EMC08

SoC Guide

Version #: **1.0**

Revision date: **Nov. 3, 10**

Version Control

Revision	Authors	Date	Comments
1.0	Thiago Santos	3-Nov-10	Initial Version
1.1	Vinícius Amaral	3-Nov-10	Revision in Initial version

Table of Contents

EMC08

1 Introduction

EMC08 project is an 8-bit Microcontroller internal IP that could be used in future projects that require, embedded DIGITAL, AMS, RF and DSP blocks as part of the automotive system application.

The 8-bit Microcontroller has a CPU optimized for control applications, extensive Boolean processing capabilities, 4K bytes of on-chip program memory (ROM) address space, 128 bytes of on-chip data RAM, 32 bidirectional and 8 unidirectional and individually addressable I/O lines, three 24-bit timer/counters, full-duplex UART, vector interrupt structure with two priority levels.

The Analog blocks are basically 8-bit digital to analog converter and low noise 2.4 GHz RF Transceiver and Receiver digital wireless protocol based. The DSP will act as a baseband processing stage, which means it will be in charge of performing several algorithms for both the transmitter and the receiver.

This SoC Guide is focused on Digital module. The Analog and DSP modules will not be part of this team development, they are considered separated IPs which can be integrated in the SoC. However, there are two essentials analog sub-modules that are necessary to digital operation:

- Phase Locked Loop – PLL
- Power On Reset

The main clock can be provided by an external crystal oscillator, or optionally, can be used the 20MHz Low-Jitter Oscillator, an analog block. The PLL provides the clock used for memories and Power on Reset provides a reliable start up of the digital core. Other analog blocks are optionally too, as well as DSP module.

1.1 Overview

The EMC08 is 8 Bit Microcontroller based on Intel 8051 instruction set. This CPU is optimized for control applications and extensive Boolean processing capabilities, containing 4K bytes of on-chip program memory (ROM) address space, 128 bytes of on-chip data RAM, 32 bidirectional and individually addressable I/O lines, three 24-bit timer/counters, full-duplex UART, vector interrupt structure with two priority levels.

1.2 Features

The system includes these distinctive features:

- 8-bit CPU
  - Based on Intel 8051 Instruction Set
- 128 bytes of on-chip Data RAM
- 4K bytes of on-chip Program ROM
- Four 8-bit bidirectional parallel ports
- 8 Interrupt Sources - 7 vectors
  - External Interrupts 0 and 1
  - Timers 0, 1 and 2

- Serial Reception and Serial Transmission
- Transceiver Reception or Transmission

- Two 24-bit Timers
- One 24 bit Flywheel Angle Counter
- Serial Transmission
  - Synchronous or Asynchronous modes
  - Full Duplex capable
  - Fosc/2, Fosc/32, Fosc/64, 9600bps, 19200bps, 57600bps or 115200bps

1.3 Modes of Operation

The system has two basic modes of operation:

- Free Run Mode: General purpose
- Test Mode: ATPG test purpose

1.4 Block Diagram

Provide a top-level diagram that shows the functional organization of the system. Figure 1 is a block diagram of the system.

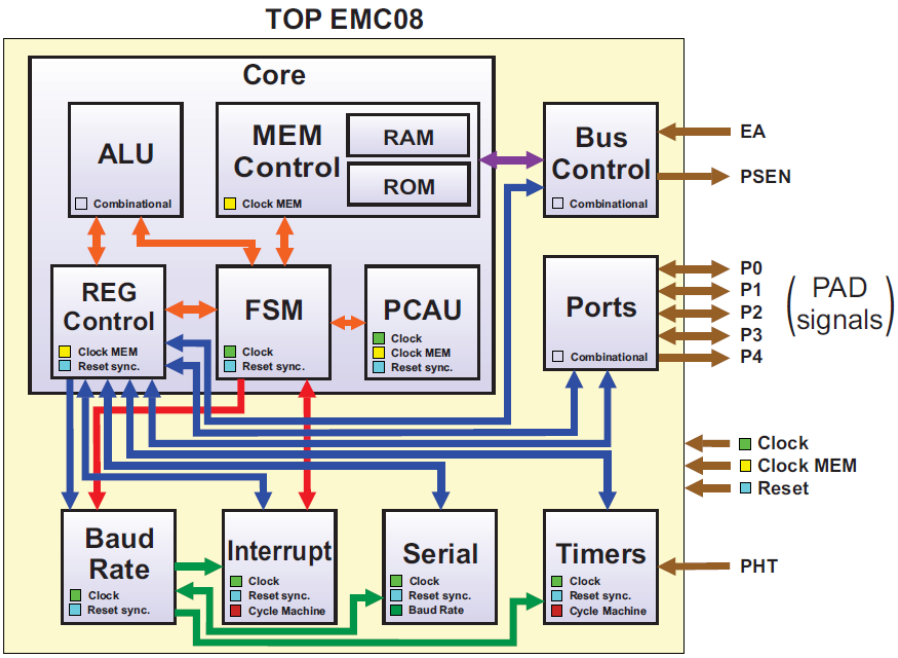


Figure 1 – System Block Diagram

1.5 System Memory Map

The system has three main internal storage locations: Internal ROM, Internal RAM and Special Function Registers Area. The Internal ROM is read only memory that stores the program data and

has 4Kbytes. The data storage area is divided into RAM and SFR. The first stores general data of the program and the second stores special configuration and internal state registers.

There are 4Kbytes of Internal ROM Memory, 128 Bytes of RAM Memory and 128 special Function Registers. The address space for Data and Program memory are different, but is shared between Data and SFR. Two modes of addressing are used: Direct and Indirect. The same address can represent different memory locations depending on the mode it's addressed. There is also the possibility of Byte or Bit addressing modes to RAM and SFR.

1.6 Device Memory Map

The tables 1 and 2 show the memory map for internal RAM and SFR area.

Table 1 – RAM Memory Map

Byte Address	Register	Access	Reset Value	Bit addres
General Registers				
0x00 – 0x07	R0 – R7: Registers Bank 0	R/W	0xXXXX_XXXX	--
0x08 – 0x0F	R0 – R7: Registers Bank 1	R/W	0xXXXX_XXXX	--
0x10 – 0x17	R0 – R7: Registers Bank 2	R/W	0xXXXX_XXXX	--
0x18 – 0x1F	R0 – R7: Registers Bank 3	R/W	0xXXXX_XXXX	--
0x20 – 0x2F	Bit addrable RAM Area	R/W	0xXXXX_XXXX	0x00 – 0x7F
0x30 – 0x7F	Ganeral Purpose RAM	R/W	0xXXXX_XXXX	--

Table 2– Special Function Registers Map

Byte Address	Register	Access	Reset Value	Bit addres
General Registers				
0x80	P0 – Port 0	R/W	0x0000_0000	0x80 .. 0x87
0x81	SP – Stack Pointer	R/W	0x0000_0111	--
0x82	DPL – Data Pointer Low	R/W	0x0000_0000	--
0x83	DPH – Data Pointer High	R/W	0x0000_0000	--
0x84	ACRL – Angle Counter Register Low	R/W	0x0000_0000	--
0x85	ACRM – Angle Counter Register Middle	R/W	0x0000_0000	--
0x86	ACRH – Angle Counter Register High	R/W	0x0000_0000	--
0x87	PCON – Power Control	R/W	0x0000_0000	--
0x88	TMOD – Timer/Counter Mode Control	R/W	0x0000_0000	0x88 .. 0x8F
0x89	TCON – Timer /Counter Control	R/W	0x0000_0000	--
0x8A	TL0 – Timer 0 Low Nibble	R/W	0x0000_0000	--
0x8B	TL1 – Timer 1 Low Nibble	R/W	0x0000_0000	--
0x8C	TH0 – Timer 0 High Nibble	R/W	0x0000_0000	--
0x8D	TH1 – Timer 1 High Nibble	R/W	0x0000_0000	--
0x8E	TM0 – Timer 0 Middle Nibble	R/W	0x0000_0000	--
0x8F	TM1 – Timer 1 Middle Nibble	R/W	0x0000_0000	--
0x90	P1 – Port 1	R/W	0x0000_0000	0x90 .. 0x97
0x91 – 0x97	Reserved	--	--	--
0x98	SCON – Serial Control	R/W	0x0000_0000	0x98 .. 0x9F
0x99	SBUF – Serial Buffer	R/W	0x0000_0000	--
0x9A – 0x9F	Reserved	--	--	--
0xA0	P2 – Port 2	R/W	0x0000_0000	0xA0 .. 0xA7
0xA1 – 0xA7	Reserved	--	--	--
0xA8	IE – Interrupt Enable	R/W	0x0000_0000	0xA8 .. 0xAF
0xA9	ACRL – Angle Counter Register Low	R/W	0x0000_0000	--
0xAA	ACRM – Angle Counter Register Middle	R/W	0x0000_0000	--
0xAB	ACRH – Angle Counter Register High	R/W	0x0000_0000	--
0xAC – 0xAE	Reserved	--	--	--

0xAF	P4 – Port 4	R/W	0x0000_0000	--
0xB0	P3 – Port 3	R/W	0x0000_0000	0xB0 .. 0xB7
0xB1 – 0xB7	Reserved	--	--	--
0xB8	IP – Interrupt Priority	R/W	0x0000_0000	0xB8 .. 0xBF
0XB9	SMAP8	R/W	0x0000_0000	--
0xBA	TACPL – Timer 2 Angle Clock Period Low	R/W	0x0000_0000	--
0xBB	TACPH – Timer 2 Angle Clock Period high	R/W	0x0000_0000	--
0xBC	RX1	R/W	0x0000_0000	--
0xBD	RX0	R/W	0x0000_0000	--
0xBE	TX1	R/W	0x0000_0000	--
0xBF	TX0	R/W	0x0000_0000	--
0xC0	P0EN – Port 0 Enable	R/W	0x0000_0000	0xC0 .. 0xC7
0xC1 – 0xC7	Reserved	--	--	--
0xC8	P1EN – Port 1 Enable	R/W	0x0000_0000	0xC8 .. 0xCF
0xC9 – 0xCF	Reserved	--	--	
0xD0	PSW – Program Status Word	R/W	0x0000_0000	0xD0 .. 0xD7
0xD1 – 0xD7	Reserved	--	--	--
0xD8	TCON2 – Timer 2 Control	R/W	0x0000_0000	0xD8 .. 0xDF
0xD9 – 0xDF	Reserved	--	--	--
0xE0	ACC – Accumulator	R/W	0x0000_0000	0xE0 .. 0xE7
0xE1 – 0xE7	Reserved	--	--	--
0xE8	P2EN – Port 2 Enable	R/W	0x0000_0000	0xE8 .. 0xEF
0xE9 – 0xEF	Reserved	--	--	--
0xF0	B – General Purpose Register	R/W	0x0000_0000	0xF0 .. 0xF7
0xF1 – 0xF7	Reserved	--	--	--
0xF8	P3EN – Port 3 Enable	R/W	0x0000_0000	0xF8 .. 0xFF
0xF9 – 0xFF	Reserved	--	--	--

1.7 Register Summary

Table 3 shows the format for a register summary table.

Table 3 – Register Summary

Byte Address Name		**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0x80 P0	R	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
	W								
	Reset	--	--	--	--	--	--	--	--
	Bit addr	0x87	0x86	0x85	0x84	0x83	0x82	0x81	0x80
0x81 SP	R	SP							
	W								
	Reset	0	0	0	0	0	1	1	1
	Bit addr	Not Bit addrable							
0x82 DPL	R	DPL							
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addrable							
0x83 DPH	**R**	DPH							
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addrable							
0x84 ACRL	**R**	ACRL							
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addrable							

Byte Address Name		7	6	5	4	3	2	1	0
0x85 ACRM	R	ACRM							
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addrable							
0x86 ACRH	R	ACRH							
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addressable							
0x87 PCON	R	PCON							
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addressable							
0x88 TCON	R	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	0x8F	0x8E	0x8D	0x8C	0x8B	0x8A	0x89	0x88
0x89 TMOD	R	GATE T1	0	M1 T1	M0 T1	0	GATE T0	M1 T0	M0 T0
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addressable							
0x8A TL0	R	TL0							
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addressable							
0x8B TL1	R	TL1							
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addressable							
0x8C TH0	R	TH0							
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addressable							
0x8D TH1	R	TH1							
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addressable							
0x8E TM0	R	TM0							
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addressable							
0x8F TM1	R	TM1							
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addressable							
0x90 P1	R	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
	W								
	Reset	--	--	--	--	--	--	--	--
	Bit addr	0x97	0x96	0x95	0x94	0x93	0x92	0x91	0x90
0x98 SCON	R	SM0	SM1	SM2	REN	TB8	RB8	TI	RI
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	0x9F	0x9E	0x9D	0x9C	0x9B	0x9A	0x99	0x98
0x99 SBUF	R	SBUF							
	W								

Byte Address Name		7	6	5	4	3	2	1	0
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addressable							
0xA0 P2	R W	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
	Reset	--	--	--	--	--	--	--	--
	Bit addr	0xA7	0xA6	0xA5	0xA4	0xA3	0xA2	0xA1	0xA0
0xA8 IE	R W	EA	ETXRX	ET2	ES	ET1	EX1	ET0	EX0
	Reset	0	0	0	0	0	0	0	0
	Bit addr	0xAF	0xAE	0xAD	0xAC	0xAB	0xAA	0xA9	0xA8
0xA9 ACRL	R W	ACR [7:0]							
	Reset	--	--	--	--	--	--	--	--
	Bit addr	Not Bit addressable							
0xAA ACRM	R W	ACR[15:8]							
	Reset	--	--	--	--	--	--	--	--
	Bit addr	Not Bit addressable							
0xAB ACRH	R W	ACR[23:16]							
	Reset	--	--	--	--	--	--	--	--
	Bit addr	Not Bit addressable							
0xAF P4	R W	P4.7	P4.6	P4.5	P4.4	P4.3	P4.2	P4.1	P4.0
	Reset	--	--	--	--	--	--	--	--
	Bit addr	Not Bit addressable							
0xB0 P3	R W	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0
	Reset	--	--	--	--	--	--	--	--
	Bit addr	0xB7	0xB6	0xB5	0xB4	0xB3	0xB2	0xB1	0xB0
0xB8 IP	R	0	PTXRX	PT2	PS	PT1	PX1	PT0	PX0
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	0xBF	0xBE	0xBD	0xBC	0xBB	0xBA	0xB9	0xB8
0xB9 SMAP8	R W	SMAP8							
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addressable							
0xBA SMAP8	R W	TACPL							
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addressable							
0xBB SMAP8	R	0	0	0	0	0	0	TACPH	
	W								
	Reset	0	0	0	0	0	0	0	0
0xBC RX1	R W	RX1							
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addressable							
0xBD RX0	R W	RX0							
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addressable							
0xBE	R	TX0							

Byte Address Name		7	6	5	4	3	2	1	0
TX0	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addressable							
0xBF TX1	R	TX1							
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	Not Bit addressable							
0xC0 P0EN	R	P0EN.7	P0EN.6	P0EN.5	P0EN.4	P0EN.3	P0EN.2	P0EN.1	P0EN.0
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	0xC7	0xC6	0xC5	0xC4	0xC3	0xC2	0xC1	0xC0
0xC8 P1EN	R	P1EN.7	P1EN.6	P1EN.5	P1EN.4	P1EN.3	P1EN.2	P1EN.1	P1EN.0
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	0xCF	0xCE	0xCD	0xCC	0xCB	0xCA	0xC9	0xC8
0xD0 PSW	R	CY	AC	F0	RS1	RS0	OV	0	P
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	0xD7	0xD6	0xD5	0xD4	0xD3	0xD2	0xD1	0xD0
0xD8 TCON2	R	0	TR2	TF2	DFP			DFSEL	EDGSEL
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	0xDF	0xDE	0xDD	0xDC	0xDB	0xDA	0xD9	0xD8
0xE0 ACC	R	ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	0xE7	0xE6	0xE5	0xE4	0xE3	0xE2	0xE1	0xE0
0xE8 P2EN	R	P2EN.7	P2EN.6	P2EN.5	P2EN.4	P2EN.3	P2EN.2	P2EN.1	P2EN.0
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	0xEF	0xEE	0xED	0xEC	0xEB	0xEA	0xE9	0xE8
0xF0 B	R	B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	0xF7	0xF6	0xF5	0xF4	0xF3	0xF2	0xF1	0xF0
0xF8 P3EN	R	P3EN.7	P3EN.6	P3EN.5	P3EN.4	P3EN.3	P3EN.2	P3EN.1	P3EN.0
	W								
	Reset	0	0	0	0	0	0	0	0
	Bit addr	0xFF	0xFE	0xFD	0xFC	0xFB	0xFA	0xF9	0xF8

2 Signal Description

The following sections will describe the system pinout, properties and detailed discussion of signals that connect off chip.

2.1 System Pinout

Figure 2 shows the external pinout diagram of the system.

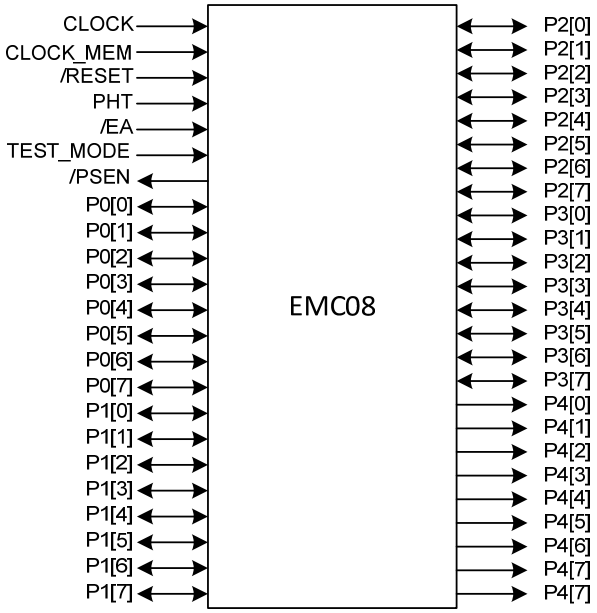


Figure 2 – Pinout Diagram

2.2 Signal Properties Summary

Table 4 – Signal Properties

Name	Secondary	Port	Function	Reset	I/O	Termination		Loading pF
						Type	Direction	
CLOCK	--	--	System Clock	-	I	Active	Pull up and Pull Down	10
CLOCK_MEM	--	--	Memory Clock	-	I	Active	Pull up and Pull Down	10
RESET	--	--	Synchronous System Reset	-	I	Active	Pull up and Pull Down	10
P0[7:0]	SI[7:0]	P0[7:0]	Bidirectional Port 0	-	I/O	Active	Pull up and Pull Down	10
P1[7:0]	SI[15:8]	P1[7:0]	Bidirectional Port 1	-	I/O	Active	Pull up and Pull Down	10
P2[7:0]	SO[7:0]	P2[7:0]	Bidirectional Port 2	-	I/O	Active	Pull up and Pull Down	10
P3[7:0]	--	P3[7:0]	Bidirectional Port 3	-	I/O	Active	Pull up and Pull Down	10
P4[7:0]	SO[15:8]	P4[7:0]	Output Port 4	0000 0000	O	Active	Pull up and Pull Down	10
PHT	--	--	Flywheel Tooth Sensor Input	-	I	Active	Pull up and Pull Down	10
/EA	SCAN_ENABLE	--	External Access Input	-	I	Active	Pull up and Pull	10

Name	Secondary	Port	Function	Reset	I/O	Termination		Loading
							Down	
TEST_MODE	--	--	Test Mode Enable Input	-	I	Active	Pull up and Pull Down	10
/PSEN	--	--	Program Store Enable Output	1	O	Active	Open Drain Output	10

2.3 Detailed Signal Descriptions

The following table describes each signal listed.

Table 5 – Detailed Signal Descriptions

Signal	I/O	Description	
CLOCK	I	System Clock. Generated by PLL analog Module.	
		State Meaning	Asserted – Memory Clock High Level Negation – Memory Clock low Level
		Timing	Assertion/Negation – 50ns Period, 50% Duty Cycle
CLOCK_ MEM	I	Memory Clock. Generated by PLL analog Module.	
		State Meaning	Asserted – Memory Clock High Level Negation – Memory Clock low Level
		Timing	Assertion/Negation – 25ns Period, 50% Duty Cycle
/RESET	I	Synchronous System Reset	
		State Meaning	Asserted – Reset Inactive Negated – Reset Active
		Timing	Assertion/Negation – May occur at any time. Must remain asserted for normal device operation
P0[0:7]	I/O	Bidirectional Port 0	
		State Meaning	Asserted – High Level Input or Output Negated – Low Level Input or Output
		Timing	Input Assertion/Negation – May occur at any time Output Assertion/Negation – Synchronous with Memory Clock
P1[0:7]	I/O	Bidirectional Port 1	
		State Meaning	Asserted – High Level Input or Output Negated – Low Level Input or Output
		Timing	Input Assertion/Negation – May occur at any time Output Assertion/Negation – Synchronous with Memory Clock
P2[0:7]	I/O	Bidirectional Port 2	
		State Meaning	Asserted – High Level Input or Output Negated – Low Level Input or Output
		Timing	Input Assertion/Negation – May occur at any time Output Assertion/Negation – Synchronous with Memory Clock
P3[0:7]	I/O	Bidirectional Port 3	
		State Meaning	Asserted – High Level Input or Output Negated – Low Level Input or Output
		Timing	Input Assertion/Negation – May occur at any time Output Assertion/Negation – Synchronous with Memory Clock
P4[0:7]	O	Output Port 0	
		State Meaning	Asserted – High Level Output Negated – Low Level Output
		Timing	Assertion/Negation – Synchronous with Memory Clock
PHT	I	Flywheel Tooth Sensor Input	
		State Meaning	Asserted – Flywheel Tooth present Negated – Flywheel Tooth absent

Signal	I/O	Description	
		Timing	Assertion/Negation – May be asserted any time
EA	I	External Access Input	
		State Meaning	Asserted – High Level Output Negated – Low Level Output
		Timing	Assertion/Negation – Synchronous with Memory Clock
/PSEN	O	Program Store Enable Output	
		State Meaning	Asserted – High Level Output Negated – Low Level Output
		Timing	Assertion/Negation – Synchronous with Memory Clock
TEST_MODE	I	Test Mode Enable Input	
		State Meaning	Asserted – Test Mode enabled Negated – Test Mode disabled
		Timing	Assertion/Negation – Synchronous with Memory Clock
VDD	I	Digital Supply 1.62V to 1.98V	
VDDR	I	Digital Supply 3.0V to 3.6V	
GND	I	Digital Ground	

3 System Clock Description

The digital module uses four domains of clock: two external signals and two internally derived signals.

CLOCK: A main clock of 20 MHz. This clock is input of top module and is distributed for all sequential blocks. This signal comes from Low-Jitter Oscillator analog block, or, if this analog block is not being used, from the external crystal oscillator.

