



Application Note

AN_180

FT232H MPSSE Example - USB Current Meter using the SPI Interface

Rev 1.0

Issue Date: 2011-10-17

This application note gives an example of interfacing an FT232H device to two SPI Analog-to-Digital Converters, to create a device which measures the VBUS voltage and current flowing in a USB connection. A Visual Basic 2008 program is used to control the MPSSE on the FT232H using D2XX commands, to take readings from the ADCs and display the results on the PC.

Use of FTDI devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify and hold harmless FTDI from any and all damages, claims, suits or expense resulting from such use.



Table of Contents

1	Introduction	2
1.1	FT232H Device	3
1.2	UM232H Module	3
1.3	Other Modules.....	3
2	Example Circuit	4
3	Software	9
3.1	Command Set.....	9
3.2	Initialisation Code.....	10
3.3	Measurement Code	10
3.4	Charting Code	11
3.5	Closing the Device	12
3.6	Code Examples.....	12
4	Using the Demo.....	15
4.1	Driver Installation.....	15
4.2	EEPROM Contents	15
4.3	Running the Demo	16
4.4	Additional Notes	17
5	Schematic	18
6	Contact Information.....	19
	Appendix A – References	20
	Document References	20
	Acronyms and Abbreviations.....	20
	Appendix B – List of Tables & Figures	21
	Appendix C – Revision History	22



1 Introduction

This application note gives an example of using the Multi-Protocol Synchronous Serial Engine (MPSSE) on the FT232H device. It uses D2xx commands to configure the MPSSE to interface with two Analog to Digital Converters (ADCs) using the Serial Peripheral Interface (SPI). It is controlled by software running on the PC written in Visual Basic 2008. The example includes the following items:

- Discover FTDI devices connected to the computer and open the desired device
- Configure the MPSSE for SPI communication using D2xx commands (SPI Mode 0 in this case)
- Use Chip-selects to communicate with multiple SPI devices
- Configure GPIO to read and write values on the GPIO pins

The application takes the form of a simple voltage and current meter, which can be connected in-line with a separate USB link to measure the VBUS voltage and how much current the USB peripheral is drawing from the host. An analog conditioning circuit is used to convert the voltage and current into suitable ranges to be measured by the ADCs. The FT232H then reads these values from the ADCs and sends them through a separate USB connection to be displayed on the Visual Basic program.

This application note details writing an application which controls the MPSSE directly, but a DLL is also available which contains functions to configure the MPSSE and communicate over SPI. Further details can be found at: <http://www.ftdichip.com/Support/SoftwareExamples/MPSSE.htm>

Figure 1 below shows the circuit connected between the laptop and a mouse. It measures the VBUS current and VBUS voltage in the connection to the mouse (red line) and sends it to the laptop on a separate USB connection (blue line). Two host ports are therefore required on the laptop.

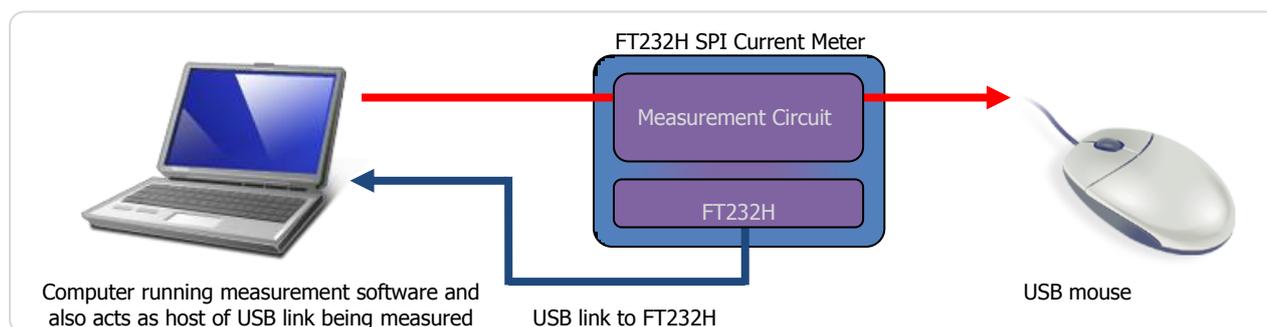


Figure 1: Measuring the connection between a laptop and USB mouse

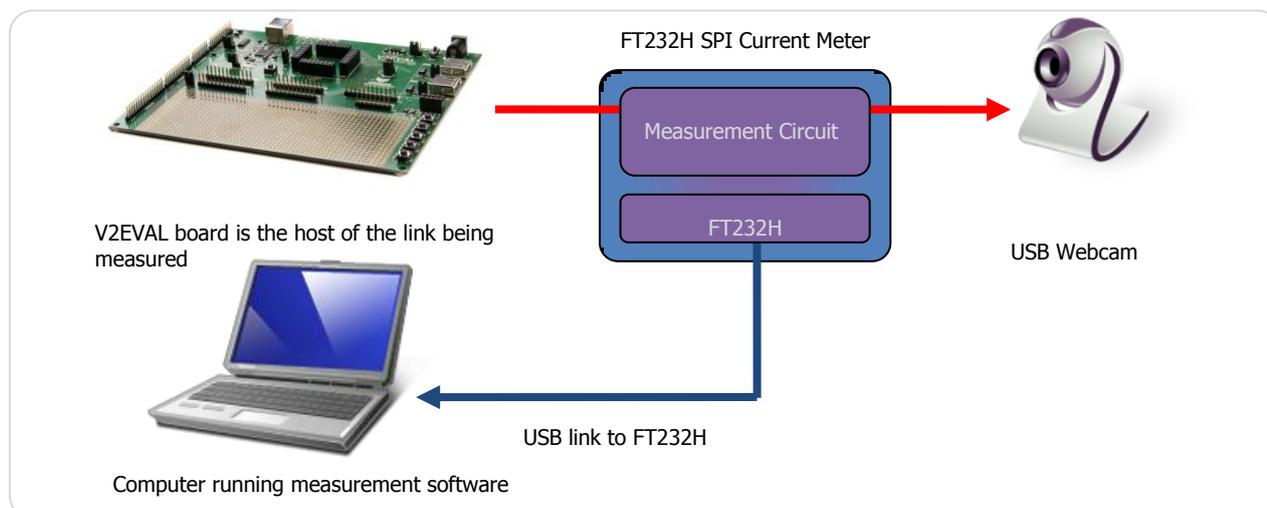


Figure 2: Measuring the current drawn from the V2-EVAL by the Webcam



The example in Figure 2 shows the circuit connected between the host port of a Vinculum-II V2EVAL (Rev 2) board and a webcam to measure the current drawn by the camera. The V2EVAL board contains a Vinculum II device, which is a 2-port USB host controller IC from FTDI. More information on this device, along with links to evaluation boards and application examples, can be found at the following page: <http://www.ftdichip.com/Products/ICs/VNC2.htm>

1.1 FT232H Device

This application uses the FT232H single-channel Hi-Speed device. However, the FT2232H and FT4232H devices also feature MPSSE engines and will be virtually identical in operation.

