCATION](#) 2
- #define [DEVICE_REVISION_LOCATION](#) 3

- `#define ATCA_ZONE_CA2_DATA ((uint8_t)0x00)`
- `#define ATCA_ZONE_CA2_CONFIG ((uint8_t)0x01)`
- `#define ATCA_ECC204_DEVICE_ID ((uint8_t)0x5A)`
- `#define ATCA_TA010_DEVICE_ID ((uint8_t)0x6A)`
- `#define ATCA_SHA104_DEVICE_ID ((uint8_t)0x35)`
- `#define ATCA_SHA105_DEVICE_ID ((uint8_t)0x3B)`
- `#define SHA_MODE_TARGET_TEMPKEY ((uint8_t)0x00)`
- `#define SHA_MODE_TARGET_MSGDIGBUF ((uint8_t)0x40)`
- `#define SHA_MODE_TARGET_OUT_ONLY ((uint8_t)0xC0)`
- `#define ATCA_STRINGIFY(x) #x`
- `#define ATCA_TOSTRING(x) ATCA_STRINGIFY(x)`
- `#define ATCA_TRACE(s, m) atca_trace(s)`

24.130.1 Detailed Description

Single aggregation point for all CryptoAuthLib header files.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.130.2 Macro Definition Documentation

24.130.2.1 ATCA_SHA256_BLOCK_SIZE

```
#define ATCA_SHA256_BLOCK_SIZE (64u)
```

Library Configuration File - All build attributes should be included in `atca_config.h`

24.130.2.2 SHA_MODE_TARGET_MSGDIGBUF

```
#define SHA_MODE_TARGET_MSGDIGBUF ((uint8_t)0x40)
```

Place resulting digest both in Output buffer and Message Digest Buffer

24.130.2.3 SHA_MODE_TARGET_OUT_ONLY

```
#define SHA_MODE_TARGET_OUT_ONLY ((uint8_t)0xC0)
```

Place resulting digest both in Output buffer ONLY

24.130.2.4 SHA_MODE_TARGET_TEMPKEY

```
#define SHA_MODE_TARGET_TEMPKEY ((uint8_t)0x00)
```

Place resulting digest both in Output buffer and TempKey

24.131 atca_hal.c File Reference

low-level HAL - methods used to setup indirection to physical layer interface. this level does the dirty work of abstracting the higher level ATCAIFace methods from the low-level physical interfaces. Its main goal is to keep low-level details from bleeding into the logical interface implementation.

```
#include "cryptoauthlib.h"
#include "atca_hal.h"
```

Data Structures

- struct [atca_hal_list_entry_t](#)
Structure that holds the hal/phy mapping for different interface types.

Functions

- ATCA_STATUS [hal_iface_register_hal](#) (ATCAIFaceType iface_type, ATCAHAL_t *hal, ATCAHAL_t **old_hal, ATCAHAL_t *phy, ATCAHAL_t **old_phy)
Register/Replace a HAL with a.
- ATCA_STATUS [hal_iface_init](#) (ATCAIFaceCfg *cfg, ATCAHAL_t **hal, ATCAHAL_t **phy)
Standard HAL API for ATCA to initialize a physical interface.
- ATCA_STATUS [hal_iface_release](#) (ATCAIFaceType iface_type, void *hal_data)
releases a physical interface, HAL knows how to interpret hal_data
- ATCA_STATUS [hal_check_wake](#) (const uint8_t *response, int response_size)
Utility function for hal_wake to check the reply.
- uint8_t [hal_is_command_word](#) (uint8_t word_address)
Utility function for hal_wake to check the reply.

24.131.1 Detailed Description

low-level HAL - methods used to setup indirection to physical layer interface. this level does the dirty work of abstracting the higher level ATCAIFace methods from the low-level physical interfaces. Its main goal is to keep low-level details from bleeding into the logical interface implementation.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.132 atca_hal.h File Reference

low-level HAL - methods used to setup indirection to physical layer interface

```
#include <stdlib.h>
#include "atca_config.h"
#include "atca_status.h"
#include "atca_iface.h"
```

Data Structures

- struct [atca_hal_kit_phy_t](#)
- struct [atca_hal_shm_t](#)

Macros

- **#define ATCA_POLLING_INIT_TIME_MSEC 1**
- **#define ATCA_POLLING_FREQUENCY_TIME_MSEC 2**
- **#define ATCA_POLLING_MAX_TIME_MSEC 2500**
- **#define ATCA_HAL_CONTROL_WAKE (0U)**
Execute the hardware specific wake - generally only for kits.
- **#define ATCA_HAL_CONTROL_IDLE (1U)**
Execute the hardware specific idle - generally only for kits.
- **#define ATCA_HAL_CONTROL_SLEEP (2U)**
Execute the hardware specific sleep - generally only for kits.
- **#define ATCA_HAL_CONTROL_RESET (3U)**
Execute the hardware specific reset - generally only for kits.
- **#define ATCA_HAL_CONTROL_SELECT (4U)**
Select the device - assert CS, open device, etc.
- **#define ATCA_HAL_CONTROL_DESELECT (5U)**
Select the device - de-assert CS, release device, etc.
- **#define ATCA_HAL_CHANGE_BAUD (6U)**
Change the datarate of the phy.
- **#define ATCA_HAL_FLUSH_BUFFER (7U)**
If the phy has a buffer make sure all bytes are transmitted.
- **#define ATCA_HAL_CONTROL_DIRECTION (8U)**
Set the PIN mode (in vs out)

Typedefs

- typedef void * **hal_mutex_t**
Generic mutex type definition for most systems.

Functions

- ATCA_STATUS [hal_iface_init](#) ([ATCAIfaceCfg](#) *cfg, [ATCAHAL_t](#) **hal, [ATCAHAL_t](#) **phy)
Standard HAL API for ATCA to initialize a physical interface.
- ATCA_STATUS [hal_iface_release](#) ([ATCAIfaceType](#) iface_type, void *hal_data)
releases a physical interface, HAL knows how to interpret hal_data
- ATCA_STATUS [hal_check_wake](#) (const uint8_t *response, int response_size)
Utility function for hal_wake to check the reply.
- void [atca_delay_ms](#) (uint32_t ms)
Timer API for legacy implementations.
- void [atca_delay_us](#) (uint32_t delay)
This function delays for a number of microseconds.
- void [hal_delay_ms](#) (uint32_t delay)
Timer API implemented at the HAL level.
- void [hal_delay_us](#) (uint32_t delay)

This function delays for a number of microseconds.

- ATCA_STATUS [hal_create_mutex](#) (void **ppMutex, const char *pName)

Optional hal interfaces.

- ATCA_STATUS [hal_init_mutex](#) (void *pMutex, bool shared)
- ATCA_STATUS [hal_destroy_mutex](#) (void *pMutex)
- ATCA_STATUS [hal_lock_mutex](#) (void *pMutex)
- ATCA_STATUS [hal_unlock_mutex](#) (void *pMutex)
- ATCA_STATUS [hal_alloc_shared](#) (void **pShared, size_t size, const char *pName, bool *initialized)
- ATCA_STATUS [hal_free_shared](#) (void *pShared, size_t size)
- ATCA_STATUS [hal_iface_register_hal](#) (ATCAIfaceType iface_type, ATCAHAL_t *hal, ATCAHAL_t **old_hal, ATCAHAL_t *phy, ATCAHAL_t **old_phy)

Register/Replace a HAL with a.

- uint8_t [hal_is_command_word](#) (uint8_t word_address)

Utility function for hal_wake to check the reply.

24.132.1 Detailed Description

low-level HAL - methods used to setup indirection to physical layer interface

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.133 hal_all_platforms_kit_hidapi.c File Reference

HAL for kit protocol over HID for any platform.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "hidapi.h"
#include "atca_hal.h"
#include "hal/kit_protocol.h"
```

Functions

- ATCA_STATUS [hal_kit_hid_init](#) (ATCAIface iface, ATCAIfaceCfg *cfg)
HAL implementation of Kit USB HID init.
- ATCA_STATUS [hal_kit_hid_post_init](#) (ATCAIface iface)
HAL implementation of Kit HID post init.
- ATCA_STATUS [hal_kit_hid_send](#) (ATCAIface iface, uint8_t word_address, uint8_t *txdata, int txlength)
HAL implementation of kit protocol send over USB HID.
- ATCA_STATUS [hal_kit_hid_receive](#) (ATCAIface iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxlength)
HAL implementation of send over USB HID.
- ATCA_STATUS [hal_kit_hid_control](#) (ATCAIface iface, uint8_t option, void *param, size_t paramlen)
Perform control operations for the kit protocol.
- ATCA_STATUS [hal_kit_hid_release](#) (void *hal_data)
Close the physical port for HID.

24.133.1 Detailed Description

HAL for kit protocol over HID for any platform.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.134 hal_freertos.c File Reference

FreeRTOS Hardware/OS Abstraction Layer.

```
#include "atca_hal.h"
#include "FreeRTOS.h"
#include "semphr.h"
#include "task.h"
```

Macros

- #define **ATCA_MUTEX_TIMEOUT** portMAX_DELAY

Functions

- void * **hal_malloc** (size_t size)
- void **hal_free** (void *ptr)
- void **hal_rtos_delay_ms** (uint32_t delay)
This function delays for a number of milliseconds.
- ATCA_STATUS **hal_create_mutex** (void **ppMutex, const char *pName)
Optional hal interfaces.
- ATCA_STATUS **hal_destroy_mutex** (void *pMutex)
- ATCA_STATUS **hal_lock_mutex** (void *pMutex)
- ATCA_STATUS **hal_unlock_mutex** (void *pMutex)

24.134.1 Detailed Description

FreeRTOS Hardware/OS Abstraction Layer.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.135 hal_gpio_harmony.c File Reference

ATCA Hardware abstraction layer for GPIO.

```
#include "atca_hal.h"
```

Functions

- ATCA_STATUS [hal_gpio_init](#) ([ATCAIface](#) iface, [ATCAIfaceCfg](#) *cfg)
Initialize a gpio interface using given config.
- ATCA_STATUS [hal_gpio_post_init](#) ([ATCAIface](#) iface)
Post Init for gpio hal.
- ATCA_STATUS [hal_gpio_send](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *pin_state, int unused_↵ param)
Set the state of the pin.
- ATCA_STATUS [hal_gpio_receive](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *pin_state, uint16_↵ t *unused_param)
Read the state of the pin.
- ATCA_STATUS [hal_gpio_control](#) ([ATCAIface](#) iface, uint8_t option, void *param, size_t paramlen)
- ATCA_STATUS [hal_gpio_release](#) (void *hal_data)
Release and clean up the HAL.

24.135.1 Detailed Description

ATCA Hardware abstraction layer for GPIO.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.135.2 Function Documentation

24.135.2.1 hal_gpio_init()

```
ATCA_STATUS hal_gpio_init (  
    ATCAIface iface,  
    ATCAIfaceCfg * cfg )
```

Initialize a gpio interface using given config.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.135.2.2 hal_gpio_post_init()

```
ATCA_STATUS hal_gpio_post_init (  
    ATCAIface iface )
```

Post Init for gpio hal.

Returns

ATCA_SUCCESS

24.135.2.3 hal_gpio_receive()

```
ATCA_STATUS hal_gpio_receive (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * pin_state,
    uint16_t * unused_param )
```

Read the state of the pin.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

<i>iface</i>	Interface context
<i>word_address</i>	Unused parameter
<i>pin_state</i>	Pin state to output
<i>unused_param</i>	Unused parameter

24.135.2.4 hal_gpio_release()

```
ATCA_STATUS hal_gpio_release (
    void * hal_data )
```

Release and clean up the HAL.

Parameters

<i>in</i>	<i>hal_data</i>	opaque pointer to hal data structure - known only to the HAL implementation
-----------	-----------------	---

Returns

ATCA_SUCCESS

24.135.2.5 hal_gpio_send()

```
ATCA_STATUS hal_gpio_send (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * pin_state,
    int unused_param )
```

Set the state of the pin.

Returns

ATCA_SUCCESS

Parameters

<i>iface</i>	Interface context
<i>word_address</i>	Unused parameter
<i>pin_state</i>	Pin state to output
<i>unused_param</i>	Unused parameter

24.136 hal_i2c_harmony.c File Reference

ATCA Hardware abstraction layer for SAMD21 I2C over Harmony PLIB.

```
#include <string.h>
#include <stdio.h>
#include "cryptoauthlib.h"
```

Functions

- ATCA_STATUS [hal_i2c_discover_buses](#) (int i2c_buses[], int max_buses)
discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge
- ATCA_STATUS [hal_i2c_discover_devices](#) (int bus_num, ATCAIfaceCfg cfg[], int *found)
discover any CryptoAuth devices on a given logical bus number
- ATCA_STATUS [hal_i2c_init](#) (ATCAIface iface, ATCAIfaceCfg *cfg)
hal_i2c_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIface instances using the same bus, and you can have multiple ATCAIface instances on multiple i2c buses, so hal_i2c_init manages these things and ATCAIface is abstracted from the physical details.
- ATCA_STATUS [hal_i2c_post_init](#) (ATCAIface iface)
HAL implementation of I2C post init.
- ATCA_STATUS [hal_i2c_send](#) (ATCAIface iface, uint8_t word_address, uint8_t *txdata, int txlength)
HAL implementation of I2C send over START.
- ATCA_STATUS [hal_i2c_receive](#) (ATCAIface iface, uint8_t address, uint8_t *rxdata, uint16_t *rxlength)
HAL implementation of I2C receive function for START I2C.
- ATCA_STATUS [change_i2c_speed](#) (ATCAIface iface, uint32_t speed)
method to change the bus speec of I2C
- ATCA_STATUS [hal_i2c_control](#) (ATCAIface iface, uint8_t option, void *param, size_t paramlen)
Perform control operations for the kit protocol.
- ATCA_STATUS [hal_i2c_release](#) (void *hal_data)
manages reference count on given bus and releases resource if no more refences exist

24.136.1 Detailed Description

ATCA Hardware abstraction layer for SAMD21 I2C over Harmony PLIB.

This code is structured in two parts. Part 1 is the connection of the ATCA HAL API to the physical I2C implementation. Part 2 is the Harmony I2C primitives to set up the interface.

Prerequisite: add SERCOM I2C Master Polled support to application in Atmel Studio

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.137 hal_i2c_start.c File Reference

ATCA Hardware abstraction layer for SAMD21 I2C over START drivers.

```
#include <string.h>
#include <stdio.h>
#include <atmel_start.h>
#include <hal_gpio.h>
#include <hal_delay.h>
#include "hal_i2c_start.h"
#include "atca_start_config.h"
#include "atca_start_iface.h"
#include "cryptoauthlib.h"
```

Functions

- ATCA_STATUS [hal_i2c_discover_buses](#) (int i2c_buses[], int max_buses)
discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge
- ATCA_STATUS [hal_i2c_discover_devices](#) (int bus_num, [ATCAIfaceCfg](#) cfg[], int *found)
discover any CryptoAuth devices on a given logical bus number
- ATCA_STATUS [hal_i2c_init](#) (void *hal, [ATCAIfaceCfg](#) *cfg)
hal_i2c_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIface instances using the same bus, and you can have multiple ATCAIface instances on multiple i2c buses, so hal_i2c_init manages these things and ATCAIface is abstracted from the physical details.
- ATCA_STATUS [hal_i2c_post_init](#) ([ATCAIface](#) iface)
HAL implementation of I2C post init.
- ATCA_STATUS [hal_i2c_send](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *txdata, int txlength)
HAL implementation of I2C send over START.
- ATCA_STATUS [hal_i2c_receive](#) ([ATCAIface](#) iface, uint8_t address, uint8_t *rxdata, uint16_t *rxlength)
HAL implementation of I2C receive function for START I2C.
- ATCA_STATUS [hal_i2c_wake](#) ([ATCAIface](#) iface)
wake up CryptoAuth device using I2C bus
- ATCA_STATUS [hal_i2c_idle](#) ([ATCAIface](#) iface)
idle CryptoAuth device using I2C bus
- ATCA_STATUS [hal_i2c_sleep](#) ([ATCAIface](#) iface)
sleep CryptoAuth device using I2C bus
- ATCA_STATUS [hal_i2c_release](#) (void *hal_data)
manages reference count on given bus and releases resource if no more references exist

24.137.1 Detailed Description

ATCA Hardware abstraction layer for SAMD21 I2C over START drivers.

This code is structured in two parts. Part 1 is the connection of the ATCA HAL API to the physical I2C implementation. Part 2 is the START I2C primitives to set up the interface.

Prerequisite: add SERCOM I2C Master Polled support to application in Atmel Studio

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.138 hal_i2c_start.h File Reference

ATCA Hardware abstraction layer for SAMD21 I2C over START drivers.

```
#include "atmel_start.h"
#include <stdlib.h>
#include "cryptoauthlib.h"
```

Data Structures

- struct [i2c_start_instance](#)

Typedefs

- typedef void(* [start_change_baudrate](#)) ([ATCAIface](#) iface, uint32_t speed)
- typedef struct [i2c_start_instance](#) [i2c_start_instance_t](#)

24.138.1 Detailed Description

ATCA Hardware abstraction layer for SAMD21 I2C over START drivers.

Prerequisite: add SERCOM I2C Master Polled support to application in Atmel Studio

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.139 hal_kit_bridge.c File Reference

Kit Bridging HAL for cryptoauthlib. This is not intended to be a zero copy driver. It should work with any interface that confirms to a few basic requirements: a) will accept an arbitrary number of bytes and packetize it if necessary for transmission, b) will block for the duration of the transmit.

```
#include "cryptoauthlib.h"
#include "atca_hal.h"
#include "hal_kit_bridge.h"
```

Functions

- ATCA_STATUS [hal_kit_attach_phy](#) ([ATCAIfaceCfg](#) *cfg, [atca_hal_kit_phy_t](#) *phy)
Helper function that connects a physical layer context structure that will be used by the kit protocol bridge.
- ATCA_STATUS [hal_kit_init](#) ([ATCAIface](#) iface, [ATCAIfaceCfg](#) *cfg)
HAL implementation of Kit USB HID init.
- ATCA_STATUS [hal_kit_post_init](#) ([ATCAIface](#) iface)
HAL implementation of Kit HID post init.
- ATCA_STATUS [hal_kit_send](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *txdata, int txlength)
HAL implementation of kit protocol send over USB HID.
- ATCA_STATUS [hal_kit_receive](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxsize)
HAL implementation of send over USB HID.
- ATCA_STATUS [hal_kit_control](#) ([ATCAIface](#) iface, uint8_t option, void *param, size_t paramlen)
Kit Protocol Control.
- ATCA_STATUS [hal_kit_release](#) (void *hal_data)
Close the physical port for HID.

24.139.1 Detailed Description

Kit Bridging HAL for cryptoauthlib. This is not intended to be a zero copy driver. It should work with any interface that confirms to a few basic requirements: a) will accept an arbitrary number of bytes and packetize it if necessary for transmission, b) will block for the duration of the transmit.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.140 hal_kit_bridge.h File Reference

Kit Bridging HAL for cryptoauthlib. This is not intended to be a zero copy driver. It should work with any interface that confirms to a few basic requirements: a) will accept an arbitrary number of bytes and packetize it if necessary for transmission, b) will block for the duration of the transmit.

Macros

- #define **BRIDGE_PROTOCOL_VERSION** (2)
- #define **HAL_KIT_COMMAND_SEND** 0x01
- #define **HAL_KIT_COMMAND_RECV** 0x02
- #define **HAL_KIT_COMMAND_WAKE** 0x03
- #define **HAL_KIT_COMMAND_IDLE** 0x04
- #define **HAL_KIT_COMMAND_SLEEP** 0x05
- #define **HAL_KIT_HEADER_LEN** (3)

Functions

- ATCA_STATUS [hal_kit_attach_phy](#) ([ATCAIfaceCfg](#) *cfg, [atca_hal_kit_phy_t](#) *phy)

Helper function that connects a physical layer context structure that will be used by the kit protocol bridge.

24.140.1 Detailed Description

Kit Bridging HAL for cryptoauthlib. This is not intended to be a zero copy driver. It should work with any interface that confirms to a few basic requirements: a) will accept an arbitrary number of bytes and packetize it if necessary for transmission, b) will block for the duration of the transmit.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.141 hal_linux.c File Reference

Timer Utility Functions for Linux.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>
#include "atca_hal.h"
```

Functions

- void [hal_delay_us](#) (uint32_t delay)
This function delays for a number of microseconds.
- void [hal_delay_ms](#) (uint32_t delay)
Timer API implemented at the HAL level.
- ATCA_STATUS [hal_create_mutex](#) (void **ppMutex, const char *pName)
Optional hal interfaces.
- ATCA_STATUS [hal_destroy_mutex](#) (void *pMutex)
- ATCA_STATUS [hal_lock_mutex](#) (void *pMutex)
- ATCA_STATUS [hal_unlock_mutex](#) (void *pMutex)
- ATCA_STATUS [hal_check_pid](#) (hal_pid_t pid)
Check if the pid exists in the system.

24.141.1 Detailed Description

Timer Utility Functions for Linux.

Copyright

(c) 2015-2018 Microchip Technology Inc. and its subsidiaries.

24.142 hal_linux_i2c_userspace.c File Reference

ATCA Hardware abstraction layer for Linux using I2C.

```
#include <cryptoauthlib.h>
#include <linux/i2c-dev.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include "atca_hal.h"
```

Data Structures

- struct [atca_i2c_host_s](#)

Typedefs

- typedef struct [atca_i2c_host_s](#) [atca_i2c_host_t](#)

Functions

- ATCA_STATUS [hal_i2c_init](#) ([ATCAIface](#) iface, [ATCAIfaceCfg](#) *cfg)
hal_i2c_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIface instances using the same bus, and you can have multiple ATCAIface instances on multiple i2c buses, so hal_i2c_init manages these things and ATCAIface is abstracted from the physical details.
- ATCA_STATUS [hal_i2c_post_init](#) ([ATCAIface](#) iface)
HAL implementation of I2C post init.
- ATCA_STATUS [hal_i2c_send](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *txdata, int txlength)
HAL implementation of I2C send over START.
- ATCA_STATUS [hal_i2c_receive](#) ([ATCAIface](#) iface, uint8_t address, uint8_t *rxdata, uint16_t *rxlength)
HAL implementation of I2C receive function for START I2C.
- ATCA_STATUS [hal_i2c_control](#) ([ATCAIface](#) iface, uint8_t option, void *param, size_t paramlen)
Perform control operations for the kit protocol.
- ATCA_STATUS [hal_i2c_release](#) (void *hal_data)
manages reference count on given bus and releases resource if no more references exist

24.142.1 Detailed Description

ATCA Hardware abstraction layer for Linux using I2C.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.143 hal_linux_uart_userspace.c File Reference

ATCA Hardware abstraction layer for Linux using UART.

```
#include "cryptoauthlib.h"
#include "atca_hal.h"
#include <unistd.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <termios.h>
```

Data Structures

- struct [atca_uart_host_s](#)

Typedefs

- typedef struct [atca_uart_host_s](#) [atca_uart_host_t](#)

Functions

- ATCA_STATUS [hal_uart_init](#) ([ATCAIface](#) iface, [ATCAIfaceCfg](#) *cfg)
HAL implementation of UART init.
- ATCA_STATUS [hal_uart_post_init](#) ([ATCAIface](#) iface)
HAL implementation of UART post init.
- ATCA_STATUS [hal_uart_send](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *txdata, int txlength)
HAL implementation of UART send.
- ATCA_STATUS [hal_uart_receive](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxlength)
HAL implementation of UART receive function.
- ATCA_STATUS [hal_uart_control](#) ([ATCAIface](#) iface, uint8_t option, void *param, size_t paramlen)
Perform control operations for the UART.
- ATCA_STATUS [hal_uart_release](#) (void *hal_data)
manages reference count on given bus and releases resource if no more references exist

24.143.1 Detailed Description

ATCA Hardware abstraction layer for Linux using UART.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.143.2 Function Documentation

24.143.2.1 [hal_uart_control\(\)](#)

```
ATCA_STATUS hal_uart_control (  
    ATCAIface iface,  
    uint8_t option,  
    void * param,  
    size_t paramlen )
```

Perform control operations for the UART.

Parameters

in	<i>iface</i>	Interface to interact with.
in	<i>option</i>	Control parameter identifier
in	<i>param</i>	Optional pointer to parameter value
in	<i>paramlen</i>	Length of the parameter

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.143.2.2 hal_uart_init()

```
ATCA_STATUS hal_uart_init (
    ATCAIface iface,
    ATCAIfaceCfg * cfg )
```

HAL implementation of UART init.

this implementation assumes UART SERIAL PORT peripheral has been enabled by user . It only initialize an UART interface using given config.

Parameters

in	<i>hal</i>	pointer to HAL specific data that is maintained by this HAL
in	<i>cfg</i>	pointer to HAL specific configuration data that is used to initialize this HAL

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.143.2.3 hal_uart_post_init()

```
ATCA_STATUS hal_uart_post_init (
    ATCAIface iface )
```

HAL implementation of UART post init.

Parameters

in	<i>iface</i>	instance
----	--------------	----------

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.143.2.4 hal_uart_receive()

```
ATCA_STATUS hal_uart_receive (
    ATCAIface iface,
```

```
uint8_t word_address,  
uint8_t * rxdata,  
uint16_t * rxlength )
```

HAL implementation of UART receive function.

Parameters

in	<i>iface</i>	Device to interact with.
in	<i>word_address</i>	device transaction type
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.143.2.5 hal_uart_release()

```
ATCA_STATUS hal_uart_release (  
    void * hal_data )
```

manages reference count on given bus and releases resource if no more refences exist

Parameters

in	<i>hal_data</i>	- opaque pointer to hal data structure - known only to the HAL implementation
----	-----------------	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.143.2.6 hal_uart_send()

```
ATCA_STATUS hal_uart_send (  
    ATCAIface iface,  
    uint8_t word_address,  
    uint8_t * txdata,  
    int txlength )
```

HAL implementation of UART send.

Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	transaction type
in	<i>txdata</i>	data to be send to device
in	<i>txlength</i>	pointer to space to bytes to send

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.144 hal_sam0_i2c_asf.c File Reference

ATCA Hardware abstraction layer for SAMD21 I2C over ASF drivers.

```
#include <asf.h>
#include <string.h>
#include <stdio.h>
#include "hal_sam0_i2c_asf.h"
#include "cryptoauthlib.h"
```

Functions

- ATCA_STATUS [hal_i2c_discover_buses](#) (int i2c_buses[], int max_buses)
discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge
- ATCA_STATUS [hal_i2c_discover_devices](#) (int bus_num, [ATCAIfaceCfg](#) cfg[], int *found)
discover any CryptoAuth devices on a given logical bus number
- ATCA_STATUS [hal_i2c_init](#) (void *hal, [ATCAIfaceCfg](#) *cfg)
hal_i2c_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIface instances using the same bus, and you can have multiple ATCAIface instances on multiple i2c buses, so hal_i2c_init manages these things and ATCAIface is abstracted from the physical details.
- ATCA_STATUS [hal_i2c_post_init](#) ([ATCAIface](#) iface)
HAL implementation of I2C post init.
- ATCA_STATUS [hal_i2c_send](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *txdata, int txlength)
HAL implementation of I2C send over START.
- ATCA_STATUS [hal_i2c_receive](#) ([ATCAIface](#) iface, uint8_t address, uint8_t *rxdata, uint16_t *rxlength)
HAL implementation of I2C receive function for START I2C.
- ATCA_STATUS [hal_i2c_wake](#) ([ATCAIface](#) iface)
wake up CryptoAuth device using I2C bus
- ATCA_STATUS [hal_i2c_idle](#) ([ATCAIface](#) iface)
idle CryptoAuth device using I2C bus
- ATCA_STATUS [hal_i2c_sleep](#) ([ATCAIface](#) iface)
sleep CryptoAuth device using I2C bus
- ATCA_STATUS [hal_i2c_release](#) (void *hal_data)
manages reference count on given bus and releases resource if no more references exist

24.144.1 Detailed Description

ATCA Hardware abstraction layer for SAMD21 I2C over ASF drivers.

This code is structured in two parts. Part 1 is the connection of the ATCA HAL API to the physical I2C implementation. Part 2 is the ASF I2C primitives to set up the interface.

Prerequisite: add SERCOM I2C Master Polled support to application in Atmel Studio

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.145 hal_sam0_i2c_asf.h File Reference

ATCA Hardware abstraction layer for SAMD21 I2C over ASF drivers.

```
#include <asf.h>
#include "cryptoauthlib.h"
```

Data Structures

- struct [i2c_sam0_instance](#)

Typedefs

- typedef void(* [sam0_change_baudrate](#)) ([ATCAIface](#) iface, uint32_t speed)
- typedef struct [i2c_sam0_instance](#) [i2c_sam0_instance_t](#)

24.145.1 Detailed Description

ATCA Hardware abstraction layer for SAMD21 I2C over ASF drivers.

Prerequisite: add SERCOM I2C Master Polled support to application in Atmel Studio

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.146 hal_sam_i2c_asf.c File Reference

ATCA Hardware abstraction layer for SAM flexcom & twi I2C over ASF drivers.

```
#include <asf.h>
#include <string.h>
#include <stdio.h>
#include "cryptoauthlib.h"
#include "hal_sam_i2c_asf.h"
```

Functions

- ATCA_STATUS [hal_i2c_discover_buses](#) (int i2c_buses[], int max_buses)
discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge
- ATCA_STATUS [hal_i2c_discover_devices](#) (int bus_num, [ATCAIfaceCfg](#) cfg[], int *found)
discover any CryptoAuth devices on a given logical bus number
- ATCA_STATUS [hal_i2c_init](#) (void *hal, [ATCAIfaceCfg](#) *cfg)
hal_i2c_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIface instances using the same bus, and you can have multiple ATCAIface instances on multiple i2c buses, so hal_i2c_init manages these things and ATCAIface is abstracted from the physical details.
- ATCA_STATUS [hal_i2c_post_init](#) ([ATCAIface](#) iface)
HAL implementation of I2C post init.
- ATCA_STATUS [hal_i2c_send](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *txdata, int txlength)
HAL implementation of I2C send over START.
- ATCA_STATUS [hal_i2c_receive](#) ([ATCAIface](#) iface, uint8_t address, uint8_t *rxdata, uint16_t *rxlength)
HAL implementation of I2C receive function for START I2C.
- ATCA_STATUS [hal_i2c_wake](#) ([ATCAIface](#) iface)
wake up CryptoAuth device using I2C bus
- ATCA_STATUS [hal_i2c_idle](#) ([ATCAIface](#) iface)
idle CryptoAuth device using I2C bus
- ATCA_STATUS [hal_i2c_sleep](#) ([ATCAIface](#) iface)
sleep CryptoAuth device using I2C bus
- ATCA_STATUS [hal_i2c_release](#) (void *hal_data)
manages reference count on given bus and releases resource if no more references exist

24.146.1 Detailed Description

ATCA Hardware abstraction layer for SAM flexcom & twi I2C over ASF drivers.

This code is structured in two parts. Part 1 is the connection of the ATCA HAL API to the physical I2C implementation. Part 2 is the ASF I2C primitives to set up the interface.

Prerequisite: add "TWI - Two-Wire Interface (Common API) (service)" module to application in Atmel Studio

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.147 hal_sam_i2c_asf.h File Reference

ATCA Hardware abstraction layer for SAMG55 I2C over ASF drivers.

```
#include <asf.h>
#include "cryptoauthlib.h"
```

Data Structures

- struct [i2c_sam_instance](#)

Typedefs

- typedef void(* **sam_change_baudrate**) ([ATCAIface](#) iface, uint32_t speed)
- typedef struct [i2c_sam_instance](#) **i2c_sam_instance_t**

24.147.1 Detailed Description

ATCA Hardware abstraction layer for SAMG55 I2C over ASF drivers.

Prerequisite: add "TWI - Two-Wire Interface (Common API) (service)" module to application in Atmel Studio

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.148 hal_sam_timer_asf.c File Reference

ATCA Hardware abstraction layer for SAMD21 timer/delay over ASF drivers.

```
#include <asf.h>
#include <delay.h>
#include "atca_hal.h"
```

Functions

- void [atca_delay_10us](#) (uint32_t delay)
This function delays for a number of tens of microseconds.
- void [atca_delay_us](#) (uint32_t delay)
This function delays for a number of microseconds.
- void [atca_delay_ms](#) (uint32_t ms)
Timer API for legacy implementations.

24.148.1 Detailed Description

ATCA Hardware abstraction layer for SAMD21 timer/delay over ASF drivers.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.149 hal_spi_harmony.c File Reference

ATCA Hardware abstraction layer for SPI over Harmony PLIB.

```
#include <string.h>
#include <stdio.h>
#include "atca_config.h"
#include "cryptoauthlib.h"
#include "atca_hal.h"
#include "atca_device.h"
#include "definitions.h"
#include "talib/talib_defines.h"
#include "talib/talib_fce.h"
```

Functions

- ATCA_STATUS [hal_spi_discover_buses](#) (int spi_buses[], int max_buses)
discover spi buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge
- ATCA_STATUS [hal_spi_discover_devices](#) (int bus_num, ATCAIfaceCfg cfg[], int *found)
discover any TA10x devices on a given logical bus number
- ATCA_STATUS [hal_spi_init](#) (ATCAIface iface, ATCAIfaceCfg *cfg)
initialize an SPI interface using given config
- ATCA_STATUS [hal_spi_post_init](#) (ATCAIface iface)
HAL implementation of SPI post init.
- ATCA_STATUS [hal_spi_select](#) (ATCAIface iface)
HAL implementation to assert the device chip select.
- ATCA_STATUS [hal_spi_deselect](#) (ATCAIface iface)
HAL implementation to deassert the device chip select.
- ATCA_STATUS [hal_spi_send](#) (ATCAIface iface, uint8_t word_address, uint8_t *txdata, int txlength)
HAL implementation of SPI send over Harmony.
- ATCA_STATUS [hal_spi_receive](#) (ATCAIface iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxlength)
HAL implementation of SPI receive function for HARMONY SPI.
- ATCA_STATUS [hal_spi_control](#) (ATCAIface iface, uint8_t option, void *param, size_t paramlen)
Perform control operations for the kit protocol.
- ATCA_STATUS [hal_spi_release](#) (void *hal_data)
manages reference count on given bus and releases resource if no more references exist

24.149.1 Detailed Description

ATCA Hardware abstraction layer for SPI over Harmony PLIB.

This code is structured in two parts. Part 1 is the connection of the ATCA HAL API to the physical SPI implementation. Part 2 is the Harmony SPI primitives to set up the interface.

Prerequisite: add SERCOM SPI Master Interrupt support to application in Mplab Harmony 3

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.150 hal_swi_gpio.c File Reference

ATCA Hardware abstraction layer for 1WIRE or SWI over GPIO.

```
#include "cryptoauthlib.h"
#include "hal_swi_gpio.h"
```

Functions

- ATCA_STATUS [hal_swi_gpio_init](#) ([ATCAIface](#) iface, [ATCAIfaceCfg](#) *cfg)
initialize an GPIO interface using given config
- ATCA_STATUS [hal_swi_gpio_post_init](#) ([ATCAIface](#) iface)
HAL implementation of GPIO post init.
- ATCA_STATUS [hal_swi_gpio_send](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *txdata, int txlength)
HAL implementation of bit banging send over Harmony.
- ATCA_STATUS [hal_swi_gpio_receive](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxlength)
HAL implementation of bit banging receive from HARMONY.
- ATCA_STATUS [hal_swi_gpio_control](#) ([ATCAIface](#) iface, uint8_t option, void *param, size_t paramlen)
Perform control operations.
- ATCA_STATUS [hal_swi_gpio_release](#) (void *hal_data)
releases resource if no more communication

24.150.1 Detailed Description

ATCA Hardware abstraction layer for 1WIRE or SWI over GPIO.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.150.2 Function Documentation

24.150.2.1 hal_swi_gpio_control()

```
ATCA_STATUS hal_swi_gpio_control (
    ATCAIface iface,
    uint8_t option,
    void * param,
    size_t paramlen )
```

Perform control operations.

Parameters

in	<i>iface</i>	Interface to interact with.
in	<i>option</i>	Control parameter identifier
in	<i>param</i>	Optional pointer to parameter value
in	<i>paramlen</i>	Length of the parameter

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.150.2.2 hal_swi_gpio_init()

```
ATCA_STATUS hal_swi_gpio_init (
    ATCAIface iface,
    ATCAIfaceCfg * cfg )
```

initialize an GPIO interface using given config

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.150.2.3 hal_swi_gpio_post_init()

```
ATCA_STATUS hal_swi_gpio_post_init (
    ATCAIface iface )
```

HAL implementation of GPIO post init.

Parameters

in	<i>iface</i>	ATCAIface instance
----	--------------	--------------------

Returns

ATCA_SUCCESS

24.150.2.4 hal_swi_gpio_receive()

```
ATCA_STATUS hal_swi_gpio_receive (
    ATCAIface iface,
```

24.150 hal_swi_gpio.c File Reference

```
uint8_t word_address,  
uint8_t * rxdata,  
uint16_t * rxlength )
```

HAL implementation of bit banging receive from HARMONY.

Parameters

in	<i>iface</i>	Device to interact with.
in	<i>word_address</i>	device transaction type
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.150.2.5 hal_swi_gpio_release()

```
ATCA_STATUS hal_swi_gpio_release (  
    void * hal_data )
```

releases resource if no more communication

Parameters

in	<i>hal_data</i>	- opaque pointer to hal data structure - known only to the HAL implementation
----	-----------------	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.150.2.6 hal_swi_gpio_send()

```
ATCA_STATUS hal_swi_gpio_send (  
    ATCAIface iface,  
    uint8_t word_address,  
    uint8_t * txdata,  
    int txlength )
```

HAL implementation of bit banging send over Harmony.

Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	device transaction type
in	<i>txdata</i>	pointer to space to bytes to send
in	<i>txlength</i>	number of bytes to send

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.151 hal_swi_gpio.h File Reference

ATCA Hardware abstraction layer for SWI over GPIO drivers.

```
#include <stdlib.h>
#include "cryptoauthlib.h"
#include "atca_status.h"
#include "atca_hal.h"
#include "atca_config.h"
```

Macros**Macros for Bit-Banged 1WIRE Timing**

Times to drive bits at 230.4 kbps.

- #define **tPUP** 0
- #define **tDSCHG** 150
- #define **tRESET** 96
- #define **tRRT** 1
- #define **tDRR** 1
- #define **tMSDR** 2
- #define **tHTSS** 150
- #define **tDACK** 2
- #define **tDACK_DLY** [atca_delay_us\(tDACK\)](#)
- #define **tRRT_DLY** [atca_delay_ms\(tRRT\)](#)
- #define **tDRR_DLY** [atca_delay_us\(tDRR\)](#)
- #define **tMSDR_DLY** [atca_delay_us\(tMSDR\)](#)
- #define **tDSCHG_DLY** [atca_delay_us\(tDSCHG\)](#)
- #define **tRESET_DLY** [atca_delay_us\(tRESET\)](#)
- #define **tHTSS_DLY** [atca_delay_us\(tHTSS\)](#)
- #define **tLOW0_MIN** 6
- #define **tLOW0_MAX** 16
- #define **tLOW1_MIN** 1
- #define **tLOW1_MAX** 2
- #define **tRCV_MIN** 4
- #define **tRCV_MAX** 6
- #define **tBIT_MIN** (tLOW0_MIN + tPUP + tRCV_MIN)
- #define **tBIT_MAX** 75
- #define **tWAKEUP** 1
- #define **tLOW0_TYPICAL** (tLOW0_MIN + ((tLOW0_MAX - tLOW0_MIN) / 2))
- #define **tLOW1_TYPICAL** (tLOW1_MIN + ((tLOW1_MAX - tLOW1_MIN) / 2))
- #define **tBIT_TYPICAL** (tBIT_MIN + ((tBIT_MAX - tBIT_MIN) / 2))
- #define **tLOW0_HDL** [atca_delay_us\(11\)](#)
- #define **tRD_HDL** [atca_delay_us\(1\)](#)
- #define **tLOW1_HDL** [atca_delay_us\(1\)](#)
- #define **tRCV0_HDL** [atca_delay_us\(11\)](#)
- #define **tRCV1_HDL** [atca_delay_us\(14\)](#)
- #define **tRD_DLY** [atca_delay_us\(1\)](#)
- #define **tHIGH_SPEED_DLY** [atca_delay_us\(1\)](#)
- #define **tSWIN_DLY** [atca_delay_us\(1\)](#)
- #define **tLOW0_DLY** [atca_delay_us\(tLOW0_TYPICAL\)](#)
- #define **tLOW1_DLY** [atca_delay_us\(tLOW1_TYPICAL\)](#)
- #define **tBIT_DLY** [atca_delay_us\(tBIT_TYPICAL\)](#)
- #define **tRCV0_DLY** [atca_delay_us\(tBIT_TYPICAL - tLOW0_TYPICAL\)](#)

- #define **tRCV1_DLY** [atca_delay_us](#)(tBIT_TYPICAL - tLOW1_TYPICAL)
- #define **send_logic0_1wire**(...) send_logic_bit(__VA_ARGS__, ATCA_GPIO_LOGIC_BIT0)
- #define **send_logic1_1wire**(...) send_logic_bit(__VA_ARGS__, ATCA_GPIO_LOGIC_BIT1)
- #define **send_ACK_1wire**(...) send_logic0_1wire(__VA_ARGS__)
- #define **send_NACK_1wire**(...) send_logic1_1wire(__VA_ARGS__)
- #define **ATCA_1WIRE_RESET_WORD_ADDR** 0x00
- #define **ATCA_1WIRE_SLEEP_WORD_ADDR** 0x01
- #define **ATCA_1WIRE_SLEEP_WORD_ADDR_ALTERNATE** 0x02
- #define **ATCA_1WIRE_COMMAND_WORD_ADDR** 0x03
- #define **ATCA_1WIRE_RESPONSE_LENGTH_SIZE** 0x01
- #define **ATCA_1WIRE_BIT_MASK** 0x80
- #define **ATCA_GPIO_WRITE** 0
- #define **ATCA_GPIO_READ** 1
- #define **ATCA_GPIO_INPUT_DIR** 0
- #define **ATCA_GPIO_OUTPUT_DIR** 1
- #define **ATCA_GPIO_LOGIC_BIT0** 0
- #define **ATCA_GPIO_LOGIC_BIT1** 1
- #define **ATCA_GPIO_ACK** ATCA_GPIO_LOGIC_BIT0
- #define **ATCA_GPIO_CLEAR** 0
- #define **ATCA_GPIO_SET** 1
- #define **ATCA_MIN_RESPONSE_LENGTH** 4
- #define **PIN_INPUT_DIR**(pin) PORT_GroupInputEnable(GET_PORT_GROUP(pin), GET_PIN_↔ MASK(pin))
- #define **PIN_OUTPUT_DIR**(pin) PORT_GroupOutputEnable(GET_PORT_GROUP(pin), GET_PIN_↔ MASK(pin))

Macros for Bit-Banged SWI Timing

Times to drive bits at 230.4 kbps.

