ESET 269 – Embedded Systems Development in C

Timers

(Delays With Systick & Timer32)

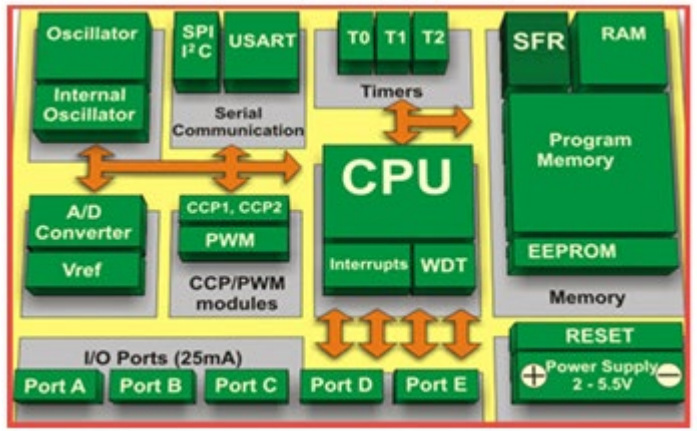
Dr. Garth V. Crosby

Microcontrollers

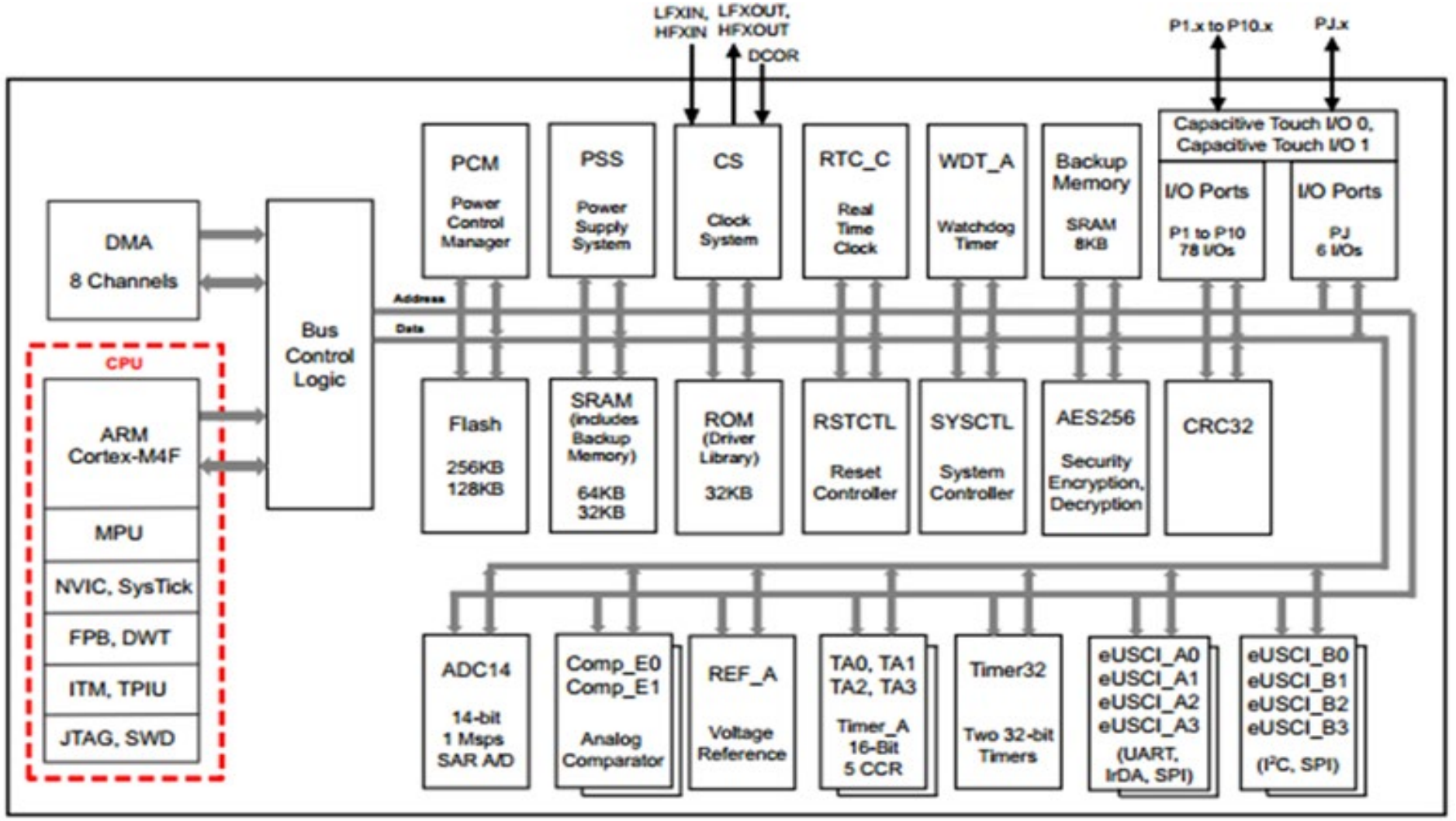
- 8051- Intel, PIC – Microchip, AVR – Atmel, ARM
- ARM processors are a family of central processing units (CPUs)based on a reduced instruction set computer (RISC) architecture
- ARM stands for Advanced RISC Machine
- The ARM cortex-M is a group of 32-bit RISC ARM processor cores licensed by ARM holdings (limited).
  - Optimized for low-cost & energy-efficient IC
  - Embedded in tens of billions of consumer devices
  - Licensed to NXP, STMicroelectronics and Texas Instruments
- The TI MSP432 is a based on the ARM Cortex-M4F CPU