CLOCK_MEM: A derived and synchronous clock for memories. PLL analog block will send this signal to core with twice the clock frequency. It was chosen to use this clock because the memories are very fast, and there are a lot of instructions that needs many reads and writes operations. As is not possible change the main clock frequency because it is a project specification, the design team opts to use another clock domain to control memories. So, it is possible make read or write memory operations in each semi-clock period.

cycle_machine: The cycle machine is an internal clock signal produced by Baud Rate digital module and sent to Interrupt and Timers modules. Each instruction of instruction set can be performed in one or two cycles of cycle machine. One cycle machine has two periods of main clock.

baud_rate_trans: Baud rate transition signal is an internal clock signal, derived from main clock too, and it is a 16 times faster than Baud rate signal. It is generated by Baud Rate digital module and is sent to Serial module only. This clock signal is variable and can assume a multiple value of clock period.

The following figure illustrates the relation between the three fixed clock domains.

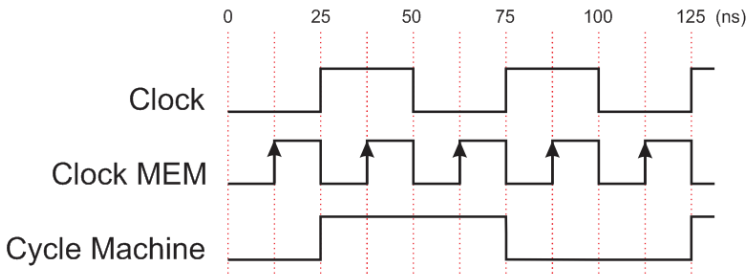


Figure 3 – Relation between fixed clock domains

4 Modes of Operation

4.1 Overview

The system has two basic modes of operation, the Test Mode, for ATPG test proposes; and the Free Run Mode, for general propose.

4.2 Free Run Mode

Free Run is the normal mode of operation. In this mode the microcontroller executes instructions provided by internal or external ROM and all blocks can work according it specific function. The functionality of each block will be discussed later in this document.

4.3 Test Mode

There are thirty-four pins used for DFT in this project, one is exclusive for DFT and others are shared with the default system functions. The chip has a dedicated pin, TEST_MODE used to set the chip mode to Free Run Mode or to Test Mode. This dedicated pin is active high, so for a normal operation, this input must remain in low level. The second test mode control signal is shared with the EA pin. This input has the function of activate the scan mode. There are sixteen scan chains in the chip. The scan chains inputs and outputs are shared with port P0, P1, P2 and P4. Ports P0 and P1 pins are used as scan chain inputs and P2 and P4 are used as outputs. The ports P0, P1 and P2 are bidirectional ports, so when the chip is set to Test Mode these ports direction are automatically set to match the inputs/outputs needs. The same system pins CLOCK and CLOCK_MEM are used for scan chains.

The test controller is implemented through a TM pin. During the execution test it value is equal at one, if in any moment the value of TM pin change to zero, the chip exit of test mode. There are too the scan enable pin, that make the shift in scan chains for improve the more testability resources.

5 Resets and Interrupts

The system has a synchronous reset, low activated. The interrupt module is responsible to generate interrupt requests to Core. It evaluates the priority of different interrupt sources which can occur at same time and decides what of them should be executed.

5.1 Overview

EMC08 chips uses different management for reset and interrupts systems.

The reset signal is as input of chip and is redistributed by Core to other module. There are no reset vectors.

Interrupts are treated by Interrupt module, which evaluates the interruption flags present at the SFR and communicates with Core sending interrupt requests.

5.2 Vectors

The interrupt block has a total of seven interrupt vectors: two external interrupts (INT0 and INT1), three timer interrupts (Timers 0, 1, and 2), serial port interrupt (TX and RX), and transceiver interrupt (TXRX).

The vector address depends on the source of the interrupt, as shown in table below.

5.2.1 Vector Table

The table below provides a table that lists all interrupt vectors for the system.

Table 6 – Vector Summary

Number	Description	Address
IE0	External Interrupt 0	0003H
TF0	Timer 0 overflow Interrupt	000BH
IE1	External Interrupt 1	0013H
TF1	Timer 1 overflow Interrupt	001BH
TF2	Timer 1 overflow Interrupt	0023H
RI + TI	Serial Transmission or Reception Interrupt	002BH
TXRX	Tranceiver Interrupt	0033H

5.2.2 Vector Base Register

Not Applicable.

5.3 Resets

The “Power-On-Reset” analog module is used to provide a reliable start up of the digital core. The circuit asserts the reset signal after a fixed delay triggered, that need to be grater then 250 ns due to memory initialization time. For reset requests after initialization, the reset input signal (/RESET) must be asserted for at least one machine cycle.

The reset signal is processed by the Core, which sends a derived signal (core_reset_o) to all other modules. This signal is synchronized with negative edge of clock due to initialization aspects of the system.

5.3.1 Reset Summary Table

The system has a single reset activated by an external input.

Table 7 – Reset Summary

Reset	Priority	Source	Characteristics
System Reset	10	External	Synchronous, Active Low

5.4 Interrupts

The chip has eight interruption sources, three external and five internal. The external sources are: external interrupts 0 and 1 that uses ports P3[2] and P3[3] respectively, and Transceiver Reception or Transmission. The internal sources are: Timers 0, 1 and 2, and Serial Transmission or Reception.

5.4.1 Interrupt Summary Table

List all sources of interrupt service requests in a table. Show pertinent information concerning each interrupt, including interrupt name, source of service request, priority (highest priority first) and vector information. Table below shows example information.

Table 8 – Interrupt Summary

Interrupt	Address	Vector	Priority	Source	Description
EXTERNAL 0	0003	000	IP[0]	Port P3[2]	Low Level or Falling Edge Active, depending on bit IT0
TIMER 0	0013	001	IP[1]	Timer 0 Module	Timer 0 Overflow
EXTERNAL 1	000B	010	IP[2]	Port P3[3]	Low Level or Falling Edge Active, depending on bit IT1
TIMER 1	001B	011	IP[3]	Timer 1 Module	Timer 1 Overflow
TIMER 2	0023	100	IP[4]	Timer 2 Module	Timer 2 Overflow
SERIAL	002B	101	IP[5]	Serial Module	Serial Transmission or Reception
TRANSCEIVER TXRX	0033	110	IP[6]	RF Module	Transceiver Transmission or Reception

5.4.2 Interrupt Summary Table

The figure below shows how the IE and IP registers and the polling sequence work to determine which if any interrupt will be serviced.

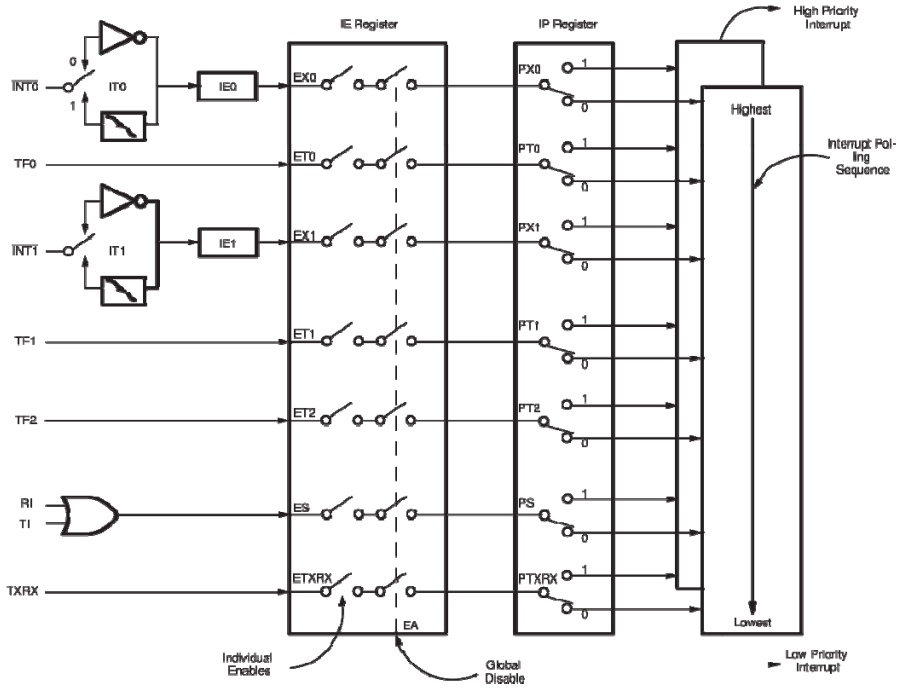


Figure 4 – Interrupt Controller Algorithm Model

The External Interrupts INT0 and INT1 can each be either level activated or transition-activated, depending on bits IT0 and IT1 in Register TCON. The flags that actually generate these interrupts are cleared by the hardware when the service routine is vectored to only if the interrupt was

transition-activated. If the interrupt was level-activated, then the external requesting source is what controls the request flag, rather than the on-chip hardware.

The Timer 0, Timer 1 and Timer 2 Interrupts are generated by TF0, TF1 (TCON) and TF2 (TCON2) which are set by a rollover in their respective Timer registers. When a timer interrupt is generated, the flag that generated it is cleared by the on-chip hardware when the service routine is vectored to.

The Serial Port Interrupt is generated by the logical OR of RI and TI. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine will normally have to determine whether it was RI or TI that generated the interrupt, and the bit will have to be cleared in software.

All of the bits that generate interrupts can be set or cleared by software, with the same result as though it had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be canceled in software.

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE. IE contains also a global disable bit, EA, which disables all interrupts at once. Refers to REGS section for IE register details.

6 Core Block Description

6.1 Introduction

The CORE of EMC08 module is composed by 4 major modules that are FSM, Memory Control, Registers Control and ALU, as showed in figure 1.

The main function of core is fetching the instructions either from memory (user instructions) or hardware based instructions, decode the op-code (if has any), identify the sub-module responsible to compute the desired data and wait for its response.

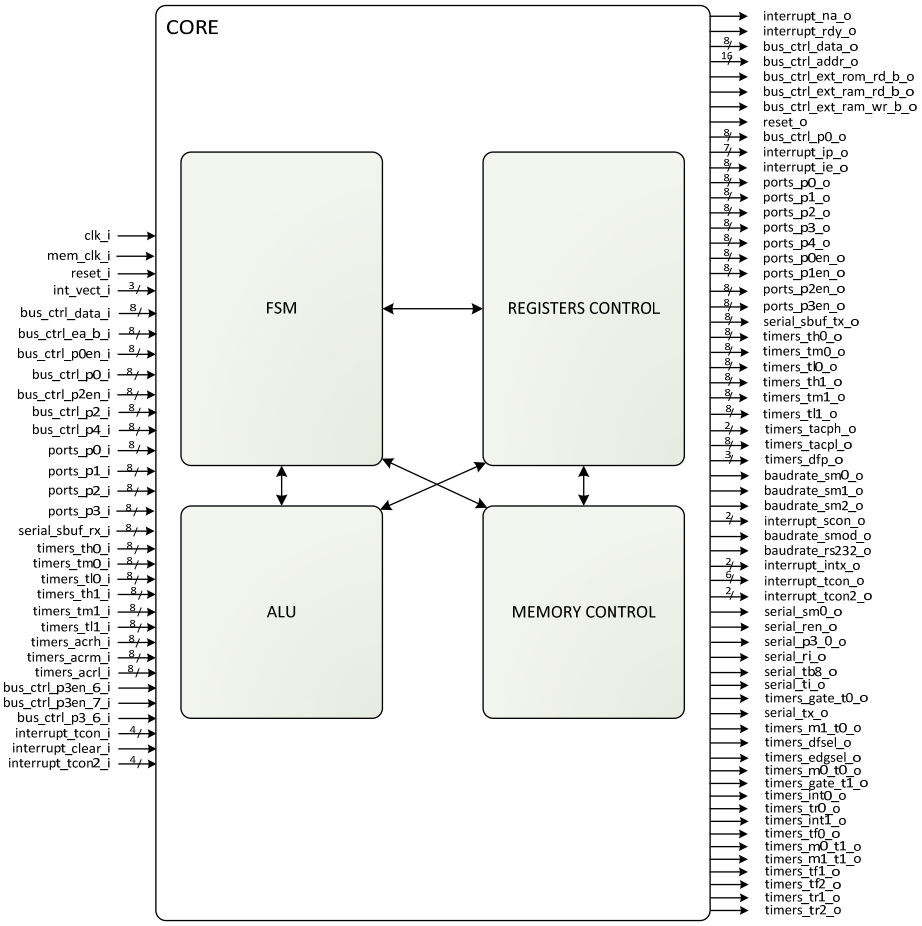


Figure 5 – Core block diagram

6.2 Overview

6.2.1 Core

The core is a group of sub-modules which have arithmetic, logical and data manipulation capabilities. These capabilities will be activated by software or hardware demand. Mainly these software-based instructions resize in an internal 4k bytes ROM memory but may reside in an external ROM or even internal/external RAM.

The communication between CORE and other modules may differ from one module to another. Some of them will interact with CORE only by reading/writing special flags stored in SFR space such as Timers, Serial and Ports Blocks; while orders may need an additional flags (register) and bus inside the module which hold a status of the signal and/or operation such as Interrupt Block.

6.2.2 FSM

The FSM module represents the finite state machine responsible to manage instructions fetched from memory or another peripheral and send control signals depending on which sub-module must be activated.

Data instructions and jump instructions are managed by FSM while arithmetical, logical and Boolean instructions are managed by ALU.

In addition, FSM module must control the PC register, incrementing it based on the previous instruction or based on address offset from a normal program flow or even increment based in a hardware-like instruction. Figure 2 show the FSM diagram composed by five states (including reset) which are activated by positive edge or negative edge of the clock signal.

This way the design implements a fixed architecture that allows us to create and implement consistent instructions.

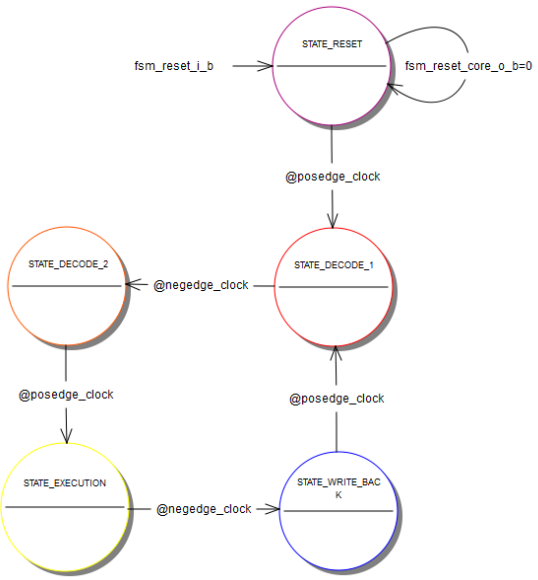


Figure 6 – FSM states

6.2.3 ALU

The ALU is the module responsible for any Boolean or arithmetic operation needed by the core. It's divided into six parts or units: Selector, Mult, Div, Add/Sub, Boolean and BCD Setting. The Selector was changed from a sub-module to a multiplexer that will receive the instruction opcode and operands from the FSM and decide what unit is the responsible for the desired operation. The operation will be executed and the result will be returned to the FSM for future usage or storage.

6.2.3.1 Selector

The Selector unit is responsible for receive the opcode decoded from FSM that translated these opcodes to a new one that contain only ALU instructions and identify what kind of operation should occur. So it sends the operands to the specific unit that will perform that operation and send back to FSM the result.

6.2.3.2 BCD Setting

The BCD Setting unit is responsible by implementation of the DA (Decimal adjust Accumulator for Addition) instruction. This instruction adjusts the eight-bit value in the Accumulator resulting from the earlier addition of two variables (each in packed-BCD format), producing two four-bit digits. Any ADD or ADDC instruction may have been used to perform the addition.

If Accumulator bits 3-0 are greater than nine (xxxx1010-xxxx1111), or if the AC flag is one, six is added to the Accumulator producing the proper BCD digit in the low order nibble. This internal

addition would set the carry flag if a carry-out of the low order four-bit field propagated through all high-order bits, but it would not clear the carry flag otherwise.

If the carry flag is now set, or if the four high-order bits now exceed nine (1010xxxx111xxxx), these high-order bits are incremented by six, producing the proper BCD digit in the high-order nibble. Again, this would set the carry flag if there was a carry out of the high-order bits, but wouldn't clear the carry. The carry flag thus indicates if the sum of the original two BCD variables is greater than 100, allowing multiple precision decimal addition. OV is not affected.

All of this occurs during the one instruction cycle. Essentially, this instruction performs the decimal conversion by adding 00H, 06H, 60H, or 66H to the Accumulator, depending on initial Accumulator and PSW conditions.

6.2.3.3 Mult

The Mult unit is responsible by implementation of multiplication operation. This operation multiplies the unsigned eight-bit integers in the Accumulator and register B. The Iow-order byte of the sixteen-bit product is left in the Accumulator, and the high order byte in B. If the product is greater than 255 (0FFH) the overflow flag is set; otherwise it is cleared. The carry flag is always cleared.

6.2.3.4 Add/Sub

The Add/Sub unit is responsible by implementation of addition and subtraction operation.

The ADD operation simultaneously adds the byte variable indicated, the carry flag and the Accumulator contents, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not out of bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

The SUBB operation subtracts the indicated variable and the carry flag together from the Accumulator, leaving the result in the Accumulator. SUBB sets the carry (borrow) flag if a borrow is needed for bit 7, and clears C otherwise. (If C was set before executing a SUBB instruction, this indicates that a borrow was needed for the previous step in a multiple precision subtraction, so the carry is subtracted from the Accumulator along with the source operand). AC is set if a borrow is needed for bit 3, and cleared otherwise. OV is set if a borrow is needed into bit 6, but not into bit 7, or into bit 7, but not bit 6.

When subtracting signed integers OV indicates a negative number product when a negative value is subtracted from a positive value or a positive result when a positive number is subtracted from a negative number.

The source operand allows four addressing modes: register, direct, register-indirect, or immediate.

6.2.3.5 Div

The Div unit is responsible by implementation of division operation. This operation divides the unsigned eight-bit integer in the Accumulator by the unsigned eight-bit integer in register B. The Accumulator receives the integer part of the quotient; register B receives the integer remainder.

The carry and OV flags will be cleared. Exception: if B had originally contained 00H, the values returned in the Accumulator and B register will be undefined and the overflow flag will be set. The carry flag is cleared in any case.

6.2.3.6 Boolean

The Boolean unit is responsible by implementation of all Boolean operations of ALU, i.e., and, or, xor, not, and comparison.

6.2.4 Memory Control

The overall function of the Memory Control unit is receiving addresses from the FSM and decides where to look up the data. There are three types of memory avaiable: Data Memory, Program Memory that utilizes RAMs and ROMs, respectively, for storage. The FSM will identify if the desired data is stored in the Data Memory or Program Memory and send signals requesting the data from the Memory Control.

There are 128 Bytes of RAM and 4Kbytes of ROM internally to the Core. The address space for Data and Program memory are different, but is shared between Data and SFR. Two modes of addressing are used: Direct and Indirect. The same address can represent different memory locations depending on the mode it's addressed.

All internal memories are embedded in the Memory Control. The comunnication beetween FSM and memory control is made by different data and adress buses for ROM and RAM, which means that the FSM can request more than one memory access each time. There is a signal that indicates if the address represents a byte or Bit addr. This bit and byte address mapping will be described in next sections.

There is a signal that indicates if the RAM access is internal or external. If the requested data is located at internal RAM, the memory access will perform the operation. If the access is requesting an external RAM read or write, the memory control must transfer the request to the Bus Control module.

The location of ROM access is determined by two features: the external EA pin and the address. When the EA port is active, all program data accesses are external. If the EA port is not active, the location is indicated by the desired address. If the address is between 0000H and 0FFFH the access is to internal ROM, if the address is between 1000H and FFFFH the access is to external ROM. Internal accesses are made directly by the memory control and external accesses are transferred to the bus control module.

6.2.5 Registers Control

The Special Function Registers consists of 128 8-Bit registers used for configuration, temporary storage and internal control, but only 40 of these registers are actually used. Some of these registers can be written or/and read by more than one external module as shown in the table below.

Table 9 – Special Function Registers Accesses

Register	Core		Ports		Serial		Interrupt		Timers		Baud Rate		Bus Control	
	RD	WR	RD	WR	RD	WR	RD	WR	RD	WR	RD	WR	RD	WR
ACC	X	X												
B	X	X												

PSW	X	X												
IP	X	X					X							
IE	X	X					X							
P0	X	X	X	X									X	X
P1	X	X	X	X										
P2	X	X	X	X									X	X
P3	X	X	X	X	X	X	X							X
P4	X	X	X										X	
P0EN	X	X	X											X
P1EN	X	X	X											
P2EN	X	X	X											X
P3EN	X	X	X			X		X						X
SCON	X	X			X	X	X	X			X			
TCON	X	X					X	X	X	X				
TCON2	X	X					X	X	X	X				
SBUF (RX)	X	X				X								
SBUF (TX)	X	X			X									
TH0	X	X							X	X				
TH1	X	X							X	X				
TM0	X	X							X	X				
TM1	X	X							X	X				
TL0	X	X							X	X				
TL1	X	X							X	X				
TMOD	X	X							X					
RX0	X	X												
RX1	X	X												
TX0	X	X												
TX1	X	X												
SMAP8	X	X												
TACPL	X	X							X	X				
TACPH	X	X							X	X				
PCON	X	X									X			
DPH	X	X												
DPL	X	X												
ACRH	X	X							X	X				
ACRM	X	X							X	X				
ACRL	X	X							X	X				
SP	X	X												

At any moment, several modules can read any register without problems, but only one module can write data to each of them. The Register Control is the unit responsible to decide which module will have the write permission. These accesses can be decided looking into other registers, giving priorities to some modules or through control signals sent by the FSM.

The FSM submodule has the maximum priority to write the registers. When a write instruction have to be made, only this submodule will have access to the desired registers, all other modules attempts to write some data will be ignored. The conflicts between two or more modules and the way they are resolved are described in the table below

Table 10 – Special Function Registers Write Conflicts

Shared Register	Write Conditions
P3[1:0], P3EN[1:0]	The FSM will provide a signal *reg_ctrl_fsm_serial_tx* that informs to the serial modules when there is some data to be transmitted. The reception is enabled by the REN register. When the serial transmission/reception is enabled the Serial module will have access to

	the registers, Ports attempt to write will be ignored.
P3[3:2], P3EN[3:2]	When the External Interruptions are enabled, the ports are configured as inputs and the Interrupt module will have write permission to the registers, otherwise the ports are used as general purpose input/output and will be configured by the user.
P0, P2, P3[7:6], P4, P0EN, P2EN, P3EN[7:6]	This conflict is resolved by an internal signal sent by the Memory Control submodule. When an external memory access is requested to the Memory Control, this module will send a signal indicating to the Register Control that the bus control need write permission to the registers.
Interrupt Flags TF0, TF1, TF2, RI, TI	This conflict is resolved by an internal signal sent by the Interruption module. The interruption module has the priority to write the flags, once that in some occasions these flags must be cleared. When the Interruption module needs to clear one of these signals, the *interrupt_clear* signal is asserted, and the write permission to the flag is granted only to the Interruption module. Otherwise, each module can write to their own flags.