Some features of the FT232H include:

- Single channel USB to serial / parallel ports with a variety of configurations.
- Entire USB protocol handled on the chip. No USB specific firmware programming required.
- USB 2.0 Hi-Speed (480Mbits/Second) and Full Speed (12Mbits/Second) compatible.
- Various interfaces including
 - Asynchronous serial UART interface option with full hardware handshaking / modem signals (up to 12Mbaud).
 - Multi-Protocol Synchronous Serial Engine (MPSSE) to simplify synchronous serial protocol (USB to JTAG, I2C, SPI or bit-bang) design (Up to 30 Mbits/Sec).
 - Asynchronous 245 FIFO mode (up to 8 MByte/Sec).
 - Synchronous 245 parallel FIFO mode (up to 35 Mbyte/Sec).
 - Half duplex FT1248 interface with a configurable width, bi-directional data bus (1, 2, 4 or 8 bits wide).
 - Bit-bang Mode interface option with RD# and WR strobes
- FTDI's royalty-free Virtual Com Port (VCP) and Direct (D2xx) drivers eliminate the requirement for USB driver development in most cases.
- Highly integrated design includes +5V to +3.3V/+1.8V LDO regulator for VCORE, integrated POR
- Operation configuration mode and USB Description strings configurable in external EEPROM over the USB interface.
- Configurable I/O drive strength (4, 8, 12 or 16mA) and slew rate.
- Low operating and USB suspend current.
- Supports self-powered, bus powered and high-power bus powered USB configurations.
- Compact 48-pin Lead Free LQFP or QFN package

More details on this IC can be found at: <http://www.ftdichip.com/Products/ICs/FT232H.htm>

1.2 UM232H Module

In this example, the UM232H module is used. This is a development module based around the FT232H device. It can be plugged into a breadboard or IC socket and includes all components needed to get started immediately using the FT232H. For a more comprehensive feature list and technical details, see http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_UM232H.pdf

1.3 Other Modules

The C232HM cables from FTDI, which are based upon the FT232H device, can also be used for this application, in place of the UM232H. The flying leads provide all ADBus signals as well as a power output.

For further details, see: <http://www.ftdichip.com/Products/Cables/USBMPSSE.htm>

The FT2232H and FT4232H devices from FTDI also include MPSSE engines, and can be used instead of the FT232H. Evaluation modules are available for these devices at the links below:

http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_FT2232H_Mini_Module.pdf

http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_FT4232H_Mini_Module.pdf



2 Example Circuit

The hardware consists of a small board with the UM232H and the analog circuitry as shown in Figure 3 and Figure 4.



Figure 3 Hardware used for the FT232H USB Current Meter

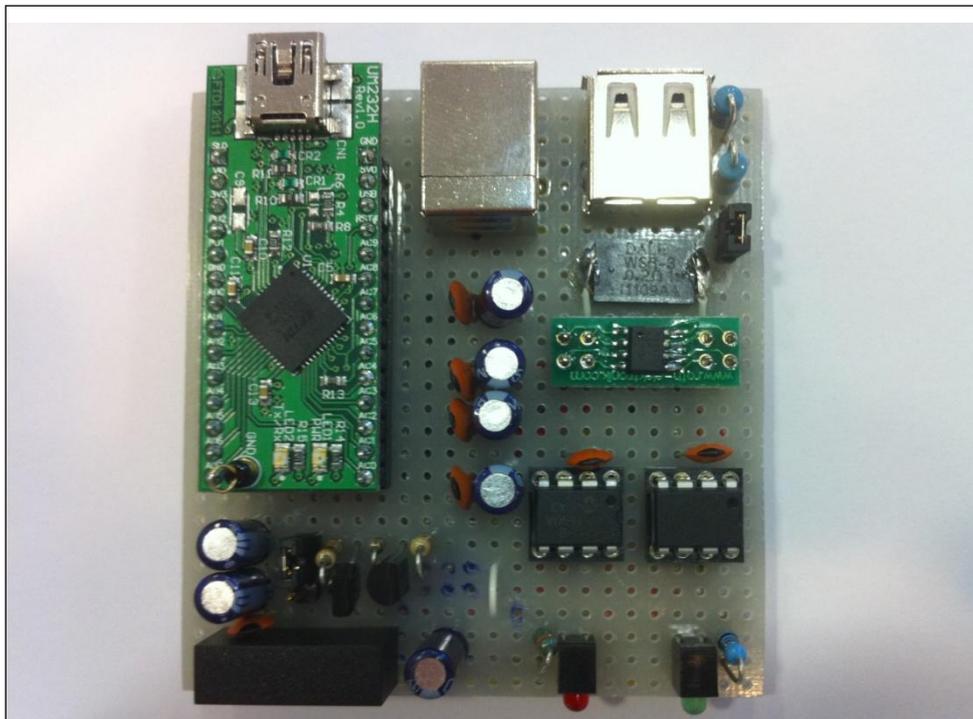


Figure 4 Prototype board used for the FT232H USB Current Meter



The block diagram of the hardware is shown below. A full schematic can be found in Section 5.

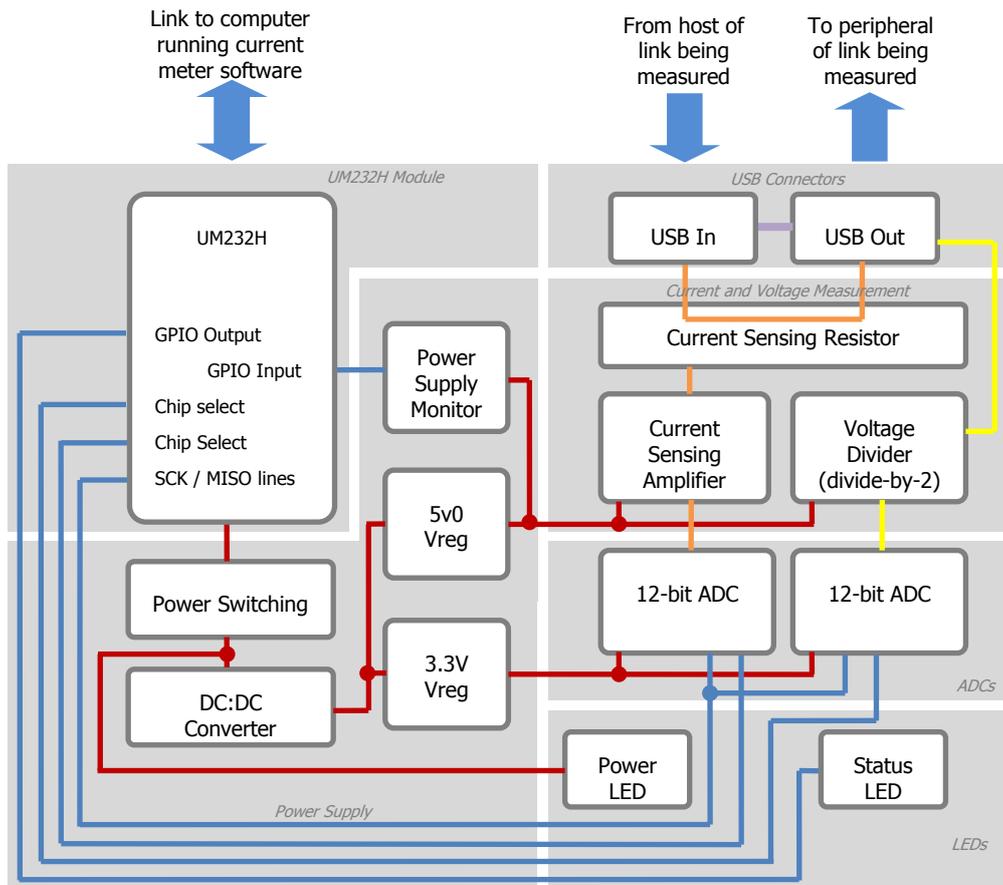


Figure 5 Block diagram of the hardware

The main components are:

UM232H Module

This evaluation module contains the FT232H device and all other circuitry required for the FT232H to operate, such as the mini-B USB connector, the crystal and an EEPROM device to store custom settings.

The datasheet at the link below contains further details of this module:

http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_UM232H.pdf

Power Supply

This design uses the FT232H PWREN# signal to enable power to the circuit once enumeration of the FT232H has completed. It also disables power when in USB suspend mode to meet the USB specification of 2.5mA max in suspend mode. It uses a P-channel MOSFET to switch the 5V supply to the DC:DC converter, which in turn powers the remainder of the circuit.