- #define **BIT_DELAY_1L** [atca_delay_us](#)(4)
- #define **BIT_DELAY_1H** [atca_delay_us](#)(4)
should be 4.34 us, is 4.05us
- #define **BIT_DELAY_5** [atca_delay_us](#)(26)
- #define **BIT_DELAY_7** [atca_delay_us](#)(34)
- #define **RX_TX_DELAY** [atca_delay_us](#)(65)
- #define **ATCA_SWI_WAKE_WORD_ADDR** ((uint8_t)0x00)
- #define **ATCA_SWI_CMD_WORD_ADDR** ((uint8_t)0x77)
- #define **ATCA_SWI_TX_WORD_ADDR** ((uint8_t)0x88)
- #define **ATCA_SWI_IDLE_WORD_ADDR** ((uint8_t)0xBB)
- #define **ATCA_SWI_SLEEP_WORD_ADDR** ((uint8_t)0xCC)
- #define **ATCA_SWI_BIT_MASK** 0x01
- enum **protocol_type** { **ATCA_PROTOCOL_1WIRE** , **ATCA_PROTOCOL_SWI** , **NO_OF_PROTOCOL** }
- enum **delay_type** {
 LOGIC0_1 , **LOGIC0_2** , **LOGIC0_3** , **LOGIC0_4** ,
 LOGIC1_1 , **LOGIC1_2** , **NO_OF_DELAYS** }

24.151.1 Detailed Description

ATCA Hardware abstraction layer for SWI over GPIO drivers.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.151.2 Macro Definition Documentation

24.151.2.1 ATCA_SWI_WAKE_WORD_ADDR

```
#define ATCA_SWI_WAKE_WORD_ADDR ((uint8_t)0x00)
```

SWI WORD Address

24.151.2.2 BIT_DELAY_1L

```
#define BIT_DELAY_1L atca_delay_us(4)
```

delay macro for width of one pulse (start pulse or zero pulse) should be 4.34 us, is 4.05 us

24.151.2.3 BIT_DELAY_5

```
#define BIT_DELAY_5 atca_delay_us(26)
```

time to keep pin high for five pulses plus stop bit (used to bit-bang CryptoAuth 'zero' bit) should be 26.04 us, is 26.92 us

24.151.2.4 BIT_DELAY_7

```
#define BIT_DELAY_7 atca_delay_us(34)
```

time to keep pin high for seven bits plus stop bit (used to bit-bang CryptoAuth 'one' bit) should be 34.72 us, is 35.13 us

24.151.2.5 RX_TX_DELAY

```
#define RX_TX_DELAY atca_delay_us(65)
```

turn around time when switching from receive to transmit should be 93 us (Setting little less value as there would be other process before these steps)

24.152 hal_swi_uart.c File Reference

ATCA Hardware abstraction layer for SWI over UART drivers.

```
#include "cryptoauthlib.h"
```

Functions

- ATCA_STATUS [hal_swi_init](#) ([ATCAIface](#) iface, [ATCAIfaceCfg](#) *cfg)
initialize an SWI interface using given config
- ATCA_STATUS [hal_swi_post_init](#) ([ATCAIface](#) iface)
HAL implementation of SWI post init.
- ATCA_STATUS [hal_swi_send](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *txdata, int txlength)
HAL implementation of SWI send command over UART.
- ATCA_STATUS [hal_swi_receive](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxlength)
HAL implementation of SWI receive function over UART.
- ATCA_STATUS [hal_swi_wake](#) ([ATCAIface](#) iface)
Send Wake flag via SWI.
- ATCA_STATUS [hal_swi_sleep](#) ([ATCAIface](#) iface)
Send Sleep flag via SWI.
- ATCA_STATUS [hal_swi_idle](#) ([ATCAIface](#) iface)
Send Idle flag via SWI.
- ATCA_STATUS [hal_swi_control](#) ([ATCAIface](#) iface, uint8_t option, void *param, size_t paramlen)
Perform control operations for the kit protocol.
- ATCA_STATUS [hal_swi_release](#) (void *hal_data)
manages reference count on given bus and releases resource if no more references exist

24.152.1 Detailed Description

ATCA Hardware abstraction layer for SWI over UART drivers.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.153 hal_timer_start.c File Reference

ATCA Hardware abstraction layer for SAMD21 I2C over START drivers.

```
#include <hal_delay.h>
#include "atca_hal.h"
```

Functions

- void [atca_delay_us](#) (uint32_t delay)
This function delays for a number of microseconds.
- void [atca_delay_10us](#) (uint32_t delay)
This function delays for a number of tens of microseconds.
- void [atca_delay_ms](#) (uint32_t ms)
Timer API for legacy implementations.

24.153.1 Detailed Description

ATCA Hardware abstraction layer for SAMD21 I2C over START drivers.

Prerequisite: add SERCOM I2C Master Polled support to application in Atmel Studio

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.154 hal_uart_harmony.c File Reference

ATCA Hardware abstraction layer for SWI uart over Harmony PLIB.

```
#include "atca_config.h"
#include "cryptoauthlib.h"
```

Functions

- ATCA_STATUS [hal_uart_init](#) ([ATCAIface](#) iface, [ATCAIfaceCfg](#) *cfg)
Initialize an uart interface using given config.
- ATCA_STATUS [hal_uart_post_init](#) ([ATCAIface](#) iface)
HAL implementation of SWI post init.
- ATCA_STATUS [hal_uart_send](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *txdata, int txlength)
Send byte(s) via SWI.
- ATCA_STATUS [hal_uart_receive](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxlength)
Receive byte(s) via SWI.
- ATCA_STATUS [hal_uart_control](#) ([ATCAIface](#) iface, uint8_t option, void *param, size_t paramlen)
- ATCA_STATUS [hal_uart_release](#) (void *hal_data)
Manages reference count on given bus and releases resource if no more reference(s) exist.

Variables

- PLIB_SWI_SERIAL_SETUP [serial_setup](#)

24.154.1 Detailed Description

ATCA Hardware abstraction layer for SWI uart over Harmony PLIB.

This code is structured in two parts. Part 1 is the connection of the ATCA HAL API to the physical I2C implementation. Part 2 is the Harmony UART (ring buffer mode) primitives to set up the interface.

Copyright

(c) 2015-2018 Microchip Technology Inc. and its subsidiaries.

24.154.2 Function Documentation

24.154.2.1 hal_uart_init()

```
ATCA_STATUS hal_uart_init (
    ATCAIface iface,
    ATCAIfaceCfg * cfg )
```

Initialize an uart interface using given config.

Parameters

in	<i>hal</i>	opaque pointer to HAL data
in	<i>cfg</i>	interface configuration

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.154.2.2 hal_uart_post_init()

```
ATCA_STATUS hal_uart_post_init (
    ATCAIface iface )
```

HAL implementation of SWI post init.

Parameters

in	<i>iface</i>	ATCAIface instance
----	--------------	--------------------

Returns

ATCA_SUCCESS

24.154.2.3 hal_uart_receive()

```
ATCA_STATUS hal_uart_receive (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * rxdata,
    uint16_t * rxlength )
```

Receive byte(s) via SWI.

Parameters

in	<i>iface</i>	Device to interact with.
in	<i>word_address</i>	device transaction type
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.154.2.4 hal_uart_release()

```
ATCA_STATUS hal_uart_release (
    void * hal_data )
```

Manages reference count on given bus and releases resource if no more reference(s) exist.

Parameters

in	<i>hal_data</i>	opaque pointer to hal data structure - known only to the HAL implementation
----	-----------------	---

Returns

ATCA_SUCCESS

24.154.2.5 hal_uart_send()

```
ATCA_STATUS hal_uart_send (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * txdata,
    int txlength )
```

Send byte(s) via SWI.

Parameters

in	<i>iface</i>	interface of the logical device to send data to
in	<i>word_address</i>	device transaction type
in	<i>txdata</i>	pointer to bytes to send
in	<i>txlength</i>	number of bytes to send

Returns

ATCA_SUCCESS

24.154.3 Variable Documentation

24.154.3.1 serial_setup

```
PLIB_SWI_SERIAL_SETUP serial_setup
```


Initial value:

```
= {
    .parity      = PLIB_SWI_PARITY_NONE,
    .dataWidth   = PLIB_SWI_DATA_WIDTH,
    .stopBits    = PLIB_SWI_STOP_BIT
}
```

24.155 hal_uc3_i2c_asf.c File Reference

ATCA Hardware abstraction layer for SAMV71 I2C over ASF drivers.

```
#include <asf.h>
#include <string.h>
#include <stdio.h>
#include "cryptoauthlib.h"
#include "hal_uc3_i2c_asf.h"
```

Functions

- ATCA_STATUS [hal_i2c_discover_buses](#) (int i2c_buses[], int max_buses)
discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge
- ATCA_STATUS [hal_i2c_discover_devices](#) (int bus_num, [ATCAIfaceCfg](#) cfg[], int *found)
discover any CryptoAuth devices on a given logical bus number
- ATCA_STATUS [hal_i2c_init](#) (void *hal, [ATCAIfaceCfg](#) *cfg)
hal_i2c_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIface instances using the same bus, and you can have multiple ATCAIface instances on multiple i2c buses, so hal_i2c_init manages these things and ATCAIface is abstracted from the physical details.
- ATCA_STATUS [hal_i2c_post_init](#) ([ATCAIface](#) iface)
HAL implementation of I2C post init.
- ATCA_STATUS [hal_i2c_send](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *txdata, int txlength)
HAL implementation of I2C send over START.
- ATCA_STATUS [hal_i2c_receive](#) ([ATCAIface](#) iface, uint8_t address, uint8_t *rxdata, uint16_t *rxlength)
HAL implementation of I2C receive function for START I2C.
- ATCA_STATUS [change_i2c_speed](#) ([ATCAIface](#) iface, uint32_t speed)
method to change the bus speed of I2C
- ATCA_STATUS [hal_i2c_wake](#) ([ATCAIface](#) iface)
wake up CryptoAuth device using I2C bus
- ATCA_STATUS [hal_i2c_idle](#) ([ATCAIface](#) iface)
idle CryptoAuth device using I2C bus
- ATCA_STATUS [hal_i2c_sleep](#) ([ATCAIface](#) iface)
sleep CryptoAuth device using I2C bus
- ATCA_STATUS [hal_i2c_release](#) (void *hal_data)
manages reference count on given bus and releases resource if no more references exist

24.155.1 Detailed Description

ATCA Hardware abstraction layer for SAMV71 I2C over ASF drivers.

This code is structured in two parts. Part 1 is the connection of the ATCA HAL API to the physical I2C implementation. Part 2 is the ASF I2C primitives to set up the interface.

Prerequisite: add SERCOM I2C Master Polled support to application in Atmel Studio

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.156 hal_uc3_i2c_asf.h File Reference

ATCA Hardware abstraction layer for SAMV71 I2C over ASF drivers.

```
#include <asf.h>
#include "twi.h"
```

Data Structures

- struct [atcal2Cmaster](#)
this is the hal_data for ATCA HAL for ASF SERCOM

Macros

- #define **MAX_I2C_BUSES** 3

Typedefs

- typedef struct [atcal2Cmaster](#) **ATCAI2CMaster_t**
this is the hal_data for ATCA HAL for ASF SERCOM

Functions

- ATCA_STATUS [change_i2c_speed](#) ([ATCAIface](#) iface, uint32_t speed)
method to change the bus speed of I2C

24.156.1 Detailed Description

ATCA Hardware abstraction layer for SAMV71 I2C over ASF drivers.

Prerequisite: add SERCOM I2C Master Polled support to application in Atmel Studio

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.157 hal_uc3_timer_asf.c File Reference

ATCA Hardware abstraction layer for SAM4S I2C over ASF drivers.

```
#include <asf.h>
#include <delay.h>
#include "atca_hal.h"
```

Functions

- void [atca_delay_us](#) (uint32_t delay)
This function delays for a number of microseconds.
- void [atca_delay_10us](#) (uint32_t delay)
This function delays for a number of tens of microseconds.
- void [atca_delay_ms](#) (uint32_t ms)
Timer API for legacy implementations.

24.157.1 Detailed Description

ATCA Hardware abstraction layer for SAM4S I2C over ASF drivers.

Prerequisite: add "Delay routines (service)" module to application in Atmel Studio

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.158 hal_windows.c File Reference

ATCA Hardware abstraction layer for windows timer functions.

```
#include "atca_hal.h"
#include <windows.h>
#include <math.h>
```

Functions

- void [hal_delay_us](#) (uint32_t delay)
This function delays for a number of microseconds.
- void [hal_delay_ms](#) (uint32_t delay)
Timer API implemented at the HAL level.
- ATCA_STATUS [hal_create_mutex](#) (void **ppMutex, const char *pName)
Optional hal interfaces.
- ATCA_STATUS [hal_destroy_mutex](#) (void *pMutex)
- ATCA_STATUS [hal_lock_mutex](#) (void *pMutex)
- ATCA_STATUS [hal_unlock_mutex](#) (void *pMutex)
- ATCA_STATUS [hal_check_pid](#) (hal_pid_t pid)
Check if the pid exists in the system.

24.158.1 Detailed Description

ATCA Hardware abstraction layer for windows timer functions.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.159 hal_windows_kit_uart.c File Reference

ATCA Hardware abstraction layer for Windows using UART.

```
#include "cryptoauthlib.h"
#include "atca_hal.h"
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <string.h>
```

Data Structures

- struct [atca_uart_host_s](#)

Typedefs

- typedef struct [atca_uart_host_s](#) [atca_uart_host_t](#)

Functions

- ATCA_STATUS [hal_uart_init](#) ([ATCAIface](#) iface, [ATCAIfaceCfg](#) *cfg)
HAL implementation of UART init.
- ATCA_STATUS [hal_uart_post_init](#) ([ATCAIface](#) iface)
HAL implementation of UART post init.
- ATCA_STATUS [hal_uart_send](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *txdata, int txlength)
HAL implementation of UART send.
- ATCA_STATUS [hal_uart_receive](#) ([ATCAIface](#) iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxlength)
HAL implementation of UART receive function.
- ATCA_STATUS [hal_uart_control](#) ([ATCAIface](#) iface, uint8_t option, void *param, size_t paramlen)
Perform control operations for the UART.
- ATCA_STATUS [hal_uart_release](#) (void *hal_data)
manages reference count on given bus and releases resource if no more references exist

24.159.1 Detailed Description

ATCA Hardware abstraction layer for Windows using UART.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.159.2 Function Documentation

24.159.2.1 hal_uart_control()

```
ATCA_STATUS hal_uart_control (
    ATCAIface iface,
    uint8_t option,
    void * param,
    size_t paramlen )
```

Perform control operations for the UART.

Parameters

in	<i>iface</i>	Interface to interact with.
in	<i>option</i>	Control parameter identifier
in	<i>param</i>	Optional pointer to parameter value
in	<i>paramlen</i>	Length of the parameter

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.159.2.2 hal_uart_init()

```
ATCA_STATUS hal_uart_init (
    ATCAIface iface,
    ATCAIfaceCfg * cfg )
```

HAL implementation of UART init.

this implementation assumes UART SERIAL PORT peripheral has been enabled by user . It only initialize an UART interface using given config.

Parameters

in	<i>hal</i>	pointer to HAL specific data that is maintained by this HAL
in	<i>cfg</i>	pointer to HAL specific configuration data that is used to initialize this HAL

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.159.2.3 hal_uart_post_init()

```
ATCA_STATUS hal_uart_post_init (
    ATCAIface iface )
```

HAL implementation of UART post init.

Parameters

in	<i>iface</i>	instance
----	--------------	----------

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.159.2.4 hal_uart_receive()

```
ATCA_STATUS hal_uart_receive (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * rxdata,
    uint16_t * rxlength )
```

HAL implementation of UART receive function.

Parameters

in	<i>iface</i>	Device to interact with.
in	<i>word_address</i>	device transaction type
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.159.2.5 hal_uart_release()

```
ATCA_STATUS hal_uart_release (
    void * hal_data )
```

manages reference count on given bus and releases resource if no more refences exist

Parameters

in	<i>hal_data</i>	- opaque pointer to hal data structure - known only to the HAL implementation
----	-----------------	---

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.159.2.6 hal_uart_send()

```
ATCA_STATUS hal_uart_send (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * txdata,
    int txlength )
```

HAL implementation of UART send.

Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	transaction type
in	<i>txdata</i>	data to be send to device
in	<i>txdata</i>	pointer to space to bytes to send
in	<i>len</i>	number of bytes to send

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.160 kit_protocol.c File Reference

Microchip Crypto Auth hardware interface object.

```
#include <stdlib.h>
#include <stdio.h>
#include <limits.h>
#include "atca_compiler.h"
#include "kit_protocol.h"
#include "atca_helpers.h"
```

Macros

- #define KIT_MAX_SCAN_COUNT 8
- #define KIT_MAX_TX_BUF 32
- #define KIT_TA_WORDADD_SEND_LEN 20

Functions

- const char * [kit_id_from_devtype](#) (ATCADeviceType devtype)
- const char * [kit_interface_from_kittype](#) (ATCAKitType kittype)
- const char * [kit_interface](#) (ATCAKitType kittype)

24.160.1 Detailed Description

Microchip Crypto Auth hardware interface object.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.161 kit_protocol.h File Reference

```
#include "cryptoauthlib.h"
```

Macros

- #define **KIT_TX_WRAP_SIZE** (10)
- #define **KIT_MSG_SIZE** (32u)
- #define **KIT_RX_WRAP_SIZE** (KIT_MSG_SIZE + 6u)

Functions

- ATCA_STATUS **kit_init** ([ATCAIface](#) iface, [ATCAIfaceCfg](#) *cfg)
- ATCA_STATUS **kit_post_init** ([ATCAIface](#) iface)
- ATCA_STATUS **kit_send** ([ATCAIface](#) iface, uint8_t word_address, uint8_t *txdata, int txlength)
- ATCA_STATUS **kit_receive** ([ATCAIface](#) iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxsize)
- ATCA_STATUS **kit_control** ([ATCAIface](#) iface, uint8_t option, void *param, size_t paramlen)
- ATCA_STATUS **kit_release** (void *hal_data)
- ATCA_STATUS **kit_wrap_cmd** ([ATCAIface](#) iface, uint8_t word_address, const uint8_t *txdata, int txlen, char *pkitchmd, int *nkitchmd)
- ATCA_STATUS **kit_parse_rsp** (const char *pkitchbuf, int nkitbuf, uint8_t *kitstatus, uint8_t *rxdata, int *datasize)
- ATCA_STATUS **kit_wake** ([ATCAIface](#) iface)
- ATCA_STATUS **kit_idle** ([ATCAIface](#) iface)
- ATCA_STATUS **kit_sleep** ([ATCAIface](#) iface)
- ATCA_STATUS **kit_phy_send** ([ATCAIface](#) iface, uint8_t *txdata, int txlength)
- ATCA_STATUS **kit_phy_receive** ([ATCAIface](#) iface, uint8_t *rxdata, int *rxsize)
- const char * [kit_id_from_devtype](#) (ATCADeviceType devtype)
- const char * [kit_interface_from_kittype](#) (ATCAKitType kittype)
- const char * [kit_interface](#) (ATCAKitType kittype)

24.161.1 Detailed Description

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.162 swi_uart_samd21_asf.c File Reference

ATXMEGA's ATCA Hardware abstraction layer for SWI interface over UART drivers.

```
#include <stdlib.h>
#include <stdio.h>
#include "swi_uart_samd21_asf.h"
#include "atca_helpers.h"
```

Functions

- ATCA_STATUS [swi_uart_init](#) ([ATCASWIMaster_t](#) *instance)
Implementation of SWI UART init.
- ATCA_STATUS [swi_uart_deinit](#) ([ATCASWIMaster_t](#) *instance)
Implementation of SWI UART deinit.
- void [swi_uart_setbaud](#) ([ATCASWIMaster_t](#) *instance, uint32_t baudrate)
implementation of SWI UART change baudrate.
- void [swi_uart_mode](#) ([ATCASWIMaster_t](#) *instance, uint8_t mode)
implementation of SWI UART change mode.
- void [swi_uart_discover_buses](#) (int swi_uart_buses[], int max_buses)
discover UART buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge
- ATCA_STATUS [swi_uart_send_byte](#) ([ATCASWIMaster_t](#) *instance, uint8_t data)
HAL implementation of SWI UART send byte over ASF. This function send one byte over UART.
- ATCA_STATUS [swi_uart_receive_byte](#) ([ATCASWIMaster_t](#) *instance, uint8_t *data)
HAL implementation of SWI UART receive bytes over ASF. This function receive one byte over UART.

Variables

- struct port_config [pin_conf](#)

24.162.1 Detailed Description

ATXMEGA's ATCA Hardware abstraction layer for SWI interface over UART drivers.

Prerequisite: add UART Polled support to application in Atmel Studio

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.163 swi_uart_samd21_asf.h File Reference

ATXMEGA's ATCA Hardware abstraction layer for SWI interface over UART drivers.

```
#include <asf.h>
#include "cryptoauthlib.h"
```

Data Structures

- struct [atcaSWImaster](#)
this is the hal_data for ATCA HAL for ASF SERCOM

Macros

- #define [MAX_SWI_BUSES](#) 6
- #define [RECEIVE_MODE](#) 0
- #define [TRANSMIT_MODE](#) 1
- #define [RX_DELAY](#) 10
- #define [TX_DELAY](#) 90
- #define [DEBUG_PIN_1](#) EXT2_PIN_5
- #define [DEBUG_PIN_2](#) EXT2_PIN_6

Typedefs

- typedef struct [atcaSWImaster](#) [ATCASWIMaster_t](#)
this is the hal_data for ATCA HAL for ASF SERCOM

Functions

- ATCA_STATUS [swi_uart_init](#) ([ATCASWIMaster_t](#) *instance)
Implementation of SWI UART init.
- ATCA_STATUS [swi_uart_deinit](#) ([ATCASWIMaster_t](#) *instance)
Implementation of SWI UART deinit.
- void [swi_uart_setbaud](#) ([ATCASWIMaster_t](#) *instance, uint32_t baudrate)
implementation of SWI UART change baudrate.
- void [swi_uart_mode](#) ([ATCASWIMaster_t](#) *instance, uint8_t mode)
implementation of SWI UART change mode.
- void [swi_uart_discover_buses](#) (int swi_uart_buses[], int max_buses)
discover UART buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge
- ATCA_STATUS [swi_uart_send_byte](#) ([ATCASWIMaster_t](#) *instance, uint8_t data)
HAL implementation of SWI UART send byte over ASF. This function send one byte over UART.
- ATCA_STATUS [swi_uart_receive_byte](#) ([ATCASWIMaster_t](#) *instance, uint8_t *data)
HAL implementation of SWI UART receive bytes over ASF. This function receive one byte over UART.

24.163.1 Detailed Description

ATXMEGA's ATCA Hardware abstraction layer for SWI interface over UART drivers.

Prerequisite: add UART Polled support to application in Atmel Studio

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.164 swi_uart_start.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <peripheral_clk_config.h>
#include "swi_uart_start.h"
#include "atca_helpers.h"
```

Macros

- #define **USART_BAUD_RATE**(baud, sercom_freq) (65536 - ((65536 * 16.0F * baud) / sercom_freq))

Functions

- ATCA_STATUS [swi_uart_init](#) ([ATCASWIMaster_t](#) *instance)
Implementation of SWI UART init.
- ATCA_STATUS [swi_uart_deinit](#) ([ATCASWIMaster_t](#) *instance)
Implementation of SWI UART deinit.
- void [swi_uart_setbaud](#) ([ATCASWIMaster_t](#) *instance, uint32_t baudrate)
implementation of SWI UART change baudrate.
- void [swi_uart_mode](#) ([ATCASWIMaster_t](#) *instance, uint8_t mode)
implementation of SWI UART change mode.
- void [swi_uart_discover_buses](#) (int swi_uart_buses[], int max_buses)
discover UART buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge
- ATCA_STATUS [swi_uart_send_byte](#) ([ATCASWIMaster_t](#) *instance, uint8_t data)
HAL implementation of SWI UART send byte over ASF. This function send one byte over UART.
- ATCA_STATUS [swi_uart_receive_byte](#) ([ATCASWIMaster_t](#) *instance, uint8_t *data)
HAL implementation of SWI UART receive bytes over ASF. This function receive one byte over UART.

24.164.1 Detailed Description

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.165 swi_uart_start.h File Reference

```
#include <stdlib.h>
#include "atmel_start.h"
#include "cryptoauthlib.h"
```

Data Structures

- struct [atcaSWImaster](#)
this is the hal_data for ATCA HAL for ASF SERCOM

Macros

- #define [MAX_SWI_BUSES](#) 6
- #define [RECEIVE_MODE](#) 0
- #define [TRANSMIT_MODE](#) 1
- #define [RX_DELAY](#) 10
- #define [TX_DELAY](#) 93

Typedefs

- typedef struct [atcaSWImaster](#) [ATCASWIMaster_t](#)
this is the hal_data for ATCA HAL for ASF SERCOM

Functions

- ATCA_STATUS [swi_uart_init](#) ([ATCASWIMaster_t](#) *instance)
Implementation of SWI UART init.
- ATCA_STATUS [swi_uart_deinit](#) ([ATCASWIMaster_t](#) *instance)
Implementation of SWI UART deinit.
- void [swi_uart_setbaud](#) ([ATCASWIMaster_t](#) *instance, uint32_t baudrate)
implementation of SWI UART change baudrate.
- void [swi_uart_mode](#) ([ATCASWIMaster_t](#) *instance, uint8_t mode)
implementation of SWI UART change mode.
- void [swi_uart_discover_buses](#) (int swi_uart_buses[], int max_buses)
discover UART buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge
- ATCA_STATUS [swi_uart_send_byte](#) ([ATCASWIMaster_t](#) *instance, uint8_t data)
HAL implementation of SWI UART send byte over ASF. This function send one byte over UART.
- ATCA_STATUS [swi_uart_receive_byte](#) ([ATCASWIMaster_t](#) *instance, uint8_t *data)
HAL implementation of SWI UART receive bytes over ASF. This function receive one byte over UART.

24.165.1 Detailed Description

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.166 atca_host.c File Reference

Host side methods to support CryptoAuth computations.

```
#include "atca_host.h"
#include "crypto/atca_crypto_sw_sha2.h"
#include "cal_internal.h"
```

24.166.1 Detailed Description

Host side methods to support CryptoAuth computations.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.167 atca_host.h File Reference

Definitions and Prototypes for ATCA Utility Functions.

```
#include <stdint.h>
#include "cryptoauthlib.h"
#include "calib/calib_basic.h"
#include "atca_host_config_check.h"
```

Data Structures

- struct [atca_temp_key](#)
Structure to hold TempKey fields.
- struct [atca_include_data_in_out](#)
Input / output parameters for function atca_include_data().
- struct [atca_nonce_in_out](#)
Input/output parameters for function atca_nonce().
- struct [atca_io_decrypt_in_out](#)
- struct [atca_verify_mac](#)
- struct [atca_secureboot_enc_in_out](#)
- struct [atca_secureboot_mac_in_out](#)
- struct [atca_mac_in_out](#)
Input/output parameters for function atca_mac().
- struct [atca_hmac_in_out](#)
Input/output parameters for function atca_hmac().
- struct [atca_gen_dig_in_out](#)
Input/output parameters for function atcah_gen_dig().
- struct [atca_diversified_key_in_out](#)
Input/output parameters for function atcah_gendivkey().
- struct [atca_write_mac_in_out](#)
Input/output parameters for function atcah_write_auth_mac() and atcah_privwrite_auth_mac().

- struct [atca_derive_key_in_out](#)
Input/output parameters for function atcah_derive_key().
- struct [atca_derive_key_mac_in_out](#)
Input/output parameters for function atcah_derive_key_mac().
- struct [atca_decrypt_in_out](#)
Input/output parameters for function atca_decrypt().
- struct [atca_check_mac_in_out](#)
Input/output parameters for function atcah_check_mac().
- struct [atca_resp_mac_in_out](#)
*Input/Output parameters for calculating the output response mac in SHA105 device. Used with the atcah_gen_↔
output_resp_mac() function.*
- struct [atca_verify_in_out](#)
Input/output parameters for function atcah_verify().
- struct [atca_gen_key_in_out](#)
*Input/output parameters for calculating the PubKey digest put into TempKey by the GenKey command with the atcah_↔
_gen_key_msg() function.*
- struct [atca_sign_internal_in_out](#)
*Input/output parameters for calculating the message and digest used by the Sign(internal) command. Used with the
atcah_sign_internal_msg() function.*
- struct [atca_session_key_in_out](#)
*Input/Output paramters for calculating the session key by the nonce command. Used with the atcah_gen_session_↔
_key() function.*
- struct [atca_delete_in_out](#)
Input/Output paramters for calculating the mac.Used with Delete command.

Macros

Definitions for ATECC Message Sizes to Calculate a SHA256 Hash

"||" is the concatenation operator. The number in braces is the length of the hash input value in bytes.

- #define **ATCA_MSG_SIZE_NONCE** (55)
RandOut{32} || NumIn{20} || OpCode{1} || Mode{1} || LSB of Param2{1}.
- #define **ATCA_MSG_SIZE_MAC** (88)
*(Key or TempKey){32} || (Challenge or TempKey){32} || OpCode{1} || Mode{1} || Param2{2} || (OTP0_7 or 0){8} ||
(OTP8_10 or 0){3} || SN8{1} || (SN4_7 or 0){4} || SN0_1{2} || (SN2_3 or 0){2}*
- #define **ATCA_MSG_SIZE_HMAC** (88u)
- #define **ATCA_MSG_SIZE_GEN_DIG** (96)
KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0_1{2} || 0{25} || TempKey{32}.
- #define **ATCA_MSG_SIZE_DIVERSIFIED_KEY** (96)
ParentKey{32} || OtherData{4} || SN8{1} || SN0_1{2} || 0{25} || InputData{32}.
- #define **ATCA_MSG_SIZE_DERIVE_KEY** (96)
KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0_1{2} || 0{25} || TempKey{32}.
- #define **ATCA_MSG_SIZE_DERIVE_KEY_MAC** (39)
KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0_1{2}.
- #define **ATCA_MSG_SIZE_ENCRYPT_MAC** (96)
KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0_1{2} || 0{25} || TempKey{32}.
- #define **ATCA_MSG_SIZE_SESSION_KEY** (96)
TransportKey{32} || 0x15{1} || 0x00{1} || KeyId{2} || SN8{1} || SN0_1{2} || 0{25} || Nonce{32}.
- #define **ATCA_MSG_SIZE_DELETE_MAC** (96)
Hmac/SecretKey{32} || 0x13{1} || 0x00{1} || 0x0000{2} || SN8{1} || SN0_1{2} || 0{25} || Nonce{32}.
- #define **ATCA_MSG_SIZE_RESPONSE_MAC** (97)
*SlotKey{32} || Opcode{1} || Param1{1} || Param2{2} || SN8{1} || SN0_1{2} || 0{25} || client_Resp{32} ||
checkmac_result{1}.*
- #define **ATCA_MSG_SIZE_PRIVWRITE_MAC** (96)
KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0_1{2} || 0{21} || PlainText{36}.

- #define **ATCA_COMMAND_HEADER_SIZE** (4)
- #define **ATCA_GENDIG_ZEROS_SIZE** (25)
- #define **ATCA_GENDIVKEY_ZEROS_SIZE** (25)
- #define **ATCA_WRITE_MAC_ZEROS_SIZE** (25)
- #define **ATCA_DELETE_MAC_ZEROS_SIZE** (25)
- #define **ATCA_RESP_MAC_ZEROS_SIZE** (25)
- #define **ATCA_PRIVWRITE_MAC_ZEROS_SIZE** (21)
- #define **ATCA_PRIVWRITE_PLAIN_TEXT_SIZE** (36)
- #define **ATCA_DERIVE_KEY_ZEROS_SIZE** (25)
- #define **ATCA_HMAC_BLOCK_SIZE** (64u)
- #define **ATCA_ENCRYPTION_KEY_SIZE** (64)

Definition for TempKey Mode

- #define **MAC_MODE_USE_TEMPKEY_MASK** ((uint8_t)0x03)
mode mask for MAC command when using TempKey

Typedefs

- typedef struct [atca_temp_key](#) **atca_temp_key_t**
Structure to hold TempKey fields.
- typedef struct [atca_nonce_in_out](#) **atca_nonce_in_out_t**
- typedef struct [atca_io_decrypt_in_out](#) **atca_io_decrypt_in_out_t**
- typedef struct [atca_verify_mac](#) **atca_verify_mac_in_out_t**
- typedef struct [atca_secureboot_enc_in_out](#) **atca_secureboot_enc_in_out_t**
- typedef struct [atca_secureboot_mac_in_out](#) **atca_secureboot_mac_in_out_t**
- typedef struct [atca_mac_in_out](#) **atca_mac_in_out_t**
- typedef struct [atca_gen_dig_in_out](#) **atca_gen_dig_in_out_t**
Input/output parameters for function atcah_gen_dig().
- typedef struct [atca_diversified_key_in_out](#) **atca_diversified_key_in_out_t**
Input/output parameters for function atcah_gendivkey().
- typedef struct [atca_write_mac_in_out](#) **atca_write_mac_in_out_t**
Input/output parameters for function atcah_write_auth_mac() and atcah_privwrite_auth_mac().
- typedef struct [atca_check_mac_in_out](#) **atca_check_mac_in_out_t**
Input/output parameters for function atcah_check_mac().
- typedef struct [atca_resp_mac_in_out](#) **atca_resp_mac_in_out_t**
*Input/Output parameters for calculating the output response mac in SHA105 device. Used with the atcah_gen↔
output_resp_mac() function.*
- typedef struct [atca_verify_in_out](#) **atca_verify_in_out_t**
- typedef struct [atca_gen_key_in_out](#) **atca_gen_key_in_out_t**
*Input/output parameters for calculating the PubKey digest put into TempKey by the GenKey command with the atcah↔
_gen_key_msg() function.*
- typedef struct [atca_sign_internal_in_out](#) **atca_sign_internal_in_out_t**
*Input/output parameters for calculating the message and digest used by the Sign(internal) command. Used with the
atcah_sign_internal_msg() function.*
- typedef struct [atca_session_key_in_out](#) **atca_session_key_in_out_t**
*Input/Output paramters for calculating the session key by the nonce command. Used with the atcah_gen_session↔
_key() function.*
- typedef struct [atca_delete_in_out](#) **atca_delete_in_out_t**
Input/Output paramters for calculating the mac.Used with Delete command.

Functions

- ATCA_STATUS **atcah_nonce** (struct [atca_nonce_in_out](#) *param)
- ATCA_STATUS **atcah_mac** (struct [atca_mac_in_out](#) *param)
- ATCA_STATUS **atcah_check_mac** (struct [atca_check_mac_in_out](#) *param)
- ATCA_STATUS **atcah_hmac** (struct [atca_hmac_in_out](#) *param)
- ATCA_STATUS **atcah_gen_dig** (struct [atca_gen_dig_in_out](#) *param)
- ATCA_STATUS **atcah_gendivkey** (struct [atca_diversified_key_in_out](#) *param)
- ATCA_STATUS **atcah_gen_mac** (struct [atca_gen_dig_in_out](#) *param)
- ATCA_STATUS **atcah_write_auth_mac** (struct [atca_write_mac_in_out](#) *param)
- ATCA_STATUS **atcah_privwrite_auth_mac** (struct [atca_write_mac_in_out](#) *param)
- ATCA_STATUS **atcah_derive_key** (struct [atca_derive_key_in_out](#) *param)
- ATCA_STATUS **atcah_derive_key_mac** (struct [atca_derive_key_mac_in_out](#) *param)
- ATCA_STATUS **atcah_decrypt** (struct [atca_decrypt_in_out](#) *param)
- ATCA_STATUS **atcah_sha256** (uint32_t len, const uint8_t *message, uint8_t *digest)
- uint8_t * **atcah_include_data** (struct [atca_include_data_in_out](#) *param)
- ATCA_STATUS **atcah_gen_key_msg** (struct [atca_gen_key_in_out](#) *param)
- ATCA_STATUS **atcah_config_to_sign_internal** (ATCADeviceType device_type, struct [atca_sign_internal_in_out](#) *param, const uint8_t *config)
- ATCA_STATUS **atcah_sign_internal_msg** (ATCADeviceType device_type, struct [atca_sign_internal_in_out](#) *param)
- ATCA_STATUS **atcah_verify_mac** ([atca_verify_mac_in_out_t](#) *param)
- ATCA_STATUS **atcah_secureboot_enc** ([atca_secureboot_enc_in_out_t](#) *param)
- ATCA_STATUS **atcah_secureboot_mac** ([atca_secureboot_mac_in_out_t](#) *param)
- ATCA_STATUS **atcah_encode_counter_match** (uint32_t counter_value, uint8_t *counter_match_value)
- ATCA_STATUS **atcah_io_decrypt** (struct [atca_io_decrypt_in_out](#) *param)
- ATCA_STATUS **atcah_ecc204_write_auth_mac** (struct [atca_write_mac_in_out](#) *param)
- ATCA_STATUS **atcah_gen_session_key** ([atca_session_key_in_out_t](#) *param)
- ATCA_STATUS **atcah_gen_output_resp_mac** (struct [atca_resp_mac_in_out](#) *param)

24.167.1 Detailed Description

Definitions and Prototypes for ATCA Utility Functions.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.168 atca_host_config_check.h File Reference

Consistency checks for configuration options.

Macros

- #define [ATCAH_INCLUDE_DATA](#) (DEFAULT_ENABLED)
- #define [ATCAH_NONCE](#) (DEFAULT_ENABLED)
- #define [ATCAH_IO_DECRYPT](#) (DEFAULT_ENABLED)
- #define [ATCAH_VERIFY_MAC](#) (DEFAULT_ENABLED)
- #define [ATCAH_SECUREBOOT_ENC](#) (DEFAULT_ENABLED)
- #define [ATCAH_SECUREBOOT_MAC](#) (DEFAULT_ENABLED)
- #define [ATCAH_MAC](#) (DEFAULT_ENABLED)
- #define [ATCAH_CHECK_MAC](#) (DEFAULT_ENABLED)
- #define [ATCAH_GEN_OUTPUT_RESP_MAC](#) (DEFAULT_ENABLED)
- #define [ATCAH_HMAC](#) (DEFAULT_ENABLED)
- #define [ATCAH_GENDIG](#) (DEFAULT_ENABLED)
- #define [ATCAH_GENDIVKEY](#) (DEFAULT_ENABLED)
- #define [ATCAH_GEN_MAC](#) (DEFAULT_ENABLED)
- #define [ATCAH_WRITE_AUTH_MAC](#) (DEFAULT_ENABLED)
- #define [ATCAH_PRIVWRITE_AUTH_MAC](#) (DEFAULT_ENABLED)
- #define [ATCAH_DERIVE_KEY](#) (DEFAULT_ENABLED)
- #define [ATCAH_DERIVE_KEY_MAC](#) (DEFAULT_ENABLED)
- #define [ATCAH_DECRYPT](#) (DEFAULT_ENABLED)
- #define [ATCAH_SHA256](#) (DEFAULT_ENABLED)
- #define [ATCAH_GEN_KEY_MSG](#) (DEFAULT_ENABLED)
- #define [ATCAH_CONFIG_TO_SIGN_INTERNAL](#) (DEFAULT_ENABLED)
- #define [ATCAH_SIGN_INTERNAL_MSG](#) (DEFAULT_ENABLED)
- #define [ATCAH_ENCODE_COUNTER_MATCH](#) (DEFAULT_ENABLED)
- #define [ATCAH_GEN_SESSION_KEY](#) (DEFAULT_ENABLED)
- #define [ATCAH_DELETE_MAC](#) (CALIB_DELETE_EN)
- #define [ATCAC_SW_SHA2_256](#) (DEFAULT_ENABLED)

24.168.1 Detailed Description

Consistency checks for configuration options.

Copyright

(c) 2015-2021 Microchip Technology Inc. and its subsidiaries.

24.168.2 Macro Definition Documentation

24.168.2.1 ATCAH_CHECK_MAC

```
#define ATCAH_CHECK_MAC (DEFAULT_ENABLED)
```

Requires: ATCAH_CHECK_MAC ATCAC_SW_SHA2_256

Supported API's: atcah_check_mac

Enable ATCAH_CHECK_MAC to perform the checkmac operation to generate client response on the host side

24.168.2.2 ATCAH_CONFIG_TO_SIGN_INTERNAL

```
#define ATCAH_CONFIG_TO_SIGN_INTERNAL (DEFAULT_ENABLED)
```

Requires: ATCAH_CONFIG_TO_SIGN_INTERNAL

Supported API's: atcah_config_to_sign_internal

Enable ATCAH_CONFIG_TO_SIGN_INTERNAL to populate the slot_config, key_config, and is_slot_locked fields in the [atca_sign_internal_in_out](#) structure from the provided config zone

24.168.2.3 ATCAH_DECRYPT

```
#define ATCAH_DECRYPT (DEFAULT_ENABLED)
```

Requires: ATCAH_DECRYPT

Supported API's: atcah_decrypt

Enable ATCAH_DECRYPT to decrypt 32-byte encrypted data received with the Read command

24.168.2.4 ATCAH_DELETE_MAC

```
#define ATCAH_DELETE_MAC (CALIB_DELETE_EN)
```

Requires: ATCAH_DELETE_MAC ATCAC_SW_SHA2_256

Supported API's: atcah_delete_mac

Enable ATCAH_DELETE_MAC to calculate the mac

24.168.2.5 ATCAH_DERIVE_KEY

```
#define ATCAH_DERIVE_KEY (DEFAULT_ENABLED)
```

Requires: ATCAH_DERIVE_KEY ATCAC_SW_SHA2_256

Supported API's: atcah_derive_key

Enable ATCAH_DERIVE_KEY to derive a key with a key and TempKey

24.168.2.6 ATCAH_DERIVE_KEY_MAC

