USB SERIAL PROTOCOL

REV 15

This guide describes how you can communicate with the DIY-Thermocam V1/V2 over the USB serial connection.

The maximum transfer speed is 12 Mbit/s and is always active, no matter what baud rate you use to connect to the device from your PC. This allows you to receive thermal frames with high speed. The device needs to be turned on, connected to the PC over USB and set to live mode.

For a closer look at the code implementation of the USB serial connection, check out the file "Connection.h" inside the folder "Hardware" on the Github. The thermal live viewer implements most parts of the protocol, so have a look at the Python source code included in the download archive of the software.

All commands are transmitted as decimal bytes. In order to start the serial connection, send the value 100 (CMD_START) to the device connected at port COM10, then wait for the acknowledge byte (100 dec).

The number of bytes returned by each command is listed in the return part of each command. Whenever a command does not send data back, it will send its own command ID as acknowledge (ACK) if successful and zero (0) if not successful. In case data needs to be transmitted after the command itself, it is marked as payload in the description, followed by the number of bytes that need to be transmitted and their description.

- **100: CMD_START**
  - **Description:** Set the device into serial mode and waits for the next command from the list
  - **Returns** (1 byte): ACK or 0

- **200: CMD_END**
  - **Description:** Exit serial mode and set the device back to live mode
  - **Returns** (1 byte): ACK or 0

- **110: CMD_GET_RAWLIMITS**
  - **Description:** Get the lepton raw limits (14-bit) of the current frame
  - **Returns** (4 byte): minTemp MSB, minTemp LSB, maxTemp MSB, maxTemp LSB

- **111: CMD_GET_RAWDATA**
  - **Description:** Get the lepton raw values, 9600 byte (Lepton2) or 38400 byte (Lepton3)
  - **Explanation:** 14-bit raw values (0 … 16383) for each pixel, equivalent to infrared intensity
  - **Returns** (9600 / 38400 byte): 4800 / 19200 * (rawData[i] MSB, rawData[i] LSB)

- **112: CMD_GET_CONFIGDATA**
  - **Description:** Get the configuration data, that does not change if not adjusted manually
  - **Returns** (10 byte):
    - leptonVersion (1 Byte): 0 = Lepton2 shutter, 1 = Lepton3 shutter, 2 = Lepton2 no shutter
    - rotationEnabled (1 Byte): 0 = Normal orientation, 1 = Device rotated by 180 degree
    - colorScheme (1 Byte): Number definitions can be looked up in the file "GlobalDefines.h"
    - tempFormat (1 Byte): 0 = Celsius, 1 = Fahrenheit
    - spotEnabled (1 Byte): 0 = Disabled, 1 = Spot temperature displayed
    - colorbarEnabled (1 Byte): 0 = Disabled, 1 = Colorbar displayed
    - minMaxEnabled (1 Byte): 0 = Disabled, 1 = Min, 2 = Max, 3 = Both
    - textColor (1 byte): 0 = White, 1 = Black, 2 = Red, 3 = Green, 4 = Blue
    - filtertype (1 byte): 0 = Disabled, 1 = Gaussian blur, 2 = Box blur
    - adjustLimits (1 Byte): 0 = Manual mode, limits fixed, 1 = Auto mode, limits adjust

- **113: CMD_GET_CALSTATUS**
  - **Description:** Get the calibration status from the device and seconds left for warmup
  - **Returns** (2 byte): calStatus (0 = warmup, 1 = standard, 2 = manual), seconds left for warmup

- **114: CMD_GET_CALIBDATA**
  - **Description:** Get the calibration data, offset first, then slope
  - **Explanation:** Required for raw-to-absolute temperature conversion with the following formula: *absTemp = rawValue * calibrationSlope + calibrationOffset*
  - **Returns** (8 byte): two float, split to four bytes each, LSB first, MSB at the end

- **115: CMD_GET_SPOTTEMP**
  - **Description:** Get the spot temperature from the MLX90614 sensor, in Fahrenheit or Celcius
  - **Returns** (4 byte): Float split into four bytes, LSB first, then MSB

- **116: CMD_SET_TIME**
  - **Description:** Set the time on the device, used to sync with PC time
  - **Explanation:** The temperature format is "*%Y-%m-%d %H:%M:%S\n*", with %Y = year, %m = month, %d = day, %H = hour, %M = month and %S = second
  - **Returns** (1 byte): ACK or 0

