LPC1768 Timer

- Using delays in a software code is usual in embedded programing. It can be done with a for loop iterating for a few 1000 cycles.
- But these types of delays need not be accurate and fundamentally, it is not a good programming practice.
- So, TIMER/COUNTER is programmed to count the time interval between events.
- It counts the cycle of the peripheral clock or an externally-supplied clock.

Timer v/s Counter

- **Timer**
  - In general, it is a specific type of clock used to measure the time interval or delay
  - It requires a clock to work
  - Suppose 1 kHz is the i/p clock frequency of the timer, the

    Clock period = 1/ frequency
    = 1/1k
    = 1 ms
    - i.e., 1000 clock counts provide 1 second of time interval / delay
- **Counter**
  - It counts the external events or external clock ticks
  - It is used to measure the frequency of the external clock ticks

- LPC1768/LPC1769 has four 32-bit Timer blocks, TIMER0-TIMER3.
- Each Timer block can be used as a ‘Timer’ or as a ‘Counter’.
- They are designed to count the peripheral clock or external clock signals and optionally generate interrupts at the specified timer/count values.
- It can be also used to demodulate PWM signals given as input.

- Each Timer module has its own **Timer Counter (TC)** and **Prescale Register(PR)** associated with it.

- **Timer Counter (TC) (32 bit):** This is the main counting register. TC incremented by 1, for every **'PR+1'** clock cycles – where PR is the value stored in Prescale Register
  - It can count from 0 to $2^{32} - 1$

- **Prescale Register (PR) (32 bit):** If **PR is 0** then TC is incremented for every **1 clock cycle of the peripheral clock (P_{CLK})**. If **PR=1** then TC is incremented every 2 clock cycles of P_{CLK} and so on.
  - ➔ The job of the PR is to divide the **P_{CLK}** to a suitable value and lower the speed of timer clock (T_{CLK}) that is fed to the TC

- **PC: Prescale Counter Register (PC) (32 bit)** – This register increments on every PCLK.
- When PC reaches the value in PR , PC is reset back to 0 and TC is incremented by 1.
- Hence if PR=0 then TC Increments on every PCLK.
- If PR=9 then TC Increments on every 10th cycle of PCLK.
- Hence by selecting an appropriate prescale value we can control the resolution of the timer.

Obtaining desired frequency / PR

W.K.T the job of PR is to divide the PCLK to a suitable value and provide a low speed T_{CLK} to TC

PR = (Freq of P_{CLK} / Freq of T_{CLK}) -1

$$PR = \frac{P_{CLK}}{T_{CLK}} - 1$$

The general formula for Timer resolution at a P_{CLK} and a given value for PR is given as,

$T_{RES} = (PR+1)/P_{CLK}$

Here, the resolution is also the time delay required to increment TC by 1.

Hence, PR value for 1 micro-second resolution at 3 Mhz P_{CLK} is,

$PR_{1us} = (3Mhz * 1uS) -1 = (3*10^{6} * 10^{-6}) -1 = 2$

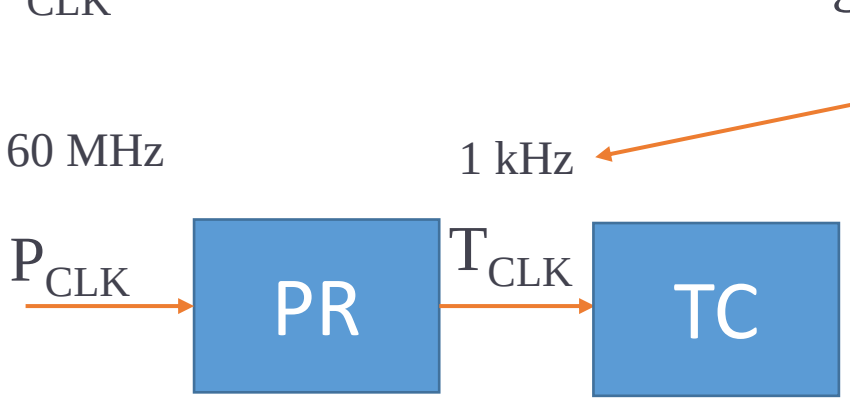
Eg: Given P_{CLK} = 60 MHz, obtain the PR to generate T_{CLK} of 1 MHz

$$PR = \frac{P_{CLK}}{T_{CLK}} - 1$$

$$PR = \frac{60M}{1M} \text{-}1$$

PR= 59

Given P_{CLK} = 60 MHz obtain the PR to generate T_{CLK} of 1ms tick



$$PR = \frac{P_{CLK}}{T_{CLK}} - 1$$

PR +1 = 60 M / 1k

PR = 59999

What is the maximum delay that can be achieved when P_{CLK} = 11MHz and PR = 10.

