Mini-Circuits®

Programming Manual

For the

USB RF Switch Matrices

Contents

1- Overview

This programming Manual is intended for customers wishing to create their own interface for Mini-Circuits' USB RF Switch Matrices.

Mini-Circuits offers support for USB Portable Test Equipment (PTE) in Windows® and Linux® Operating Systems, in a 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.

RF Switch Controller

Product Name	Version	Download	Description / Instructions	Models Supported
RF Switch Controller - Setup	A9	Download	RF Switch Controller GUI program for Windows 32/64 bit - Latest Version - Setup.	
RF Switch Controller - CD	A9	Download	Latest Version of the entire Switch Controller CD: GUI program, DLL COM Objects 32/64 bit, Linux Support and Documents. When extracting the files after download, keep the folder names.	
MCL_RF_Switch_Controller.dll	Dec 06, 2011	Download	Dll - ActiveX com object file. Registering to Windows is required. Recommended for 32 bit programming.	
mcl_RF_Switch_Controller64.dll	Dec 06, 2011	Download	Dll - .NET Class Library. Recommended for 64/32 bit programming.	
Programming Manual	Apr, 16, 2012	Download	PDF File: Detailed Guide for Programmers.	
Project Examples	Apr, 16, 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.	USB-1SPDT-A18 (XL) USB-2SPDT-A18 (XL) USB-3SPDT-A18 (XL) USB-4SPDT-A18 (XL)

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.

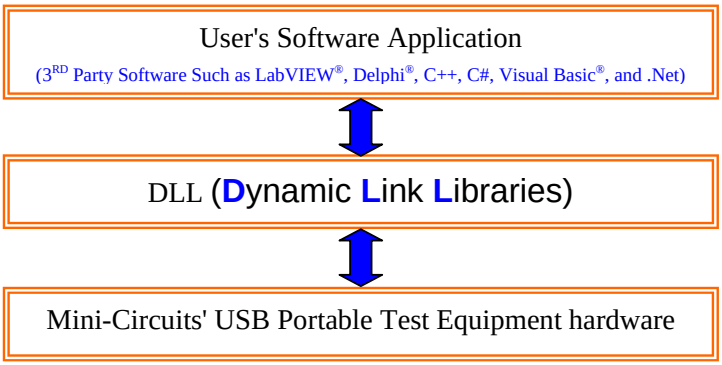


Figure 2 – DLL Interface

Mini-Circuits' provides two DLLs files:

1. ActiveX® com object - *MCL_RF_Switch_Controller.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_RF_Switch_Controller64.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_RF_Switch_Controller.dll **- ActiveX® com object**	***MCL_RF_Switch_Controller64.dll*** **- .NET Class Library**
Instructions ▪ For **32bit** Windows OS, copy MCL_RF_Switch_Controller.dll to windows\system32 folder ▪ For **64bit** Windows OS, copy MCL_RF_Switch_Controller.dll to windows\SysWOW64 folder ▪ Register the DLL, see instructions below	**Instructions** ▪ For **32bit** Windows OS copy MCL_RF_Switch_Controller64.dll to windows\system32 folder ▪ For **64bit** Windows OS copy MCL_RF_Switch_Controller64.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_RF_Switch_Controller.dll, 32-bit DLL, on a 32-bit Windows OS?

