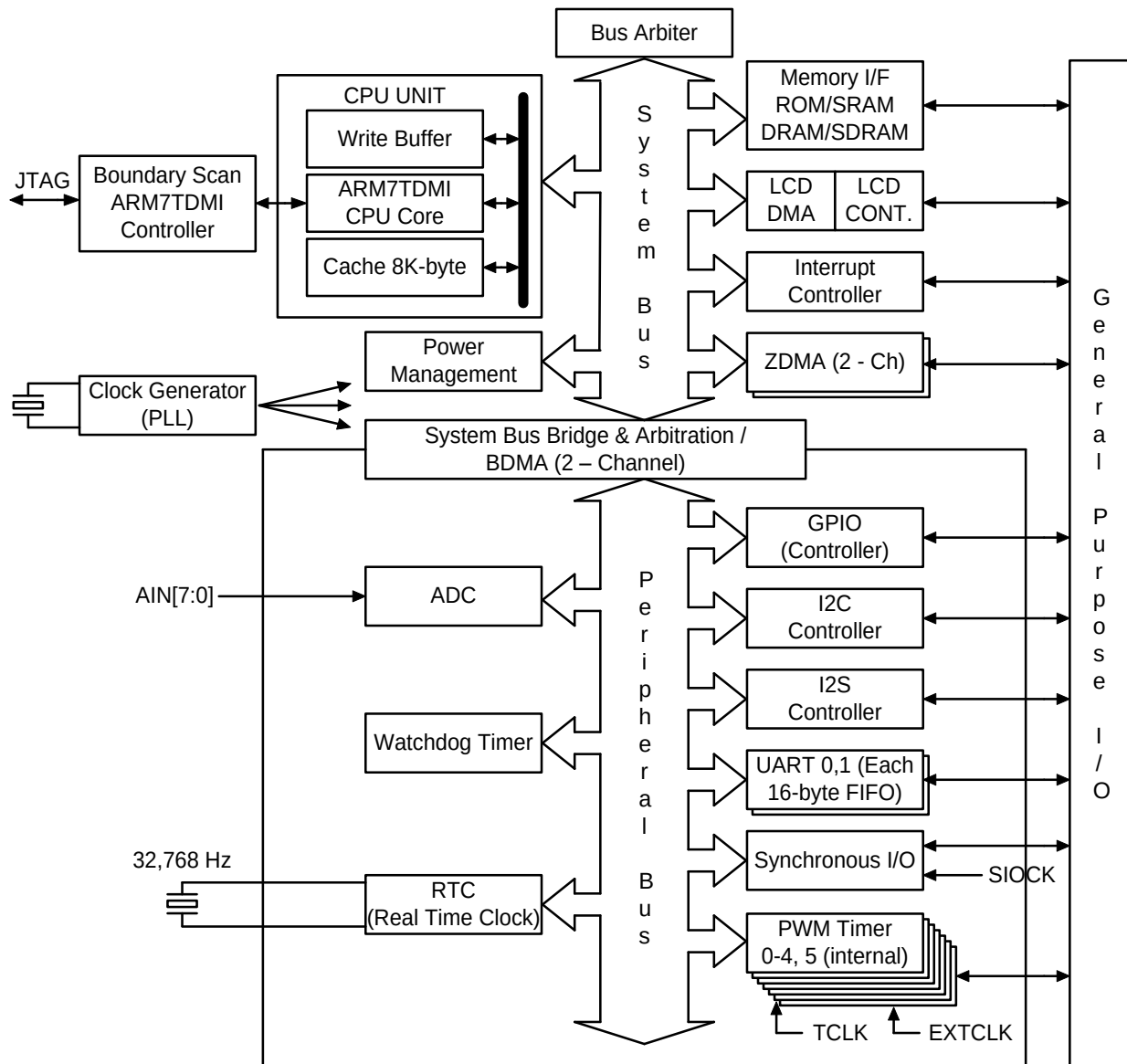


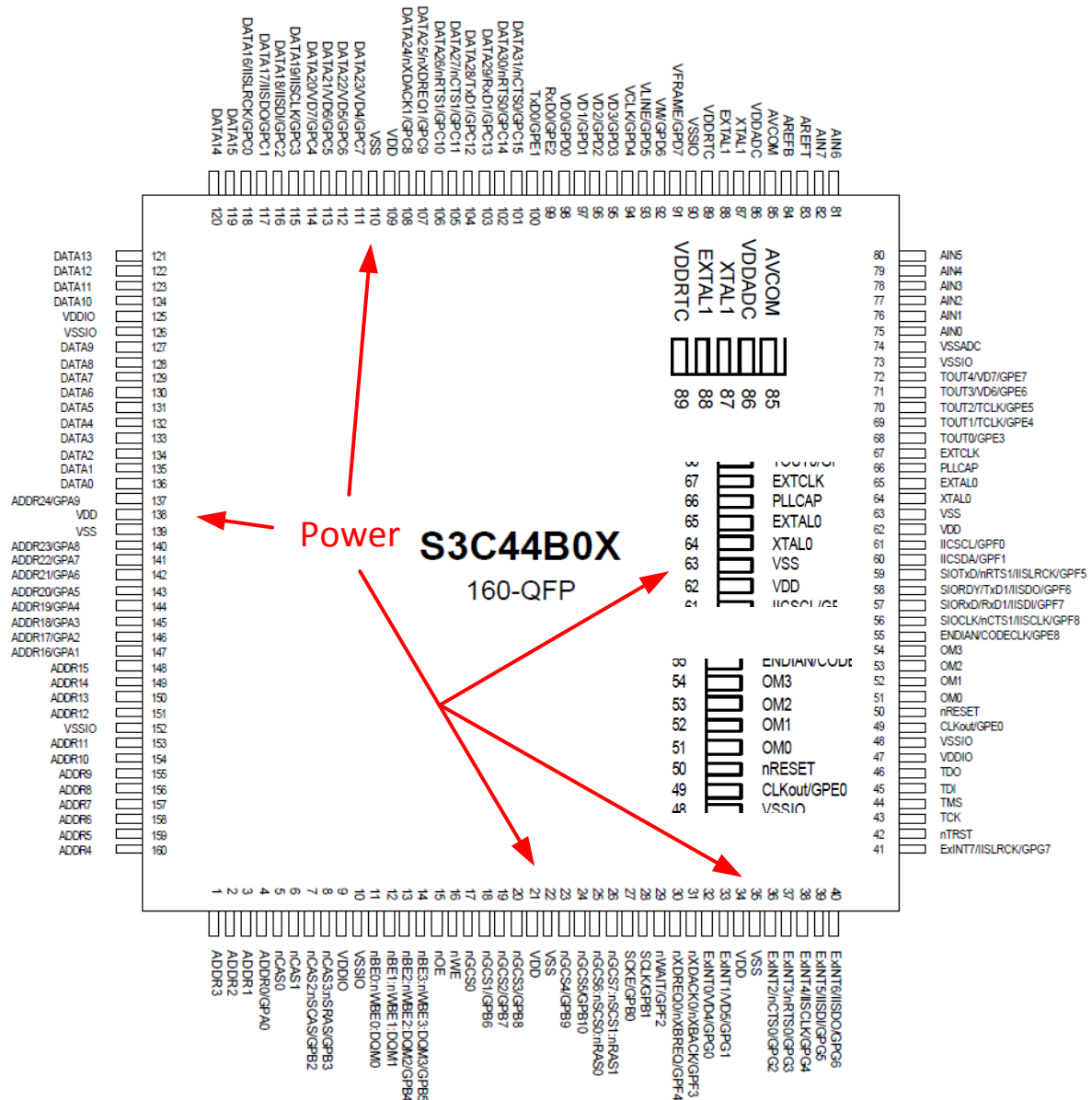
Chapter 6 Clock, Reset and Timers

6.1 S3C44B0X Internal Architecture:



High speed components are connected to the **System Bus (high speed)** whilst slow I/O devices are connected to the **Peripheral Bus (slow speed)**.

A single clock source is used as a reference and from which different clock frequency can be derived for different components.



How to make it work ?

Essentials things:

- Power (distributed power input)
- Clock (Crystal or oscillator)
- Reset circuit
- Memory (internal or external; flash and RAM)
- Reset handler (Boot program)

Flash is for storing boot program; RAM is for program/data

THEN YOUR PROGRAM !!!

