

Blue Melon  
BlueStep  
BM1001 / BM1002  
Communcation protocol

Blue Melon v.o.f.

22nd February 2006

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.0.1	Contacting Blue Melon . . . . .	2
<b>2</b>	<b>Protocol specification</b>	<b>3</b>
2.1	SPI <sup>2</sup> C protocol . . . . .	3
2.1.1	Addressing . . . . .	3
2.1.2	Initialisatie . . . . .	4
2.1.3	Writing . . . . .	4
2.1.4	Reading . . . . .	5
2.2	X <sup>2</sup> C protocol . . . . .	5
2.2.1	Protocol . . . . .	6
2.2.2	Reading information . . . . .	7
2.2.3	Reading variables . . . . .	7
2.2.4	Updating variables . . . . .	10
2.2.5	Writing a variable . . . . .	10
2.2.6	Errors . . . . .	11
<b>A</b>	<b>Connector pin layout and dimensions</b>	<b>12</b>

# Chapter 1

## Introduction

This document tells the story about the communication protocol used between the BlueStep and the PC.

### 1.0.1 Contacting Blue Melon

If you still have questions or problems operating your BlueStep after reading the manual you can contact Blue Melon. Before contacting us please have a look at the Frequently Asked Questions section on our website "[www.bluemelon.org](http://www.bluemelon.org)".

[www.BlueMelon.org](http://www.BlueMelon.org)

e-mail: [Support@BlueMelon.nl](mailto:Support@BlueMelon.nl)

telnr: (+31)(0)594-213462

fax: (+31)(0)594-213674

Postal address: Postbus 11, 9843 ZG Grijpskerk, The Netherlands.

## Chapter 2

# Protocol specification

### 2.1 SPI<sup>2</sup>C protocol

The SPI<sup>2</sup>C protocol is a network layer on top of SPI, which resembles I<sup>2</sup>C , but which takes into account the limitations of micro systems.

A SPI<sup>2</sup>C system consists of a master and a slave. Which device is a master and which is a slave is defined by the SPI protocol. A SPI<sup>2</sup>C master therefore needs a SPI master and API, while a SPI<sup>2</sup>C slave needs a SPI slave and corresponding API.

#### 2.1.1 Addressing

A SPI<sup>2</sup>C connections supports addressing, this in contrast to the SPI protocol which supports point-to-point connections. Addressing allows a SPI<sup>2</sup>C slave to be implemented as router. The router can send its received data to one or more virtual or physical devices. In case of physical devices these are connected via SPI<sup>2</sup>C or I<sup>2</sup>C to the slave. In some circumstances a device controller thus acts as master and slave at the same time, however this always involves two different physical connections.

Table 2.1: Address bits

A6	A5	A4	A3	A2	A1	A0	R/W
----	----	----	----	----	----	----	-----

The address consists of a byte (table 2.1). The highest seven bits represent the addresses 0 to 127. The lowest bit is a R/W flag. If this bit is high de connection is in write mode, if it is low the connection is in read mode.

### 2.1.2 Initialisatie

A SPI<sup>2</sup>C transmission is always initiated by the master. The initialization is put into effect by sending the start byte and an address byte, refer to table 2.2. A connected slave should respond with a reply consisting of an ACK<sup>1</sup> or NAK<sup>2</sup> byte. In case of a NAK the transmission is ended. In case of an ACK the data connection will be put into write or read mode depending on the R/W bit in the address byte.

Table 2.2: Initialization

Byte	Master	Slave
0	START	
1	address	
2		ACK or NAK

### 2.1.3 Writing

When the initialization is ended by an ACK<sup>3</sup> and the R/W flag in the address represents write mode, the master will have to send a message containing a length byte and several data bytes (table 2.3). The length of a message is represented by a byte, this allows message lengths of up to 255 bytes. As a special case the length field can be zero, which results in a message without data bytes. These zero-length messages can be used to scan the bus for connected devices. When a complete message has been transmitted the slave will respond with an ACK<sup>4</sup> or NAK<sup>5</sup> byte. If a NAK was received the message should be considered not sent.

Table 2.3: Write cycle

Byte	Master	Slave
3	length	
4...4+L	data[n]	
5+L		ACK or NAK

---

<sup>1</sup>The ACK byte is valued at 0xAA hex or 170 decimal.

<sup>2</sup>The NAK byte is valued at 0xEE hex or 238 decimal.

<sup>3</sup>The ACK byte is valued at 0xAA hex or 170 decimal.

<sup>4</sup>The ACK byte is valued at 0xAA hex or 170 decimal.

<sup>5</sup>The NAK byte is valued at 0xEE hex or 238 decimal.

