



USC-216

User Guide

Original

www.sysacom.com

Table of Contents

1.	INTRODUCTION.....	3
2.	THE CD-ROM	3
3.	HOW TO USE THE FILES	3
3.1	THE ADDICT TECHNOLOGIES USB DRIVER.....	3
3.1.1	The INF file.....	4
3.1.2	The SYS file.....	4
3.2	PROGRAMMING INSTRUCTION	5
3.2.1	Command.....	5
3.2.1.1	Set SPI mode.....	5
3.2.1.2	Get SPI mode	5
3.2.1.3	Set SPI Clock mode	6
3.2.1.4	Get SPI Clock mode.....	8
3.2.1.5	Set SPI slave select.....	8
3.2.1.6	Get SPI slave select.....	9
3.2.1.7	Test SPI port	9
3.2.1.8	Request byte to send.....	9
3.2.2	Data transmission.....	11
3.2.2.1	USB host to SPI device	11
3.2.2.2	SPI device to USB host	12
3.2.3	Interface between the API and the driver	13
3.3	SETUP PROJECT	14
3.3.1	Installation.....	14
3.3.2	Utilization.....	16
4.	SPECIFICATION	20
5.	WARRANTY.....	21

1. Introduction

Congratulations on your purchase of the USC-216 Isolated USB to SPI converter from Addict Technologies. It is the ideal module to connect any electronic design having an SPI port to a computer having a USB port.

This document is focused on helping you with your design of the software drivers and the Application-Programming Interface (API) used to communicate with your end application. For hardware details, please refer to the product datasheet.

Note: The code given as an example assumes that the PC will be a USB host. The USC-216 was not designed to be used as a USB On-the-Go.

2. The CD-ROM

The CD-ROM provided with your USC-216 contains many files. They have been carefully selected to help you in designing a functioning software in a short time and gives some hints on how the USB works.

Files on the CD-ROM are:

USC-216-00.01.pdf	The USC-216 datasheet.
USC-216-URG.01.pdf	The USC-216 user guide.
USC216.inf	A file use to inform Windows on how to load the USB driver.
USC216.sys	The Addict technologies USB driver,
USC216.exe	API to test USC-216,

3. How to use the files

This section explains how to use the driver files provided on the CD-ROM.

3.1 The Addict technologies USB driver

The Addict technologies USB driver provided can be use to perform basic USB communication. This is a bulk mode USB port using endpoint 0 for enumeration, endpoint 1 for reception (IN endpoint) and endpoint 2 for transmission (OUT endpoint)*.

* Note: If you're not familiar with USB endpoint or the USB protocol in general, good books are available to help you learning it quickly. An example is, "USB complete" from Jan Axelson. She describes the USB hardware in great details. Also, "Programming the Windows driver model" from Walter Oney explains how to write a Windows Driver.

3.1.1 The INF file

The file “USC216.inf” is used to inform Windows on how to load your driver. This is the file that the Windows plug and play engine will look for upon detection of a device on the USB port.

A first section that is very important in this file is the vendor ID and the product ID. To ensure that Windows matches the right driver with the right device, the USB enumeration process asks the target unit to return its Vendor ID and its Product ID. It then searches in all the INF files retrieved for the one that matches. In our sample file,

```
[Addict_Technologies]
%VID_0525&PID_A533.DeviceDesc% = DriverInstall,USB\VID_0525&PID_A533
```

The 0x0525 code represents the vendor ID and the 0xA533 code represents product ID. You can use any code for developing your application in laboratories as long as they match the one returned by your target device and that they are not currently in use by another USB device connected to your computer. Take note that for the deployment of your product, you must get your manufacturer code and vendor ID code directly from the USB organization. They can be contacted at www.usb.org

Another very important line is the one that tell Windows which driver to load and which service binary to start.

```
[USC216.CopyFiles]
USC216.sys
```

```
ServiceBinary = %10%\system32\drivers\USC216.sys
```

If you change the name of the USC216.sys file, you must also change the name in the following two sections. For complete information on the INF file, refer to the Walter Oney book.

3.1.2 The SYS file

The SYS file is the driver itself. It was built from the source files that can be found in the source folder. It is an adaptation of some samples available on the Windows Driver development Kit (Windows DDK). It was built using the Windows DDK to be compatible with Windows XP. If you need a driver for a different platform, you will need to recompile those files using the Windows DDK. This is a free development tool that can be downloaded directly from the Microsoft web site.

3.2 Programming instruction

This section provides information about the protocol to be used to communicate with the USC-216. In addition, this section provides a complete explanation of the interface between the Application-Programming Interface (API) and the driver to design your API effectively. So the protocol and the interface between the API, the driver and the USC-216 must be followed to operate correctly.

