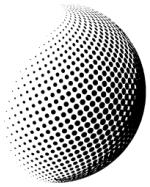

MERIDIAN
Innovation

Meridian Innovation MI48xx USB Interface Protocol

Reference Manual

Revision 1.0.3 – Jun 2024

Contents

1. OVERVIEW

The USB Interface Communication Protocol described here underlies the communication between the host system and Meridian Innovation's thermal imaging processor MI48XX in the following two cases:

- The MI48Ax is on an evaluation kit with a USB interface, where the EVK provides a bridge between the USB and the SPI/I2C interfaces of the MI48Ax.
- The MI48Bx is in Meridian Innovation's mobile phone attachment XPRO, with USB-C interface connector.
- The MI48Dx supports both USB mode or SPI/I2C interface mode. Depends on the logic level of the MODE pin during power up, it works as USB mode (MODE=1) or SPI/I2C mode (MODE=0).

This communication protocol must be implemented in the host system, in order to allow it to exchange control and status information with the MI48xx, as well as read out the thermal image data. The conceptual overview of the system is shown in Fig. 1.

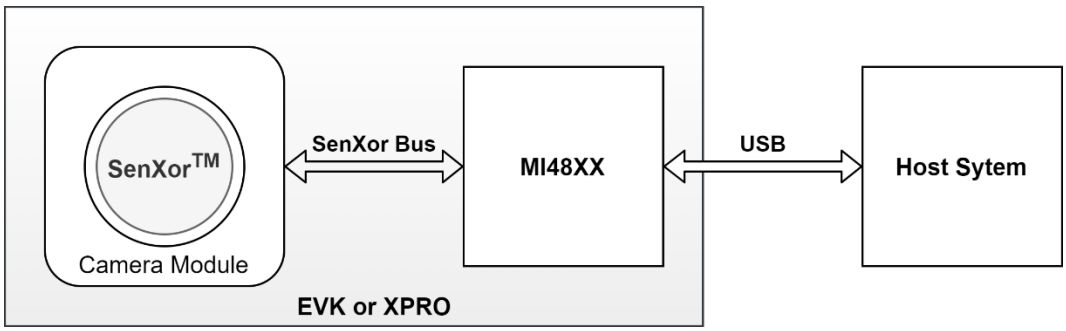

Fig. 1. A conceptual overview of connecting the MI48XX Thermal Imaging Processor to a Host System via USB. The Host System could be a computer with a USB interface (if the MI48XX is part of the Evaluation Kit MI48A0P0) or an Android-based mobile device (if the MI48XX is on the XPRO phone attachment with USB-C connector MI0800X0G).

From the perspective of the host, the MI48XX would be enumerated as a Virtual Comport, described in the USB Communication Device Class (CDC) reference.

The communication protocol consists of *command-acknowledge* type of messages exchanged between the host and the MI48xx via the USB physical interface. The following section describes the overall message format and the supported commands and acknowledges.

2. COMMUNICATION PROTOCOL

2.1. Message Format

All commands and acknowledge messages conform to the format shown in Table 1, and must be interpreted left to right.

Table 1. MESSAGE FORMAT

Message Delimiter “ #”	Message Length	Command or Acknowledge Name	Data	Check Sum
4 bytes	4 bytes	4 bytes	Variable length	4 bytes

2.1.1. Message Delimiter

The message begins with a message delimiter that consists of 4 bytes: three space characters followed by the “#” symbol. The message delimiter is essential for distinguishing the start of a message.

2.1.2. Message contents

The shaded columns in Table 1 represent the message contents. It includes

- 4 bytes that state the length of the entire message contents (including the Message Length field itself) in number of bytes; The number is encoded in 4 hexadecimal ASCII characters.
- 4-byte command or acknowledge name (refer to Table 2); It is naturally encoded in 4 ASCII characters.
- A data field of variable length that depends on the command or acknowledge, and can be 0 bytes too. The data values may be encoded in hexadecimal ASCII characters, or as raw bytes – this also depends on the type of command.

2.1.3. Check Sum

The check sum provides a way to verify the integrity of the message. It is obtained by

- summing all bytes within the message contents,
- retaining the least significant 16 bits, and
- encoding them as ASCII characters (hence the 4 bytes).

2.2. Supported Commands

The communication between the host system and the MI48xx via USB interface consists of a series of command messages sent by the host, to each of which the MI48XX responds

with an acknowledge message. However, when the MI48XX starts thermal data acquisition through the attached camera module, it initiates acknowledge messages on its own.

The host can issue two types of commands:

- Write Register Command, named “WREG”, for writing to a register of the MI48, and,
- Read Register Command, named “RREG”, for reading a register of the MI48XX
- Read Multiple Registers Command, named “RRSE”, for reading multiple registers of the MI48XX in one time.

The MI48XX can issue three types of acknowledges:

- Write Register Acknowledge, named “WREG”, when it has acted on a WREG command from host successfully
- Read Register Acknowledge, named “RREG”, yielding the contents of the register requested by the host via a “RREG” command.
- Read Multiple Register Acknowledge, named “RRSE”, yielding the contents of the registers requested by the host via a “RRSE” command.
- Frame Acknowledge, named “GFRA”, when it yields thermal image data

Table 2 and Table 3 detail out the contents of the supported command and acknowledge messages, along with some examples.