```
#define ATCAH_DERIVE_KEY_MAC (DEFAULT_ENABLED)
```

Requires: ATCAH_DERIVE_KEY_MAC ATCAC_SW_SHA2_256

Supported API's: atcah_derive_key_mac

Enable ATCAH_DERIVE_KEY_MAC to calculate the input MAC for a DeriveKey command

24.168.2.7 ATCAH_ENCODE_COUNTER_MATCH

#define ATCAH_ENCODE_COUNTER_MATCH (DEFAULT_ENABLED)

Requires: ATCAH_ENCODE_COUNTER_MATCH

Supported API's: atcah_encode_counter_match

Enable ATCAH_ENCODE_COUNTER_MATCH to build the counter match value that needs to be stored in a slot

24.168.2.8 ATCAH_GEN_KEY_MSG

#define ATCAH_GEN_KEY_MSG (DEFAULT_ENABLED)

Requires: ATCAH_SHA256 ATCAC_SW_SHA2_256

Supported API's: atcah_gen_key_msg

Enable ATCAH_GEN_KEY_MSG to calculate the PubKey digest created by GenKey and saved to TempKey

24.168.2.9 ATCAH_GEN_MAC

#define ATCAH_GEN_MAC (DEFAULT_ENABLED)

Requires: ATCAH_GEN_MAC ATCAC_SW_SHA2_256

Supported API's: atcah_gen_mac

Enable ATCAH_GEN_MAC to generate mac with session key with a plain text

24.168.2.10 ATCAH_GEN_OUTPUT_RESP_MAC

#define ATCAH_GEN_OUTPUT_RESP_MAC (DEFAULT_ENABLED)

Requires: ATCAH_GEN_OUTPUT_RESP_MAC ATCAC_SW_SHA2_256

Supported API's: atcah_gen_output_resp_mac

Enable ATCAH_GEN_OUTPUT_RESP_MAC to generate output response mac

24.168.2.11 ATCAH_GEN_SESSION_KEY

#define ATCAH_GEN_SESSION_KEY (DEFAULT_ENABLED)

Requires: ATCAH_GEN_SESSION_KEY ATCAC_SW_SHA2_256

Supported API's: atcah_gen_session_key

Enable ATCAH_GEN_SESSION_KEY to calculate the session key for the ECC204

24.168.2.12 ATCAH_GENDIG

```
#define ATCAH_GENDIG (DEFAULT_ENABLED)
```

Requires: ATCAH_GENDIG ATCAC_SW_SHA2_256

Supported API's: atcah_gen_dig

Enable ATCAH_GENDIG to combine the current TempKey with a stored value

24.168.2.13 ATCAH_GENDIVKEY

```
#define ATCAH_GENDIVKEY (DEFAULT_ENABLED)
```

Requires: ATCAH_GENDIVKEY ATCAC_SW_SHA2_256

Supported API's: atcah_gendivkey

Enable ATCAH_GENDIVKEY to generate the diversified key

24.168.2.14 ATCAH_HMAC

```
#define ATCAH_HMAC (DEFAULT_ENABLED)
```

Requires: ATCAH_HMAC ATCAC_SW_SHA2_256 ATCAH_INCLUDE_DATA

Supported API's: atcah_hmac

Enable ATCAH_HMAC to generate an HMAC / SHA-256 hash of a key and other information

24.168.2.15 ATCAH_INCLUDE_DATA

```
#define ATCAH_INCLUDE_DATA (DEFAULT_ENABLED)
```

Requires: ATCAH_INCLUDE_DATA

Supported API's: atcah_include_data

Enable ATCAH_INCLUDE_DATA to copy otp and sn data into a command buffer

24.168.2.16 ATCAH_IO_DECRYPT

```
#define ATCAH_IO_DECRYPT (DEFAULT_ENABLED)
```

Requires: ATCAH_IO_DECRYPT ATCAC_SW_SHA2_256

Supported API's: atcah_io_decrypt

Enable ATCAH_IO_DECRYPT to decrypt data that's been encrypted by the IO protection key. The ECDH and KDF commands on the ATECC608 are the only ones that support this operation

24.168.2.17 ATCAH_MAC

```
#define ATCAH_MAC (DEFAULT_ENABLED)
```

Requires: ATCAH_MAC ATCAC_SW_SHA2_256 ATCAH_INCLUDE_DATA

Supported API's: atcah_mac

Enable ATCAH_MAC to generate an SHA-256 digest (MAC) of a key, challenge, and other information

24.168.2.18 ATCAH_NONCE

```
#define ATCAH_NONCE (DEFAULT_ENABLED)
```

Requires: ATCAH_NONCE ATCAC_SW_SHA2_256

Supported API's: atcah_nonce

Enable ATCAH_NONCE to calculate host side nonce with the parameters passed

24.168.2.19 ATCAH_PRIVWRITE_AUTH_MAC

```
#define ATCAH_PRIVWRITE_AUTH_MAC (DEFAULT_ENABLED)
```

Requires: ATCAH_PRIVWRITE_AUTH_MAC ATCAC_SW_SHA2_256

Supported API's: atcah_privwrite_auth_mac

Enable ATCAH_PRIVWRITE_AUTH_MAC to calculate the input MAC for the PrivWrite command

24.168.2.20 ATCAH_SECUREBOOT_ENC

```
#define ATCAH_SECUREBOOT_ENC (DEFAULT_ENABLED)
```

Requires: ATCAH_SECUREBOOT_ENC ATCAC_SW_SHA2_256

Supported API's: atcah_secureboot_enc

Enable ATCAH_SECUREBOOT_ENC to encrypt the digest for the SecureBoot command when using the encrypted digest / validating mac option

24.168.2.21 ATCAH_SECUREBOOT_MAC

```
#define ATCAH_SECUREBOOT_MAC (DEFAULT_ENABLED)
```

Requires: ATCAH_SECUREBOOT_MAC ATCAC_SW_SHA2_256

Supported API's: atcah_secureboot_mac

Enable ATCAH_SECUREBOOT_MAC to calculates the expected MAC returned from the SecureBoot command when verification is a success

24.168.2.22 ATCAH_SHA256

```
#define ATCAH_SHA256 (DEFAULT_ENABLED)
```

Requires: ATCAH_SHA256 ATCAC_SW_SHA2_256

Supported API's: atcah_sha256

Enable ATCAH_SHA256 to create a SHA256 digest on a little-endian system

24.168.2.23 ATCAH_SIGN_INTERNAL_MSG

```
#define ATCAH_SIGN_INTERNAL_MSG (DEFAULT_ENABLED)
```

Requires: ATCAH_SIGN_INTERNAL_MSG ATCAC_SW_SHA2_256

Supported API's: atcah_sign_internal_msg

Enable ATCAH_SIGN_INTERNAL_MSG to build the full message that would be signed by the Sign(Internal) command

24.168.2.24 ATCAH_VERIFY_MAC

```
#define ATCAH_VERIFY_MAC (DEFAULT_ENABLED)
```

Requires: ATCAH_VERIFY_MAC ATCAC_SW_SHA2_256

Supported API's: atcah_verify_mac

Enable ATCAH_VERIFY_MAC to calculate the expected MAC on the host side for the Verify command

24.168.2.25 ATCAH_WRITE_AUTH_MAC

```
#define ATCAH_WRITE_AUTH_MAC (DEFAULT_ENABLED)
```

Requires: ATCAH_WRITE_AUTH_MAC ATCAC_SW_SHA2_256

Supported API's: atcah_write_auth_mac ECC204 specific API's: atcah_ecc204_write_auth_mac

Enable ATCAH_WRITE_AUTH_MAC to calculate the input MAC for the Write command

24.169 atca_jwt.c File Reference

Utilities to create and verify a JSON Web Token (JWT)

```
#include "cryptoauthlib.h"
#include "atca_helpers.h"
#include "crypto/atca_crypto_sw_sha2.h"
#include "jwt/atca_jwt.h"
#include <stdio.h>
```

24.169.1 Detailed Description

Utilities to create and verify a JSON Web Token (JWT)

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.170 atca_jwt.h File Reference

Utilities to create and verify a JSON Web Token (JWT)

```
#include "cryptoauthlib.h"
```

24.170.1 Detailed Description

Utilities to create and verify a JSON Web Token (JWT)

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.171 atca_mbedtls_interface.h File Reference

Configuration Check for MbedTLS Integration Support.

```
#include "atca_config_check.h"  
#include "mbedtls/mbedtls_config.h"
```

Data Structures

- struct [atcac_x509_ctx](#)

Macros

- #define [ATCAC_SHA1_EN](#) (DEFAULT_DISABLED)
- #define [ATCAC_SHA256_EN](#) (DEFAULT_DISABLED)
- #define [ATCAC_SHA384_EN](#) (DEFAULT_DISABLED)
- #define [ATCAC_SHA512_EN](#) (DEFAULT_DISABLED)
- #define [ATCAC_AES_CMAC_EN](#) (DEFAULT_DISABLED)
- #define [ATCAC_AES_GCM_EN](#) (DEFAULT_DISABLED)
- #define [ATCAC_PKEY_EN](#) (DEFAULT_ENABLED)
- #define [HOSTLIB_CERT_EN](#) (DEFAULT_ENABLED)

Typedefs

- typedef struct [atcac_x509_ctx](#) **atcac_x509_ctx_t**

24.171.1 Detailed Description

Configuration Check for MbedTLS Integration Support.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.171.2 Macro Definition Documentation

24.171.2.1 ATCAC_AES_CMAC_EN

```
#define ATCAC_AES_CMAC_EN (DEFAULT_DISABLED)
```

Indicates if this module is a provider of an AES-CMAC implementation

24.171.2.2 ATCAC_AES_GCM_EN

```
#define ATCAC_AES_GCM_EN (DEFAULT_DISABLED)
```

Indicates if this module is a provider of an AES-GCM implementation

24.171.2.3 ATCAC_PKEY_EN

```
#define ATCAC_PKEY_EN (DEFAULT_ENABLED)
```

Indicates if this module is a provider of a generic asymmetric cryptography implementation

24.171.2.4 ATCAC_SHA1_EN

```
#define ATCAC_SHA1_EN (DEFAULT_DISABLED)
```

Indicates if this module is a provider of a SHA1 implementation

24.171.2.5 ATCAC_SHA256_EN

```
#define ATCAC_SHA256_EN (DEFAULT_DISABLED)
```

Indicates if this module is a provider of a SHA256 implementation

24.171.2.6 ATCAC_SHA384_EN

```
#define ATCAC_SHA384_EN (DEFAULT_DISABLED)
```

Indicates if this module is a provider of a SHA384 implementation

Disabled by default. Use FEATURE_ENABLED to use SHA384

24.171.2.7 ATCAC_SHA512_EN

```
#define ATCAC_SHA512_EN (DEFAULT_DISABLED)
```

Indicates if this module is a provider of a SHA512 implementation

Disabled by default. Use FEATURE_ENABLED to use SHA512

24.171.2.8 HOSTLIB_CERT_EN

```
#define HOSTLIB_CERT_EN (DEFAULT_ENABLED)
```

Indicates if this module is a provider of x509 certificate handling

24.172 atca_mbedtls_wrap.c File Reference

Wrapper functions to replace cryptoauthlib software crypto functions with the mbedtls equivalent.

```
#include "atca_config_check.h"
#include "mbedtls/mbedtls_config.h"
#include <stdlib.h>
#include "mbedtls/cmac.h"
#include "mbedtls/ctr_drbg.h"
#include "mbedtls/pk.h"
#include "mbedtls/ecdh.h"
#include "mbedtls/ecp.h"
#include "mbedtls/entropy.h"
#include "mbedtls/x509_crt.h"
#include "mbedtls/oid.h"
#include "cryptoauthlib.h"
#include "atca_mbedtls_wrap.h"
#include "atca_mbedtls_patch.h"
#include "crypto/atca_crypto_sw.h"
#include "atcacert/atcacert_client.h"
#include "atcacert/atcacert_def.h"
#include "mbedtls/library/pk_wrap.h"
#include "atcacert/atcacert_der.h"
```

Macros

- `#define mbedtls_calloc` calloc
- `#define mbedtls_free` free

Functions

- ATCA_STATUS [atcac_sw_random](#) (uint8_t *data, size_t data_size)
Return Random Bytes.
- ATCA_STATUS [atcac_aes_gcm_aad_update](#) (struct [atcac_aes_gcm_ctx](#) *ctx, const uint8_t *aad, const size_t aad_len)
Update the GCM context with additional authentication data (AAD)
- ATCA_STATUS [atcac_aes_gcm_encrypt_start](#) (struct [atcac_aes_gcm_ctx](#) *ctx, const uint8_t *key, const uint8_t key_len, const uint8_t *iv, const uint8_t iv_len)
Initialize an AES-GCM context.
- ATCA_STATUS [atcac_aes_gcm_encrypt_update](#) (struct [atcac_aes_gcm_ctx](#) *ctx, const uint8_t *plaintext, const size_t pt_len, uint8_t *ciphertext, size_t *ct_len)
Encrypt a data using the initialized context.
- ATCA_STATUS [atcac_aes_gcm_encrypt_finish](#) (struct [atcac_aes_gcm_ctx](#) *ctx, uint8_t *tag, size_t tag_len)
Get the AES-GCM tag and free the context.
- ATCA_STATUS [atcac_aes_gcm_decrypt_start](#) (struct [atcac_aes_gcm_ctx](#) *ctx, const uint8_t *key, const uint8_t key_len, const uint8_t *iv, const uint8_t iv_len)
Initialize an AES-GCM context for decryption.
- ATCA_STATUS [atcac_aes_gcm_decrypt_update](#) (struct [atcac_aes_gcm_ctx](#) *ctx, const uint8_t *ciphertext, const size_t ct_len, uint8_t *plaintext, size_t *pt_len)
Decrypt ciphertext using the initialized context.
- ATCA_STATUS [atcac_aes_gcm_decrypt_finish](#) (struct [atcac_aes_gcm_ctx](#) *ctx, const uint8_t *tag, size_t tag_len, bool *is_verified)
Compare the AES-GCM tag and free the context.
- ATCA_STATUS [atcac_sw_sha1_init](#) (struct [atcac_sha1_ctx](#) *ctx)
Initialize context for performing SHA1 hash in software.
- ATCA_STATUS [atcac_sw_sha1_update](#) (struct [atcac_sha1_ctx](#) *ctx, const uint8_t *data, size_t data_size)
Add data to a SHA1 hash.
- ATCA_STATUS [atcac_sw_sha1_finish](#) (struct [atcac_sha1_ctx](#) *ctx, uint8_t digest[ATCA_SHA1_DIGEST_SIZE])
Complete the SHA1 hash in software and return the digest.
- ATCA_STATUS [atcac_aes_cmac_init](#) (struct [atcac_aes_cmac_ctx](#) *ctx, const uint8_t *key, const uint8_t key_len)
Initialize context for performing CMAC in software.
- ATCA_STATUS [atcac_aes_cmac_update](#) (struct [atcac_aes_cmac_ctx](#) *ctx, const uint8_t *data, const size_t data_size)
Update CMAC context with input data.
- ATCA_STATUS [atcac_aes_cmac_finish](#) (struct [atcac_aes_cmac_ctx](#) *ctx, uint8_t *cmac, size_t *cmac_size)
Finish CMAC calculation and clear the CMAC context.
- ATCA_STATUS [atcac_sha256_hmac_init](#) (struct [atcac_hmac_ctx](#) *ctx, struct [atcac_sha256_ctx](#) *sha256_ctx, const uint8_t *key, const uint8_t key_len)
Initialize context for performing HMAC (sha256) in software.
- ATCA_STATUS [atcac_sha256_hmac_update](#) (struct [atcac_hmac_ctx](#) *ctx, const uint8_t *data, size_t data_size)
Update HMAC context with input data.
- ATCA_STATUS [atcac_sha256_hmac_finish](#) (struct [atcac_hmac_ctx](#) *ctx, uint8_t *digest, size_t *digest_len)
Finish CMAC calculation and clear the HMAC context.
- ATCA_STATUS [atcac_pk_init](#) (struct [atcac_pk_ctx](#) *ctx, const uint8_t *buf, size_t buflen, uint8_t key_type, bool pubkey)
Set up a public/private key structure for use in asymmetric cryptographic functions.
- ATCA_STATUS [atcac_pk_init_pem](#) (struct [atcac_pk_ctx](#) *ctx, const uint8_t *buf, size_t buflen, bool pubkey)
Set up a public/private key structure for use in asymmetric cryptographic functions.

- ATCA_STATUS [atcac_pk_free](#) (struct [atcac_pk_ctx](#) *ctx)
Free a public/private key structure.
- ATCA_STATUS [atcac_pk_public](#) (struct [atcac_pk_ctx](#) *ctx, uint8_t *buf, size_t *buflen)
Get the public key from the context.
- int [atca_mbedtls_random_ctx](#) (void *ctx, unsigned char *data, size_t data_size)
mbedtls f_rng callback interface to fill a buffer with random bytes
- ATCA_STATUS [atcac_pk_sign](#) (struct [atcac_pk_ctx](#) *ctx, const uint8_t *digest, size_t dig_len, uint8_t *signature, size_t sig_buf_size, size_t *sig_len)
Perform a signature with the private key in the context.
- ATCA_STATUS [atcac_pk_verify](#) (struct [atcac_pk_ctx](#) *ctx, const uint8_t *digest, size_t dig_len, const uint8_t *signature, size_t sig_len)
Perform a verify using the public key in the provided context.
- ATCA_STATUS [atcac_pk_derive](#) (struct [atcac_pk_ctx](#) *private_ctx, struct [atcac_pk_ctx](#) *public_ctx, uint8_t *buf, size_t *buflen)
Execute the key agreement protocol for the provided keys (if they can)
- int [atca_mbedtls_pk_init_ext](#) (ATCADevice device, mbedtls_pk_context *pkey, const uint16_t slotid)
Initializes an mbedtls pk context for use with EC operations.
- int [atca_mbedtls_pk_init](#) (mbedtls_pk_context *pkey, const uint16_t slotid)
Initializes an mbedtls pk context for use with EC operations.
- ATCA_STATUS [atcac_parse_der](#) (struct [atcac_x509_ctx](#) **cert, [cal_buffer](#) *der)
- ATCA_STATUS [atcac_get_subject](#) (const struct [atcac_x509_ctx](#) *cert, [cal_buffer](#) *cert_subject)
- ATCA_STATUS [atcac_get_subj_public_key](#) (const struct [atcac_x509_ctx](#) *cert, [cal_buffer](#) *subj_public_key)
- ATCA_STATUS [atcac_get_subj_key_id](#) (const struct [atcac_x509_ctx](#) *cert, [cal_buffer](#) *subj_public_key_id)
- ATCA_STATUS [atcac_get_issue_date](#) (const struct [atcac_x509_ctx](#) *cert, [cal_buffer](#) *not_before, uint8_t *fmt)
- ATCA_STATUS [atcac_get_expire_date](#) (const struct [atcac_x509_ctx](#) *cert, [cal_buffer](#) *not_after, uint8_t *fmt)
- ATCA_STATUS [atcac_get_issuer](#) (const struct [atcac_x509_ctx](#) *cert, [cal_buffer](#) *issuer_buf)
- ATCA_STATUS [atcac_get_cert_sn](#) (const struct [atcac_x509_ctx](#) *cert, [cal_buffer](#) *cert_sn)
- ATCA_STATUS [atcac_get_auth_key_id](#) (const struct [atcac_x509_ctx](#) *cert, [cal_buffer](#) *auth_key_id)
- void [atcac_x509_free](#) (void *cert)

Variables

- const mbedtls_pk_info_t [atca_mbedtls_eckey_info](#)

24.172.1 Detailed Description

Wrapper functions to replace cryptoauthlib software crypto functions with the mbedtls equivalent.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.172.2 Function Documentation

24.172.2.1 atcac_aes_cmac_finish()

```
ATCA_STATUS atcac_aes_cmac_finish (
    struct atcac_aes_cmac_ctx * ctx,
    uint8_t * cmac,
    size_t * cmac_size )
```

Finish CMAC calculation and clear the CMAC context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a aes-cmac context
out	<i>cmac</i>	cmac value
in, out	<i>cmac_size</i>	length of cmac

24.172.2.2 atcac_aes_cmac_init()

```
ATCA_STATUS atcac_aes_cmac_init (
    struct atcac_aes_cmac_ctx * ctx,
    const uint8_t * key,
    const uint8_t key_len )
```

Initialize context for performing CMAC in software.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a aes-cmac context
in	<i>key</i>	key value to use
in	<i>key_len</i>	length of the key

24.172.2.3 atcac_aes_cmac_update()

```
ATCA_STATUS atcac_aes_cmac_update (
    struct atcac_aes_cmac_ctx * ctx,
    const uint8_t * data,
    const size_t data_size )
```

Update CMAC context with input data.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a aes-cmac context
in	<i>data</i>	input data
in	<i>data_size</i>	length of input data

24.172.2.4 atcac_aes_gcm_aad_update()

```
ATCA_STATUS atcac_aes_gcm_aad_update (
    struct atcac_aes_gcm_ctx * ctx,
    const uint8_t * aad,
    const size_t aad_len )
```

Update the GCM context with additional authentication data (AAD)

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	AES-GCM Context
in	<i>aad</i>	Additional Authentication Data
in	<i>aad_len</i>	Length of AAD

24.172.2.5 atcac_aes_gcm_decrypt_finish()

```
ATCA_STATUS atcac_aes_gcm_decrypt_finish (
    struct atcac_aes_gcm_ctx * ctx,
    const uint8_t * tag,
    size_t tag_len,
    bool * is_verified )
```

Compare the AES-GCM tag and free the context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	AES-GCM Context
in	<i>tag</i>	GCM Tag to Verify
in	<i>tag_len</i>	Length of the GCM tag
out	<i>is_verified</i>	Tag verified as matching

24.172.2.6 atcac_aes_gcm_decrypt_start()

```
ATCA_STATUS atcac_aes_gcm_decrypt_start (
    struct atcac_aes_gcm_ctx * ctx,
    const uint8_t * key,
    const uint8_t key_len,
    const uint8_t * iv,
    const uint8_t iv_len )
```

Initialize an AES-GCM context for decryption.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	AES-GCM Context
in	<i>key</i>	AES Key
in	<i>key_len</i>	Length of the AES key - should be 16 or 32
in	<i>iv</i>	Initialization vector input
in	<i>iv_len</i>	Length of the initialization vector

24.172.2.7 atcac_aes_gcm_decrypt_update()

```
ATCA_STATUS atcac_aes_gcm_decrypt_update (
    struct atcac_aes_gcm_ctx * ctx,
    const uint8_t * ciphertext,
    const size_t ct_len,
    uint8_t * plaintext,
    size_t * pt_len )
```

Decrypt ciphertext using the initialized context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	AES-GCM Context
in	<i>ciphertext</i>	Ciphertext to decrypt
in	<i>ct_len</i>	Length of the ciphertext
out	<i>plaintext</i>	Resulting decrypted plaintext
in, out	<i>pt_len</i>	Length of the plaintext buffer

24.172.2.8 atcac_aes_gcm_encrypt_finish()

```
ATCA_STATUS atcac_aes_gcm_encrypt_finish (
    struct atcac_aes_gcm_ctx * ctx,
    uint8_t * tag,
    size_t tag_len )
```

Get the AES-GCM tag and free the context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	AES-GCM Context
out	<i>tag</i>	GCM Tag Result
in	<i>tag_len</i>	Length of the GCM tag

24.172.2.9 atcac_aes_gcm_encrypt_start()

```
ATCA_STATUS atcac_aes_gcm_encrypt_start (
    struct atcac_aes_gcm_ctx * ctx,
    const uint8_t * key,
    const uint8_t key_len,
    const uint8_t * iv,
    const uint8_t iv_len )
```

Initialize an AES-GCM context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	AES-GCM Context
in	<i>key</i>	AES Key
in	<i>key_len</i>	Length of the AES key - should be 16 or 32
in	<i>iv</i>	Initialization vector input
in	<i>iv_len</i>	Length of the initialization vector

24.172.2.10 atcac_aes_gcm_encrypt_update()

```
ATCA_STATUS atcac_aes_gcm_encrypt_update (
    struct atcac_aes_gcm_ctx * ctx,
    const uint8_t * plaintext,
    const size_t pt_len,
    uint8_t * ciphertext,
    size_t * ct_len )
```

Encrypt a data using the initialized context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	AES-GCM Context
in	<i>plaintext</i>	Input buffer to encrypt
in	<i>pt_len</i>	Length of the input
out	<i>ciphertext</i>	Output buffer
in, out	<i>ct_len</i>	Length of the ciphertext buffer

24.172.2.11 atcac_pk_derive()

```
ATCA_STATUS atcac_pk_derive (
    struct atcac_pk_ctx * private_ctx,
    struct atcac_pk_ctx * public_ctx,
    uint8_t * buf,
    size_t * buflen )
```

Execute the key agreement protocol for the provided keys (if they can)

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.172.2.12 atcac_pk_free()

```
ATCA_STATUS atcac_pk_free (
    struct atcac_pk_ctx * ctx )
```

Free a public/private key structure.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a pk context
----	------------	-------------------------

24.172.2.13 atcac_pk_init()

```
ATCA_STATUS atcac_pk_init (
    struct atcac\_pk\_ctx * ctx,
    const uint8_t * buf,
    size_t buflen,
    uint8_t key_type,
    bool pubkey )
```

Set up a public/private key structure for use in asymmetric cryptographic functions.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a pk context
in	<i>buf</i>	buffer containing a pem encoded key
in	<i>buflen</i>	length of the input buffer
in	<i>pubkey</i>	buffer is a public key

24.172.2.14 atcac_pk_init_pem()

```
ATCA_STATUS atcac_pk_init_pem (
    struct atcac\_pk\_ctx * ctx,
    const uint8_t * buf,
    size_t buflen,
    bool pubkey )
```

Set up a public/private key structure for use in asymmetric cryptographic functions.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a pk context
in	<i>buf</i>	buffer containing a pem encoded key
in	<i>buflen</i>	length of the input buffer
in	<i>pubkey</i>	buffer is a public key

24.172.2.15 atcac_pk_public()

```
ATCA_STATUS atcac_pk_public (
    struct atcac_pk_ctx * ctx,
    uint8_t * buf,
    size_t * buflen )
```

Get the public key from the context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.172.2.16 atcac_pk_sign()

```
ATCA_STATUS atcac_pk_sign (
    struct atcac_pk_ctx * ctx,
    const uint8_t * digest,
    size_t dig_len,
    uint8_t * signature,
    size_t sig_buf_size,
    size_t * sig_len )
```

Perform a signature with the private key in the context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.172.2.17 atcac_pk_verify()

```
ATCA_STATUS atcac_pk_verify (
    struct atcac_pk_ctx * ctx,
    const uint8_t * digest,
    size_t dig_len,
    const uint8_t * signature,
    size_t sig_len )
```

Perform a verify using the public key in the provided context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.172.2.18 atcac_sha256_hmac_finish()

```
ATCA_STATUS atcac_sha256_hmac_finish (
    struct atcac_hmac_ctx * ctx,
    uint8_t * digest,
    size_t * digest_len )
```

Finish CMAC calculation and clear the HMAC context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a sha256-hmac context
out	<i>digest</i>	hmac value
in, out	<i>digest_len</i>	length of hmac

24.172.2.19 atcac_sha256_hmac_init()

```
ATCA_STATUS atcac_sha256_hmac_init (
    struct atcac_hmac_ctx * ctx,
    struct atcac_sha2_256_ctx * sha256_ctx,
    const uint8_t * key,
    const uint8_t key_len )
```

Initialize context for performing HMAC (sha256) in software.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a sha256-hmac context
in	<i>sha256_ctx</i>	pointer to a sha256 context
in	<i>key</i>	key value to use
in	<i>key_len</i>	length of the key

24.172.2.20 atcac_sha256_hmac_update()

```
ATCA_STATUS atcac_sha256_hmac_update (
    struct atcac_hmac_ctx * ctx,
```

```
const uint8_t * data,  
size_t data_size )
```

Update HMAC context with input data.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a sha256-hmac context
in	<i>data</i>	input data
in	<i>data_size</i>	length of input data

24.172.2.21 atcac_sw_random()

```
ATCA_STATUS atcac_sw_random (  
    uint8_t * data,  
    size_t data_size )
```

Return Random Bytes.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.172.2.22 atcac_sw_sha1_finish()

```
ATCA_STATUS atcac_sw_sha1_finish (  
    struct atcac_sha1_ctx * ctx,  
    uint8_t digest[ATCA_SHA1_DIGEST_SIZE] )
```

Complete the SHA1 hash in software and return the digest.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a hash context
out	<i>digest</i>	output buffer (20 bytes)

24.172.2.23 atcac_sw_sha1_init()

```
ATCA_STATUS atcac_sw_sha1_init (
    struct atcac_sha1_ctx * ctx )
```

Initialize context for performing SHA1 hash in software.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a hash context
----	------------	---------------------------

24.172.2.24 atcac_sw_sha1_update()

```
ATCA_STATUS atcac_sw_sha1_update (
    struct atcac_sha1_ctx * ctx,
    const uint8_t * data,
    size_t data_size )
```

Add data to a SHA1 hash.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a hash context
in	<i>data</i>	input data buffer
in	<i>data_size</i>	input data length

24.173 atca_openssl_interface.c File Reference

Crypto abstraction functions for external host side cryptography.

```
#include "cryptoauthlib.h"
#include "crypto/atca_crypto_sw.h"
#include <openssl/bn.h>
#include <openssl/bio.h>
#include <openssl/cmac.h>
```

```
#include <openssl/ec.h>
#include <openssl/evp.h>
#include <openssl/hmac.h>
#include <openssl/pem.h>
#include <openssl/rand.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>
```

Data Structures

- struct [atca_evp_ctx](#)

Functions

- ATCA_STATUS [atcac_sw_random](#) (uint8_t *data, size_t data_size)
Return Random Bytes.
- ATCA_STATUS [atcac_aes_gcm_aad_update](#) (struct [atcac_aes_gcm_ctx](#) *ctx, const uint8_t *aad, const size_t aad_len)
Update the GCM context with additional authentication data (AAD)
- ATCA_STATUS [atcac_aes_gcm_encrypt_start](#) (struct [atcac_aes_gcm_ctx](#) *ctx, const uint8_t *key, const uint8_t key_len, const uint8_t *iv, const uint8_t iv_len)
Initialize an AES-GCM context.
- ATCA_STATUS [atcac_aes_gcm_encrypt_update](#) (struct [atcac_aes_gcm_ctx](#) *ctx, const uint8_t *plaintext, const size_t pt_len, uint8_t *ciphertext, size_t *ct_len)
Encrypt a data using the initialized context.
- ATCA_STATUS [atcac_aes_gcm_encrypt_finish](#) (struct [atcac_aes_gcm_ctx](#) *ctx, uint8_t *tag, size_t tag_len)
Get the AES-GCM tag and free the context.
- ATCA_STATUS [atcac_aes_gcm_decrypt_start](#) (struct [atcac_aes_gcm_ctx](#) *ctx, const uint8_t *key, const uint8_t key_len, const uint8_t *iv, const uint8_t iv_len)
Initialize an AES-GCM context for decryption.
- ATCA_STATUS [atcac_aes_gcm_decrypt_update](#) (struct [atcac_aes_gcm_ctx](#) *ctx, const uint8_t *ciphertext, const size_t ct_len, uint8_t *plaintext, size_t *pt_len)
Decrypt ciphertext using the initialized context.
- ATCA_STATUS [atcac_aes_gcm_decrypt_finish](#) (struct [atcac_aes_gcm_ctx](#) *ctx, const uint8_t *tag, size_t tag_len, bool *is_verified)
Compare the AES-GCM tag and free the context.
- ATCA_STATUS [atcac_sw_sha1_init](#) (struct [atcac_sha1_ctx](#) *ctx)
Initialize context for performing SHA1 hash in software.
- ATCA_STATUS [atcac_sw_sha1_update](#) (struct [atcac_sha1_ctx](#) *ctx, const uint8_t *data, size_t data_size)
Add data to a SHA1 hash.
- ATCA_STATUS [atcac_sw_sha1_finish](#) (struct [atcac_sha1_ctx](#) *ctx, uint8_t digest[ATCA_SHA1_DIGEST_SIZE])
Complete the SHA1 hash in software and return the digest.
- ATCA_STATUS [atcac_aes_cmac_init](#) (struct [atcac_aes_cmac_ctx](#) *ctx, const uint8_t *key, const uint8_t key_len)
Initialize context for performing CMAC in software.
- ATCA_STATUS [atcac_aes_cmac_update](#) (struct [atcac_aes_cmac_ctx](#) *ctx, const uint8_t *data, const size_t data_size)
Update CMAC context with input data.
- ATCA_STATUS [atcac_aes_cmac_finish](#) (struct [atcac_aes_cmac_ctx](#) *ctx, uint8_t *cmac, size_t *cmac_size)

Finish CMAC calculation and clear the CMAC context.

- ATCA_STATUS [atcac_sha256_hmac_init](#) (struct [atcac_hmac_ctx](#) *ctx, struct [atcac_sha256_ctx](#) *sha256_ctx, const uint8_t *key, const uint8_t key_len)

Initialize context for performing HMAC (sha256) in software.

- ATCA_STATUS [atcac_sha256_hmac_update](#) (struct [atcac_hmac_ctx](#) *ctx, const uint8_t *data, size_t data_size)

Update HMAC context with input data.

- ATCA_STATUS [atcac_sha256_hmac_finish](#) (struct [atcac_hmac_ctx](#) *ctx, uint8_t *digest, size_t *digest_len)

Finish CMAC calculation and clear the HMAC context.

- ATCA_STATUS [atcac_pk_init](#) (struct [atcac_pk_ctx](#) *ctx, const uint8_t *buf, size_t buflen, uint8_t key_type, bool pubkey)

Set up a public/private key structure for use in asymmetric cryptographic functions.

- ATCA_STATUS [atcac_pk_init_pem](#) (struct [atcac_pk_ctx](#) *ctx, const uint8_t *buf, size_t buflen, bool pubkey)

Set up a public/private key structure for use in asymmetric cryptographic functions.

- ATCA_STATUS [atcac_pk_free](#) (struct [atcac_pk_ctx](#) *ctx)

Free a public/private key structure.

- ATCA_STATUS [atcac_pk_public](#) (struct [atcac_pk_ctx](#) *ctx, uint8_t *buf, size_t *buflen)

Get the public key from the context.

- ATCA_STATUS [atcac_pk_sign](#) (struct [atcac_pk_ctx](#) *ctx, const uint8_t *digest, size_t dig_len, uint8_t *signature, size_t sig_buf_size, size_t *sig_len)

Perform a signature with the private key in the context.

- ATCA_STATUS [atcac_pk_verify](#) (struct [atcac_pk_ctx](#) *ctx, const uint8_t *digest, size_t dig_len, const uint8_t *signature, size_t sig_len)

Perform a verify using the public key in the provided context.

- ATCA_STATUS [atcac_pk_derive](#) (struct [atcac_pk_ctx](#) *private_ctx, struct [atcac_pk_ctx](#) *public_ctx, uint8_t *buf, size_t *buflen)

Execute the key agreement protocol for the provided keys (if they can)

- ATCA_STATUS [atcac_parse_der](#) (struct [atcac_x509_ctx](#) **cert, [cal_buffer](#) *der)
- ATCA_STATUS [atcac_get_subject](#) (const struct [atcac_x509_ctx](#) *cert, [cal_buffer](#) *cert_subject)
- ATCA_STATUS [atcac_get_subj_public_key](#) (const struct [atcac_x509_ctx](#) *cert, [cal_buffer](#) *subj_public_key)
- ATCA_STATUS [atcac_get_subj_key_id](#) (const struct [atcac_x509_ctx](#) *cert, [cal_buffer](#) *subj_public_key_id)
- ATCA_STATUS [atcac_get_issuer](#) (const struct [atcac_x509_ctx](#) *cert, [cal_buffer](#) *issuer_buf)
- ATCA_STATUS [atcac_get_auth_key_id](#) (const struct [atcac_x509_ctx](#) *cert, [cal_buffer](#) *auth_key_id)
- ATCA_STATUS [atcac_get_issue_date](#) (const struct [atcac_x509_ctx](#) *cert, [cal_buffer](#) *not_before, uint8_t *fmt)
- ATCA_STATUS [atcac_get_expire_date](#) (const struct [atcac_x509_ctx](#) *cert, [cal_buffer](#) *not_after, uint8_t *fmt)
- ATCA_STATUS [atcac_get_cert_sn](#) (const struct [atcac_x509_ctx](#) *cert, [cal_buffer](#) *cert_sn)
- void [atcac_x509_free](#) (void *cert)

24.173.1 Detailed Description

Crypto abstraction functions for external host side cryptography.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.173.2 Function Documentation

24.173.2.1 atcac_aes_cmac_finish()

```
ATCA_STATUS atcac_aes_cmac_finish (
    struct atcac_aes_cmac_ctx * ctx,
    uint8_t * cmac,
    size_t * cmac_size )
```

Finish CMAC calculation and clear the CMAC context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a aes-cmac context
out	<i>cmac</i>	cmac value
in, out	<i>cmac_size</i>	length of cmac

24.173.2.2 atcac_aes_cmac_init()

```
ATCA_STATUS atcac_aes_cmac_init (
    struct atcac_aes_cmac_ctx * ctx,
    const uint8_t * key,
    const uint8_t key_len )
```

Initialize context for performing CMAC in software.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a aes-cmac context
in	<i>key</i>	key value to use
in	<i>key_len</i>	length of the key

24.173.2.3 atcac_aes_cmac_update()

```
ATCA_STATUS atcac_aes_cmac_update (
    struct atcac_aes_cmac_ctx * ctx,
    const uint8_t * data,
    const size_t data_size )
```

Update CMAC context with input data.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a aes-cmac context
in	<i>data</i>	input data
in	<i>data_size</i>	length of input data

24.173.2.4 atcac_aes_gcm_aad_update()

```
ATCA_STATUS atcac_aes_gcm_aad_update (
    struct atcac_aes_gcm_ctx * ctx,
    const uint8_t * aad,
    const size_t aad_len )
```

Update the GCM context with additional authentication data (AAD)

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	AES-GCM Context
in	<i>aad</i>	Additional Authentication Data
in	<i>aad_len</i>	Length of AAD

24.173.2.5 atcac_aes_gcm_decrypt_finish()

```
ATCA_STATUS atcac_aes_gcm_decrypt_finish (
    struct atcac_aes_gcm_ctx * ctx,
    const uint8_t * tag,
    size_t tag_len,
    bool * is_verified )
```

Compare the AES-GCM tag and free the context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	AES-GCM Context
in	<i>tag</i>	GCM Tag to Verify
in	<i>tag_len</i>	Length of the GCM tag
out	<i>is_verified</i>	Tag verified as matching

24.173.2.6 atcac_aes_gcm_decrypt_start()

```

ATCA_STATUS atcac_aes_gcm_decrypt_start (
    struct atcac_aes_gcm_ctx * ctx,
    const uint8_t * key,
    const uint8_t key_len,
    const uint8_t * iv,
    const uint8_t iv_len )

```

Initialize an AES-GCM context for decryption.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	AES-GCM Context
in	<i>key</i>	AES Key
in	<i>key_len</i>	Length of the AES key - should be 16 or 32
in	<i>iv</i>	Initialization vector input
in	<i>iv_len</i>	Length of the initialization vector

24.173.2.7 atcac_aes_gcm_decrypt_update()

```

ATCA_STATUS atcac_aes_gcm_decrypt_update (
    struct atcac_aes_gcm_ctx * ctx,
    const uint8_t * ciphertext,
    const size_t ct_len,
    uint8_t * plaintext,
    size_t * pt_len )

```

Decrypt ciphertext using the initialized context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	AES-GCM Context
in	<i>ciphertext</i>	Ciphertext to decrypt
in	<i>ct_len</i>	Length of the ciphertext
out	<i>plaintext</i>	Resulting decrypted plaintext
in, out	<i>pt_len</i>	Length of the plaintext buffer

24.173.2.8 atcac_aes_gcm_encrypt_finish()

```
ATCA_STATUS atcac_aes_gcm_encrypt_finish (
    struct atcac_aes_gcm_ctx * ctx,
    uint8_t * tag,
    size_t tag_len )
```

Get the AES-GCM tag and free the context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	AES-GCM Context
out	<i>tag</i>	GCM Tag Result
in	<i>tag_len</i>	Length of the GCM tag

24.173.2.9 atcac_aes_gcm_encrypt_start()

```
ATCA_STATUS atcac_aes_gcm_encrypt_start (
    struct atcac_aes_gcm_ctx * ctx,
    const uint8_t * key,
    const uint8_t key_len,
    const uint8_t * iv,
    const uint8_t iv_len )
```

Initialize an AES-GCM context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	AES-GCM Context
in	<i>key</i>	AES Key
in	<i>key_len</i>	Length of the AES key - should be 16 or 32
in	<i>iv</i>	Initialization vector input
in	<i>iv_len</i>	Length of the initialization vector

24.173.2.10 atcac_aes_gcm_encrypt_update()

```
ATCA_STATUS atcac_aes_gcm_encrypt_update (
    struct atcac_aes_gcm_ctx * ctx,
    const uint8_t * plaintext,
    const size_t pt_len,
    uint8_t * ciphertext,
    size_t * ct_len )
```

Encrypt a data using the initialized context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	AES-GCM Context
in	<i>plaintext</i>	Input buffer to encrypt
in	<i>pt_len</i>	Length of the input
out	<i>ciphertext</i>	Output buffer
in, out	<i>ct_len</i>	Length of the ciphertext buffer

24.173.2.11 atcac_pk_derive()

```
ATCA_STATUS atcac_pk_derive (
    struct atcac_pk_ctx * private_ctx,
    struct atcac_pk_ctx * public_ctx,
    uint8_t * buf,
    size_t * buflen )
```

Execute the key agreement protocol for the provided keys (if they can)

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.173.2.12 atcac_pk_free()

```
ATCA_STATUS atcac_pk_free (
    struct atcac_pk_ctx * ctx )
```

Free a public/private key structure.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a pk context
----	------------	-------------------------

24.173.2.13 atcac_pk_init()

```
ATCA_STATUS atcac_pk_init (
    struct atcac_pk_ctx * ctx,
    const uint8_t * buf,
    size_t buflen,
    uint8_t key_type,
    bool pubkey )
```

Set up a public/private key structure for use in asymmetric cryptographic functions.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a pk context
in	<i>buf</i>	buffer containing a pem encoded key
in	<i>buflen</i>	length of the input buffer
in	<i>pubkey</i>	buffer is a public key

24.173.2.14 atcac_pk_init_pem()

```
ATCA_STATUS atcac_pk_init_pem (
    struct atcac_pk_ctx * ctx,
    const uint8_t * buf,
    size_t buflen,
    bool pubkey )
```

Set up a public/private key structure for use in asymmetric cryptographic functions.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a pk context
in	<i>buf</i>	buffer containing a pem encoded key
in	<i>buflen</i>	length of the input buffer
in	<i>pubkey</i>	buffer is a public key

24.173.2.15 atcac_pk_public()

```
ATCA_STATUS atcac_pk_public (
    struct atcac_pk_ctx * ctx,
    uint8_t * buf,
    size_t * buflen )
```

Get the public key from the context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.173.2.16 atcac_pk_sign()

```
ATCA_STATUS atcac_pk_sign (
    struct atcac_pk_ctx * ctx,
    const uint8_t * digest,
    size_t dig_len,
    uint8_t * signature,
    size_t sig_buf_size,
    size_t * sig_len )
```

Perform a signature with the private key in the context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.173.2.17 atcac_pk_verify()

```
ATCA_STATUS atcac_pk_verify (
    struct atcac_pk_ctx * ctx,
    const uint8_t * digest,
    size_t dig_len,
    const uint8_t * signature,
    size_t sig_len )
```

Perform a verify using the public key in the provided context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.173.2.18 atcac_sha256_hmac_finish()

```
ATCA_STATUS atcac_sha256_hmac_finish (
    struct atcac_hmac_ctx * ctx,
    uint8_t * digest,
    size_t * digest_len )
```

Finish CMAC calculation and clear the HMAC context.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a sha256-hmac context
out	<i>digest</i>	hmac value
in, out	<i>digest_len</i>	length of hmac

24.173.2.19 atcac_sha256_hmac_init()

```
ATCA_STATUS atcac_sha256_hmac_init (
    struct atcac_hmac_ctx * ctx,
    struct atcac_sha2_256_ctx * sha256_ctx,
    const uint8_t * key,
    const uint8_t key_len )
```

Initialize context for performing HMAC (sha256) in software.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a sha256-hmac context
in	<i>sha256_ctx</i>	pointer to a sha256 context
in	<i>key</i>	key value to use
in	<i>key_len</i>	length of the key

24.173.2.20 atcac_sha256_hmac_update()

```
ATCA_STATUS atcac_sha256_hmac_update (
    struct atcac_hmac_ctx * ctx,
```



```
const uint8_t * data,  
size_t data_size )
```

Update HMAC context with input data.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a sha256-hmac context
in	<i>data</i>	input data
in	<i>data_size</i>	length of input data

24.173.2.21 atcac_sw_random()

```
ATCA_STATUS atcac_sw_random (  
    uint8_t * data,  
    size_t data_size )
```

Return Random Bytes.

Returns

ATCA_SUCCESS on success, otherwise an error code.

24.173.2.22 atcac_sw_sha1_finish()

```
ATCA_STATUS atcac_sw_sha1_finish (  
    struct atcac_sha1_ctx * ctx,  
    uint8_t digest[ATCA_SHA1_DIGEST_SIZE] )
```

Complete the SHA1 hash in software and return the digest.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	<i>ctx</i>	pointer to a hash context
out	<i>digest</i>	output buffer (20 bytes)

24.173.2.23 atcac_sw_sha1_init()

```
ATCA_STATUS atcac_sw_sha1_init (
    struct atcac_sha1_ctx * ctx )
```

Initialize context for performing SHA1 hash in software.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	ctx	pointer to a hash context
----	-----	---------------------------

24.173.2.24 atcac_sw_sha1_update()

```
ATCA_STATUS atcac_sw_sha1_update (
    struct atcac_sha1_ctx * ctx,
    const uint8_t * data,
    size_t data_size )
```

Add data to a SHA1 hash.

Returns

ATCA_SUCCESS on success, otherwise an error code.

Parameters

in	ctx	pointer to a hash context
in	data	input data buffer
in	data_size	input data length

24.174 atca_openssl_interface.h File Reference

OpenSSL Integration Support.