A 3-way jumper allows the power to the DC:DC converter to be always-on (VBUS) or switched by PWREN# (VBUS SW)

A DC:DC converter boosts the 5V power from the Power Switching circuit to 12V. Regulators are then used to drop this 12V supply down to create a 5V rail for the current measurement amplifier and 3.3V for the ADCs. This provides a more stable supply for the analog circuitry compared to powering the 5V circuitry and 3.3V regulator directly from the USB supply. Although not used here, it also provides an optional 12V supply for other circuitry if required.



A supply monitor IC is used to detect when the supply to the analog circuitry fails or drops to a lower than normal value. The Microchip MCP120-450DI/TO was used in this application. It pulls its open-collector output low in this case, which is connected to an input GPIO pin on the FT232H. In addition to indicating to the application that the measurements taken may be invalid, this also provides an example of reading a GPIO pin using the MPSSE. The datasheet for the MCP120 can be found in the Appendix A – References section.

ADCs

The circuit uses two 12-bit ADCs which are connected to the SPI interface of the UM232H. One ADC measures the output of the current sensing amplifier (which is adjusted via an output resistor to give a 0 to +3.3V range), and the other measures the VBUS voltage via a divider (which divides the normal VBUS level of 5V by two as the maximum ADC input level is 3.3V). Although dual channel ADCs are available, two separate devices were used intentionally to demonstrate the use of the chip select in this case.

The SPI Master In Slave Out (MISO) and Clock (SCK) lines are connected to both ADCs from the FT232H. Each ADC also has its own chip select (CS) line. Since the ADC does not require any data to be sent to it from the Master, the Master Out Slave In (MOSI) line is not connected. However, in other applications it would be connected to the appropriate pin of the peripheral devices.

SPI Mode 0 was used in this application, which means that data is clocked in on a positive edge and the clock idle state is low. The initial pin states of the FT232H and the MPSSE data commands were selected to implement this mode in the MPSSE.

The datasheet for the MCP3201-B ADC which was used in the prototype can be found in the Appendix A – References section.

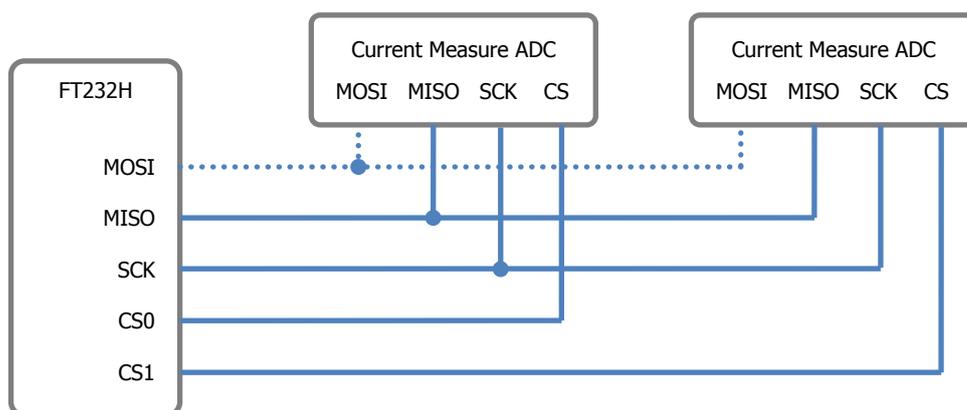


Figure 6 SPI connections between UM232H and ADCs

LEDs

A pin is used to drive the green Status LED on the board. In this example, the LED is driven by one of the upper bits of the same port used for SPI. It is turned on by driving the pin low, and must be supplied from 3.3V because this is the I/O level used on the pins of the UM232H module in this application. To avoid noise on the 3.3V rail used for the ADC, the LED is powered from the 3.3V output provided by the FT232H device (which has its own internal regulator and can provide 3.3V to external circuits).

The LED is turned on by the application software after the MPSSE is initialised and blinks off briefly each time a reading is taken. As well as indicating the status of the unit, it also presents an example of how to control GPIO through the MPSSE.

A red LED is powered from the 5V rail from the Power Switching circuit to indicate that the power is turned on.



Current and Voltage Measurement

The current measurement circuit consists of a current sensing resistor and current sensing differential amplifier. In this application, the Maxim MAX4172 was used. The datasheet can be found in the Appendix A – References section. This has further information on the device and recommendations for layout etc.

The current flows between the 5V (VBUS) pins of the USB 'In' and USB 'Out' connectors through the current sensing resistor, which converts the current flowing into a small voltage across the resistor. In this case, a 0.2R resistor is used which gives a range of between 0mV and 100mV depending on the current flowing. This resistor was chosen because 0mV – 100mV is the normal input range for the differential amplifier which is used to measure the voltage across the resistor. At 500mA flowing, $(0.2 \text{ Ohms} * 0.5 \text{ Amps}) = 100\text{mV}$ across the resistor.

This method provides a reasonably accurate current measurement but is not entirely un-intrusive since the resistor will cause a corresponding voltage drop in the supply to the USB peripheral being measured. The selection of a resistor value depends on the amplifier used, and is a trade-off between achieving good accuracy without causing excessive voltage drop to the peripheral being measured.

A jumper is provided to allow a multimeter to be connected in series with the unit to verify the current measured by the circuit.

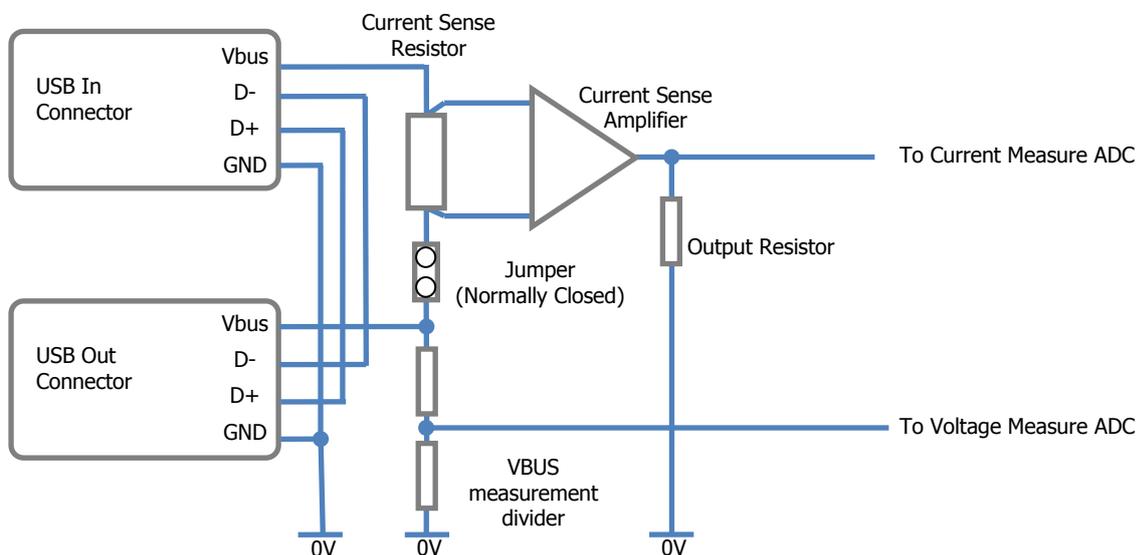


Figure 7 Current and Voltage Measurement

The amplifier outputs a current which is proportional to the voltage across the sense resistor. An external output resistor then converts this into a voltage which can be measured by the ADC. In this case the 3.3K resistor provides an output range of 0V to 3.3V.

