MICROSAR Crypto

Technical Reference

Version 1.3

Authors	Thorsten Albers
Status	Not Released

Document Information

History

Author	Date	Version	Remarks
Thorsten Albers	2010-10-29	1.0	Creation of document
Thorsten Albers	2011-08-25	1.1	General update
Thorsten Albers	2011-10-26	1.2	Integration description updated
Thorsten Albers	2013-07-24	1.3	Add new functions, general update

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_DET.pdf	2.2.1
[2]	AUTOSAR	AUTOSAR_SWS_DEM.pdf	2.2.0
[3]	AUTOSAR	AUTOSAR_BasicSoftwareModules.pdf	V1.0.0
[4]	NIST	ADVANCED ENCRYPTION STANDARD (AES) (http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf)	Nov. 2001
[5]	RFC	US Secure Hash Algorithm 1 (SHA1)	RFC 3174
[6]	RSA	A Method for Obtaining Digital Signatures and Public-Key Cryptosystems (http://people.csail.mit.edu/rivest/Rsapaper.pdf)	1977
[7]	cryptovision	Cryptographic Library for Embedded Systems (cv_act_libES.pdf)	v1.2.1 Vector 1.14 (2011-06-10)

Scope of the Document

This technical reference describes the general use of the Crypto basis software. Please refer to your Release Notes to get a detailed description of the platform (host, compiler) your Vector Ethernet Bundle has been configured for.

Caution
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector’s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

Illustrations

Tables

Component History

The component history gives an overview over the important milestones that are supported in the different versions of the component.

Component Version	New Features
1.0	created
1.2.x	replace cryptovision lib with new version (additional crypto functions), add ECC domain parameters
1.4.x	Add generation and validation API for ECDSA signatures (curve ANSIp256r1)

Table 1-1 Component history

1 Introduction

This document describes the functionality, API and configuration of the MICROSAR BSW-internal module CRYPTO. This is

Supported AUTOSAR Release:	none	
Supported Configuration Variants:	none	
Vendor ID:	CRYPTO_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	CRYPTO_MODULE_ID	255 decimal (according to ref. [3], complex device driver)
Module Instance:	CRYPTO_INSTANCE_ID	104 decimal (default value, chosen by Vector)

The Crypto module is only a service module for stack internal usage (e.g. by TLS). It offers encryption and hash functionality. Most crypto and hash functions in this module are implemented by cryptovision, therefore this module is always delivered as a pre-compiled library to ensure protection of cryptovision IP.

1.1 Architecture Overview

The following figure shows where the CRYPTO is located in the MICROSAR architecture.

Figure 1-1 Dependencies to Crypto

No external application may access the services of this module directly. This module is only used stack-internally.

The Crypto module consists of two parts – a crypto library implemented by cryptovision (third party), and some extensions implemented by Vector. Crypto APIs of the cryptovision library are called directly from the stack modules, their names are esl_<function-name>. There are no wrappers for these functions to give them the modules prefix ‘Crypto_’, the same is valid for types used by these functions.

2 Functional Description

2.1 Features

The following feature lists describes the main functionality of this service module.

The "supported" and "not supported" features are presented in the following two tables. For further information of not supported features also see chapter 6.

The following features are supported:

Supported Feature
AES encryption and decryption
SHA-1 hash algorithm
SHA-256 hash algorithm
MD5 hash algorithm
SHA-1 HMAC
SHA-256 HMAC
MD5 HMAC
RSA signature generation and validation
RSA encryption and decryption
ECDSA signature handling (for one single elliptic curve)
RC2 decryption
3DES decryption

Table 2-1 Supported features

The following features are not supported:

Not Supported Feature

Table 2-2 Not supported features

2.2 Initialization

The Crypto module has to be initialized by calling the Crypto_InitMemory service and the Crypto_Init service.

An external random number generator (used by Crypto) has to be initialized **before** calling Crypto_Init.

2.3 States

The Crypto module is operational after initialization.

2.4 Main Functions

The Crypto module does not have a main function.

2.5 Error Handling

2.5.1 Development Error Reporting

Development errors are reported to the DET using the service `Det_ReportError()` as specified in [1], if development error reporting is enabled (i.e. pre-compile parameter `CRYPTO_DEV_ERROR_DETECT==STD_ON`).

The reported CRYPTO_MODULE_ID is 255. The CRYPTO_INSTANCE_ID is 104.

