BitBang UART Embedded

January 10, 2025

1 Bit Banging : Introduction

Bit banging is a technique for serial communication where data transmission (TX) and reception (RX) are managed by software controlling the GPIO pins rather than relying on dedicated hardware peripheral of UART.

2 Bit Banging : Tx

2.1 Explanation

In bit-banging transmission (TX), the microcontroller generates the required signal waveform by manipulating output pins at precise time intervals. This involves toggling the GPIO pin for each bit like the start bit, data bits, optional parity, and a stop bit.

2.2 Code

In the below code snippet following is done:

- Libraries are included
- SysTick Registers are defined from their locations given in datasheet
- SysTick control register bits are defined and UART pins are defined
- Global variables for tx are initialized

```
#include <stdint.h>
#include <stdbool.h>
#include "tm4c123gh6pm.h"
#include "uart.h"
#include "printf.h"
#include "trace.h"

/* SysTick Registers Define */
#define STCTRL *((volatile long *) 0xE000E010)
#define STRELOAD *((volatile long*) 0xE000E014)
#define STCURRENT *((volatile long *) 0xE000E018)

/* SysTick Control Registers Define */
#define ENABLE (1 << 0) // Enable SysTick Counter
#define CLKINT (1 << 2) // System CLK is used by writing 1 here
#define STINT  (1 << 1) // SysTick Interrupt is enabled
#define CLOCK_MHZ 16 // System CLK Speed

/* UART Port A Pins Define */
#define UART_Tx_Pin 0x02
```

```
21
22 /* Global variables used for tx */
23 volatile uint_8 tx_data = 0;
24 volatile uint_8 tx_index = 0;
25 volatile uint_8 tx_parity = 0;
26 volatile bool tx_busy = false;
27 volatile bool partity = false;
28
29 /* Baud Rate for the bit banging communication */
30 #define BAUDRATE 9600
```

UART and SysTick are initialized. SysTick timer calculation are done as follows :
The time required to transmit one bit is given by:

$$T_{\text{bit}} = \frac{1}{\text{Baud Rate}} = \frac{1}{9600} = 0.000\,104\,166\,7\,\text{s}\,(104.17\ \mu\text{s}).$$

For 10 bits (1 start bit, 8 data bits, and 1 stop bit):

$$T_{\text{total, 10}} = 10 \times T_{\text{bit}} = 10 \times 0.000\,104\,166\,7\,\text{s} = 0.001\,041\,667\,\text{s}\,(1.041\ \text{ms}).$$

For 11 bits (1 start bit, 8 data bits, 1 parity bit, and 1 stop bit):

$$T_{\text{total, 11}} = 11 \times T_{\text{bit}} = 11 \times 0.000\,104\,166\,7\,\text{s} = 0.001\,145\,833\,\text{s}\,(1.146\ \text{ms}).$$

The SysTick reload value is calculated using the system clock frequency $f_{\text{sys}} = 16\,\text{MHz}$:

$$\text{SysTick Reload Value} = T_{\text{total}} \times f_{\text{sys}}.$$

For 10 bits:

$$\text{Reload Value (10 bits)} = 0.001\,041\,667\,\text{s} \times 16\,\text{MHz} = 16667.$$

For 11 bits:

$$\text{Reload Value (11 bits)} = 0.001\,145\,833\,\text{s} \times 16\,\text{MHz} = 18333.$$

- SysTick Reload Value for 10 bits: **16667**
- SysTick Reload Value for 11 bits: **18333**

```
1  void UART_TxBitBang_Init(){
2      /* GPIO Port A Configuration */
3      SYSCTL_RCGCGPIO_R |= (1<<0); // Enable clock for Port A
4      GPIO_PORTA_DEN_R |= 0x02; // Digital Enable Port A pins 0
5      GPIO_PORTA_DIR_R |= 0x02 ; // Set Port A pin 1 to output
6      GPIO_PORTA_DATA_R |= 0x02; // Set Porta A pin 1 (output) to idle
           state high
7  }
8
9  void SystTick_Init(){
10     if (parity==false){
11         STRELOAD = 0x411B; // Calculated from baud rate, (1/baud_rate)
               *no_of_bits*clk_freq, (1/9600)*10*16000000
12     } else {
13         STRELOAD = 0x479D; // Calculated from baud rate, (1/baud_rate)
               *no_of_bits*clk_freq, (1/9600)*11*16000000
14     }
15     STCTRL |= (CLKINT | STINT);
16 }
```

