TIMERS

Creation Guide

Version #: 0.1

Revision date: **NOV 30, 2010**

Version Control

Revision	Authors	Date	Comments
0.1	Inácio Mendonça / Eloi Magalhães	30/NOV/2010	Initial Version

1 TIMERS

1.1 Introduction

The TIMERs Block (timer 0, timer 1 and timer 2), being two on general purpose (timer 0 and 1) and an on specific purpose (timer 2).

The timers 0 and 1 can assume the timer function or counter depending on the configurations attributed to the same by the application (software), still possessing two modes of operation, increasing and decreasing.

The timer 2 has his functionality focused for the section automotive being used as an angle counter in a jagged wheel in the which lacks a tooth that through his occurrence allows to synchronize the counting previously stored in a register with the current counting obtained by a turn of the jagged wheel, being then that validated result and stored in a register, and through these data stored in the register makes possible the counting of turns in the motor of the vehicle (RPM) and it provides to the system (CPU) to evaluate the automobile is accelerating or slowing down so that of ownership of those registrations to increase or to reduce the flow of injection of combustible mixture and the speed of the ignition, that system FlyWheel is called.

Timers 0 and 1 were based on the architecture of the microcontroller 8051, but the timer 2 is a specific application in the automotive sector, its development was based on specifications supplied by the customer.

This new design was derived from the design of microcontroller EMC08.

Was studied by the atmel datasheets and Intel, the operation of timers 0 and 1 configurations and their vectors. Since the second timer was through information given by the project architect and documentation provided by the representative of Magneti Marelli.

After that the first versions were coded block timers.

1.2 Module History

Tables 1-6 lists the history of the module versions and implementations.

Table 1 - Timer 0 and 1 History

Module Version	Derived From	1st Implemented On	Headline
Timer0/1 (0.1)	EMC08 SOW V.1.0	EMC08	Start version.
Timer0/1 (0.2)	Timer0/1 (0.1)	EMC08	Change RESET for active in fall edge.
Timer0/1 (0.3)	Timer0/1 (0.2)	EMC08	Repair bug in registers, it don't updated values.
Timer0/1 (0.4)	Timer0/1 (0.3)	EMC08	New version of code, change variables and correct other bug's.
Timer0/1 (0.5)	Timer0/1 (0.4)	EMC08	Fix error in mode selection, only work in a mode 0 and 3.
Timer0/1 (0.6)	Timer0/1 (0.5)	EMC08	Update input of tag for overflow.
Timer0/1 (0.5)	Timer0/1 (0.6)	EMC08	Reduces number of variables.
Timer0/1 (0.6)	Timer0/1 (0.5)	EMC08	Fix bug in a time of values change.
Timer0/1 (0.7)	Timer0/1 (0.6)	EMC08	Rename variables
Timer0/1 (0.8)	Timer0/1 (0.7)	EMC08	Remove latch inferred
Timer0/1 (0.9)	Timer0/1 (0.8)	EMC08	Fixed bugs in outputs.
Timer0/1 (0.1.0)	Timer0/1 (0.9)	EMC08	Fixed bugs related in LEC Verify tool.
Timer0/1 (1.0)	Timer0/1 (0.1.0)	EMC08	Finish Version.

Table 2 – Angle Clock Generator (ACG) block History

timers_timer2_acg (0.1)	EMC08 SOW V.1.0	EMC08	Start version.
timers_timer2_acg (0.2)	timers_timer2_acg (0.1)	EMC08	Remove input of timer control and respective block.
timers_timer2_acg (0.3)	timers_timer2_acg (0.2)	EMC08	Change acr_inc variable in the line 145.
timers_timer2_acg (0.4)	timers_timer2_acg (0.3)	EMC08	Added timers_sfr_tcon2_tf2_o into reset block, because it isn't trated.
timers_timer2_acg (0.4)	timers_timer2_acg (0.4)	EMC08	Removed two output port's, TACPH and TACPL because the block can't write in this registers.
timers_timer2_acg (0.5)	timers_timer2_acg (0.4)	EMC08	Fix bug's in the variables of state machine.
timers_timer2_acg (0.5)	timers_timer2_acg (0.5)	EMC08	Fix all bug's related in LEC tool.
timers_timer2_acg (1.0)	timers_timer2_acg (0.5)	EMC08	Finish Version.