What is a Microcontroller?

- An IC which contains a CPU, ROM, RAM, I/O ports, communication peripherals, etc.
  - A microprocessor is only a CPU on an IC
- Serves as an interface between a system and physical world
  - Control motors, read voltages, communicate to PC, turn on relays, etc.
- Key component in embedded systems
  - Microcontroller (or processor) with external devices attached programmed for a specific dedicated purpose



MSP432 Block Diagram



Creating Delays

- Delays so far have been executing a loop for a certain number of iterations
  - Programmer is not sure what the exact delay will be
- MSP432 has multiple timers (counters) that can be used to create hardware delays
  - Reliable and know exactly how long a delay is
- Instead of executing a loop for a number of iterations, a hardware counter will count a specified number of times

MSP432 Timers

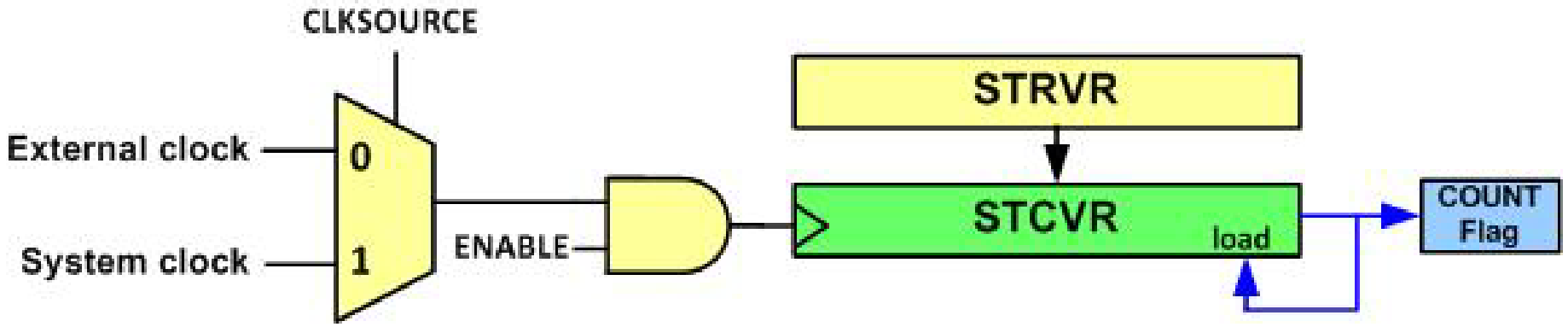
- 3 different timers in MSP432
  - System Tick Time (SysTick) – 24 bit down counter found in every ARM microcontroller.
  - Timer 32 – 32 bit down counter which can be configured with a frequency divider.
  - Timer A – 16-bit up/down counters that have multiple configuration options for use and frequency division.

