TF1
MEMS Tunable Optical Filter

Revision 3.8

May 2021

TABLE OF CONTENTS

1 In This Manual

This manual describes the optical, electrical and mechanical specifications of the TF MEMS Tunable Filter. It additionally details the communication protocol required to drive the device.

2 Description

sercalo's Optical Tunable Filter is based on MEMS technology and is designed for ITU O, C and L band with 50 GHz or 100 GHz channel spacing. The highly reliable tuning mechanism uses an integrated micro-mirror with switching time below 50 ms and insertion loss below 4 dB. The component is compliant to Telcordia 1221 reliability standards and RoHS requirements 2015/863/EU.

sercalo's Tunable Filter is composed of an optical system and an electrical driver interface (see Figure 1). The device can be independently controlled by a UART interface with TTL/CMOS voltage levels or an SMBus/I^2C interface.

The device operating principle is depicted in Figure 2. Light from the input fiber is collimated on a fused silica grating. The grating diffracts the light with a distinct angle for each wavelength. Light is then reflected by a MEMS mirror onto the output collimator which only couples a small fraction of it into the output fiber. By modifying the MEMS mirror tilt angle user can choose the central wavelength of the filter.

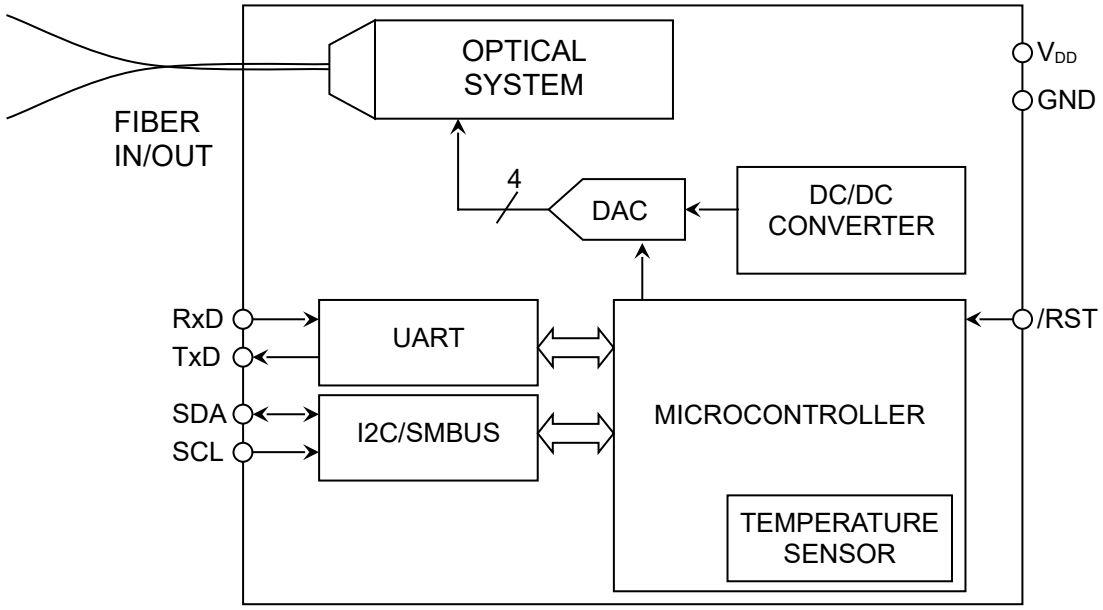

Figure 1 – Functional block diagram

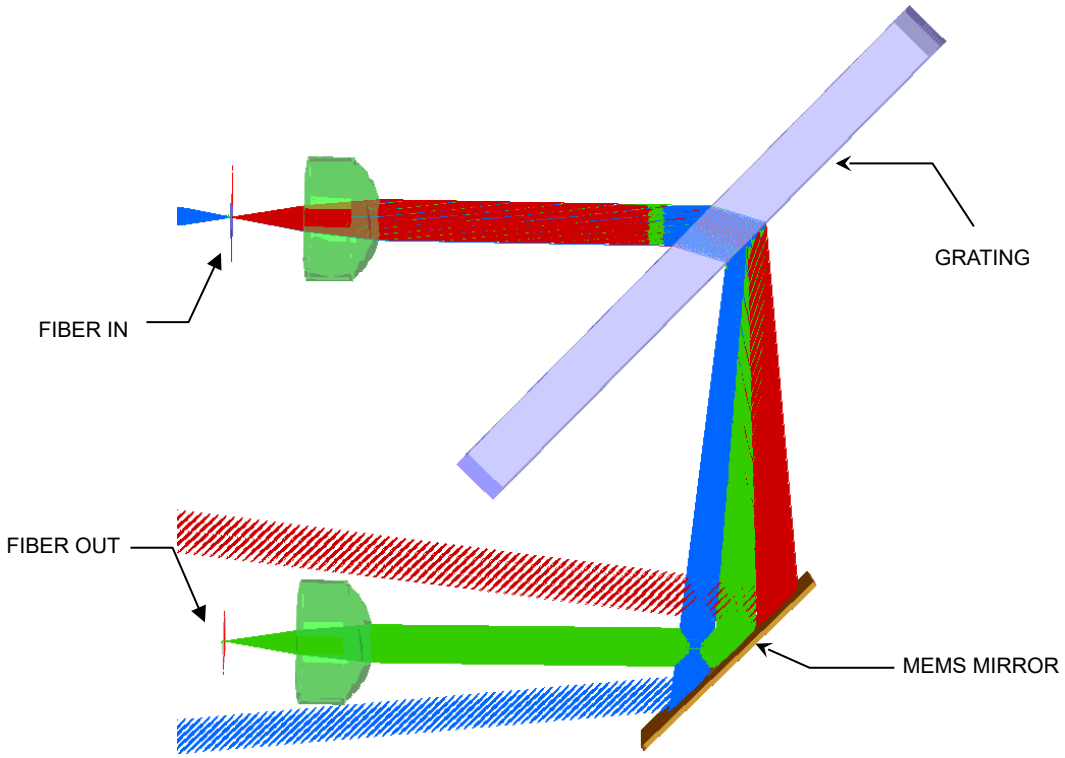

Figure 2 – Tunable filter operating principle

3 Connector Pin Assignments

The pin assignments for the connector is detailed in Table 1.

Pin	Name	Description
1	GND	Ground
2	VDD	Supply voltage
3	-	Reserved
4	TX	UART TX
5	-	Reserved
6	RX	UART RX
7	$\overline{\text{RST}}$	Active-low system reset
8	SDA	SMBus/I²C Serial Data
9	SCL	SMBus/I²C Serial Clock
10	GND	Ground

Table 1 – Connector pin assignments

Please note the following points:

➔ **UART TX** signal carries data *from* the product, while **UART RX** signal carries data *to* the product.