6.3 Features

- Core features: 8-bit CPU;
- Extensive Boolean processing (single-bit logic) capabilities;
- 128 bytes of on-chip Data RAM;
- 4K bytes of on-chip Program ROM;
- 128 8-Bit Special Function Registers;
- 255 arithmetical, boolean and logical instructions;
- 4 cycle state machine;
- 2 clock cycles per machine cycle.

6.4 Modes of operation

Basically the core operates in single mode all the time. Meaning that it will read the program from ROM memory and fetching the next instruction until find an interruption. This way the core module receives interruptions from other modules through the Interrupt block. The current instruction must be finished before threat the incoming interruption.

After the previous request has been completed the core (through the FSM sub-module) switches back to program memory, reading and fetching software-based instructions.

6.5 External signal description

Core module connects externally direct only with RST, VDD and VSS pin. All other external pins are connected to other modules that compound the EMC08.

6.6 Detailed signal description

The complete interface of Core module is described below. The interface description includes internal ports and external pins.

Table 11 – Interface description

Signal	I/O	Description		Reset
core_reset_i	I	System reset signal		N/A
		State Meaning	Asserted: Reset inactive Negated: Reset active	
		Timing	Synchronous signal	
core_clk_i	I	System Main clock originated by analog PLL module		N/A
		State Meaning	Asserted: clock high cycle Negated: clock low cycle	
		Timing	50ns period, 50% duty cycle	
core_mem_clk_i	I	System Memory clock originated by analog PLL module		N/A
		State Meaning	Asserted: clock high cycle Negated: clock low cycle	
		Timing	25ns period, 50% duty cycle	
core_bus_control_p0_en_i	I	Port 0 Enable signal incoming from Bus Control		0
		State Meaning	Asserted: Port set as output Negated: Port set as input	
		Timing	Asserted any time, registered at Memory Clock positive edge	
core_bus_control_p0_i	I	Port 0 signal incoming from Bus Control		00000000
		State Meaning	Asserted: High level being driven to port Negated: Low level being driven to port	
		Timing	Asserted any time, registered at Memory Clock positive edge	
core_bus_control_p2_en_i	I	Port 2 Enable signal incoming from Bus Control		0
		State Meaning	Asserted: Port set as output Negated: Port set as input	
		Timing	Asserted any time, registered at Memory Clock positive edge	
core_bus_control_p2_i	I	Port 2 signal incoming from Bus Control		00000000
		State Meaning	Asserted: High level being driven to port Negated: Low level being driven to port	
		Timing	Asserted any time, registered at Memory Clock positive edge	
core_bus_control_p3_6_en_i	I	Port 3.6 Enable signal incoming from Bus Control		0
		State Meaning	Asserted: Port set as output Negated: Port set as input	
		Timing	Asserted any time, registered at Memory Clock positive edge	
core_bus_control_p3_6_i	I	Port 3.6 signal incoming from Bus Control		0
		State Meaning	Asserted: High level being driven to port Negated: Low level being driven to port	
		Timing	Asserted any time, registered at Memory Clock positive edge	
core_bus_control_p3_7_en_i	I	Port 3.7 Enable signal incoming from Bus Control		0
		State Meaning	Asserted: Port set as output Negated: Port set as input	

Signal	I/O	Description					Reset
		Timing	Asserted any time, registered at Memory Clock positive edge				
core_bus_control_p3_7_i	I	Port 3.7 signal incoming from Bus Control					0
		State Meaning	Asserted: High level being driven to port Negated: Low level being driven to port				
		Timing	Asserted any time, registered at Memory Clock positive edge				
core_bus_control_p4_i	I	Port 4 signal incoming from Bus Control					00000000
		State Meaning	Asserted: High level to be driven to port Negated: Low level to be driven to port				
		Timing	Asserted any time, registered at Memory Clock positive edge				
core_bus_control_data_i	I	External data read by bus control					00000000
		State Meaning	Asserted: High level to be driven to port Negated: Low level to be driven to port				
		Timing	Asserted any time				
core_interrupt_vect_i[2:0]	I	Interruption Signal					000
		State Meaning	Bit 2	Bit 1	Bit 0	Interruption	
			0	0	0	None	
			0	0	1	External 0	
			0	1	0	Timer 0	
			0	1	1	External 1	
			1	0	0	Timer 1	
			1	0	1	Timer 2	
			1	1	0	Serial (TX or RX)	
			1	1	1	Transceiver	
		Timing	Asserted any time, evaluated before a new instruction is fetched				
core_interrupt_tcon_i	I	TCON Register signals being drive from Interruption module. Compound by TFTXRX and TF2 IE0 flags respectively.					0000
		State Meaning	Asserted: High level to be stored to register Negated: Low level to be stored to register				
		Timing	Asserted any time, registered at Memory Clock positive edge				
core_interrupt_tcon2_i	I	TCON2 Register signals being drive from Interruption module. Compound by TF1, TF0, IE1, IE0 flags respectively.					00
		State Meaning	Asserted: High level to be stored to register Negated: Low level to be stored to register				
		Timing	Asserted any time, registered at Memory Clock positive edge				
core_interrupt_clear_i	I	Interruption clear request signal sent by interruption module.					0
		State Meaning	Asserted: Requesting an flag clear Negated: No clear to be made				
		Timing	Asserted any time, evaluated at Memory Clock positive edge				
core_bus_control_p0_i	I	Port 0 signal incoming from Bus Control					0
		State Meaning	Asserted: High level being driven to port Negated: Low level being driven to port				
		Timing	Asserted any time				
		State Meaning	State Meaning				
		Timing	Timing				

<table>
<tr><th>Signal</th><th>I/O</th><th colspan="6">Description</th><th>Reset</th></tr>
<tr><td rowspan="11"></td><td rowspan="11"></td><td colspan="6"></td><td rowspan="11"></td></tr>
<tr><td rowspan="10">State Meaning</td><td colspan="5">Interruption to be treated</td></tr>
<tr><td>Bit 2</td><td>Bit 1</td><td>Bit 0</td><td colspan="2">Int</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td colspan="2">None</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td colspan="2">INT0</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td colspan="2">TF0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td colspan="2">INT1</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td colspan="2">TF1</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td colspan="2">TF2</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td colspan="2">RI + TI</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td colspan="2">TX/RX</td></tr>
<tr><td>Timing</td><td colspan="6">Assertion: May be asserted any time
Negation: May be negated any time.</td></tr>
<tr><td rowspan="3">ext_ram_rd</td><td rowspan="3">O</td><td colspan="7">External RAM Read Signal</td><td rowspan="3">1</td></tr>
<tr><td>State Meaning</td><td colspan="6">Asserted: External RAM Read Disabled
Negated: External RAM Read Enabled</td></tr>
<tr><td>Timing</td><td colspan="6">Assertion: Synchronous with external clock
Negation: Synchronous with external clock</td></tr>
<tr><td rowspan="3">ext_ram_wr</td><td rowspan="3">O</td><td colspan="7">External RAM Write Signal</td><td rowspan="3">1</td></tr>
<tr><td>State Meaning</td><td colspan="6">Asserted: External RAM Write Disabled
Negated: External RAM Write Enabled</td></tr>
<tr><td>Timing</td><td colspan="6">Assertion: Synchronous with external clock
Negation: Synchronous with external clock</td></tr>
<tr><td rowspan="3">ext_rom_rd</td><td rowspan="3">O</td><td colspan="7">External ROM Read Signal</td><td rowspan="3">1</td></tr>
<tr><td>State Meaning</td><td colspan="6">Asserted: External ROM Read Disabled
Negated: External ROM Read Enabled</td></tr>
<tr><td>Timing</td><td colspan="6">Assertion: Synchronous with external clock
Negation: Synchronous with external clock</td></tr>
<tr><td rowspan="3">ext_addr [15:0]</td><td rowspan="3">O</td><td colspan="7">External memory address bus</td><td rowspan="3">XXXXH</td></tr>
<tr><td>State Meaning</td><td colspan="6">Asserted: n/a
Negated: n/a</td></tr>
<tr><td>Timing</td><td colspan="6">Assertion: Must be available before read/write signals be asserted, synchronous with external clock
Negation: Synchronous with external clock</td></tr>
<tr><td rowspan="3">ext_data [7:0]</td><td rowspan="3">O</td><td colspan="7">External memory data bus</td><td rowspan="3">XXH</td></tr>
<tr><td>State Meaning</td><td colspan="6">Asserted: n/a
Negated: n/a</td></tr>
<tr><td>Timing</td><td colspan="6">Assertion: Synchronous with external clock
Negation: Synchronous with external clock</td></tr>
<tr><td rowspan="3">ext_rom_rd</td><td rowspan="3">O</td><td colspan="7">External ROM Read Signal</td><td rowspan="3">1</td></tr>
<tr><td>State Meaning</td><td colspan="6">Asserted: External ROM Read Disabled
Negated: External ROM Read Enabled</td></tr>
<tr><td>Timing</td><td colspan="6">Assertion: Synchronous with external clock
Negation: Synchronous with external clock</td></tr>
</table>

6.7 Memory map and register definition

The memory map for Core consists of 128 Bytes RAM Memory, 4Kbytes ROM and 128 8-Bit SFR. Some areas are bit and byte addressable and others are only byte addressable. These details are indicated as follows.

6.8 Memory map

6.8.1 Data Memory

The Data Memory consists of a 128 Bytes RAM that can be divided into three sections: Registers Banks that stores the R0-R7 registers and is selected through a bit in the PSW register, Bit addrable General Purpose and Byte Addressable General Purpose. The address length is 8-Bit and the memory division is showed in table below.

Table 12 – Data Memory Map

Byte Address		Bit addr							
Not Bit addrable	7F 30	General Purpose RAM							
Bit addrable	2F	7F	7E	7D	7C	7B	7A	79	78
	2E	77	76	75	74	73	72	71	70
	2D	6F	6E	6D	6C	6B	6A	69	68
	2C	67	66	65	64	63	62	61	60
	2B	5F	5E	5D	5C	5B	5A	59	58
	2A	57	56	55	54	53	52	51	50
	29	4F	4E	4D	4C	4B	4A	49	48
	28	47	46	45	44	43	42	41	40
	27	3F	3E	3D	3C	3B	3A	39	38
	26	37	36	35	34	33	32	31	30
	25	2F	2E	2D	2C	2B	2A	29	28
	24	27	26	25	24	23	22	21	20
	23	1F	1E	1D	1C	1B	1A	19	18
	22	17	16	15	14	13	12	11	10
	21	0F	0E	0D	0C	0B	0A	09	08
	20	07	06	05	04	03	02	01	00
Not Bit addrable	1F 18	BANK 3							
	17 10	BANK 2							
	0F 08	BANK 1							
	07 00	BANK 0							

6.9 Program Memory

The internal program memory consists of a 4Kbytes ROM memory with 16-Bit length addresses. The memory space is continuous but some areas dedicated to interruptions routines. If the interruptions are not used, these areas are available for general use. The full program memory space is described below.

Table 13 – Program Memory Map

Description	Address
General Use	FFFFH
	0034H
Interruption Routines	0033H
	002BH
	0023H
	001BH

	0013H
	000BH
	0003H
Reset	0000H

6.9.1 Special Function Registers

The Special Function Registers are Core internal registers that stores data for configuration, control, temporary storage and others. There is an available area for up to of 128 8-Bit registers, but only thirty eight registers are implemented at this moment. Generally hits registers are byte addressable, but some of them are Bit addressable too. The location and address mode of each one of the registers is described below.

Table 14 – Special Function Register Map

Register	Bit addr								Byte Address
Reserved	Not Bit addressable								FF – F9
P3EN	FF	FE	FD	FC	FB	FA	F9	F8	F8
Reserved	Not Bit addressable								F7 – F1
B	F7	F6	F5	F4	F3	F2	F1	F0	F0
Reserved	Not Bit addressable								EF – E9
P2EN	EF	EE	ED	EC	EB	EA	E9	E8	E8
Reserved	Not Bit addressable								E7 – E1
ACC	E7	E6	E5	E4	E3	E2	E1	E0	E0
Reserved	Not Bit addressable								DF – D9
TCON2	DF	DE	DD	DC	DB	DA	D9	D8	D8
Reserved	Not Bit addressable								D7 – D1
PSW	D7	D6	D5	D4	D3	D2	D1	D0	D0
Reserved	Not Bit addressable								CF – C9
P1EN	CF	CE	CD	CC	CB	CA	C9	C8	C8
Reserved	Not Bit addressable								C7 – C1
P0EN	C7	C6	C5	C4	C3	C2	C1	C0	C0
TX0	Not Bit addressable								BF
TX1	Not Bit addressable								BE
RX0	Not Bit addressable								BD
RX1	Not Bit addressable								BC
TACPH	Not Bit addressable								BB
TACPL	Not Bit addressable								BA
SMAP8	Not Bit addressable								B9
IP	BF	BE	BD	BC	BB	BA	B9	B8	B8
Reserved	Not Bit addressable								B7 – B1
P3	B7	B6	B5	B4	B3	B2	B1	B0	B0
P4	Not Bit addressable								AF
Reserved	Not Bit addressable								AE – A9
IE	AF	AE	AD	AC	AB	AA	A9	A8	A8
Reserved	Not Bit addressable								A7 – A1
P2	A7	A6	A5	A4	A3	A2	A1	A0	A0
Reserved	Not Bit addressable								9F – 9A
SBUF	Not Bit addressable								99
SCON	9F	9E	9D	9C	9B	9A	99	98	98
Reserved	Not Bit addressable								97 – 91
P1	97	96	95	94	93	92	91	90	90
TM1	Not Bit addressable								8F
TM0	Not Bit addressable								8E
TH1	Not Bit addressable								8D
TH0	Not Bit addressable								8C

TL1	Not Bit addressable								8B
TL0	Not Bit addressable								8A
TMOD	Not Bit addressable								89
TCON	8F	8E	8D	8C	8B	8A	89	88	88
PCON	Not Bit addressable								87
ACRH	Not Bit addressable								86
ACRM	Not Bit addressable								85
ACRL	Not Bit addressable								84
DPH	Not Bit addressable								83
DPL	Not Bit addressable								82
SP	Not Bit addressable								81
P0	87	86	85	84	83	82	81	80	80

6.10 Register Description

6.10.1 P0 – Port 0 Input/Output

Bit	7	6	5	4	3	2	1	0
R W	**P0.7**	**P0.6**	**P0.5**	**P0.4**	**P0.3**	**P0.2**	**P0.1**	**P0.0**
Reset	1	1	1	1	1	1	1	1

Table 15 – P0 Description

Field	Description
P0.[7:0]	Port 0 Input/Output Level 0: Low level Output/Input. 1: High Level Output/Input.

6.10.2 P1 – Port 1 Input/Output

Bit	7	6	5	4	3	2	1	0
R W	**P1.7**	**P1.6**	**P1.5**	**P1.4**	**P1.3**	**P1.2**	**P1.1**	**P1.0**
Reset	1	1	1	1	1	1	1	1

Table 16 – P1 Description

Field	Description
P1.[7:0]	Port 1 Input/Output Level 0: Low level Output/Input. 1: High Level Output/Input.

6.10.3 P2 – Port 2 Input/Output

Bit	7	6	5	4	3	2	1	0
R W	**P2.7**	**P2.6**	**P2.5**	**P2.4**	**P2.3**	**P2.2**	**P2.1**	**P2.0**

Reset	1	1	1	1	1	1	1	1

Table 17 – P2 Description

Field	Description
P2.[7:0]	Port 2 Input/Output Level 0: Low level Output/Input. 1: High Level Output/Input.

6.10.4 P3 – Port 3 Input/Output

Bit	7	6	5	4	3	2	1	0
R W	**P3.7**	**P3.6**	**P3.5**	**P3.4**	**P3.3**	**P3.2**	**P3.1**	**P3.0**
Reset	1	1	1	1	1	1	1	1

Table 18 – P3 Description

Field	Description
P3.[7:0]	Port 3 Input/Output Level 0: Low level Output/Input. 1: High Level Output/Input.

6.10.5 P4 – Port 4 Output

Bit	7	6	5	4	3	2	1	0
R								
W	**P4.7**	**P4.6**	**P4.5**	**P4.4**	**P4.3**	**P4.2**	**P4.1**	**P4.0**
Reset	0	0	0	0	0	0	0	0

Table 19 – P4 Description

Field	Description
P4.[7:0]	Port 4 Output Level 0: Low level Output. 1: High level Output.

6.10.6 P0EN – Port 0 Enable

Bit	7	6	5	4	3	2	1	0
R W	**P0EN.7**	**P0EN.6**	**P0EN.5**	**P0EN.4**	**P0EN.3**	**P0EN.2**	**P0EN.1**	**P0EN.0**
Reset	0	0	0	0	0	0	0	0

Table 20 – P0EN Description

Field	Description

P0EN.[7:0]	Port 0 I/O Enable Bit 0: Pin set as input. 1: Pin set as output.

6.10.7 P1EN – Port 1 Enable

Bit	7	6	5	4	3	2	1	0
R W	**P1EN.7**	**P1EN.6**	**P1EN.5**	**P1EN.4**	**P1EN.3**	**P1EN.2**	**P1EN.1**	**P1EN.0**
Reset	0	0	0	0	0	0	0	0

Table 21 – P1EN Description

Field	Description
P1EN.[7:0]	Port 1 I/O Enable Bit 0: Pin set as input. 1: Pin set as output.

6.10.8 P2EN – Port 2 Enable

Bit	7	6	5	4	3	2	1	0
R W	**P2EN.7**	**P2EN.6**	**P2EN.5**	**P2EN.4**	**P2EN.3**	**P2EN.2**	**P2EN.1**	**P2EN.0**
Reset	0	0	0	0	0	0	0	0

Table 22 – P2EN Description

Field	Description
P2EN.[7:0]	Port 2 I/O Enable Bit 0: Pin set as input. 1: Pin set as output.

6.10.9 P3EN – Port 3 Enable

Bit	7	6	5	4	3	2	1	0
R W	**P3EN.7**	**P3EN.6**	**P3EN.5**	**P3EN.4**	**P3EN.3**	**P3EN.2**	**P3EN.1**	**P3EN.0**
Reset	0	0	0	0	0	0	0	0

Table 23 – P3EN Description

Field	Description
P3EN.[7:0]	Port 3 I/O Enable Bit 0: Pin set as input. 1: Pin set as output.

6.10.10 SP – Stack Pointer

Bit	7	6	5	4	3	2	1	0
R W	**SP[7:0]**							
Reset	0	0	0	0	0	1	1	1

Table 24 – SP Description

Field	Description
SP	Stack Pointer

6.10.11 DPL – Data Pointer Low

Bit	7	6	5	4	3	2	1	0
R W	**DPL[7:0]**							
Reset	0	0	0	0	0	0	0	0

Table 25 – DPL Description

Field	Description
DPL[7:0]	Data Pointer Lower Nibble

6.10.12 DPH – Data Pointe High

Bit	7	6	5	4	3	2	1	0
R W	**DPH[7:0]**							
Reset	0	0	0	0	0	0	0	0

Table 26 – DPH Description

Field	Description
DPH[7:0]	Data Pointer Higher Nibble

6.10.13 ACRL – Angle Counter Register Low

Bit	7	6	5	4	3	2	1	0
R W	**ACRL[7:0]**							
Reset	0	0	0	0	0	0	0	0

Table 27 – ACRL Description

Field	Description

ACRL[7:0]	Angle Counter Register Lower Nibble

6.10.14 ACRM – Angle Counter Register Middle

Bit	7	6	5	4	3	2	1	0
R W	**ACRM[7:0]**							
Reset	0	0	0	0	0	0	0	0

Table 28 – ACRM Description

Field	Description
ACRM[7:0]	Angle Counter Register Middle Nibble

6.10.15 ACRH – Angle Counter Register High

Bit	7	6	5	4	3	2	1	0
R W	**ACRH[7:0]**							
Reset	0	0	0	0	0	0	0	0

Table 29 – ACRH Description

Field	Description
ACRH[7:0]	Angle Counter Register Higher Nibble

6.10.16 PCON – Power Control

Bit	7	6	5	4	3	2	1	0
R	**SMOD**					**RS232**		
W								
Reset	0	0	0	0	0	0	0	0

Table 30 – PCON Description

Field	Description
SMOD	Double baud rate bit. If Timer 1 is used to generate baud rate and SMOD=1, the baud rate is doubled when the Serial Port is used in modes 1, 2 or 3.
RS232	RS232 communication mode active or not

6.10.17 TCON – Timer/Counter Control

Bit	7	6	5	4	3	2	1	0
R	**TF1**	**TR1**	**TF0**	**TR0**	**IE1**	**IT1**	**IE0**	**IT0**

W								
Reset	0	0	0	0	0	0	0	0

Table 31 – TCON Description

Field	Description
TF1	Timer 1 overflow flag. Set by hardware when the Timer 1 overflows. Cleared as processor vectors to the interrupt service routine.
TR1	Timer 1 run control bit. Set/cleared by software to turn Timer 1 ON/OFF.
TF0	Timer 0 overflow flag. Set by hardware when the Timer 0 overflows. Cleared as processor vectors to the interrupt service routine.
TR0	Timer 0 run control bit. Set/cleared by software to turn Timer 0 ON/OFF.
IE1	External Interrupt 1 edge flag. Set by hardware when External Interrupt edge is detected. Cleared by hardware when interrupt is processed.
IT1	Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.
IE0	External Interrupt 0 edge flag. Set by hardware when External Interrupt edge is detected. Cleared by hardware when interrupt is processed.
IT0	Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.