System Tick (SysTick) Timer

- Nowadays, all microcontrollers comes with a on-chip Timer/Counter.
- SysTick is a 24-bit down counter driven by the master clock (MCLK, the CPU clock)
- Systick
  - Down counts from initial value to 0
  - When it reaches 0, on the next clock pulse, it underflows and raises the COUNT flag
  - It then reloads the initial value and starts over
- The initial value of SysTick can be set to a value between 0x000000 and 0xFFFFFF

System Tick (SysTick) Timer

- STCVR – SysTick current value register, the counter of the SysTick timer
- STRVR – SysTick reload value register, value to preset the counter to once 0 is reached on count
- COUNT flag – Indicates if counter reached 0



SysTick Registers

- There are 3 registers in the SysTick module
  - Systick Reload Value Register (STRVR)
  - Systick Current Value Register (STCVR)
  - Systick Control and Status Register (STCSR): use to start the Systick counter among other things

SysTick Control Register

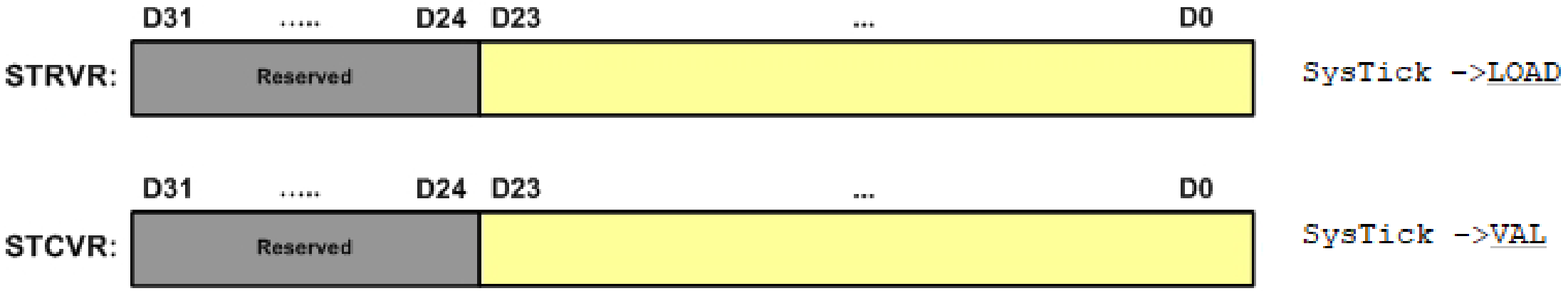
- 32 bit register, but not all the bits are used because they are reserved (not accessible)

bit	Name	Description
0	ENABLE	Enable (0: the counter is disabled, 1: enables SysTick to begin counting down)
1	TICKINT	Interrupt Enable 0: Interrupt generation is disabled, 1: when SysTick counts to 0 an interrupt is generated (See Chapter 6)
2	CLKSOURCE	Clock Source 0: External clock 1: System clock
16	COUNTFLAG	Count Flag 0: the SysTick has not counted down to zero since the last time this bit was read 1: the SysTick has counted down to zero *Note: this flag is cleared by reading the STCSR register.*

```
SysTick ->CTRL
```

STRVR & STCVR

- SysTick Reload Value Register – Contains the value to reload SysTick counter to
- SysTick Current Value Register – Contains the current count value of the SysTick counter



SysTick

- The system clock is 3 MHz (speed of MSP432)
- The counter is 24-bits, contains 16,777,216 counts, and each count is 0.333 µs or 333.3 ns
  - A full count is a delay of about 5.59 seconds
- Effective delay is $\frac{\#\ of\ counts}{CLK\ Frequency}$
- The # counts - 1 is the value that is loaded into the STRVR
  - The -1 is because zero gets a count
- When the STCVR register reaches 0, the COUNTFLAG is set to 1. It is 0 otherwise