Table 3 – Prescaler Digital Clock Filter (PDCF) block History

timers_timer2_pdcf (0.1)	EMC08 SOW V.1.0	EMC08	Start Version.
timers_timer2_pdcf (0.2)	timers_timer2_pdcf (0.1)	EMC08	Fix bug in prescaler, adjust counter.
timers_timer2_pdcf (0.2)	timers_timer2_pdcf (0.2)	EMC08	Fix all bug's related in LEC tool.
timers_timer2_pdcf (1.0)	timers_timer2_pdcf (0.2)	EMC08	Finish Version.

Table 4 – Programmable Digital Filter (PDF) block History

timers_timer2_pdcf (0.1)	EMC08 SOW V.1.0	EMC08	Start Version.
timers_timer2_pdcf (0.2)	timers_timer2_pdcf (0.1)	EMC08	Fix bug with the filter validation.
timers_timer2_pdcf (0.3)	timers_timer2_pdcf (0.2)	EMC08	Remove one flip-flop of the sync, because it has one more.
timers_timer2_pdcf (0.4)	timers_timer2_pdcf (0.3)	EMC08	Fix all bug's related in LEC tool.
timers_timer2_pdcf (1.0)	timers_timer2_pdcf (0.4)	EMC08	Finish Version.

Table 5 –Top Timer 2 block History

top_timer2 (0.1)	EMC08 SOW V.1.0	EMC08	Start Version.
top_timer2 (0.2)	top_timer2 (0.1)	EMC08	Added input for bit tf2 (tcon2 register).
top_timer2 (0.3)	top_timer2 (0.2)	EMC08	Remove input of timer control and respective block.
top_timer2 (0.4)	top_timer2 (0.3)	EMC08	Removed two output ports TACPH, TACPL.
top_timer2 (1.0)	top_timer2 (0.3)	EMC08	Finish Version.

Table 6 –Top Timers block History

top_timers (0.1)	Timers Block Guide v.1.0	EMC08	Start Version.
top_timers (0.2)	top_timers (0.1)	EMC08	Added news connections for timer 0 and 1, the same is missing.
top_timers (0.3)	top_timers (0.2)	EMC08	Added input ports for the bits: tf0, tf1 and tf2.
top_timers (0.4)	top_timers (0.3)	EMC08	Added input to machine cycle.
top_timers (0.5)	top_timers (0.4)	EMC08	Removed output ports TACPH and TACPL.
top_timers (1.0)	top_timers (0.5)	EMC08	Finish Version.

1.3 Features

These devices were designed to act as counters timers. It can be activated externally or internally via software and its main features are:

- Clock Generator
- 24-bit Registers
- Up-Counter
- Down-Counter

1.3.1 TIMER 2

The timer 2 is a device for automotive applications, his main characteristic is to provide ECU of the vehicle to make possible the:

- Regulate times of injection
- Regulate the ignition progress
- Control enrichment of the combustible mixture in acceleration
- Cut of fuel in the phase of diminish of the motor
- Administration of the rotation of the motor in the slow march
- Limitation of the maximum rotation of the motor

1.4 Architecture and Design Hierarchy

The operation of timer/counter 0 and 1 are identical, using the descriptions for both below worked.

There are two ways to activate the timer; The timer 0 leaves in low level the bit **GATETx** inside **TMOD** register and set a bit **TRx** from **TCON** register**.** After this, it will now operates as a counter updating the register values **TLx**, **TMx**, **THx** all 8-bit, and finally reaching the maximum value (FFFFFF hex), on the next cycle, an overflow will occur, setting the interrupt flag, **TFx** on **TCON** register (bits 7 and 5).

