TIVA LaunchPad Clock Sources

MOSC, PIOSC, LFIOSC, Hibernation Clock

Fundamental Clock Sources

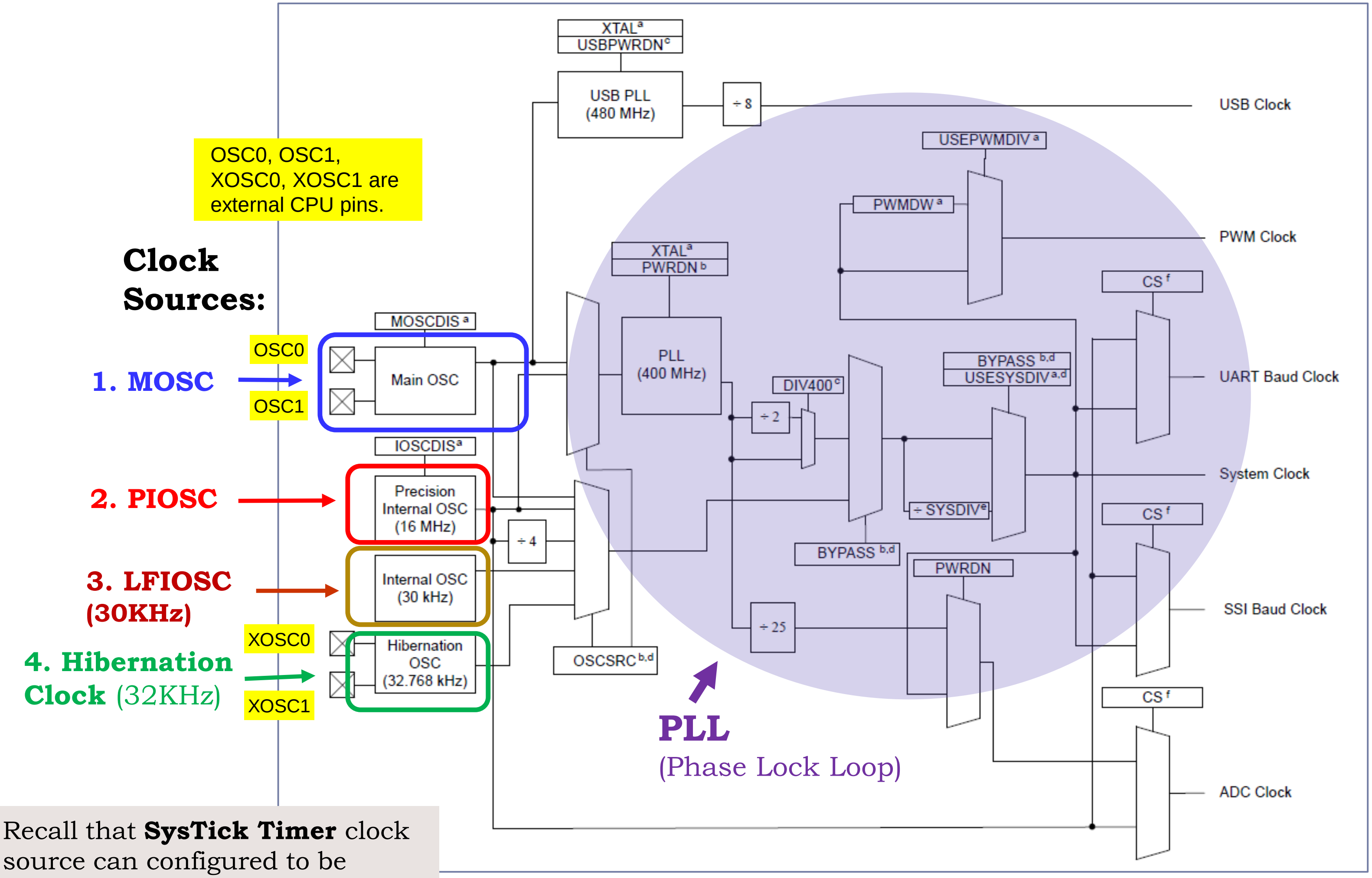
The Tiva LaunchPad ARM processor can derive its clock from various clock sources:

1. Precision Internal Oscillator (**PIOSC**)
   - On-chip clock @ 16MHz +/- 3%.
   - Low-cost clock source; does not require external components.
2. Main Oscillator (**MOSC**) using either,
   - External crystal connects to OSC0 (input) & OSC1 (output) pins.
   - Crystal must be between 5 MHz to 25 MHz (inclusive).
   - An external single-ended clock source to OSC0 in.
3. Internal **30 kHz** Oscillator **(LFIOSC)**
   - for use during Deep-Sleep power-saving modes.
4. Hibernation Module Clock Source (**32.768 kHz**)
   - Clocked by 32.768 kHz oscillator.
   - Clock can be used as system clock during Hibernation mode.
   - Provides Real-Time Clock (RTC) source.
   - Crystal connects to XOSC0 (input) & XOSC1 (output) pins.

PIOSC or MOSC can be used to drive the Phase Lock Loop (PLL).

Source: Tiva C Series TM4C123G LaunchPad Evaluation Board – User Guide (spmu296.pdf, p260)

Processor Clock Tree



OSC0, OSC1, XOSC0, XOSC1 are external CPU pins.

Clock Sources:

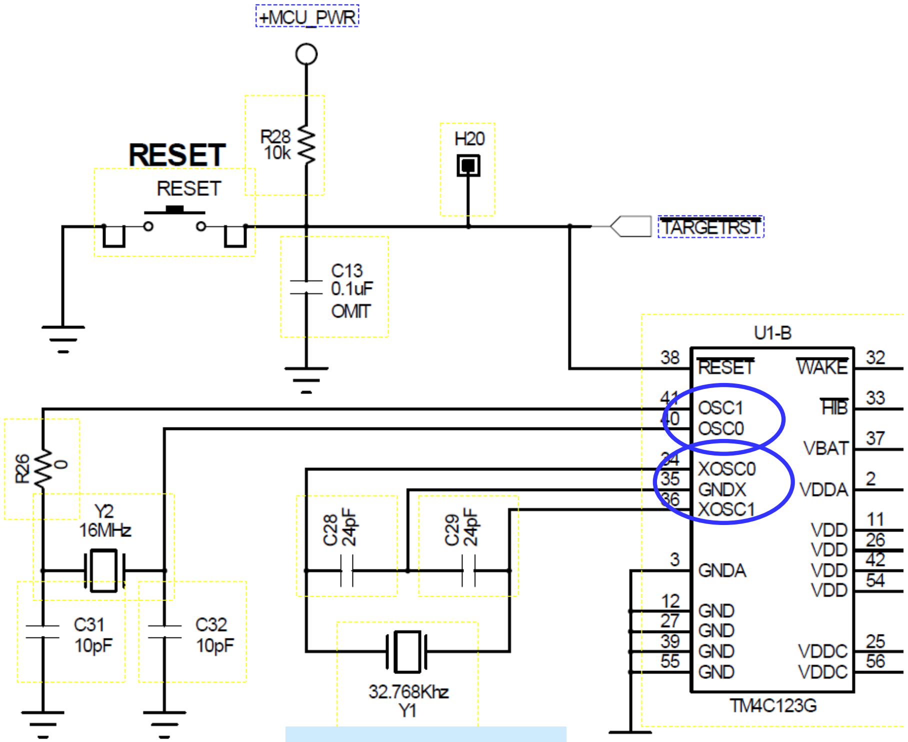
1. **MOSC** (OSC0, OSC1)
2. **PIOSC**
3. **LFIOSC (30KHz)**
4. **Hibernation Clock** (32KHz) (XOSC0, XOSC1)

PLL
(Phase Lock Loop)

Recall that **SysTick Timer** clock source can configured to be either **system clock** or **PIOSC/4**.

Source: Tiva C Series TM4C123GH6PM7 Data Sheet (spmu376e.pdf, p222)

Tiva LaunchPad Clock Schematics

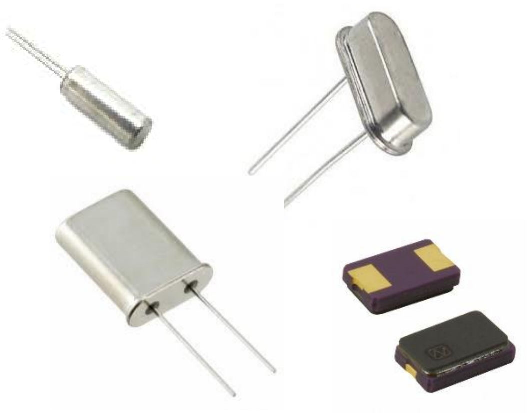


MOSC (16MHz)

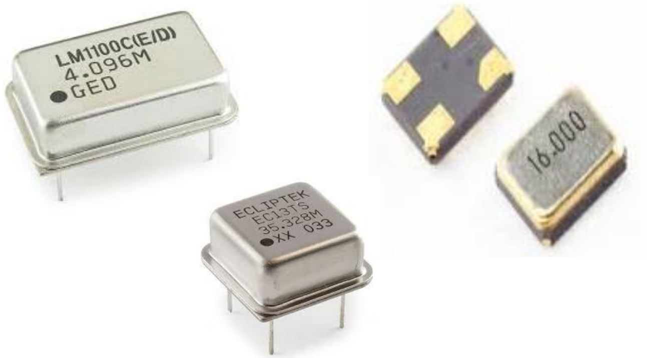
32.768 kHz

Source: Tiva LaunchPad TM4C123G User Manual (spmu296.pdf, p21)

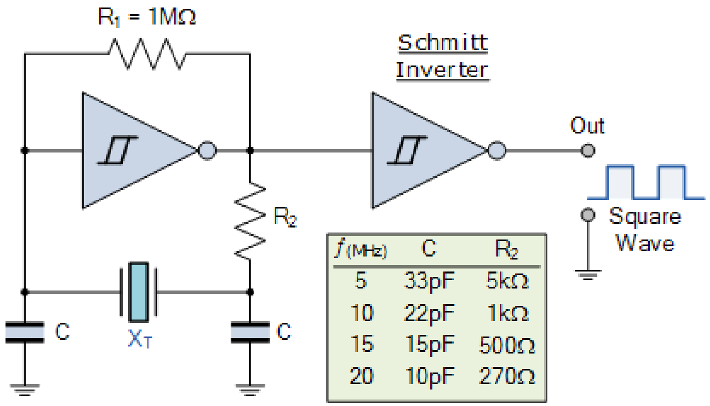
Fundamental Clock Sources



Crystals



Crystal Oscillators



f (MHz)	C	R$_2$
5	33pF	5kΩ
10	22pF	1kΩ
15	15pF	500Ω
20	10pF	270Ω

- A crystal oscillator output a TTL-compatible clock.
- A crystal requires an oscillator circuit in order to output a clock source. See an example of oscillator circuit at left.
- In short, (crystal + oscillator circuit) = oscillator.

Phase Lock Loop (PLL)

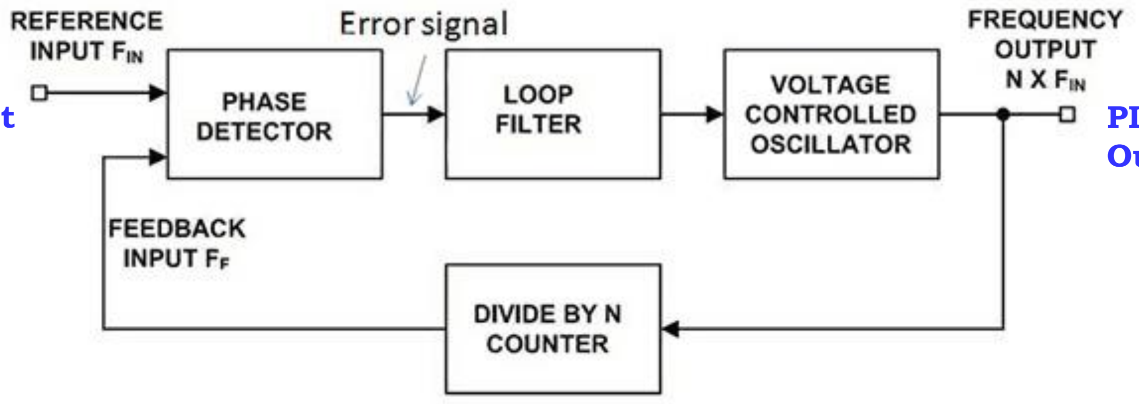
Feedback System to produce Stable High Frequency Clock from Low Frequency Clock

PLL Fundamentals

A **Phase Locked Loop** (PLL) is a feedback system that produces a stable **high frequency clock** from a **low frequency clock** input. It includes a VCO, phase detector & low-pass filter within its feedback loop.

The PLL forces the VCO to replicate & track the phase (and therefore frequency), thus producing a stable clock output. When this is achieved, we say the PLL is locked.

Reference Clock Input *(from clock oscillator)*



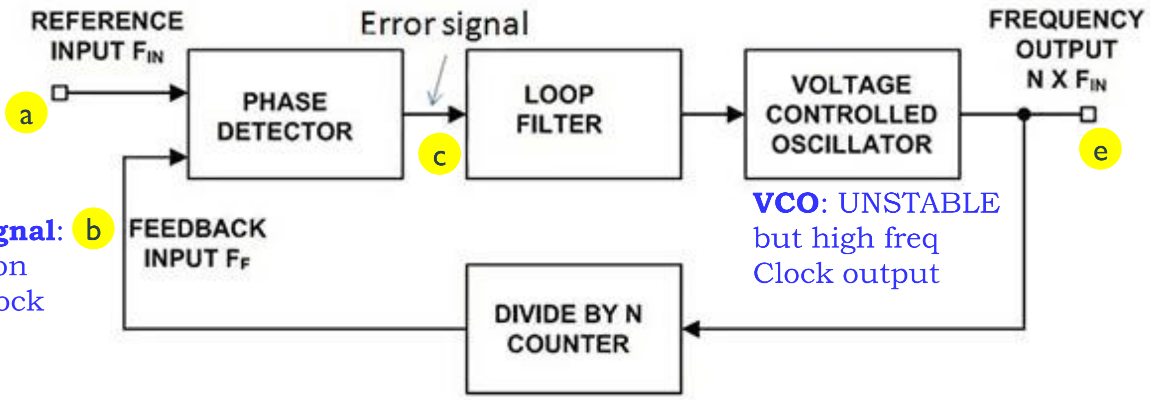
PLL Clock Output

Basic Blocks of PLL:

- **Reference Clock:** Input clock to PLL.
- **Phase Detector:** compares feedback clock to reference clock. Produces an Error signal proportional to phase difference.
- **Loop Filter:** LPF. To keep only the fundamental frequencies to compare and generate the Error signal.
- **VCO:** produces PLL output clock which is controlled by the Error signal.