6.2 CPU Clock

- A microprocessor based system is a synchronous system
- All activities are synchronized with a master clock signal - CPU clock (the clock source of the entire system)
- A *Phase Lock Loop* circuit is used to generate a high frequency master clock from a low frequency source.
- Timing circuits of different peripheral devices will generate different clock frequencies for different requirement.
- Advanced CPU, like SoC (System on Chip), has already integrated with different peripheral devices (or controllers) on-chip, all the required clock signals are generated by an internal timing generator.
- Wrong concept: the clock rate determines the performance of the CPU.
- For example: a system with 1.6 GHz CPU is not necessarily double the performance of a system using CPU runs at 800 MHz.
- Reasons: Architecture difference – CISC versus RISC
 - Pipeline
 - System bus speed
 - Cache size
 - SRAM/SDRAM/DDR/DDR2 and speed
 - Harddisk speed



Crystals (2 pins):

1. HC-49/SD crystal (SMD)
2. HC-49/US crystal
3. HC-49/U crystal

Oscillators (4 pins):

4. DIL-14 "Full size" metal case
5. DIL-8 "Half size" metal case
6. DIL-8 "Half size" plastic case
7. 5x7mm metal case (SMD)

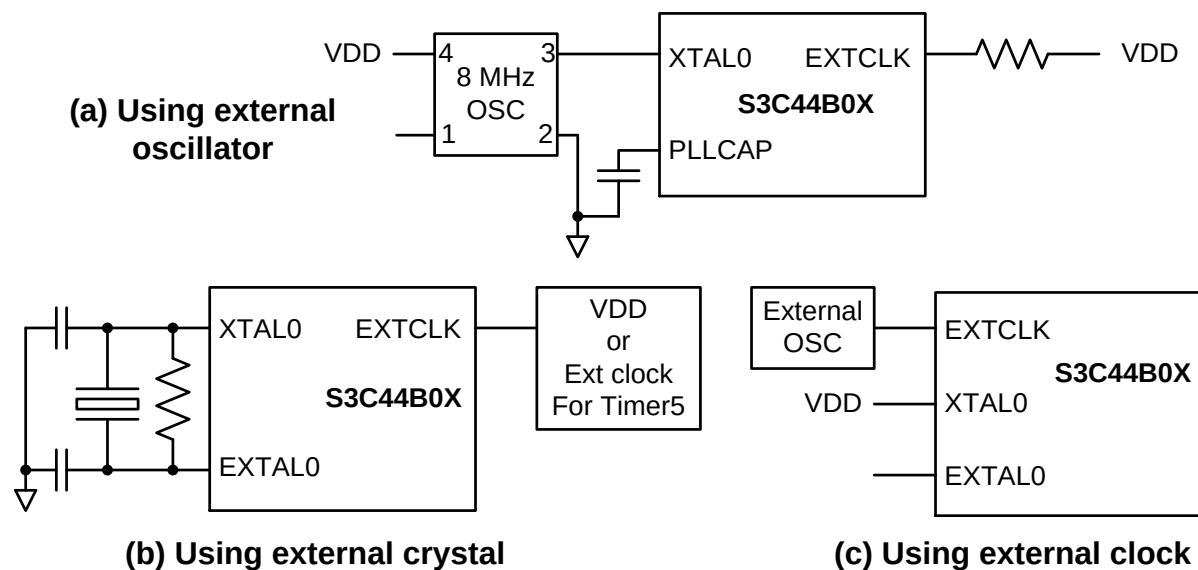
The 4 pins are typically as follow:

- | | |
|-----------|-------------------------------|
| 1. Enable | 3. Output |
| 2. GND | 4. VCC (typically 3.3V or 5V) |

Pin 1 – GND : disable oscillator output

Pin 1 – VCC (or floating) : enable oscillator output

Metal case – GND : for shielding

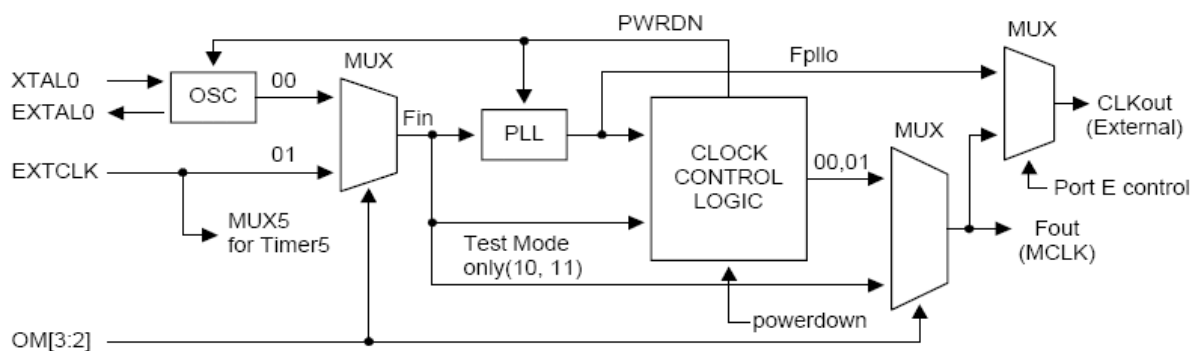


Main Oscillator Circuit Example

6.3 CPU Clock generator of S3C44B0X

6.3.1 Clock generator

- The clock generator can generate the required clock signals for the CPU as well as peripherals.
- It can be controlled to supply or disconnect the clock to each peripheral block by S/W, which will reduce the power consumption.
- The following is the block diagram of the clock generator.



- The main clock source comes from **an external clock** or **external crystal** connecting to an oscillator amplifier.
- The generator also has a PLL (Phase-Locked-Loop) which takes the low frequency input from either the oscillator amplifier or external clock to generate a high frequency clock required by S3C44B0X.
- The clock generator block has the logic to generate a stable clock frequency after a reset or a stop mode.

6.3.2 Clock source selection **at boot-up**

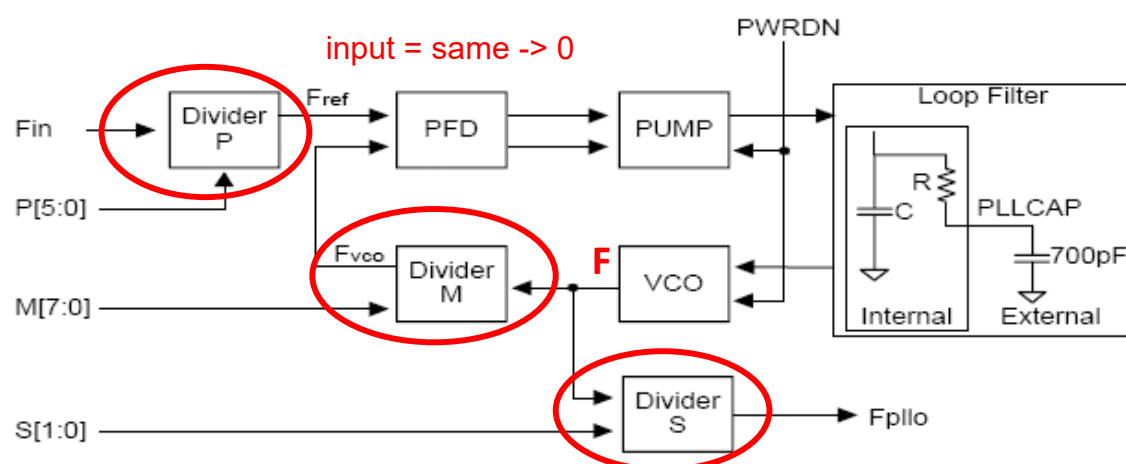
- When S3C44B0X boots up, a clock source is required and it is determined from the logic combination of **two mode input pins: OM[3:2]**.
- The OM[3:2] status is latched internally by referring to the logic level of OM3 and OM2 pins at the rising edge of nRESET.
- The clock selection is shown as follow:

Mode OM[3:2]	Clock Source	Crystal Driver	PLL starting state	Fout
00	Crystal clock	enable	Enable*	PLL Output*
01	Ext. Clock	disable	enable*	PLL Output*
10 or 11	Test mode			

** Although the PLL starts just after a reset, the PLL output cannot be used as Fout until the S/W writes valid settings to the PLLCON register. Before this valid setting, the clock from crystal oscillator or Ext. clock source will be used as Fout directly. Even if the user wants to maintain the default value of PLLCON register, the user should write the same value into PLLCON register. – i.e. at boot up, CPU will follow the clock source and changing clock can only be effective after writing to PLLCON register.*

6.3.3 PLL (Phase Lock Loop)

- The PLL circuit is used to synchronize the frequency and phase of an output signal with a reference input signal.



