Schrittmotoren, Steuerungen, Bewegungssysteme

SmartStep Protokoll FW 2.x

17.03.2026

1. Allgemeine Protokollbeschreibung

1.1 Protokollrahmen

Frame:

STX	Length	Dest. Addr.	TT/Src.Addr.	Payload	CRC 16Bit

- *STX*: Startzeichen = 0x02

- *Length*: Länge des Telegramms
  Anzahl Bytes ab Zieladresse inkl. CRC

- *Dest.Addr.*: Zieladresse der Schrittmotorkarte
  Adresse des Empfängers. Adresse 0 ist
  Broadcast. Gültige Kartenadressen: 1 - 31

- *TT*: Telegrammtyp (2 Bit :Bits 6 und 7) 0 = Request

  1 = Response
  2 = Spontan

- *Src.Addr.*: Quelladresse: 6 Bit
  Adresse des Senders
  Gültige Kartenadressen: 1 - 31

- *Payload*: Nutzdaten

- *CRC*: 2 Byte, MSB zuerst
  Generatorpolynom: x^16 + x^12 + x^5 +1 (CCITT)
  Startwert: 0
  Berechnung siehe Anhang A.1 und A.2

1.2 Payload

Die Nutzdaten entsprechen, bis auf Channel = 8 (Spontanmeldung einer Karte), dem Protokoll der Firmwareversion 1.x. Diesem Protokoll wurde der Rahmen aus 1. hinzugefügt, um ein Bussystem zu ermöglichen. Aus diesem Grund ist das Byte Length hier redundant.

Request

Channel	Length	Command	Data[0 .. n]

- *Channel*: Kanal auf der Schrittmotorkarte
  1 Byte

  1 - 5: Antriebskanäle
  6: Kanal für weitere Hardware auf der Karte (z.B. I/O)
  7: Kanal für Parametereinstellungen
  8: Spontanmeldung einer Karte
  32: Quittungs- (Acknowledge) oder Antwortkanal (Response)

- *Length*: Länge Data +Command, also n+1

- *Command*: Kommando des Telegramms

- *Data*: Parameter des Kommandos

Response

Channel = 32	Length	Response Type	Command	Data[0 .. n]

- *Channel*: Immer 32

- *Length*: Response Type + Command + Data, also n+1

- *Response Type*:
  ERROR_CODE = 0
  RESPONSE = 1
  SPONTANEOUS = 2
  DEBUG_MSG = 3

- *Command*: Ist immer das Kommando vom Request - Telegramm

- Data[0..n]: Response-Typ ERROR_CODE:
  In diesem Fall ist Data immer ein Byte, das den Error Code enthält.

  Error Codes:
  EC_OK = 0
  EC_UNKNOWN_COMMAND = 1
  EC_INVALID_PARAMETER = 2
  EC_INVALID_CHANNEL = 3
  EC_TIMEOUT = 4
  EC_WRITE_ERROR = 5
  EC_EXCEPTION = 6
  EC_INVALID_FILE_FORMAT = 7
  EC_CHECKSUM_ERROR = 8
  EC_HEXFILE_TOO_LONG = 9
  EC_FILE_NOT_FOUND = 10
  EC_NO_CONNECTION = 11
  EC_INVALID_BOARD_ADDRESS = 12

Response-Typ RESPONSE:
In diesem Fall enthält das Array Data[] die angefragten Daten.

1.3 Request / Response

Alle Kommandos, sei es zur Steuerung oder zur Abfrage eines Wertes, werden mit dem Telegrammtyp Request gesendet (vgl. 1.1 Protokollrahmen). Die Steuerung beantwortet die Anfrage mit einem Telegramm vom Typ Response. Response - Telegramme werden immer mit der Kanalnummer 32 gesendet. Sie können entweder angefragte Daten oder einen Fehlercode enthalten.

