Sommario

DEVICE SERCOM

1. Introduction

The SERCOM device is a tool established in the CPU that allows to transmit and receive data via serial communication ports. The received data are stored in a buffer of adjustable size. Also receiving can be adjusted automatically by the application of the XON-XOFF protocol. To transmit data using fixed memory area formed by 64 byte. Since that receive and send buffers are different, you can create a shortcut both Half-Duplex and Full-Duplex links.
Using the capabilities of the low level of this device you can implement in your QCL code the highest level communication protocol allowing serial communication with different equipment (example: bar-code reader, linear position sensors with serial data transmission, small printer, ecc.).

1.1 Installation

1.1.1 Device declaration in the configuration file (.CNF)

Declaration of the used device (internal device): device name assigned, SERCOM, sampling time, serial port, size of receive buffer.

```
;-------------------------------
;Internal device declaration
;-------------------------------
INTDEVICE
...
<device_name>         SERCOM TCamp serial_port dim_buffer_rx
...
```

Where:

INTDEVICE	is a keyword indicating the beginning of the definition of internal devices,
<device_name	is the device name,
SERCOM	is the keyword that identifies the device described in this document,
TCamp	is the sampling time of the device,
serial_port	is the definition of the type of serial port connected (0 = Prog, 1 = User, the ID number of the other serial portsdepends on the hardware and firmware of the instrument used).
dim_buffer_rx	is the number of bytes to use for the receive circular buffer. If the number is omitted from the X character, or is less than the minimum of 48, It is set to the minimum value. The maximum value allowed is 32000.

The Declaration of the hardware used in the “BUS” section of the configuration unit you will have to refer to the firmware of the hardware itself.

```
Example
;-------------------------------
;Internal device declaration
;-------------------------------
INTDEVICE
...
DevSer       SERCOM         0001   0000   0100
...
```

1.2 Receive and send buffer

The receive and transmit buffers are distinguished, you can then make an Half-Duplex and Full-Duplex connections.

1.2.1 Receive buffer

The receive buffer is a memory FIFO type (First Input First Output). The device provides a parameter (ibyte1), in response to a command (RECEIVED), the first data received. There is also a parameter that contains the number of received data and in the buffer (nrx). With this informations it is easy to write a program in QCL which might “empty” the buffer as arriving data.
The buffer is “circular”: once you run out of space destined to receive subsequent data go to overwrite previously received (if you don't use the XON-XOFF). The latter event is reported in the err parameter.
The receive buffer size can be set in the configuration file at the time of the device declaration. It's important to correctly adjust the receive buffer to prevent the data loss. For example, if a instrument normally broadcasts a 100 characters string and the buffer has a capacity to store 50 characters, or stops the transmission of the string or you lose the data. If the transmitter instrument manages the XON-XOFF protocol the device interrupts its transmission by sending the XOFF character, reactivating the XON character when buffer space is freed.

1.2.2 Transmission buffer

The transmit buffer is fixed-size. It is composed from variables “*obyteNN*”, with NN from 1 to 64. In the case of the transmission

you can write every single byte in transmission. With a single command (*SEND*) you will be able to transfer a message consisting of a *ntx* number of byte (up to 64 max). If this is not sufficient, it's always possible to transmit multiple consecutive packets to complete the message. The use of a subsequent packets require the receiving system does not time-out may be turned on receiving between a character and the next (see section “Limitations”).

1.2.3 Flow control

The device uses a software flow control, composed from send/receive of two characters (*XOn* and *XOff*) enhance enable or temporarily block the transmission of the external device to the Qmove to allow the processing of data already received and emptying buffer. Flow control can be enabled as an automatic feature to avoid filling the receive buffer.
The threshold of *XOn* and *XOff* are respectively set by the parameters: *xonlim* and *xofflim*.

1.3 Commands and parameters table

1.3.1 Symbols used

The parameter name, condition or command is shown back to the left side of the table.

R
Indicates if its parameter or state is retentive (upon initialization of the device maintains the previously defined state), or the state assumes upon initialization of the device.
If the device does not require initialization the “R” field indicates the value that the parameter or state take to the power up of the card.
R = Retentive
0 = Upon initialization of the device the value is forced to zero.
1 = Upon initialization of the device the value is forced to one.
- = Upon initialization of the device is presented significant value.

D
Indicates the size of the parameter.
F = Flag
B = Byte
W = Word
L = Long
S = Single Float

1.3.1.1 Conditions

Describes all the conditions necessary so that the parameter is considered correct or because the command is accepted.
In some cases, limit values are specified for the acceptance of the parameter: if introduced any values outside the limits set, the data is however accepted; therefore appropriate controls of the application must be provided to ensure the proper functioning.
For the execution of a command, all conditions must be met; otherwise the command does not executed.