The reported service IDs identify the services which are described in section 4.4. The following table presents the service IDs and the related services:

Service ID	Service
--	Crypto_InitMemory
0x01	CRYPTO_API_ID_INIT
0x02	CRYPTO_API_ID_GET_VERSION_INFO

Table 2-3 Service IDs

The errors reported to DET are described in the following table:

Error Code		Description
0x01	CRYPTO_E_NOT_INITIALIZED	A CRYPTO service was called without initializing the module first by calling Crypto_Init
0x02	CRYPTO_E_INV_POINTER	An CRYPTO service was called with a zero pointer as parameter
0x03	CRYPTO_E_INV_PARAM	An CRYPTO service was called with an invalid parameter

Table 2-4 Errors reported to DET

2.5.2 Production Code Error Reporting

Production code related errors are reported to the DEM using the service `Dem_ReportErrorStatus()` as specified in [2], if production error reporting is enabled (i.e. pre-compile parameter `CRYPTO_PROD_ERROR_DETECT==STD_ON`).

The errors reported to DEM are described in the following table:

Error Code	Description
none	-

Table 2-5 Errors reported to DEM

3 Integration

This chapter gives necessary information for the integration of the MICROSAR CRYPTO into an application environment of an ECU.

3.1 Scope of Delivery

The delivery of the CRYPTO contains the files which are described in the chapters 3.1.1 and 3.1.2:

3.1.1 Static Files

File Name	Source Code Delivery	Object Code Delivery	Description
Crypto.c	■		Implementation
Crypto.h	■	■	API declaration
Crypto_Cfg.h	■		parameter configuration declaration
Crypto_Priv.h	■		Component local macro and variable declaration
Crypto_Types.h	■	■	Types header for API of Crypto
ESLib.h	■	■	cryptovision header
ESLib_ERC.h	■	■	cryptovision header
ESLib_platform_t.h	■	■	cryptovision header
ESLib_RNG.h	■	■	cryptovision header
ESLib_t.h	■	■	cryptovision header
ESLib_V1.1_compatibility.h	■	■	cryptovision header
ESLib.lib		■	cryptovision library (release or debug version)

Table 3-1 Static files

3.1.2 Dynamic Files

The dynamic files are generated by the configuration tool [config tool].

File Name	Description
none	-

Table 3-2 Generated files

3.2 Include Structure

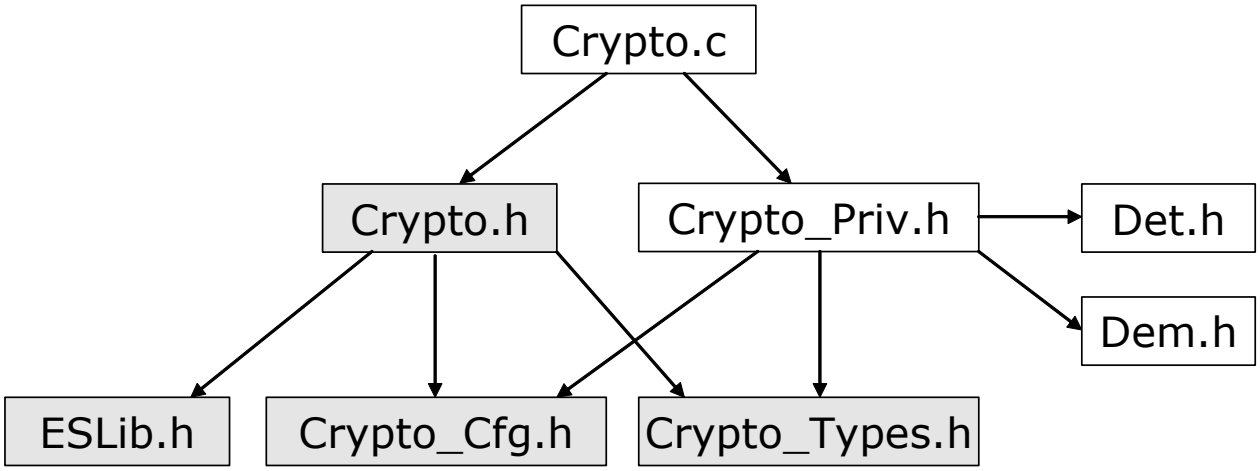

Figure 3-1 Include structure

Figure 3-1 shows the include structure of the module Crypto. Only ‘Crypto.h’ needs to be included by modules that want to use cryptographic functionality from the Crypto module, the ESLib is encapsulated.