6.10.18 TMOD – Timer/Counter Mode Control

Bit	7	6	5	4	3	2	1	0
R	**GATE T1**		**M1 T1**	**M0 T1**	**GATE T0**		**M1 T0**	**M0 T0**
W								
Reset	0	0	0	0	0	0	0	0

Table 32 – TMOD Description

Field	Description			
GATE T1	When TR1 (in TCON) is set and GATE T1=1, Timer/Counter 1 will run only while INT1 pin is high (hardware control). When GATE T1=0, Timer/Counter1 will run only while TR1=1 (software control).			
TMOD[6]	Reserved			
M[1:0] T1	Timer 1 selector 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
GATE T0	When TR0 (in TCON) is set and GATE T0=1, Timer/Counter 0 will run only while INT0 pin is high (hardware control). When GATE T0=0, Timer/Counter0 will run only while TR0=1 (software control).			
TMOD[2]	Reserved			
M[1:0] T0	Timer 1 selector 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

6.10.19 TL1 – Timer 1 Low

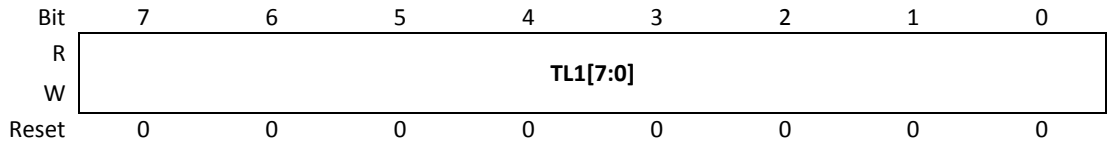


Table 33 – TL1 Description

Field	Description
TL1[7:0]	Timer 1 Lower Nibble

6.10.20 TL0 – Timer 0 Low

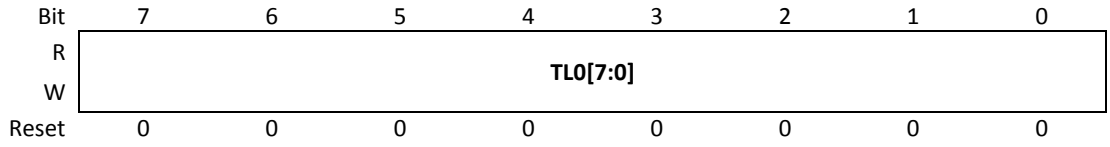
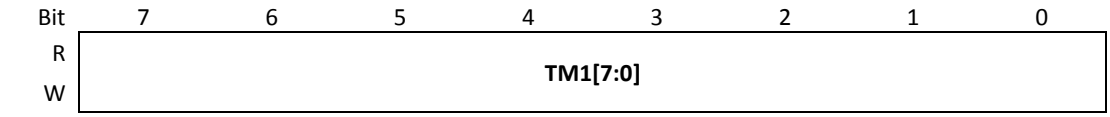


Table 34 – TL0 Description

Field	Description
TL0[7:0]	Timer 0 Lower Nibble

6.10.21 TM1 – Timer 1 Middle



Reset	0	0	0	0	0	0	0	0

Table 35 – TM1 Description

Field	Description
TM1[7:0]	Timer 1 Middle Nibble

6.10.22 TM0 – Timer 0 Middle

Bit	7	6	5	4	3	2	1	0
R W	**TM0[7:0]**							
Reset	0	0	0	0	0	0	0	0

Table 36 – TM0 Description

Field	Description
TM0[7:0]	Timer 0 Middle Nibble

6.10.23 TH1 – Timer 1 High

Bit	7	6	5	4	3	2	1	0
R W	**TH1[7:0]**							
Reset	0	0	0	0	0	0	0	0

Table 37 – TH1 Description

Field	Description
TH1[7:0]	Timer 1 Higher Nibble

6.10.24 TH0 – Timer 0 High

Bit	7	6	5	4	3	2	1	0
R W	**TH0[7:0]**							
Reset	0	0	0	0	0	0	0	0

Table 38 – TH0 Description

Field	Description
TH0[7:0]	Timer 0 Higher Nibble

6.10.25 SCON – Serial Port Control

Bit	7	6	5	4	3	2	1	0

R W	**SM0**	**SM1**	**SM2**	**REN**	**TB8**	**RB8**	**TI**	**RI**
Reset	0	0	0	0	0	0	0	0

Table 39 – SCON Description

Field	Description				
SM[1:0]	Timer 1 selector bit.				
	SM0	SM1	Operating Mode	Description	Baud Rate
	0	0	0	Shift Register	Fosc/N
	0	1	1	8-Bit UART	Variable
	1	0	2	9-Bit UART	Fosc/64 or Fosc/32
	1	1	3	9-Bit UART	Variable
SM2	Enables the multiprocessor communication feature in modes 2 and 3. In mode 2 or 3, if SM2=1 then RI will not be active if the received 9th data bit (RB8) is 0. In mode 1, if SM2=1 then RI will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0.				
REN	Set/Cleared by software to enable/disable reception.				
TB8	The 9th bit will be transmitted in modes 2 and 3. Set/cleared by software.				
RB8	In modes 2 and 3, is the 9th data bit was received. In mode 1, if SM2=0, RB8 is stop bit that was received. In mode 0, RB8 is not used.				
TI	Transmit interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or halfway through the stop bit time in the other modes (except see SM2).				
RI	Receive interrupt flag. Set by hardware at the end of the 8th bit in mode 0, or halfway through the stop bit time in the other modes (except see SM2). Must be cleared by software.				

6.10.26 SBUF – Serial Buffer

Bit	7	6	5	4	3	2	1	0
R W	**SBUF[7:0]**							
Reset	0	0	0	0	0	0	0	0

Table 40– SBUF Description

Field	Description
SBUF[7:0]	SBUF is actually two separate registers: a transmit buffer and a receive buffer register. When data is moved to SBUF, it goes to transmit buffer where it is held for serial transmission. (Moving a byte to SBUF initiates the transmission). When data is moved from SBUF, it comes from the receive buffer.

6.10.27 IE – Interrupt Enable

Bit	7	6	5	4	3	2	1	0
R W	**EA**	**ETXRX**	**ET2**	**ES**	**ET1**	**EX1**	**ET0**	**EX0**
Reset	0	0	0	0	0	0	0	0

Table 41 – IE Description

Field	Description
EA	Enable all interrupts. If EA=0, no interrupt will be acknowledged. If EA=1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
ETXRX	Transceiver interrupt enable bit
ET2	Timer 2 interrupt enable bit.
ES	Serial port interrupt enable bit.
ET1	Timer 1 overflow interrupt enable bit.
EX1	External interrupt 1 enable bit.
ET0	Timer 0 overflow interrupt enable bit.
EX0	External interrupt 0 enable bit.

6.10.28 IP – Interrupt Priority

Bit	7	6	5	4	3	2	1	0
R W		**PTXRX**	**PT2**	**PS**	**PT1**	**PX1**	**PT0**	**PX0**
Reset	0	0	0	0	0	0	0	0

Table 42 – IE Description

Field	Description
IP[7]	Reserved
TXRX	Transceiver interrupt priority bit.
PT2	Timer 2 interrupt priority bit.
PS	Serial port interrupt priority bit.
PT1	Timer 1 interrupt priority bit.
PX1	External Interrupt 1 priority bit.

PT0	Timer 0 interrupt priority bit.
PX0	External Interrupt 0 priority bit.

6.10.29 SMAP8

Bit	7	6	5	4	3	2	1	0
R W	**SMAP8[7:0]**							
Reset	0	0	0	0	0	0	0	0

Table 43 – SMAP8 Description

Field	Description
SMAP8[7:0]	Input from the ADC from MAP sensors

6.10.30 TACPL – Timer 2 Angle Clock Period Low

Bit	7	6	5	4	3	2	1	0
R W	**TACPL[7:0]**							
Reset	0	0	0	0	0	0	0	0

Table 44 – TACPL Description

Field	Description
TACPL[7:0]	Timer 2 Low Byte Angle Clock Period.

6.10.31 TACPH – Timer 2 Angle Clock Period High

Bit	7	6	5	4	3	2	1	0
R							**TACPH[1]**	**TACPH[0]**
W								
Reset	0	0	0	0	0	0	0	0

Table 45 – TACPH Description

Field	Description
TACPH[1:0]	Timer 2 High Byte Angle Clock Period.

6.10.32 TX1

Bit	7	6	5	4	3	2	1	0
R W	**TX1[7:0]**							

Reset	0	0	0	0	0	0	0	0

Table 46 – TX1 Description

Field	Description
TX1[7:0]	High Byte Transmitter Data Register

6.10.33 TX0

Bit	7	6	5	4	3	2	1	0
R W	**TX0[7:0]**							
Reset	0	0	0	0	0	0	0	0

Table 47 – TX0 Description

Field	Description
TX0[7:0]	Low Byte Transmitter Data Register

6.10.34 RX1

Bit	7	6	5	4	3	2	1	0
R W	**RX1[7:0]**							
Reset	0	0	0	0	0	0	0	0

Table 48 – RX1 Description

Field	Description
RX1[7:0]	High Byte Receiver Data Register

6.10.35 RX0

Bit	7	6	5	4	3	2	1	0
R W	**RX0[7:0]**							
Reset	0	0	0	0	0	0	0	0

Table 49 – RX0 Description

Field	Description
RX0[7:0]	Low Byte Receiver Data Register

6.10.36 PSW - Program Status Word

Bit	7	6	5	4	3	2	1	0

R	**CY**	**AC**	**F0**	**RS1**	**RS0**	**OV**		**P**
W								
Reset	0	0	0	0	0	0	0	0

Table 50 – PSW Description

Field	Description				
CY	Carry flag receives carry out from bit 7 of ALU operands.				
AC	Auxiliary carry flag receives carry out from bit 3 of addition operands.				
F0	General purpose status flag.				
RS[1:0]	Register Bank Select				
	RS0	RS1	Register Bank	Address Range	
	0	0	0	00H	07H
	0	1	1	08H	0FH
	1	0	2	10H	17H
	1	1	3	18H	1FH
OV	Overflow flag set by arithmetic operations				
PSW[1]	Reserved				
P	Parity of accumulator set by hardware to 1 if it contains an odd number of 1s. Otherwise it is reset to 0.				

6.10.37 TCON2 – Timer 2 Control

Bit	7	6	5	4	3	2	1	0
R		**TR2**	**TF2**	**DFP[4:2]**			**DFSEL**	**EDGSEL**
W								
Reset	0	0	0	0	0	0	0	0

Table 51 – TCON2 Description

Field	Description
TCON2[7]	Reserved
TR2	Timer 2 run control bit. Set/cleared by software to turn Timer 2 ON/OFF.
TF2	Timer 2 overflow flag. Set by hardware when the Timer 2 overflows. Cleared as processor vectors to the interrupt service routine.
DFP[4:2]	Digital Filter Clock Period
DFSEL	Digital Filter Sampling Selection 0: DF output Sampling S2 selected

	1: DF output Sampling S3 selected
EDGSEL	Rise-Fall Edge selection 0: Fall edge selection 1: Rise edge selection

6.10.38 ACC – Accumulator

Bit	7	6	5	4	3	2	1	0
R W	**ACC[7:0]**							
Reset	0	0	0	0	0	0	0	0

Table 52 – ACC Description

Field	Description
ACC[7:0]	Accumulator data

6.10.39 B – General Purpose Register

Bit	7	6	5	4	3	2	1	0
R W	**B[7:0]**							
Reset	0	0	0	0	0	0	0	0

Table 53 – ACC Description

Field	Description
B[7:0]	B data

6.11 Functional Description

The core has a complex set of inputs and outputs that varies along with the entire system state which means that simulate this system requires a complete set of stimulus provided by near modules.

At first glance, a reset stimuli will record 0 (zero) on the Program Counter, ALU registers, MEM_CTRL and REG_CTRL flags and restart the FSM to state S0; the following table shows the registers that must be zeroed.

Table 47 – Functional Description

Sub-module	Register	Reset value
PC	previous_address	16'b0
PC	Address	16'b0
FSM	State	d'0 (S0)
FSM	Opcode	7'b0

FSM	operand1	8'b0
FSM	operand2	8'b0
ALU	operand1	8'b0
ALU	operand2	8'b0
ALU	Opcode	7'b0
ALU	Accumulator	8'b0

Many registers must be zeroed but the great majority resides in the SFR area. Refer to section 1.6.2 to know which registers must be zeroed and what initial value they must hold. All the previous explanation about how to reset the core module is just a simple description of the reset core behavior while the /RESET pin is active.

The core can access the current instruction by fetching it from the memory address pointed by PC, identify its type (by using an opcode set previously defined) and do a lot of computations accordingly to.

The sub-module FSM will be responsible to control the PC increment, identify the instruction, call the MEM_CTRL module, retrieve the operands, if is any, and send it to the correct sub-module. If any instruction or internal instruction, i.e. interrupt, need access to internal Special Functions Registers, FSM will ask the REG_CTRL module to do it so.

If any logical or arithmetical instruction is called the FSM will pass all useful information to the ALU module trough the ALU's sub-module called Selector, who in turn will send it to inner sub-modules.

After complete all related computations the result should be stored. An entire cycle is called Machine Cycle.

The core module does not generate any interrupt. Instead, receive every interrupt from other modules of the system either by hardware or software means.

6.12 Initialization Information

The Core starts its continuous operation after a Reset signal is applied. The initial configurations of the external modules are described in table below. After this initialization, user should provide instruction to modify these configurations:

Table 54 – Initial Configuration

Module	Mode of Operation
Ports	P0: Output P1: Output P2: Output P3: Output P4: Output
Bus Control	Set by external memory configuration pins (EA_b)
Timer	Timer 0: 24-Bit Up Counter - Off Timer 1: 24-Bit Up Counter - Off Timer 2: Two Sample Mode, Fall Edge - Off
Interrupt	All Interrupts Disabled, All Priorities set to Low
Baud Rate	Oscillator frequency / N
Serial	Mode 0: 8-Bit Data, fixed Baud Rate

Below are described some examples of user instructions that can be used to change the peripherals configuration.

Set ports P0, P1, P2 and P3 as inputs:

```
MOV P0EN, #F0h          // Moves F0h to P0 Enable Register
```

Set Timer 0 as down counter controlled by software, with enabled interruption and begins counting:

```
MOV A, #00000011b;      // Loads 00000011b to Accumulator
ORL A, TMOD             // A OR TMOD: Sets the desired bits in TMOD
SETB EA;                // Enable Interrupts
SETB ET0;               // Set Timer 0 Interruption
SETB TR1;               // Starts count by setting TR1 flag
```

6.13 Instruction Set Description

6.13.1 ACALL:

Function: Absolute Call

Description:

The ACALL instruction calls a subroutine located at the specified address. The PC is incremented twice to obtain the address of the following instruction. The 16-bit PC is then stored on the stack (low-order byte first) and the stack pointer is incremented twice. No flags are affected.

The address of the subroutine is calculated by combining the 5 high-order bits of the incremented PC (for A15-A11), the 3 high-order bits of the ACALL instruction opcode (for A10-A8), and the second byte of the instruction (for A7-A0). The subroutine that is called must be located in the same 2KByte block of program memory as the opcode following the ACALL instruction.

Table 55 – ACALL Description.

Instruction	Bytes	Cycles	Encoding	Operation
ACALL	2	2	| a10 a9 a8 1 | 0 0 0 1 | | a7 a6 a5 a4 | a3 a2 a1 a0 |	(PC) ← (PC) + 2 (SP) ← (SP) + 1 ((SP)) ← (PC_{7-0}) (SP) ← (SP) + 1 ((SP)) ← (PC_{15-8}) (PC_{10-0}) ← page address

6.13.2 ADD:

Function: Add

Description:

The ADD instruction adds a byte value to the accumulator and stores the results back in the accumulator. Several of the flag registers are affected.

Table 50 – ADD Description.

Instruction	Bytes	Cycles	Encoding	Operation
ADD_A_RR	1	1	| 0 0 1 0 | 1 r r r |	A ← A + RR

ADD_A_D	2	1	| 0 0 1 0 | 0 1 0 1 | |direct address |	A ← A + D
ADD_A_ATRI	1	1	| 0 0 1 0 | 0 1 1 i |	A ← A + ATRI
ADD_A_DATA	2	1	| 0 0 1 0 | 0 1 0 0 | | immediate data |	A ← A + DATA

6.13.3 ADDC:

Function: Add with Carry

Description:

The ADDC instruction adds a byte value and the value of the carry flag to the accumulator. The results of the addition are stored back in the accumulator. Several of the flag registers are affected.

Table 51 – ADDC Description.

Instruction	Bytes	Cycles	Encoding	Operation
ADDC_A_RR	1	1	| 0 0 1 1 | 1 r r r |	A ← A + C + RR
ADDC_A_D	2	1	| 0 0 1 1 | 0 1 0 1 | | direct address |	A ← A + C + D
ADDC_A_ATRI	1	1	| 0 0 1 1 | 0 1 1 i |	A ← A + C + ATRI
ADDC_A_DATA	2	1	| 0 0 1 1 | 0 1 0 0 | | immediate data |	A ← A + C + DATA

6.13.4 AJMP:

Function: Absolute Jump

Description:

The AJMP instruction transfers program execution to the specified address. The address is formed by combining the 5 high-order bits of the address of the following instruction (for A15-A11), the 3 high-order bits of the opcode (for A10-A8), and the second byte of the instruction (for A7-A0). The destination address must be located in the same 2KByte block of program memory as the opcode following the AJMP instruction. No flags are affected.

Table 52 – AJMP Description.

Instruction	Bytes	Cycles	Encoding	Operation
AJMP	2	2	| a10 a9 a8 0 | 0 0 0 1 | | a7 a6 a5 a4 | a3 a2 a1 a0 |	(PC) ← (PC) + 2 ($PC_{10\text{-}0}$) ← page address

6.13.5 ANL:

Function: Logical-AND for byte/bit variables.

Description:

The ANL instruction performs a bitwise logical AND operation between the specified byte or bit operands and stores the result in the destination operand.

Note:

When this instruction is used to modify an output port, the value used as the port data will be read from the output data latch, not the input pins of the port.

Table 53 – ANL Description.

Instruction	Bytes	Cycles	Encoding	Operation
ANL_A_RR	1	1	| 0 1 0 1 | 1 r r r |	A ← A & RR
ANL_A_D	2	1	| 0 1 0 1 | 0 1 0 1 | | direct address |	A ← A & D
ANL_A_ATRI	1	1	| 0 1 0 1 | 0 1 1 i |	A ← A & ATRI
ANL_A_DATA	2	1	| 0 1 0 1 | 0 1 0 0 | | immediate data |	A ← A & DATA
ANL_D_A	2	1	| 0 1 0 1 | 0 0 1 0 | | direct address |	D ← D & A
ANL_D_DATA	3	2	| 0 1 0 1 | 0 0 1 1 | | direct address | | immediate data |	D ← D & DATA
ANL_C_BIT	2	2	| 1 0 0 0 | 0 0 1 0 | | Bit addr |	C ← C & BIT
ANL_C_NBIT	2	2	| 1 0 1 1 | 0 0 0 0 | | Bit addr |	C ← C & NBIT

6.13.6 CJNE:

Function: Compare and Jump if Not Equal.

Description:

The **CJNE** instruction compares the first two operands and branches to the specified destination if their values are not equal. If the values are the same, execution continues with the next instruction.

Table 54 – CJNE Description.

Instruction	Bytes	Cycles	Encoding	Operation
CJNE_A_D	3	2	| 1 0 1 1 | 0 1 0 1 | | direct address | | rel address |	PC ← PC + 3 IF A <> D THEN PC ← PC + relative offset IF A < D THEN C ← 1 ELSE C ← 0
CJNE_A_DATA	3	2	| 1 0 1 1 | 0 1 0 0 | | immediate address | | rel address |	PC ← PC + 3 IF A <> DATA THEN PC ← PC + relative offset IF A < DATA THEN C ← 1 ELSE C ← 0
CJNE_RR_DATA	3	2	| 1 0 1 1 | 1 r r r | |immediate data | | rel address |	PC ← PC + 3 IF RR <> DATA THEN PC ← PC + relative offset IF RR < DATA THEN C ← 1 ELSE C ← 0
CJNE_ATRI_DATA	3	2	| 1 0 1 1 | 0 1 1 i | | immediate address | |	PC ← PC + 3 IF ATRI <> DATA

			rel address |	THEN PC ← PC + relative offset IF ATRI < DATA THEN C ← 1 ELSE C ← 0

6.13.7 CLR:

Function: Clear Accumulator or Clear bit.

Description:

The CLR instruction sets the specified destination operand to a value of 0.

Table 55 – CLR Description.

Instruction	Bytes	Cycles	Encoding	Operation
CLR_A	1	1	| 1 1 1 0 | 0 1 0 0 |	A ← 0
CLR_C	1	1	| 1 1 0 0 | 0 0 1 1 |	C ← 0
CLR_BIT	2	1	| 1 1 0 0 | 0 0 1 0 | | Bit addr|	BIT ← 0

6.13.8 CPL:

Function: Complement Accumulator or Complement bit.

Description:

The CPL instruction logically complements the value of the specified destination operand and stores the result back in the destination operand. Bits that previously contained a 1 will be changed to a 0 and bits that previously contained a 0 will be changed to a 1.

Table 56 – CPL Description.

Instruction	Bytes	Cycles	Encoding	Operation
CPL_A	1	1	| 1 1 1 1 | 0 1 0 0 |	A ← /A
CPL_C	1	1	| 1 0 1 1 | 0 0 1 1 |	C ← /C
CPL_BIT	2	1	| 1 0 1 1 | 0 0 1 0 | | Bit addr |	BIT ← /BIT

6.13.9 DA:

Function: Decimal-adjust Accumulator for Addition

Description:

The DA instruction adjusts the 8-bit value in the accumulator to correspond to binary-coded decimal (BCD) format. This instruction begins by testing the low-order nibble of the accumulator. If the AC flag is set or if the low 4 bits of the accumulator exceed a value of 9, the accumulator is incremented by 6. The high-order nibble is then tested. If the carry flag is set or if the high 4 bits of the accumulator exceed a value of 9, the value 60h is added to the accumulator.

This instruction performs a decimal conversion by adding 00h, 06h, or 66h to the accumulator depending on the initial contents of the PSW and accumulator.

Table 57 – DA Description.

Instruction	Bytes	Cycles	Encoding	Operation
DA_A	1	1	| 1 1 0 1 | 0 1 0 0 |	contents of Accumulator are BCD IF [(A3-0 > 9) | (AC = 1)] THEN (A3-0) ← (A3-0) + 6 AND IF [(A7-4 > 9) | (C = 1)] THEN (A7-4) ← (A7-4) + 6

6.13.10 DEC:

Function: Decrement.

Description:

The DEC instruction decrements the specified operand by 1. An original value of 00h underflows to 0FFh. No flags are affected by this instruction.

Table 58 – DEC Description.

Instruction	Bytes	Cycles	Encoding	Operation
DEC_A	1	1	| 0 0 0 1 | 0 1 0 0 |	A ← A - 1
DEC_RR	1	1	| 0 0 0 1 | 1 r r r |	RR ← RR - 1
DEC_D	2	1	| 0 0 0 1 | 0 1 0 1 | | direct address |	D ← D - 1
DEC_ATRI	1	1	| 0 0 0 1 | 0 1 1 i |	ATRI ← ATRI - 1

6.13.11 DIV:

Function: Divide

Description:

The DIV instuction divides the unsigned 8-bit integer in the accumulator by the unsigned 8-bit integer in register B. After the division, the quotient is stored in the accumulator and the remainder is stored in the B register. The carry and OV flags are cleared.

If the B register begins with a value of 00h the division operation is undefined, the values of the accumulator and B register are undefined after the division, and the OV flag will be set indicating a division-by-zero error.

Table 59 – DIV Description.