Configuring SysTick

- Right now, the SysTick will be used with the following
  - System clock source, D2 = 1
  - Interrupt generation disabled, D1 = 0
- Enabling the SysTick (D0 = 1) will start the counter on the next clock cycle. It will continue to recount on its own as long as it is enabled

Determining if Count is Reached

- Continuously check the COUNTFLAG of the STCSR (D16) with a loop
  - Known as polling. Polling can potentially block other code

Checking for the raised COUNT flag

Steps to Use SysTick

1. Set the LOAD value for the number of counts
2. Set the CTRL for system clock, no interrupt generation, and initially disabled
3. Whenever you want to count, enable SysTick
4. Poll COUNTFLAG until count is reached
5. Disable SysTick once count reached
6. To reuse SysTick
   - Place LOAD value
   - Enable SysTick

Example

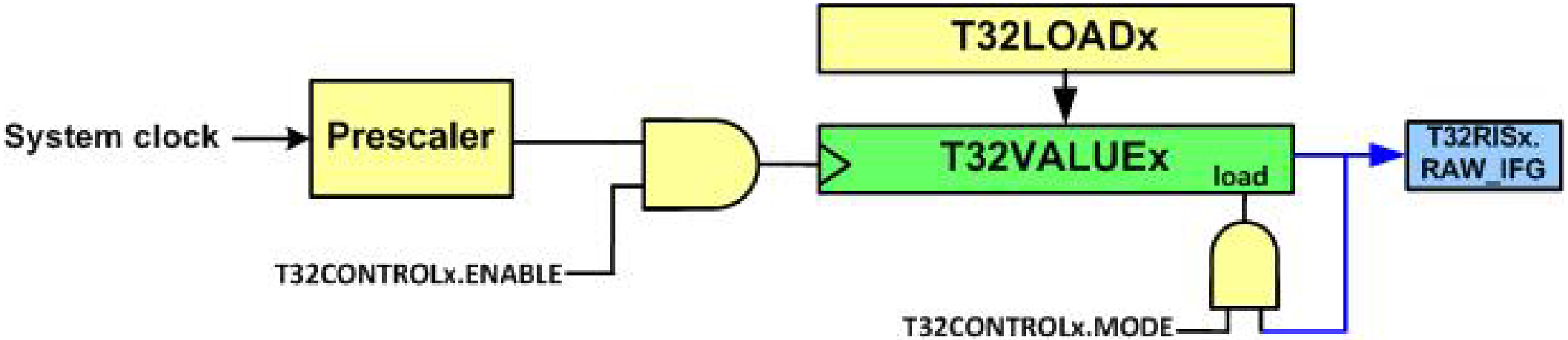
```
//Set up Port 1 Pin 0
P1 ->SEL0 =0x00;
P1 ->SEL1 = 0x00;
P1 ->DIR |=0x01;

//set up SysTick
SysTick ->LOAD = 9000000-1; //delay of 3 seconds
SysTick ->CTRL |=0x4; //system CLK, no interrupt, and disabled

P1 ->OUT |=0x01; //Turn on Pin 0
SysTick ->CTRL |=0x1; //Enable SysTick

while((SysTick ->CTRL & 0x10000)==0) //while COUNTFLAG not set
{
  //wait
}

P1 ->OUT &=~0x01; //Turn off Pin 0
SysTick ->CTRL &=~0x1; //disable SysTick
```

(0x4 = 0100)

