Project/Programming Questions
Introduction to Embedded Systems – Part I

Project 1

Write an 8051 assembly language program to send a block of 100 bytes from the external data memory starting at address 0x100 through the serial port UART. Configure the UART for the settings: baud = 9600, 8 data bits, 1 stop bit and parity = even.

Project 2

Write an embedded C program (for Keil cross compiler) for implementing even parity enabled data reception for 8051. On receiving a character, the parity of it is verified with the parity bit received. If the parity bit is non-matching, the ASCII character ‘NAK’ is sent to the serial port. The serial data communication is implemented through interrupts. The communication settings are: baud = 9600, 8 data bits, 1 stop bit and parity = even.

Project 3

Design the embedded hardware and firmware in Assembly language for an 8051 based home security system for intruder detection as per the requirements given below.
1. There are two intruder detection sensors placed at two locations within the premise of the home. The intruder detector sensor generates a low pulse upon detecting an intruder.
2. When the intruder detector sensor output is low, a speaker/buzzer connected to a port pins produces an audible alarm for 2 minutes and an LED corresponding to the intruder detection sensor which generated the intruder detection signal is turned ON during the alarm period.

Project 4

Write an embedded C program (for Keil C51 cross compiler) to display the character string “HELLO WORLD!” on the first line of a 2 line 16 character HD44780 standard compatible LCD and the character string “HAVE A NICE DAY” on second line of the LCD as per the requirements given below.
1. The LCD interface details are given as:
   a. LCD Enable (EN)= Port Pin P3.2
   b. LCD Read Write (RW) = Port Pin P3.3
   c. LCD Register Select (RS) = Port Pin P3.4
   d. LCD Backlight enable = Port Pin P3.5
   e. LCD Data Port = Port 0 (P0)
2. The string constants are stored in code (program memory)
3. The Embedded C program should implement the following LCD interface routines
   a. void assert_lcd_enable(void) – Function for enabling LCD
   b. void lcd_command_write(unsigned char data_byte) – Function for writing a character to the command register of LCD
   c. unsigned char lcd_command_read(void) – Function for reading the command register of LCD. Returns the command register value
   d. void lcd_data_reg_write(unsigned char data_byte) – Function for writing a character to the data register of LCD
   e. unsigned char lcd_data_reg_read(void) – Function for reading the data register (at the current address) of LCD. Returns the data register value
   f. void lcd_busy(void) – Function for checking the busy flag of LCD. Returns only when the busy flag is reset
   g. void lcd_clear(void) – Function for clearing the LCD

Shibu K V
Introduction to Embedded Systems – Part I

h. `char lcd_initialize(void)` – Function for initializing LCD. Returns 1 if succeeds else returns zero.

i. `void MoveCursorTo(unsigned char x, unsigned char y)` – Function for moving Cursor to Line y and character x in the display area for a 2 line 16 character LCD, y represents the number of lines x represents the character position within a line. y varies from 0 to 1 and x varies from 0 to 15. x and y in combination sets the address of the Display Data (DD)RAM. For line 1, first character (x=0, y=0) the DD(RAM address is 0x80. For line 2 first character (x=0, y=1) the DD(RAM address is 0xC0.

j. `char *display_string(char* str)` – Function for displaying a character string in current Cursor position. Input parameter str - pointer to the string which needs to be displayed on the LCD screen. Return Value - Pointer to the char string which is displayed on LCD

k. `void display_char(char ch)` - Function for displaying a character in current Cursor position. Input : ch - character to be displayed on the LCD screen.

4. Code the milliseconds and microseconds delays needed for the initialization routine of LCD in embedded c

5. Refer to the Digital clock case study for the initialization sequence for HD44780 standard compatible LCD

Project 5

An embedded device application talks to an external device using MODBUS protocol over RS-485 interface. The response from the device is stored in a character array of 20 bytes. The first byte of the array represents the ID of the device, second byte as the ‘function code’ and the third byte onwards as the MODBUS register values. The MODBUS registers are 16 bit wide and they store the data in IEE 754 Floating point format (for single precision). The device sends the response byte in the format, the lower order IEE 754 format data byte first; the next higher order, the next higher order and the last byte will be the MS byte. Write an embedded C program to print the first register details in floating point representation.