- **117: CMD_GET_TEMPPOINTS**
  - **Description:** Get the temperature points that are shown on the screen
  - **Explanation:** First two bytes are index value (1 = (0,0) … 19200 = (159, 119)), second two bytes are the raw value. If the index is 0, the point is not set, otherwise evaluate the raw value
  - **Returns** (384 byte): 96 points (96 x 4), 2 bytes index, 2 bytes raw value, MSB first, then LSB

- **118: CMD_SET_LASER**
  - **Description:** Toggles the laser marker on the device (DIY-Thermocam V1 only)
  - **Returns** (1 byte): ACK or 0

- **119: CMD_GET_LASER**
  - **Description:** Get the current laser state (DIY-Thermocam V1 only)
  - **Returns** (1 byte): 0 = Disabled, 1 = Enabled

- **120: CMD_SET_SHUTTERRUN**
  - **Description:** Run the shutter flat-field-correction (FFC) on the FLIR Lepton
  - **Returns** (1 byte): ACK or 0

- **121: CMD_SET_SHUTTERMODE**
  - **Description:** Set the Lepton shutter FFC mode to automatic or manual
  - **Payload** (1 byte): 0 = Manual, 1 = Automatic
  - **Returns** (1 byte): ACK or 0

- **122: CMD_SET_FILTERTYPE**
  - **Description:** Set the type of filter used for the lepton raw values
  - **Payload** (1 byte): 0 = Disabled, 1 = Gaussian blur, 2 = Box blur
  - **Returns** (1 byte): ACK or 0

- **123: CMD_GET_SHUTTERMODE**
  - **Description:** Get the current shutter FFC mode from the FLIR Lepton
  - **Returns** (1 byte): 0 = Manual, 1 = Auto, 2 = None (no shutter attached)

- **124: CMD_GET_BATTERYSTATUS**
  - **Description:** Get the current battery status in percentage

    - **Returns** (1 byte): 0 = 0% (empty), 100 = 100% (fully charged)

- **125**: **CMD_SET_CALSLOPE**
    - **Description:** Set the calibration slope
    - **Payload** (4 byte): Float split into four bytes, LSB first, MSB at the end
    - **Returns** (1 byte): ACK or 0

- **126**: **CMD_SET_CALOFFSET**
    - **Description:** Set the calibration offset
    - **Payload** (4 byte): Float split into four bytes, LSB first, MSB at the end
    - **Returns** (1 byte): ACK or 0

- **127**: **CMD_GET_DIAGNOSTIC**
    - **Description:** Get the diagnostic information containing the hardware status
    - **Explanation:** The following components are checked: Spot Sensor (0), Display (1), Visual Camera (2), Touch Screen (3), SD Card (4), Battery Gauge (5), Lepton I2C (6), Lepton SPI (7)
    - **Returns** (1 byte): Diagnostic byte, False when component not working: (diagnostic >> device) & 1

- **128**: **CMD_GET_VISUALIMG**
    - **Description:** Sends the visual image with 320x240 or 640x480 resolution as JPEG byte stream
    - **Payload** (1 byte): 0 = Low resolution (320x240), for streaming, 1 = High resolution (640x480), for saving
    - **Returns** (Variable bytes, as defined in length):
        - The first two bytes are the length, MSB first, then LSB
        - JPEG data starts with two bytes 0xFF & 0xD8 and ends with two bytes 0xFF & 0xD9
        - If the display rotation is enabled, 100 bytes of EXIF information are inserted, starting at offset 20 dec (0xFF & 0xE9) and ending at offset 119 dec, the picture information follows afterwards

- **129**: **CMD_GET_FWVERSION**
    - **Description:** Get the current firmware version installed on the device
    - **Returns** (1 byte): 100 equals 1.00, 255 equals 2.55

- **130**: **CMD_SET_LIMITS**
    - **Description:** Set adjust limits to lock limit or automatic mode
    - **Explanation:** When using lock limits, the raw value limits from the last frame are locked
    - **Payload** (1 byte): 0 = Lock limits, 1 = Auto mode, limits adjust
    - **Returns** (1 byte): ACK or 0

