EIBA Handbook Series

Release 3.0

Volume 3:
System Specifications

Part 6: Application Interfaces

Chapter 1: Application Programmer's Interface

08.03.1999

Table of Contents

Page left blank intentionally.

1. Characteristics of BCU RAM and EEPROM Variables

1.1 API RAM Variables and RAM Space

1.1.1 Registers B to N

The RAM area of 50h to 5Ch can be used as register memory space or as temporary RAM storage by the application programmer.

Attention: ***"registers" B to N may also be used by ROM functions or for parameter passing or as temporary variables. See description of API functions!***

Registers can be used to store parameters for functions to be called.

Attention: Before the next complete execution of the user program all registers are cleared by the system firmware.

1.1.2 Stack Space available for the Internal Application

Internal applications can use a stack of t.b.d. bytes maximum for own purposes. If an application uses more stack space, this can result in severe conflicts with the stack needs by the BCU firmware.

1.1.3 Internal Application (i.e. "User") RAM

To be completed.

1.2 API EEPROM Variables and EEPROM Space

1.2.1 Group_object Table Pointer "CommsTabPtr" and "CommsTabPtr2"

CommsTabPtr is a one byte EEPROM variable. CommsTabPtr contains the byte offset to the location of the group_object table in EEPROM, if it is not zero. If CommsTabPtr is not zero then the group_object table starts at address 100H+[CommsTabPtr].

If CommsTabPtr is zero then the EEPROM variable CommsTabPtr2 exists which contains a Big Endian absolute pointer to the group_object table location.

1.2.2 User Initialisation Routine Pointer "UsrInitPtr" and "UsrIntiPtr2"

If CommsTabPtr is not zero then UsrInitPtr exists and contains the byte offset to the start address of the user initialisation routine in EEPROM. The initialisation routine starts at 100H+[UsrInitPtr].

If CommsTabPtr is zero then the EEPROM variable UsrInitPtr2 exists which contains a Big Endian absolute pointer to the start address of the user initialisation routine.

The initialisation routine is called once at the startup time of the internal user application. It must be written as a subroutine, i.e. it must be terminated by "rts".

1.2.3 User Program Pointer "UsrPrgPtr" and "UsrPrgPtr2"

If CommsTabPtr is not zero then UsrPrgPtr exists and contains the byte offset to the start address of the internal user application program in EEPROM. Then the internal user application program starts at 100H+[UsrPrgPtr].

If CommsTabPtr is zero then the EEPROM variable UsrPrgPtr2 exists which contains a Big Endian absolute pointer to the start address of the internal user application program.

This routine is called periodically if the BCU is in normal operation mode and the PEI type expected is equal to the PEI adaptor's PEI type measured by the A/D converter of the BCU.

This routine must be written as a subroutine, i.e. it must be terminated by "rts".

1.2.4 User Save Routine Pointer "UsrSavPtr" and "UsrSavPtr2"

If CommsTabPtr is not zero then UsrSavPtr exists and contains the byte offset to the start address of the user save routine in EEPROM. The user save routine starts at 100H+[UsrSavPtr].

If CommsTabPtr is zero then the EEPROM variable UsrSavPtr2 exists which contains a Big Endian absolute pointer to the start address of the user save routine.

This routine is called if a bus power loss is detected at a bus-powered device and the internal user application program is not inactivated.

After calling this routine the BCU is reset. Nonetheless this routine must be written as a subroutine, i.e. it must be terminated by "rts".

1.2.5 Internal User Application (i.e. "user") EEPROM space

This is the EEPROM space usable by the internal user application program and the optional application programmer-writtten PEI driver for PEI 10.

Size: To be completed.

1.3 API Tables

1.3.1 OR_TAB

Symbol: **OR_TAB**

Address: Mask 0010h: 0020H
Mask 0011h: 0020H
Mask 0020h: 0020H
Mask 1013h: 0020H

OR_TAB

Addr. (hex)	Value (binary)
20	0000 0001
21	0000 0010
22	0000 0100
23	0000 1000
24	0001 0000
25	0010 0000
26	0100 0000
27	1000 0000

1.3.2 AND_TAB

Symbol: **AND_TAB**

Address: Mask 0010h: 0028H
Mask 0011h: 0028H
Mask 0020h: 0028H
Mask 1013h: 0028H

AND_TAB

Addr.(hex)	Value (binary)
28	1111 1110
29	1111 1101
2A	1111 1011
2B	1111 0111
2C	1110 1111
2D	1101 1111
2E	1011 1111
2F	0111 1111

Both tables are helpfull for **MASK**-operation. If you need a value of these tables, you need only a one-byte-offset. This is helpfull for loops.

Example:

```
        ldx     #AND_TAB ;ptr to AND_TAB
loop    lda     ...
        and     ,X       ;MASK-operation
        sta     ...
        :
        incx
        bra     loop
```

1.4 System Timer

There are 4 software-timers: 0 to3.

The timers 0 and 1 are reserved for the system software.

The timers 2 and 3 are available to the user.

The timer 2 is also used by the debounce-function. Do not use it a second time, if debouncing is used.

A timer can be used in one of two operation modes.

In operation mode, 0 a timer is initialized with a run-time. If this time is expired, then this is flagged. The timer may be restarted during operation.

In operation mode 1, the timer calculates the time since the last call to the timer function. The user must take care to avoid range overflows.

A timer has a resolution of 8 bits. It can be operated in five different time-ranges. E.g. a timer-value of 5 in time-range 2 means about 40ms.

Range	Time-Unit
1	±. 0.5 ms
2	± 8.0 ms
3	± 130 ms
4	± 2.1 s
5	± 33 s

Note: A not-initialized timer has the status of an expired timer.

1.5 User Timer

There is another set of support routines that can be used to define additional User timers.

Each pair of User-Timers can have a different time base. But each timer must be updated at least one time per timer-tick in its own time base.

A description-block in EEPROM is used to describe the User timers. Each User timer function needs a pointer to this description block.

The structure of the EEPROM description block is:

Offset in block	Length (bytes)	Meaning
0	1	Pointer to RAM-data
1	1	Time base1 0 and 1
2	1	Time base 2 and 3
…		...

