Application Note

AN_188

C232HM_MPSSE_Cable_in_USB_to_SPI_Interface

Document Reference No.: FT_000513

Version 1.0

Issue Date: 2011-10-17

This application note gives an example on how to configure the FTDI C232HM Hi-Speed USB 2.0 cable as a USB to Serial Peripheral Interface (SPI). All source code required is also included.

Future Technology Devices International Limited (FTDI)
Unit 1, 2 Seaward Place, Glasgow G41 1HH, United Kingdom
Tel.: +44 (0) 141 429 2777 Fax: + 44 (0) 141 429 2758
Web Site: http://ftdichip.com

Table of Contents

1 Introduction

This document gives an example of using the FTDI C232HM Hi-Speed USB cable by configuring the Multi Protocol Synchronous Serial Engine of the cable as a Serial Peripheral Interface (SPI). The cable contains the FTDI FT232H chip which may be configured to enable the MPSSE. A simple console application, written in "C" and using the libMPSSE-SPI API library, illustrates how the SPI interface is realized in software. The interface works as a SPI master, controlling slave SPI chips on the bus.

For further explanation and description of the MPSSE, please refer to application notesAN_135_MPSSE_BASICS and AN_108 Command Processor for MPSSE.

The sample application code is neither guaranteed nor supported by FTDI.

To demonstrate the SPI interface of the C232HM, the Microchip PICkit™ Serial SPI Demo Board is used. For detailed electrical and mechanical specification of C232HM cable, refer to the **C232HM MPSSE Cable** datasheet.

1.1 Scope

This document is designed as an introduction to using the FTDI C232HM cable with the SPI protocol. Not all SPI configuration modes will be discussed. It is assumed that the user is familiar with loading FTDI drivers and using Microsoft Visual Studio 2010.

1.2 PICkit Serial SPI Demo Board and CM232H

The SPI client demonstration hardware is made by Microchip, and features seven individually selectable SPI client devices. The SPI header pins can be easily connected to the CM232H's fly-wire sockets as shown in Figure 1.1 and Table 1.3.

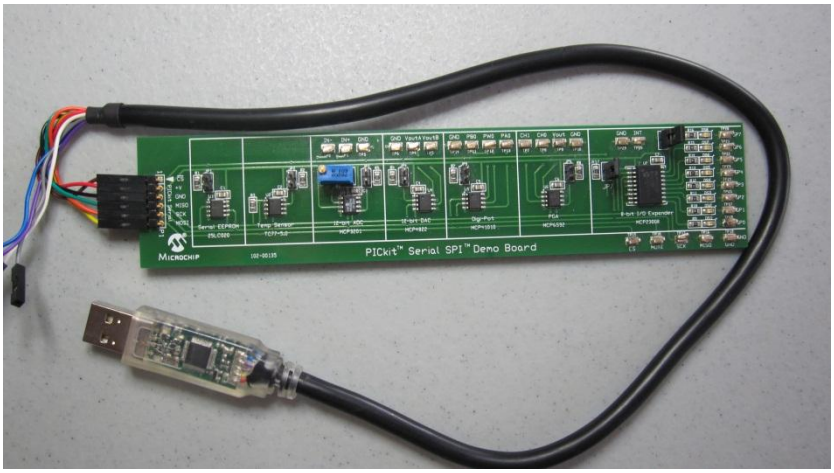

Figure 1.1 C232HM Cable and SPI Demo Board

1.3 Introduction to SPI Interface

The SPI (Serial Peripheral Interface) is a master/slave synchronous serial bus that consists of 4 signals. Both command and data signals are sent across the interface. The SPI master initiates all data transactions. Full duplex data transfers can be made up to 30 Mbits/sec with the C232HM. There is no fixed bit length in SPI. A generic SPI system consists of the following signals and is illustrated in Figure 1.2.

- Serial Clock (SCLK) from master to slave.
- Serial Data Out (also called Master Out Slave In or MOSI) from master.
- Serial Data In (also called Master In Slave Out or MISO) from slave.
- Chip Select (CS) from master.

