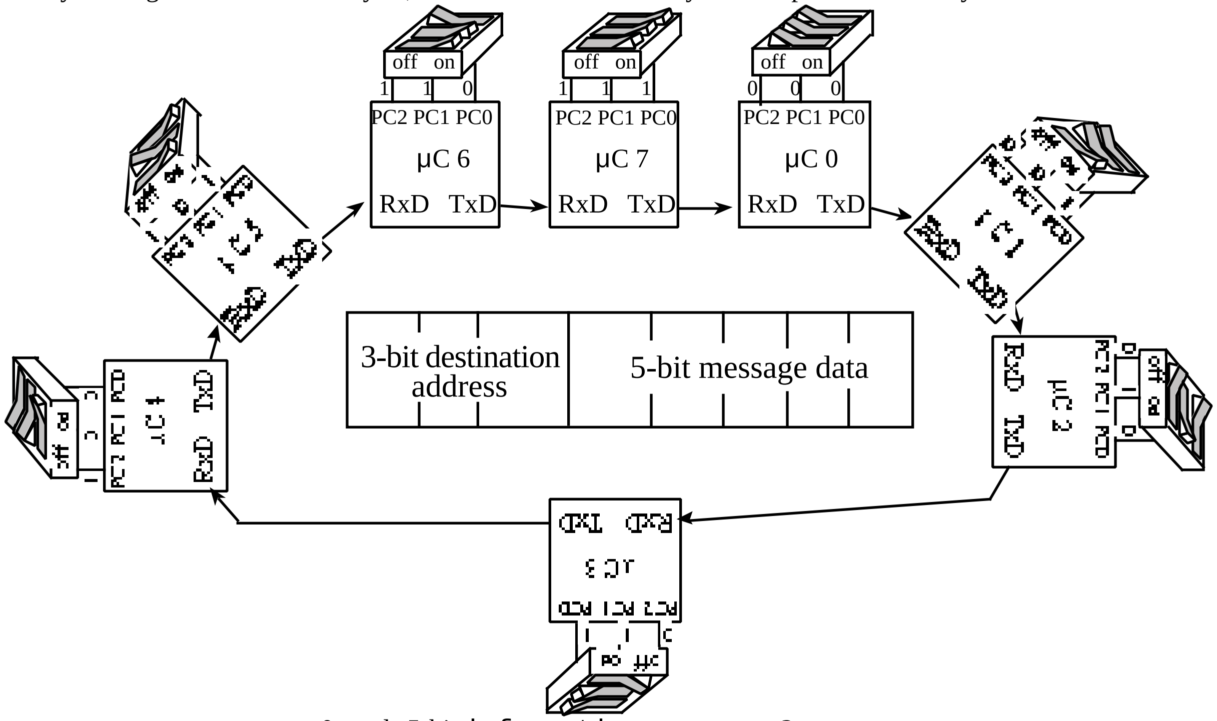
Jonathan W. Valvano First:_________________ Last:_____________________
April 5, 2001, 11:00am-11:50am

This is an open book, open notes exam. You must put your answers on these pages only, you can use the back. You have 50 minutes, so please allocate your time accordingly. ***Please read the entire quiz before starting.***

(60) Question 1. The objective of this problem is to design a communication network using eight single chip microcomputers. The microcomputers are connected together by simple digital lines. The channel is virtually noise-free. The asynchronous simplex serial communication channels link the four computers in a circle. Each 8-bit message consists of a 3-bit destination address, and 5 bits of data. The message data structure has two fields. In memory messages are stored as 2 bytes, but when transmitted they are compressed into 1 byte:



Seven steps occur as computer 0 sends 5-bit `information` to computer 2:

the main program on computer 0 creates the data and passes it to its interrupt software. E.g.,

```
void main(void){ char information; // 5 bits
  RingInit();  // initialize SCI, data structures
  while(1){
    // create the 5-bit information
    RingSend(2,information); // send to computer 2
  }
}
```

the interrupt software on computer 0 transmits the message to computer 1
the interrupt software on computer 1 receives the message
    and notices the destination address does not match its computer number (DIP switch)
the interrupt software on computer 1 retransmits the message to computer 2
the interrupt software on computer 2 receives the message
    and notices the destination address does match its computer number (DIP switch)
the interrupt software on computer 2 passes the message to the main program on computer 2
the main program on computer 2 accepts the message E.g.,

```
void main(void){ char information; // 5 bits
  RingInit();  // initialize
  while(1){
    information=RingAccept(); // waits for message
    // process the 5-bit information
  }
}
```

Part a) The TxD SCI0 output of one computer is directly connected to the RxD SCI0 input of the next computer in the ring. Do the grounds need to be connected together?

