Discussion of exam problems and scores

QUIZ: Configure for using PA4 for output

```
#define GPIO_PORTD_DATA_R   (*((volatile unsigned long *)0x400073FC))
#define GPIO_PORTD_DIR_R    (*((volatile unsigned long *)0x40007400))
#define GPIO_PORTD_AFSEL_R  (*((volatile unsigned long *)0x40007420))
#define GPIO_PORTD_DEN_R    (*((volatile unsigned long *)0x4000751C))
#define SYSCTL_RCGC2_R      (*((volatile unsigned long *)0x400FE108))
void PortD_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000008;  // 1) activate clock for Port D
  delay = SYSCTL_RCGC2_R;        // allow time for clock to start
  GPIO_PORTD_DIR_R = 0x0F;       // 2) set direction register
  GPIO_PORTD_AFSEL_R = 0x00;     // 3) regular port function
  GPIO_PORTD_DEN_R = 0xFF;       // 4) enable digital port
}
unsigned long PortD_Input(void){
  return (GPIO_PORTD_DATA_R>>4); // read PD7-PD4 inputs
}
void PortD_Output(unsigned long data){
  GPIO_PORTD_DATA_R = data;      // write PD3-PD0 outputs
}
```

Text sections

4.2.1 Bit-specific addressing

4.2.2 Switches and LEDs

Software Module Interaction

- Two or more software modules may access different elements of a shared I/O port
  - Each module must employ ***friendly*** operations when accessing/altering elements of the port
    - Only the elements the software module is ***authorized*** to modify can be modified
    - Elements used by other modules must not be modified

Remember: initialization “ritual”

Address	7	6	5	4	3	2	1	0	Name
400F.E108	GPIOH	**GPIOG**	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA	SYSCTL_RCGC2_R
4000.73FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTD_DATA_R
4000.7400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTD_DIR_R
4000.7420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTD_AFSEL_R
4000.751C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTD_DEN_R

- **Initialization (executed once at beginning)**
  1. Turn on clock in **`SYSCTL_RCGC2_R`**
  2. Wait two bus cycles
  3. Set *DIR* to 1 for output or 0 for input
  4. Clear *AFSEL* bits to 0 to select regular I/O
  5. Set *DEN* bits to 1 to enable data pins
- **Input/output from pin**
  6. Read/write **`GPIO_PORTX_DATA_R`**

Clock Gating

A through F on our MCU

Selected I/O Port Registers

Address	7	6	5	4	3	2	1	0	Name
$400F.E108			GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA	SYSCTL_RCGC2_R
$4000.43FC			DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTA_DATA_R
$4000.4400			DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTA_DIR_R
$4000.4420			SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTA_AFSEL_R
$4000.451C			DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTA_DEN_R
$4000.53FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTB_DATA_R
$4000.5400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTB_DIR_R
$4000.5420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTB_AFSEL_R
$4000.551C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTB_DEN_R
$4000.63FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTC_DATA_R
$4000.6400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTC_DIR_R
$4000.6420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTC_AFSEL_R
$4000.651C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTC_DEN_R
$4000.73FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTD_DATA_R
$4000.7400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTD_DIR_R
$4000.7420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTD_AFSEL_R
$4000.751C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTD_DEN_R
$4002.43FC							DATA	DATA	GPIO_PORTE_DATA_R
$4002.4400							DIR	DIR	GPIO_PORTE_DIR_R
$4002.4420							SEL	SEL	GPIO_PORTE_AFSEL_R
$4002.451C							DEN	DEN	GPIO_PORTE_DEN_R

EACH REGISTER IS 32 BITS WIDE (BITS 8-31 not used)

The complete list is in the datasheet of LM4F120 – Table 10-6 on p.632.
Hardcopy was provided in class.

IO register offsets

Table 10-6. GPIO Register Map

Offset	Name	Type	Reset	Description
0x000	GPIODATA	R/W	0x0000.0000	GPIO Data
0x400	GPIODIR	R/W	0x0000.0000	GPIO Direction
0x404	GPIOIS	R/W	0x0000.0000	GPIO Interrupt Sense

Source: p.631 of LM4F120-Tiva datasheet (linked on webpage, hardcopy of this table was provided in class)

Explain!

```
#define GPIO_PORTD_DATA_R   (*((volatile unsigned long *)0x400073FC))
#define GPIO_PORTD_DIR_R    (*((volatile unsigned long *)0x40007400))
#define GPIO_PORTD_AFSEL_R  (*((volatile unsigned long *)0x40007420))
#define GPIO_PORTD_DEN_R    (*((volatile unsigned long *)0x4000751C))
#define SYSCTL_RCGC2_R      (*((volatile unsigned long *)0x400FE108))
void PortD_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000008;  // 1) activate clock for Port D
  delay = SYSCTL_RCGC2_R;        // allow time for clock to start
  GPIO_PORTD_DIR_R = 0x0F;       // 2) set direction register
  GPIO_PORTD_AFSEL_R = 0x00;     // 3) regular port function
  GPIO_PORTD_DEN_R = 0xFF;       // 4) enable digital port
}
unsigned long PortD_Input(void){
  return (GPIO_PORTD_DATA_R>>4); // read PD7-PD4 inputs
}
void PortD_Output(unsigned long data){
  GPIO_PORTD_DATA_R = data;      // write PD3-PD0 outputs
}
```

I/O Port Bit-Specific

- I/O Port bit-specific addressing is used to access port data register
  - Define address offset as $4*2^b$, where b is the selected bit position
  - 256 possible bit combinations (0-8)
  - Add offsets for each bit selected to base address for the port
  - Example: Port A, bits 1,2,3

If we wish to access bit	*Constant*
7	0x0200
6	0x0100
5	0x0080
4	0x0040
3	0x0020
2	0x0010
1	0x0008
0	0x0004

Port A = 0x4000.4000
0x4000.4000+0x0008+0x0010+0x0020
= 0x4000.4038

Provides friendly and atomic access to port pins

Address	Contents
0x4000 4000	
0x4000 4001	
0x4000 4002	
0x4000 4003	
0x4000 4004	
0x4000 4005	
0x4000 4006	
0x4000 4007	
0x4000 4008	
0x4000 4009	
0x4000 400A	
0x4000 400B	
0x4000 400C	
0x4000 400D	
0x4000 400E	
0x4000 400F	

