RUNTIME GPIO CONTROL FOR CP210X

1. Introduction

Some CP210x devices include GPIO pins that can be controlled by the PC using a Dynamic Link Library (DLL). The default configuration of these pins is controlled by the software included in AN721, "CP210x/CP211x Device Customization Guide," available on the interface application note web page (http://www.silabs.com/products/Interface/Pages/interface-application-notes.aspx). This document discusses reading and writing the state of these pins with software.

The CP210x Port Read/Write Example illustrates how the GPIO latch can be read from and written to using the *CP210xRuntime.DLL*. The functions in this API (*CP210xRT_ReadLatch()* and *CP210xRT_WriteLatch()*) give host-based software access to the CP210x device's GPIO latch using the USB connection as shown in Figure 1.

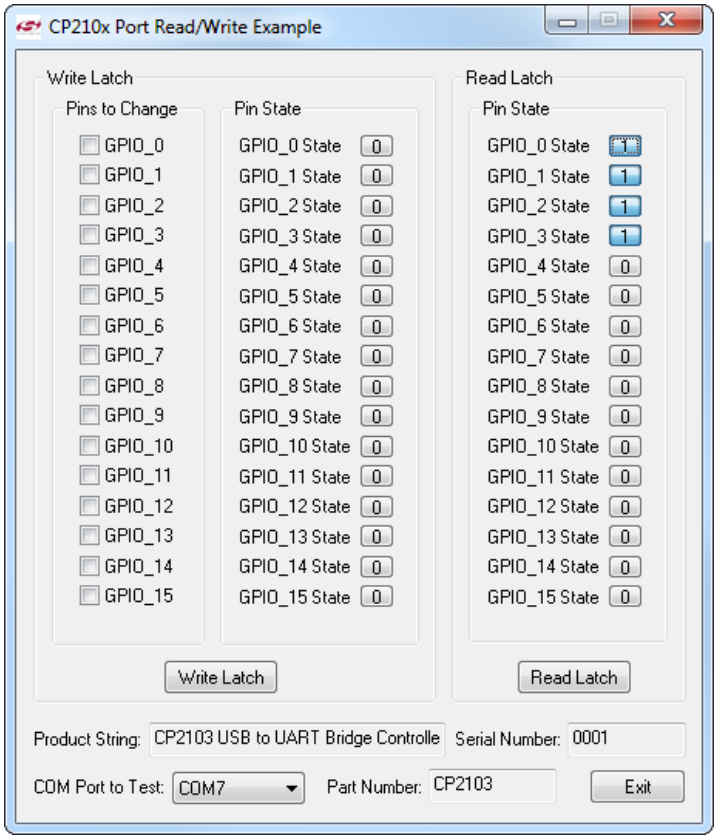


Figure 1. Main Window

The main window of the CP210x Port Read/Write Example contains one section to write the GPIO latch and another to read the GPIO latch. Below that is a list to select a COM port and display boxes for the device part number, product string, and serial number.

To write new values to the latch, select which GPIO pins to update, and set the pin state for each. Any GPIO pins not selected to change will remain static when the "Write Latch" button is pressed. The "Write Latch" button calls the *CP210xRT_WriteLatch()* function followed by the *CP210xRT_ReadLatch()* function, which updates the values displayed in the Read Latch portion of the dialog. At any time, the "Read Latch" button can be pressed to read in the current GPIO pin state of the device.

2. Creating Custom Applications using *CP210xRuntime.DLL*

Custom applications can use the CP210x Runtime API implemented in *CP210xRuntime.DLL*. To use functions implemented in *CP210xRuntime.DLL*, link *CP210xRuntime.LIB* with your Visual C++ 6.0 application. Include *CP210xRuntimeDLL.h* in any file that calls functions implemented in *CP210xRuntime.DLL*.

3. 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.

- *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.
- *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. Type definitions and constants are defined in the file CP210xRuntimeDLL.h.

Note: Functions calls into this API are blocked until completed. This can take several milliseconds depending on USB traffic.

3.1. CP210xRT_ReadLatch

Description: Gets the current port latch value from the device.

Supported Devices: CP2103, CP2104, CP2105, CP2108

Location: `CP210x Runtime DLL`

Prototype: `CP210x_STATUS  CP210xRT_ReadLatch(HANDLE Handle, LPWORD Latch)`

Parameters:
1. Handle—Handle to the Com port returned by *CreateFile()*.
2. Latch—Pointer for 4-byte return GPIO latch value [Logic High = 1, Logic Low = 0].

Return Value: CP210x_STATUS = CP210x_SUCCESS,
CP210x_INVALID_HANDLE,
CP210x_DEVICE_IO_FAILED
CP210x_FUNCTION_NOT_SUPPORTED

3.2. CP210xRT_WriteLatch

Description: Sets the current port latch value for the device.

