
tutorials
by
Cytron
Technologies

UART – Universal Asynchronous Receiver and Transmitter

By: WW Kong

RH2T Magazine Vol.3, Dec 2010

Were you ever disheartened when you ran out of ideas interfacing your microcontroller with PC? Well, not anymore as UART, despite being one of the pioneers in communication protocol happens to be one of the solutions

1.0 Introduction

The Universal Asynchronous Receiver/Transmitter (UART) controller is the key component of the serial communications subsystem of a computer. UART is also a common integrated feature in most microcontrollers. The UART takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes. Serial transmission of digital information (bits) through a single wire or other medium is much more cost effective than parallel transmission through multiple wires. Communication can be “full duplex” (both send and receive at the same time) or “half duplex” (devices take turns transmitting and receiving).

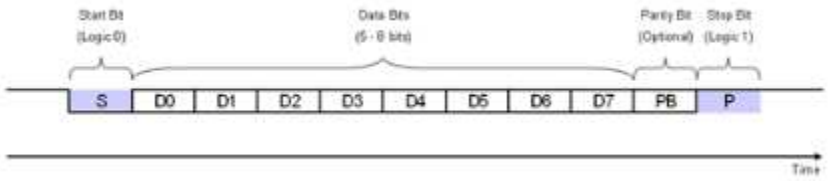
2.0 The Asynchronous Receiving and Transmitting Protocol

Asynchronous transmission allows data to be transmitted without the sender having to send a clock signal to the receiver. In this case, the sender and receiver must agree on timing parameters (Baud Rate) prior transmission and special bits are added to each word to synchronize the sending and receiving units. In asynchronous transmission, the sender sends a Start bit, 5 to 8 data bits (LSB first), an optional Parity bit, and then 1, 1.5 or 2 Stop bits.

When a word is passed to the UART for asynchronous transmissions, the Start bit is added at beginning of the word. The Start bit is used to inform the receiver that a word of data is about to be send, thereby forcing the clock in the receiver to be in sync with the clock in the transmitter. It is important to note that the frequency drift between these two clocks must not exceed 10%. In other words, both the transmitter and receiver must have identical baud rate.

After the Start bit, the individual bits of the word of data are sent, beginning with the Least Significant Bit (LSB). When data is fully transmitted, an optional parity bit is sent to the transmitter. This bit is usually used by receiver to perform simple error checking. Lastly, Stop bit will be sent to indicate the end of transmission.

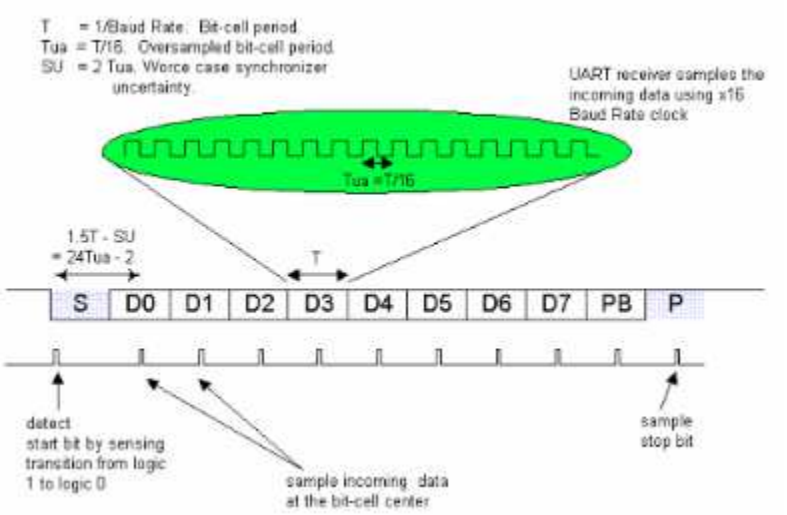
When the receiver has received all of the bits in the data word, it may check for the Parity Bits (both sender and receiver must agree on whether a Parity Bit is to be used), and then the receiver searches for a Stop Bit. If the Stop Bit does not appear when it is supposed to, the UART considers the entire word to be garbled and will report a Framing Error to the host processor when the data word is read. Common reason for the occurrence of Framing Error is that the sender and receiver clocks were not running at the same speed, or that the signal was interrupted.