How a PLL Works ….

Clock Input: STABLE, low frequency.



PLL Output = System Clock

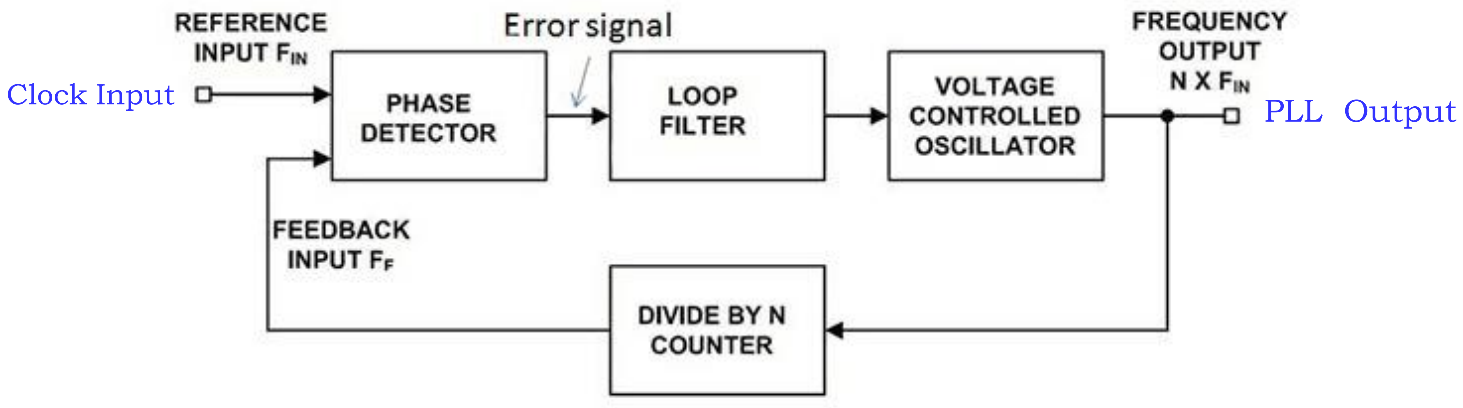
Feedback signal: for comparison with input clock

VCO: UNSTABLE but high freq Clock output

Basic Operation of PLL:

- PLL output signal (e) is always a multiple of the input clock.
  - Signals at (a) & (b) must always have same phase difference (locked).
- The phase difference produces an error signal (c).
- VCO produces an output clock that is proportional to (c).
  - If (e) deviates, it will cause a phase difference between (a) & (b), which in turn causes a change in (c) & VCO output will then be adjusted to correct the phase difference between (a) & (b).

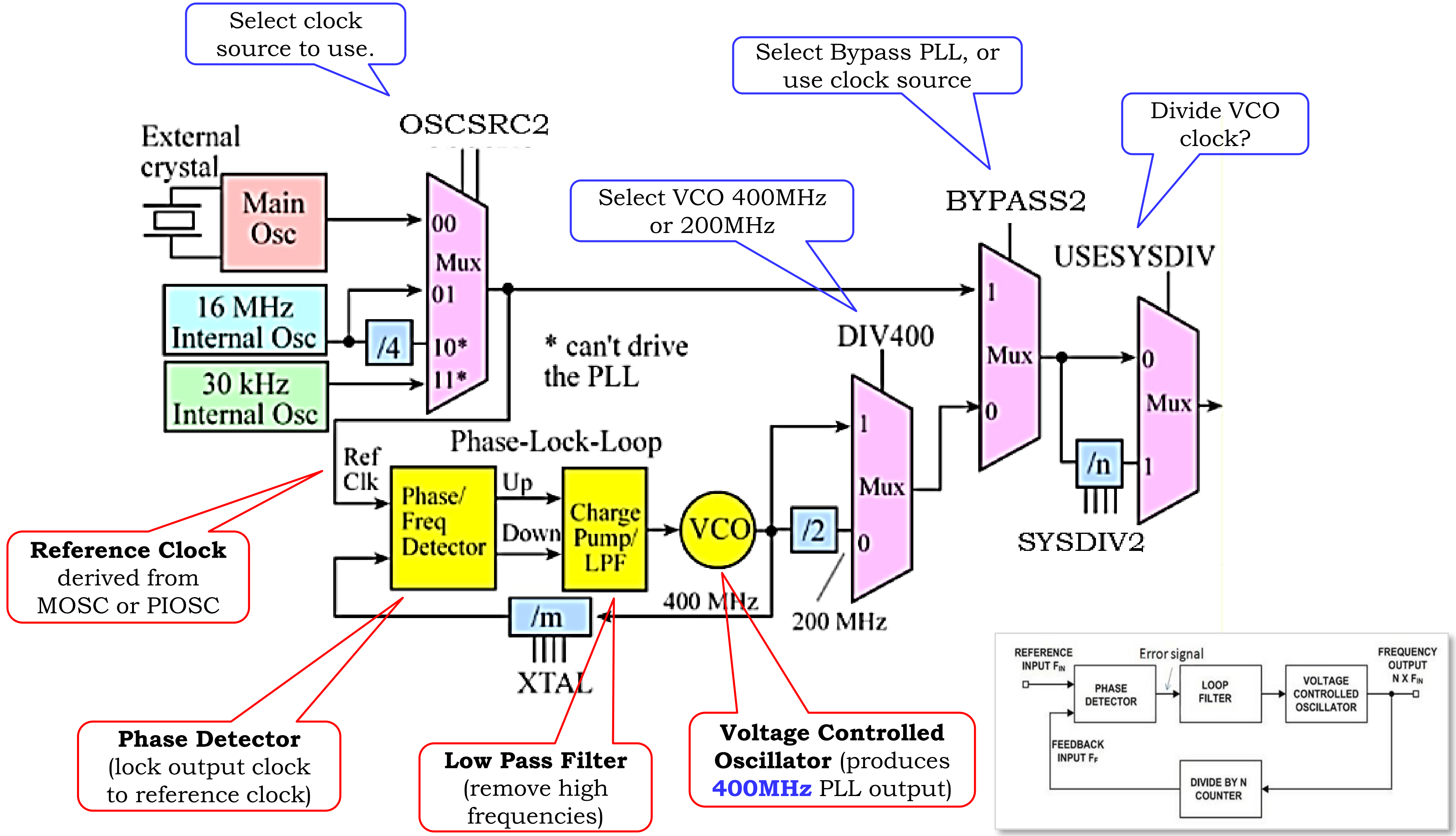
Why use PLL?



Why do we need a PLL?

- Crystal oscillator circuits can provide low frequency, very stable clocks (e.g 16MHz).
- Processor needs a high frequency clock (e.g. 80MHz or higher).
  - However VCO clock circuit output is normally not sufficiently stable enough for use as CPU clock.
  - Through feedback loop control, PLL circuit will force VCO output to track a reference stable clock to produce a stable clock output.

LaunchPad PLL Block Diagram



Select clock source to use.

Select Bypass PLL, or use clock source

Divide VCO clock?

Select VCO 400MHz or 200MHz

Reference Clock derived from MOSC or PIOSC

Phase Detector (lock output clock to reference clock)

Low Pass Filter (remove high frequencies)

Voltage Controlled Oscillator (produces **400MHz** PLL output)

PLL Setup for LaunchPad

- The Tiva LaunchPad PLL is configured through direct register writes to the Run-Mode Clock Configuration (**RCC** & **RCC2**) registers.
- Other registers involved are:
  - **MISC** (Mask Interrupt Status & Clear)
  - **RIS** (Raw Interrupt Status)
  - **PLLSTAT** (PLL Status).
- All the above are part of a set of System Control Registers (SYSCTRL).

(see next slide & also TM4C123GH6PM Datasheet, pg 232).

Registers Needed for PLL Setup

Register	Function
RCC	Run Mode Clock Configuration register. Legacy register for compatibility with older version Tiva chipset. Some functions in RCC are repeated in RCC2 register. Used to set external crystal frequency, select main oscillator.
RCC2	Run Mode Clock Configuration register 2. Used to set PLL clock divisor, enable PLL, …
RIS	Raw Interrupt Status register. Used to read interrupt status.
MISC	Masked Interrupt Status & Clear register. Used to read interrupt status and clear interrupts.
PLLSTAT	PLL Status register. Only 1 bit of register is used. Used to check if PLL has been locked to set frequency.

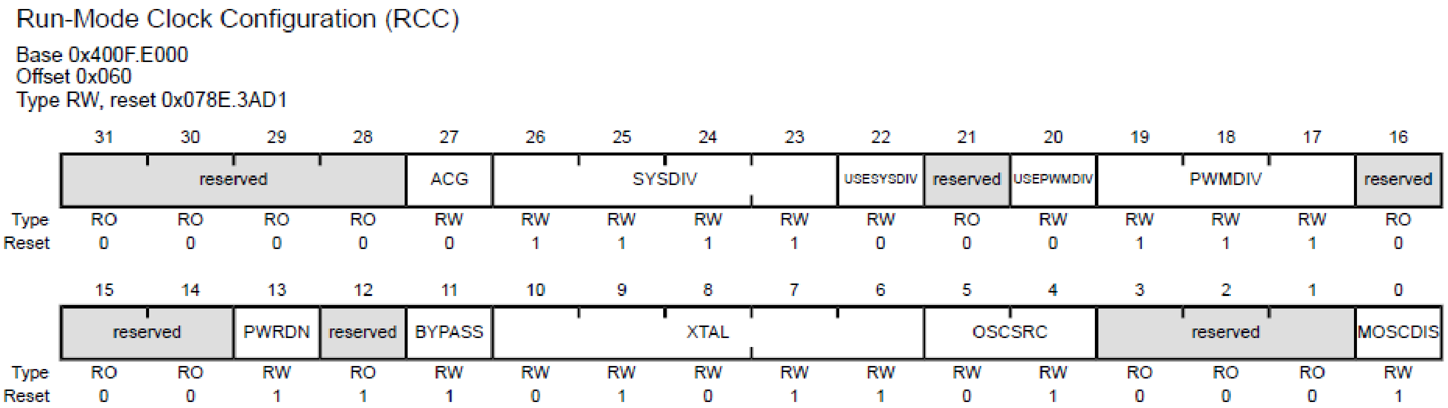
RCC & RCC2 Registers

RCC & RCC2 for PLL Configuration

RCC & RCC2 Registers

- Both **RCC** & **RCC2** registers are used to configure the PLL.
- RCC2 extends some fields of the RCC register, like providing more clock configurations.
- If RCC2 is used, USERCC2 bit (found in RCC2 register, bit 31) must be set.
- During initial power-on, the PLL is NOT used (means, PLL is bypassed) & processor clock is taken directly from the clock source (e.g MOSC).
    - In the case of the Tiva launchPad, MOSC is 16 MHz since a 16 MHz external crystal is used.
- When the PLL has been initialized and **locked**, software can then switch the processor clock to use the PLL output.

Run-Mode Clock Configuration (RCC Register)



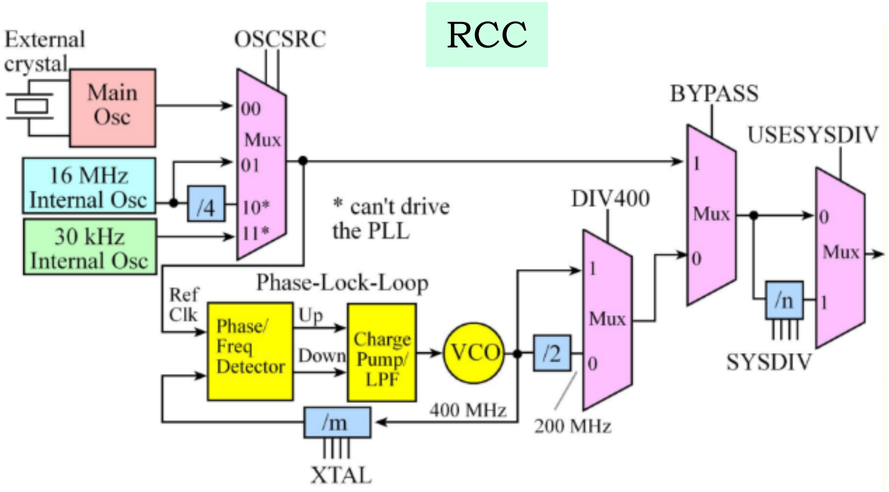
Run-Mode Clock Configuration (RCC)

Base 0x400F.E000
Offset 0x060
Type RW, reset 0x078E.3AD1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved				ACG	SYSDIV				USESYSDIV	reserved	USEPWMDIV	PWMDIV			reserved
Type	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RO	RW	RW	RW	RW	RO
Reset	0	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		PWRDN	reserved	BYPASS	XTAL					OSCSRC		reserved			MOSCDIS
Type	RO	RO	RW	RO	RW	RW	RW	RW	RW	RW	RW	RW	RO	RO	RO	RW
Reset	0	0	1	1	1	0	1	0	1	1	0	1	0	0	0	1

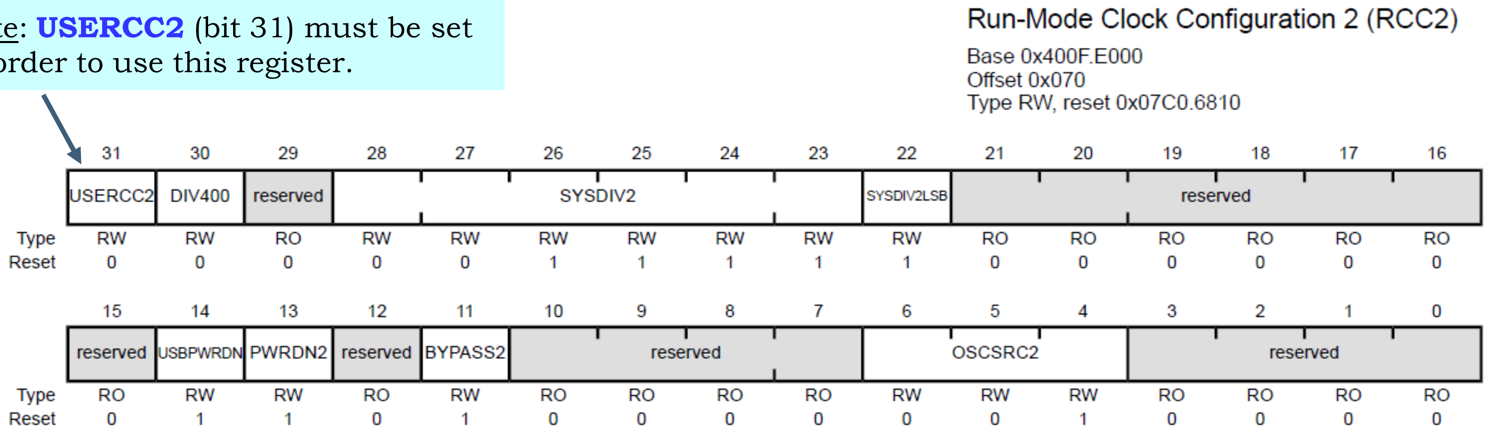
- **USESYSDIV** (bit 22)
  - ‘1’ = use system clock divider
  - ‘0’ = system clock is used undivided.
- **XTAL**: crystal value attached to the MOSC. 0x15 (16MHz).
- **OSCSRC**: oscillator source. 0x0 (MOSC); 0x01 (PIOSC-default); …
- **MOSCDIS**: ‘1’ = main oscillator disabled; ‘0’ = enabled.
- **BYPASS**: ‘0’= PLL is enabled.