```
#include "atca_config_check.h"
```

Data Structures

- struct [atcac_sha1_ctx](#)
- struct [atcac_sha2_256_ctx](#)
- struct [atcac_sha2_384_ctx](#)
- struct [atcac_sha2_512_ctx](#)
- struct [atcac_aes_cmac_ctx](#)
- struct [atcac_hmac_ctx](#)
- struct [atcac_pk_ctx](#)
- struct [atcac_x509_ctx](#)

Macros

- #define [ATCAC_SHA1_EN](#) (DEFAULT_ENABLED)
- #define [ATCAC_SHA256_EN](#) (FEATURE_ENABLED)
- #define [ATCAC_SHA384_EN](#) (FEATURE_DISABLED)
- #define [ATCAC_SHA512_EN](#) (FEATURE_DISABLED)
- #define [ATCAC_AES_CMAC_EN](#) (DEFAULT_ENABLED)
- #define [ATCAC_AES_GCM_EN](#) (DEFAULT_ENABLED)
- #define [ATCAC_PKEY_EN](#) (DEFAULT_ENABLED)
- #define [HOSTLIB_CERT_EN](#) (DEFAULT_ENABLED)

Typedefs

- typedef struct [atcac_sha1_ctx](#) [atcac_sha1_ctx_t](#)
- typedef struct [atcac_sha2_256_ctx](#) [atcac_sha2_256_ctx_t](#)
- typedef struct [atcac_sha2_384_ctx](#) [atcac_sha2_384_ctx_t](#)
- typedef struct [atcac_sha2_512_ctx](#) [atcac_sha2_512_ctx_t](#)
- typedef struct [atcac_aes_cmac_ctx](#) [atcac_aes_cmac_ctx_t](#)
- typedef struct [atcac_hmac_ctx](#) [atcac_hmac_ctx_t](#)
- typedef struct [atcac_pk_ctx](#) [atcac_pk_ctx_t](#)
- typedef struct [atcac_x509_ctx](#) [atcac_x509_ctx_t](#)

24.174.1 Detailed Description

OpenSSL Integration Support.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.174.2 Macro Definition Documentation

24.174.2.1 ATCAC_AES_CMAC_EN

```
#define ATCAC_AES_CMAC_EN (DEFAULT_ENABLED)
```

Indicates if this module is a provider of an AES-CMAC implementation

24.174.2.2 ATCAC_AES_GCM_EN

```
#define ATCAC_AES_GCM_EN (DEFAULT_ENABLED)
```

Indicates if this module is a provider of an AES-GCM implementation

24.174.2.3 ATCAC_PKEY_EN

```
#define ATCAC_PKEY_EN (DEFAULT_ENABLED)
```

Indicates if this module is a provider of a generic asymmetric cryptography implementation

24.174.2.4 ATCAC_SHA1_EN

```
#define ATCAC_SHA1_EN (DEFAULT_ENABLED)
```

Indicates if this module is a provider of a SHA1 implementation

24.174.2.5 ATCAC_SHA256_EN

```
#define ATCAC_SHA256_EN (FEATURE_ENABLED)
```

Indicates if this module is a provider of a SHA256 implementation

24.174.2.6 ATCAC_SHA384_EN

```
#define ATCAC_SHA384_EN (FEATURE_DISABLED)
```

Indicates if this module is a provider of a SHA384 implementation

Disabled by default. Use FEATURE_ENABLED to use SHA384

24.174.2.7 ATCAC_SHA512_EN

```
#define ATCAC_SHA512_EN (FEATURE_DISABLED)
```

Indicates if this module is a provider of a SHA512 implementation

Disabled by default. Use FEATURE_ENABLED to use SHA512

24.174.2.8 HOSTLIB_CERT_EN

```
#define HOSTLIB_CERT_EN (DEFAULT_ENABLED)
```

Indicates if this module is a provider of x509 certificate handling

24.175 pkcs11_attrib.c File Reference

PKCS11 Library Object Attributes Handling.

```
#include "pkcs11_config.h"
#include "pkcs11_attrib.h"
#include "cryptoauthlib.h"
#include "pkcs11_session.h"
```

Functions

- CK_RV [pkcs11_attrib_fill](#) (CK_ATTRIBUTE_PTR pAttribute, const void *pData, const CK_ULONG ulSize)
Perform the necessary checks and copy data into an attribute structure.
- CK_RV [pkcs11_attrib_value](#) (CK_ATTRIBUTE_PTR pAttribute, const CK_ULONG ulValue, const CK_ULONG ulSize)
Helper function to write a numerical value to an attribute buffer.
- CK_RV [pkcs11_attrib_false](#) (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV [pkcs11_attrib_true](#) (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV [pkcs11_attrib_empty](#) (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)

24.175.1 Detailed Description

PKCS11 Library Object Attributes Handling.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.176 pkcs11_attrib.h File Reference

PKCS11 Library Object Attribute Handling.

```
#include "cryptoauthlib.h"
#include "cryptoki.h"
#include "pkcs11_session.h"
```

Data Structures

- struct [pkcs11_attrib_model_s](#)

Typedefs

- typedef CK_RV(* [attrib_f](#)) (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- typedef struct [pkcs11_attrib_model_s](#) **pkcs11_attrib_model**
- typedef struct [pkcs11_attrib_model_s](#) * **pkcs11_attrib_model_ptr**

Functions

- CK_RV [pkcs11_attrib_fill](#) (CK_ATTRIBUTE_PTR pAttribute, const void *pData, const CK_ULONG ulSize)
Perform the nessasary checks and copy data into an attribute structure.
- CK_RV **pkcs11_attrib_value** (CK_ATTRIBUTE_PTR pAttribute, const CK_ULONG ulValue, const CK_ULONG ulSize)
Helper function to write a numerical value to an attribute buffer.
- CK_RV **pkcs11_attrib_false** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV **pkcs11_attrib_true** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV **pkcs11_attrib_empty** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)

24.176.1 Detailed Description

PKCS11 Library Object Attribute Handling.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.176.2 Typedef Documentation

24.176.2.1 [attrib_f](#)

```
typedef CK_RV(* attrib\_f) (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, pkcs11\_session\_ctx\_ptr pSession)
```

Populate an attribute based on the "object"

24.177 pkcs11_cert.c File Reference

PKCS11 Library Certificate Handling.

```
#include "cryptoauthlib.h"
#include "atcacert/atcacert_def.h"
#include "atcacert/atcacert_client.h"
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_token.h"
#include "pkcs11_cert.h"
#include "pkcs11_os.h"
#include "pkcs11_util.h"
#include "pkcs11_slot.h"
#include "pkcs11_key.h"
```

Functions

- CK_RV **pkcs11_cert_load** (pkcs11_object_ptr pObject, CK_ATTRIBUTE_PTR pAttribute, [ATCADevice](#) device)
- CK_RV **pkcs11_cert_x509_write** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV **pkcs11_cert_clear_session_cache** ([pkcs11_session_ctx_ptr](#) session_ctx)
- CK_RV **pkcs11_cert_clear_object_cache** (pkcs11_object_ptr pObject)

Variables

- const [pkcs11_attr_model](#) [pkcs11_cert_x509public_attributes](#) []
- const CK_ULONG **pkcs11_cert_x509public_attributes_count** = (CK_ULONG)(sizeof([pkcs11_cert_x509public_attributes](#)) / sizeof([pkcs11_cert_x509public_attributes](#)[0]))
- const [pkcs11_attr_model](#) [pkcs11_cert_wtlspublic_attributes](#) []
- const CK_ULONG **pkcs11_cert_wtlspublic_attributes_count** = (CK_ULONG)(sizeof([pkcs11_cert_wtlspublic_attributes](#)) / sizeof([pkcs11_cert_wtlspublic_attributes](#)[0]))
- const [pkcs11_attr_model](#) [pkcs11_cert_x509_attributes](#) []
- const CK_ULONG **pkcs11_cert_x509_attributes_count** = (CK_ULONG)(sizeof([pkcs11_cert_x509_attributes](#)) / sizeof([pkcs11_cert_x509_attributes](#)[0]))

24.177.1 Detailed Description

PKCS11 Library Certificate Handling.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.178 pkcs11_cert.h File Reference

PKCS11 Library Certificate Handling.

```
#include "pkcs11_object.h"
```

Functions

- CK_RV **pkcs11_cert_x509_write** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV **pkcs11_cert_load** (pkcs11_object_ptr pObject, CK_ATTRIBUTE_PTR pAttribute, [ATCADevice](#) device)
- CK_RV **pkcs11_cert_clear_session_cache** ([pkcs11_session_ctx_ptr](#) session_ctx)
- CK_RV **pkcs11_cert_clear_object_cache** (pkcs11_object_ptr pObject)

Variables

- const [pkcs11_attr_model](#) [pkcs11_cert_x509public_attributes](#) []
- const CK_ULONG [pkcs11_cert_x509public_attributes_count](#)
- const [pkcs11_attr_model](#) [pkcs11_cert_wtlspublic_attributes](#) []
- const CK_ULONG [pkcs11_cert_wtlspublic_attributes_count](#)
- const [pkcs11_attr_model](#) [pkcs11_cert_x509_attributes](#) []
- const CK_ULONG [pkcs11_cert_x509_attributes_count](#)

24.178.1 Detailed Description

PKCS11 Library Certificate Handling.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.179 pkcs11_config.c File Reference

PKCS11 Library Configuration.

```
#include <stdbool.h>
#include "cryptoauthlib.h"
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_slot.h"
#include "pkcs11_object.h"
#include "pkcs11_key.h"
#include "pkcs11_cert.h"
#include "pkcs11_os.h"
#include "pkcs11_util.h"
#include <limits.h>
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <errno.h>
#include <fcntl.h>
#include <dirent.h>
```

Data Structures

- struct [pkcs11_conf_filedata_s](#)

Macros

- #define [PKCS11_CONFIG_U8_MAX](#) 0xFFL
- #define [PKCS11_CONFIG_U16_MAX](#) 0xFFFFL
- #define [PKCS11_CONFIG_U32_MAX](#) 0xFFFFFFFFL

Typedefs

- typedef struct [pkcs11_conf_filedata_s](#) **pkcs11_conf_filedata**
- typedef struct [pkcs11_conf_filedata_s](#) * **pkcs11_conf_filedata_ptr**

Functions

- void **pkcs11_config_set_key_size** (pkcs11_object_ptr pObject)
- void **pkcs11_config_init_private** (pkcs11_object_ptr pObject, const char *label, size_t len)
- void **pkcs11_config_init_public** (pkcs11_object_ptr pObject, const char *label, size_t len)
- void **pkcs11_config_init_secret** (pkcs11_object_ptr pObject, const char *label, size_t len, size_t keylen)
- void **pkcs11_config_init_cert** (pkcs11_object_ptr pObject, const char *label, size_t len)
- void **pkcs11_config_split_string** (char *s, char splitter, int *argc, char *argv[])
- CK_RV **pkcs11_config_cert** (pkcs11_lib_ctx_ptr pLibCtx, pkcs11_slot_ctx_ptr pSlot, pkcs11_object_ptr p↔Object, CK_ATTRIBUTE_PTR pLabel)
- CK_RV **pkcs11_config_key** (pkcs11_lib_ctx_ptr pLibCtx, pkcs11_slot_ctx_ptr pSlot, pkcs11_object_ptr p↔Object, CK_ATTRIBUTE_PTR pLabel)
- CK_RV **pkcs11_config_remove_object** (pkcs11_lib_ctx_ptr pLibCtx, pkcs11_slot_ctx_ptr pSlot, pkcs11_↔object_ptr pObject)
- CK_RV **pkcs11_config_load_objects** (pkcs11_slot_ctx_ptr slot_ctx)
- CK_RV **pkcs11_config_load** (pkcs11_slot_ctx_ptr slot_ctx)

24.179.1 Detailed Description

PKCS11 Library Configuration.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.180 pkcs11_debug.c File Reference

PKCS11 Library Debugging.

```
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_os.h"
#include "atca_helpers.h"
```

24.180.1 Detailed Description

PKCS11 Library Debugging.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.181 pkcs11_debug.h File Reference

PKCS11 Library Debugging.

```
#include "pkcs11_config.h"
```

Macros

- `#define PKCS11_DEBUG_NOFILE(...)`
- `#define PKCS11_DEBUG(...)`
- `#define PKCS11_DEBUG_RETURN(x) { return x; }`
- `#define pkcs11_debug_attributes(x, y)`

24.181.1 Detailed Description

PKCS11 Library Debugging.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.182 pkcs11_digest.h File Reference

PKCS11 Library Digest (SHA256) Handling.

```
#include "cryptoki.h"
```

Functions

- `CK_RV pkcs11_digest_init` (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism)
Initializes a message-digesting operation using the specified mechanism in the specified session.
- `CK_RV pkcs11_digest` (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pDigest, CK_ULONG_PTR pulDigestLen)
Digest the specified data in a one-pass operation and return the resulting digest.
- `CK_RV pkcs11_digest_update` (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen)
Continues a multiple-part digesting operation.
- `CK_RV pkcs11_digest_final` (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pDigest, CK_ULONG_PTR pulDigestLen)
Finishes a multiple-part digesting operation.

24.182.1 Detailed Description

PKCS11 Library Digest (SHA256) Handling.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.183 pkcs11_encrypt.c File Reference

PKCS11 Library Encrypt Support.

```
#include "cryptoauthlib.h"
#include <limits.h>
#include "pkcs11_config.h"
#include "pkcs11_encrypt.h"
#include "pkcs11_debug.h"
#include "pkcs11_init.h"
#include "pkcs11_object.h"
#include "pkcs11_session.h"
#include "pkcs11_util.h"
#include "pkcs11_slot.h"
#include "pkcs11_key.h"
```

Functions

- CK_RV **pkcs11_encrypt_init** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hObject)
- CK_RV **pkcs11_encrypt** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pEncryptedData, CK_ULONG_PTR pulEncryptedDataLen)
- CK_RV **pkcs11_encrypt_update** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pEncryptedData, CK_ULONG_PTR pulEncryptedDataLen)
- CK_RV **pkcs11_encrypt_final** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedData, CK_ULONG_PTR pulEncryptedDataLen)
Finishes a multiple-part encryption operation.
- CK_RV **pkcs11_decrypt_init** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hObject)
- CK_RV **pkcs11_decrypt** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedData, CK_ULONG ulEncryptedDataLen, CK_BYTE_PTR pData, CK_ULONG_PTR pulDataLen)
- CK_RV **pkcs11_decrypt_update** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedData, CK_ULONG ulEncryptedDataLen, CK_BYTE_PTR pData, CK_ULONG_PTR pulDataLen)
- CK_RV **pkcs11_decrypt_final** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG_PTR pulDataLen)
Finishes a multiple-part decryption operation.

24.183.1 Detailed Description

PKCS11 Library Encrypt Support.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.184 pkcs11_encrypt.h File Reference

PKCS11 Library AES Support.

```
#include "cryptoki.h"
```

Functions

- CK_RV **pkcs11_encrypt_init** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hObject)
- CK_RV **pkcs11_encrypt** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pEncryptedData, CK_ULONG_PTR pulEncryptedDataLen)
- CK_RV **pkcs11_encrypt_update** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pEncryptedData, CK_ULONG_PTR pulEncryptedDataLen)
- CK_RV **pkcs11_encrypt_final** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedData, CK_ULONG_PTR pulEncryptedDataLen)
Finishes a multiple-part encryption operation.
- CK_RV **pkcs11_decrypt_init** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hObject)
- CK_RV **pkcs11_decrypt** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedData, CK_ULONG ulEncryptedDataLen, CK_BYTE_PTR pData, CK_ULONG_PTR pulDataLen)
- CK_RV **pkcs11_decrypt_update** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedData, CK_ULONG ulEncryptedDataLen, CK_BYTE_PTR pData, CK_ULONG_PTR pulDataLen)
- CK_RV **pkcs11_decrypt_final** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG_PTR pulDataLen)
Finishes a multiple-part decryption operation.

24.184.1 Detailed Description

PKCS11 Library AES Support.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.185 pkcs11_find.c File Reference

PKCS11 Library Object Find/Searching.

```
#include "cryptoauthlib.h"
#include "atcacert/atcacert_def.h"
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_init.h"
#include "pkcs11_os.h"
#include "pkcs11_slot.h"
#include "pkcs11_session.h"
#include "pkcs11_find.h"
#include "pkcs11_util.h"
#include "pkcs11_cert.h"
```

Functions

- CK_RV **pkcs11_find_init** (CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount)
- CK_RV **pkcs11_find_continue** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE_PTR phObject, CK_ULONG ulMaxObjectCount, CK_ULONG_PTR pulObjectCount)
- CK_RV **pkcs11_find_finish** (CK_SESSION_HANDLE hSession)
- CK_RV **pkcs11_find_get_attribute** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount)

24.185.1 Detailed Description

PKCS11 Library Object Find/Searching.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.186 pkcs11_find.h File Reference

PKCS11 Library Object Find/Searching.

```
#include "cryptoki.h"
#include "pkcs11_object.h"
```

Functions

- CK_RV **pkcs11_find_init** (CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_↔
ULONG ulCount)
- CK_RV **pkcs11_find_continue** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE_PTR ph↔
Object, CK_ULONG ulMaxObjectCount, CK_ULONG_PTR pulObjectCount)
- CK_RV **pkcs11_find_finish** (CK_SESSION_HANDLE hSession)
- CK_RV **pkcs11_find_get_attribute** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject,
CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount)

24.186.1 Detailed Description

PKCS11 Library Object Find/Searching.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.187 pkcs11_info.c File Reference

PKCS11 Library Information Functions.

```
#include "cryptoauthlib.h"
#include "pkcs11_config.h"
#include "pkcs11_init.h"
#include "pkcs11_slot.h"
#include "pkcs11_session.h"
#include "pkcs11_util.h"
#include "pkcs11_info.h"
#include <stdio.h>
```

Functions

- CK_RV **pkcs11_get_lib_info** (CK_INFO_PTR pInfo)
Obtains general information about Cryptoki.

Variables

- const char **pkcs11_lib_manufacturer_id** [] = "Microchip Technology Inc"
- const char **pkcs11_lib_description** [] = "Cryptoauthlib PKCS11 Interface"

24.187.1 Detailed Description

PKCS11 Library Information Functions.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.188 pkcs11_info.h File Reference

PKCS11 Library Information Functions.

```
#include "cryptoki.h"
```

Functions

- CK_RV **pkcs11_get_lib_info** (CK_INFO_PTR pInfo)
Obtains general information about Cryptoki.

Variables

- const char **pkcs11_lib_manufacturer_id** []
- const char **pkcs11_lib_description** []

24.188.1 Detailed Description

PKCS11 Library Information Functions.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.189 pkcs11_init.c File Reference

PKCS11 Library Init/Deinit.

```
#include "atca_device.h"
#include "hal/atca_hal.h"
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_init.h"
#include "pkcs11_os.h"
#include "pkcs11_slot.h"
#include "pkcs11_object.h"
#include "pkcs11_session.h"
#include "cryptoauthlib.h"
```

Functions

- `pkcs11_lib_ctx_ptr` **pkcs11_get_context** (void)
Retrieve the current library context.
- `CK_RV` **pkcs11_lock_context** (`pkcs11_lib_ctx_ptr` pContext)
- `CK_RV` **pkcs11_unlock_context** (`pkcs11_lib_ctx_ptr` pContext)
- `CK_RV` **pkcs11_lock_device** (`pkcs11_lib_ctx_ptr` pContext)
- `CK_RV` **pkcs11_unlock_device** (`pkcs11_lib_ctx_ptr` pContext)
- `CK_RV` **pkcs11_lock_both** (`pkcs11_lib_ctx_ptr` pContext)
- `CK_RV` **pkcs11_unlock_both** (`pkcs11_lib_ctx_ptr` pContext)
- `CK_RV` **pkcs11_init_check** (`pkcs11_lib_ctx_ptr` *ppContext, `CK_BBOOL` lock)
Check if the library is initialized properly.
- `CK_RV` **pkcs11_init** ([CK_C_INITIALIZE_ARGS](#) const *pInitArgs)
Initializes the PKCS11 API Library for Cryptoauthlib.
- `CK_RV` **pkcs11_deinit** (`CK_VOID_PTR` pReserved)

24.189.1 Detailed Description

PKCS11 Library Init/Deinit.

Copyright (c) 2017 Microchip Technology Inc. All rights reserved.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.190 pkcs11_init.h File Reference

PKCS11 Library Initialization & Context.

```
#include "atca_compiler.h"
#include "pkcs11_config.h"
#include "pkcs11_os.h"
#include "cryptoauthlib.h"
```

Data Structures

- struct [pkcs11_dev_ctx](#)
- struct [pkcs11_dev_res](#)
- struct [pkcs11_dev_state](#)
- struct [pkcs11_lib_ctx_s](#)

Macros

- #define **PKCS11_AES_OP** (0x0u)
- #define **PKCS11_DIGEST_OP_0** (0x1u)
- #define **PKCS11_DIGEST_OP_1** (0x2u)
- #define **PKCS11_AUTH_OP_0** (0x3u)
- #define **PKCS11_AUTH_OP_1** (0x4u)
- #define **PKCS11_MAX_DEV_CTX** (5u)
- #define **MAX_DIGEST_SESSIONS** (2u)
- #define **MAX_AUTH_SESSIONS** (2u)

Typedefs

- typedef struct [pkcs11_lib_ctx_s](#) [pkcs11_lib_ctx](#)

Functions

- CK_RV [pkcs11_init](#) ([CK_C_INITIALIZE_ARGS](#) const *pInitArgs)
Initializes the PKCS11 API Library for Cryptoauthlib.
- CK_RV [pkcs11_deinit](#) (CK_VOID_PTR pReserved)
- CK_RV [pkcs11_init_check](#) (pkcs11_lib_ctx_ptr *ppContext, CK_BBOOL lock)
Check if the library is initialized properly.
- pkcs11_lib_ctx_ptr [pkcs11_get_context](#) (void)
Retrieve the current library context.
- CK_RV [pkcs11_lock_context](#) (pkcs11_lib_ctx_ptr pContext)
- CK_RV [pkcs11_unlock_context](#) (pkcs11_lib_ctx_ptr pContext)
- CK_RV [pkcs11_lock_device](#) (pkcs11_lib_ctx_ptr pContext)
- CK_RV [pkcs11_unlock_device](#) (pkcs11_lib_ctx_ptr pContext)
- CK_RV [pkcs11_lock_both](#) (pkcs11_lib_ctx_ptr pContext)
- CK_RV [pkcs11_unlock_both](#) (pkcs11_lib_ctx_ptr pContext)

24.190.1 Detailed Description

PKCS11 Library Initialization & Context.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.190.2 Typedef Documentation

24.190.2.1 pkcs11_lib_ctx

```
typedef struct pkcs11_lib_ctx_s pkcs11_lib_ctx
```

Library Context

24.191 pkcs11_key.c File Reference

PKCS11 Library Key Object Handling.

```
#include "cryptoauthlib.h"
#include "crypto/atca_crypto_sw_sha1.h"
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_token.h"
#include "pkcs11_attrib.h"
#include "pkcs11_key.h"
#include "pkcs11_session.h"
#include "pkcs11_slot.h"
#include "pkcs11_util.h"
#include "pkcs11_os.h"
```

Functions

- const [pkcs11_key_info_t](#) * **pkcs11_get_object_key_type** ([ATCADevice](#) device_ctx, [pkcs11_object_ptr](#) obj↔_ptr)
- CK_RV **pkcs11_ta_get_pubkey** (CK_VOID_PTR pObject, [cal_buffer](#) *key_buffer, [pkcs11_session_ctx_ptr](#) session_ctx)
- CK_RV **pkcs11_key_write** (CK_VOID_PTR pSession, CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR p↔Attribute)
- CK_RV **pkcs11_key_generate** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phKey)
- CK_RV **pkcs11_key_generate_pair** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR p↔Mechanism, CK_ATTRIBUTE_PTR pPublicKeyTemplate, CK_ULONG ulPublicKeyAttributeCount, CK↔ATTRIBUTE_PTR pPrivateKeyTemplate, CK_ULONG ulPrivateKeyAttributeCount, CK_OBJECT_HANDLE↔_PTR phPublicKey, CK_OBJECT_HANDLE_PTR phPrivateKey)
- CK_RV **pkcs11_key_derive** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hBaseKey, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT↔_HANDLE_PTR phKey)
- CK_RV **pkcs11_key_clear_session_cache** ([pkcs11_session_ctx_ptr](#) session_ctx)
- CK_RV **pkcs11_key_clear_object_cache** ([pkcs11_object_ptr](#) pObject)

Variables

- CK_BYTE [pkcs11_ec_pbkey_asn1_hdr_p256](#) []
- CK_BYTE [pkcs11_x962_asn1_hdr_ec256](#) []
- CK_BYTE **pkcs11_key_ec_params_p256** [] = { 0x06, 0x08, 0x2a, 0x86, 0x48, 0xce, 0x3d, 0x03, 0x01, 0x07 }
- CK_BYTE [pkcs11_ec_pbkey_asn1_hdr_p224](#) []
- CK_BYTE [pkcs11_x962_asn1_hdr_ec224](#) []
- CK_BYTE **pkcs11_key_ec_params_p224** [] = { 0x06, 0x05, 0x2B, 0x81, 0x04, 0x00, 0x21 }

- CK_BYTE [pkcs11_ec_pbkey_asn1_hdr_p384](#) []
- CK_BYTE [pkcs11_key_ec_params_p384](#) [] = { 0x06, 0x05, 0x2B, 0x81, 0x04, 0x00, 0x22 }
- CK_BYTE [pkcs11_x962_asn1_hdr_ec384](#) []
- CK_BYTE [pkcs11_ec_pbkey_asn1_hdr_p521](#) []
- CK_BYTE [pkcs11_x962_asn1_hdr_ec521](#) []
- CK_BYTE [pkcs11_key_ec_params_p521](#) [] = { 0x06, 0x05, 0x2B, 0x81, 0x04, 0x00, 0x23 }
- const [pkcs11_ecc_key_info_t ec_key_data_table](#) [4]
- const [pkcs11_rsa_key_info_t rsa_key_data_table](#) [4]
- const [pkcs11_key_info_t key_data_table](#) []
- const [pkcs11_attr_model pkcs11_key_public_attributes](#) []
- const CK_ULONG [pkcs11_key_public_attributes_count](#) = (CK_ULONG)(sizeof([pkcs11_key_public_attributes](#)) / sizeof([pkcs11_key_public_attributes](#) [0]))
- const [pkcs11_attr_model pkcs11_key_private_attributes](#) []
- const CK_ULONG [pkcs11_key_private_attributes_count](#) = (CK_ULONG)(sizeof([pkcs11_key_private_attributes](#)) / sizeof([pkcs11_key_private_attributes](#) [0]))
- const [pkcs11_attr_model pkcs11_key_secret_attributes](#) []
- const CK_ULONG [pkcs11_key_secret_attributes_count](#) = (CK_ULONG)(sizeof([pkcs11_key_secret_attributes](#)) / sizeof([pkcs11_key_secret_attributes](#) [0]))

24.191.1 Detailed Description

PKCS11 Library Key Object Handling.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.192 pkcs11_key.h File Reference

PKCS11 Library Object Handling.

```
#include "pkcs11_object.h"
```

Data Structures

- struct [pkcs11_ecc_key_info_s](#)
- struct [pkcs11_rsa_key_info_s](#)
- struct [pkcs11_key_info_s](#)

Macros

- #define [PKCS11_X962_ASN1_HEADER_SZ](#) 3u
- #define [PKCS11_MAX_ECC_ASN1_HDR_SIZE](#) ATCA_ECCP256_ASN1_HDR_SIZE
- #define [PKCS11_MAX_ECC_RSA_ASN1_HDR_SIZE](#) ATCA_RSA4K_ASN1_HDR_SIZE
- #define [PKCS11_MAX_ECC_RSA_PB_KEY_SIZE](#) ATCA_MAX_ECC_RSA_PB_KEY_SIZE
- #define [PKCS11_MAX_ECC_PB_KEY_SIZE](#) TA_ECC521_PUB_KEY_SIZE
- #define [PKCS11_MAX_RSA_PB_KEY_SIZE](#) TA_KEY_TYPE_RSA4096_SIZE

Typedefs

- typedef struct [pkcs11_ecc_key_info_s](#) **pkcs11_ecc_key_info_t**
- typedef struct [pkcs11_rsa_key_info_s](#) **pkcs11_rsa_key_info_t**
- typedef struct [pkcs11_key_info_s](#) **pkcs11_key_info_t**

Functions

- CK_RV **pkcs11_key_write** (CK_VOID_PTR pSession, CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute)
- CK_RV **pkcs11_key_generate** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phKey)
- CK_RV **pkcs11_key_generate_pair** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_ATTRIBUTE_PTR pPublicKeyTemplate, CK_ULONG ulPublicKeyAttributeCount, CK_ATTRIBUTE_PTR pPrivateKeyTemplate, CK_ULONG ulPrivateKeyAttributeCount, CK_OBJECT_HANDLE_PTR phPublicKey, CK_OBJECT_HANDLE_PTR phPrivateKey)
- CK_RV **pkcs11_key_derive** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hBaseKey, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phKey)
- CK_RV **pkcs11_key_clear_session_cache** ([pkcs11_session_ctx_ptr](#) session_ctx)
- CK_RV **pkcs11_key_clear_object_cache** ([pkcs11_object_ptr](#) pObject)
- const [pkcs11_key_info_t](#) * **pkcs11_get_object_key_type** ([ATCADevice](#) device_ctx, [pkcs11_object_ptr](#) obj_ptr)
- CK_RV **pkcs11_ta_get_pubkey** (CK_VOID_PTR pObject, [cal_buffer](#) *key_buffer, [pkcs11_session_ctx_ptr](#) session_ctx)

Variables

- const [pkcs11_ecc_key_info_t](#) **ec_key_data_table** [4]
- const [pkcs11_rsa_key_info_t](#) **rsa_key_data_table** [4]
- const [pkcs11_key_info_t](#) **key_data_table** []
- const [pkcs11_attr_model](#) **pkcs11_key_public_attributes** []
- const CK_ULONG **pkcs11_key_public_attributes_count**
- const [pkcs11_attr_model](#) **pkcs11_key_private_attributes** []
- const CK_ULONG **pkcs11_key_private_attributes_count**
- const [pkcs11_attr_model](#) **pkcs11_key_secret_attributes** []
- const CK_ULONG **pkcs11_key_secret_attributes_count**
- CK_BYTE **pkcs11_ec_pbkey_asn1_hdr_p256** []
- CK_BYTE **pkcs11_x962_asn1_hdr_ec256** []
- CK_BYTE **pkcs11_key_ec_params_p256** []
- CK_BYTE **pkcs11_ec_pbkey_asn1_hdr_p224** []
- CK_BYTE **pkcs11_x962_asn1_hdr_ec224** []
- CK_BYTE **pkcs11_key_ec_params_p224** []
- CK_BYTE **pkcs11_ec_pbkey_asn1_hdr_p384** []
- CK_BYTE **pkcs11_x962_asn1_hdr_ec384** []
- CK_BYTE **pkcs11_key_ec_params_p384** []
- CK_BYTE **pkcs11_ec_pbkey_asn1_hdr_p521** []
- CK_BYTE **pkcs11_x962_asn1_hdr_ec521** []
- CK_BYTE **pkcs11_key_ec_params_p521** []

24.192.1 Detailed Description

PKCS11 Library Object Handling.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.193 pkcs11_main.c File Reference

PKCS11 Basic library redirects based on the 2.40 specification docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.html.

```
#include "cryptoki.h"
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_encrypt.h"
#include "pkcs11_init.h"
#include "pkcs11_info.h"
#include "pkcs11_slot.h"
#include "pkcs11_mech.h"
#include "pkcs11_session.h"
#include "pkcs11_token.h"
#include "pkcs11_find.h"
#include "pkcs11_object.h"
#include "pkcs11_signature.h"
#include "pkcs11_digest.h"
#include "pkcs11_key.h"
```

Functions

- **CK_RV C_Initialize** (CK_VOID_PTR pInitArgs)
Initializes Cryptoki library NOTES: If pInitArgs is a non-NULL_PTR is must dereference to a [CK_C_INITIALIZE_ARGS](#) structure.
- **CK_RV C_Finalize** (CK_VOID_PTR pReserved)
Clean up miscellaneous Cryptoki-associated resources.
- **CK_RV C_GetInfo** (CK_INFO_PTR pInfo)
Obtains general information about Cryptoki.
- **CK_RV C_GetFunctionList** (CK_FUNCTION_LIST_PTR_PTR ppFunctionList)
Obtains entry points of Cryptoki library functions.
- **CK_RV C_GetSlotList** (CK_BBOOL tokenPresent, CK_SLOT_ID_PTR pSlotList, CK_ULONG_PTR pulCount)
Obtains a list of slots in the system.
- **CK_RV C_GetSlotInfo** (CK_SLOT_ID slotID, CK_SLOT_INFO_PTR pInfo)
Obtains information about a particular slot.
- **CK_RV C_GetTokenInfo** (CK_SLOT_ID slotID, CK_TOKEN_INFO_PTR pInfo)
Obtains information about a particular token.
- **CK_RV C_GetMechanismList** (CK_SLOT_ID slotID, CK_MECHANISM_TYPE_PTR pMechanismList, CK_ULONG_PTR pulCount)
Obtains a list of mechanisms supported by a token (in a slot)

- **CK_RV C_GetMechanismInfo** (CK_SLOT_ID slotID, CK_MECHANISM_TYPE type, CK_MECHANISM_↵
INFO_PTR pInfo)
Obtains information about a particular mechanism of a token (in a slot)
- **CK_RV C_InitToken** (CK_SLOT_ID slotID, CK_UTF8CHAR_PTR pPin, CK_ULONG ulPinLen, CK_UTF8_↵
CHAR_PTR pLabel)
Initializes a token (in a slot)
- **CK_RV C_InitPIN** (CK_SESSION_HANDLE hSession, CK_UTF8CHAR_PTR pPin, CK_ULONG ulPinLen)
Initializes the normal user's PIN.
- **CK_RV C_SetPIN** (CK_SESSION_HANDLE hSession, CK_UTF8CHAR_PTR pOldPin, CK_ULONG ulOld_↵
Len, CK_UTF8CHAR_PTR pNewPin, CK_ULONG ulNewLen)
Modifies the PIN of the current user.
- **CK_RV C_OpenSession** (CK_SLOT_ID slotID, CK_FLAGS flags, CK_VOID_PTR pApplication, CK_NOTIFY
Notify, CK_SESSION_HANDLE_PTR phSession)
*Opens a connection between an application and a particular token or sets up an application callback for token inser-
tion.*
- **CK_RV C_CloseSession** (CK_SESSION_HANDLE hSession)
Close the given session.
- **CK_RV C_CloseAllSessions** (CK_SLOT_ID slotID)
Close all open sessions.
- **CK_RV C_GetSessionInfo** (CK_SESSION_HANDLE hSession, CK_SESSION_INFO_PTR pInfo)
Retrieve information about the specified session.
- **CK_RV C_GetOperationState** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pOperationState, CK_↵
_ULONG_PTR pulOperationStateLen)
Obtains the cryptographic operations state of a session.
- **CK_RV C_SetOperationState** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pOperationState, CK_ULONG ulOperationStateLen, CK_OBJECT_HANDLE hEncryptionKey, CK_OBJECT_HANDLE h_↵
AuthenticationKey)
Sets the cryptographic operations state of a session.
- **CK_RV C_Login** (CK_SESSION_HANDLE hSession, CK_USER_TYPE userType, CK_UTF8CHAR_PTR
pPin, CK_ULONG ulPinLen)
Login on the token in the specified session.
- **CK_RV C_Logout** (CK_SESSION_HANDLE hSession)
Log out of the token in the specified session.
- **CK_RV C_CreateObject** (CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_↵
ULONG ulCount, CK_OBJECT_HANDLE_PTR phObject)
Create a new object on the token in the specified session using the given attribute template.
- **CK_RV C_CopyObject** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_↵
ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phNewObject)
Create a copy of the object with the specified handle.
- **CK_RV C_DestroyObject** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject)
Destroy the specified object.
- **CK_RV C_GetObjectSize** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_↵
ULONG_PTR pulSize)
Obtains the size of an object in bytes.
- **CK_RV C_GetAttributeValue** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_↵
ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount)
Obtains an attribute value of an object.
- **CK_RV C_SetAttributeValue** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_↵
ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount)
Change or set the value of the specified attributes on the specified object.
- **CK_RV C_FindObjectsInit** (CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_↵
ULONG ulCount)
Initializes an object search in the specified session using the specified attribute template as search parameters.

- **CK_RV C_FindObjects** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE_PTR phObject, CK_↔
_ULONG ulMaxObjectCount, CK_ULONG_PTR pulObjectCount)
Continue the search for objects in the specified session.
- **CK_RV C_FindObjectsFinal** (CK_SESSION_HANDLE hSession)
Finishes an object search operation (and cleans up)
- **CK_RV C_EncryptInit** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_↔
OBJECT_HANDLE hKey)
Initializes an encryption operation using the specified mechanism and session.
- **CK_RV C_Encrypt** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen,
CK_BYTE_PTR pEncryptedData, CK_ULONG_PTR pulEncryptedDataLen)
Perform a single operation encryption operation in the specified session.
- **CK_RV C_EncryptUpdate** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ul↔
PartLen, CK_BYTE_PTR pEncryptedPart, CK_ULONG_PTR pulEncryptedPartLen)
Continues a multiple-part encryption operation.
- **CK_RV C_EncryptFinal** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pLastEncryptedPart, CK_↔
ULONG_PTR pulLastEncryptedPartLen)
Finishes a multiple-part encryption operation.
- **CK_RV C_DecryptInit** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_↔
OBJECT_HANDLE hKey)
Initialize decryption using the specified object.
- **CK_RV C_Decrypt** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedData, CK_ULONG ul↔
EncryptedDataLen, CK_BYTE_PTR pData, CK_ULONG_PTR pulDataLen)
Perform a single operation decryption in the given session.
- **CK_RV C_DecryptUpdate** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedPart, CK_↔
ULONG ulEncryptedPartLen, CK_BYTE_PTR pPart, CK_ULONG_PTR pulPartLen)
Continues a multiple-part decryption operation.
- **CK_RV C_DecryptFinal** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pLastPart, CK_ULONG_PTR
pulLastPartLen)
Finishes a multiple-part decryption operation.
- **CK_RV C_DigestInit** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism)
Initializes a message-digesting operation using the specified mechanism in the specified session.
- **CK_RV C_Digest** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen,
CK_BYTE_PTR pDigest, CK_ULONG_PTR pulDigestLen)
Digest the specified data in a one-pass operation and return the resulting digest.
- **CK_RV C_DigestUpdate** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPart↔
Len)
Continues a multiple-part digesting operation.
- **CK_RV C_DigestKey** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hKey)
Update a running digest operation by digesting a secret key with the specified handle.
- **CK_RV C_DigestFinal** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pDigest, CK_ULONG_PTR
pulDigestLen)
Finishes a multiple-part digesting operation.
- **CK_RV C_SignInit** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_↔
OBJECT_HANDLE hKey)
Initialize a signing operation using the specified key and mechanism.
- **CK_RV C_Sign** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_↔
_BYTE_PTR pSignature, CK_ULONG_PTR pulSignatureLen)
Sign the data in a single pass operation.
- **CK_RV C_SignUpdate** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen)
Continues a multiple-part signature operation.
- **CK_RV C_SignFinal** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pSignature, CK_ULONG_PTR
pulSignatureLen)
Finishes a multiple-part signature operation.

- **CK_RV C_SignRecoverInit** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey)
Initializes a signature operation, where the data can be recovered from the signature.
- **CK_RV C_SignRecover** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG_PTR pulSignatureLen)
Signs single-part data, where the data can be recovered from the signature.
- **CK_RV C_VerifyInit** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey)
Initializes a verification operation using the specified key and mechanism.
- **CK_RV C_Verify** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG ulSignatureLen)
Verifies a signature on single-part data.
- **CK_RV C_VerifyUpdate** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen)
Continues a multiple-part verification operation.
- **CK_RV C_VerifyFinal** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pSignature, CK_ULONG ulSignatureLen)
Finishes a multiple-part verification operation.
- **CK_RV C_VerifyRecoverInit** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey)
Initializes a verification operation where the data is recovered from the signature.
- **CK_RV C_VerifyRecover** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pSignature, CK_ULONG ulSignatureLen, CK_BYTE_PTR pData, CK_ULONG_PTR pulDataLen)
Verifies a signature on single-part data, where the data is recovered from the signature.
- **CK_RV C_DigestEncryptUpdate** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen, CK_BYTE_PTR pEncryptedPart, CK_ULONG_PTR pulEncryptedPartLen)
Continues simultaneous multiple-part digesting and encryption operations.
- **CK_RV C_DecryptDigestUpdate** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedPart, CK_ULONG ulEncryptedPartLen, CK_BYTE_PTR pPart, CK_ULONG_PTR pulPartLen)
Continues simultaneous multiple-part decryption and digesting operations.
- **CK_RV C_SignEncryptUpdate** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen, CK_BYTE_PTR pEncryptedPart, CK_ULONG_PTR pulEncryptedPartLen)
Continues simultaneous multiple-part signature and encryption operations.
- **CK_RV C_DecryptVerifyUpdate** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedPart, CK_ULONG ulEncryptedPartLen, CK_BYTE_PTR pPart, CK_ULONG_PTR pulPartLen)
Continues simultaneous multiple-part decryption and verification operations.
- **CK_RV C_GenerateKey** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phKey)
Generates a secret key using the specified mechanism.
- **CK_RV C_GenerateKeyPair** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_ATTRIBUTE_PTR pPublicKeyTemplate, CK_ULONG ulPublicKeyAttributeCount, CK_ATTRIBUTE_PTR pPrivateKeyTemplate, CK_ULONG ulPrivateKeyAttributeCount, CK_OBJECT_HANDLE_PTR phPublicKey, CK_OBJECT_HANDLE_PTR phPrivateKey)
Generates a public-key/private-key pair using the specified mechanism.
- **CK_RV C_WrapKey** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hWrappingKey, CK_OBJECT_HANDLE hKey, CK_BYTE_PTR pWrappedKey, CK_ULONG_PTR pulWrappedKeyLen)
Wraps (encrypts) the specified key using the specified wrapping key and mechanism.
- **CK_RV C_UnwrapKey** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hUnwrappingKey, CK_BYTE_PTR pWrappedKey, CK_ULONG ulWrappedKeyLen, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulAttributeCount, CK_OBJECT_HANDLE_PTR phKey)
Unwraps (decrypts) the specified key using the specified unwrapping key.

- CK_RV **C_DeriveKey** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_↔
_OBJECT_HANDLE hBaseKey, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulAttributeCount, CK_↔
OBJECT_HANDLE_PTR phKey)
Derive a key from the specified base key.
- CK_RV **C_SeedRandom** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pSeed, CK_ULONG ul↔
SeedLen)
Mixes in additional seed material to the random number generator.
- CK_RV **C_GenerateRandom** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR RandomData, CK_↔
ULONG ulRandomLen)
Generate the specified amount of random data.
- CK_RV **C_GetFunctionStatus** (CK_SESSION_HANDLE hSession)
Legacy function - see PKCS#11 v2.40.
- CK_RV **C_CancelFunction** (CK_SESSION_HANDLE hSession)
Legacy function.
- CK_RV **C_WaitForSlotEvent** (CK_FLAGS flags, CK_SLOT_ID_PTR pSlot, CK_VOID_PTR pRserve↔)
Wait for a slot event (token insertion, removal, etc) on the specified slot to occur.

24.193.1 Detailed Description

PKCS11 Basic library redirects based on the 2.40 specification docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.html.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.194 pkcs11_mech.c File Reference

PKCS11 Library Mechanism Handling.