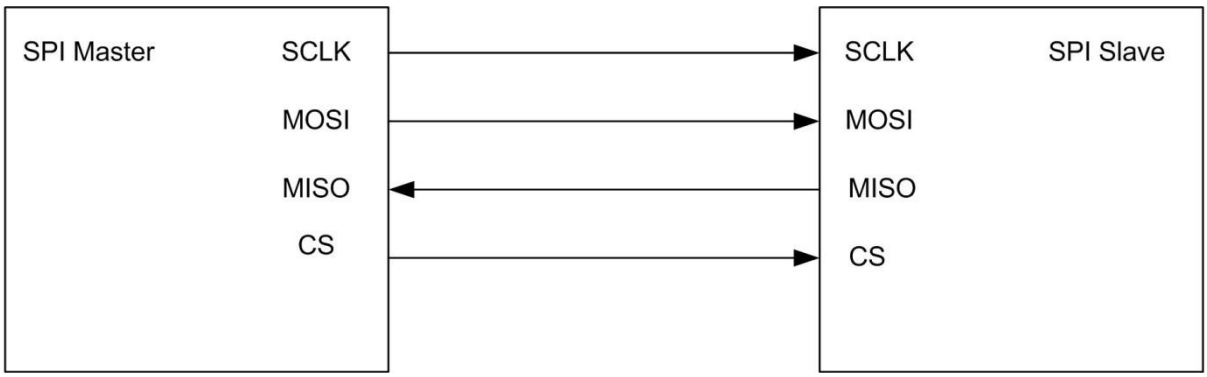

Figure 1.2 Basic SPI System

1.3.1 SPI Operating Modes

The C232HM always acts as the SPI master. In addition to SPI signals, the C232HM cable has 4 additional Chip Select lines (GPIOL [3:0]) that are used to access up to 5 SPI slave devices. These signals are controlled by libMPSSE-SPI API commands.

As SPI data is shifted out of the master and in to a slave device, SPI data will also be shifted out from the slave and clocked in to the master. Depending on which type of slave device is being implemented, data can be shifted MSB first or LSB first. Slave devices can have active low or active high chip select inputs. Figure 1.3 shows an example SPI timing diagram.

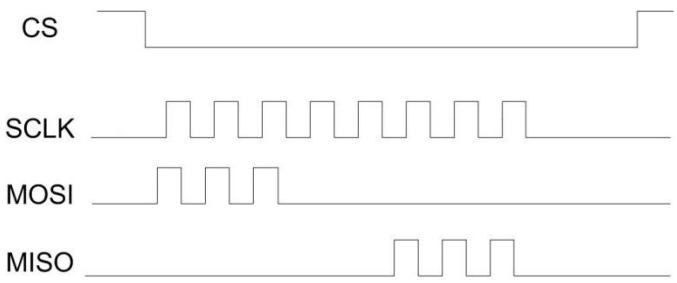

Figure 1.3 Example SPI Timing Diagram

The SPI device in the above example uses SPI Mode 0, with active low Chip Select

The SPI interface has 4 unique modes of clock phase (CPHA) and clock polarity (CPOL), known as Mode 0, Mode 1, Mode 2 and Mode 3. Table 1.1 summarizes these modes.

For CPOL = 0, the base (inactive) level of SCLK is 0.

In this mode:

- When CPHA = 0, data will be transferred on the rising edge of SCLK.
- When CPHA = 1, data will be transferred on the falling edge of SCLK

For CPOL =1, the base (inactive) level of SCLK is 1.

In this mode:

- When CPHA = 0, data will be transferred on the falling edge of SCLK
- When CPHA = 1, data will be transferred on the rising edge of SCLK.

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Table 1.1 Clock Phase/Polarity

It is worth noting that the SPI slave interface can be implemented in various ways. The C232HM cable can be configured to handle MODE 0 and MODE 2.

It is recommended that designers review the SPI Slave data sheet to determine the SPI mode implementation.

1.4 CM232H SPI/GPIO Interface and PICkit Interface

The CM232H IO lines used for SPI and GPIO are listed in Table 1.2.