The amplifier is designed for current sensing, since it can measure the voltage between two terminals of a resistor instead of a voltage with respect to ground. It can then convert this floating voltage difference into a ground referenced output which is independent of the common mode voltage at the input terminals. Some examples are given below:

USB Current	Voltage across Rsense (Voltage into Amplifier)	Amplifier Output Current	Voltage out of Amplifier (Voltage into ADC)	ADC Reading
0mA	0mV	0mA	0V	0x000
250mA	50mV	0.5mA	1.65V	0x800
500mA	100mV	1mA	3.3V	0xFFFF

Table 1 Current sensing examples



The voltage measurement uses a divide-by-2 resistor divider to convert the nominal 5V USB bus voltage into a voltage suitable for the ADC input (range 0 – 3.3V). The ADC will normally measure a voltage of 2.5V.

The measurement point is taken after the sense resistor so that the reading reflects the actual voltage presented to the peripheral being measured. Due to the drop across the resistor, the reading will be up to 100mV below the actual Vbus being output by the host port depending on the current being drawn, as will be observed in the voltage measurement in the application.

For both current and voltage measurements, the PC software calculates the actual value to be displayed based upon the ADC reading.

USB Connectors

The hardware has three USB connections in total.

The first is the mini-B female connector on the UM232H. It is used to connect the FT232H to the PC running the application which controls the MPSSE and displays the readings.

The other two are used to connect the hardware board in-line with an existing USB link. The Upstream, or In, connection is a USB type-B socket and a standard A-B USB cable is used to connect this to the host of the link being tested. The peripheral being measured then connects to the USB type A female socket on the hardware unit and will behave as if it was connected directly to the host.

The USB data lines between these connections should be kept as short as possible and equal in length, since they form part of the USB link between the host and peripheral, to avoid issues when measuring Hi-speed devices.



3 Software

The software was written in Visual Basic 2008. It is designed to run on the Windows OS but the same principle can be used when writing applications for Linux and Mac systems. The demo is provided with a zip file which contains an executable version of the software, and folders containing the source code (Visual Basic 2008), schematic and FT_Prog template for the EEPROM of the FT232H. This can be obtained from the link below:

<http://www.ftdichip.com/Support/SoftwareExamples/MPSSE/i-SPI.zip>

Note that the software and source code is provided as an example only and is not guaranteed or supported by FTDI. The code is designed to demonstrate interfacing to the FT232H device, and to improve readability it does not include the full set of error checking and performance enhancements that a final application would use.

The application software is shown in Figure 8. It displays the instantaneous current and voltage as values. A scrolling graph is used to give a profile of the current as it changes. A bar-graph is also provided which shows the instantaneous USB current.

In the screen-shot shown, a USB flash drive was connected (drawing approximately 55mA), a virus scan was then run on the drive causing the current to vary, and the 'safely remove hardware' option was then used (the current drops to 1mA as the drive was still connected to the port).

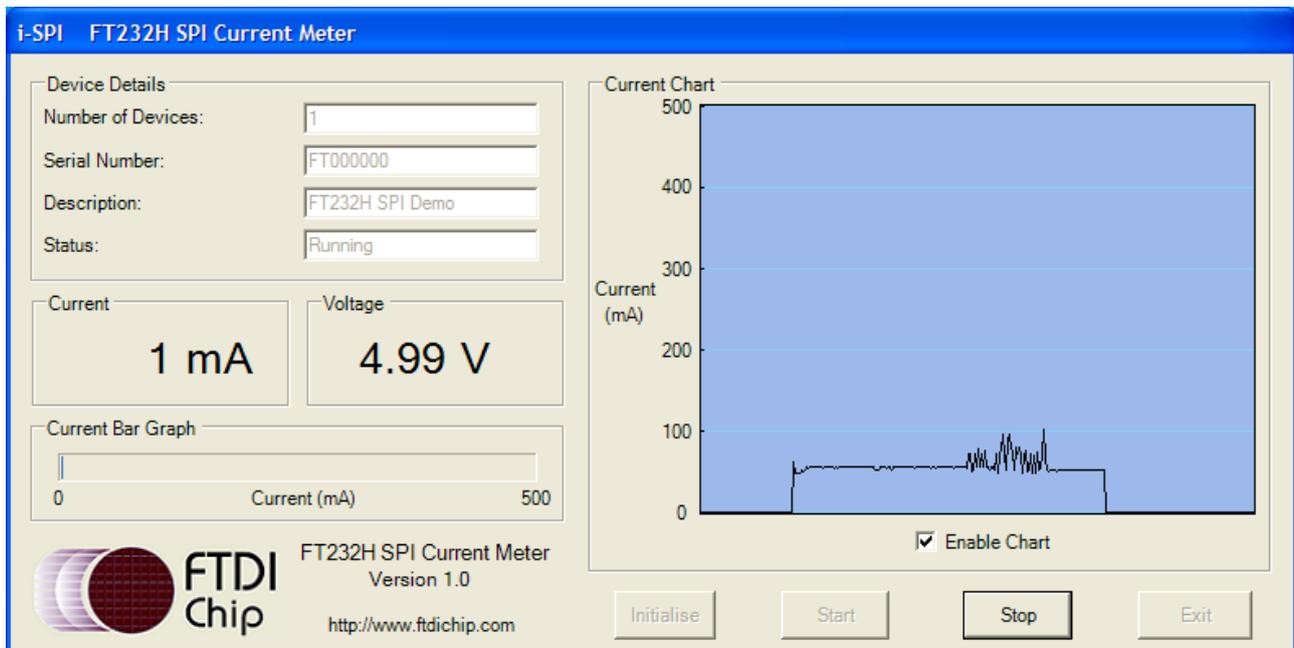


Figure 8

i-SPI software - User Interface of the Visual Basic application

3.1 Command Set

Some settings for the FT232H device itself are configured using standard D2xx commands. An example of using these commands to set the Latency Timer is shown in Section 3.6 (Code Examples) below. The commands are described in the D2xx Programmers Guide, which can be found at the following link:

[http://www.ftdichip.com/Support/Documents/ProgramGuides/D2XX_Programmer's_Guide\(FT_000071\).pdf](http://www.ftdichip.com/Support/Documents/ProgramGuides/D2XX_Programmer's_Guide(FT_000071).pdf)

Other settings use specific MPSSE commands once MPSSE mode is selected. An example of setting the Clock Divisor is shown in Section 3.6 (Code Examples) below. The full set of commands is described in Application Note AN_108 which can be found at the following link:

http://www.ftdichip.com/Support/Documents/AppNotes/AN_108_Command_Processor_for_MPSSE_and_MCU_Host_Bus_Emulation_Modes.pdf



Note that the example code performs each operation as a separate write to the FT232H. It is also possible to put multiple commands into a single buffer and then send that buffer. For example, the command to write the GPIO to enable Chip Select 0 and the command to read two bytes from the ADC could be written to a single buffer and then sent. This technique can be used to improve throughput if required.

3.2 Initialisation Code

This section of code (Button3_Click) runs when the Initialise button is pressed, to configure the device. The Initialisation routine is responsible for carrying out the tasks listed below:

- Open the required device:
 - Find out how many FTDI devices are connected (e.g. 3 devices found).
 - For each device found, get the description string and check if it is 'FT232H SPI Demo' (e.g. if 3 devices are found, use a loop to get the string for indexes 0 to 2). If a device with 'FT232H SPI Demo' is found, get its index number.
 - Open the required device by its index found in the above step
- Initialise the MPSSE
 - Reset the device
 - Set bit mode to MPSSE
 - Set the Latency Timer (left at the default of 16ms in this example, [see Section 3.6](#))
 - Set Timeouts (left at 0 for read and write, which disables timeouts)
 - Disable the /5 clock divider
 - Set the clock divisor register ([see Section 3.6](#))
- Check if MPSSE is operational:
 - Send a bad command 0xAA to the MPSSE to check if it responds correctly (expect it to return 0xFA followed by echoing the invalid command 0xAA)
 - Send a bad command 0xAB to the MPSSE to check if it responds correctly (expect it to return 0xFA followed by echoing the invalid command 0xAB)
- Ensure that the loop-back mode is disabled before sending any data
- Set the I/O pins to the required directions and states ([see Section 3.6 for an example](#))
- Update program status variables to show that Initialisation is now complete

3.3 Measurement Code

This section of code (Button1_Click) runs when the Start button is pressed, to take measurements. It uses commands to set/read the upper bits of the ADbus port which are used for the Chip Select outputs, PWRGOOD (supply voltage monitor) input and status LED output. It then uses data shifting commands to read in the data from the ADCs over the SPI bus.