Instruction	Bytes	Cycles	Encoding	Operation
DIV_AB	1	4	| 1 0 0 0 | 0 1 0 0 |	A[15:8] ← A / B B[7:0] ← A % B

6.13.12 DJNZ:

Function: Decrement and Jump if Not Zero.

Description:

The DJNZ instruction decrements the byte indicated by the first operand and, if the resulting value is not zero, branches to the address specified in the second operand.

Table 60 – DJNZ Description.

Instruction	Bytes	Cycles	Encoding	Operation
DJNZ_RR	2	2	| 1 1 0 1 | 1 r r r |	(PC) ← (PC) + 2

			| rel address |	RR ← RR - 1 IF [RR > 0 or RR < 0] THEN (PC) ← (PC) + rel
DJNZ_D	3	2	| 1 1 0 1 | 0 1 0 1 | | direct address | | rel address |	(PC) ← (PC) + 2 D ← D - 1 IF [D > 0 or D < 0] THEN (PC) ← (PC) + rel

6.13.13 INC:

Function: Increment.

Description:

The INC instruction increments the specified operand by 1. An original value of 0FFh or 0FFFFh overflows to 00h or 0000h. No flags are affected by this instruction.

Note: When this instruction is used to modify an output port, the value used as the port data is read from the output data latch, not the input pins of the port.

Table 61 – INC Description.

Instruction	Bytes	Cycles	Encoding	Operation
INC_A	1	1	| 0 0 0 0 | 0 1 0 0 |	A ← A + 1
INC_RR	1	1	| 0 0 0 0 | 1 r r r |	RR ← RR + 1
INC_D	2	1	| 0 0 0 0 | 0 1 0 1 | | direct address |	D ← D + 1
INC_ATRI	1	1	| 0 0 0 0 | 0 1 1 i |	ATRI ← ATRI + 1
INC_DPTR	1	2	| 1 0 1 0 | 0 0 1 1 |	DPTR ← DPTR + 1

6.13.14 JB:

Function: Jump if Bit set.

Description:

The JB instruction branches to the address specified in the second operand if the value of the bit specified in the first operand is 1. The bit that is tested is not modified. No flags are affected by this instruction.

Table 62 – JB Description.

Instruction	Bytes	Cycles	Encoding	Operation
JB	3	2	| 0 0 1 0 | 0 0 0 0 | | Bit addr | | rel address |	(PC) ← (PC) + 3 IF (bit) = 1 THEN (PC) ← (PC) + rel

6.13.15 JBC:

Function: Jump if Bit is set and Clear bit.

Description:

The JBC instruction branches to the address specified in the second operand if the value of the bit specified in the first operand is 1.Otherwise, execution continues with the next instruction.

If the bit specified in the first operand is set, it is cleared. No flags are affected by this instruction.

Note: When this instruction is used to modify an output port, the value used as the port data is read from the output data latch, not the input pins of the port.

Table 63 – JBC Description.

Instruction	Bytes	Cycles	Encoding	Operation
JBC	3	2	| 0 0 0 1 | 0 0 0 0 | | Bit addr | | rel address |	(PC) ← (PC) + 3 IF (bit) = 1 THEN (bit) ← 0 (PC) ← (PC) + rel

6.13.16 JC:

Function: Jump if Carry is set.

Description:

The JC instruction branches to the specified address if the carry flag is set. Otherwise, execution continues with the next instruction. No flags are affected by this instruction.

Table 64 – JC Description.

Instruction	Bytes	Cycles	Encoding	Operation
JC	2	2	| 0 1 0 0 | 0 0 0 0 | | rel address |	(PC) ← (PC) + 2 IF (C) = 1 THEN (PC) ← (PC) + rel

6.13.17 JMP:

Function: Jump indirect.

Description:

The JMP instruction transfers execution to the address generated by adding the 8-bit value in the accumulator to the 16-bit value in the DPTR register. Neither the accumulator nor the DPTR register are altered. No flags are affected by this instruction.

Table 65 – JMP Description.

Instruction	Bytes	Cycles	Encoding	Operation
JMP_A_DPTR	1	2	| 0 1 1 1 | 0 0 1 1 |	(PC) ← (A) + (DPTR)

6.13.18 JNB:

Function: Jump if Bit Not set.

Description:

The JNB instruction branches to the specified address if the specified bit operand has a value of 0. Otherwise, execution continues with the next instruction. No flags are affected by this instruction.

Table 66 – JNB Description.

Instruction	Bytes	Cycles	Encoding	Operation
JNB	3	2	| 0 0 1 1 | 0 0 0 0 |	(PC) ← (PC) + 3

			| Bit addr | | rel address |	IF (bit) = 0 THEN (PC) ← (PC) + rel

6.13.19 JNC:

Function: Jump if Carry not set.

Description:

The JNC instruction transfers program control to the specified address if the carry flag is 0. Otherwise, execution continues with the next instruction. No flags are affected by this instruction.

Table 67 – JNC Description.

Instruction	Bytes	Cycles	Encoding	Operation
JNC	2	2	| 0 1 0 1 | 0 0 0 0 | | rel address |	(PC) ← (PC) + 2 IF (C) = 0 THEN (PC) ← (PC) + rel

6.13.20 JNZ:

Function: Jump if Accumulator Not Zero.

Description:

The JNZ instruction transfers control to the specified address if the value in the accumulator is not 0. If the accumulator has a value of 0, the next instruction is executed. Neither the accumulator nor any flags are modified by this instruction.

Table 68 – JNZ Description.

Instruction	Bytes	Cycles	Encoding	Operation
JNZ	2	2	| 0 1 1 1 | 0 0 0 0 | | rel address |	(PC) ← (PC) + 2 IF (A) != 0 THEN (PC) ← (PC) + rel

6.13.21 JZ:

Function: Jump if Accumulator Zero.

Description:

The JZ instruction transfers control to the specified address if the value in the accumulator is 0. Otherwise, the next instruction is executed. Neither the accumulator nor any flags are modified by this instruction.

Table 69 – JZ Description.

Instruction	Bytes	Cycles	Encoding	Operation
JZ	2	2	| 0 1 1 0 | 0 0 0 0 | | rel address |	(PC) ← (PC) + 2 IF (A) = 0 THEN (PC) ← (PC) + rel

6.13.22 LCALL:

Function: Long call.

Description:

The LCALL instruction calls a subroutine located at the specified address. This instruction first adds 3 to the PC to generate the address of the next instruction. This result is pushed onto the stack low-byte first and the stack pointer is incremented by 2. The high-order and low-order bytes of the PC are loaded from the second and third bytes of the instruction respectively. Program execution is transferred to the subroutine at this address. No flags are affected by this instruction.

Table 70 – LCALL Description.

Instruction	Bytes	Cycles	Encoding	Operation
LCALL	3	2	| 0 0 0 1 | 0 0 1 0 | | addr15-addr8 | | addr7-addr0 |	(PC) ← (PC) + 3 (SP) ← (SP) + 1 ((SP)) ← (PC7-0) (SP) ← (SP) + 1 ((SP)) ← (PC15-8) (PC) ← addr15-0

6.13.23 LJMP:

Function: Long Jump.

Description:

The LJMP instruction transfers program execution to the specified 16-Bit addr. The PC is loaded with the high-order and low-order bytes of the address from the second and third bytes of this instruction respectively. No flags are affected by this instruction.

Table 71 – LJMP Description.

Instruction	Bytes	Cycles	Encoding	Operation
LJMP	3	2	| 0 0 0 0 | 0 0 1 0 | | addr15-addr8 | | addr7-addr0 |	(PC) ← addr15-0

6.13.24 MOV:

Function: Move byte variable or move bit data or load Data Pointer with a 16-bit constant.

Description:

The MOV instruction moves data bytes between the two specified operands. The byte specified by the second operand is copied to the location specified by the first operand. The source data byte is not affected.

Table 72 – MOV Description.

Instruction	Bytes	Cycles	Encoding	Operation
MOV_A_RR	1	1	| 1 1 1 0 | 1 r r r |	A ← RR
MOV_A_D	2	1	| 1 1 1 0 | 0 1 0 1 | | direct address |	A ← D
MOV_A_ATRI	1	1	| 1 1 1 0 | 0 1 1 i |	A ← ATRI
MOV_A_DATA	2	1	| 0 1 1 1 | 0 1 0 0 | | immediate data |	A ← DATA

MOV_RR_A	1	1	| 1 1 1 1 | 1 r r r |	RR ← A
MOV_RR_D	2	2	| 1 0 1 0 | 1 r r r | | direct address |	RR ← D
MOV_RR_DATA	2	1	| 0 1 1 1 | 1 r r r | | immediate data |	RR ← DATA
MOV_D_A	2	1	| 1 1 1 1 | 0 1 0 1 | | direct address |	D ← A
MOV_D_RR	2	2	| 1 0 0 0 | 1 r r r | | direct address |	D ← RR
MOV_D_D	3	2	| 1 0 0 0 | 0 1 0 1 | | dir addr src | | dir addr dest |	D ← D
MOV_D_ATRI	2	2	| 1 0 0 0 | 0 1 1 i | | direct address |	D ← ATRI
MOV_D_DATA	3	2	| 0 1 1 1 | 0 1 0 1 | | direct address | | immediate data |	D ← DATA
MOV_ATRI_A	1	1	| 1 1 1 1 | 0 1 1 i |	ATRI ←A
MOV_ATRI_D	2	2	| 1 0 1 0 | 0 1 1 i | | direct address |	ATRI ←D
MOV_ATRI_DATA	2	1	| 0 1 1 1 | 0 1 1 i | | immediate data |	ATRI ←DATA

6.13.25 MOVC:

Function: Move Code byte.

Description:

The MOVC instruction moves a byte from the code or program memory to the accumulator.

Table 73 – MOVC Description.

Instruction	Bytes	Cycles	Encoding	Operation
MOVC_A_ATDPTR	1	2	| 1 0 0 1 | 0 0 1 1 |	A ← A + DPTR
MOVC_A_ATPC	1	2	| 1 0 0 0 | 0 0 1 1 |	PC ← PC + 1 A ← A + PC

6.13.26 MOVX:

Function: Move External.

Description:

The MOVX instruction transfers data between the accumulator and external data memory. External memory may be addressed via 16-bits in the DPTR register or via 8-bits in the R0 or R1 registers. When using 8-Bit addring, Port 2 must contain the high-order byte of the address.

Table 74 – MOVX Description.

Instruction	Bytes	Cycles	Encoding	Operation
MOVX_A_ATRI	1	2	| 1 1 1 0 | 0 0 1 i |	A ← ATRI
MOVX_A_ATDPTR	1	2	| 1 1 1 0 | 0 0 0 0 |	A ← DPTR
MOVX_ATRI_A	1	2	| 1 1 1 1 | 0 0 1 i |	ATRI ← A
MOVX_ATDPTR_A	1	2	| 1 1 1 1 | 0 0 0 0 |	DPTR ← A

6.13.27 MUL:

Function: Multiply.

Description:

The MUL instruction multiplies the unsigned 8-bit integer in the accumulator and the unsigned 8-bit integer in the B register producing a 16-bit product. The low-order byte of the product is returned in the accumulator. The high-order byte of the product is returned in the B register. The OV flag is set if the product is greater than 255 (0FFh), otherwise it is cleared. The carry flag is always cleared.

Table 75 – MUL Description.

Instruction	Bytes	Cycles	Encoding	Operation
MUL_AB	1	4	| 1 0 1 0 | 0 1 0 0 |	(A)7-0 ← (A) * (B) (B)15-8

6.13.28 NOP:

Function: No Operation.

Description:

The NOP instruction does nothing. Execution continues with the next instruction. No registers or flags are affected by this instruction. NOP is typically used to generate a delay in execution or to reserve space in code memory.

Table 76 – NOP Description.

Instruction	Bytes	Cycles	Encoding	Operation
NOP	1	1	| 0 0 0 0 | 0 0 0 0 |	PC ← PC + 1

6.13.29 ORL:

Function: Logical-OR for byte/bit variables.

Description:

The ORL instruction performs a bitwise logical OR operation on the specified operands, the result of which is stored in the destination operand.

Note: When this instruction is used to modify an output port, the value used as the port data will be read from the output data latch, not the input pins of the port.

Table 77 – ORL Description.

Instruction	Bytes	Cycles	Encoding	Operation
ORL_A_RR	1	1	| 0 1 0 0 | 1 r r r |	A ← A | RR
ORL_A_D	2	1	| 0 1 0 0 | 0 1 0 1 | | direct address |	A ← A | D
ORL_A_ATRI	1	1	| 0 1 0 0 | 0 1 1 i |	A ← A | ATRI
ORL_A_DATA	2	1	| 0 1 0 0 | 0 1 0 0 | | immediate data |	A ← A | DATA
ORL_D_A	2	1	| 0 1 0 0 | 0 0 1 0 | | direct address |	D ← D | A
ORL_D_DATA	3	2	| 0 1 0 0 | 0 0 1 1 | | direct address | | immediate data |	D ← D | DATA
ORL_C_BIT	2	2	| 0 1 1 1 | 0 0 1 0 |	C ← C | BIT

			| Bit addr |	
ORL_C_NBIT	2	2	| 1 0 1 0 | 0 0 0 0 | | Bit addr |	C ← C | /BIT

6.13.30 POP:

Function: Pop from stack.

Description:

The POP instruction reads a byte from the address indirectly referenced by the SP register. The value read is stored at the specified address and the stack pointer is decremented. No flags are affected by this instruction.

Table 78 – POP Description.

Instruction	Bytes	Cycles	Encoding	Operation
POP	2	2	| 1 1 0 1 | 0 0 0 0 | | direct address |	(direct) ← ((SP)) (SP) ← (SP) - 1

6.13.31 PUSH:

Function: Push onto stack.

Description:

The PUSH instruction increments the stack pointer and stores the value of the specified byte operand at the internal RAM address indirectly referenced by the stack pointer. No flags are affected by this instruction.

Table 79 – PUSH Description.

Instruction	Bytes	Cycles	Encoding	Operation
PUSH	2	2	| 1 1 0 0 | 0 0 0 0 | | direct address |	(SP) ← (SP) + 1 ((SP)) ← (direct)

6.13.32 RET:

Function: Return from subroutine.

Description:

The RET instruction pops the high-order and low-order bytes of the PC from the stack (and decrements the stack pointer by 2). Program execution resumes from the resulting address which is typically the instruction following an ACALL or LCALL instruction. No flags are affected by this instruction.

Table 80 – RET Description.

Instruction	Bytes	Cycles	Encoding	Operation
RET	1	2	| 0 0 1 0 | 0 0 1 0 |	(PC15-8) ← ((SP)) (SP) ← (SP) - 1 (PC7-0) ← ((SP)) (SP) ← (SP) - 1

6.13.33 RETI:

Function: Return from interrupt.

Description:

The RETI instruction is used to end an interrupt service routine. This instruction pops the high-order and low-order bytes of the PC (and decrements the stack pointer by 2) and restores the interrput logic to accept additional interrupts. No other registers are affected by this instruction.

The RETI instruction does not restore the PSW to its value before the interrupt. The interrupt service routine must save and restore the PSW.

Execution returns to the instruction immediately after the point at which the interrupt was detected. If another interrupt was pending when the RETI instruction is executed, one instruction at the return address is executed before the pending interrupt is processed.

Table 81 – RETI Description.

Instruction	Bytes	Cycles	Encoding	Operation
RETI	1	2	| 0 0 1 1 | 0 0 1 0 |	(PC15-8) ← ((SP)) (SP) ← (SP) - 1 (PC7-0) ← ((SP)) (SP) ← (SP) - 1

6.13.34 RL:

Function: Rotate Accumulator Left.

Description:

The RL instruction rotates the eight bits in the accumulator left one bit position. Bit 7 of the accumulator is rotated into bit 0, bit 0 into bit 1, bit 1 into bit 2, and so on. No flags are affected by this instruction.

Table 82 – RL Description.

Instruction	Bytes	Cycles	Encoding	Operation
RL_A	1	1	| 0 0 1 0 | 0 0 1 1 |	(An + 1) ← (An) n = 0 - 6 (A0) ← (A7)

6.13.35 RLC:

Function: Rotate Accumulator Left through the Carry flag.

Description:

The RLC instruction rotates the eight bits in the accumulator and the one bit in the carry flag left one bit position. Bit 7 of the accumulator is rotated into the carry flag while the original value of the carry flag is rotated into bit 0 of the accumulator. Bit 0 of the accumulator is rotated into bit 1, bit 1 into bit 2, and so on. No other flags are affected by this operation.

Table 83 – RLC Description.

Instruction	Bytes	Cycles	Encoding	Operation
RLC_A	1	1	| 0 0 1 1 | 0 0 1 1 |	(An + 1) ← (An) n = 0 - 6 (A0) ← (C)

				(C) ← (A7)

6.13.36 RR:

Function: Rotate Accumulator Right.

Description:

The RR instruction rotates the eight bits in the accumulator right one bit position. Bit 0 of the accumulator is rotated into bit 7, bit 7 into bit 6, and so on. No flags are affected by this instruction.

Table 84 – RR Description.

Instruction	Bytes	Cycles	Encoding	Operation
RR_A	1	1	| 0 0 0 0 | 0 0 1 1 |	(An) ← (An + 1) n = 0 - 6 (A7) ← (A0)

6.13.37 RRC:

Function: Rotate Accumulator Right through Carry flag.

Description:

The RRC instruction rotates the eight bits in the accumulator and the one bit in the carry flag right one bit position. Bit 0 of the accumulator is rotated into the carry flag while the original value of the carry flag is rotated in to bit 7 of the accumulator. Bit 7 of the accumulator is rotated into bit 6, bit 6 into bit 5, and so on. No other flags are affected by this instruction.

Table 85 – RRC Description.

Instruction	Bytes	Cycles	Encoding	Operation
RRC_A	1	1	| 0 0 0 1 | 0 0 1 1 |	(An) ← (An + 1) n = 0 - 6 (A7) ← (C) (C) ← (A0)

6.13.38 SETB:

Function: Set Bit.

Description:

The SETB instruction sets the bit operand to a value of 1. This instruction can operate on the carry flag or any other directly addressable bit. No flags are affected by this instruction.

Table 86 – SETB Description.

Instruction	Bytes	Cycles	Encoding	Operation
SETB_C	1	1	| 1 1 0 1 | 0 0 1 1 |	C ← 1
SETB_BIT	2	1	| 1 1 0 1 | 0 0 1 0 | | Bit addr |	BIT ← 1

6.13.39 SJMP:

Function: Short Jump.

Description:

The SJMP instruction transfers execution to the specified address. The address is calculated by adding the signed relative offset in the second byte of the instructoin to the address of the following instruction. The range of destination addresses is from 128 before thenext instruction to 127 bytes after the next instruction.

Table 87 – SJMP Description.

Instruction	Bytes	Cycles	Encoding	Operation
SJMP	2	2	| 1 0 0 0 | 0 0 0 0 | | rel address |	(PC) ← (PC) + 2 (PC) ← (PC) + rel

6.13.40 SUBB:

Function: Subtract with borrow.

Description:

The SUBB instruction subtracts the specified byte variable and the carry flag from the accumulator. The result is stored in the accumulator. This instruction sets the carry flag if a borrow is required for bit 7 of the result. If no borrow is required, the carry flag is cleared.

Table 88 – SUBB Description.

Instruction	Bytes	Cycles	Encoding	Operation
SUBB_A_RR	1	1	| 1 0 0 1 | 1 r r r |	A ←A - C – RR
SUBB_A_D	2	1	| 1 0 0 1 | 0 1 0 1 | | direct address |	A ← A - C – D
SUBB_A_ATRI	1	1	| 1 0 0 1 | 0 1 1 i |	A ← A - C – ATRI
SUBB_A_DATA	2	1	| 1 0 0 1 | 0 1 0 0 | | immediate address |	A ← A - C - DATA

6.13.41 SWAP:

Function: Swap nibbles within the Accumulator.

Description:

The SWAP instruction exchanges the low-order and high-order nibbles within the accumulator. No flags are affected by this instruction.

Table 89 – SWAP Description.

Instruction	Bytes	Cycles	Encoding	Operation
SWAP_A	1	1	| 1 1 0 0 | 0 1 0 0 |	(A3-0) ↔ (A7-4)

6.13.42 XCH:

Function: Exchange Accumulator with byte variable.

Description:

The XCH instruction loads the accumulator with the byte value of the specified operand while simultaneously storing the previous contents of the accumulator in the specified operand.

Table 90 – XCH Description.

Instruction	Bytes	Cycles	Encoding	Operation

XCH_A_RR	1	1	| 1 1 0 0 | 1 r r r |	A ↔ RR
XCH_A_D	2	1	| 1 1 0 0 | 0 1 0 1 | | direct address |	A ↔ D
XCH_A_ATRI	1	1	| 1 1 0 0 | 0 1 1 i |	A ↔ ATRI

6.13.43 XCHD:

Function: Exchange Digit.

Description:

The XCHD instruction exchanges the low-order nibble of the accumulator with the low-order nibble of the specified internal RAM location. The internal RAM is accessed indirectly through R0 or R1. The high-order nibbles of each operand are not affected.

Table 91 – XCHD Description.

Instruction	Bytes	Cycles	Encoding	Operation
XCHD_A_ATRI	1	1	| 1 1 0 1 | 0 1 1 i |	A[3:0] ↔ RR[3:0]

6.13.44 XRL:

Function: Logical Exclusive-OR for byte/bit variables.

Description:

The XRL instruction performs a logical exclusive OR operation between the specified operands. The result is stored in the destination operand.

Note: When this instruction is used to modify an output port, the value used as the port data is read from the output data latch, not the pins of the port.

Table 92 – XRL Description.

Instruction	Bytes	Cycles	Encoding	Operation
XRL_A_RR	1	1	| 0 1 1 0 | 1 r r r |	A ← A ^ RR
XRL_A_D	2	1	| 0 1 1 0 | 0 1 0 1 | | direct address |	A ← A ^ D
XRL_A_ATRI	1	1	| 0 1 1 0 | 0 1 1 i |	A ← A ^ ATRI
XRL_A_DATA	2	1	| 0 1 1 0 | 0 1 0 0 | | immediate data |	A ← A ^ DATA
XRL_D_A	2	1	| 0 1 1 0 | 0 0 1 0 | | direct address |	D ← D ^ A
XRL_D_DATA	3	2	| 0 1 1 0 | 0 0 1 1 | | direct address | | immediate address |	D ← D ^ DATA

7 Memories Block Description

7.1 Introduction

Most of all IC projects have used memory modules. The difference among them is physical and structural, in order to attempt requirements efficiently. Today, there are many types of memories designed, because the range of applications is very large. Common types of memory used are RAM and ROM technologies. They are very popular on IC projects. RAM memory, in many times, was

designed to store all data information (values used by instructions and design units). ROM memory stores all program information (set of instructions) to perform its execution inside the core module.

The necessity of memory modules in many projects justifies the importance of data storage; however, in this project this is not different. The interactivity with many devices asking for places to store all the information and execute them at different times inside the microcontroller core 8051. Then, the design of memory modules makes sense.

