TF40 LiDAR

1. Product Overview

TF40 is an mm-level accuracy LiDAR with its range up to 40m.

Features:

- High accuracy
- Tiny
- Small light spot
- Visible laser, easier for aiming

2. Technical specifications

Table 1 Main parameters of TF40

Parameters name		Standard version
Product performance	Range	0.04-40m@90% reflectivity, 0.04-20m@10% reflectivity
	Accuracy	±2mm
	Distance resolution	1mm
	Frame rate	5Hz
	Repeatability	1σ: <2mm
	Working temperature	-10~50℃
	Enclosure rating	/
Optical parameters	Light source	LD
	Wavelength	635nm
	Laser class	CLASS 2 (EN 60825)
	Detection angle	<1mrad
Electrical parameters	Supply voltage	3.3V
	Average current	≤180mA
	Power consumption	≤0.6W
	Peak current	≤180mA
	Communication voltage level	LVTTL (3.3V)

	Communication protocol	UART
Others	Dimension	44mm*64.5mm*23mm (L*W*H)
	Enclosure Material	Aluminum alloy +ABS
	Storage temperature	-30~70℃
	Weight	57g±3g
	Cable length	35cm

3. Product Appearance

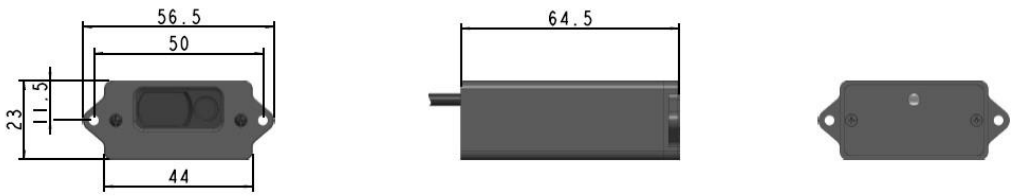

Figure 1 TF40 dimension

4. Line description

Table 1 TF40 Pin function description

No.	Color	Pin	Function
1	Red	VCC	3.3V supply
2	Black	GND	Ground wire
3	Yellow	TTL_TXD	Serial port output
4	Blue	TTL_RXD	Serial port input

5. Communication protocol

Table 2 TF40 Serial communication protocol

Communication protocol	UART
Baud rate	38400
Databits	8
Stopbits	1
parity bit	none

5.1 Serial command and data frame format

Command frame format:

1Byte	1Byte	2Bytes	2Bytes	2Bytes
Address code	Function code	Starting address	Number of registers（N）	CRC

Data frame format:

1Byte	1Byte	1Bytes	2*N Bytes	2Bytes
Address code	Function code	Byte count	Register value	CRC

5.2 Error response:

1Byte	1Byte	1Bytes	2Bytes
Address code	Error code	Exception code	CRC

Error code: 0x83 = Function code + 0x80

Exception code:

0x01: Wrong function code

0x02: Wrong starting address

0x03: Wrong number of registers

0x04: Wrong register value

CRC code calculation method:

CRC is calculated from the beginning of the address code to the byte before CRC code. The 8-bit byte of the CRC16 is first and the upper 8 bits are followed.

5.3 Register address and data format

Register address	Register description	Return value format
0x00 0x0F	Measuring distance	Measuring distance 4Bytes (Big-Endian)

Example:

Read measuring distance:

Description	Address	Function	Starting address	Number of register	CRC
Send:	0x01	0x03	0x00 0x0F	0x00 0x02	0xF4 0x08

Normal response (E.g. Measuring distance of 57.505m):

Description	Address	Function	Byte count	Register1	Register2	CRC
Response:	0x01	0x03	0x04	0x00 0x00	0xE0 0xA1	0x72 0x4B

Note: Measuring distance is four bytes long,

0x00 0x00 0xE0 0xA1 -> 0x0000E0A1 -> 57505mm

If a wrong starting address is sent, the response will be:

Description	Address	Error	Exception	CRC

Response:	0x01	0x83	0x02	0xC0 0xF1

5.4 Error Code

Registers 1 and 2 would represent error codes when an error occurs, possible error code are listed below:

Error code	Message
0xFF000000	Calculation error, measuring again
0xFE000000	The reflected signal is too weak or the measurement time is too long. The reflective surface need to be more reflective, or use cutting boards, white paper, etc.
0xFD000000	Glare detected, please avoid strong light
0xFC000000	Out of range! Please measure within it.

6. Instruction

Note that TF40 is only working on trigger mode, which can be set using instructions since TF40 will not work automatically with power on by default. One may send instructions to turn on or off laser for mounting and alignment. Also, the baud rate of the serial port is modifiable with instructions.

Table 3 TF40 Setting instructions

No.	Configuration	Instructions	Description
1	Trigger mode	01 03 00 0F 00 02 F4 08	Measure once
		01 03 00 01 00 02 95 CB	Successive measurement (5Hz)
		01 03 00 0A 00 02 E4 09	Stop successive measurement
2	Open beam	01 10 00 03 00 01 02 00 01 67 A3	If necessary, use this command to turn on the laser.

3	Close	01 10 00 03 00 01 02 00 00 A6 63	This command will turn off the laser when the laser is turned on.
4	baud rate (e.g. 115200)	01 10 00 00 00 02 04 00 01 C2 00 F3 0F	Six baud rates (4800, 9600, 19200, 38400, 57600, 115200) are supported on TF40 and please make sure using the right baud rate to switch it.

7. Attachment:

Calculate and check CRC16

```
/* CRC upper byte value table */
typedef unsigned                 char u8;
typedef unsigned short          int u16;
const u8 auchCRCHi[] = {
        0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
        0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
        0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
        0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
        0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
        0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
        0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
        0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
        0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
        0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
        0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
        0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
        0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
        0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
        0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
        0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
        0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
        0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
        0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
        0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
        0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
        0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
        0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
        0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
        0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
        0x80, 0x41, 0x00, 0xC1, 0x81, 0x40};
/* CRC lower byte value table */
const u8 auchCRCLo[] = {
        0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06,
        0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
        0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
        0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
        0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,
        0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
        0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3,
        0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
        0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
        0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,
```

```
    0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED,
    0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
    0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,
    0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
    0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
    0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
    0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E,
    0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
    0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71,
    0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
    0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
    0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
    0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B,
    0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
    0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42,
    0x43, 0x83, 0x41, 0x81, 0x80, 0x40};
//====================================================================
============
// @brief CRC16: This function return unsigned 16-bits integer CRC value
// @param[u8*] Start_Byte          The array that use to calculate CRC
// @param[u16] Num_Bytes           The number of bytes
//====================================================================
============

u16 CRC16(u8 *Start_Byte, u16 Num_Bytes)
{
    u8 uchCRCHi = 0xFF; // Initialize higher CRC byte
    u8 uchCRCLo = 0xFF; // Initialize lower CRC byte
    u16 uIndex;                 // Index in the table
    while (Num_Bytes--)
    {
        uIndex = uchCRCLo ^ *Start_Byte++; // Calculate CRC
        uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex];
        uchCRCHi = auchCRCLo[uIndex];
    }
    return (uchCRCHi << 8 | uchCRCLo);

}
```