PRODUCT GUIDELINES

USB RF Switch Software

Table of Contents

1. General notes 3
2. Control of the relays/creating your own software 3
3. Allocation between CP2108 GPIO and actuators/Indicator contacts 3
4. Overview of the CP210x Runtime API functions 3
5. Communication with the switch via USB 4
6. The CP210xRuntime API functions 5
   - 6.1 CP210xRT_ReadLatch 5
   - 6.2 CP210xRT_WriteLatch 5
   - 6.3 CP210xRT_GetPartNumber 6
   - 6.4 CP210xRT_GetDeviceProductString 6
   - 6.5 CP210xRT_GetDeviceSerialNumber 6
7. The Radiall_USBInterface API functions 7
   - 7.1 Initialize 7
   - 7.2 GetSerialNumber 7
   - 7.3 GetSwitchID 7
   - 7.4 GetDeviceType 8
   - 7.5 GetPosition 8
   - 7.6 SetPosition 8
   - 7.7 Close 8
   - 7.8 ERROR MESSAGES 9
8. Example: Using DLLs in Labview 9

1. General notes

The control of the USB switch via the silabs CP2108 USB interface controller takes place using CP210xRuntime DLL and Radiall_USBInterface DLL.
The control of the USB relay is done via a PC only.

2. Control of the relays/creating your own software

The switch actuators are controlled via the USB interface using CP210xRuntimeDLL (Dynamic Link Library). As support for the development of your own control software, you can fin information and examples on the item page at www.silabs.com.
In the process, the actuators are operated via GPIO connections. A Low-level (logic"0") on the CP2108 will switch the relay on; a high level will switch it off.

3. Allocation between CP2108 GPIO and actuators/Indicator contacts

CP210x_GPIO_0↔ Position 1	CP210x_GPIO_8↔ Indicator contact 1
CP210x_GPIO_1↔ Position 2	CP210x_GPIO_9↔ Indicator contact 2
CP210x_GPIO_2↔ Position 3	CP210x_GPIO_10↔ Indicator contact 3
CP210x_GPIO_3↔ Position 4	CP210x_GPIO_11↔ Indicator contact 4
CP210x_GPIO_4↔ Position 5	CP210x_GPIO_12↔ Indicator contact 5
CP210x_GPIO_5↔ Position 6	CP210x_GPIO_13↔ Indicator contact 6
CP210x_GPIO_6↔ Position 7	CP210x_GPIO_14↔ Indicator contact 7
CP210x_GPIO_7↔ Position 8	CP210x_GPIO_15↔ Indicator contact 8

4. Overview of the CP210x Runtime API functions

The CP210x Runtime API provides access to the GPIO port latch, and is meant for distribution with the product containing a CP210x device.
**API: Application programming Interface.*
CP210xRT_ReadLatch () Returns the GPIO port latch of a CP210x device.
CP210xRT_WriteLatch () Sets the GPIO port latch of a CP210x device.
CP210xRT_GetPartNumber () Returns the 1-byte Part Number of a CP210x device. A return value of 8 corresponds to the CP2108 used at the relay board.
CP210xRT_GetProductString () Returns the product string programmed to the device.
CP210xRT_GetDeviceSerialNumber () Returns the serial number programmed to the device.
CP210xRT_GetDeviceInterfaceString () Returns the interface string programmed to the device.

Typically, the user initiates communication with the target CP210x device by opening a handle to a COM port using CreateFile () (See AN197: Serial Communication Guide for CP210x). The handle returned allows the user to call the API functions listed above. Each of these functions is described in the following sections.

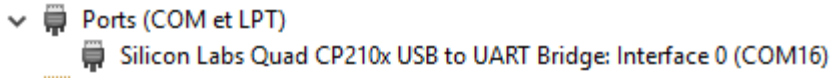
Radiall has developed another DLL (*Radiall_USBInterface.dll)* based on the provided silabs's dll which contains customized functions used to control our USB Switch.
The CP210xRuntime.DLL and *Radiall_USBInterface.dll* must always be in the same directory as the application.

5. Communication with the switch via USB

As a prerequisite for the communication with the CP2108, you have to create a "handle" to a COM port using createfile(). For more detailed information please refer to document AN197.pdf on silicon labs website.