Beispiel: Setzen des Ausgangs 1 auf der Steuerung mit der Adresse 1. Das steuernde Gerät hat die Adresse 32

Request

STX	Length Frame	DA	TT/SA	CH:HW	Length Payload	Set Output	Output	Value	CRC MSB	CRC LSB
0x02	0x09	0x01	0x20	0x06	0x03	0x01	0x01	0x01	0x50	0x7e

Feld TT/SA:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TT	TT	SA	SA	SA	SA	SA	SA
0	0	0	0	0	0	1	0

Telegrammtyp = 1, Request

Response

STX	Length Frame	DA	TT/SA	CH: Ack.	Length Payload	Response Type	Command	Error-Code	CRC MSB	CRC LSB
0x02	0x09	0x20	0x41	0x20	0x03	0x00	0x01	0x00	0x7e	0x71

Feld TT/SA:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TT	TT	SA	SA	SA	SA	SA	SA
0	1	0	0	0	0	0	1

Telegrammtyp = 1, Response

1.4 Spontanmeldungen

Spontanmeldungen werden z.B. beim Beenden eines Fahrwegs (Ready-Meldung) oder im Falle eines Alarms gesendet. Eine Ready-Meldung wird an den Sender adressiert, der den Fahrweg initiiert hat. Alarmmeldungen werden als Broadcast (Adresse 0) gesendet. Auch Spontanmeldungen müssen quit-tiert werden, ansonsten werden diese Meldungen wiederholt.

Beispiel: Eine Steuerung mit der Adresse 1 meldet die Beendigung eines Fahrwegs. Ein Sender mit der Adresse 2 hat den Fahrweg ausgelöst.

Spontanmeldung

STX	Length Frame	DA	TT/SA	CH: Spontan	Length Payload	Type Ready	Von Antrieb1	CRC MSB	CRC LSB
0x02	0x08	0x02	0x81	0x08	0x02	0xfa	0x81	0x89	0xca

Feld TT/SA:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TT	TT	SA	SA	SA	SA	SA	SA
1	0	0	0	0	0	0	1

Telegrammtyp = 2, Spontan

Quittung

STX	Length Frame	DA	TT/SA	CH: Ack.	Length Payload	Response Type	Command	Error-Code	CRC MSB	CRC LSB
0x02	0x09	0x01	0x42	0x20	0x03	0x00	0xfa	0x00	0xa9	0x3d

Feld TT/SA:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TT	TT	SA	SA	SA	SA	SA	SA
0	1	0	0	0	0	0	1

Telegrammtyp = 1, Response

2. Kommandos für Antriebskanäle

Die Kommandos sind immer an eine Kanalgruppe gebunden, d.h., dass es bei den Kanälen für den Antrieb (1 bis 5) und dem Kanal für die allgemeine Hardware der Steuerung (z.B. I/O) den gleichen Kommando-Code mit einer anderen Bedeutung geben kann.

2.1 Schrittfrequenz

Schrittfrequenz für folgende Fahrbefehle setzen.

CH:	Length Payload	Command	Data[0]	Data[1]
Ch	0x03	0x14	n LSB	n MSB

```
Ch : 1..5, je nach Steuerung
n: Wertebereich 188..65535
```

Der 16 Bit-Wert aus Value High und Value Low bestimmt die Periodendauer des Schrittimpulses. Die Frequenz des Schrittimpulses wird wie folgt berechnet:

$f_{Schritt}$ = 7,5MHz / 16Bit Wert

2.2 Drehrichtung

Drehrichtung setzen.

CH:	Length Payload	Command	Data[0]
Ch	0x02	0x15	dir

```
ch : 1..5, je nach Steuerung
dir: 0 = links, 255 = rechts
```

2.3 Weg fahren relativ

Eine vorgegebene Anzahl von Schritten ausführen.

CH:	Length Payload	Command	Data[0]	Data[1]	Data[2]
Ch	0x04	0x19	n LSB	n	n MSB

```
ch : 1..5, je nach Steuerung
n  : Anzahl Schritte
n  : 0xffffff (2^24-1) löst eine Dauerfahrt aus.
n 0 : 0 = Fahrt stoppen
```

2.4 Weg fahren absolut

Eine vorgegebene Position anfahren. Es werden so viele Schritte in der erforderlichen Richtung ausgegeben, bis der interne Schrittzähler die angegebenen Wert erreicht hat.

CH:	Length Payload	Command	Data[0]	Data[1]	Data[2]	Data[3]
Ch	0x05	0x1a	p LSB	p	p	p MSB

```
ch : 1..5, je nach Steuerung
p : Zielposition
```

2.5 Betriebsart Step/Dir oder diskrete Ansteuerung der Wicklungen

In der Betriebsart "Step/Dir" sind die unter 2.1.1 bis 2.1.4 angegebenen Kommandos zur Ausgabe von Schrittimpulsen gültig. In der Betriebsart "Diskret" können die Ströme der Motorwicklungen individuell eingestellt werden (siehe auch 2.1.6 und 2.1.7).