Basic UART packet format: 1 Start bit, 8 data bits, 1 Parity bit and 1 Stop bit

Every operation of the UART hardware is controlled by a clock signal which runs at much faster rate than the baud rate. For example, the popular 16450 UART has an internal clock that runs 16 times faster than the baud rate. This allows the UART receiver to sample the incoming data with granularity of 1/16 the baud-rate period and has greater immunity towards baud rate error.

The receiver detects the Start bit by detecting the voltage transition from logic 1 to logic 0 on the transmission line. In the case of 16450 UART, once the Start bit is detected, the next data bit's "center" can be assured to be 24 ticks minus 2 (worse case synchronizer uncertainty) later. From then on, every next data bit center is 16 clock ticks later.



Data sampling point by the UART receiver

Transmitting and receiving UARTs must be set at the same baud rate, character length, parity, and stop bits for proper operation. The typical format for serial ports used with PC connected to modems is 1 Start bit, 8 data bits, no Parity and 1 Stop bit.

3.0 The Physical Layer Standards

So far, we have discussed the software protocol of the UART. How about the physical layer standards? There are actually quite a number of different standards that utilizes similar protocol. For instances, TTL level UART, RS-232, RS-422, RS-485 and etc. We will only discuss about TTL level UART and RS-232 in this article.

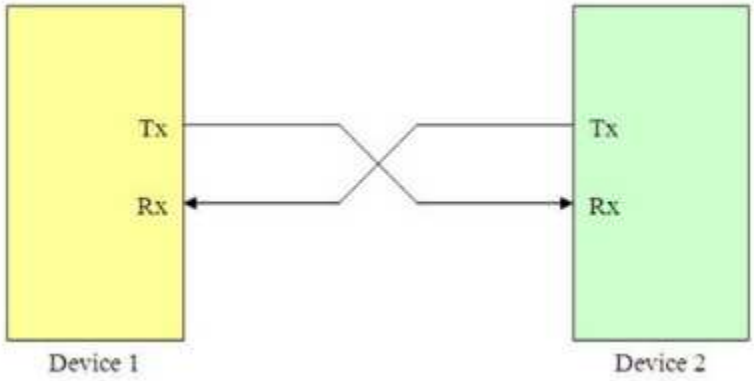
3.1 TTL level UART

Most microcontrollers with UART uses TTL (Transistor-transistor Logic) level UART. It is the simplest form of UART. Both logic 1 and 0 are represented by 5V and 0V respectively.

Logic	Voltage
Low	0V
High	5V

Voltage level for TTL level UART

The TTL level UART is commonly used in the communications between microcontrollers and ICs. Only 2 wires are required for the full duplex communications as illustrated in the picture below.



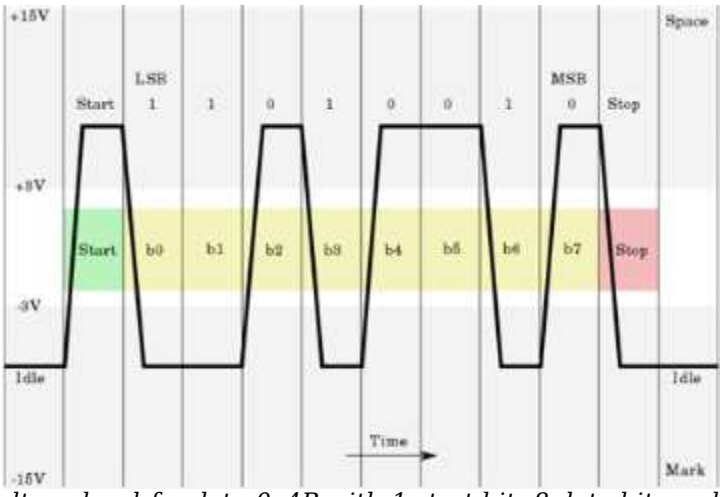
UART Communication Between 2 devices

3.2 RS-232

RS-232 (Recommended Standard 232) is a standard for serial binary data signals connecting between a Data Terminal Equipment (DTE) and a Data Communication Equipment (DCE). It is commonly used in computer serial ports. One of the significant differences between TTL level UART and RS-232 is the voltage level. Valid signals in RS-232 are ±3 to – ±15V, and signals near 0V is not a valid RS-232 level.