Table 2. SUPPORTED COMMANDS – HOST TO MI48XX

Length	Name	Data	Example
10 bytes (0x000A)	RREG	2-byte hex ASCII register address	‘ #000ARREGB6XXXX’ Read STATUS register
12 bytes (0x000C)	WREG	2-byte hex ASCII register address + 2-byte hex ASCII register value	‘ #000CWREGB102XXXX’ Write 0x02 to FRAME_MODE register
Variable Bytes	RRSE	Multiple 2-byte hex ASCII registers address, terminated by FF	‘ #0016RRSE E0E1E2E3E4E5FFXXXX’ Read SenxorID register 0xE0-0xE5, terminated by FF to indicate end

Table 3. SUPPORTED ACKNOWLEDGES – MI48XX TO HOST

Length	Name	Data	Example
10 bytes (0x000A)	RREG	2-byte hex ASCII register value	‘ #000AWREG130265’ Value of requested register
8 bytes (0x0008)	WREG	None	‘ #0008WREG01FD’ Successful Write to a register

Depends on camera module	GFRA	Thermal Data Frame, raw bytes	E.g. For MI08XXXX ‘ #2808GFRA’+ Thermal data frame bytes + ASCII encoded check sum E.g. For MI16XXXX ‘ #9B08GFRA’+ Thermal data frame bytes + ASCII encoded check sum
Variable Bytes	RRSE	4-bytes hex ASCII in {register, value} pair	‘ #0020RRSE+ E016E117E200E300E431E5500723’ Register address and value of the registers requested in pair. In above example 6 pairs of {register, value} as {E0,16}, {E1,17}, {E2,00}, {E3,00}, {E4,31}, {E5,50}

The length of the GFRA acknowledge depends on the resolution of the attached camera module as shown in Table 4.

Table 4. LENGTH OF GFRA MESSAGE CONTENTS

Camera Module	Resolution	GFRA message length
MI08XXXX	80 x 62	10,248 bytes (0x2808)
MI16XXXX	160 x 120	39,688 bytes (0x9B08)

2.2.1. Note on Encoding

Except for the Thermal Data Frame inside a GFRA acknowledge from the MI48XX to the host, all other numeric values in a message, be it command or acknowledge, are ASCII encoded hexadecimal representations. For GFRA acknowledge message, the thermal data bytes are in binary representation.

2.2.2. Interpretation of the Thermal Data Frame

Interpretation of the Thermal Data Frame should be done by an upper application level. The contents of the Thermal Data Frame is detailed out in the MI48XX Datasheet.

Note however, that although the size of the Thermal Data Frame in principle depends on whether a Frame Header is requested or not (see FRAME_MODE register 0xB1, bit 5 – NO_HEADER in MI48XX register map), the length of the USB message is invariant in this regard. Specifically, if the Frame Header is not requested (i.e. NO_HEADER bit is set to 1), the USB GFRA acknowledge will contain all 0’s in place of the HEADER of the Thermal Data Frame.

Table 5. MI08XX THERMAL DATA FRAME FORMAT INCLUDING FRAME HEADER

USB Header 12 (MI08XX)	**ASCII ‘ #’** 4 bytes **ASCII**			**Len 2808** 4 bytes			**ASCII ‘GFRA’** 4 bytes	
Reserve 80 words (MI08XX)	**Reserved** 80 words							
Frame Header 80 words (MI08XX)	**Frame counter** 1 word	**SenXor VDD** 1 word	**SenXor die temperature** 1 word	**Time stamp** 2 words	**Max pixel value** 1 word	**Min pixel value** 1 word	**CRC** 1 word	**Reserved** 72 words (MI08XX)
Tempe-rature data	80 column * 62 row Pixel Data, i.e. 4960 words (MI0802 Camera Module)							
Checksum	**Checksum** 4 bytes							

Table 6. MI16XX THERMAL DATA FRAME FORMAT INCLUDING FRAME HEADER

USB Header 12 bytes (MI16XX)	**ASCII ‘ #’** 4 bytes **ASCII**			**Len 9B08** 4 bytes			**ASCII ‘GFRA’** 4 bytes	
Reserved 160 words (MI16XX)	**Reserved** 160 words							
Reserved 160 words (MI16XX)	**Reserved** 160 words							
Reserved 160 words (MI16XX	**Reserved** 160 words							
Frame Header 160 words (MI16XX)	**Frame counter** 1 word	**SenXor VDD** 1 word	**SenXor die temperature** 1 word	**Time stamp** 2 words	**Max pixel value** 1 word	**Min pixel value** 1 word	**CRC** 1 word	**Reserved** 152 words (MI16XX)
Tempe-rature data	160 column * 120 row Pixel Data, i.e. 19,200 words (MI16XX Camera Module)							
Checksum	**Checksum** 4 bytes							

Note1: 1 word = 2 bytes

3. REFERENCE IMPLEMENTATIONS

Currently, Meridian Innovation offers the following reference software development kits (SDK) that implement the above Communication Protocol:

- **PySenXor** – Python library for communicating with the MI48XX via USB, or SPI/I2C interface, for Linux (including WSL and Raspbian), Windows, MacOS.
- **Android SDK** – Java SDK geared towards Android based applications for Mobile Phones and Tablets.

4. REVISION HISTORY

Revision	*Date*	*Comment*
1.0.1	1 Mar 2020	Combining Original Application Note and Recent Updates on USB interface protocol.
1.0.2	28 Dec 2022	Add RRSE Command.
1.0.3	19 Jun 2024	Update frame format, and also add description for MI16XXXX.

5. LEGAL INFORMATION

Meridian products are not intended for use in life support appliances, devices, or systems. Use of Meridian products for purposes other than those set forth in the accompanying NDA without the written consent of an appropriate Meridian officer is prohibited.

6. CONTACTS INFORMATION

For more information, please visit www.meridianinno.com

For sales inquiries, please email info@meridianinno.com

Headquarters: Meridian Innovation Pte. Ltd., 2 Vision Exchange, #11-08, Singapore

Company Registration Number: 201611173R