Source: Tiva C Series TM4C123GH6PM7 Data Sheet (spmu376e.pdf, p254)

Run-Mode Clock Configuration (RCC2 Register)

Note: **USERCC2** (bit 31) must be set in order to use this register.



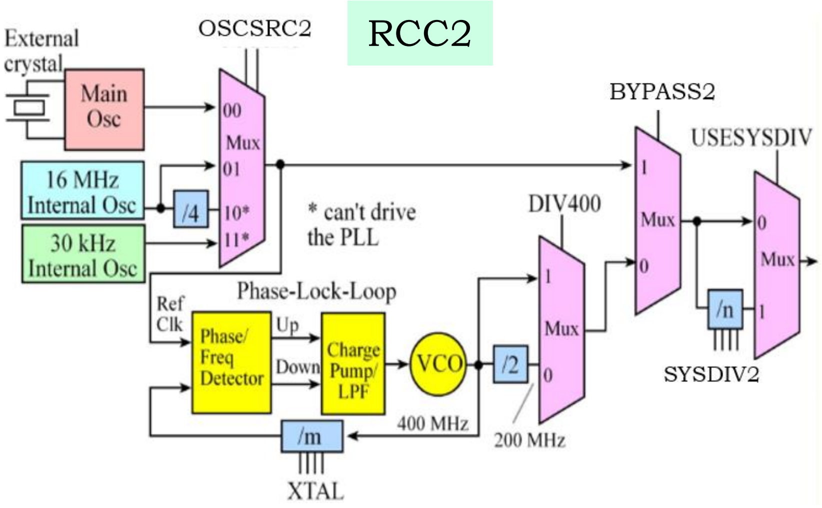
Run-Mode Clock Configuration 2 (RCC2)

Base 0x400F.E000
Offset 0x070
Type RW, reset 0x07C0.6810

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	USERCC2	DIV400	reserved	SYSDIV2						SYSDIV2LSB	reserved					
Type	RW	RW	RO	RW	RW	RW	RW	RW	RW	RW	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	USBPWRDN	PWRDN2	reserved	BYPASS2	reserved				OSCSRC2			reserved			
Type	RO	RW	RW	RO	RW	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO
Reset	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0

- **USERCC2:** ‘1’ = use RCC2 register.
- **OSCSRC2:** Oscillator Source 2. 0x0 (MOSC), 0x01 (PIOSC), ….
- **DIV400**: ‘1’ = SYSDIV2 is 7-bits; ‘0’= SYSDIV2 is 6-bits.
- **SYSDIV2LSB**: LSB for SYSDIV2 if DIV400 set.
- **BYPASS2**:
  - ‘0’ = system clock derived from PLL;
  - ‘1 = system clock derived from OSC source.
- **SYSDIV2**: VCO clock divisor.
  - *[see slide 25 to set System Clock Frequencies]*

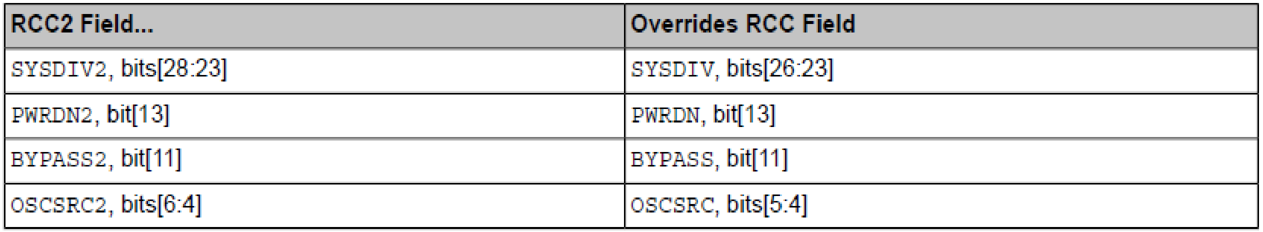


Source: Tiva C Series TM4C123GH6PM7 Data Sheet (spmu376e.pdf, p260)

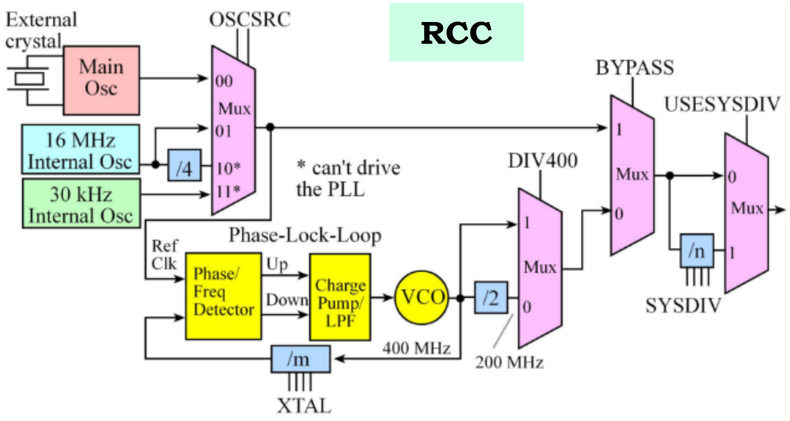
RCC2 versus RCC Register

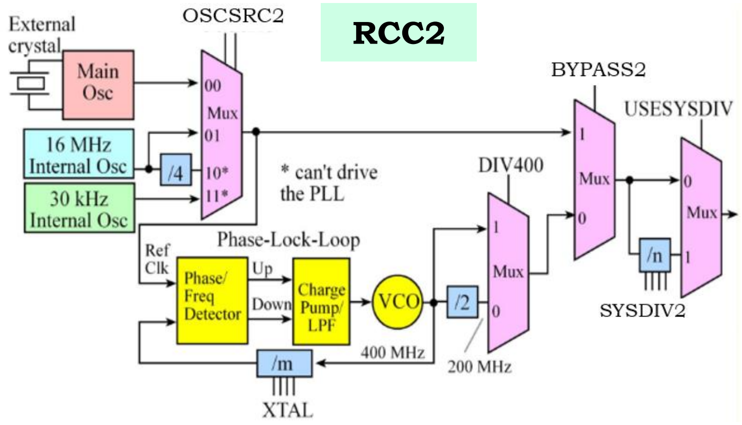
- RCC register is a legacy register (*meant for backward compatibility with older version chips*).
- Certain bit fields in RCC are repeated or extended in RCC2. Such fields in RCC2 overrides that in RCC (*see table below*).

RCC2 bits ends with a '2' suffix.



RCC2 Field...	Overrides RCC Field
SYSDIV2, bits[28:23]	SYSDIV, bits[26:23]
PWRDN2, bit[13]	PWRDN, bit[13]
BYPASS2, bit[11]	BYPASS, bit[11]
OSCSRC2, bits[6:4]	OSCSRC, bits[5:4]

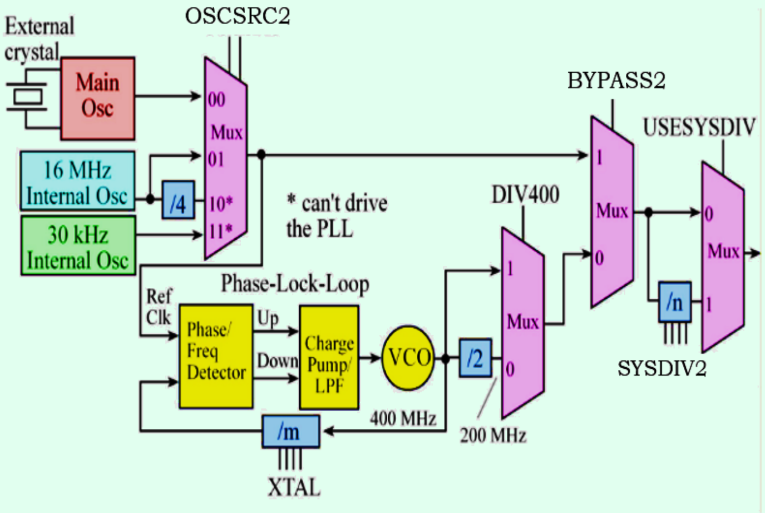




Important: When using the RCC & RCC2 registers, write to the RCC register first before writing to the RCC2 register.

Summary: Points to Note on Tiva PLL

1. PLL can be driven from the PIOSC or MOSC.
   - MOSC is an external clock source. Can be external crystal or oscillator.
   - PIOSC is a 16MHz internal clock source.
2. RCC and RCC2 registers are used to setup the PLL.
3. If BYPASS2=1 in RCC2 register, PLL is not used to generate CPU clock.
4. If USESYSDIV=1 in RCC register, clock divisor is used.
5. Input clock source is synthesized to a 400MHz internal PLL clock (through a VCO) which is used to generate the processor clock.
6. VCO output freq = 400 MHz.
7. If DIV400 = ‘0’, the VCO output freq is halved (= 200 MHz) before it is used.
8. VCO output is divided down to obtain the CPU clock.



PLL Setup & Initialization

RCC, RCC2, RIS, MISC, PLLSTAT

PLL Initialization

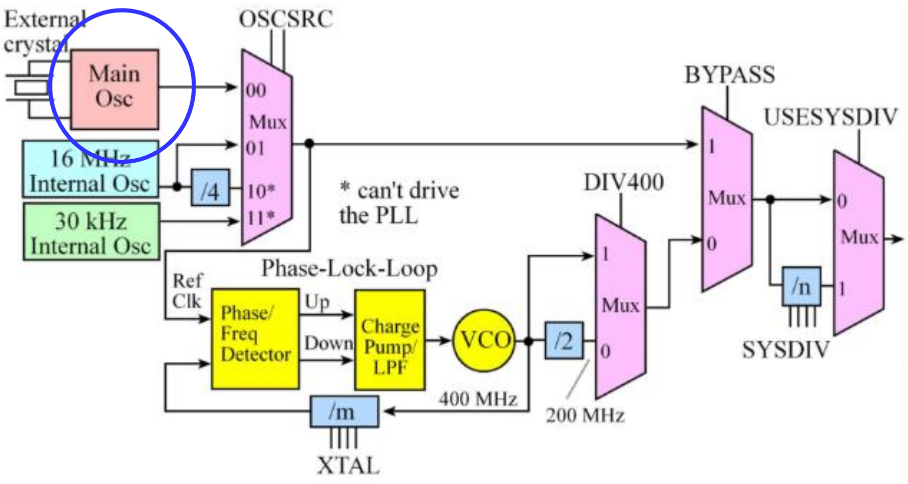
General Steps to setup PLL in Tiva LaunchPad:

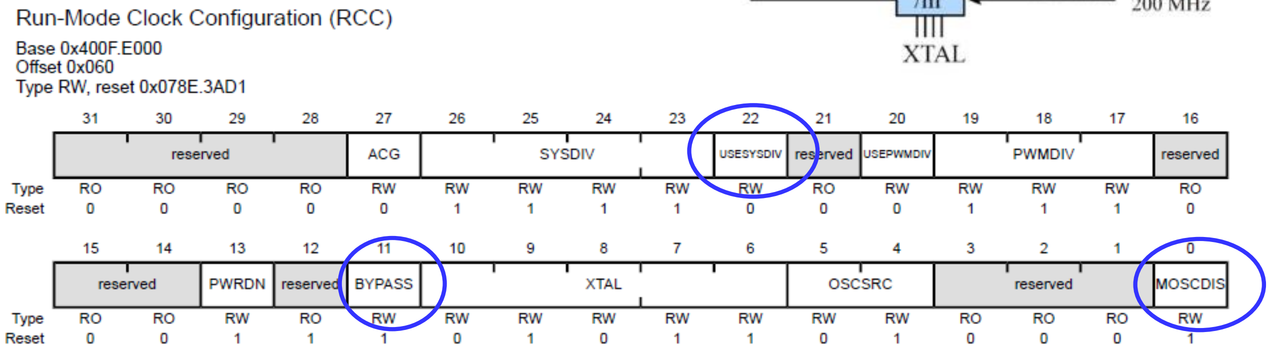
1. Bypass PLL, enable main oscillator (MOSC) & disable clock division during PLL setup.
2. Specify clock crystal (MOSC = 16MHz) & enable system clock divider.
3. Set DIV400 bit to 1 (use 400MHz PLL clock).
4. Set to use main oscillator.
5. Set processor clock speed.
6. Activate PLL.
7. Wait for PLL to lock.
8. Enable PLL.

PLL Initialization

Step 1:

- Bypass PLL & enable main oscillator (MOSC) during PLL setup.
  - Through clearing the MOSCDIS in RCC register.
  - Ensure main oscillator clock has stabilized & clear interrupt status before beginning PLL initialization.