- ➔ The UART interface uses TTL/CMOS voltage levels. A voltage translator is required in order to communicate with the driver board using an RS-232 interface (see Chapter 5.2).
- ➔ **System reset**, **UART RX/TX** and **SMBus/I²C SDA/SCL** signals are connected to internal high impedance pull-up resistors. Unused pins can be let unconnected.
- ➔ Let reserved pins unconnected.

4 Product Dimensions

Figure 3 depicts the devices layouts, view from pin side. All dimensions are shown in millimeters.

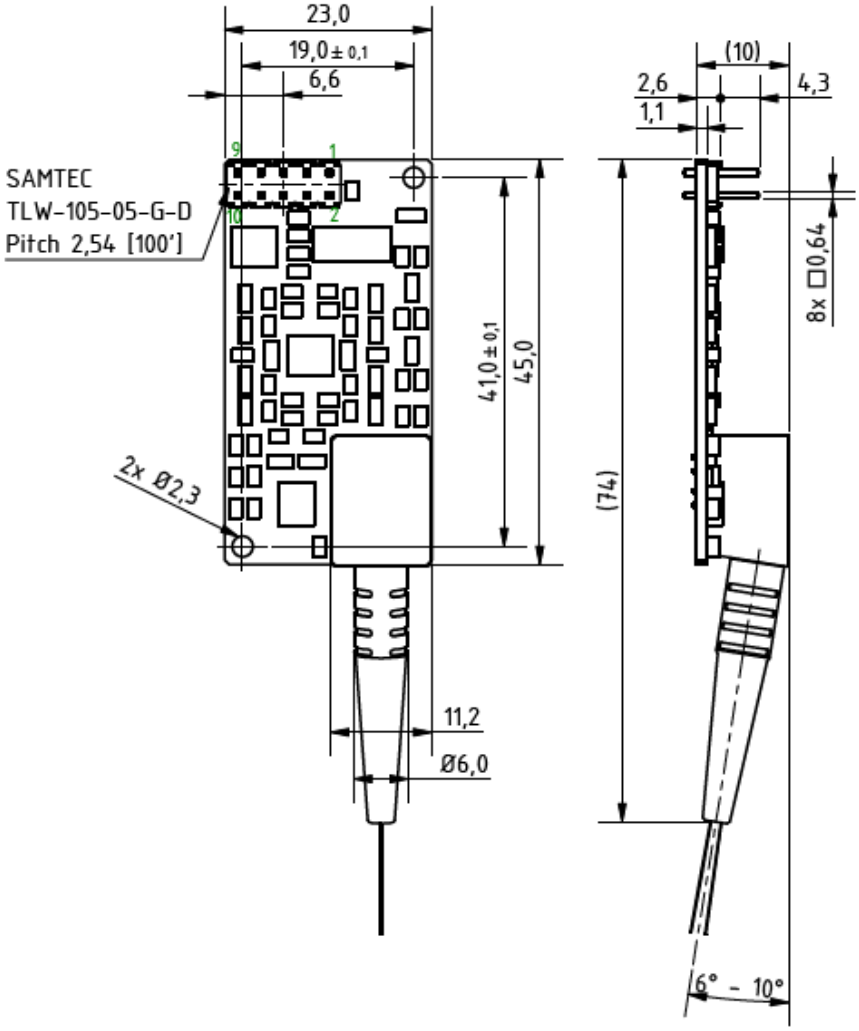

Figure 3 – Device layout

5 Using the UART Interface

The device is equipped with a Universal Asynchronous Receiver-Transmitter (UART) with TTL/CMOS-compatible voltage levels. UART TX signal carries data *from* the product, while UART RX signal carries data *to* the product.

Table 2 lists the requirements for the UART settings. The default baud rate (after power on or reset) is 9600 baud. Baud rate remains in effect otherwise changed by the user.

Parameter	Factory setting
Transmission rate [bps]	9600
Data bits	8
Parity	none
Stop bits	1
Flow control	none

Table 2 – RS-232 settings after power-on or reset

5.1 Command syntax

Commands consist of ASCII characters strings having the following structure:

`CMD <param1> <…> [opt_param1] […]`↵

where:

- **`CMD`** is the command
- **`<…>`** angle brackets indicate that the enclosed parameter is *mandatory*
- **`[…]`** square brackets indicate that the enclosed parameter is *optional*
- ↵ is the string terminator (end of line).

The following conventions are adopted:

- Commands can be written either in upper-case or lower-case characters.
- Commands and parameters are separated by one or more spaces (ASCII 0x20).
- The system recognizes as end of line any of the forms: LF (Line feed, '\n', ASCII 0x0A) or CR (Carriage return, '\r', ASCII 0x0D), or CR followed by LF (CR+LF, 0x0D 0x0A). Replies always end with the sequence CR+LF.
- The device always replies to commands. The reply can be a command-dependent acknowledgment or an error message.
- Error messages always start with ERR, followed by a space and context-dependent additional data.
- The user should not attempt to send the device a new command until it has completed the current command.

5.2 Connect the UART interface to a PC

The UART interface can be directly connected to microcontrollers and any other electronic device with TTL/CMOS-compatible voltage levels, i.e. logic high ('1') is represented by V_{DD}, while a logic low ('0') is 0V. This is not the case for serial ports on personal computers whose voltage levels complies with the RS-232 standard. By the RS-232 standard a logic high ('1') is

represented by a negative voltage (from -3 to -25V) while a logic low ('0') transmits a positive voltage (from +3 to +25V)[1].

One solution is to use the MAX232[2] integrated circuit or one of the many derivatives to convert signals from a RS-232 serial port to TTL signals. The other way is to use a USB-UART bridge based on CP2102 or FT232R integrated circuits which directly converts USB to UART. They are easily available on the market (some suppliers are listed on Table 3).

Supplier	Part Number
DigiKey	768-1017-ND
Mouser	895-TTL-232R-5V-PCB
eXtreme Electronics	SKU00201
Sparkfun	FTDI Basic Breakout - 5V
Adafruit	954

Table 3 – Example of USB-to-UART bridge suppliers

If a Windows-based PC and HyperTerminal are used, the suggested configuration is displayed in Figure 4.