The code runs in a loop until the Stop button is pressed.

- Set initial pin state with Clock low, both Chip Selects high (not asserted) and the Status LED on ([see Section 3.6 for an example](#))
- Read the PWRGOOD line ([see Section 3.6 for an example](#))
 - Send a Read Data Bits command
 - Read the returned byte
 - Mask off the required bit for the PWRGOOD input (bit 6)
 - Set Status to 'Power Supply Out of Range' if found to be low
- Set new pin state with CS0 low (to select ADC0 and start its conversion) and the Status LED off
- Read the result from ADC0 (current measurement)
 - Send command to clock bytes in to the MPSSE, number of bytes set to 0001 (= 2 bytes)
 - Read the two returned bytes ([see Section 3.6 for an example](#))
 - Calculate the current based upon the returned value
 - Store the current at the next location in a circular buffer and store the index of this location which is used when painting the scrolling chart on the screen
- Set new pin state with CS0 high again
- Set new pin state with CS1 low (to select ADC1 and start its conversion)
- Read the result from ADC1 (voltage measurement)
 - Send command to clock bytes in to the MPSSE, number of bytes set to 0001 (= 2 bytes)



- Read the two returned bytes (see Section 3.6 for an example)
- Calculate the voltage based upon the returned value
- Set new pin state with CS1 high again (back to idle state)
- Update the meter displays, bargraph and scrolling chart
- Go back to start of measurement loop (unless Stop button was pressed)

The measurement loop takes the current and voltage measurement data and updates the instantaneous values in the meter-style display and the bargraph. It also stores the current values in an array so that they can be plotted on the scrolling chart in a separate routine.

The Button4_Click routine runs what the Stop button is pressed. The only function of this routine is to set the Logging_Enabled variable to False so that the measurement loop will stop.

3.4 Charting Code

The chart painting subroutine (Panel1_Paint) plots the scrolling chart on the screen using a panel control. The entire chart is re-drawn each time a sample is taken from the ADCs.

The measurement routine puts the sample results into an array with 340 elements (one for each horizontal pixel of the chart). Each time a measurement is taken, it puts the result into the next available location in the array, in a form of circular buffer which helps to make the recording and display of the measurements more efficient. It also updates an index value which specifies which array element to use as the first data point for the chart plotting subroutine.

For example, when the measurement routine has added a new result in location 337, the chart will be drawn starting from element 338 as shown below. Because the last value measured is the last value to be plotted and is at the highest X value in the chart, the chart will scroll from right to left by one pixel each time a new point is added. To make a continuous line, each point is drawn as a line from the previous coordinate to the current coordinate.

Since the chart is 250 pixels high, each pixel is equivalent to 2mA. The Y axis is reversed and so a Y value of 250 is 0mA and a value of 0 is 500mA. This is handled in the application software.

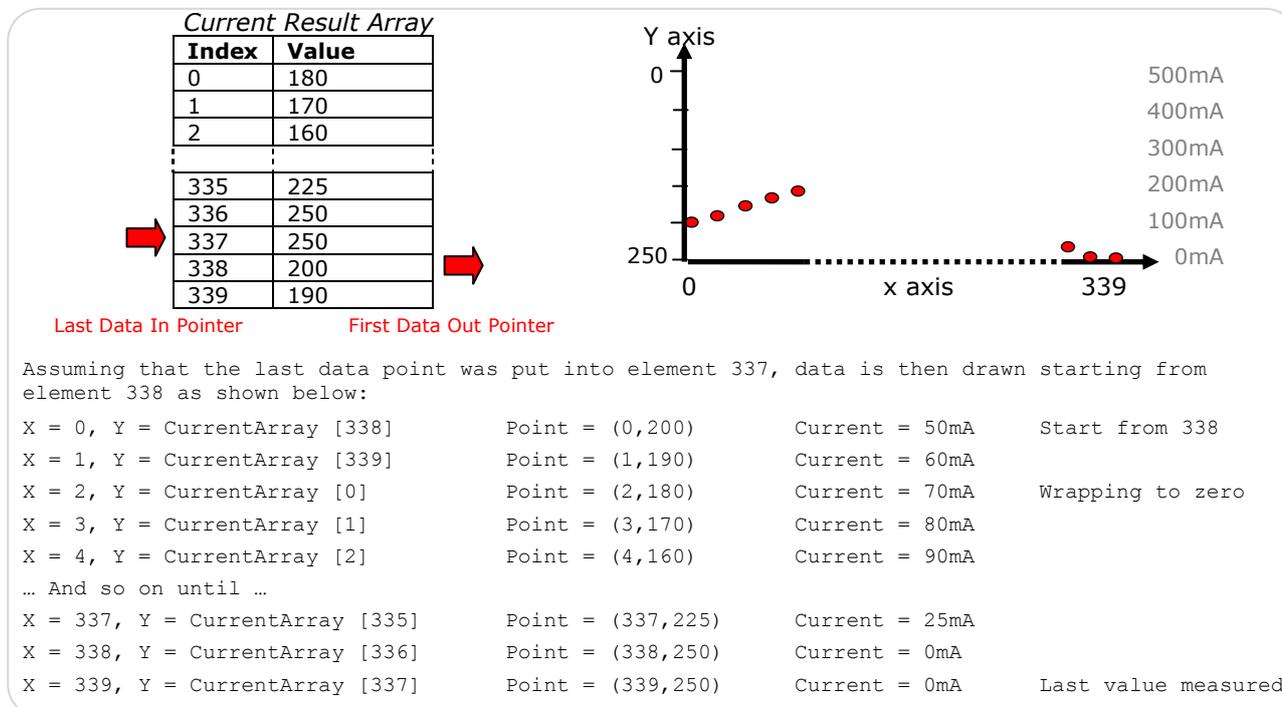


Figure 9 Buffer and Charting Routine

More complicated algorithms could be used to allow for re-sizing of the panel and make the drawing more efficient etc. but this would increase the complexity of the code beyond the scope of this application note.



3.5 Closing the Device

The Button2_Click routine is called when the Exit button is pressed, the application uses the FT_Close function to close the device if it had been previously initialised. It then updates the Status window to show 'Closed', and then closes the main window after a short delay.

3.6 Code Examples

The code sections below illustrate some of the commands used earlier in Section 3. The full code listing is included in the project zip file.