Run-Mode Clock Configuration (RCC)

Base 0x400F.E000
Offset 0x060
Type RW, reset 0x078E.3AD1

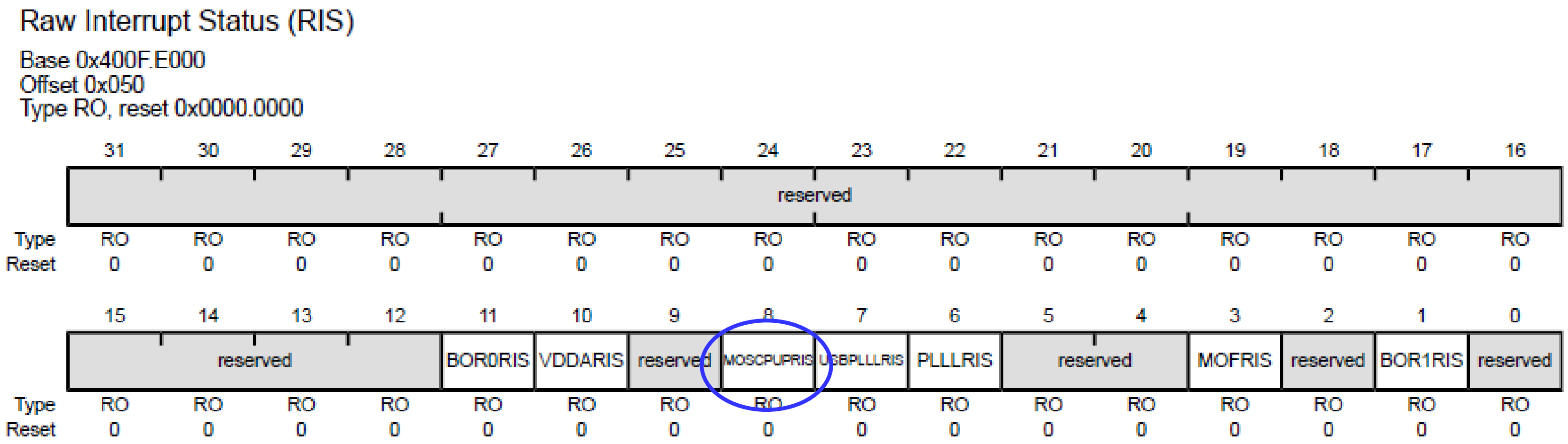
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved				ACG	SYSDIV				USESYSDIV	reserved	USEPWMDIV	PWMDIV			reserved
Type	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RO	RW	RW	RW	RW	RO
Reset	0	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		PWRDN	reserved	BYPASS	XTAL					OSCSRC		reserved			MOSCDIS
Type	RO	RO	RW	RO	RW	RW	RW	RW	RW	RW	RW	RW	RO	RO	RO	RW
Reset	0	0	1	1	1	0	1	0	1	1	0	1	0	0	0	1

```
SYSCTL->RCC |= SYSCTL_RCC_BYPASS;   /* 1. bypass the PLL for now */
/* 1. enable main osc during PLL setup, do not use clock divider */
SYSCTL->RCC &= ~(SYSCTL_RCC_MOSCDIS | SYSCTL_RCC_USESYSDIV);
/* Wait for main clock to be ready */
while( 0 == (SYSCTL->RIS & SYSCTL_RIS_MOSCPUPRIS) ){};
SYSCTL->MISC = SYSCTL_RIS_MOSCPUPRIS; // clear MOSC power-up intr status
```

Bit definitions are in header file: TMC123GH6PM7.h

Raw Interrupt Status (RIS) Register



Raw Interrupt Status (RIS)

Base 0x400F.E000
Offset 0x050
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

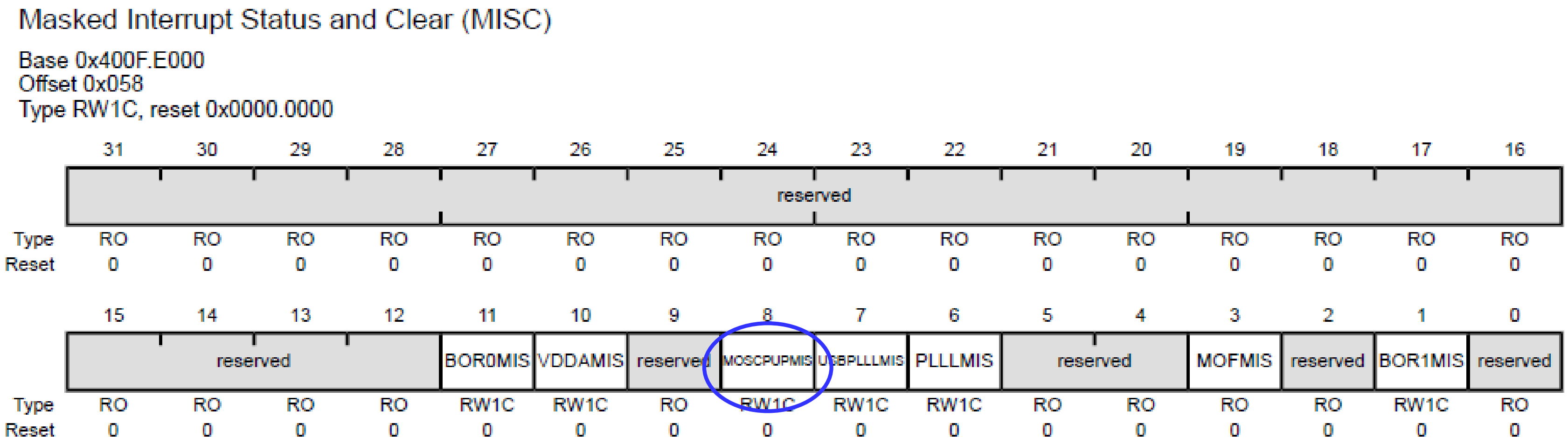
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				BOR0RIS	VDDARIS	reserved	MOSCPUPRIS	USBPLLLRIS	PLLLRIS	reserved		MOFRIS	reserved	BOR1RIS	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- This register indicates status for the System Control to get the interrupt status for the corresponding interrupt in the **RIS** register.

- **MOSCPUPRIS** (MOSC Power Up) bit:
  - '0': Main Oscillator (MOSC) has NOT reached set frequency.
  - '1': MOSC has reached set frequency.
  - bit is cleared by writing a '1' to the MOSCPUPMIS bit in **MISC** register. *(see next slide)*

Source: Tiva C Series TM4C123GH6PM7 Data Sheet (spmu376e.pdf, p244)

Masked Interrupt Status and Clear (MISC Register)



Masked Interrupt Status and Clear (MISC)

Base 0x400F.E000
Offset 0x058
Type RW1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				BOR0MIS	VDDAMIS	reserved	MOSCPUPMIS	USBPLLLMIS	PLLLMIS	reserved		MOFMIS	reserved	BOR1MIS	reserved
Type	RO	RO	RO	RO	RW1C	RW1C	RO	RW1C	RW1C	RW1C	RO	RO	RO	RO	RW1C	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- Read this register to get the interrupt status for the corresponding interrupt in the **RIS** register.

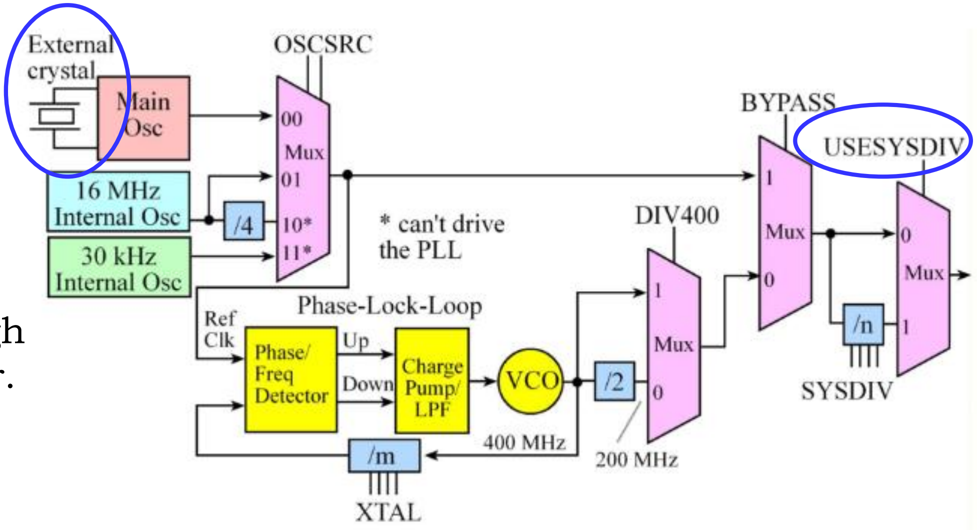
- **MOSCPUPMIS** bit: writing a ‘1’ to this bit clears the MOSCPUPRIS bit in **RIS** register.
  - ‘0’: MOSC PLL has NOT reached set frequency.
  - ‘1’: MOSC PLL has reached set frequency.

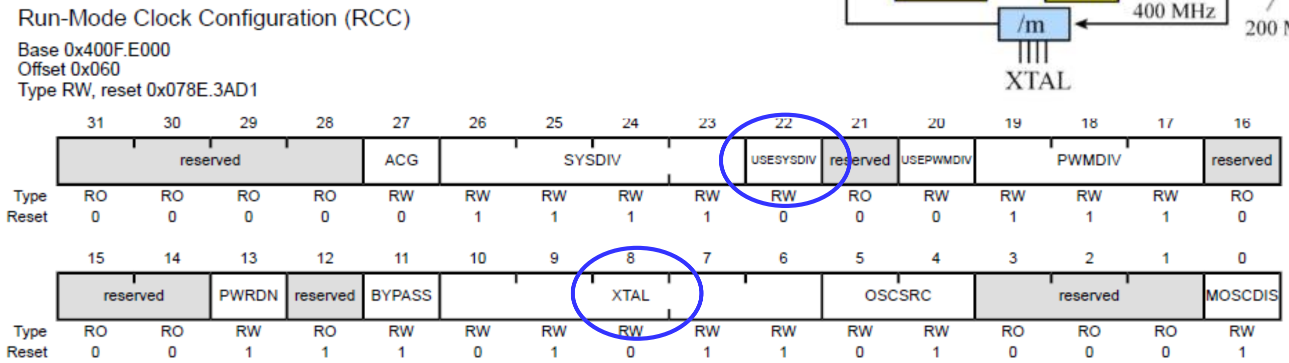
Source: Tiva C Series TM4C123GH6PM7 Data Sheet (spmu376e.pdf, p249)

PLL Initialization

Step 2:

- Specify clock crystal value (16MHz) through the XTAL bits in RCC register.
- *Tiva LaunchPad is loaded with an external 16MHz crystal*
- Enable system clock divider through the USESYSDIV bit in RCC register.





Run-Mode Clock Configuration (RCC)

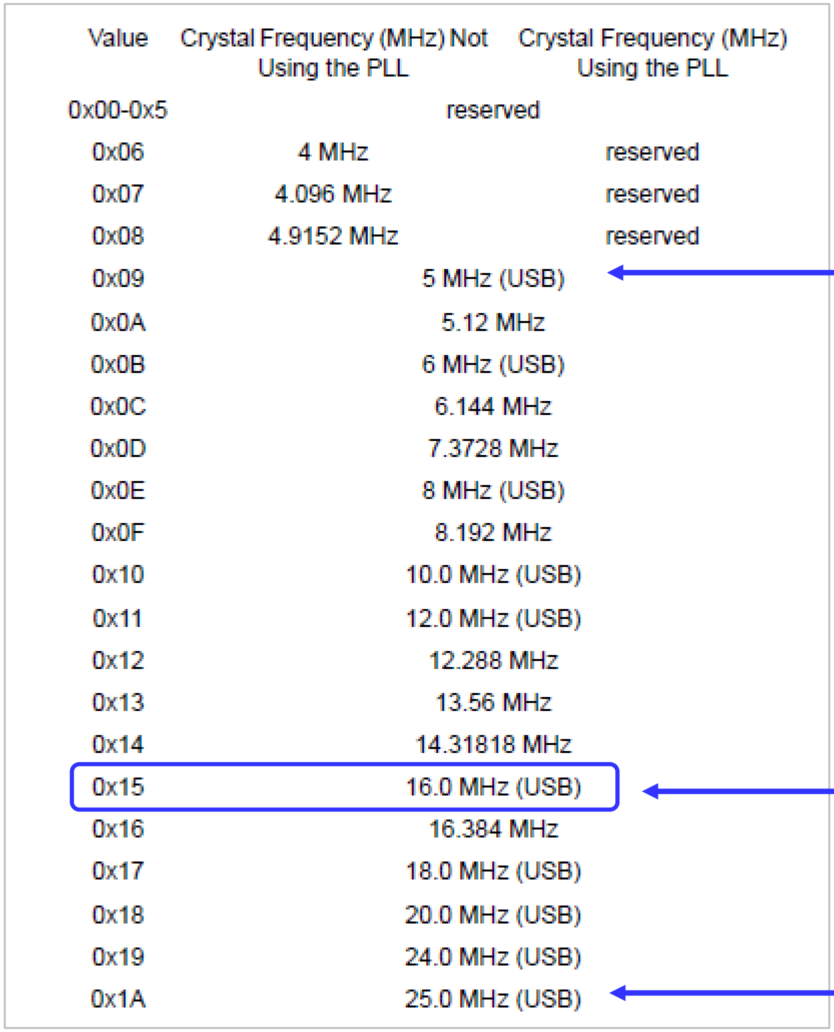
Base 0x400F.E000
Offset 0x060
Type RW, reset 0x078E.3AD1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved				ACG	SYSDIV				USESYSDIV	reserved	USEPWMDIV	PWMDIV			reserved
Type	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RO	RW	RW	RW	RW	RO
Reset	0	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		PWRDN	reserved	BYPASS	XTAL					OSCSRC		reserved			MOSCDIS
Type	RO	RO	RW	RO	RW	RW	RW	RW	RW	RW	RW	RW	RO	RO	RO	RW
Reset	0	0	1	1	1	0	1	0	1	1	0	1	0	0	0	1

```
/* from header file:      */
#define SYSCTL_RCC_XTAL_M    0x000007C0
#define SYSCTL_RCC_SYSDIV_M  0x07800000

SYSCTL->RCC &= ~(SYSCTL_RCC_XTAL_M | SYSCTL_RCC_SYSDIV_M);
SYSCTL->RCC |= SYSCTL_RCC_XTAL_16MHZ  /* 2. select 16MHz xtal */
             | SYSCTL_RCC_USESYSDIV; /*  & enable system clk divider  */
```

XTAL Field Definitions (RCC Register)



