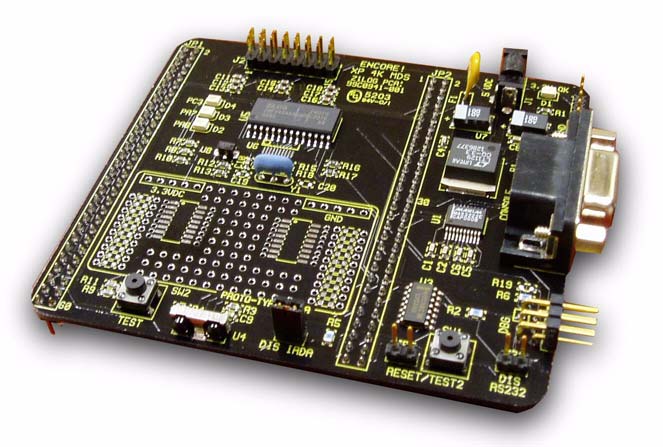
Z8 UART communication

Glenn Lopez  
EMBEDDED SYSTEMS  
Lab 4: Student Report

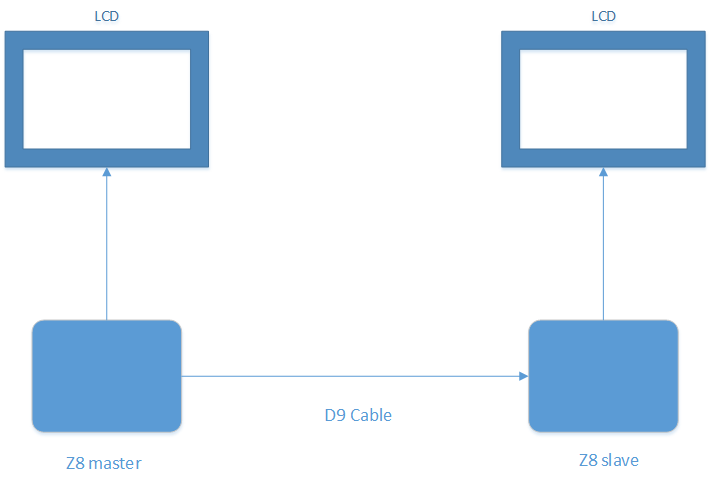


# INTRODUCTION

This lab actively introduces teamwork and **UART concepts** to students by allowing one group of 2 to communicate with each other via serial by sending/receiving packets using two separate Z8 uC. In this lab the master sends a 6 byte packet to the slave. The first byte is the ascii character for “start of text”, the next one is the address of the slave microprocessor. These are then followed by the command and data bytes.

We make sure to end the transmission with ascii “end of text”, but before “end of text” we must send a block check character. **The block check character (BCC)** is calculated by adding up the hex values of all the characters in the packet, inverting it then adding 1 (two’s compliment). This is done in order check that the packets sent by the master are the same as the characters received by the slave. If the sum of the hex values of our packet is equal to zero (0x00), we will assume the correct data is received. The information is sent serially at a baud rate of 38600bps.

# Equiptment Used

* **Zilog Z8 (2x):** We need two for master and slave. Can be purchased from Element14 or eBay for a cheap price of 50 dollars (0.225 BTC @ 210 exchange rate)
* **HD44780 (2x):** We need two to display errors and packets. You can get 10 for 5 dollars on eBay or Element14
* **D9 Serial cable (1x):** You may need a gender changer. This is used as a medium for exchanging packets (fansy smansy poetic word for “a coper wire used to transmit signals”)