- **131**: **CMD_SET_TEXTCOLOR**
    - **Description:** Set the text color used to display the elements in live mode
    - **Payload** (1 byte): 0 = White, 1 = Black, 2 = Red, 3 = Green, 4 = Blue
    - **Returns** (1 byte): ACK or 0

- **132**: **CMD_SET_COLORSCHEME**
    - **Description:** Set current color scheme used for the conversion, see page 1 for numbers
    - **Payload** (1 byte): color scheme number
    - **Returns** (1 byte): ACK or 0

- **133**: **CMD_SET_TEMPFORMAT**
    - **Description:** Set the temperature format to Celsius or Fahrenheit
    - **Payload** (1 byte): 0 = Celsius, 1 = Fahrenheit
    - **Returns** (1 byte): ACK or 0

- **134**: **CMD_SET_SHOWSPOT**
    - **Description:** Show spot temperature information in the image or not
    - **Payload** (1 byte): 0 = Disabled, 1 = Enabled

- **Returns** (1 byte): ACK or 0

- **135**: CMD_SET_SHOWCOLORBAR
  - **Description:** Show color bar information in the image or not
  - **Payload** (1 byte): 0 = Disabled, 1 = Enabled
  - **Returns** (1 byte): ACK or 0

- **136**: CMD_SET_SHOWMINMAX
  - **Description:** Show the hottest, coldest or both points in the image or not
  - **Payload** (1 byte): 0 = Disabled, 1 = Min, 2 = Max, 3 = Both
  - **Returns** (1 byte): ACK or 0

- **137**: CMD_SET_TEMPPOINTS
  - **Description:** Set the temperature points to be shown on the screen
  - **Explanation:** First two bytes are index value (0 = (0,0) … 19200 = (159, 119)), second two bytes are the raw value. If the index is 0, the point is not set, otherwise evaluate the raw value
  - **Payload** (384 byte): 96 points (96 x 4), 2 byte index, 2 bytes raw value, MSB first, then LSB
  - **Returns** (1 byte): ACK or 0

- **138**: CMD_GET_HWVERSION
  - **Description:** Get the hardware version of the device
  - **Returns** (1 byte): 0 = DIY-Thermocam V1, 1 = DIY-Thermocam V2

- **139**: CMD_SET_ROTATION
  - **Description:** Sets the display rotation on the device
  - **Payload** (1 byte): 0 = normal orientation, 1 = 180° rotated
  - **Returns** (1 byte): ACK or 0

- **140**: CMD_SET_CALIBRATION
  - **Description:** Run the calibration method to calculate slope and offset
  - **Returns** (102 byte): Status in percentage (0…100) and ACK when finished

- **141**: CMD_GET_HQRESOLUTION
  - **Description:** Get the HQ resolution status for the DIY-Thermocam V2
  - **Explanation:** The normal rendering resolution is 160 x 120, the HQ rendering resolution is 320 x 240
  - **Returns** (1): 0 = Normal rendering, 1 = HQ resolution rendering

- **150**: CMD_FRAME_RAW
  - **Description:** Get a raw frame containing the raw data and additional information
  - **Explanation:** First byte is the command ID:
    - 180 = push button was pressed short; user wants to save a thermal image
    - 181 = touch screen was pressed short; user wants to save a visual image
    - 182 = push button was pressed long; user wants to start/stop recording a video
    - 183 = no button pressed: normal frame, followed by the byte stream
  - **Returns:** If the command ID is 183, the following bytes are appended as payload: Raw data (9600 or 38400 byte), Raw limits (4 byte), Spot temperature (4 byte), Calibration data (8 byte)

- **151**: CMD_FRAME_COLOR
  - **Description:** Command IDs equal to the normal raw-frame, the lepton values are converted to RGB565 colors
  - **Explanation:** The raw-data to color conversion is done according to the current selected color scheme

- **152**: **CMD_FRAME_DISPLAY**
  - **Description:** Command IDs equal to the normal raw-frame, but the payload is the display framebuffer only
  - **Explanation:** On the DIY-Thermocam V2 with HQ resolution enabled, the size is 320 x 240 pixel (153600 byte), otherwise it is 160 x 120 pixel (38400 byte). Each pixel is represented by a RGB565 color (16-bit)

- **153**: **CMD_FRAME_SAVE**
  - **Description:** Save a frame [raw file + visual image (if activated) + converted image (if activated)] to the internal storage of the Thermocam
  - **Returns** (1 byte): ACK or 0