21.05.24

|  |
| --- |
| Line by line Serial data communication |
|  |  |  |
|  |  | Go all at same time = Parallel data communication |
|  |  | They are sent (transfer) |
|  |  |  |
|  |  | They are 8- bit data transferred. |
|  |  |  |
|  |  | MS B = Most significant bit |
|  |  |  |
|  |  | LS B = Least significant bit |
|  |  |  |
|  |  | They two method in serial data communication. |
|  |  | eg: 0110011010011010 |
|  |  |  |
|  |  |  |
|  |  | 16 10 0 |
|  |  | MSB LSB |
|  |  |  |
|  |  | Synchronous |
|  |  |  |
|  |  | They data & clock are Some at a time. |
|  |  |  |
|  |  | Asynchronous |
|  |  |  |
|  |  | Not at same time. |
|  |  |  |
|  |  | Baudrate = is like a bridge to Transmitter and receiver to Perform in time (or type of command) Transmit speed (data per sec) Space (bits Per sec). |
|  |  |  |
|  |  | Bit rate - Per second (frequency of clock) time. |
|  |  |  |
|  |  |  |
|  |  | UART = serial communication. |
|  |  |  |
|  |  | UASRT = Univered Asynchronous. |
|  |  |  |
|  |  | • Synchronous Reciever Transmitter. |
|  |  |  |
|  |  | Types |
|  |  |  |
|  |  | Duplex - Data can be transmitted & received and do in same time |
|  |  |  |
|  |  | Simplex =Data flows in one direction only. In transmit & receive process. |
|  |  | One can transmit , one can receive. |
|  |  |  |
|  |  | Half Duplex = Devices can both send and receive, but not at the same time. Cannot do both process |
|  |  | in same way or in one way. (Do both directions) |
|  |  | Eg: walkie-talkie |
|  |  |  |
|  |  | Full Duplex = both devices can send and receive data at same time. In both direction. |
|  |  | Eg: phone ,etc.. |
|  |  | Baudrate have two parity |
|  |  |  |
|  |  | \* Old Parity, |
|  |  |  |
|  |  | \*Even Parity. |
|  |  |  |
|  |  | We use Hardware for now ( we have circuit is uart -2 ) |
|  |  |  |
|  |  | by using UART-2 ( it controls i know my thoughts) |
|  |  |  |
|  |  | \* blind led, |
|  |  |  |
|  |  | \* flash. |
|  |  |  |
|  |  | we have this circuit (SEM32F446RE) |
|  |  |  |
|  |  | Both the function do some |
|  |  | In this 16 MHz (circuit) |
|  |  |  |
|  |  | UART = Full Duplex |
|  |  |  |
|  |  | 1 bit at start - value of start 0 (always) |
|  |  |  |
|  |  | Stop 1 or 2 bit - value of stop is 1. |
|  |  |  |
|  |  | 0 = Full (Data have), |
|  |  |  |
|  |  | 1 = Empty (not have data). |
|  |  |  |
|  |  | When it changes or we change it , we can use for data process , |
|  |  | check by the status register ( what is process or status of that work (maybe anything in circuit) |

Codeing

#include "stm32f4xx.h"

#include <stdint.h>

#include <string.h>

#define GPIOAEN (1U<<0)

#define PA2MOD (2U<<4)

#define PA2AF (7U<<8)

#define APB1EN (1U<<17)

#define USART2\_TX (1U<<3)

#define USART2EN (1U<<13)

#define SR\_TXE (1U<<7)

#define SYS\_FREQ 16000000

#define APB1\_CLK SYS\_FREQ

#define UART\_BAUDRATE 115200

void uart2\_init(void);

void uart2\_write(const char \*str);

static void uart\_set\_baudrate(USART\_TypeDef \*USARTx, uint32\_t PeriphClk, uint32\_t BaudRate);

static uint16\_t compute\_uart\_bd(uint32\_t PeriphClk, uint32\_t BaudRate);

int main()

{

uart2\_init();

while(1)

{

const char \*mystr ="boobathi \r \n";

uart2\_write(mystr);

}

return 0;

}

void uart2\_init(void)

{

/\*\*\*\*\*Configure uart gpio pin\*\*\*\*\*\*/

/\*Enable clock access to gpioa \*/

RCC->AHB1ENR |= GPIOAEN;

/Set PA2 mode to alternate function mode/

GPIOA->MODER |= PA2MOD;

/Set PA2 alternate function type to UART\_TX (AF07)/

GPIOA->AFR[0] |= PA2AF;

/\*\*\*\*\*Configure uart module \*\*\*\*\*\*/

/\*Enable clock access to uart2 \*/

RCC->APB1ENR |= APB1EN;

/Configure baudrate/

uart\_set\_baudrate(USART2,APB1\_CLK,UART\_BAUDRATE);

//USART2->BRR = ((APB1\_CLK + (BaudRate/2U))/BaudRate) ;

/Configure the transfer direction/

USART2->CR1 = USART2\_TX;

/Enable uart module/

USART2->CR1 |= USART2EN;

}

static void uart\_set\_baudrate(USART\_TypeDef \*USARTx, uint32\_t PeriphClk, uint32\_t BaudRate)

{

USARTx->BRR = compute\_uart\_bd(PeriphClk,BaudRate);

}

static uint16\_t compute\_uart\_bd(uint32\_t PeriphClk, uint32\_t BaudRate)

{

return ((PeriphClk + (BaudRate/2U))/BaudRate);

}

void uart2\_write(const char \*str)

{

for (int i=0;i< strlen(str);i++){

USART2->DR = (str[i] & 0xFF);

while(!(USART2->SR & SR\_TXE)){

}

}

/Make sure the transmit data register is empty/

/Write to transmit data register/

}