Using D2xx Commands (Setting Latency Timer)

This routine uses the D2xx command FT_SetLatencyTimer to configure the value of the Latency Timer on the FT232H. This timer sets the interval at which partially filled buffers will be flushed back to the PC.

```
' Set latency timer to 16ms
FT_Status = FT_SetLatencyTimer(FT_Handle, 16)

' Check if the function failed...
If FT_Status <> FT_OK Then
    CommLostError("SetLatencyTimer Failed")
    Exit Sub
End If
```

Using MPSSE Commands (Setting Clock Divisor)

This routine uses the MPSSE command 0x86 to set the clock divisor.

```
' set TCK/SK divisor to 50KHz
' See command processor for MPSSE application note for further information
' TCK      = 60MHz / ((1 + [ (0xValueH * 256) OR 0xValueL ] ) * 2)
'          = 60000000 / ((1 + [ ( 0x02 * 256) OR 0x57 ] ) * 2)
'          = 60000000 / ((1 + [ ( 512 + 87 ] ) * 2)
'          = 60000000 / ((1 + 512 + 87) * 2)
'          = 60000000 / 600 * 2
'          = 50000
'          = 50kHz

SendBuffer(0) = &H86      ' MPSSE command to set the divisor
SendBuffer(1) = &H57      ' Low value = 0x57
SendBuffer(2) = &H2       ' High value = 0x02

' Write buffer to the device
FT_Status = FT_Write_Bytes(FT_Handle, SendBuffer(0), 3, BytesWritten)

' Check if the function failed...
If FT_Status <> FT_OK Then
    CommLostError("Set Divider Failed")
    Exit Sub
End If

' Note: The application could also check BytesWritten to ensure that the
' correct number of bytes were sent
```

Setting Port Data and Direction

This routine sets the direction and state of the I/O pins. Three bytes are written to the MPSSE, which consist of the command itself, the pin values, and the pin directions.

```
' Bits assigned on FT232H AD bus
' 0      Out      SPI CLK (SCK)
' 1      Out      SPI DO (MOSI)
' 2      In       SPI DI (MISO)
' 3      Out      SPI CS0
' 4      Out      SPI CS1
' 5      Out      Not used (optional Error LED)
' 6      In       PWRGOOD input from power supply voltage monitor
```



```
'      7      Out      Status LED on when pin low (flashes on during SPI read)

' Note: Default state of CLK is 0 since using SPI Mode 0

' Set the required values for the pins
LED_State = LED_Status_On      ' Status LED on to indicate MPSSE configured 0x1x xxxx
CS_State = CS_None             ' No chip select lines asserted yet      xxx1 1xxx
                                ' State of the other SPI lines        x1xx x110
Pin_Values = Pin_Defaults Or LED_State Or CS_State      ' 0111 1110

SendBuffer(0) = &H80           ' MPSSE Command to set low bits of port
SendBuffer(1) = Pin_Values     '
SendBuffer(2) = Pin_Directions ' Defined above as 1011 1011

' Write command to the device
FT_Status = FT_Write_Bytes(FT_Handle, SendBuffer(0), 3, BytesWritten)

' Check if the function failed...
If FT_Status <> FT_OK Then
    CommLostError("Set Pin States Failed")
    Exit Sub
End If
```

Reading a Port Pin (Reading the PWRGOOD Pin)

This routine tells the MPSSE to read the state of the I/O pins. Command 0x81 is for a read of the AD0-AD7 pins. After the MPSSE runs this command, the chip will send a byte of data to the driver on the PC containing the pin states. A subroutine is then called to read this data (see [Read Data from the FT232H](#)).

```
' Read PWRGOOD line

' Put the command into the buffer
SendBuffer(0) = &H81      ' Command to read data bits (low byte) which is AD0 - AD7

' Send the buffer
FT_Status = FT_Write_Bytes(FT_Handle, SendBuffer(0), 1, BytesWritten)

' Check if the send was successful, display message box if not
If FT_Status <> FT_OK Then
    CommLostError("Read Pwrgood Failed")
    Exit Sub
End If

' Short delay
Sleep(10)

' Now read the result returned

Receive_Data(1)          ' See 'Read Data from the FT232H' below

If (BytesRead = 1) Then
    ' Mask off and test bit 6 of the port
    If (ReceiveBuffer(0) And &H40) = 0 Then
        ' If the line is low, the supply is out of range
        TextBox5.Text = "Power supply out of range"
    Else
        ' Otherwise running normally
        TextBox5.Text = "Running"
    End If
Else
    Receive_Error()
    Exit Sub
End If
```

Reading SPI Data

This command causes the MPSSE to clock in two bytes of data from the ADC. The 0x20 command clocks data in on the positive edge since this application uses SPI Mode 0. After the MPSSE runs this command



to clock in two bytes to the chip itself, the chip will send this data to the driver on the PC. A subroutine is then called to read this data (see [Read Data from the FT232H](#)).

```
' Send the command to clock the SPI data in
SendBuffer(0) = &H20      ' MPSSE command to read bytes in from SPI
SendBuffer(1) = &H1      ' 0001 means clock in two bytes
SendBuffer(2) = &H0      ' ADC returns two bytes

' Write the command to the FTDI device
FT_Status = FT_Write_Bytes(FT_Handle, SendBuffer(0), 3, BytesWritten)

' If function failed to run, update status box text and return
If FT_Status <> FT_OK Then
    CommLostError("SPI Read Failed")
    Exit Sub
End If
Sleep(10)

' Read the data from the FT232H
Receive_Data(2)          ' See 'Read Data from the FT232H' below

' Calculate the current from the two bytes received
' ADC outputs 15 bits XX0RRRRR RRRRRR where R is the result (bits 11:0)
' Reading 16 bits so ReceiveBuffer(0) = XX0RRRRR and ReceiveBuffer(1) = RRRRRRRX
If (BytesRead = 2) Then
    result16 = (((ReceiveBuffer(1) / 2) And &H7F) + ((ReceiveBuffer(0) And &H1F) * 128))
    current = ((result16 / 4096) * 500)  ' Scale result to 0 - 500mA
Else
    Receive_Error()
    Exit Sub
End If
```

Read Data from the FT232H

When the FT232H reads data from the ADC or from the port pins, it sends it to the PC to be held in the driver buffer. This subroutine reads the data from the driver buffer so that it can be used by the application.

```
Private Sub Receive_Data(ByVal BytesToRead As Integer)
    ' Function takes number of bytes to be read

    NumBytesInQueue = 0
    QueueTimeOut = 0

    ' Check how many bytes are in the buffer
    ' Wait until bytes are available, drop out of loop if not received within a count of 500
    While ((NumBytesInQueue < BytesToRead) And (QueueTimeOut < 500))
        FT_Status = FT_GetQueueStatus(FT_Handle, NumBytesInQueue)
        Sleep(1)
        QueueTimeOut = QueueTimeOut + 1
    End While

    ' Check if the loop above exited due to data or timing out
    If (QueueTimeOut > 499) Then
        BytesRead = 0
        Exit Sub
    Else
        ' Otherwise it was data being received
        BytesRead = 0

        ' Read required number of bytes from the driver
        FT_Status = FT_Read_Bytes(FT_Handle, ReceiveBuffer(0), BytesToRead, BytesRead)

        ' Check if the function failed...
        If FT_Status <> FT_OK Then
            BytesRead = 0
            Exit Sub
        End If

        ' calling function checks if correct number of bytes received
    End If
```



4 Using the Demo

4.1 Driver Installation

The first step is to ensure that the driver installed supports the FT232H. If the computer does not already have the FTDI D2xx driver installed, this can be obtained from the drivers page at the link below or from Windows Update if the computer is connected to the internet. It is recommended that older FTDI drivers are un-installed before installing the newer ones. For more information on installing the driver and un-installing older drivers which are already on your system, please refer to the FTDI driver installation guides.

The demo uses the D2xx driver. For Windows, this driver is part of the Combined Driver Model (CDM) driver. It is recommended that the latest driver available from the page below is used.