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

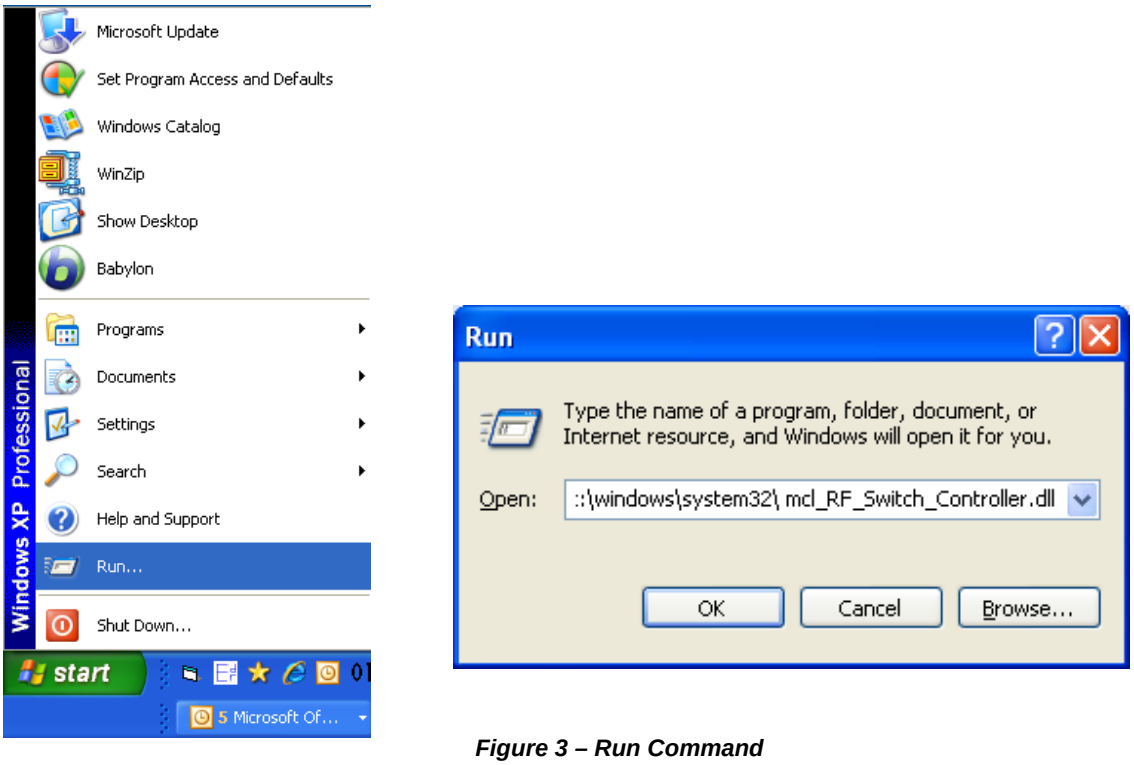


Figure 3 – Run Command

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

- Run the Command Prompt as Administrator, see figure 4

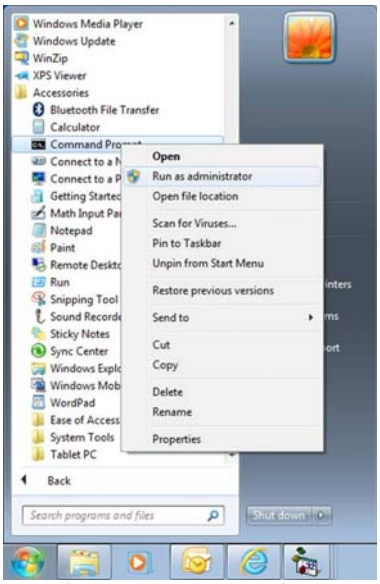


Figure 4 – Command Prompt

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

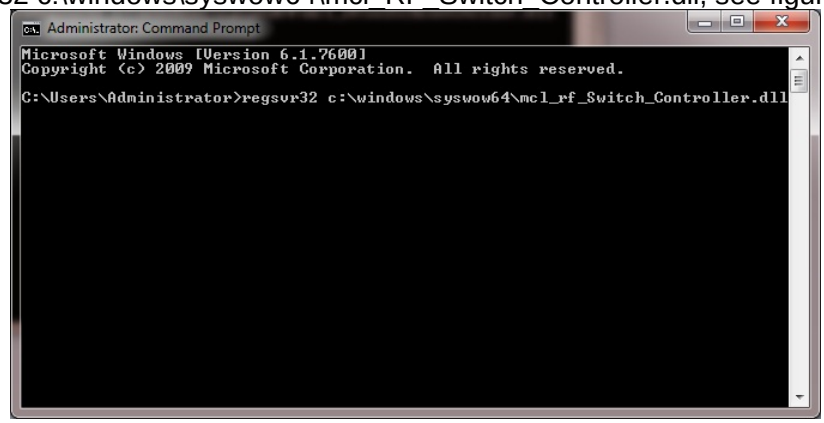


Figure 5 – Type command

- Click Enter, see figure 6.