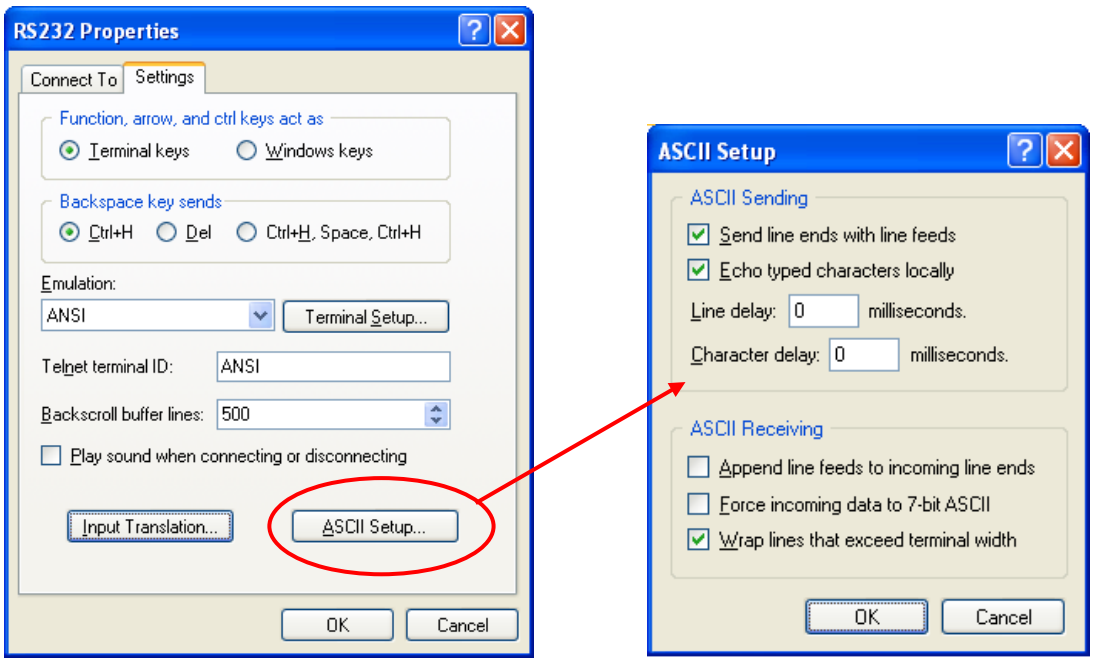

Figure 4 – HyperTerminal RS-232 properties and ASCII setup.

6 Using the SMBus/I²C Interface

The device is equipped with an SMBus serial I/O peripheral that is compliant with both the System Management Bus Specification and the I²C-Bus Specification. SMBus is a trademark of Intel; I²C is a trademark of Phillips Semiconductor.

[1] https://en.wikipedia.org/wiki/RS-232#Voltage_levels

[2] https://en.wikipedia.org/wiki/MAX232

SDA and SCL I/O pins are compatible with power supplies as low as 3.3V and as high as 5.0V. External pull-up resistors are required for full-speed operation.

The device is provided with a factory-programmed address of 0xFE. User can modify the default address using the dedicated command (see Chapter 9.8 for details). The new address is stored in the internal flash memory and is preserved during reset and power-off.

Each transmission is terminated with a Packet Error Code (PEC) byte, compliant to SMBus specification version 2.0 and 1.1. The byte is calculated over the entire message including the address and read/write bit (see Chapter 6.5 for details).

Please note that the device, after a successful write sequence, delay any further transmission by holding SCK low until the command is executed.

Parameters of type **signed short**, **unsigned short**, and **char** are transmitted using one byte (8 bits).

Parameters of type **signed int** and **unsigned int** are transmitted using two bytes (16 bits). The high-order byte (most significant bit, MSB) is transmitted first, followed by the low-order byte (least significant bit, LSB).

Parameters of type **float** are transmitted using four bytes (32-bits). The format used follows the IEEE-754 standard. The high-order byte (MSB) is transmitted first.

6.1 Send commands to device

A write sequence consists of the following elements (see Figure 5):

1. A *start* condition: it consists of a falling edge on SDA while SCL is high.
2. The *address byte*, i.e. 7 bits corresponding to the device address and the R/W bit set as “write” (the R/W bit is set to logic 1 to indicate a “read” operation and cleared to logic 0 to indicate a “write” operation).
3. A *command code*: refer to Chapter 9 for a list of available command codes.
4. The *parameter(s) length*, i.e. the number *n* of bytes composing the following optional parameters. This field is mandatory: if no parameters are required, set it to 0x00.
5. *Optional parameters*: refer to Chapter 9 for a detailed explanation.
6. *CRC-8.*
7. A *stop* condition: it consists of a rising edge on SDA while SCL is high.

The device replies to each received byte with an “ACK” as a confirmation. An ACK consists of a low level on SDA sampled when SCL is high. If the ACK signal is not received, master must abort the transfer.

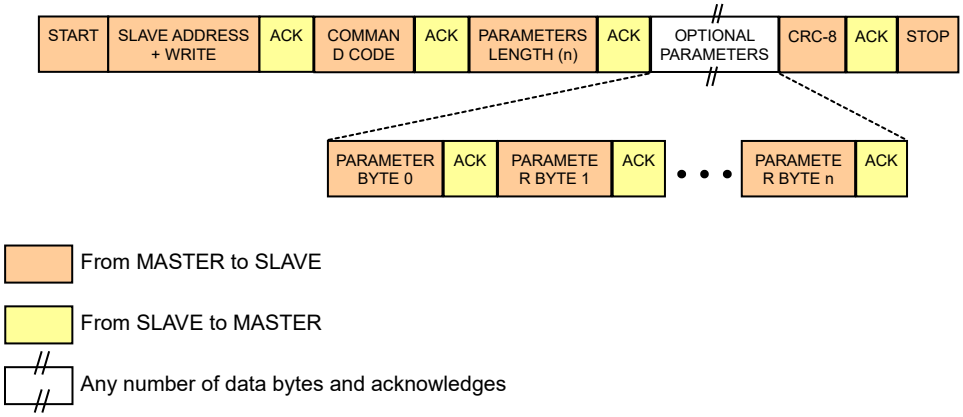

Figure 5 – Typical successful write (master to slave)

6.2 Receive data from device

After a successful transmission, as detailed in Paragraph 6.1, device always has a reply that can be optionally read by master. A read sequence for a successfully executed command consists of the following elements (see Figure 6):

1. A *start* condition: it consists of a falling edge on SDA while SCL is high.
2. The *address byte*, i.e. 7 bits corresponding to the device address and the R/W bit set as “read” (the R/W bit is set to logic 1 to indicate a “read” operation and cleared to logic 0 to indicate a “write” operation).
3. A *command code*: it is the last executed command, which this reply refers.
4. The *parameter(s) length*, i.e. the number *n* of bytes composing the following optional parameters. If no parameters will follow, this field is set to 0x00.
5. *Optional parameters*: refer to Chapter 9 for a detailed explanation.
6. *CRC-8*.
7. A *stop* condition: it consists of a rising edge on SDA while SCL is high.

The master must replies to each received byte with an “ACK” as a confirmation. An ACK consists of a low level on SDA sampled when SCL is high. Only the last byte must be confirmed with a “NACK”. A NACK consists of a high level on SDA sampled when SCL is high.

