

Project Name: Example_Half_Duplex_UART

Programming Language: C

Associated Part Families: CY8C24x23, CY8C27x43, CY8C29x66, CY8C24x94, CY8C21x34

Software Version: PSoC Designer 5.0

Project Objective

To implement an 8 bit Half Duplex UART by using Dynamic Reconfiguration and a single digital communication block.

Overview

This example project demonstrates a serial port interface between a PSoC and the PC with a baud rate of 19.2Kbps. Hyper Terminal is used as a user interface on the PC side to view the characters transmitted and received over the serial port. Half Duplex UART is implemented using Dynamic Reconfiguration where the same digital block is shared by the Transmitter (TX8) and the Receiver (RX8).

User Module List and Placement

The following table lists user modules used in this project and the hardware resources occupied by each user module.

User Module	Placement
Counter8_1	DBB01 (Base Configuration)
RX8_1	DCB02 (Receiver Configuration)
TX8_1	DCB02 (Transmitter Configuration)

User Module Parameter Settings

The following tables show the user module parameter settings for each of the user modules used in the project.

Counter8_1		
Parameter	Value	Comments
Clock	VC1	This parameter is left at its default setting. The clock setting is going to be over ridden by the ClockSync parameter
ClockSync	Use SysClock Direct	This selects SysClk as the clock source. This setting also overrides the "Clock" parameter.
Enable	High	This enables the counter.
CompareOut	None	The CompareOut output is not used in this application.
TerminalCountOut	None	The TerminalCountOut is not used in this application.
Period	155	The divider of the counter is set to 156 (Period + 1). This generates a 153.846KHz at the counter output. This in turn sets the baud rate of the UART to 19230 bps
CompareValue	78	This parameter sets the Duty Cycle of the Counter Output. Duty cycle is set to 50%
CompareType	Less Than or Equal To	This value sets the divider of the counter as Period+1.
InterruptType	Terminal Count	This parameter is not used
InvertEnable	Normal	This sets the Enable of Counter to Active High

Notes:

- The Counter8_1 UM has been included in this project as a baud rate generator for the RX8_1 and TX8_1 UMs. This provides more flexibility as various baud rates may be obtained by changing the Period value of the counter on the fly. Alternately VC1 / VC2 and VC3 may also be used as the clock source to the RX8 and TX8 modules thus saving a digital block
- The Counter8_1 UM is placed in the base configuration and is never unloaded. This ensures that this module is running throughout the application irrespective of any other configuration being loaded or unloaded. Hence the clock supply to the transmitter and receiver UMs is continuous all the time.

RX8_1		
Parameter	Value	Comments
Clock	DBB01	The clock to the RX8 is derived from the Counter
Input	Row_0_Input_2	Port pin P1.6 has been assigned as the input for the half duplex UART. The input from this pin to the RX8_1 block is routed via the Row_0_Input_1 net.
ClockSync	Sync To SysClock	As the clock to the RX8 is derived from SysClk, the clock sync has to be set to "SyncToSysClk"
RxCmdBuffer	Disable	Buffer for Command Processing is disables.
RxBufferSize	16 Bytes	NA
RX Output	None	RX output is not used
Data Clock Out	None	Data clock is not used
InvertInput	Normal	The input signal is not inverted.

Notes:

- The clock to the RX8 user module should be 8 times the desired baud rate

TX8_1		
Parameter	Value	Comments
Clock	DBB01	The clock to the transmitter module must be 8 times the output baud rate. DBB01 output clock is 153.846KHz which is 8 times 19.230kbps
Output	Row_0_Output_3	Port pin P2.7 has been assigned as the output for the half duplex UART. The output from the TX8_1 block is routed to this pin via the Row_0_Output_3 net and Global_Out_Even7.
TX Interrupt Mode	TxRegEmpty	This parameter is not used and hence is left at its default value.
ClockSynch	Sync to SysClock	Clock is synchronized with the SysClock.
Data Clock Out	None	

Notes:

- The clock to the TX8 user module should be 8 times the desired baud rate

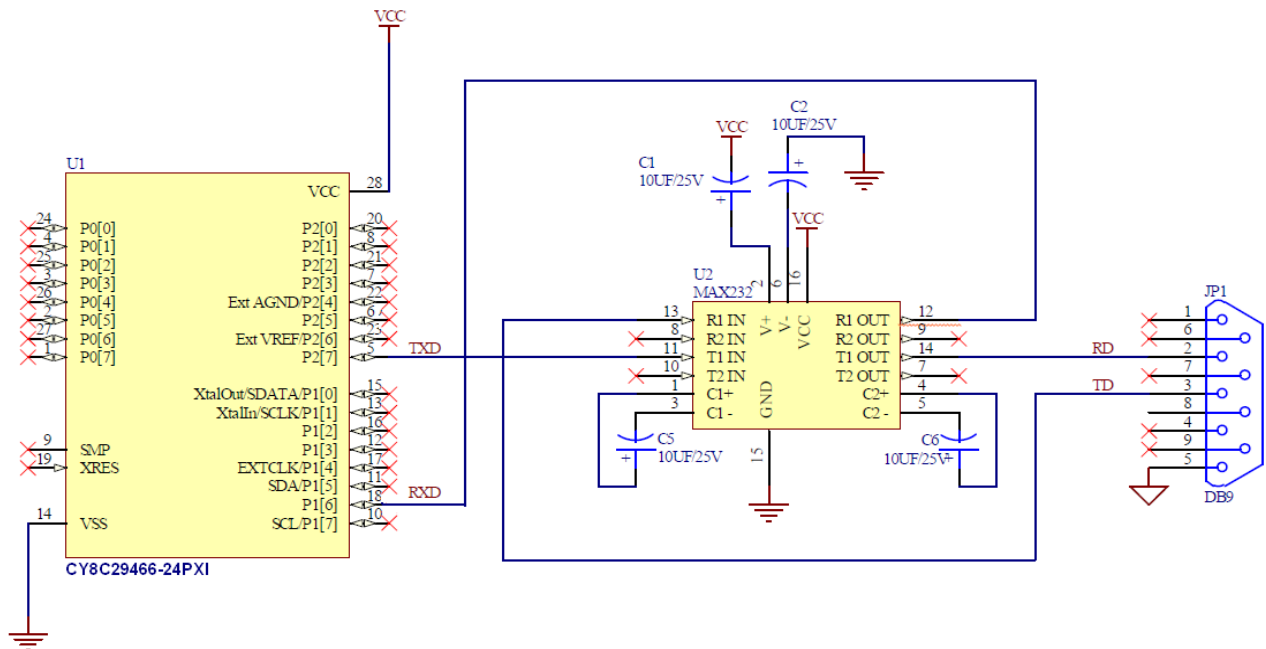
Global Resources

Important Global Resources		
Parameter	Value	Comments
CPU Clock	SysClk/2	Sets the CPU frequency to 12MHz

- All other global resources are left at their default, as they are not specific to this project.

Hardware Connections

The Schematic of the project is shown below:



MAX232, an RS232 transceiver is used to translate the +/-10V RS232 signals to TTL level signals of the PSOC. JP1 is a 9 pin Female serial port connector which is used to connect the project with a PC.

The project can be tested using CY3210 – PSOC Eval1 board. This board has an RS232 transceiver and also a Serial port connector. To test the project using the CY3210 board, the following connections have to be made.

- Connect P16 of J8 to RX of J13
- Connect P27 of J7 to TX of J13

Operation

On reset, all hardware settings from the device configuration are loaded into the device and main.c is executed. Before the program enters main.c, the base configuration with the Counter8_1 gets loaded. The following operations are performed in main.c.

1. Counter8_1 is started in the beginning of main.c. Once started, the Counter8_1 is never stopped, nor the base configuration unloaded in the project. This ensures that the Counter8_1 is always running generating the baud clock. As the ClockSys parameter of the counter is set to "Use Sysclk Direct", the clock input to the counter is 24MHz. The counter divides the 24MHz clock by 156, generating 153.846KHz clock. This is 8 times the final baud rate of 19.2Kbps that we wish our UART to communicate. Alternately, the baud rate may also be generated using various combinations of VC1, VC2 and VC3.
 - VC1 = 12; VC2 = 13; Clock input to Counter = VC2
 - VC1 =NA; VC2 =NA; VC3 Source = SysClk; VC3 Divider = 156; Clock input to Counter = VC3

The advantage of using VC1 / VC2 / VC3 is that the digital block used for the Counter8_1 is saved.

2. A "1" is written to Bit-7 of PRT2DR register. This is a workaround for the following condition. When the transmitter configuration is unloaded, P2[7] is still connected to the global bus and to the Row_0_Output3 net. When the TX8 block is unloaded from DCB02, the output P2[7] goes low. This will be treated as a Start bit by the PC and will result in a framing error condition. To avoid this condition, before unloading the Transmitter configuration, P2[7] is disconnected from the Global bus by clearing Bit-7 of PRT2GS register. The 1 written to Bit-7 of PRT2DR register will maintain a HIGH state on the TX pin, thus

preventing the false Start bit. Whenever the Transmitter configuration is loaded, P2[7] is connected to the Global bus.

3. The Receiver configuration is loaded and RX8_1 is enabled with no parity.
4. The RX8_1 is polled for an input character, by using the RX8_1_cGetChar function
5. On receiving a character, the Receiver configuration is unloaded
6. The Transmitter configuration is loaded
7. The TX pin is connected to the Global bus
8. The received character is transmitted on the Serial bus
9. The TX pin is disconnected from the Global bus
10. The Transmitter configuration is unloaded.
11. Steps #3 to #10 are repeated in an infinite loop

Testing the Project

To test the project, Hyper-terminal (or any other terminal program) may be used. The following explains how Hyper-terminal may be configured in Windows.

1. Connect the CY3210 board to the PC serial port using a Serial port cable.
2. Start Hyper Terminal using Start -> Program Files -> Accessories -> Communication -> HyperTerminal
3. Enter a Name for the connection, for example "COM1_19200" and select OK.
4. In the "ConnectTo" option select the desired serial port (for example, COM1) from 'Connect using' list Click and click OK
5. In the COM1 properties, configure the following parameters
 - Bits per second = 19200
 - Data bits = 8
 - Parity = None
 - Stop Bits = 1
 - Flow Control = None
 - Click OK
6. At this point, Hyper Terminal will connect to COM1. We have some more configurations to do before the Project may be tested.
7. Click on the "Disconnect" Icon.
8. Click on File -> Properties -> Settings -> ASCII Setup
9. Enable "Echo typed characters locally"
10. Click OK
11. Click on the "Connect" icon
12. Now, Hyper Terminal is ready to be used with the UART example project

Now type characters and see them get echoed on the Hyper-terminal window. Each character will be displayed twice on the HyperTerminal window. The first is the local echo and the second is the character echoed by the PSoC.

Example - PSoC is a registered trademark of Cypress Semiconductor Corp. "Programmable System-on-Chip," PSoC Designer, and PSoC Express are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

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, 2008. 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.

This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. 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' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.