- VCO (Voltage Controlled Oscillator) generates an output frequency (F) proportional to an input DC voltage derived from a loop filter.
- $F_{ref} = \frac{F_{in}}{P}$, $F_{vco} = \frac{F}{M}$
- The PFD (Phase Difference Detector) monitors the phase difference between the F_{ref} and F_{vco} and generates a control signal when a difference is detected.
- According to the control signal, the PUMP circuit and loop filter will vary the DC voltage to alter the VCO output frequency F .
- If the frequency and phase are matched, the VCO control voltage remains unchanged, i.e.,

$$F_{ref} = F_{vco} \Rightarrow \frac{F_{in}}{P} = \frac{F}{M} \Rightarrow F = \frac{MF_{in}}{P}$$

- The master clock F_{pllo} is obtained by dividing the VCO output frequency by a factor of S .
- The output clock frequency F_{pllo} is related to the reference input clock frequency F_{in} by the following equation:

$$F_{pllo} = \frac{m \times F_{in}}{p \times 2^s}$$

$$m = M \text{ (the value for divider } M) + 8,$$

$$p = P \text{ (the value for divider } P) + 2$$

6.3.4 Setting up the CPU clock

- The master CPU clock frequency depends on the frequency of the source (external crystal, oscillator or clock) and the setting of PLL control register.
- The relationship of various parameters to set the CPU clock frequency are:

$$F_{pllo} = \frac{m \times F_{in}}{p \times 2^s}, \quad 20\text{MHz} \leq F_{pllo} \leq 66\text{MHz},$$

$$m = MDIV + 8,$$

$$p = PDIV + 2, \quad s = SDIV$$

Example:

If $F_{in}=14.318\text{Mhz}$ and $F_{out}=60\text{Mhz}$, we have:

$$MDIV = 59 = (111011)_2$$

$$PDIV = 6 = (110)_2$$

$$\text{and } SDIV = 1$$

PLL CONTROL REGISTER (PLLCON)

Register	Address	R/W	Description	Reset Value
PLLCON	0x01D80000	R/W	PLL configuration Register	0x38080

PLLCON	Bit	Description	Value
MDIV	[19:12]	Main divider control	111011
PDIV	[9:4]	Pre-divider control	110
SDIV	[1:0]	Post divider control	1

$$PLLCON = 11\ 1011\ 0000\ 0110\ 0001 = 0x3B061$$

PLL VALUE SELECTION GUIDE

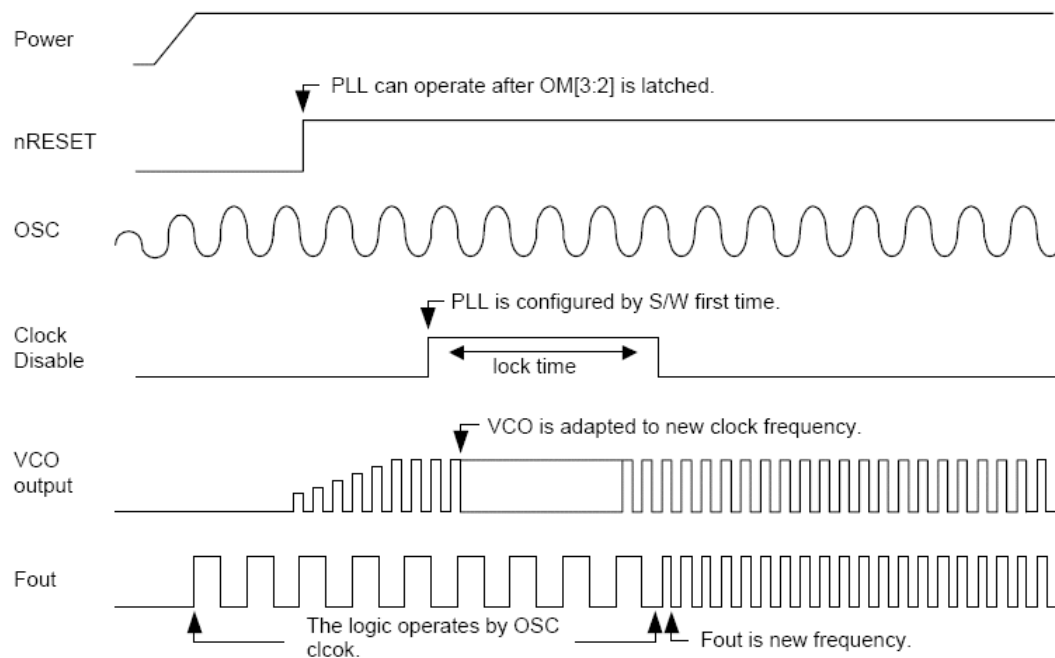
1. $F_{pllo} \times 2^s < 170\text{ MHz}$.
2. S should be as large as possible.
3. $1\text{ MHz} \leq \frac{F_{in}}{p} < 2\text{ MHz}$.

6.3.5 Clock control logic

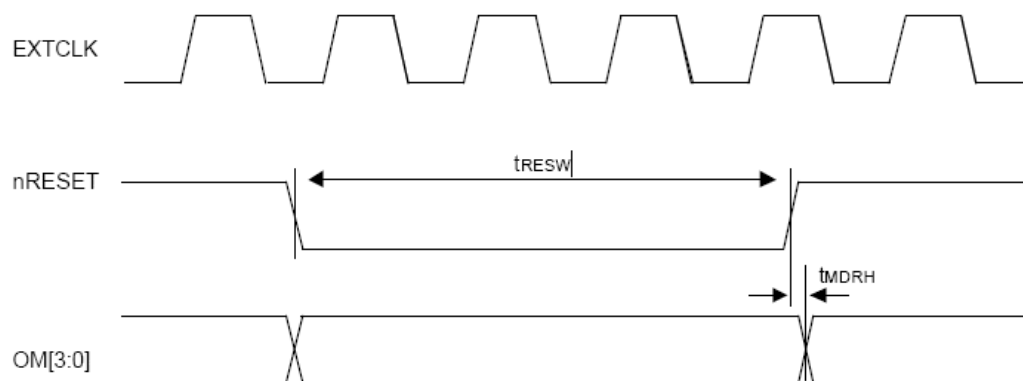
- The clock control logic is also activated at power-on reset and wake-up from power-down mode.
- It determines the clock source: PLL or oscillator clock.
- It disables the *FOUT* until PLL output is stabilized when the PLL is configured to a new frequency value.
- The lock time is the time required for PLL output stabilization and it should be longer than 208 μ s.
- The lock-time is inserted automatically after reset and wake-up from STOP or SL_IDLE mode.

6.3.6 Power-On Reset

- The crystal oscillator begins oscillation within several milliseconds.
- When OSC clock is stabilized and nRESET is released, the PLL starts to operate according to the default PLL configuration.
- However PLL is commonly known to be unstable after power on reset, so *Fin* fed directly to *Fout* instead of the *Fpllo* (PLL output) before the S/W newly configures the PLLCON.
- Even if the user wants to use the default value of PLLCON register after Reset, the user **MUST** write the same value into PLLCON register by S/W.
- The PLL begins the lockup sequence again toward the new frequency only after the S/W configures the PLL with a new frequency.
- *Fout* can be configured to be the PLL output (*Fpllo*) immediately after lock time.