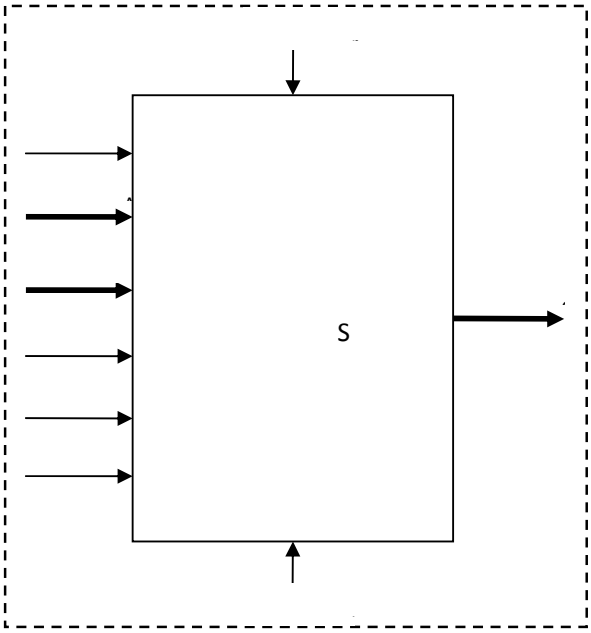


Figure 7 – SPRAM block diagram

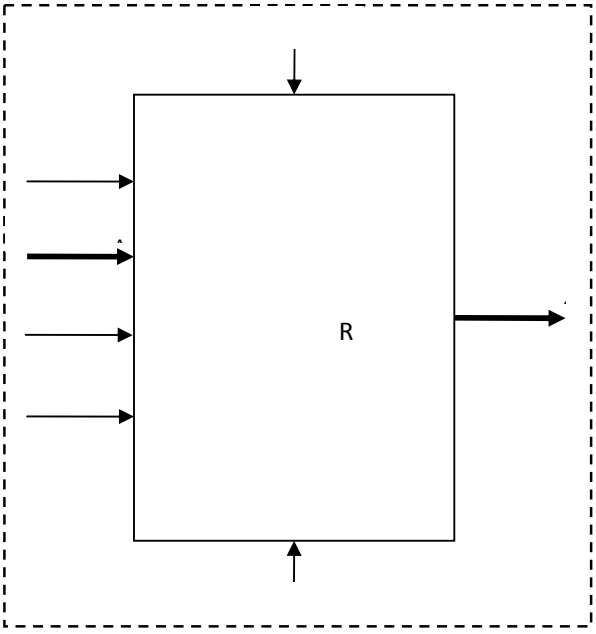


Figure 8 – ROM block diagram

7.2 Overview

The RAM memory (Figure 1), in this case (SPRAM) is responsible to store and provide data information. In other words, there are input signals incoming from external on-chip devices such as

central processor unit. The clock signal defines the speed through the frequency of working. When the clock is switching between 0 and 1 at determined frequency, an address is set if the enable signal to turn on all the functionalities of SPRAM. It can read or write in a SPRAM, is necessary address information to read/write, the idea can be as “read from somewhere/write in somewhere”. To read, change the output enable and to write, change the write enable according to the specification signals. When a data is written, the activated signals are address (to assign location in the memory), enable write action through write enable signal, enable to unlock module operation and data signals to set the data.

To read data, the only differences is not necessary data signals and write enable is turned off, instead read enable (turned on).

ROM memory (Figure 2) is another type of memory, commonly used to read permanent data once programmed (set up) never can be changed, only read. The idea is similar to read function in SPRAM. Data instructions are provided inside it and can be read through output bus. Only applying the address location of desirable data, it must be available on output bus. Of course, clock and enable signals must be activated according to specifications.

7.3 Features

XFAB Compiler Features

- 0.18µm CMOS
  - Processes xc018/xh018
- Separate input and output data buses
- No DC current path in memory schematics
- Typical access time 2.29ns (64kbit instance)
- Power consumption 0.131mW/MHz (64kbit instance)
- 134570 bits/mm2 (64kbit instance)
- Wide range of instance sizes available
  - RAM sizes from 64 to 512k bits
  - Word length from 2 to 128 bits in steps of 1
  - Address range from 32 to 8k words
- Wide operating range
  - **Absolute** Maximum Rating
    - VDD (DC Supply Voltage)
      - **-0.5 to 2.5** (V)
    - Tj (Junction Temperature Range)
      - **-55 to +150** (°C)
  - **Recommended** Operating Conditions
    - VDD (DC Supply Voltage)
      - **1.62 to 1.98** (V)
    - Tj (Junction Temperature Range)
      - **-40 to +125** (°C)

XFAB SPRAM IP

- Cell name = SPRAM128X8
- Cell function = single-port RAM
- Number of words = 128
- Number of data bits = 8
- Number of address bits = 7

- CMOS processes = xh018 LP3MOS 1.8/3.3 low power CMOS module
- Nominal power supply voltage = 1.8V
- Number of rows = 32
- Number of columns = 32
- Metal module = 4METALS

XFAB ROM IP

- Cell name = ROM4096X8
- Cell function = ROM
- Number of words = 4096
- Number of data bits = 8
- Number of address bits = 12
- CMOS processes = xh018 LP3MOS 1.8/3.3 low power CMOS module
- Nominal power supply voltage = 1.8V
- Number of rows = 256
- Number of columns = 128
- Metal module = 4METALS

7.4 Modes of operation

A brief description of mode operations in relation the pins:

7.4.1 SPRAM

- Read Mode
  - **Output Enable** pin on;
  - **Write Enable** pin off;
  - **Address bus** must be set up to locate data;
  - **Data input bus** can be specified, but will be ignored (doesn't write mode);
  - **Data output bus** searched at the address will be available at output port;
- Write mode
  - **Write Enable** pin on;
  - **Output Enable** pin off;
  - **Data address bus** must be specified to define where the data will be placed;
  - Set up **data value** at data bus to pass the value;

7.4.2 ROM

- Read Mode
  - **Output Enable** pin on;
  - **Write Enable** pin off;
  - **Data Address bus** must be set up to locate data;
  - **Data input bus** can be specified, but will be ignored (don't write);
  - **Data output** searched at the address will be available at output port;

NOTE: Obviously, to use any memory, the **enable** pin must be turned on.

7.5 External signal description

This section list and describe the signals that do, or may, connect off chip, and provide the necessary documentation for the customer.

The X-FAB memory compiler generates a memory block **with an internal power/ground supply structure**, but **without a fixed memory external power/ground supply structure**. The user himself is responsible for connecting all memory power/ground supply pins and for the power, ground supply routing from the chip supply pads to the memory supply pins.

The width of the memory external supply wires W_{VDD} and W_{GND} to the memory power and ground supply pins must be wide enough to prevent both electromigration problems and signal integrity problems due to a voltage drop on the memory power supply (see **MEMORY POWER GROUND SUPPLY WIDTH**).

To prevent a too big voltage drop on its power/ground wires the memory block should be placed as close as possible to the chip supply pads.

It is recommended to place additional capacitance between power and ground supply wires near the memory block. All wires used for power/ground connections should have equal resistance.

There are supply pins on the bottom side of the memory block only. All of them have to be connected to prevent violation messages because of unconnected pins.

7.6 Detailed signal descriptions

A table of signal properties, as shown, with a detailed discussion of signals can be viewed in the datasheet of XFAB® (all rights reserved). Also include appropriate timing diagrams and both SPRAM and ROM are available.

Table 56 – SPRAM 128x8 Interface description

Signal	I/O	Description		Reset
CLK	I	Clock signal input memory		N/A
		State Meaning	Asserted: n/a Negated: n/a	
		Timing	Assertion: 50% duty cycle Negation: 50% duty cycle	
/WEB	I	Read write control input. Write is active low.		N/A

Signal	I/O	Description		Reset
		State Meaning	Asserted: write inside memory disable Negated: write inside memory enable	
		Timing	Assertion: n/a Negation: Read enable signal must be high to avoid conflicts. Data must be ready to write in the data bus.	
/ENB	I	Enable active low		N/A
		State Meaning	Asserted: The memory is not enabled to use. Negated: The memory is enabled.	
		Timing	Assertion: may occur at any time synchronous to an external clock. Negation: may occur at any time synchronous to an external clock.	
ADR[6:0]	I	Address input bus		N/A
		State Meaning	Asserted: n/a Negated: n/a	
		Timing	Assertion: May occur at any time, according to specifications of the time diagram. Negation: May occur at any time, according to specifications of the time diagram.	
Q[7:0]	O	Data output bus		N/A
		State Meaning	Asserted: n/a Negated: n/a	
		Timing	Assertion: May occur at any time, according to specifications of the time diagram. Negation: May occur at any time, according to specifications of the time diagram.	
/OEB	I	Output enable active low		N/A
		State Meaning	Asserted: output bus disable to read Negated: output bus enabled to read	
		Timing	Assertion: n/a Negation: n/a	
ramvdd	I	Power supply		N/A
		State Meaning	Asserted: Activate the module Negated: Turn off the module	
		Timing	Assertion: Start startup time. Negation: n/a	
ramgnd	I	Ground supply		N/A
		State Meaning	Asserted: n/a Negated: n/a	
		Timing	Assertion: n/a Negation: n/a	
D[7:0]	I	Data signals input		N/A
		State Meaning	Asserted: n/a Negated: n/a	
		Timing	Assertion: May occur at any time, according to specifications of the time diagram. Negation: May occur at any time, according to specifications of the time diagram.	

Note: Data latches are transparent if memory is selected for write operation and previous memory operation is finished. Address latches are transparent if memory is selected for read/write operation and previous operation and previous memory operation is finished.

Keeping ENB high when memory is not active prevents unnecessary power consumption.

More information about SPRAM 128X8 at XFAB® Data Sheet

Table 57 – ROM 4Kx8

Signal	I/O	Description		Reset
CLK	I	Clock signal input memory		N/A
		State Meaning	Asserted: accept all signals activated by high level Negated: accept all signals activated by low level	
		Timing	Assertion: 50% (default frequency is 12Mhz) Negation: 50%	
/ENB	I	Enable active low		N/A
		State Meaning	Asserted: Keeping high prevents unnecessary power consumption. Negated: enable the memory module.	
		Timing	Assertion: n/a (timing information XFAB DATASHEET) Negation: n/a (timing information XFAB DATASHEET)	
ADR[6:0]	I	Address inputs		N/A
		State Meaning	Asserted: provides data location Negated: provides data location	
		Timing	Assertion: n/a (timing information XFAB DATASHEET) Negation: n/a (timing information XFAB DATASHEET)	
Q[7:0]	O	Data output bus		N/A
		State Meaning	Asserted: n/a Negated: n/a	
		Timing	Assertion: n/a (timing information XFAB DATASHEET) Negation: n/a (timing information XFAB DATASHEET)	
/OEB	I	Output enable active low		N/A
		State Meaning	Asserted: read closed Negated: turns Q[7:0] available to read	
		Timing	Assertion: n/a (timing information XFAB DATASHEET) Negation: n/a (timing information XFAB DATASHEET)	
ramvdd	I	Power supply		N/A
		State Meaning	Asserted: activate the module Negated: Turn off the module	
		Timing	Assertion: n/a Negation: n/a	
ramgnd	I	Ground supply		N/A
		State Meaning	Asserted: Provides power to module Negated: Turn off the module	
		Timing	Assertion: n/a Negation: n/a	

Note:

Reference voltage for delay measurement is 0.5 of power supply.

Reference voltages for transition time measurement/assignment are 1.0 of power supply and 0.9 of power supply.

More information about ROM 4Kx8 sees XFAB® Data Sheet.

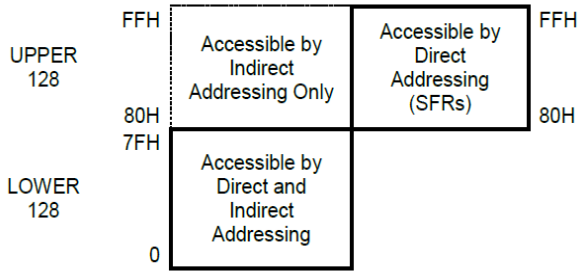
7.7 Memory map and register definition

The memory map for SPRAM and ROM interface registers consists of 8 bit registers with no special requirements. The memory map and registers details are in the following sections.

7.7.1 Memory map

The microcontroller has 128 bytes of on-chip RAM plus a number of Special Function Registers (SRFs). The memory space is show divided into three blocks, which are generally referred to as Lower 128, the Upper 128, and Special Function Registers space. The Special Function Registers are inside Core, Lower 128 is the Memory IP from X-FAB and Upper 128 bytes remaining are only in 8052 models. Internal Data Memory is mapped in Table 7.

Table 58 – Internal Data Memory



Where, SFRs e UPPER 128 bytes have the same address space (80H-FFH). To **DIRECT ACCESS**, the instruction:

MOV 80H, #AAH

This is an example of direct access, it will write AAH to address 80H.

These instructions:

MOV R0, #80H

MOV @R0, #BBH

Is an example for **INDIRECT ACCESS**, write the desirable address at R0 or R1 (only these), and BBH is written at location 80H of the data RAM.

7.7.2 Data Memory – SPRAM

The **LOWER 128** bytes can be accessed directly or indirectly and the lowest 32 bytes (00H to 1FH) are mapped as 4 banks of 8 registers. Program instructions call out these registers as R0 through R7. Two bits in the Program Status Word (PSW) select which register bank in use. This allows more efficient use of code space, since register instructions are shorter than instructions that use direct addressing.

RESET initializes the Stack Pointer to location 07H and it is incremented once to start from location 08H which is the first register (R0) of the second register bank. Thus, in order to use more than one register bank, the SP should be initialized to a different location of the RAM where it is not used for data storage (ie, higher part of the RAM).

Bit addrable Location: The next 16 bytes above the register banks form a block of bit-addressable memory space. The Bit addres in this area are 00H through 7FH. One way is to refer their addresses, i.e., 00H to 7FH. The other way is with reference to bytes 20H to 2FH. Thus, bits 0-7 can also be

referred to as bits 20.0-20.7, and bits 08H-0FH are the same as 21.0-21.7 and so on. Each of the 16 bytes in this segment can also be addressed as a byte.

General Purpose Area: Bytes 30H through 7FH are available to the user as data RAM. However, if the stack pointer has been initialized to this area enough number of bytes should be left aside to prevent SP data destruction. By default, the stack pointer (SP) is initialized at address 07H after a reset. This causes the stack to begin at location 08H. But it can start at the **General Purpose Area**. Stack Pointer is an 8-bit register used during a PUSH, POP, CALL, RET, or RETI instructions.

Table 59 - Internal Data Memory Organization

Byte Address		Bit Address							
Not Bit Addressable	7F 30	General Purpose RAM							
Bit Addressable Location	2F	7F	7E	7D	7C	7B	7A	79	78
	2E	77	76	75	74	77	72	71	70
	2D	6F	6E	6D	6C	6B	6A	69	68
	2C	67	66	65	64	63	62	61	60
	2B	5F	5E	5D	5C	5B	5A	59	58
	2A	57	56	55	54	53	52	51	50
	29	4F	4E	4D	4C	4B	4A	49	48
	28	47	46	45	44	43	42	41	40
	27	3F	3E	3D	3C	3B	3A	39	38
	26	37	36	35	34	33	32	31	30
	25	2F	2E	2D	2C	2B	2A	29	28
	24	27	26	25	24	23	22	21	20
	23	1F	1E	1D	1C	1B	1A	19	18
	22	17	16	15	14	13	12	11	10
	21	0F	0E	0D	0C	0B	0A	09	08
	20	07	06	05	04	03	02	01	00
Not Bit Addressable	1F 18	BANK 3							
	17 10	BANK 2							
	0F 08	BANK 1							
	07 00	BANK 0							

Internal Data Memory is always one byte (8 bits) wide, which implies an address space of only 256 bytes. However, the addressing modes for internal RAM can in fact accommodate 384 bytes, using a simple trick. Direct addresses higher than 7FH access one memory space, and indirect addresses higher than 7FH access one memory space, and indirect addresses higher than 7FH access a different memory space. So, the Upper 128 (in models with 256 bytes of RAM) and SFR space occupying the same block of addresses, 80H through FFH, although they are physically separate entities. The table 10 shows the SFR memory map organization. However, this project will use only direct access to access the **Data Memory**, because 8051 microcontroller has, in fact, 128 bytes of memory for Data Storage. Then, there are the 128 lower bytes and for upper bytes, it will use the SFRs only. Also, the UPPER memory will not be used; once the model 8052 has 256 bytes of physical RAM available to share the address range 80H-FFH.

Physically, the SFR is separated from Lower 128 bytes. It is on the Core module and its map is shown in Table below.

Table 60 – SFR Memory List

ACC	B
PSW	IP
IE	P0
P1	P2

P3	P4
P0EN	P1EN
P2EN	P3EN
SCON	TCON
TCON2	SBUF
TH0	TH1
TM0	TM1
TL0	TL1
TMOD	TX0
RX0	TX1
RX1	SMAP8
TACPH	TACPL
PCON	DPH
DPL	ACRH
ACRM	ACRL
SP	

7.7.3 Program Memory – ROM

Figure below shows a map of the lower part of the Program Memory. After reset, the CPU begins execution from location 0000H. As shown in this Figure, each interrupt is assigned a fixed location in Program Memory. The interrupt causes the CPU to jump to that location, where it commences execution of the service routine. External Interrupt 0, for example, is assigned to location 0003H. If External Interrupt 0 is going to be used, its service routine must begin at location 0003H. If the interrupt is not going to be used, its service location is available as general purpose Program Memory. This mechanism is used to perform interruption from external devices such as keypads, serial communications and so forth. More hardware resources results in power control.

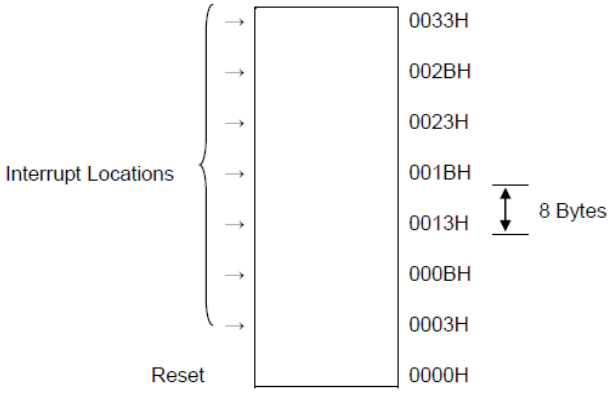


Figure 9 – Program Memory

The interrupt service locations are spaced at 8-byte intervals: 0003H for External Interrupt 0, 000BH for Timer 0, 0013H for External Interrupt 1, 001BH for Timer 1, etc. If an interrupt service routine is short enough (as is often the case in control applications), it can reside entirely within that 8-byte interval. Longer service routines can use a jump instruction to skip over subsequent interrupt locations, if other interrupts are in use.

The read strobe to external ROM, PSEN_b is used for all external program fetches. PSEN_b is not activated for internal program fetches, because it is used for external memory access only. But in this project won't be used.

7.8 Functional Description

As mentioned at the beginning of this document. The Memory ROM IP provided by XFAB® is composed by 4K bytes (4096 bytes; 4096 x 8) and is used as we call instruction/code memory. Then, it is not writable. It used to store data instruction of the program being executed fetching everyone at the initial address and incrementing the PC by the size of that instruction in execution. It is also used as code space for interruption. Each section of 8 bytes starting at address 03H is a location where the correspondent interruption jumps to execute the respective routine. If a routine is big enough to overlap its address space, then, the last instruction must be a jump to another section of code to execute code remaining. This information is helpful to understand addressing behavior of ROM when the same is working.

The memory used as RAM is the SPRAM. The IP Memory contains 128 bytes and will be used to store data information about registers in its major, for general purpose from programs. The stack pointer is implemented on this memory. As the manuals describe, the inputs of this block is designed to read and write data inside it. External signals cannot read and write at the same time. There are timing specifications at XFAB Data Sheet for more information about it (accessing time, slope time, setup and hold time, etc.). See more detailed description of time diagrams following.

