USB 4fach-Relaiskarte

Version 07/13

Best.-Nr. 39 39 05

Anleitung zum Erstellen eigener Applikationen zur Relaiskarte

Allgemeine Hinweise

Die Ansteuerung der Relais über den Silabs CP2104 USB-Interface Controller erfolgt unter Nutzung der CP210xRuntimeDLL. Die mitgelieferten Source-Code Beispiele (Visual Basic .Net, VC++) können entsprechend den Anforderungen anpasst und erweitert werden.

Die Ansteuerung der Relais erfolgt ausschließlich vom PC aus. Ein Betrieb ohne PC mit Windows-Betriebssystem wird nicht unterstützt.

Die Beispiele zeigen nicht alle Möglichkeiten und sollen nur das Funktionsprinzip der Steuerungsbefehle verdeutlichen.

Installation der Software / Beispiele

Um sicherzustellen dass stets die aktuelle Version verwendet wird, finden Sie die zugehörige Software im Downloadbereich der Artikelseite unter www.conrad.com. Der Start der Installation erfolgt durch Aufruf der „Setup.exe“. Nach dem Kopieren der Programmdateien erfolgt automatisch der Aufruf der Treiberinstallation. Folgen Sie dabei den Anweisungen der Treiberinstallationssoftware. Aktuelle Treiber finden Sie ggf. unter www.silabs.com.

Es wird empfohlen, nach Abschluss der Installation den PC neu zu starten.

Verbinden Sie jetzt die Relaiskarte mit einem freien USB-Port Ihres Computers. Hierfür besitzt die Relaiskarte eine USB-Schnittstelle (USB-B-Buchse). Bei erstmaligem Anschluss erkennt Windows die neue Hardware und schließt die Treiberinstallation ab. Dieser Vorgang kann abhängig vom Betriebssystem mehrere Minuten in Anspruch nehmen.

Ansteuerung der Relais / Erstellen eigener Software

Die Relaiskarte wird über das USB-Interface unter Nutzung der CP210xRuntimeDLL (Dynamic Link Library) gesteuert. Als Hilfestellung zur Entwicklung eigener Steuerungssoftware finden Sie Informationen und Beispiele als Download auf der Artikelseite unter www.conrad.com.

Eine detaillierte Anleitung zu den DLL-Funktionen finden sie in der AN223.pdf, Kapitel 7 „Creating Custom Applications using CP210xRuntime.DLL“. Die Definitionen und Konstanten befinden sich in der „CP210xRuntimeDLL.h“.

Hinweise:

- Die CP210xRuntime.DLL muss sich stets im selben Verzeichnis befinden wie die Anwendung.
- Als Grundlage für die Entwicklung eigener Programme empfiehlt es sich, unter Visual C++6.0 das Projekt „CP210xPortReadWriteExample_SRC“ als Vorlage zu verwenden. Unter VB.Net 2010 kann das „USB_4X_RELAIS_DEMO“ als Vorlage verwendet werden.
- Die Relais werden dabei über die GPIO Anschlüsse geschaltet. Ein LOW-Pegel (logisch „0“) am CP2104 führt zum Einschalten der Relais, ein High-Pegel zum Abschalten.
- Weitere Funktionsaufrufe der API sind blockiert, bis der aktuelle Funktionsaufruf abgeschlossen ist. Dies kann, abhängig vom USB-Datenverkehr, mehrere Millisekunden in Anspruch nehmen.