The COM port (in the example:"COM16) to be used corresponds to that of "Silicon-Labs CP210x USB to UART Bridge (See device manager)

You can have 4 interfaces (4 COMPorts) any of them can be used in our application or if this is disturbing you can uninstall 3 by right clicking on it > uninstall



Example:
Creation of a "Handle" under vb.Net):

```
'Open a file for usb communication'
    USBDevice = CreateFile("\\.\" & COMPort, FileAccess.GENERIC_READ Or
    FileAccess.GENERIC_WRITE, _0, 0, OPEN_EXISTING,
    _FileAttributes.FILE_FLAG_OVERLAPPED Or FileAttributes.FILE_ATTRIBUTE_NORMAL, 0)
```

This "Handle" (in the example "USBDevice") can be used to call the CP210xRuntime API functions. After completion of the communication, this "handle" must be disabled again:

```
    CloseHandle(USBDevice)
    USBDevice = INVALID_HANDLE_VALUE
```

6. The CP210xRuntime API functions

6.1 CP210xRT_ReadLatch

Description: Reading the GPIO switching status (and therefore, the actuator switching status)

Prototype: CP210x_STATUSCP210xRT_ReadLatch (HANDLE handle, LPBYTE Latch)

Parameter:

1. Handle: Handle for COM port, created by CreateFile ().
2. Latch: Pointer to 1 Byte as return of the GPIO switching status (logical high=1, low=0)

Return value for the function: CP210x_STATUS=
CP210x_SUCCESS (&H0),
CP210x_INVALID_HANDLE (&H1),
CP210x_DEVICE_IO_FAILED (&H3),
CP210x_FUNCTION_NOT_SUPPORTED (&H4)

6.2 CP210xRT_WriteLatch

Description: Setting the GPIO switching status (and therefore, the relay switching status)

Prototype: CP210x_STATUSCP210xRT_WriteLatch (HANDLE handle, BYTE Mask, BYTE Latch)

Parameter:

1. Handle: Handle for COM port, created by CreateFile().
2. Mask : Definition of the GPIOs to be changed(change = 1, Keep=0)
3. Latch: 1-byte value to change the GPIOs(Logical high =1, LOW=0)

Return value for the function: CP210x_STATUS=
CP210x_SUCCESS (&H0),
CP210x_INVALID_HANDLE (&H1),
CP210x_DEVICE_IO_FAILED (&H3),
CP210x_FUNCTION_NOT_SUPPORTED (&H4)

Example: Switching on the position 1, Mask=15, Latch=1

6.3 CP210xRT_GetPartNumber

Description: Reading out the component name of the CP210x interface component.
Prototype: CP210x_STATUS CP210xRT_GetPartNumber (HANDLE handle, LPBYTE PartNum)
Parameter:

1. Handle: Handle for COM port, created by CreateFile().
2. Latch: Pointer to a byte which encodes the component number of the CP210x
   A return value of 8 corresponds to the CP2108 used here.

Return value for the function: CP210x_STATUS=
CP210x_SUCCESS (&H0),
CP210x_INVALID_HANDLE (&H1),
CP210x_DEVICE_IO_FAILED (&H3),

6.4 CP210xRT_GetDeviceProductString

Description: Reading out the product string of the cp210x interface component
Prototype: CP210xRT_GetDeviceProductString (HANDLE cyHandle, LPVOID lpProduct,LPBYTE lpbLength, BOOL bConvertToASCII=TRUE)
Parameter:

1. Handle: Handle for COM port, created by CreateFile().
2. Product: Pointer to the product string (“CP2108 USB to UART Bridge Controller”)
3. ConvertToASCII: Conversion to ASCII true/false (select true!).

6.5 CP210xRT_GetDeviceSerialNumber

Description: Reading out the serial number of the cp210x interface component
Prototype: CP210xRT_GetDeviceSerialNumber (HANDLE cyHandle, LPVOID lpSerialNumber, LPBYTE lpbLength, BOOL bConvertToASCII=TRUE)
Parameter:

1. Handle: Handle for COM port, created by CreateFile().
2. Product: Pointer to the product string (“CP2108 USB to UART Bridge Controller”)
3. ConvertToASCII: Conversion to ASCII true/false (select true!).

7. The Radiall_USBInterface API functions

7.1 Initialize

Description: Method used to initialize the communication: open the communication handle
Prototype: Initialize (ByVal COMPort As String) As Boolean
Parameters: Handle—Handle to the Com port returned by CreateFile ().
If *SUCCESS* Return Value: Boolean state of the initialization: true
Else