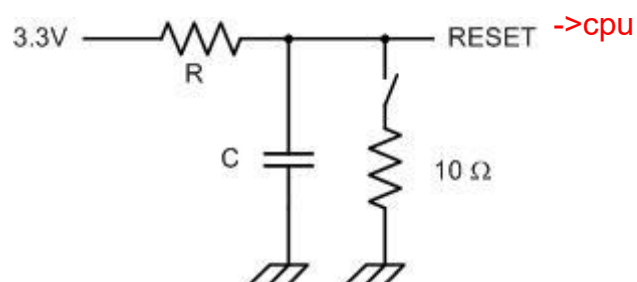


Power-On Reset Sequence

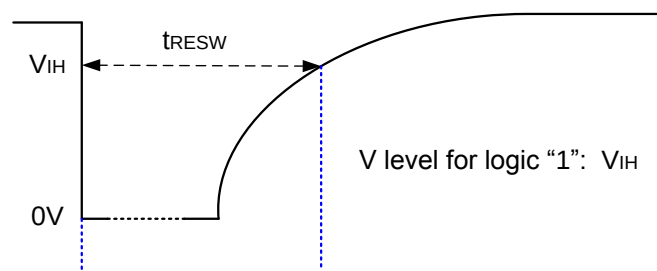


Reset assert time after clock stabilization t_{RESW} (min) = **4 clock periods**

6.3.6.1 Use RC reset circuit



A simple RC Reset Circuit



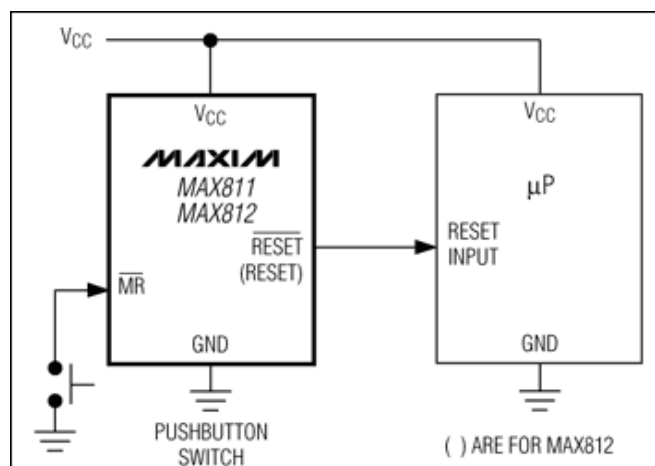
<http://www.csgnetwork.com/rctimecalc2.html>

Use tools from the above web site:

Enter: Supply Voltage
 Capacitance
 Target voltage
 Required time

The value of required resistance will be calculated.

6.3.6.2 Use Reset IC

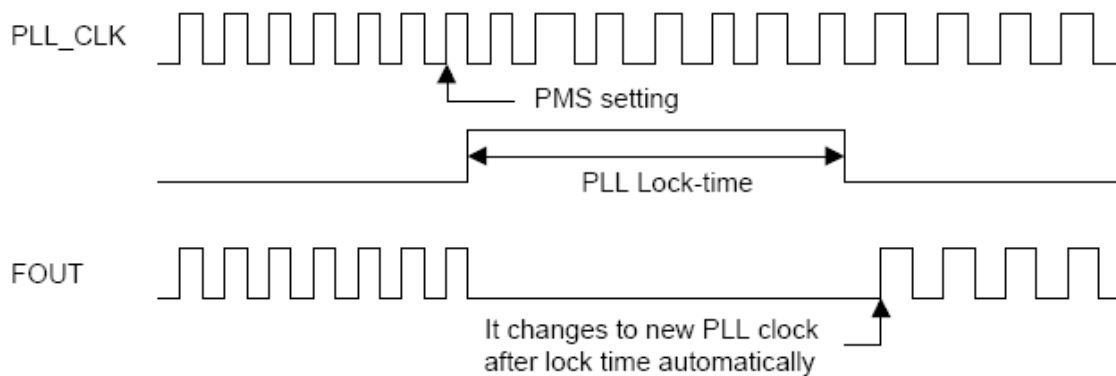


Reset IC

Key Specifications: Supervisors (1 Monitored Voltage)					
Part Number	Reset Thresh. (V)	Reset Out	t_{RESET}	Watchdog Feature	Supervisor Features
			min		
MAX811	2.5 to 3.3	Active Low Push-Pull	85ms to 300ms	No Watchdog	Manual Reset
MAX812	3.3 to 5.5	Active High Push-Pull			

6.3.7 Change *Fpllo* during normal operation mode

- The PLL lock time is automatically inserted when new P, M and S values are applied.
- During the lock time, the clock is not supplied to the internal blocks in S3C440X, i.e. **CPU is NOT working**
- The timing diagram is as follow:

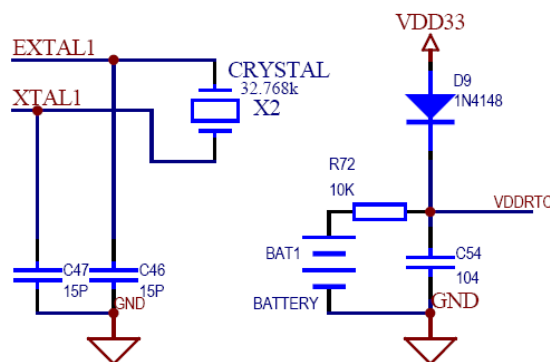
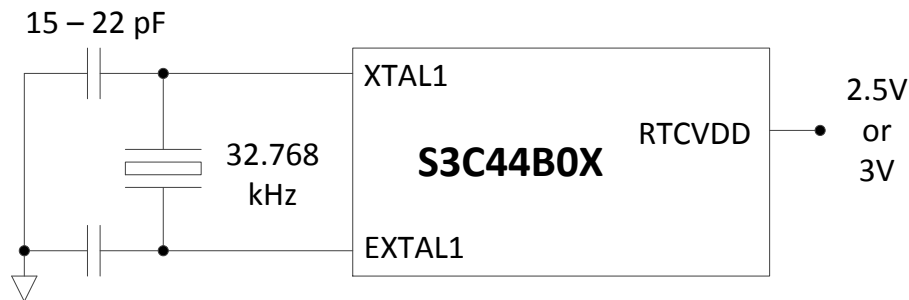


The Case that Changes Slow Clock by Setting PMS Value

6.4 Real Time Clock (RTC)

- It is not the CPU clock.
- A RTC may be implemented in form of a discrete IC or integrated inside a CPU.
- It keeps track of the current time even when power supply is switched off.
- An alternative power source (typically a lithium battery since it is recharged when power is turned on) must be installed to maintain the RTC normal operation against situations like power-off or power failure.
- It may provide more functions like alarm and calendar.
- Many applications requires the use of RTC, for example:
 - Real-time watch / Stop watch/ Calendar
 - Scheduled data logger
 - Reminding for services
 - Attendance record
 - Time Stamping
 - Automatic electricity ON/OFF control
- RTC offers certain benefits:
 - Low power consumption (important when running from alternate power);
 - Free the main system for time-critical tasks;
 - Sometimes more accurate than other methods (although personal computer RTC's are often inaccurate).
- Most RTCs use a crystal oscillator of frequency 32768 Hz.
- Since $32768 = 2^{15}$, it can be conveniently manipulated by using binary counter; calibration and error adjustment can also be easily implemented.

6.4.1 RTC of S3C44B0X



RTC crystal circuit diagram

- The RTC unit **MUST** be operated by a backup battery while the system power is off.
- The data include second, minute, hour, date, day, month, and year, and are represented in 8-bit BCD format
- It includes the following features:
 - BCD number: second, minute, hour, date, day, month, year
 - Leap year generator
 - Alarm function: alarm interrupt or wake-up from power down mode.
 - Independent power pin (VDDRTC)

- The RTC alarm register, RTCALM, determines the alarm enable/disable and the condition of the alarm time setting.

Tick time interrupt

- The RTC tick time is used for interrupt request.
- The TICNT register has an interrupt enable bit and the count value for the interrupt.
- The count value reaches '0' when the tick time interrupt occurs. Then the period of interrupt is as follow:

$$\text{Period} = (n + 1)/128 \text{ second}$$

n : Tick time count value (1 – 127)

6.4.2 Setting up the RTC

- The RTCCON register consists of 4 bits such as the RTCEN, which controls the read/write enable of the BCD registers, CLKSEL, CNTSEL, and CLKRST for testing.
- RTCEN bit can control all interfaces between the CPU and the RTC, **so it should be set to 1 in an RTC control routine to enable data read/write after a system reset.**
- Also before power off, the RTCEN bit should be cleared to 0 to prevent inadvertent writing into RTC registers.

REAL TIME CLOCK CONTROL REGISTER (RTCCON)

Register	Address	R/W	Description	Reset Value
RTCCON	0x01D70040(L) 0x01D70043(B)	R/W (by byte)	RTC control Register	0x0

RTCCON	Bit	Description	Initial State
CLKRST	[3]	RTC clock count reset 0 = No reset, 1 = Reset	0
CNTSEL	[2]	BCD count select 0 = Merge BCD counters 1 = Reserved (Separate BCD counters)	0
CLKSEL	[1]	BCD clock select 0 = XTAL 1/2 ¹⁵ divided clock 1 = Reserved (XTAL clock only for test)	0

RTCEN	[0]	RTC read/write enable 0 = Disable, 1 = Enable If RTC read/write feature is enabled, The STOP current will be consumed excessively. To reduce STOP current, this bit should be 0 while not accessing RTC. Although this bit is 0, the RTC clock is still alive.	0
-------	-----	---	---

NOTES:

1. All RTC registers have to be accessed by byte unit using STRB and LDRB instructions or char type pointer.
2. (L): When the endian mode is little endian. (B): When the endian mode is Big endian.

