ELEC 464 : MICROCOMPUTER SYSTEM DESIGN
1996/97 WINTER SESSION TERM 1

Solutions to Assignment 8

Question 1

There are, of course, many possible solutions. The program below is an ad-hoc solution. This approach is suitable for relatively simple problems such as this. For more complex problems a more structured approach using one or more state machines, foreground/background tasks and timers would be more appropriate.

```
/*
   Sample program for ELEC 464
   microcontroller assignment.

   Ed Casas, 96/10/30

   The code required to implement the bonus
   solution gets compiled in if the
   preprocessor symbol BONUS is defined. */

/* The 8051.h include file is specific to the
   Hi-Tec 8051 C compiler.  It defines the
   8051 I/O ports (in this case, P1 and
   P3). */

#include <8051.h>

#define u_char unsigned char

/* The various LED patterns that are output
   to port P1 to generate the traffic light
   displays.  P1.7 to P1.5 (the MS 3 bits)
   drive the Red/Yellow/Green traffic light
   LEDs and P1.4 to P1.3 drive the Red/Green
   pedestrian crossing LEDs.  These are
   active-low outputs so setting a bit to ’0’
   (output low) causes the corresponding LED
   to be turned on. The symbol LED_xy
   corresponds to a traffic light of colour x
   and pedestrian light of colour y where
   R=red, Y=yellow and G=green. */

                          /* R Y G R  G X X X */
#define LED_ON  0x07      /* 0 0 0 0  0 1 1 1 */
#define LED_OFF 0xff      /* 1 1 1 1  1 1 1 1 */
#define LED_GR  0xcf      /* 1 1 0 0  1 1 1 1 */
#define LED_YR  0xaf      /* 1 0 1 0  1 1 1 1 */
#define LED_RG  0x77      /* 0 1 1 1  0 1 1 1 */
#define LED_RR  0x6f      /* 0 1 1 0  1 1 1 1 */

/* Patterns required to "blink" the lights.
   Notation is same as above with O=off. */

#define LED_OR  0xef      /* 1 1 1 0  1 1 1 1 */
#define LED_RO  0x7f      /* 0 1 1 1  1 1 1 1 */
```

```
/* Macro to convert a digit (0-9) into the value
   required to display a number in binary on the
   MS 4 bits of P1. */

#define NUMTOLED(n)  ( ~((n)<<4) | 0xf )

/* A bit mask to extract the bit of P3 that
   contains the current push button
   status. This input is active-low (when the
   button is pushed the input goes low). */

#define BUTTONMASK 0x04

/* The sequence of test and group ID patterns
   to be displayed on the LEDs when the
   program starts up. */

#define NPATTERN 4
u_char testpattern [NPATTERN] = {
  LED_ON, LED_OFF, NUMTOLED(9), NUMTOLED(6) } ;

/* Timing loop constant.  ’MSLOOP’ is the
   number of empty "for" loops required to
   cause a delay of about one millisecond.
   NOTE: This is compiler- and hardware-
   dependent. */

#define MSLOOP 55

/* Global flag indicating "button pushed"
   state. */

char pushed ;

/* Wait for ’ms’ milliseconds.  The button is
   polled every millisecond and the global
   "pushed" flag is set if the button is
   pushed. ’ms’ must be >=1 for the button to
   be polled.  */

void waitms ( int ms )
{
  int i ;

  while ( ms-- > 0 ) {
    i = 100 ;
    /* 1 ms delay: */
    for ( i=MSLOOP ; i > 0 ; i-- ) ;
    if ( ( P3 & BUTTONMASK ) == 0 )
      pushed = 1 ;
  }
}


/* Set the LEDs.  The value ’bits’ is written
   to port P1. */

void setleds (u_char bits)
{
  P1 = bits ;
```

```
}


#ifdef BONUS

/* Output ’n’ sequences of p1 and p2 with a
   500 ms period. The blink frequency is 2 Hz
   and the duration is n/2 seconds. */

void blink ( u_char p1, u_char p2, u_char n )
{
  for ( ; n > 0 ; n-- ) {
    setleds ( p1 ) ;
    waitms ( 250 ) ;
    setleds ( p2 ) ;
    waitms ( 250 ) ;
  }
}

/* As above, but terminates when button is
   pushed. */

void waitblink ( u_char p1, u_char p2 )
{
  char i ;
  while ( ! pushed ) {
    setleds ( p1 ) ;
    for ( i=0 ; !pushed && i<25 ; i++ )
      waitms ( 10 ) ;
    setleds ( p2 ) ;
    for ( i=0 ; !pushed && i<25 ; i++ )
      waitms ( 10 ) ;
  }
}

#endif

void main(void)
{
  char i ;

  /* Show the test and ID patterns on the
     LEDs.  */

  pushed = 0 ;
  while ( ! pushed ) {
     for ( i=0 ; i<NPATTERN ; i++ ) {

       /* show a test/ID pattern for 1 second */

        setleds ( testpattern[i] ) ;
        waitms ( 1000 ) ;

        /* flash LED’s to indicate end of a
           pattern in case a pattern repeats
           or is all-zero */

        setleds ( LED_OFF ) ; waitms ( 100 ) ;
        setleds ( LED_ON ) ;  waitms (  50 ) ;
        setleds ( LED_OFF ) ; waitms ( 100 ) ;
      }
  }

  /* end of diagnostics, assume no pedestrian
     yet */

  pushed = 0 ;
  setleds ( LED_GR ) ;
```

```
  while ( 1 ) {     /* loop forever */

    /* wait for pedestrian */
    while ( ! pushed ) {
#ifdef BONUS
      waitblink ( LED_GR, LED_OR ) ;
#else
      waitms ( 1 ) ;
#endif
    }

    /* turn traffic light yellow, wait 1 s */
    setleds ( LED_YR ) ;
    waitms ( 1000 ) ;

    /* turn traffic light red, pedestrian
    light green, reset "pedestrian is
    waiting" flag, and wait for 1 s */

    setleds ( LED_RG ) ;
    pushed = 0 ;
    waitms ( 1000 ) ;

    /* turn pedestrian light red for 2 s */
#ifdef BONUS
    blink ( LED_RR, LED_RO, 4 ) ;
#else
    setleds ( LED_RR ) ;
    waitms ( 2000 ) ;
#endif

    /* turn traffic light green for 5 s */
#ifdef BONUS
    blink ( LED_GR, LED_OR, 10 ) ;
#else
    setleds ( LED_GR ) ;
    waitms ( 5000 ) ;
#endif

  }

}
```