Zuordnung zwischen CP2104 GPIO und Relais:

CP210x_GPIO_0 ↔ Relais 1

CP210x_GPIO_1 ↔ Relais 2

CP210x_GPIO_2 ↔ Relais 3

CP210x_GPIO_3 ↔ Relais 4

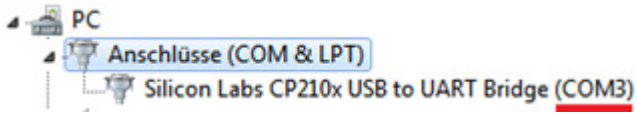
Übersicht über die CP210x Runtime API Funktionen:

- CP210xRT_ReadLatch(): Lesen der Schaltzustände der GPIOs. Die Relais werden über GPIO 0…3 angesteuert.
- CP210xRT_WriteLatch(): Ändern des GPIO Schaltzustand. Die Relais werden dabei über logisch “0” eingeschaltet. Diese können dabei einzeln geschaltet werden, ohne die anderen zu beeinflussen.
- CP210xRT_GetPartNumber(): Auslesen des Interfacebaustein-Typs. Ein Rückgabewert von 4 entspricht dem verwendeten CP2104.
- CP210xRT_GetDeviceSerialNumber(): Auslesen der Seriennummer des CP2104-Bausteins. Falls mehrere Relaiskarten verbunden sind, kann diese Nummer zur Zuordnung genutzt werden.
- CP210xRT_GetDeviceProductString: Bezeichnung des Bausteins auslesen.

Kommunikation mit der Relaiskarte über USB:

Vorraussetzung für die Kommunikation mit dem CP2104 ist, dass zuerst ein „Handle“ zu einem COM-Port durch Createfile() erstellt wird. Detaillierte Informationen hierzu enthält das Dokument „AN197.pdf“.

Der zu verwendende COM-Port (im Beispiel „COM3“) entspricht dem der „Silicon-Labs CP210x USB to UART Bridge“ (siehe Gerätemanager).



Beispiel: Erstellen eines „Handles“ unter Visual C++ 6.0 (im Beispiel: COM3):

(siehe AN197, Kap 2. “Opening a COM Port”)

HANDLE hMasterCOM = CreateFile („\\\\.\\COM3“, GENERIC_READ | GENERIC_WRITE, 0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL | FILE_FLAG_OVERLAPPED, 0);

Über dieses „Handle“ (im Beispiel: „hMasterCOM“) können dann die CP210xRuntime API-Funktionen aufgerufen werden. Nach Beenden der Kommunikation muss dieses „Handle“ wieder geschlossen werden:

CloseHandle(hMasterCOM);

hMasterCOM = INVALID_HANDLE_VALUE;

Die CP210xRuntime API Funktionen

CP210xRT_ReadLatch

Beschreibung: Den GPIO-Schaltzustand (und damit die Relaisschaltzustände) des CP2104 lesen.

Prototyp:

CP210x_STATUS CP210xRT_ReadLatch(HANDLE Handle, LPBYTE Latch)

Parameter:

1. Handle: Handle zum COM-Port, erstellt durch CreateFile().
2. Latch: Pointer auf 1-Byte als Rückgabe des GPIO Schaltzustands (Logisch High=1, Low=0).

Rückgabewert der Funktion: CP210x_STATUS =
CP210x_SUCCESS (&H0),
CP210x_INVALID_HANDLE (&H1),
CP210x_DEVICE_IO_FAILED (&H3),
CP210x_FUNCTION_NOT_SUPPORTED (&H4)

CP210xRT_WriteLatch

Beschreibung: Den GPIO-Schaltzustand (und damit die Relaisschaltzustände) des CP2104 setzen.

Prototyp:

CP210x_STATUS CP210xRT_WriteLatch(HANDLE Handle, BYTE Mask, BYTE Latch)

Parameter:

1. Handle: Handle zum COM-Port, erstellt durch CreateFile().
2. Mask: Festlegung, welche GPIOs geändert werden sollen (Ändern = 1, Beibehalten = 0).
3. Latch: 1-Byte Wert, um die GPIOs zu ändern (Logisch High=1, LOW=0).

Rückgabewert der Funktion: CP210x_STATUS =
CP210x_SUCCESS (&H0),
CP210x_INVALID_HANDLE (&H1),
CP210x_DEVICE_IO_FAILED (&H3)
CP210x_FUNCTION_NOT_SUPPORTED (&H4)

Beispiel: Alle Relais einschalten. Mask = 15 (dez), Latch = 0

CP210xRT_GetPartNumber

Beschreibung: Die Bauteilbezeichnung des CP210x Interfacebausteins auslesen.