```
#include "pkcs11_config.h"  
#include "pkcs11_debug.h"  
#include "pkcs11_init.h"  
#include "pkcs11_mech.h"  
#include "pkcs11_slot.h"  
#include "cryptoauthlib.h"
```

Data Structures

- struct [pkcs11_mech_table_e](#)

Macros

- #define **PKCS11_MECH_ECC508_EC_CAPABILITY** (CKF_EC_F_P | CKF_EC_NAMEDCURVE | CKF_↔
EC_UNCOMPRESS)
- #define **TABLE_SIZE**(x) sizeof(x) / sizeof(x[0])

Typedefs

- typedef struct [pkcs11_mech_table_e](#) **pkcs11_mech_table_e**
- typedef struct [pkcs11_mech_table_e](#) * **pkcs11_mech_table_ptr**

Functions

- CK_RV **pkcs11_mech_get_list** (CK_SLOT_ID slotID, CK_MECHANISM_TYPE_PTR pMechanismList, CK_ULONG_PTR pulCount)
- CK_RV **pkcs_mech_get_info** (CK_SLOT_ID slotID, CK_MECHANISM_TYPE type, CK_MECHANISM_↔INFO_PTR pInfo)

24.194.1 Detailed Description

PKCS11 Library Mechanism Handling.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.195 pkcs11_mech.h File Reference

PKCS11 Library Mechanism Handling.

```
#include "cryptoki.h"
```

Functions

- CK_RV **pkcs11_mech_get_list** (CK_SLOT_ID slotID, CK_MECHANISM_TYPE_PTR pMechanismList, CK_ULONG_PTR pulCount)
- CK_RV **pkcs_mech_get_info** (CK_SLOT_ID slotID, CK_MECHANISM_TYPE type, CK_MECHANISM_↔INFO_PTR pInfo)

24.195.1 Detailed Description

PKCS11 Library Mechanism Handling.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.196 pkcs11_object.c File Reference

PKCS11 Library Object Handling Base.

```
#include "cryptoauthlib.h"
#include "atcacert/atcacert_def.h"
#include "cryptoki.h"
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_init.h"
#include "pkcs11_slot.h"
#include "pkcs11_session.h"
#include "pkcs11_util.h"
#include "pkcs11_object.h"
#include "pkcs11_os.h"
#include "pkcs11_find.h"
#include "pkcs11_key.h"
#include "pkcs11_cert.h"
```

Functions

- CK_RV **pkcs11_object_alloc** (CK_SLOT_ID slotId, pkcs11_object_ptr *ppObject)
- CK_RV **pkcs11_object_free** (pkcs11_object_ptr pObject)
- CK_RV **pkcs11_object_check** (pkcs11_object_ptr *ppObject, CK_OBJECT_HANDLE hObject)
- CK_RV **pkcs11_object_get_handle** (pkcs11_object_ptr pObject, CK_OBJECT_HANDLE_PTR phObject)
- CK_RV **pkcs11_object_get_owner** (pkcs11_object_ptr pObject, CK_SLOT_ID_PTR pSlotId)
- CK_RV **pkcs11_object_get_name** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV **pkcs11_object_get_class** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV **pkcs11_object_get_type** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV **pkcs11_object_get_destroyable** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV **pkcs11_object_get_size** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_ULONG_PTR pulSize)
- CK_RV **pkcs11_object_find** (CK_SLOT_ID slotId, pkcs11_object_ptr *ppObject, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount)
- CK_RV **pkcs11_object_create** (CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phObject)
Create a new object on the token in the specified session using the given attribute template.
- CK_RV **pkcs11_object_destroy** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject)
Destroy the specified object.
- CK_RV **pkcs11_object_deinit** (pkcs11_lib_ctx_ptr pContext)
- ATCA_STATUS **pkcs11_object_load_handle_info** ([ATCADevice](#) device, pkcs11_lib_ctx_ptr pContext)
- CK_RV **pkcs11_object_is_private** (pkcs11_object_ptr pObject, CK_BBOOL *is_private, [pkcs11_session_ctx_ptr](#) pSession)

Checks the attributes of the underlying cryptographic asset to determine if it is a private key - this changes the way the associated public key is referenced.

Variables

- `pkcs11_object_cache_t pkcs11_object_cache` [PKCS11_MAX_OBJECTS_ALLOWED]
- const `pkcs11_attr_model pkcs11_object_monotonic_attributes` []
- const CK_ULONG `pkcs11_object_monotonic_attributes_count` = (CK_ULONG)(sizeof(`pkcs11_object_monotonic_attributes`) / sizeof(`pkcs11_object_monotonic_attributes` [0]))

24.196.1 Detailed Description

PKCS11 Library Object Handling Base.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.197 pkcs11_object.h File Reference

PKCS11 Library Object Handling.

```
#include "cryptoauthlib.h"
#include "cryptoki.h"
#include "pkcs11_config.h"
#include "pkcs11_attr.h"
```

Data Structures

- struct `pkcs11_object_s`
- struct `pkcs11_object_cache_s`

Macros

- #define `PKCS11_OBJECT_FLAG_DESTROYABLE` (0x01U)
- #define `PKCS11_OBJECT_FLAG_MODIFIABLE` (0x02U)
- #define `PKCS11_OBJECT_FLAG_DYNAMIC` (0x04U)
- #define `PKCS11_OBJECT_FLAG_SENSITIVE` (0x08U)
- #define `PKCS11_OBJECT_FLAG_TA_TYPE` (0x10U)
- #define `PKCS11_OBJECT_FLAG_TRUST_TYPE` (0x20U)
- #define `PKCS11_OBJECT_FLAG_CERT_CACHE` (0x40U)
- #define `PKCS11_OBJECT_FLAG_KEY_CACHE` (0x80U)
- #define `PKCS11_OBJECT_FLAG_KEY_CACHE_COMPLEMENT` ~(PKCS11_OBJECT_FLAG_KEY_CACHE & 0xffu)
- #define `PKCS11_OBJECT_FLAG_CERT_CACHE_COMPLEMENT` ~(PKCS11_OBJECT_FLAG_CERT_CACHE & 0xffu)

Typedefs

- typedef struct `pkcs11_object_s` `pkcs11_object`
- typedef struct `pkcs11_object_cache_s` `pkcs11_object_cache_t`

Functions

- CK_RV **pkcs11_object_alloc** (CK_SLOT_ID slotId, pkcs11_object_ptr *ppObject)
- CK_RV **pkcs11_object_free** (pkcs11_object_ptr pObject)
- CK_RV **pkcs11_object_check** (pkcs11_object_ptr *ppObject, CK_OBJECT_HANDLE hObject)
- CK_RV **pkcs11_object_find** (CK_SLOT_ID slotId, pkcs11_object_ptr *ppObject, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount)
- CK_RV **pkcs11_object_is_private** (pkcs11_object_ptr pObject, CK_BBOOL *is_private, [pkcs11_session_ctx_ptr](#) pSession)
Checks the attributes of the underlying cryptographic asset to determine if it is a private key - this changes the way the associated public key is referenced.
- CK_RV **pkcs11_object_deinit** (pkcs11_lib_ctx_ptr pContext)
- CK_RV **pkcs11_object_get_owner** (pkcs11_object_ptr pObject, CK_SLOT_ID_PTR pSlotId)
- ATCA_STATUS **pkcs11_object_load_handle_info** ([ATCADevice](#) device, pkcs11_lib_ctx_ptr pContext)
- CK_RV **pkcs11_object_get_class** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV **pkcs11_object_get_name** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV **pkcs11_object_get_type** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV **pkcs11_object_get_destroyable** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV **pkcs11_object_get_size** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_ULONG_PTR pulSize)
- CK_RV **pkcs11_object_get_handle** (pkcs11_object_ptr pObject, CK_OBJECT_HANDLE_PTR phObject)
- CK_RV **pkcs11_object_create** (CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phObject)
Create a new object on the token in the specified session using the given attribute template.
- CK_RV **pkcs11_object_destroy** (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject)
Destroy the specified object.

Variables

- [pkcs11_object_cache_t](#) **pkcs11_object_cache** []
- const [pkcs11_attr_model](#) **pkcs11_object_monotonic_attributes** []
- const CK_ULONG **pkcs11_object_monotonic_attributes_count**

24.197.1 Detailed Description

PKCS11 Library Object Handling.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.198 pkcs11_os.c File Reference

PKCS11 Library Operating System Abstraction Functions.

```
#include "pkcs11_os.h"
#include "pkcs11_util.h"
#include "pkcs11_init.h"
```

Functions

- CK_RV [pkcs11_os_create_mutex](#) (CK_VOID_PTR_PTR ppMutex)
Application callback for creating a mutex object.
- CK_RV [pkcs11_os_destroy_mutex](#) (CK_VOID_PTR pMutex)
- CK_RV [pkcs11_os_lock_mutex](#) (CK_VOID_PTR pMutex)
- CK_RV [pkcs11_os_unlock_mutex](#) (CK_VOID_PTR pMutex)
- CK_RV [pkcs11_os_alloc_shared_ctx](#) (void **ppShared, size_t size)
- CK_RV [pkcs11_os_free_shared_ctx](#) (void *pShared, size_t size)

24.198.1 Detailed Description

PKCS11 Library Operating System Abstraction Functions.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.199 pkcs11_os.h File Reference

PKCS11 Library Operating System Abstraction.

```
#include "cryptoki.h"
#include "cryptoauthlib.h"
```

Macros

- `#define ATCA_SHARED_MUTEX_NAME "atca_shared_mutex"`
- `#define pkcs11_os_malloc hal_malloc`
- `#define pkcs11_os_free hal_free`

Functions

- CK_RV [pkcs11_os_create_mutex](#) (CK_VOID_PTR_PTR ppMutex)
Application callback for creating a mutex object.
- CK_RV [pkcs11_os_destroy_mutex](#) (CK_VOID_PTR pMutex)
- CK_RV [pkcs11_os_lock_mutex](#) (CK_VOID_PTR pMutex)
- CK_RV [pkcs11_os_unlock_mutex](#) (CK_VOID_PTR pMutex)
- CK_RV [pkcs11_os_alloc_shared_ctx](#) (void **ppShared, size_t size)
- CK_RV [pkcs11_os_free_shared_ctx](#) (void *pShared, size_t size)

24.199.1 Detailed Description

PKCS11 Library Operating System Abstraction.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.200 pkcs11_session.c File Reference

PKCS11 Library Session Handling.

```
#include "cryptoauthlib.h"
#include "host/atca_host.h"
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_session.h"
#include "pkcs11_token.h"
#include "pkcs11_init.h"
#include "pkcs11_slot.h"
#include "pkcs11_object.h"
#include "pkcs11_os.h"
#include "pkcs11_util.h"
#include "pkcs11_key.h"
#include "pkcs11_cert.h"
```

Functions

- [pkcs11_session_ctx_ptr](#) **pkcs11_get_session_context** (CK_SESSION_HANDLE hSession)
- CK_RV **pkcs11_session_check** ([pkcs11_session_ctx_ptr](#) *pSession, CK_SESSION_HANDLE hSession)
Check if the session is initialized properly.
- CK_RV **pkcs11_reserve_resource** (pkcs11_lib_ctx_ptr pContext, [pkcs11_session_ctx_ptr](#) pSession, uint8_t resource)
- CK_RV **pkcs11_release_resource** (pkcs11_lib_ctx_ptr pContext, [pkcs11_session_ctx_ptr](#) pSession, uint8_t resource)
- CK_RV **pkcs11_session_open** (CK_SLOT_ID slotID, CK_FLAGS flags, CK_VOID_PTR pApplication, CK_NOTIFY notify, CK_SESSION_HANDLE_PTR phSession)
- CK_RV **pkcs11_session_close** (CK_SESSION_HANDLE hSession)
- CK_RV [pkcs11_session_closeall](#) (CK_SLOT_ID slotID)
Close all sessions for a given slot - not actually all open sessions.
- CK_RV **pkcs11_session_get_info** (CK_SESSION_HANDLE hSession, CK_SESSION_INFO_PTR pInfo)
Obtains information about a particular session.
- CK_RV [pkcs11_session_login](#) (CK_SESSION_HANDLE hSession, CK_USER_TYPE userType, CK_UTF8CHAR_PTR pPin, CK_ULONG ulPinLen)
- CK_RV **pkcs11_session_logout** (CK_SESSION_HANDLE hSession)

24.200.1 Detailed Description

PKCS11 Library Session Handling.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.201 pkcs11_session.h File Reference

PKCS11 Library Session Management & Context.

```
#include "cryptoki.h"
#include "pkcs11_config.h"
#include "cal_internal.h"
```

Data Structures

- struct [pkcs11_session_mech_ctx_s](#)
- struct [pkcs11_session_ctx_s](#)

Typedefs

- typedef struct [pkcs11_session_mech_ctx_s](#) **pkcs11_session_mech_ctx**
- typedef struct [pkcs11_session_mech_ctx_s](#) * **pkcs11_session_mech_ctx_ptr**
- typedef struct [pkcs11_session_ctx_s](#) **pkcs11_session_ctx**
- typedef struct [pkcs11_session_ctx_s](#) * **pkcs11_session_ctx_ptr**

Functions

- [pkcs11_session_ctx_ptr](#) **pkcs11_get_session_context** (CK_SESSION_HANDLE hSession)
- CK_RV **pkcs11_session_check** ([pkcs11_session_ctx_ptr](#) *pSession, CK_SESSION_HANDLE hSession)
Check if the session is initialized properly.
- CK_RV **pkcs11_session_get_info** (CK_SESSION_HANDLE hSession, CK_SESSION_INFO_PTR pInfo)
Obtains information about a particular session.
- CK_RV **pkcs11_session_open** (CK_SLOT_ID slotID, CK_FLAGS flags, CK_VOID_PTR pApplication, CK_NOTIFY notify, CK_SESSION_HANDLE_PTR phSession)
- CK_RV **pkcs11_session_close** (CK_SESSION_HANDLE hSession)
- CK_RV [pkcs11_session_closeall](#) (CK_SLOT_ID slotID)
Close all sessions for a given slot - not actually all open sessions.
- CK_RV [pkcs11_session_login](#) (CK_SESSION_HANDLE hSession, CK_USER_TYPE userType, CK_UTF8CHAR_PTR pPin, CK_ULONG ulPinLen)
- CK_RV **pkcs11_session_logout** (CK_SESSION_HANDLE hSession)
- CK_RV **pkcs11_reserve_resource** ([pkcs11_lib_ctx_ptr](#) pContext, [pkcs11_session_ctx_ptr](#) pSession, uint8_t resource)
- CK_RV **pkcs11_release_resource** ([pkcs11_lib_ctx_ptr](#) pContext, [pkcs11_session_ctx_ptr](#) pSession, uint8_t resource)

24.201.1 Detailed Description

PKCS11 Library Session Management & Context.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.201.2 Typedef Documentation

24.201.2.1 [pkcs11_session_ctx](#)

```
typedef struct pkcs11\_session\_ctx\_s pkcs11\_session\_ctx
```

Session Context

24.202 pkcs11_signature.c File Reference

PKCS11 Library Sign/Verify Handling.

```
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_init.h"
#include "pkcs11_signature.h"
#include "pkcs11_object.h"
#include "pkcs11_session.h"
#include "pkcs11_util.h"
#include "cryptoauthlib.h"
#include "pkcs11_slot.h"
#include "pkcs11_key.h"
#include "atcacert/atcacert_der.h"
```

Functions

- CK_RV **pkcs11_signature_sign_init** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey)
Initialize a signing operation using the specified key and mechanism.
- CK_RV **pkcs11_signature_sign** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG_PTR pulSignatureLen)
Sign the data in a single pass operation.
- CK_RV **pkcs11_signature_sign_continue** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen)
Continues a multiple-part signature operation.
- CK_RV **pkcs11_signature_sign_finish** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pSignature, CK_ULONG_PTR pulSignatureLen)
Finishes a multiple-part signature operation.
- CK_RV **pkcs11_signature_verify_init** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey)
Initializes a verification operation using the specified key and mechanism.
- CK_RV **pkcs11_signature_verify** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG ulSignatureLen)
Verifies a signature on single-part data.
- CK_RV **pkcs11_signature_verify_continue** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen)
Continues a multiple-part verification operation.
- CK_RV **pkcs11_signature_verify_finish** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pSignature, CK_ULONG ulSignatureLen)
Finishes a multiple-part verification operation.

24.202.1 Detailed Description

PKCS11 Library Sign/Verify Handling.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.203 pkcs11_signature.h File Reference

PKCS11 Library Sign/Verify Handling.

```
#include "cryptoki.h"
```

Functions

- CK_RV **pkcs11_signature_sign_init** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey)
Initialize a signing operation using the specified key and mechanism.
- CK_RV **pkcs11_signature_sign** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG_PTR pulSignatureLen)
Sign the data in a single pass operation.
- CK_RV **pkcs11_signature_sign_continue** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen)
Continues a multiple-part signature operation.
- CK_RV **pkcs11_signature_sign_finish** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pSignature, CK_ULONG_PTR pulSignatureLen)
Finishes a multiple-part signature operation.
- CK_RV **pkcs11_signature_verify_init** (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey)
Initializes a verification operation using the specified key and mechanism.
- CK_RV **pkcs11_signature_verify** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG ulSignatureLen)
Verifies a signature on single-part data.
- CK_RV **pkcs11_signature_verify_continue** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen)
Continues a multiple-part verification operation.
- CK_RV **pkcs11_signature_verify_finish** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pSignature, CK_ULONG ulSignatureLen)
Finishes a multiple-part verification operation.

24.203.1 Detailed Description

PKCS11 Library Sign/Verify Handling.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.204 pkcs11_slot.c File Reference

PKCS11 Library Slot Handling.

```
#include "cryptoauthlib.h"
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_init.h"
#include "pkcs11_slot.h"
#include "pkcs11_info.h"
#include "pkcs11_util.h"
#include "pkcs11_object.h"
#include "pkcs11_os.h"
#include <stdio.h>
```

Functions

- pkcs11_slot_ctx_ptr **pkcs11_slot_get_context** (pkcs11_lib_ctx_ptr lib_ctx, CK_SLOT_ID slotID)
Retrieve the current slot context.
- pkcs11_slot_ctx_ptr **pkcs11_slot_get_new_context** (pkcs11_lib_ctx_ptr lib_ctx)
- CK_VOID_PTR **pkcs11_slot_initslots** (CK_ULONG pulCount)
- CK_RV **pkcs11_slot_deinitslots** (pkcs11_lib_ctx_ptr lib_ctx)
- CK_RV **pkcs11_slot_config** (CK_SLOT_ID slotID)
- CK_RV **pkcs11_slot_init** (CK_SLOT_ID slotID)
This is an internal function that initializes a pkcs11 slot - it must already have the locks in place before being called.
- CK_RV **pkcs11_slot_get_list** (CK_BBOOL tokenPresent, CK_SLOT_ID_PTR pSlotList, CK_ULONG_PTR pulCount)
- CK_RV **pkcs11_slot_get_info** (CK_SLOT_ID slotID, CK_SLOT_INFO_PTR pInfo)
Obtains information about a particular slot.

24.204.1 Detailed Description

PKCS11 Library Slot Handling.

The nomenclature here can lead to some confusion - the pkcs11 slot is not the same as a device slot. So for example each slot defined here is a specific device (most systems would have only one). The "slots" as defined by the device specification would be enumerated separately as related to specific supported mechanisms as cryptographic "objects".

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.205 pkcs11_slot.h File Reference

PKCS11 Library Slot Handling & Context.

```
#include "pkcs11_init.h"
#include "cryptoauthlib.h"
```

Data Structures

- struct [pkcs11_slot_ctx_s](#)

Macros

- #define **SLOT_STATE_UNINITIALIZED** (0U)
- #define **SLOT_STATE_CONFIGURED** (1U)
- #define **SLOT_STATE_READY** (2U)

Typedefs

- typedef struct [pkcs11_slot_ctx_s](#) [pkcs11_slot_ctx](#)

Functions

- CK_RV **pkcs11_slot_init** (CK_SLOT_ID slotID)
This is an internal function that initializes a pkcs11 slot - it must already have the locks in place before being called.
- CK_RV **pkcs11_slot_config** (CK_SLOT_ID slotID)
- CK_VOID_PTR **pkcs11_slot_initslots** (CK_ULONG pulCount)
- CK_RV **pkcs11_slot_deinitslots** (pkcs11_lib_ctx_ptr lib_ctx)
- pkcs11_slot_ctx_ptr **pkcs11_slot_get_context** (pkcs11_lib_ctx_ptr lib_ctx, CK_SLOT_ID slotID)
Retrieve the current slot context.
- pkcs11_slot_ctx_ptr **pkcs11_slot_get_new_context** (pkcs11_lib_ctx_ptr lib_ctx)
- CK_RV **pkcs11_slot_get_list** (CK_BBOOL tokenPresent, CK_SLOT_ID_PTR pSlotList, CK_ULONG_PTR pulCount)
- CK_RV **pkcs11_slot_get_info** (CK_SLOT_ID slotID, CK_SLOT_INFO_PTR pInfo)
Obtains information about a particular slot.

24.205.1 Detailed Description

PKCS11 Library Slot Handling & Context.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.205.2 Typedef Documentation

24.205.2.1 pkcs11_slot_ctx

```
typedef struct pkcs11_slot_ctx_s pkcs11_slot_ctx
```

Slot Context

24.206 pkcs11_token.c File Reference

PKCS11 Library Token Handling.

```
#include "cryptoauthlib.h"
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_token.h"
#include "pkcs11_slot.h"
#include "pkcs11_info.h"
#include "pkcs11_util.h"
#include "pkcs11_object.h"
#include "pkcs11_key.h"
#include "pkcs11_cert.h"
#include "pkcs11_session.h"
```

Functions

- CK_RV **pkcs11_token_init** (CK_SLOT_ID slotID, CK_UTF8CHAR_PTR pPin, CK_ULONG ulPinLen, CK_UTF8CHAR_PTR pLabel)
- CK_RV **pkcs11_token_get_access_type** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV **pkcs11_token_get_writable** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV **pkcs11_token_get_storage** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV **pkcs11_token_get_info** (CK_SLOT_ID slotID, CK_TOKEN_INFO_PTR pInfo)
Obtains information about a particular token.
- CK_RV **pkcs11_token_random** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pRandomData, CK_ULONG ulRandomLen)
Generate the specified amount of random data.
- CK_RV **pkcs11_token_convert_pin_to_key** (const CK_UTF8CHAR_PTR pPin, const CK_ULONG ulPinLen, const CK_UTF8CHAR_PTR pSalt, const CK_ULONG ulSaltLen, CK_BYTE_PTR pKey, CK_ULONG ulKeyLen, [pkcs11_slot_ctx_ptr](#) slot_ctx)
- CK_RV **pkcs11_token_set_pin** (CK_SESSION_HANDLE hSession, CK_UTF8CHAR_PTR pOldPin, CK_ULONG ulOldLen, CK_UTF8CHAR_PTR pNewPin, CK_ULONG ulNewLen)

24.206.1 Detailed Description

PKCS11 Library Token Handling.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.207 pkcs11_token.h File Reference

PKCS11 Library Token Management & Context.

```
#include "pkcs11_init.h"
#include "pkcs11_session.h"
```

Macros

- **#define ATCA_SERIAL_NUM_SIZE** (9)

Functions

- CK_RV **pkcs11_token_init** (CK_SLOT_ID slotID, CK_UTF8CHAR_PTR pPin, CK_ULONG ulPinLen, CK_UTF8CHAR_PTR pLabel)
- CK_RV **pkcs11_token_get_access_type** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV **pkcs11_token_get_writable** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV **pkcs11_token_get_storage** (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute, [pkcs11_session_ctx_ptr](#) pSession)
- CK_RV **pkcs11_token_get_info** (CK_SLOT_ID slotID, CK_TOKEN_INFO_PTR pInfo)
Obtains information about a particular token.
- CK_RV **pkcs11_token_convert_pin_to_key** (const CK_UTF8CHAR_PTR pPin, const CK_ULONG ulPinLen, const CK_UTF8CHAR_PTR pSalt, const CK_ULONG ulSaltLen, CK_BYTE_PTR pKey, CK_ULONG ulKeyLen, [pkcs11_slot_ctx_ptr](#) slot_ctx)
- CK_RV **pkcs11_token_random** (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pRandomData, CK_ULONG ulRandomLen)
Generate the specified amount of random data.
- CK_RV **pkcs11_token_set_pin** (CK_SESSION_HANDLE hSession, CK_UTF8CHAR_PTR pOldPin, CK_ULONG ulOldLen, CK_UTF8CHAR_PTR pNewPin, CK_ULONG ulNewLen)

24.207.1 Detailed Description

PKCS11 Library Token Management & Context.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.208 pkcs11_util.c File Reference

PKCS11 Library Utility Functions.

```
#include "pkcs11_util.h"
```

Functions

- void **pkcs11_util_escape_string** (CK_UTF8CHAR_PTR buf, CK_ULONG buf_len)
- CK_RV **pkcs11_util_convert_rv** (ATCA_STATUS status)
- int **pkcs11_util_memset** (void *dest, size_t destsz, int ch, size_t count)

24.208.1 Detailed Description

PKCS11 Library Utility Functions.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.209 pkcs11_util.h File Reference

PKCS11 Library Utilities.

```
#include "pkcs11_config.h"
#include "cryptoki.h"
#include "cryptoauthlib.h"
```

Macros

- `#define PKCS11_UTIL_ARRAY_SIZE(x) sizeof(x) / sizeof(x[0])`

Functions

- void **pkcs11_util_escape_string** (CK_UTF8CHAR_PTR buf, CK_ULONG buf_len)
- CK_RV **pkcs11_util_convert_rv** (ATCA_STATUS status)
- int **pkcs11_util_memset** (void *dest, size_t destsz, int ch, size_t count)

24.209.1 Detailed Description

PKCS11 Library Utilities.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.210 atca_wolfssl_interface.c File Reference

Crypto abstraction functions for external host side cryptography.

```
#include "cryptoauthlib.h"
```

24.210.1 Detailed Description

Crypto abstraction functions for external host side cryptography.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.211 atca_wolfssl_interface.h File Reference

Configuration Check for WolfSSL Integration Support.