Table 61 – ROM 4Kx8 Timing Specifications

Corner		Best				Typical				Worst				Unit
PVT		Fast 1.98V -40°C				Typical 1.8V 25°C				Fast 1.62V 125°C				
Specification	Symbol	0.02ns. input slope		0.04ns. input slope		0.02ns. input slope		0.04ns. input slope		0.02ns. input slope		0.04ns. input slope		
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
Output data access time	ACCESS_TIME		2.18		2.29		3.05		3.17		4.70		4.85	ns
Cycle time	CYCLE_TIME	2.23		2.24		3.06		3.07		4.65		4.66		ns
Cycle low pulse width	CLK_LOW_TIME	0.66		0.77		0.92		1.02		1.37		1.48		ns
Cycle high pulse width	CLK_HIGH_TIME	0.23		0.62		0.31		0.72		0.51		0.91		ns
Address setup time	ADR_SETUP_TIME	0.51		0.48		0.75		0.72		1.21		1.18		ns
Address hold time	ADR_HOLD_TIME	0.49		0.61		0.65		0.77		1.00		1.11		ns
Enable setup time	ENB_SETUP_TIME	0.68		0.84		0.96		1.10		1.50		1.59		ns
Enable hold time	ENB_HOLD_TIME	0.47		0.59		0.63		0.73		0.97		1.04		ns
Output disable to output tristate	HIGH_Z_TIME		0.17		0.23		0.23		0.31		0.34		0.43	ns
Output enable to data valid	LOW_Z_TIME		0.17		0.26		0.25		0.35		0.40		0.50	ns
Output data slope with Cload=0.4pf	Q_SLOPE_TIME		0.32		0.32		0.43		0.43		0.66		0.66	ns

Table 62 - ROM 4Kx8 Timing Specifications

Corner		Best		Typical		Worst		Unit
PVT		Fast 1.98V -40°C		Typical 1.8V 25°C		Fast 1.62V 125°C		
Specification	Symbol	Min	Max	Min	Max	Min	Max	
Clock transition time	CLK_RISE_TIME		2		3		3	ns
VDD turning on to memory readiness	STARTUP_TIME		250		250		250	ns

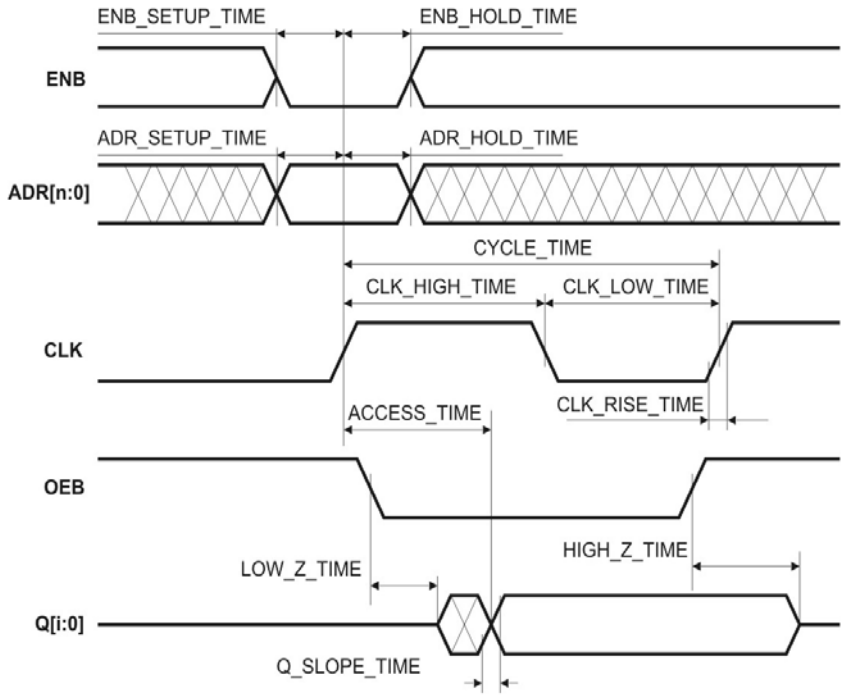


Figure 10 – ROM READ CYCLE time diagram

Table 63 - SPRAM 128x8 Timing Specifications

Corner		Best				Typical				Worst				Unit
PVT		Fast 1.98V -40°C				Typical 1.8V 25°C				Fast 1.62V 125°C				
Specification	Symbol	0.02ns. input slope		0.04ns. input slope		0.02ns. input slope		0.04ns. input slope		0.02ns. input slope		0.04ns. input slope		
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
Output data access time	ACCESS_TIME		1.21		1.30		1.77		1.91		2.85		2.99	ns
Cycle time	CYCLE_TIME	1.30		1.67		1.86		1.99		2.95		2.95		ns
Cycle low pulse width	CLK_LOW_TIME	0.40		0.68		0.57		0.75		0.90		1.00		ns
Cycle high pulse width	CLK_HIGH_TIME	0.17		0.17		0.25		0.60		0.41		0.41		ns
Address setup time	ADR_SETUP_TIME	0.42		0.40		0.61		0.57		0.96		0.92		ns
Address hold time	ADR_HOLD_TIME	0.07		0.15		0.11		0.18		0.18		0.23		ns
Input data setup time	D_SETUP_TIME	0.10		0.11		0.13		0.11		0.16		0.12		ns
Input data hold time	D_HOLD_TIME	0.28		0.33		0.41		0.46		0.66		0.70		ns
Write control setup time	WEB_SETUP_TIME	0.54		0.56		0.73		0.72		1.09		1.06		ns
Write control hold time	WEB_HOLD_TIME	0.14		0.20		0.20		0.25		0.32		0.36		ns
Enable setup time	ENB_SETUP_TIME	0.58		0.56		0.81		0.79		1.20		1.18		ns
Enable hold time	ENB_HOLD_TIME	0.12		0.23		0.18		0.28		0.31		0.38		ns
Output disable to output tristate	HIGH_Z_TIME		0.20		0.25		0.26		0.34		0.40		0.50	ns
Output enable to valid data	LOW_Z_TIME		0.19		0.28		0.28		0.38		0.45		0.56	ns
Output data slope with Cload = 0.4pf	Q_SLOPE_TIME		0.30		0.30		0.41		0.41		0.64		0.64	ns

Table 64 - SPRAM 128x8 Timing Specification

Corner		Best		Typical		Worst		
PVT		Fast 1.98V -40°C		Typical 1.8V 25°C		Fast 1.62V 125°C		
Specification	Symbol	Min	Max	Min	Max	Min	Max	
Clock transition time	CLK_RISE_TIME		2		3		3	

VDD turning on to memory readiness	START UP_TI ME		250. 0		250. 0		250. 0	

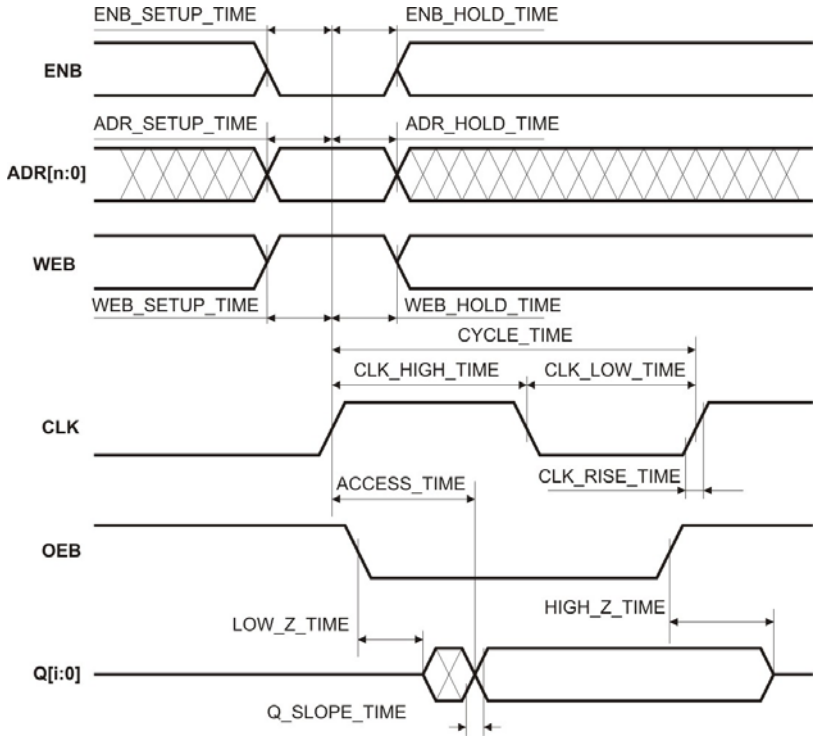


Figure 11 - SPRAM READ CYCLE time diagram

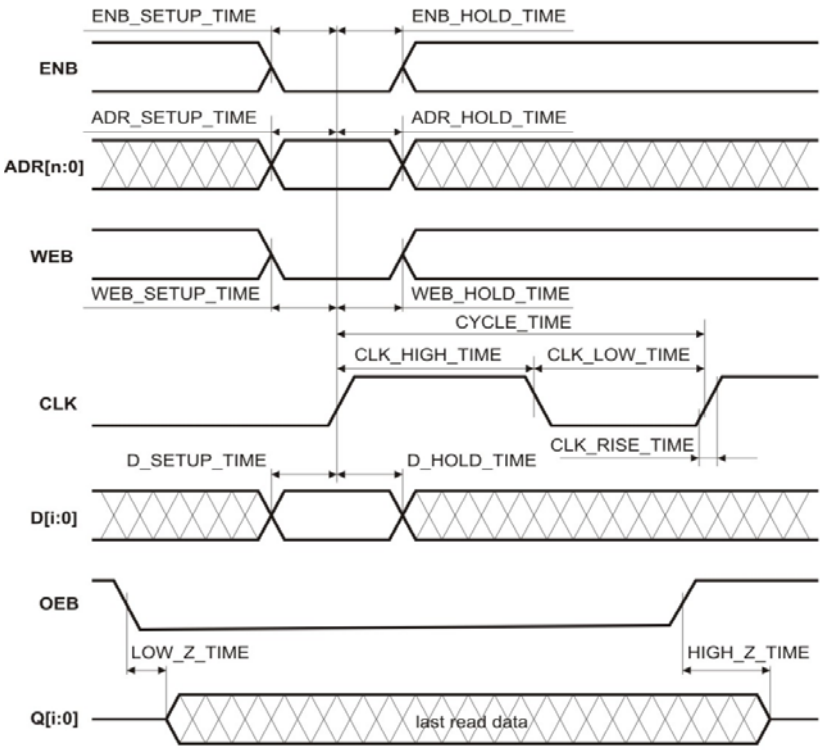


Figure 12 - SPRAM WRITE CYCLE time diagram

7.9 Extra Information

Not applicable.

7.10 Memory Power Ground Supply

The X-FAB memory compiler generates a memory block with an internal power/ground supply, but without a fixed memory external power/ground supply structure. The user himself is responsible for connecting all memory power/ground supply pins and for the power, ground supply routing from the chip supply pads to the memory supply pins. This allows the user to save chip area by adapting the memory power supply to his project layout. But the following memory Place & Route recommendations should be adhered and described in future versions of this document.

7.11 Initialization Information

This topic will cover only the reset state of SFRs. The lower 128 bytes (128 x 8) of SPRAM memory and ROM memory data sheets provided by XFAB®, does not contain any information about reset pins, signals and behaviors.

Following is described how states of SFRs are after a power up or a reset.

Table 65 - SFRs after power on or reset.

Table 2. Contents of the SFRs after reset

Register	Value in Binary
*ACC	00000000
*B	00000000
*PSW	00000000
SP	00000111
DPTR	
DPH	00000000
DPL	00000000
*P0	11111111
*P1	11111111
*P2	11111111
*P3	11111111
*IP	8051 XXX00000, 8052 XX000000
*IE	8051 0XX00000, 8052 0X000000
TMOD	00000000
*TCON	00000000
*+T2CON	00000000
TH0	00000000
TL0	00000000
TH1	00000000
TL1	00000000
+TH2	00000000
+TL2	00000000
*SCON	00000000
SBUF	Indeterminate
PCON	HMOS 0XXXXXXX CHMOS 0XXX0000

X = Undefined
* = Bit Addressable
+ = 8052 only

The internal reset algorithm writes 0s to all the SFRs except the port latches, the Stack Pointer, and SBUF. The port latches are initialized to FFH, the Stack Pointer to 07H and SBUF is indeterminate. This table is from Intel 8051/8052 manuals (see Bibliography section at the beginning of this document), then this project is customized and there is a feasibility of changes to be done. In other words, more special function registers have been added to this table. As far as possible, the changes will be defined and the table updated.

8 Bus Control Block Description

8.1 Introduction

This module is responsible to management of external signal of EMC08, they are: EA_b and PSEN_b.

Accesses to external Memory use signal PSEN_b (program store enable) as the read strobe.

If the EA_b pin is strapped to Vss, then all program fetches are directed to external ROM.

Accesses to external memory are of two types:

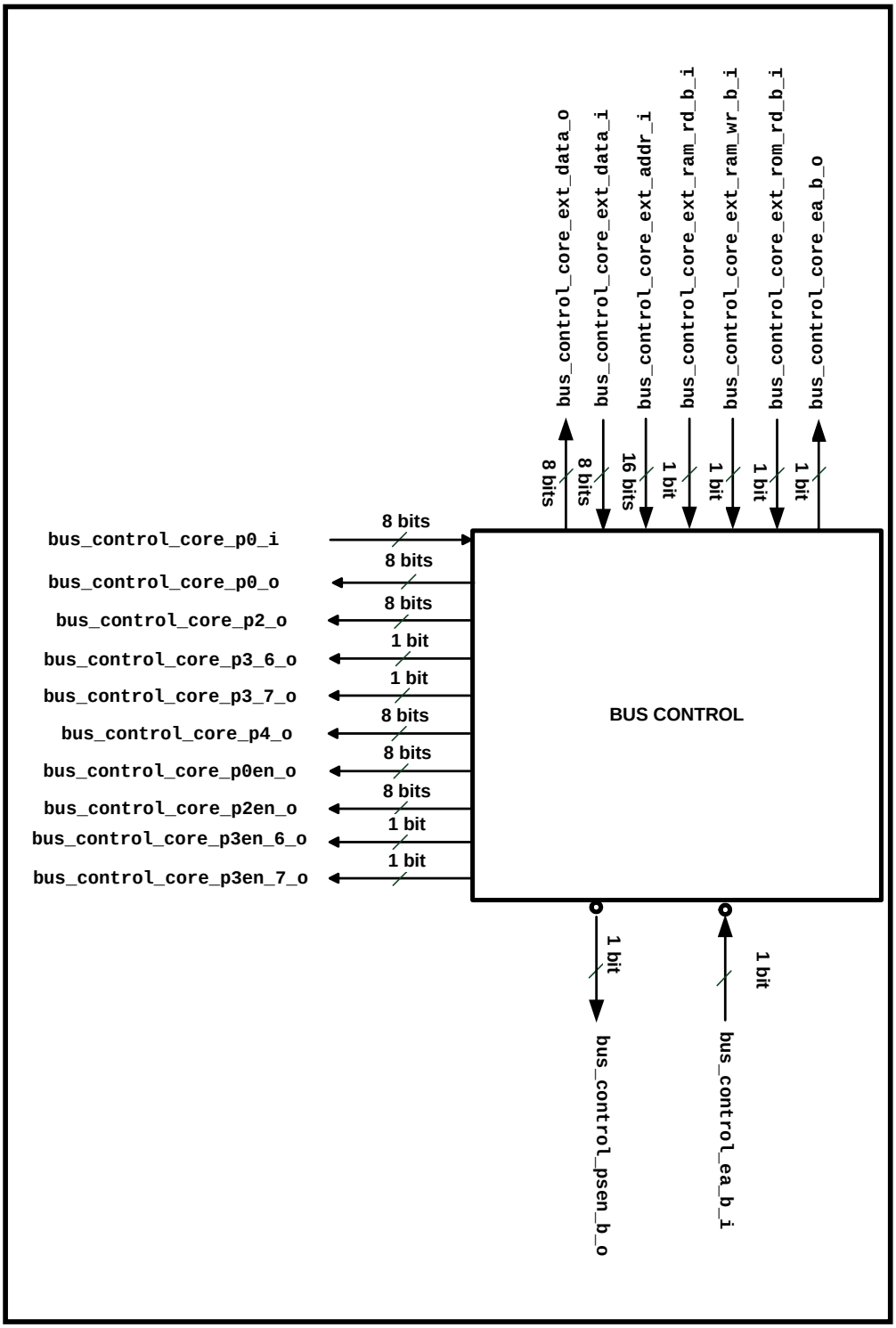
- accesses to external Program Memory ;
- accesses to external Data Memory.

External Data Memory addresses can be either 1 or 2 bytes wide.

- One-byte addresses are often used in conjunction with one or more other I/O lines to page the RAM;

Two-byte address can also be used, in which case the high address byte is emitted at Port 2 and the low byte of the address is driving by Port 4.

Figure 13 – Bus Control block diagram



8.2 Overview

This module is responsible to management of external signals of EMC08, they are: EA_b and PSEN_b.

The read strobe to external ROM, PSEN_b is used for all external program fetches. PSEN_b is not activated for internal program fetches.

The lowest 4K bytes of Program Memory can be either in the on-chip ROM or in an external ROM. This selection is made by strapping the EA_b pin to either VCC or Vss. This project uses only 4K bytes of ROM internal

When EA_b is connected at the VCC performs the internal ROM program memory, otherwise you run the program from external ROM, but within the rules of addresses for memories that can be 4K. If the EA_b pin is strapped to Vcc, then program fetches to addresses 0000H through OFFFH are directed to the internal ROM. Program fetches to addresses 1000H through FFFFH are directed to external ROM.

If the EA_b pin is strapped to Vss, then all program fetches are directed to external ROM. Fetches from external Program Memory always use a 16-Bit addr. Accesses to external Data Memory can use either a 16-Bit addr or an 8-Bit addr.

Whenever a 16-Bit addr is used, the high byte of the address comes out on Port 2, where it is held for the duration of the read or writes cycle.

If an 8-Bit addr is being used (MOVX @Ri, where i can be 1 or 0), the contents of the Port 2 SFR remain at the Port 2 pins throughout the external memory cycle. This will facilitate paging and the low byte of the address is driving by Port 4.

Accesses to external Data Memory use RD_b or WR_b (alternate functions of P3.7and P3.6) to strobe the memory.

8.3 Features

The Bus Control features are:

- ✓ Two 8-bit bidirectional parallel ports;
- ✓ Two 1-bit bidirectional parallel ports;
- ✓ One 8-bits unidirectional parallel ports;
- ✓ P3.6 – WR_b;
- ✓ P3.7 – RD_b;
- ✓ One 8-Bit addr port;
- ✓ 8-bit Data Bus;
- ✓ 16-Bit addr Bus;
- ✓ Access to external program memory (ROM);
  - Provide to include ROM external with the size: 4KB;
  - Signals: PSEN_b, EA_b ;
- ✓ Access to external data memory (RAM) ;
  - Provide to include RAM external with size the 64K Bytes;

- Signals: RD_b, WR_b;

✓ Data Bus (8-bit)

- Port 0 can be switchable to a bidirectional data bus

✓ Address Bus (16-bit)

- Port 2 can be switchable to output upper address byte the external memory (higher address bus)
- Port 4 is output lower address Byte to the external memory (lower address bus)

8.4 Modes of operation

The bus control supply access to external data memory and external program memory. There are two types of memory accesses: accesses external to Program Memory (ROM) and accesses to external Data Memory (RAM). Accesses to external RAM use WR_b or RD_b (alternate functions of P3.7 and P3.6) to strobe the memory.

Fetches from external ROM always use a 16-Bit addr. Accesses to external RAM can use either Bit addr or an 8-Bit addr. The lower 4K bytes of ROM can be either in the on-chip ROM or in an external ROM. This selection is made by strapping the EA_b pin to either Vcc or Vss.

The ROM less parts must have this pin externally strapped to VSS to enable them to execute properly. The read strobe to external ROM, PSEN_b is used for all external program fetches. PSEN_b is not activated for internal program fetches.

8.5 External signal description

The Table below show external signal description of the Bus Control:

Table 66 – Signal Description

Pin name	Description
bus_control_core_p0_i [7:0]	Data[7:0] external memory
bus_control_core_p0_o [7:0]	Data[7:0] external memory
bus_control_core_p2_o [7:0]	Higher Address [15:8] external memory
bus_control_core_p3_6_o [1:0]	P3[6]- WR_b (external data memory write strobe)
bus_control_core_p3_7_o [1:0]	P3[7]- RD_b (external data memory read strobe)
bus_control_core_p4_o [7:0]	Lower address bus output
bus_control_core_p0en_o [7:0]	Enable Port 0
bus_control_core_p2en_o [7:0]	Enable Port 2
bus_control_core_p3en_6_o [1:0]	Enable Pin 3.6
bus_control_core_p3en_7_o [1:0]	Enable Pin 3.7
bus_control_core_ext_data_o [7:0]	External memory data bus
bus_control_ea_b_i [1:0]	External Memory Access Enable
bus_control_psen_b_o [1:0]	Program Store Enable
bus_control_core_ext_ram_rd_b_i [1:0]	External RAM read signal
bus_control_core_ext_ram_wr_b_i [1:0]	External RAM write signal
bus_control_core_ext_rom_rd_b_i [1:0]	External ROM read signal
bus_control_core_ext_addr_i [15:0]	External memory address bus
bus_control_core_ext_data_i [7:0]	External memory data bus
bus_control_core_ea_b_o [1:0]	External Memory Access Enable

8.6 Detailed signal descriptions

The Table below shows the signals interface description of the bus control

Table 67 – Interface description

Signal	I/O	Description	
bus_control_ea_b_i [1:0]	I	The signal EA_b defines if fetches will be in internal ROM and external ROM doing strapped to Vcc or Vss	
		State Meaning	Asserted: must be connected to Vcc (1) for internal program execution (ROM) Negated: must be connected to Vss (0) for external program execution (ROM)
		Combinational	
bus_control_psen_b_o [1:0]	O	The signal PSEN_b is used for read strobe for all external program fetches	
		State Meaning	Asserted: To external program fetches of ROM
		Combinational	
bus_control_core_p3_7_o [1:0]	O	This signal permit read in external RAM uses pin P3.7 of P3 port	
		State Meaning	Asserted: if RD is low Negated: if RD is high
		Combinational	
bus_control_core_p3_6_o [1:0]	O	This signal allows the write in the external RAM and uses pin P3.6 of P3 port	
		State Meaning	Asserted: if WR is low Negated: if WR is high
		Combinational	
bus_control_core_ext_ram_rd_b_i [1:0]	I	External RAM Read Signal	
		State Meaning	Asserted: External RAM Read Disabled Negated: External RAM Read Enabled
		Combinational	
bus_control_core_ext_ram_wr_b_i [1:0]	I	External RAM Write Signal	
		State Meaning	Asserted: External RAM Write Disabled Negated: External RAM Write Enabled
		Combinational	
bus_control_core_ext_rom_rd_b_i [1:0]	I	External ROM Read Signal	
		State Meaning	Asserted: External ROM Read Disabled Negated: External ROM Read Enabled
		Combinational	
bus_control_core_ext_addr_i [15:0]	I	External memory address bus	
		State Meaning	Asserted: n/a Negated: n/a
		Combinational	
bus_control_core_ext_data_i [7:0]	I	External memory data bus (input)	
		State Meaning	Asserted: n/a Negated: n/a
		Combinational	
bus_control_core_ext_data_o [7:0]	O	External memory data bus (output)	
		State Meaning	Asserted: n/a

			Negated: n/a
		Combinational	
bus_control_core_ea_b_o [1:0]	O	The signal EA_b defines if fetches will be in internal ROM and external ROM doing strapped to Vcc or Vss	
		State Meaning	Asserted: must be connected to Vcc (1) for internal program execution (ROM) Negated: must be connected to Vss (0) for external program execution (ROM)
		Combinational	
bus_control_core_p0_i [7:0]	I	Signal Bus Control pass data to Core	
		State Meaning	Asserted: n/a Negated: n/a
		Combinational	
bus_control_core_p0_o [7:0]	O	Signal Bus Control which pass data to Memory	
		State Meaning	Asserted: n/a Negated: n/a
		Combinational	
bus_control_core_p2_o [7:0]	O	This signal send the Higher Address [15:8] external memory	
		State Meaning	Asserted: n/a Negated: n/a
		Combinational	
bus_control_core_p4_o [7:0]	O	This signal send the Lower Address [7:0] external memory	
		State Meaning	Asserted: n/a Negated: n/a
		Combinational	
bus_control_core_p0en_o [7:0]	O	This signal enable the Port 0	
		State Meaning	Asserted: n/a Negated: n/a
		Combinational	
bus_control_core_p3en_6_o [1:0]	O	This signal enable the Pin 3.6 of the Port 3	
		State Meaning	Asserted: n/a Negated: n/a
		Combinational	
bus_control_core_p3en_7_o [1:0]	O	This signal enable the Pin 3.7 of the Port 3	
		State Meaning	Asserted: n/a Negated: n/a
		Combinational	
bus_control_core_p2en_o [7:0]	O	This signal enable Port 2	
		State Meaning	Asserted: n/a Negated: n/a
		Combinational	

8.7 Data External Memory - RAMX

The following figure shows a hardware configuration for accessing up to 2K bytes of external RAM.

The CPU in this case is executing from internal ROM. Port 4 serves an address bus to the RAM, and 3 lines of Port 2 are being used to page the RAM. The CPU generates RD_b and WR_b signals as needed during external RAM accesses.