CH:	Length Payload	Command	Data[0]
Ch	0x02	0x2b	mode

```
ch : 1..5, je nach Steuerung
mode: 0 = Step/Dir, 1 = Diskret
```

2.6 Diskrete Wicklungsströme ausgeben

Die Spulenströme werden direkt ausgegeben. Voraussetzung ist, dass die Steuerung in den Modus "Diskret" gesetzt wurde (siehe 2.1.5). Das MSB eines Spulenwertes gibt die Polarität des Stroms an.

CH:	Length Payload	Command	Data[0]	Data[1]	Data[2]	Data[3]
Ch	0x05	0x2c	A LSB	A MSB	BLSB	B MSB

```
ch : 1..5, je nach Steuerung

A: Strom für Wicklung A
       A MSB: 0..1 = Polarität des Stroms
       A LSB: Wertebereich: 0..245

B: Strom für Wicklung B
       B MSB: 0..1 = Polarität des Stroms
       B LSB: Wertebereich: 0..245
```

2.7 Motorendstufe abschalten

Die Motorendstufe wird abgeschaltet.

CH:	Length Payload	Command
Ch	0x01	0x09

ch : 1..5, je nach Steuerung

2.8 Motorendstufe einschalten

Die Motorendstufe wird eingeschaltet.

CH:	Length Payload	Command
Ch	0x01	0xf1

ch : 1..5, je nach Steuerung

2.9 Antriebsstatus 1 lesen

Statusbits eines Antriebskanals lesen

Request

CH:	Length Payload	Command
Ch	0x01	0x1d

Response

CH:	Length Payload	Response Type	Command	Data[0]	Data[1]
0x20	0x04	0x01	0x1d	n LSB	n MSB

ch : 1..5, je nach Steuerung

Data[0] : Bit 0: Busy Flag
Bit 1: -
Bit 2: Kanal hat eine erfolgreiche Referenzfahrt durchgeführt
Bit 3: -
Bit 4: -
Bit 5: Zeigt an, dass der nächste Weg zeitlich befrenzt mit maximalem Fahrstrom gefahren wird
Bit 6: -
Bit 7: -

Data[1] : Modus für Referenzfahrt
0: Aus
1: Referenzfahrt mit Endschalter
2: Referenzfahrt mit Stallguard
3: Referenzfahrt ohne Stallguard und ohne Endschalter (bis Anschlag)

3. Kommandos für Parameter

3.1 Parameter schreiben

CH:	Command	Parametername (C-String)	Arrayindex		Parametertyp	Data[0..n]
0x07	0x07	pname	aindex LSB	aindex MSB	ptype	value

pname: Name des Parameters als nullterminierter String (C-String)
aindex: Falls ein Parameter ein Array darstellt, z.B. beim Modus der I/O, dann ist dies der Arrayindex. Besteht der Parameter nur aus einem Element, ist der Arrayindex = 0.

ptype:
DPT_U8 = 0
DPT_U16 = 1
DPT_U32 = 2
DPT_S8 = 4
DPT_S16 = 5
DPT_S32 = 7
DPT_STRING = 8

value: Dies sind die Datenbytes. Die Anzahl richtet sich nach dem Parametertyp. Es wird immer das LSB zuerst übertragen.

Antwort auf ein "Transfer Parameter"

CH:	Commad	Error-Code
0x07	0x08	error

```
error: P2_EC_NO_ERROR                          = 0
       P2_EC_PARMETER_NOT_FOUND                = 1
       P2_EC_PARAMETER_TYPE_MISMATCH           = 2
       P2_EC_PARAMETER_INDEX_OUT_OF_RANGE      = 3
       P2_EC_TIMEOUT                           = 4
       P2_EC_PARAMETER_VALUE_OUT_OF_RANGE      = 5
       P2_EC_NO_CONNECTION                     = 6
       P2_EC_OPERATION_CANCALLED               = 7
       P2_EC_PARAMETER_INVALID                 = 255
```

3.2 Parameter lesen

CH:	Command	Parametername (C-String)	Arrayindex	
0x07	0x06	pname	aindex LSB	aindex MSB

pname: Name des Parameters als nullterminierter String (C-String)

aindex: Falls ein Parameter ein Array darstellt, z.B. beim Modus der I/O, dann ist dies der Arrayindex. Besteht der Parameter nur aus einem Element, ist der Arrayindex = 0.

Die Antwort auf ein "Request Parameter" ist ein "Transfer Parameter" Telegramm.

