EUCHNER
More than safety.

EKS on Siemens S7-300 – checking CRC

Contents

Components/modules used

EUCHNER

Description	Order no./item designation
EKS Profibus	084800 / EKS-A-IDX-G01-ST09/03
EKS Electronic-Key	077859 / EKS-A-K1RDWT32-EU 084735 / EKS-A-K1BKWT32-EU 091045 / EKS-A-K1BLWT32-EU 094839 / EKS-A-K1GNWT32-EU 094840 / EKS-A-K1YEWT32-EU

Tip: More information and downloads about the aforementioned EUCHNER products can be found at www.EUCHNER.de. Simply enter the order number in the search box.

Others

Description	Item
S7-300, CPU 315F-2 PN/DP	6ES7315-2FJ14-0AB0

Functional description

General

The EKS is connected to a Siemens S7-300 PLC via the Profibus. The CRC is to be calculated over the Electronic-Key content. This program is based on the data from the Electronic-Key already having been read into a data block. An example for reading the data can be found in AP000169-3-... for a Profibus EKS and in AP000169-4... for a Profinet EKS. Note that data block DB1 has a different structure for the two versions.

Example of an Electronic-Key structure

The data on the Electronic-Key are structured as follows:

Byte no.	Description	Type	Length	Explanation
103 – 104	KEYCRC	CRC	2 bytes	Checksum over a certain part of the Electronic-Key as copy protection. Refer to the EKM manual for details about the CRC.
105 – 112	Expiry date	Date	8 bytes	Electronic-Key expiry date.
113 – 114	Authorization level	Word	2 bytes	Authorization level for access to the machine.
115	Department	Byte	1 byte	Number describing a limited quantity of machines or installations.
116 – 123	KeyID	KeyID	8 bytes	The KeyID is a number that is permanently pre-programmed on the Electronic-Key by EUCHNER. This number is different for each Electronic-Key. This number can be used to identify workers.

CRC definition

The KeyCRC is configured as follows in the EKM. Calculation begins with byte 105 and extends over 11 bytes. The CRC is on the Electronic-Key from byte 103.

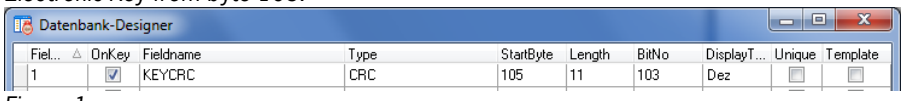

Figure 1

Programming in the control system

Global data block

A global data block is created, which must already contain the Electronic-Key content when the check of the CRC is called.
The data are created in a structured manner in the data block for reading, with all data items longer than one byte being created as individual bytes to circumvent the even-numbered alignment in the control system.

DB1, ReadBufferEKS

The data block shown in Figure 2 is suitable for example AP000169-3…, in which the EKS is used with Profibus.
Data DB1 has a somewhat different structure with a Profinet EKS. Bytes 1 to 3 are not used for Profinet (ReadKeyCount, ReadStartAddress, ReadNumberBytes). The corresponding lines are omitted in DB1 for the EKS Profinet.