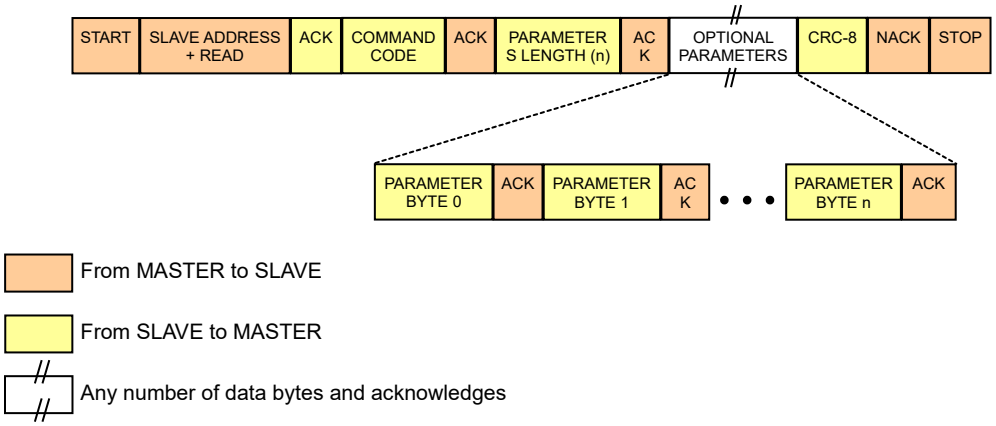

Figure 6 – Typical successful read (slave to master)

6.3 Receive an error from device

If the previous command returns an error, it can be optionally read and detect by the master. A read sequence for a command returning an error consists in the following elements:

1. A *start* condition: it consists of a falling edge on SDA while SCL is high.
2. The *address byte*, i.e. 7 bits corresponding to the device address and the R/W bit set as “read” (the R/W bit is set to logic 1 to indicate a “read” operation and cleared to logic 0 to indicate a “write” operation).
3. A *command code* plus the value 0x80.
4. The *error number*: see Chapter 10 for a list of error codes.
5. *CRC-8*.
6. A *stop* condition: it consists of a rising edge on SDA while SCL is high.

The master must replies to each received byte with an “ACK” as a confirmation. An ACK consists of a low level on SDA sampled when SCL is high. Only the last byte must be confirmed with a “NACK”. A NACK consists of a high level on SDA sampled when SCL is high.

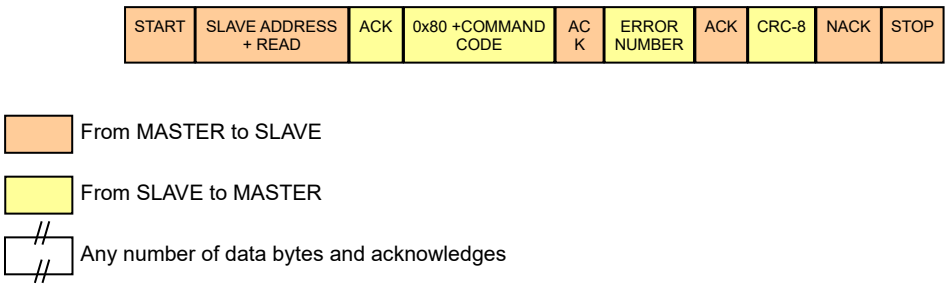

Figure 7 – Device returns an error (slave to master)

6.4 Hold master communication sequence

If the device is addressed during the execution of a command, it pulls down the SCL line in order to force the master into a wait state. By releasing the SCL line, the device indicates that

internal processing is terminated and that transmission may be continued. The maximum duration for command execution can vary depending on the command.

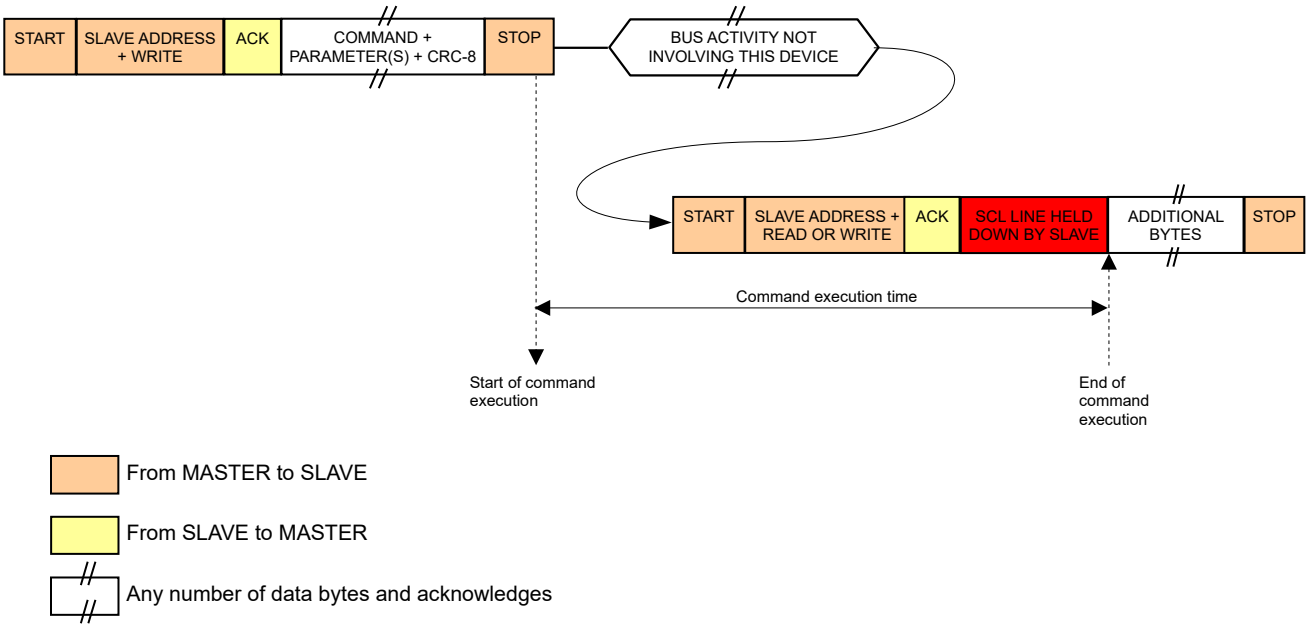

Figure 8 – Communication sequence with SCL line held down until command is executed

