International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-2 Issue-2, December 2012

Study of Uart Transimitter in Microcontroler

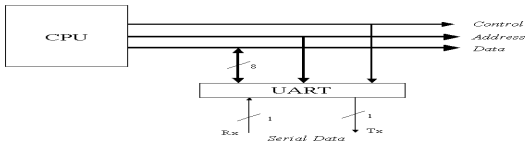
Manchineni Vijay Kumar, Suresh Angadi

Abstract:- UART- Universal Asynchronous Receiver Transmitter, generally it is used for better transmission of serial data that is it either transmit or receives data serially with the help of shift register. It consist frame format, one start bit (usually low), 5-8 data bit, one optional parity bit and one stop bit (opposite polarity of start bit). Asynchronous means by using start and stop bit we transmit data, there is no need of sending (PAD) that is ASCII (SYN) for synchronizing transmitter and receiver. It transmits 9600 to 38400bps for transmitting data bit. Whole process of serial transmission is based upon the principle of shift register.

Keywords: UART, RDR, USART, DTE, DCE

I. INTRODUCTION

The Universal Asynchronous Receiver Transmitter (UART) is a popular and widely-used device for data communication in the field of telecommunication. There are different versions of UARTs in the Industry. (UART) An integrated circuit used for serial communications, containing a transmitter (parallel-to serial converter) and a receiver (serial-to-parallel converter), each clocked separately.



Another status register bit says whether the UART has received a byte from the serial line, in which case the computer should read it from the Receive Data Register (RDR). If another byte is received before the previous one is read, the UARTs will signal an “overrun” error via another status bit. The UARTs may be set up to interrupt the computer when data is received or when ready to transmit more data.

II. THE UART: WHAT IT IS AND HOW IT WORKS

The Universal Asynchronous Receiver/Transmitter (UART) controller is the key component of the serial communications subsystem of a computer. 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 is commonly used with modems and for non-networked communication between computers, terminals and other devices.

There are two primary forms of serial transmission: Synchronous and Asynchronous. Depending on the modes that are supported by the hardware, the name of the communication sub-system will usually include a A if it supports Asynchronous communications, and a S if it supports Synchronous communications. Both forms are described below. Some common acronyms are: UART Universal Asynchronous Receiver/Transmitter USART Universal Synchronous-Asynchronous Receiver/Transmitter

2.1) Synchronous Serial Transmission

Synchronous serial transmission requires that the sender and receiver share a clock with one another, or that the sender provide a strobe or other timing signal so that the receiver knows when to “read” the next bit of the data. In most forms of serial Synchronous communication, if there is no data available at a given instant to transmit, a fill character must be sent instead so that data is always being transmitted. Synchronous communication is usually more efficient because only data bits are transmitted between sender and receiver, and synchronous communication can be more costly if extra wiring and circuits are required to share a clock signal between the sender and receiver.

A form of Synchronous transmission is used with printers and fixed disk devices in that the data is sent on one set of wires while a clock or strobe is sent on a different wire. Printers and fixed disk devices are not normally serial devices because most fixed disk interface standards send an entire word of data for each clock or strobe signal by using a separate wire for each bit of the word. In the PC industry, these are known as Parallel devices.

The standard serial communications hardware in the PC does not support Synchronous operations. This mode is described here for comparison purposes only.

2.2) Asynchronous Serial Transmission

Asynchronous transmission allows data to be transmitted without the sender having to send a clock signal to the receiver. Instead, the sender and receiver must agree on timing parameters in advance and special bits are added to each word which are used to synchronize the sending and receiving units.

When a word is given to the UART for Asynchronous transmissions, a bit called the "Start Bit" is added to the beginning of each word that is to be transmitted. The Start Bit is used to alert the receiver that a word of data is about to be sent, and to force the clock in the receiver into synchronization with the clock in the transmitter. These two clocks must be accurate enough to not have the frequency drift by more than 10% during the transmission of the remaining bits in the word. (This requirement was set in the days of mechanical teleprinters and is easily met by modern electronic equipment.) After the Start Bit, the individual bits of the word of data are sent, with the Least Significant Bit (LSB) being sent first. Each bit in the transmission is transmitted for exactly the same amount of time as all of the other bits, and the receiver “looks” at the wire at approximately halfway through the period assigned to each bit to determine if the bit is a 1 or a 0.

