EE 3921

Dr. Johnson

Homework 8

1 – Given the driver code below; P_base is the pio base address, with the pio ports tied to LEDs and configured as outputs only, what is the LED flashing pattern for the code snippet below. Assume usleep is available. 40pts

foo_drv.h

```
#include "alt_types.h"
#include "io.h"

#define pio_fee(base)   IORD(base, 0)
#define pio_fie(base, val)  IOWR(base, 0, val)
#define pio_foe(base)   IORD(base, 3)
#devine pio_fum(base)  IOWR(base, 3, 0x0F)

void pio_foo(alt_u32 base, alt_u8 val);
```

foo_dvr.c

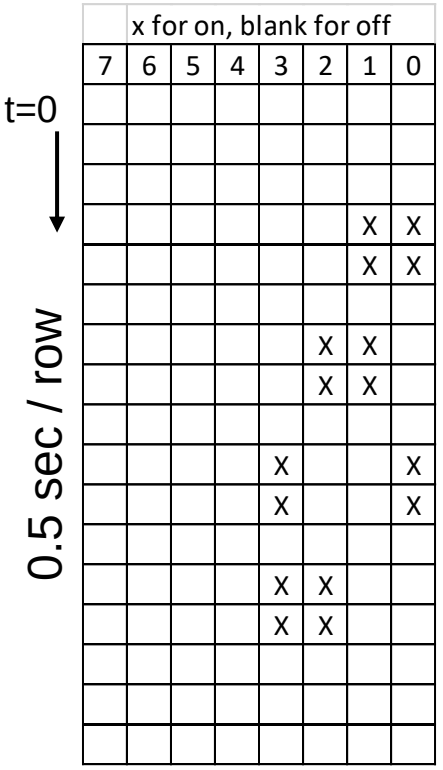
```
#include “foo_drv.h"

void pio_foo(alt_u32 bse, alt_u8 val){
 int i;
   for (i = 0; i < val, i++){
       pio_fie(bse, 3*i);
       usleep(1000000);
       pio_fie(bse, 0);
       usleep(500000);
   }
```

code snippit

```
…
int ptn = 0;
pio_foo(P_base, 5);
}
…
```

LED Pattern



2 – In EE2920 you needed to use a digital line sensor. The sensor required youto pull a pin high for 100us, then change the same pin to an input and determine when it fell below the input logic threshold to a 0. 60pts

a) Identify the Platform Designer parameters for an 8 bit PIO to support this operation.
b) Write a snippit of code to force the pin(bit 4) high for 100us, and then release the pin. Include a test for when the input is recognized as a 0 with approximately 10us resolution. (you can assume usleep() is available)

8 bit
set bidir

enable
individual bit
setting

```
int transition = 0;
int foo;
foo = IORD_ALTERA_AVALON_PIO_DIRECTION(PIO_BASE) | 0x10;                 // read dir and set desired bit
IOWR_ALTERA_AVALON_PIO_DIRECTION(PIO_BASE, foo);                         // write direction
IOWR_ALTERA_AVALON_PIO_SET_BITS(PIO_BASE, 0x10);                         // set pin 4 high(indiv bit setting)
usleep(100);
foo = = IORD_ALTERA_AVALON_PIO_DIRECTION(PIO_BASE)  & ~ 0x10;            // read dir and clr desired bit
IOWR_ALTERA_AVALON_PIO_DIRECTION(PIO_BASE, foo);                         // write direction
while(IORD_ALTERA_AVALON_PIO_DATA(PIO_BASE) & 0x10){                     // check for 1 on input pin 4
    usleep(10);
    transition++;
}
```

8 bit
set bidir

```
int transition = 0;
IOWR_ALTERA_AVALON_PIO_DIRECTION(PIO_BASE, (IORD_ALTERA_AVALON_PIO_DIRECTION(PIO_BASE) | 0x10));      // set pin 4 as output
IOWR_ALTERA_AVALON_PIO_DATA(PIO_BASE, (IORD_ALTERA_AVALON_PIO_DATA(PIO_BASE) | 0x10));                // set pin 4 high
usleep(100);
IOWR_ALTERA_AVALON_PIO_DIRECTION(PIO_BASE, (IORD_ALTERA_AVALON_PIO_DIRECTION((PIO_BASE) & ~0x10));   // set pin 4 to input
while(IORD_ALTERA_AVALON_PIO_DATA(PIO_BASE) & 0x10){                                                  // check for 1 on input pin 4
    usleep(10);
    transition++;
}
```

2 – In EE2920 you needed to use a digital line sensor. The sensor required you to pull a pin high for 100us, then change the same pin to an input and determine when it fell below the input logic threshold to a 0. 60pts

a) Identify the Platform Designer parameters for an 8 bit PIO to support this operation.

b) Write a snippit of code to force the pin(bit 4) high for 100us, and then release the pin. Include a test for when the input is recognized as a 0 with approximately 10us resolution. (you can assume usleep() is available)

8 bit
set bidir
enable individual bit setting

```
int transition = 0;
int foo;
foo = IORD(PIO_BASE,1) | 0x10;          // read dir and set desired bit
IOWR(PIO_BASE, 1, foo);                 // write direction
IOWR(PIO_BASE, 4, 0x10);                // set pin 4 high(indiv bit setting)
usleep(100);
foo = IORD(PIO_BASE,1) & ~ 0x10;        // read dir and clr desired bit
IOWR(PIO_BASE, 1,foo);                  // set pin 4 to input
while(IORD(PIO_BASE,0) & 0x10){         // check for 1 on input pin 4
    usleep(10);
    transition++;
}
```

8 bit
set bidir

```
int transition = 0;
IOWR(PIO_BASE, 1, (IORD(PIO_BASE, 1) | 0x10));   // set pin 4 as output
IOWR(PIO_BASE, 0, (IORD(PIO_BASE, 0) | 0x10));   // set pin 4 high
usleep(100);
IOWR(PIO_BASE, 1, (IORD(PIO_BASE, 1) & ~0x10)); // set pin 4 to input
while(IORD(PIO_BASE, 0) & 0x10){                 // check for 1 on input pin 4
    usleep(10);
    transition++;
}
```