6.5 Example of CRC-8 generation

The SMBus/I²C uses a PEC (packet error code) compliant to SMBus specification version 1.1 and 2.0. The polynomial used is x^8+x^2+x+1 i.e. the CRC-8-ATM HEC algorithm, initialized to zero. An example of a C function that performs the calculation is presented below.

```
unsigned char crc8(unsigned char MsgLen, const unsigned char *Msg){
/* Lookup table for CRC codes
// CRC-8, poly = x^8 + x^2 + x^1 + x^0, init = 0 */

static unsigned char crc8_table[256] = {
   0x00, 0x07, 0x0E, 0x09, 0x1C, 0x1B, 0x12, 0x15, 0x38, 0x3F, 0x36, 0x31,
   0x24, 0x23, 0x2A, 0x2D, 0x70, 0x77, 0x7E, 0x79, 0x6C, 0x6B, 0x62, 0x65,
   0x48, 0x4F, 0x46, 0x41, 0x54, 0x53, 0x5A, 0x5D, 0xE0, 0xE7, 0xEE, 0xE9,
   0xFC, 0xFB, 0xF2, 0xF5, 0xD8, 0xDF, 0xD6, 0xD1, 0xC4, 0xC3, 0xCA, 0xCD,
   0x90, 0x97, 0x9E, 0x99, 0x8C, 0x8B, 0x82, 0x85, 0xA8, 0xAF, 0xA6, 0xA1,
   0xB4, 0xB3, 0xBA, 0xBD, 0xC7, 0xC0, 0xC9, 0xCE, 0xDB, 0xDC, 0xD5, 0xD2,
   0xFF, 0xF8, 0xF1, 0xF6, 0xE3, 0xE4, 0xED, 0xEA, 0xB7, 0xB0, 0xB9, 0xBE,
   0xAB, 0xAC, 0xA5, 0xA2, 0x8F, 0x88, 0x81, 0x86, 0x93, 0x94, 0x9D, 0x9A,
   0x27, 0x20, 0x29, 0x2E, 0x3B, 0x3C, 0x35, 0x32, 0x1F, 0x18, 0x11, 0x16,
   0x03, 0x04, 0x0D, 0x0A, 0x57, 0x50, 0x59, 0x5E, 0x4B, 0x4C, 0x45, 0x42,
   0x6F, 0x68, 0x61, 0x66, 0x73, 0x74, 0x7D, 0x7A, 0x89, 0x8E, 0x87, 0x80,
   0x95, 0x92, 0x9B, 0x9C, 0xB1, 0xB6, 0xBF, 0xB8, 0xAD, 0xAA, 0xA3, 0xA4,
   0xF9, 0xFE, 0xF7, 0xF0, 0xE5, 0xE2, 0xEB, 0xEC, 0xC1, 0xC6, 0xCF, 0xC8,
   0xDD, 0xDA, 0xD3, 0xD4, 0x69, 0x6E, 0x67, 0x60, 0x75, 0x72, 0x7B, 0x7C,
   0x51, 0x56, 0x5F, 0x58, 0x4D, 0x4A, 0x43, 0x44, 0x19, 0x1E, 0x17, 0x10,
   0x05, 0x02, 0x0B, 0x0C, 0x21, 0x26, 0x2F, 0x28, 0x3D, 0x3A, 0x33, 0x34,
   0x4E, 0x49, 0x40, 0x47, 0x52, 0x55, 0x5C, 0x5B, 0x76, 0x71, 0x78, 0x7F,
   0x6A, 0x6D, 0x64, 0x63, 0x3E, 0x39, 0x30, 0x37, 0x22, 0x25, 0x2C, 0x2B,
   0x06, 0x01, 0x08, 0x0F, 0x1A, 0x1D, 0x14, 0x13, 0xAE, 0xA9, 0xA0, 0xA7,
   0xB2, 0xB5, 0xBC, 0xBB, 0x96, 0x91, 0x98, 0x9F, 0x8A, 0x8D, 0x84, 0x83,
   0xDE, 0xD9, 0xD0, 0xD7, 0xC2, 0xC5, 0xCC, 0xCB, 0xE6, 0xE1, 0xE8, 0xEF,
```

```
    0xFA, 0xFD, 0xF4, 0xF3 };

register unsigned char crcVal = 0x00;

while (MsgLen-- > 0) {
   crcVal = crc8_table[crcVal ^ *Msg++];
   }

return crcVal;
}
```

7 Power Management Modes

The device has a low power mode that permits saving current, e.g. in battery-supplied applications. User can switch between normal and low power mode using the dedicated command (see Chapter 7 for additional details).

In low power mode, the DC/DC converter and the DAC are shut down, while UART and SMBus/I^{2}C interfaces are let active. The current consumption is reduced, but the Tunable Filter cannot be operated and the optical path is open. The device enters in low power mode after each power on or reset and remains in this status until changed by the user.

8 System reset

A system reset is performed:

- By asserting the system reset ($\overline{\text{RST}}$) signal (see Table 1)
- By using the reset command (see Paragraph 9.2)
- After any powering on

The reset brings the device to its initial state:

- The device enters in low-power mode (see Paragraph 7)
- The receive and transmit buffers of the UART and SMBus/I2C interfaces are flushed and any pending error is cleared

9 Command Reference

The available commands are listed in Table 4 and detailed in the following subsections. Use binary command formatting for SMBus/I²C bus, ASCII command formatting for UART.

Command		Description
Binary	**ASCII**	
0x01	ID	Returns the equipment identification
0x02	RST	Resets the device
0x03	POW	Returns or changes the power mode of the device
0x04	ERM	Returns or changes the error returning mode
0x08	TMP	Returns the temperature of the microcontroller
0x10	UART	Returns or changes the baud rate of the UART
0x11	PTY	Returns or changes the UART parity
0x20	IIC	Returns or changes the address for SMBus/I²C
0x50	SET	Sets the network configuration
0x51	POS	Returns the network configuration
0x52	CHSET	Sets the specified user-defined channel
0x53	CHGET	Returns the position of the specified user-defined channel
0x54	CHMOD	Modifies the specified user-defined channel
0x55	WVL	Returns or sets the output wavelength
0x56	WVMIN	Returns the minimum selectable wavelength
0x57	WVMAX	Returns the maximum selectable wavelength

Table 4 – List of available commands

9.1 ID – Returns the equipment identification

Returns the system identification, which consists of the model, the serial number and the firmware version. The three fields are separated by pipe characters '|' (ASCII 0x7C).

Parameters:			
	PRODUCT	(string)	product model
	S/N	(string)	serial number
	FW_REV	(string)	firmware version

Use with UART:

Syntax: **ID**↵

Reply on success: **ID <PRODUCT>|<S/N>|<FW_REV>**

Example: **ID**↵

ID TF|2010-20-002|1.2

Use with SMBus/I²C bus:

Syntax: **ADDR+W,0x01,0x00,CRC8**

Reply on success: **ADDR+R,0x01,STR_LEN,PRODUCT,'|',S/N,'|',FW_REV,CRC8**

Example:
(T) FE 01 00 55
(R) FF 01 0A 54 46 7C 4E 2F 41 7C 35 2E 31 16

9.2 RST – Resets the board

Resets the system.

Parameters: none

Use with UART:

Syntax: **RST**↵

Reply on success: **RST**

Example:
RST↵
RST

Use with SMBus/I²C bus:

Syntax: **ADDR+W,0x02,0x00,CRC8**

Reply on success: **ADDR+R,0x02,0x00,CRC8**

Example:
(T) FE 02 00 6A
(R) FF 02 00 01

9.3 POW – Returns or changes the power mode of the device

If called without parameters, returns 0 if the device is in low power mode or 1 otherwise. If **POW_MODE** is entered, switch to low power mode if its value is 0, or switch to normal power mode if the value is 1. The MEMS control only works when the power mode is set to 1. This parameter returns to the default value after any reset or power-on.

