Mini-Circuits®

Programming Manual

For the

USB/RS232 to SPI Converter

Contents

1- Overview

This programming Manual is intended for customers wishing to create their own interface for Mini-Circuits' USB/RS232 to SPI Converters.

Mini-Circuits offers support for USB Portable Test Equipment (PTE) in Windows® and Linux® Operating Systems, in a for variety of programming environments including third-party applications such as LabVIEW® and MATLAB® through .NET assembly and ActiveX® Controls to write your own customized control applications.

Mini-Circuits' CD package Includes: GUI program installation, DLL Objects 32/64 bit, Linux Support, project examples for 3RD party software and Documents. The latest CD version is available for download at http://www.minicircuits.com/support/software_download.html , see Figure 1.

RS232/USB To SPI

Product Name	Version	Download	Description / Instructions	Models Supported
RS232/USB To SPI Converter - Setup	A0	Download	RS232/USB To SPI GUI program for Windows 32/64 bit - Latest Version - Setup.	
RS232/USB To SPI Converter - CD	A0	Download	Latest Version of the entire RS232/USB to SPI CD: GUI program, DLL COM Objects 32/64 bit, Linux Support and Documents. When extracting the files after download, keep the folder names.	
MCL_RS232_USB_To_SPI.dll	May 1, 2011	Download	Dll - ActiveX com object file. Registering to Windows is required. Recommended for 32 bit programming.	
MCL_RS232_USB_To_SPI_64.dll	May 1, 2011	Download	Dll - .NET Class Library. Recommended for 64/32 bit programming.	
Programming Manual	May 1, 2012	Download	PDF File: Detailed Guide for Programmers.	
Project Examples	May 1, 2012	Download	Projects Examples for several Programming environments such as: VB6, VB.NET, C#, C++, Delphi, LabView, Matlab, LINUX. When extracting the Zip file after download: keep the folder names.	RS232/USB-SPI RS232/USB-SPI-N

Figure 1 – Download Screen

2 - Operating in a Windows® Environment 32/64Bits OS with USB HID Support

The DLL Object (Dynamic Link Library) - Concept:

Dynamic **L**ink **L**ibrary is Microsoft's implementation of the shared library concept in the Microsoft Windows® environment.

DLLs provide a mechanism for shared code and data, allowing a developer of shared code/data to upgrade functionality without requiring applications to be re-linked or recompiled.

Mini-Circuits' CD package provides DLL Objects in order to allow your own Software Application to interface with the functions of the Mini-Circuits' USB Portable Test Equipment hardware, see Figure 2.

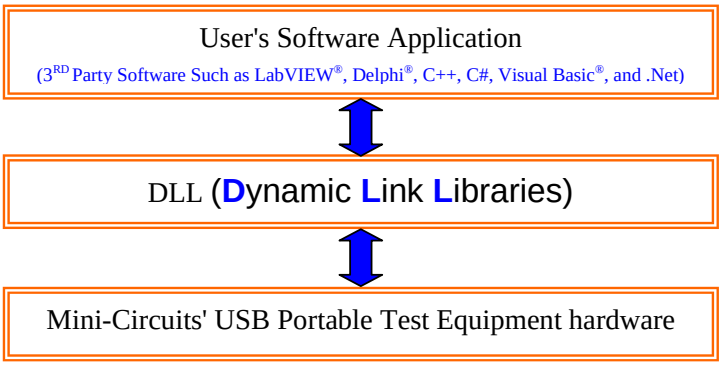

Figure 2 – DLL Interface

Mini-Circuits' provides two DLLs files:

1. ActiveX® com object - *MCL_RS232_USB_To_SPI.dll*→ Click to download
   http://www.minicircuits.com/support/software_download.html
   ActiveX® com object can be used in any programming environment that supports ActiveX® objects - third party COM (Component Object Model) compliant application. The ActiveX® DLL should be registered using RegSvr32 (see pages 5 and 6 - Register an ActiveX® DLL).