Value	Crystal Frequency (MHz) Not Using the PLL	Crystal Frequency (MHz) Using the PLL
0x00-0x5	reserved	
0x06	4 MHz	reserved
0x07	4.096 MHz	reserved
0x08	4.9152 MHz	reserved
0x09	5 MHz (USB)	
0x0A	5.12 MHz	
0x0B	6 MHz (USB)	
0x0C	6.144 MHz	
0x0D	7.3728 MHz	
0x0E	8 MHz (USB)	
0x0F	8.192 MHz	
0x10	10.0 MHz (USB)	
0x11	12.0 MHz (USB)	
0x12	12.288 MHz	
0x13	13.56 MHz	
0x14	14.31818 MHz	
0x15	16.0 MHz (USB)	
0x16	16.384 MHz	
0x17	18.0 MHz (USB)	
0x18	20.0 MHz (USB)	
0x19	24.0 MHz (USB)	
0x1A	25.0 MHz (USB)	

Min value of external crystal = 5MHz

LaunchPad external crystal = 16MHz

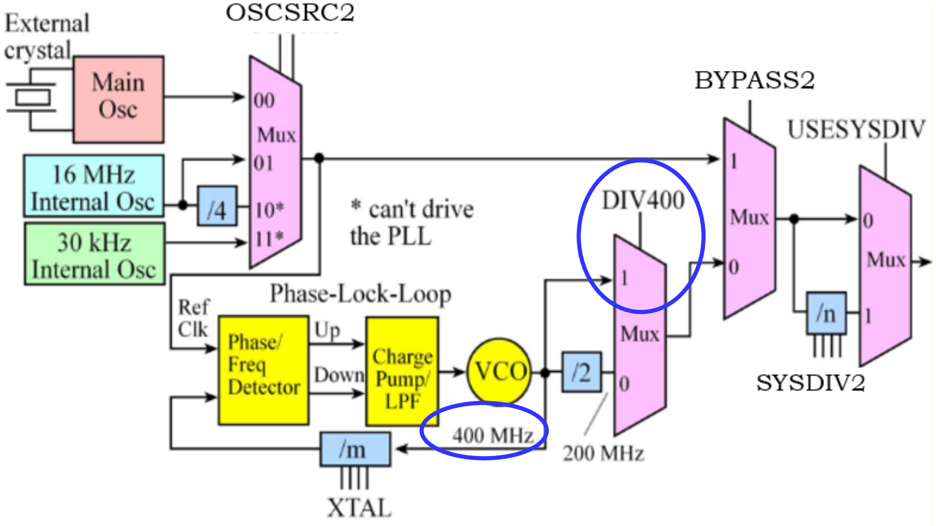
Max value of external crystal = 25MHz

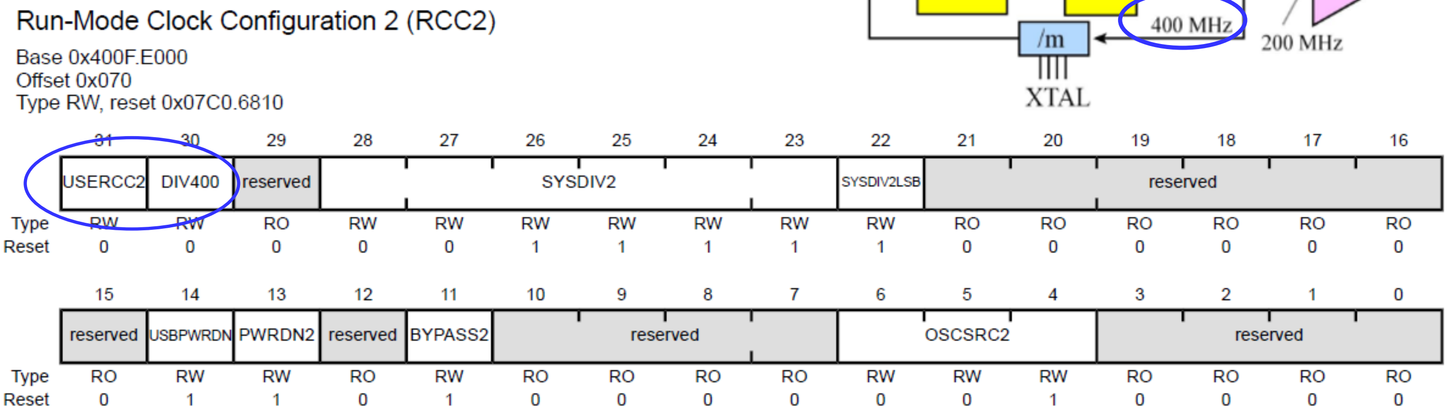
Source: Tiva C Series TM4C123GH6PM7 Data Sheet (spmu376e.pdf, p256)

PLL Initialization

Step 3:

- Set DIV400 bit in RCC2 register to 1 to use 400 MHz VCO & 7-bit divisor.
- Note: In order to program RCC2, bit USERRCC2 has to be set first.





Run-Mode Clock Configuration 2 (RCC2)

Base 0x400F.E000
Offset 0x070
Type RW, reset 0x07C0.6810

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	USERCC2	DIV400	reserved	SYSDIV2						SYSDIV2LSB	reserved					
Type	RW	RW	RO	RW	RW	RW	RW	RW	RW	RW	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0

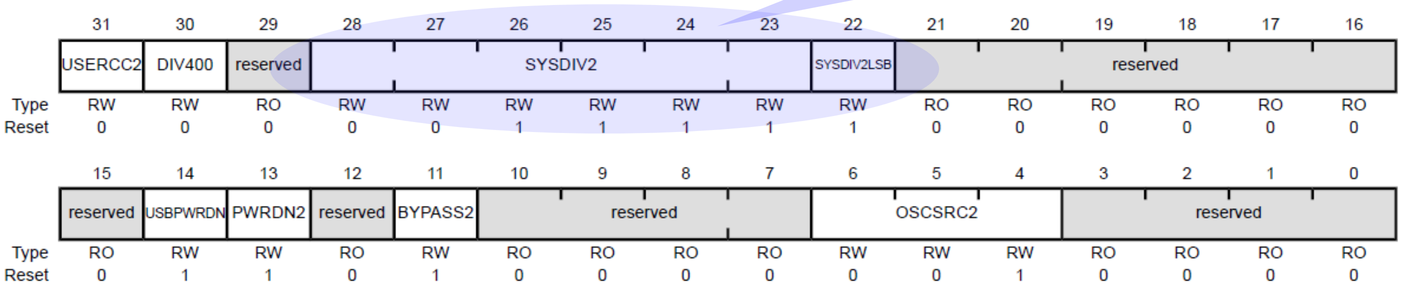
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	USBPWRDN	PWRDN2	reserved	BYPASS2	reserved				OSCSRC2			reserved			
Type	RO	RW	RW	RO	RW	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO
Reset	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0

```
SYSCTL->RCC2 |= SYSCTL_RCC2_USERCC2; /* this bit must be set to use RCC2     */
SYSCTL->RCC2 |= SYSCTL_RCC2_DIV400;  /* 3. set DIV400=1(use 400MHz PLL clk)  */
```

VCO Clock Division

- **PLL VCO output frequency = 400 MHz**
- Tiva LaunchPad design:
  - **VCO Divisor = (SYSDIV2+1)** , or
  - **SYSDIV2 = (VCO Divisor-1)**
- SYSDIV2 field in RCC2 register provides the clock divisor.
  - If **DIV400 = 0**:
    - VCO = 200 MHz (clock is divided by 2 before clock divisor is applied).
    - **SYSDIV2** (clock divisor) is **6-bits** => division of up to 64 (= 2^6).
  - If **DIV400 = 1**:
    - VCO clock = 400 MHz (no VCO clock division).
    - **SYSDIV2LSB** is appended to **SYSDIV2** to get **7-bits**. => division of up to 128 (=2^7)

[SYSDIV2:SYSDIV2LSB] -> (6 + 1)
bits for clock divisor



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	USERCC2	DIV400	reserved	SYSDIV2						SYSDIV2LSB	reserved					
Type	RW	RW	RO	RW	RW	RW	RW	RW	RW	RW	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	USBPWRDN	PWRDN2	reserved	BYPASS2	reserved				OSCSRC2			reserved			
Type	RO	RW	RW	RO	RW	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO
Reset	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0

RCC2 register

Source: Tiva C Series TM4C123GH6PM7 Data Sheet (spmu376e.pdf, p224)

SYSDIV2 (VCO Clock Divisor)

- In Step 3, we have set DIV400 = ‘1’ which means the divisor is 7 bits.
- In this case, VCO clock divisor is made up of the **concatenation** of SYSDIV2 (6 bits) & SYSDIV2LSB (1 bit); total of 7 bits.

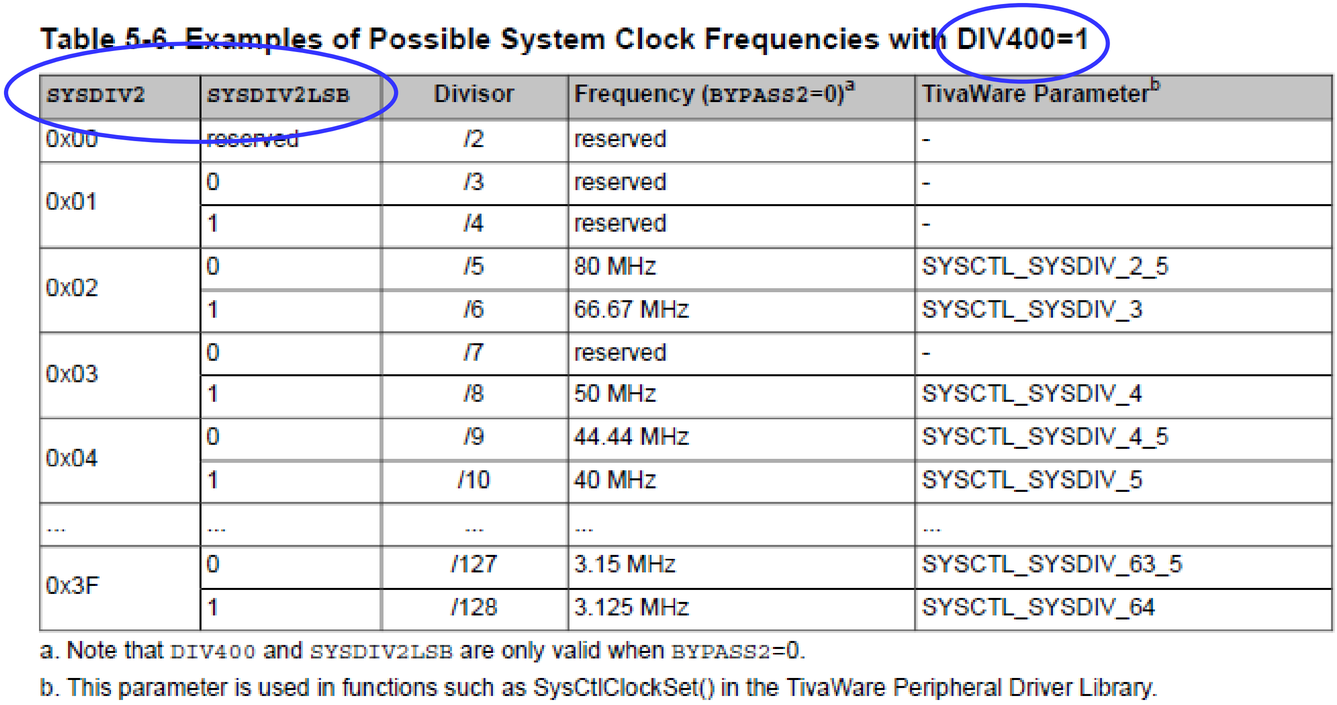


Table 5-6. Examples of Possible System Clock Frequencies with DIV400=1

SYSDIV2	SYSDIV2LSB	Divisor	Frequency (BYPASS2=0)[a]	TivaWare Parameter[b]
0x00	reserved	/2	reserved	-
0x01	0	/3	reserved	-
	1	/4	reserved	-
0x02	0	/5	80 MHz	SYSCTL_SYSDIV_2_5
	1	/6	66.67 MHz	SYSCTL_SYSDIV_3
0x03	0	/7	reserved	-
	1	/8	50 MHz	SYSCTL_SYSDIV_4
0x04	0	/9	44.44 MHz	SYSCTL_SYSDIV_4_5
	1	/10	40 MHz	SYSCTL_SYSDIV_5
...	...	...	...	...
0x3F	0	/127	3.15 MHz	SYSCTL_SYSDIV_63_5
	1	/128	3.125 MHz	SYSCTL_SYSDIV_64

[a] Note that DIV400 and SYSDIV2LSB are only valid when BYPASS2=0.

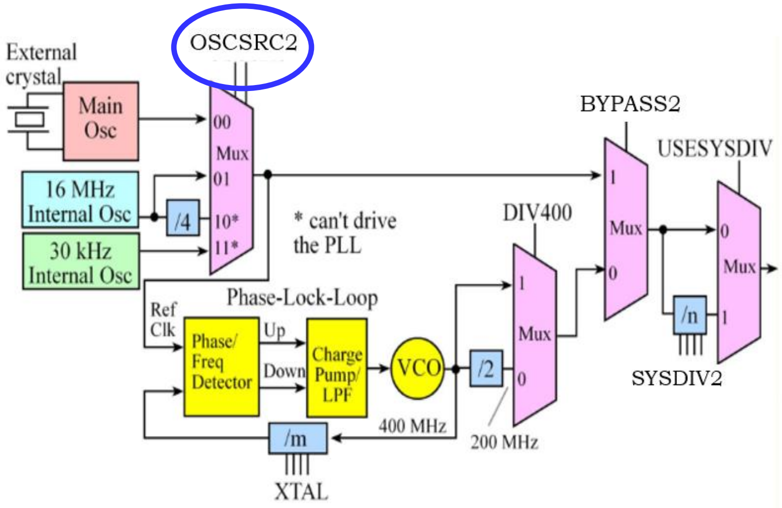
[b] This parameter is used in functions such as SysCtlClockSet() in the TivaWare Peripheral Driver Library.