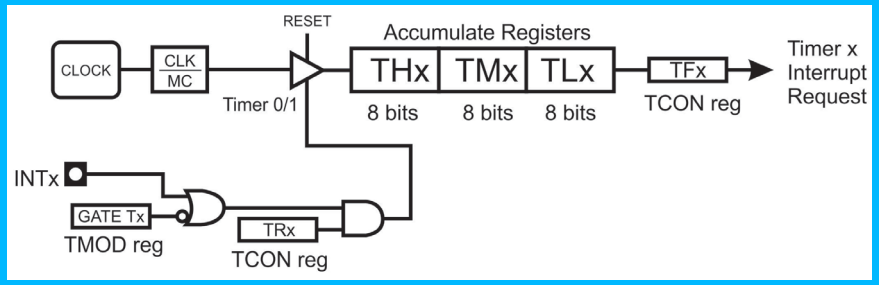

Figure 1 – Functional Block's of timer 0 and 1.

If the interrupt timers are enabled (register IE, bits: EA, ET2, ET1 e ET0), this may act in interrupts, otherwise whenever there is an overflow, the counter continues to count indefinitely until the flag TRx have been reset.

The timers block consists of a sequential counter that can act as a up-counter or down-counter, depending on the value of the control variables "incr or decr" in RTL code.

When the control variable "incr" is set, the counter is now incremented by adding 1 to its value.

If the control variable "decr" is set, the counter is now decremented by subtracting its value to 1.

When an overflow occurs in the sum (16 bits), the interrupt flag TFX, which is located in register TCON, is set in the transition from all 1s to all 0s.

The 24-Bit register consists of three 8 bits registers (TH1/TM1/TL1). Setting the run flag (TR1) does not clear the registers. Mode 0 operation is the same for Timer 0 as for Timer 1. The corresponding Timers 0-1 signals are TR0-1/TF0-1/INT0-1/TH0-1/TM0-1/TL0-1. There are two different GATE bits, one for Timer 1 (TMOD.7) and one for Timer 0 (TMOD.3).

In RTL the name of the registers are respectively: timers_sfr_th0_i, timers_sfr_tm0_i timers_sfr_tl0_i and, for the inputs and timers_sfr_th0_o, timers_sfr_tm0_o and timers_sfr_tl0_o for the outputs, because you need a feedback block to have a controlling value of the registers

Figure 1 describes exactly the run control timer, the RTL was developed based on the same.

1.5 Timers 0 and 1 Description

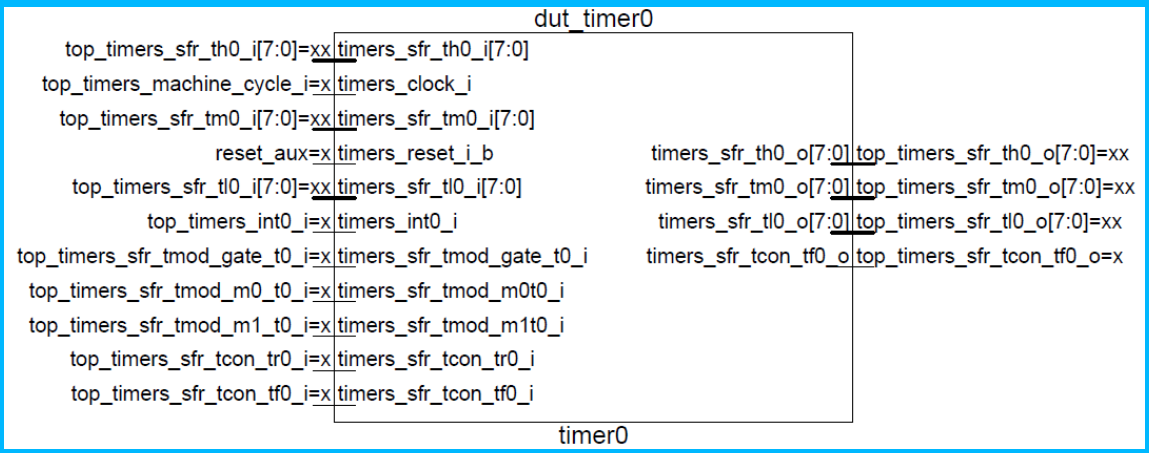

Figure 2 – Top Level of the timer 0

in Figure 2 we have the block diagram of the timer 0 in which shows all inputs and outputs which will be detailed to follow:

1.5.1 Special Functions Registers (SFR)

The timers 0 and 1 have three inputs and three outputs of 8 bits each that connect the special function registers, which are responsible for storing the count value of the timer. The input is used to check the current value of the output, if the timer is turned off then the output will receive the input value remained so until it is turned on again and from there the output will be updated with the new count value.

1.5.1.1 Detailed Signal Descriptions

Signal	I/O	Description		Reset
timers_sfr_tmod_gate_t0_i	I	**State Meaning**	Asserted: if INT0 = 1 Timer 0 Run. Negated: if TR0 = 1 Timer 0 Run.	0
timers_reset_i_b	I	**State Meaning**	Asserted: Not reset system. Negated: Reset system.	0
timers_clock_i	I	Clock interface to work timer/counter.		1
		State Meaning	Asserted: High Level of clock. Negated: Low Level of clock.	
		Timing	Asserted: Synchronous with external clock. Negated: Synchronous with external clock.	
timers_sfr_th0_i	I	8-bit Register (SFR)		0
timers_sfr_tm0_i	I	8-bit Register (SFR)		0
timers_sfr_tl0_i	I	8-bit Register (SFR)		0
timers_sfr_th0_o	O	8-bit Register (SFR)		0
timers_sfr_tm0_o	O	8-bit Register (SFR)		0
timers_sfr_tl0_o	O	8-bit Register (SFR)		0
timers_int0_i	I	External on/off timer 0		x
		State Meaning	Asserted: Turn on Timer 1 if GATE T1 = 0. Negated: Turn off Timer 1.	
		Timing	Asserted: Synchronous with external clock. Negated: Synchronous with external clock.	
timers_sfr_tmod_m0t0_i timers_sfr_tmod_m1t0_i	I	Timer 0 mode select bit		
		M1	M0	Operating Mode	Description
		0	0	0	24-bit up Timer
		0	1	1	Reserved

		1	0	2	Reserved
		1	1	3	24-bit down Timer
timers_sfr_tcon_tf0_i	Timer 0 overflow flag input to update the output is an overflow does not occur.				
timers_sfr_tcon_tf0_o	Timer 0 overflow flag. Set by hardware when the Timer 0 overflows. Cleared as processor vectors to the interrupt service routine.				
timers_sfr_tcon_tr0_i	Timer 0 run control bit. Set/cleared by software to turn Timer 0 ON/OFF.				

1.6 Block Timers Implementation

The Block of the Timers operates with two clocks, a coming of Baude Rate that it synchronizes the Timer 0 and 1 and another generated by the oscillator that synchronizes the Timer 2.
The component more critic of the Timer is it of the block SYNC should have the purpose of maintaining the metaestability of the Timer 2, without which will affect all the functionality of the Timer 2.