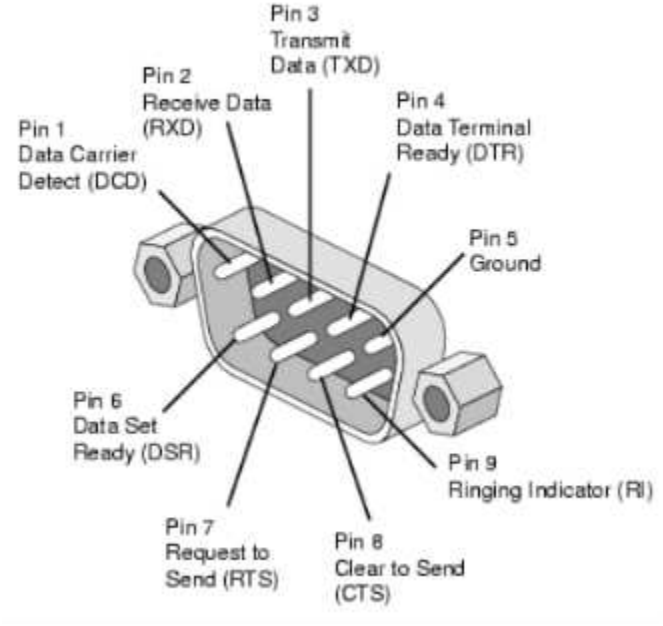
Logic	Voltage
Low	+3 to +15V
High	-3 to -15V

Voltage level for RS-232



RS-232 voltage level for data 0x4B with 1 start bit, 8 data bits and 1 stop bit

Besides voltage level, the RS-232 also has a few extra pins specifically designed for the communication between PC and modem. The pinouts of the DB-9 and their functions are shown below.



Pinouts of a serial port

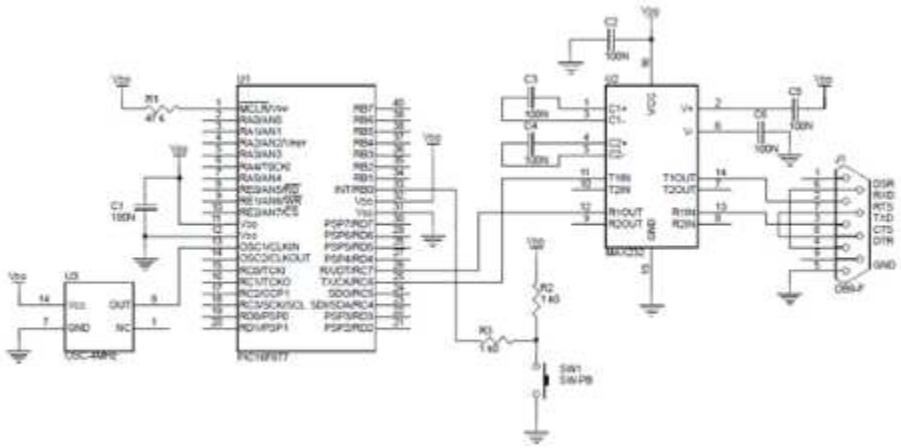
Name	Pin	Description
Transmitted Data (TxD)	3	Serial data output
Received Data (RxD)	2	Serial data input
Request to Send (RTS)	7	This line informs the DCE (Modem) that the DTE (PC) is ready to exchange data
Clear to Send (CTS)	8	This line indicates that the DCE is ready to exchange data
Data Terminal Ready (DTR)	4	Asserted by DTE to indicate that it is ready to be connected
Data Set Ready (DSR)	6	Asserted by DCE to indicate the DCE is powered on and is ready to receive commands or data for transmission from the DTE
Data Carrier Detect (DCD)	1	Asserted by DCE when a connection has been established with remote equipment
Ring Indicator (RI)	9	Asserted by DCE when it detects a ring signal from the telephone line