```
#include "atca_config_check.h"
```

24.211.1 Detailed Description

Configuration Check for WolfSSL Integration Support.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

24.212 atca_wolfssl_internal.h File Reference

WolfSSL Integration Support.

24.212.1 Detailed Description

WolfSSL Integration Support.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

Index

- `_array_to_code`
 - `cryptoauthlib.library`, 336
 - `_ascii_kit_host_context`, 355
 - `_atcacert_convert_bytes`
 - `cryptoauthlib.atcacert`, 326
 - `_atcacert_convert_enum`
 - `cryptoauthlib.atcacert`, 326
 - `_check_type_rationality`
 - `cryptoauthlib.library`, 336
 - `_convert_pointer_to_list`
 - `cryptoauthlib.library`, 336
 - `_ctype_from_definition`
 - `cryptoauthlib.library`, 336
 - `_def_`
 - `cryptoauthlib.atcacert.atcacert_cert_element_t`, 397
 - `cryptoauthlib.atcacert.atcacert_device_loc_t`, 404
 - `cryptoauthlib.iface._ATCAHID`, 357
 - `cryptoauthlib.iface._ATCAKIT`, 360
 - `cryptoauthlib.iface._ATCAUART`, 363
 - `cryptoauthlib.iface.ATCAIfaceCfg`, 414
 - `_def_to_field`
 - `cryptoauthlib.library`, 337
 - `_fields_`
 - `cryptoauthlib.atcab.atca_aes_cbc_ctx`, 367
 - `cryptoauthlib.atcab.atca_aes_ccmac_ctx`, 368
 - `cryptoauthlib.atcab.atca_aes_ccm_ctx`, 368
 - `cryptoauthlib.atcab.atca_aes_cmac_ctx`, 369
 - `cryptoauthlib.atcab.atca_aes_ctr_ctx`, 370
 - `cryptoauthlib.atcab.atca_aes_gcm_ctx`, 371
 - `cryptoauthlib.atcab.atca_sha256_ctx`, 385
 - `cryptoauthlib.atcacert.atcacert_comp_data_t`, 401
 - `cryptoauthlib.atcacert.atcacert_tm_utc_t`, 407
 - `cryptoauthlib.device.AesEnable`, 366
 - `cryptoauthlib.device.Atecc508aConfig`, 421
 - `cryptoauthlib.device.Atecc608Config`, 423
 - `cryptoauthlib.device.Atsha204aConfig`, 425
 - `cryptoauthlib.device.ChipMode508`, 430
 - `cryptoauthlib.device.ChipMode608`, 430
 - `cryptoauthlib.device.ChipOptions`, 431
 - `cryptoauthlib.device.Counter204`, 448
 - `cryptoauthlib.device.CountMatch`, 448
 - `cryptoauthlib.device.I2cEnable`, 456
 - `cryptoauthlib.device.KeyConfig`, 457
 - `cryptoauthlib.device.SecureBoot`, 470
 - `cryptoauthlib.device.SlotConfig`, 471
 - `cryptoauthlib.device.UseLock`, 475
 - `cryptoauthlib.device.VolatileKeyPermission`, 476
 - `cryptoauthlib.device.X509Format`, 477
 - `cryptoauthlib.iface._ATCACUSTOM`, 356
 - `cryptoauthlib.iface._ATCAI2C`, 358
 - `cryptoauthlib.iface._ATCAIfaceParams`, 359
 - `cryptoauthlib.iface._ATCASPI`, 361
 - `cryptoauthlib.iface._ATCASWI`, 362
 - `cryptoauthlib.iface._U_Address`, 365
 - `cryptoauthlib.library.cal_buffer`, 427
 - `_force_local_library`
 - `cryptoauthlib.library`, 337
 - `_get_attribute_from_ctypes`
 - `cryptoauthlib.library`, 337
 - `_get_field_definition`
 - `cryptoauthlib.library`, 337
 - `_iface_load_default_config`
 - `cryptoauthlib.iface`, 333
 - `_is_pointer`
 - `cryptoauthlib.library`, 338
 - `_kit_host_map_entry`, 364
 - `_map_`
 - `cryptoauthlib.iface._ATCAI2C`, 358
 - `cryptoauthlib.iface.ATCAIfaceCfg`, 414
 - `_obj_to_code`
 - `cryptoauthlib.library`, 338
 - `_object_definition_code`
 - `cryptoauthlib.library`, 338
 - `_pointer_to_code`
 - `cryptoauthlib.library`, 338
 - `_structure_to_code`
 - `cryptoauthlib.library`, 339
 - `_structure_to_string`
 - `cryptoauthlib.library`, 339
 - `_to_code`
 - `cryptoauthlib.library`, 339
- `address`
 - `atca_iface.h`, 556
 - `ATCAIfaceCfg`, 413
- `api_206a.c`, 479
 - `sha206a_authenticate`, 480
 - `sha206a_check_dk_useflag_validity`, 480
 - `sha206a_check_pk_useflag_validity`, 481
 - `sha206a_diversify_parent_key`, 481
 - `sha206a_generate_challenge_response_pair`, 481
 - `sha206a_generate_derive_key`, 482
 - `sha206a_get_data_store_lock_status`, 482
 - `sha206a_get_dk_update_count`, 483
 - `sha206a_get_dk_useflag_count`, 483
 - `sha206a_get_pk_useflag_count`, 483
 - `sha206a_read_data_store`, 484
 - `sha206a_verify_device_consumption`, 484

- sha206a_write_data_store, 485
- api_206a.h, 485
 - sha206a_authenticate, 486
 - sha206a_check_dk_useflag_validity, 487
 - sha206a_check_pk_useflag_validity, 487
 - sha206a_diversify_parent_key, 487
 - sha206a_generate_challenge_response_pair, 488
 - sha206a_generate_derive_key, 488
 - sha206a_get_data_store_lock_status, 489
 - sha206a_get_dk_update_count, 489
 - sha206a_get_dk_useflag_count, 489
 - sha206a_get_pk_useflag_count, 490
 - sha206a_read_data_store, 490
 - sha206a_verify_device_consumption, 491
 - sha206a_write_data_store, 491
- ascii_kit_host.c, 494
 - kit_host_init, 494
 - kit_host_init_phy, 495
- ascii_kit_host.h, 495
 - kit_host_init, 497
 - kit_host_init_phy, 498
 - kit_host_map_entry_t, 497
 - KIT_MESSAGE_SIZE_MAX, 496
- ATCA_ALLOC_FAILURE
 - atca_status.h, 558
- ATCA_ASSERT_FAILURE
 - atca_status.h, 558
- ATCA_BAD_OPCODE
 - atca_status.h, 558
- ATCA_BAD_PARAM
 - atca_status.h, 558
- atca_basic.c, 515
- atca_basic.h, 523
- atca_cfgs.c, 532
- atca_cfgs.h, 532
- ATCA_CHECK_INVALID_MSG
 - atca_config_check.h, 535
- atca_check_mac_in_out, 371
 - slot_key, 372
 - target_key, 372
- ATCA_CHECKMAC_VERIFY_FAILED
 - atca_status.h, 558
- ATCA_COMM_FAIL
 - atca_status.h, 558
- atca_compiler.h, 533
 - UNUSED_VAR, 533
- atca_config_check.h, 533
 - ATCA_CHECK_INVALID_MSG, 535
 - ATCA_UNUSED_VAR_CHECK, 535
 - ATCA_USE_ATCAB_FUNCTIONS, 535
 - ATCAB_AES_GFM_EN, 535
 - ATCAB_GENKEY_MAC_EN, 536
 - ATCAB_INFO_LATCH_EN, 536
 - ATCAB_VERIFY_MAC_EN, 536
 - ATCAB_WRITE_EN, 536
 - ATCAC_CERT_ADD_EN, 536
 - ATCAC_RANDOM_EN, 537
 - ATCAC_SHA1_EN, 537
 - ATCAC_SHA256_EN, 537
 - ATCAC_SHA384_EN, 537
 - ATCAC_SHA512_EN, 537
 - ATCAC_SIGN_EN, 537
 - ATCAC_VERIFY_EN, 538
 - LIBRARY_BUILD_EN_CHECK, 538
 - MULTIPART_BUF_EN, 538
- ATCA_CRYPT_AES_CMEN
 - crypto_sw_config_check.h, 649
- ATCA_CRYPT_AES_GCMEN
 - crypto_sw_config_check.h, 649
- atca_crypto_hw_aes.h, 639
- atca_crypto_hw_aes_cbc.c, 640
- atca_crypto_hw_aes_ccmac.c, 640
- atca_crypto_hw_aes_ccm.c, 641
- atca_crypto_hw_aes_cmec.c, 641
- atca_crypto_hw_aes_ctr.c, 642
- atca_crypto_pad.c, 642
- atca_crypto_pbkdf2.c, 642
- ATCA_CRYPT_SHA1_EN
 - crypto_sw_config_check.h, 650
- ATCA_CRYPT_SHA256_EN
 - crypto_sw_config_check.h, 650
- ATCA_CRYPT_SHA2_EN
 - crypto_sw_config_check.h, 650
- ATCA_CRYPT_SHA2_HMAC_CTR_EN
 - crypto_sw_config_check.h, 650
- ATCA_CRYPT_SHA2_HMAC_EN
 - crypto_sw_config_check.h, 650
- ATCA_CRYPT_SHA384_EN
 - crypto_sw_config_check.h, 650
- ATCA_CRYPT_SHA512_EN
 - crypto_sw_config_check.h, 651
- atca_crypto_sw.h, 643
- atca_crypto_sw_aes_cmec.c, 643
- atca_crypto_sw_aes_gcm.c, 644
- atca_crypto_sw_sha1.c, 644
- atca_crypto_sw_sha1.h, 645
- atca_crypto_sw_sha2.c, 645
- atca_crypto_sw_sha2.h, 645
- ATCA_CUSTOM_IFACE
 - ATCAIface (atca_), 176
- atca_debug.c, 538
- atca_decrypt_in_out, 372
- atca_delay_10us
 - Hardware abstraction layer (hal_), 222
- atca_delay_ms
 - Hardware abstraction layer (hal_), 222
- atca_delay_us
 - Hardware abstraction layer (hal_), 223
- atca_delete_in_out, 372
- atca_derive_key_in_out, 373
- atca_derive_key_mac_in_out, 373
- atca_device, 374
 - device_state, 374
 - mlface, 374
- atca_device.c, 539
- atca_device.h, 539

- atca_devtypes.h, 540
- atca_diversified_key_in_out, 375
- atca_evp_ctx, 375
- ATCA_EXECUTION_ERROR
 - atca_status.h, 559
- ATCA_FUNC_FAIL
 - atca_status.h, 559
- atca_gen_dig_in_out, 375
- ATCA_GEN_FAIL
 - atca_status.h, 559
- atca_gen_key_in_out, 376
- atca_hal.c, 659
- atca_hal.h, 659
- atca_hal_kit_phy_t, 377
 - hal_data, 377
 - packet_alloc, 377
 - packet_free, 377
 - recv, 377
 - send, 377
- atca_hal_list_entry_t, 378
 - phy, 378
- atca_hal_shm_t, 378
- ATCA_HEALTH_TEST_ERROR
 - atca_status.h, 559
- atca_helpers.c, 541
 - atcab_base64decode, 542
 - atcab_base64decode_, 543
 - atcab_base64encode, 543
 - atcab_base64encode_, 544
 - atcab_bin2hex, 544
 - atcab_bin2hex_, 545
 - atcab_hex2bin, 545
 - atcab_reversal, 546
 - isAlpha, 546
 - isBase64, 547
 - isBase64Digit, 547
 - isBlankSpace, 547
 - isDigit, 548
 - isHex, 548
 - isHexAlpha, 548
 - isHexDigit, 549
 - packHex, 549
- atca_helpers.h, 550
- ATCA_HID_IFACE
 - ATCAIface (atca_), 176
- atca_hmac_in_out, 378
- atca_host.c, 703
- atca_host.h, 703
- atca_host_config_check.h, 706
 - ATCAH_CHECK_MAC, 707
 - ATCAH_CONFIG_TO_SIGN_INTERNAL, 707
 - ATCAH_DECRYPT, 708
 - ATCAH_DELETE_MAC, 708
 - ATCAH_DERIVE_KEY, 708
 - ATCAH_DERIVE_KEY_MAC, 708
 - ATCAH_ENCODE_COUNTER_MATCH, 708
 - ATCAH_GEN_KEY_MSG, 709
 - ATCAH_GEN_MAC, 709
 - ATCAH_GEN_OUTPUT_RESP_MAC, 709
 - ATCAH_GEN_SESSION_KEY, 709
 - ATCAH_GENDIG, 709
 - ATCAH_GENDIVKEY, 710
 - ATCAH_HMAC, 710
 - ATCAH_INCLUDE_DATA, 710
 - ATCAH_IO_DECRYPT, 710
 - ATCAH_MAC, 710
 - ATCAH_NONCE, 711
 - ATCAH_PRIVWRITE_AUTH_MAC, 711
 - ATCAH_SECUREBOOT_ENC, 711
 - ATCAH_SECUREBOOT_MAC, 711
 - ATCAH_SHA256, 711
 - ATCAH_SIGN_INTERNAL_MSG, 712
 - ATCAH_VERIFY_MAC, 712
 - ATCAH_WRITE_AUTH_MAC, 712
- ATCA_I2C_GPIO_IFACE
 - ATCAIface (atca_), 176
- atca_i2c_host_s, 380
- ATCA_I2C_IFACE
 - ATCAIface (atca_), 176
- atca_iface, 380
 - hal, 380
 - hal_data, 380
 - mlfaceCFG, 380
 - phy, 380
- atca_iface.c, 551
- atca_iface.h, 553
 - address, 556
- atca_iface_is_kit
 - ATCAIface (atca_), 176
- atca_iface_is_swi
 - ATCAIface (atca_), 176
- atca_include_data_in_out, 381
- ATCA_INVALID_ID
 - atca_status.h, 559
- ATCA_INVALID_SIZE
 - atca_status.h, 559
- atca_io_decrypt_in_out, 381
- atca_jwt.c, 712
- atca_jwt.h, 713
- ATCA_KIT_IFACE
 - ATCAIface (atca_), 176
- atca_mac_in_out, 381
- atca_mbedtls_eckey_s, 382
- atca_mbedtls_eckey_t
 - mbedtls Wrapper methods (atca_mbedtls_), 259
- atca_mbedtls_interface.h, 713
 - ATCAC_AES_CMAC_EN, 714
 - ATCAC_AES_GCM_EN, 714
 - ATCAC_PKEY_EN, 714
 - ATCAC_SHA1_EN, 714
 - ATCAC_SHA256_EN, 714
 - ATCAC_SHA384_EN, 714
 - ATCAC_SHA512_EN, 715
 - HOSTLIB_CERT_EN, 715
- atca_mbedtls_pk_init
 - mbedtls Wrapper methods (atca_mbedtls_), 259

- atca_mbedtls_pk_init_ext
 - mbedtls Wrapper methods (atca_mbedtls_), 260
- atca_mbedtls_random_ctx
 - mbedtls Wrapper methods (atca_mbedtls_), 260
- atca_mbedtls_wrap.c, 715
 - atcac_aes_cmac_finish, 717
 - atcac_aes_cmac_init, 718
 - atcac_aes_cmac_update, 718
 - atcac_aes_gcm_aad_update, 719
 - atcac_aes_gcm_decrypt_finish, 719
 - atcac_aes_gcm_decrypt_start, 720
 - atcac_aes_gcm_decrypt_update, 720
 - atcac_aes_gcm_encrypt_finish, 721
 - atcac_aes_gcm_encrypt_start, 721
 - atcac_aes_gcm_encrypt_update, 722
 - atcac_pk_derive, 722
 - atcac_pk_free, 722
 - atcac_pk_init, 723
 - atcac_pk_init_pem, 723
 - atcac_pk_public, 724
 - atcac_pk_sign, 724
 - atcac_pk_verify, 724
 - atcac_sha256_hmac_finish, 724
 - atcac_sha256_hmac_init, 725
 - atcac_sha256_hmac_update, 725
 - atcac_sw_random, 726
 - atcac_sw_sha1_finish, 726
 - atcac_sw_sha1_init, 727
 - atcac_sw_sha1_update, 727
- ATCA_NO_DEVICES
 - atca_status.h, 559
- atca_nonce_in_out, 382
- ATCA_NOT_INITIALIZED
 - atca_status.h, 560
- ATCA_NOT_LOCKED
 - atca_status.h, 560
- atca_openssl_interface.c, 727
 - atcac_aes_cmac_finish, 729
 - atcac_aes_cmac_init, 730
 - atcac_aes_cmac_update, 730
 - atcac_aes_gcm_aad_update, 731
 - atcac_aes_gcm_decrypt_finish, 731
 - atcac_aes_gcm_decrypt_start, 732
 - atcac_aes_gcm_decrypt_update, 732
 - atcac_aes_gcm_encrypt_finish, 733
 - atcac_aes_gcm_encrypt_start, 733
 - atcac_aes_gcm_encrypt_update, 734
 - atcac_pk_derive, 734
 - atcac_pk_free, 734
 - atcac_pk_init, 735
 - atcac_pk_init_pem, 735
 - atcac_pk_public, 736
 - atcac_pk_sign, 736
 - atcac_pk_verify, 736
 - atcac_sha256_hmac_finish, 736
 - atcac_sha256_hmac_init, 737
 - atcac_sha256_hmac_update, 737
 - atcac_sw_random, 738
 - atcac_sw_sha1_finish, 738
 - atcac_sw_sha1_init, 739
 - atcac_sw_sha1_update, 739
- atca_openssl_interface.h, 739
 - ATCAC_AES_CMAC_EN, 740
 - ATCAC_AES_GCM_EN, 740
 - ATCAC_PKEY_EN, 741
 - ATCAC_SHA1_EN, 741
 - ATCAC_SHA256_EN, 741
 - ATCAC_SHA384_EN, 741
 - ATCAC_SHA512_EN, 741
 - HOSTLIB_CERT_EN, 741
- ATCA_PARITY_ERROR
 - atca_status.h, 560
- ATCA_PARSE_ERROR
 - atca_status.h, 560
- atca_platform.h, 556
- atca_resp_mac_in_out, 383
- ATCA_RESYNC_WITH_WAKEUP
 - atca_status.h, 560
- ATCA_RX_CRC_ERROR
 - atca_status.h, 560
- ATCA_RX_FAIL
 - atca_status.h, 560
- ATCA_RX_NO_RESPONSE
 - atca_status.h, 560
- ATCA_RX_TIMEOUT
 - atca_status.h, 561
- atca_secureboot_enc_in_out, 384
- atca_secureboot_mac_in_out, 384
- atca_session_key_in_out, 384
- ATCA_SHA256_BLOCK_SIZE
 - cryptoauthlib.h, 658
- atca_sha256_ctx, 385
- atca_sign_internal_in_out, 386
- ATCA_SMALL_BUFFER
 - atca_status.h, 561
- ATCA_SPI_GPIO_IFACE
 - ATCAIface (atca_), 176
- atca_spi_host_s, 386
- ATCA_SPI_IFACE
 - ATCAIface (atca_), 176
- atca_status.h, 557
 - ATCA_ALLOC_FAILURE, 558
 - ATCA_ASSERT_FAILURE, 558
 - ATCA_BAD_OPCODE, 558
 - ATCA_BAD_PARAM, 558
 - ATCA_CHECKMAC_VERIFY_FAILED, 558
 - ATCA_COMM_FAIL, 558
 - ATCA_EXECUTION_ERROR, 559
 - ATCA_FUNC_FAIL, 559
 - ATCA_GEN_FAIL, 559
 - ATCA_HEALTH_TEST_ERROR, 559
 - ATCA_INVALID_ID, 559
 - ATCA_INVALID_SIZE, 559
 - ATCA_NO_DEVICES, 559
 - ATCA_NOT_INITIALIZED, 560
 - ATCA_NOT_LOCKED, 560

ATCA_PARITY_ERROR, [560](#)
 ATCA_PARSE_ERROR, [560](#)
 ATCA_RESYNC_WITH_WAKEUP, [560](#)
 ATCA_RX_CRC_ERROR, [560](#)
 ATCA_RX_FAIL, [560](#)
 ATCA_RX_NO_RESPONSE, [560](#)
 ATCA_RX_TIMEOUT, [561](#)
 ATCA_SMALL_BUFFER, [561](#)
 ATCA_STATUS_CRC, [561](#)
 ATCA_STATUS_ECC, [561](#)
 ATCA_STATUS_SELFTEST_ERROR, [561](#)
 ATCA_STATUS_UNKNOWN, [561](#)
 ATCA_SUCCESS, [561](#)
 ATCA_TIMEOUT, [561](#)
 ATCA_TOO_MANY_COMM_RETRIES, [562](#)
 ATCA_TX_FAIL, [562](#)
 ATCA_TX_TIMEOUT, [562](#)
 ATCA_UNIMPLEMENTED, [562](#)
 ATCA_USE_FLAGS_CONSUMED, [562](#)
 ATCA_WAKE_FAILED, [562](#)
 ATCA_WAKE_SUCCESS, [562](#)
 ATCA_STATUS_CRC
 [atca_status.h](#), [561](#)
 ATCA_STATUS_ECC
 [atca_status.h](#), [561](#)
 ATCA_STATUS_SELFTEST_ERROR
 [atca_status.h](#), [561](#)
 ATCA_STATUS_UNKNOWN
 [atca_status.h](#), [561](#)
 ATCA_SUCCESS
 [atca_status.h](#), [561](#)
 ATCA_SWI_GPIO_IFACE
 ATCAIface ([atca_](#)), [176](#)
 ATCA_SWI_IFACE
 ATCAIface ([atca_](#)), [176](#)
 ATCA_SWI_WAKE_WORD_ADDR
 [hal_swi_gpio.h](#), [684](#)
 atca_temp_key, [387](#)
 ATCA_TIMEOUT
 [atca_status.h](#), [561](#)
 ATCA_TOO_MANY_COMM_RETRIES
 [atca_status.h](#), [562](#)
 ATCA_TX_FAIL
 [atca_status.h](#), [562](#)
 ATCA_TX_TIMEOUT
 [atca_status.h](#), [562](#)
 atca_uart_host_s, [387](#)
 ATCA_UART_IFACE
 ATCAIface ([atca_](#)), [176](#)
 ATCA_UNIMPLEMENTED
 [atca_status.h](#), [562](#)
 ATCA_UNUSED_VAR_CHECK
 [atca_config_check.h](#), [535](#)
 ATCA_USE_ATCAB_FUNCTIONS
 [atca_config_check.h](#), [535](#)
 ATCA_USE_FLAGS_CONSUMED
 [atca_status.h](#), [562](#)
 atca_utils_sizes.c, [563](#)
 atca_verify_in_out, [387](#)
 atca_verify_mac, [388](#)
 atca_version.h, [564](#)
 ATCA_WAKE_FAILED
 [atca_status.h](#), [562](#)
 ATCA_WAKE_SUCCESS
 [atca_status.h](#), [562](#)
 atca_wolfssl_interface.c, [775](#)
 atca_wolfssl_interface.h, [775](#)
 atca_wolfssl_internal.h, [776](#)
 atca_write_mac_in_out, [388](#)
 atcab_aes
 Basic Crypto API methods ([atcab_](#)), [97](#)
 [cryptoauthlib.atcab](#), [278](#)
 atcab_aes_cbc_decrypt_block
 [cryptoauthlib.atcab](#), [278](#)
 ATCAB_AES_CBC_DECRYPT_EN
 [crypto_hw_config_check.h](#), [647](#)
 atcab_aes_cbc_encrypt_block
 [cryptoauthlib.atcab](#), [279](#)
 ATCAB_AES_CBC_ENCRYPT_EN
 [crypto_hw_config_check.h](#), [647](#)
 atcab_aes_cbc_init
 [cryptoauthlib.atcab](#), [279](#)
 ATCAB_AES_CBCMAC_EN
 [crypto_hw_config_check.h](#), [647](#)
 atcab_aes_cbcmac_finish
 [cryptoauthlib.atcab](#), [280](#)
 atcab_aes_cbcmac_init
 [cryptoauthlib.atcab](#), [280](#)
 atcab_aes_cbcmac_update
 [cryptoauthlib.atcab](#), [280](#)
 atcab_aes_ccm_aad_finish
 [cryptoauthlib.atcab](#), [281](#)
 atcab_aes_ccm_aad_update
 [cryptoauthlib.atcab](#), [281](#)
 atcab_aes_ccm_decrypt_finish
 [cryptoauthlib.atcab](#), [281](#)
 atcab_aes_ccm_decrypt_update
 [cryptoauthlib.atcab](#), [282](#)
 ATCAB_AES_CCM_EN
 [crypto_hw_config_check.h](#), [647](#)
 atcab_aes_ccm_encrypt_finish
 [cryptoauthlib.atcab](#), [282](#)
 atcab_aes_ccm_encrypt_update
 [cryptoauthlib.atcab](#), [282](#)
 atcab_aes_ccm_init
 [cryptoauthlib.atcab](#), [283](#)
 atcab_aes_ccm_init_rand
 [cryptoauthlib.atcab](#), [283](#)
 atcab_aes_cmac_finish
 [cryptoauthlib.atcab](#), [284](#)
 atcab_aes_cmac_init
 [cryptoauthlib.atcab](#), [284](#)
 atcab_aes_cmac_update
 [cryptoauthlib.atcab](#), [284](#)
 atcab_aes_ctr_decrypt_block
 [cryptoauthlib.atcab](#), [285](#)

- ATCAB_AES_CTR_EN
 - crypto_hw_config_check.h, [647](#)
- atcab_aes_ctr_encrypt_block
 - cryptoauthlib.atcab, [285](#)
- atcab_aes_ctr_init
 - cryptoauthlib.atcab, [285](#)
- atcab_aes_ctr_init_rand
 - cryptoauthlib.atcab, [286](#)
- ATCAB_AES_CTR RAND_IV_EN
 - crypto_hw_config_check.h, [648](#)
- atcab_aes_decrypt
 - Basic Crypto API methods (atcab_), [98](#)
 - cryptoauthlib.atcab, [286](#)
- atcab_aes_decrypt_ext
 - Basic Crypto API methods (atcab_), [98](#)
- atcab_aes_encrypt
 - Basic Crypto API methods (atcab_), [99](#)
 - cryptoauthlib.atcab, [287](#)
- atcab_aes_encrypt_ext
 - Basic Crypto API methods (atcab_), [99](#)
- ATCAB_AES_EXTRAS_EN
 - crypto_hw_config_check.h, [648](#)
- atcab_aes_gcm_aad_update
 - Basic Crypto API methods (atcab_), [100](#)
 - cryptoauthlib.atcab, [287](#)
- atcab_aes_gcm_aad_update_ext
 - Basic Crypto API methods (atcab_), [100](#)
- atcab_aes_gcm_decrypt_finish
 - Basic Crypto API methods (atcab_), [101](#)
 - cryptoauthlib.atcab, [288](#)
- atcab_aes_gcm_decrypt_finish_ext
 - Basic Crypto API methods (atcab_), [101](#)
- atcab_aes_gcm_decrypt_update
 - Basic Crypto API methods (atcab_), [102](#)
 - cryptoauthlib.atcab, [288](#)
- atcab_aes_gcm_decrypt_update_ext
 - Basic Crypto API methods (atcab_), [102](#)
- atcab_aes_gcm_encrypt_finish
 - Basic Crypto API methods (atcab_), [103](#)
 - cryptoauthlib.atcab, [288](#)
- atcab_aes_gcm_encrypt_finish_ext
 - Basic Crypto API methods (atcab_), [103](#)
- atcab_aes_gcm_encrypt_update
 - Basic Crypto API methods (atcab_), [103](#)
 - cryptoauthlib.atcab, [289](#)
- atcab_aes_gcm_encrypt_update_ext
 - Basic Crypto API methods (atcab_), [104](#)
- atcab_aes_gcm_init
 - Basic Crypto API methods (atcab_), [104](#)
 - cryptoauthlib.atcab, [289](#)
- atcab_aes_gcm_init_ext
 - Basic Crypto API methods (atcab_), [105](#)
- atcab_aes_gcm_init_rand
 - Basic Crypto API methods (atcab_), [105](#)
 - cryptoauthlib.atcab, [289](#)
- atcab_aes_gfm
 - Basic Crypto API methods (atcab_), [106](#)
 - cryptoauthlib.atcab, [290](#)
- ATCAB_AES_GFM_EN
 - atca_config_check.h, [535](#)
- ATCAB_AES_UPDATE_EN
 - crypto_hw_config_check.h, [648](#)
- atcab_base64decode
 - atca_helpers.c, [542](#)
 - Basic Crypto API methods (atcab_), [106](#)
- atcab_base64decode_
 - atca_helpers.c, [543](#)
 - Basic Crypto API methods (atcab_), [107](#)
- atcab_base64encode
 - atca_helpers.c, [543](#)
 - Basic Crypto API methods (atcab_), [107](#)
- atcab_base64encode_
 - atca_helpers.c, [544](#)
 - Basic Crypto API methods (atcab_), [108](#)
- atcab_bin2hex
 - atca_helpers.c, [544](#)
 - Basic Crypto API methods (atcab_), [108](#)
- atcab_bin2hex_
 - atca_helpers.c, [545](#)
 - Basic Crypto API methods (atcab_), [109](#)
- atcab_challenge
 - Basic Crypto API methods (atcab_), [109](#)
 - cryptoauthlib.atcab, [290](#)
- atcab_challenge_seed_update
 - Basic Crypto API methods (atcab_), [110](#)
 - cryptoauthlib.atcab, [291](#)
- atcab_checkmac
 - Basic Crypto API methods (atcab_), [110](#)
 - cryptoauthlib.atcab, [291](#)
- atcab_checkmac_with_response_mac
 - Basic Crypto API methods (atcab_), [111](#)
- atcab_cmp_config_zone
 - Basic Crypto API methods (atcab_), [111](#)
 - cryptoauthlib.atcab, [291](#)
- atcab_counter
 - Basic Crypto API methods (atcab_), [112](#)
 - cryptoauthlib.atcab, [292](#)
- atcab_counter_increment
 - Basic Crypto API methods (atcab_), [112](#)
 - cryptoauthlib.atcab, [292](#)
- atcab_counter_read
 - Basic Crypto API methods (atcab_), [112](#)
 - cryptoauthlib.atcab, [292](#)
- atcab_derivekey
 - Basic Crypto API methods (atcab_), [113](#)
 - cryptoauthlib.atcab, [293](#)
- atcab_derivekey_ext
 - Basic Crypto API methods (atcab_), [113](#)
- atcab_ecdh
 - Basic Crypto API methods (atcab_), [114](#)
 - cryptoauthlib.atcab, [293](#)
- atcab_ecdh_base
 - Basic Crypto API methods (atcab_), [114](#)
 - cryptoauthlib.atcab, [293](#)
- atcab_ecdh_enc
 - Basic Crypto API methods (atcab_), [115](#)

- cryptoauthlib.atcab, [294](#)
- atcab_ecdh_ioenc
 - Basic Crypto API methods (atcab_), [115](#)
 - cryptoauthlib.atcab, [294](#)
- atcab_ecdh_tempkey
 - Basic Crypto API methods (atcab_), [116](#)
 - cryptoauthlib.atcab, [295](#)
- atcab_ecdh_tempkey_ioenc
 - Basic Crypto API methods (atcab_), [116](#)
 - cryptoauthlib.atcab, [295](#)
- atcab_gendig
 - Basic Crypto API methods (atcab_), [116](#)
 - cryptoauthlib.atcab, [296](#)
- atcab_gendivkey
 - Basic Crypto API methods (atcab_), [117](#)
- atcab_genkey
 - Basic Crypto API methods (atcab_), [117](#)
 - cryptoauthlib.atcab, [296](#)
- atcab_genkey_base
 - Basic Crypto API methods (atcab_), [118](#)
 - cryptoauthlib.atcab, [296](#)
- atcab_genkey_ext
 - Basic Crypto API methods (atcab_), [118](#)
- ATCAB_GENKEY_MAC_EN
 - atca_config_check.h, [536](#)
- atcab_get_device
 - Basic Crypto API methods (atcab_), [119](#)
 - cryptoauthlib.atcab, [297](#)
- atcab_get_device_address
 - Basic Crypto API methods (atcab_), [119](#)
- atcab_get_device_type
 - Basic Crypto API methods (atcab_), [119](#)
 - cryptoauthlib.atcab, [297](#)
- atcab_get_device_type_ext
 - Basic Crypto API methods (atcab_), [119](#)
- atcab_get_pubkey
 - Basic Crypto API methods (atcab_), [120](#)
 - cryptoauthlib.atcab, [297](#)
- atcab_get_pubkey_ext
 - Basic Crypto API methods (atcab_), [120](#)
- atcab_get_zone_size
 - Basic Crypto API methods (atcab_), [121](#)
- atcab_get_zone_size_ext
 - Basic Crypto API methods (atcab_), [121](#)
- atcab_hex2bin
 - atca_helpers.c, [545](#)
 - Basic Crypto API methods (atcab_), [121](#)
- atcab_hmac
 - Basic Crypto API methods (atcab_), [122](#)
 - cryptoauthlib.atcab, [298](#)
- atcab_hw_sha2_256
 - Basic Crypto API methods (atcab_), [122](#)
 - cryptoauthlib.atcab, [298](#)
- atcab_hw_sha2_256_finish
 - Basic Crypto API methods (atcab_), [123](#)
 - cryptoauthlib.atcab, [298](#)
- atcab_hw_sha2_256_init
 - Basic Crypto API methods (atcab_), [123](#)
- cryptoauthlib.atcab, [299](#)
- atcab_hw_sha2_256_update
 - Basic Crypto API methods (atcab_), [123](#)
 - cryptoauthlib.atcab, [299](#)
- atcab_idle
 - Basic Crypto API methods (atcab_), [124](#)
- atcab_info
 - Basic Crypto API methods (atcab_), [124](#)
 - cryptoauthlib.atcab, [299](#)
- atcab_info_base
 - Basic Crypto API methods (atcab_), [124](#)
 - cryptoauthlib.atcab, [300](#)
- atcab_info_chip_status
 - Basic Crypto API methods (atcab_), [125](#)
- atcab_info_ext
 - Basic Crypto API methods (atcab_), [125](#)
- atcab_info_get_latch
 - Basic Crypto API methods (atcab_), [126](#)
 - cryptoauthlib.atcab, [300](#)
- ATCAB_INFO_LATCH_EN
 - atca_config_check.h, [536](#)
- atcab_info_lock_status
 - Basic Crypto API methods (atcab_), [126](#)
- atcab_info_set_latch
 - Basic Crypto API methods (atcab_), [126](#)
 - cryptoauthlib.atcab, [300](#)
- atcab_init
 - Basic Crypto API methods (atcab_), [127](#)
 - cryptoauthlib.atcab, [301](#)
- atcab_init_device
 - Basic Crypto API methods (atcab_), [127](#)
- atcab_init_ext
 - Basic Crypto API methods (atcab_), [127](#)
- atcab_is_ca2_device
 - Basic Crypto API methods (atcab_), [128](#)
- atcab_is_ca_device
 - Basic Crypto API methods (atcab_), [128](#)
- atcab_is_config_locked
 - Basic Crypto API methods (atcab_), [128](#)
- atcab_is_config_locked_ext
 - Basic Crypto API methods (atcab_), [129](#)
- atcab_is_data_locked
 - Basic Crypto API methods (atcab_), [129](#)
- atcab_is_data_locked_ext
 - Basic Crypto API methods (atcab_), [129](#)
- atcab_is_locked
 - Basic Crypto API methods (atcab_), [130](#)
 - cryptoauthlib.atcab, [301](#)
- atcab_is_private_ext
 - Basic Crypto API methods (atcab_), [130](#)
- atcab_is_slot_locked
 - Basic Crypto API methods (atcab_), [131](#)
 - cryptoauthlib.atcab, [301](#)
- atcab_is_slot_locked_ext
 - Basic Crypto API methods (atcab_), [131](#)
- atcab_is_ta_device
 - Basic Crypto API methods (atcab_), [132](#)
- atcab_kdf

- Basic Crypto API methods (atcab_), 132
- cryptoauthlib.atcab, 302
- atcab_lock
 - Basic Crypto API methods (atcab_), 132
 - cryptoauthlib.atcab, 302
- atcab_lock_config_zone
 - Basic Crypto API methods (atcab_), 133
 - cryptoauthlib.atcab, 303
- atcab_lock_config_zone_crc
 - Basic Crypto API methods (atcab_), 133
 - cryptoauthlib.atcab, 303
- atcab_lock_config_zone_ext
 - Basic Crypto API methods (atcab_), 134
- atcab_lock_data_slot
 - Basic Crypto API methods (atcab_), 134
 - cryptoauthlib.atcab, 303
- atcab_lock_data_slot_ext
 - Basic Crypto API methods (atcab_), 134
- atcab_lock_data_zone
 - Basic Crypto API methods (atcab_), 135
 - cryptoauthlib.atcab, 304
- atcab_lock_data_zone_crc
 - Basic Crypto API methods (atcab_), 135
 - cryptoauthlib.atcab, 304
- atcab_lock_data_zone_ext
 - Basic Crypto API methods (atcab_), 135
- atcab_mac
 - Basic Crypto API methods (atcab_), 136
 - cryptoauthlib.atcab, 304
- atcab_nonce
 - Basic Crypto API methods (atcab_), 136
 - cryptoauthlib.atcab, 305
- atcab_nonce_base
 - Basic Crypto API methods (atcab_), 137
 - cryptoauthlib.atcab, 305
- atcab_nonce_load
 - Basic Crypto API methods (atcab_), 137
 - cryptoauthlib.atcab, 306
- atcab_nonce_rand
 - Basic Crypto API methods (atcab_), 138
 - cryptoauthlib.atcab, 306
- atcab_nonce_rand_ext
 - Basic Crypto API methods (atcab_), 138
- ATCAB_PBKDF2_SHA256_EN
 - crypto_sw_config_check.h, 651
- atcab_priv_write
 - Basic Crypto API methods (atcab_), 139
 - cryptoauthlib.atcab, 307
- atcab_random
 - Basic Crypto API methods (atcab_), 139
 - cryptoauthlib.atcab, 307
- atcab_random_ext
 - Basic Crypto API methods (atcab_), 139
- atcab_read_bytes_zone
 - Basic Crypto API methods (atcab_), 140
 - cryptoauthlib.atcab, 307
- atcab_read_config_zone
 - Basic Crypto API methods (atcab_), 140
 - cryptoauthlib.atcab, 308
- atcab_read_config_zone_ext
 - Basic Crypto API methods (atcab_), 141
- atcab_read_enc
 - Basic Crypto API methods (atcab_), 141
 - cryptoauthlib.atcab, 308
- atcab_read_pubkey
 - Basic Crypto API methods (atcab_), 142
 - cryptoauthlib.atcab, 309
- atcab_read_pubkey_ext
 - Basic Crypto API methods (atcab_), 142
- atcab_read_serial_number
 - Basic Crypto API methods (atcab_), 143
 - cryptoauthlib.atcab, 309
- atcab_read_serial_number_ext
 - Basic Crypto API methods (atcab_), 143
- atcab_read_sig
 - Basic Crypto API methods (atcab_), 143
 - cryptoauthlib.atcab, 309
- atcab_read_zone
 - Basic Crypto API methods (atcab_), 144
 - cryptoauthlib.atcab, 310
- atcab_read_zone_ext
 - Basic Crypto API methods (atcab_), 144
- atcab_release
 - Basic Crypto API methods (atcab_), 145
 - cryptoauthlib.atcab, 310
- atcab_release_ext
 - Basic Crypto API methods (atcab_), 145
- atcab_reversal
 - atca_helpers.c, 546
 - Basic Crypto API methods (atcab_), 145
- atcab_secureboot
 - Basic Crypto API methods (atcab_), 146
 - cryptoauthlib.atcab, 310
- atcab_secureboot_mac
 - Basic Crypto API methods (atcab_), 146
 - cryptoauthlib.atcab, 311
- atcab_selftest
 - Basic Crypto API methods (atcab_), 147
 - cryptoauthlib.atcab, 311
- atcab_sha
 - Basic Crypto API methods (atcab_), 147
 - cryptoauthlib.atcab, 312
- atcab_sha_base
 - Basic Crypto API methods (atcab_), 148
 - cryptoauthlib.atcab, 312
- atcab_sha_end
 - Basic Crypto API methods (atcab_), 149
 - cryptoauthlib.atcab, 313
- atcab_sha_hmac
 - Basic Crypto API methods (atcab_), 149
 - cryptoauthlib.atcab, 313
- atcab_sha_hmac_ext
 - Basic Crypto API methods (atcab_), 149
- atcab_sha_hmac_finish
 - Basic Crypto API methods (atcab_), 150
 - cryptoauthlib.atcab, 314

- atcab_sha_hmac_init
 - Basic Crypto API methods (atcab_), 151
 - cryptoauthlib.atcab, 314
- atcab_sha_hmac_update
 - Basic Crypto API methods (atcab_), 151
 - cryptoauthlib.atcab, 314
- atcab_sha_read_context
 - Basic Crypto API methods (atcab_), 151
 - cryptoauthlib.atcab, 315
- atcab_sha_start
 - Basic Crypto API methods (atcab_), 152
 - cryptoauthlib.atcab, 315
- atcab_sha_update
 - Basic Crypto API methods (atcab_), 152
 - cryptoauthlib.atcab, 315
- atcab_sha_write_context
 - Basic Crypto API methods (atcab_), 152
 - cryptoauthlib.atcab, 316
- atcab_sign
 - Basic Crypto API methods (atcab_), 153
 - cryptoauthlib.atcab, 316
- atcab_sign_base
 - Basic Crypto API methods (atcab_), 153
 - cryptoauthlib.atcab, 316
- atcab_sign_ext
 - Basic Crypto API methods (atcab_), 154
- atcab_sign_internal
 - Basic Crypto API methods (atcab_), 154
 - cryptoauthlib.atcab, 317
- atcab_sleep
 - Basic Crypto API methods (atcab_), 155
- atcab_updateextra
 - Basic Crypto API methods (atcab_), 155
 - cryptoauthlib.atcab, 317
- atcab_verify
 - Basic Crypto API methods (atcab_), 155
 - cryptoauthlib.atcab, 317
- atcab_verify_extern
 - Basic Crypto API methods (atcab_), 156
 - cryptoauthlib.atcab, 318
- atcab_verify_extern_ext
 - Basic Crypto API methods (atcab_), 157
- atcab_verify_extern_mac
 - Basic Crypto API methods (atcab_), 157
 - cryptoauthlib.atcab, 319
- atcab_verify_extern_stored_mac
 - cryptoauthlib.atcab, 319
- atcab_verify_invalidate
 - Basic Crypto API methods (atcab_), 158
 - cryptoauthlib.atcab, 320
- ATCAB_VERIFY_MAC_EN
 - atca_config_check.h, 536
- atcab_verify_stored
 - Basic Crypto API methods (atcab_), 158
 - cryptoauthlib.atcab, 320
- atcab_verify_stored_ext
 - Basic Crypto API methods (atcab_), 159
- atcab_verify_stored_mac
 - Basic Crypto API methods (atcab_), 159
 - cryptoauthlib.atcab, 321
- atcab_verify_stored_with_tempkey
 - Basic Crypto API methods (atcab_), 160
- atcab_verify_validate
 - Basic Crypto API methods (atcab_), 160
 - cryptoauthlib.atcab, 321
- atcab_version
 - Basic Crypto API methods (atcab_), 161
- atcab_wakeup
 - Basic Crypto API methods (atcab_), 161
- atcab_write
 - Basic Crypto API methods (atcab_), 161
 - cryptoauthlib.atcab, 322
- atcab_write_bytes_zone
 - Basic Crypto API methods (atcab_), 162
 - cryptoauthlib.atcab, 322
- atcab_write_config_counter
 - Basic Crypto API methods (atcab_), 163
 - cryptoauthlib.atcab, 323
- atcab_write_config_zone
 - Basic Crypto API methods (atcab_), 163
 - cryptoauthlib.atcab, 323
- atcab_write_config_zone_ext
 - Basic Crypto API methods (atcab_), 163
- ATCAB_WRITE_EN
 - atca_config_check.h, 536
- atcab_write_enc
 - Basic Crypto API methods (atcab_), 164
 - cryptoauthlib.atcab, 323
- atcab_write_pubkey
 - Basic Crypto API methods (atcab_), 164
 - cryptoauthlib.atcab, 324
- atcab_write_pubkey_ext
 - Basic Crypto API methods (atcab_), 165
- atcab_write_zone
 - Basic Crypto API methods (atcab_), 165
 - cryptoauthlib.atcab, 324
- atcab_write_zone_ext
 - Basic Crypto API methods (atcab_), 167
- atcac_aes_cmac_ctx, 393
- ATCAC_AES_CMAC_EN
 - atca_mbedtls_interface.h, 714
 - atca_openssl_interface.h, 740
 - crypto_sw_config_check.h, 651
- atcac_aes_cmac_finish
 - atca_mbedtls_wrap.c, 717
 - atca_openssl_interface.c, 729
- atcac_aes_cmac_init
 - atca_mbedtls_wrap.c, 718
 - atca_openssl_interface.c, 730
- atcac_aes_cmac_update
 - atca_mbedtls_wrap.c, 718
 - atca_openssl_interface.c, 730
- atcac_aes_gcm_aad_update
 - atca_mbedtls_wrap.c, 719
 - atca_openssl_interface.c, 731
- atcac_aes_gcm_ctx, 393

- atcac_aes_gcm_decrypt_finish
 - atca_mbedtls_wrap.c, [719](#)
 - atca_openssl_interface.c, [731](#)
- atcac_aes_gcm_decrypt_start
 - atca_mbedtls_wrap.c, [720](#)
 - atca_openssl_interface.c, [732](#)
- atcac_aes_gcm_decrypt_update
 - atca_mbedtls_wrap.c, [720](#)
 - atca_openssl_interface.c, [732](#)
- ATCAC_AES_GCM_EN
 - atca_mbedtls_interface.h, [714](#)
 - atca_openssl_interface.h, [740](#)
 - crypto_sw_config_check.h, [651](#)
- atcac_aes_gcm_encrypt_finish
 - atca_mbedtls_wrap.c, [721](#)
 - atca_openssl_interface.c, [733](#)
- atcac_aes_gcm_encrypt_start
 - atca_mbedtls_wrap.c, [721](#)
 - atca_openssl_interface.c, [733](#)
- atcac_aes_gcm_encrypt_update
 - atca_mbedtls_wrap.c, [722](#)
 - atca_openssl_interface.c, [734](#)
- ATCAC_CERT_ADD_EN
 - atca_config_check.h, [536](#)
 - crypto_sw_config_check.h, [651](#)
- atcac_hmac_ctx, [394](#)
- ATCAC_PBKDF2_SHA256_EN
 - crypto_sw_config_check.h, [651](#)
- atcac_pk_ctx, [394](#)
- atcac_pk_derive
 - atca_mbedtls_wrap.c, [722](#)
 - atca_openssl_interface.c, [734](#)
- atcac_pk_free
 - atca_mbedtls_wrap.c, [722](#)
 - atca_openssl_interface.c, [734](#)
- atcac_pk_init
 - atca_mbedtls_wrap.c, [723](#)
 - atca_openssl_interface.c, [735](#)
- atcac_pk_init_pem
 - atca_mbedtls_wrap.c, [723](#)
 - atca_openssl_interface.c, [735](#)
- atcac_pk_public
 - atca_mbedtls_wrap.c, [724](#)
 - atca_openssl_interface.c, [736](#)
- atcac_pk_sign
 - atca_mbedtls_wrap.c, [724](#)
 - atca_openssl_interface.c, [736](#)
- atcac_pk_verify
 - atca_mbedtls_wrap.c, [724](#)
