Technofour UTUSB – Commands for FTDI

(Also, Refer Command packets UTUSB - FTDI.xls)

This document is divided into three sections.

A. Global variables and defines used in the document

B. Some useful functions to set various parameters

C. Recommended Initial Sequencing of functions in your project

A. Global variables and defines used in the document

```
#define T4UTUSB_AUTO_PRR          2000

#define T4UTUSB_MAX_RANGE_DATADEPTH  9

#define T4UTUSB_MAX_RANGE_PULSETYPE  3 // 0:Bipolar, 1:Unipolar, 2:Spike
#define T4UTUSB_PULSE_BIPOLAR              0
#define T4UTUSB_PULSE_UNIPOLAR             1
#define T4UTUSB_PULSE_SPIKE                2

#define T4UTUSB_MAX_RANGE_SAMPRATE   4 // 0:100MHz, 1:50MHz, 2:25MHz,
3:12.5MHz
#define T4UTUSB_100MHZ          0
#define T4UTUSB_50MHZ           1
#define T4UTUSB_25MHZ           2
#define T4UTUSB_12_5MHZ         3

#define T4UTUSB_MAX_RANGE_VPULSE    5 // 0:40V, 1:70V, 2:100V, 3:150V, 4:200V
#define T4UTUSB_40V             0
#define T4UTUSB_70V             1
#define T4UTUSB_100V            2
#define T4UTUSB_150V            3
#define T4UTUSB_200V            4

#define T4UTUSB_MAX_RANGE_PROBETYPE  3 // 0:PE, 1:TR, 2:Through
#define T4UTUSB_PR_TYPE_PE                 0
#define T4UTUSB_PR_TYPE_TR                 1
#define T4UTUSB_PR_TYPE_THRU               2

#define T4UTUSB_MAX_RANGE_LPF       5 // 0:27MHz, 1:15MHz, 2:10MHz, 3:6MHz,
4:4Mhz
#define T4UTUSB_LPF_27MHZ         0
#define T4UTUSB_LPF_15MHZ         1
#define T4UTUSB_LPF_10MHZ         2
#define T4UTUSB_LPF_6MHZ          3
#define T4UTUSB_LPF_4MHZ          4

#define T4UTUSB_MAX_RANGE_HPF       4 // 0:0.5MHz, 1:1Mhz, 2:2MHz, 3:4MHz
#define T4UTUSB_HPF_500KHZ             0
#define T4UTUSB_HPF_1MHZ               1
#define T4UTUSB_HPF_2MHZ               2
```

```
#define T4UTUSB_HPF_4MHZ              3

#define T4UTUSB_MAX_RANGE_CYCLES    4
#define T4UTUSB_CYCLE1                0
#define T4UTUSB_CYCLE2                1
#define T4UTUSB_CYCLE4                2
#define T4UTUSB_CYCLE8                3

#define T4UTUSB_MIN_PRR          40     // 40 Hz ie 255 *100 usec
#define T4UTUSB_MAX_PRR          2000 // 2000 Hz ie 5 *100 usec

#define T4UTUSB_MAX_DELAY             255
#define T4UTUSB_MAX_DATALENGTH      (64*1024) // Max range = 256 * 2^range =
64k

#define T4UTUSB_MAX_RANGE_AVG       6
#define T4UTUSB_AVG_1            0
#define T4UTUSB_AVG_2            1
#define T4UTUSB_AVG_4            2
#define T4UTUSB_AVG_8            3
#define T4UTUSB_AVG_16           4
#define T4UTUSB_AVG_32           5

#define T4UTUSB_MAX_RANGE_TRIGGER    3
#define T4UTUSB_TRIGGER_OFF             0
#define T4UTUSB_TRIGGER_INT             1
#define T4UTUSB_TRIGGER_EXT             2

#define T4UTUSB_MIN_DAC_GAIN       20.0
#define T4UTUSB_MAX_DAC_GAIN       50.0
#define T4UTUSB_PA_GAIN               20.0
#define T4UTUSB_BOOST_GAIN          16.0
#define T4UTUSB_MAX_GAIN   (T4UTUSB_PA_GAIN + T4UTUSB_MAX_DAC_GAIN +
T4UTUSB_BOOST_GAIN)  // ie 86.

FT_HANDLE fthandle; // Global variable
FT_STATUS ftstatus;
unsigned char buf[5];
DWORD byteswritten;

int lpf[T4UTUSB_MAX_RANGE_LPF] = {0, 7, 6, 5, 4}; // lpf bits for 27, 16, 10, 6, 4 MHz
int fbits[T4UTUSB_MAX_RANGE_SAMPRATE]  = {1, 2, 4, 8}; // freq = 0:100MHz,
1:50MHz, 2:25:MHZ, 3:12.5MHz
int v[T4UTUSB_MAX_RANGE_VPULSE]  = {0, 4, 5, 6, 7};// 40,70,100,150,200V
```

```
int nc[T4UTUSB_MAX_RANGE_CYCLES] = {1, 2, 4, 8}; // No of Cycles

double nMultiplier;
#define UT_CONSTATNT 12.276
#define USB_DATA_HEADER_SIZE   9
unsigned char header[USB_DATA_HEADER_SIZE] =  {0xff, 0x00, 0xaa, 0x55, 0xdd, 0x22,
0xbb, 0x44, 0x00}; // last byte represents size
```