Pinouts and description of a serial port

4.0 Interfacing between TTL level UART and RS-232

From previous discussions, we know that microcontrollers make use of TTL level UART while the PC serial port uses RS-232. Since both standards uses similar software protocol, both of them are able to communicate via UART. However, because of the differences in voltage level and polarity, we will need a level shifter to interface the TTL

level UART with the RS-232. Nowadays, this can be easily done with the commonly available IC such as the MAX232 from Maxim.

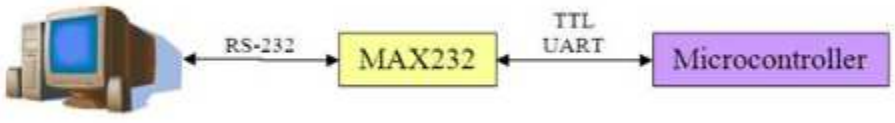


Example of interfacing microcontroller to PC serial port

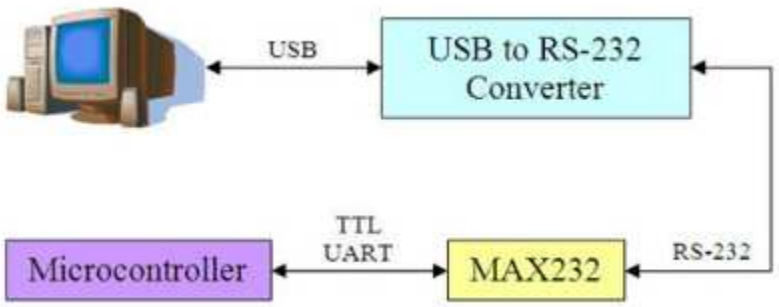
5.0 Summary

In short, UART is the simplest form of communication between microcontroller and PC. However, due to the mushrooming growth of technology, serial port is slowly being replaced by other means of communication port. Nevertheless, serial communication is still possible even without a physical serial port on your PC. For example, the USB can be treated as a serial port after the signal from microcontroller is converted using the USB to RS-232 converter. In order to gain more understanding on this converter, feel free to refer to the USB to UART converter (UC00A) from Cytron as it is a readily available device that provides communication between UART and USB via the USB to RS-232 converter. If you have any inquiry, please do come to our technical forum to discuss

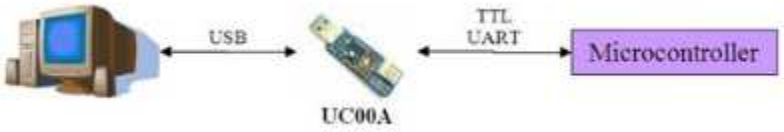
USB to UART converter (UC00A) from Cytron.



Conventional way of interfacing microcontroller to PC via serial port



Interfacing microcontroller to PC via USB to RS-232 Converter



Interfacing microcontroller to PC via USB to UART Converter (UC00A)

Reference:

~~http://www.cmosoxod.com/micro_uart.htm~~ (Error!)

http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter

http://retired.beyondlogic.org/serial/serial.htm

Related Posts:

-  Low Cost Wireless UART
-  USB to UART Converter, UC00A
-  Crazy Sales 2011
-  SD02B Stepper Driver Features UART Interface
-  LCD: Interfacing with PIC Microcontrollers Part 2

Published by: ober on February 16, 2012.

2 Responses to "UART – Universal Asynchronous Receiver and Transmitter"

1. *J Hughes* says:
   July 8, 2013 at 9:53 am

   Your references:

   http://www.beyondlogic.org/serial/serial.htm

   AND

   http://www.cmosexod.com/micro_uart.htm

   are 404!

2. *cytron* says:
   July 20, 2013 at 11:43 pm

   Thanks for the info!

Leave a Reply

Name (required)

Mail (will not be published) (required)

Website

Submit Comment

Help us improve the wiki Send Your Comments