WKT

$$PR = \frac{P_{CLK}}{T_{CLK}} - 1$$

10+1 = 11M/T_{CLK}

T_{CLK} = 1MHz

So, the resolution / delay is T_{RES} = 1/1M = 1µS

Maximum delay = 1µ * 2^{32} = 4294.9672 S

Maximum delay = 4294.9672 / 60 = 71.58 min

Maximum delay = 1 hr 19 min

CTCR: Count Control register

- Used to select Timer/Counter Mode.
- When CTCR = 0➔ Timer Mode is selected.

TCR: Timer Control Register : 1st two bits (bit0 & bit1)are utilized here.

- Used to enable, disable and reset TC.
- When bit0 = 1 timer is enabled. When bit0 = 0 timer disabled.
- When bit1=1 TC and PC are set to zero.
- Rest of the bits of TCR are reserved.

Match Register

- Each Timer has **four 32-bit Match Registers, MR0-MR3**
- Each Timer has **two 32-bit Capture Registers, CR0, CR1**.
- Timer 0,1,3 have two Match outputs while Timer 2 has four.

- A Match Register: Contains a specific value set by the user. (User will store the count value here)
- When the Timer starts – every time after TC is incremented, the value in TC is compared with match register.
- If it matches then Timer can be Reset or can generate an interrupt as defined in **Match Control register**.

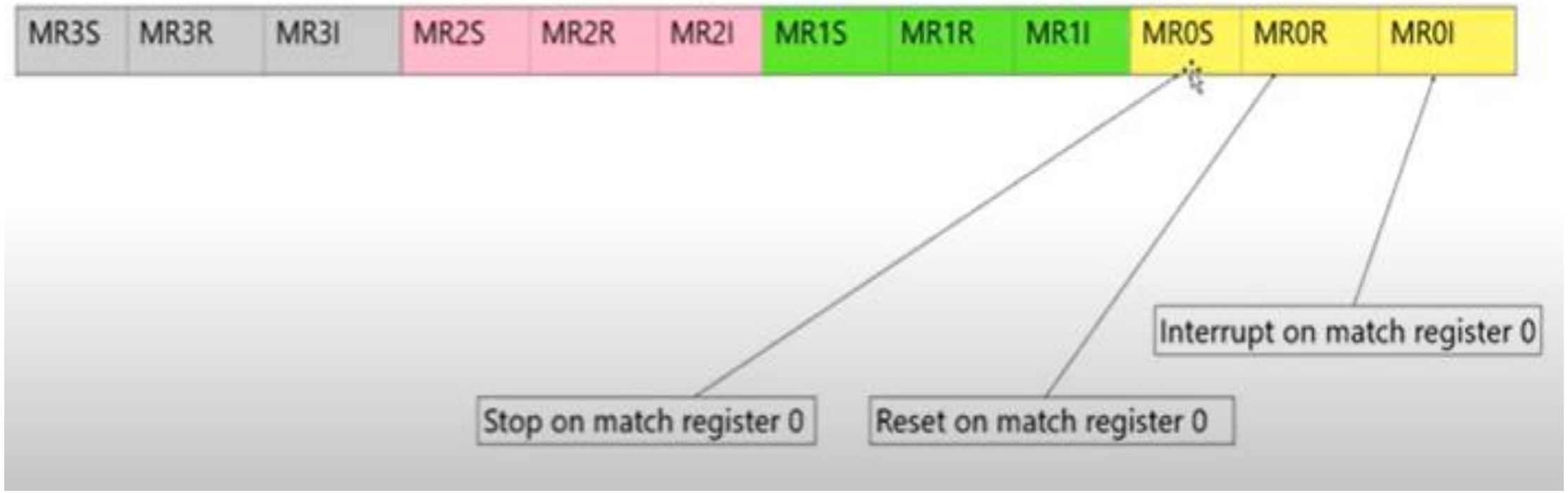
MCR: Match Control register – It is used to control operations that can be done when the value in MR matches the value in TC.