3.3 Dependencies to other modules

Crypto calls the following **external** function **Appl_Crypto_GetRandArray()** to initialize its own internal random generator. This function fills a given array with random data. The function prototype is defined as external inside the Crypto.c file, so there is no need for a corresponding header include.

void Appl_Crypto_GetRandArray(uint8* TgtDataPtr, uint16 TgtLen);

Take care that this external random generator is initialized before it is used by Crypto.

4 API Description

4.1 Interfaces Overview

The CRYPTO provides the following services:

- Crypto_InitMemory()
- Crypto_Init()
- Crypto_CheckInit()
- Crypto_GetVersionInfo()
-
- Crypto_HmacMd5Init()
- Crypto_HmacMd5Encode()
- Crypto_HmacMd5End()
- Crypto_HmacSha256Init()
- Crypto_HmacSha256Encode()
- Crypto_HmacSha256End()
-
- Crypto_initVerifyRSAMD5_V15 ()
- Crypto_updateVerifyRSAMD5_V15 ()
- Crypto_finalizeVerifyRSAMD5_V15 ()
- Crypto_SignRSACRTgen_V15 ()
- Crypto_GenerateEcdsaSignature ()
- Crypto_ValidateEcdsaSignature ()
-
- Crypto_StirRNG ()

Table 4-1 CRYPTO API

4.2 Type Definitions

The types defined by the CRYPTO are described in this chapter.

Type Name	C-Type	Description	Value Range
Crypto_StateType	uint8	Data type for internal module state	CRYPTO_STATE_UNINIT, CRYPTO_STATE_INIT

Table 4-2 Type definitions

4.3 Structures

Crypto_HmacMd5StoreType

Struct Element Name	C-Type	Description	Value Range
Store	eslt_WorkSpaceMD5	MD5 workspace (from cryptovision lib)	-
KeyDataLocIpad	uint8[]	inner padding	-
KeyDataLocOpad	uint8[]	outer padding	-

Table 4-3 Crypto_HmacMd5StoreType

Crypto_HmacSha256StoreType

Struct Element Name	C-Type	Description	Value Range
Store	eslt_WorkSpaceSHA256	MD5 workspace (from cryptovision lib)	-
KeyDataLocIpad	uint8[]	inner padding	-
KeyDataLocOpad	uint8[]	outer padding	-

4.4 Services provided by CRYPTO

4.4.1 Crypto_InitMemory

Prototype	
`void` **`Crypto_InitMemory`** `(void)`	
Parameter	
void	
Return Code	
void	void
Functional Description	
Init internal module state variables.	
Particularities and Limitations	
Has to be called before any other calls to the module. Not reentrant	
Pre-Conditions	
none	
Call Context	

system startup / task level

Table 4-4 Crypto_InitMemory

4.4.2 Crypto_Init

Prototype	
`void Crypto_Init (void)`	
Parameter	
void	
Return Code	
void	void
Functional Description	
Initialization of the Crypto component.	
Particularities and Limitations	
This function has to be called after Crypto_InitMemory() and before any other function of this module. Not reentrant	
Pre-Conditions	
none	
Call Context	
system startup / task level	

Table 4-5 Crypto_Init

4.4.3 Crypto_CheckInit

Prototype	
`boolean Crypto_CheckInit (void)`	
Parameter	
void	
Return Code	
boolean	> TRUE module has been initialized > FALSE module has not been initialized
Functional Description	
Check if Crypto_Init was called. This is needed because most crypto functions are implemented by a third party and are called directly, so no DET-checks can be added here.	
Particularities and Limitations	
Has to be called before usage of the module	

Pre-Conditions
none
Call Context
initialization

Table 4-6 Crypto_CheckInit

4.4.4 Crypto_GetVersionInfo

Prototype	
`void` **`Crypto_GetVersionInfo`** `(Std_VersionInfoType *VersionInfoPtr)`	
Parameter	
VersionInfoPtr	pointer for version information
Return Code	
void	void
Functional Description	
Get Crypto software version. Read the module id, the vendor id and the software implementation version.	
Particularities and Limitations	
none	
Pre-Conditions	
none	
Call Context	
task level	

