CY3240-I2USB

I2C-USB Bridge Guide

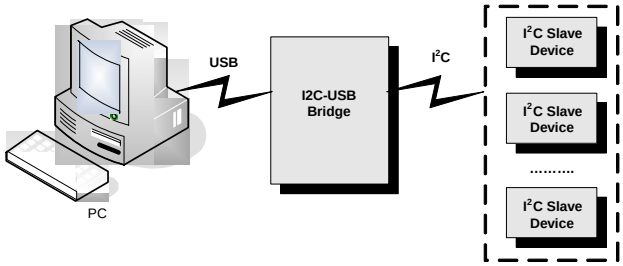
Author: Petro Kobluk, Valeriy Kyrynyuk
Associated Project: Yes
Associated Part Family: CY8C24894
Software Version: PSoC Designer™ v. 4.3
Associated Application Notes: AN2352

Introduction

The main purpose of the I2C-USB bridge is to test, tune, and debug programs that have an I^2C slave interface. This application can also be useful for data acquisition and regulation under PC control. A wide variety of devices can be connected to the PC using this bridge. Among them are: EEPROMs, realtime clocks, ADC/DAC converters, LCD displays, regulated DC/DC converters, port expanders and many other devices incorporating the I^2C interface. The number of devices that can be connected is constrained only by the I^2C address limit and physical ability of the I^2C bus.

The I2C-USB bridge is connected to the PC in the same way as an HID device and requires no additional driver when connected to a PC installed with Windows®. This I2C-USB bridge works as a master in the I^2C bus and is controlled by a PC program via the USB. In addition, a demonstration PC program is included with the project to demonstrate bridge operation with connected I^2C slaves.

Figure 1. I2C-USB Bridge Between PC and I^2C Slaves

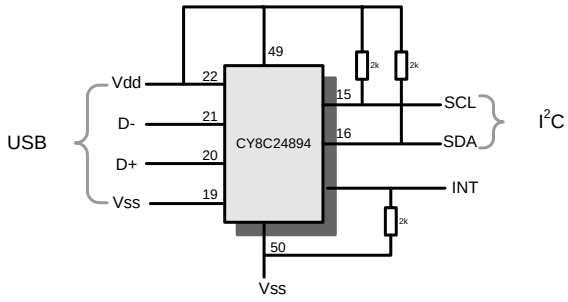


Because this I2C-USB bridge uses only a small subset of the PSoC device's resources, it has a substantial potential for implementation of additional functions or support of other interfaces.

Description of I2C-USB Bridge

The external pins are shown in Figure 2.

Figure 2. Simplified Schematic of I2C-USB Bridge



The device meets the requirements of the I^2C specification for standard and fast speed I^2C devices and the requirements for USB HID devices. The bridge is powered by the USB and consumes less than 500 mA. The device can be configured to several I^2C clock rates: 50 kHz, 100 kHz, and 400 kHz.

Several factors determine the maximum limit of I^2C devices that can be connected to the bus and the bus wire length: bus clock rate, capacitance and resistance of the bus wire, and capacitance of the I^2C device's pins. For details refer to the *THE I^2C -BUS SPECIFICATION* [1].

The USB communication function uses two packet types: one for input, and another for output data flow. The size of both packets is 64 bytes. The maximum bandwidth of this configuration is 64 bytes/s. This is enough for most I2C-USB bridge applications.

The "Int" pin is a pulled-down bidirectional pin that can be used as additional signal between the bridge and I^2C slave device, for sleep mode control, etc.

Bridge Circuit

Appendix B shows the bridge schematic.

The bridge has over-voltage and over-current protection on the USB and I^2C slave sides, provides three power modes for a connected slave device, and includes a level translator between “I2C” and “Int” pins on the slave and bridge sides, which allows it to work with any slave device regardless of the power supply voltage.

There are three power modes that can be chosen: external power (bridge does not provide power supply on Vdev pin), +3.3V, and +5V of bridge power supply (U1). The PSoC® device takes control over U1 by use of the “Enable” and “VoltCnt” pins. PSoC can detect the presence of a power supply on Vdev using the “Vsense” pin. The bridge also turns off the internal power supply if external voltage is connected.

U4 provides over-voltage protection of the USB line from the bridge side. U6 provides over-current protection of the bridge from the slave device. U5 provides voltage level translation for the I^2C lines.