2. .NET Class Library - *MCL_RS232_USB_To_SPI64.dll*→ Click to download
   http://www.minicircuits.com/support/software_download.html
   .NET object – a logical unit of functionality that runs under the control of the .NET

2.1 - Software supported by ActiveX® and .NET Class Library

MCL_RS232_USB_To_SPI.dll - ActiveX® com object	MCL_RS232_USB_To_SPI64.dll - .NET Class Library
Instructions ▪ For **32bit** Windows OS, copy MCL_RS232_USB_To_SPI.dll to windows\system32 folder ▪ For **64bit** Windows OS, copy MCL_RS232_USB_To_SPI.dll to windows\SysWOW64 folder ▪ Register the DLL, see instructions below	**Instructions** ▪ For **32bit** Windows OS copy MCL_RS232_USB_To_SPI64.dll to windows\system32 folder ▪ For **64bit** Windows OS copy MCL_RS232_USB_To_SPI64.dll to windows\SysWOW64 folder ▪ DLL Registry is not required
Visual Studio 6 (VC++,VB®) NI LabVIEW® 8.0 or newer MATLAB® 7 or newer Delphi® Borland C++ Agilent VEE® Python	NI CVI NET (VC++, VB.net, C# 2003,2005,2008,2010) NI LabVIEW®_2009 or newer MATLAB® 2008 or newer Delphi® Borland C++

* Additional 3RD party software are supported, contact Mini-Circuits for details.

How to register mcl_pm.dll, 32-bit DLL, on a 32-bit Windows operating system?

Open the Run Command from the Start Menu and type
regsvr32 c:\windows\system32\MCL_RS232_USB_To_SPI.dll

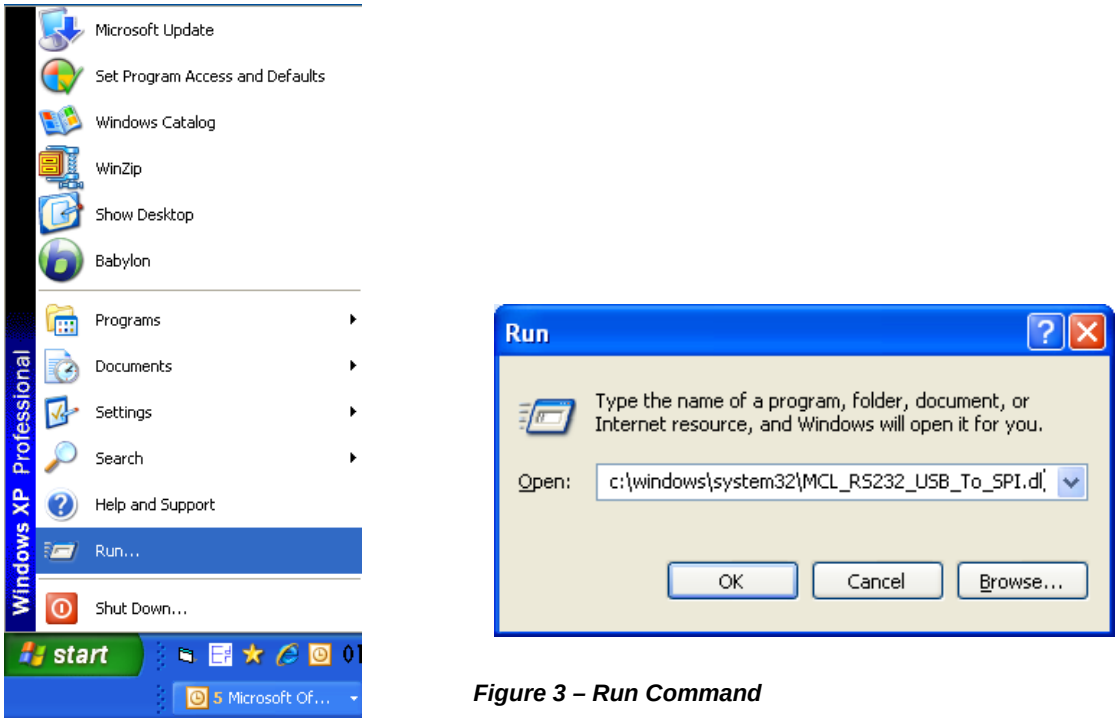

Figure 3 – Run Command

How to register MCL_RS232_USB_To_SPI.dll, 32-bit DLL on a 64-bit Windows OS?