Time bases:

Number	Minimum Time Resolution
0	ca. 130ms
1	ca. 260ms
2	ca. 520ms
3	ca. 1.0s
4	ca. 2.1s
5	ca. 4.2s
6	ca. 8.4s
7	ca. 17s
8	ca. 34s
9	ca. 1.1min
10	ca. 2.2min
11	ca. 4.5min
12	ca. 9.0min
13	ca. 18min
14	ca. 35min
15	ca. 1.2h

For each user timer, a byte must be reserved in RAM. All of these bytes must by allocated in one block. The first byte in this block corresponds with the first user timer, the second byte corresponds with the second user timer and so on. The bit 7 in each byte is reserved, the bits 0-6 is used for the timer value (0-127). This value will be decremented only if you use the (update function) U_GetTM. You can set the user timer if you use the function **U_SetTM**, or you can load the corresponding RAM directly with the value. The bit7 must be 0.

1 In the mask-version 1.0 only the low nibble is evaluated. It serves as the time base for both timers.

2. API Function Reference

The API function reference lists all the API functions that support the task of the programmer of the internal user application. The interface to the functions is a pure assembler interface. If you want to interface at e.g. C language level then you must write your own interface adaptations (header files).

The estimated value given for "Watchdog-time" is an indication of how much watchdog-time this routine needs for processing. I.e. processing-time since latest triggering of watchdog (inside that routine).

This value is necessary to estimate when the watchdog must be triggered.

2.1 Communication Object Manipulation Functions

2.1.1 U_flagsGet Reading RAM Flags

Symbol: **U_flagsGet**

Description: gets the RAM-flags of the specified communication object.

Inputs: A = communication object number

	Call Address	Outputs	Changed Registers
Mask 0010h	0C8CH	See below	A, X, RegC, RegJ
Mask 0011h	0C9DH	See below	A, X, RegC, RegJ
Mask 0012h	0C9DH	See below	A, X, RegC, RegJ
Mask 0020h	505AH	See below	A, X, RegC, RegJ
Mask 1013h	3D14H	See below	A, X, RegJ RegA = RAM-flags

Watchdog-time: (to be completed) µs

Outputs: RegB

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Meaning	undefined Mask 1013h: zero				update flag	data request flag	transmission status	

2.1.2 U_flagsSet Writing RAM flags

Symbol: **U_flagsSet**

Description: sets the RAM-flags of the specified communication object.

Inputs: A = communication object number

RegB = RAM-flags

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Meaning	undefined				update flag	data request flag	transmission status	

	Call Address	Outputs	Changed Registers
Mask 0010h	0C94H	None	A, X, RegB, RegC, RegJ
Mask 0011h	0CB3H	None	A, X, RegB, RegC, RegJ
Mask 0012h	0CB3H	None	A, X, RegB, RegC, RegJ
Mask 0020h	505HD	None	A, X, RegB, RegC, RegJ
Mask 1013h	3D11H	None	A, X, RegB, RegJ

Watchdog-time: (to be completed) µs

2.1.3 U_testObj Testing RAM flags

Symbol: **U_testObj**

Description: fetches the RAM-flags from the specified object and resets the Update-flag.

Inputs: A = communication object number

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	0CA5H		A, X, RegB, RegC, RegJ
Mask 0012h	0CA5H		A, X, RegB, RegC, RegJ
Mask 0020h	507BH		A, X, RegB, RegC, RegJ
Mask 1013h	3D17H		A, X, RegJ

Watchdog-time: (to be completed) µs

Outputs: RegC = RAM-flags (right adjusted)

Zero-flag = **NOT** Update-flag

2.1.4 U_transRequest Setting Transmit Request

Symbol: **U_transRequest**

Description: sets a transmit-request in the specified communication object. If a telegram is being currently transmitted for this object then the transmit-request is not set.

Inputs: A = communication object number

	Call Address	Outputs	Changed Registers
Mask 0010h	0D91H		A, X, RegB, RegC, RegJ
Mask 0011h	0DB9H		A, X, RegB, RegC, RegJ
Mask 0012h	0DB9H		A, X, RegB, RegC, RegJ
Mask 0020h	507EH		A, X, RegB, RegC, RegJ
Mask 1013h	3D0EH		A, X, RegJ

Watchdog-time: (to be completed) µs

Outputs: Carry
0 = OK
1 = transmit-request not set

Note

- Mask 1013h: Due to a slower bus-transmission speed its strongly recommended to evaluate the return value of this function. Changing the object value while transmission-request is set may cause unpredictable telegrams on the bus.

2.1.5 Standard Callback

Symbol: **AL_SAPcallback**

Description: Complete Handling of GroupCommunicationObjects and User-EIB-Objects

Inputs: X Index to message

Carry 0 : Polling mode: call one times in a cycle

1 : Message mode: call if a message for an object is received either a Communication Object or a EIB Object (SAP = FFh)

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	-	-	-
Mask 0012h	-	-	-
Mask 0020h	5081H	See below	A, X, RegB -RegN
Mask 1013h	-	-	-

Watchdog-time: (to be completed) µs

Outputs: X Index to message

A state
- 0 Ok (Return messagebuffer to system)
- 1 break and free buffer
- 2 hold buffer and break
- 3 send message and break

Example

```
AppCallBack
            bcc     Nomsg
* Message received
            lda     EIBMSG_ASAP,x       Load SAP number
            cmp     #SPECIAL_SAP
            bne     STDcallback
*handle special SAP
            ...
            ...

STDcallback
            sec                         ;mark message
Nomsg       jmp     AL_SAPcallback      ;Use standard callback
```

2.2 EEPROM Manipulation Support Functions

2.2.1 Eewrite Writing to EEPROM

Symbol: **EEwrite**

Description: writes a byte to the specified location in memory.
The write operation takes up to 20ms.
Attention: Update checksum if necessary!