3.2.1 Commands

Different commands can be sent to the USC-216 to communicate, configure or test the SPI port. This next section explains how to use all those commands.

3.2.1.1 Set SPI mode

This command is used to set the mode of the SPI port and to handle the INT pin. The master mode indicates that the USC-216 has the control of the SPI clock. For the slave mode, the SPI master device controls the SPI clock, so the USC-216 has no control of it. The USC-216 can send or received data in 8 bits or 16 bits mode. The INT pin is used only in slave mode to indicate to the SPI master device that data is ready to be sent. When this command is sent to the USC-216, an Acknowledge is responded by the USC-216 to confirm that the change is really done. This Acknowledge represents the same command sent to the USC-216. It is gathered in the unit until the API has read it. If the Acknowledge received by the host is not the same that the API sent, an error has occurred.

Set SPI mode				
Byte #	Name	Value	Description	Default value
0	Instruction	0x02	Set mode of SPI port	
1	Mode	0x00	Slave mode	Master
		0x01	Master mode	
2	Bit mode	0x00	8 bits mode	8 bits
		0x01	16 bits mode	
3	INT pin	0x00	Disable INT pin	INT pin Disabled
		0x01	Enable INT pin	

Table 1: Set SPI mode

3.2.1.2 Get SPI mode

This command is used to get the SPI mode and the INT pin. The answer returned to the host is the instruction value of the get command associated with the 3 bytes containing the status of each mode and INT pin. The USC-216 makes accessible the response immediately after it's treatment. The answer is stored in the converter until the API reads it.

Get SPI mode			
Byte #	Name	Value	Description
0	Instruction	0x03	Get mode of SPI port

Table 2: Get SPI mode

Return Get SPI mode			
Byte #0	Byte #1	Byte #2	Byte #3
Instruction(0x03)	Mode	Bit mode	INT pin

Table 3: Return Get SPI mode

3.2.1.3 Set SPI Clock mode

This command is used to set the baud rate, phase and polarity of the SPI clock. The frequency of the SPI serial clock in master mode is set with two fields, the SPI Baud Rate Prescale Divisor field (SPPR) and the SPI Baud Rate Divisor field (SPR). Both fields are on three bits and the baud rate divisor equation is as follow.

$$BaudRate = \frac{24Mbps}{(Prescale\ Divisor \cdot Divisor)}$$

Equation 1: Baudrate equation

For example, having a baud rate of 500Kbps we may have a prescale divisor of 6 (0x05) and a divisor of 8 (0x02).

Field	Description
SPPR[2 :0]	SPI Baud Rate Prescale Divisor: This 3-bit field selects one of eight divisors for the SPI baud rate prescale.
SPR[2:0]	SPI Baud Rate Divisor: This 3-bit field selects one of eight divisors for the SPI baud rate.

Table 4: Baud rate field description

SPPR2: SPPR1: SPPR0	Prescale Divisor	SPR2: SPR1: SPR0	Divisor
0:0:0	1	0:0:0	2
0:0:1	2	0:0:1	4
0:1:0	3	0:1:0	8
0:1:1	4	0:1:1	16
1:0:0	5	1:0:0	32
1:0:1	6	1:0:1	64
1:1:0	7	1:1:0	128
1:1:1	8	1:1:1	256

Table 5: Value of Baud rate

The USC-216 allows the user to set the clock polarity and the clock phase. The clock polarity selects an inverter or non-inverter bit that is in series with the SPI clock. This controls the active level of the SPI clock. The clock phase chooses between two different clock phase relationships (clock and data). The next figure shows the application of these possibilities.