Address	Name	Type	Initial val	Comment
0.0		STRUCT		
+0.0	ReadEKSStatus	BYTE	B#16#0	Statusbyte from EKS
+1.0	ReadKeyCount	BYTE	B#16#0	Counter for keys
+2.0	ReadStartAddress	BYTE	B#16#0	First byte
+3.0	ReadNumberBytes	BYTE	B#16#0	Number of bytes read
+4.0	ReadCRC_00	BYTE	B#16#0	CRC Byte 0
+5.0	ReadCRC_01	BYTE	B#16#0	CRC Byte 1
+6.0	ReadDate_00	BYTE	B#16#0	Date Byte 0
+7.0	ReadDate_01	BYTE	B#16#0	Date Byte 1
+8.0	ReadDate_02	BYTE	B#16#0	Date Byte 2
+9.0	ReadDate_03	BYTE	B#16#0	Date Byte 3
+10.0	ReadDate_04	BYTE	B#16#0	Date Byte 4
+11.0	ReadDate_05	BYTE	B#16#0	Date Byte 5
+12.0	ReadDate_06	BYTE	B#16#0	Date Byte 6
+13.0	ReadDate_07	BYTE	B#16#0	Date Byte 7
+14.0	ReadAuthorization_00	BYTE	B#16#0	Access Level Byte 0
+15.0	ReadAuthorization_01	BYTE	B#16#0	Access Level Byte 1
+16.0	ReadDepartment	BYTE	B#16#0	Department
+17.0	ReadKeyID_00	BYTE	B#16#0	KeyID Byte 0
+18.0	ReadKeyID_01	BYTE	B#16#0	KeyID Byte 1
+19.0	ReadKeyID_02	BYTE	B#16#0	KeyID Byte 2
+20.0	ReadKeyID_03	BYTE	B#16#0	KeyID Byte 3
+21.0	ReadKeyID_04	BYTE	B#16#0	KeyID Byte 4
+22.0	ReadKeyID_05	BYTE	B#16#0	KeyID Byte 5
+23.0	ReadKeyID_06	BYTE	B#16#0	KeyID Byte 6
+24.0	ReadKeyID_07	BYTE	B#16#0	KeyID Byte 7
+26.0	Buffer	ARRAY[0..5]		NC for filling up to 32 bytes
*1.0		BYTE		
=32.0		END_STRUCT		

Figure 2

DB11, instance module for FB2

A DB is used as an instance module so that function module FB2 can be supplemented with static variables. In the example, DB11 is created for this purpose. For example, further evaluation of the Electronic-Key can be programmed directly after a positive CRC check.

STL program for calculating the CRC

The calculation program is programmed in FB2 in this example. The CRC is to be calculated over an odd number of data items, so the checksum must be calculated in three steps.
All words that are completely within the user data are read and calculated in the first step. In the second step, the last byte of the user data is read and is supplemented with a 0. The KeyID is then added to the CRC in the last step.
The data are read byte-by-byte based on the arrangement in the little-endian format. The first byte is shifted 8 bits to the left in each case, and the second byte is then read.

Tip: If the CRC to be calculated is calculated over an even-numbered user-data range, the entire calculation can be performed in a single step if the data are sequential.

Description of the interface

Input data
None.

Output data
Message as to whether the CRC is correct.

Input/output data
None.

Static data
None.

Temporary data
Counter for the loops in step 1 and step 3 and calculated value of the CRC.

Name	Data type	Address	Start value	Comment
IN		0.0		
OUT		0.0		
CRC_ok	Bool	0.0	FALSE	CRC is identical
IN_OUT		0.0		
STAT		0.0		
TEMP		0.0		
Tcounter	Int	0.0		Counter
CRC	Word	2.0		Calculated value of CRC

Figure 3

Changed registers
A1, A2, SW, AR1, DBR1

Unchanged registers
AR2, DBR2

System functions used
None.

Global data
Data block DB1 must contain the data of the EKS Electronic-Key that has already been read.
The content of data block DB1 is not changed.