Inputs: A = value to write
X = offset in EEPROM

	Call Address	Outputs	Changed Registers
Mask 0010h	0C2DH	None	RegB, RegC, RegH
Mask 0011h	0C38H	None	RegB, RegC, RegH
Mask 0012h	0C38H	None	RegB, RegC, RegH
Mask 0020h	503FH	None	RegB, RegC, RegH
Mask 1013h	3D08H	None	RegB, RegC, RegH

Watchdog-time: internally set by function

2.2.2 EEsetChecksum Updating the checksum

Symbol: **EEsetChecksum**

Description: updates the checksum byte of the EEPROM. This function must be called if any byte inside the check range is modified by the user.

Inputs: None

	Call Address	Outputs	Changed Registers
Mask 0010h	0C5DH	None	A, X, RegB, RegC, RegH
Mask 0011h	0C68H	None	A, X, RegB, RegC, RegH
Mask 0012h	0C68H	None	A, X, RegB, RegC, RegH
Mask 0020h	503CH	None	A, X, RegB, RegC, RegH
Mask 1013h	3D05H	None	A, X, RegB, RegC, RegH

Watchdog-time: internally set by function

2.2.3 U_EE_WriteBlock Write Block to EEPROM

Symbol: **U_EE_WriteBlock**

Description : Write a block of 4 bytes to the EEPROM

Inputs RegH:RegI Memory Address
RegK,RegL,RegM,RegN Memory Block to write

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	-	-	-
Mask 0012h	-	-	-
Mask 0020h	5099H	None	A, X, RegB, RegC, RegJ
Mask 1013h	-	-	-

Watchdog-time: (to be completed) µs

2.3 Application Support Functions

2.3.1 U_debounce Debouncing

Symbol: **U_debounce**

Description: debounces a complete byte. As long as the debounce-time is not yet expired or the value is changing the latest debounced value is returned.

- Mask 00xx: UserRAM = last input value
  UserRAM+1 = debounced value

  Note: It is not necessary to initialize the software timer 2.

- Mask 10xx: UserRAM1 = last input value
  UserRAM1+1 = debounced value

  Note: It is not necessary to initialize the system-timer #2.

Inputs: A = value (byte) to debounce
X = debounce-time in 0.5 ms-units

	Call Address	Outputs	Changed Registers
Mask 0010h	0C64H	See below	X, RegB, RegC, RegD, RegE, RegF, RegG
Mask 0011h	0C75H	See below	X, RegB, RegC, RegD, RegE, RegF, RegG
Mask 0012h	0C75H	See below	X, RegB, RegC, RegD, RegE, RegF, RegG
Mask 0020h	C5051H	See below	X, RegB, RegC, RegD, RegE, RegF, RegG
Mask 1013h	3D26H	See below	X, RegB, RegC, RegD, RegE, RegF, RegG

Watchdog-time: (to be completed) µs

Outputs: A = debounced value (byte)

Effects:

- Mask 00xx uses software-timer 2
  changed RAM-locations: UserRAM, UserRAM + 1
- Mask 10xx uses system-timer #2
  changed RAM-locations: UserRAM1, UserRAM1 + 1

Symbol: **U_deb10**

Description: This function is the same as U_debounce except that a fixed debounce-time of 10 ms is used. Therefore the X register has not to be loaded.

Inputs: A = value (byte) to debounce

	Call Address	Outputs	Changed Registers
Mask 0010h	-		
Mask 0011h	0C73H	See U_debounce	See U_debounce
Mask 0012h	0C73H	See U_debounce	See U_debounce
Mask 0020h	504BH	See U_debounce	See U_debounce
Mask 1013h	3D23H	See U_debounce	See U_debounce

Symbol: **U_deb30**

Description: This function is the same as U_debounce except that a fixed debounce-time of 30 ms is used.

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	0C6FH	See U_debounce	See U_debounce
Mask 0012h	0C6FH	See U_debounce	See U_debounce
Mask 0020h	504EH	See U_debounce	See U_debounce
Mask 1013h	3D20H	See U_debounce	See U_debounce

Watchdog-time: (to be completed) µs

2.3.2 U_delMsgs Ignoring messages to the user

Symbol: **U_delMsgs**[2]

Description: removes any messages addressed to the user program. Call this function in the USER main routine, if you do not expect any messages. Otherwise the buffer resources in the BCU may become exhausted due to external faults.

Inputs: None

	Call Address	Outputs	Changed Registers
Mask 0010h	0C82H	None	A, X, RegB
Mask 0011h	0C93H	None	A, X, RegB
Mask 0012h	0C93H	None	A, X, RegB
Mask 0020h	5057H	None	A, X, RegB
Mask 1013h	3D89H	None	A, X, RegB

Watchdog-time: (to be completed) µs

2 The same effect as calling this function can be achieved, for the mask-version 1.1 only, if bit 7 in the EEPROM-parameter RouteCnt is set.

2.3.3 U_readAD Doing AD conversion

Symbol: **U_readAD**

Description: starts A/D conversion for the specified port (not necessarily a PEI I/O port!) and reads the result. This operation is repeated the specified number of times. The sum of the values of all read operations is returned.

Note: This function is valid for A/D conversion of any analog port values in the bus access module, no matter whether the port line is connected to a PEI input line.

Inputs: A = channel number of analog port
X = number of read operations

	Call Address	Outputs	Changed Registers
Mask 0010h	0D35H	See below	A, X
Mask 0011h	0D54H	See below	A, X
Mask 0012h	0D54H	See below	A, X
Mask 0020h	506CH	See below	A, X
Mask 1013h	3D38H	See below A = value of last AD-conversion	A, X

Watchdog-time: (to be completed) µs

Outputs: RegD:RegE = sum of read values. Precision is implementation-specific.