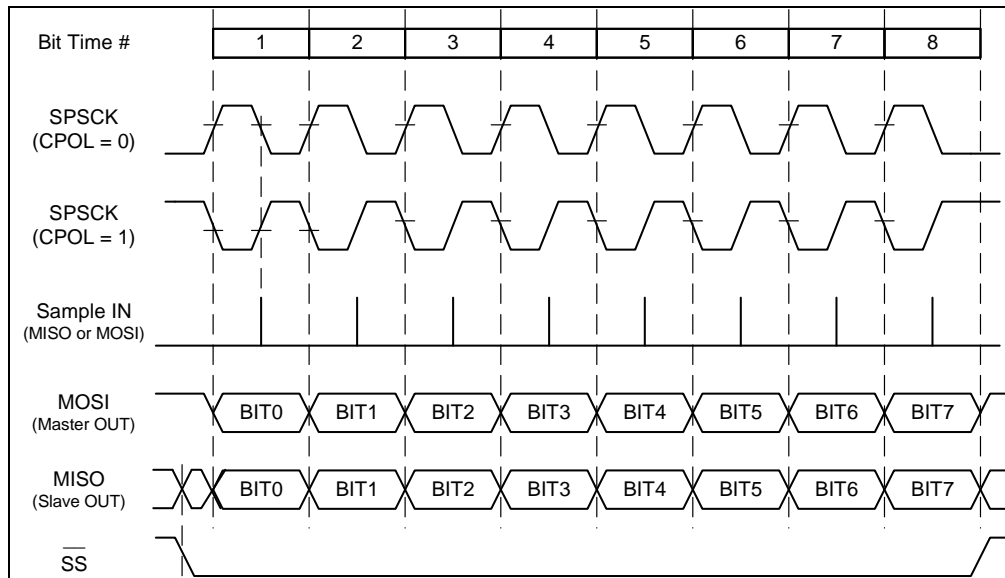


Figure 1: SPI clock format (CPHA = 1)

When CPHA = 1, the slave begins to drive when SS reaches active low. In this case the slave select (\overline{SS}) is set to automatic. On the first SPSCCK edge, the data is defined and shifts the first data onto the transmission line. The next edge causes the sampling of the bit on the transmission line. The third edge shifts the byte of one bit onto the transmission line and the fourth edge onto the sampling etc.

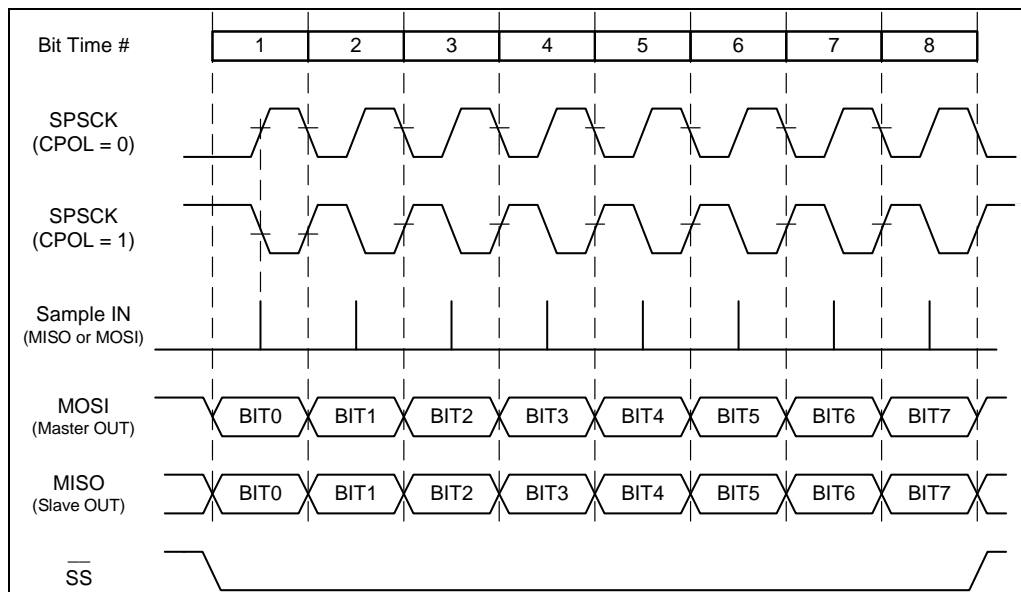


Figure 2: SPI clock format (CPHA = 0)

When CPHA = 0, the slave and the master device begin to drive when SS reaches active low. In this case, the slave select (\overline{SS}) is set to automatic. The first SPSCCK edge initiates the sampling of the bit on the transmission line. The second edge shifts the byte of one bit onto the transmission line and the third edge onto the sampling etc.

When this command is sent to the USC-216, an Acknowledge is responded by the USC-216 to confirm that the change is really done. This Acknowledge represents the same command send to the USC-216. It is gathered in the unit until the API has read it. If the Acknowledge received by the host is not the same that the API sent, an error has occurred.

Set SPI Clk mode				
Byte #	Name	Value	Description	Default value
0	Instruction	0x04	Set clock of SPI port	
1	SPPR	0xXX	SPI baud rate prescale divisor	0x07
2	SPR	0xXX	SPI baud rate divisor	0x07
3	Clock phase	0x00	Active high SPI clock	0x01
		0x01	Active low SPI clock	
4	Clock polarity	0x00	First edge on SPSCCK occurs at the middle of the cycle of a data transfer	0x01
		0x01	First edge on SPSCCK occurs at the start of the cycle of a data transfer	

Table 6: Set SPI clock mode

3.2.1.4 Get SPI Clock mode

This command is used to get the SPI clock status. The answer returned to the host by the USC-216 is the instruction value of the get command associated with the 4 bytes containing the value of each parameter. The USC-216 makes accessible the response immediately after it's treatment. The answer is stored in the converter until the API reads it.