Example of setting the time (sec, min and hr)

BCD SECOND REGISTER (BCDSEC)

Register	Address	R/W	Description	Reset Value
BCDSEC	0x01D70070(L) 0x01D70073(B)	R/W (by byte)	BCD second Register	Undef.

BCDSEC	Bit	Description	Initial State
Reserved	[7]		–
SECDATA	[6:4]	BCD value for second from 0 to 5	–
	[3:0]	from 0 to 9	–

BCD MINUTE REGISTER (BCDMIN)

Register	Address	R/W	Description	Reset Value
BCDMIN	0x01D70074(L) 0x01D70077(B)	R/W (by byte)	BCD minute Register	Undef.

BCDMIN	Bit	Description	Initial State
Reserved	[7]		–
MINDATA	[6:4]	BCD value for minute from 0 to 5	–
	[3:0]	from 0 to 9	–

BCD HOUR REGISTER (BCDHOUR)

Register	Address	R/W	Description	Reset Value
BCDHOUR	0x01D70078(L) 0x01D7007B(B)	R/W (by byte)	BCD second Register	Undef.

BCDHOUR	Bit	Description	Initial State
Reserved	[7:6]		–
HOURLDATA	[5:4]	BCD value for hour from 0 to 2	–
	[3:0]	from 0 to 9	–

For example, the simplest way to set the time to 11:25:07 is:

```
rBCDHOUR = 0x11; // BCDHOUR   = 0001 0001 in binary format
rBCDMIN  = 0x25; // BCDMIN    = 0010 0101 in binary format
rBCDSEC  = 0x07; // BCDSEC    = 0000 0111 in binary format
```

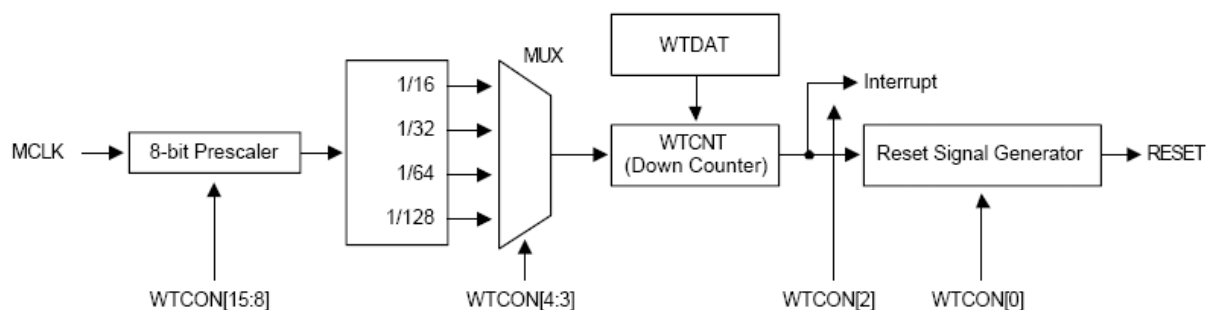
Setting up the date and alarm are similar as above.

6.5 Watchdog timer

- After power-on, the watchdog timer is responsible for reset (reboot) the CPU when it is malfunction and remained in a halt-state and cannot resume normal operation.
- The watchdog timer generates the reset signal for 128 MCLK cycles and the CPU starts to execute the Reset Handler (boot program – i.e. reboot).
- Watchdog should be **OFF** during program development
- Watchdog can be **ON** after program verification

6.5.1 Watchdog timer operation

- The functional block diagram of the watchdog timer is shown in the figure below:



- Watchdog timer clock - MCLK
- The MCLK frequency is prescaled first, and the resulting frequency is divided again.
- The prescaler values range from 0 to $2^8 - 1$.
- The frequency division factor can be selected as 16, 32, 64, or 128.
- The following equation is used to calculate the watchdog timer clock frequency and the duration of each timer clock cycle:

$$t_{\text{watchdog}} = 1 / (\text{MCLK} / (\text{Prescaler value} + 1) / \text{Div_factor})$$

6.5.2 Operation issues

- When the watchdog timer is enabled first, the value of WTDAT (watchdog timer data register) cannot be automatically reloaded into the WTCNT (timer counter).
- For this reason, an initial value **must be written to WTCNT** (watchdog timer count register) **before the watchdog timer starts**.
- When S3C44B0X is in debug mode using Embedded ICE, the watchdog timer will not be operated.

6.5.3 Watchdog timer registers

- The watchdog timer has 3 registers for setting-up its operation and details can be found in the databook.

Watchdog Timer Control Register (WTCON)

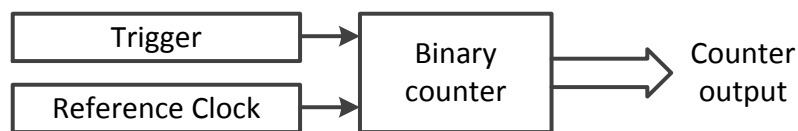
Watchdog Timer Data Register (WTDAT)

Watchdog Timer Count Register (WTCNT)

6.6 Timers

6.6.1 Simple timer

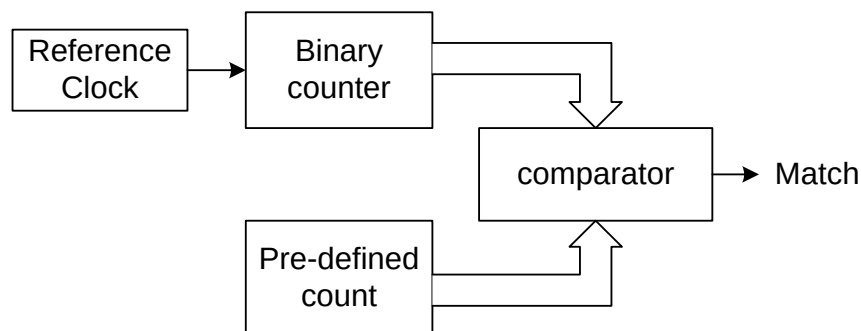
- A timer basically is counting circuit.
- Its counting speed is according to its reference clock frequency.
- The longer the counter bit length, the longer time that can be counted.



- Application – **measure the width of a pulse**

$$\text{pulse width} = \frac{\text{count}}{\text{reference clock frequency}}$$

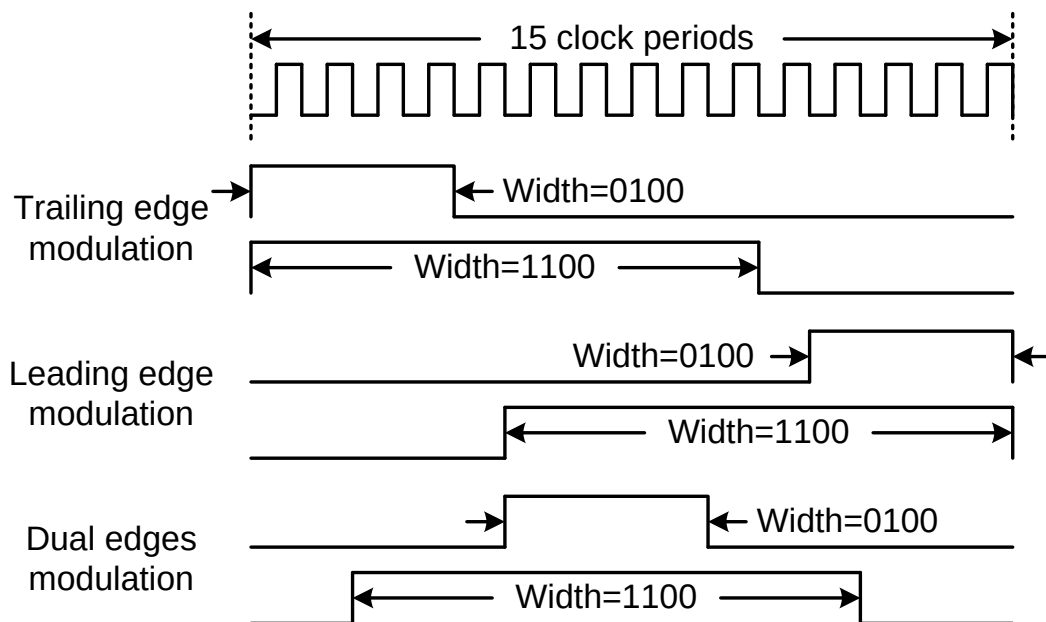
- An output is produced if the counter's out matches a pre-defined counter value.