Note:

- Mask 1013h: If only one AD-conversion was requested the contents of the Accu is identical with the contents of RegD:RegE

2.3.4 U_map Characterisation function

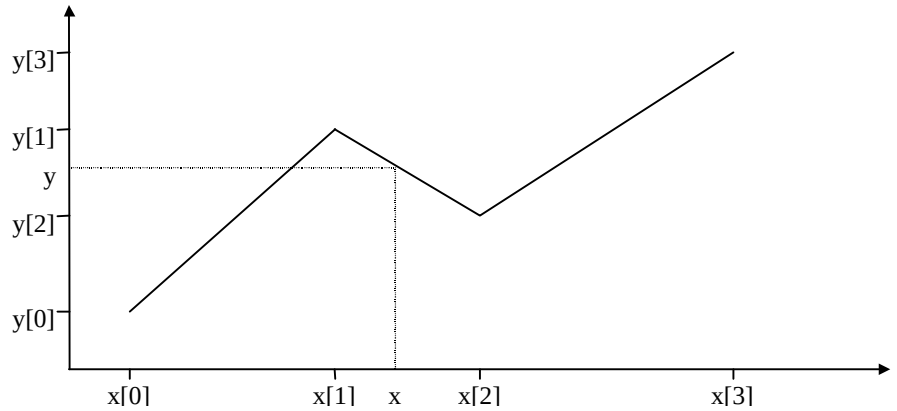
Symbol: **U_map**

Description: maps the input value by the use of a conversion table. Input value and result are 16-bit signed integer numbers and cover all 4 quadrants. For the conversion a table of x-y-value pairs is used. The input value must be inside the x-value range. For values in between two x-y-value pairs linear interpolation is used.

The interpolation-formula used is:

$$Y = [(X-X1)*(Y2-Y1)]/(X2-X1) + Y1$$

All intermediate results must not exceed the signed integer value range.



The conversion table is assumed to be in EEPROM and has the following format: (x-values in ascending order, LSByte is the high byte).

Length (x-y-pair count) (1 byte)	
X0	Y0
X1	Y1
...	...
low word	high word

Inputs: RegB:RegC = value to be mapped (signed integer)
X = pointer to conversion table (EEPROM-offset)

	Call Address	Outputs	Changed Registers
Mask 0010h	0C9BH	See below	A, X, RegD, RegE, RegF, RegG, RegH, RegI
Mask 0011h	0CBAH	See below	A, X, RegD, RegE, RegF, RegG, RegH, RegI
Mask 0012h	0CBAH	See below	A, X, RegD, RegE, RegF, RegG, RegH, RegI
Mask 0020h	5069H	See below	A, X, RegD, RegE, RegF, RegG, RegH, RegI
Mask 1013h	3D86H	See below	A, X, RegD, RegE, RegF, RegG, RegH, RegI

Watchdog-time: (to be completed) µs

Outputs: RegB:RegC = result (signed integer)
Carry = 0 OK
1 conversion error, result not valid

Note:

- Masks 0010h, 0011h: These mask versions only operate in the first quadrant!

2.3.5 U_GetAccess Get actual Access Level

Symbol: **U_GetAccess**

Description : gets the actual accesslevel of the system

Inputs None

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	-	-	-
Mask 0012h	-	-	-
Mask 0020h	5096H	A = Accesslevel	None
Mask 1013h	-	-	-

Watchdog-time: (to be completed) µs

2.3.6 U_SetPollingRsp Set Polling Response

Symbol: **U_SetPollingRsp**

Description : Set the value to response at poll request.

Inputs A Value to response

Reserved Values

FFh	No user running / not set from user
F0h-Fdh	Reserved
FEh	Fill character no slave responding
00h	Set at Userstartup

Bit 7 1: Systemstate
0: Userstate

Bit 6 Error

Bit 5 Alarm

Bit 4 Reseverd

Bit 0-3: Usercode

For more informations please refer to Part 3/7 "Interworking".

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	-	-	-
Mask 0012h	-	-	-
Mask 0020h	50AEH	None	None
Mask 1013h	-	-	-

2.4 PEI Support Functions

2.4.1 U_ioAST Binary Port Access

Symbol: **U_ioAST**

Description: Digital ports can be accessed by the function U_ioAST that allows reading and/or writing digital I/O pins 1 to 4. Reading or writing can be selected for each PEI I/O line via I/O flags. Accumulator and RegB serve as parameters for U_ioAST.

Via I/O-flags reading or writing can be selected for each bit/pin. For both input and output the bits are mapped to the PEI-port as follows:

I/O-Bit #	Input # or Output #	PEI-Pin
0	1	3
2	2	2
2	3	4
3	4	7

Encoding of read / write bits: 0 means off, i.e. 0 V

1 means on, i.e. 5 VDC

Inputs: Accumulator: for I/O flags and values

bit #	7	6	5	4	3	2	1	0
meaning	I/0-flags 0 = read 1 = write				write bit values (valid only if write selected for that bit)			
I/O bit #	3	2	1	0	3	2	1	0

	Call Address	Outputs	Changed Registers
Mask 0010h	0DA7H	See below	A, X, RegB, RegC, RegD
Mask 0011h	0DCFH	See below	A, X, RegB, RegC, RegD
Mask 0012h	0DCFH	See below	A, X, RegB, RegC, RegD
Mask 0020h	5066H	See below	A, X, RegB, RegC, RegD
Mask 1013h	3D35h	See below	A, X, RegB, RegC

Watchdog-time: (to be completed) µs

Outputs: RegB: for bit values to be read

bit #	7	6	5	4	3	2	1	0
meaning	bit values of A before read/write				read bit values (valid only if read selected for that bit)			
I/O bit #	3	2	1	0	3	2	1	0

Note: Digital Input Value Sampling

Digital input value sampling must be done in the internal user application program with the help of several calls of U_ioAST. This means, that the sampling rate per time unit and debouncing is left to the user application programmer.

2.4.2 S_AstShift / S_LastShift Data-Block Exchange via Serial Synchronous PEI (PEI-Type 14 only)

Symbol: **S_AstShift / S_LastShift**

Description: The specified data block is exchanged via the serial PEI-interface. The synchronous protocol, that runs at PEI type 14, and the data block format is described in Chapter 3/6/3 “External Message Interface” of this handbook.

The function "S_AstShift" uses a ca. 130 ms timeout and the function "S_LAstShift" uses a ca. 1 s timeout. I.e. the data block exchange must be completed within these timeout times.