Get SPI clock mode			
Byte #	Name	Value	Description
0	Instruction	0x05	Get clock status of SPI port

Table 7: Get SPI clock mode

Return Get SPI clock mode				
Byte #0	Byte #1	Byte #2	Byte #3	Byte #4
Instruction (0x05)	SPPR	SPR	Clock phase	Clock polarity

Table 8: Return Get SPI Clock mode

3.2.1.5 Set SPI slave select

This command is used to set the status of the slave select pin of the SPI port. This pin selects a slave device to send it some data and offers three possibilities: Automatic, High and Low. When the slave select is set to Automatic, the SPI module handles the SS pin. If the Low status or the High status is selected, this pin is used like an I/O pin and the converter drives the pin Low or High until another SPI slave select command is set. When a slave select command is sent to the USC-216, an Acknowledge is responded by the USC-216 to confirm that the change is really done. This Acknowledge represents the same command sent to the USC-216. It is stored in

converter until the API has read it. If the Acknowledge received by the host is not the same that the API sent, an error has occurred.

SPI Slave select				
Byte #	Name	Value	Description	Default value
0	Instruction	0x06	Set slave select	
1	Status	0x01	Automatic	0x01
		0x02	Drive high	
		0x03	Drive low	

Table 9: Set SPI slave select

3.2.1.6 *Get SPI slave select*

This command is used to get the SPI slave select. The message returned to host is the instruction value of the get command associated with the status of the slave select. The USC-216 makes accessible the response immediately after it's treatment. The answer is stored in the converter until the API reads it.

Get SPI slave select			
Byte #	Name	Value	Description
0	Instruction	0x07	Get slave select of SPI port

Table 10: Get SPI slave select

Return Get SPI slave select	
Byte #0	Byte #1
Instruction (0x07)	Status slave select

Table 11: Return Get SPI slave select

3.2.1.7 *Test SPI port*

This command is used to test the SPI port by sending the same byte in continuous loop. When this command is sent to the USC-216, an Acknowledge is responded by the USC-216 to confirm that the test is started. This Acknowledge represents the same command sent to the USC-216. The answer is stored in converter until the API reads it.

Test SPI port			
Byte #	Name	Value	Description
0	Instruction	0x08	Get clock of SPI port
1	State	0x00	State of the test (0x00 = Stop or 0x01 = Run)

Table 12: Test SPI port

3.2.1.8 *Request byte to send*

This command is used to ask the USC-216 how many bytes are empty in the SPI buffer. This buffer is used to store the data sent from host before sending it to the SPI port. It is 835 bytes in length. This request must be done before sending the data to the USC-216. If an overrun occurs the SPI buffer will reset to prevent a possible error. This command assures that the number of bytes sent won't exceed the length of the SPI buffer.

Request byte to send			
Byte #	Name	Value	Description
0	Instruction	0x09	Request how many byte is empty in SPI buffer

Table 13: Request byte to send

Return request byte to send	
Byte #0	Byte #1 & #2
Instruction (0x09)	Number of byte can send (0x0000 to 0x0343)

Table 14: Return Request byte to send

3.2.2 Data transmission

The firmware uses circular buffers to handle the data on both sides. The buffers are 835 bytes in length. The converter has complete control of the buffers and no code is necessary to control them. These next parts explain how to send and to receive data with the USC-216.

3.2.2.1 USB host to SPI device

The USC-216 may be set in master or slave mode. The method to send data to the USC-216 in both modes is the same but not for starting the SPI transmission. After configuring the SPI port, we must know how many bytes we can send to the USC-216 without overrunning the SPI buffer. This information is amassed with the “request byte to send” command viewed previously. The number of bytes sent in a packet is 64. If the data contains more than that, the driver automatically separates into packets of 64 bytes. In a data transmission from the USB host to the USC-216, the first packet contains the instructions associated with the number of bytes to send and the data. The next few packets contain the rest of the data.

Data packet (1 st)			
Byte #	Name	Value	Description
0	Instruction	0x01	The mode is write data
1:2	Nb of byte	0x0XXX	The number of byte to send (0 to 835 bytes)
3:63	Data	Byte	Byte to send to SPI port

Data packet (2 nd)			
Byte #	Name	Value	Description
0:63	Data	Byte	Byte to send to SPI port is more that 61 bytes

...