Colour	C232HM PCB Pin Number	Name	Type	Description
Orange	2	SK	Output	**Serial Clock**
Yellow	3	DO	Output	**Serial Data Out (MOSI)**
Green	4	DI	Input	**Serial Data In (MISO)**
Brown	5	CS	Output	**Serial Chip Select**
Grey	6	GPIOL0	Output	**Additional (spare) Chip Select**
Purple	7	GPIOL1	Output	**Additional (spare) Chip Select**
White	8	GPIOL2	Output	**Additional (spare) Chip Select**
Blue	9	GPIOL3	Output	Additional (spare) Chip Select

Table 1.2 CM232H SPI & GPIO IOs

The CM232H Cable to PICkit connections are listed in Table 1.3**.**

CM232H Cable Flywire Colour	CM232H Cable Wire Function	PICKit SPI Connector Name
Brown	Serial Chip Select (CS)	CS
Red	VBUS	+V
Black	GND	GND
Green	Serial Data In (DI)	MISO
Orange	Serial Clock	SCK
Yellow	Serial Data Out (DO)	MOSI

Table 1.3 C232HM Cable to PICkit Connections

2 Application Code Details

This application demonstrates the SPI capability of the C232HM cable by interfacing it to the MCP23S08 8 bit I/O Expander chip. The application uses SPI to write data to the GPIO pins of the MCP23S08 chip. The GPIO outputs are connected to LEDs, which give a visual indication of the written data. Select the MCP23S08 and enable the LEDs by connecting jumpers JP7 and JP8 on the PICKit demo board.

2.1 SPI Client Details

The MCP23S08 chip uses SPI Mode 0 (SCLK base polarity is logic low, data is transferred on rising edge of SCLK). Chip select (CS) polarity is active low. The following registers are configured to drive the external LEDs:

- Control Byte
- I/O Direction (IODIR) register
- Output Latch (OLAT) register

When sending these commands, chip select is driven low.

The control byte is always the first byte sent to the MCP23S08. Pins A0 and A1 are hardware address pins that can be used to select multiple 23S08 devices (Of course, the spare CS lines can be used for the same purpose). The control byte also specifies if the operation is read or write. The format of the control byte (for the demo board) is shown in Table 2.1.

0	1	0	0	0	A1	A0	R/W

Table 2.1 MCP23S08 Control Byte

In the SPI demo board, pins A1 and A0 are pulled low – these bits are always zero. R/W is 0 for a write, and 1 for a read operation. For this demo, the value of the control byte will be 0x40.

Register Name	Address (Hex)	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IODIR	00	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0
OLAT	0A	OL7	OL6	OL5	OL4	OL3	OL2	OL1	OL0

Table 2.2 MCP23S08 Configuration Registers (subset)

To configure the data bus to outputs, the IODIR register needs to be set to all zeros.

The Output Latch register can be written with any single byte hex value. This value will be displayed in binary form on the external LEDs.

The control byte, IODIR and OLAT values are sent to the MCP23S08 as shown in Figure 2.1.

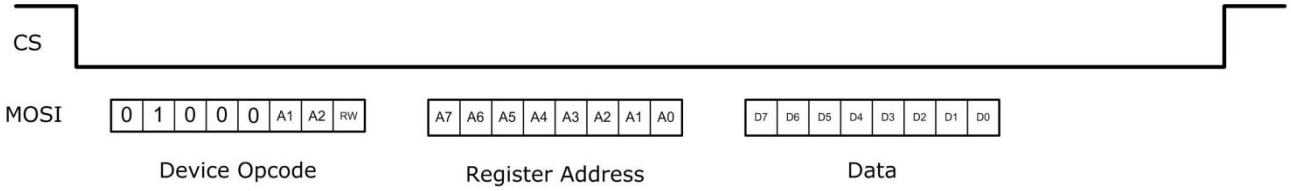

Figure 2.1 SPI Write Operation

To summarize, the application makes 2 SPI write operations to configure the 23S08 IO pins and to write data to these pins. Table 2.3 illustrates the register values that are used.