B. Some useful functions to set various parameters

1. int OpenFTDI()

Parameters: None

Return :

T4UTUSB_ERR_OPEN : Error opening USB port
T4UTUSB_ERR_INIT : Error initializing UTUSB
T4UTUSB_OK : Success opening USB port

Description: Use this function in the beginning of your code to open USB port for data and parameter communication.

FTDI Code:

```
ftstatus = FT_OpenEx("UTUSB", FT_OPEN_BY_DESCRIPTION, &fthandle);
```

2. void CloseFTDI()

Parameters: None

Return : None

Description: Use this function while exiting the code.

FTDI Code:

```
FT_Close(fthandle);
```

3. int SetTriggerMode(int nTriggerMode, int nAutoPRR, int nPRRate);

(Refer Linux Command packets xls: 'T' Settings)

Parameters:

int nTriggerMode:

Valid Values:
T4UTUSB_TRIGGER_OFF
T4UTUSB_TRIGGER_INT
T4UTUSB_TRIGGER_EXT

int nAutoPRR:

Valid Values:
0 : Manual Pulse Repetition Rate
1 : Automatic Pulse Repetition Rate

int nPRRate: Value of Pulse Repetition Rate (valid only for non-Auto PRR)

Limits :T4UTUSB_MIN_PRR - T4UTUSB_MAX_PRR

Return :

T4UTUSB_ERR_LIMIT: Parameter out of limit

T4UTUSB_OK : Success
> 0 : FTDI error

Description: To set Trigger mode parameters

FTDI Code:

```
buf[0] = 'T';
buf[1] = nTriggerMode;
double TimePeriod = nPRRAuto ? (1.0 / T4UTUSB_AUTO_PRR) : (1.0 /nPRR);
 // PRR is expressed as freq
TimePeriod *= 1000000.; // expressed as micro sec
TimePeriod /= 100. ; // expressed in terms of 100 usec
buf[2] = (char)TimePeriod;
buf[3] = buf[4] = 0;
return FT_Write(fthandle, buf, 5, &byteswritten);
```

4. int SetParameters(int samp_rate, int nProbeType, int nLpf, int nHpf, int average);

(Refer Linux Command packets xls: 'S' Settings)

Parameters:

int samp_rate: Sampling Rate

Valid Values:

T4UTUSB_100MHZ : 100 MHz
T4UTUSB_50MHZ : 50 MHz
T4UTUSB_25MHZ : 25 MHz
T4UTUSB_12_5MHZ : 12.5 MHz

Limits: 0 - T4UTUSB_MAX_RANGE_SAMPRATE

int nProbeType: Probe Type

Valid Values:

T4UTUSB_PR_TYPE_PE :Pulse Echo
T4UTUSB_PR_TYPE_TR : Transmit Receive
T4UTUSB_PR_TYPE_THRU : Through Transmission

Limits: 0 - T4UTUSB_MAX_RANGE_PROBETYPE

int nLpf: Low Pass Filter

Valid Values:

T4UTUSB_LPF_27MHZ : 27 MHz
T4UTUSB_LPF_15MHZ : 15 MHz
T4UTUSB_LPF_10MHZ : 10 MHz
T4UTUSB_LPF_6MHZ : 6 MHz

T4UTUSB_LPF_4MHZ : 4 MHz

Limits: 0 - T4UTUSB_MAX_RANGE_LPF

int nHpf: High Pass Filter

Valid Values:

T4UTUSB_HPF_500KHZ : 500 KHz
T4UTUSB_HPF_1MHZ : 1MHz
T4UTUSB_HPF_2MHZ : 2KHz
T4UTUSB_HPF_4MHZ : 4Hz

Limits: 0 - T4UTUSB_MAX_RANGE_HPF

int average: Average number of A-Scans

Valid Values:

T4UTUSB_AVG_1 : No average
T4UTUSB_AVG_2 : Load data buffer with an average of 2 A-Scans.
T4UTUSB_AVG_4 : Load data buffer with an average of 4 A-Scans.
T4UTUSB_AVG_8 : Load data buffer with an average of 8 A-Scans.
T4UTUSB_AVG_16 : Load data buffer with an average of 16 A-Scans.
T4UTUSB_AVG_32 : Load data buffer with an average of 32 A-Scans.

Limits: 0 - T4UTUSB_MAX_RANGE_AVG

Return :

T4UTUSB_ERR_LIMIT: Parameter out of limit
T4UTUSB_OK : Success
> 0 : FTDI error

Description: Use this function to set sampling rate, average and filters

FTDI Code:

```
buf[0] = 'S';
buf[1] = fbits[samp_rate];
buf[2] = nProbeType == T4UTUSB_PR_TYPE_THRU ? T4UTUSB_PR_TYPE_TR :
nProbeType; // 0:PE, 1:TR

buf[3] = lpf[nLpf] | (nHpf << 3); // bits 2..0: lpf, bits 4.3: hpf
buf[4] = 0;
return FT_Write(fthandle, buf, 5, &byteswritten);
```

5. int SetPulse(int nPulseType, int nVpulse, int nCycles, double nProbefreq, int nDamping);

(Refer Linux Command packets xls: 'P' Settings)

Parameters:

int nPulseType: Bipolar / Unipolar or Spike trigger pulse

Valid Values:

T4UTUSB_PULSE_BIPOLAR : Bipolar

T4UTUSB_PULSE_UNIPOLAR : Unipolar

T4UTUSB_PULSE_SPIKE : Spike

Limits: 0 - T4UTUSB_MAX_RANGE_PULSETYPE

int nVpulse:

Valid Values:

T4UTUSB_40V : 40 Volt

T4UTUSB_70V : 70 Volts

T4UTUSB_100V : 100 Volts

T4UTUSB_150V : 150 Volts

T4UTUSB_200V : 200 Volts

Limits: 0 - T4UTUSB_MAX_RANGE_VPULSE

int nCycles: Number of Bipolar square cycles

Valid Values for nPulseType = T4UTUSB_PULSE_BIPOLAR

T4UTUSB_CYCLE1 : 1 cycle

T4UTUSB_CYCLE2 : 2 cycles

T4UTUSB_CYCLE4 : 4 cycles

T4UTUSB_CYCLE8 : 8 cycles

Limits: 0 - T4UTUSB_MAX_RANGE_CYCLES for Bipolar Pulse

Value = T4UTUSB_CYCLE1 for Non-Bipolar Pulse

int nProbefreq: Probe frequency in Hz

int nDamping: Disable / Enable Active damping

Limits: 0 - 1

Return :

T4UTUSB_ERR_LIMIT: Parameter out of limit

T4UTUSB_OK : Success

> 0 : FTDI error

Description: Use this function to set Trigger pulse related parameters.