The data format is shown below:

Message length in bytes (1 byte)	data

Inputs: X = pointer to data block

	Call Address		Outputs	Changed Registers
	S_AstShift	S_LAstShift		
Mask 0010h	1103H	1101H	See below	A, RegB, RegC, RegD, RegE, RegF, RegG, RegH, RegI
Mask 0011h	1117H	1115H	See below	A, RegB, RegC, RegD, RegE, RegF, RegG, RegH, RegI
Mask 0012h	1117H	1115H	See below	A, RegB, RegC, RegD, RegE, RegF, RegG, RegH, RegI
Mask 0020h	5042H	5045H	See below	A, RegB, RegC, RegD, RegE, RegF, RegG, RegH, RegI
Mask 1013h	3D3Eh	3D3Bh	See below	A, RegB, RegC, RegD, RegE, RegF, RegG, RegH, RegI

Outputs: X = pointer to data block (contains response)
Carry: 0 = communication Ok
1 = communication failed

Note: When using the serial interface, it is very important to change the "own" handshake signal not before the end of the stop bit. Otherwise, the transmission will fail.

2.4.3 U_SerialShift Byte exchange via Serial PEI

Symbol: **U_SerialShift**

Description : The specified byte is exchanged via the serial PEI-interface. The function U_SerialShift" uses a ca. 130 ms-timeout. I.e. the byte-exchange must be completed within the timeout-time.

Inputs: A = Byte to be transferred

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	-	-	-
Mask 0012h	0C90H	See below	A, RegB, RegC, RegD, RegE, RegF, RegG,RegI
Mask 0020h	5048H	See below	A, RegB, RegC, RegD, RegE, RegF, RegG,RegI
Mask 1013h	3D41H	See below	A, RegB, RegC, RegD, RegE, RegF, RegG,RegH

Watchdog-time: (to be completed) µs

Outputs: A = received byte
Carry = 0 : communication O.K.
1 : communication failed

Note: **Important Remarks to the functions S_AstShift, S_LastShift and U_SerialShift**

1. These functions may only be used in combination with PEI-type 14.
2. Long wait times may cause serious problems on the bus (busy-acknowledges).

2.5 Timer Functions

2.5.1 TM_Load Starting the timer

Symbol: **TM_Load**

Description: The specified timer is initialized with operation mode and time-range.
In addition, in operation mode 0 the run-time is set and the timer is started.

Inputs: A = (see below)

X = run-time (operation mode 0 only)

Input in A:

- Mask 0010h, 0011h, 0012h, 0020h:

Bit #	7	6	5	4	3	2	1	0
Meaning	must be 0		timer #		operation mode	timer range		

- Mask 1013h:

Bit #	7	6	5	4	3	2	1	0
Meaning	timer #				operation mode	timer range		

	Call Address	Outputs	Changed Registers
Mask 0010h	0E0CH	None	A, X, RegB, RegC, RegD, RegE, RegF
Mask 0011h	0E2BH	None	A, X, RegB, RegC, RegD, RegE, RegF
Mask 0012h	0E2BH	None	A, X, RegB, RegC, RegD, RegE, RegF
Mask 0020h	5039H	None	A, X, RegB, RegC, RegD, RegE, RegF
Mask 1013h	3D1AH	None	A, X, RegB, RegC, RegD, RegE, RegF

Watchdog-time: (to be completed) µs

2.5.2 TM_GetFlg Reading the Timer Status

Symbol: **TM_GetFlg**

Description: The timer-status is returned.
In operation mode 0, it returns if the run-time is expired.
In operation mode 1, the time since the last call to this function is returned.

Inputs: A = timer number

	Call Address	Outputs	Changed Registers
Mask 0010h	0E2AH		A, X, RegB, RegC, RegD, RegE
Mask 0011h	0E49H		A, X, RegB, RegC, RegD, RegE
Mask 0012h	0E49H		A, X, RegB, RegC, RegD, RegE
Mask 0020h	5036H		A, X, RegB, RegC, RegD, RegE
Mask 1013h	3D1DH		A, X, RegB, RegC, RegD, RegE

Watchdog-time: (to be completed) µs

Outputs:

Operation Mode 0	Operation Mode 1
Carry = 0 time not yet expired 1 time expired	A = time since the last call

2.5.3 U_SetTM Setting a User Timer

Symbol: **U_SetTM**

Description: loads a User-Timer.

Inputs:

- Mask 0010h, 0011h, 0012h, 0020h:

  A = user-timer number
  X = pointer to EEPROM description block
  RegE = time to be set

- Mask 1013h:

  A = See table below

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Meaning	Location of timer-RAM-block 0 = UserRAM1 1 = UserRAM2	User-Timer number						

  X = pointer to EEPROM-description-block

  RegE = (see below)

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Meaning	not evaluated	time to be set						

	Call Address	Outputs	Changed Registers
Mask 0010h	0D8AH	None	A, X, RegB, RegC, RegD
Mask 0011h	0DB3H	None	A, X, RegB, RegC, RegD
Mask 0012h	0DB3H	None	A, X, RegB, RegC, RegD
Mask 0020h	506FH	None	A, X, RegB, RegC, RegD
Mask 1013h	3D32H	None	A, X RegE if Bit #7 is set

Watchdog-time: (to be completed) µs

Symbol: **U_SetTMx**

Description: This function is the same as U_SetTM except that the pointer to the pointer to EEPROM description block is fetched from the byte directly before the user main program.

Inputs: See U_SetTM

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	0DAFH	See U_SetTM	See U_SetTM
Mask 0012h	0DAFH	See U_SetTM	See U_SetTM
Mask 0020h	5072H	See U_SetTM	See U_SetTM
Mask 1013h	3D2FH	See U_SetTM	See U_SetTM

Watchdog-time: (to be completed) µs

Note: If you load the timer with the timer value 0, the timer is always expired. If you load the timer with 1, the timer can be immediate expired, because the timer tolerance is one timer tick. If you load the timer with a value >127, the bit7 will be ignored.

2.5.4 U_GetTM Reading the User Timer Status

Symbol: **U_GetTM**

Description: gets the status of a user timer. This function is the only function that updates this (specified) timer.