Data packet (X th)			
Byte #	Name	Value	Description
0:63	Data	Byte	The rest of the byte to send to SPI port

Table 15: Send data to USC-216

The USC-216 stored the data received in the circular buffer. In master mode, the converter starts the transmission after receiving the last byte of the last packet (all of the data). In slave mode, the INT pin must be active to indicate to the master device that the data is ready to send. The master device will enable the clock and the transmission will begin. This pin can be set before or after sending the data.

During the transmission, this same process can be executed to send data while in a continuous state.

3.2.2.2 SPI device to USB host

The USC-216 can be set in either master or slave mode. The method used by the USC-216 to send data to the USB host is the same for both modes (master and slave), but the method to read data on the SPI port is different.

In the slave mode, the SPI master device controls the SPI clock and the slave select line. When the SPI master device sends data to USC-216, this data is stored in a circular buffer and is immediately sent to the USB host.

In master mode, the USC-216 controls the SPI clock and the slave select line. Thus, to be able to read the SPI data from the slave device we must first activate the clock by sending data to it. This will in turn allow us to simultaneously read the SPI data. To enable the reading mode, the instructions of “send data to USC-216” must be changed but the transmission protocol should remain the same. When the new byte is received from the SPI port it is automatically sent to the USB host.

Data packet (1st)			
Byte #	Name	Value	Description
0	Instruction	0x11	The mode is read and write data
1:2	Nb of byte	0x0XXX	The number of byte to receive and to send (0 to 835 bytes)
3:63	Data	Byte	Byte to send to SPI port

....

Data packet (Xth)			
Byte #	Name	Value	Description
0:63	Data	Byte	The rest of the byte to send to SPI port

Table 16: Receive and send data from USC-216

During the reception phase, the same process can be executed to receive data in a continuous state.

Regardless of the mode, (master or slave) the format of the packet when the data is sent from the USC-216 to the USB host is always the same. All of the packets sent by the USC-216 to the host contain instructions, that at least one byte has been received by the SPI port. This is followed by info about the number of bytes in the packet and the bytes read. If the circular buffer contains more than 62 bytes then the data will be divided into multiple packets.

Data received from USC-216			
Byte #	Name	Value	Description
0	Instruction	0x0F	Data from SPI port
1	Nb of byte	0xXX	Number of byte in the packet
2:63	Data	Byte	Byte received from SPI port

Table 17: Data received from USC-216

3.2.3 Interface between the API and the driver

The USC-216 works in bulk mode, so the driver must be built to handle this type of communication. The method used to receive data in the bulk mode is polling type; this also allows the USB host to control all transfers. The driver does the USB enumeration automatically.

The link between the driver and the API is a globally unique identifier (GUID). The GUID is also a 128-bit number that uniquely identifies an object. Once the driver has been identified as a unique object by the GUID, the software is able to communicate and becomes fully functional.

The GUID of the driver of Addict Technologies is : `51fe8f30-ef12-11dd-ba2f-0800200c9a66`

This GUID must be used by the API to find the USB device.

The API must initialize one handle for the OUT pipe and one for the IN pipe. As seen previously, the OUT pipe corresponds to the endpoint 2 and the IN pipe to the endpoint 1. This handle will be used to communicate with the USC-216.[†]

When the device's handles are done, the API can exchange data with the USC-216. The two functions that are used are the WriteFile and ReadFile. Communication is established between the API and the USC-216 using these two functions. The next figures show a BULK transfer using IN or OUT transaction.

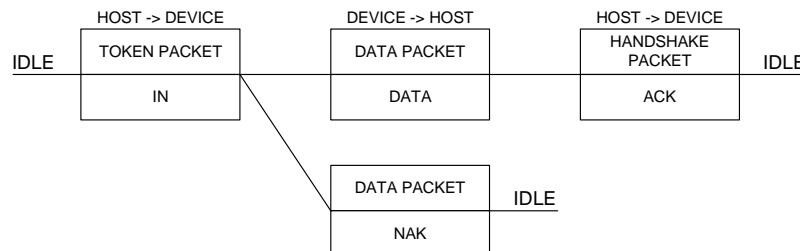


Figure 3: Bulk IN transaction[‡]

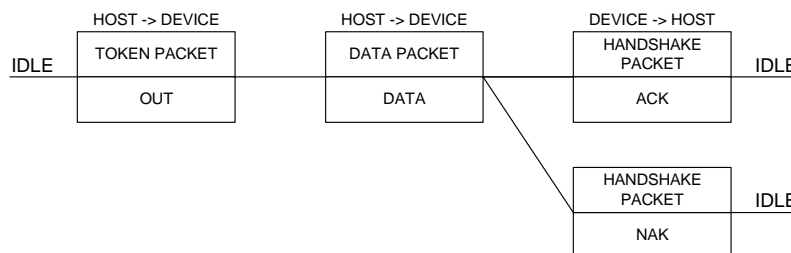


Figure 4: Bulk OUT transaction[‡]

When the USC-216 has data ready to be sent, the API must begin the communication with a ReadFile function, but the API is not alerted that something is ready to be read. To ease the

[†] Note: “USB COMPLETE SECOND EDITION” from Jan Axelson describes how to find a device and which function must be use.