Two LEDs show information about bridge status. “Status” LED (green) lights when bridge is enumerated into USB, and blinks during communication between the PC and the bridge. The “Dev Pow” LED (red) lights vividly if +5V of internal power is connected to Vdev, a weak light informs of a connection of +3.3V to Vdev, and the light blinks when external power is connected to Vdev.

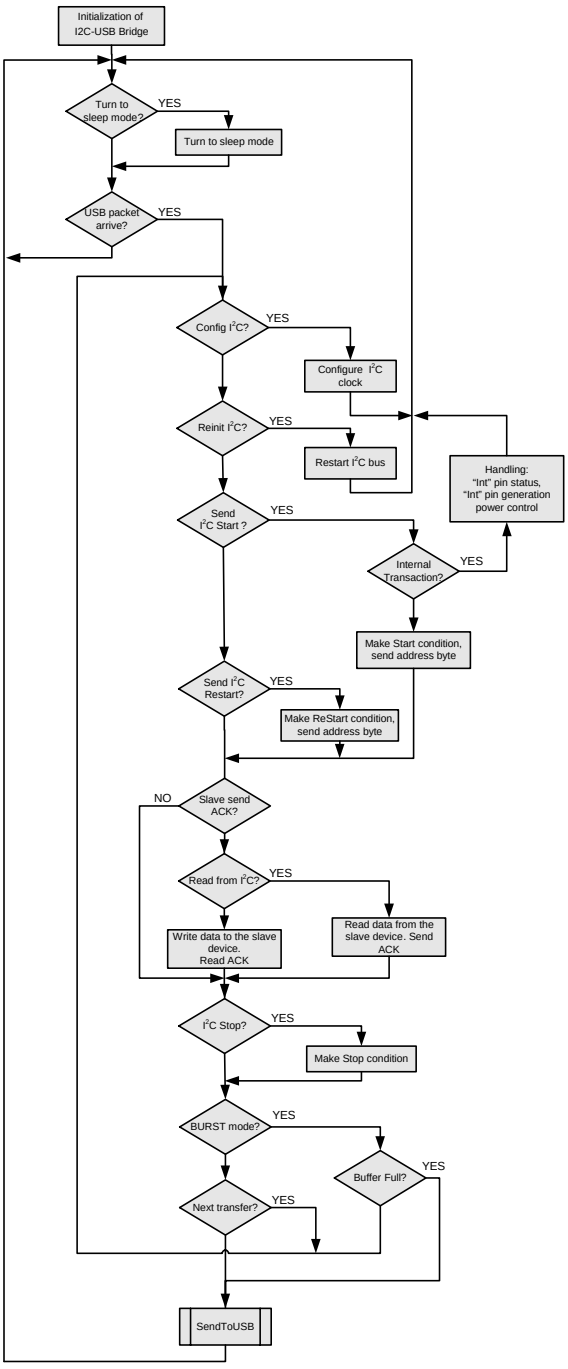
Internal Structure

The flowchart schematic of the bridge operation algorithm is depicted in Figure 3. The main program continuously checks for the presence of a new input packet from the USB. During processing of this packet, the I^2C transfer is completed and the result is sent in a USB output packet. Both packets should be formed according to the I2C-USB bridge protocol, which is described below.

The input packets can contain information about the reconfiguration or initialization of the I^2C bus driver. In this case, the I^2C transfer is not performed. The device also complies with the USB specification for switching connected devices into suspend mode. If there is no activity on the USB bus then the device will switch to sleep mode.

We examine the worst-case situation, which can occur during I^2C transfer. After sending the address byte, if the I^2C slave device does not confirm by ACK or if the bridge device receives NACK while sending the data bytes, then the program breaks the current I^2C transfer and outputs a USB packet containing information about the error status of the last transaction. If during the I^2C transfer the bus does not respond (starvation of I^2C bus) then the new incoming USB packet breaks the current transaction and the program jumps to processing the new packet.

Figure 3. Flowchart of Bridge Program



Protocol Description

The protocol describes the structure of USB input and output packets (Figure 4). The USB packet can contain several I^2C transfers. The total size of I^2C transfers into input/output packet cannot exceed 64 bytes. The 80h bit-mask into the “length” byte indicates whether the current transfer is the last into packet (bit = 0) or next transfer is present (bit = 1). This algorithm allows the bridge to attain the maximum speed and flexibility during communication with I^2C devices. The first byte in the transfer of an incoming packet is always a “control” byte. The “control” byte determines how the program will treat the subsequent bytes in the packet. The details of the “control” byte are shown in Tables 1 and 2.