- Run the Command Prompt as Administrator, see figure 4

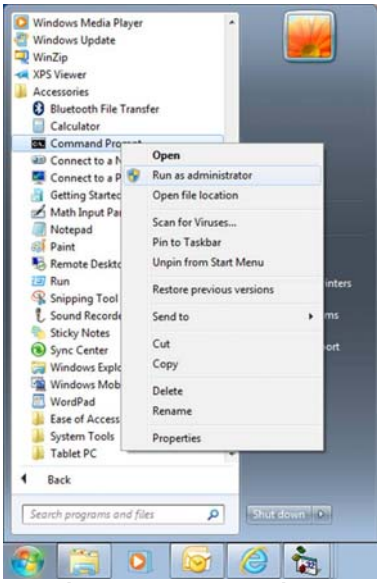

Figure 4 – Command Prompt

- Type regsvr32 c:\windows\syswow64\ MCL_RS232_USB_To_SPI.dll, see figure 5

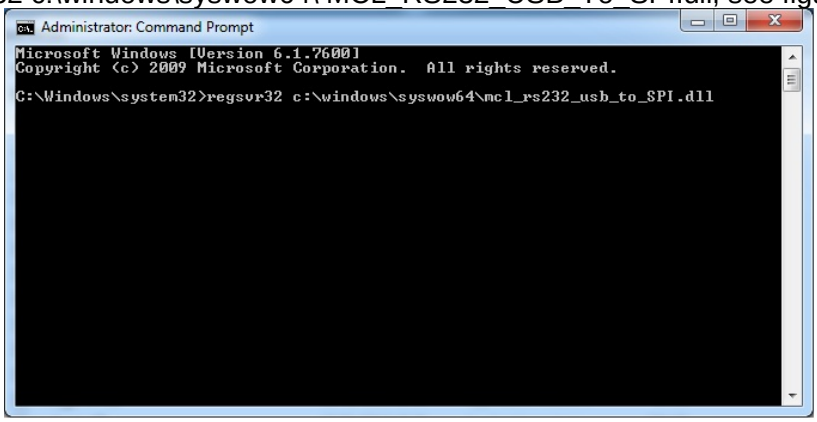

Figure 5 – Type command

- Click Enter, see figure 6.

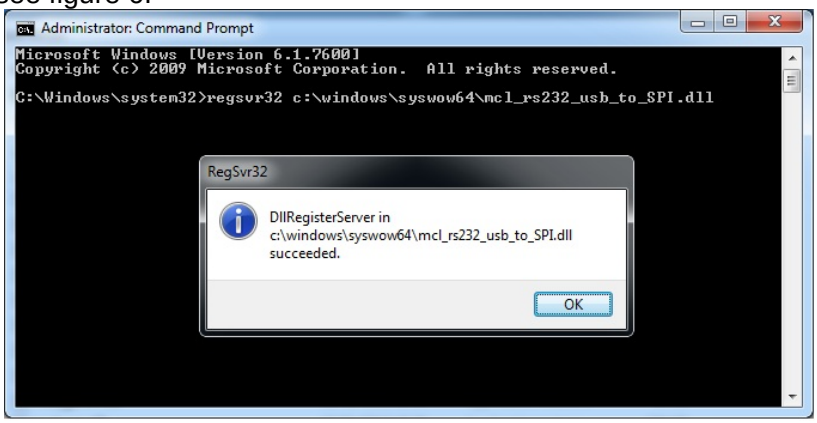

Figure 6 – Registration succeeded

2.2 - DLL Structure (both *MCL_RS232_USB_To_SPI.dll* and *MCL_RS232_USB_To_SPI64.dll*)