Table 4-7 Crypto_GetVersionInfo

4.4.5 Crypto_HmacMd5Init

Prototype	
`void` **`Crypto_HmacMd5Init`** `(Crypto_HmacMd5StoreType *HmacStorePtr, const uint8 *KeyDataPtr, uint32 KeyLenByte)`	
Parameter	
HmacStorePtr	pointer for the temporary data
KeyDataPtr	pointer to the key data
KeyLenByte	key data length in bytes
Return Code	
void	void

Functional Description
Initialize hmac MD5 calculation.
Particularities and Limitations
none
Pre-Conditions
none
Call Context
task level

Table 4-8 Crypto_HmacMd5Init

4.4.6 Crypto_HmacMd5Encode

Prototype	
`void` **`Crypto_HmacMd5Encode`** `(Crypto_HmacMd5StoreType *HmacStorePtr, const uint8 *SrcDataPtr, uint32 SrcLenByte)`	
Parameter	
HmacStorePtr	pointer for the temporary data
SrcDataPtr	pointer to the raw data
SrcLenByte	raw data length in bytes
Return Code	
void	void
Functional Description	
Proceed the hmac MD5 calculation.	
Particularities and Limitations	
none	
Pre-Conditions	
none	
Call Context	
task level	

Table 4-9 Crypto_HmacMd5Encode

4.4.7 Crypto_HmacMd5End

Prototype
`void` **`Crypto_HmacMd5End`** `(Crypto_HmacMd5StoreType *HmacStorePtr, uint8 *TgtDataPtr)`

Parameter	
HmacStorePtr	pointer for the temporary data
TgtDataPtr	pointer for the decoded data
Return Code	
void	void
Functional Description	
Finalize the hmac MD5 calculation.	
Particularities and Limitations	
none	
Pre-Conditions	
none	
Call Context	
task level	

Table 4-10 Crypto_HmacMd5End

4.4.8 Crypto_HmacSha256Init

Prototype
`void Crypto_HmacSha256Init (Crypto_HmacSha256StoreType *HmacStorePtr, const uint8 *KeyDataPtr, uint32 KeyLenByte)`
Parameter
HmacStorePtr — pointer for the temporary data
KeyDataPtr — pointer to the key data
KeyLenByte — key data length in bytes
Return Code
void — void
Functional Description
Initialize hmac SHA256 calculation.
Particularities and Limitations
none
Pre-Conditions
none
Call Context
task level

Table 4-11 Crypto_HmacSha256Init

4.4.9 Crypto_HmacSha256Encode

Prototype	
`void` **`Crypto_HmacSha256Encode`** `(Crypto_HmacSha256StoreType *HmacStorePtr, const uint8 *SrcDataPtr, uint32 SrcLenByte)`	
Parameter	
HmacStorePtr	pointer for the temporary data
SrcDataPtr	pointer to the raw data
SrcLenByte	raw data length in bytes
Return Code	
void	void
Functional Description	
Proceed the hmac SHA256 calculation.	
Particularities and Limitations	
none	
Pre-Conditions	
none	
Call Context	
task level	

Table 4-12 Crypto_HmacSha256Encode

4.4.10 Crypto_HmacSha256End

Prototype	
`void` **`Crypto_HmacSha256End`** `(Crypto_HmacSha256StoreType *HmacStorePtr, uint8 *TgtDataPtr)`	
Parameter	
HmacStorePtr	pointer for the temporary data
TgtDataPtr	pointer for the decoded data
Return Code	
void	void
Functional Description	
Finalize the hmac SHA256 calculation.	
Particularities and Limitations	
none	
Pre-Conditions	
none	
Call Context	

task level

Table 4-13 Crypto_HmacSha256End

4.4.11 Crypto_initVerifyRSAMD5_V15

Prototype	
`eslt_ErrorCode` **`Crypto_initVerifyRSAMD5_V15`** `(eslt_WorkSpaceRSAMD5ver *workSpace, eslt_Length keyPairModuleSize, const eslt_Byte *keyPairModule, eslt_Length publicKeyExponentSize, const eslt_Byte *publicKeyExponent)`	
Parameter	
workSpace	pointer to signature workspace
keyPairModuleSize	size (byte) of RSA module
keyPairModule	pointer to RSA module
publicKeyExponentSize	size (byte) of RSA public exponent
publicKeyExponent	pointer to RSA public exponent
Return Code	
eslt_ErrorCode	> ESL_ERC_NO_ERROR initialization was successful > ESL_ERC_WS_TOO_SMALL given workspace is too small > others other init errors
Functional Description	
Initialize verification of RSA with MD5 digital signature Functionality analog to esl_initVerifyRSASHA1_V15(() of cryptovision.	
Particularities and Limitations	
workspace has to be initialized before	
Pre-Conditions	
none	
Call Context	
task level	