Device Opcode (control byte)	Register Address	Data	Function
0x40	0x00	0x00	Set GPIO to Outputs
0x40	0x0A	0xAA (for example)	Data to Display

Table 2.3 SPI Write Commands used in Demo Application

In the following code example in section 3, these commands are implemented in **C** as follows:

```
/*Write command to configure 23S08's IODIR register as all outputs*/
sizeToTransfer=24;  //3 Bytes Opcodes and Data
sizeTransfered=0;
buffer[0]=0x40; // Opcode to select device
buffer[1]=0x00; // Opcode for IODIR register
buffer[2]=0x00; // Data Packet - Make GPIO pins outputs
status = p_SPI_Write(ftHandle, buffer, sizeToTransfer, &sizeTransfered,
        SPI_TRANSFER_OPTIONS_SIZE_IN_BITS|
        SPI_TRANSFER_OPTIONS_CHIPSELECT_ENABLE|
        SPI_TRANSFER_OPTIONS_CHIPSELECT_DISABLE);

Sleep(5);  // short gap between writes

/* Write Data to 23S08's OLAT Register */
sizeToTransfer=24;  // 3 Bytes Opcodes and Data
sizeTransfered=0;
buffer[0]=0x40;  //  Opcode to select device
buffer[1]=0x0A;  //  Opcode for OLAT Register
buffer[2]=LEDS;  //  Data to write to OLAT & LEDs
status = p_SPI_Write(ftHandle, buffer, sizeToTransfer, &sizeTransfered,
        SPI_TRANSFER_OPTIONS_SIZE_IN_BITS|
        SPI_TRANSFER_OPTIONS_CHIPSELECT_ENABLE|
        SPI_TRANSFER_OPTIONS_CHIPSELECT_DISABLE);
```

In addition to the SPI write commands, SCLK frequency, SPI operation mode and Chip Select polarity need to be setup. These parameters are configured by the following code:

```
channelConf.ClockRate = 50000;

channelConf.configOptions = SPI_CONFIG_OPTION_MODE0| SPI_CONFIG_OPTION_CS_DBUS3|
SPI_CONFIG_OPTION_CS_ACTIVELOW;
```

The above commands setup the MPSSE cable to output SCLK at 50 KHz, use SPI Mode 0, assign Chip Select to pin DB3 and make Chip Select active low.

3 SPI Demo Code Listing

The source code can be obtained from the link below:

http://www.ftdichip.com/Support/SoftwareExamples/MPSSE/SPI_Daynamic.zip

Note that the software and source code is provided as an example only and is not guaranteed or supported by FTDI.

```
/* Project: libMPSSE-SPI
 * Module: SPI Sample Application - Interfacing CM232H Cable to MCP23S08 8 Bit I/O Expander
 * Refer to Applications Note AN_188 for operational details
 * FTDI-USA Apps Project
 * Revision History:
 * 1.0  Initial version
 *
 *
 */

#include<stdio.h>
#include<stdlib.h>
#ifdef _WIN32
#include<windows.h>
#endif

#include "libMPSSE_spi.h"
#include "ftd2xx.h"

#ifdef _WIN32
        #define GET_FUN_POINTER  GetProcAddress
#endif

#define SPI_DEVICE_BUFFER_SIZE          256
#define SPI_WRITE_COMPLETION_RETRY      10
#define CHANNEL_TO_OPEN                 0       // 0 for first available channel
#define SPI_SLAVE_0                             0
#define SPI_SLAVE_1                             1
#define SPI_SLAVE_2                             2

// Options-Bit0: If this bit is 1 then it means that the transfer size provided is in bytes
#define SPI_TRANSFER_OPTIONS_SIZE_IN_BYTES                      0x00000001
// Options-Bit0: If this bit is 1 then it means that the transfer size provided is in bytes
#define SPI_TRANSFER_OPTIONS_SIZE_IN_BITS                       0x00000001
// Options-Bit1: if BIT1 is 1 then CHIP_SELECT line will be enables at start of transfer
#define SPI_TRANSFER_OPTIONS_CHIPSELECT_ENABLE          0x00000002
// Options-Bit2: if BIT2 is 1 then CHIP_SELECT line will be disabled at end of transfer
#define SPI_TRANSFER_OPTIONS_CHIPSELECT_DISABLE                 0x00000004

typedef FT_STATUS (*pfunc_SPI_GetNumChannels)(uint32 *numChannels);
pfunc_SPI_GetNumChannels p_SPI_GetNumChannels;
typedef FT_STATUS (*pfunc_SPI_GetChannelInfo)(uint32 index, FT_DEVICE_LIST_INFO_NODE *chanInfo);
pfunc_SPI_GetChannelInfo p_SPI_GetChannelInfo;
typedef FT_STATUS (*pfunc_SPI_OpenChannel)(uint32 index, FT_HANDLE *handle);
pfunc_SPI_OpenChannel p_SPI_OpenChannel;
typedef FT_STATUS (*pfunc_SPI_InitChannel)(FT_HANDLE handle, ChannelConfig *config);
pfunc_SPI_InitChannel p_SPI_InitChannel;
typedef FT_STATUS (*pfunc_SPI_CloseChannel)(FT_HANDLE handle);
pfunc_SPI_CloseChannel p_SPI_CloseChannel;
typedef FT_STATUS (*pfunc_SPI_Read)(FT_HANDLE handle, uint8 *buffer, uint32 sizeToTransfer, uint32
*sizeTransfered, uint32 options);
pfunc_SPI_Read p_SPI_Read;
typedef FT_STATUS (*pfunc_SPI_Write)(FT_HANDLE handle, uint8 *buffer, uint32 sizeToTransfer, uint32
*sizeTransfered, uint32 options);
pfunc_SPI_Write p_SPI_Write;


typedef FT_STATUS (*pfunc_SPI_IsBusy)(FT_HANDLE handle, bool *state);
pfunc_SPI_IsBusy p_SPI_IsBusy;
```

```
uint32 channels;
uint8  LEDS = 0x00;
FT_HANDLE ftHandle;
ChannelConfig channelConf;
uint8 buffer[SPI_DEVICE_BUFFER_SIZE];

FT_STATUS write_byte()
// write_byte function configures IODIR and OLAT Registers

{
        uint32 sizeToTransfer = 0;
        uint32 sizeTransfered=0;
        bool writeComplete=0;
        uint32 retry=0;
        bool state;
        FT_STATUS status;

        // Write command to configure 23S08's IODIR register as all outputs
        sizeToTransfer=24;  // 3 Bytes Opcodes and Data
        sizeTransfered=0;
        buffer[0]=0x40; // Opcode to select device
        buffer[1]=0x00; // Opcode for IODIR register
        buffer[2]=0x00; // Data Packet - Make GPIO pins outputs
        status = p_SPI_Write(ftHandle, buffer, sizeToTransfer, &sizeTransfered,
                SPI_TRANSFER_OPTIONS_SIZE_IN_BITS|
                SPI_TRANSFER_OPTIONS_CHIPSELECT_ENABLE|
                SPI_TRANSFER_OPTIONS_CHIPSELECT_DISABLE);


        Sleep(5);  // short gap between writes

        // Write Data to 23S08's OLAT Register
        sizeToTransfer=24;  // 3 Bytes Opcodes + Data
        sizeTransfered=0;
        buffer[0]=0x40;  //  Opcode to select device
        buffer[1]=0x0A;  //  Opcode for OLAT Register
        buffer[2]=LEDS;  //  Data to write to OLAT & LEDs
        status = p_SPI_Write(ftHandle, buffer, sizeToTransfer, &sizeTransfered,
                SPI_TRANSFER_OPTIONS_SIZE_IN_BITS|
                SPI_TRANSFER_OPTIONS_CHIPSELECT_ENABLE|
                SPI_TRANSFER_OPTIONS_CHIPSELECT_DISABLE);

        return status;
}


int main()
{
#ifdef _WIN32
#ifdef _MSC_VER
        HMODULE h_libMPSSE;
#else
        HANDLE h_libMPSSE;
#endif
#endif


FT_STATUS status;

uint8 address=0;

// Setup SCLK, SPI Mode, Latency Timer, Chip Select Pin and Chip Select Polarity
channelConf.ClockRate = 50000;
channelConf.LatencyTimer= 255;
channelConf.configOptions = SPI_CONFIG_OPTION_MODE0| SPI_CONFIG_OPTION_CS_DBUS3|
SPI_CONFIG_OPTION_CS_ACTIVELOW;

// Direction and Value of GPIO Pins (for dir: 0=in, 1=out)
channelConf.Pin = 0x00000000;

// Load libMPSSE
#ifdef _WIN32
```

```
#ifdef _MSC_VER
h_libMPSSE = LoadLibrary(L"libMPSSE.dll");

#endif
#endif// init function pointers
p_SPI_GetNumChannels=(pfunc_SPI_GetNumChannels)GET_FUN_POINTER(h_libMPSSE,
"SPI_GetNumChannels");
p_SPI_GetChannelInfo = (pfunc_SPI_GetChannelInfo)GET_FUN_POINTER(h_libMPSSE, "SPI_GetChannelInfo");
p_SPI_OpenChannel = (pfunc_SPI_OpenChannel)GET_FUN_POINTER(h_libMPSSE, "SPI_OpenChannel");
p_SPI_InitChannel = (pfunc_SPI_InitChannel)GET_FUN_POINTER(h_libMPSSE, "SPI_InitChannel");
p_SPI_Read = (pfunc_SPI_Read)GET_FUN_POINTER(h_libMPSSE, "SPI_Read");
p_SPI_Write = (pfunc_SPI_Write)GET_FUN_POINTER(h_libMPSSE, "SPI_Write");
p_SPI_CloseChannel = (pfunc_SPI_CloseChannel)GET_FUN_POINTER(h_libMPSSE,"SPI_CloseChannel");

status = p_SPI_GetNumChannels(&channels);
printf("Number of available SPI channels = %d\n",channels);

status = p_SPI_OpenChannel(CHANNEL_TO_OPEN,&ftHandle);

status = p_SPI_InitChannel(ftHandle,&channelConf);
printf("Enter a hex value to display in binary -  ");

//read in a hex value from standard input
scanf_s("%x" , &LEDS);
printf("LED Display = %x   \n",LEDS);

// Call Write Byte Function to activate LEDs on GPIO pins
write_byte();

printf("End of SPI Demo");
status = p_SPI_CloseChannel(ftHandle);


}
```