- Application - **to “time” and trigger an event**
- If the timer can be restarted automatically, periodic and repetitive action can be carried out by means of interrupt to save the resource of the CPU.
- For example, a data logger is used to sample the operating conditions of a machine, say every 5 seconds, to determine if attention is required.

6.6.2 PWM (Pulse Width Modulation) timer

- If a timer can be used to convert a digital value into a duty cycle representation, it is called a PWM timer.
- The pulse width control can be achieved by varying the starting point (leading edge modulation), ending point (trailing edge modulation) or both starting and ending points (dual edges modulation).
- Example: A 4-bit counter can represent 0 to 15. It requires 15 clock periods to cover the width representation. Therefore, 15 clock periods is one cycle and each clock period equates to an increment of one bit.



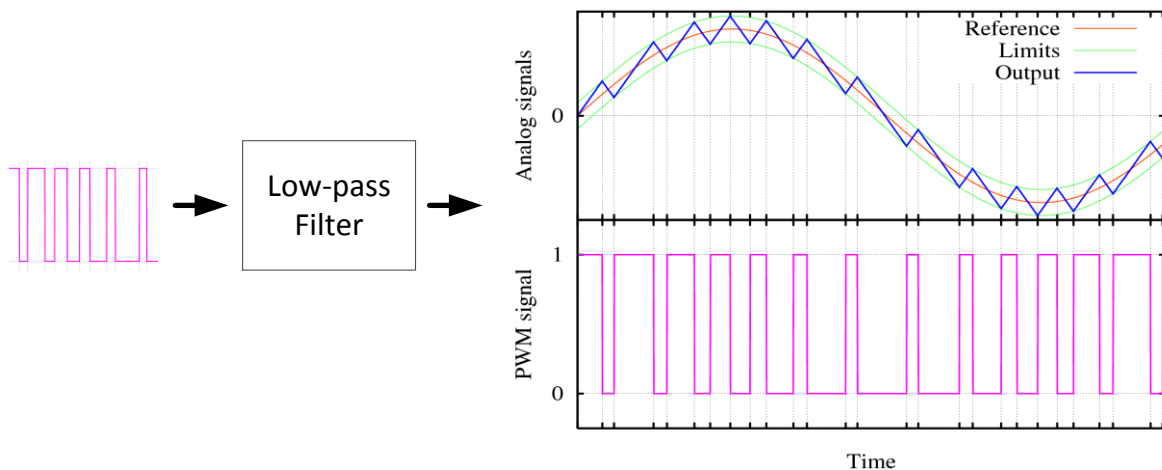
- $\text{duty_cycle} = \frac{\text{pulse width}}{\text{period}}$
- duty_cycle varies from 0.0 (equivalent to 00...0) to 1.0 (equivalent to 111...1)
- The equivalent DC output over one cycle is:

$$V_{eqv} = V \times \text{duty_cycle}$$

6.6.2.1 PWM applications

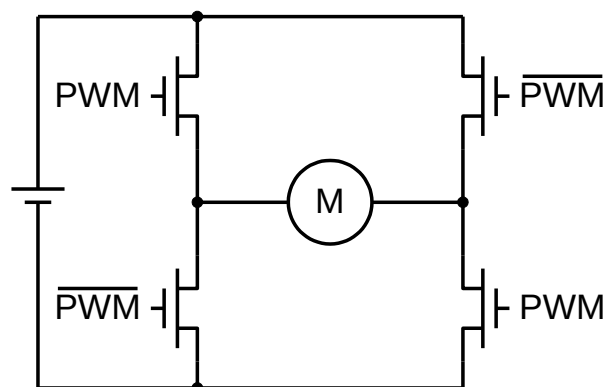
A. Analogue signal compression/decompression

Analogue signal can be represented by “one” bit in digital domain and reproduced by connecting a low-pass filter to the PWM output as shown in the following.



B. Motor speed control

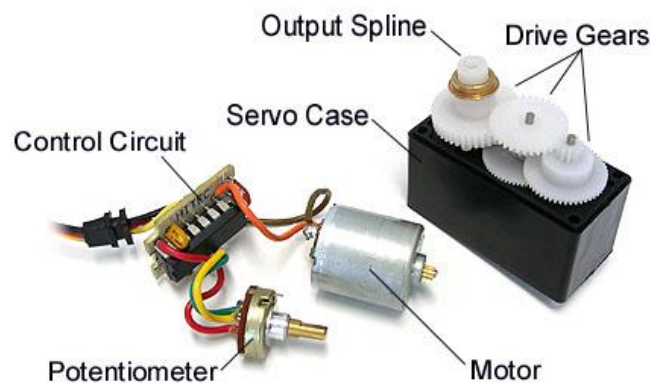
The speed of a motor can be controlled by varying the supply voltage which in turn can be achieved by adjusted PWM duty cycle.



C. Servo motor control

A servo motor is a rotary actuator that allows for precise control of angular position.

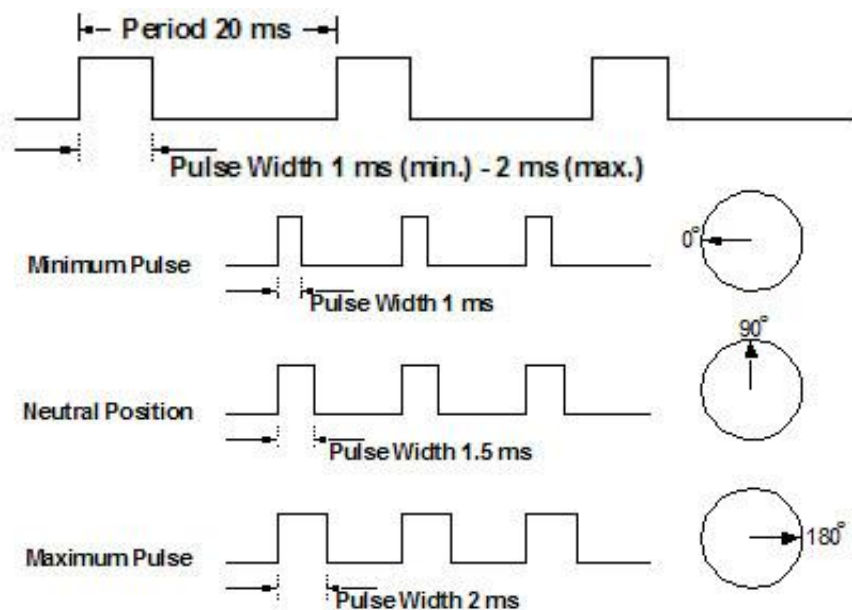
When a servo is commanded to move a position and it will hold that position even with external assertion of force. The maximum amount of force the servo can exert is the torque rating of the servo.



A repetitive pulse is sent to the servo motor.

Angle of rotation is determined by the pulse width of the signal (motor dependent).

A neutral position is always around 1.5 milliseconds (ms).