Parameters: **POW_MODE** (char, 0 or 1) power status

Default: **0**

Use with UART:

Syntax: **POW [POW_MODE]**↵

Reply on success: **POW <POW_MODE>**

Example: query the current power mode:
POW↵
POW 0
change the current power mode:
POW 0↵
POW 0

Use with SMBus/I²C bus:

Query the current power mode:

Syntax: **ADDR+W,0x03,0x00,CRC8**

Reply on success: **ADDR+R,0x03,0x01,POW_MODE,CRC8**

Example:
(T) FE 03 00 7F
(R) FF 03 01 00 79

Change the current power mode:

Syntax: **ADDR+W,0x03,0x01,POW_MODE,CRC8**

Reply on success: **ADDR+R,0x03,0x01,POW_MODE,CRC8**

Examples:
(T) FE 03 01 00 6F (set low power mode)
(R) FF 03 01 00 79

(T) FE 03 01 01 68 (set normal power mode)
(R) FF 03 01 01 7E

9.4 ERM – Returns or changes the UART error returning mode

If called without parameters, returns 0 if the errors are identified with integer numbers or 1 if the errors are returned in plain text. If **ERR_MODE** is entered, switch to verbose mode if its value is 1, or switch to number mode if the value is 0. This parameter returns to the default value after any reset or power-on.

Parameters: **ERR_MODE** (char, 0 or 1) error returning mode

Default: **1**

Use with UART:

Syntax: **ERM [ERR_MODE]**↵

Reply on success: **ERM <ERR_MODE>**

Example: query the current error returning mode:
ERM↵
ERM 0
change the current error returning mode:
ERM 0↵
ERM 0

Use with SMBus/I²C bus:

Query the current error returning mode:

Syntax: **ADDR+W,0x04,0x00,CRC8**

Reply on success: **ADDR+R,0x04,0x01,ERR_MODE,CRC8**

Example:
(T) FE 04 00 14
(R) FF 04 01 00 6F

Change the current error returning mode:

Syntax: **ADDR+W,0x04,0x01,ERR_MODE,CRC8**

Reply on success: **ADDR+R,0x04,0x01,ERR_MODE,CRC8**

Examples:

```
(T) FE 04 01 00 79   (set number mode)
(R) FF 04 01 00 6F

(T) FE 04 01 01 7E   (set verbose mode)
(R) FF 04 01 01 68
```

9.5 TMP – Returns the temperature of the microcontroller

Returns the temperature of the microcontroller unit, expressed in Celsius degrees.

Parameters: **TEMP** (char) temperature

Use with UART:

Syntax: **TMP**↵

Reply on success: **TMP <TEMP>**

Example:

```
TMP↵
TMP 38
```

Use with SMBus/I²C bus:

Syntax: **ADDR+W,0x08,0x00,CRC8**

Reply on success: **ADDR+R,0x08,0x01,TEMP,CRC8**

Example:

```
(T) FE 08 00 E8
(R) FF 08 01 1D C6
```

9.6 UART – Changes the speed of the UART interface

If called without parameters, returns the speed of the UART port. If **UART_BAUD** is entered, changes the baud rate of the UART accordingly to Table 5. This parameter returns to the default value after any reset or power-on.

Value	Baud rate
0	9600 (default)
1	19200
2	38400
3	57600
4	115200

Table 5 – List of available baud rates

Parameters: **UART_BAUD** (char, 0 to 4) port speed

Default: **0**

Use with UART:

Syntax: **UART [UART_BAUD]**↵

Reply on success: **UART <UART_BAUD>**

Example: query the current baud rate:

```
UART↵
UART 3
```

change the baud rate:

```
UART 2↵
UART 2
```

Use with SMBus/I²C bus:

Query the current baud rate:

Syntax: **ADDR+W,0x10,0x00,CRC8**

Reply on success: **ADDR+R,0x10,0x01,UART_BAUD,CRC8**

Example:

```
(T) FE 10 00 17
(R) FF 10 01 00 3F
```

Change the current baud rate:

Syntax: **ADDR+W,0x10,0x01,UART_BAUD,CRC8**

Reply on success: **ADDR+R,0x10,0x01,UART_BAUD,CRC8**

Examples:

```
(T) FE 10 01 00 70   (set 9600 bps)
(R) FF 10 01 00 66

(T) FE 10 01 04 6C   (set 115200 bps)
(R) FF 10 01 04 7A
```

9.7 PTY – Changes the parity setting of the UART interface

If called without parameters, returns the parity setting of the UART port. If **UART_PARITY** is entered, changes the parity setting of the UART accordingly to Table 6. This parameter returns to the default value after reset or power-on.

Value	Parity
0	none (default)
1	even
2	odd
3	mark
4	space

Table 6 – List of available parity settings

Parameters: **UART_PARITY** (char, 0 to 4) UART parity

Default: **0**

Use with UART:

Syntax: **PTY [UART_PARITY]↵**

Reply on success: **PTY <UART_PARITY>**

Example: query the current parity:
PTY↵
PTY 3
change the parity:
PTY 2↵
PTY 2

Use with SMBus/I²C bus:

Query the current parity:

Syntax: **ADDR+W,0x11,0x00,CRC8**

Reply on success: **ADDR+R,0x11,0x01,UART_PARITY,CRC8**

Example:
(T) FE 11 00 02
(R) FF 11 01 00 0D

Change the current parity:

Syntax: **ADDR+W,0x11,0x01,UART_PARITY,CRC8**

Reply on success: **ADDR+R,0x11,0x01,UART_PARITY,CRC8**

Examples:
(T) FE 11 01 00 1B (disable parity)
(R) FF 11 01 00 0D

(T) FE 11 01 01 1C (set even parity)
(R) FF 11 01 01 0A

9.8 IIC – Returns or changes the address for SMBus/I²C

If called without parameters, this command returns the current address of the device, as a decimal number. If **IIC_ADDR** is entered, changes the address of the device. The new value is stored in the internal flash memory and is preserved during reset and power-off.

Parameters: **IIC_ADDR** (char, 0 to 255) SMBus/I²C address

Default: **254**

Use with UART:

Syntax: **IIC [IIC_ADDR]**↵

Reply on success: **IIC <IIC_ADDR>**

Example: query the current address:
IIC↵
IIC 254
modify the address:
IIC 2↵
IIC 2

Use with SMBus/I²C bus:

Query the current address:

Syntax: **ADDR+W,0x20,0x00,CRC8**

Reply on success: **ADDR+R,0x20,0x01,IIC_ADDR,CRC8**

Example:
(T) FE 20 00 EE
(R) FF 20 01 FE 73

Modify the address:

Syntax: **ADDR+W,0x20,0x01,IIC_ADDR,CRC8**

Reply on success: **ADDR+R,0x20,0x01,IIC_ADDR,CRC8**

Example:
(T) FE 20 01 A0 F8 (new address is 0xA0)
(R) FF 20 01 A0 EE

9.9 SET – Moves the MEMS mirror to the specified position

Moves the MEMS mirror to the (x,y) coordinates. The signs of x and y define the tilt direction of the micro-mirror. The absolute value is proportional to the full scale of the digital to analog converter. Only one semi-axes is active at any moment, so two parameters are always 0: if x is positive, assign this value at x+ and set x- to 0, if x is negative assign this value at x- and set x+ to 0. The same applies to y.

