LAB 7

SERIAL COMMUNICATION

1. INTRODUCTION

This lab gives information about serial communication of CSM12C32. What is the UART? Explain RS-232 serial communication standards. What is the Baud Rate? Search all of these on Internet. Read following documents for CSM12C32 serial communication basics.

HCS12 UART.pdf (Find Transmit and Receiver flowcharts)

MC9S12C Datasheet.pdf p389-413 (Read SCI registers features)

2. RECEIVE and TRANSMIT MODE

CSM12C32 has internal UART hardware. It sends and receives eight or nine bit serial data. Before sending or receiving any data, you should initialize some registers. These are SCIBDL, SCIBDH, SCICR1 and SCICR2. Read the pages 389-413 of MC9S12C Datasheet.pdf on Ninova. Following code is simple UART initialization code. Try to understand register values.

```
void sci_init(void){
 SCIBDL = 0x34; /*Configure baud rate at 9600 bps with Baud=16/(16*0x34) */
 SCIBDH = 0x00; /*an SCI clock modulo of 8MHz*/
 SCICR1 = 0x00; /*8 data bits, no parity*/
 SCICR2 = 0x0C; /*Enable Tx, Rx Bits*/
}
```

a. Transmit Mode

In order to transmit any eight bit data, you should write the data to SCIDRL register. Then processor sends data bit by bit. If you are sending sequential data, before sending each eight bit you should check *Transmission Complete Flag* (sixth bit of SCISR1). If this flag is set, no transmission progress. It means you can write your data to SCIDRL. Otherwise, you need to wait transmission complete.

b. Receive Mode

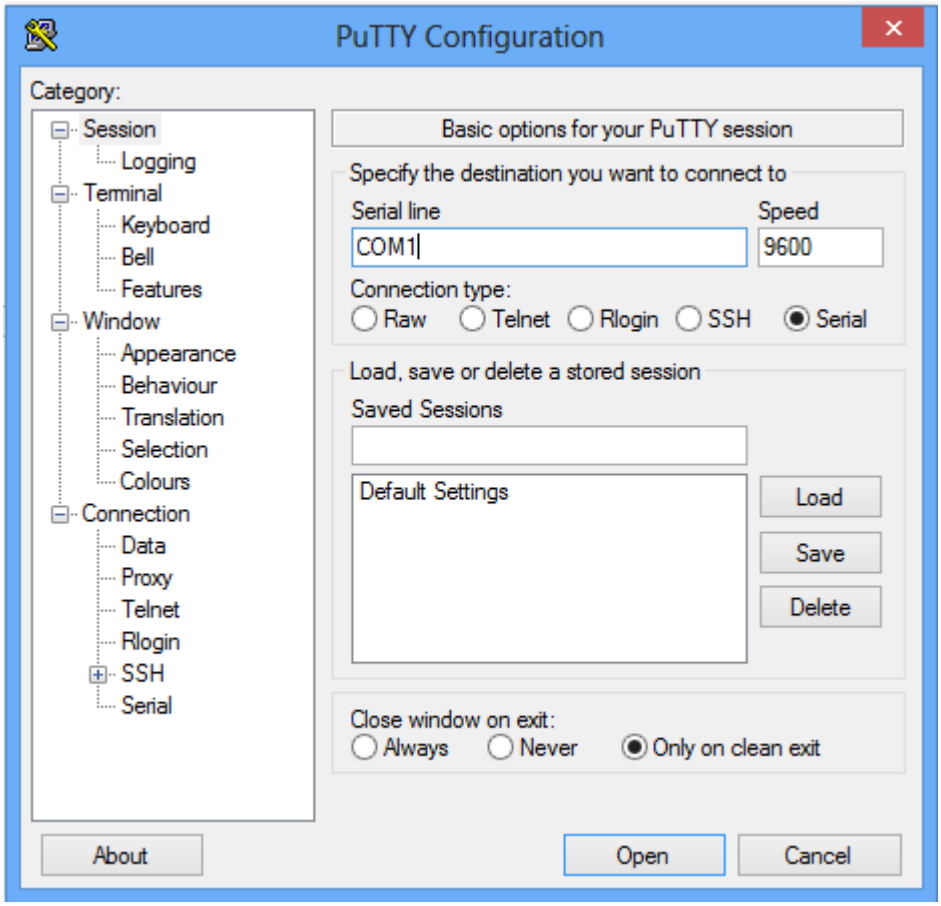
Receiving is a little bit different from transmitting procedure. If any eight bit data has received. *Receive Data Register Full Flag* (fifth bit of SCISR1) is set. Then, you can read SCIDRL register. It has the received data. If *Receive Data Register Full Flag* is reset, data is not received.

3. USAGE of SERIAL TERMINAL (PUTTY)

CSM12C32 has one UART port. It is used for both programming and serial terminal. To use as a serial terminal, follow the below instructions.

1- Load your program to microcontroller.
2- Run the program using green arrow.
3- Close current window which is True-Time Simulator & Real Time Debugger
4- Run Putty on desktop
5- Select Serial
6- Set baud rate 9600 and select correct com port (it is usually COM1)
7- Click Open

You should close the Putty to program microcontroller again.



4. EXPERIMENT

a. Transmit Only

Complete the following C code. This code prints HELLO WORLD! on serial terminal Putty, if SW1 is pressed.

```
#include <hidef.h>      /* common defines and macros */
#include <mc9s12c32.h>     /* derivative information */
#pragma LINK_INFO DERIVATIVE "mc9s12c32"

unsigned char text[]="HELLO WORLD!";

void sci_init(void){

 SCIBDL = 0x34; /*Configure baud rate 9600 bps*/
 SCIBDH = 0x00; /*an SCI clock modulo of 8MHz*/

 SCICR1 = 0x00; /*8 data bits, no parity*/
 SCICR2 = 0x0C; /*Enable Tx, Rx Bits*/
}

void send_char(unsigned char value){
////////////////////////////////
///Your send_char function/////
////////////////////////////////
}

void print_string(unsigned char * ptr){
////////////////////////////////////
///Your print_string function/////
////////////////////////////////////
}

void main(void) {
  DDRE &= 0xFE; //PortE.0 is input for SW1
  sci_init();

  for (;;){
      if( !(PORTE & 0x01) )  //when SW1 is pressed
        print_string(text);
  }

}
```

b. Echo Mode

Complete the following C code. This code receives typed string on serial terminal Putty. If you press the *enter*, your program will print string which is typed until the *enter*.

```
#include <hidef.h>      /* common defines and macros */
#include <mc9s12c32.h>     /* derivative information */
#pragma LINK_INFO DERIVATIVE "mc9s12c32"

unsigned char text[20]; //Char Buffer
static int count=0;
unsigned char temp;

void sci_init(void){
 SCIBDL = 0x34; /*Configure baud rate at 9600 bps */
 SCIBDH = 0x00; /*an SCI clock modulo of 8MHz*/

 SCICR1 = 0x00; /*8 data bits, no parity*/
 SCICR2 = 0x0C; /*Receiver Full Interrupt, Enable Tx, Rx Bits*/
}

void send_char(unsigned char value){
////////////////////////////////
///Your send_char function/////
////////////////////////////////
}

void print_string(unsigned char * ptr){
////////////////////////////////////
///Your print_string function/////
////////////////////////////////////
}

void main(void) {

  sci_init();

  for (;;){
      ////////////////////////////////////////
      ///Your receive character algorithm/////
      ////////////////////////////////////////
  }

}
```

In the report, explain UART mechanism and each step of the codes.