Source: Tiva C Series TM4C123GH6PM7 Data Sheet (spmu376e.pdf, p224)

PLL Initialization

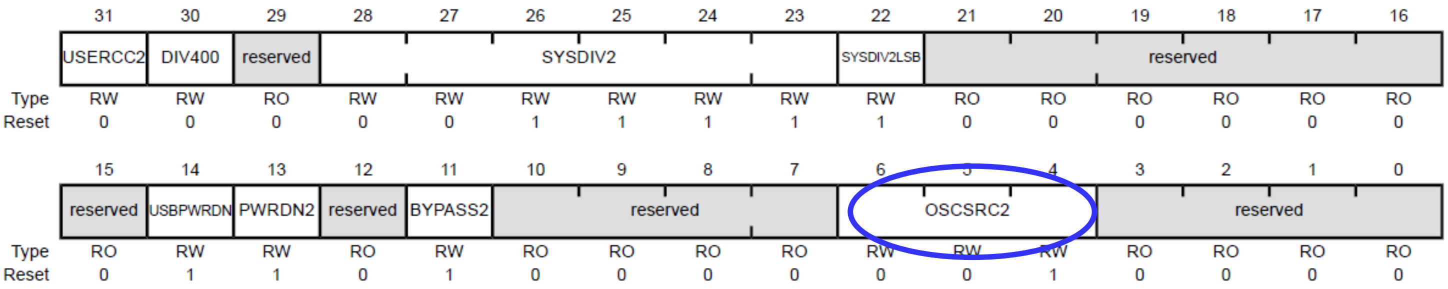
Step 4:

- Set to use main oscillator.
- Through setting the Oscillator Source bits: OSCSRC2 in RCC2 register.



Run-Mode Clock Configuration 2 (RCC2)

Base 0x400F.E000
Offset 0x070
Type RW, reset 0x07C0.6810

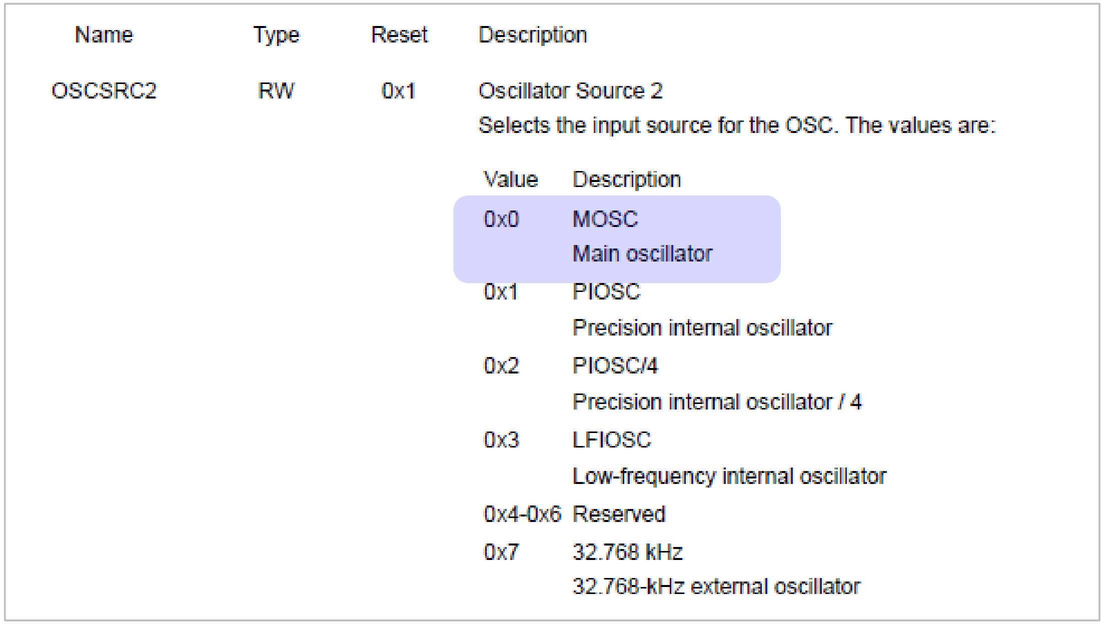


	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	USERCC2	DIV400	reserved	SYSDIV2						SYSDIV2LSB	reserved					
Type	RW	RW	RO	RW	RW	RW	RW	RW	RW	RW	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	USBPWRDN	PWRDN2	reserved	BYPASS2	reserved				OSCSRC2			reserved			
Type	RO	RW	RW	RO	RW	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO
Reset	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0

```
SYSCTL->RCC2 &= ~(0x7f<<SYSCTL_RCC2_SYSDIV2_DIV400_S);  /* clear divisor bits */
SYSCTL->RCC2 &= ~SYSCTL_RCC2_OSCSRC2_M;
SYSCTL->RCC2 |= SYSCTL_RCC2_OSCSRC2_MO; // 4. select main oscillator
```

OSCSRC2 bits (RCC2 Register)



Name	Type	Reset	Description
OSCSRC2	RW	0x1	Oscillator Source 2 Selects the input source for the OSC. The values are:

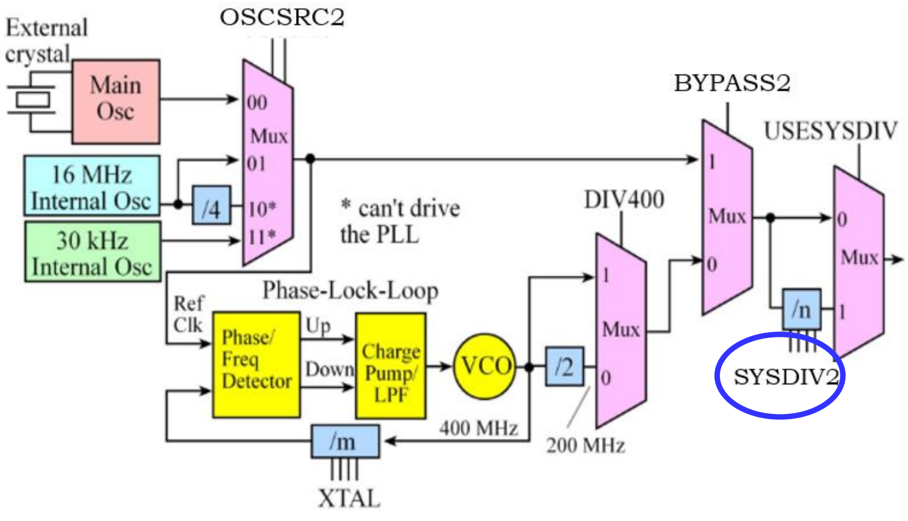
Value	Description
0x0	MOSC Main oscillator
0x1	PIOSC Precision internal oscillator
0x2	PIOSC/4 Precision internal oscillator / 4
0x3	LFIOSC Low-frequency internal oscillator
0x4-0x6	Reserved
0x7	32.768 kHz 32.768-kHz external oscillator

Source: Tiva C Series TM4C123GH6PM7 Data Sheet (spmu376e.pdf, p262)

PLL Initialization

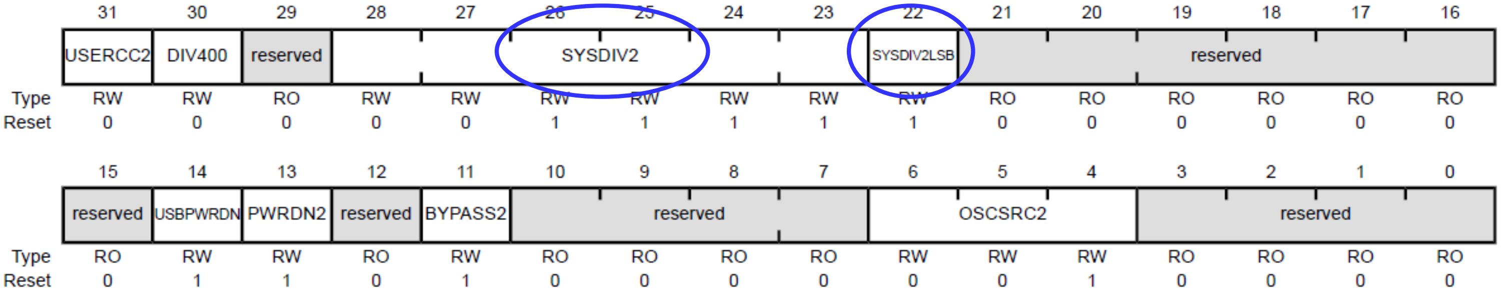
Step 5:

- Set processor clock speed through setting the PLL VCO clock divisor.
- Clock divisor is set through the SYSDIV2 & SYSDIV2LSB bits in RCC2 register.



Run-Mode Clock Configuration 2 (RCC2)

Base 0x400F.E000
Offset 0x070
Type RW, reset 0x07C0.6810



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	USERCC2	DIV400	reserved	SYSDIV2						SYSDIV2LSB	reserved					
Type	RW	RW	RO	RW	RW	RW	RW	RW	RW	RW	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	USBPWRDN	PWRDN2	reserved	BYPASS2	reserved				OSCSRC2			reserved			
Type	RO	RW	RW	RO	RW	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO
Reset	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0

```
SYSCTL->RCC2 |= (4U<<22); /* 5. set clock speed */
```

System Divisor (RCC2 Register)

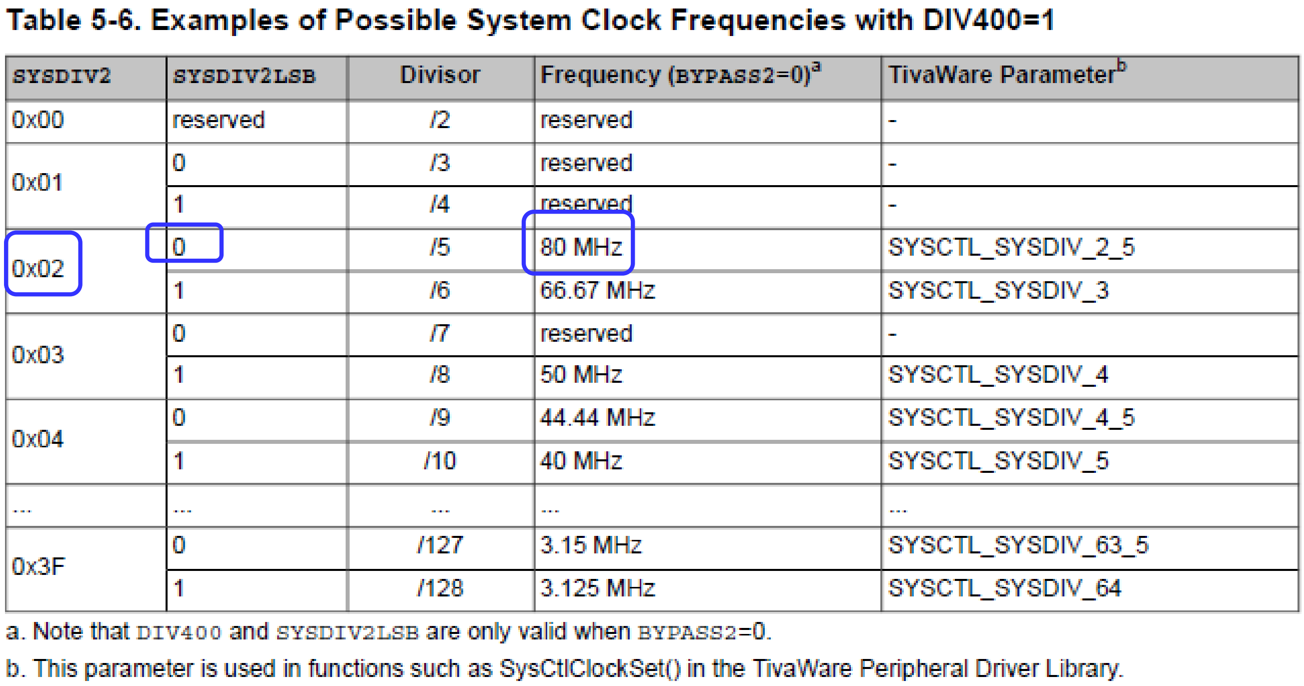


Table 5-6. Examples of Possible System Clock Frequencies with DIV400=1

SYSDIV2	SYSDIV2LSB	Divisor	Frequency (BYPASS2=0)[a]	TivaWare Parameter[b]
0x00	reserved	/2	reserved	-
0x01	0	/3	reserved	-
	1	/4	reserved	-
0x02	0	/5	80 MHz	SYSCTL_SYSDIV_2_5
	1	/6	66.67 MHz	SYSCTL_SYSDIV_3
0x03	0	/7	reserved	-
	1	/8	50 MHz	SYSCTL_SYSDIV_4
0x04	0	/9	44.44 MHz	SYSCTL_SYSDIV_4_5
	1	/10	40 MHz	SYSCTL_SYSDIV_5
...	...	...	...	...
0x3F	0	/127	3.15 MHz	SYSCTL_SYSDIV_63_5
	1	/128	3.125 MHz	SYSCTL_SYSDIV_64

a. Note that DIV400 and SYSDIV2LSB are only valid when BYPASS2=0.

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

- If 80MHz system clock is required, we set Divisor = (VCO freq)/80MHz = 400MHz/80MHz = 5.
  - Bits [SYSDIV2:SYSDIV2LSB] = 5 – 1 = 4 = [0b010.0].
    => SYSDIV2 = 0x02 or 0b010 & SYSDIV2LSB = 0.