DLLs Functions MCL_RS232_USB_To_SPI.dll and MCL_RS232_USB_To_SPI64.dll

1. Int Send_SPI(String str_to_send)
2. Int Read_ModelName(String ModelName)
3. Int Read_SN(String SN)
4. String Read_SPI(Short NoOfBit, Short WaitForReady,String str_Ret)
5. Void Connect()
6. Void Disconnect()

Note: The DLL is useful only for converting USB to SPI Register, otherwise in case of converting RS232 to SPI see page 8.

Functions Description:

1. Int Send_SPI(String str_to_send)
   Sending SPI Data Out:
   The function returns 1 on success.

2. Int Read_ModelName(String ModelName)
   Getting the Device Model Name:
   The function returns 1 on success

3. Int Read_SN(String SN)
   Getting the Device Serial Number:
   The function returns 1 on success

4. String Read_SPI(Short NoOfBit , Short WaitForReady , String str_Ret)

   Receiving SPI Data:

   BoOfBit= The Number Of Bits to Read from SPI.
   WaitForReady if>0 then wait for "Ready Bit" to go Low.
   If = 0 no need to wait for "Ready Bit".
   Str_Ret will have the Reading Data. (Optional *string SN)

5. Void Connect()
   - Open Connection.

6. Void Disconnect()
   - Close connection. It is strongly recommended to disconnect the device before ending the program.

In case of converting RS232 to SPI, create a serial RS232 connection as follows:
Setup programming: Baud=9600, Parity=E, Data_Bits=8

Connect RS232 cable from 9 pin connector to the Computer RS232 port.
Connect to USB socket to PC or to 5 Volt adaptor.

Communication based on sending and receiving ASCII data over RS232 port.

1. Sending SPI Data OUT: Send the text "B[Binary Data]E" (MSB send first).

   The device will return "ACK".
   Example: The command "B0110111011011011E" will cause 16 Bits send to SPI Data out.
   The device will return "ACK".

2. Getting the Device Model Name: Send the text "M"

   The device will return [DeviceModelName]

3. Getting the Device Serial Number: Send the text "S"

   The device will return [DeviceSerialNumber].

4. Receiving SPI Data: Send the Text "R##E"

   ##=number of bits to get from SPI.
   The device will return "B[Binary Data]E"

   Example: The command "R16" will cause to read 16 bit from SPI DATA IN.

   The device will return "BXXXXXXXXXXXXXXXXE"
   X="0" or "1".

5. Wait for "Ready bit" to go Low then Receiving SPI Data: Send the Text "RR##E"
   ##=number of bits to get from SPI.

   The device will return "B[Binary Data]E"

   Example: The command "R16" will cause to read 16 bit from SPI DATA IN.

   The device will return "BXXXXXXXXXXXXXXXXE"
   X="0" or "1".

2.3 - Sample code

The CD package also includes a number of sample programs developed to show you how to write your own programs. The sample programs were developed in Visual C++®, Visual Basic®, C# and LabVIEW®. The sample programs provide an excellent starting point to write your own applications.

The complete project examples are available for download at:
http://www.minicircuits.com/support/software_download.html

3 - Operating in a Linux® Environment 32/64Bits OS with USB HID Support

The RS232/USB to SPI Converter is based on 2 options of controls:

A. HID USB control
B. RS232 Control.

For the first option: convert USB to SPI:

To open a connection to the power sensor, Vendor ID and Product ID are required:

- Mini-Circuits Vendor ID is: 0x20CE
- RS232/USB to SPI Converter ID is: 0x25

The communication with the sensor is done by USB Interrupt.
The transmitted and received buffer sizes are 64 Bytes.

