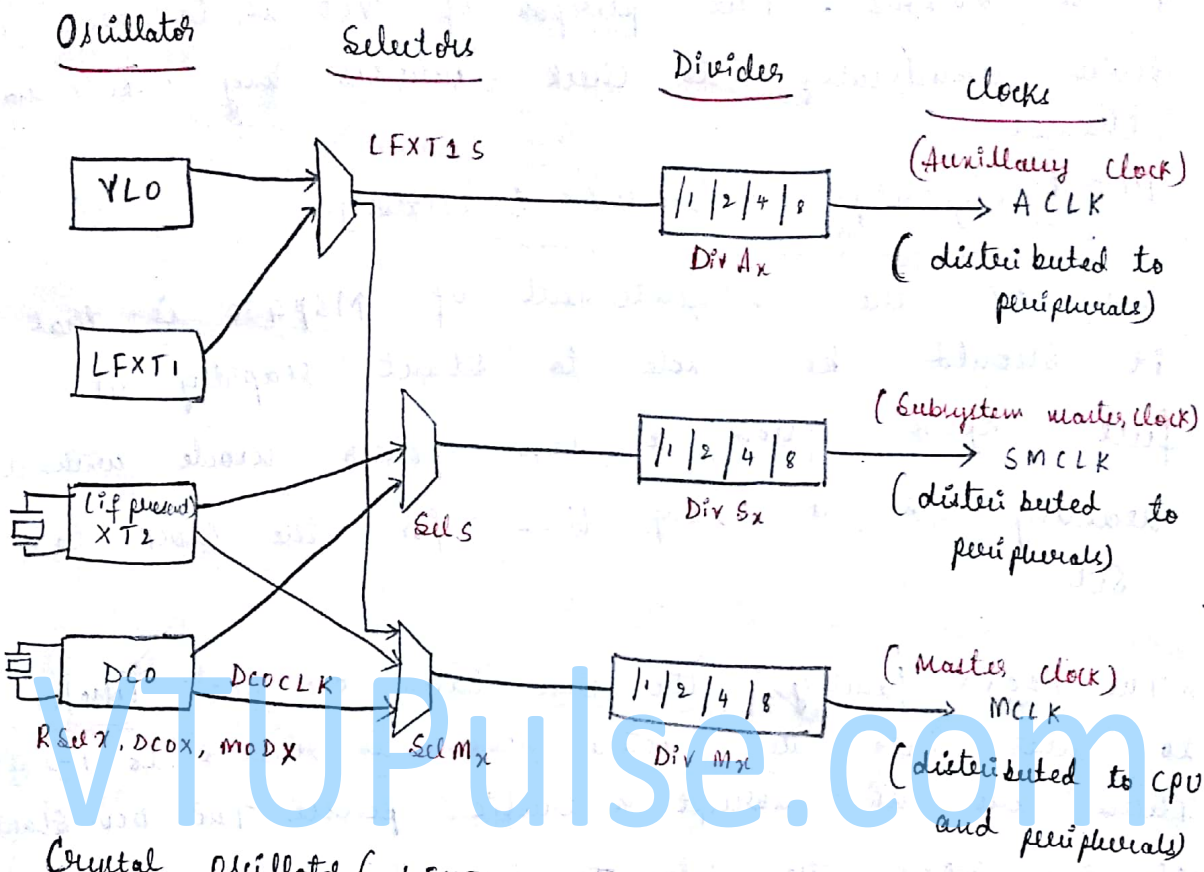
10/10/17

Module 3: Clock System, Interrupts and operating modes

Clock System: A clock module is used in microcontrollers to drive CPU and peripherals. The basic clock module + (BCM+) for the MSP430F20XX is as shown.



Crystal Oscillator (LFXT1 and XT2):

Crystals are used when an accurate and stable frequency is needed. Crystals are cut from high quality quartz with specific orientations to give them high stability.

Oscillator circuits are integrated into MSP430 and the Crystal should be connected to pins X_{IN} and X_{OUT}. This circuit is also known as pierce oscillator which also requires capacitors one to ground from each pin.

Internal very low power low frequency oscillator:

The VLO is an internal RC oscillator that runs at around 12 KHz and can be used instead of LFXT1 in some devices.

It Saves the cost and Space required for a Crystal and also reduces the current drawn.