This command is not available when the board is in power saving mode (see command POW).

Parameters:			
	X_NEG	(integer, 0 to 65535)	x- axis
	X_POS	(integer, 0 to 65535)	x+ axis
	Y_NEG	(integer, 0 to 65535)	y- axis
	Y_POS	(integer, 0 to 65535)	y+ axis

Use with UART:

Syntax: **SET <X_NEG> <X_POS> <Y_NEG> <Y_POS>↵**

Reply on success: **SET <X_NEG> <X_POS> <Y_NEG> <Y_POS>**

Example:
SET 2000 0 500 0↵
SET 2000 0 500 0

Use with SMBus/I²C bus:

Syntax:
ADDR+W,0x50,0x08,
X_NEG[MSB],X_NEG[LSB],X_POS[MSB],X_POS[LSB],
Y_NEG[MSB],Y_NEG[LSB],Y_POS[MSB],Y_POS[LSB],CRC8

Reply on success:
ADDR+R,0x50,0x08,
X_NEG[MSB],X_NEG[LSB],X_POS[MSB],X_POS[LSB],
Y_NEG[MSB],Y_NEG[LSB],Y_POS[MSB],Y_POS[LSB],CRC8

Example:
(T) FE 50 08 61 A8 00 00 00 00 48 44 81
(R) FF 50 08 61 A8 00 00 00 00 48 44 9E

9.10 POS – Returns the current position of the MEMS mirror

Returns the actual (x,y) coordinates of the mirror. The signs of x and y define the tilt direction of the micro-mirror. The absolute value is proportional to the full scale of the digital to analog

converter. This command is not available when the board is in power saving mode (see command POW).

Parameters:			
	X_NEG	(integer, 0 to 65535)	x- axis
	X_POS	(integer, 0 to 65535)	x+ axis
	Y_NEG	(integer, 0 to 65535)	y- axis
	Y_POS	(integer, 0 to 65535)	y+ axis

Use with UART:

Syntax: **POS**↵

Reply on success: **POS <X_NEG> <X_POS> <Y_NEG> <Y_POS>**

Example:
```
POS↵
POS 2000 0 500 0
```

Use with SMBus/I²C bus:

Syntax: **ADDR+W,0x51,0x00,CRC8**

Reply on success:
```
ADDR+R,0x51,0x08,
     X_NEG[MSB],X_NEG[LSB],X_POS[MSB],X_POS[LSB],
     Y_NEG[MSB],Y_NEG[LSB],Y_POS[MSB],Y_POS[LSB],CRC8
```

Example:
```
(T) FE 51 00 59
(R) FF 51 08 00 00 7A 10 00 00 25 AA F2
```

9.11 CHSET – Sets the specified user-defined channel

Moves the internal MEMS mirror to the coordinates stored in the memory location **P**. If **P** is empty, an error message is returned. This command is not available when the board is in power saving mode (see command POW).

Parameters:			
	P	(integer, 0 to 127)	memory location

Use with UART:

Syntax: **CHSET <P>**↵

Reply on success: **CHSET <P>**

Example:
```
CHSET 1↵
CHSET 1
```

Use with SMBus/I²C bus:

Syntax: **ADDR+W,0x52,0x02,P[MSB],P[LSB],CRC8**

Reply on success: **ADDR+R,0x52,0x02,P[MSB],P[LSB],CRC8**

Example:
```
(T) FE 52 02 00 02 53     (set channel 2)
(R) FF 52 02 00 02 31
```

9.12 CHGET – Returns the content of the specified user-defined channel

Returns the (x,y) coordinates stored in memory location **P**. The signs of x and y define the tilt direction of the micro-mirror. The absolute value is proportional to the full scale of the digital to analog converter.

Parameters:			
	P	(integer, 0 to 127)	memory location
	X_NEG	(integer, 0 to 65535)	x- axis
	X_POS	(integer, 0 to 65535)	x+ axis
	Y_NEG	(integer, 0 to 65535)	y- axis
	Y_POS	(integer, 0 to 65535)	y+ axis

Use with UART:

Syntax: **CHGET <P>**↵

Reply on success: **CHGET <P> <X_NEG> <X_POS> <Y_NEG> <Y_POS>**

Example:
```
CHGET 1↵
CHGET 1 0 45 1050 0
```

Use with SMBus/I²C bus:

Syntax: **ADDR+W,0x53,0x02,P[MSB],P[LSB],CRC8**

Reply on success:
```
ADDR+R,0x53,0x0A,P[MSB],P[LSB]
     X_NEG[MSB],X_NEG[LSB],X_POS[MSB],X_POS[LSB],
     Y_NEG[MSB],Y_NEG[LSB],Y_POS[MSB],Y_POS[LSB],CRC8
```

Example:
```
(T) FE 53 02 00 05 50
(R) FF 53 0A 00 05 A0 00 00 00 00 00 FE 01 93
```

9.13 CHMOD – Modifies the specified user-defined channel

Saves in the memory location **P** the mirror position defined by the coordinates (x,y). The signs of x and y define the tilt direction of the micro-mirror. The absolute value is proportional to the full scale of the digital to analog converter. Only one semi-axes is active at any moment, so two parameters are always 0: if x is positive, assign this value at x+ and set x- to 0, if x is negative assign this value at x- and set x+ to 0. The same applies to y.

Warning: the overwritten points will be permanently lost!

Parameters:			
	P	(integer, 0 to 127)	memory location
	X_NEG	(integer, 0 to 65535)	x- axis
	X_POS	(integer, 0 to 65535)	x+ axis
	Y_NEG	(integer, 0 to 65535)	y- axis
	Y_POS	(integer, 0 to 65535)	y+ axis

Use with UART:

Syntax: **CHMOD <P> <X_NEG> <X_POS> <Y_NEG> <Y_POS>**↵

Reply on success: **CHMOD <P> <X_NEG> <X_POS> <Y_NEG> <Y_POS>**

Example:
```
CHMOD 1 0 45 1050 0↵
CHMOD 1 0 45 1050 0
```

Use with SMBus/I²C bus:

Syntax:
```
ADDR+W,0x54,0x0A,P[MSB],P[LSB]
      X_NEG[MSB],X_NEG[LSB],X_POS[MSB],X_POS[LSB],
      Y_NEG[MSB],Y_NEG[LSB],Y_POS[MSB],Y_POS[LSB],CRC8
```

Reply on success:
```
ADDR+R,0x54,0x10,P[MSB],P[LSB]
      X_NEG[MSB],X_NEG[LSB],X_POS[MSB],X_POS[LSB],
      Y_NEG[MSB],Y_NEG[LSB],Y_POS[MSB],Y_POS[LSB],CRC8
```

Example:
```
(T) FE 54 0A 00 01 00 00 0A 70 02 15 00 00 A9
(R) FF 54 0A 00 01 00 00 0A 70 02 15 00 00 3D
```

9.14 WVL – Returns or sets the output wavelength

If called without parameters, returns the central wavelength of the filter expressed in nanometers. If **LAMBDA** is entered, moves the central wavelength of the filter to the desired value. The range of available wavelengths **LAMBDA** is comprised between the minimum and maximum values that can be found using the commands WVMIN and WVMAX. This command is not available when the board is in power saving mode.