FTDI Code:

```
buf[0] = 'P';
```

```
double T = 1000000000. / nProbefreq; // express as n sec
T = T / 10. ; // in terms of 10 n sec
T /= 2; // total period divided in high & low pulse time
if (nPulseType == T4UTUSB_PULSE_BIPOLAR)
        buf[1] = buf[2] = char(T);
if (nPulseType == T4UTUSB_PULSE_UNIPOLAR) {
        buf[1] = char(T);
        buf[2] = 0;
}
if (nPulseType == T4UTUSB_PULSE_SPIKE){
        buf[1] = 2;
        buf[2] = 0;
}
if (nDamping == 0)
        buf[3] = 0;
else
        buf[3] = (T + T)  > 255 ? 255 : (char)(T+T);
buf[4] = v[nVpulse] | (nc[nCycles] << 4);
return FT_Write(fthandle, buf, 5, &byteswritten);
```

6. int DelayRange(double nDelayMS, double nRangeMS, double nZeroMS, int nVelocity);

(Refer Linux Command packets xls: 'D' Settings)

Parameters:

double nDelayMS: Post trigger Delay in **microSec**

double nRangeMS: Range in **microSec**

double nZeroMS: Probe 'Zero' (Probe Delay) in **microSec**

int nVelocity: Velocity of sound in m / sec

Return :

T4UTUSB_ERR_LIMIT	: Parameter out of limit
T4UTUSB_OK	: Success
> 0	: FTDI error

Description: Use this function to set post trigger delay, range, probe zero (Probe delay) and velocity. Considering new velocity and probe zero, DLL converts range into no of samples and delay is converted into micro-sec. If they fall within limits, parameters are set.

FTDI Code:

```
buf[0] = 'D';
buf[1] = (har)nDelayMS + (char)nZeroMS;
buf[2] = nRangeIndex;
```

(Find nRangeIndex here. It has to be in between 0 to 9, depending upon data length. nRangeMS has to be converted to number of samples depending upon range in microsec and velocity. Then select data length, which is just greater than required range in samples
eg if range = 2000 sample, data length should be 2048 …refer xls sheet such that
data length = 256 * 2 ^ 3 = 2048 which is > 2000
so, here, nRangeIndex will be 3)

```
buf[3] = 0;
return FT_Write(fthandle, buf, 5, &byteswritten);
```

e.g if range is 3000 samples

7. int T4UTUSBSetGain(double nGain);

(Refer Linux Command packets xls: 'G' Settings)

Parameters:
double nGain: Gain Set by User
Limits: 0 – T4UTUSB_MAX_GAIN

Return :
T4UTUSB_ERR_LIMIT: Gain out of limit
T4UTUSB_OK : Success
> 0 : FTDI error

Description: Use this function to set gain

FTDI Code:

```
buf[0] = 'G';
double g1, m1;
unsigned short g2;
int boost = (nGain > (T4UTUSB_PA_GAIN + T4UTUSB_MAX_DAC_GAIN)) ? 1 : 0;
if (nGain <= T4UTUSB_MIN_DAC_GAIN) {
        buf[1] = buf[2] = 0;
        m1 = pow(10., double((T4UTUSB_PA_GAIN - nGain) / 20.));
        nMultiplier = 1. / m1;
}

if (nGain > T4UTUSB_MIN_DAC_GAIN && nGain <= T4UTUSB_MAX_GAIN) {
```

```
        g1 = (nGain - (boost ? (T4UTUSB_PA_GAIN + T4UTUSB_BOOST_GAIN) :
T4UTUSB_PA_GAIN));
        g2 = (unsigned short)(g1 * UT_CONSTANT);
        buf[1] = (g2 & 0x0300) >> 8;
        buf[2] = g2 & 0x00ff;
        nMultiplier = 1.0;
}
if (nGain > T4UTUSB_MAX_GAIN) {
        buf[1] = 0x03;
        buf[2] = (unsigned char)0xff;
        nMultiplier = pow(10., double((T4UTUSB_MAX_GAIN - nGain) / 20.));
}
buf[3] = boost;
buf[4] = 0;
return FT_Write(fthandle, buf, 5, &byteswritten);
```