- DEVICE_NOT_FOUND
- INVALID_PARAMETER

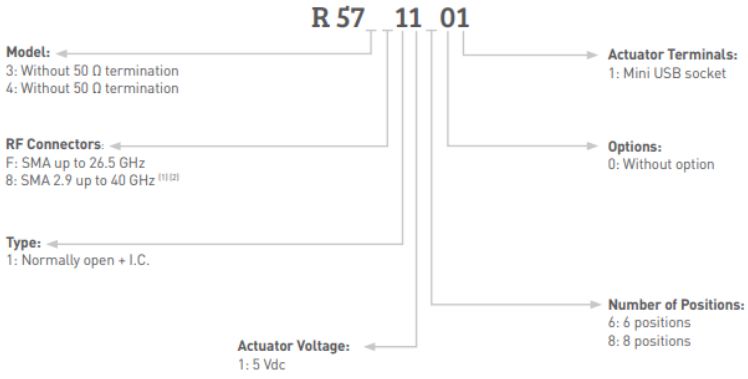
7.2 GetSerialNumber

Description: Method used to get the serial number of the switch
Prototype: GetSerialNumber () As String
Uses "CP210xRT_GetDeviceSerialNumber » of *CP210xRuntime.DLL*
If SUCCESS Returns: Serial number as a string
Else

- INVALID_HANDLE

7.3 GetSwitchID

Description: Method used to get the Switch ID
Prototype: GetSwitchID () As String
Uses the "CP210xRTGetDeviceProductString" of *CP210xRuntime.DLL*
If SUCCESS Returns: SwitchID as a string
Depending on the switch, the return value will be as below:



Else
• INVALID_HANDLE

7.4 GetDeviceType

Description: Method used to get the Part Number
Prototype: GetDeviceType () As String
By calling the function ReadDeviceType which reads the 8th digit of the SwitchID from GetSwitchID, if Digit = 6 then Device Type = SP6T, if Digit=8 then Device Type = SP8T
If SUCCESS Returns: Device Type as a string (SP6T or SP8T)
Else

- GLOBAL_DATA_ERROR

N.B: This method is a private function used in SetPosition or GetPosition. It is not accessible directly by the user. But by reading the deviceType returned, the SetPosition and GetPosition will use this function.

7.5 GetPosition

Description: Method used to get the actual position of the switch
Prototype: GetPosition()As Integer
Uses the "CP210xRT_ReadLatch" of *CP210xRuntime.DLL*
If SUCCESS Returns: Position as an integer
Else

- GLOBAL_DATA_ERROR
- INVALID_HANDLE

7.6 SetPosition

Description: Method used to set the position of the switch
Prototype: SetPosition () As Boolean
Parameter: Target position of the switch (must be between 0 and 6 or 8)
Uses the "CP210xRT_WriteLatch" of *CP210xRuntime.DLL*
If SUCCESS Returns: Boolean state if the position is set.
Else

- INVALID_PARAMETER
- INVALID_HANDLE

7.7 Close

Description: Method used to close the communication
Prototype: Close () As Boolean
If SUCCESS Returns: when USBDevice = INVALID_HANDLE_VALUE then close Handle
Else
• DO NOTHING

7.8 ERROR MESSAGES

Description: Function used to convert a return Code into String
Supported Devices: CP2108
Prototype: GetErrorMessage (ByVal ErrorCode As ReturnCodes) As String
If SUCCESS Returns: ErrorCode as a string:
"Command Failed","Device IO Failed","Device Not Found","File Error","Function Not Supported","Global Data Error","Invalid Access Type","Invalid Handle","Invalid Parameter","Success"
Else