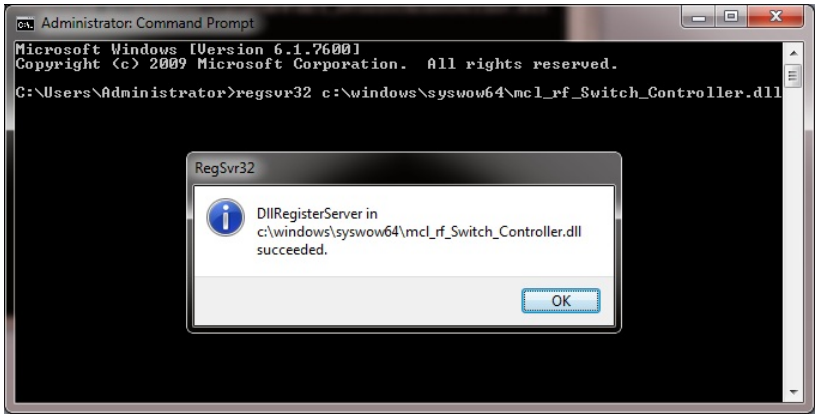


Figure 6 – Registration succeeded

2.2 - DLL Structure (both *MCL_RF_Switch_Controller.dll* and *MCL_RF_Switch_Controller64.dll*)

DLLs Functions MCL_RF_Switch_Controller.dll and MCL_RF_Switch_Controller64.dll

1. Int Connect(Optional String SN)
2. Int ConnectByAddress(Optional Short Address)
3. Void Disconnect()
4. Int GetSwitchesStatus(Int StatusRet)
5. Int Read_Model_Name(String ModelName)
6. Int Read_SN(String SN)
7. Int Set_Switch(String SwitchName, Int Val)
8. Int Set_SwitchesPort (Byte Val)
9. Int Set_Address(Int Address)
10. Int Get_Address()
11. Int Get_Avaliable_SN_List(String SN_List)
12. Int Get_Avaliable_Address_List(String Add_List)
13. Float GetDeviceTemperature(Short TSensor)
14. Int GetHeatAlarm()

Functions Description:

1. Int Connect(Optional String SN)

   SN parameter is needed when more than one Switch Box is connected to the computer.
   SN is the Serial Number of the Switch Box and can be ignored if using only one box.

2. Int ConnectByAddress(Optional Short Address)

   Address parameter represents the Address of the Switch Box.The address can be any integer number from 1 to 255 and can be changed by the SetAddress function (function 9).

   Address parameter is needed if more than one Switch Box is connected to PC.In this case the connection to the Switch Box is by Address instead of SN.This is an alternative to Function 1 (connect by SN).

3. Void Disconnect()

   Close connection to the Switch Box. It is strongly recommended to use this function to disconnect the device before ending the program.
   Shutting down the program without disconnecting the device may result in connection problem to the device. To resolve the issue shout down the program, then unplug the Switch Box from the computer and plug it back in before starting the program again.

4. Int GetSwitchesStatus(Int StatusRet)

   StatusRet - Contains the status for each switch in the box.
   The LSB bit represents switch "A" then "B" etc...
   For example: if StatusRet=12 (B00001100) - Switch C and D are ENERGIZED, all others DE-ENERGIZED.

   The function returns a non-zero value upon success.

5. Int Read_Model_Name(String ModelName)

   ModelName - Contains the Switch Box Model Name
   The function returns a non-zero value upon success.

6. Int Read_SN(String SN)

   SN- Contains the Switch Box Serial Number
   The function returns a non-zero value upon success.

7. Int Set_Switch(String SwitchName, Int Val)

   SwitchName – Contains the switch letter can be "A","B","C" or "D" (model must have the switch designated)
   Val - Contains 0 for DE-ENERGIZED any other value (1 to 255) for ENERGIZED.

   The function returns a non-zero value upon success.

8. Int Set_SwitchesPort (Byte Val)

   This function allows you to set all switches in the Matrix to the desired state with a single command.
   Each bit in the "Val" parameter controls the state of a different switch ('0'=De-Energized, '1'=Energized). LSB controls Switch A.