USB Drivers: <http://www.ftdichip.com/Drivers/D2XX.htm>

Installation Guides: <http://www.ftdichip.com/Support/Documents/InstallGuides.htm>

Note that the UM232H requires some pins on its DIP header to be connected in order to power up. The module will not power up or be recognised by the computer if it is connected to a USB port whilst none of the DIP pins are connected. As a minimum, the following pins must be connected:

- Pin 2 to Pin 3 Powers the I/O supply of the FT232H from the 3.3V output
- Pin 26 to Pin 27 Powers the 5V supply of the module from the USB 5V

4.2 EEPROM Contents

The FT232H uses an external EEPROM to hold configuration data. The UM232H can be used for SPI communication in its default configuration as supplied (since MPSSE mode is entered via a software command and does not need to be selected in the EEPROM), but some changes were made for this application as shown below.

USB String Descriptors

- The Product Description string was set to 'FT232H SPI Demo' so that the software can find the demo board if multiple FTDI devices are connected to the computer. *The example software provided requires this as it checks the string of each connected FTDI device in order to open it. However, the software could be changed to search for the device in other ways, for example by serial number or by simply opening the first device connected regardless of string or serial number.*
- The Serial number was set to FT000001

Hardware Specific -> Port A

- The Hardware was set to '245 FIFO'. *A command is used in the application software itself to switch to MPSSE mode as this mode cannot be directly selected in EEPROM.*
- The Driver was set to D2xx as the Virtual COM port is not required. *D2xx is always loaded but if the selection had been set to VCP then the VCP layer would have been loaded on top of the D2xx driver.*

A template for the EEPROM contents is included in the project zip file (created with FT_Prog 2.4).

The FT_Prog software can be used to modify the EEPROM contents. This utility uses the existing USB connection on the module to read and program the EEPROM. After connecting the module to the PC and installing the drivers, open the FT_Prog software (see the note on pin connections in section 4.1 if the module does not power up). The EEPROM can then be read, programmed with manual selections or a template file can be loaded and programmed. The FT_Prog utility and a user guide with further instructions are available at the link below:

http://www.ftdichip.com/Support/Utilities.htm#FT_Prog

4.3 Running the Demo

If the previous steps have already been completed, the demo can be run as follows:

- Connect the USB port of the UM232H on the hardware board to a USB port on the PC (the black cable in Figure 10) and the red power LED will illuminate. *If the LED does not illuminate, check that the drivers were installed correctly as per the installation guide in section 4.1.*



Figure 10 Connections to the hardware unit

- Connect a standard USB cable between a USB host and the 'USB In' connector of the hardware. (the silver cable in Figure 10)
- Start the demo application by double-clicking on the 'FT232H SPI Current Meter.exe' file as shown in Figure 11 below. *The application can be run by double-clicking the FT232H SPI Current Meter.exe file in the i-SPI folder within the project zip file. Note that when re-building the program from the source code provided, the exe file will be in \\i-SPI|i-SPI Source Code\bin.*

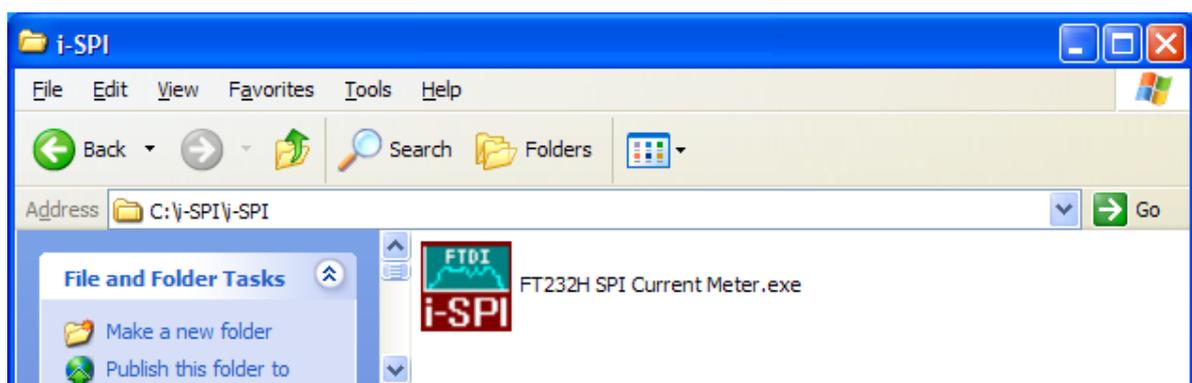


Figure 11 FT232H SPI Current Meter.exe file used to run the application

- Click the Initialise button and number of FTDI devices found will be shown. The buttons are shown in the screen shot in Figure 8. *With the current meter connected to the USB port, at least 1 device should be found. The count includes all FTDI devices currently connected to the computer (remember that an FT232H will show as 2 devices and an FT4232H will show as 4 devices).*



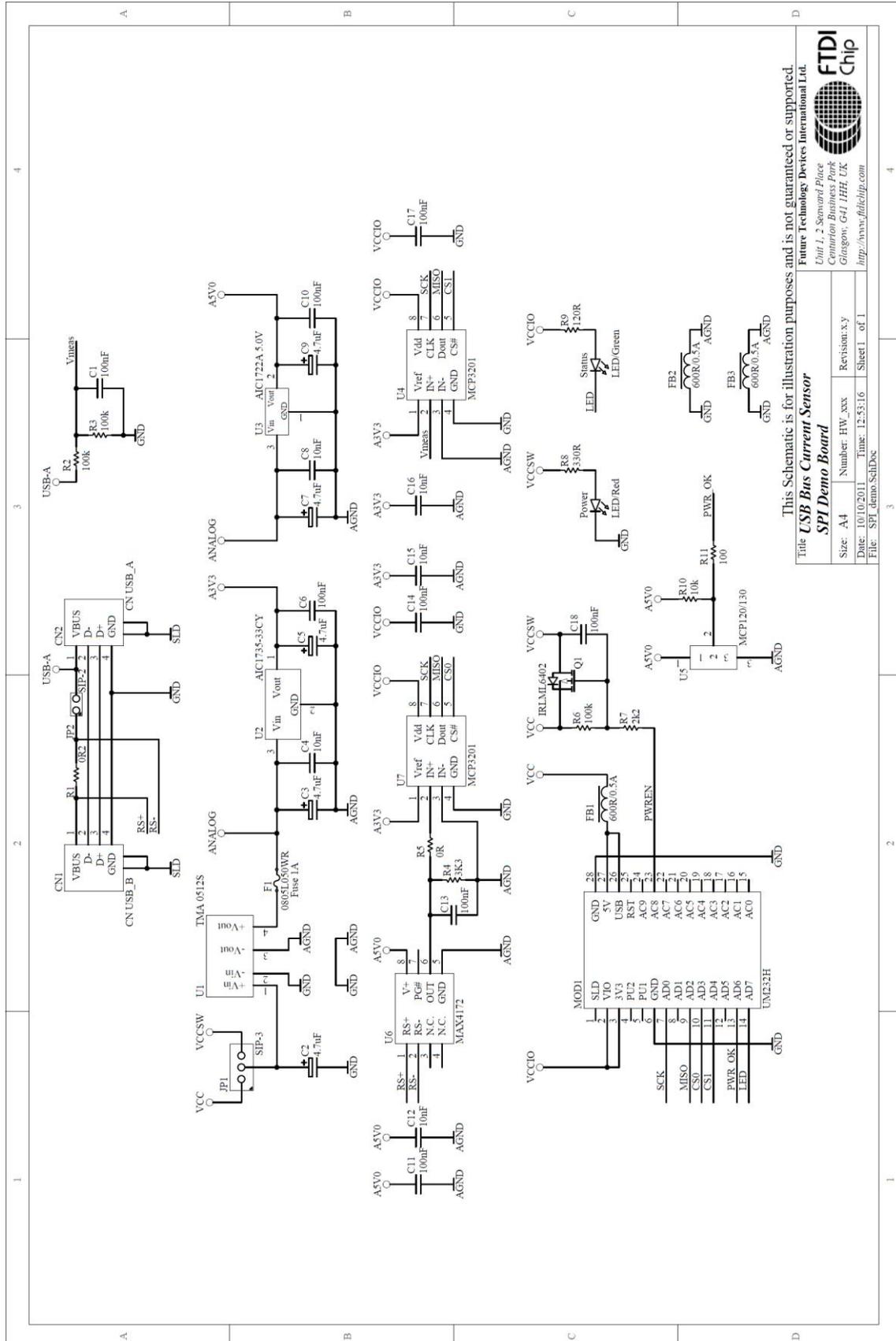
- If a device with correct string (FT232H SPI Demo) is found, it will be opened and the serial number will be displayed. *Otherwise, the Status box will show 'No SPI Devices Found'. In this case, check that the description string was correctly programmed using FT_Prog as shown in section 4.2.*
- The Green status light will illuminate to show that the MPSSE is now configured successfully and the software will show status as *Ready*. *The Start button in the application will also then be enabled since the hardware is ready to start taking measurements.*
- Click the Start button in the application and measurements will begin. The status window will show 'Running'. The green Status light will briefly blink off for each measurement, and the window will show the current and voltage on the meter-style readout. The scrolling chart and the current bargraph will also display the measured current in graphical form. *Note that if the voltage monitor IC detects a drop in the analog power supply inside the unit, the status will change to 'Power Supply out of Range' and measurements shown may be invalid due to the incorrect supply voltage.*
- Connect a USB peripheral to the 'USB Out' connector of the current meter hardware (an FTDI USB-serial converter is connected in Figure 10).
- Click the Stop button in the application to end the measurement run. The status window in the software will show 'Stopped'. The green Status light will stop blinking and stay on.
- Click Exit to close the FTDI device and exit the program. The Status window in the software will change to 'Closed' and the green Status LED will go off. The application will then close.

