UNC Charlotte–ECGR4101/5101/6090-Midterm Exam –11/3/04

Name: ___**SOLUTION**_____ User ID _____

Question	1	2	3	4	Total
Score	/30	/20	/40	/60	/150

1) Write a fully functional subroutine called "initLEDs" that will set up the 30262-SKP board LEDs and turn them all off. Include full comments. To make the function work, you will need to ensure some files are available during compile. List all the files that need to be present for the function to compile. (30 points)

```
#include "sfr262.h"

#define RED_LED (p8_0) /* from board schem.*/
#define YEL_LED (p7_4)
#define GRN_LED (p7_2)
#define LED_ON  (0)   /* 0 is ON for LEDs */
#define LED_OFF (1)

#define DIR_IN  (0)
#define DIR_OUT (1)

 void init_LEDs() {
   pd8_0 = pd7_4 = pd7_2 = DIR_OUT;
   RED_LED = YEL_LED = GRN_LED = LED_ON;
   RED_LED = YEL_LED = GRN_LED = LED_OFF;
 }
```

2) What are the minimum and maximum baud (bit per second) rates at which UART1 of a 20 MHz M16C30262 can communicate? Remember to take advantage of the internal clock source selection options. (20 points)

Maximum=20/16*1 =1.25Mbps (10 points)

Minimum=20/16*256*32=0.1525Kbps (10 points)

3) Write C code to configure the timers to generate an interrupt every 60 seconds (or as close as possible). Generate an output pulse on TA1OUT. Assume the MCU's clock runs at 20MHz. What is the actual period of the signal generated? (40 points)

To get 60 sec, need to cascade two timers. By cascading TA0 with TA1:

void init_timers(){
TA0MR=x80; //Timer A0 in timer mode with f32 clock (7.5 points)
TA0MR=x05; //Timer A1 in event count mode (7.5 points)
Ta0=0xf425; //Gives overflow every 0.1 sec. (5points)
Ta0=0x0258; //Gives overflow every 60 sec. (5 points)
TABSR=0x03; //start the timers (5 points)
TGRSR=0x02; //to cascade the two timers. (5 points)
}
Actual period for this solution is 60 sec. (5 points)

Consider the Lab 4 assignment.

a. Write a flowchart showing the program flow (20 points)

b. Write a subroutine to set up the receive and transmit UART0 (40 points)

Lab 4: In this lab you will be performing serial communications with polling. This lab will use the on-board UART to communicate between two boards. The LED's will be used for signaling and the LCD can be used to display debugging information. This lab must be demonstrated to the TA.
You will be expected to listen for several different commands. All valid commands will be transmitted in uppercase. The commands that are valid are R, Y, and G toggle the respective red, yellow, and green LEDs. All other characters should be considered invalid and should trigger the invalid response.