This Comes at the cost of accuracy and stability. The frequency range of VLO in F2013 is around 4 to 20 KHz. The purpose of VLO is to make the device periodically to check whether any inputs have changed.

DCO [Digitally Controlled Oscillator]

One of the requirement of MSP430 is that it should be able to start rapidly at full Speed from a low power mode without waiting for a long time for the clock to Set.

In The F20xx family the time taken by DCO 11/10/17 to enter into the active mode is reduced to 1-2 μs. There are no abrupt or erratic pulses. The DCO starts Cleanly after the delay The frequency Can be Controlled through a Set of bits in the modules registers at 3 levels. The first 2 levels Set the DCO to a Constant frequency

RSel x : This Selects one of the 16 values Coarse ranges of frequency. The overall range is about 0.09 - 20MHz

DCOx : Selects one of the 8 Steps within each range. Each Step increases the frequency by about 8%

MODx : Finer Control of the average frequency is obtained by modulating the frequency of oscillator b/w Selected Value of DCO and the next Step up DCO+1. Each period of 32 Clock Cycles Contains MOD Cycles with the higher frequency.

Given by DCO+1 and 32-MOD cycles with lower frequency given by DCO. Thus the average period over this 32 cycles is

$$T_{average} = \frac{MOD * T_{DCO+1} + (32 - MOD)\, T_{DCO}}{32}$$

$$f = \frac{1}{T}$$

$$frequency = \frac{32}{MOD\left(\frac{1}{f_{DCO+1}}\right) + \frac{(32 - MOD)\, T_{DCO}}{f_{DCO}}}$$

$$\boxed{frequency = \frac{32\, f_{DCO}\, f_{DCO+1}}{MOD\, f_{DCO} + (32 - MOD)\, f_{DCO+1}}}$$

Control of the clock module through Status register.

The clock module is controlled by 4 bits in the Status register as well as its own peripheral registers. If all bits are cleared the device works / operates in full power i.e active mode. To make the device work in different low power mode these bits have to be set accordingly.

CPUOFF: Setting this bit disables master clock which stops the CPU and any peripherals that use master clock.

SCG1: System clock generator
Setting this bit disables Subsystem master clock (SMCLK) and peripherals that use it.

SCG0: Setting this bit disable the DC generator for the DCO (it disables the frequency locked loop FLL in MSP430X4XX)

OSCOFF: Disables VLO and LFXT1. by setting this bit.

INTERRUPTS

12/10/17

→ Interrupts are commonly used for a range of applications.

* Urgent / Immediate tasks that must be executed at the higher priority than the main code.

* Infrequent tasks such as handling slow inputs from humans. This saves the overhead of regular polling.

* Waking the CPU from sleep mode.

* Making calls to an operating system.

→ ISR / IH : The code to handle an interrupt is called an interrupt handler or interrupt service routine. Each interrupt has an flag which is set when the condition for interrupt occurs.

→ Most interrupts are maskable i.e they are effective only if GIE bit is set in the Status register.

→ Both the enable bit in the module and GIE must be set for interrupts to be generated.

→ The non maskable interrupts can't be suppressed by clearing GIE.

→ MSP430 uses vectored interrupts wherein the address of each ISR i.e. the vector is stored in a vector table at defined address in memory.

→ Each vector is associated with unique interrupt but some source share a vector.

→ Each interrupt vector has a distinct priority which is used to select the interrupt with highest priority if more than one interrupt is active.

→ priorities are fixed and can't be changed by the user.

A higher vector address indicates a higher priority. The reset vector has address 0XFFFE which gives it the highest priority.

→ Interrupts must be handled in such a way that the code that was interrupted can be resumed or continued without any error. i.e the value in CPU registers must be restored. There are 2 ways for this.

1> Copies of all the registers are saved on the stack automatically as soon as an interrupt has to be processed.

Disadvantage: Time required is more i.e response to the interrupt is delayed. An alternative to this is to use a second set of registers.

2> The opposite approach is for the hardware to save only the absolute minimum which is the return address in the program counter PC. This is faster and it is upto the user to save and restore the values of critical registers like status registes.

Inupt

** What happens when an interrupt is requested

A sequence of operations occurs when an interrupt is raised.

It starts when a flag bit is set in the module when the condition for interrupt occurs. (eg: TAIFG i.e timer A interrupt flag is set when counter TAR returns to zero). This is passed to the logic that controls interrupts if the corresponding enable bit is also set. This is pass The request for an interrupt is finally passed to the CPU if GIE bit is set. The hardware then performs the following steps to launch the ISR.