Inputs:

- Mask 0010h, 0011h, 0012h, 0020h:

  A = user-timer number
  X = pointer to EEPROM-description-block

- Mask 1013h:

  A = See table below
  X = pointer to EEPROM-description-block

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Meaning	Location of timer-RAM-block 0 = UserRAM1 1 = UserRAM2	User-Timer number						

	Call Address	Outputs	Changed Registers
Mask 0010h	0D4DH	See below	A, X, RegB, RegC, RegD
Mask 0011h	0D71H	See below	A, X, RegB, RegC, RegD
Mask 0012h	0D71H	See below	A, X, RegB, RegC, RegD
Mask 0020h	5060H	See below	A, X, RegB, RegC, RegD
Mask 1013h	3D2CH	See below	A, X, RegB, RegC, RegD

Watchdog-time: (to be completed) µs

Outputs: Zero-Flag = 0 timer not yet expired
1 timer expired

Note: Periodically update the User-Timer. Otherwise some "timer-ticks" may be lost.

Symbol: **U_GetTMx**

Description: This function is the same as U_GetTM, except that the pointer to the pointer to EEPROM description block is fetched from the byte directly before the user main program.

Inputs: See U_GetTM

	Call Address	Outputs	Changed Registers
Mask 0010h	-		-
Mask 0011h	0D6CH	See U_GetTM	See U_GetTM
Mask 0012h	0D6CH	See U_GetTM	See U_GetTM
Mask 0020h	5063H	See U_GetTM	See U_GetTM
Mask 1013h	3D29H	See U_GetTM	See U_GetTM

Watchdog-time: (to be completed) µs

2.5.5 U_Delay

Symbol: **U_Delay** **Delay**

Description: waits the specified amount of time.
The delay is based upon the internal hardware-timer.

Inputs: A = delay time in 0.5 ms

Note: No delay times above 15 ms should be used.

	Call Address	Outputs	Changed Registers
Mask 0010h	0DDBH	None	A, X, RegB
Mask 0011h	0DFAH	None	A, X, RegB
Mask 0012h	0DFAH	None	A, X, RegB
Mask 0020h	5054H	None	A, X, RegB
Mask 1013h	3D02H	None	A, X, RegB

Watchdog-time: triggers watchdog internally

Th example below shows how a delay can be implemented using the user timers.

RAM-Data:

```
            UsrTmr0         DS          1
```

EEPROM-Data:

```
            UsrTmr          DB          LowByte(UsrTmr0)
                            DB          0               ;time base = 130ms
```

Code:

```
            ... other code ...
            ; start of timer
                            lda         #10
                            sta         RegE            ;time = 1.3 sec
                            lda         #0              ;user-timer 0
                            ldx         #LowByte(UsrTmr)    ;ptr to description
                            jsr         U_SetTM
            ... other code ...


        ;or other way to start a timer
        lda     #10
        sta     UsrTmr0

; check of timer status
        lda     #0      ;user-timer 0
        ldx     #LowByte(UsrTmr)    ;ptr to description
        jsr     U_GetTM
        beq     ...     ;branch if timer
;                        expired
... other code ...      Message handling functions
```

2.5.6 BCU 2 Timer System

For the interpretation of the following 3 API-functions U_TS_Set, U_TS_Del and U_TS_Seti, please refer to the system implementation descriptions in part 9/4 "BCU's and BIM's".

The BCU 2 has a maximum of 3 system timers (1-3) available for the user. If a serial protocol is active it will use timer 1. If the U_debounce function is used, it will use timer 2.

The following modes and scales are possible:

- Modes

TS_TYPE_EVT	020h	Event only used with TS_Seti fixed scale of 6.6ms generates an event in PEI-driver event queue.
TS_TYPE_MSG	030h	Message generates a timerindication message
TS_TYPE_MSG_CYC	040h	Messagecylic generates cyclic a timerindication message
TS_TYPE_DOWN	050h	Downcounter reserved for compatible function used with TM_Load
TS_TYPE_DIFF	060h	Differencecounter reserved for compatible function used with TM_Load

- Scales

TS_SCALE041MS	00h	0.416	ms
TS_SCALE6_6MS	02h	6.6	ms
TS_SCALE106S	04h	106	ms
TS_SCALE1_6S	06h	1,6	sec
TS_SCALE26S	08h	26	sec

- Cyclic Scales (only for type messagecyclic)

TS_100MS	04h	100	ms
TS_1SEC	08h	1	sec
TS_1MIN	0ch	1	min

2.5.6.1 U_TS_Set Set a BCU 2 Timer

Symbol: **U_TS_Set**

Description : Set an BCU 2 Timer

Inputs

X	Timernumber
A	Timermode
RegB	Timer Scale / Cyclic Scale
RegC	TimerValue (not for Mode TS_TYPE_MSG_CYC)
RegD	TimerParameter (only Mode TS_TYPE_MSG)

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	-	-	-
Mask 0012h	-	-	-
Mask 0020h	5090H	None	A, X, RegB-RegF
Mask 1013h	-	-	-

Watchdog-time: (to be completed) µs

2.5.6.2 U_TS_Del Delete BCU 2 Timer

Symbol: **U_TS_Del**

Description: Delete Timer

Inputs: X Timernumber

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	-	-	-
Mask 0012h	-	-	-
Mask 0020h	5093H	None	None
Mask 1013h	-	-	-

Watchdog-time: (to be completed) µs

2.5.6.3 U_TS_Seti Set Timer Interrupt Event

Symbol: **U_TS_Seti**

Description: Set BCU 2 Timer for Event Generation (only use in Interrupts)

Inputs:

X	Timernumber
iRegD	message queue ID or TS_TYPE_EVT
iRegC	Timervalue
A	Timer Parameter max. 6 bit

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	-	-	-
Mask 0012h	-	-	-
Mask 0020h	50ABH	None	None
Mask 1013h	-	-	-

Watchdog-time: (to be completed) µs

2.6 Message Handling Functions

2.6.1 AllocBuf Buffer Allocation

Symbol: **AllocBuf**

Description: allocates a message buffer.