 - atca_openssl_interface.c, [736](#)
- ATCAC_PKEY_EN
 - atca_mbedtls_interface.h, [714](#)
 - atca_openssl_interface.h, [741](#)
 - crypto_sw_config_check.h, [652](#)
- ATCAC_RANDOM_EN
 - atca_config_check.h, [537](#)
 - crypto_sw_config_check.h, [652](#)
- atcac_sha1_ctx, [394](#)
- ATCAC_SHA1_EN
 - atca_config_check.h, [537](#)
 - atca_mbedtls_interface.h, [714](#)
 - atca_openssl_interface.h, [741](#)
 - crypto_sw_config_check.h, [652](#)
- ATCAC_SHA256_EN
 - atca_config_check.h, [537](#)
 - atca_mbedtls_interface.h, [714](#)
 - atca_openssl_interface.h, [741](#)
 - crypto_sw_config_check.h, [652](#)
- atcac_sha256_hmac_finish
 - atca_mbedtls_wrap.c, [724](#)
 - atca_openssl_interface.c, [736](#)
- atcac_sha256_hmac_init
 - atca_mbedtls_wrap.c, [725](#)
 - atca_openssl_interface.c, [737](#)
- atcac_sha256_hmac_update
 - atca_mbedtls_wrap.c, [725](#)
 - atca_openssl_interface.c, [737](#)
- atcac_sha2_256_ctx, [394](#)
- atcac_sha2_384_ctx, [394](#)
- atcac_sha2_512_ctx, [394](#)
- ATCAC_SHA384_EN
 - atca_config_check.h, [537](#)
 - atca_mbedtls_interface.h, [714](#)
 - atca_openssl_interface.h, [741](#)
 - crypto_sw_config_check.h, [652](#)
- ATCAC_SHA512_EN
 - atca_config_check.h, [537](#)
 - atca_mbedtls_interface.h, [715](#)
 - atca_openssl_interface.h, [741](#)
 - crypto_sw_config_check.h, [652](#)
- ATCAC_SIGN_EN
 - atca_config_check.h, [537](#)
 - crypto_sw_config_check.h, [653](#)
- atcac_sw_random
 - atca_mbedtls_wrap.c, [726](#)
 - atca_openssl_interface.c, [738](#)
- atcac_sw_sha1_finish
 - atca_mbedtls_wrap.c, [726](#)
 - atca_openssl_interface.c, [738](#)
- atcac_sw_sha1_init
 - atca_mbedtls_wrap.c, [727](#)
 - atca_openssl_interface.c, [739](#)
- atcac_sw_sha1_update
 - atca_mbedtls_wrap.c, [727](#)
 - atca_openssl_interface.c, [739](#)
- ATCAC_VERIFY_EN
 - atca_config_check.h, [538](#)
 - crypto_sw_config_check.h, [653](#)
- atcac_x509_ctx, [395](#)
- atcacert.h, [564](#)
- atcacert_build_state_s, [395](#)
- atcacert_build_state_t
 - Certificate manipulation methods (atcacert_), [188](#)
- atcacert_cert_element_s, [395](#)
- atcacert_cert_element_t
 - Certificate manipulation methods (atcacert_), [188](#)

- atcacert_cert_loc_s, [397](#)
- atcacert_cert_loc_t
 - Certificate manipulation methods (atcacert_), [188](#)
- atcacert_cert_sn_src_e
 - Certificate manipulation methods (atcacert_), [189](#)
- atcacert_cert_sn_src_t
 - Certificate manipulation methods (atcacert_), [188](#)
- atcacert_cert_type_e
 - Certificate manipulation methods (atcacert_), [190](#)
- atcacert_cert_type_t
 - Certificate manipulation methods (atcacert_), [188](#)
- atcacert_check_config.h, [565](#)
- atcacert_client.c, [566](#)
- atcacert_client.h, [566](#)
 - atcacert_read_cert, [567](#)
 - atcacert_read_cert_ext, [568](#)
 - atcacert_read_cert_size, [568](#)
 - atcacert_read_cert_size_ext, [569](#)
 - atcacert_write_cert, [569](#)
 - atcacert_write_cert_ext, [570](#)
- atcacert_create_csr
 - cryptoauthlib.atcacert, [326](#)
- atcacert_create_csr_pem
 - cryptoauthlib.atcacert, [326](#)
- atcacert_date.c, [570](#)
- atcacert_date.h, [571](#)
- atcacert_date_cmp
 - Certificate manipulation methods (atcacert_), [192](#)
- atcacert_date_dec
 - Certificate manipulation methods (atcacert_), [192](#)
 - cryptoauthlib.atcacert, [327](#)
- atcacert_date_dec_compcert
 - Certificate manipulation methods (atcacert_), [192](#)
 - cryptoauthlib.atcacert, [327](#)
- atcacert_date_dec_compcert_ext
 - Certificate manipulation methods (atcacert_), [193](#)
- atcacert_date_enc
 - Certificate manipulation methods (atcacert_), [194](#)
 - cryptoauthlib.atcacert, [328](#)
- atcacert_date_enc_compcert
 - Certificate manipulation methods (atcacert_), [194](#)
 - cryptoauthlib.atcacert, [328](#)
- atcacert_date_enc_compcert_ext
 - Certificate manipulation methods (atcacert_), [194](#)
- atcacert_date_from_asn1_tag
 - Certificate manipulation methods (atcacert_), [195](#)
- atcacert_date_get_max_date
 - Certificate manipulation methods (atcacert_), [195](#)
 - cryptoauthlib.atcacert, [328](#)
- atcacert_decode_pem
 - atcacert_pem.h, [579](#)
- atcacert_decode_pem_cert
 - atcacert_pem.h, [580](#)
- atcacert_decode_pem_csr
 - atcacert_pem.h, [580](#)
- atcacert_def.c, [573](#)
- atcacert_def.h, [573](#)
- atcacert_def_s, [402](#)
- atcacert_def_t
 - Certificate manipulation methods (atcacert_), [188](#)
- atcacert_der.c, [575](#)
- atcacert_der.h, [576](#)
- atcacert_der_dec_ecdsa_sig_value
 - Certificate manipulation methods (atcacert_), [196](#)
- atcacert_der_dec_integer
 - Certificate manipulation methods (atcacert_), [196](#)
- atcacert_der_dec_length
 - Certificate manipulation methods (atcacert_), [197](#)
- atcacert_der_enc_ecdsa_sig_value
 - Certificate manipulation methods (atcacert_), [197](#)
- atcacert_der_enc_integer
 - Certificate manipulation methods (atcacert_), [198](#)
- atcacert_der_enc_length
 - Certificate manipulation methods (atcacert_), [198](#)
- atcacert_device_loc_s, [403](#)
- atcacert_device_loc_t
 - Certificate manipulation methods (atcacert_), [189](#)
- atcacert_device_zone_e
 - Certificate manipulation methods (atcacert_), [190](#)
- atcacert_device_zone_t
 - Certificate manipulation methods (atcacert_), [189](#)
- ATCACERT_E_BAD_CERT
 - Certificate manipulation methods (atcacert_), [186](#)
- ATCACERT_E_BAD_PARAMS
 - Certificate manipulation methods (atcacert_), [186](#)
- ATCACERT_E_BUFFER_TOO_SMALL
 - Certificate manipulation methods (atcacert_), [186](#)
- ATCACERT_E_DECODING_ERROR
 - Certificate manipulation methods (atcacert_), [186](#)
- ATCACERT_E_ELEM_MISSING
 - Certificate manipulation methods (atcacert_), [186](#)
- ATCACERT_E_ELEM_OUT_OF_BOUNDS
 - Certificate manipulation methods (atcacert_), [187](#)
- ATCACERT_E_ERROR
 - Certificate manipulation methods (atcacert_), [187](#)
- ATCACERT_E_INVALID_DATE
 - Certificate manipulation methods (atcacert_), [187](#)
- ATCACERT_E_INVALID_TRANSFORM
 - Certificate manipulation methods (atcacert_), [187](#)
- ATCACERT_E_SUCCESS
 - Certificate manipulation methods (atcacert_), [187](#)
- ATCACERT_E_UNEXPECTED_ELEM_SIZE
 - Certificate manipulation methods (atcacert_), [187](#)
- ATCACERT_E_UNIMPLEMENTED
 - Certificate manipulation methods (atcacert_), [187](#)
- ATCACERT_E_VERIFY_FAILED
 - Certificate manipulation methods (atcacert_), [187](#)
- atcacert_encode_pem
 - atcacert_pem.h, [581](#)
- atcacert_encode_pem_cert
 - atcacert_pem.h, [581](#)
- atcacert_encode_pem_csr
 - atcacert_pem.h, [582](#)
- atcacert_get_auth_key_id
 - Certificate manipulation methods (atcacert_), [199](#)
- atcacert_get_cert_sn

- Certificate manipulation methods (atcacert_), 199
- atcacert_get_expire_date
 - Certificate manipulation methods (atcacert_), 200
- atcacert_get_issue_date
 - Certificate manipulation methods (atcacert_), 200
- atcacert_get_issuer
 - Certificate manipulation methods (atcacert_), 201
- atcacert_get_response
 - cryptoauthlib.atcacert, 329
- atcacert_get_subj_key_id
 - Certificate manipulation methods (atcacert_), 201
- atcacert_get_subj_public_key
 - Certificate manipulation methods (atcacert_), 202
- atcacert_get_subject
 - Certificate manipulation methods (atcacert_), 202
- atcacert_host_hw.c, 577
- atcacert_host_hw.h, 577
- atcacert_host_sw.c, 577
- atcacert_host_sw.h, 578
- atcacert_max_cert_size
 - cryptoauthlib.atcacert, 329
- atcacert_pem.c, 578
- atcacert_pem.h, 579
 - atcacert_decode_pem, 579
 - atcacert_decode_pem_cert, 580
 - atcacert_decode_pem_csr, 580
 - atcacert_encode_pem, 581
 - atcacert_encode_pem_cert, 581
 - atcacert_encode_pem_csr, 582
- atcacert_read_cert
 - atcacert_client.h, 567
 - cryptoauthlib.atcacert, 329
- atcacert_read_cert_ext
 - atcacert_client.h, 568
- atcacert_read_cert_size
 - atcacert_client.h, 568
- atcacert_read_cert_size_ext
 - atcacert_client.h, 569
- atcacert_std_cert_element_e
 - Certificate manipulation methods (atcacert_), 191
- atcacert_std_cert_element_t
 - Certificate manipulation methods (atcacert_), 189
- atcacert_tm_utc_s, 406
- atcacert_tm_utc_t
 - Certificate manipulation methods (atcacert_), 189
- atcacert_transform_e
 - Certificate manipulation methods (atcacert_), 191
- atcacert_write_cert
 - atcacert_client.h, 569
 - cryptoauthlib.atcacert, 330
- atcacert_write_cert_ext
 - atcacert_client.h, 570
- ATCADevice (atca_), 171
 - atGetIFace, 172
 - deleteATCADevice, 173
 - initATCADevice, 173
 - newATCADevice, 173
 - releaseATCADevice, 174
- ATCAH_CHECK_MAC
 - atca_host_config_check.h, 707
- ATCAH_CONFIG_TO_SIGN_INTERNAL
 - atca_host_config_check.h, 707
- ATCAH_DECRYPT
 - atca_host_config_check.h, 708
- ATCAH_DELETE_MAC
 - atca_host_config_check.h, 708
- ATCAH_DERIVE_KEY
 - atca_host_config_check.h, 708
- ATCAH_DERIVE_KEY_MAC
 - atca_host_config_check.h, 708
- ATCAH_ENCODE_COUNTER_MATCH
 - atca_host_config_check.h, 708
- ATCAH_GEN_KEY_MSG
 - atca_host_config_check.h, 709
- ATCAH_GEN_MAC
 - atca_host_config_check.h, 709
- ATCAH_GEN_OUTPUT_RESP_MAC
 - atca_host_config_check.h, 709
- ATCAH_GEN_SESSION_KEY
 - atca_host_config_check.h, 709
- ATCAH_GENDIG
 - atca_host_config_check.h, 709
- ATCAH_GENDIVKEY
 - atca_host_config_check.h, 710
- ATCAH_HMAC
 - atca_host_config_check.h, 710
- ATCAH_INCLUDE_DATA
 - atca_host_config_check.h, 710
- ATCAH_IO_DECRYPT
 - atca_host_config_check.h, 710
- ATCAH_MAC
 - atca_host_config_check.h, 710
- ATCAH_NONCE
 - atca_host_config_check.h, 711
- ATCAH_PRIVWRITE_AUTH_MAC
 - atca_host_config_check.h, 711
- ATCAH_SECUREBOOT_ENC
 - atca_host_config_check.h, 711
- ATCAH_SECUREBOOT_MAC
 - atca_host_config_check.h, 711
- ATCAH_SHA256
 - atca_host_config_check.h, 711
- ATCAH_SIGN_INTERNAL_MSG
 - atca_host_config_check.h, 712
- ATCAH_VERIFY_MAC
 - atca_host_config_check.h, 712
- ATCAH_WRITE_AUTH_MAC
 - atca_host_config_check.h, 712
- ATCAHAL_t, 411
- atcal2Cmaster, 411
- ATCAIface (atca_), 174
 - ATCA_CUSTOM_IFACE, 176
 - ATCA_HID_IFACE, 176
 - ATCA_I2C_GPIO_IFACE, 176
 - ATCA_I2C_IFACE, 176
 - atca_iface_is_kit, 176

- atca_iface_is_swi, [176](#)
- ATCA_KIT_IFACE, [176](#)
- ATCA_SPI_GPIO_IFACE, [176](#)
- ATCA_SPI_IFACE, [176](#)
- ATCA_SWI_GPIO_IFACE, [176](#)
- ATCA_SWI_IFACE, [176](#)
- ATCA_UART_IFACE, [176](#)
- ATCAIfaceType, [176](#)
- atcontrol, [177](#)
- atgetifacecfg, [177](#)
- atgetifacehaldat, [178](#)
- atidle, [178](#)
- atinit, [178](#)
- atreceive, [179](#)
- atsend, [179](#)
- atsleep, [180](#)
- atwake, [180](#)
- deleteATCAIface, [180](#)
- ifacecfg_set_address, [181](#)
- ifacetype_is_kit, [181](#)
- initATCAIface, [181](#)
- releaseATCAIface, [182](#)
- ATCAIfaceCfg, [412](#)
 - address, [413](#)
- ATCAIfaceType
 - ATCAIface (atca_), [176](#)
- atCalcCrc
 - calib_command.c, [591](#)
 - calib_command.h, [612](#)
- ATCAPacket, [416](#)
- atcaSWImaster, [418](#)
- atCheckCrc
 - calib_command.c, [592](#)
 - calib_command.h, [613](#)
- atcontrol
 - ATCAIface (atca_), [177](#)
- atCRC
 - calib_command.c, [592](#)
 - calib_command.h, [613](#)
- atecc508a_config_s, [420](#)
- ATECC508A_DEVICE_CONFIG
 - test_device, [352](#)
- ATECC508A_DEVICE_CONFIG_VECTOR
 - test_device, [352](#)
- atecc608_config_s, [422](#)
- ATECC608_DEVICE_CONFIG
 - test_device, [353](#)
- ATECC608_DEVICE_CONFIG_VECTOR
 - test_device, [353](#)
- atGetIFace
 - ATCADevice (atca_), [172](#)
- atgetifacecfg
 - ATCAIface (atca_), [177](#)
- atgetifacehaldat
 - ATCAIface (atca_), [178](#)
- atidle
 - ATCAIface (atca_), [178](#)
- atInfo
 - calib_command.c, [592](#)
 - calib_command.h, [613](#)
- atinit
 - ATCAIface (atca_), [178](#)
- atIsECCFamily
 - calib_command.c, [593](#)
 - calib_command.h, [614](#)
- atIsSHAFamily
 - calib_command.c, [593](#)
 - calib_command.h, [614](#)
- atPause
 - calib_command.c, [593](#)
 - calib_command.h, [614](#)
- atreceive
 - ATCAIface (atca_), [179](#)
- atsend
 - ATCAIface (atca_), [179](#)
- atsha204a_config_s, [424](#)
- ATSHA204A_DEVICE_CONFIG
 - test_device, [353](#)
- ATSHA204A_DEVICE_CONFIG_VECTOR
 - test_device, [354](#)
- atsleep
 - ATCAIface (atca_), [180](#)
- attrib_f
 - pkcs11_attrib.h, [743](#)
- attributes
 - pkcs11_object_s, [466](#)
- Attributes (pkcs11_attrib_), [260](#)
 - ec_key_data_table, [271](#)
 - key_data_table, [271](#)
 - pkcs11_attrib_fill, [269](#)
 - pkcs11_cert_wtlspublic_attributes, [271](#)
 - pkcs11_cert_x509_attributes, [271](#)
 - pkcs11_cert_x509public_attributes, [272](#)
 - pkcs11_deinit, [269](#)
 - pkcs11_ec_pbkey_asn1_hdr_p224, [272](#)
 - pkcs11_ec_pbkey_asn1_hdr_p256, [272](#)
 - pkcs11_ec_pbkey_asn1_hdr_p384, [272](#)
 - pkcs11_ec_pbkey_asn1_hdr_p521, [272](#)
 - pkcs11_init, [270](#)
 - pkcs11_key_private_attributes, [273](#)
 - pkcs11_key_public_attributes, [273](#)
 - pkcs11_key_secret_attributes, [273](#)
 - pkcs11_object_monotonic_attributes, [273](#)
 - pkcs11_os_create_mutex, [270](#)
 - pkcs11_session_closeall, [270](#)
 - pkcs11_session_login, [270](#)
 - pkcs11_token_init, [270](#)
 - pkcs11_x962_asn1_hdr_ec224, [273](#)
 - pkcs11_x962_asn1_hdr_ec256, [274](#)
 - pkcs11_x962_asn1_hdr_ec384, [274](#)
 - pkcs11_x962_asn1_hdr_ec521, [274](#)
 - rsa_key_data_table, [274](#)
- atwake
 - ATCAIface (atca_), [180](#)
- Basic Crypto API methods (atcab_), [88](#)
 - atcab_aes, [97](#)

atcab_aes_decrypt, 98
atcab_aes_decrypt_ext, 98
atcab_aes_encrypt, 99
atcab_aes_encrypt_ext, 99
atcab_aes_gcm_aad_update, 100
atcab_aes_gcm_aad_update_ext, 100
atcab_aes_gcm_decrypt_finish, 101
atcab_aes_gcm_decrypt_finish_ext, 101
atcab_aes_gcm_decrypt_update, 102
atcab_aes_gcm_decrypt_update_ext, 102
atcab_aes_gcm_encrypt_finish, 103
atcab_aes_gcm_encrypt_finish_ext, 103
atcab_aes_gcm_encrypt_update, 103
atcab_aes_gcm_encrypt_update_ext, 104
atcab_aes_gcm_init, 104
atcab_aes_gcm_init_ext, 105
atcab_aes_gcm_init_rand, 105
atcab_aes_gfm, 106
atcab_base64decode, 106
atcab_base64decode_, 107
atcab_base64encode, 107
atcab_base64encode_, 108
atcab_bin2hex, 108
atcab_bin2hex_, 109
atcab_challenge, 109
atcab_challenge_seed_update, 110
atcab_checkmac, 110
atcab_checkmac_with_response_mac, 111
atcab_cmp_config_zone, 111
atcab_counter, 112
atcab_counter_increment, 112
atcab_counter_read, 112
atcab_derivekey, 113
atcab_derivekey_ext, 113
atcab_ecdh, 114
atcab_ecdh_base, 114
atcab_ecdh_enc, 115
atcab_ecdh_ioenc, 115
atcab_ecdh_tempkey, 116
atcab_ecdh_tempkey_ioenc, 116
atcab_gendig, 116
atcab_gendivkey, 117
atcab_genkey, 117
atcab_genkey_base, 118
atcab_genkey_ext, 118
atcab_get_device, 119
atcab_get_device_address, 119
atcab_get_device_type, 119
atcab_get_device_type_ext, 119
atcab_get_pubkey, 120
atcab_get_pubkey_ext, 120
atcab_get_zone_size, 121
atcab_get_zone_size_ext, 121
atcab_hex2bin, 121
atcab_hmac, 122
atcab_hw_sha2_256, 122
atcab_hw_sha2_256_finish, 123
atcab_hw_sha2_256_init, 123
atcab_hw_sha2_256_update, 123
atcab_idle, 124
atcab_info, 124
atcab_info_base, 124
atcab_info_chip_status, 125
atcab_info_ext, 125
atcab_info_get_latch, 126
atcab_info_lock_status, 126
atcab_info_set_latch, 126
atcab_init, 127
atcab_init_device, 127
atcab_init_ext, 127
atcab_is_ca2_device, 128
atcab_is_ca_device, 128
atcab_is_config_locked, 128
atcab_is_config_locked_ext, 129
atcab_is_data_locked, 129
atcab_is_data_locked_ext, 129
atcab_is_locked, 130
atcab_is_private_ext, 130
atcab_is_slot_locked, 131
atcab_is_slot_locked_ext, 131
atcab_is_ta_device, 132
atcab_kdf, 132
atcab_lock, 132
atcab_lock_config_zone, 133
atcab_lock_config_zone_crc, 133
atcab_lock_config_zone_ext, 134
atcab_lock_data_slot, 134
atcab_lock_data_slot_ext, 134
atcab_lock_data_zone, 135
atcab_lock_data_zone_crc, 135
atcab_lock_data_zone_ext, 135
atcab_mac, 136
atcab_nonce, 136
atcab_nonce_base, 137
atcab_nonce_load, 137
atcab_nonce_rand, 138
atcab_nonce_rand_ext, 138
atcab_priv_write, 139
atcab_random, 139
atcab_random_ext, 139
atcab_read_bytes_zone, 140
atcab_read_config_zone, 140
atcab_read_config_zone_ext, 141
atcab_read_enc, 141
atcab_read_pubkey, 142
atcab_read_pubkey_ext, 142
atcab_read_serial_number, 143
atcab_read_serial_number_ext, 143
atcab_read_sig, 143
atcab_read_zone, 144
atcab_read_zone_ext, 144
atcab_release, 145
atcab_release_ext, 145
atcab_reversal, 145
atcab_secureboot, 146
atcab_secureboot_mac, 146

- atcab_selftest, [147](#)
- atcab_sha, [147](#)
- atcab_sha_base, [148](#)
- atcab_sha_end, [149](#)
- atcab_sha_hmac, [149](#)
- atcab_sha_hmac_ext, [149](#)
- atcab_sha_hmac_finish, [150](#)
- atcab_sha_hmac_init, [151](#)
- atcab_sha_hmac_update, [151](#)
- atcab_sha_read_context, [151](#)
- atcab_sha_start, [152](#)
- atcab_sha_update, [152](#)
- atcab_sha_write_context, [152](#)
- atcab_sign, [153](#)
- atcab_sign_base, [153](#)
- atcab_sign_ext, [154](#)
- atcab_sign_internal, [154](#)
- atcab_sleep, [155](#)
- atcab_updateextra, [155](#)
- atcab_verify, [155](#)
- atcab_verify_extern, [156](#)
- atcab_verify_extern_ext, [157](#)
- atcab_verify_extern_mac, [157](#)
- atcab_verify_invalidate, [158](#)
- atcab_verify_stored, [158](#)
- atcab_verify_stored_ext, [159](#)
- atcab_verify_stored_mac, [159](#)
- atcab_verify_stored_with_tempkey, [160](#)
- atcab_verify_validate, [160](#)
- atcab_version, [161](#)
- atcab_wakeup, [161](#)
- atcab_write, [161](#)
- atcab_write_bytes_zone, [162](#)
- atcab_write_config_counter, [163](#)
- atcab_write_config_zone, [163](#)
- atcab_write_config_zone_ext, [163](#)
- atcab_write_enc, [164](#)
- atcab_write_pubkey, [164](#)
- atcab_write_pubkey_ext, [165](#)
- atcab_write_zone, [165](#)
- atcab_write_zone_ext, [167](#)
- isAlpha, [167](#)
- isBase64, [168](#)
- isBase64Digit, [168](#)
- isBlankSpace, [168](#)
- isDigit, [169](#)
- isHex, [169](#)
- isHexAlpha, [170](#)
- isHexDigit, [170](#)
- packHex, [170](#)
- Basic Crypto API methods for CryptoAuth Devices
 - (calib_), [202](#)
 - calib_ca2_get_addr, [207](#)
 - calib_ca2_is_config_locked, [207](#)
 - calib_ca2_is_data_locked, [208](#)
 - calib_ca2_is_locked, [208](#)
 - calib_exit, [209](#)
 - calib_get_addr, [209](#)
 - calib_get_zone_size, [209](#)
 - calib_idle, [210](#)
 - calib_info, [210](#)
 - calib_info_base, [211](#)
 - calib_info_chip_status, [211](#)
 - calib_info_lock_status, [212](#)
 - calib_info_privkey_valid, [212](#)
 - calib_sleep, [212](#)
 - calib_wakeup, [213](#)
 - calib_wakeup_i2c, [213](#)
 - bind_host_and_secure_element_with_io_protection
 - secure_boot.c, [499](#)
 - secure_boot.h, [501](#)
 - BIT_DELAY_1L
 - hal_swi_gpio.h, [684](#)
 - BIT_DELAY_5
 - hal_swi_gpio.h, [684](#)
 - BIT_DELAY_7
 - hal_swi_gpio.h, [684](#)
 - buf
 - cal_buffer_s, [428](#)
 - cal_buf_read_bytes
 - cal_buffer.c, [583](#)
 - cal_buffer.h, [586](#)
 - cal_buf_read_number
 - cal_buffer.c, [584](#)
 - cal_buffer.h, [586](#)
 - cal_buf_write_bytes
 - cal_buffer.c, [584](#)
 - cal_buffer.h, [587](#)
 - cal_buf_write_number
 - cal_buffer.c, [584](#)
 - cal_buffer.h, [587](#)
 - cal_buffer.c, [582](#)
 - cal_buf_read_bytes, [583](#)
 - cal_buf_read_number, [584](#)
 - cal_buf_write_bytes, [584](#)
 - cal_buf_write_number, [584](#)
 - cal_buffer.h, [585](#)
 - cal_buf_read_bytes, [586](#)
 - cal_buf_read_number, [586](#)
 - cal_buf_write_bytes, [587](#)
 - cal_buf_write_number, [587](#)
 - cal_buffer_s, [427](#)
 - buf, [428](#)
 - len, [428](#)
 - cal_internal.h, [588](#)
 - calib_aes.c, [588](#)
 - calib_aes_gcm.c, [589](#)
 - calib_aes_gcm.h, [589](#)
 - calib_basic.c, [589](#)
 - calib_ca2_get_addr
 - Basic Crypto API methods for CryptoAuth Devices
 - (calib_), [207](#)
 - calib_ca2_is_config_locked
 - Basic Crypto API methods for CryptoAuth Devices
 - (calib_), [207](#)
 - calib_ca2_is_data_locked

- Basic Crypto API methods for CryptoAuth Devices (calib_), 208
- calib_ca2_is_locked
 - Basic Crypto API methods for CryptoAuth Devices (calib_), 208
- calib_checkmac.c, 590
- calib_command.c, 591
 - atCalcCrc, 591
 - atCheckCrc, 592
 - atCRC, 592
 - atInfo, 592
 - atIsECCFamily, 593
 - atIsSHAFamily, 593
 - atPause, 593
 - isATCAError, 594
- calib_command.h, 594
 - atCalcCrc, 612
 - atCheckCrc, 613
 - atCRC, 613
 - atInfo, 613
 - atIsECCFamily, 614
 - atIsSHAFamily, 614
 - atPause, 614
 - isATCAError, 615
- calib_config_check.h, 615
 - CALIB_INFO_LATCH_EN, 617
 - CALIB_LOCK_CA2_EN, 617
 - CALIB_LOCK_EN, 617
 - CALIB_READ_EN, 617
 - CALIB_SHA_CONTEXT_EN, 618
 - CALIB_SHA_EN, 618
 - CALIB_SHA_HMAC_EN, 618
 - CALIB_SIGN_CA2_EN, 618
 - CALIB_SIGN_EN, 618
 - CALIB_UPDATEEXTRA_EN, 618
 - CALIB_VERIFY_EN, 619
 - CALIB_VERIFY_MAC_EN, 619
 - CALIB_VERIFY_STORED_EN, 619
 - CALIB_WRITE_ENC_EN, 619
- calib_counter.c, 620
- calib_delete.c, 620
- calib_derivekey.c, 621
- calib_device.h, 621
- calib_ecdh.c, 624
- calib_execute_command
 - calib_execution.c, 625
 - calib_execution.h, 627
- calib_execution.c, 625
 - calib_execute_command, 625
 - calib_get_execution_time, 626
- calib_execution.h, 626
 - calib_execute_command, 627
 - calib_get_execution_time, 628
- calib_exit
 - Basic Crypto API methods for CryptoAuth Devices (calib_), 209
- calib_gendig.c, 628
- calib_genkey.c, 629
- calib_get_addr
 - Basic Crypto API methods for CryptoAuth Devices (calib_), 209
- calib_get_execution_time
 - calib_execution.c, 626
 - calib_execution.h, 628
- calib_get_zone_size
 - Basic Crypto API methods for CryptoAuth Devices (calib_), 209
- calib_helpers.c, 629
- calib_hmac.c, 630
- calib_idle
 - Basic Crypto API methods for CryptoAuth Devices (calib_), 210
- calib_info
 - Basic Crypto API methods for CryptoAuth Devices (calib_), 210
- calib_info.c, 630
- calib_info_base
 - Basic Crypto API methods for CryptoAuth Devices (calib_), 211
- calib_info_chip_status
 - Basic Crypto API methods for CryptoAuth Devices (calib_), 211
- CALIB_INFO_LATCH_EN
 - calib_config_check.h, 617
- calib_info_lock_status
 - Basic Crypto API methods for CryptoAuth Devices (calib_), 212
- calib_info_privkey_valid
 - Basic Crypto API methods for CryptoAuth Devices (calib_), 212
- calib_kdf.c, 631
- calib_lock.c, 631
- CALIB_LOCK_CA2_EN
 - calib_config_check.h, 617
- CALIB_LOCK_EN
 - calib_config_check.h, 617
- calib_mac.c, 632
- calib_nonce.c, 632
- calib_packet.c, 633
- calib_packet.h, 634
- calib_packet_cache_s, 428
- calib_privwrite.c, 634
- calib_random.c, 635
- calib_read.c, 635
- CALIB_READ_EN
 - calib_config_check.h, 617
- calib_secureboot.c, 636
- calib_selftest.c, 636
- calib_sha.c, 637
- CALIB_SHA_CONTEXT_EN
 - calib_config_check.h, 618
- CALIB_SHA_EN
 - calib_config_check.h, 618
- CALIB_SHA_HMAC_EN
 - calib_config_check.h, 618
- calib_sign.c, 637

- CALIB_SIGN_CA2_EN
 - calib_config_check.h, 618
- CALIB_SIGN_EN
 - calib_config_check.h, 618
- calib_sleep
 - Basic Crypto API methods for CryptoAuth Devices (calib_), 212
- calib_updateextra.c, 638
- CALIB_UPDATEEXTRA_EN
 - calib_config_check.h, 618
- calib_verify.c, 638
- CALIB_VERIFY_EN
 - calib_config_check.h, 619
- CALIB_VERIFY_MAC_EN
 - calib_config_check.h, 619
- CALIB_VERIFY_STORED_EN
 - calib_config_check.h, 619
- calib_wakeup
 - Basic Crypto API methods for CryptoAuth Devices (calib_), 213
- calib_wakeup_i2c
 - Basic Crypto API methods for CryptoAuth Devices (calib_), 213
- calib_write.c, 639
- CALIB_WRITE_ENC_EN
 - calib_config_check.h, 619
- Certificate manipulation methods (atcacert_), 182
 - atcacert_build_state_t, 188
 - atcacert_cert_element_t, 188
 - atcacert_cert_loc_t, 188
 - atcacert_cert_sn_src_e, 189
 - atcacert_cert_sn_src_t, 188
 - atcacert_cert_type_e, 190
 - atcacert_cert_type_t, 188
 - atcacert_date_cmp, 192
 - atcacert_date_dec, 192
 - atcacert_date_dec_compcert, 192
 - atcacert_date_dec_compcert_ext, 193
 - atcacert_date_enc, 194
 - atcacert_date_enc_compcert, 194
 - atcacert_date_enc_compcert_ext, 194
 - atcacert_date_from_asn1_tag, 195
 - atcacert_date_get_max_date, 195
 - atcacert_def_t, 188
 - atcacert_der_dec_ecdsa_sig_value, 196
 - atcacert_der_dec_integer, 196
 - atcacert_der_dec_length, 197
 - atcacert_der_enc_ecdsa_sig_value, 197
 - atcacert_der_enc_integer, 198
 - atcacert_der_enc_length, 198
 - atcacert_device_loc_t, 189
 - atcacert_device_zone_e, 190
 - atcacert_device_zone_t, 189
 - ATCACERT_E_BAD_CERT, 186
 - ATCACERT_E_BAD_PARAMS, 186
 - ATCACERT_E_BUFFER_TOO_SMALL, 186
 - ATCACERT_E_DECODING_ERROR, 186
 - ATCACERT_E_ELEM_MISSING, 186
 - ATCACERT_E_ELEM_OUT_OF_BOUNDS, 187
 - ATCACERT_E_ERROR, 187
 - ATCACERT_E_INVALID_DATE, 187
 - ATCACERT_E_INVALID_TRANSFORM, 187
 - ATCACERT_E_SUCCESS, 187
 - ATCACERT_E_UNEXPECTED_ELEM_SIZE, 187
 - ATCACERT_E_UNIMPLEMENTED, 187
 - ATCACERT_E_VERIFY_FAILED, 187
 - atcacert_get_auth_key_id, 199
 - atcacert_get_cert_sn, 199
 - atcacert_get_expire_date, 200
 - atcacert_get_issue_date, 200
 - atcacert_get_issuer, 201
 - atcacert_get_subj_key_id, 201
 - atcacert_get_subj_public_key, 202
 - atcacert_get_subject, 202
 - atcacert_std_cert_element_e, 191
 - atcacert_std_cert_element_t, 189
 - atcacert_tm_utc_t, 189
 - atcacert_transform_e, 191
 - CERTTYPE_CUSTOM, 190
 - CERTTYPE_X509, 190
 - CERTTYPE_X509_FULL_STORED, 190
 - DATEFMT_ISO8601_SEP, 188
 - DEVZONE_CONFIG, 191
 - DEVZONE_DATA, 191
 - DEVZONE_DEDICATED_DATA, 191
 - DEVZONE_GENKEY, 191
 - DEVZONE_NONE, 191
 - DEVZONE_OTP, 191
 - SNSRC_DEVICE_SN, 190
 - SNSRC_DEVICE_SN_HASH, 190
 - SNSRC_DEVICE_SN_HASH_POS, 190
 - SNSRC_DEVICE_SN_HASH_RAW, 190
 - SNSRC_PUB_KEY_HASH, 190
 - SNSRC_PUB_KEY_HASH_POS, 190
 - SNSRC_PUB_KEY_HASH_RAW, 190
 - SNSRC_SIGNER_ID, 190
 - SNSRC_STORED, 190
 - SNSRC_STORED_DYNAMIC, 190
 - STDCERT_NUM_ELEMENTS, 191
 - TF_BIN2HEX_LC, 191
 - TF_BIN2HEX_SPACE_LC, 191
 - TF_BIN2HEX_SPACE_UC, 191
 - TF_BIN2HEX_UC, 191
 - TF_HEX2BIN_LC, 191
 - TF_HEX2BIN_SPACE_LC, 191
 - TF_HEX2BIN_SPACE_UC, 191
 - TF_HEX2BIN_UC, 191
 - TF_NONE, 191
 - TF_REVERSE, 191
- CERTTYPE_CUSTOM
 - Certificate manipulation methods (atcacert_), 190
- CERTTYPE_X509
 - Certificate manipulation methods (atcacert_), 190
- CERTTYPE_X509_FULL_STORED
 - Certificate manipulation methods (atcacert_), 190
- cfg_ateccx08a_i2c_default

- cryptoauthlib.iface, [333](#)
- cfg_ateccx08a_kithid_default
 - cryptoauthlib.iface, [334](#)
- cfg_ateccx08a_swi_default
 - cryptoauthlib.iface, [334](#)
- cfg_atsha20xa_i2c_default
 - cryptoauthlib.iface, [334](#)
- cfg_atsha20xa_kithid_default
 - cryptoauthlib.iface, [334](#)
- cfg_atsha20xa_swi_default
 - cryptoauthlib.iface, [334](#)
- change_i2c_speed
 - Hardware abstraction layer (hal_), [223](#)
- check_rationality
 - cryptoauthlib.library.AtcaStructure, [418](#)
 - cryptoauthlib.library.AtcaUnion, [419](#)
- check_status
 - cryptoauthlib.status, [347](#)
- CK_AES_CBC_ENCRYPT_DATA_PARAMS, [432](#)
- CK_AES_CCM_PARAMS, [432](#)
- CK_AES_CTR_PARAMS, [432](#)
- CK_AES_GCM_PARAMS, [432](#)
- CK_ARIA_CBC_ENCRYPT_DATA_PARAMS, [432](#)
- CK_ATTRIBUTE, [433](#)
- CK_C_INITIALIZE_ARGS, [433](#)
- CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS, [433](#)
- CK_CAMELLIA_CTR_PARAMS, [433](#)
- CK_CCM_PARAMS, [433](#)
- CK_CMS_SIG_PARAMS, [434](#)
- CK_DATE, [434](#)
- CK_DES_CBC_ENCRYPT_DATA_PARAMS, [434](#)
- CK_DSA_PARAMETER_GEN_PARAM, [434](#)
- CK_ECDH1_DERIVE_PARAMS, [434](#)
- CK_ECDH2_DERIVE_PARAMS, [435](#)
- CK_ECDH_AES_KEY_WRAP_PARAMS, [435](#)
- CK_ECMQV_DERIVE_PARAMS, [435](#)
- CK_FUNCTION_LIST, [435](#)
- CK_GCM_PARAMS, [436](#)
- CK_GOSTR3410_DERIVE_PARAMS, [436](#)
- CK_GOSTR3410_KEY_WRAP_PARAMS, [436](#)
- CK_INFO, [436](#)
- CK_KEA_DERIVE_PARAMS, [437](#)
- CK_KEY_DERIVATION_STRING_DATA, [437](#)
- CK_KEY_WRAP_SET_OAEP_PARAMS, [437](#)
- CK_KIP_PARAMS, [437](#)
- CK_MECHANISM, [437](#)
- CK_MECHANISM_INFO, [438](#)
- CK_OTP_PARAM, [438](#)
- CK_OTP_PARAMS, [438](#)
- CK_OTP_SIGNATURE_INFO, [438](#)
- CK_PBE_PARAMS, [438](#)
- CK_PKCS5_PBKD2_PARAMS, [439](#)
- CK_PKCS5_PBKD2_PARAMS2, [439](#)
- CK_RC2_CBC_PARAMS, [439](#)
- CK_RC2_MAC_GENERAL_PARAMS, [439](#)
- CK_RC5_CBC_PARAMS, [440](#)
- CK_RC5_MAC_GENERAL_PARAMS, [440](#)
- CK_RC5_PARAMS, [440](#)
- CK_RSA_AES_KEY_WRAP_PARAMS, [440](#)
- CK_RSA_PKCS_OAEP_PARAMS, [440](#)
- CK_RSA_PKCS_PSS_PARAMS, [441](#)
- CK_SEED_CBC_ENCRYPT_DATA_PARAMS, [441](#)
- CK_SESSION_INFO, [441](#)
- CK_SKIPJACK_PRIVATE_WRAP_PARAMS, [441](#)
- CK_SKIPJACK_RELAYX_PARAMS, [442](#)
- CK_SLOT_INFO, [442](#)
- CK_SSL3_KEY_MAT_OUT, [442](#)
- CK_SSL3_KEY_MAT_PARAMS, [442](#)
- CK_SSL3_MASTER_KEY_DERIVE_PARAMS, [443](#)
- CK_SSL3_RANDOM_DATA, [443](#)
- CK_TLS12_KEY_MAT_PARAMS, [443](#)
- CK_TLS12_MASTER_KEY_DERIVE_PARAMS, [443](#)
- CK_TLS_KDF_PARAMS, [443](#)
- CK_TLS_MAC_PARAMS, [444](#)
- CK_TLS_PRF_PARAMS, [444](#)
- CK_TOKEN_INFO, [444](#)
- CK_VERSION, [444](#)
- CK_WTLS_KEY_MAT_OUT, [445](#)
- CK_WTLS_KEY_MAT_PARAMS, [445](#)
- CK_WTLS_MASTER_KEY_DERIVE_PARAMS, [445](#)
- CK_WTLS_PRF_PARAMS, [445](#)
- CK_WTLS_RANDOM_DATA, [445](#)
- CK_X9_42_DH1_DERIVE_PARAMS, [446](#)
- CK_X9_42_DH2_DERIVE_PARAMS, [446](#)
- CK_X9_42_MQV_DERIVE_PARAMS, [446](#)
- CL_HashContext, [446](#)
- class_id
 - pkcs11_object_s, [466](#)
- class_type
 - pkcs11_object_s, [466](#)
- config_path
 - pkcs11_lib_ctx_s, [464](#)
- Configuration (cfg_), [171](#)
- count
 - pkcs11_object_s, [466](#)
- crypto_hw_config_check.h, [646](#)
 - ATCAB_AES_CBC_DECRYPT_EN, [647](#)
 - ATCAB_AES_CBC_ENCRYPT_EN, [647](#)
 - ATCAB_AES_CBCMAC_EN, [647](#)
 - ATCAB_AES_CCM_EN, [647](#)
 - ATCAB_AES_CTR_EN, [647](#)
 - ATCAB_AES_CTR_RAND_IV_EN, [648](#)
 - ATCAB_AES_EXTRAS_EN, [648](#)
 - ATCAB_AES_UPDATE_EN, [648](#)
- crypto_sw_config_check.h, [648](#)
 - ATCA_CRYPT_AES_CMEN, [649](#)
 - ATCA_CRYPT_AES_GCM_EN, [649](#)
 - ATCA_CRYPT_SHA1_EN, [650](#)
 - ATCA_CRYPT_SHA256_EN, [650](#)
 - ATCA_CRYPT_SHA2_EN, [650](#)
 - ATCA_CRYPT_SHA2_HMAC_CTR_EN, [650](#)
 - ATCA_CRYPT_SHA2_HMAC_EN, [650](#)
 - ATCA_CRYPT_SHA384_EN, [650](#)
 - ATCA_CRYPT_SHA512_EN, [651](#)
 - ATCAB_PBKDF2_SHA256_EN, [651](#)
 - ATCAC_AES_CMEN, [651](#)

- ATCAC_AES_GCM_EN, [651](#)
- ATCAC_CERT_ADD_EN, [651](#)
- ATCAC_PBKDF2_SHA256_EN, [651](#)
- ATCAC_PKEY_EN, [652](#)
- ATCAC_RANDOM_EN, [652](#)
- ATCAC_SHA1_EN, [652](#)
- ATCAC_SHA256_EN, [652](#)
- ATCAC_SHA384_EN, [652](#)
- ATCAC_SHA512_EN, [652](#)
- ATCAC_SIGN_EN, [653](#)
- ATCAC_VERIFY_EN, [653](#)
- MAX_AES_CMAC_CTX_SIZE, [653](#)
- MAX_AES_GCM_CTX_SIZE, [653](#)
- MAX_HMAC_CTX_SIZE, [653](#)
- cryptoauthlib, [275](#)
- cryptoauthlib.atcab, [275](#)
 - atcab_aes, [278](#)
 - atcab_aes_cbc_decrypt_block, [278](#)
 - atcab_aes_cbc_encrypt_block, [279](#)
 - atcab_aes_cbc_init, [279](#)
 - atcab_aes_cbcmac_finish, [280](#)
 - atcab_aes_cbcmac_init, [280](#)
 - atcab_aes_cbcmac_update, [280](#)
 - atcab_aes_ccm_aad_finish, [281](#)
 - atcab_aes_ccm_aad_update, [281](#)
 - atcab_aes_ccm_decrypt_finish, [281](#)
 - atcab_aes_ccm_decrypt_update, [282](#)
 - atcab_aes_ccm_encrypt_finish, [282](#)
 - atcab_aes_ccm_encrypt_update, [282](#)
 - atcab_aes_ccm_init, [283](#)
 - atcab_aes_ccm_init_rand, [283](#)
 - atcab_aes_cmac_finish, [284](#)
 - atcab_aes_cmac_init, [284](#)
 - atcab_aes_cmac_update, [284](#)
 - atcab_aes_ctr_decrypt_block, [285](#)
 - atcab_aes_ctr_encrypt_block, [285](#)
 - atcab_aes_ctr_init, [285](#)
 - atcab_aes_ctr_init_rand, [286](#)
 - atcab_aes_decrypt, [286](#)
 - atcab_aes_encrypt, [287](#)
 - atcab_aes_gcm_aad_update, [287](#)
 - atcab_aes_gcm_decrypt_finish, [288](#)
 - atcab_aes_gcm_decrypt_update, [288](#)
 - atcab_aes_gcm_encrypt_finish, [288](#)
 - atcab_aes_gcm_encrypt_update, [289](#)
 - atcab_aes_gcm_init, [289](#)
 - atcab_aes_gcm_init_rand, [289](#)
 - atcab_aes_gfm, [290](#)
 - atcab_challenge, [290](#)
 - atcab_challenge_seed_update, [291](#)
 - atcab_checkmac, [291](#)
 - atcab_cmp_config_zone, [291](#)
 - atcab_counter, [292](#)
 - atcab_counter_increment, [292](#)
 - atcab_counter_read, [292](#)
 - atcab_derivekey, [293](#)
 - atcab_ecdh, [293](#)
 - atcab_ecdh_base, [293](#)
 - atcab_ecdh_enc, [294](#)
 - atcab_ecdh_ioenc, [294](#)
 - atcab_ecdh_tempkey, [295](#)
 - atcab_ecdh_tempkey_ioenc, [295](#)
 - atcab_gendig, [296](#)
 - atcab_genkey, [296](#)
 - atcab_genkey_base, [296](#)
 - atcab_get_device, [297](#)
 - atcab_get_device_type, [297](#)
 - atcab_get_pubkey, [297](#)
 - atcab_hmac, [298](#)
 - atcab_hw_sha2_256, [298](#)
 - atcab_hw_sha2_256_finish, [298](#)
 - atcab_hw_sha2_256_init, [299](#)
 - atcab_hw_sha2_256_update, [299](#)
 - atcab_info, [299](#)
 - atcab_info_base, [300](#)
 - atcab_info_get_latch, [300](#)
 - atcab_info_set_latch, [300](#)
 - atcab_init, [301](#)
 - atcab_is_locked, [301](#)
 - atcab_is_slot_locked, [301](#)
 - atcab_kdf, [302](#)
 - atcab_lock, [302](#)
 - atcab_lock_config_zone, [303](#)
 - atcab_lock_config_zone_crc, [303](#)
 - atcab_lock_data_slot, [303](#)
 - atcab_lock_data_zone, [304](#)
 - atcab_lock_data_zone_crc, [304](#)
 - atcab_mac, [304](#)
 - atcab_nonce, [305](#)
 - atcab_nonce_base, [305](#)
 - atcab_nonce_load, [306](#)
 - atcab_nonce_rand, [306](#)
 - atcab_priv_write, [307](#)
 - atcab_random, [307](#)
 - atcab_read_bytes_zone, [307](#)
 - atcab_read_config_zone, [308](#)
 - atcab_read_enc, [308](#)