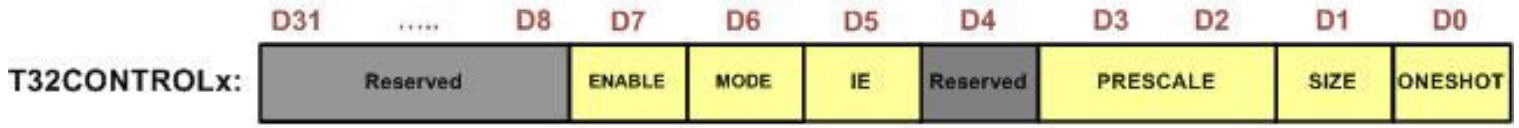
(0x10000 = 1 0000 0000 0000 0000

Timer 32

- Similar in use to SysTick, but is 32 –bits in size and has the following additional options
  - Frequency divide (prescale) system clock
  - Run mode select
  - One-shot enable



Timer 32 Control Register

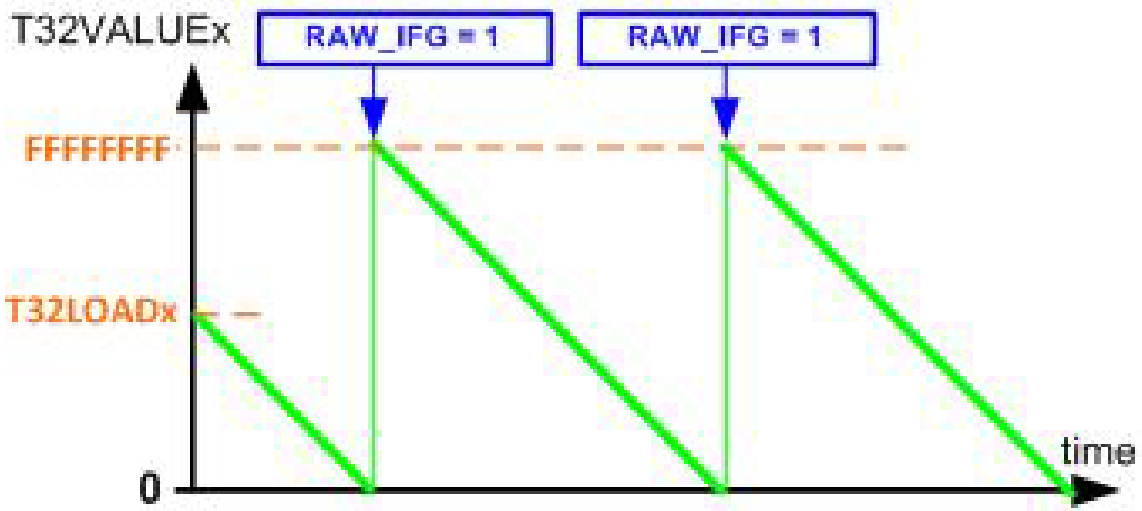


bit	Name	Description
7	ENABLE	Enable (0: the timer is disabled, 1: enables timer to begin counting down)
6	MODE	Mode bit 0: Free-running mode (The timer rolls over to its maximum value) 1: Periodic mode (The timer is reloaded with the value of the T32LOADx register)
5	IE	Interrupt Enable bit 0: Timer interrupt disabled 1: Timer interrupt enabled
3-2	PRESCALE	Prescale bits 00: clock is divided by 1 01: clock is divided by 16 10: clock is divided by 256 11: Reserved
1	SIZE	Selects 16-bit or 32-bit counter operation 0: 16-bit counter 1: 32-bit counter
0	ONESHOT	Selects one-shot or wrapping counter mode: 0: wrapping mode (The timer continues counting when it reaches to zero) 1: one-shot (The timer stops when it reaches to zero)