4 Summary

The hardware and source code described in this application note provide a starting point for developing applications to enable USB to SPI communication using the FTDI C232HM MPSSE cable and a wide variety of SPI based client devices.

The C232HM MPSSE cable and the PICKit SPI Demo board are both available from Allied Electronics, Digi-Key, Future Electronics and Mouser Electronics.

The source code is available as a Microsoft Visual Studio 2010 project.

Visual Studio 2010 Express Website

Download source available from:

http://www.ftdichip.com/Support/SoftwareExamples/MPSSE/SPI_Daynamic.zip

5 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.

Appendix A – References

Document References

C232HM Datasheet, FTDI Ltd.

FT232H Datasheet, FTDI Ltd.

AN_178 User Guide for LibMPSSE-SPI, FTDI Ltd.

AN_135 MPSSE Basics, FTDI Ltd

AN_108 Command Processor for MPSSE, FTDI Ltd.

PICKit SPI Demo Board Users Guide, Microchip Inc.

MCP23008/23S08 Datasheet, Microchip Inc.

AN_972 I/O Expansion using the MCP23X08, Microchip Inc.

Acronyms and Abbreviations

Terms	Description
USB	Universal Serial Bus
SPI	Serial Peripheral Interface
MPSSE	Multi-Protocol Synchronous Serial Engine
IODIR	IO Pin Direction register (MCP23S08)
OLAT	Output Latch register (MCP23S08)
PICKit	Demonstration board made by Microchip

Appendix B – List of Tables & Figures

List of Tables

List of Figures

Appendix C – Revision History

Document Title:	AN_188 Using MPSSE Cable as USB to SPI Interface
Document Reference No.:	DocNo000513
Clearance No.:	FTDI# 224
Product Page Link:	C232HM MPSSE Cable Product Page
Document Feedback:	Send Feedback

Revision	Changes	Date
1.0	Initial Release	2011-10-17