- Unknown Error

8. Example: Using DLLs in Labview

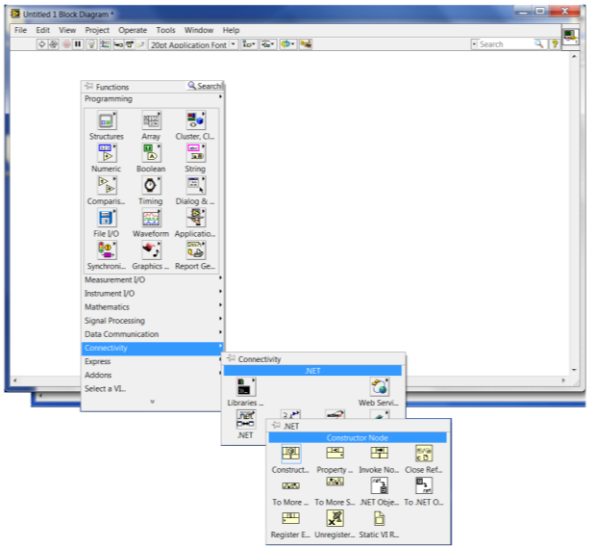


Figure 1 : Calling API from Connectivity

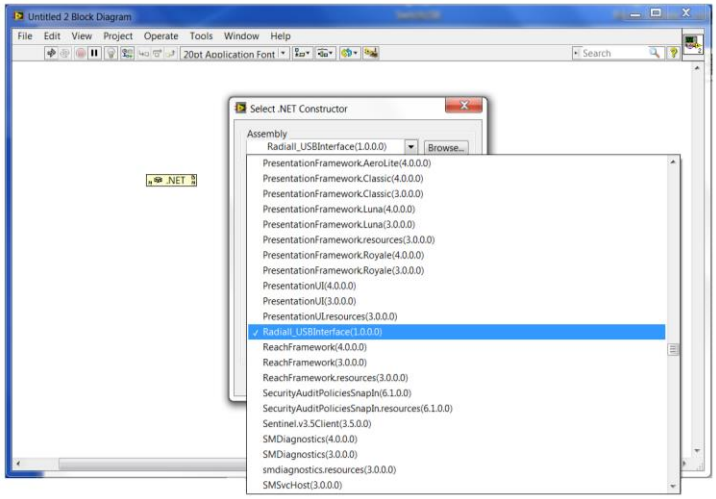


Figure 2 : "Choosing Radiall_USBInterface"

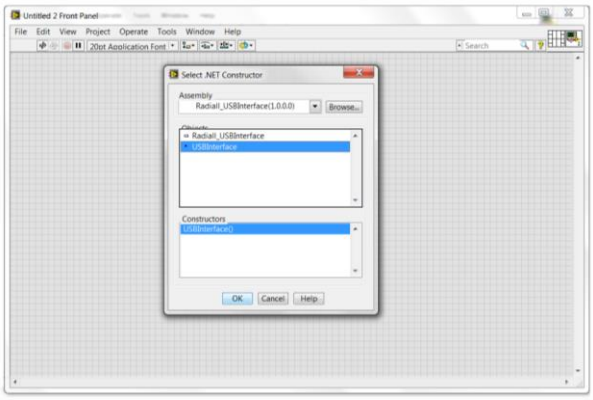


Figure 3 : USB Interface

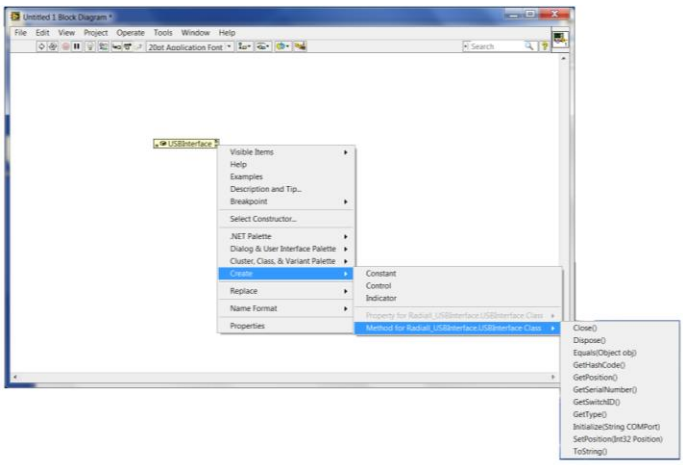


Figure 4: all functions in Radiall's API are available

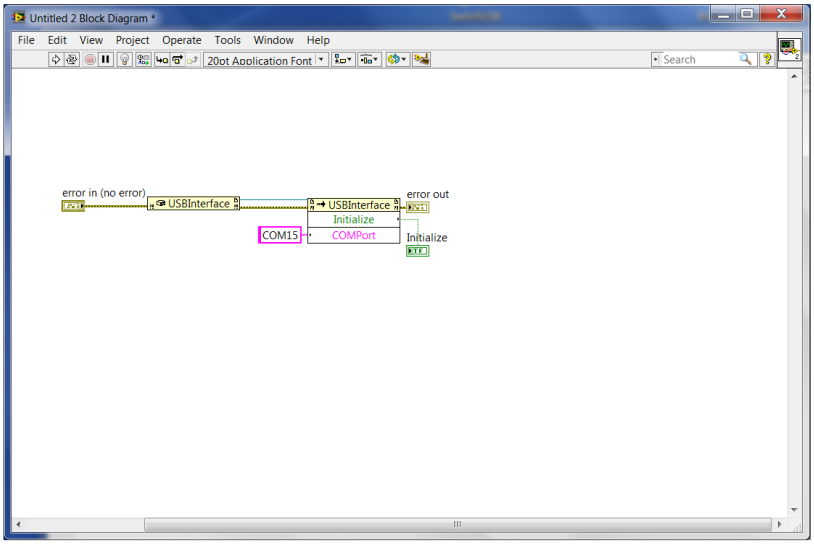


Figure 5 : example to initialize the PORT Communication