1) Any currently executing instruction is completed if the CPU was active when the interrupt was requested. MCLK is started if the CPU was off.

2) The PC which points to the next instruction is pushed on to the stack (return address).

3) The status register SR is pushed on to the stack.

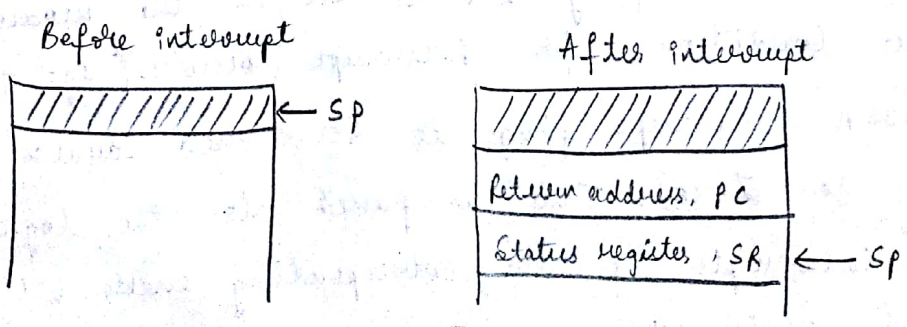
4) The interrupt with the highest priority is selected if multiple interrupts are waiting for service.

5) The interrupt request flag is cleared automatically for vectors that have a single source. Flags remains set for servicing by software if the vector has multiple sources which applies to TAIFG.

6) The status register is cleared which has 2 effects.
   i) Further maskable interrupts are disabled because the GIE bit is cleared but non-maskable interrupts remain active.
   ii) It terminates the low power modes. i.e the device operates at full power.

7) The interrupt vector is loaded into the PC and the CPU starts to execute ISR at that address.

This sequence takes 6 clock cycles in MSP430 before the ISR commences. The stack at this point is as shown.



The delay b/w an interrupt being request and start of the ISR is called as "latency"

If the cpu is already running it is given by the time taken to execute the current instruction + the 6 cycles needed to launch the interrupt sequence. This should be calculated for the slowest instruction to get the worst case delay.

Format I instructions take upto 6 clock cycles.

Therefore the overall latency is 12 clock cycles.

The time required to start MCLK replaces the duration of current instruction execution if the device was in a low power mode.

An ISR must always finish with the RETI instruction (return from interrupt instruction) which does the following

i) The status register is popped from the stack. All previous settings of GIE and mode control bits are now in effect irrespective of the settings during ISR. This re-enables maskable interrupts and restores the previous low power modes if any.

ii) The PC is popped from the stack and execution continues from the point where it was interrupted. or the cpu stops and the device returns to its low power mode if only instruction before the interrupts. This takes 5 cycles. The stack pointer is restored to its state.

Non maskable interrupts 13/10/17

All the non-maskable interrupt source share a single vector, which have the highest address. Therefore non-maskable interrupts have highest priority except for the reset vector. 3 modules can request non-maskable interrupt.

1) Oscillator fault (OFIFG): This flag is set by a power up clear (PUC) as a warning that the oscillator may not get destabilized.

Therefore this interrupt should not be enabled until the oscillator is running correctly.

2) Access violation to flash memory (ACCVIFG)

3) An active edge on the external pin $\overline{RST}$/NMI pin if it has been configured for interrupts rather than reset. The function of $\overline{RST}$/NMI is configured in the control register for watchdog timer module (WDTCTL). By default this pin is active low $\overline{RST}$ pin. This can be switched to a non-maskable interrupt input with WDTNMI bit.

Interrupt Service Routine

Low power modes of operation

MSP430 was designed for low power applications and this is reflected in a range of low power modes of operation. There are 5 modes in total of which 2 are rarely used in current devices.

The modes are for the F2013 with $V_{CC} = 3V$, DCO running at 1MHz and LFXT1 at 32kHz from a crystal.