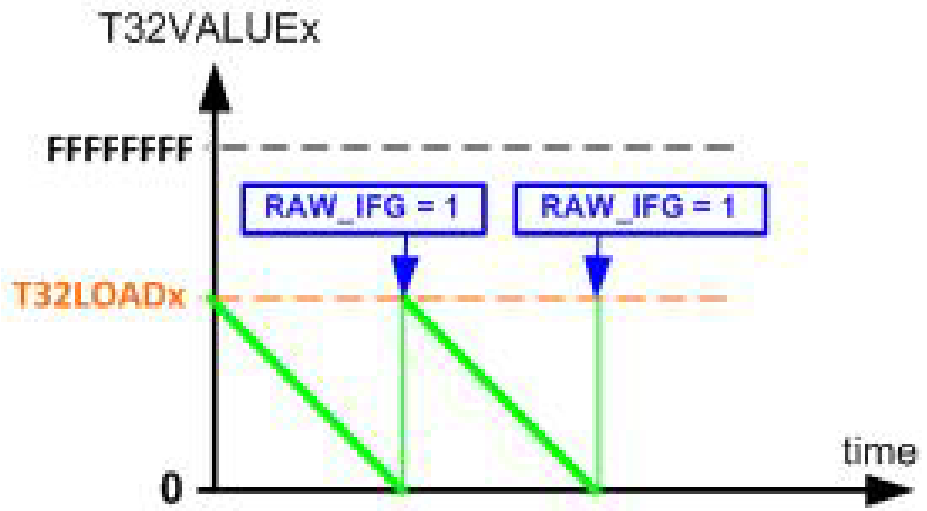
```
TIMER32_1 ->CONTROL
TIMER32_2 ->CONTROL
```

Timer 32 Modes

- Free running resets to 0xFFFFFFFF when counter reaches 0
- Periodic resets the TIMER32 Load register value when counter reaches 0



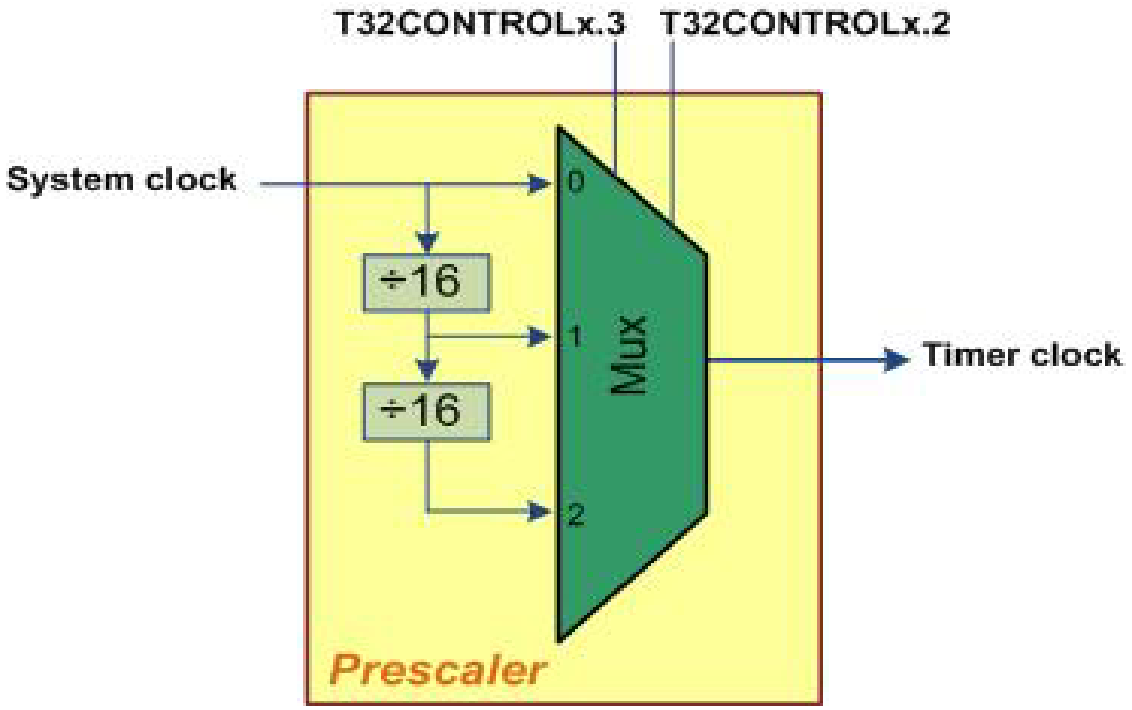
(a) Free running (MODE = 0)