Table 4-14 Crypto_initVerifyRSAMD5_V15

4.4.12 Crypto_updateVerifyRSAMD5_V15

Prototype	
`eslt_ErrorCode` **`Crypto_updateVerifyRSAMD5_V15`** `(eslt_WorkSpaceRSAMD5ver *workSpace, eslt_Length inputSize, const eslt_Byte *input)`	
Parameter	
workSpace	pointer to signature workspace

inputSize	size (byte) of input data
input	pointer to input data
Return Code	
eslt_ErrorCode	> ESL_ERC_NO_ERROR update was successful > others update was not successful
Functional Description	
Update verification of RSA with MD5 digital signature Functionality analog to esl_updateVerifyRSASHA1_V15(() of cryptovision.	
Particularities and Limitations	
Crypto_initVerifyRSAMD5_V15() has to be executed before	
Pre-Conditions	
none	
Call Context	
task level	

Table 4-15 Crypto_updateVerifyRSAMD5_V15

4.4.13 Crypto_finalizeVerifyRSAMD5_V15

Prototype	
`eslt_ErrorCode` **`Crypto_finalizeVerifyRSAMD5_V15`** `(eslt_WorkSpaceRSAMD5ver *workSpace, eslt_Length signatureSize, const eslt_Byte *signature)`	
Parameter	
workSpace	pointer to signature workspace
signatureSize	max signature size
signature	pointer to where the signature shall be stored
Return Code	
eslt_ErrorCode	> ESL_ERC_NO_ERROR finalization was successful > others finalization was not successful
Functional Description	
Finalize verification of RSA with MD5 digital signature Functionality analog to esl_finalizeVerifyRSASHA1_V15(() of cryptovision.	
Particularities and Limitations	
Crypto_initSignRSAMD5_V15() has to be executed before	
Pre-Conditions	
none	
Call Context	
task level	

Table 4-16 Crypto_finalizeVerifyRSAMD5_V15

4.4.14 Crypto_SignRSACRTgen_V15

Prototype
`eslt_ErrorCode` **`Crypto_SignRSACRTgen_V15 (eslt_WorkSpaceRSACRTsig *workSpace, uint16 keyPairPrimePSize, CONSTuint8 *keyPairPrimeP, uint16 keyPairPrimeQSize, CONSTuint8 *keyPairPrimeQ, uint16 privateKeyExponentDPSize, CONSTuint8 *privateKeyExponentDP, uint16 privateKeyExponentDQSize, CONSTuint8 *`**`*privateKeyExponentDQ, uint16 privateKeyInverseQISize, CONSTuint8 *privateKeyInverseQI, uint16 hashSize, CONSTuint8 *hashPtr, uint16 *signatureSize, CONSTuint8 *signature)`

Parameter	
workSpace	pointer to signature workspace
keyPairPrimePSize	size (byte) of RSA key parameter P
keyPairPrimeP	pointer to RSA key parameter P
keyPairPrimeQSize	size (byte) of RSA key parameter Q
keyPairPrimeQ	pointer to RSA key parameter Q
privateKeyExponentDPSize	size (byte) of RSA key parameter DP
privateKeyExponentDP	pointer to RSA key parameter DP
privateKeyExponentDQSize	size (byte) of RSA key parameter DQ
privateKeyExponentDQ	pointer to RSA key parameter DQ
privateKeyInverseQISize	size (byte) of RSA key parameter QI
privateKeyInverseQI	pointer to RSA key parameter QI
hashSize	size (byte) of input / hash
hashPtr	pointer to input / hash
signatureSize	max signature size
signature	pointer to where the signature shall be stored

Return Code	
eslt_ErrorCode	> ESL_ERC_NO_ERROR initialization was successful > ESL_ERC_WS_TOO_SMALL given workspace is too small > others other init errors

Functional Description
Calculate RSA digital signature using a pre-calculated hash value Functionality analog to esl_initSignRSACRTSHA1_V15(), esl_updateSignRSACRTSHA1_V15() and esl_finalizeSignRSACRTSHA1_V15() of cryptovision.

Particularities and Limitations
Workspace has to be initialized before. Derived from function using RSA and MD5.