Beispiel: Parameter 'vsense' auf 160mV setzen.

Parameter schreiben

CH:	Command	Parametername (C-String)	Arrayindex		Parametertyp	Data[0..n]
0x07	0x07	'vsense',0	0	0	0	1

Das Feld Parametername wird in diesem Beispiel wie folgt kodiert:

```
0x76 0x73 0x65 0x6e 0x73 0x65 0x00
```

Response

CH:	Commad	Error-Code
0x07	0x08	00

3.3 Alle Parameter im Flash speichern

CH:	Command
0x07	0x09

3.4 Liste der Parameter

3.4.1 Allgemein

3.4.1.1 Fahrstrom

Stromeinstellung während einer Fahrt

Parametername	Array size	Parametertyp	Data[0]	Data[1]
drivecurrent_1	0	DPT_U16	value LSB	value MSB

```
value: 0 .. 100 [%]
```

3.4.1.2 Haltestrom

Stromeinstellung während des Stillstands

Parametername	Array size	Parametertyp	Data[0]	Data[1]
holdcurrent_1	0	DPT_U16	value LSB	value MSB

```
value: 0 .. 100 [%]
```

3.4.1.3 Haltestromverzögerung

Verzögerungszeit für Umschaltung von Fahr- auf Haltestrom

Parametername	Array size	Parametertyp	Data[0]	Data[1]
holddelay_1	0	DPT_U16	value LSB	value MSB

```
value: 0 .. 65535 [ms]
```

3.4.1.4 Overdrive-Strom

Anfahrstrom, wenn Overdrive-Flag gesetzt ist

Parametername	Array size	Parametertyp	Data[0]	Data[1]
overdrivelevel_1	0	DPT_U16	value LSB	value MSB

```
value: 0 .. 100 [%]
```

3.4.1.5 Overdrive-Zeit

Verzögerungszeit für Umschaltung von Overdrive- auf Fahrstrom

Parametername	Array size	Parametertyp	Data[0]	Data[1]
overdrivetime_1	0	DPT_U16	value LSB	value MSB

```
value: 0 .. 65535 [ms]
```

3.4.1.6 Mikroschrittauflösung

Parametername	Array size	Parametertyp	Data[0]
microstep_res	0	DPT_U8	value

```
value: Mikroschritte pro 90°
       0 = 256
       1 = 128
       2 = 64
       3 = 32
       4 = 16
       5 = 8
       6 = 4
       7 = 2 (halfstep)
       8 = 1 (fullstep)
```

3.4.1.7 Start-Stopp Frequenz

Parametername	Array size	Parametertyp	Data[0]	Data[1]
startstopfrequ_1	0	DPT_U16	value LSB	value MSB

```
value: 7,5MHz / (115 .. 30000) [Hz]
```

3.4.1.8 Drehrichtung umkehren

Parametername	Array size	Parametertyp	Data[0]
invertdir_1	0	DPT_U8	value

```
value:       0 = normal
             1 = Richtung invertiert
```

3.4.1.9 Polarität Referenzschalter

Parametername	Array size	Parametertyp	Data[0]
invertrefini_1	0	DPT_U8	value

```
value:      0 = Normally open
            1 = Normally closed
```

3.4.1.10 Absolute Referenzposition

Dieser Wert wird nach einer erfolgreichen Referenzfahrt in den Positionszähler übernommen.

Parametername	Array size	Parametertyp	Data[0]	Data[1]	Data[2]	Data[3]
abscounterrefpos_1	0	DPT_U32	value LSB	value	value	value MSB

```
value: 0 .. (2^32)-1
```

3.4.1.11 Spontanmeldungen freigeben

Spontanmeldungen sind per Vorgabe freigegeben.

Parametername	Array size	Parametertyp	Data[0]
enspontmsg	0	DPT_U8	value

```
value:      0 = Spontanmeldungen gesperrt
            1 = Spontanmeldungen freigegeben
```

3.4.1.12 Entprellzeit für digitale Eingänge

Parametername	Array size	Parametertyp	Data[0]	Data[1]
debouncetime	0	DPT_U16	value LSB	value MSB

```
value: 0 .. 65535 [ms]
```

3.4.2 Chopper Configuration

3.4.2.1 Chopper mode

Parametername	Array size	Parametertyp	Data[0]
choppermode	0	DPT_U8	value

```
value:      0 = Standard (spread cycle)
            1 = Constant toff
```