(b) Periodic (MODE = 1)

Timer 32 Prescale

- Divides the system clock by 16 or 256

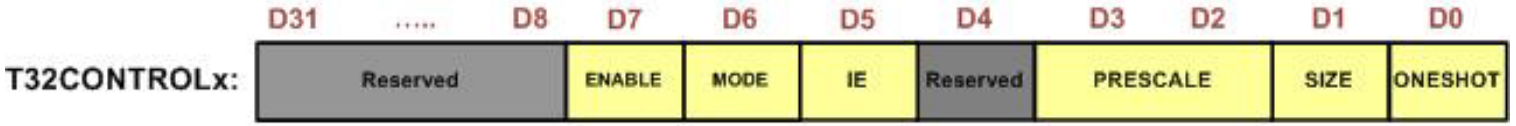


One Shot vs. Wrapping

- One shot will automatically stop the timer when it reaches 0
  - Will not reload and count down again
  - Do not need to reenable timer
- Wrapping will reload the timer and restart another countdown

Configuring Timer32

- Common configuration
  - Periodic Mode
  - No prescale
  - 32-bit size
  - Wrapping Mode



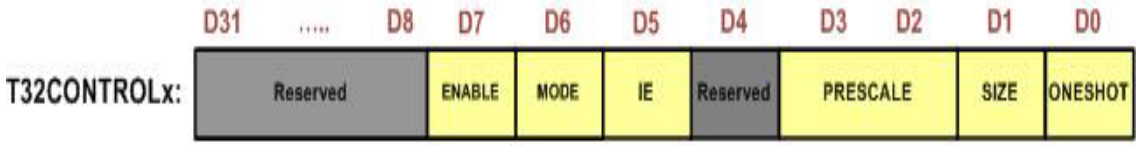
Determining if Count Reached

- When the timer reaches 0, the first bit in the T32RIS register goes to 1. It is 0 otherwise
  - This must be reset back to 0 using the T32INTCLR register

Using Timer32

1. Set the LOAD for the number of counts
2. Set the CONTROL for mode, prescale, bit size, and one-shot or wrapping. Initially disabled.
3. Whenever you want to count, enable Timer32
4. Poll the RIS if it is 1 to determine if count is finished
5. Set INTCLR to 0 when count is finished
6. Disable the timer (if not in one shot mode)
7. To reuse Timer32
   - Place LOAD value
   - Enable Timer

Example




```
//Set up Port 2 Pin 0
P2 ->SEL0 =0x00;
P2 ->SEL1 = 0x00;
P2 ->DIR |=0x01;

//set up Timer32
TIMER32_1 ->LOAD = 3000000-1;
TIMER32_1 ->CONTROL |=0x42; //periodic mode, no interurpt or prescale, 32-bit size, wrapping mode
P2 ->OUT |=0x01; //turn on pin 0
TIMER32_1 ->CONTROL |=0x80; //enable timer
while((TIMER32_1 ->RIS & 1)!=1) //while count not done
{
  //wait
}
TIMER32_1->INTCLR &=~0x01; //set INTCLR to 0 after count is reached
P2 ->OUT &=~0x01; //Turn off Pin 0
TIMER32_1 ->CONTROL &=~0x80; //disable Timer32
```

(0x42 = 0100 0010)

(0x80 = 1000 0000)

One Shot Example

(0x43 = 0100 0011)

```
TIMER32_1 ->CONTROL |= 0x43; //configure one shot mode
TIMER32_1 ->LOAD = 3000000-1; //load value into timer
TIMER32_1 ->CONTROL |=0x80; //start timer
while((TIMER32_1 ->RIS & 1) ==0) //wait
{
  //wait
}
TIMER32_1->INTCLR = 0; //clear count flag
```

*To reuse timer, place a value in the LOAD register.

Timer A

- Will not be covering TimerA
- Most advanced and versatile timer on the MSP432
- Common uses of TimerA
  - Capture and compare – determine frequency of digital signal
  - PWM for motor control
  - Count number of occurrences on digital pins