0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	

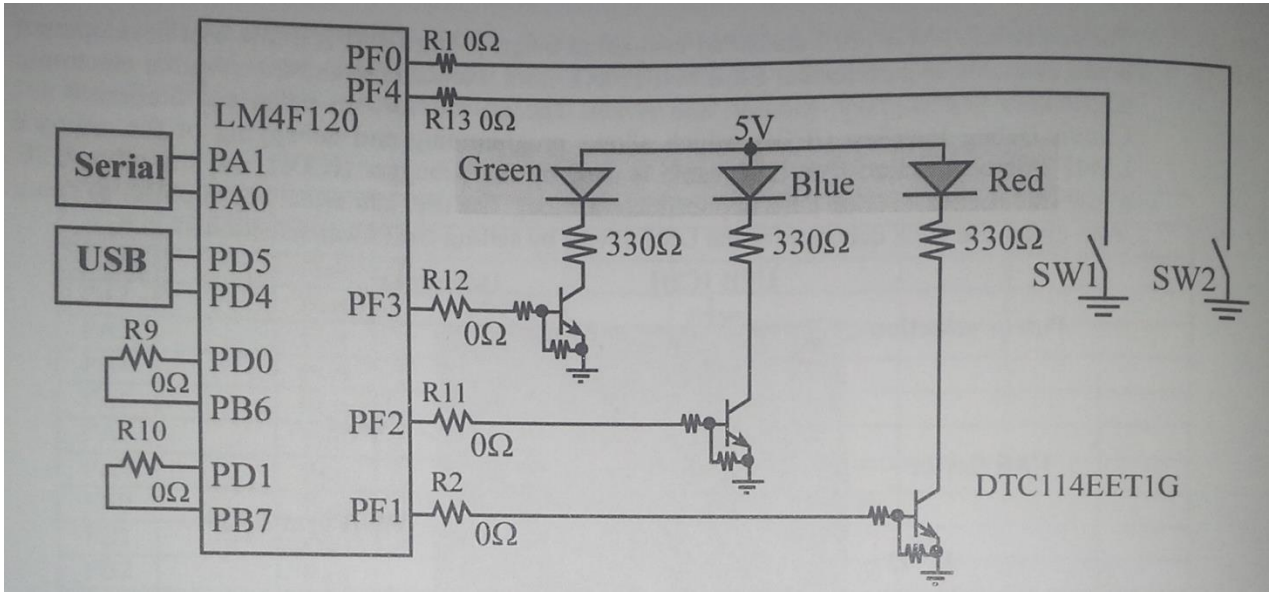
0x4000 41D8	
0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	

0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	
0x4000 4	

End of bit-specific block

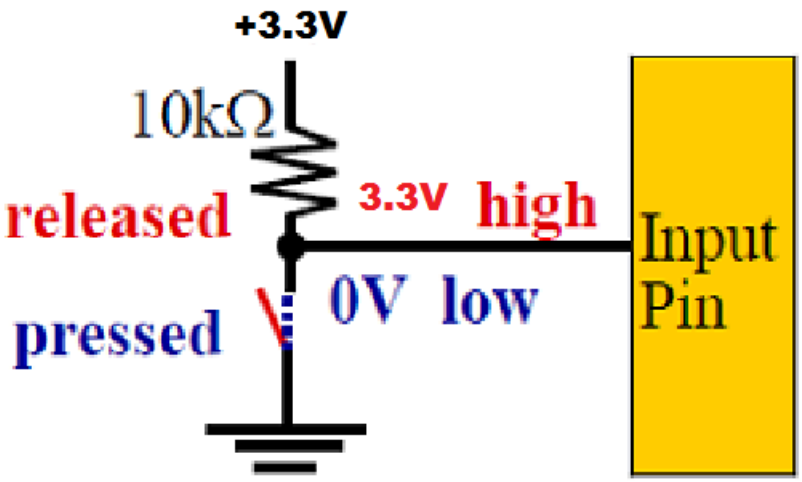
On the prev. memory diagram:

- Fill out all the missing address bits
- Write next to each word what specific bits correspond to it
- Calculate the address of the block corresponding to the following bit combinations:
  - PA7, 6, and 5
  - PB3, 2, 1, 0
  - PC6, 4, 2, 0

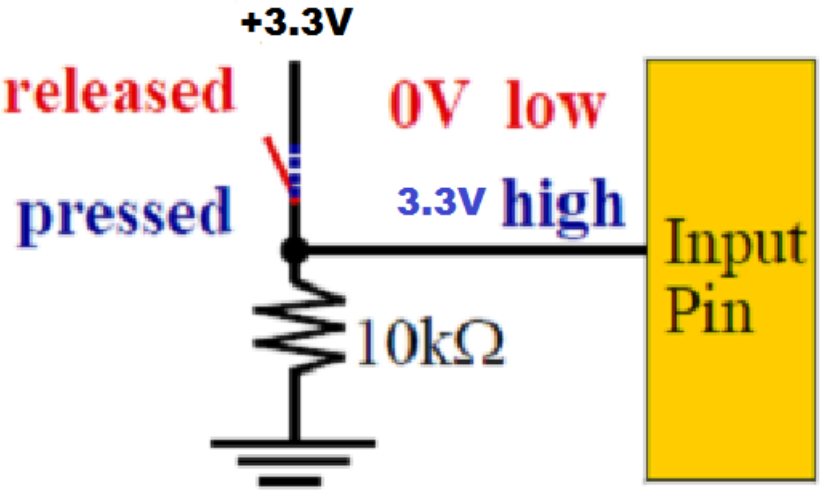


Remember: The Stellaris board already has 2 switches and 3 LEDs connected to MCU pins.
But if we need more …

Switch Configuration

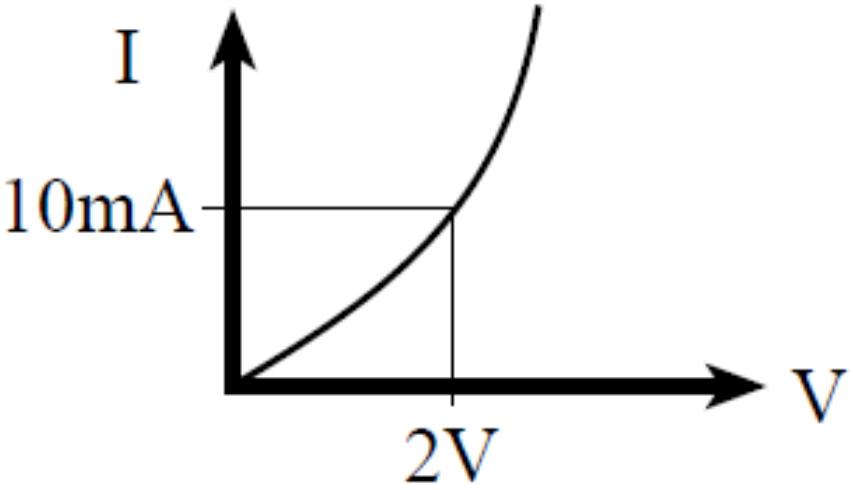


negative – pressed = ‘0’



positive – pressed = ‘1’

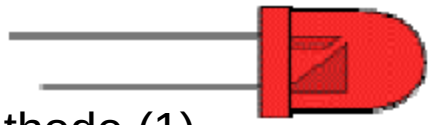
LED Interfacing



LED current v. voltage

Brightness = power = V*I

anode (+)

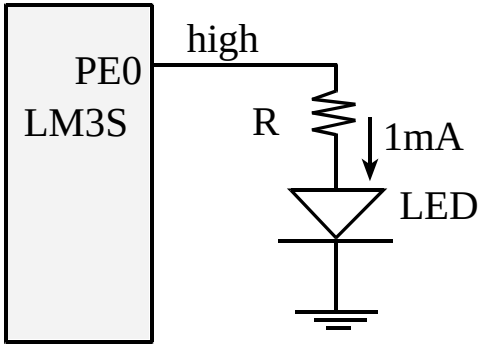


cathode (1)

"*big voltage connects to big pin*"