Note: There is a distinction between long and short buffers. But non-system-software requires only long buffers.

Inputs: Carry: = buffer type
0 = short buffer
1 = long buffer

	Call Address	Outputs	Changed Registers
Mask 0010h	116AH	See below	A
Mask 0011h	117EH	See below	A
Mask 0012h	117EH	See below	A
Mask 0020h	5000H	See below	A
Mask 1013h	3D8FH	See below	A

Watchdog-time: (to be completed) µs

Outputs: X = pointer to buffer
Carry = 1 buffer allocated
0 no buffer allocated (all buffers in use) (X invalid)

2.6.2 FreeBuf Buffer Release

Symbol: **FreeBuf**

Description: releases a previously allocated buffer.

Inputs: X = pointer to buffer

	Call Address	Outputs	Changed Registers
Mask 0010h	118CH	None	A, X, RegB
Mask 0011h	11A0H	None	A, X, RegB
Mask 0012h	11A0H	None	A, X, RegB
Mask 0020h	5006H	None	A, X, RegB
Mask 1013h	3D8C	None	A, X, RegB

Watchdog-time: (to be completed) µs

2.6.3 PopBuf Message request

Symbol: **PopBuf**

Description: searches for a certain message type.

Inputs:

- Mask 0010h, 0011h, 0012h, 0020h:

  A = message and buffer type

Bit #	7	6	5	4	3	2	1	0
Meaning	buffer type 0 = short 1 = long	layer address			must be 0			

- Mask 1013h:

  A = message and buffer type

Bit #	7	6	5	4	3	2	1	0
Meaning	buffer type 0 = short 1 = long	layer address			requested service or 0			

	Call Address	Outputs	Changed Registers
Mask 0010h	11ACH	See below	A, RegB
Mask 0011h	11C0H	See below	A, RegB
Mask 0012h	11C0H	See below	A, RegB
Mask 0020h	5003H	See below	A, RegB
Mask 1013h	3D92H	See below	A, RegB

Watchdog-time: (to be completed) µs

Outputs: X = pointer to buffer
Carry = 1 = message found
0 = no such message (X invalid)

2.6.4 U_MS_Post Post Messages

Symbol: **U_MS_Post**

Description: Post the message pointed by X to the message queue A

Inputs:

X pointer to message

A message queue ID

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	-	-	-
Mask 0012h	-	-	-
Mask 0020h	508DH	None	A, X, RegB, RegL, RegM, RegN
Mask 1013h	-	-	-

Watchdog-time: (to be completed) µs

Example

```
        ...
jsr     PopBuf          ;get message from user queue
lda     #TASK_PM_ID     ;load queue/task id from PEI
jsr     U_MS_Post
...
```

2.6.5 U_MS_Switch Message redirection

Symbol: **U_MS_Switch**

Description: redirects messages from one task to an other ,all messages posted to queue ID X now posted to queue ID A.

Inputs:

X message queue ID to redirect

A new destination for queue X

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	-	-	-
Mask 0012h	-	-	-
Mask 0020h	508AH	None	A, X, RegC
Mask 1013h	-	-	-

Watchdog-time: (to be completed) µs

Example:

```
                ldx     #TASK_AL_ID   ;load queue/task id from applicationlayer
        lda     #TASK_US_ID   ;load queue/task id from user
        jsr     U_MS_Switch   ;now all messages for the applicationlayer
                              ;are send to the user
    .... receive messages
                .... handle messages for applicationlayer
```

```
        clrx
clra
jsr     U_MS_Switch   ;reset all redirections
```

2.7 Arithmetic Functions

2.7.1 multDE_FG Unsigned Integer Multiplication

Symbol: **multDE_FG**

Description: multiplies the unsigned integer values in the registers RegD:RegE and RegF:RegG.

Inputs: RegD:RegE = factor
RegF:RegG = factor

	Call Address	Outputs	Changed Registers
Mask 0010h	0B3CH	See below	A, X
Mask 0011h	0B4BH	See below	A, X
Mask 0012h	0B4BH	See below	A, X
Mask 0020h	5033H	See below	A, X
Mask 1013h	3D80H	See below	A, X

Watchdog-time: (to be completed) µs

Outputs: RegB:RegC = product
Carry = 0 ok
1 overflow

Note:

- Mask 1013h: In case of overflow the product is RegB:RegC = 0xFFFF.

2.7.2 divDE_BC Unsigned Integer Division

Symbol: **divDE_BC**

Description: divides the unsigned integer value in the registers RegD:RegE by the unsigned integer value in the registers RegB:RegC.

Inputs: RegD:RegE = dividend
RegB:RegC = divisor

	Call Address	Outputs	Changed Registers
Mask 0010h	0AFCH	See below	A, X, RegB, RegC
Mask 0011h	0B0BH	See below	A, X, RegB, RegC
Mask 0012h	0B0BH	See below	A, X, RegB, RegC
Mask 0020h	5030H	See below	A, X, RegB, RegC
Mask 1013h	3D83H	See below	A, X, RegB, RegC

Watchdog-time: (to be completed) µs

Outputs: RegF:RegG = quotient
RegD:RegE = remainder
Carry = 0 ok
1 divide by zero

2.7.3 FP_Flt2Int Float to Integer

Symbol: **FP_Flt2Int**

Description : Converts a 4 byte IEEE floatingpoint to2 byte integer

Inputs:

X	pointer to 32 bit float
RegF	offset to exponent

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	-	-	-
Mask 0012h	-	-	-
Mask 0020h	5087H	See below	A, X, RegD, RegE
Mask 1013h	-	-	-

Watchdog-time: (to be completed) µs

Outputs RegB:RegC 2 byte integer

2.7.4 FP_Int2Flt Integer to Float

Symbol: **FP_Int2Flt**

Description : Converts an 2 byte integer to 4 byte IEEE floatingpoint

Inputs:

RegB:RegC 2 byte integer

X pointer to 32 bit float

RegF offset to exponent

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	-	-	-
Mask 0012h	-	-	-
Mask 0020h	5084H	See below	A, X, RegD
Mask 1013h	-	-	-

Watchdog-time: (to be completed) µs

Outputs 4-Byte IEEE float at address X

2.8 Miscellaneous Functions

2.8.1 shlA*n* Accu shift left n

Symbol: **shlA*n***

Description: shifts the accu left by *n* bits. The values possible for *n* are 4, 5, 6, and 7. The new bits are set to zero.

