Program Structure (2A)

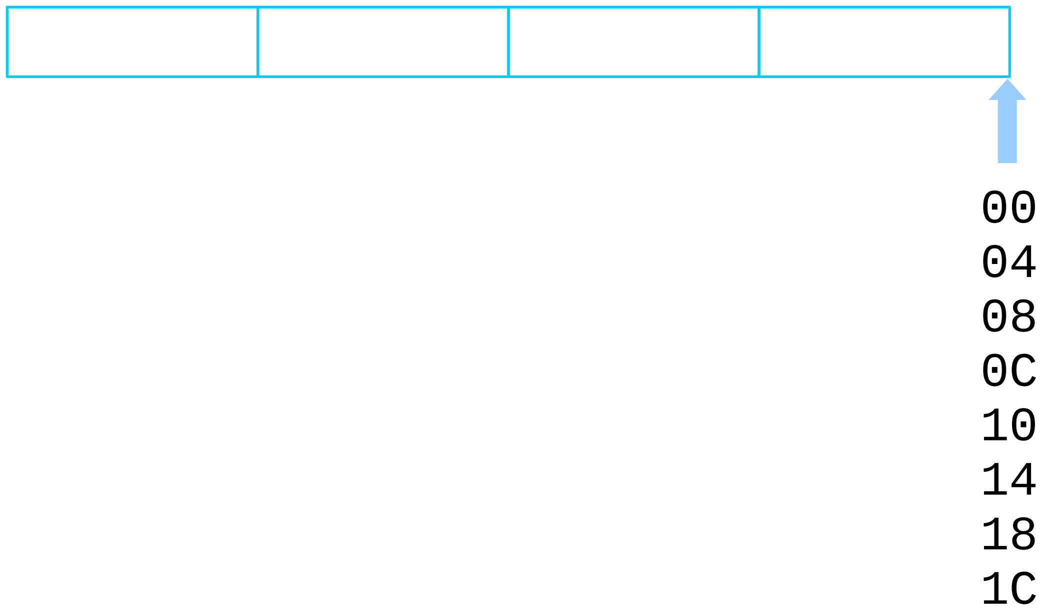
Young Won Lim
5/28/18

Based on Embedded Software in C for an ARM Cortex M
http://users.ece.utexas.edu/~valvano/Volume1/

Port Address

```
#define GPIO_PORTA_DATA_R    (*((volatile unsigned long *) 0x400043FC))
#define GPIO_PORTA_DIR_R     (*((volatile unsigned long *) 0x40004400))
#define GPIO_PORTA_DEN_R     (*((volatile unsigned long *) 0x4000451C))
#define SYSCTL_PRGPIO_R      (*((volatile unsigned long *) 0x400FEA08))
```

volatile – no optimization



Punctuations

;	end of a **statement**
:	defines a **label**
,	separates **elements** of a list
( )	start and end of a **parameter list**
{ }	start and end of a **compound statement**
[ ]	start and end of a **array index**
" "	start and end of a **string**
' '	start and etop of a **character constant**

Single character operators

=	assignment statement
&	address of
?	selection
<	less than
>	greater than
!	logical not (true to false, false to true)
~	bitwise not (1's complement)
+	addition
-	subtraction
*	multiply / pointer reference
/	divide
%	modulo, division remainder
|	bitwise or
&	bitwise and / address of
^	bitwise exclusive or
.	used to access parts of a structure

Multiple character operators

==	equal to
<=	less than or equal to
>=	greater than or equal to
!=	not equal to
<<	shift left
>>	shift right
++	increment
--	decrement
&&	logical and
||	logical or
->	pointer to a structure

Multiple character compound operators

+=	assign add value to
-=	assign subtract value to
*=	assign multiply value to
/=	assign divide value to
|=	assign or value to
&=	assign and value to
^=	assign exclusive or value to
<<=	assign shift value left
>>=	assign shift value right
%=	assign modulo divide value to

Precedence

Precedence	Operators Associativity	
Highest	() [ ] . -> ++(postfix) --(postfix)	left to right
	++(prefix) --(prefix) ! (not) ~(not) sizeof(type) +(unary) –(unary) &(address) *(dereference)	right to left
	* / %	left to right
	+ -	left to right
	<< >>	left to right
	< <= > >=	left to right
	== !=	left to right
	&	left to right
	^	left to right
	|	left to right
	&&	left to right
	||	left to right
	? :	right to left
	= += -= *= /= %= <<= >>= |= &= ^=	right to left
lowest	,	left to right

Port IO

```
void Lock_Init(void){
  volatile unsigned long delay;

  SYSCTL_PRGPIO_R |= 0x01;        // activate clock for Port A
  delay = SYSCTL_PRGPIO_R;        // allow time for clock to start
  GPIO_PORTA_DIR_R = 0x80;        // set PA7 to output and PA6-0 to input
  GPIO_PORTA_DEN_R = 0xFF;        // enable digital port
}
```

If then else

```
#define PORTB (*((volatile unsigned long *) 0x400053FC))
#define PORTE (*((volatile unsigned long *) 0x400243FC))

void Example(void){
  if ((PORTE & 0x04)==0) {       /* test bit 2 of PORTE */
    PORTB = 0;                   /* if PORTE bit 2 is 0, then make PORTB=0 */
  } else {
    PORTB = 100;                 /* if PORTE bit 0 is not 0, then make PORTB=100 */
  }
}
```