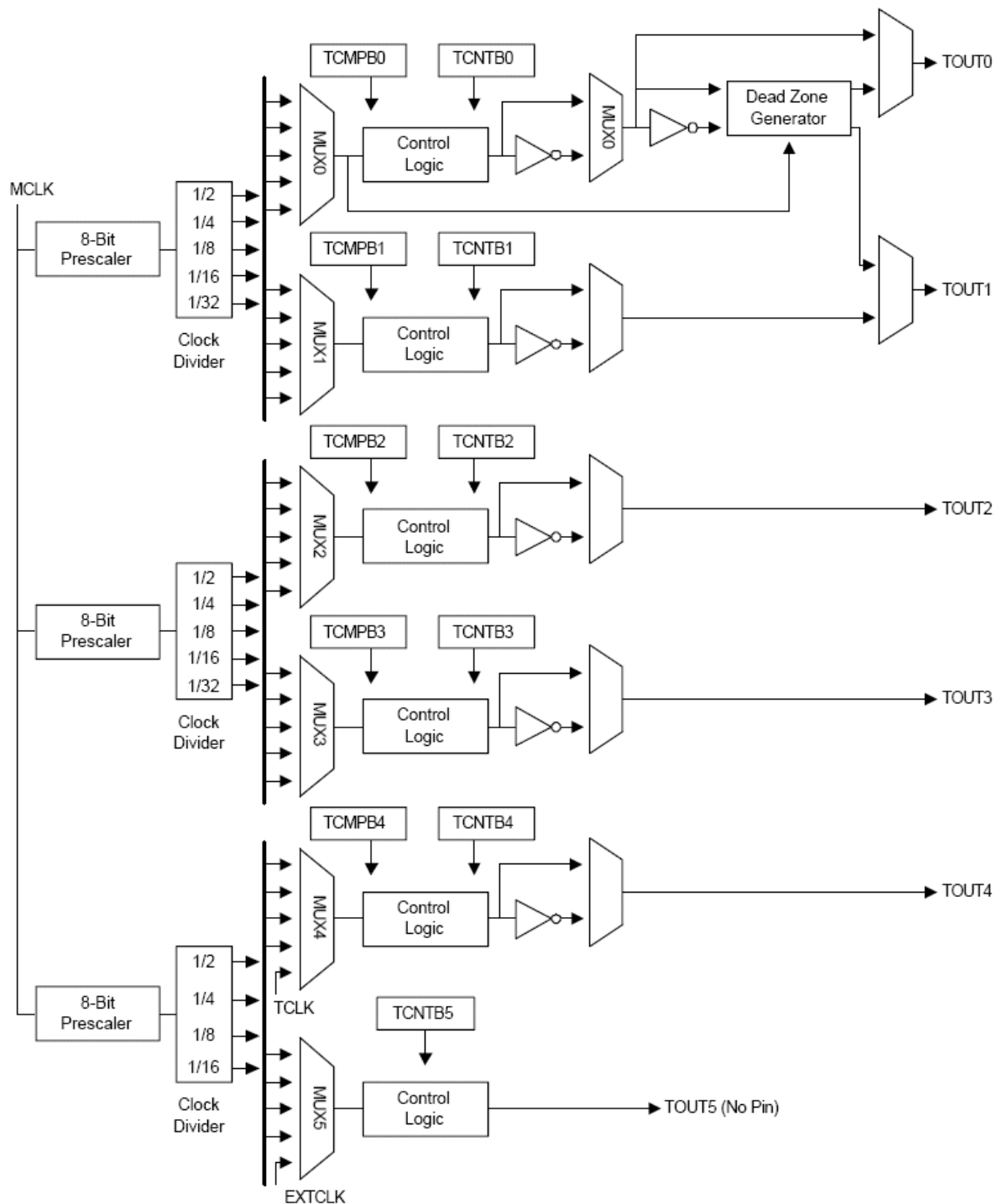


Application: robot joints – forward /backward Rotation

6.7 PWM timer of S3C44B0X

- The S3C44B0X has six 16-bit timers, each timer can operate in interrupt-based or DMA-based mode.
- Each timer has its own 16-bit down-counter which is driven by the timer clock.
- When the **down-counter reaches zero**, the **timer interrupt request is generated** to inform the CPU that the timer operation has been completed.
- The clock frequency fed to the timers can be adjusted via three 8-bit prescalers, two 5-bit dividers and one 4-bit divider.
- 8-bit prescaler ranges from 1 to 255.
- Values of 5-bit divider: {1/2, 1/4, 1/8, 1/16, 1/32}.
- Values of 4-bit divider: {1/2, 1/4, 1/8, 1/16}.
- An 8-bit prescaler and an independent 4-bit divider make the following output frequencies:

4-bit divider settings	minimum resolution (prescaler = 1)	maximum resolution (prescaler = 255)	maximum interval (TCNTBn = 65535)
1/2 (MCLK = 66 MHz)	0.030 μ s (33.0 MHz)	7.75 μ s (58.6 KHz)	0.50 sec
1/4 (MCLK = 66 MHz)	0.060 μ s (16.5 MHz)	15.5 μ s (58.6 KHz)	1.02 sec
1/8 (MCLK = 66 MHz)	0.121 μ s (8.25 MHz)	31.0 μ s (29.3 KHz)	2.03 sec
1/16 (MCLK = 66 MHz)	0.242 μ s (4.13 MHz)	62.1 μ s (14.6 KHz)	4.07 sec
1/32 (MCLK = 66 MHz)	0.485 μ s (2.06 MHz)	125 μ s (7.32 KHz)	8.13 sec



16-bit PWM Timer Block Diagram

NOTE:

The timers 0, 1, 2, 3 and 4 are able to produce PWM output.

Timer 5 is an internal timer with no output pins.

Timer 0 has a dead-zone generator, which is used with a large current device.

6.7.1 Timer operation

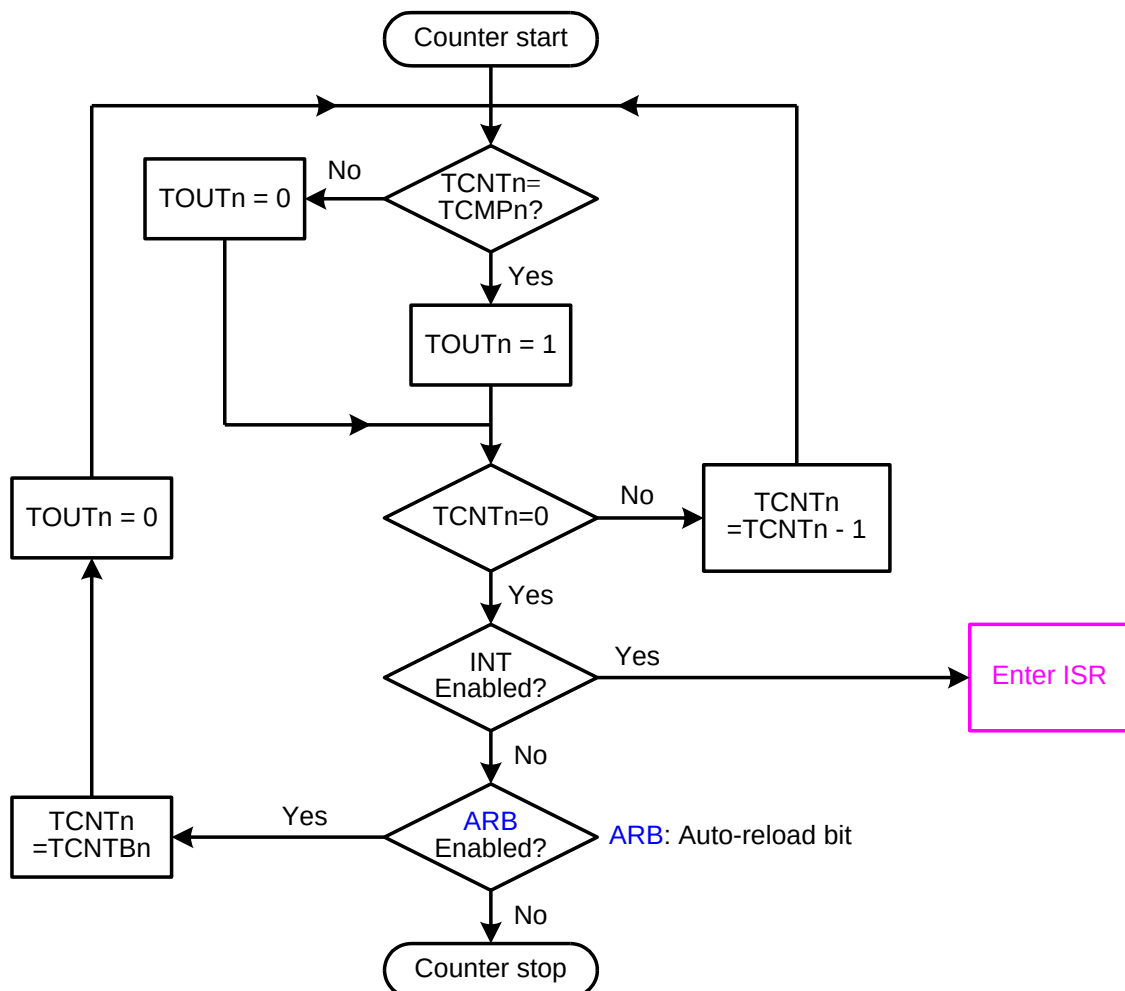
TCNTn: internal timer counter register – a down-counter and not visible to user.

TCNTBn: timer counter buffer register – for loading the timer value.

TCNTOn: timer count observation register – store the current counter value of the timer.

TCMPBn: timer compare buffer register – its value will be compared with the timer down-counter's value.

- The timer can be programmed to “time” once or repetitively using auto-load function.
- For auto-reload operation, TCNTn will be loaded with the value stored in TCNTBn when TCNTn reaches zero.