Pre-Conditions
none

Call Context
task level

Table 4-17 Crypto_SignRSACRTgen_V15

4.4.15 Crypto_GenerateEcdsaSignature

Prototype	
`Std_ReturnType` **`Crypto_GenerateEcdsaSignature`** `(eslt_WorkSpaceEcP *EcWorkSpPtr, uint8 *SignatureValuePtr, const uint8 *PrivKeyPtr, const uint8 *DigestValuePtr, uint16 *SignatureValueLen, uint16 PrivKeyLen, uint8 DigestValueLen, boolean BerEncoded)`	
Parameter	
EcWorkSpPtr	Pointer to the Crypto workspace that is used for signature generation
SignatureValuePtr	Pointer to the Signature value buffer
PrivKeyPtr	Pointer to the private key information that shall be used for this signature
DigestValuePtr	Pointer to the digest value that is used signed
SignatureValueLen	Pointer to the length of the signature, shall be maximum buffer size when calling this function and will hold the generated signature value length when returning
PrivKeyLen	Length of the private key
DigestValueLen	Length of the digest value
BerEncoded	Signature Value is BER encoded
Return Code	
Std_ReturnType	> E_OK Signature generated successfully and stored at SignatureValuePtr with SignatureValueLen > E_NOT_OK Error during signature generation. No signature value is available.
Functional Description	
This function is used to generate a ECDSA signature using ANSIp256r1 curve.	
Particularities and Limitations	
none	
Pre-Conditions	
none	
Call Context	
task level	

Table 4-18 Crypto_GenerateEcdsaSignature

4.4.16 Crypto_ValidateEcdsaSignature

Prototype	
`Std_ReturnType` **`Crypto_ValidateEcdsaSignature`** `(eslt_WorkSpaceEcP *EcWorkSpPtr, const uint8 *SignatureValuePtr, const uint8 *PubKeyPtr, const uint8 *DigestValuePtr, uint16 SignatureValueLen, uint16 PubKeyLen, uint8 DigestValueLen, boolean BerEncoded)`	
Parameter	
EcWorkSpPtr	Pointer to the Crypto workspace that is used for signature validation
SignatureValuePtr	Pointer to the Signature value buffer
PubKeyPtr	Pointer to the public key information that shall be used to verify this signature
DigestValuePtr	Pointer to the digest value against which the signature shall be verified
SignatureValueLen	Length of the signature
PubKeyLen	Length of the public key
DigestValueLen	Length of the digest value
BerEncoded	Signature Value is BER encoded
Return Code	
Std_ReturnType	> E_OK Signature validation finished successfully. The signature is valid. > E_NOT_OK Error during signature validation. The signature is invalid.
Functional Description	
This function is used to validate a ECDSA signature using ANSIp256r1 curve.	
Particularities and Limitations	
none	
Pre-Conditions	
none	
Call Context	
task level	

Table 4-19 Crypto_ValidateEcdsaSignature

4.4.17 Crypto_StirRNG

Prototype	
`void` **`Crypto_StirRNG`** `(const uint16 InputLen, const uint8 *InputPtr)`	
Parameter	
InputLen	The length (byte) of the input data
InputPtr	Pointer to the input data for stirring
Return Code	
void	void

Functional Description
Extern function needed by the ESLib of CV to get a random number.
Particularities and Limitations
esl_initRNG() has to be executed before
Pre-Conditions
none
Call Context
task level

Table 4-20 Crypto_StirRNG

4.4.18 Other APIs (cryptovision)

All other APIs are described in [7]. Those functions are implemented in the cryptovision library, and they are called with their original names.

4.5 Services used by CRYPTO

In the following table services provided by other components, which are used by the CRYPTO are listed. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
DET (optional)	Det_ReportError
Application	Appl_Crypto_GetRandArray

Table 4-21 Services used by the CRYPTO

4.6 Call-back Functions

none

4.7 Call-out Functions

none

5 Configuration

The CRYPTO module can not be configured by the system integrator.

6 AUTOSAR Standard Compliance

6.1 Deviations

There is no AUTOSAR standard for the CRYPTO Module, so there can’t be any deviations.

6.2 Additions/ Extensions

not relevant

6.3 Limitations

none

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
-	-

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
TLS	Transport Layer Security

Table 7-2 Abbreviations

8 Contact

Visit our website for more information on

- News
- Products
- Demo software
- Support
- Training data
- Addresses

www.vector.com