Active mode: The CPU, all clocks and enabled modules are active current $I \approx 300\ \mu A$.
The MSP430 starts up in this mode which must be used when the CPU is required. An interrupt automatically switches the device to the active mode. The current can be reduced by running the MSP430 at the lowest supply voltage consistent at with the frequency of master clock. Vcc can be lowered upto 1.8V for $F_{DCO} = 1MHz$. giving $I \approx 200\ \mu A$

1) **low power mode 0: (LPM0)**

The CPU and MCLK are disabled, SMCLK and ACLK remain active, current $I \approx 85\ \mu A$. This mode is used when the CPU is not required but some module require a fast clock from SMCLK and DCO.

2) **low power mode 3: (LPM3)**

The CPU, MCLK, SMCLK and DCO are disabled. only the ACLK remains active, current $I \approx 1\ \mu A$.
This is the standard low power mode where the device must wake itself at regular intervals and therefore need slow clock. This mode is also require

if the MSP430 must maintain real clock & time clock (RTC). The current can be further reduced to 0.5 μA by using VLO instead of external crystal in MSP430F5xxx/F6xxx if f_{ACLK} need not be accurate

3) Low power mode 4 (LPM4): CPU and all clocks are disabled. The current $I \approx 0.1\,\mu A$. The device can be awakened only by an external signal. This is called as RAM retention mode. 14/10/17

When a device is in low power mode, it is described as sleeping and waking the device indicates it returns to its active mode. To enter into low power mode 3, the instruction is BIS.W #LPM3, SR the symbol LPM3 is defined in the header file. The MSP430 can be awakened only by interrupt. So these must be enabled by setting GIE bit in the Status register.

Waking from a low power mode

An interrupt is needed to awaken the MSP430. The processor handles an interrupt from low power mode in the same way as in active mode. The only difference is that master clock must be first started. So that the CPU can handle the interrupt. MCLK is automatically started by hardware for servicing the interrupts and requires no intervention from the programmer.

Returning from a low power mode to main function

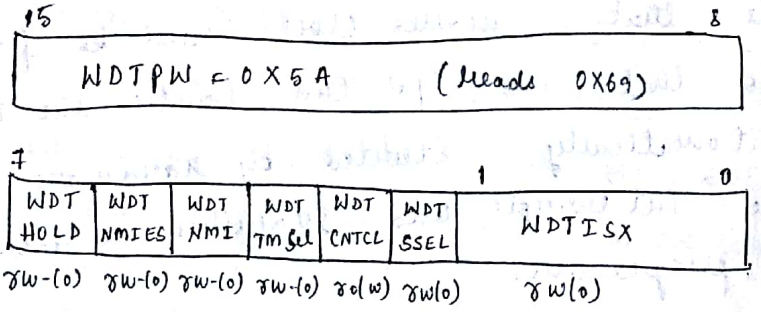
To return to the function that puts the device into low power mode, we must clear all the low power bits in the Status register before it is restored at the end of interrupt service routine.

Timers: Watchdog timer (WDT)

The main purpose of the watchdog timer is to protect the System against failure of the Software Such as program becoming trapped in an unintended infinite loop. If WDT is left to itself it Counts up to a max value and resets the MSP430 when it reaches its limit Therefore the code must keep Clearing the Counter before the limit is reached to prevent reset.

The operation of WDT is Controlled by the 16-bit register named WDTCTL (watchdog timer Control register) It is protected against accidental writes by requiring the password WDTPW = 0X5A in upper byte. A reset will occur if the value with an incorrect password in written to WDTCTL. Reading WDTCTL register returns 0X69 will be upper byte. Therefore reading WDTCTL and writing the value back 16/10/17 violates the password and Causes the reset.

The lower byte of WDTCTL Contains the bits that Control the operation of WDT.



WDT Hold=1, disable the WDT when WDT hold = 1.

WDT NMIES (watchdog timer non maskable interrupt edge Select) when

WDT NMI = 1

0 → Selects raising edge of NMI

1 → Selects falling edge of NMI

WDTNMI

Selects $\overline{RST}$/NMI function
- 0 → $\overline{SRTS}$ function
- 1 → NMI

WDTTMSEL → WDT timer mode
- 0 → Supervised mode
- 1 → Interval timer

WDTCNTCL → WDT Counter clear
- 0 → No action
- 1 → Clears WDTCNT register.