- Bits 0,1,2 are for MR0. Bits 3,4,5 for MR1. Bits 6,7,8 for MR2. Bits 9,10,11 for MR3.

For MR0:

When value of MR0 matches TC value,

- Bit 0: Interrupt on MR0
  - Bit 0 = 1 ➔ Trigger an interrupt
  - Bit 0 = 0 ➔ Disable the interrupt
- Bit 1: Reset on MR0
  - Bit 1 = 1 ➔ TC will be reset
  - Bit 1 = 0 ➔ Disable the TC reset
- Bit 2: Stop on MR0
  - Bit 2 = 1 ➔ TC & PC will stop
  - Bit 2 = 0 ➔ Disable the stop
- Similarly, bits 3-5 , 6-8 , 9-11 are for MR1 , MR2 , MR3 respectively.


MR3S
MR3R
MR3I
MR2S
MR2R
MR2I
MR1S
MR1R
MR1I
MR0S
MR0R
MR0I
Interrupt on match register 0
Stop on match register 0
Reset on match register 0

External Match Output - When a match register (MR3:0) equals the timer counter (TC) this output can either toggle, go low, go high, or do nothing depending on the value in the External Match Register (EMR).

EMR: External Match Register – It provides both status and control of External Match Output Pins.

- First four bits are for EM0 to EM3. (For each timer)
- Next 8 bits are for EMC0 to EMC3 in pairs of 2.

•Bit 0 – EM0: External Match 0: When a match occurs between TC and MR0, depending on bits[5:4] i.e., EMC0, this bit can either toggle, go LOW, go HIGH, or do nothing. This bit is driven to MATx.0 where x=Timer number. (→ external pin)

•Similarly for Bits 1, 2 & 3.

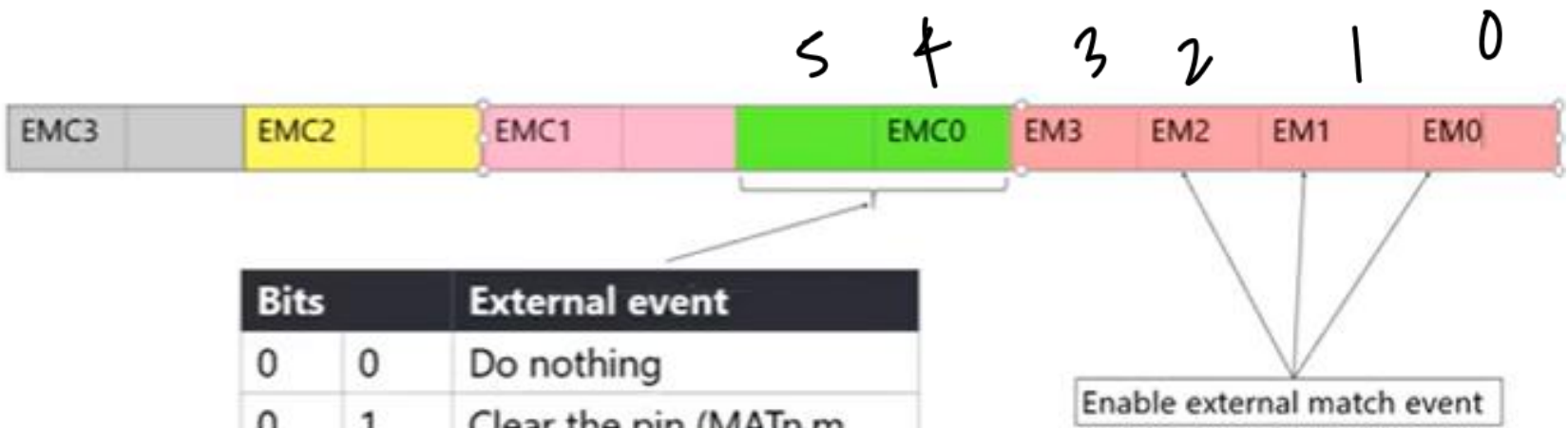
•Bits[5:4] – EMC0: External Match 0: The values in these bits select the functionality of EM0 as follows:

- 0x0: Do nothing
- 0x1: Clear the corresponding External Match to 0 (MATx.m pin is LOW).
- 0x2: Set the corresponding External Match to 1 (MATx.m pin is HIGH).
- 0x3: Toggle the corresponding External Match output.

•Similarly for Bits[7:6] – EMC1, Bits[9,8] – EMC2, Bits[11:10] – EMC3.