### 2.1.4 Reading

When the initialization is ended by an ACK<sup>6</sup> and the R/W flag in the address represents read mode, the master will have to send a message containing a length field containing the amount of data bytes which it wants to receive. A length field is represented by a byte, which constraints the size of data packet to 255 bytes. As a special case the length field can be zero, in this case no data bytes are expected as a reply. These zero-length messages can be used to scan the bus for connected devices. When a slave receives such a zero-length message it should not respond by sending data bytes. A slave will pad a message with zeros when it has not enough data to send. After a complete message has been transmitted the slave will have to correspond with an ACK<sup>7</sup> or NAK<sup>8</sup> byte. In case of a NAK<sup>9</sup> the controller should consider the data as not received. A NAK cannot occur when there was not enough data to send to the master.

Table 2.4: Read cycle

Byte	Master	Slave
3	length	
4...4+L		data[n]
5+L		ACK or NAK

## 2.2 X<sup>2</sup>C protocol

The X<sup>2</sup>C protocol was developed by Blue Melon to facilitate the exchange of data represented by variables from a micro system to a controlling system. The X<sup>2</sup>C protocol is supported over X<sup>2</sup>C and SPI<sup>2</sup>C connections. See appendix 2.1 for information on SPI<sup>2</sup>C .

Via X<sup>2</sup>C one can obtain information of a certain device. This information consists of data (values) and a description of the data. More specifically the information consists of values of variables and a description of the variables (eg. name, minimum value , maximum value, etc.). Furthermore X<sup>2</sup>C can be used to set the variables to a given value.

A X<sup>2</sup>C system always consists of a master and one or more slaves. Although

---

<sup>6</sup>The ACK byte is valued at 0xAA hex or 170 decimal.

<sup>7</sup>The ACK byte is valued at 0xAA hex or 170 decimal.

<sup>8</sup>The NAK byte is valued at 0xEE hex or 238 decimal.

<sup>9</sup>The NAK byte is valued at 0xEE hex or 238 decimal.

multiple masters are supported by I<sup>2</sup>C , these are currently not supported in X<sup>2</sup>C .

### 2.2.1 Protocol

The packets which are sent and received in the X<sup>2</sup>C protocol have a similar format. The master always start with a command. Depending on this command the slave will respond with an answer. The master can choose to ignore the answer. When a command arrives at a slave which has an answer ready, this answer will be overwritten by the answer on the last received command. The protocol uses Intel byte order, which means that in case of a word (int16) or dword (int32) the LSB is ordered before the MSB.

Table 2.5: Command

COMMAND byte	length byte	data[n] bytes...
--------------	-------------	------------------

A X<sup>2</sup>C command (table 2.5) consists of a COMMAND byte (commands listed in table 2.6) and a length byte and data bytes. The value of the length byte determines the number of following data bytes. The transmission of a command may be terminated at any time.

Table 2.6: COMMAND bytes

Symbol	Hex value	appendix
X2C_UPDATE_VAR	0x11	<a href="#">2.2.3</a>
X2C_GET_VAR	0x22	<a href="#">2.2.3</a>
X2C_SET_VAR	0x33	<a href="#">2.2.5</a>
X2C_GET_INFO	0x44	<a href="#">2.2.2</a>
X2C_ERROR	0xEE	<a href="#">2.2.6</a>

A X<sup>2</sup>C answer (table 2.7), consists of a COMMAND byte (commands listed in table 2.6) followed by a number of data blocks. The number of data blocks and their format depends on the COMMAND byte value. In any case the command byte in the answer will correspond to the COMMAND byte value of the previously sent command.

Table 2.7: Answer

0xAA	Length	COMMAND byte	block[0]	blocks[1]	...	csum[0]	csum[1]
------	--------	--------------	----------	-----------	-----	---------	---------

## 2.2.2 Reading information

To readout information of a device the master will have to send a X2C\_GET\_INFO command (table 2.8). Subsequently the master can receive the answer from the slave. The slave will return an answer consisting of its name, version and number of devices variables (table 2.9). The name and version field are formatted as strings. Each string consists of its length followed by character data. The total length of an answer is given by  $2+(namelength)+(versionlength)$ .

Table 2.8: Info read command

0x44	0x00
------	------

Table 2.9: Info read answer

0x44	
(int8) name length	(string) name
(int8) version length	(string) version
(int8) variable count	

## 2.2.3 Reading variables

To retrieve a variable from a device the master can send a X2C\_GET\_VAR command (table 2.10). The command has a length of 1 byte, its data consists of the index of the to be retrieved variable. The answer (table 2.11) contains a command byte and the index of the variable which correspond to the similar bytes of the command. Furthermore the answer contains the name of the variable. The name is formatted as a string. In general a string consists of its length followed by character data. Finally the answer contains a number of flags in the form of a bitmask (table 2.13) and a description of the value type of the variable (table 2.12). In variables which have an 'Enumeration' type this enumeration is received subsequently. In variables of other scalar types the maximum, minimum and metric will be received. A variable of type 'Enumeration' cannot have an assigned metric. Refer to table 2.14 and table 2.15 for information on enumerations and scalar values.