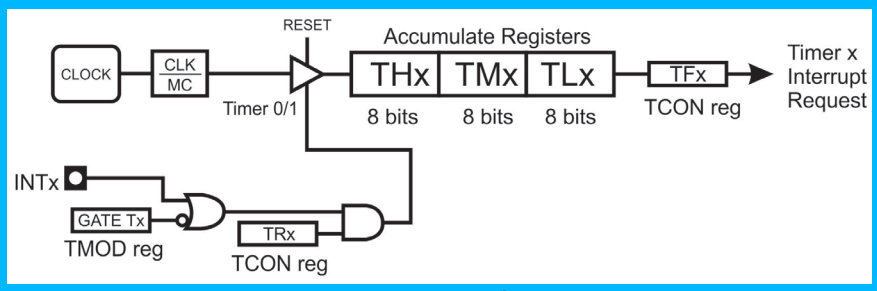

Figura 3 – Block Diagram of Timers 0 and 1

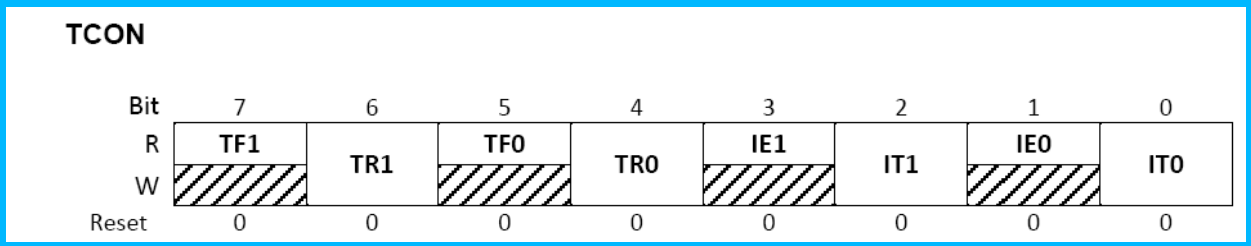

Figura 4 – TCON Register

The operation of timer/counter 0 and 1 are identical, using the descriptions for both below worked.

There are two ways to activate the timer; The timer 0 leaves in low level the bit **GATETx** inside **TMOD** register and set a bit **TRx** from **TCON** register**.** After this, it will now operates as a counter updating the register values **TLx**, **TMx**, **THx** all 8-bit, and finally reaching the maximum value (FFFFFF hex), on the next cycle, an overflow will occur, setting the interrupt flag, **TFx** on **TCON** register (bits 7 and 5) (Figure 2).
If the interrupt timers are enabled (register IE, bits: EA, ET2, ET1 e ET0), this may act in interrupts, otherwise whenever there is an overflow, the counter continues to count indefinitely until the flag TRx have been reset.

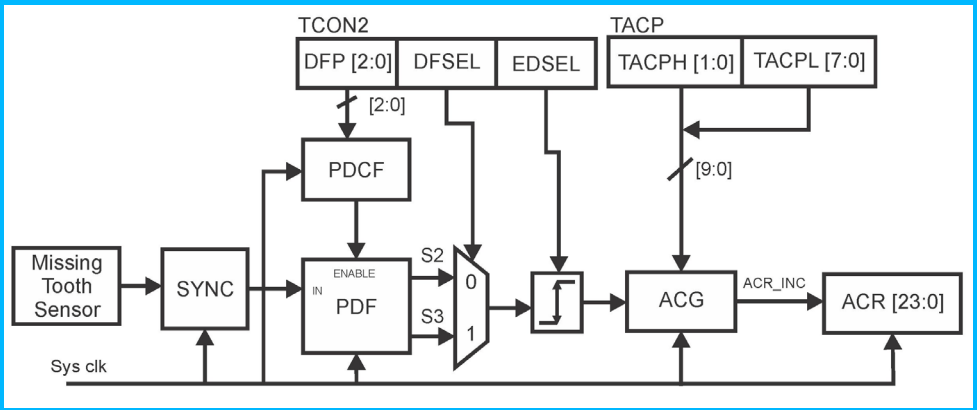

Figura 5 – Block Diagram of Timer 2

Theory of Operation:

The Figure 1 above display the diagram blocks of the Timer 2.
The sign that comes from Missing Tooth Sensor raisin for the block SYNC, composed of two Flip Flops, to provide the metaestability of the system is sent to PDF (Digital Programmable Filter) that accomplishes the sampling of the sign that comes in the way SYNC.
DFP (Prescaler Digital Filter Clock Period) is a registered that stores the divisor that will be used by PDCF (Prescaler Digital Clock Filter) that accomplishes the division of the frequency of the clock and he/she sends the sign that enables PDF to accomplish the sampling of the sign of SYNC.
The sign of PDF is sent to DFSEL (Digital Filter Selector) that operates in two: manners S2 (makes the validation of two samples) and S3 (makes the validation of three samples) that it is sent to EDSEL (Digital Edge Selector) after having validated that transform a wide pulse in a narrow pulse synchronized in the frequency of the clock that makes possible ACG (Angle Clock Generator) to accomplish the activities of the state machine.