Figure 4. Format of I^2C Transfer

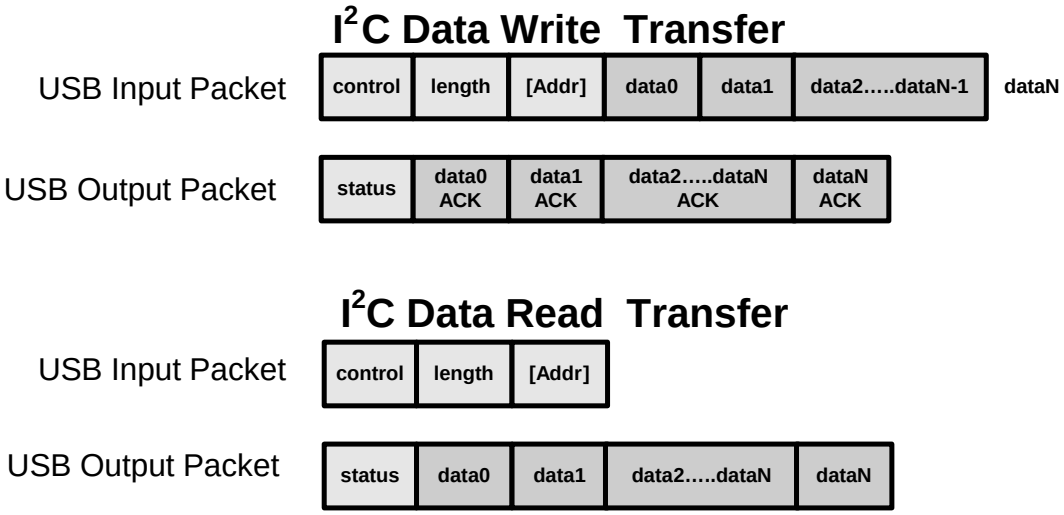


Table 1. “Control“ Byte Description

Bit	Description
0	Write/read I^2C operation
1	Make “Start” condition, send address byte
2	Make “ReStart” condition, send address byte
3	Make “Stop” condition after data transfer
4	Reinitialize I^2C bus
5	Make I^2C Reconfiguration
6, 7	Choose bus 00 – I^2C 01 – SPI (reserved) 10 – UART (reserved) 11 – LIN (reserved)

When bit 5 is set, bits 2 and 3 define the I^2C clock rate and all other low bits will not be processed. If bit 4 is set, the low 4 bits also will not be processed and there will be no I^2C transfer (refer to Figure 3).

Table 2. “Control” Byte Bits during I^2C Reconfiguration

Control Byte Bit 3, 2	I^2C Clock Rate
00	100 kHz
01	400 kHz
10	50 kHz
11	Reserved

The next byte of the incoming packet contains the length of the data array in bytes. This is the array of bytes to be written to or read from the I^2C slave device. The address byte, if present, is not included in the "length" byte. The maximum length of data in one packet is 61 bytes. If more than 61 data bytes need to be sent or received then the transfer will be performed using several USB packets. In the first packet the "Start" or "Restart" flags must be set and in the last packet the "Stop" flags in the "control" byte must be set.

After the "length" byte, an "Address" byte might be present. This byte holds the address of the I^2C slave that will be used in the I^2C transfer. The "Address" byte will be present in the packet only if the "Start" or "Restart" flag is set in the "control" byte. Its value must be in the range 0..127. The 128..255 address is reserved for internal bridge operation, as power control or "Int" pin control.

For Power and "Int" pin control, the address byte should be 0x80. If the "read operation" bit into the "control" byte is set, then the current power mode and current I^2C speed value will be returned. Also, the "status" byte will indicate the presence of Vdev voltage on the bridge (0x04 mask in "status" byte) and "Int" pin status (0x02 mask in "status" byte). When a write operation is performed, the data byte should contain the power mode byte. Power mode byte options include:

- 0: external power
- 1: +5V internal power
- 2: +3.3V internal power

If the "Restart" bit is set into the "control" byte then the "Int" pin generates a short interrupt positive pulse, which can be used to awaken the slave device.