A
Indicates the access mode.
R = Read.
W = Write.
RW = Read / Write.

1.3.2 Commands

The available commands to manage the device are listed under the priority order descending. The device executes all commands received within the same sampling time starting from the one with the highest priority. For example if the device receives the same sampling time CLOSECOM and OPENCOM commands, first execute the OPENCOM command and than CLOSECOM command leaving the communication port closed.

Name	D	R	A	Conditions	Description
OPENCOM	/	/	/	st_opencom = 0	**Open Serial communication** Open the serial communication (the device then committed the communications port). The st_opencom state becomes to 1.
CLOSECOM	/	/	/	/	**Close Serial communication** Close t he serial communication (the device then not committed the communication port). The st_opencom state becomes to 0.

Name	D	R	A	Conditions	Description
SEND	/	/	/	ntx > 0 st_opencom = 1	**Send** Send the composite string in the transmission buffer for the number of characters set in the “ntx” parameter.
RECEIVED	/	/	/	ntx > 0 st_opencom = 1	**Received** If “ nrx ” > 0, draws the first character received and puts it in “ibyte1”, then 1 decrements the variable “nrx”.
FLUSH	/	/	/	st_opencom = 1	**FLush buffer** Execute a reset of the “nrx” variable, reset the “st_rxoff” and “st_txoff” states and reset the “ibyte1” variable and all the “obyte1-64” transmit buffer.
CLRWDATA	/	/	/	/	**Clear Warning Data** Reset the reporting of wdata parameter.
CLRWCMD	/	/	/	/	**Clear Warning Command** Reset the reporting of wcmd parameter.

1.3.3 Parameters

Name	D	R	A	Conditions	Description
mode	B	R	RW	-	**Mode** Defines the operating mode of the device as: **0** = Normal, without the flow control **1** = Software flow control (Xon - Xoff) **2** = A special mode to speed up the extraction of characters from the buffer. The command RECEIVED the characters received are presented in the obyte1÷64 parameters instead of the ibyte parameter. This new mode does not support the XO-XOFF (then resumes operation of mode 0). Valid range: 0 ÷ 2. N.B.: changing the operating mode,possible at any time, leads to reinitialize the device with consequent zeroing of the receive buffer
brate	L	R	RW	-	**Baud rate** Serial baud rate. Valid values: 4800, 9600, 19200, 38400, 57600.
datab	B	R	RW	-	**Data bit** Valid values: 7, 8
stopb	B	R	RW	-	**Stop bit** Valid values: 1, 2
par	B	R	RW	-	**Parity** (0 ÷ 2) **0** = none, **1** = even, **2** = odd.
xonlim	W	R	RW	-	**XOn threshold** It's the value that identifies the threshold of XOn. When the nrx parameter is less of this threshold is sent one XON character to the external unit. The default value is 16. You must set the xonlim less than xofflim. If this condition is not met, the default value is forced to 16.
xofflim	W	R	RW	-	**XOff threshold** It is the value that identifies the XOff threshold. When the nrx parameter is greater than this threshold is sent one XOFF character to the external connected unit. The defaul values is 16. You must set the xofflim greater than xonlim. If this condition is not met, the default value is forced equal to “buffer dimension” - 16.
ntx	W	0	RW	-	**Number Trasmitted Bytes** (1 ÷ 64) It is the number of characters to be sent with the SEND command.
nrx	W	0	R	-	**Number Received Bytes** (1 ÷ max.dim. buffer rx) Is the number of characters received from the serial buffer.
ibyte1	B	0	R	-	**Represents the receive buffer** After the RECEIVED command in this parameter is uploaded the first byte received retrieved from the receive buffer.
obyte1÷64	B	0	RW	-	**Represents the transmit buffer** Output byte n.1÷64. Are the 64 byte that composed the transmission buffer.
err	-	-	-	-	**Errors** Indicates if the errors occurred in the protocol. **0** = no errors **1** = overflow of receiving buffer; indicates that you have received more characters than the buffer capacity.
serr	-	-	-	-	**Serial Errors** Indicates if errors occurred in serial communication. The parameter is updated for each error encountered. The value persists until a subsequent error or to a writing on the same with the QCL. **0** = no errors **1** = parity error **2** = framing error **3** = overrun error
st_sended	F	0	R	-	**Sended** Activation indicates completion of the transmission of a message. The state is reset with the SEND command.