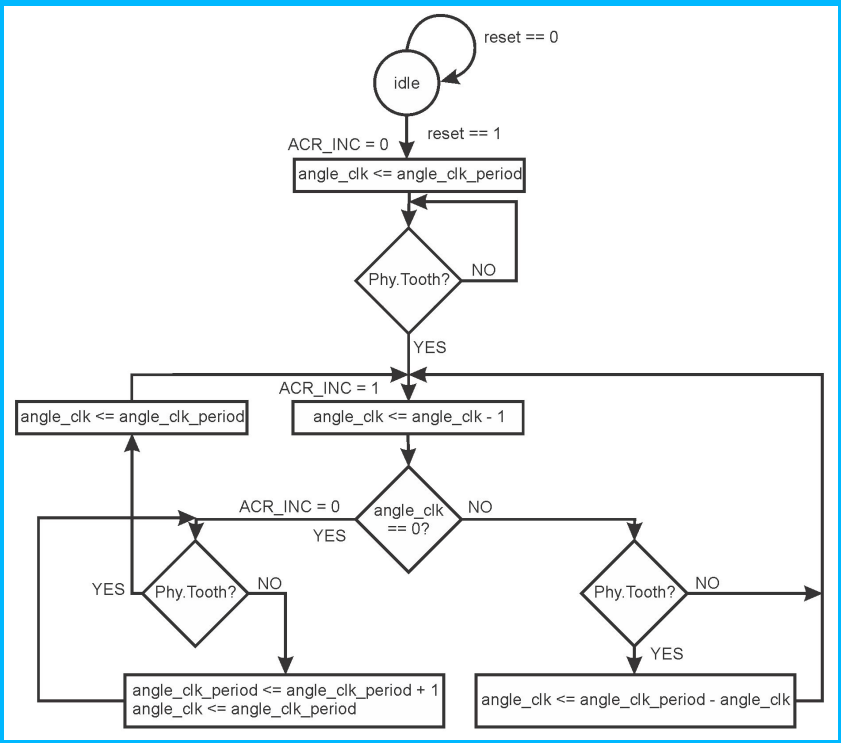

Figura 6 – AGC Microcode

The Figure 6 above display the microcode of operation of ACG.

The sign coming ACR_INC of the block ACG enables the register ACR of 24 bits to store or not the counting of the clock.
The data stored in the register ACR will be treated by ECU of the vehicle that will make possible to maintain the acting of operation of the motor.

1.7 Block Timers Configuration Parameters

No applicable to the block Timers

1.8 Block Timers Performance

The Block Timers has your limited performance for the clock that influences the power and speed.
Any technique was not applied critical to make the requirements of the specifications.
The speed of operation of the block Timers is a factor limitante in virtue of compatibilizar a sign of low frequency (Missing Sensor Tooth) with one of high frequency (Clock).
No critical paths in the block Timers.

1.9 Block Timers Design Trade-offs

No applicable to the block Timers

1.10 <Functional Block Name> Reuse Issues

As potential reuse of blocks of the implementation, can be mentioned the blocks of PDCF (Prescaler Digital Clock Period) and the state machine that it represents the microcode of the block ACG.

1.11 Design Methodology

No applicable to the block Timers

1.12 Tools Flow

The tools of Cadence were used, simulator SimVision, RTL Compiler, LEC in support to the development of the project.
In beginning the simulator was used SimVision to accomplish analyses of the behavior of the implemented block.
Soon afterwards the tool RTL Compiler was used to execute the synthesis, and after, the tool LEC to accomplish the equivalence among it Designates (RTL) him and Netlist (synthesis).