[‡] This figure is taken form the “USB COMPLETE SECOND EDITION” from Jan Axelson.

process a thread can be declared. This thread asks the USC-216 if it has something to send with the token packet. The USC-216 responds with the data it has to send or with a no acknowledge (NAK).

3.3 Setup project

The next section explains how to install the USC-216 interface and the Addict Technologies driver on your computer. Followed by an explication on how to install and how to use the Interface.

3.3.1 Installation

To install the USC-216 Interface software on your computer; execute “SETUP.EXE” from the CD-ROM. Follow each step of the installation process to complete the installation.



Figure 5: USC-216 Interface setup welcome screen.

Figure 5 shows the setup welcome screen. Click “Next” to go to next step.

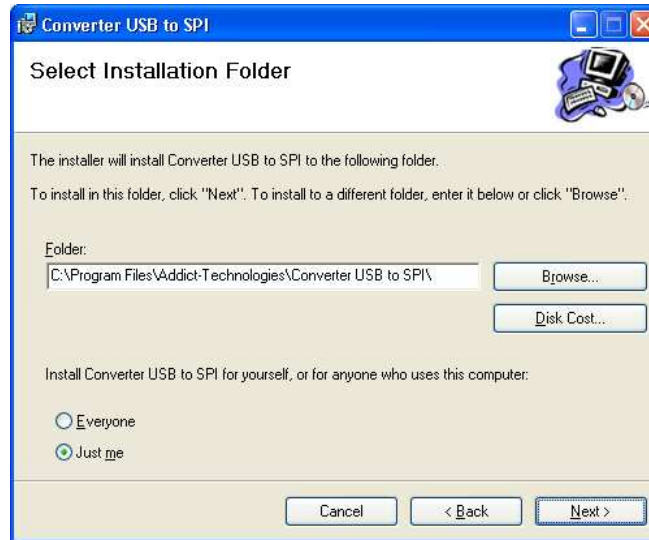


Figure 6: USC-216 Interface setup installation folder window.

Figure 6 shows the destination folder window. Enter the folder where you want to install the USC-216 Interface software and click “Next”.

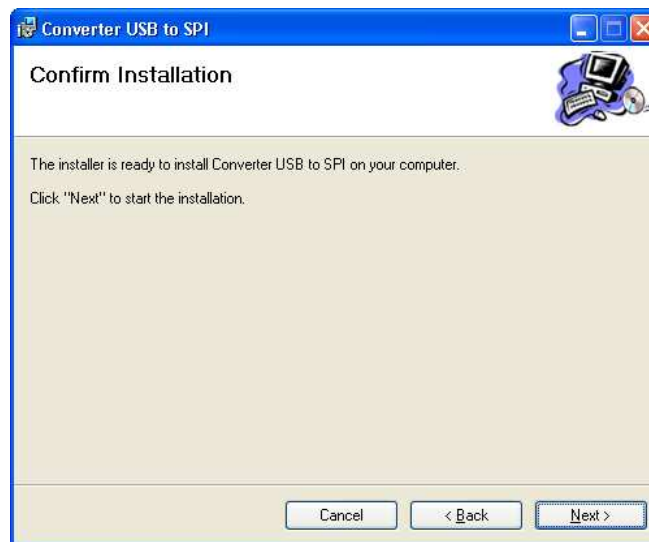


Figure 7: USC-216 Interface setup confirmation window.

Figure 7 shows the confirmation screen. Click “Next” to proceed with the installation or “Cancel” to stop the installation process.