SYSDIV2 (6 bits)

Recall: SYSDIV2 = (VCO Divisor-1)

SYSDIV2LSB (1 bit)

```
SYSCTL->RCC2 |= (4U<<22); /* 5. set clock speed 80MHz*/
```

Source: Tiva C Series TM4C123GH6PM7 Data Sheet (spmu376e.pdf, p222)

Ex 1: System Divisor (RCC2 Register)

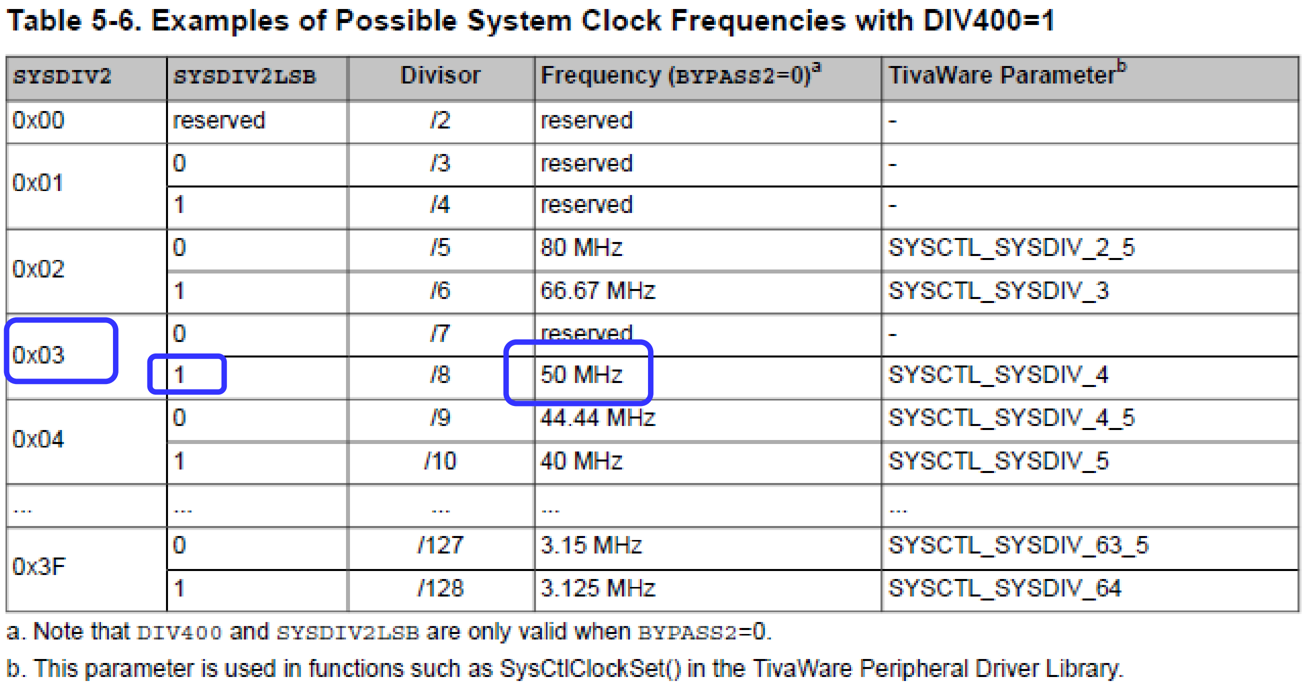


Table 5-6. Examples of Possible System Clock Frequencies with DIV400=1

SYSDIV2	SYSDIV2LSB	Divisor	Frequency (BYPASS2=0)[a]	TivaWare Parameter[b]
0x00	reserved	/2	reserved	-
0x01	0	/3	reserved	-
	1	/4	reserved	-
0x02	0	/5	80 MHz	SYSCTL_SYSDIV_2_5
	1	/6	66.67 MHz	SYSCTL_SYSDIV_3
0x03	0	/7	reserved	-
	1	/8	50 MHz	SYSCTL_SYSDIV_4
0x04	0	/9	44.44 MHz	SYSCTL_SYSDIV_4_5
	1	/10	40 MHz	SYSCTL_SYSDIV_5
...	...	...	...	...
0x3F	0	/127	3.15 MHz	SYSCTL_SYSDIV_63_5
	1	/128	3.125 MHz	SYSCTL_SYSDIV_64

a. Note that DIV400 and SYSDIV2LSB are only valid when BYPASS2=0.

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

- If 50MHz system clock is required, what is the value of Divisor we should set?
  - 400MHz/50MHz = 8 = Divisor
  - Bits [SYSDIV2:SYSDIV2LSB] = 8 – 1 = 7 = [0b011.1].
- Write the C code:

SYSDIV2 (6 bits)

SYSDIV2LSB (1 bit)

```
SYSCTL->RCC2 |= (7U<<22); /* 5. set clock speed 50MHz */
```

Ex 2: System Divisor (RCC2 Register)

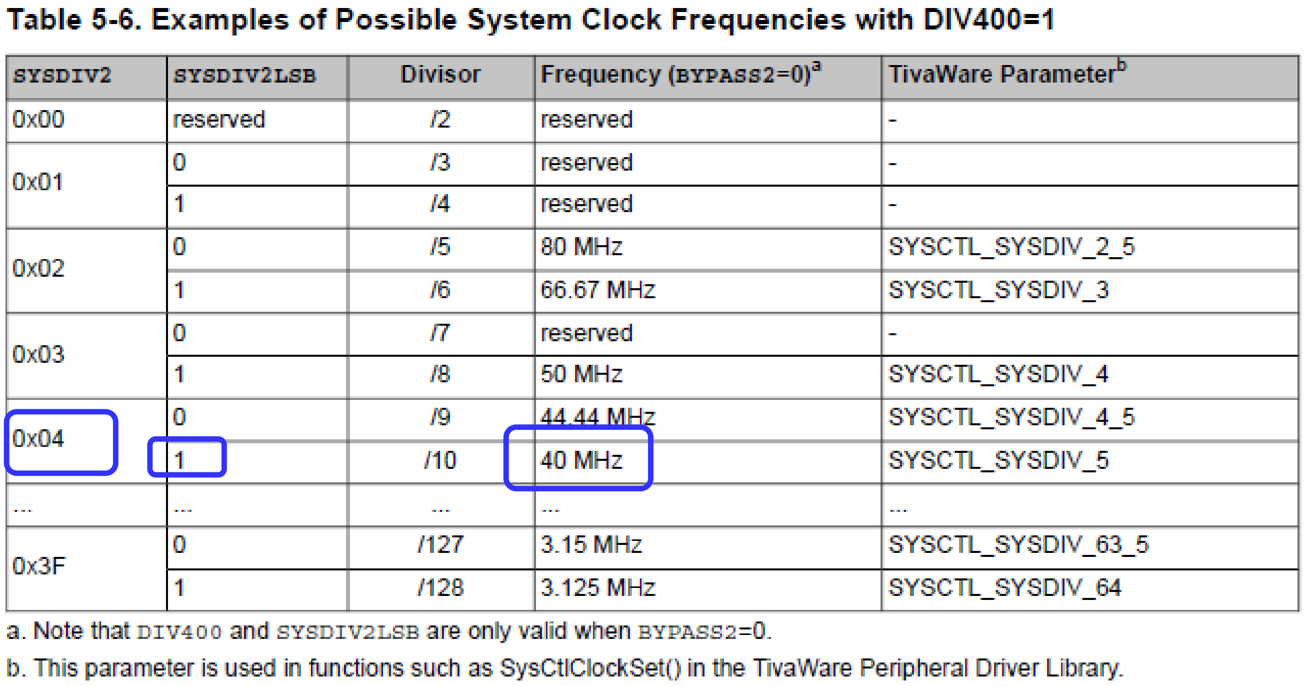


Table 5-6. Examples of Possible System Clock Frequencies with DIV400=1

SYSDIV2	SYSDIV2LSB	Divisor	Frequency (BYPASS2=0)[a]	TivaWare Parameter[b]
0x00	reserved	/2	reserved	-
0x01	0	/3	reserved	-
	1	/4	reserved	-
0x02	0	/5	80 MHz	SYSCTL_SYSDIV_2_5
	1	/6	66.67 MHz	SYSCTL_SYSDIV_3
0x03	0	/7	reserved	-
	1	/8	50 MHz	SYSCTL_SYSDIV_4
0x04	0	/9	44.44 MHz	SYSCTL_SYSDIV_4_5
	1	/10	40 MHz	SYSCTL_SYSDIV_5
...	...	...	...	...
0x3F	0	/127	3.15 MHz	SYSCTL_SYSDIV_63_5
	1	/128	3.125 MHz	SYSCTL_SYSDIV_64

[a] Note that DIV400 and SYSDIV2LSB are only valid when BYPASS2=0.

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

- If 40MHz system clock is required, what is the value of Divisor we should set?
  - 400MHz/40MHz = 10 = Divisor
  - Bits [SYSDIV2:SYSDIV2LSB] = 10 – 1 = 9 = [0b100.1].
- Write the C code:

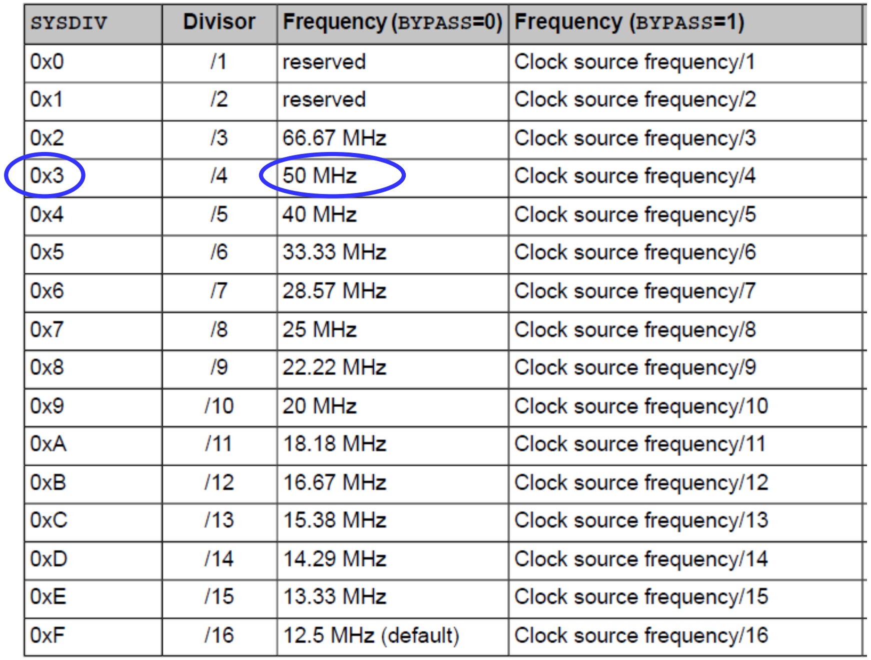
SYSDIV2 (6 bits)

SYSDIV2LSB (1 bit)

```
SYSCTL->RCC2 |= (9U<<22); /* 5. set clock speed 40MHz */
```

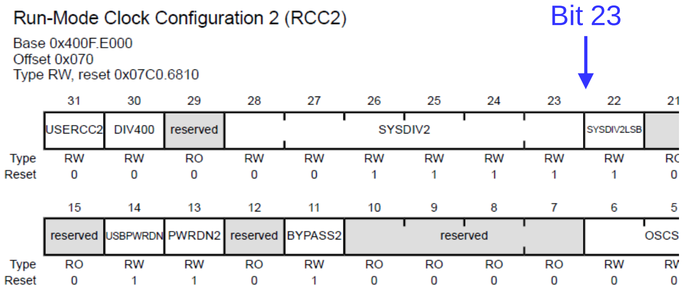
SYSDIV2 (VCO Clock Divisor)

- If we set DIV400 = ‘0’, the divisor is 6 bits.



SYSDIV	Divisor	Frequency (BYPASS=0)	Frequency (BYPASS=1)
0x0	/1	reserved	Clock source frequency/1
0x1	/2	reserved	Clock source frequency/2
0x2	/3	66.67 MHz	Clock source frequency/3
0x3	/4	50 MHz	Clock source frequency/4
0x4	/5	40 MHz	Clock source frequency/5
0x5	/6	33.33 MHz	Clock source frequency/6
0x6	/7	28.57 MHz	Clock source frequency/7
0x7	/8	25 MHz	Clock source frequency/8
0x8	/9	22.22 MHz	Clock source frequency/9
0x9	/10	20 MHz	Clock source frequency/10
0xA	/11	18.18 MHz	Clock source frequency/11
0xB	/12	16.67 MHz	Clock source frequency/12
0xC	/13	15.38 MHz	Clock source frequency/13
0xD	/14	14.29 MHz	Clock source frequency/14
0xE	/15	13.33 MHz	Clock source frequency/15
0xF	/16	12.5 MHz (default)	Clock source frequency/16

- If 50MHz system clock is required:
  - 200MHz/50MHz = 4 = Divisor
  - SYSDIV2 = 4 – 1 = 3



Run-Mode Clock Configuration 2 (RCC2)

Base 0x400F.E000
Offset 0x070
Type RW, reset 0x07C0.6810

Bit 23

	31	30	29	28	27	26	25	24	23	22	21
	USERCC2	DIV400	reserved	SYSDIV2						SYSDIV2LSB	
Type	RW	RW	RO	RW	RW	RW	RW	RW	RW	RW	RO
Reset	0	0	0	0	0	1	1	1	1	1	0

	15	14	13	12	11	10	9	8	7	6	5
	reserved	USBPWRDN	PWRDN2	reserved	BYPASS2	reserved				OSCS	
Type	RO	RW	RW	RO	RW	RO	RO	RO	RO	RW	RW
Reset	0	1	1	0	1	0	0	0	0	0	0

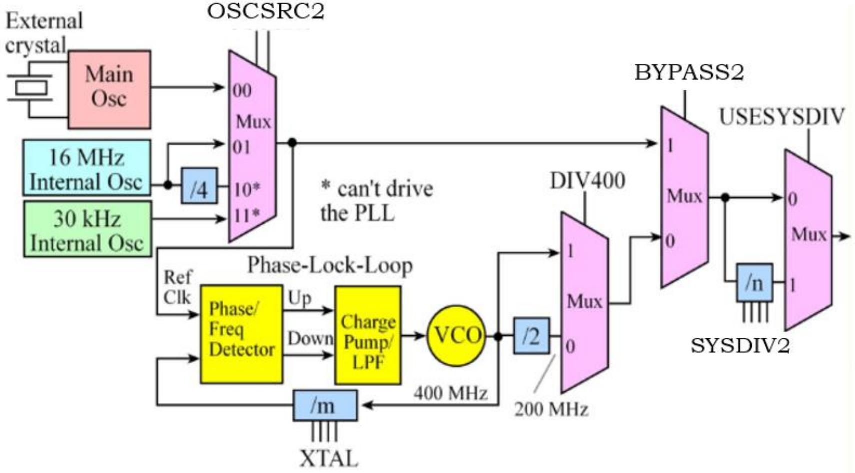
```
SYSCTL->RCC2 |= (3U<<23); /* set clock speed 50MHz */
```

Source: Tiva C Series TM4C123GH6PM7 Data Sheet (spmu376e.pdf, p223)

PLL Initialization

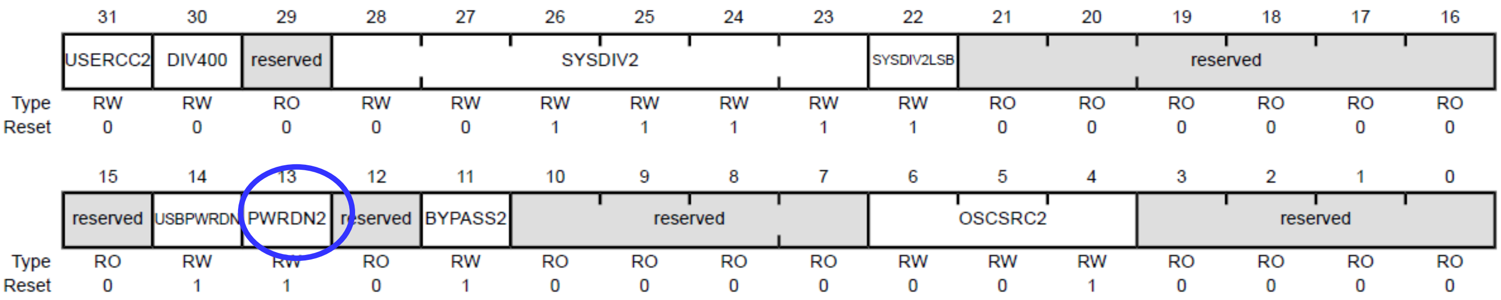
Step 6:

- Activate the PLL through clearing the PWRDN2 bit in the RCC2 register.