Prototyp:

CP210x_STATUS CP210xRT_GetPartNumber(HANDLE Handle, LPBYTE PartNum)

Parameter:

1. Handle: Handle zum COM-Port, erstellt durch CreateFile().
2. PartNum: Pointer auf ein Byte, welches die Bauteilnummer des CP210x codiert.
   Ein Rückgabewert von 4 entspricht dem hier verwendeten CP2104.

Rückgabewert der Funktion: CP210x_STATUS =

CP210x_SUCCESS (&H0)

CP210x_INVALID_HANDLE (&H1)

CP210x_DEVICE_IO_FAILED (&H3)

CP210xRT_GetDeviceProductString

Beschreibung: Den Productstring des Cp210x Interfacebausteins auslesen.

Prototyp:

CP210xRT_GetDeviceProductString(HANDLE cyHandle, LPVOID lpProduct, LPBYTE lpbLength, BOOL bConvertToASCII = TRUE)

Parameter:

1. Handle: Handle zum COM-Port, erstellt durch CreateFile().
2. Product: Pointer auf den Product-String („CP2104 USB to UART Bridge Controller").
3. ConvertToASCII: Konvertierung nach ASCII true/false (**true wählen!**).

CP210xRT_GetDeviceSerialNumber

Beschreibung: Die Seriennummer des Cp210x Interfacebausteins auslesen.

Prototyp:

CP210xRT_GetDeviceSerialNumber(HANDLE cyHandle, LPVOID lpSerialNumber, LPBYTE lpbLength, BOOL bConvertToASCII = TRUE)

Parameter:

1. Handle: Handle zum COM-Port, erstellt durch CreateFile().
2. SerialNumber: Pointer auf den Product-String („CP2104 USB to UART Bridge Controller").
3. ConvertToASCII: Konvertierung nach ASCII true/false (**true wählen!**).

USB 4x relay board

Version 07/13

Item no. 39 39 05

Instructions for creating custom applications for the relay card

General notes

The control of the relays via the Silabs CP2104 USB interface controller takes place using CP210xRuntimeDLL. The source code samples (Visual Basic .Net, VC++) provided can be adapted and extended according to your requirements.

The control of the relays is done via the PC only. An operation of the product without a PC with Windows operating system is not supported.

The examples do not show all options and they are intended only to illustrate the functional principle of the control commands.

Installation of the software / examples

To ensure you always use the latest software version, you will find the associated software in the download section of the item page at

www.conrad.com. The installation is started by clicking on “Setup.exe”. After the program files are copied, the driver installation will start automatically. In this process, follow the instructions of the driver installation software. If required, you can find the latest drivers at: www.silabs.com.

After completion of the installation, we recommend rebooting the PC.

Connect the relay card with a free USB port of your computer. For this purpose, the relay card features a USB interface (USB-B socket). When the product is connected for the first time, Windows recognises the new hardware and completes the driver installation. Depending on the operating system, this

process may take several minutes.

Control of the relays / creating your own software

The relay card is controlled via the USB interface using the CP210xRuntimeDLL (Dynamic Link Library). As support for the development of your own control software, you can find information and examples available as download on the item page at www.conrad.com.

You can find detailed instructions for the DLL functions in the file AN223.pdf, section 7 “Creating Custom Applications using CP210xRuntime.DLL”. For definitions and constants please refer to “CP210xRuntimeDLL.h”.

Tips:

- The CP210xRuntime.DLL must always be in the same directory as the application.
- As basis for the development of your own programs, we recommend using the project “CP210xPortReadWriteExample_SRC” as a template under Visual C++6.0. Under VB.Net 2010, “USB_4X_RELAIS_DEMO” can be used as a template.
- In the process, the relays are operated via the GPIO connections. A LOW-level (logic “0”) on the CP2104 will switch the relay on; a high-level will switch it off.
- Any further function call of the API is blocked until the current function call is completed. Depending on the USB data traffic, this may take several milliseconds.