Transmit Array should be 64 bytes [Byte 0][Byte1][Byte2]……[Byte 63]
Receive Array contains 64 bytes [Byte 0][Byte1][Byte2]……[Byte 63]

Commands List

#	Description	Command Code – Byte[0]	Additional Transmitted Bytes
1	Get device Model Name	40	--
2	Get device Serial Number	41	--
3	Send SPI Out	6	Byte[1] – Number of Data Bits Byte[2+N] – Data
4	Set pulse Width	8	Byte[1] - Pulse Width in micro seconds

* See detailed description on pages 10 - 13

1. Get the device Model Name:

To get the devise Model Name, code number 40 should be sent

Transmit Array

- Byte[0]=40
- Bytes[1] through [63] are NC - Not Care

Received Array

The Model Name will be returned in the receive array of ASCII characters. End of Model Name is signified by a 0 value.

- Byte[0]=40
- Byte[1] to the byte before the 0 value = Model Name
- All bytes after the 0 value up to byte [63] contain random values

2. Get Device Serial Number

To get the device Serial Number, code number 41 should be sent

Transmit Array

- Byte[0]=41
- Bytes[1] through [63] are NC - Not Care

Received Array

The Serial Number will be returned in the receive array of ASCII characters. End of S/N is signified by a 0 value.

- Byte[0]=41
- Byte[1] to the byte before the 0 value = Serial Number
- All bytes after the 0 value up to byte [63] contain random values

3. Send SPI OUT

Transmit Array

- Byte[0]=6
- Byte[1]=N - The number of data bits to send
- Byte[2] Byte[N+2]= the data value to send = 1 or 0
- Bytes[3] through [63] are NC - Not Care

Byte [0]	Byte [1]	Byte [2]	Byte [3]	Byte [4]	Byte [5]	Byte [6]
6	4	1	0	0	1	1

Byte [N]		Byte [63]
0		

Received Array

- Byte[0]=6
- Bytes[1] through [63] contain random values

4. Set the Pulse Width of the SPI, Data, Clock and LE

Transmit Array

- Byte[0]=8
- Byte[1]=Pulse Width in micro seconds
- Bytes[2] through [63] are NC - Not Care

Received Array

- Byte[0]=8
- Bytes[1] through [63] contain random values

For the second option convert RS232 to SPI:

Linux programmers need to create connection to serial RS232 port with the following:
Setup programming: Baud=9600, Parity=E, Data_Bits=8

Connect RS232 cable from 9 pin connector to the Computer RS232 port.
Connect to USB socket to PC or to 5 Volt adaptor.

Communication based on sending and receiving ASCII data over RS232 port.

1. Sending SPI Data OUT: Send the text "B[Binary Data]E" (MSB send first).

   The device will return "ACK".
   Example: The command "B0110111011011011E" will cause 16 Bits send to SPI Data out.
   The device will return "ACK".

2. Getting the Device Model Name: Send the text "M"

   The device will return [DeviceModelName]

3. Getting the Device Serial Number: Send the text "S"

   The device will return [DeviceSerialNumber].

4. Receiving SPI Data: Send the Text "R##E"

   ##=number of bits to get from SPI.
   The device will return "B[Binary Data]E"

   Example: The command "R16" will cause to read 16 bit from SPI DATA IN.

   The device will return "BXXXXXXXXXXXXXXXXE"
   X="0" or "1".

5. Wait for "Ready bit" to go Low then Receiving SPI Data: Send the Text "RR##E"
   ##=number of bits to get from SPI.

   The device will return "B[Binary Data]E"

   Example: The command "R16" will cause to read 16 bit from SPI DATA IN.

   The device will return "BXXXXXXXXXXXXXXXXE"
   X="0" or "1".

3.1 – Sample code

The Linux Folder in the CD package contains the following:

- usb2IO.c example source code using the libhid & libusb libraries to open the USB HID device.

The complete project samples are available on the CD or at:
http://www.minicircuits.com/support/software_download.html