Manuscript published on 30 December 2012.
* Correspondence Author (s)
Manchineni Vijay Kumar*, Student of KL University, Department of ECE, KL University, Vaddeswaram, Vijayawada, Andhra Pradesh, India.
Suresh Angadi, Assistant Professor, Department of ECE, KL University, Vaddeswaram, Vijayawada, Andhra Pradesh, India.

For example, if it takes two seconds to send each bit, the receiver will examine the signal to determine if it is a 1 or a 0 after one second has passed, then it will wait two seconds and then examine the value of the next bit, and so on.

The sender does not know when the receiver has “looked” at the value of the bit. The sender only knows when the clock says to begin transmitting the next bit of the word.

When the entire data word has been sent, the transmitter may add a Parity Bit that the transmitter generates. The Parity Bit may be used by the receiver to perform simple error checking. Then at least one Stop Bit is sent by the transmitter.

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 looks 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. The usual cause of a Framing Error is that the sender and receiver clocks were not running at the same speed, or that the signal was interrupted.

Regardless of whether the data was received correctly or not, the UART automatically discards the Start, Parity and Stop bits. If the sender and receiver are configured identically, these bits are not passed to the host.

If another word is ready for transmission, the Start Bit for the new word can be sent as soon as the Stop Bit for the previous word has been sent.

Because asynchronous data is “self are synchronizing”, if there is no data to transmit, the transmission line can be idle.

2.3) Other UART Functions

In addition to the basic job of converting data from parallel to serial for transmission and from serial to parallel on reception, a UART will usually provide additional circuits for signals that can be used to indicate the state of the transmission media, and to regulate the flow of data in the event that the remote device is not prepared to accept more data. For example, when the device connected to the UART is a modem, the modem may report the presence of a carrier on the phone line while the computer may be able to instruct the modem to reset itself or to not take calls by raising or lowering one more of these extra signals. The function of each of these additional signals is defined in the EIA RS232-C standard.

2.4) The RS232-C and V.24 Standards

In most computer systems, the UART is connected to circuitry that generates signals that comply with the EIA RS232-C specification. There is also a CCITT standard named V.24 that mirrors the specifications included in RS232-C.

2.4.1) RS232-C Bit Assignments (Marks and Spaces)

In RS232-C, a value of 1 is called a Mark and a value of 0 is called a Space. When a communication line is idle, the line is said to be “Marking”, or transmitting continuous 1 value.

The Start bit always has a value of 0 (a Space). The Stop Bit always has a value of 1 (a Mark). This means that there will always be a Mark (1) to Space (0) transition on the line at the start of every word, even when multiple word are transmitted back to back. This guarantees that sender and receiver can resynchronize their clocks regardless of the content of the data bits that are being transmitted.

The idle time between Stop and Start bits does not have to be an exact multiple (including zero) of the bit rate of the communication link, but most UARTs are designed this way for simplicity.

In RS232-C, the "Marking" signal (a 1) is represented by a voltage between -2 VDC and -12 VDC, and”Spacing" signal (a 0) is represented by a voltage between 0 and +12 VDC. The transmitter is supposed to send +12 VDC or -12 VDC, and the receiver is supposed to allow for some voltage loss in long cables. Some transmitters in low power devices (like portable computers) sometimes use only +5 VDC and -5 VDC, but these values are still acceptable to a RS232-C receiver, provided that the cable lengths are short.

2.4.2) RS232-C Break Signal

RS232-C also specifies a signal called a Break, which is caused by sending continuous Spacing values (no Start or Stop bits). When there is no electricity present on the data circuit, the line is considered to be sending Break.

The Break signal must be of a duration longer than the time it takes to send a complete byte plus Start, Stop and Parity bits. Most UARTs can distinguish between a Framing Error and a Break, but if the UART cannot do this, the Framing Error detection can be used to identify Breaks.

In the days of tele printers, when numerous printers around the country were wired in series (such as news services), any unit could cause a Break by temporarily opening the entire circuit so that no current flowed. This was used to allow a location with urgent news to interrupt some other location that was currently sending information.