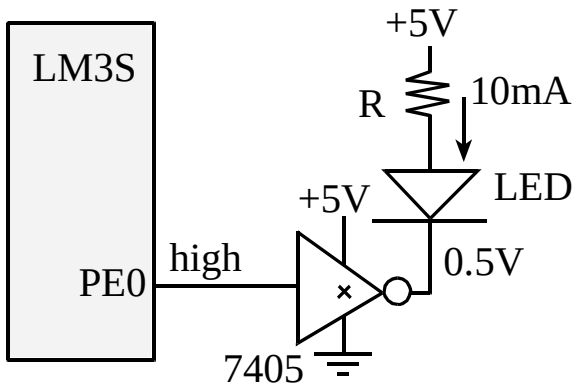
LED Interfacing

R = (3V – 1.5)/0.001
= 1.5 kOhm



LED current < 8 ma

R = (5.0-2-0.5)/0.01
= 220 Ohm



LED current > 8 ma

LED may contain several diodes in series

How do we define the constant PA5 for bit-specific addressing?

```
void Switch_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000001;  // activate clock for Port A
  delay = SYSCTL_RCGC2_R;        // allow time for clock to start
  GPIO_PORTA_DIR_R &= ~0x20;     // PA5 input
  GPIO_PORTA_AFSEL_R &= ~0x20;   // PA5 regular port function
  GPIO_PORTA_DEN_R |= 0x20;      // enable PA5 digital port
}
unsigned long Switch_Input(void){
  return PA5; // return 0x20(pressed) or 0(not pressed)
}
unsigned long Switch_Input2(void){
  return (GPIO_PORTA_DATA_R&0x20); // 0x20(pressed) or 0(not pressed)
}
```

Program 4.2a. Software interface for a switch on PA5 (Switch_xxx.zip).

How do we define the constant PA5 for bit-specific addressing?

```
#define PA5    (*((volatile unsigned long *)0x40004080))
```

```
void Switch_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000001;  // activate clock for Port A
  delay = SYSCTL_RCGC2_R;        // allow time for clock to start
  GPIO_PORTA_DIR_R &= ~0x20;     // PA5 input
  GPIO_PORTA_AFSEL_R &= ~0x20;   // PA5 regular port function
  GPIO_PORTA_DEN_R |= 0x20;      // enable PA5 digital port
}
unsigned long Switch_Input(void){
  return PA5; // return 0x20(pressed) or 0(not pressed)
}
unsigned long Switch_Input2(void){
  return (GPIO_PORTA_DATA_R&0x20); // 0x20(pressed) or 0(not pressed)
}
```

Program 4.2a. Software interface for a switch on PA5 (Switch_xxx.zip).

Make PA7 output, the set, clear and toggle (use both methods!)

```
#define PA5    (*((volatile unsigned long *)0x40004080))
```

```
void Switch_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000001;  // activate clock for Port A
  delay = SYSCTL_RCGC2_R;        // allow time for clock to start
  GPIO_PORTA_DIR_R &= ~0x20;     // PA5 input
  GPIO_PORTA_AFSEL_R &= ~0x20;   // PA5 regular port function
  GPIO_PORTA_DEN_R |= 0x20;      // enable PA5 digital port
}
unsigned long Switch_Input(void){
  return PA5; // return 0x20(pressed) or 0(not pressed)
}
unsigned long Switch_Input2(void){
  return (GPIO_PORTA_DATA_R&0x20); // 0x20(pressed) or 0(not pressed)
}
```

Program 4.2a. Software interface for a switch on PA5 (Switch_xxx.zip).

Your turn: Configure PB7 and PB5 for output …

```
#define PA5    (*((volatile unsigned long *)0x40004080))
```

```
void Switch_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000001;  // activate clock for Port A
  delay = SYSCTL_RCGC2_R;        // allow time for clock to start
  GPIO_PORTA_DIR_R &= ~0x20;     // PA5 input
  GPIO_PORTA_AFSEL_R &= ~0x20;   // PA5 regular port function
  GPIO_PORTA_DEN_R |= 0x20;      // enable PA5 digital port
}
unsigned long Switch_Input(void){
  return PA5; // return 0x20(pressed) or 0(not pressed)
}
unsigned long Switch_Input2(void){
  return (GPIO_PORTA_DATA_R&0x20); // 0x20(pressed) or 0(not pressed)
}
```

Program 4.2a. Software interface for a switch on PA5 (Switch_xxx.zip).

… then set set 7 and clear 5 (use both methods!)

Example 4.1: Generating out-of-phase square waves on PG1 and PG0

```
PG10 EQU 0x4002600C
Start
     LDR R1, =SYSCTL_RCGC2_R
     LDR R0, [R1]       ; previous
     ORR R0, R0, #0x00000040   ; G clock
     STR R0, [R1]
     NOP
     NOP                ; time to finish
     LDR R1, =GPIO_PORTG_DIR_R
     LDR R0, [R1]       ; previous
     ORR R0, R0,#0x03 ; PG1,PG0 output
     STR R0, [R1]       ; set direction
     LDR R1, =GPIO_PORTG_AFSEL_R
     LDR R0, [R1]       ; previous
     BIC R0, R0,#0x03 ; disable alt funct
     STR R0, [R1]       ; set alt funct
     LDR R1, =GPIO_PORTG_DEN_R
     LDR R0, [R1]       ; previous
     ORR R0, R0, #0x03 ; PG1,PG0 digital
     STR R0, [R1]       ; set digital
     LDR R1, =PG10
     MOV R0, #0x01      ; initial output
loop STR R0, [R1]
     EOR R0, R0, #0x03 ; toggle
     B   loop
```

```
// C implementation
void main(void){
volatile unsigned long delay;

// Port G clock
  SYSCTL_RCGC2_R |= 0x40;
  delay = SYSCTL_RCGC2_R;

// PG1 PG0 outputs
  GPIO_PORTG_DIR_R |= 0x03;

// normal function
  GPIO_PORTG_AFSEL_R &= ~0x03;

// enable digital I/O on PG1-0
  GPIO_PORTG_DEN_R |= 0x03;

// initial output
  PG10 = 0x01; // PG1=0; PG0=1

  while(1){
    PG10 ^= 0x03; // toggle
  }
```

Modify the code to use port E instead!

```
PG10 EQU 0x4002600C
Start
     LDR R1, =SYSCTL_RCGC2_R
     LDR R0, [R1]       ; previous
     ORR R0, R0, #0x00000040   ; G clock
     STR R0, [R1]
     NOP
     NOP                ; time to finish
     LDR R1, =GPIO_PORTG_DIR_R
     LDR R0, [R1]       ; previous
     ORR R0, R0,#0x03 ; PG1,PG0 output
     STR R0, [R1]       ; set direction
     LDR R1, =GPIO_PORTG_AFSEL_R
     LDR R0, [R1]       ; previous
     BIC R0, R0,#0x03 ; disable alt funct
     STR R0, [R1]       ; set alt funct
     LDR R1, =GPIO_PORTG_DEN_R
     LDR R0, [R1]       ; previous
     ORR R0, R0, #0x03 ; PG1,PG0 digital
     STR R0, [R1]       ; set digital
     LDR R1, =PG10
     MOV R0, #0x01      ; initial output
loop STR R0, [R1]
     EOR R0, R0, #0x03 ; toggle
     B   loop
```

```
// C implementation
void main(void){
volatile unsigned long delay;

// Port G clock
  SYSCTL_RCGC2_R |= 0x40;
  delay = SYSCTL_RCGC2_R;

// PG1 PG0 outputs
  GPIO_PORTG_DIR_R |= 0x03;

// normal function
  GPIO_PORTG_AFSEL_R &= ~0x03;

// enable digital I/O on PG1-0
  GPIO_PORTG_DEN_R |= 0x03;

// initial output
  PG10 = 0x01; // PG1=0; PG0=1

  while(1){
    PG10 ^= 0x03; // toggle
  }
```

Checkpoint 4.8: Assume the instructions **`STR EOR B loop`** in Program 4.4 takes 2, 1, and 4 bus cycles respectively to execute, and assume the bus frequency is 50 MHz. Estimate the frequency of each square wave in Example 4.1?