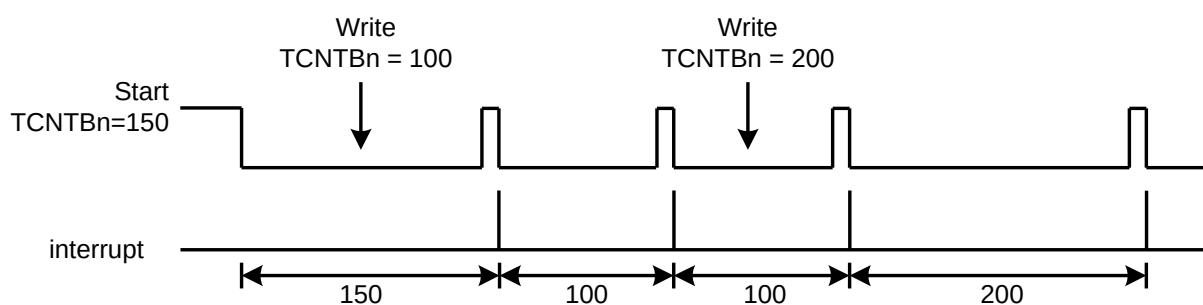
6.7.2 Initialization or reloading of timer

To initialize or writing a new value to the timer, the TCNTBn must first be written with a starting value by manual update.

- 1) Write the initial value into TCNTBn and TCMPBn
- 2) Set the manual update bit of the corresponding timer; and set TOUTn (inverter bit = 0/1, TOUTn = 0/1)
- 3) Set the start bit of the corresponding timer to start the timer.
- 4) Reset manual update bit of the timer.

6.7.3 Illustration of timer operation

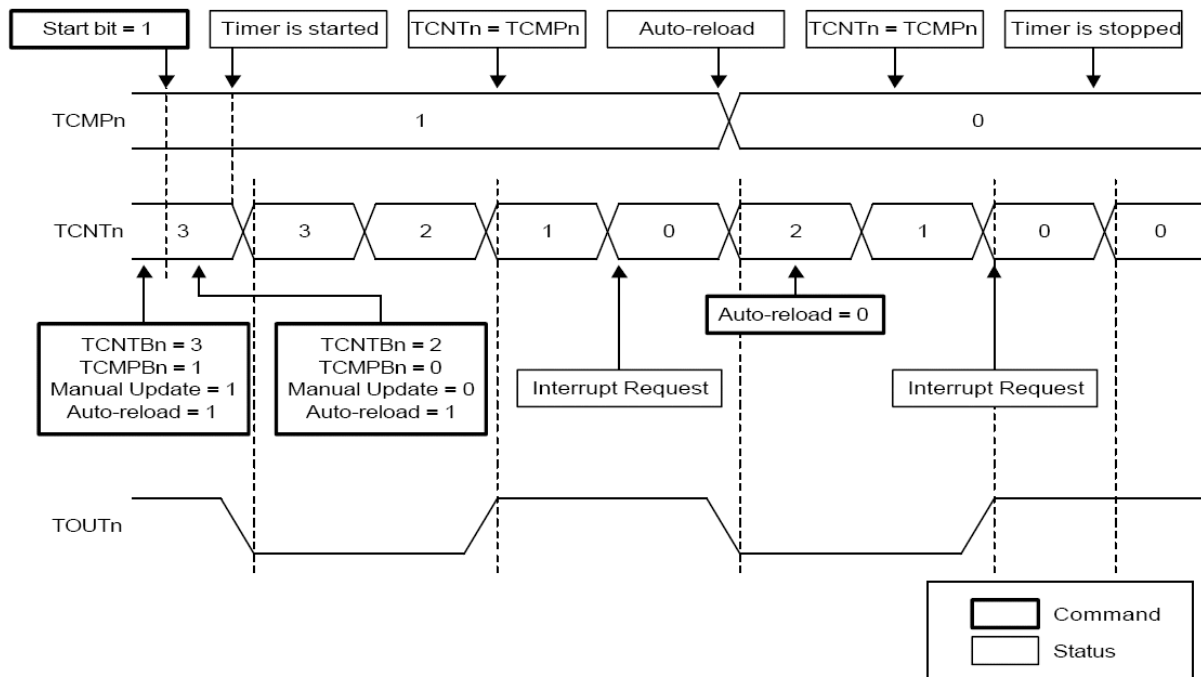
- For TCMPBn = 0: the timer is basically a simple timer.
- It simply performs a down-count operation and generates an interrupt when TCNTn reaches zero if interrupt of timer n is enabled.
- The TOUTn remains unchanged throughout the operation.
- The following figure shows the operation with auto-reload bit enabled:



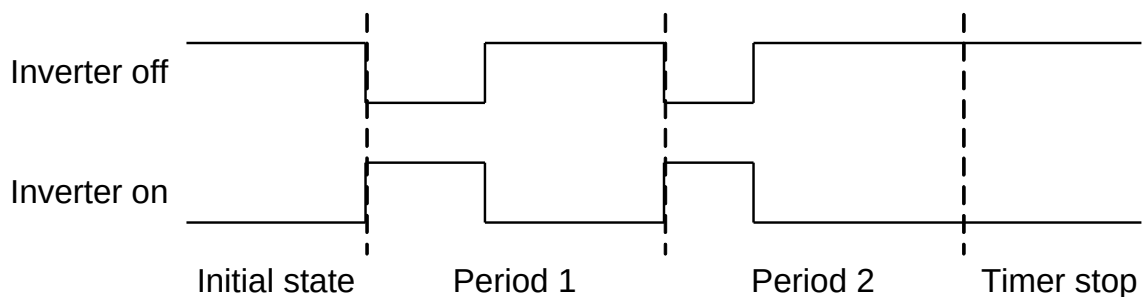
6.7.4 Illustration of PWM timer operation

- For TCMPBn $\neq$ 0: the timer is basically used as a PWM timer
- It performs down-count operation and compares its value with the value stored in TCMPn.
- The TOUTn toggles its output when TCNTn=TCMPBn.

- It also generates an interrupt when TCNTn reaches zero if interrupt of timer n is enabled.
- The following figure shows the operation:



- By loading a different value to TCMPBn every cycle (fixed value for TCNTBn), the duty-cycle for every cycle becomes programmable.



$$\text{Inverter off: Duty_cycle} = \frac{\text{TCMPBn}}{\text{TCNTBn}}$$

$$\text{Inverter on: Duty_cycle} = \frac{\text{TCNTBn} - \text{CMPBn}}{\text{TCNTBn}}$$

Register	Address	R/W	Description	Reset Value
TCFG0	0x01D50000	R/W	Configures the three 8-bit prescalers	0x00000000

TCFG0	Bit	Description	Initial State
Dead zone length	[31:24]	These 8 bits determine the dead zone length. The 1 unit time of the dead zone length is equal to the 1 unit time of timer 0.	0x00
Prescaler 2	[23:16]	These 8 bits determine prescaler value for Timer 4 & 5	0x00
Prescaler 1	[15:8]	These 8 bits determine prescaler value for Timer 2 & 3	0x00
Prescaler 0	[7:0]	These 8 bits determine prescaler value for Timer 0 & 1	0x00

Register	Address	R/W	Description	Reset Value
TCFG1	0x01D50004	R/W	6-MUX & DMA mode selection register	0x00000000

TCFG1	Bit	Description	Initial State
DMA mode	[27:4]	Select DMA request channel 0000 = No select (all interrupt) 0001 = Timer0 0011 = Timer2 0010 = Timer1 0100 = Timer3 0101 = Timer4 0110 = Timer5 0111 = Reserved	000
MUX 5	[23:20]	Select MUX input for PWM Timer5. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = EXTCLK	000
MUX 4	[19:16]	Select MUX input for PWM Timer4. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = TCLK	000
MUX 3	[15:12]	Select MUX input for PWM Timer3. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = 1/32	000
MUX 2	[11:8]	Select MUX input for PWM Timer2. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = 1/32	000
MUX 1	[7:4]	Select MUX input for PWM Timer1. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = 1/32	000
MUX 0	[3:0]	Select MUX input for PWM Timer0. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = 1/32	000

Register	Address	R/W	Description	Reset Value
TCNTB0	0x01D5000C	R/W	Timer 0 count buffer register: 16-bit	0x00000000
TCMPB0	0x01D50010	R/W	Timer 0 compare buffer register: 16-bit	0x00000000