3.4.2.2 TBL - Blank time selection

Parametername	Array size	Parametertyp	Data[0]
blanktime	0	DPT_U8	value

```
value:      0 = 16 Clocks
            1 = 24 Clocks
            2 = 32 Clocks
            3 = 54 Clocks
```

3.4.2.3 HDEC - Hysteresis decrement

Parametername	Array size	Parametertyp	Data[0]
hdec	0	DPT_U8	value

```
value:      0 = 16 Clocks
            1 = 32 Clocks
            2 = 48 Clocks
            3 = 64 Clocks
```

3.4.2.4 HEND - Hysteresis low value

Parametername	Array size	Parametertyp	Data[0]
hend	0	DPT_U8	value

```
value:      0 = -3
            1 = -2
            2 = -1
            3 = 0
            4 = 1
            .
            .
            15 = 12
```

3.4.2.5 HSTR - Hysteresis start value

Parametername	Array size	Parametertyp	Data[0]
hstrt	0	DPT_U8	value

```
value:      0 = 1
            1 = 2
            .
            .
            7 = 8
```

3.4.2.6 RNDTF - Random off time

Parametername	Array size	Parametertyp	Data[0]
randomtofftime	0	DPT_U8	value

```
value:      0 = fixed
            1 = random
```

3.4.2.7 FDMODE - Disable current comparator

Parametername	Array size	Parametertyp	Data[0]
fdmode	0	DPT_U8	value

```
value:      0 = enable
            1 = disable
```

3.4.2.8 Sine wave offset

Parametername	Array size	Parametertyp	Data[0]
sinewaveoffset	0	DPT_U8	value

```
value:      0 = -3
            1 = -2
            2 = -1
            3 = 0
            4 = 1
            .
            .
            15 = 12
```

3.4.2.9 Fast decay time setting

Parametername	Array size	Parametertyp	Data[0]
fastdecaytime	0	DPT_U8	value

```
value:      0 = slow
            1 = 32 clk
            2 = 64 clk
            3 = 96 clk
            4 = 128 clk
            5 = 160 clk
            6 = 192 clk
            7 = 224 clk
            8 = 256 clk
            9 = 288 clk
            10 = 320 clk
            11 = 352 clk
            12 = 384 clk
            13 = 416 clk
            14 = 448 clk
            15 = 480 clk
```

3.4.2.8 TOFF - Driver off time

Parametername	Array size	Parametertyp	Data[0]
offtime	0	DPT_U8	value

```
value:      0 = disable
            1 = 1
            2 = 2
            .
            .
            15 = 15
```

3.4.3 Drive Configuration

3.4.3.1 SLPH - Slope control high

Parametername	Array size	Parametertyp	Data[0]
slopectrlhigh	0	DPT_U8	value

```
value:      0 = min
            1 = min + tc
            2 = med + tc
            3 = max
```

3.4.3.2 SLPL - Slope control low

Parametername	Array size	Parametertyp	Data[0]
slopectrllow	0	DPT_U8	value

```
value:      0 = min
            1 = min
            2 = med
            3 = max
```

3.4.3.3 DISS2G - Disable short to GND protection

Parametername	Array size	Parametertyp	Data[0]
diss2g	0	DPT_U8	value

```
value:      0 = enabled
            1 = disabled
```

3.4.3.4 TS2G - Short to GND protection timer

Parametername	Array size	Parametertyp	Data[0]
ts2g	0	DPT_U8	value

```
value:      0 = 3,2µs
            1 = 1,6µs
            2 = 1,25µs
            3 = 0,8µs
```

3.4.3.5 V-Sense

Parametername	Array size	Parametertyp	Data[0]
vsense	0	DPT_U8	value

v

alue: 0 = 300mV
1 = 160mV

3.4.4 Stall Guard Current Setting

3.4.4.1 CS - Current scale

Parametername	Array size	Parametertyp	Data[0]
currentscale	0	DPT_U8	value

value: 0..31 entspricht 0..100% Strangströme

3.4.4.2 SGT - Stallguard threshold value

Parametername	Array size	Parametertyp	Data[0]
sgt	0	DPT_U8	value

value: -64 .. +63 (Zweierkomplement)

3.4.4.3 SFILT - Stallguard filter

Parametername	Array size	Parametertyp	Data[0]
sgfilter	0	DPT_U8	value

value:
0 = Standard mode
1 = Filtered mode

4. Kommandos für allgemeine Hardwarefunktionen

4.1 Digitale Eingänge entprellt lesen

Request

CH:	Length Payload	Command
0x06	0x01	0x02

Response

CH:	Length Payload	Response Type	Command	Data[0]	Data[1]	Data[2]
0x20	0x05	0x01	0x02	n LSB	n	n MSB