WDTSSEL → WDT Source Select (Select Clock pulse)
- 0 → SMCLK
- 1 → ACLK

WDTISX → WDT Interval Select
- 00 → Clock Signal / 32,768 (2^{15})
- 01 → Clock Signal / 8192 (2^{13})
- 10 → Clock Signal / 512 (2^9)
- 11 → Clock Signal / 64 (2^6)

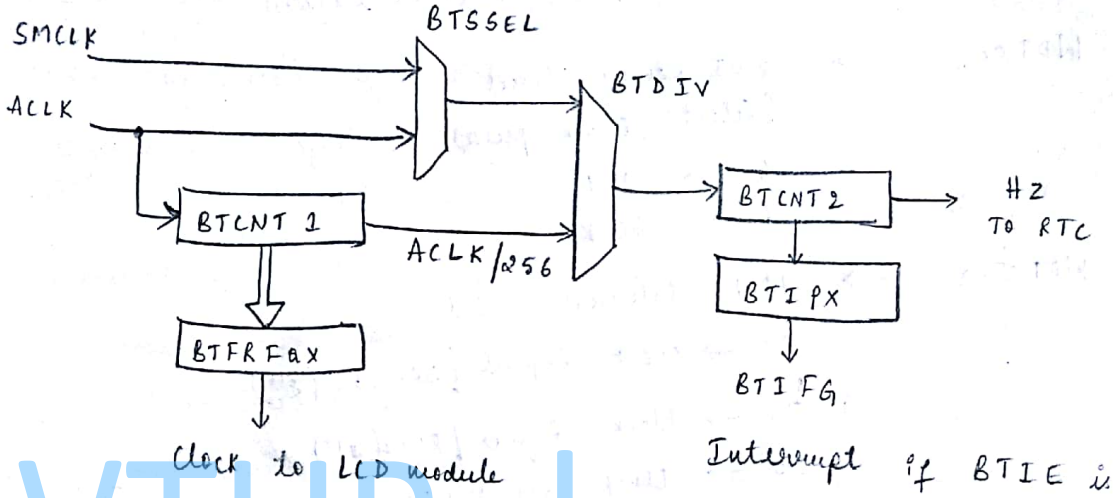
Failsafe Clock Source for WDT+ Refer notes

Watchdog timer as interval timer.

WDT can be used as interval timer if its protection function is not desired. The WDTTMSEL bit in WDTCTL register. Should be set for interval timer mode. The time periods are same as before and the WDT interrupt flag (WDTIFG) is set when the time reaches its limit but no reset occurs. The Counter rolls over and restarts from zero. An interrupt is requested if WDTIE bit IE register is set if GIE is set. The WDTIFG is automatically cleared when the interrupt is serviced.

Basic timer 1:

It is present in all MSP430 X F4XX families. It provides clock to the LCD module and generates periodic interrupts. There are two 8 bit Counters in basic timer 1 which can be used independantly or Cascaded for longer intervals. The block diagram of basic timer 1 is as shown.



Real time Clock (RTC):

26/10/17

A RTC Counts Seconds, minutes, hours, days, months and years. It can be Configured in Calender mode by Setting RTCMODEX bits to 11.

It can be

In the Control register RTCCTL as shown

7	6	5	4	3	2	1	0
RTCBCD	RTCHOLD	RTCMODEX	RTCTEVX			RTCIE	RTCFG

RTC BCD: Selects BCD Counting for RTC, applies to Calender mode.

0 → Binary

1 → BCD.

RTC HOLD: 0 → Operational (Calender mode & Counter mode)

1 → Disabling RTC

RTCMODEX : 11 → Calender mode

RTCTEVX : RTC time event

In Calender mode . 00 → minute change
01 → hour change
10 → Everyday at midnight (00:00)
11 → Everyday at noon. (12:00)

In Counter mode. 00 → 8-bit overflow
01 → 16-bit overflow
10 → 24-bit overflow
11 → 32-bit overflow

RTCIE : RTC interrupt enable

If the bit is set. then RTC module can send an interrupt

RTCFG : RTC interrupt flag

The current time and date are held in a set of registers that contains the following bytes

i) Second - RTCSEC
ii) Minute - RTCMIN
iii) Hour - RTCHOUR
iv) Day of week - RTCDOW
v) Day of Month - RTCDAY
vi) Month - RTCMON
vii) Year - RTCYEARL
viii) Century - RTCYEARH.

The registers are arranged in pairs So that they can be accessed as words.