Inputs: A = value

	Call Address				Outputs	Changed Registers
	shlA4	shlA5	shlA6	shlA7		
Mask 0010h	0B9AH	0B99H	0B98H	0B97H	A = shifted value	None
Mask 0011h	0BA9H	0BA8H	0BA7H	0BA6H	A = shifted value	None
Mask 0012h	0BA9H	0BA8H	0BA7H	0BA6H	A = shifted value	None
Mask 0020h	5018H	501BH	501EH	5021H	A = shifted value	None
Mask 1013h	3D59H	3D5CH	3D5FH	3D62H	A = shifted value	None

Watchdog-time: (to be completed) µs

2.8.2 shrA*n* Accu shift right n

Symbol: **shrA*n***

Description: shifts the accu right by *n* bits. The values possible for *n* are 4, 5, 6, and 7. The new bits are set to zero.

Inputs: A = value

	Call Address				Outputs	Changed Registers
	shrA4	shrA5	shrA6	shrA7		
Mask 0010h	0BDAH	0BD9H	0BD8H	0BD7H	A = shifted value	None
Mask 0011h	0BE9H	0BE8H	0BE7H	0BE6H	A = shifted value	None
Mask 0012h	0BE9H	0BE8H	0BE7H	0BE6H	A = shifted value	None
Mask 0020h	5024H	5027H	502AH	502DH	A = shifted value	None
Mask 1013h	3D4DH	3D50H	3D53H	3D56H	A = shifted value	None

Watchdog-time: (to be completed) µs

2.8.3 rolAn Accu rotate Left

Symbol: **rolA*n***

Description: rotates the accu left by *n* bits. The values possible for *n* are 1, 2, 3, 4 and 7.

Inputs: A = value

	Call Address					Outputs	Changed Registers
	rolA1	rolA2	rolA3	rolA4	rolA7		
Mask 0010h	-	-	-	-	-	-	-
Mask 0011h	-	-	-	-	-	-	-
Mask 0012h	0AF4H	0AF2H	0AF0H	0AEEH	0AECH	A = rotated value	None
Mask 0020h	5009H	500CH	500FH	5012H	5015H	A = rotated value	None
Mask 1013h	3D77H	3D74H	3D71H	3D6EH	3D6BH	A = rotated value	None

Watchdog-time: (to be completed) µs

2.8.4 U_SetBit Bit write

Symbol: **U_SetBit**

Description: sets the specified bit in register RegH.

Inputs: A = bit number
RegH = byte to modify
Carry = bit value to set

	Call Address	Outputs	Changed Registers
Mask 0010h	0DF9H	A = RegH = modified byte	RegB
Mask 0011h	0E18H	A = RegH = modified byte	RegB
Mask 0012h	0E18H	A = RegH = modified byte	RegB
Mask 0020h	5078H	A = RegH = modified byte	RegB
Mask 1013h	3D47H	A = RegH = modified byte	RegB

Watchdog-time: (to be completed) µs

2.8.5 U_GetBit

Symbol: **U_GetBit**

Description: reads the specified bit in register RegH.

Inputs: A = bit number
RegH = byte to be read from

	Call Address	Outputs	Changed Registers
Mask 0010h	0DEDH	See below	A, X, RegB
Mask 0011h	0E0CH	See below	A, X, RegB
Mask 0012h	0E0CH	See below	A, X, RegB
Mask 0020h	5075H	See below	A, X, RegB
Mask 1013h	3D44H	See below	A, RegB

Watchdog-time: (to be completed) µs

Outputs: Zero-flag = 0 if bit set
1 if bit clear

2.9 Loadable PEI-Support

2.9.1 U_FT12_Reset Reset FT 1.2 Protocol

Symbol: **U_FT12_Reset**

Description: Reset FT 1.2 protocol with specified baudrate

Inputs: A Baudrate

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	-	-	-
Mask 0012h	-	-	-
Mask 0020h	509CH	None	A, X, RegB-RegN
Mask 1013h	-	-	-

Watchdog-time: (to be completed) µs

2.9.2 U_FT12_GetStatus Get FT 1.2 Protocol Status

Symbol: **U_FT12_GetStatus**

Description: Gets the FT 1.2 Protocollstatus

Inputs: Carry 0 get status
1 force status request

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	-	-	-
Mask 0012h	-	-	-
Mask 0020h	509FH	See below	A
Mask 1013h	-	-	-

Watchdog-time: (to be completed) µs

Outputs: Carry = 0 no new State
Carry = 1 new state
A = 0 State OK
≠ 0 State not OK

2.9.3 U_SCI_Init Initialize Serial Communication Interface

Symbol: **U_SCI_Init**

Description: Initialize the SCI of the 68HC05BE12
(for Serial Asynchronous Communication)
(enable SCI-System , disable SPI-System , 8 databits, receiver enable, transmitter enable configure PCMUX, set Baudrate)

Inputs A Baudrate

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	-	-	-
Mask 0012h	-	-	-
Mask 0020h	50A2H	None	t.b.d.
Mask 1013h	-	-	-

Watchdog-time: (to be completed) µs

2.9.4 U_SPI_Init Initialize Serial Peripal Interface

Symbol: **U_SPI_Init**

Description: Initialize the SPI of the 68HC05BE12
(for Serial Synchronous Communication)
(enable SPI-System, disable SCI-System, configure PCMUX, set baudrate from EE_SyncRate, CPOL and CPHA from EE_ConfigDes)

Inputs None

	Call Address	Outputs	Changed Registers
Mask 0010h	-	-	-
Mask 0011h	-	-	-
Mask 0012h	-	-	-
Mask 0020h	50A5H	None	t.b.d.
Mask 1013h	-	-	-

Watchdog-time: (to be completed) µs