To do for next time:
Read and understand the remaining examples in 4.2.2

QUIZ: Bit-specific addressing

```
#define P[redacted]  (*((volatile unsigned long *)0x40024004))
void LED_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= [redacted];
  delay = SYSCTL_RCGC2_R;
  GPIO_PORTE_DIR_R |= [redacted]
  GPIO_PORTE_AFSEL_R &= [redacted]
  GPIO_PORTE_DEN_R |= [redacted]
}
```

Fill in the missing parts of the”ritual”

QUIZ: Bit-specific addressing

```
#define PE0    (*((volatile unsigned long *)0x40024004))
void LED_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000010;  // activate clock for Port E
  delay = SYSCTL_RCGC2_R;        // allow time for clock to start
  GPIO_PORTE_DIR_R |= 0x01;      // PE0 output
  GPIO_PORTE_AFSEL_R &= ~0x01;   // PE0 regular port function
  GPIO_PORTE_DEN_R |= 0x01;      // enable PE0 digital port
}
```

QUIZ: Bit-specific addressing

```
#define PE0    (*((volatile unsigned long *)0x40024004))
void LED_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000010;  // activate clock for Port E
  delay = SYSCTL_RCGC2_R;        // allow time for clock to start
  GPIO_PORTE_DIR_R |= 0x01;      // PE0 output
  GPIO_PORTE_AFSEL_R &= ~0x01;   // PE0 regular port function
  GPIO_PORTE_DEN_R |= 0x01;      // enable PE0 digital port
}
```

Write in C a function that toggles the LED connected to pin PE0

```
#define PE0    (*((volatile unsigned long *)0x40024004))
void LED_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000010;  // activate clock for Port E
  delay = SYSCTL_RCGC2_R;        // allow time for clock to start
  GPIO_PORTE_DIR_R |= 0x01;      // PE0 output
  GPIO_PORTE_AFSEL_R &= ~0x01;   // PE0 regular port function
  GPIO_PORTE_DEN_R |= 0x01;      // enable PE0 digital port
}
```

```
void LED_Toggle(void){
  PE0 = PE0^1; // toggle LED
}
```

4.3 PLL

- LM4F120 has 16 MHz crystal
- Remember "**CV^2F**" power fmla. for CMOS!

- A CMOS circuit operates with V_{DD} = 3.3V. What is the power savings if V_{DD} is reduced to 2.5V?

4.3 PLL

- LM4F120 has 16 MHz crystal
- Remember "**CV^2F**" power fmla. for CMOS!

- A CMOS circuit operates with f = 40 MHz. What is the power savings if:
  - f is reduced to 20 MHz?
  - f is increased to 80 MHz?

PIOSC Specifications

Table 22-15. PIOSC Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
F_{PIOS}[a]	Internal 16-MHz precision oscillator frequency variance (factory calibrated at 25 °C C and 3.3 V) across the specified voltage and temperature range	-	-	±3%	-

a. This parameter value remains valid if recalibration occurs.

Low-Frequency Internal Oscillator (LFIOSC) Specifications

Table 22-16. Low-Frequency internal Oscillator Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
F_{LFIOSC}	Low-frequency internal oscillator (LFIOSC) frequency	10	33	90	KHz

Hibernation Clock Source Specifications

Table 22-17. Hibernation Oscillator Input Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
$F_{HIBLFIOSC}$	Hibernation low frequency internal oscillator (HIB LFIOSC) frequency	10	33	90	KHz
C_1, C_2	External load capacitance on XOSC0, XOSC1 pins[a]	12	-	24	pF

What is the power savings when LM4F120 hibernates?

Source: Tiva™ TM4C1233H6PM Microcontroller (identical to LM4F120H5QR) DATA SHEET

- Multiple clock sources for microcontroller system clock
  - Precision Oscillator (PIOSC): On-chip resource providing a 16 MHz ±1% frequency at room temperature
    - 16 MHz ±3% across temperature
    - Can be recalibrated with 7-bit trim resolution
    - Software power down control for low power modes
  - Main Oscillator (MOSC): A frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the `OSC0` input pin, or an external crystal is connected across the `OSC0` input and `OSC1` output pins.
    - External crystal used with or without on-chip PLL: select supported frequencies from 4 MHz to 25 MHz.
    - External oscillator: from DC to maximum device speed
  - Low Frequency Internal Oscillator (LFIOSC): On-chip resource used during power-saving modes
  - Hibernate RTC oscillator clock that can be configured to be the 32.768-kHz external oscillator source from the Hibernation (HIB) module or the HIB Low Frequency clock source (HIB LFIOSC), which is located within the Hibernation Module.

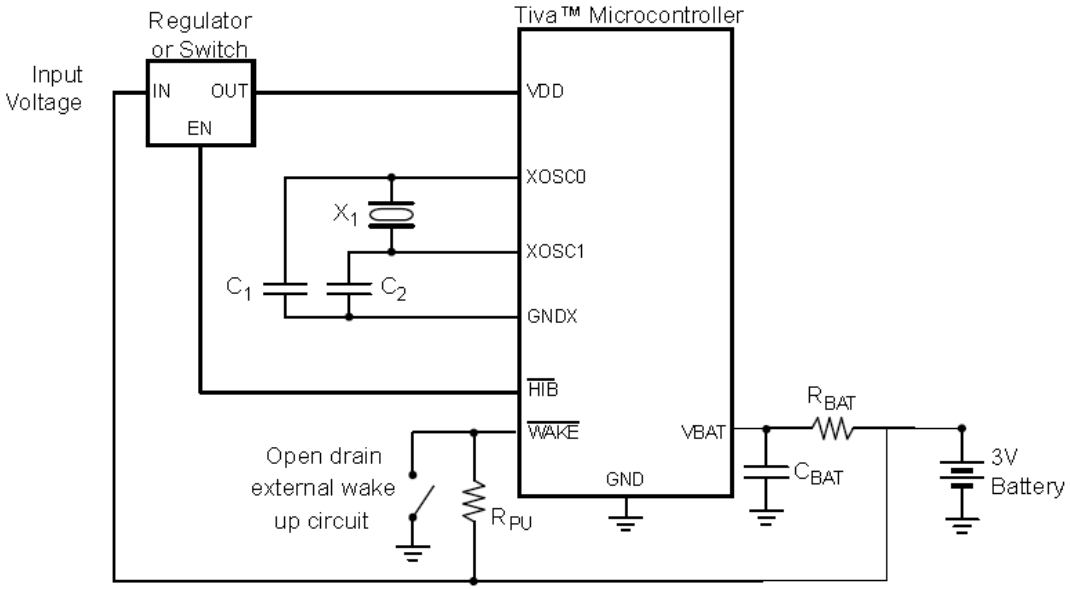
Real Time Clock (see next slide)