16
RTCYEAR => RTCYEARH : RTCYEARL

RTCTIMO => RTCMIN : RTCSEC

The clock need 1 #2 inputs taken from basic timer 1. RTC module has an interrupt flag RTCFG and the corresponding enable bit RTCIE. The flag is set every minute, every hour, everyday at midnight or everyday at noon depending on RTCTEVx. Interrupts are generated by real time clock if RTCIF is set. The interrupt vector is shared with BT1. Both BTIFG and RTCFG flags are set when interrupt occurs and is cleared automatically when it is service.

31/10/17

Imp ** Timer A :

It is the general purpose timer and it is included in all the devices. It consists of 2 main parts.

i> Timer block : It is based on TAR register (Timer A register).

The TAIFG is set when Counter #0. rolls over to 0.

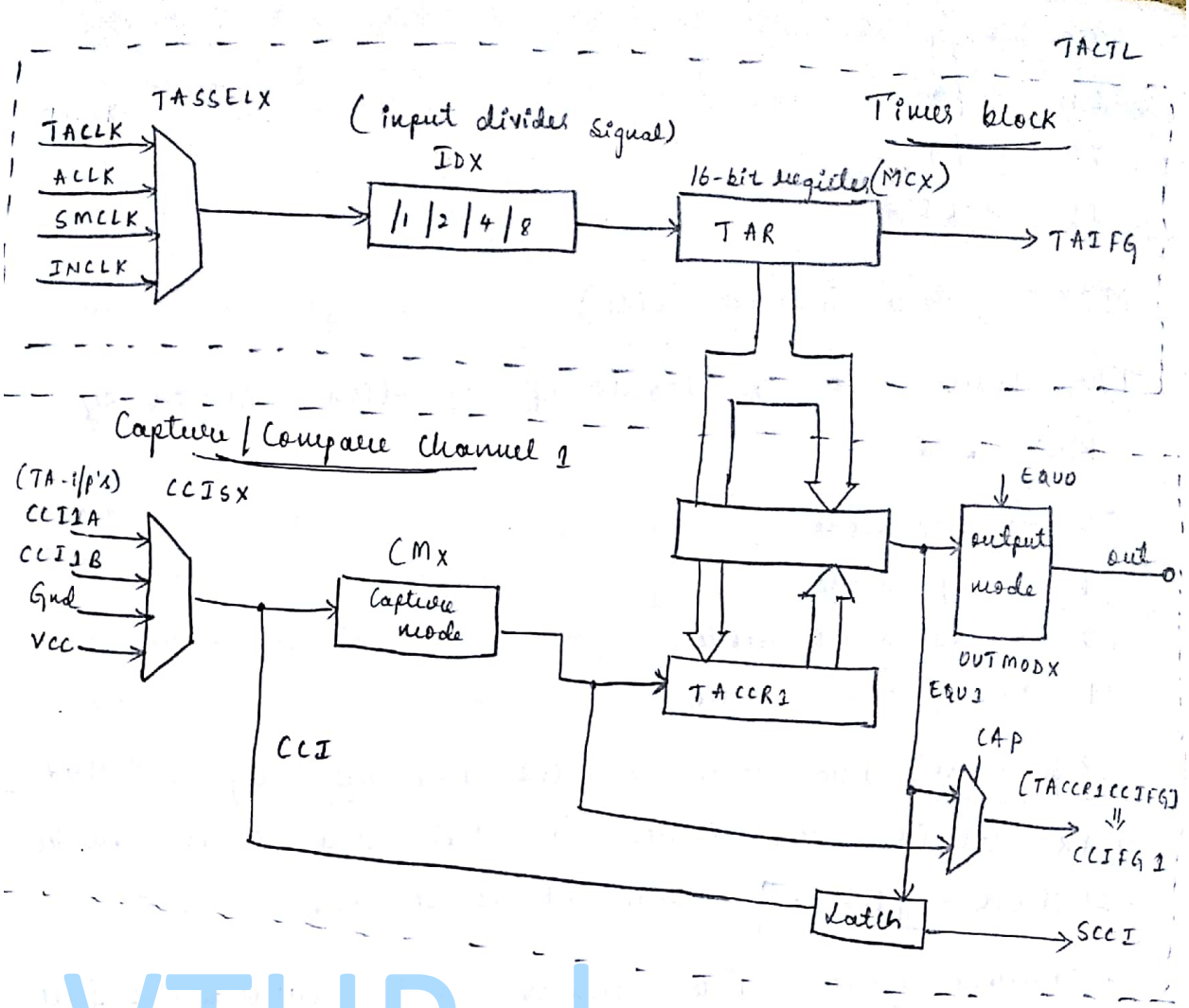
ii> Capture Compare Channels :

Capture : Capture an input or record the time (VALUE in TAR) at which the input changes in TACCRn (Timer A Capture Compare register).