 - atcab_read_pubkey, [309](#)
 - atcab_read_serial_number, [309](#)
 - atcab_read_sig, [309](#)
 - atcab_read_zone, [310](#)
 - atcab_release, [310](#)
 - atcab_secureboot, [310](#)
 - atcab_secureboot_mac, [311](#)
 - atcab_selftest, [311](#)
 - atcab_sha, [312](#)
 - atcab_sha_base, [312](#)
 - atcab_sha_end, [313](#)
 - atcab_sha_hmac, [313](#)
 - atcab_sha_hmac_finish, [314](#)
 - atcab_sha_hmac_init, [314](#)
 - atcab_sha_hmac_update, [314](#)
 - atcab_sha_read_context, [315](#)
 - atcab_sha_start, [315](#)
 - atcab_sha_update, [315](#)
 - atcab_sha_write_context, [316](#)

- atcab_sign, 316
- atcab_sign_base, 316
- atcab_sign_internal, 317
- atcab_updateextra, 317
- atcab_verify, 317
- atcab_verify_extern, 318
- atcab_verify_extern_mac, 319
- atcab_verify_extern_stored_mac, 319
- atcab_verify_invalidate, 320
- atcab_verify_stored, 320
- atcab_verify_stored_mac, 321
- atcab_verify_validate, 321
- atcab_write, 322
- atcab_write_bytes_zone, 322
- atcab_write_config_counter, 323
- atcab_write_config_zone, 323
- atcab_write_enc, 323
- atcab_write_pubkey, 324
- atcab_write_zone, 324
- cryptoauthlib.atcab.atca_aes_cbc_ctx, 367
 - _fields_, 367
- cryptoauthlib.atcab.atca_aes_cbcmac_ctx, 367
 - _fields_, 368
- cryptoauthlib.atcab.atca_aes_ccm_ctx, 368
 - _fields_, 368
- cryptoauthlib.atcab.atca_aes_cmac_ctx, 369
 - _fields_, 369
- cryptoauthlib.atcab.atca_aes_ctr_ctx, 370
 - _fields_, 370
- cryptoauthlib.atcab.atca_aes_gcm_ctx, 370
 - _fields_, 371
- cryptoauthlib.atcab.atca_hmac_sha256_ctx, 379
- cryptoauthlib.atcab.atca_sha256_ctx, 385
 - _fields_, 385
- cryptoauthlib.atcacert, 325
 - _atcacert_convert_bytes, 326
 - _atcacert_convert_enum, 326
 - atcacert_create_csr, 326
 - atcacert_create_csr_pem, 326
 - atcacert_date_dec, 327
 - atcacert_date_dec_compcert, 327
 - atcacert_date_enc, 328
 - atcacert_date_enc_compcert, 328
 - atcacert_date_get_max_date, 328
 - atcacert_get_response, 329
 - atcacert_max_cert_size, 329
 - atcacert_read_cert, 329
 - atcacert_write_cert, 330
- cryptoauthlib.atcacert.atcacert_cert_element_t, 396
 - _def_, 397
- cryptoauthlib.atcacert.atcacert_cert_loc_t, 397
- cryptoauthlib.atcacert.atcacert_cert_sn_src_t, 398
- cryptoauthlib.atcacert.atcacert_cert_type_t, 399
- cryptoauthlib.atcacert.atcacert_comp_data_t, 400
 - _fields_, 401
- cryptoauthlib.atcacert.atcacert_date_format_t, 401
- cryptoauthlib.atcacert.atcacert_def_t, 402
- cryptoauthlib.atcacert.atcacert_device_loc_t, 404
 - _def_, 404
- cryptoauthlib.atcacert.atcacert_device_zone_t, 405
- cryptoauthlib.atcacert.atcacert_std_cert_element_t, 405
- cryptoauthlib.atcacert.atcacert_tm_utc_t, 407
 - _fields_, 407
- cryptoauthlib.atcacert.atcacert_transform_t, 407
- cryptoauthlib.atcacert.CertStatus, 428
- cryptoauthlib.atcaenum, 330
- cryptoauthlib.atcaenum.AtcaEnum, 410
- cryptoauthlib.atjwt, 331
- cryptoauthlib.atjwt.HwEcAlgorithm, 453
 - sign, 454
- cryptoauthlib.atjwt.HwHmacAlgorithm, 454
 - sign, 454
 - verify, 455
- cryptoauthlib.atjwt.PyJWT, 468
- cryptoauthlib.device, 331
- cryptoauthlib.device.AesEnable, 366
 - _fields_, 366
- cryptoauthlib.device.Atecc508aConfig, 420
 - _fields_, 421
- cryptoauthlib.device.Atecc608Config, 422
 - _fields_, 423
- cryptoauthlib.device.Atsha204aConfig, 424
 - _fields_, 425
- cryptoauthlib.device.ChipMode508, 429
 - _fields_, 430
- cryptoauthlib.device.ChipMode608, 430
 - _fields_, 430
- cryptoauthlib.device.ChipOptions, 431
 - _fields_, 431
- cryptoauthlib.device.Counter204, 447
 - _fields_, 448
- cryptoauthlib.device.CountMatch, 448
 - _fields_, 448
- cryptoauthlib.device.I2cEnable, 456
 - _fields_, 456
- cryptoauthlib.device.KeyConfig, 457
 - _fields_, 457
- cryptoauthlib.device.SecureBoot, 470
 - _fields_, 470
- cryptoauthlib.device.SlotConfig, 471
 - _fields_, 471
- cryptoauthlib.device.UseLock, 475
 - _fields_, 475
- cryptoauthlib.device.VolatileKeyPermission, 475
 - _fields_, 476
- cryptoauthlib.device.X509Format, 476
 - _fields_, 477
- cryptoauthlib.exceptions, 332
- cryptoauthlib.exceptions.AssertionFailure, 366
- cryptoauthlib.exceptions.BadArgumentError, 425
- cryptoauthlib.exceptions.BadCrcError, 426
- cryptoauthlib.exceptions.BadOpcodeError, 426
- cryptoauthlib.exceptions.CheckmacVerifyFailedError, 429
- cryptoauthlib.exceptions.CommunicationError, 447
- cryptoauthlib.exceptions.ConfigZoneLockedError, 447

- cryptoauthlib.exceptions.CrcError, [449](#)
- cryptoauthlib.exceptions.CryptoError, [450](#)
- cryptoauthlib.exceptions.DataZoneLockedError, [450](#)
- cryptoauthlib.exceptions.EccFaultError, [451](#)
- cryptoauthlib.exceptions.ExecutionError, [452](#)
- cryptoauthlib.exceptions.FunctionError, [452](#)
- cryptoauthlib.exceptions.GenericError, [452](#)
- cryptoauthlib.exceptions.HealthTestError, [453](#)
- cryptoauthlib.exceptions.InvalidIdentifierError, [456](#)
- cryptoauthlib.exceptions.InvalidSizeError, [457](#)
- cryptoauthlib.exceptions.LibraryLoadError, [458](#)
- cryptoauthlib.exceptions.LibraryMemoryError, [458](#)
- cryptoauthlib.exceptions.LibraryNotInitialized, [459](#)
- cryptoauthlib.exceptions.NoDevicesFoundError, [459](#)
- cryptoauthlib.exceptions.NoResponseError, [460](#)
- cryptoauthlib.exceptions.NoUseFlagError, [460](#)
- cryptoauthlib.exceptions.ParityError, [460](#)
- cryptoauthlib.exceptions.ParseError, [461](#)
- cryptoauthlib.exceptions.ReceiveError, [468](#)
- cryptoauthlib.exceptions.ReceiveTimeoutError, [469](#)
- cryptoauthlib.exceptions.ResyncWithWakeupError, [469](#)
- cryptoauthlib.exceptions.StatusUnknownError, [473](#)
- cryptoauthlib.exceptions.TimeOutError, [473](#)
- cryptoauthlib.exceptions.TransmissionError, [473](#)
- cryptoauthlib.exceptions.TransmissionTimeoutError, [474](#)
- cryptoauthlib.exceptions.UnimplementedError, [474](#)
- cryptoauthlib.exceptions.UnsupportedInterface, [474](#)
- cryptoauthlib.exceptions.WakeFailedError, [476](#)
- cryptoauthlib.exceptions.ZoneNotLockedError, [477](#)
- cryptoauthlib.h, [656](#)
 - ATCA_SHA256_BLOCK_SIZE, [658](#)
 - SHA_MODE_TARGET_MSGDIGBUF, [658](#)
 - SHA_MODE_TARGET_OUT_ONLY, [658](#)
 - SHA_MODE_TARGET_TEMPKEY, [658](#)
- cryptoauthlib.iface, [333](#)
 - _iface_load_default_config, [333](#)
 - cfg_ateccx08a_i2c_default, [333](#)
 - cfg_ateccx08a_kithid_default, [334](#)
 - cfg_ateccx08a_swi_default, [334](#)
 - cfg_atsha20xa_i2c_default, [334](#)
 - cfg_atsha20xa_kithid_default, [334](#)
 - cfg_atsha20xa_swi_default, [334](#)
- cryptoauthlib.iface._ATCACUSTOM, [355](#)
 - _fields_, [356](#)
- cryptoauthlib.iface._ATCAHID, [356](#)
 - _def_, [357](#)
- cryptoauthlib.iface._ATCAI2C, [357](#)
 - _fields_, [358](#)
 - _map_, [358](#)
- cryptoauthlib.iface._ATCAIfaceParams, [359](#)
 - _fields_, [359](#)
- cryptoauthlib.iface._ATCAKIT, [360](#)
 - _def_, [360](#)
- cryptoauthlib.iface._ATCASPI, [361](#)
 - _fields_, [361](#)
- cryptoauthlib.iface._ATCASWI, [362](#)
 - _fields_, [362](#)
- cryptoauthlib.iface._ATCAUART, [363](#)
 - _def_, [363](#)
- cryptoauthlib.iface._U_Address, [365](#)
 - _fields_, [365](#)
- cryptoauthlib.iface.ATCADeviceType, [408](#)
- cryptoauthlib.iface.ATCAIfaceCfg, [413](#)
 - _def_, [414](#)
 - _map_, [414](#)
- cryptoauthlib.iface.ATCAIfaceType, [414](#)
- cryptoauthlib.iface.ATCAKitType, [415](#)
- cryptoauthlib.library, [335](#)
 - _array_to_code, [336](#)
 - _check_type_rationality, [336](#)
 - _convert_pointer_to_list, [336](#)
 - _ctype_from_definition, [336](#)
 - _def_to_field, [337](#)
 - _force_local_library, [337](#)
 - _get_attribute_from_ctypes, [337](#)
 - _get_field_definition, [337](#)
 - _is_pointer, [338](#)
 - _obj_to_code, [338](#)
 - _object_definition_code, [338](#)
 - _pointer_to_code, [338](#)
 - _structure_to_code, [339](#)
 - _structure_to_string, [339](#)
 - _to_code, [339](#)
 - ctypes_to_bytes, [339](#)
 - get_cryptoauthlib, [340](#)
 - get_ctype_array_instance, [340](#)
 - get_ctype_by_name, [340](#)
 - get_ctype_structure_instance, [340](#)
 - get_device_name, [340](#)
 - get_device_name_with_device_id, [341](#)
 - get_device_type_id, [341](#)
 - get_size_by_name, [341](#)
 - load_cryptoauthlib, [341](#)
- cryptoauthlib.library._Cypeliterator, [364](#)
- cryptoauthlib.library.AtcaReference, [416](#)
- cryptoauthlib.library.AtcaStructure, [417](#)
 - check_rationality, [418](#)
 - from_definition, [418](#)
- cryptoauthlib.library.AtcaUnion, [419](#)
 - check_rationality, [419](#)
 - from_definition, [419](#)
- cryptoauthlib.library.cal_buffer, [427](#)
 - _fields_, [427](#)
- cryptoauthlib.sha206_api, [342](#)
 - sha206a_authenticate, [342](#)
 - sha206a_check_dk_useflag_validity, [342](#)
 - sha206a_check_pk_useflag_validity, [343](#)
 - sha206a_diversify_parent_key, [343](#)
 - sha206a_generate_challenge_response_pair, [343](#)
 - sha206a_generate_derive_key, [344](#)
 - sha206a_get_data_store_lock_status, [344](#)
 - sha206a_get_dk_update_count, [345](#)
 - sha206a_get_dk_useflag_count, [345](#)
 - sha206a_get_pk_useflag_count, [345](#)
 - sha206a_read_data_store, [345](#)

- sha206a_verify_device_consumption, 346
- sha206a_write_data_store, 346
- cryptoauthlib.status, 347
 - check_status, 347
- cryptoauthlib.status.Status, 471
- cryptoauthlib.tng, 348
 - tng_atcacert_device_public_key, 348
 - tng_atcacert_max_device_cert_size, 348
 - tng_atcacert_max_signer_cert_size, 349
 - tng_atcacert_read_device_cert, 349
 - tng_atcacert_read_signer_cert, 349
 - tng_atcacert_root_cert, 350
 - tng_atcacert_root_cert_size, 350
 - tng_atcacert_root_public_key, 350
 - tng_atcacert_signer_public_key, 351
 - tng_get_device_pubkey, 351
- cryptoauthlib_mock.atcab_mock, 389
- ctypes_to_bytes
 - cryptoauthlib.library, 339
- DATEFMT_ISO8601_SEP
 - Certificate manipulation methods (atcacert_), 188
- deleteATCADevice
 - ATCADevice (atca_), 173
- deleteATCAIface
 - ATCAIface (atca_), 180
- dev_lock
 - pkcs11_dev_state, 462
- dev_lock_enabled
 - pkcs11_lib_ctx_s, 464
- dev_state
 - pkcs11_lib_ctx_s, 464
- device_execution_time_t, 451
- device_state
 - atca_device, 374
- devtype_names_t, 451
- DEVZONE_CONFIG
 - Certificate manipulation methods (atcacert_), 191
- DEVZONE_DATA
 - Certificate manipulation methods (atcacert_), 191
- DEVZONE_DEDICATED_DATA
 - Certificate manipulation methods (atcacert_), 191
- DEVZONE_GENKEY
 - Certificate manipulation methods (atcacert_), 191
- DEVZONE_NONE
 - Certificate manipulation methods (atcacert_), 191
- DEVZONE_OTP
 - Certificate manipulation methods (atcacert_), 191
- ec_key_data_table
 - Attributes (pkcs11_attrib_), 271
- from_definition
 - cryptoauthlib.library.AtcaStructure, 418
 - cryptoauthlib.library.AtcaUnion, 419
- get_cryptoauthlib
 - cryptoauthlib.library, 340
- get_ctype_array_instance
 - cryptoauthlib.library, 340
- get_ctype_by_name
 - cryptoauthlib.library, 340
- get_ctype_structure_instance
 - cryptoauthlib.library, 340
- get_device_name
 - cryptoauthlib.library, 340
- get_device_name_with_device_id
 - cryptoauthlib.library, 341
- get_device_type_id
 - cryptoauthlib.library, 341
- get_size_by_name
 - cryptoauthlib.library, 341
- hal
 - atca_iface, 380
 - hal_all_platforms_kit_hidapi.c, 661
 - hal_check_wake
 - Hardware abstraction layer (hal_), 224
 - hal_create_mutex
 - Hardware abstraction layer (hal_), 224
 - hal_data
 - atca_hal_kit_phy_t, 377
 - atca_iface, 380
 - hal_delay_ms
 - Hardware abstraction layer (hal_), 224
 - hal_delay_us
 - Hardware abstraction layer (hal_), 225
 - hal_freertos.c, 662
 - hal_gpio_harmony.c, 662
 - hal_gpio_init, 663
 - hal_gpio_post_init, 663
 - hal_gpio_receive, 663
 - hal_gpio_release, 664
 - hal_gpio_send, 664
 - hal_gpio_init
 - hal_gpio_harmony.c, 663
 - hal_gpio_post_init
 - hal_gpio_harmony.c, 663
 - hal_gpio_receive
 - hal_gpio_harmony.c, 663
 - hal_gpio_release
 - hal_gpio_harmony.c, 664
 - hal_gpio_send
 - hal_gpio_harmony.c, 664
 - hal_i2c_control
 - Hardware abstraction layer (hal_), 225
 - hal_i2c_discover_buses
 - Hardware abstraction layer (hal_), 226
 - hal_i2c_discover_devices
 - Hardware abstraction layer (hal_), 227
 - hal_i2c_harmony.c, 665
 - hal_i2c_idle
 - Hardware abstraction layer (hal_), 227
 - hal_i2c_init
 - Hardware abstraction layer (hal_), 228, 229
 - hal_i2c_post_init
 - Hardware abstraction layer (hal_), 230
 - hal_i2c_receive

- Hardware abstraction layer (hal_), 231
- hal_i2c_release
 - Hardware abstraction layer (hal_), 232
- hal_i2c_send
 - Hardware abstraction layer (hal_), 233
- hal_i2c_sleep
 - Hardware abstraction layer (hal_), 234
- hal_i2c_start.c, 666
- hal_i2c_start.h, 667
- hal_i2c_wake
 - Hardware abstraction layer (hal_), 235
- hal_iface_init
 - Hardware abstraction layer (hal_), 235
- hal_iface_register_hal
 - Hardware abstraction layer (hal_), 236
- hal_iface_release
 - Hardware abstraction layer (hal_), 236
- hal_is_command_word
 - Hardware abstraction layer (hal_), 236
- hal_kit_attach_phy
 - Hardware abstraction layer (hal_), 237
- hal_kit_bridge.c, 667
- hal_kit_bridge.h, 668
- hal_kit_control
 - Hardware abstraction layer (hal_), 237
- hal_kit_hid_control
 - Hardware abstraction layer (hal_), 237
- hal_kit_hid_init
 - Hardware abstraction layer (hal_), 238
- hal_kit_hid_post_init
 - Hardware abstraction layer (hal_), 238
- hal_kit_hid_receive
 - Hardware abstraction layer (hal_), 239
- hal_kit_hid_release
 - Hardware abstraction layer (hal_), 239
- hal_kit_hid_send
 - Hardware abstraction layer (hal_), 239
- hal_kit_init
 - Hardware abstraction layer (hal_), 240
- hal_kit_post_init
 - Hardware abstraction layer (hal_), 240
- hal_kit_receive
 - Hardware abstraction layer (hal_), 241
- hal_kit_release
 - Hardware abstraction layer (hal_), 241
- hal_kit_send
 - Hardware abstraction layer (hal_), 241
- hal_linux.c, 669
- hal_linux_i2c_userspace.c, 669
- hal_linux_uart_userspace.c, 670
 - hal_uart_control, 671
 - hal_uart_init, 672
 - hal_uart_post_init, 672
 - hal_uart_receive, 672
 - hal_uart_release, 673
 - hal_uart_send, 673
- hal_rtos_delay_ms
 - Hardware abstraction layer (hal_), 242
- hal_sam0_i2c_asf.c, 674
- hal_sam0_i2c_asf.h, 675
- hal_sam_i2c_asf.c, 675
- hal_sam_i2c_asf.h, 676
- hal_sam_timer_asf.c, 677
- hal_spi_control
 - Hardware abstraction layer (hal_), 242
- hal_spi_deselect
 - Hardware abstraction layer (hal_), 243
- hal_spi_discover_buses
 - Hardware abstraction layer (hal_), 243
- hal_spi_discover_devices
 - Hardware abstraction layer (hal_), 243
- hal_spi_harmony.c, 678
- hal_spi_init
 - Hardware abstraction layer (hal_), 244
- hal_spi_post_init
 - Hardware abstraction layer (hal_), 244
- hal_spi_receive
 - Hardware abstraction layer (hal_), 245
- hal_spi_release
 - Hardware abstraction layer (hal_), 245
- hal_spi_select
 - Hardware abstraction layer (hal_), 245
- hal_spi_send
 - Hardware abstraction layer (hal_), 246
- hal_swi_control
 - Hardware abstraction layer (hal_), 246
- hal_swi_gpio.c, 679
 - hal_swi_gpio_control, 679
 - hal_swi_gpio_init, 680
 - hal_swi_gpio_post_init, 680
 - hal_swi_gpio_receive, 680
 - hal_swi_gpio_release, 681
 - hal_swi_gpio_send, 681
- hal_swi_gpio.h, 682
 - ATCA_SWI_WAKE_WORD_ADDR, 684
 - BIT_DELAY_1L, 684
 - BIT_DELAY_5, 684
 - BIT_DELAY_7, 684
 - RX_TX_DELAY, 684
- hal_swi_gpio_control
 - hal_swi_gpio.c, 679
- hal_swi_gpio_init
 - hal_swi_gpio.c, 680
- hal_swi_gpio_post_init
 - hal_swi_gpio.c, 680
- hal_swi_gpio_receive
 - hal_swi_gpio.c, 680
- hal_swi_gpio_release
 - hal_swi_gpio.c, 681
- hal_swi_gpio_send
 - hal_swi_gpio.c, 681
- hal_swi_idle
 - Hardware abstraction layer (hal_), 247
- hal_swi_init
 - Hardware abstraction layer (hal_), 247
- hal_swi_post_init

Hardware abstraction layer (hal_), 247
 hal_swi_receive
 Hardware abstraction layer (hal_), 248
 hal_swi_release
 Hardware abstraction layer (hal_), 248
 hal_swi_send
 Hardware abstraction layer (hal_), 249
 hal_swi_sleep
 Hardware abstraction layer (hal_), 249
 hal_swi_uart.c, 684
 hal_swi_wake
 Hardware abstraction layer (hal_), 250
 hal_timer_start.c, 685
 hal_uart_control
 hal_linux_uart_userspace.c, 671
 hal_windows_kit_uart.c, 694
 hal_uart_harmony.c, 686
 hal_uart_init, 687
 hal_uart_post_init, 688
 hal_uart_receive, 688
 hal_uart_release, 689
 hal_uart_send, 689
 serial_setup, 689
 hal_uart_init
 hal_linux_uart_userspace.c, 672
 hal_uart_harmony.c, 687
 hal_windows_kit_uart.c, 694
 hal_uart_post_init
 hal_linux_uart_userspace.c, 672
 hal_uart_harmony.c, 688
 hal_windows_kit_uart.c, 695
 hal_uart_receive
 hal_linux_uart_userspace.c, 672
 hal_uart_harmony.c, 688
 hal_windows_kit_uart.c, 695
 hal_uart_release
 hal_linux_uart_userspace.c, 673
 hal_uart_harmony.c, 689
 hal_windows_kit_uart.c, 695
 hal_uart_send
 hal_linux_uart_userspace.c, 673
 hal_uart_harmony.c, 689
 hal_windows_kit_uart.c, 697
 hal_uc3_i2c_asf.c, 690
 hal_uc3_i2c_asf.h, 691
 hal_uc3_timer_asf.c, 692
 hal_windows.c, 692
 hal_windows_kit_uart.c, 693
 hal_uart_control, 694
 hal_uart_init, 694
 hal_uart_post_init, 695
 hal_uart_receive, 695
 hal_uart_release, 695
 hal_uart_send, 697
 handle
 pkcs11_object_cache_s, 465
 Hardware abstraction layer (hal_), 214
 atca_delay_10us, 222
 atca_delay_ms, 222
 atca_delay_us, 223
 change_i2c_speed, 223
 hal_check_wake, 224
 hal_create_mutex, 224
 hal_delay_ms, 224
 hal_delay_us, 225
 hal_i2c_control, 225
 hal_i2c_discover_buses, 226
 hal_i2c_discover_devices, 227
 hal_i2c_idle, 227
 hal_i2c_init, 228, 229
 hal_i2c_post_init, 230
 hal_i2c_receive, 231
 hal_i2c_release, 232
 hal_i2c_send, 233
 hal_i2c_sleep, 234
 hal_i2c_wake, 235
 hal_iface_init, 235
 hal_iface_register_hal, 236
 hal_iface_release, 236
 hal_is_command_word, 236
 hal_kit_attach_phy, 237
 hal_kit_control, 237
 hal_kit_hid_control, 237
 hal_kit_hid_init, 238
 hal_kit_hid_post_init, 238
 hal_kit_hid_receive, 239
 hal_kit_hid_release, 239
 hal_kit_hid_send, 239
 hal_kit_init, 240
 hal_kit_post_init, 240
 hal_kit_receive, 241
 hal_kit_release, 241
 hal_kit_send, 241
 hal_rtos_delay_ms, 242
 hal_spi_control, 242
 hal_spi_deselect, 243
 hal_spi_discover_buses, 243
 hal_spi_discover_devices, 243
 hal_spi_init, 244
 hal_spi_post_init, 244
 hal_spi_receive, 245
 hal_spi_release, 245
 hal_spi_select, 245
 hal_spi_send, 246
 hal_swi_control, 246
 hal_swi_idle, 247
 hal_swi_init, 247
 hal_swi_post_init, 247
 hal_swi_receive, 248
 hal_swi_release, 248
 hal_swi_send, 249
 hal_swi_sleep, 249
 hal_swi_wake, 250
 kit_id_from_devtype, 250
 kit_interface, 250
 kit_interface_from_kittype, 250

- MAX_SWI_BUSES, 221
- swi_uart_deinit, 250
- swi_uart_discover_buses, 251
- swi_uart_init, 251
- swi_uart_mode, 252
- swi_uart_receive_byte, 252
- swi_uart_send_byte, 253
- swi_uart_setbaud, 253
- Host side crypto methods (atcah_), 253
- HOSTLIB_CERT_EN
 - atca_mbedtls_interface.h, 715
 - atca_openssl_interface.h, 741
- i2c_sam0_instance, 455
- i2c_sam_instance, 455
- i2c_start_instance, 455
- ifacecfg_set_address
 - ATCAIface (atca_), 181
- ifacetype_is_kit
 - ATCAIface (atca_), 181
- init_args
 - pkcs11_lib_ctx_s, 464
- initATCADevice
 - ATCADevice (atca_), 173
- initATCAIface
 - ATCAIface (atca_), 181
- initialized
 - pkcs11_lib_ctx_s, 464
- io_protection_key.h, 498
- isAlpha
 - atca_helpers.c, 546
 - Basic Crypto API methods (atcab_), 167
- isATCAError
 - calib_command.c, 594
 - calib_command.h, 615
- isBase64
 - atca_helpers.c, 547
 - Basic Crypto API methods (atcab_), 168
- isBase64Digit
 - atca_helpers.c, 547
 - Basic Crypto API methods (atcab_), 168
- isBlankSpace
 - atca_helpers.c, 547
 - Basic Crypto API methods (atcab_), 168
- isDigit
 - atca_helpers.c, 548
 - Basic Crypto API methods (atcab_), 169
- isHex
 - atca_helpers.c, 548
 - Basic Crypto API methods (atcab_), 169
- isHexAlpha
 - atca_helpers.c, 548
 - Basic Crypto API methods (atcab_), 170
- isHexDigit
 - atca_helpers.c, 549
 - Basic Crypto API methods (atcab_), 170
- JSON Web Token (JWT) methods (atca_jwt_), 258
- key_data_table
 - Attributes (pkcs11_attrib_), 271
- kit_host_init
 - ascii_kit_host.c, 494
 - ascii_kit_host.h, 497
- kit_host_init_phy
 - ascii_kit_host.c, 495
 - ascii_kit_host.h, 498
- kit_host_map_entry_t
 - ascii_kit_host.h, 497
- kit_id_from_devtype
 - Hardware abstraction layer (hal_), 250
- kit_interface
 - Hardware abstraction layer (hal_), 250
- kit_interface_from_kittype
 - Hardware abstraction layer (hal_), 250
- KIT_MESSAGE_SIZE_MAX
 - ascii_kit_host.h, 496
- kit_protocol.c, 697
- kit_protocol.h, 698
- len
 - cal_buffer_s, 428
- lib_lock
 - pkcs11_lib_ctx_s, 464
- LIBRARY_BUILD_EN_CHECK
 - atca_config_check.h, 538
- load_cryptoauthlib
 - cryptoauthlib.library, 341
- MAX_AES_CMAC_CTX_SIZE
 - crypto_sw_config_check.h, 653
- MAX_AES_GCM_CTX_SIZE
 - crypto_sw_config_check.h, 653
- MAX_HMAC_CTX_SIZE
 - crypto_sw_config_check.h, 653
- MAX_SWI_BUSES
 - Hardware abstraction layer (hal_), 221
- mbedtls Wrapper methods (atca_mbedtls_), 258
 - atca_mbedtls_eckey_t, 259
 - atca_mbedtls_pk_init, 259
 - atca_mbedtls_pk_init_ext, 260
 - atca_mbedtls_random_ctx, 260
- memory_parameters, 459
- mlface
 - atca_device, 374
- mlfaceCFG
 - atca_iface, 380
- MULTIPART_BUF_EN
 - atca_config_check.h, 538
- newATCADevice
 - ATCADevice (atca_), 173
- object
 - pkcs11_object_cache_s, 465
- packet_alloc
 - atca_hal_kit_phy_t, 377

packet_free
 atca_hal_kit_phy_t, 377
 packHex
 atca_helpers.c, 549
 Basic Crypto API methods (atcab_), 170
 pkcs11_mech_table_e, 461
 phy
 atca_hal_list_entry_t, 378
 atca_iface, 380
 pkcs11_attr.c, 742
 pkcs11_attr.h, 742
 attrib_f, 743
 pkcs11_attr_fill
 Attributes (pkcs11_attr_), 269
 pkcs11_attr_model_s, 461
 pkcs11_cert.c, 743
 pkcs11_cert.h, 744
 pkcs11_cert_wtlspublic_attributes
 Attributes (pkcs11_attr_), 271
 pkcs11_cert_x509_attributes
 Attributes (pkcs11_attr_), 271
 pkcs11_cert_x509public_attributes
 Attributes (pkcs11_attr_), 272
 pkcs11_conf_filedata_s, 461
 pkcs11_config.c, 745
 pkcs11_debug.c, 746
 pkcs11_debug.h, 747
 pkcs11_deinit
 Attributes (pkcs11_attr_), 269
 pkcs11_dev_ctx, 462
 pkcs11_dev_res, 462
 pkcs11_dev_state, 462
 dev_lock, 462
 resources, 463
 pkcs11_digest.h, 747
 pkcs11_ec_pbkey_asn1_hdr_p224
 Attributes (pkcs11_attr_), 272
 pkcs11_ec_pbkey_asn1_hdr_p256
 Attributes (pkcs11_attr_), 272
 pkcs11_ec_pbkey_asn1_hdr_p384
 Attributes (pkcs11_attr_), 272
 pkcs11_ec_pbkey_asn1_hdr_p521
 Attributes (pkcs11_attr_), 272
 pkcs11_ecc_key_info_s, 463
 pkcs11_encrypt.c, 748
 pkcs11_encrypt.h, 748
 pkcs11_find.c, 749
 pkcs11_find.h, 750
 pkcs11_info.c, 750
 pkcs11_info.h, 751
 pkcs11_init
 Attributes (pkcs11_attr_), 270
 pkcs11_init.c, 752
 pkcs11_init.h, 752
 pkcs11_lib_ctx, 753
 pkcs11_key.c, 754
 pkcs11_key.h, 755
 pkcs11_key_info_s, 463
 pkcs11_key_private_attributes
 Attributes (pkcs11_attr_), 273
 pkcs11_key_public_attributes
 Attributes (pkcs11_attr_), 273
 pkcs11_key_secret_attributes
 Attributes (pkcs11_attr_), 273
 pkcs11_lib_ctx
 pkcs11_init.h, 753
 pkcs11_lib_ctx_s, 463
 config_path, 464
 dev_lock_enabled, 464
 dev_state, 464
 init_args, 464
 initialized, 464
 lib_lock, 464
 slot_cnt, 464
 slots, 465
 pkcs11_main.c, 757
 pkcs11_mech.c, 761
 pkcs11_mech.h, 762
 pkcs11_object.c, 763
 pkcs11_object.h, 764
 pkcs11_object_cache_s, 465
 handle, 465
 object, 465
 pkcs11_object_monotonic_attributes
 Attributes (pkcs11_attr_), 273
 pkcs11_object_s, 465
 attributes, 466
 class_id, 466
 class_type, 466
 count, 466
 pkcs11_os.c, 765
 pkcs11_os.h, 766
 pkcs11_os_create_mutex
 Attributes (pkcs11_attr_), 270
 pkcs11_rsa_key_info_s, 466
 pkcs11_session.c, 767
 pkcs11_session.h, 767
 pkcs11_session_ctx, 768
 pkcs11_session_closeall
 Attributes (pkcs11_attr_), 270
 pkcs11_session_ctx
 pkcs11_session.h, 768
 pkcs11_session_ctx_s, 466
 pkcs11_session_login
 Attributes (pkcs11_attr_), 270
 pkcs11_session_mech_ctx_s, 467
 pkcs11_signature.c, 769
 pkcs11_signature.h, 770
 pkcs11_slot.c, 770
 pkcs11_slot.h, 771
 pkcs11_slot_ctx, 772
 pkcs11_slot_ctx
 pkcs11_slot.h, 772
 pkcs11_slot_ctx_s, 467
 read_key, 468
 pkcs11_token.c, 772

pkcs11_token.h, [773](#)
pkcs11_token_init
 Attributes (pkcs11_attrib_), [270](#)
pkcs11_util.c, [774](#)
pkcs11_util.h, [775](#)
pkcs11_x962_asn1_hdr_ec224
 Attributes (pkcs11_attrib_), [273](#)
pkcs11_x962_asn1_hdr_ec256
 Attributes (pkcs11_attrib_), [274](#)
pkcs11_x962_asn1_hdr_ec384
 Attributes (pkcs11_attrib_), [274](#)
pkcs11_x962_asn1_hdr_ec521
 Attributes (pkcs11_attrib_), [274](#)

read_key
 pkcs11_slot_ctx_s, [468](#)
recv
 atca_hal_kit_phy_t, [377](#)
releaseATCADevice
 ATCADevice (atca_), [174](#)
releaseATCAIface
 ATCAIface (atca_), [182](#)
resources
 pkcs11_dev_state, [463](#)
rsa_key_data_table
 Attributes (pkcs11_attrib_), [274](#)
RX_TX_DELAY
 hal_swi_gpio.h, [684](#)

secure_boot.c, [499](#)
 bind_host_and_secure_element_with_io_protection,
 [499](#)
 secure_boot_process, [500](#)
secure_boot.h, [500](#)
 bind_host_and_secure_element_with_io_protection,
 [501](#)
 secure_boot_process, [501](#)
secure_boot_config_bits, [469](#)
secure_boot_memory.h, [502](#)
secure_boot_parameters, [470](#)
secure_boot_process
 secure_boot.c, [500](#)
 secure_boot.h, [501](#)
send
 atca_hal_kit_phy_t, [377](#)
serial_setup
 hal_uart_harmony.c, [689](#)
setup.BinaryDistribution, [426](#)
setup.CryptoAuthCommandBuildExt, [449](#)
setup.CryptoAuthCommandInstall, [449](#)
sha1_routines.c, [654](#)
sha1_routines.h, [654](#)
sha206a_authenticate
 api_206a.c, [480](#)
 api_206a.h, [486](#)
 cryptoauthlib.sha206_api, [342](#)
sha206a_check_dk_useflag_validity
 api_206a.c, [480](#)
 api_206a.h, [487](#)
 cryptoauthlib.sha206_api, [342](#)
sha206a_check_pk_useflag_validity
 api_206a.c, [481](#)
 api_206a.h, [487](#)
 cryptoauthlib.sha206_api, [343](#)
sha206a_diversify_parent_key
 api_206a.c, [481](#)
 api_206a.h, [487](#)
 cryptoauthlib.sha206_api, [343](#)
sha206a_generate_challenge_response_pair
 api_206a.c, [481](#)
 api_206a.h, [488](#)
 cryptoauthlib.sha206_api, [343](#)
sha206a_generate_derive_key
 api_206a.c, [482](#)
 api_206a.h, [488](#)
 cryptoauthlib.sha206_api, [344](#)
sha206a_get_data_store_lock_status
 api_206a.c, [482](#)
 api_206a.h, [489](#)
 cryptoauthlib.sha206_api, [344](#)
sha206a_get_dk_update_count
 api_206a.c, [483](#)
 api_206a.h, [489](#)
 cryptoauthlib.sha206_api, [345](#)
sha206a_get_dk_useflag_count
 api_206a.c, [483](#)
 api_206a.h, [489](#)
 cryptoauthlib.sha206_api, [345](#)
sha206a_get_pk_useflag_count
 api_206a.c, [483](#)
 api_206a.h, [490](#)
 cryptoauthlib.sha206_api, [345](#)
sha206a_read_data_store
 api_206a.c, [484](#)
 api_206a.h, [490](#)
 cryptoauthlib.sha206_api, [345](#)
sha206a_verify_device_consumption
 api_206a.c, [484](#)
 api_206a.h, [491](#)
 cryptoauthlib.sha206_api, [346](#)
sha206a_write_data_store
 api_206a.c, [485](#)
 api_206a.h, [491](#)
 cryptoauthlib.sha206_api, [346](#)
sha2_routines.c, [655](#)
sha2_routines.h, [655](#)
SHA_MODE_TARGET_MSGDIGBUF
 cryptoauthlib.h, [658](#)
SHA_MODE_TARGET_OUT_ONLY
 cryptoauthlib.h, [658](#)
SHA_MODE_TARGET_TEMPKEY
 cryptoauthlib.h, [658](#)
sign
 cryptoauthlib.atjwt.HwEcAlgorithm, [454](#)
 cryptoauthlib.atjwt.HwHmacAlgorithm, [454](#)
slot_cnt
 pkcs11_lib_ctx_s, [464](#)

- slot_key
 - atca_check_mac_in_out, [372](#)
- slots
 - pkcs11_lib_ctx_s, [465](#)
- SNSRC_DEVICE_SN
 - Certificate manipulation methods (atcacert_), [190](#)
- SNSRC_DEVICE_SN_HASH
 - Certificate manipulation methods (atcacert_), [190](#)
- SNSRC_DEVICE_SN_HASH_POS
 - Certificate manipulation methods (atcacert_), [190](#)
- SNSRC_DEVICE_SN_HASH_RAW
 - Certificate manipulation methods (atcacert_), [190](#)
- SNSRC_PUB_KEY_HASH
 - Certificate manipulation methods (atcacert_), [190](#)
- SNSRC_PUB_KEY_HASH_POS
 - Certificate manipulation methods (atcacert_), [190](#)
- SNSRC_PUB_KEY_HASH_RAW
 - Certificate manipulation methods (atcacert_), [190](#)
- SNSRC_SIGNER_ID
 - Certificate manipulation methods (atcacert_), [190](#)
- SNSRC_STORED
 - Certificate manipulation methods (atcacert_), [190](#)
- SNSRC_STORED_DYNAMIC
 - Certificate manipulation methods (atcacert_), [190](#)
- Software crypto methods (atcac_), [214](#)
- STDCERT_NUM_ELEMENTS
 - Certificate manipulation methods (atcacert_), [191](#)
- swi_uart_deinit
 - Hardware abstraction layer (hal_), [250](#)
- swi_uart_discover_buses
 - Hardware abstraction layer (hal_), [251](#)
- swi_uart_init
 - Hardware abstraction layer (hal_), [251](#)
- swi_uart_mode
 - Hardware abstraction layer (hal_), [252](#)
- swi_uart_receive_byte
 - Hardware abstraction layer (hal_), [252](#)
- swi_uart_samd21_asf.c, [699](#)
- swi_uart_samd21_asf.h, [700](#)
- swi_uart_send_byte
 - Hardware abstraction layer (hal_), [253](#)
- swi_uart_setbaud
 - Hardware abstraction layer (hal_), [253](#)
- swi_uart_start.c, [701](#)
- swi_uart_start.h, [702](#)
- symmetric_authenticate
 - symmetric_authentication.c, [492](#)
 - symmetric_authentication.h, [493](#)
- symmetric_authentication.c, [492](#)
 - symmetric_authenticate, [492](#)
- symmetric_authentication.h, [493](#)
 - symmetric_authenticate, [493](#)
- target_key
 - atca_check_mac_in_out, [372](#)
- test_device, [352](#)
 - ATECC508A_DEVICE_CONFIG, [352](#)
 - ATECC508A_DEVICE_CONFIG_VECTOR, [352](#)
 - ATECC608_DEVICE_CONFIG, [353](#)
 - ATECC608_DEVICE_CONFIG_VECTOR, [353](#)
 - ATSHA204A_DEVICE_CONFIG, [353](#)
 - ATSHA204A_DEVICE_CONFIG_VECTOR, [354](#)
- test_iface, [354](#)
- TF_BIN2HEX_LC
 - Certificate manipulation methods (atcacert_), [191](#)
- TF_BIN2HEX_SPACE_LC
 - Certificate manipulation methods (atcacert_), [191](#)
- TF_BIN2HEX_SPACE_UC
 - Certificate manipulation methods (atcacert_), [191](#)
- TF_BIN2HEX_UC
 - Certificate manipulation methods (atcacert_), [191](#)
- TF_HEX2BIN_LC
 - Certificate manipulation methods (atcacert_), [191](#)
- TF_HEX2BIN_SPACE_LC
 - Certificate manipulation methods (atcacert_), [191](#)
- TF_HEX2BIN_SPACE_UC
 - Certificate manipulation methods (atcacert_), [191](#)
- TF_HEX2BIN_UC
 - Certificate manipulation methods (atcacert_), [191](#)
- TF_NONE
 - Certificate manipulation methods (atcacert_), [191](#)
- TF_REVERSE
 - Certificate manipulation methods (atcacert_), [191](#)
- tflxtls_cert_def_4_device.c, [502](#)
- tflxtls_cert_def_4_device.h, [503](#)
- TNG API (tng_), [81](#)
 - tng_atcacert_device_public_key, [83](#)
 - tng_atcacert_max_device_cert_size, [83](#)
 - tng_atcacert_max_signer_cert_size, [83](#)
 - tng_atcacert_read_device_cert, [84](#)
 - tng_atcacert_read_signer_cert, [84](#)
 - tng_atcacert_root_cert, [85](#)
 - tng_atcacert_root_cert_size, [85](#)
 - tng_atcacert_root_public_key, [85](#)
 - tng_atcacert_signer_public_key, [86](#)
 - tng_get_device_cert_def, [86](#)
 - tng_get_device_cert_def_ext, [86](#)
 - tng_get_device_pubkey, [87](#)
 - tng_map_get_device_cert_def, [87](#)
- tng_atca.c, [503](#)
- tng_atca.h, [504](#)
- tng_atcacert_client.c, [504](#)
- tng_atcacert_client.h, [505](#)
- tng_atcacert_device_public_key
 - cryptoauthlib.tng, [348](#)
 - TNG API (tng_), [83](#)
- tng_atcacert_max_device_cert_size
 - cryptoauthlib.tng, [348](#)
 - TNG API (tng_), [83](#)
- tng_atcacert_max_signer_cert_size
 - cryptoauthlib.tng, [349](#)
 - TNG API (tng_), [83](#)
- tng_atcacert_read_device_cert
 - cryptoauthlib.tng, [349](#)
 - TNG API (tng_), [84](#)
- tng_atcacert_read_signer_cert
 - cryptoauthlib.tng, [349](#)

- TNG API (tng_), [84](#)
- tng_atcacert_root_cert
 - cryptoauthlib.tng, [350](#)
 - TNG API (tng_), [85](#)
- tng_atcacert_root_cert_size
 - cryptoauthlib.tng, [350](#)
 - TNG API (tng_), [85](#)
- tng_atcacert_root_public_key
 - cryptoauthlib.tng, [350](#)
 - TNG API (tng_), [85](#)
- tng_atcacert_signer_public_key
 - cryptoauthlib.tng, [351](#)
 - TNG API (tng_), [86](#)
- tng_get_device_cert_def
 - TNG API (tng_), [86](#)
- tng_get_device_cert_def_ext
 - TNG API (tng_), [86](#)
- tng_get_device_pubkey
 - cryptoauthlib.tng, [351](#)
 - TNG API (tng_), [87](#)
- tng_map_get_device_cert_def
 - TNG API (tng_), [87](#)
- tng_root_cert.c, [505](#)
- tng_root_cert.h, [506](#)
- tnglora_cert_def_1_signer.c, [506](#)
- tnglora_cert_def_1_signer.h, [507](#)
- tnglora_cert_def_2_device.c, [507](#)
- tnglora_cert_def_2_device.h, [507](#)
- tnglora_cert_def_4_device.c, [508](#)
- tnglora_cert_def_4_device.h, [508](#)
- tngtls_cert_def_1_signer.c, [509](#)
- tngtls_cert_def_1_signer.h, [509](#)
- tngtls_cert_def_2_device.c, [510](#)
- tngtls_cert_def_2_device.h, [510](#)
- tngtls_cert_def_3_device.c, [510](#)
- tngtls_cert_def_3_device.h, [511](#)
- trust_pkcs11_config.c, [498](#)
- UNUSED_VAR
 - atca_compiler.h, [533](#)
- verify
 - cryptoauthlib.atjwt.HwHmacAlgorithm, [455](#)
- wpc_apis.c, [511](#)
- wpc_apis.h, [512](#)
- wpccert_client.c, [513](#)
 - wpccert_read_cert, [513](#)
 - wpccert_read_mfg_cert, [513](#)
- wpccert_client.h, [514](#)
 - wpccert_read_cert, [514](#)
 - wpccert_read_mfg_cert, [515](#)
- wpccert_read_cert
 - wpccert_client.c, [513](#)
 - wpccert_client.h, [514](#)
- wpccert_read_mfg_cert
 - wpccert_client.c, [513](#)
 - wpccert_client.h, [515](#)