Symbol table

	Status	Symbol	Address	Data type	Comment
1		Calculate CRC	FB 2	FB 2	
2		COMPLETE RESTART	OB 100	OB 100	Complete Restart
3		Data FB1	DB 10	FB 1	
4		Data FB2	DB 11	FB 2	
5		DIS_AIRT	SFC 41	SFC 41	Delay the Higher Priority Interrupts and Asynchronous Errors
6		DPRD_DAT	SFC 14	SFC 14	Read Consistent Data of a Standard DP Slave
7		DPWR_DAT	SFC 15	SFC 15	Write Consistent Data to a Standard DP Slave
8		EKSIn	E 256.1	BOOL	Fla if key plugged
9		EKSInCount	EB 257	BYTE	Counter of EKS
10		EKSMemIn	EB 256	BYTE	First byte of input buffer EKS
11		EKSMemOut	AB 256	BYTE	First byte of output buffer EKS
12		EN_AIRT	SFC 42	SFC 42	Enable Higher Priority Interrupts and Asynchronous Errors
13		F_CTRL_1	FB 273	FB 273	
14		F_CTRL_2	FB 274	FB 274	F_: Test Block an Programm Run Control
15		F_DIAG_N	FB 275	FB 275	F_: Diagnosticbuffer Message with non CPU-Stop
16		F_GLOBDB	DB 545	DB 545	F_: F_Global_Data Block
17		F_IO_CGP	FB 272	FB 272	F_: Driver Block In-Output with Channel Granular Passivation
18		Globaler Speicher	DB 3	DB 3	
19		I/O_FLT1	OB 82	OB 82	I/O Point Fault 1
20		Main Program	OB 1	OB 1	
21		PROG_ERR	OB 121	OB 121	Programming Error
22		RDSYSST	SFC 51	SFC 51	Read a System Status List or Partial List
23		Read EKS	FB 1	FB 1	
24		ReadBufferEKS	DB 1	DB 1	
25		STP	SFC 46	SFC 46	Change the CPU to STOP
26		VAT_1	VAT 1		
27		VAT_2	VAT 2		
28		WriteBufferEKS	DB 2	DB 2	

Figure 4

STL program in FB2 – Calculate CRC

The program shown in Figure 5a is suitable for example AP000169-3..., in which the EKS is used with Profibus.
As data block DB1 has a slightly different structure with an EKS Profinet, byte 3 must be used as the start instead of byte 6. Three status bytes are missing in case of Profinet EKS, which the EKS Profibus sends, and the data range is therefore shifted forward by three bytes.
The corresponding program line is then:

LAR1 P#DBX3.0

```
// The CRC is calculated to suit the specified Electronic-Key structure
// Calculation is done in three steps, which are not always required depending on data configuration
// Calculation must take place in three steps only if an odd-numbered amount of user data is read with-
out KeyId
// All words, except for the last byte, are calculated in the first step
// The individual byte, extended by an inserted 0, is calculated in the second step
// The corresponding KeyID is calculated in the third step

// Calculation of the first part via the user data on the Electronic-Key
      L     5                           // Number of words in the first step
      T     #TCounter                   // Initiate counter for adding
      L     0                           // Initiate CRC calculation value
      T     #CRC
      AUF   "ReadBufferEKS"             // Read data from the DB1
      LAR1  P#DBX 3.0                   // Beginning from the word following the CRC
LOP1: L     B [AR1,P#1.0]               // Load next word (offset 1)
      SLW   8
      L     B [AR1,P#0.0]
      +I
      L     #CRC                        // Add to existing CRC
      +I
      T     #CRC                        // Newly calculated CRC
      L     P#2.0                       // Increase pointer by 2 bytes
      +AR1                              // Address register
      L     #TCounter                   // Decrease counter by 1
      DEC   1
      T     #TCounter
      L     0                           // End of loop when value reaches 0
      ==I
      SPBN  LOP1
```

Figure 5a

```
// Special calculation, because an odd-numbered amount of bytes are on the Electronic-Key in the CRC
      L     B [AR1,P#0.0]               // Load next word, because a 0 is inserted it is only the byte
      L     #CRC                        // Add to existing CRC
      +I
      T     #CRC                        // Newly calculated CRC
      L     P#1.0                       // Increase pointer by 1 byte
     +AR1
```