Compare : The current value of TAR is compared with the value stored in TACCRn and update an output when they match.

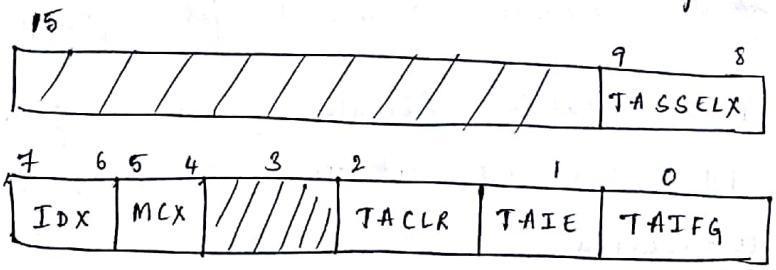
There are 3 Capture Compare channels and all channels within timer A share the same timer block Therefore there is only one TAR So that the actions performed by different channels are synchronized.

The block diagram of timer A is as shown.



TACCTLn

Timer block: It consists of 16-bit timer register TAR controlled by TACTL register as shown



The Clock can be chosen from 4 sources using TASSELX bits

- 0 0 → TACLK → External clock
- 0 1 → ACLK → Internal clock and slow clock (32KHz)
- 1 0 → SMCLK → Internal clock and fast clock (MHz)
- 1 1 → INCLK → External clock and can be connected through an inverter to the pin of TACLK so that INCLK = $\overline{\text{TACLK}}$

IDX: The frequency of input signal or incoming clock can be divided by 2, 4 or 8 by configuring the IDX bits.

00 → /1
01 → /2
10 → /4
11 → /8

MCX: (Mode Control bits)

The timer has 4 modes of operation selected by MCX bits.

00 → Stop mode
01 → up mode
10 → Continuous mode
11 → up down mode.

Stop mode: The time is halted and all register including TAR retain their values so that the timer can be restarted from where it is stopped.

Continuous mode: The Counter counts through its full range from 0000 to FFFFH, after which it overflows and rolls back to zero. Its period is $2^{16} = 65536$ convenient for capturing inputs.

up mode: The Counter counts from 0 upto the value in TACCR0. It returns to zero on the next clock transition.

Period is TACCR0+1.

eg: If TACCR0 = 4, Sequence is 0, 1, 2, 3, 4, 0
period = 4+1 = 5.

And it is used for pulse width modulation.

up down mode: The Counter counts from 0 upto TACCR0 and counts down again to 0 and repeats.

The period is 2 X TACCR0

eg: If TACCR0 = 3, Sequence is 0, 1, 2, 3, 2, 1, 0
period = 2 X 3 = 6

TACLR : The count in TAR and the divider can be cleared by setting the TACLR bit in TACTL register.

TAIFG : The flag TAIFG in TACTL is set when timer rolls over to zero and the maskable interrupt is requested if the TAIE bit is set.

Capture Compare Channels

Timer A has 3 channels of which channel 0 is used to set the limit in & for counting in up and up down mode. Each channel is controlled by a register TACCTLn which is a 16 bit register as shown

15–14	13–12	11	10	9	8
CMx	CCISx	SCS	SCCI		CAP

7–5	4	3	2	1	0
OUTMODx	CCIE	CCI	OUT	COV	CCIFG

The central feature of each channel is capture compare register, TACCRn.

Capture mode : ⊛ An event can be a rising edge, falling edge or either edge on the input according to capture mode bits CMx.

⊛ The CCISx bits selects the input to be captured.

⊛ SCS – Synchronization of capture signal

⊛ SCCI – Synchronization of capture compare input.

⊛ CAP – Capture compare bit 0 → compare, 1 → capture

The state of the selected i/p can be read in the CCI bit of TACCTL.

COV : The Capture overflow bit COV is set if another Capture occurs before TACCRn has been read following the previous event

00 → CCI1A
01 → CCI1B
10 → Gnd } Capturing for Software
11 → Vcc }