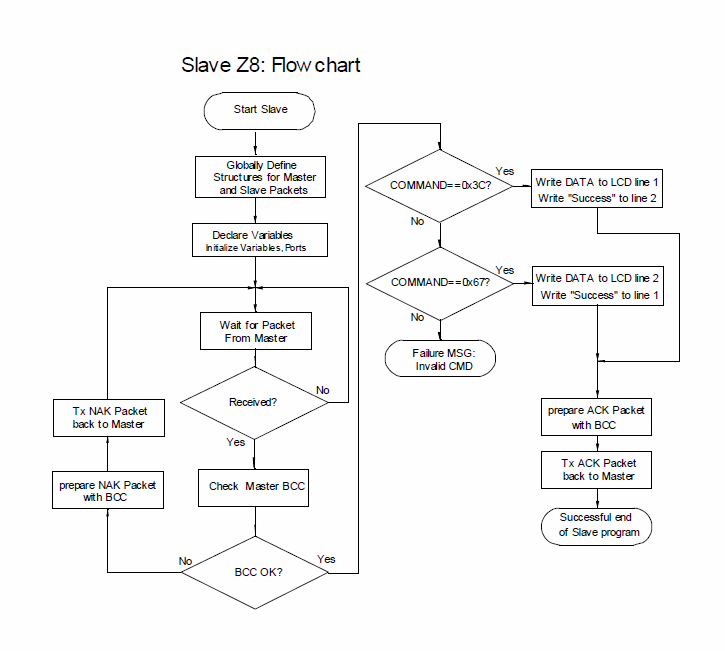
## DISCUSSIONS

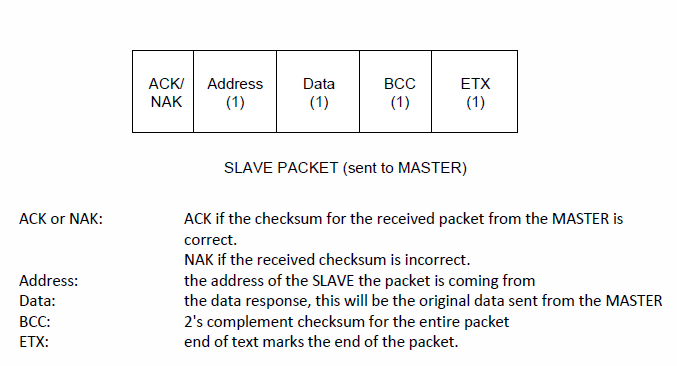
Here’s how the communication between the two Zilogs work…

### The SLAVE

As soon as the slave receives the data from the master, it takes the sum the bytes. If the sum is not equal to zero then something is wrong (there is an error somewhere), so the slave sends a NAK to the master telling the master “hey, I didn’t quite understand what you just said, can you repeat it?”. Now, if the sum of the bytes in the packet is equal to zero, then the slave interprets the payload and writes the data byte to the line directed by

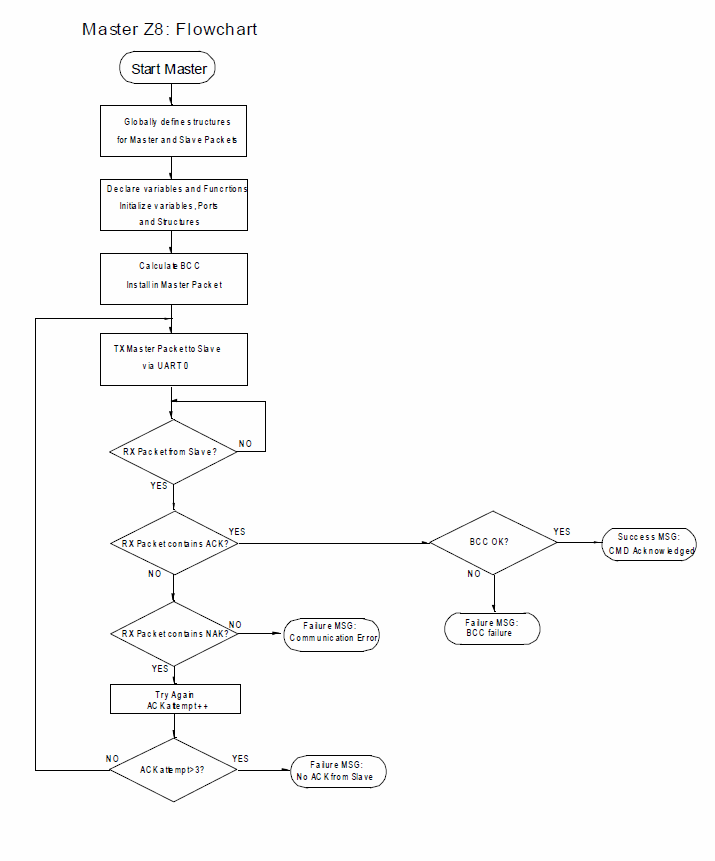
the command byte. The slave tells the master “I hear you loud and clear” by sending an ACK packet.