In modern systems there are two types of Break signals. If the Break is longer than 1.6 seconds, it is considered a "Modem Break", and some modems can be programmed to terminate the conversation and go on-hook or enter the modems' command mode when the modem detects this signal. If the Break is smaller than 1.6 seconds, it signifies a Data Break and it is up to the remote computer to respond to this signal. Sometimes this form of Break is used as an Attention or Interrupt signal and sometimes is accepted as a substitute for the ASCII CONTROL-C character.

Marks and Spaces are also equivalent to “Holes” and “No Holes” in paper tape systems.

Note: Breaks cannot be generated from paper tape or from any other byte value, since bytes are always sent with Start and Stop bit. The UART is usually capable of generating the continuous Spacing signal in response to a special command from the host processor.

2.4.3) RS232-C DTE and DCE Devices

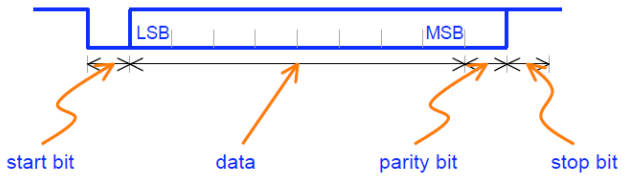
The RS232-C specification defines two types of equipment: the Data Terminal Equipment (DTE) and the Data Carrier Equipment (DCE). Usually, the DTE device is the terminal (or computer), and the DCE is a modem. Across the phone line at the other end of a conversation, the receiving modem is also a DCE device and the computer that is connected to that modem is a DTE device. The DCE device receives signals on the pins that the DTE device transmits on, and vice versa.

When two devices that are both DTE or both DCE must be connected together without a modem or a similar media translator between them, a NULL modem must be used. The NULL modem electrically re-arranges the cabling so that the transmitter output is connected to the receiver input on the other device, and vice versa.

Similar translations are performed on all of the control signals so that each device will see what it thinks are DCE (or DTE) signals from the other device. The number of signals generated by the DTE and DCE devices is not symmetrical. The DTE device generates fewer signals for the DCE device than the DTE device receives from the DCE.

III. SERIAL TRANSMISSION:

- Data transmission is made by the UART in a serial way, by 11-bit blocks:
  - A 0 bit marks the starting point of the block
  - Eight bits for data
  - One parity bit
  - A 1 bit marking the end of the block
- The transmission and reception lines should hold a 1 when no data is transmitted
  - The parity bit is set to 1 or 0, depending on the number of 1's.
  - The transmission speed is fixed, measured in bauds.



UART transmission

IV. DESIGN:

This design can also be instantiated many times to get multiple UARTs in the same device. For easy instantiation of the design into a larger implementation, the bi-directional data bus is separated into two buses, DIN and DOUT, instead of using tri-state buffers. The transmitter and receiver both share a common internal Clk16X clock. This internal clock which needs to be 16 times of the desired baud rate clock frequency is obtained from the on-board clock through the MCLK input directly.

USART chips have both synchronous and asynchronous modes. In *synchronous* transmission, the clock data is recovered separately from the data stream and no start/stop bits are used. An asynchronous transmission sends no characters over the interconnection when the transmitting device has nothing to send; but a synchronous interface must send "pad" characters to maintain synchronization between the receiver and transmitter. The usual filler is the ASCII "SYN" character. This may be done automatically by the transmitting device.

V.FEATURES

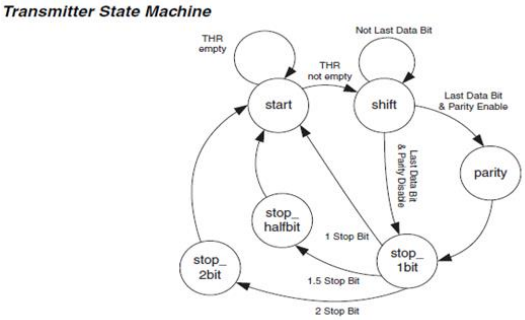
- Functionally compatible with the NS16450 UART
- Raster performance than industry standard hardwired devices
- Inserts or extracts standard asynchronous communication bits (start, stop and parity) to or from the serial data
- Holding and shifting registers eliminate the need for precise synchronization between the CPU and serial data
- Standard CPU Interface
- Separate interrupt lines for data received (Rx Rdyn) and data transmitted (Tx Rdyn)
- A common interrupt line for all internal UART data and error events. Interrupt conditions include: receiver line errors, receiver buffer available, transmit buffer empty and when a modem status flag change is detected.
- Fully-prioritized interrupt system control
- MODEM interface functions (CTS, RTS, DSR, DTR, RI and DCD)
- Fully programmable serial interface characteristics:
  – 5, 6, 7 or 8-bit characters
  – Even, odd, or no-parity bit generation and detection
  – 1, 1.5 or 2-stop bit generation and detection
- False start bit detection
- Line breaks generation and detection
- Interactive control signaling and status reporting capabilities
- Separate input and output data buses for use as an embedded module in a larger design
- Transmitter enabled by new data writes to transmit holding register
- Receiver synchronizes off the start bit
- Receiver samples all incoming bits at the center of each bit