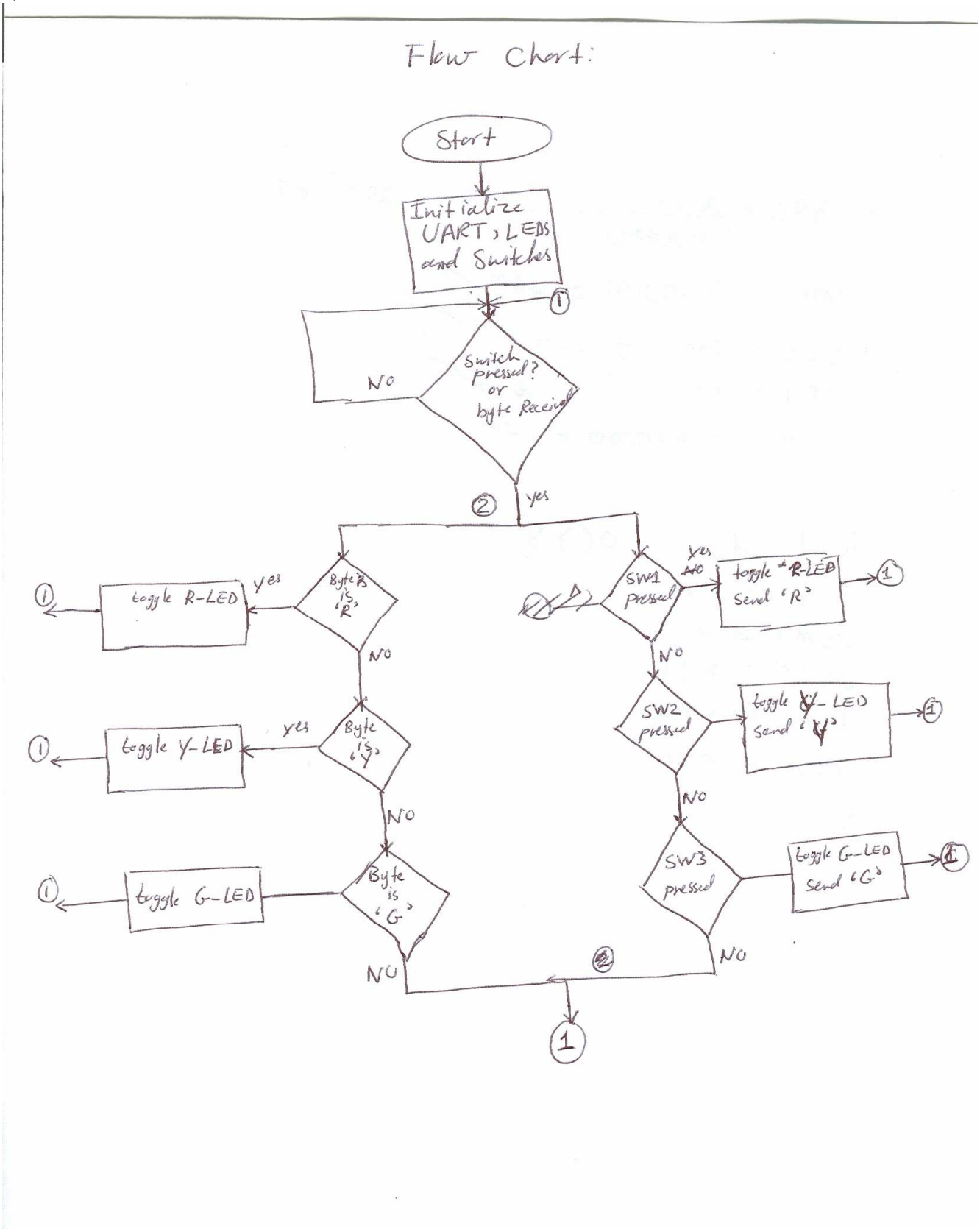
1. The program should always be polling for a new character.
2. If a character is received it should be checked for validity.
3. If the command is valid the program should act accordingly.
4. Once the command has been processed the program should poll for the next command.

Requirements
Req. 1 – The code generated is written in C for the SKP16C26.
Req. 2 – The code is well commented and easy to follow
Req. 3 – The serial communications should operate at 1200 baud, even parity, 8 data bits, one stop bit.
Req. 4 – The two student boards will be connected via the UART0 transmit, receive, and ground pins. The transmit pin on one board will be connected to the receive pin on the other board.
Req. 5 – Both boards will wait for a button press on its own board or a byte to be received from the other attached board.
Req. 6 – If "R" is received from the other board, then the Red LED is inverted on the board.
Req. 7 – If "Y" is received from the other board, then the Yellow LED is inverted on the board.
Req. 8 – If "G" is received from the other board, then the Green LED is inverted on the board.
Req. 9 – If SW1 is pressed, then the Red LED is inverted on the board and the character "R" is sent to the other board.
Req. 10 – If SW2 is pressed, then the Yellow LED is inverted on the board and the character "Y" is sent to the other board.
Req. 11 – If SW3 is pressed, then the Green LED is inverted on the board and the character "G" is sent to the other board.
Req. 12 – Once a command is processed the program returns to polling.
Req. 13 – The color of the LEDs on both boards must always match.

b)We need just the UART initialization as follows:

```
void init_UART(void){
u0brg=129;              //using f32 to get 1200bps              (10 points)
u0mr=0x65;      // to get 8 data bits, 1 stop bit, even parity.  (10 points)
u0c0=0x01;              //to get f32                            (10 points)
u0c1=0x05;              //to enable transmitter and receiver.   (10 points)
}
```

a)


Flow Chart:
Start
Initialize UART, LEDs and Switches
Switch pressed? or byte Received
NO
yes
Byte is 'R'
toggle R-LED
Byte is 'Y'
toggle Y-LED
Byte is 'G'
toggle G-LED
SW1 pressed
toggle R-LED
Send 'R'
SW2 pressed
toggle Y-LED
Send 'Y'
SW3 pressed
toggle G-LED
Send 'G'