Source: Tiva™ TM4C1233H6PM Microcontroller (identical to LM4F120H5QR) DATA SHEET, p.52

FYI

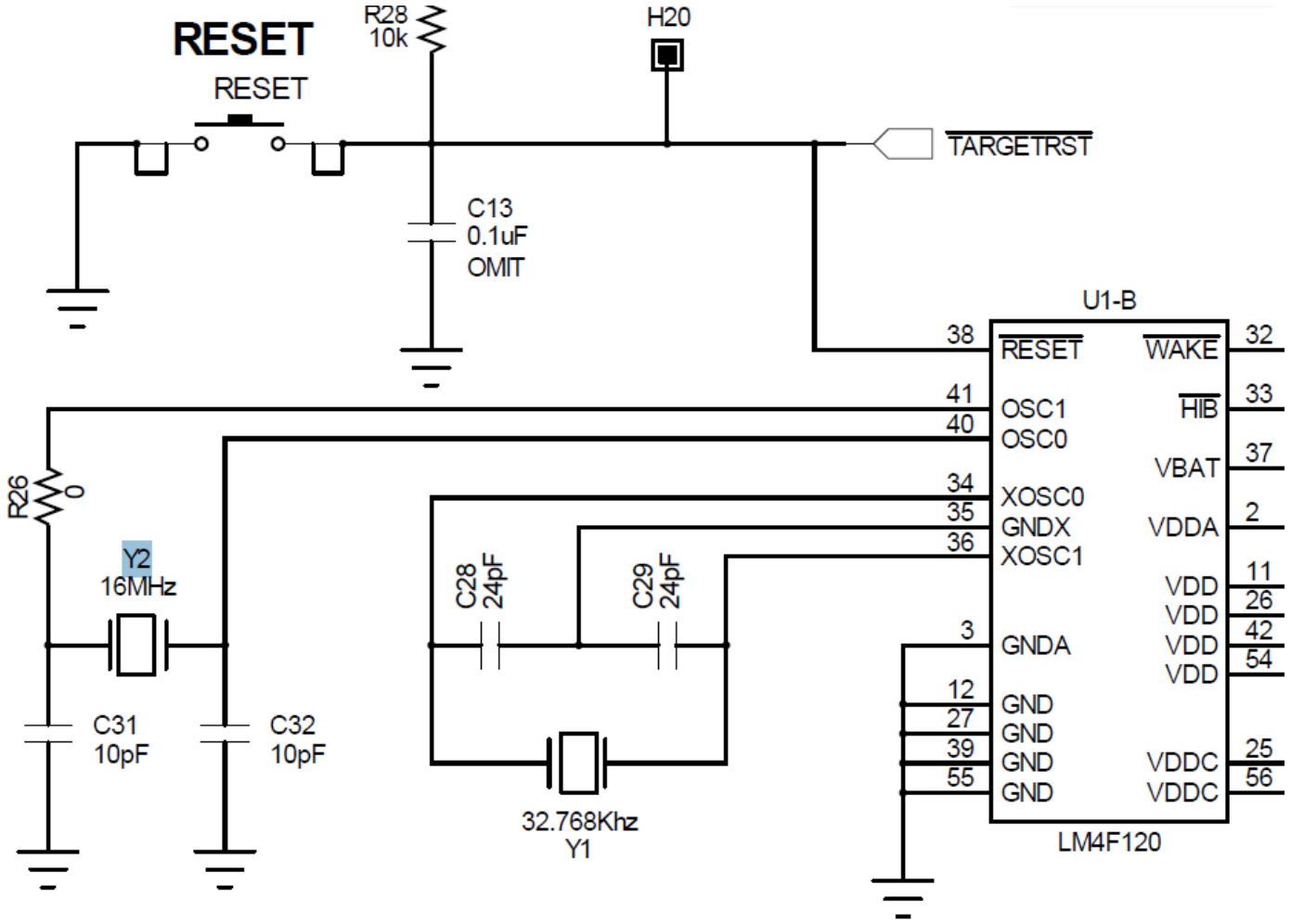
7.3.2 Hibernation Clock Source

In systems where the Hibernation module is used, the module must be clocked by an external source that is independent from the main system clock, even if the RTC feature is not used. An external oscillator or crystal is used for this purpose. To use a crystal, a 32.768-kHz crystal is connected to the XOSC0 and XOSC1 pins. Alternatively, a 32.768-kHz oscillator can be connected to the XOSC0 pin, leaving XOSC1 unconnected. Care must be taken that the voltage amplitude of the 32.768-kHz



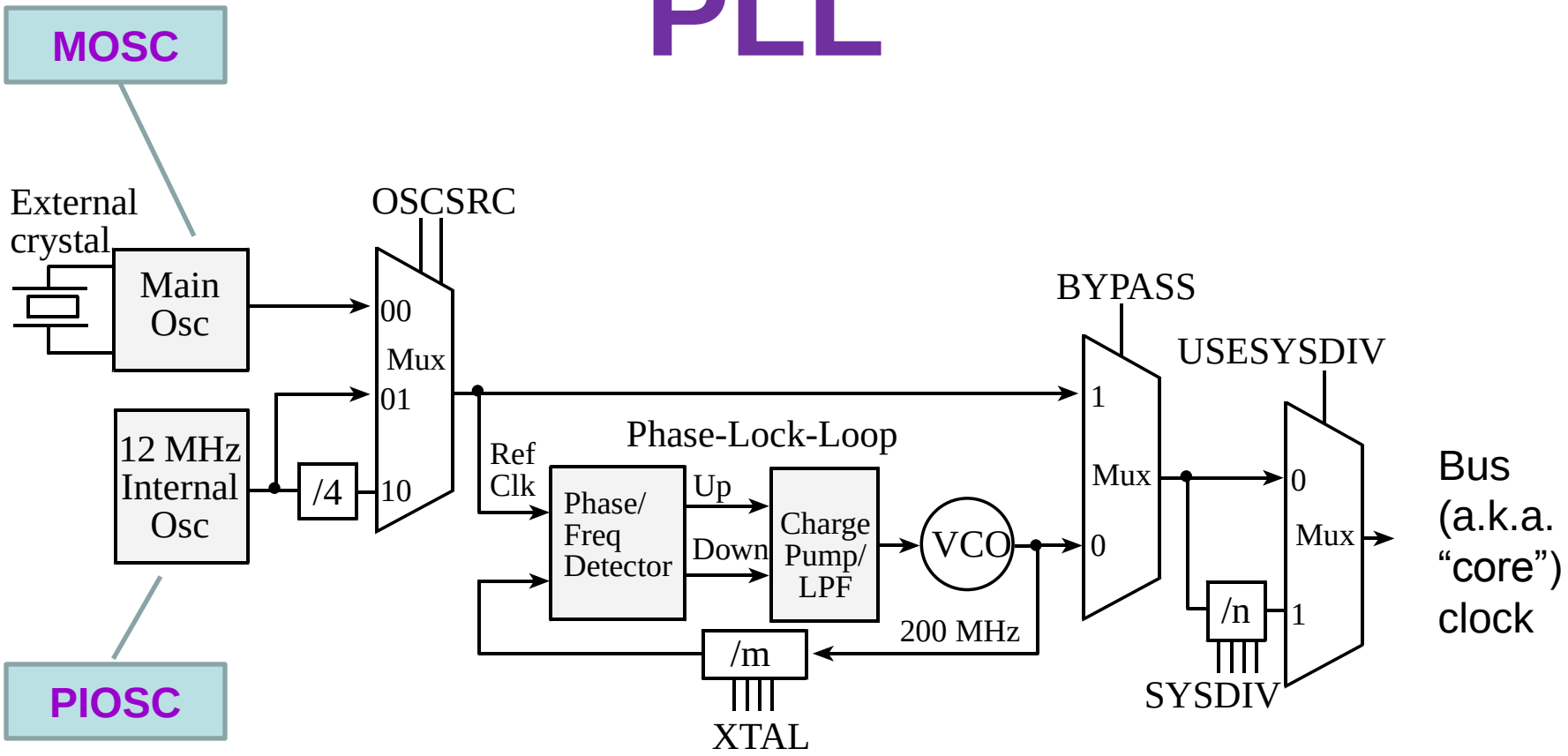
Source: Tiva™ TM4C1233H6PM Microcontroller (identical to LM4F120H5QR) DATA SHEET