VI. TRANSMITTER:

The serial transmitter section consists of an 8-bit Transmitter Hold Register (THR) and Transmitter Shift Register (TSR). There are two ways to indicate the status of THR: an independent TxRDY output pin or the THRE flag in the Line Status Register (LSR).. After the data is loaded in THR, the THR empty flag in LSR will be reset to logic 0, and pin TxRDYn will go inactive high.

The serial data transmission will be automatically enabled after the data is loaded into THR. First a start bit (logic 0) is transmitted and the data in THR is parallel loaded to TSR automatically. This automatic sequencing cause the frames to be transmitted back-to-back which Increases the transmission bandwidth. When no transmission is taking place, the SOUT pin is held in the high state.

The behavior of the transmitter is controlled by the FSM (Finite State Machine) shown in Figure



<Start>: When the UART is reset by the MR pin, the transmitter FSM will be reset to this state. When in this state, the transmitter is waiting to assert the start bit. A start bit will be asserted as soon as the THR is not empty. Once a low SOUT (start bit) is asserted, the FSM will switch to <shift> state.

<Parity>: When the FSM is in this state, the last data bit is still in transmission. When the transmission is complete, the FSM will assert the parity bit. Once the parity bit is asserted, the FSM switch to the <stop_1bit> state.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

<Shift>: When the FSM is in this state, it is waiting for the last (most significant) data bit to be shifted out. After the last data bit is shifted out, the FSM will switch to <parity> state if parity is enabled. Otherwise, it will switch to <stop_1bit> state.

<stop_1bit>: No matter if the stop bit is configured to be 1, 1.5 or 2 bits long, the FSM will always switch to this state, wait for a baud clock cycle, and then assert the stop bit(s). For one stop bit, the FSM switches back to the <start> state and waits to assert the start bit of another frame. For 1.5 stop bits, it switches to <stop_ half bit> state and stays there for just half a baud clock cycle before

Switching to <start> state. For two stop bits, it switches to <stop_2bit> state then switches back to <start> state. Note that the stop bit(s) is asserted at the time when the FSM is leaving the <stop_1_bit> state.

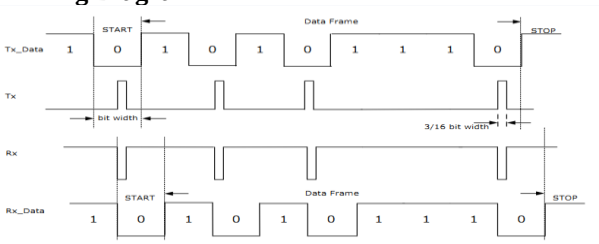
<stop_2bit>: When the FSM is in this state, the first stop bit is in transmission. It waits for a baud clock cycle, then asserts the second stop bit and switches to the <start> state.

<Stop _ half bit>: This state is for 5-bit data bits with a 1.5 stop bit. The FSM will stay in this state for only half baud clock cycle and then switch to <start> state.

VII. CONCLUSIONS

The parallel side of a UART's usually connected to the bus of a computer. When the computer writes a byte to the UART's Transmit Data Register (TDR), the UARTs will start to transmit it on the serial line. The UART's status register contains a flag bit which the computer can read to see if the UART is ready to transmit another byte.

Timing Diagram



REFERENCES

1. http://www.spel.com/Technology.html
2. http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter
3. http://www.latticesemi.com/
4. http://www.freebsd.org/doc/en/articles/serial-uart/index.html

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)