Allocation between CP2104 GPIO and relays:

CP210x_GPIO_0 ↔ relay 1

CP210x_GPIO_1 ↔ relay 2

CP210x_GPIO_2 ↔ relay 3

CP210x_GPIO_3 ↔ relay 4

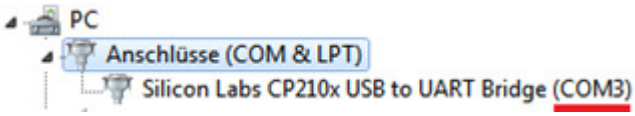
Overview of the CP210x Runtime API functions:

- CP210xRT_ReadLatch(): Reading the switching status of the GPIOs. The relays are controlled via GPIO 0…3.
- CP210xRT_WriteLatch(): Changing the GPIO switching status. The relays are switched on via the logical value “0”. Thereby, the relays can be switched individually without affecting the others.
- CP210xRT_GetPartNumber(): Reading out the interface component type. A return value of 4 corresponds to the CP2104 used at the relay board.
- CP210xRT_GetDeviceSerialNumber(): Reading out the serial number of the CP2104 component. If multiple relay cards are connected, this number can be used for allocation.
- CP210xRT_GetDeviceProductString: Reading out the name of the component.

Communication with the relay card via USB:

As a prerequisite for the communication with the CP2104, you have to create a "handle" to a COM port first using Createfile(). For more detailed information please refer to document "AN197.pdf".

The COM port (in the example: "COM3") to be used corresponds to that of "Silicon-Labs CP210x USB to UART Bridge" (see device manager).



Example: Creation of a "Handle" under Visual C++ 6.0 (in the example: COM3):

(see AN197, section 2. "Opening a COM Port")

HANDLE hMasterCOM = CreateFile („\\\\.\\COM3", GENERIC_READ | GENERIC_WRITE, 0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL | FILE_FLAG_OVERLAPPED, 0);

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

CloseHandle(hMasterCOM);

hMasterCOM = INVALID_HANDLE_VALUE;

The CP210xRuntime API functions

CP210xRT_ReadLatch

Description: Reading the GPIO switching status (and therefore, the relay switching status) of the CP2104.

Prototype:

CP210x_STATUS CP210xRT_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)

CP210xRT_WriteLatch

Description: Setting the GPIO switching status (and therefore the switching status of the relays) of the CP2104.

Prototype:

CP210x_STATUS CP210xRT_WriteLatch(HANDLE Handle, BYTE Mask, BYTE Latch)

Parameter:

1. Handle: Handle for the 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 values of the function: CP210x_STATUS =
CP210x_SUCCESS (&H0),
CP210x_INVALID_HANDLE (&H1),
CP210x_DEVICE_IO_FAILED (&H3)
CP210x_FUNCTION_NOT_SUPPORTED (&H4)

Example: Switching on all of the relays. Mask = 15 (dec), Latch = 0

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 the COM port, created by CreateFile().
2. PartNum: Pointer to a byte which encodes the component number of the CP210x.
   A return value of 4 corresponds to the CP2104 used here.

Return value of the function: CP210x_STATUS =

CP210x_SUCCESS (&H0)

CP210x_INVALID_HANDLE (&H1)

CP210x_DEVICE_IO_FAILED (&H3)

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 the COM port, created by CreateFile().
2. Product: Pointer to the product string (“CP2104 USB to UART Bridge Controller”).
3. ConvertToASCII: Conversion to ASCII true/false (select true!).

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 the COM port, created by CreateFile().
2. SerialNumber: Pointer to the product string (“CP2104 USB to UART Bridge Controller”).
3. ConvertToASCII: Conversion to ASCII true/false (select true!).