

SOFTWARE UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER

Ankur Khetrpal
Computer Engineering
Delhi College of Engineering
Delhi University

ankurkhetrapal@ieee.org

Software Universal Asynchronous Receiver Transmitter

Introduction

UART or Universal Asynchronous Receiver Transmitter is used for serial data communication; it converts bytes of data to and from asynchronous start-stop bit streams represented as binary electrical impulses. UARTs are commonly used in conjunction with other communication standards such as RS-232.

The word "asynchronous" indicates that UARTs recover character timing information from the data stream, using designated "start" and "stop" bits to indicate the framing of each character. In *synchronous* transmission, the clock data is recovered separately from the data stream and no start/stop bits are used.

Operation

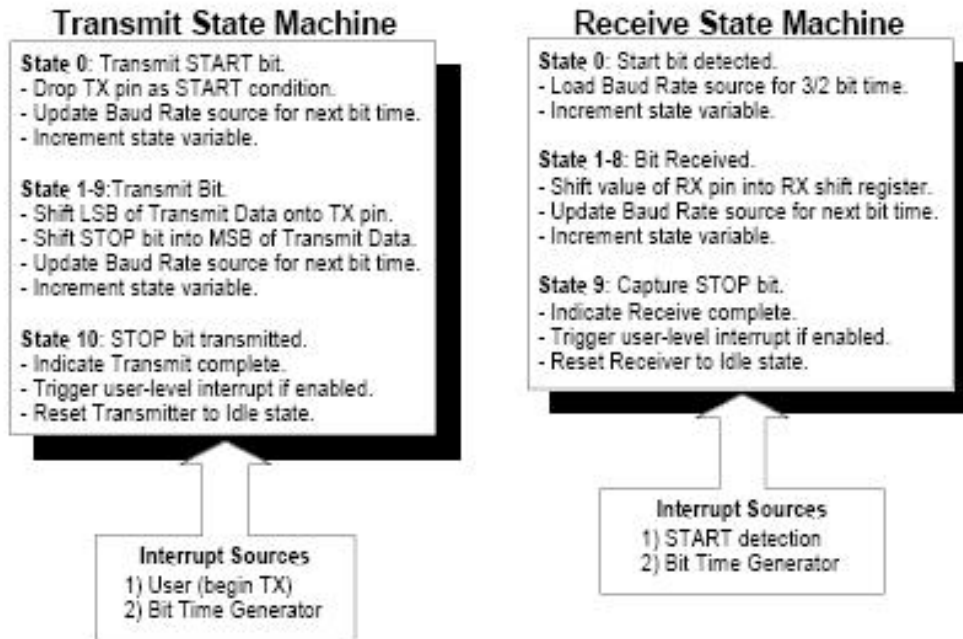
In idle state, the line is held at logic '1'. When data is to be transmitted, the start bit, which is a logic '0' is transmitted for a period of 1 bit time. This causes a '1' to '0' transition at the line. This can be detected by the receiver and is used as a notification that data is coming.

The data bits are then transferred serially, starting with the Least Significant Bit. The logic value corresponding to the data bit being transmitted is held at the line for 1 bit time.

When data bits have been transmitted, a stop bit is transferred. A stop bit is a logical '1' and puts the line back to idle state before a new start bit followed by the data can be transmitted.

The receiver must sample the data during every bit time in order to receive the data properly. The sampling frequency is generally taken to be a multiple of the baud rate so that all the bits are samples correctly.

When the above functions are managed by software, the resulting module is known as software UART. The essential trade-off to consider when implementing a software UART is between hardware usage and speed/efficiency. Designs that utilize more hardware are likely to consume less CPU bandwidth and allow higher bit rates.



Key Features

1. An interface similar to hardware UART with transmit and receive interrupts.
2. Interrupt driven implementation.
3. Full duplex communication with data rates up to 19.2kbps when operated at 12Mhz
4. Minimal hardware usage.

Implementation

The following modules are implemented in the solution:

1. void get_rx_pin_status(void)
Returns 0 or 1 dependent on whether the receive pin is high or low.
2. void cout_high(void)
Sets the transmit pin to the high state.
3. void cout_low(void)
Sets the transmit pin to the low state.

4. void idle(void)

Background functions to execute while waiting for input or output.

5. void timer_set()

Sets the timer to 3 times the baud rate, enables interrupt and reset on every match of the timer counter.

6. void UartTxRx(void)

It is the ISR that manages transmission and receiving of data and error handling.

7. void init_uart(void)

Initializes the uart to idle state and sets appropriate flags.

8. void _putchar(char ch)

Defines the frame to be transmitted and outputs the byte serially on the TX pin and initializes flags to initiate transfer on the next interrupt.

9. char _getchar(void)

Returns the oldest received byte from the input buffer (FIFO structure).

10. void flush_input_buffer(void)

Clears the input buffer.

Error Handling

The implemented software UART checks the input stream for framing errors and overflows. These conditions are explicitly handled by the UART. A framing error occurs in case a logic '0' stop bit is received by the UART. In case of a framing error, the received byte is rejected and is not stored in the input buffer.

References

1. *LPC 2129 User Manual* from Philips
2. *Insider's Guide to ARM* from Hitex
3. *Software UART Examples Application note AN115* from Silicon Laboratories Inc. www.silabs.com
4. *Software UART using bit banging* published in *Embedded Systems Programming*, December 1991.
5. *User Manual for Keil* available at www.keil.com