Name	D	R	A	Conditions	Description
st_rxoff	F	0	R	mode = 1	**XOff on receive** If mode = 1 indicates that the system is in XOff in the receiving because the number of data in the receive buffer has fallen below the threshold of control. Is reset automatically when the program sends the XOn character after the receive buffer has been “emptied” with the RECEIVED command until re-entry in the control threshold, or with a FLUSH command.
st_txoff	F	0	R	-	**Flow control XOff** If mode = 1 indicates that the system received the XOff character during the transmission. Resets automatically when it receives the XOn character or with FLUSH command.
wdata	F	0	R	-	**Warning Data** This bit indicates that an attempt was made to insert an invalid value in a parameter.
wcmd	F	0	R	-	**Warning Command** This bit indicates that it did not execute a command because they lack the necessary conditions
st_opencom	F	0	R	-	**Open communication port** Activation indicates that the device is working the serial communications port. To set this state use the OPENCOM command, for reset the CLOSECOM command

1.4 Limitations

The following is a list of some limitations on the use of SERCOM device:

- You cannot use the device with the systems that control the time-out between characters, if the length of the message to be transmitted exceeds the size of the transmit buffer; between the sending of the first 64 bytes, and the next, the QCL should intervene to rewrite the transmit buffer.
- The maximum speed is 57600 baud, the minimum is 4800;
- If the connection is made to systems that use flow controls other than XOn/XOff, you must implement the control itself using the QCL. If systems use a hardware flow control, they aren't linkable;
- If the connection is made towards systems in a daisy-chain where each component execute the echo of received characters, you must be attention to the propagation speed of the data because the echo function should be performed using the QCL.
- Not all combinations of values of DataBits, StopBits and ParityBit parameters are enable. If you use a combination of not enabled values, When you try to open the serial port using the OPENCOM command the device generates a warning command.

Below is a diagram of the combinations:

	DataBits	StopBits	ParityBit
Combinations May Not Work	7	1	0
	7	1	1
	7	1	2
	7	2	0
	7	2	1
	7	2	2
	8	2	0
	8	2	1
	8	2	2

1.5 Application example

The following are some examples of how to use the device.
In the examples the device will be DevSer.

1.5.1 Transmission of a string

Transmission of a control string in ASCII format to a instrument series DIN HB237.04A.

The string to be transmitted is: **{13QL1234561000@**

The code you have to write is as follows:

```
DevSer:mode = 0
...
address = 13
quota = 123456
...
DevSer:obyte1 = 123               ; character {
tmp = address
tmp1 = tmp / 10
DevSer:obyte2 = tmp1 + 48         ; character 1
```

```
tmp = tmp - (tmp1 * 10)
DevSer:obyte3 = tmp + 48          ; character 3
DevSer:obyte4 = 81                ; character Q
DevSer:obyte5 = 76                ; character L
tmp = quota
tmp1 = tmp / 100000
DevSer:obyte6 = tmp1 + 48         ; character 1
tmp = tmp - (tmp1 * 100000)
tmp1 = tmp / 10000
DevSer:obyte7 = tmp1 + 48         ; character 2
tmp = tmp - (tmp1 * 10000)
tmp1 = tmp / 1000
DevSer:obyte8 = tmp1 + 48         ; character 3
tmp = tmp - (tmp1 * 1000)
tmp1 = tmp / 100
DevSer:obyte9 = tmp1 + 48         ; character 4
tmp = tmp - (tmp1 * 100)
tmp1 = tmp / 10
DevSer:obyte10 = tmp1 + 48        ; character 5
tmp = tmp - (tmp1 * 10)
DevSer:obyte11 = tmp1 + 48        ; character 6
DevSer:obyte12 = 49               ; character 1
DevSer:obyte13 = 48               ; character 0
DevSer:obyte14 = 48               ; character 0
DevSer:obyte15 = 48               ; character 0
DevSer:obyte16 = 64               ; character @

DevSer:ntx = 16                   ; sets the send of 16 characters

SEND DevSer                       ; Send the string
WAIT DevSer:st_sended             ; and wait for end of transmission
```

1.5.2 Receiving a string

Receiving a string from an HB237.04A instrument in ASCII format and executing the echo of each character received.

The string to be received is: **[13RS123456123456@**

The code you have to write is:

```
DevSer:mode = 0
sequence = 0
...
WAIT DevSer:nrx                          ; wait character reception
RECEIVED DevSer
DevSer:obyte1 = DevSer:ibyte1            ; execution of local echo
DevSer:ntx = 1                           ; sets of 1 character sending
SEND DevSer                              ; Send the string
IF sequence EQ 0
  IF DevSer:ibyte1 EQ 91                 ; check character [
    address = 0
    sequence = 1
  ENDIF
ENDIF
IF sequence EQ 1
address = DevSer:ibyte1 * 10
sequence = 2
ENDIF
IF sequence EQ 2
  address = address + DevSer:ibyte1
  IF address EQ 13                     ; check address value
    sequence = 3
  ELSE
    sequence = 0
  ENDIF
ENDIF
...
WAIT DevSer:st_sended                    ; wait for end of echo transmission
```