4.4 Additional Notes

It is recommended that the demo board is always powered (i.e. the UM232H USB port is connected to an active host port on the computer) whenever the USB link being measured is powered. This avoids having voltages at the inputs to the measurement circuits in the current meter hardware whilst the measurement circuits are un-powered.

The current meter configuration in this application note has a maximum measurement range of 500mA. It is therefore not suitable for measuring USB Battery Charging applications if they have a charge current of more than 500mA. The value of the sense resistor can be changed to 0.1 Ohms to provide a measurement range of 0 to 1A although this may reduce the accuracy at lower currents. More advanced systems with several current ranges could also be designed, for example by using the GPIO pins of the FT232H to control relays to set the range.

5 Schematic



This Schematic is for illustration purposes and is not guaranteed or supported.
Future Technology Devices International Ltd.
 Unit 1, 2 Seaward Place
 Centurion Business Park
 Glasgow, G41 1HH, UK
<http://www.ftdichip.com>

SPI Demo Board
 Size: A4 Number: HW_XXX Revision: XY
 Date: 10/10/2011 Time: 12:53:16 Sheet 1 of 1
 File: SPI_demo_SchDoc



6 Contact Information

Head Office – Glasgow, UK

Future Technology Devices International Limited
Unit 1, 2 Seaward Place, Centurion Business Park
Glasgow G41 1HH
United Kingdom
Tel: +44 (0) 141 429 2777
Fax: +44 (0) 141 429 2758

E-mail (Sales) sales1@ftdichip.com
E-mail (Support) support1@ftdichip.com
E-mail (General Enquiries) admin1@ftdichip.com

Branch Office – Hillsboro, Oregon, USA

Future Technology Devices International Limited (USA)
7235 NW Evergreen Parkway, Suite 600
Hillsboro, OR 97123-5803
USA
Tel: +1 (503) 547 0988
Fax: +1 (503) 547 0987

E-Mail (Sales) us.sales@ftdichip.com
E-Mail (Support) us.support@ftdichip.com
E-Mail (General Enquiries) us.admin@ftdichip.com

Branch Office – Taipei, Taiwan

Future Technology Devices International Limited
(Taiwan)
2F, No. 516, Sec. 1, NeiHu Road
Taipei 114
Taiwan, R.O.C.
Tel: +886 (0) 2 8791 3570
Fax: +886 (0) 2 8791 3576

E-mail (Sales) tw.sales1@ftdichip.com
E-mail (Support) tw.support1@ftdichip.com
E-mail (General Enquiries) tw.admin1@ftdichip.com

Branch Office – Shanghai, China

Future Technology Devices International Limited
(China)
Room 408, 317 Xianxia Road,
Shanghai, 200051
China
Tel: +86 21 62351596
Fax: +86 21 62351595

E-mail (Sales) cn.sales@ftdichip.com
E-mail (Support) cn.support@ftdichip.com
E-mail (General Enquiries) cn.admin@ftdichip.com

Web Site

<http://ftdichip.com>

Distributor and Sales Representatives

Please visit the Sales Network page of the [FTDI Web site](#) for the contact details of our distributor(s) and sales representative(s) in your country.

System and equipment manufacturers and designers are responsible to ensure that their systems, and any Future Technology Devices International Ltd (FTDI) devices incorporated in their systems, meet all applicable safety, regulatory and system-level performance requirements. All application-related information in this document (including application descriptions, suggested FTDI devices and other materials) is provided for reference only. While FTDI has taken care to assure it is accurate, this information is subject to customer confirmation, and FTDI disclaims all liability for system designs and for any applications assistance provided by FTDI. Use of FTDI devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify and hold harmless FTDI from any and all damages, claims, suits or expense resulting from such use. This document is subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document. Neither the whole nor any part of the information contained in, or the product described in this document, may be adapted or reproduced in any material or electronic form without the prior written consent of the copyright holder. Future Technology Devices International Ltd, Unit 1, 2 Seaward Place, Centurion Business Park, Glasgow G41 1HH, United Kingdom. Scotland Registered Company Number: SC136640



Appendix A – References

Document References

<http://www.ftdichip.com/Support/SoftwareExamples/MPSSE/i-SPI.zip>

[AN_108 Command Processor for MPSSE and MCU Host Bus Emulation Modes](#)

[D2XX Programmer's Guide](#)

[AN_135 MPSSE Basics](#)

[V2EVAL Rev2 Datasheet](#)

[MAX4172 Current Sense Amplifier Datasheet](#)

[MCP3201 ADC Datasheet](#)

[MCP120 Voltage Supervisor IC Datasheet](#)

Acronyms and Abbreviations

Terms	Description
USB	Universal Serial Bus
USB-IF	USB Implementers Forum
ADC	Analog to Digital Converter
MPSSE	Multi-Protocol Synchronous Serial Engine
SPI	Serial Peripheral Interface



Appendix B – List of Tables & Figures

List of Tables

Table 1	Current sensing examples	7
----------------	---------------------------------------	----------

List of Figures

Figure 1:	Measuring the connection between a laptop and USB mouse	2
Figure 2:	Measuring the current drawn from the V2-EVAL by the Webcam.....	2
Figure 3	Hardware used for the FT232H USB Current Meter	4
Figure 4	Prototype board used for the FT232H USB Current Meter	4
Figure 5	Block diagram of the hardware	5
Figure 6	SPI connections between UM232H and ADCs	6
Figure 7	Current and Voltage Measurement	7
Figure 8	i-SPI software - User Interface of the Visual Basic application	9
Figure 10	Connections to the hardware unit	16
Figure 11	FT232H SPI Current Meter.exe file used to run the application	16



Appendix C – Revision History

Document Title: AN_180 FT232H MPSSE Example - USB Current Meter using the SPI Interface
Document Reference No.: FT_000497
Clearance No.: FT_225
Product Page: <http://www.ftdichip.com/Products/ICs/FT232H.htm>
Document Feedback: [Send Feedback](#)

Revision	Changes	Date
1.0	Initial Release	2011-10-17