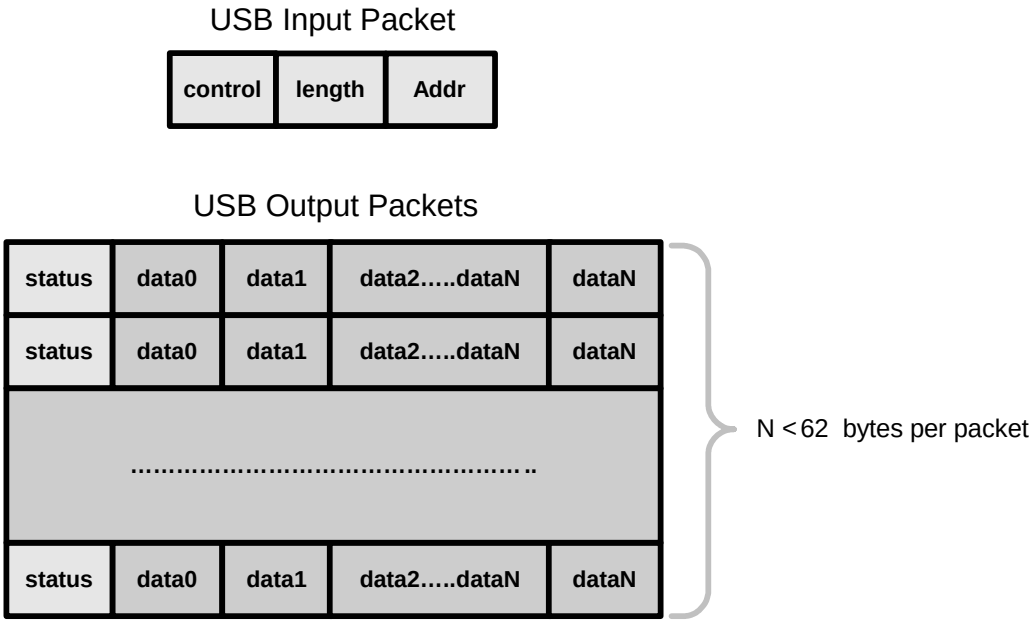
Interpretation of data in the input/output packet will depend on whether the I^2C performs a read or write operation. If data is to be written to the I^2C slave then the data bytes will go immediately after the "address" or "length" byte. If data is to be read from the I^2C slave then the input packet will contain no additional information.

The first byte in the transfer of the output packet always contains the "status" byte of the I^2C transfer. If status is zero then the bridge cannot find the I^2C slave device and all transfer with the device is canceled. In this case, all packet bytes that follow must be ignored. If the "status" byte is non-zero then the slave device is present and the bytes that follow are valid. If data was written to the slave device then the output frame contains an array of bytes that indicate the ACK ('1') or NACK ('0') status of each byte that was written. The last byte received by a slave device has a NACK status (zero value). All bytes that follow are ignored by the slave (perhaps the device's read buffer is full). The length of the response data array is the same length as the input packet data array.

If there is read data from the I^2C slave device then the bytes containing this data follow the "status" byte in the output packet. The I2C-USB bridge sends an ACK signal while reading the data from the slave device and sends a NACK once the last byte has been read.

Also, BURST mode was implemented into the bridge. This mode is necessary for high-speed data acquisition and allows the device to read 1000..5000 samples per second from the I^2C slave. To switch the bridge to this mode only the one header transfer should be sent with 40h bit set in the "length" byte – and after this the bridge will switch to BURST mode and continuously send the results of the header transfer into the output packet. The output packet contains several transfer results. For example, if I^2C transfer reads 8 bytes from the slave, the output packet will contain $N = INT(\frac{62}{1+8}) = 6$ read slave samples. Each time the output buffer is filled it will be sent to the USB host.

Figure 5. BURST Mode Format



Appendix A. References

[1]. THE I2C-BUS SPECIFICATION.