Run-Mode Clock Configuration 2 (RCC2)

Base 0x400F.E000
Offset 0x070
Type RW, reset 0x07C0.6810



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	USERCC2	DIV400	reserved	SYSDIV2						SYSDIV2LSB	reserved					
Type	RW	RW	RO	RW	RW	RW	RW	RW	RW	RW	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0

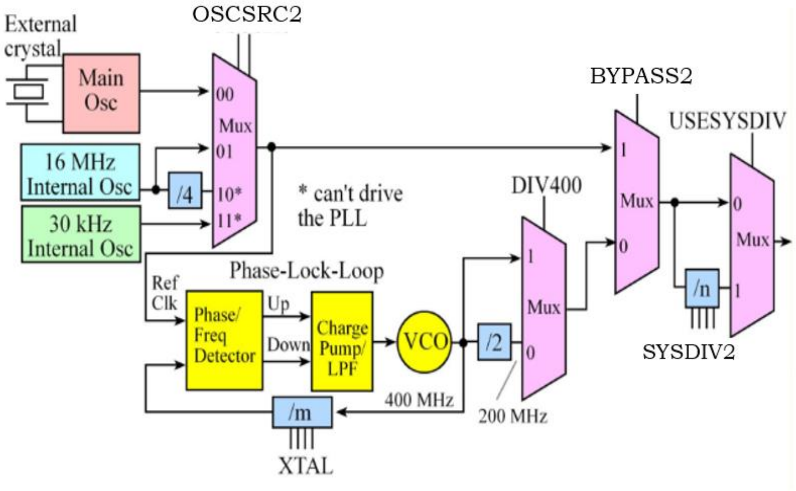
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	USBPWRDN	PWRDN2	reserved	BYPASS2	reserved				OSCSRC2			reserved			
Type	RO	RW	RW	RO	RW	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO
Reset	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0

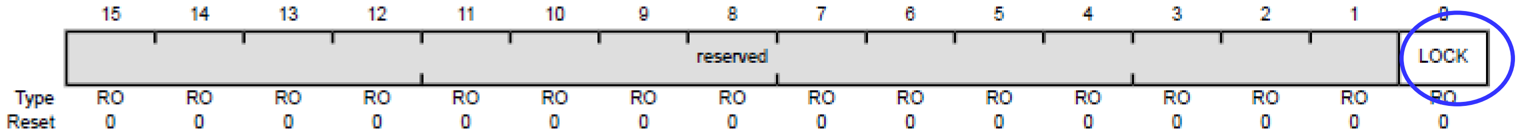
```
SYSCTL->RCC2 &= ~SYSCTL_RCC2_PWRDN2; /* 6. activate PLL  */
```

PLL Initialization

Step 7:

- Wait for the PLL to lock through checking the LOCK bit in the PLLSTAT register.



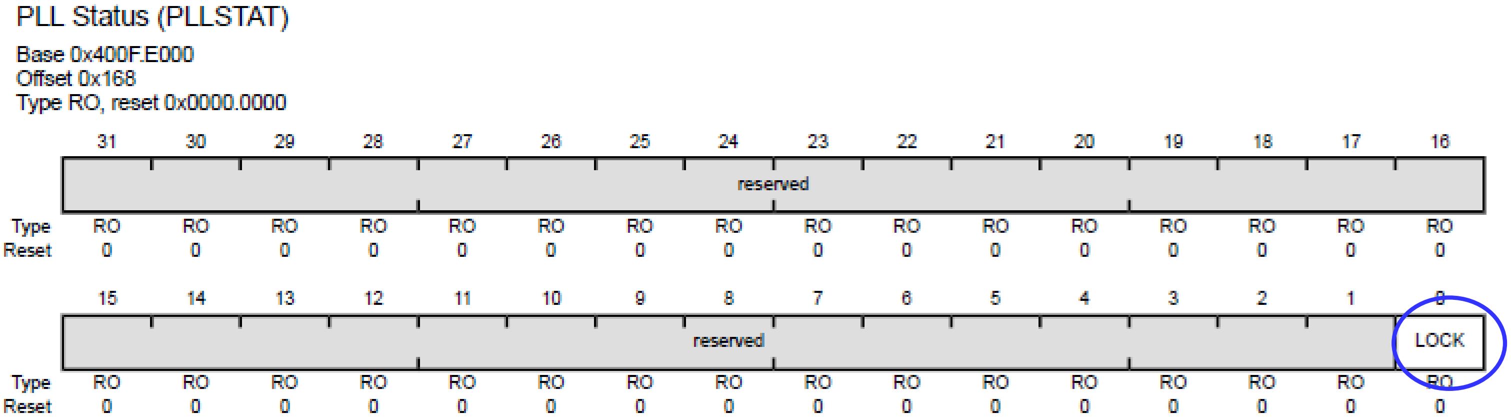


	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															LOCK
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PLLSTAT register

```
while( 0 == (SYSCTL->PLLSTAT & SYSCTL_PLLSTAT_LOCK)){}; /* 7. wait for PLL to lock  */
```

PLL Status (PLLSTAT) Register



PLL Status (PLLSTAT)

Base 0x400F.E000
Offset 0x168
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															LOCK
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- Read this register to get the lock status of the PLL.
- Only 1 bit is used.

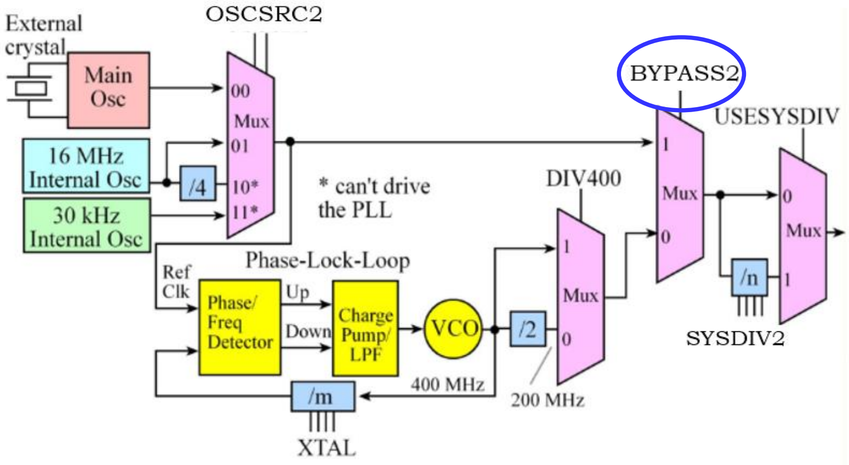
- **LOCK** bit:
  - ‘0’: PLL is not powered and not locked.
  - ‘1’: PLL is powered and locked.

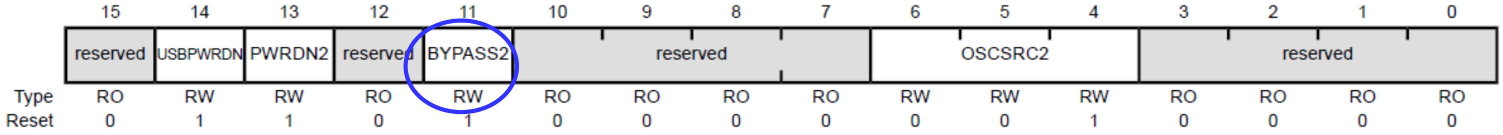
Source: Tiva C Series TM4C123GH6PM7 Data Sheet (spmu376e.pdf, p249)

PLL Initialization

Step 8:

- Enable the PLL once lock status has been verified. Done through clearing the BYPASS2 bit in the RCC2 register.





	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	USBPWRDN	PWRDN2	reserved	BYPASS2	reserved				OSCSRC2			reserved			
Type	RO	RW	RW	RO	RW	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO
Reset	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0

RCC2 register

```
SYSCTL->RCC2 &= ~SYSCTL_RCC2_BYPASS2;       /* 8. enable PLL */
```

PLL Initialization

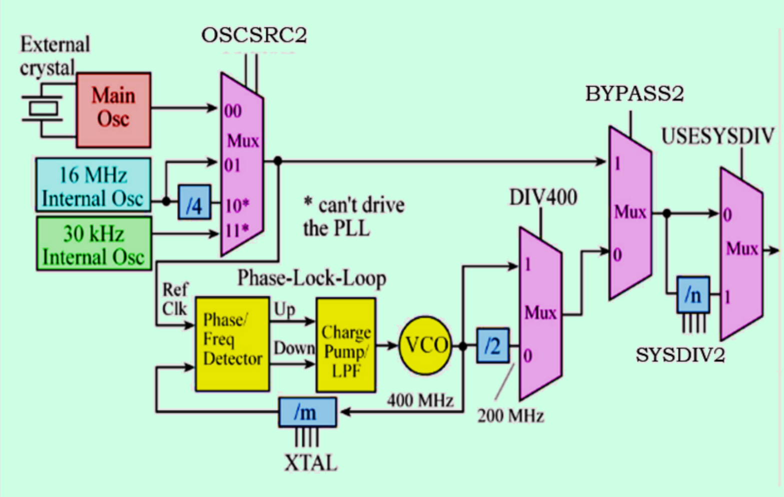
- Let’s put all the initialization codes from Steps 1 to 8 together … ….
- In our KEIL µVison C program, PLL initialization codes can be found in file ‘**system_TM4C123.c**’.
  - Look into the file for function: **`SystemInit()`**.
- Bit fields are defined in header file ‘*TM4C123GH6PM7.h*’.

Configuring PLL *(Example Code)*

```
void SystemInit (void)
{
    /**** … other program codes …. *****/

    SYSCTL->RCC |= SYSCTL_RCC_BYPASS;   /* 1. bypass the PLL for now   */
    /* 1. enable MOSC, disable SYSDIv */
    SYSCTL->RCC &= ~(SYSCTL_RCC_MOSCDIS|SYSCTL_RCC_USESYSDIV);
    while( 0 == (SYSCTL->RIS & SYSCTL_RIS_MOSCPUPRIS) ){};  /* Wait main clock ready */

    SYSCTL->MISC = SYSCTL_RIS_MOSCPUPRIS; /* clear MOSC power-up intr status          */
    SYSCTL->RCC &= ~(SYSCTL_RCC_XTAL_M | SYSCTL_RCC_SYSDIV_M);
    SYSCTL->RCC |= SYSCTL_RCC_XTAL_16MHZ  /* 2. select 16MHz xtal                     */
                 | SYSCTL_RCC_USESYSDIV;  /*    & enable system clk divider           */

    /** Enable RCC2 **/
    SYSCTL->RCC2 |= SYSCTL_RCC2_USERCC2;  /* this bit must be set to use RCC2         */
    SYSCTL->RCC2 |= SYSCTL_RCC2_DIV400;   /* 3. set DIV400=1(use 400MHz PLL clk)      */
    SYSCTL->RCC2 &= ~(0x7f<<SYSCTL_RCC2_SYSDIV2_DIV400_S); /* clear divisor bits      */
    SYSCTL->RCC2 &= ~SYSCTL_RCC2_OSCSRC2_M;
    SYSCTL->RCC2 |= SYSCTL_RCC2_OSCSRC2_MO  /* 4. select main oscillator              */
                 | (4U<<SYSCTL_RCC2_SYSDIV2_DIV400_S); /* 5. set clock speed          */
    SYSCTL->RCC2 &= ~SYSCTL_RCC2_PWRDN2;    /* 6. activate PLL                        */

    while( 0 == (SYSCTL->PLLSTAT & SYSCTL_PLLSTAT_LOCK)){}; /* 7. wait till PLL locks */
    SYSCTL->RCC2 &= ~SYSCTL_RCC2_BYPASS2;   /* 8. enable PLL                          */
}
```

Points to Note on Tiva PLL:

1. PLL can be driven from the PIOSC or MOSC.
   - MOSC is an external clock source.
   - PIOSC is a 16MHz internal clock source.
2. RCC and RCC2 registers are used to setup the PLL.
3. If BYPASS2=1 in RCC2 register, PLL is not used to generate CPU clock.
4. If USESYSDIV=1 in RCC register, clock divisor is used.



5. Input clock source is synthesized to a 400MHz internal PLL clock (through a VCO) which is used to generate the processor clock.
6. VCO output freq = 400 MHz.
7. If DIV400 = ‘0’, the VCO output freq is halved (= 200 MHz) before it is used.
8. VCO output is divided down to obtain the processor clock.
   - **DIV400=1**: VCO freq = 400MHz, clock divisor = 7 bits, divisor is made up of SYSDIV2 (6 bits) & SYSDIV2LSB (1 bit, LSB), Bits [SYSDIV2:SYSDIV2LSB] = clock divisor-1.
   - **DIV400=0**: VCO freq – 200MHz, clock divisor = 6 bits, divisor is made up of SYSDIV2, Bits [SYSDIV2] = clock divisor-1.

Review Questions

1. Which are the 2 clock sources that can be used to drive the PLL on the LaunchPad?
2. What function does the PLLSTAT register serve?
3. What do we mean when we say a ‘PLL has locked’ ?
4. If DIV400 bit in RCC2 is ‘1’, what is the maximum system division possible? Show how you get the number. What is the resultant processor clock speed?
5. What is the USERCC2 bit in RCC2 register used for?
6. I would like to set a bus speed of 25MHz, what is the system clock divisor value to use in RCC2? Assume DIV400 = ‘1’. Write the C code line to set the bus speed.
7. I would like to set a bus speed of 20MHz, what is the system clock divisor value to use in RCC2? Assume DIV400 = ‘0’. Write the C code line to set the bus speed.