   For example Set_Switches(5) will set switch A and C to Energized, and all others De-Energized. (Decimal 5 is Binary 00000101: bits A and C are set to "1"=Energized).

   The function returns a non-zero value upon success.

9. Int Set_Address(Int Address)

   Set the address of the unit. The address can be any number in the 1 to 255 range.
   The function returns a non-zero value upon success.

10. Int Get_Address()

    The function returns the device address.
    If fails - Returns 0.

11. Int Get_Avaliable_SN_List(String SN_List)

    SN_List contains a list of all available Serial Numbers (separated by a single space character): [SN1] [SN2] [SN3] [SN4]....

    The function returns a non-zero value upon success.

12. Int Get_Avaliable_Address_List(String Add_List)

    Add_List contains a list of all available addresses (separated by a single space character).

    The function returns a non-zero value upon success.

13. Float GetDeviceTemperature(Short TSensor)

    Get the device temperature.
    The device may have 2 internal temperature sensors.
    Tsensor =1 for sensor number 1 or 2 for sensor no 2.
    The function returns the temperature value in Celsius.

14. Int GetHeatAlarm()

    The function returns 1 if the temperature of either of the temperature sensors exceeds 50° Celsius, (usually the temperature should not exceeded this value), else return 0.

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

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

To open a connection to the RF Switch Box, Vendor ID and Product ID are required:

- Mini-Circuits Vendor ID is: 0x20CE
- USB RF Switch Box Product ID is: 0x22

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	Set Switch A (ENERGIZED or DE-ENERGIZED	1	Byte[1] – 0 DE-ENERGIZED or 1 ENERGIZED
4	Set Switch B (ENERGIZED or DE-ENERGIZED	2	Byte[1] – 0 DE-ENERGIZED or 1 ENERGIZED
5	Set Switch C (ENERGIZED or DE-ENERGIZED	3	Byte[1] – 0 DE-ENERGIZED or 1 ENERGIZED
6	Set Switch D (ENERGIZED or DE-ENERGIZED	4	Byte[1] – 0 DE-ENERGIZED or 1 ENERGIZED
7	Set all switches	9	Byte[1] – see description
8	Get Switches Status	15	

* See detailed description on pages 11 - 13

1. Get Device Model Name

Transmit Array

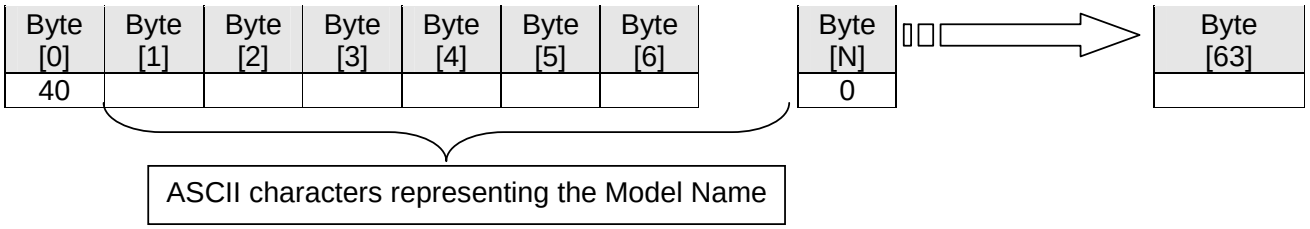
To get the Device Model Name, code number 40 should be sent.

- 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:

Transmit Array

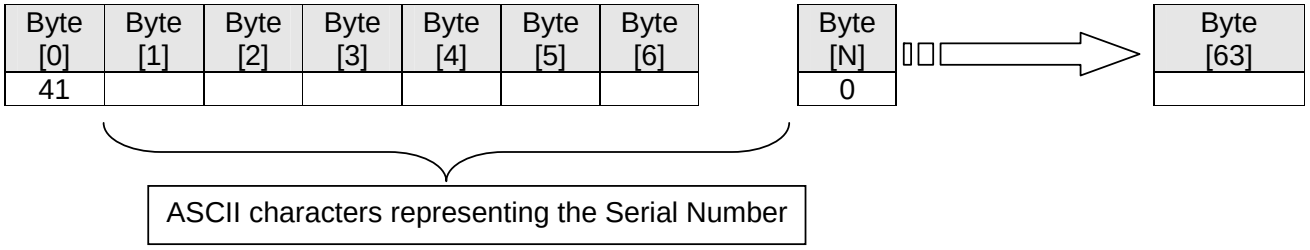
To get the Device Serial Number, code number 41 should be sent