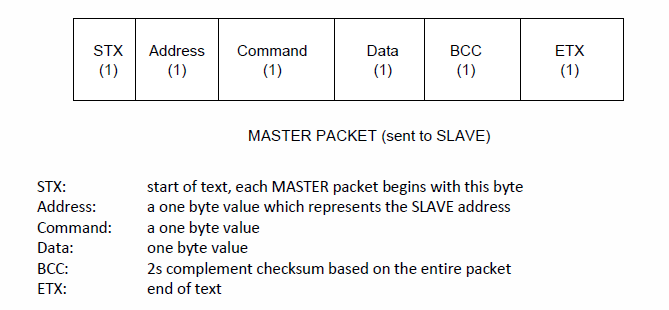




### THE MASTER

If the master receives a packet from the slave’ it will check the first byte (the ACK/NACK byte). When it contains ACK, the master will do a check to see if the sum of the packet is equal to zero. As soon as this happens, the master will write a “CMD acknowledged” to the masters LCD. If the sum of the packets is not equal to zero, the master LCD will display “BCC failure”.





### THE END

If the slave’s packet contains neither a ACK nor a NAK as the first byte, the master sends “Communication error” to it’s LCD. If the slave’s packet contains a NAK, the master attempts to resend the data. If the master attempts this 3 times and receives a NAK each time, it sends “No ACK from Slave” to it’s LCD.

## CONCLUSSION

This lab (a little outdated) provides a great bridge for applying what we learned in Digital communication to microcontrollers. Although I favor Arm9 over Z8, the entire labs (lab1 to lab4) definitely shows that once you know how to program in C, you can take that knowledge and apply it to any microcontroller. It helped me learn the importance of reading datasheets. This lab creates an opportunity for students to experiment with the concepts they learned over the summer and although the program worked as required, I think the use of pointers would of improved how the program worked.

CODE for SLAVE

#include <ez8.h>  
#include <stdio.h>  
#include <string.h>  
#include <uart.h>  
#define BUSY 0x80  
#define READY 0  
#define CLEAR\_LCD 0x01  
#define FIRSTLINE 0x80  
#define SECONDLINE 0xC0  
#define STX 0x02       //defining stx as having value 0x02  
#define M\_ADDR 0x11       //"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" 0x11  
#define S\_ADDR 0x55        //"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" 0x55  
#define ACK 0x06       //"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" 0x06  
#define NAK 0x15       //"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" 0x15  
#define ETX 0x03       //"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" 0x03  
  
typedef struct{        //Define the structure of mpacket  
    unsigned char stx;     //start of text  
    unsigned char address;    //address of master  
    unsigned char command;    //command to be executed  
    unsigned char data;     //data of the packet  
    unsigned char bcc;     //bcc sent by master  
    unsigned char etx;     //end of text  
 }mpacket;  
   
typedef struct{          // defining the structure of the slave packet  
    unsigned char ack;     //acknowledge  
    unsigned char address;    //address of the master  
    unsigned char data;     //data received  
    unsigned char bcc;     //calculated bcc  
    unsigned char etx;     //end of text  
 }spacket;  
   
int nm, ns;  
 mpacket m, \*p\_m;        //structure pointer of m packet  
 spacket s, \*p\_s;        //structure pointer for s packet  
   
 long int delay1; //delay 1  
 long int delay2; //delay 2  
   
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
   
//uart function prototypes  
mpacket init\_m(mpacket m);//function that will initiate m packet  
spacket init\_s(spacket s);//function that will initaite s packet   
unsigned char bcc\_check;//function that checks master bcc and adds it to see if it's correct  
void bcc\_slave(void);  
void bcc\_master(void);  
void master\_write(void);//function that sends packet  
void slave\_packet\_read(void);//function that reads the packet  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
//LCD FUNCTIONS  
void init\_ports(void);  
void delay(unsigned int time);  
void lcd\_ready(void);  
void soft\_reset(void);  
void data\_write(unsigned char data);  
void cmd\_write(unsigned char cmd);  
void init\_lcd(void);  
unsigned char rd\_busy(void);  
int waitkeypressed(void);  
int waitkeyreleased(void);  
unsigned char anykey(void);  
void lcd\_write\_string(char \*p\_message);  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
void main(void)  
{  
 int Nakcount=0; //variable  
 char good[]="ALL GOOD IN DA HOOD!";//string that writes good   
 char bad[]= "BAD BCC!";//string that writes bad bcc  
 char success[]= "SUCCESS!one1!";//string that writes Sucess!!  
 char fail[]= "lawl~ FAIL!";// strign that writes fail!  
 unsigned char check;//defining an unsigned char as check  
 char bcc\_check;  
 unsigned int cnt\_m;//defining an unsigned integer cnt\_m which will be used to  
 unsigned int cnt\_s;//defining an unsigned integer cnt\_m which will be used to  
 unsigned char\* p\_list;//structure pointer for master list  
 unsigned char\* ps\_list;//structure pointer for   
 char loop=0x00; //char varable called loop  
   
 init\_ports();//initialize ports  
 soft\_reset();//initialize soft reset  
 init\_lcd();//initialize lcd  
 cmd\_write(CLEAR\_LCD);//clear lcd  
   