FYI: Our board comes with both crystals: MOSC (Y2) and hibernation module (Y1)



Source: Stellaris® LM4F120 LaunchPad Evaluation Board User Manual

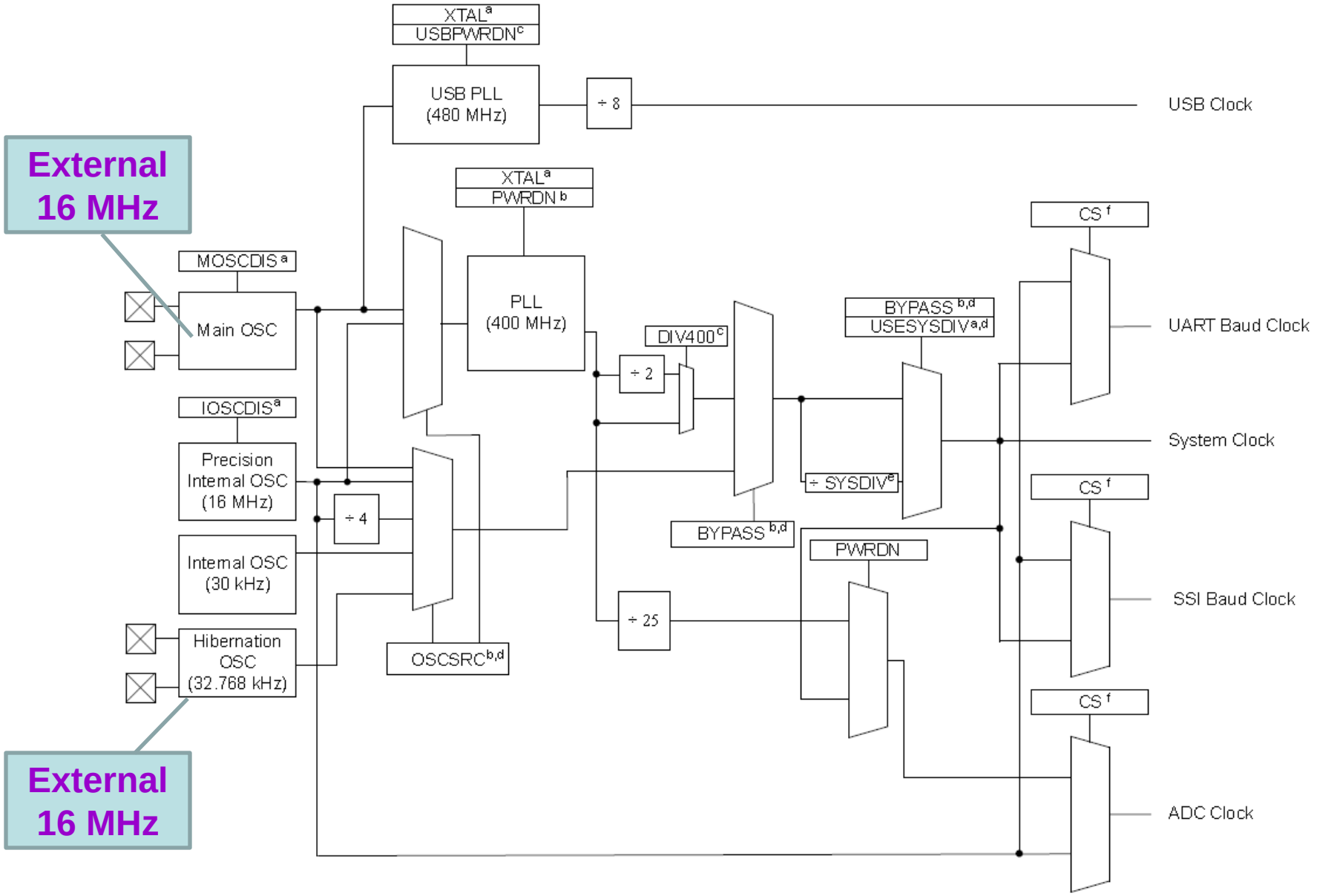
PLL



Internal oscillator requires minimal power but is imprecise
External crystal provides stable bus clock
LM4F120 is equipped with a 16 MHz crystal, and the bus clock is typically set at 50 MHz

FYI

Figure 5-5. Main Clock Tree



Source: Tiva™ TM4C1233H6PM Microcontroller (identical to LM4F120H5QR) DATA SHEET

Table 5-4. Possible System Clock Frequencies Using the SYSDIV Field

SYSDIV	Divisor	Frequency (BYPASS=0)	Frequency (BYPASS=1)	TivaWare™ Parameter[a]
0x0	/1	reserved	Clock source frequency/1	SYSCTL_SYSDIV_1
0x1	/2	reserved	Clock source frequency/2	SYSCTL_SYSDIV_2
0x2	/3	66.67 MHz	Clock source frequency/3	SYSCTL_SYSDIV_3
0x3	/4	50 MHz	Clock source frequency/4	SYSCTL_SYSDIV_4
0x4	/5	40 MHz	Clock source frequency/5	SYSCTL_SYSDIV_5
0x5	/6	33.33 MHz	Clock source frequency/6	SYSCTL_SYSDIV_6
0x6	/7	28.57 MHz	Clock source frequency/7	SYSCTL_SYSDIV_7
0x7	/8	25 MHz	Clock source frequency/8	SYSCTL_SYSDIV_8
0x8	/9	22.22 MHz	Clock source frequency/9	SYSCTL_SYSDIV_9
0x9	/10	20 MHz	Clock source frequency/10	SYSCTL_SYSDIV_10
0xA	/11	18.18 MHz	Clock source frequency/11	SYSCTL_SYSDIV_11
0xB	/12	16.67 MHz	Clock source frequency/12	SYSCTL_SYSDIV_12
0xC	/13	15.38 MHz	Clock source frequency/13	SYSCTL_SYSDIV_13
0xD	/14	14.29 MHz	Clock source frequency/14	SYSCTL_SYSDIV_14
0xE	/15	13.33 MHz	Clock source frequency/15	SYSCTL_SYSDIV_15
0xF	/16	12.5 MHz (default)	Clock source frequency/16	SYSCTL_SYSDIV_16

a. This parameter is used in functions such as SysCtlClockSet() in the TivaWare Peripheral Driver Library.