- 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 Serial Number 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. Set Switch A: Available for all models

Transmit Array

To set Switch A, code number 1 should be sent in Byte[0], and Byte[1] should be 0 for DE-ENERGIZED or 1 for ENERGIZED

- Byte[0]=1
- Byte[1] = 0 for DE-ENERGIZED or 1 for ENERGIZED
- Bytes[2] through [63] are NC - Not Care

Received Array

The returned array of 64 bytes will be as follows:

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

4. Set Switch B: NOT Available for the USB-1SPDT-A18

Transmit Array

To set Switch B, code number 2 should be sent in Byte[0], and Byte[1] should be 0 for DE-ENERGIZED or 1 for ENERGIZED

- Byte[0]=2
- Byte[1] = 0 for DE-ENERGIZED or 1 for ENERGIZED
- Bytes[2] through [63] are NC - Not Care

Received Array

The returned array of 64 bytes will be as follows:

- Byte[0]=2
- Byte[1] through [63] contain random values

5. Set Switch C: NOT Available for the USB-1SPDT-A18 or USB-2SPDT-A8

Transmit Array

To set Switch C, code number 3 should be sent in Byte[0], and Byte[1] should be 0 for DE-ENERGIZED or 1 for ENERGIZED

- Byte[0]=3
- Byte[1] = 0 for DE-ENERGIZED or 1 for ENERGIZED
- Bytes[2] through [63] are NC - Not Care

Received Array

The returned array of 64 bytes will be as follows:

- Byte[0]=3
- Byte[1] through [63] contain random values

6. Set Switch D: Available for the USB-4SPDT-A18 ONLY

Transmit Array

To set Switch D, code number 4 should be sent in Byte[0], and Byte[1] should be 0 for DE-ENERGIZED or 1 for ENERGIZED

- Byte[0]=4
- Byte[1] = 0 for DE-ENERGIZED or 1 for ENERGIZED
- Bytes[2] through [63] are NC - Not Care

Received Array

The returned array of 64 bytes will be as follows:

- Byte[0]=4
- Byte[1] through [63] contain random values

7. Set all switches:

Handles all switches in one command

Transmit Array

- Byte[0]=9
- Byte[1]= Set Switches state(Switch A- is LSB)
- Bytes[2] through [63] are NC - Not Care

Bits used in Byte[1]:
The USB-1SPDT-A18 contains one Switch – bit LSB (0)
The USB-2SPDT-A18 contains two Switches – Bits: LSB (0) =A, 1=B
The USB-3SPDT-A18 contains three Switches – Bits: LSB (0) =A, 1=B, 2=C
The USB-4SPDT-A18 contains four Switches – Bits: LSB (0) =A, 1=B, 2=C, 3=D

For example, to set Switches A, B and C to Energized (1) and Switch D to DE-Energized (0) the following code should be sent:

- Byte[0]=9
- Byte[1]=7

Byte[1] → 00000111=7

Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3 **Switch D**	Bit 2 **Switch C**	Bit 1 **Switch B**	Bit 0 **Switch A** (LSB)
0	0	0	0	0	1	1	1

Received Array

- Byte[0]=9
- Bytes[1] through [63] contain random values

8. Get Switches Status

Transmit Array

- Byte[0]=15
- Bytes[1] through [63] are NC - Not Care

Received Array

- Byte[0]=15
- Bytes[1]= Switches Status (LSB is for Switch A)
- Bytes[2] through [63] contain random values

3.1 – Sample code

The Linux Folder in the CD package contains the following:

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

Linux Project Examples are also available on the CD package or can be downloaded at:
http://www.minicircuits.com/support/software_download.html