Part b) Each microcomputer has a 3-bit DIP switch specifying its computer number (0-7). You may assume that each computer has a unique 3-bit address, and that all eight computers exist. The switch being on (0Ω, 0V) signifies a zero address bit, and the switch being off (open, +5V from pull-up) signifies a one address bit. Show the hardware interface to Port C bits 2,1,0 of the 3 DIP switches for one single chip microcomputer system. Except for the setting of the DIP switch, the network hardware/software is the same for all four computers. Please label chip numbers but not pin numbers. The switches will bounce when toggled, but is hardware debouncing necessary?

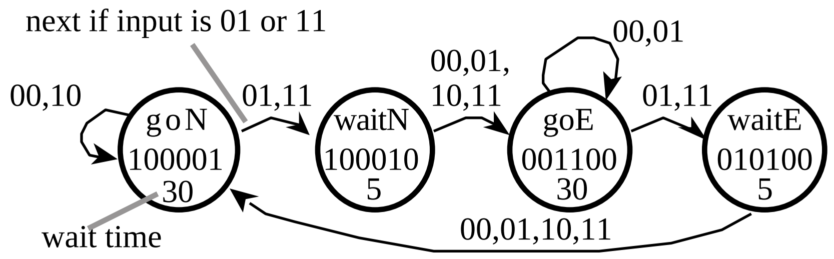
Part c) Give the `RingInit` `RingAccept` `RingSend` functions along with the interrupt handlers required. Since this is similar to SCI12A, you can simply edit the programs of SCI12A into your solution. You may use the `RxFifo` and `TxFifo` FIFO queues without showing the FIFO function definitions. The same network communication software will be placed in each system (you can read the computer address from the DIP switches connected to an input port). The `RingInit` will initialize all data structures, and configure the microcomputer. ***Maximize bandwidth.*** The function `RingSend` will initiate a send message communication. `RingSend` will wait if the message can not be sent (e.g., fifo full). The subroutine `RingAccept` will check to see if a message has been received for this computer. The return value is the 5-bit information field. If no incoming message is available, `RingAccept` will wait. The interrupt processes will perform the actual serial I/O operations. ***Good interrupt software contains NO gadfly loops.*** In this simple system you may assume the destination node exists and is operational. Comments will be graded.

```
#include "RxFifo.h"
#include "TxFifo.h"
#define TDRE 0x80
#define RDRF 0x20
void InitSCI(void){

  RxInitFifo();

  TxInitFifo();

  SC0BDH=0;

  SC0BDL=13;

  SC0CR1=0;

  SC0CR2=0x2C;

asm(" cli");

}
```

```
char InChar(void){ char letter;

  while (RxGetFifo(&letter) == 0){};

  return(letter);

}


void OutChar(char data){

  while (TxPutFifo(data) == 0){};

  SC0CR2=0xAC;

}




#pragma interrupt_handler SciHandler
void SciHandler(void){ char data;

  if(SC0SR1 & RDRF)

    RxPutFifo(SC0DRL);

  if(SC0SR1 & TDRE){

    if(TxGetFifo(&data))

      SC0DRL=data;

    else

      SC0CR2=0x2c;

  }

}

extern void SciHandler();
#pragma abs_address:0xffd6
void (*SCI_interrupt_vector[])() = { SciHandler };
#pragma end_abs_address
#include "RxFifo.c"
#include "TxFifo.c"
```

Part d) What is the bandwidth when sending a continuous data stream from computer 0 to computer 2? For this answer you may assume the other 6 computers are idle. Specify units and show the equations that justify your answer.

(40) Question 2. This Moore FSM controls traffic at an intersection. The 2-bit inputs and the 6-bit outputs are given in binary. The wait times are given in seconds. The controller sequence is output, wait, input, and jump to next state, depending on the input. You are allowed to make minor modifications to the linked structure, as long as the original FSM is implemented. Port C bits 1,0 are sensor inputs, and Port B bits 5-0 are traffic light outputs.



```
const struct State {
  unsigned char Out;             /* Output to Port B */
  unsigned short Time;           /* Time in sec to wait in seconds */
  const struct State *Next[4];}; /* Next if input=00,01,10,11*/
typedef const struct State StateType;
#define goN   &fsm[0]
#define waitN &fsm[1]
#define goE   &fsm[2]
#define waitE &fsm[3]
StateType fsm[4]={
  {0x21, 30,{goN,waitN,goN,waitN}}, /* goN   state */
  {0x22,  5,{goE,goE,goE,goE}},     /* waitN state */
  {0x0C, 30,{goE,goE,waitE,waitE}}, /* goE   state */
  {0x14,  5,{goN,goN,goN,goN}}};    /* waitE state */
```

The FSM controller must run in the background using output compare, RTI or TOF interrupts.

Part a) Show the ritual that initializes the traffic light system. The initial state is `goN`. The main program, which you do not write, will call the ritual, then perform other unrelated tasks.

Part b) Show the ISR that runs the controller in the background. NO BACKWARD JUMPS ALLOWED!