Source: Tiva™ TM4C1233H6PM Microcontroller (identical to LM4F120H5QR) DATA SHEET

The SYSDIV2 field in the **RCC2** register is 2 bits wider than the SYSDIV field in the **RCC** register so that additional larger divisors up to /64 are possible, allowing a lower system clock frequency for improved Deep Sleep power consumption. When using the PLL, the VCO frequency of 400 MHz is predivided by 2 before the divisor is applied. The divisor is equivalent to the SYSDIV2 encoding plus 1. Table 5-5 shows how the SYSDIV2 encoding affects the system clock frequency, depending on whether the PLL is used (BYPASS2=0) or another clock source is used (BYPASS2=1). For a list of possible clock sources, see Table 5-3 on page 211.

Table 5-5. Examples of Possible System Clock Frequencies Using the SYSDIV2 Field

SYSDIV2	Divisor	Frequency (BYPASS2=0)	Frequency (BYPASS2=1)	TivaWare Parameter[a]
0x00	/1	reserved	Clock source frequency/1	SYSCTL_SYSDIV_1
0x01	/2	reserved	Clock source frequency/2	SYSCTL_SYSDIV_2
0x02	/3	66.67 MHz	Clock source frequency/3	SYSCTL_SYSDIV_3
0x03	/4	50 MHz	Clock source frequency/4	SYSCTL_SYSDIV_4
0x04	/5	40 MHz	Clock source frequency/5	SYSCTL_SYSDIV_5
...	...	...	...	...
0x09	/10	20 MHz	Clock source frequency/10	SYSCTL_SYSDIV_10
...	...	...	...	...
0x3F	/64	3.125 MHz	Clock source frequency/64	SYSCTL_SYSDIV_64

a. This parameter is used in functions such as SysCtlClockSet() in the TivaWare Peripheral Driver Library.

Source: Tiva™ TM4C1233H6PM Microcontroller (identical to LM4F120H5QR) DATA SHEET

Your turn!

Using the 16 MHz CLK as the source, how should we set the multiplier to obtain:

- A 50 MHz clock?
- A reduction in power by 75%?

QUIZ: Bit-specific addressing

4.8 Fill in the following table with the bit-specific assembly definitions (the first one is done)

Port	Bits	Definition
A	5	`PA5 EQU 0x40004080`
B	1,0	
F	7,5	
G	3,2,0	

4.9 Fill in the following table with the bit-specific C definitions (the first one is done)

Port	Bits	Definition
A	5	`#define PA5 (*((volatile unsigned long *)0x40004080))`
C	3,1	
D	7,6,5	
F	7,0	

Designing a complete Hw-Sw system:

Example 4.2: The goal is to develop a means for the microcontroller to turn on and to turn off an AC-powered appliance. The interface will use a solid state relay (SSR) having a control portion equivalent to an LED with parameters of 2V and 10 mA. Include appropriate functions.

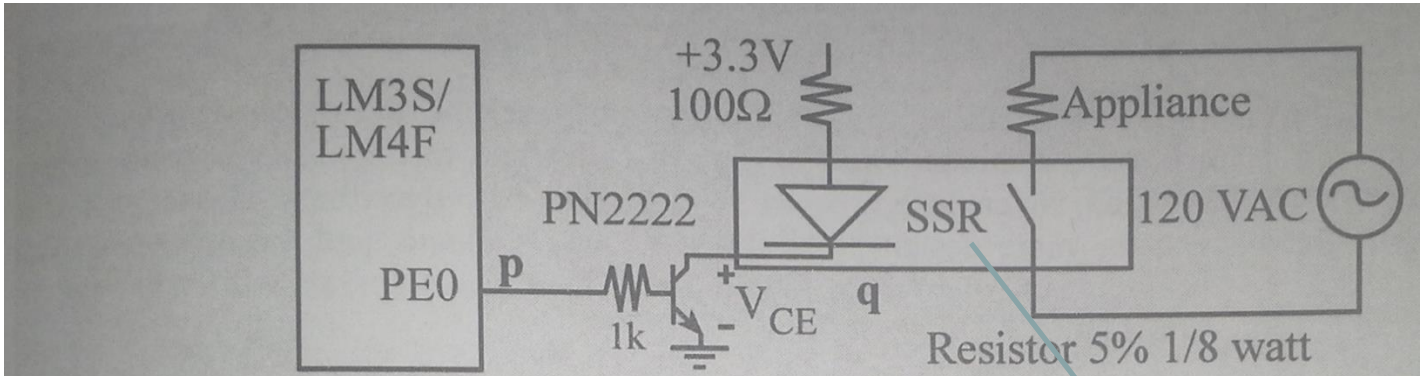


Figure 4.15. Solid state relay interface using a PN2222 NPN transistor.

SSR = Solid-State Relay

PN2222 specs:
- max. ICE = 150 mA
- VCE = 0.3V

What is there to design?

Designing a complete Hw-Sw system:

Example 4.2: The goal is to develop a means for the microcontroller to turn on and to turn off an AC-powered appliance. The interface will use a solid state relay (SSR) having a control portion equivalent to an LED with parameters of 2V and 10 mA. Include appropriate functions.

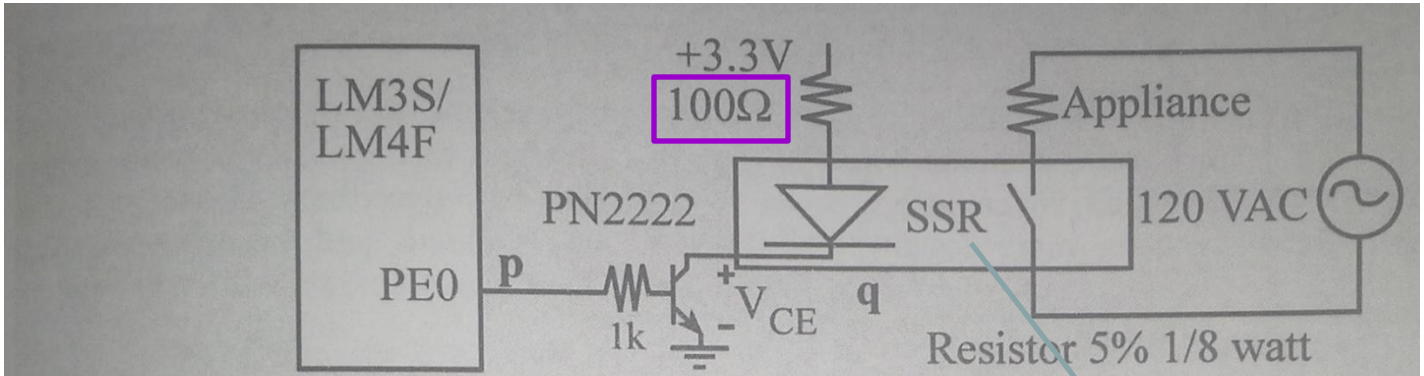


Figure 4.15. Solid state relay interface using a PN2222 NPN transistor.