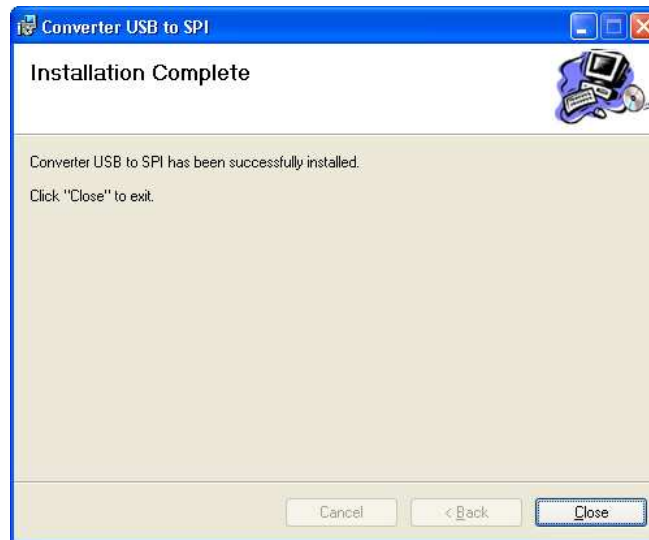


Figure 8: USC-216 Interface setup completed window.

Figure 8 shows the installation complete window. Click on "Close" to exit the USC-216 Interface software setup.

3.3.2 Utilization

The USC-216 Interface can configure, send or receive data from the SPI port. This interface is easy to use.

First of all, the USC-216 must be connected to the USB port and the driver must be installed. The SPI port must be powered on. When the API performs, this message will appear if a side is not connected. Check the USB connection and SPI power supply. Figure 9 shows the error message.



Figure 9: USC-216 is not found

When the USC-216 is disconnected from the host when the SPI is running, an error message will appear. Check the connection of the both sides and press ok. To leave the interface, press 'Annuler'. Figure 10 shows the error message.

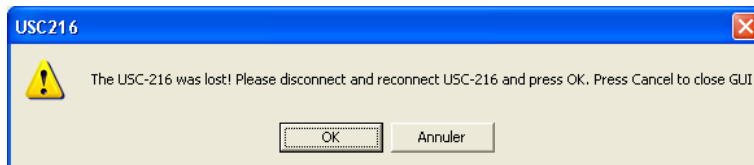


Figure 10: USC-216 was lost

When everything is properly connected on both sides, the USC-216 interface appears and the communication with the USC-216 can start. Figure 11 shows the interface. The next section describes how to use this interface.

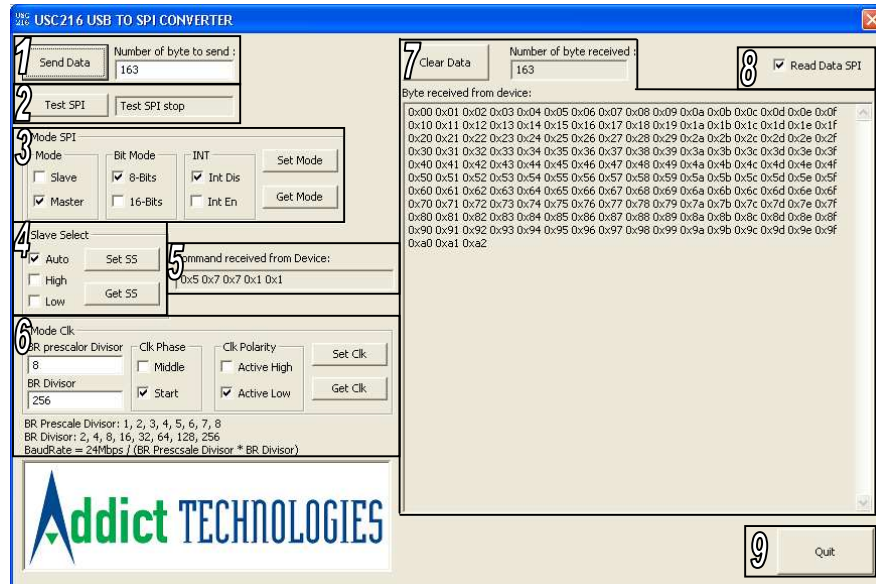


Figure 11: USC-216 interface

Case #1 is used to send data to the SPI port. In the white box, enter the number of bytes to send. The bytes sent begin with 0 and increment at each byte in the packet. The maximum number of bytes that the interface can send is 65500 bytes. Press the ‘Send Data’ button to send data.

Case #2 is used to start the SPI test. This test always sends the same byte to the SPI port. The ‘Test SPI’ button starts and stops the test. When the test is running the text changes to ‘Test SPI run’. This test is used to fix the SPI port.

Case #3 is used to configure the SPI port and the INT pin. When the API is started the default configuration is present on the screen. The radio button allows the user to change the mode, and the bit mode as well as the INT pin configuration. To send the SPI configuration press on the “Set Mode” button. To see the current SPI configuration press the ‘Get Mode’ button, this will update the radio buttons and the return message will appear in Case #5.

Case #4 is used to configure the slave select status. The slave select by default is Auto. The radio button allows the user to change the slave select status. To send the slave select configuration press on the ‘Set SS’ button. To see the current slave select configuration press on the ‘Get Mode’ button, this will update the radio buttons and the return message will appear in Case #5.