   
   
   
 while(loop<1) //keep going until loop is less than 1  
 {  
  loop=0x00; //loop is 0  
  while(!kbhit());//loop here until byte in UART Rx buffer   
     
  m.stx  =getch(); //initializing m packet struct for stx  
    
  while(m.stx !=STX){ // if m.stx is not 0x02;  
  m.stx=  getch(); //initializing m packet struct for stx  
  }  
    
  m.address =getch(); //initializing m packet struct for address  
  m.command =getch(); //initializing m packet struct for command  
  m.data  =getch(); // initializing m packet struct for data  
  m.bcc   =getch(); // initializing m packet struct for bcc  
  m.etx  =getch(); // initializing m packet struct for etx  
   
  bcc\_check = (m.stx + m.address + m.command + m.data + m.etx+ m.bcc); //calcuation 2's complement checksum  
   
  if(bcc\_check!=0x00)  // if BCC Checking is not zero  proceed to the next step  
  {  
      
     
   s.ack = NAK; // initalize slave packet with info  
   s.address = S\_ADDR;//  
   s.data = m.data; //  
   s.etx = ETX; //  
   s.bcc = (~(s.ack + s.address + s.data + s.etx) + 1); //calcutoion 2's complement checksum for the slave packet  
   
   putch(s.ack); //there putchar send the data to the master one byte at a time  
   putch(s.address);   
   putch(s.data);   
   putch(s.etx);    
   putch(s.bcc);   
     
   Nakcount++; //increment Nakcount variable  
   while(Nakcount<3) { //keep going until Nakcount is less than 3  
    cmd\_write(FIRSTLINE);  
    lcd\_write\_string(bad);//brings "BAD BCC"  
      
    }  
     
  }  
    
  if(bcc\_check == 0x00) // if BCC chekcing is zero proceed to the next step  
  {    
   if (m.command == 0x3C) //if master  command is 0x67, proceed to the next step  
   {  
    cmd\_write(FIRSTLINE); //first line LCD  
    data\_write(m.data);//bring the master data  
    cmd\_write(SECONDLINE);// second lince LCD  
    lcd\_write\_string (success);///brings  " sucess"string on LCD  
     
    s.ack = ACK;  
    s.address = M\_ADDR;  
    s.data= m.data;  
    s.etx = ETX;  
    s.bcc=(~(s.ack + s.address + s.data + s.etx)+1);  
      
    putch(s.ack);//there putchar send the data to the master one byte at a time   
    putch(s.address);  
    putch(s.data);  
    putch(s.etx);  
    putch(s.bcc);  
      
    loop=1; // stop the while loop   
      
   }  
   else if(m.command == 0x67) //if master  command is 0x67, proceed to the next step  
   {  
    cmd\_write(FIRSTLINE);//first line LCD  
    lcd\_write\_string (success);//bring "sucess" string on LCD  
    cmd\_write(SECONDLINE);// second lince LCD  
    data\_write(m.data);//bring the master data  
     
    s.ack = ACK;  
    s.address = M\_ADDR;   
    s.data= m.data; //Save the values somewhere in memory  
    s.etx = ETX; //Bcc is 0x15  
    s.bcc=(~(s.ack + s.address + s.data + s.etx)+1);// //Calculation for BCC for Slave   
      
    putch(s.ack); //there putchar send the data to the master one byte at a time   
    putch(s.address);  
    putch(s.data);  
    putch(s.etx);  
    putch(s.bcc);   
      
    loop=1; // stop the while loop   
   }  
   else  
   {  
   cmd\_write(FIRSTLINE); //first line LCD  
   lcd\_write\_string("Invalid CMD"); //failure MSG   
   loop=1; // stop the while loop   
   }  
  }  
 }   
return;  
}