## Appendix A: C LANGUAGE SOFTWARE INTERFACE

`259macros.h`

- Include file for all I/O declarations and definitions. Function prototypes for 259library.c.

`259library.c`

- Core functions to access the DE1 board I/O. Include this file as part of your C program.

Add "`259library.c`" in the Altera Monitor Program's list of source files. At the top of your own C files, put `#include "259macros.h"`. *You should read the contents of both files.*

### Graphics (VGA) – draw pixels and put characters on the VGA output

`MAX_X_PIXELS` and `MAX_Y_PIXELS`, `MAX_X_CHARS` and `MAX_Y_CHARS`

- The size of the VGA graphics screen in pixels, and size of the text screen in characters.

```
void initScreen()
```

- Initializes the graphics framebuffer. Always use this, or you may get unreliable colours.

```
void fillScreen( unsigned short colour )
```

- Fills the entire screen with the same colour pixels. Use this to erase the screen.

```
void drawPixel( int x, int y, unsigned short colour )
```

- Writes the specified colour value to one pixel at location <x,y>. The top-left corner is <0,0>.

```
unsigned short makeColour( int r, int g, int b )
```

- Returns the colour value formed by mixing the values specified by r, g, b. The values of r, g, and b represent an intensity level between 0 and 63 each (0=dark/black, 63=bright).

```
unsigned short *getPixelAddr( int x, int y )
```

- Returns the address of position <x,y> in the pixel frame buffer. The top-left corner is <0,0>. An out-of-range location is clamped to the inner edge of the screen (so you'll see a mistake).

```
void drawChar( int x, int y, int c )
```

- Writes the ASCII character c to location <x,y>. To erase it later, write a ' ' space.

```
unsigned char *getCharAddr( int x, int y )
```

- Returns the address of position <x,y> in the character buffer.

### JTAG SERIAL – put and get characters from the Terminal window.

```
int getcharJTAG()
```

- Reads a character from Terminal. It waits until a character is typed.

```
unsigned int getJTAG_nowait()
```

- Reads "raw" word from Terminal. If successful, bit 15 = 1 and lower 8 bits contain character.

```
void putcharJTAG( int c )
```

- Writes character **c** to Terminal. It waits if the transmitter FIFO is full.

```
void printstringJTAG( char *psz )
```
- Writes a null-terminated string of characters to Terminal. It uses putcharJTAG().

```
void printbyteJTAG( int c )
```
- Converts the low 8-bits of **c** into a hex number and prints it as 5 letters: '0xHH '.

**PS/2 PORT: KEYBOARD AND MOUSE**

```
int getcharPS2()
```
- Reads a character from PS2 port. It waits until a character is received.

```
unsigned int getPS2_nowait()
```
- Reads a character from the PS2 port. If successful, the upper 16 bits will be non-zero and the lower 8 bits contain the received character.

```
void putcharPS2( int c )
```
- Writes character **c** to the PS2 port. It will wait if the transmitter FIFO is full.

```
void flushPS2()
```
- Discards all incoming characters from the PS2 port for the next 100ms.

```
int resetPS2()
```
- Sends reset codes and initializes the PS2 port device. Returns PS2_MOUSE or PS2_KB if successful; returning anything else indicates an error. This function repeatedly tries to reset the device until successful – this may take several tens of milliseconds.

```
int getMouseMotion( int *pdx, int *pdy, int *pbuttons )
int getMouseMotion_nowait( int *pdx, int *pdy, int *pbuttons )
```
- Checks for mouse motion and returns 1 if successful, or 0 if it detects data errors or no data. This function may discard several mouse messages if it gets confused, which may take several milliseconds and mouse movements to resolve. If successful, it modifies the integers pointed to by pdx and pdy to contain the amount of motion in "delta x" and "delta y". The status of the 3 buttons is reflected in the lowest 3 bits of the integer pointed to by pbuttons.

**COUNTER**

```
#define ONE_MS 50000          // delay_amount for 1ms of time
void timedDelay( int delay_amount )
```
- Waits until the specified delay_amount has passed since the **previous** call to timedDelay(). Upon exit, it remembers the current timestamp to prepare for the next call. The delay_amount is specified in number of 50MHz clock ticks. Thus, the following code runs exactly 1 loop iteration every 10 milliseconds:

```
while( 1 ) {
     timedDelay( 10 * ONE_MS );  // wait 10ms since last call
     printstringJTAG( "10 ms has elapsed" );
     for( i=0; i < 3000; i++ ); // waste time
}
```

NAME: _____ STUDENT #: _____

**EECE 259: Introduction to Microcomputers** **Lecture Quiz** **Mar 9, 2011**

1. Write a C and assembly program to count the number of switches that are set to '1' and place the count on the green LEDs. The program should run continuously. (Same as Study Question #1 from Homework 4)

```
#include "259macros.h"
        // defines pSWITCH and pLEDG

int main( int argc, char *argv[] )
{
```

```
.include "ubc-de1media-macros.s"

.global _start
.text
_start:
```

NAME: _____ STUDENT #: _____

**EECE 259: Introduction to Microcomputers** **Lecture Quiz** **Mar 9, 2011**

1. Write a C and assembly program to count the number of switches that are set to '1' and place the count on the green LEDs. The program should run continuously. (Same as Study Question #1 from Homework 4)

```
#include "259macros.h"
        // defines pSWITCH and pLEDG


int main( int argc, char *argv[] )
{
      int count;          // r8
      unsigned int sw;    // r9

      while(1) {
        count = 0;
        sw = *pSWITCH;
        sw = sw & 0x3ff;

        while(sw) {


            if(sw & 1)
            {
               count++;
            }

            sw = sw >> 1;

        }


        *pLEDG = count;

      }
}
```

```
.include "ubc-de1media-macros.s"

.global _start
.text

_start:

    movia   r23, IOBASE


loop:
    movi    r8, 0             // count=r8
    ldwio   r9, SWITCH(r23)   // sw=r9
    andi    r9, r9, 0x3ff

whilesw:
    beq     r9, r0, output

    andi    r10, r9, 1
    beq     r10, r0, skipcount
    addi    r8, r8, 1

skipcount:
    srli    r9, r9, 1

    br      whilesw

output:
    stwio   r8, LEDG(r23)

    br      loop



.data
    /* not used */

.end
```