Figure 5b

```
// Calculation via the KeyID
      L     4                           // Number of words, only KeyID
      T     #TCounter                   // Initiate counter for adding

LOP2: L     B [AR1,P#1.0]               // Load next word
      SLW   8
      L     B [AR1,P#0.0]
      +I
      L     #CRC                        // Add to existing CRC
      +I
      T     #CRC                        // Newly calculated CRC
      L     P#2.0                       // Increase pointer by 2 bytes
      +AR1                              // Address register
      L     #TCounter                   // Decrease counter by 1
      DEC   1
      T     #TCounter
      L     0                           // End of loop when value reaches 0
      ==I
      SPBN  LOP2
```

Figure 5c

The program shown in Figure 5d is suitable for example AP000169-3... , in which the EKS is used with Profibus.
As data block DB1 has a slightly different structure with an EKS Profinet, byte 1 must be used as the start instead of byte 4. Three status bytes are missing in case of Profinet EKS, which the EKS Profibus sends, and the data range is therefore shifted forward by three bytes.
The corresponding program line is unchanged when the symbol table is built up.

```
// Compare with the CRC on the Electronic-Key
      L     "ReadBufferEKS".ReadCRC_01  // The CRC is located from byte 1
      SLW   8
      L     "ReadBufferEKS".ReadCRC_00  //
      +I
      L     #CRC                        // Compare calculated CRC with the CRC from the Electronic-Key
      ==I
      =     #CRC_ok                     // Set return value
      BE
```

Figure 5d

FB2 call

The calculation is called only if valid new Electronic-Key content was provided, for example from an EKS reading program as in application AP000169-3... This is identified by marker M0.1 being set.

```
// Calculate the Electronic-Key CRC
      U     M      0.1                  // Check whether Electronic-Key is new
      SPBN  M001
      CALL  "Calculate CRC" , "Data FB2"
      CRC_ok:=M0.0

M001: BE                                // End of EKS reading
```

Figure 6

Important note – please observe carefully!

This document is intended for a design engineer who possesses the requisite knowledge in safety engineering and knows the applicable standards, e.g. through training for qualification as a safety engineer. Only with the appropriate qualification is it possible to integrate the introduced example into a complete safety chain.

The example represents only part of a complete safety chain and does not fulfill any safety function on its own. In order to fulfill a safety function, the energy switch-off function for the hazard location and the software within the safety evaluation must also be considered, for example.

The introduced applications are only examples for solving certain safety tasks for protecting safety doors. The examples cannot be comprehensive due to the application-dependent and individual protection goals within a machine/installation.

If questions concerning this example remain open, please contact us directly.

In accordance with Machinery Directive 2006/42/EC, the design engineer of a machine or installation is obligated to perform a risk assessment and take measures to reduce the risk. When doing this, the engineer must comply with the applicable national and international standards. Standards generally represent the current state of the art. Therefore, the design engineer should continuously inform himself about changes in the standards and adapt his considerations to them. Relevant standards include EN ISO 13849 and EN 62061. This application must be regarded only as assistance for the considerations about safety measures.

The design engineer of a machine/installation is obligated to assess the safety technology itself. The examples must not be used for assessment, because only a small excerpt of a complete safety function was considered in terms of safety engineering here.

In order to be able to use the safety switch applications correctly on safety doors, it is indispensable to observe the standards EN ISO 13849-1, EN ISO 14119 and all relevant C-standards for the respective machine type. Under no circumstances does this document replace the engineer's own risk assessment, and it cannot serve as the basis for a fault assessment.

Particularly in case of fault exclusion, it must be noted that this can be performed only by the design engineer of a machine or installation and requires a reason. General fault exclusion is not possible. More information about fault exclusion can be found in EN ISO 13849-2.

Changes to products or within assemblies from third-party suppliers used in this example can lead to the function no longer being ensured or the safety assessment having to be adapted. In any event, the information in the operating instructions on the part of EUCHNER, as well as on the part of third-party suppliers, must be used as the basis before this application is integrated into an overall safety function. If contradictions should arise between the operating instructions and this document, please contact us directly.

Use of brand names and company names

All brand names and company names stated are the property of the related manufacturer. They are used only for the clear identification of compatible peripheral devices and operating environments in relation to our products.