http://www.semiconductors.philips.com/acrobat/literature/9398/39340011.pdf

Appendix B. Schematic of Cypress I2C-USB Bridge

Figure 6. Schematic of Cypress I2C-USB Bridge

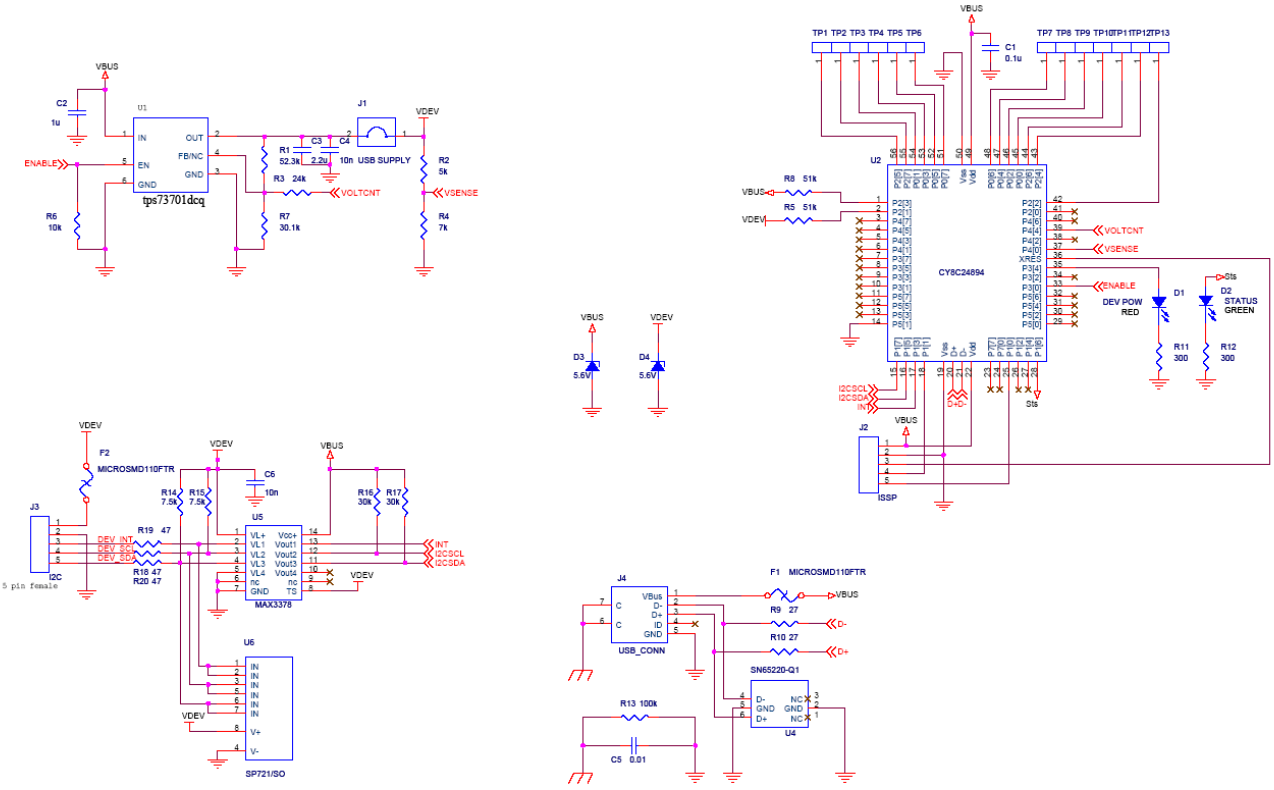
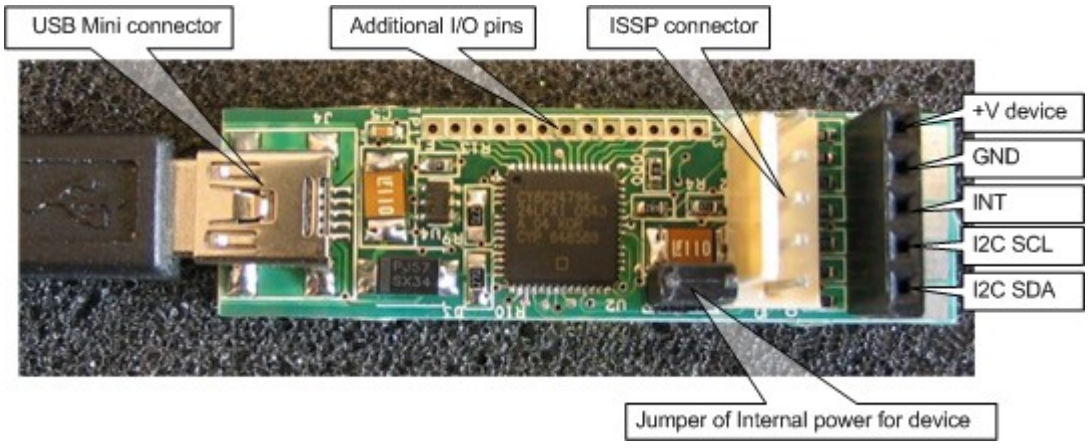


Figure 7. Cypress I2C-USB Bridge Board



Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com

© Cypress Semiconductor Corporation, 2006. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.