Supported Devices: CP2103, CP2104, CP2105, CP2108

Location: `CP210x Runtime DLL`

Prototype: `CP210x_STATUS CP210xRT_WriteLatch(HANDLE Handle, WORD Mask, WORD Latch)`

Parameters:
1. Handle—Handle to the Com port returned by *CreateFile()*.
2. Mask—Determines which pins to change [Change = 1, Leave = 0].
3. Latch—4-byte value to write to GPIO latch [Logic High = 1, Logic Low = 0]

Return Value: CP210x_STATUS = CP210x_SUCCESS,
CP210x_INVALID_HANDLE,
CP210x_DEVICE_IO_FAILED
CP210x_FUNCTION_NOT_SUPPORTED

3.3. CP210xRT_GetPartNumber

Description: Gets the part number of the current device.

Supported Devices: CP2101, CP2102, CP2103, CP2104, CP2105, CP2108

Location: `CP210x Runtime DLL`

Prototype: `CP210x_STATUS CP210xRT_GetPartNumber(HANDLE Handle, LPBYTE PartNum)`

Parameters:
1. Handle—Handle to the Com port returned by *CreateFile()*.
2. PartNum—Pointer to a byte containing the return code for the part number.

Return Value: CP210x_STATUS = CP210x_SUCCESS,
CP210x_INVALID_HANDLE,
CP210x_DEVICE_IO_FAILED

3.4. CP210xRT_GetDeviceProductString

Description: Gets the product string in the current device.

Supported Devices: CP2101, CP2102, CP2103, CP2104, CP2105, CP2108

Location: `CP210x Runtime DLL`

Prototype:
```
CP210x_STATUS CP210xRT_GetDeviceProductString(HANDLE  cyHandle,
LPVOID lpProduct, LPBYTE lpbLength, BOOL bConvertToASCII = TRUE)
```

Parameters:

1. Handle—Handle to the Com port returned by *CreateFile()*.
2. lpProduct—Variable of type CP210x_PRODUCT_STRING returning the NULL terminated product string.
3. lpbLength—Length in characters (not bytes) not including a NULL terminator.
4. ConvertToASCII—Boolean that determines whether the string should be left in Unicode, or converted to ASCII. This parameter is true by default, and will convert to ASCII.

Return Value: CP210x_STATUS = CP210x_SUCCESS,
CP210x_INVALID_HANDLE,
CP210x_DEVICE_IO_FAILED
CP210x_INVALID_PARAMETER

3.5. CP210xRT_GetDeviceSerialNumber

Description: Gets the serial number in the current device.

Supported Devices: CP2101, CP2102, CP2103, CP2104, CP2105, CP2108

Location: `CP210x Runtime DLL`

Prototype:
```
CP210x_STATUS CP210xRT_GetDeviceSerialNumber(HANDLE cyHandle,
LPVOID lpProduct, LPBYTE lpbLength, BOOL bConvertToASCII = TRUE)
```

Parameters:

1. Handle—Handle to the Com port returned by *CreateFile()*.
2. lpProduct—Variable of type CP210x_SERIAL_STRING returning the NULL terminated serial string.
3. lpbLength—Length in characters (not bytes) not including a NULL terminator.
4. ConvertToASCII—Boolean that determines whether the string should be left in Unicode, or converted to ASCII. This parameter is true by default, and will convert to ASCII.

Return Value: CP210x_STATUS = CP210x_SUCCESS,
CP210x_INVALID_HANDLE,
CP210x_DEVICE_IO_FAILED
CP210x_INVALID_PARAMETER

3.6. CP210xRT_GetDeviceInterfaceString

Description: Gets the interface string of the current device.

Supported Devices: CP2105, CP2108

Location: `CP210x Runtime DLL`

Prototype:
```
CP210x_STATUS CP210xRT_GetDeviceInterfaceString(HANDLE cyHandle,
LPVOID lpInterfaceString, LPBYTE lpbLength, BOOL bConvertToASCII = TRUE)
```

Parameters:

1. Handle—Handle to the Com port returned by *CreateFile()*.
2. lpInterfaceString—Variable of type CP210x_SERIAL_STRING returning the NULL terminated interface string.
3. lpbLength—Length in characters (not bytes) not including a NULL terminator.
4. ConvertToASCII—Boolean that determines whether the string should be left in Unicode, or converted to ASCII. This parameter is true by default, and will convert to ASCII.