8. long T4UTUSBGetData(int* databuf)

Parameters:
int* databuf: integer Pointer to get data from UTUSB
Malloc Minimum size = 128*1024 * sizeof(int)

Return :
T4UTUSB_ERR_NULLBUF: Null buffer pointer sent
T4UTUSB_ERR_DATA : Data Read Error
0 : Data not yet ready
> 0 : Data transferred to databuf, indicates data size

Description: Use this function to get data in user buffer.

FTDI Code:

```
DWORD numbytes, bytesread;
ftstatus = FT_GetQueueStatus(fthandle, &numbytes); // Bytes Available ?
if (ftstatus == FT_OK) {
        if (numbytes) { // numbytes are available
                ftstatus = FT_Read(fthandle, threadQ, numbytes, &bytesread); // Read
bytes in threadQ, bytesread = actual bytes read
                if (ftstatus == FT_OK) {
                        Read data here
                        For each triggering pulse, You will read number of bytes
                        depending upon range and delay
```

Each response will contain header of size
USB_DATA_HEADER_SIZE. First eight bytes should match with header[] as defined above.

Let int index = The [8]th byte of header corresponds to number of bytes of data.
Data_bytes = 256 * 2 ^ index

Data is 8 bit data.

So Read Data_bytes. Subtract 128 from each value so as to get signed data.

Multipy by nMultiplier.

After reading all samples, send next trigger pulse
If you want to average n no of samples, do it here

```
            }
        }
}
```

C. Recommended Initial Sequencing of functions in your project

```
Initialize buffer to hold UT A-Scan data
int* databuf;
databuf = new int[128*1024];
int nDataReady = 0;

// Open USB port
T4UTUSBOpen();

T4UTUSBSetGain(double nGain);

// While setting following parameters, set trigger mode 'Off'.

        T4UTUSBSetTriggerMode(T4UTUSB_TRIGGER_OFF);

        T4UTUSBSetPRR(nAutoPRR, nPRRate);

T4UTUSBSetPulse(int nPulseType, int nVpulse, int nCycles, double nProbefreq, int
nDamping);

T4UTUSBSetParameters(int samp_rate, int nProbeType, int nLpf, int nHpf, int average);

T4UTUSBSetDelayRange(double nDelayMM, double nRangeMM, double nZeroMS, int
nVelocity);

// Now set trigger mode of your interest – Internal or External
T4UTUSBSetTriggerMode( nTriggerMode);

#If you have set 'External' trigger mode, Set Trigger Pulse and get data by calling:
T4UTUSBExternalTrigger();
T4USBGetData(int *databuf);  till it returns all data or error

#else
// If you have set Internal trigger mode, Keep polling for UT signals, like this..

for (;;) {
        nDataReady = T4API long T4UTUSBGetData(int* databuf);
        if (nDataReady == T4UTUSB_ERR_NULLBEF) {
        //…. Null buffer pointer sent by user
        }
        if (nDataReady == 0) {
        //…. Data not ready
        }
```

```
        if (nDataReady < 0) {
        //…. Error reading
        }
        if (nDataReady > 0) {
        //…. Valid A-Scan data frame.. Process this
        }
        Sleep(2);
}

// When you quit, Close USB port, free the memory malloc'd for data buffer

T4API void T4UTUSBClose();
delete []databuf;
```