Case #5 is used to display the last command received from USC-216.

Case #6 is used to configure the SPI clock. The baud rate speed is set with two divisible factors and the choices of each divisible factor are displayed under the box. The radio buttons allow the user to change the phase and the polarity clock. To send an SPI clock configuration press on the ‘Set Clk’ button. To see the current configuration press on the ‘Get Clk’ button, this will update the radio buttons and the return message will appear in Case #5.

Case #7 is used to display the data received. The number of bytes received is displayed in the small box and the data is displayed in the big box. To clear both box press on 'Clear Data'.

Case #8 is used to read the data from the SPI port. When this radio button is checked the unit can read at the same time that data is sent to USC-216 in master mode. USC-216 sends data that it read from the SPI port to the API and the data is displayed in 'Byte received from device' in Case #7.

Case #9 is used to close the API.

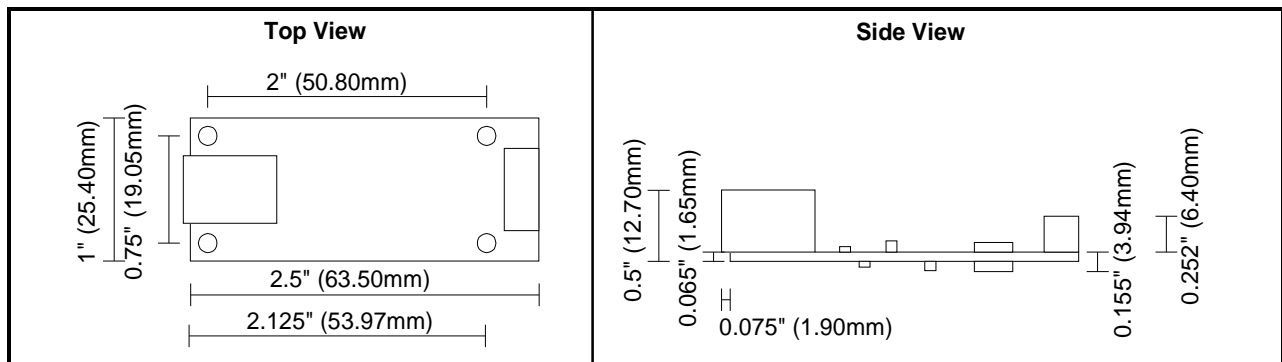
4. Specification

This section includes information about electrical and mechanical specifications of the USC-216 and associated connectors' characteristics.

Specifications	USC-216	Unit
Performance		
USB Compliance	2.0	
Data Rate		
Data < 835 bytes	1.5	Mbps
Data > 835 bytes	500	Kbps
Environment		
Operating Temp Range	-40 to 85°C	°C
Electrical		
Supply Voltage SPI Side	3.3	V
Supply Current SPI Side	35	mA
Supply Voltage USB Side	5	V
Supply Current USB Side	38	mA
Electrical isolation between USB side and SPI side	2.5	KV
Physical		
Size	2.5 x 1 x 0.58	In
	63.5 x 25.4 x 14.7	mm
Weight	0.026	lbs
	12	Grams

Connectors		
Pin	J1 ¹	J2 ²
1	VC	+3.3V
2	D+	MOSI
3	D-	GND
4	GND	SCLK
5		GND
6		SS
7		GND
8		MISO
9		GND
10		INT
11		GND
12		GPX

- 1- USB Type B connector
- 2- Molex 87831-1220



5. Warranty

Addict Technologies warrants the USC-216 Isolated USB to SPI converter to be free from malfunctions and defects in both materials and workmanship for one year from the date of purchase.

If the equipment does not function properly during the warranty period due to defects in either materials or workmanship, Addict Technologies will, at its option, either repair or replace the equipment without charge, subject to limitations stated herein. Such repair service will include all labour, as well as any necessary adjustments and / or replacement parts.

LIMITATIONS

The warranty becomes null and void if you fail to pack your USC-216 in a manner consistent with the original product packaging and damage occurs during shipment.

Addict Technologies makes no other warranties, express, implied, or of merchantability or fitness for a particular purpose for this equipment or software. Repair or replacement without charge are Addict Technologies only obligation under this warranty. Addict Technologies will not be responsible for any special, consequential or incidental damages resulting from the purchase, use, or improper functioning of this equipment regardless of the cause.

Depending on your geographical location, some limitations may not apply.

Addict Technologies
A division of Sysacom R&D plus inc
324 St-Paul st., Le Gardeur
Quebec, Canada
J5Z 4H9
www.sysacom.com