Table 2.10: Read variable command

0x22	0x01	variable index
------	------	----------------

Table 2.11: Read variabele answer

0x22	variable index
(int8) name lengte	(string) name
(int8) flag	(int8) type
enumeration or scalar, refer to tabel <a href="#">2.14</a> , <a href="#">2.15</a> or ?? respectively.	

Table 2.12: Types

Type	Hex valye
unset	0x00
unsigned byte (uint8)	0x01
signed byte (int8)	0x02
unsigned word (uint16)	0x03
signed word (int16)	0x04
unsigned dword (uint32)	0x05
signed dword (int32)	0x06
single (4 byte float)	0x07
boolean (uint8)	0x08
enumeration (uint8)	0x09

Table 2.13: Flags (mask) (The changed flag is used internally.)

Flags	Hex mask
read Only	0x01
hidden	0x02
back-up	0x04
dangerous	0x08
real-time	0x10
changed	0x80

Table 2.14: Enumeration data block

(int8) index
(int8) element count
(int8) element[n] value
(int8) element[n] name length
(string) element[n] name

Table 2.15: Scalar data block

(int32) value
(int32) maximum
(int32) minimum
(int8) metric mask, refer to table <a href="#">2.16</a>

Table 2.16: Metric

Dimension	Hex mask 0xE0
milli (m)	0x40
micro (u)	0x80
kilo (k)	0xC0
Metric	Hex mask 0x1F
ampère (A)	0x01
volt (V)	0x02
celsius (oC)	0x03
steps (p)	0x04
steps/seconde (p/s)	0x05

## 2.2.4 Updating variables

To retrieve an update value of a certain variable a `X2C_UPDATE_VAR` command can be used (table 2.17). The data block of this command is 1 byte long and consist of the index of the variable of which the value is to be retrieved. The answer consists of a `COMMAND` byte, the index of the variable and the value of this variable (table 2.18). Depending on the type of the variable the value of the variable consists of a byte in a variable of type enumeration or 4 bytes which represent a scalar value. In case of an error an error value will be returned (see appendix 2.2.6).

Table 2.17: Variable update command

0x11	0x01	variable index
------	------	----------------

Table 2.18: Variable update answer

0x11	variable index
(int8) index in enumeration	
OR	
(int32) scalar value	

## 2.2.5 Writing a variable

To write a new value to a variable the `X2C_SET_VAR` command is used (table 2.19). The data block consists of five bytes and contains both index

of the variable (one byte) and value of the variable (four bytes). If the variable is of the enumeration type the first byte of the value will contain the index in the enumeration, the other three bytes are zero. The answer will contain nothing more than the COMMAND byte (table 2.20) or an error value (see chapter 2.2.6).

Table 2.19: Variable write command

0x33	0x05	variable index
(int8) enumeration index	(int24) 0	
OR		
(int32) scalar value		

Table 2.20: Variabele write answer

0x33
------

## 2.2.6 Errors

When the master sends an incorrect command or an illegal number of arguments the slave will answer with an error message X2C\_ERROR (table 2.21).

Table 2.21: Error message

0xEE	(uint8) message length
(string) message	

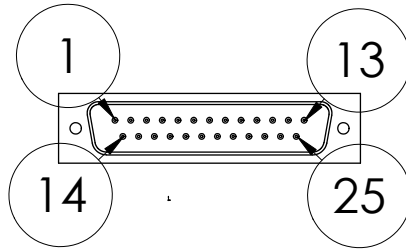


Figure A.1: Parallel port connector.

## Appendix A

# Connector pin layout and dimensions

---

<sup>1</sup>Pin 2 to Pin 7 are depicted in the default configuration. The connections can be rewired by using the BlueStep configuration tool.

<sup>2</sup>Pin 8 or 9 can be used to turn the motors on and off. The use of these pins is set up with the Blue Step configuration tool.

<sup>3</sup>Pin 1 and 14 are used by the BlueStep configuration software. Configure your CNC application to keep these pins low.

<sup>4</sup>Switch 4 is meant to be used as an emergency stop switch.

Pin 1	Reset <sup>3</sup>	Pin 14	Setup <sup>3</sup>
Pin 2	Motor 1 direction <sup>1</sup>	Pin 15	Not connected
Pin 3	Motor 1 step <sup>1</sup>	Pin 16	Relay 2
Pin 4	Motor 2 direction <sup>1</sup>	Pin 17	Relay 1
Pin 5	Motor 2 step <sup>1</sup>	Pin 18	Ground
Pin 6	Motor 3 direction <sup>1</sup>	Pin 19	Ground
Pin 7	Motor 3 step <sup>1</sup>	Pin 20	Ground
Pin 8	MOSI <sup>2</sup>	Pin 21	Ground
Pin 9	SCK <sup>2</sup>	Pin 22	Ground
Pin 10	Switch 3	Pin 23	Ground
Pin 11	Switch 2	Pin 24	Ground
Pin 12	Switch 4 <sup>4</sup>	Pin 25	Ground
Pin 13	Switch 1		

Table A.1: Pin layout Parallel port connector, refer to figure [A.1](#).