While

```
#define PORTA (*((volatile unsigned long *) 0x400043FC))
#define PORTB (*((volatile unsigned long *) 0x400053FC))

void Example(void) {                /* loop until PORTB equals 200 */
  PORTB = 0;
  while (PORTB != 200) {
    PORTA = PORTA ^ 0x08;           /* toggle PORTA bit 3 output */
    PORTB++;                        /* increment PORTB output */
  }
}
```

For loop

```
#define PORTB (*((volatile unsigned long *) 0x400053FC))

void Example(void){                                    /* loop until PORTB equals 200 */
  for (PORTB=0; PORTB != 200; PORTB++) {
    PORTA = PORTA ^ 0x08;                              /* toggle PORTA bit 3 output */
  }
}
```

Functions

```
short add(short x, short y) {
        short z;

        z = x+y;
        if ((x>0) && (y>0) && (z<0)) z=32767;          // overflow : set to 0x7FFF
        If ((x<0) && (y<0) && (z>0)) z=-32768;         // underflow : set to 0x1000
        Return (z);
}

int main(void) {
        short a, b;
        a = add(2000,2000);
        b = 0;
        while(1) {
                b = add(b,1);                          // 0,1,2, …, 32767,32767,….
        }
}
```

Functions

```
short add(short x, short y) {
        short z;
        z = x+y;                                        /* z=4000*/
        if ((x>0)&&(y>0)&&(z<0)) z=32767;
        if ((x<0)&&(y<0)&&(z>0)) z=-32768;
        return(z);                                      /* return 4000 from call*/
}

int main(void) {
        short a,b;
        a = add(2000,2000);                             /* call to add*/
        b = 0;
        while (1) {
                b = add(b,1);                           /* call to add*/
        }
}
```

Compound Statements

```
// 3 wide 16-bit signed median filter
short median(short n1,short n2,short n3){
    if (n1>n2){
      if (n2>n3)
        return(n2);                    // n1>n2,n2>n3                 n1>n2>n3
      else{
          if (n1>n3)
            return(n3);                // n1>n2,n3>n2,n1>n3           n1>n3>n2
          else
            return(n1);                // n1>n2,n3>n2,n3>n1           n3>n1>n2
          }
      }
      else{
         if (n3>n2)
            return(n2);                // n2>n1,n3>n2                 n3>n2>n1
         else{
            if (n1>n3)
               return(n1);             // n2>n1,n2>n3,n1>n3           n2>n1>n3
            else
              return(n3);              // n2>n1,n2>n3,n3>n1           n2>n3>n1
         }
     }
}
```

Source Files

```
/* ****file tm4c123gh6pm.h (actually much bigger)************ */
#define GPIO_PORTA_DATA_R       (*((volatile unsigned long *) 0x400043FC))
#define GPIO_PORTA_DIR_R        (*((volatile unsigned long *) 0x40004400))
#define GPIO_PORTA_DEN_R        (*((volatile unsigned long *) 0x4000451C))
#define SYSCTL_PRGPIO_R         (*((volatile unsigned long *) 0x400FEA08))
```

```
/* ****file LOCK.h ************ */
void Lock_Init(void);
void Lock_Set(int flag);
unsigned long Lock_Input(void);
```

Source Files

```
/* ****file Lock.C ************ */
#include "tm4c123gh6pm.h"

void Lock_Init(void){ volatile unsigned long delay;
  SYSCTL_PRGPIO_R |= 0x01;   // activate clock for Port A
  delay = SYSCTL_PRGPIO_R;   // allow time for clock to start
  GPIO_PORTA_DIR_R = 0x80;   // set PA7 to output and PA6-0 to input
  GPIO_PORTA_DEN_R = 0xFF;   // enable digital port
}

void Lock_Set(int flag){
  if(flag){
    GPIO_PORTA_DATA_R = 0x80;
  }else{
    GPIO_PORTA_DATA_R = 0;
  }
}

unsigned long Lock_Input(void){
  return GPIO_PORTA_DATA_R & 0x7F; // 0 to 127
}
```

Source Files

```
/* ****file main.c ************ */

const unsigned char key=0x23;                  // The key code 0100011 (binary)
#include "Lock.h"

void main(void){
  unsigned char input;
  unsigned long cnt;

  Lock_Init();                                 // initialize lock
  cnt = 4000;
  while(1){
    input = Lock_Input();                      // input 8 bits from parallel port A
    if (key == input) {
      cnt--;                                   // debounce switches
      if (cnt == 0) {                          // done bouncing
        Lock_Set(1);                           // unlock door
      }
    } else {
      Lock_Set(0);                             // lock the door
      cnt = 4000;
    }
  }
}
#include "Lock.c"
```

References

[1] Essential C, Nick Parlante
[2] Efficient C Programming, Mark A. Weiss
[3] C A Reference Manual, Samuel P. Harbison & Guy L. Steele Jr.
[4] C Language Express, I. K. Chun
[5] “A Whirlwind Tutorial on Creating Really Teensy ELF Executables for Linux“
http://cseweb.ucsd.edu/~ricko/CSE131/teensyELF.htm
[6] http://en.wikipedia.org
[7] http://www.muppetlabs.com/~breadbox/software/tiny/teensy.html
[8] http://csapp.cs.cmu.edu/public/ch7-preview.pdf