SSR = Solid-State Relay

PN2222 specs:
- max. ICE = 150 mA
- VCE = 0.3V

Designing a complete Hw-Sw system:

Example 4.2: The goal is to develop a means for the microcontroller to turn on and to turn off an AC-powered appliance. The interface will use a solid state relay (SSR) having a control portion equivalent to an LED with parameters of 2V and 10 mA. Include appropriate functions.

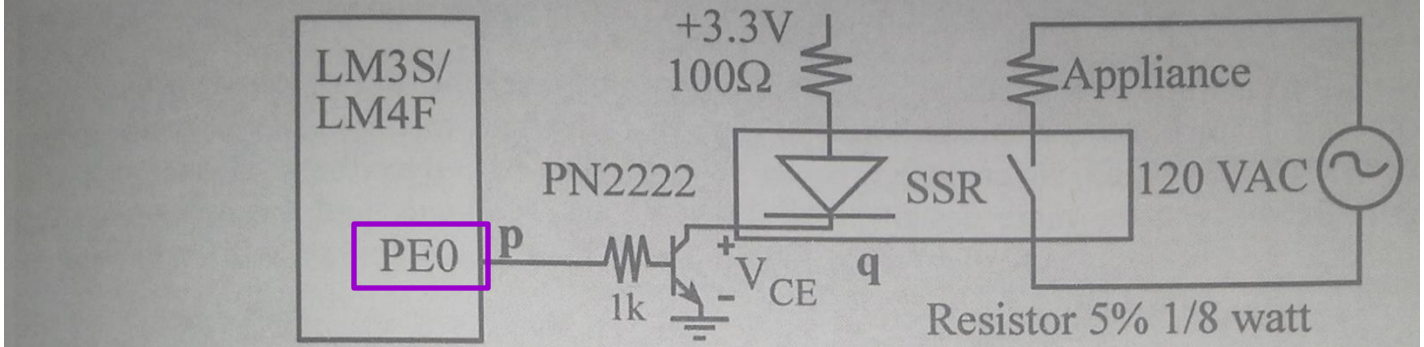


Figure 4.15. Solid state relay interface using a PN2222 NPN transistor.

QUIZ: Bit-specific addressing

```
#define P[redacted]  (*((volatile unsigned long *)0x40024004))
void LED_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= [redacted];
  delay = SYSCTL_RCGC2_R;
  GPIO_PORTE_DIR_R |= [redacted]
  GPIO_PORTE_AFSEL_R &= [redacted]
  GPIO_PORTE_DEN_R |= [redacted]
}
```

Fill in the missing parts of the”ritual”

```
#define PE0    (*((volatile unsigned long *)0x40024004))
void LED_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000010;  // activate clock for Port E
  delay = SYSCTL_RCGC2_R;        // allow time for clock to start
  GPIO_PORTE_DIR_R |= 0x01;      // PE0 output
  GPIO_PORTE_AFSEL_R &= ~0x01;   // PE0 regular port function
  GPIO_PORTE_DEN_R |= 0x01;      // enable PE0 digital port
}
```

```
void LED_Toggle(void){
  PE0 = PE0^1; // toggle LED
}
```

Write C functions that turn the SSR on and off, using toggle as template!

4.4 SysTick Timer

Timer/Counter operation

- 24-bit counter *decrements* at bus clock frequency
  - With 50 MHz bus clock, decrements occur every 20 ns
- Counting is from n → 0
  - For a count of m, (m-1) is loaded into the counter

SysTick Timer

Initialization

- Clear ENABLE to stop counter
- Specify the RELOAD value
- Clear the counter via NVIC_ST_CURRENT_R
- Set CLK-SRC=1 and specify interrupt action via INTEN in NVIC_ST_CTRL_R

Address	31-24	23-17	16	15-3	2	1	0	Name
$E000E010	0	0	COUNT	0	CLK_SRC	INTEN	ENABLE	NVIC_ST_CTRL_R
$E000E014	0	24-bit RELOAD value						NVIC_ST_RELOAD_R
$E000E018	0	24-bit CURRENT value of SysTick counter						NVIC_ST_CURRENT_R

```
#define NVIC_ST_CTRL_R          (*((volatile unsigned long *)0xE000E010))
#define NVIC_ST_RELOAD_R        (*((volatile unsigned long *)0xE000E014))
#define NVIC_ST_CURRENT_R       (*((volatile unsigned long *)0xE000E018))
void SysTick_Init(void){
  NVIC_ST_CTRL_R = 0;                   // 1) disable SysTick during setup
  NVIC_ST_RELOAD_R = 0x00FFFFFF;        // 2) maximum reload value
  NVIC_ST_CURRENT_R = 0;                // 3) any write to current clears it
  NVIC_ST_CTRL_R = 0x00000005;          // 4) enable SysTick with core clock
}
```

Address	31-24	23-17	16	15-3	2	1	0	Name
$E000E010	0	0	COUNT	0	CLK_SRC	INTEN	ENABLE	NVIC_ST_CTRL_R
$E000E014	0	24-bit RELOAD value						NVIC_ST_RELOAD_R
$E000E018	0	24-bit CURRENT value of SysTick counter						NVIC_ST_CURRENT_R

```
// The delay parameter is in units of the 6 MHz core clock. (167 ns)
void SysTick_Wait(unsigned long delay){
  volatile unsigned long elapsedTime;
  unsigned long startTime = NVIC_ST_CURRENT_R;
  do{
    elapsedTime = (startTime-NVIC_ST_CURRENT_R)&0x00FFFFFF;
  }
  while(elapsedTime <= delay);
}
// 10000us equals 10ms
void SysTick_Wait10ms(unsigned long delay){
  unsigned long i;
  for(i=0; i<delay; i++){
    SysTick_Wait(60000);  // wait 10ms
  }
}
```

Why do we need this?
A: To account for possible counter rollover!

Address	31-24	23-17	16	15-3	2	1	0	Name
$E000E010	0	0	COUNT	0	CLK_SRC	INTEN	ENABLE	NVIC_ST_CTRL_R
$E000E014	0	24-bit RELOAD value						NVIC_ST_RELOAD_R
$E000E018	0	24-bit CURRENT value of SysTick counter						NVIC_ST_CURRENT_R

SKIP 4.5 OLED

READ 4.6 Debugging monitor using LED

4.6 Debugging monitor using LED

```
#define PG2    (*((volatile unsigned long *)0x40026010))
```

```
PG2 EQU 0x40026010
Toggle
  LDR R1, =PG2
  LDR R0, [R1]          ; previous
  EOR R0, R0, #0x04     ; toggle PG2
  STR R0, [R1]          ; output
  BX  LR
```

```
void Toggle(void){
  PG2 ^= 0x04;
}
```

How many bus cycles does this code take to execute?

4.7 Performance Debugging

Using SysTick to time a segment of code

```
void main(void){ unsigned long before,elapsed,ss,tt;
  SysTick_Init();      // initialize, Program 4.7
  ss = 230400;
  before = NVIC_ST_CURRENT_R;
  tt = sqrt(ss);
  elapsed = (before-NVIC_ST_CURRENT_R)&0x00FFFFFF;
  while(1){};
}
```

Homework for Ch.4

- End of chapter **6, 7, 10, 12, 13, 17**
- Due Thu, Nov.7