Return Value: CP210x_STATUS = CP210x_SUCCESS,
CP210x_INVALID_HANDLE,
CP210x_DEVICE_IO_FAILED
CP210x_INVALID_PARAMETER

Document Change List

Revision 0.1 to Revision 0.2

- Reworded Open Drain Output description for clarity

Revision 0.2 to Revision 0.3

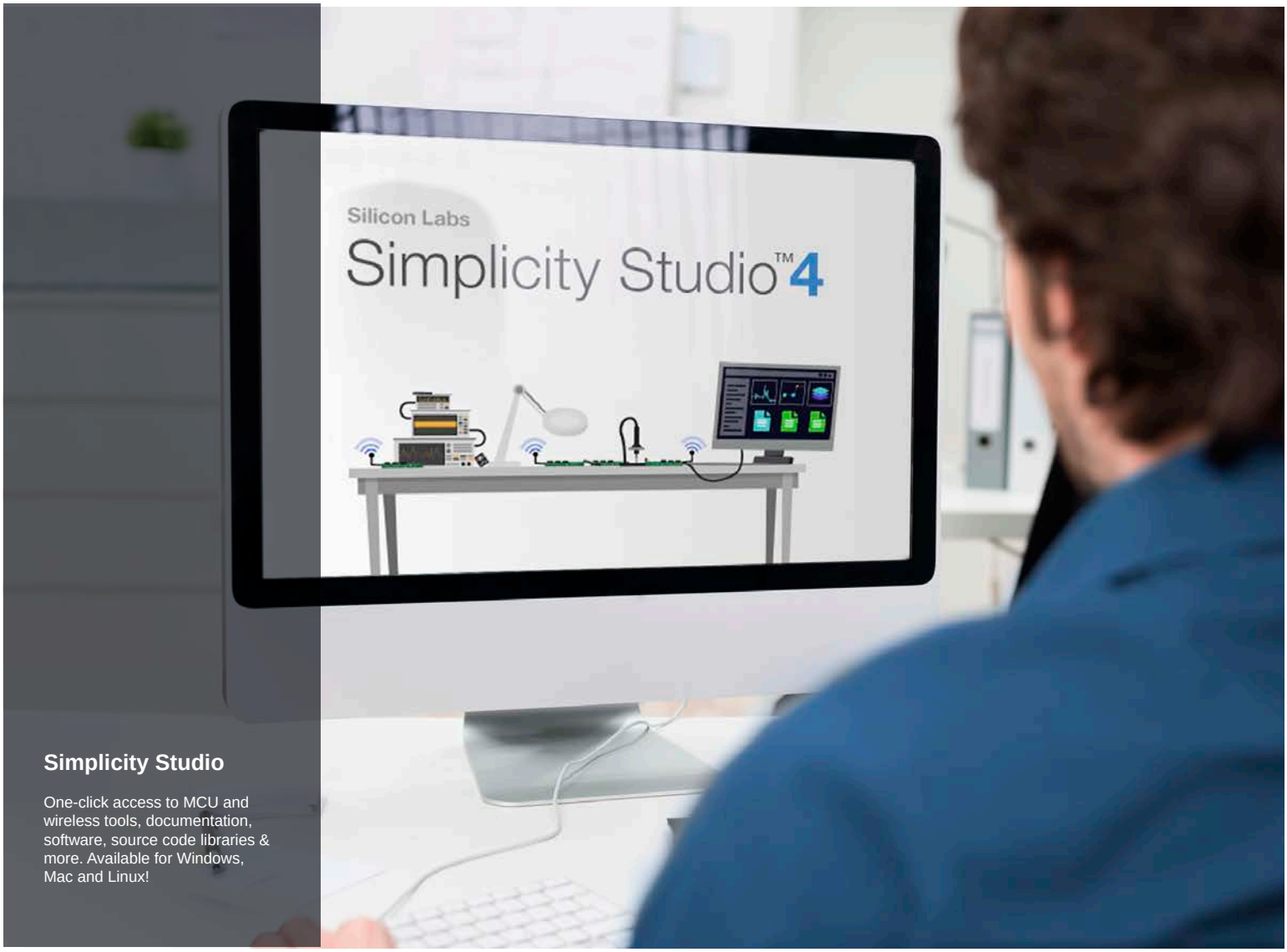
- Added CP210xRT_GetProductString
- Added CP210xRT_GetDeviceSerialNumber
- Added CP210xRT_GetDeviceProductString

Revision 0.3 to Revision 0.4

- Added support for CP2104 and CP2105
- Added CP210xRT_GetDeviceInterfaceString
- Added Table 3 and Table 4
- Added Section 5.6. CP2105 GPIO Mode and Modem Mode
- Removed the Appendix

Revision 0.4 to Revision 0.5

- Removed sections regarding port configuration.
- Updated interfaces to ReadLatch and WriteLatch from BYTE/LPBYTE to WORD/LPWORD.
- Added CP2108 as a supported device to all functions.



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com