Parameters: **LAMBDA** (float) wavelength

Use with UART:

Syntax: **WVL [LAMBDA]↵**

Reply on success: **WVL <LAMBDA>**

Example: query the current wavelength:
```
WVL↵
WVL 1560.250
```
change the current wavelength:
```
WVL 1548↵
WVL 1548.000
```

Use with SMBus/I²C bus:

Query the current wavelength:

Syntax: **ADDR+W,0x55,0x00,CRC8**

Reply on success: **ADDR+R,0x55,0x04,LAMBDA[MSB],…,LAMBDA[LSB],CRC8**

Example:
```
(T) FE 55 00 EE
(R) FF 55 04 FE 73
```

Modify the wavelength:

Syntax: **ADDR+W,0x55,0x04,LAMBDA[MSB],…,LAMBDA[LSB],CRC8**

Reply on success: **ADDR+R,0x55,0x04,LAMBDA[MSB],…,LAMBDA[LSB],CRC8**

Example: **(T) FE 55 04 44 C1 C0 00 B9** (new wavelength is 1550.0 nm)
(R) FF 55 04 44 C1 C0 00 66

9.15 WVMIN – Returns the minimum selectable wavelength

Returns the shorter wavelength **LAMBDA_MIN** selectable by the filter and expressed in nanometers.

Parameters: **LAMBDA_MIN** (float) minimum wavelength

Use with UART:

Syntax: **WVMIN**↵

Reply on success: **WVMIN <LAMBDA_MIN>**

Example: **WVMIN**↵
WVMIN 1503.990

Use with SMBus/I²C bus:

Syntax: **ADDR+W,0x56,0x00,CRC8**

Reply on success: **ADDR+R,0x56,0x04,LAMBDA_MIN[MSB],**
…,LAMBDA_MIN[LSB],CRC8

Example: **(T) FE 56 00 32**
(R) FF 56 04 44 BF 10 00 EC (min wavelength is 1528.5 nm)

9.16 WVMAX – Returns the maximum selectable wavelength

Returns the longer wavelength **LAMBDA_MAX** selectable by the filter and expressed in nanometers.

Parameters: **LAMBDA_MAX** (float) maximum wavelength

Use with UART:

Syntax: **WVMAX**↵

Reply on success: **WVMAX <LAMBDA_MAX>**

Example: **WVMAX**↵
WVMAX 1600.590

Use with SMBus/I²C bus:

Syntax: **ADDR+W,0x57,0x00,CRC8**

Reply on success: **ADDR+R,0x57,0x04,LAMBDA_MAX[MSB],**
…,LAMBDA_MAX[LSB],CRC8

Example: **(T) FE 57 00 27**
(R) FF 57 04 44 C4 40 00 42 (max wavelength is 1570.0 nm)

10 Error Codes

The following table gives information about error codes.

Error number	Description
2	CRC error – The CRC of the last SMBus/I²C command is invalid (see Paragraph 6)
3	Invalid parameter(s)
4	Command unknown
6	Buffer overrun – The command is too long and UART or SMBus/I²C receive buffer is full
8	Command unavailable because the device is in idle mode
9	The memory location of the selected channel is empty
10	Current wavelength is unknown

Table 7 – Overview of error codes

11 Absolute Maximum Ratings

Applicable absolute maximum ratings for the full operating temperature range without causing irreversible damage to the device are listed in Table 8.

	Unit	Min	Max
Operation temperature	°C	0	70
Storage temperature	°C	-40	70
Operation humidity (non condensing)	% r.h.	0	95
Maximum optical power	mW		500
Voltage on V_{DD}	V	GND-0.2	5.5
Voltage on any IO pin	V	GND-0.2	5.8
Total current sunk into V_{DD}	mA		300
Current sourced or sunk by any IO pin	mA	-100	100

Table 8 – Absolute maximum ratings

12 Optical Specifications

Please refer to the datasheet of the TF1 for the optical specifications.

13 Electrical Specifications

	Unit	Min	Typ	Max
Supply voltage	V	4.75	5	5.25
Power consumption (normal mode)	W			1
Power consumption (idle mode)	W		0.2	
SMBus/I^{2}C bus speed	kbps	100		400
UART speed	baud	9600		115200
UART Logic Level 0	V		0	0.6
UART Logic Level 1	V	2.4	5.0	
Input logic level low	V		0	0.6
Input logic level high	V	2.4	5.0	
Output logic level low	V		0	0.6
Output logic level high	V	2.6	3.3	
Reset inactive voltage[3]	V	2.4	5	
Reset active voltage	V		0	0.9
Reset pulse duration	µs	15		
Onboard pull-up resistor value	kΩ	110	165	330
Onboard pull-up voltage	V		3.3	

14 Package Specifications

	Unit	Min	Typ	Max
Pigtail length	cm	50		100
Dimensions	mm	45 x 23 x 10		
Weight	g	75		
RoHS compliance		2015/863/EU (no exceptions)		

[3] Through onboard pull-up resistor

1 Revision History

Version	Date	Modified	Approved	Changes
3.6	08.06.2020	P. Baroni	C. Marxer	Table 1: added signals description Paragraph 5: added additional details about UART Added Paragraph 8 Paragraphs 9.12 and 9.13: corrected example Paragraph 10: added additional details about error codes, removed error codes not relevant to this device Minor corrections
3.7	20.07.2020	P. Baroni	C. Marxer	Added I2C/SMBus examples Minor corrections
3.8	26.03.2021	P. Baroni	C. Marxer	Absolute maximum ratings Pull-up resistors value Minor corrections

15 Ordering Information

TF	1	C	100	-	9	N

Number of independent channels:
1 = 1 (2 fibers)

Tuning range:
O = 1260-1360 nm
C = 1528-1569 nm
L = 1570-1615 nm
C+L = 1529-1615 nm

Bandwidth FWHM:
50 = 0.3 nm
100 = 0.5 nm

Fiber type:
9 = SMF28ULTRA
(other fiber types available upon request)

Fiber sleeve type:
N = loose tube 0.9mm
B = bare fiber 0.25mm

16 Contact Information

sercalo Microtechnology Ltd

Landstrasse 151, 9494 Schaan

Principality of Liechtenstein

Tel.: +423 237 57 97

Fax: +423 237 57 48

http://www.sercalo.com

Email: info@sercalo.com