Now transmit code and interrupt handler code is addded.

```
void UART_Transmit(uint8_t data){
    while(tx_busy){
        // do nothing
    }
    tx_data = data; // Set tx_data to current data
    tx_index = 0; // Reset index to 0
    tx_busy = true; // Set busy flag to true

    if (parity==false){
        STRELOAD = 0x411B; // Calculated from baud rate, (1/baud_rate)*no_of_bits*clk_freq, (1/9600)*10*16000000
    } else {
        STRELOAD = 0x479D; // Calculated from baud rate, (1/baud_rate)*no_of_bits*clk_freq, (1/9600)*11*16000000
    }
    STCTRL |= ENABLE; // Enable SysTick timer to send data via interrupts at baud rate
}

void SysTickHandler(void){

    /* Transmit the start bit */
    if (tx_index==0){
        GPIO_PORTA_DATA_R &= ~UART_Tx_Pin; // Send UART Start Bit (1)
        tx_index += 1;
    }

    /* Transmit the 8 bits of data in UART */
    while ((tx_index > 0) & (tx_index < 8)){
        /* If tx_data has 1 at tx_index then set GPIO Port A Pin 1
         * If tx_data has 0 at tx_index then clear GPIO Port A Pin 1
         */
        if ((tx_data >> tx_index) & 1){
            GPIO_PORTA_DATA_R |= UART_Tx_Pin;
            tx_parity ^= 0x01;
        } else {
            GPIO_PORTA_DATA_R &= ~UART_Tx_Pin;
        }

        /* Increase tx_index to go to next bit in tx_data */
        tx_index += 1;
    }

    /* Send parity bit if true */
    if ((parity==true) & (tx_index==8)){
        if (tx_parity == 1){
            GPIO_PORTA_DATA_R |= UART_Tx_Pin;
        } else {
            GPIO_PORTA_DATA_R &= ~UART_Tx_Pin;
        }
        tx_index += 1;
    }

    /* Transmit stop bit as the 10th bit(if no parity) or 11th bit(if parity)in UART */
    if (((parity==false) & (tx_index==8)) | ((parity==true) & (
```

```
        tx_index==9))){
52          GPIO_PORTA_DATA_R |= UART_Tx_Pin;
53          tx_index = 0;
54          tx_busy = false
55      }
56
57      /* Disable SysTick*/
58      STCTRL &= ~ENABLE;
59  }
```

Finally here is the main function to transmit 'A', 'B' and 'C' in a loop.

```
1   void main(void){
2
3       UART_RxBitBang_Init(); // Initialize PORT A Pin 0 UART Rx
4       UART_TxBitBang_Init(); // Initialize Port A Pin 1 UART Tx
5       while (1){
6           UART_Transmit('A');
7           UART_Transmit('B');
8           UART_Transmit('C');
9       }
10  }
```

3 Bit Banging : Rx

3.1 Explanation

Bit-banging reception (RX) involves reading the state of an input pin at specific intervals to reconstruct the received data. The software must be synchronized with the incoming signal to correctly interpret the start bit and then sample each subsequent data bit at the appropriate time..

3.2 Code

In the below code snippet following is done:

- Libraries are included
- SysTick Registers are defined from their locations given in datasheet
- SysTick control register bits are defined and UART pins are defined
- Global variables for rx are initialized

```
1   #include <stdint.h>
2   #include <stdbool.h>
3   #include "tm4c123gh6pm.h"
4
5   /* SysTick Registers Define */
6   #define STCTRL *((volatile long *) 0xE000E010)
7   #define STRELOAD *((volatile long*) 0xE000E014)
8   #define STCURRENT *((volatile long *) 0xE000E018)
9
10  /* SysTick Control Registers Define */
11  #define ENABLE (1 << 0)     // Enable SysTick Counter
12  #define CLKINT (1 << 2)     // System CLK is used
13  #define STINT  (1 << 1)     // SysTick Interrupt is enabled
14  #define CLOCK_MHZ 16  // System Clock Speed in MHz
15
```

```c
/* UART Port A Pins Define */
#define UART_Tx_Pin 0x02
#define UART_Rx_Pin 0x01

/* Global variables used for RX */
volatile uint8_t rx_data = 0;
volatile uint8_t rx_index = 0;
volatile uint8_t rx_parity = 0;
volatile bool rx_ready = false;
volatile bool parity = false;

/* Baud Rate for the bit-banging communication */
#define BAUDRATE 9600
```

Now we initialize UART Rx Pin and SysTick timer.

```c
/* Initialize UART RX */
void UART_RxBitBang_Init() {
    SYSCTL_RCGCGPIO_R |= (1 << 0);        // Enable clock for Port A
    GPIO_PORTA_DEN_R |= UART_Rx_Pin;     // Digital Enable Port A Pin 0
    GPIO_PORTA_DIR_R &= ~UART_Rx_Pin;    // Set Port A Pin 0 as input
}

/* Initialize SysTick Timer */
void SystTick_Init() {
    if (parity == false) {
        STRELOAD = 0x411B;  // 10 bits at baud rate 9600 for 16 MHz
    } else {
        STRELOAD = 0x479D;  // 11 bits at baud rate 9600 for 16 MHz
    }
    STCTRL |= (CLKINT | STINT);
}
```

Now receive code and interrupt handler code is added.

```c
/* Start Receiving Data */
void UART_Receive_Start() {
    if (!rx_ready) {
        rx_data = 0;
        rx_index = 0;
        rx_parity = 0;
        rx_ready = false;

        // start bit wait (falling edge)
        while (GPIO_PORTA_DATA_R & UART_Rx_Pin);

        // Systick takes over for bit sampling
        STCTRL |= ENABLE;
    }
}

/* SysTick Handler for RX */
void SysTickHandler(void) {
    static bool start_bit_checked = false;

    /* Check Start Bit */
    if (!start_bit_checked) {
```

```
        if (!(GPIO_PORTA_DATA_R & UART_Rx_Pin)) {  // Confirm start bit is low
            start_bit_checked = true;
        } else {
            start_bit_checked = false;
            rx_ready = false;  // False start, reset
            STCTRL &= ~ENABLE; // Disable SysTick
            return;
        }
    }

    /* Sample Data Bits */
    if (rx_index < 8) {
        if (GPIO_PORTA_DATA_R & UART_Rx_Pin) {
            rx_data |= (1 << rx_index);  // Set the bit
            rx_parity ^= 0x01;           // Update parity
        }
        rx_index++;
    } else if (parity && rx_index == 8) {
        /* Sample Parity Bit */
        uint8_t received_parity = (GPIO_PORTA_DATA_R & UART_Rx_Pin) ? 1 : 0;
        if (received_parity != rx_parity) {
            rx_ready = false;  // Parity error
        }
        rx_index++;
    } else if (rx_index == (parity ? 9 : 8)) {
        /* Check Stop Bit */
        if (GPIO_PORTA_DATA_R & UART_Rx_Pin) {
            rx_ready = true;  // Data successfully received
        } else {
            rx_ready = false; // Stop bit error
        }
        rx_index++;
    }

    /* Stop SysTick after all bits received */
    if (rx_index >= (parity ? 10 : 9)) {
        STCTRL &= ~ENABLE; // Disable SysTick
        start_bit_checked = false;
    }
}
```

Finally here is the main function to receive any 8 bit data from PC and echo it back if UART Trasmit function as defined before exists.

```
void main(void) {
    UART_RxBitBang_Init();
    SystTick_Init();

    while (1) {
        UART_Receive_Start(); // Start data receive

        if (rx_ready) {
            // Echo back data if UART_Transmit defined
            UART_Transmit(rx_data);
            rx_ready = false;
```

```
12              }
13          }
14  }
```

References