MAT : Match

: Configure external match control register (optional if you want an external signal event when a match happens)



Bits		External event
0	0	Do nothing
0	1	Clear the pin (MATn.m pin becomes low)
1	0	Set the pin (MATn.m pin becomes high)
1	1	Toggle the MATn.m pin

32 16 8 4 2 1
10 0000

32 → 20

Capture Register (CAP0-CAP3)

- As the name suggests it is used to Capture Input signal. When a transition event occurs on a Capture pin, it can be used to copy the value of TC into any of the 4 Capture Register or to generate an Interrupt.

Setting up & configuring Timers in LPC1768

- All the Registers used to program and use timers are defined as members of structure (pointer) `LPC_TIMx`, where x is the Timer module from 0 to 3.
- So, for Timer0 we will use `LPC_TIM0` and so on.
- Registers can be accessed by de-referecing the pointer using "->" operator.
- For, example we can access TCR of Timer0 block as `LPC_TIM0->TCR`.

Initialisation:

1. Set appropriate value in `LPC_TIMx->CTCR`
2. Define the Prescale value in `LPC_TIMx->PR`
3. Set Value(s) in Match Register(s) if required
4. Set appropriate value in `LPC_TIMx->MCR` if using Match registers / Interrupts
5. Reset Timer – Which resets PC and TC
6. Set `LPC_TIMx->TCR` to `0x01` to Enable the Timer when required
7. Reset `LPC_TIMx->TCR` to `0x00` to Disable the Timer when required

LPC1768 Timer Prescaler Calculations:

The delay or time required for 1 clock cycle when PCLK = ‘X’ Mhz is given by :

$$T_{PCLK} = 1/PCLK_{Hz}$$
$$=1/(X * 10^6)Seconds$$

It is also the maximum resolution Timer block can provide at a given PCLK frequency of X Mhz. The general formula for Timer resolution at X Mhz PCLK and a given value for prescale (PR) is as given below:

$$T_{RES} = (PR+1)/PCLK_{Hz}$$
$$=(PR+1)/(X * 10^6)Seconds$$

Hence, we get the **Formula for Prescaler (PR) for required Timer resolution (T_{RES} in Secs)** at given **PCLK(in Hz)** frequency as:

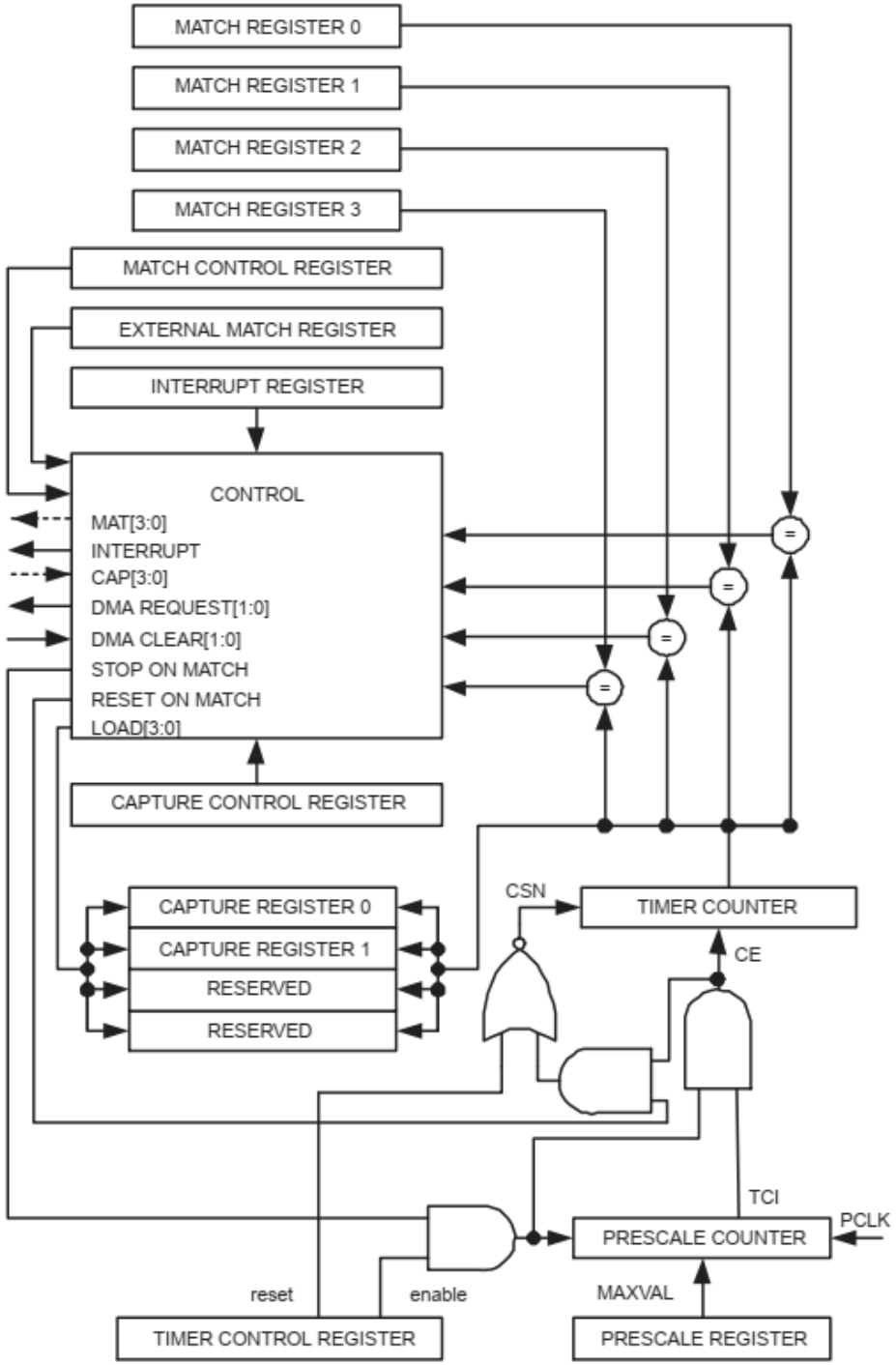
$$PR = (PCLK_{Hz} * T_{RES}) - 1$$

$$PR = ((X * 10^{6}) * T_{RES}) - 1$$

Note that here, the resolution is also the time delay required to increment TC by 1.

Hence, Prescaler value for 1 micro-second resolution at 3 Mhz PCLK is,

$$PR_{1us} = (3Mhz * 1uS) - 1 = (3*10^{6} * 10^{-6}) - 1 = 2$$



Timer Block Diagram.

Ref:

- http://www.ocfreaks.com/lpc1768-timer-programming-tutorial/
- **UM10360 LPC 176x/5x User Manual Chapter 21**

Program to get 1ms delay using timer0. Given Tres = 1us, PCLK = 3MHz

```
void initTimer0(void)
{
    LPC_TIM0->CTCR = 0x0;   //timer mode
    LPC_TIM0->TCR = 0x02; //Reset Timer and Disable timer
    LPC_TIM0->PR = 2; //Increment TC at every 2+1 clock cycles, Tres = 1us (PCLK= 3MHz)
    LPC_TIM0->MR0 = 999; // for 1 ms delay TC counts 0 to 999
    LPC_TIM0->MCR = 2; // reset TC after 1 ms delay
    LPC_TIM0->EMR = 0x20; // when match occurs bit 0 of EMR will set (set EMC0 : 100000 : 32 : 2015)
    LPC_TIM0->TCR = 0x01; //enable timer0
}
void delay(void)
{
        initTimer0();
        while(!(LPC_TIM0->EMR &1));   // Checking if set to first bit for timer 0
}
```

Q) WAP for hrena down counter with $P_{CLK} = 10\ MHz$, $T_{res} = 1\ \mu s$

A)
```
void initTimer0(){
    LPC_TIM0->CTCR = 0x0;
    LPC_TIM0->TCR = 0x2;
    LPC_TIM0->PR = 9;
    LPC_TIM0->MR0 = 999;
    LPC_TIM0->MCR = 2;
    LPC_TIM0->EMR = 0x20;
    LPC_TIM0->TCR = 0x1;
}

void main(){

    LPC_PINCON->PINSEL0 &= FF0000FF;
    LPC_GPIO0->FIODIR |= 0x00000FF0;

    LPC_PINCON->PINSEL4 &= 0xFCFFFFFF;
    LPC_GPIO2->FIODIR &= 0xFFFFEFFF; //P2.12

    int count = 15;
    while(1){
        LPC_GPIO->FIOPIN = count;
        initTimer0();
        count--;
        if(count < 0)
            count = 15;
    }
}
```