Data[0] : Digitale Eingänge bitweise nach der Entprellung

Data[1] : 0: Überstromsicherung in Ordnung
1: Überstromsicherung hat ausgelöst

Data[2] : 0: Temperatur in Ordnung
1: Übertemperatur

4.2 Analoge Eingänge lesen

Request

CH:	Length Payload	Command
0x06	0x01	0x04

Response

CH:	Length Payload	Response Type	Command	Data[0..n]
0x20	len	0x01	0x04	Data

Karte HP/LP

len = 8

Data[0,1] : Spannungmesseingang (Wert*10, Fixed Komma)
Data[2,3] : Strommesseingang (Wert*10, Fixed Komma)
Data[4,5] : Versorgungsspannung (Wert*10, Fixed Komma)

Karte Backpack

len = 20

Data[0,1] : Versorgungsspannung (Wert*10, Fixed Komma)
Data[2,3] : Messwert Analogeingang 1
Data[4,5] : Messwert Analogeingang 2
Data[6,7] : Messwert Analogeingang 3
Data[8,9] : Messwert Analogeingang 4
Data[10,11] : Messwert Analogeingang 5
Data[12,13] : Messwert Analogeingang 6
Data[14,15] : Messwert Analogeingang 7
Data[16,17] : Messwert Analogeingang 8

4.3 Bord-Temperatur lesen

Request

CH:	Length Payload	Command
0x06	0x01	0x03

Response

CH:	Length Payload	Response Type	Command	Data[0]	Data[1]
0x20	0x04	0x01	0x03	value LSB	value MSB

Karte HP/LP

value : Temperatur mit der Auflösung 0,5°C

Karte Backpack

Die Steuerung "Backpack" verfügt über keinen Temperatursensor, deshalb antwortet diese mit dem Fehlercode "*Unknown Command*".

A. Berechnung der CRC

A.1 C#

```
/// <summary>
///
/// </summary>
/// <param name="x"></param>
/// <returns></returns>
ushort _Swap(ushort x)
{
    UInt16 temp;

    temp = (ushort)(x << 8);
    temp = (ushort)(temp | (x >> 8)); return (temp);
}
```

```
/// <summary>
/// Berechnet CRC mit dem Generatorpolynom X^16+X^12+X^5+1.
/// </summary>
/// <param name="ms"></param>
/// <returns></returns>
ushort calcCrc(byte[] buffer)
{
    ushort crc = 0;

    for (int i = 0; i < buffer.Length; ++i)
    {
        crc = (ushort)(_Swap((crc)) ^ (((byte)(buffer[i])))); crc = (ushort)((crc) ^
        (((byte)((crc)) >> 4)));
        crc = (ushort)((crc ^ ((_Swap((byte)(crc)) << 4))) ^
    (((byte)(crc) << 5)));
    }

    return (crc);
}
```

A.2 C

```
*---------------------------------------------------------------------
       Funktion: _Swap
              Vertauscht die beiden Bytes eines 16Bit Integers.

       Eingang: 16Bit Integer
       Ausgang: 16Bit Integer mit vertauschten Bytes

---------------------------------------------------------------------*/

U16 _Swap(U16 x)
{
       U16 temp;

       temp = x << 8;
       temp = temp | (x >> 8);
       return (temp);
}


/*---------------------------------------------------------------------
       Funktion: CRC16
                 Berechnet CRC mit dem Generatorpolynom
                 X^16+X^12+X^5+1. Das CRC-Register wird vor dem ersten
                 Datenbyte mit 0 initialisiert. Dann wird die Funktion
                 für jedes Daten-byte aufgerufen.

                          Adresse des CRC-
       Eingang:      *acrc: Registers
                     c    : Datenbyte

       Ausgang: -

---------------------------------------------------------------------*/

void CRC16(U16 *acrc, unsigned char c)
{
       (*acrc) = (U16)(_Swap((*acrc)) ^ (((unsigned char)(c))));
       (*acrc) = (U16)((*acrc) ^ (((unsigned char)((*acrc)) >> 4)));
       (*acrc) = (U16)(((*acrc) ^ ((_Swap((unsigned char)((*acrc))) << 4))) ^
       (((unsigned char)((*acrc)) << 5)));
}
```