There can be up to 64K bytes of external Data Memory. External Data Memory addresses can be either 1 or 2 bytes wide. One-byte addresses are often used in conjunction with one or more other

I/O lines to page the RAM, as shown in Figure 2. Two-byte address can also be used, in which case the high address byte is emitted at Port 2. Otherwise the Port 2 pins continue to emit the P2 SFR content.

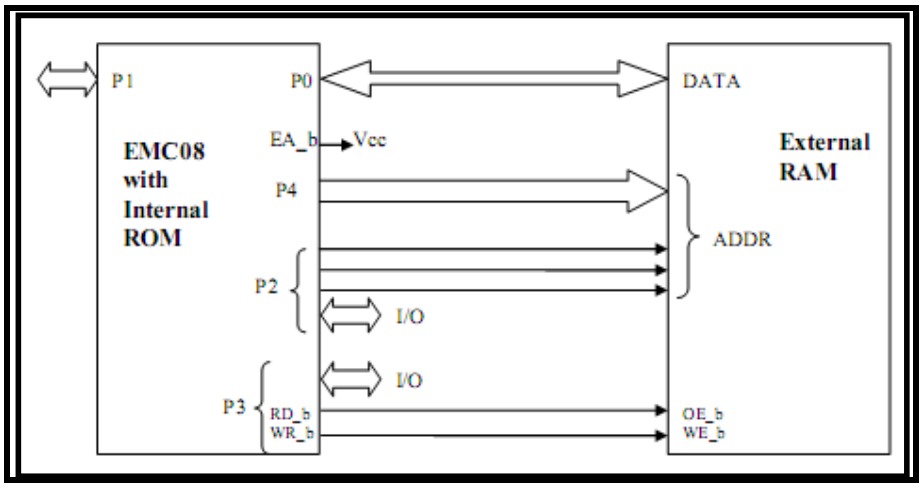


Figure 14 – Accessing External Data Memory

8.8 Accessing External Memory

Accesses to external memory are of two types: accesses to external Program Memory and accesses to external Data Memory. Accesses to external program Memory use signal PSEN_b (program store enable) as the read strobe.

Accesses to external Data Memory use RD_b or WR_b (alternate functions of P3.7and P3.6) to strobe the memory.

Fetches from external Program Memory always use a 16-Bit addr. Accesses to external Data Memory can use either a 16 Bit addr (MOVX @DPTR) or an 8-Bit addr (MOVX @Ri, where I can be 1 or 0).

Whenever a 16-Bit addr is used, the high byte of the address comes out on Port 2, where it is held for the duration of the read or write cycle.

If an 8-Bit addr is being used (MOVX @Ri, where I can be 1 or 0), the contents of the Port 2 SFR remain at the Port 2 pins throughout the external memory cycle. This will facilitate paging and the low byte of the address is driving by Port 4.

8.9 EMC08 Memory Addressing

8.9.1 Direct Addressing

In direct addressing the operand is specified by an 8-Bit addr field in the instruction. Only internal lower 128 Byte Data RAM and SFRs can be directly addressed.

8.9.2 Indirect Addressing

In indirect addressing the instruction specifies a register which contains the address of the operand. Both internal and external RAM can be indirectly addressed. The address register for 8-Bit

addres can be R0 or R1 of the selected register bank, or the Stack Pointer. The address register for 16-Bit addres can only be the 16-bit —data pointer register, DPTR.

8.10 Register Description

Sample register description: The REG sub-module of core stores the majority of Special Function Registers. In this module, the SFRs can be update or send their values to the FSM or ALU.

8.11 Functional Description

This module is responsible to management of external signals of EMC08, they are: EA_b and PSEN_b. The read strobe to external ROM, PSEN_b is used for all external program fetches. PSEN_b is not activated for internal program fetches.

The lowest 4K bytes of Program Memory can be either in the on-chip ROM or in an external ROM. This selection is made by strapping the EA_b pin to either VCC or Vss.

When EA_b is connected at the VCC performs the internal ROM program memory, otherwise you run the program from external ROM, but within the rules of addresses for memories that can be 4K. If the EA_b pin is strapped to Vcc, then program fetches to addresses 0000H through OFFFH are directed to the internal ROM. Program fetches to addresses 1000H through FFFFH are directed to external ROM.

If the EA_b pin is strapped to Vss, then all program fetches are directed to external ROM.

Fetches from external Program Memory always use a 16 Bit addr. Accesses to external Data Memory can use either a 16 Bit addr or an 8-Bit addr.

Whenever a 16-Bit addr is used, the high byte of the address comes out on Port 2, where it is held for the duration of the read or write cycle.

If an 8-Bit addr is being used (MOVX @Ri, where i can be 1 or 0), the contents of the Port 2 SFR remain at the Port 2 pins throughout the external memory cycle. This will facilitate paging and the low byte of the address is driving by Port 4.

Accesses to external Data Memory use RD_b or WR_b (alternate functions of P3.7and P3.6) to strobe the memory.

8.12 Extra Information

Not applicable

8.13 Initialization Information

The bus control initialization when the core requests access to the external ROM and to the external RAM. In the other cases, the bus control stays in an idle mode.

8.14 Application Information

Not applicable

9 Timers Block Description

9.1 Introduction

The microcontroller EMC08 has 3 timers / counters, calls of TIMER (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).

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.

Figure below shows the block diagram (top level) of the timers, they are prepared in all inputs and outputs, the arrows indicate whether the signal is unidirectional or bi-directional, many bits of each. This is a representation known as black box.

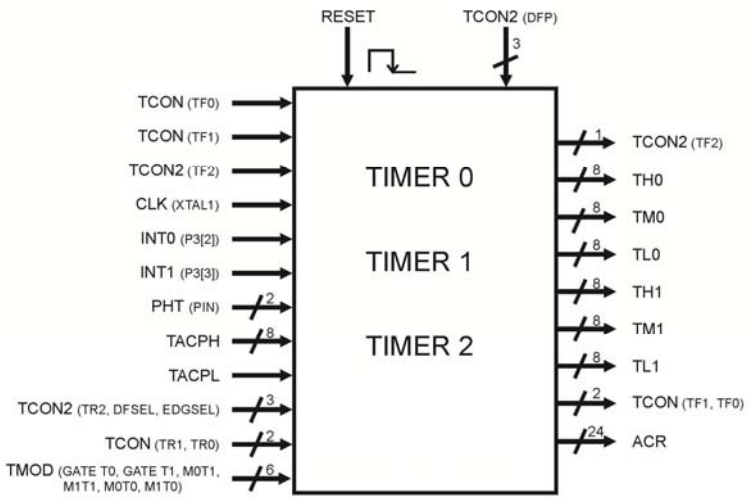


Figure 15 – Top Level of Timers

9.2 Overview

Starting from the growth of the use electronics embedded on the surface mobile vehicle did necessary the creation and development of several devices for improvement of the acting and safety of those platforms, the necessity of a new device was evident. This device was called Flywheel.

He acts in the improvement of the motor acting optimizing his consumption and efficiency providing stability in his operation.

Through analogic sensor the obtained signs are converted in digital signs after have been processed, are sent for an Unit of Control of the motor (Engine Control Unit - ECU) that it makes the necessary corrections in the injection control of the fuel mixture and ignition speed.

It is a auto-adaptive system to monitor and recognizes the changes that happen in the motor and it compensates them automatically acting in the Map Base of Fuel, progress and air flow in ECU.

This document provides all the functions and settings of timers, registers and their associated memory locations.

Further, details will be discussed about the operation of timer 2, (which is a specific application automotive), all your settings and associated records.

9.3 Functional Description

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).

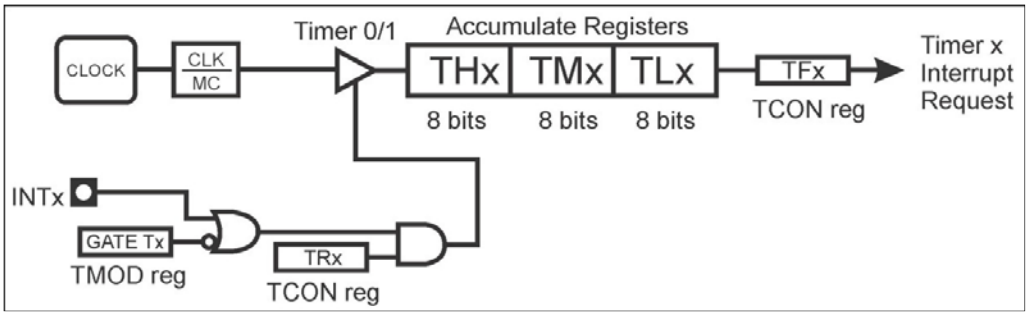


Figure 16 - TIMER 0 or 1 Functional Circuit

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 operation of Timer 2 is specific for the application what it destine being composed of a mixer of analogical sign with digital sign, to follow it will be made a description in his operation way.

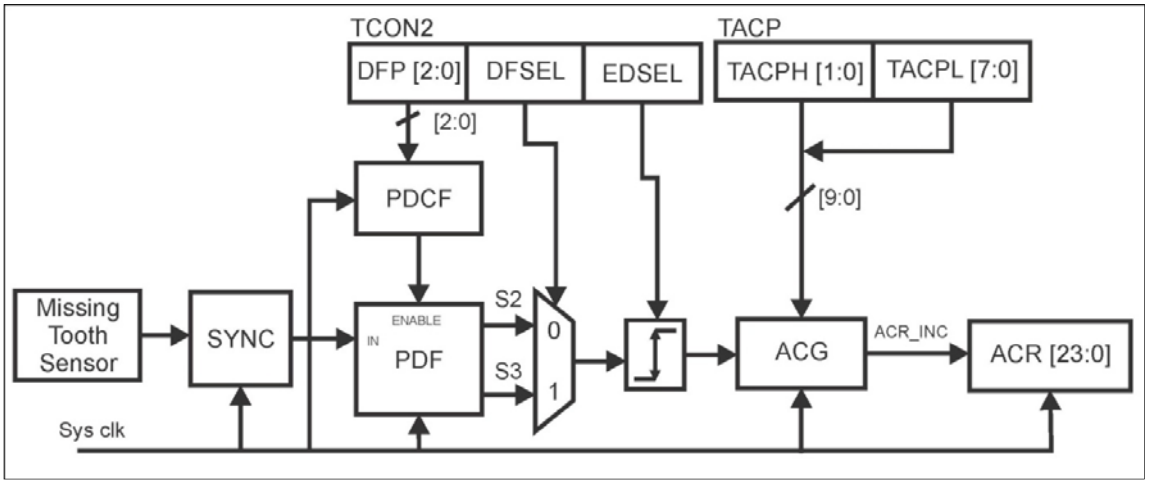


Figure 17 - Block Diagram Timer 2

The process of data collection done by the Timer 2 (Flywheel) is executed by sampling originating from of a sensor one analogical (inductive) in contact with the existent jagged wheel in the extremity of the tree of cranks (crankshaft) that sends pulses for a converter A/D that is processed by the Timer 2 and the pulse sequence stored in a 24 bits register (ACR) enabled and incapacitated by the sign acr_inc.

The received signal from analog sensor is submit for a synchronization through SYNC control block that has as purpose maintain the metastability of the signal sent for the filter PDF that is controlled by signal of the programmable filter PDCF, this being controlled by signal that comes DFP that to configure a frequency spliter of 3 bits with the purpose adapting the frequency of a coming signal of the analogic sensor, the frequency of the coming signal of oscillator.

The coming signal from PDCF filter, controls (enable) the PDF filter that sends the same to the next stage of the DFSEL selector, this stage is a sample selector the analog signal that is to still eliminate some instability existent happened in the system, that he does in two operation mode that defined for the application through the bit no. 1 from TCON2 register.

If the logical level of that bit is low (0) the selector DFSEL will select two samples same successive arrival of the sign of the filter PDF, otherwise, they will be rejected.

If the logical level of that bit goes 1 the selector DFSEL will select three samples same successive arrival of the sign of the filter PDF, otherwise, they will be rejected.

The sign will be given to the next stage, the border selector EDSEL that will choose the border of work of the next stage of ACG.

The stage of ACG will make a comparison with the coming signal from TACP register (10-bits) with the signal of the occurrence of the coming tooth of the sensor analogical.

The process of that comparison takes place with the entrance a die measured initial in a register TACP, after this when occurs a tooth the generator ACG will begin decrease the received data from TACP register in the end of counting coincides an occurrence from next tooth, the ACG generator update the ACR register (24-bits), when this is enabled by the signal acr_inc.

After updating the register ACR the generator ACG restarts the counting process for the next period of occurrence tooth, if the same happens before the finalization counting the obtained data is stored in the ACG generator and it will be decrement successively until the counting to conclude if

on this exact moment there is an occurrence of the tooth the data is stored in ACG and starts again the countdown.

In case this counting arrives at the end and there was not the occurrence of tooth that data is stored in the ACG generator and increased each period of the angle clock to that there is the coincidence of finalization the counting with a tooth occurrence, like this the process will repeat in each period of angle clock. Sometimes confirming the counting with the tooth occurrence, other times being necessary to increase or to decrease that counting with the purpose of doing the finalization counting to coincide with the tooth occurrence.

The following illustration displays the functional diagram of Angle Clock Generator (ACG).

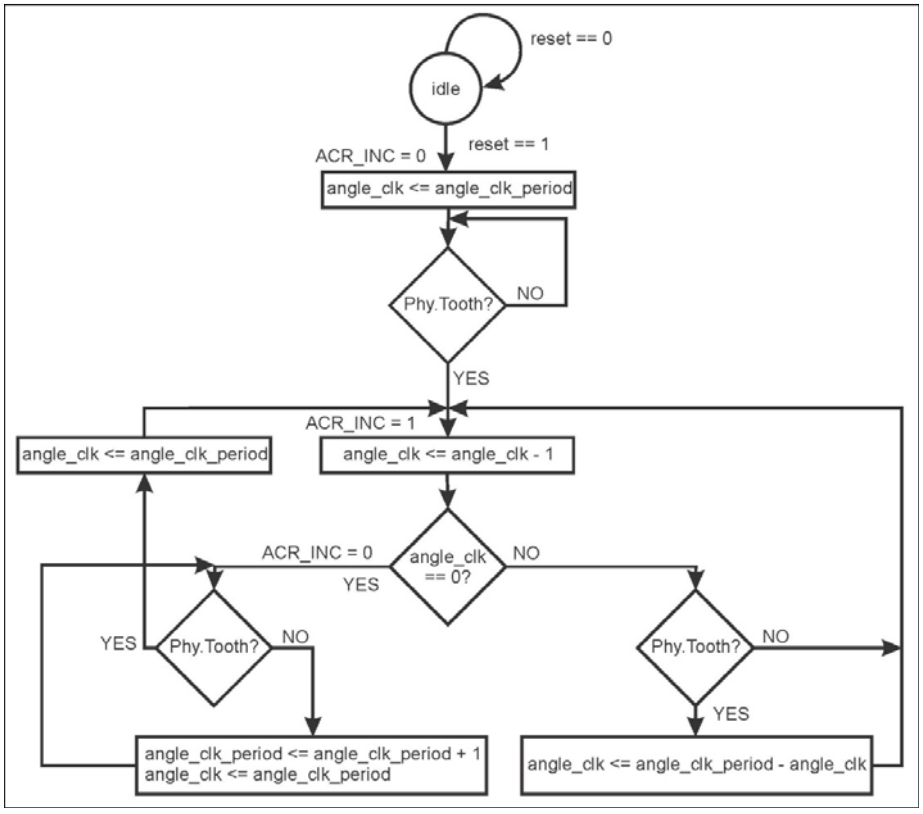


Figure 18 - Microcode Algorithm

9.4 Initialization Information

Before starting the program, case you will use one of the timers, it is advisable initialize the registers **TLx**, **TMx**, **THx**, with necessary values or zero, to ensure that no one has any value due to a chain that has not been completely discharged, also known as trash.

The folowing table shows the values of registers after reset.

Address / Offset	Register	Reset Value
D8h	TCON2	x000 0001b
88h	TCON	0000 0000b
89h	TMOD	0x00 0x00b
8Ch	TH0	0000 0000b
8Eh	TM0	0000 0000b
8Ah	TL0	0000 0000b

8Dh	TH1	0000 0000b
8Fh	TM1	0000 0000b
8Bh	TL1	0000 0000b
BBh	TACPH	xxxx xx00b
BAh	TACPL	0000 0000b

9.5 Features

9.5.1 Timers 0 and 1

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

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

9.5.2 Timer 2

The main characteristic of the timer 2 (Flywheel) is provide to ECU the possibility of:

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

9.6 Modes of operation

9.6.1 Mode 0

Either Timer 0 and Timer 1 in Mode 0 are a 24-bit Counter. As the count rolls over from all 1s to all 0s, it sets the Timer interrupt flag **TF1**. The counted input is enabled to the Timer when **TR1** = 1 and either GATE = 0 or INT1 = 1. (Setting GATE = 1 allows the Timer to be controlled by external input **INT1**, to facilitate pulse width measurements.) TR1 is a control bit in the Special Function Register **TCON.GATE** is in **TMOD**.

The 24-Bit register consists of three 8 bits registers (**TH1/TM1/TL1**). Setting the run flag (TR1) won't clear the registers.

The operation mode 0 is the same for Timer 0 and Timer 1. The corresponding Timer 0 signals are **TR0/TF0/INT0/TH0/TM0/TL0**. There are two different GATE bits, one for Timer 1 (**TMOD.7**) and one for Timer 0 (**TMOD.3**).

9.6.2 Mode 1

For this application, mode 1 is not enabled in microcontroller.

9.6.3 Mode 2

For this application, mode 2 is not enabled in microcontroller.

9.6.4 Mode 3

Either Timer 0 and Timer 1 in Mode 3 are a 24-bit Down Counter. As the count rolls over from all 0s to all 1s, it sets the Timer interrupt flag **TF1**. The counted input is enabled to the Timer when **TR1** = 1 and either **GATE** = 0 or **INT1** = 1. (Setting **GATE** = 1 allows the Timer to be controlled by external input **INT1**, to facilitate pulse width measurements.) **TR1** is a control bit in the Special Function Register **TCON.GATE** is in **TMOD**.

The 24-Bit register consists of three 8 bits registers (**TH1/TM1/TL1**). Setting the run flag (**TR1**) does not clear the registers. The operation mode 3 is the same for Timer 0 as for Timer 1. The corresponding Timer 1 signals are **TR1/TF1/INT1/TH1/TM1/TL1**. There are two different **GATE** bits, one for Timer 1 (**TMOD.7**) and Timer 0 (**TMOD.3**).

9.7 Signal Description

9.7.1 External Signal Description

PHT Digital Flywheel Tooth sensor input to Timer 2. Digital signal generated by analog sensor placed in to crankshaft.

9.7.2 Detailed Signal Descriptions

Table 68 – Interface description

Signal	I/O	Description		Reset
CLK	I	Clock interface to work timer/counter.		1
		State Meaning	Asserted: High Level of clock. Negated: Low level of clock.	
		Timing	Assertion: Synchronous with external clock. Negation: Synchronous with external clock.	
INT0	I	External on/off timer 0.		0
		State Meaning	Asserted: Turn on Timer 0 if GATE T0 = 0. Negated: Turn off Timer 0.	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
INT1	I	External on/off timer 1.		0
		State Meaning	Asserted: Turn on Timer 1 if GATE T1 = 0. Negated: Turn off Timer 1.	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
PHT	I	Digital Flywheel Tooth sensor input to Timer 2.		0
		State Meaning	Asserted: Start the Process of Timer 2. Negated: Stop the Process.	

Signal	I/O	Description		Reset
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TACPH	I/O	Accumulator (2 bits) Msb of estimated value from Angle Clock Period.		0
		State Meaning	Asserted: Not change. Negated: Update value.	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TACPL	I/O	Accumulator (8 bits) Lsb of estimated value from Angle Clock Period.		0
		State Meaning	Asserted: Not change. Negated: Update value.	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
ACR	O	Angle Clock Accumulator (3x8 bits) ACRL, ACRM, ACRL.		0
		State Meaning	Asserted: Not change. Negated: Update value.	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TCON	O	TCON Register to control Timers/Counter (bits TF1 and TF0)		0
		State Meaning	Asserted: Timer 0 and 1 overflow flag active. Negated: Timer 0 and 1 overflow flag inactive.	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TCON	I	TCON Register to control Timers/Counter Run (bits TR1 and TR0)		0
		State Meaning	Asserted: Run Timer 0 or 1. Negated: Stop Timer 0 or 1.	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TH1	O	Timer 1 Accumulator Most Significant bits (8 bits).		0
		State Meaning	Asserted: Not change. Negated: Update value.	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TM1	O	Timer 1 Accumulator Medium Significant bits (8 bits).		0
		State Meaning	Asserted: Not change. Negated: Update value.	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TL1	O	Timer 1 Accumulator Low Significant bits (8 bits).		0
		State Meaning	Asserted: Not change. Negated: Update value.	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TH0	O	Timer 0 Accumulator Most Significant bits (8 bits).		0
		State Meaning	Asserted: Not change. Negated: Update value.	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TM0	O	Timer 0 Accumulator Medium Significant bits (8 bits).		0
		State Meaning	Asserted: Not change. Negated: Update value.	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TL0	O	Timer 0 Accumulator Low Significant bits (8 bits).		0
		State Meaning	Asserted: Not change. Negated: Update value.	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TMOD.3	I	Control bit to Run or Stop Timer 0.		0

Signal	I/O	Description		Reset
(GATE T0)		State Meaning	Asserted: if INT0 = 1 Timer 0 Run. Negated: if TR0 = 1 Timer 0 Run.	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TMOD.7 (GATE T1)	I	Control bit to Run or Stop Timer 1.		0
		State Meaning	Asserted: if INT1 = 1 Timer 0 Run. Negated: if TR1 = 1 Timer 0 Run.	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TMOD.0 (M0T0)	I	Timer 0 mode selector bit (M0).		0
		State Meaning	Asserted: Ask item 1.7.1.1 Negated: Ask item 1.7.1.1	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TMOD.1 (M1T0)	I	Timer 0 mode selector bit (M1).		0
		State Meaning	Asserted: Ask item 9.3 Negated: Ask item 9.3	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TMOD.4 (M0T1)	I	Timer 1 mode selector bit (M0).		0
		State Meaning	Asserted: Ask item 9.3 Negated: Ask item 9.3	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TMOD.5 (M1T1)	I	Timer 1 mode selector bit (M1).		0
		State Meaning	Asserted: Ask item 9.3 Negated: Ask item 9.3	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TCON2.5 (TF2)	O	Timer 2 overflow flag.		0
		State Meaning	Asserted: Overflow occur Negated: Not occur overflow	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TCON2.6 (TR2)	I	Timer 2 run control bit.		0
		State Meaning	Asserted: Turn on timer 2. Negated: Turn off timer 2.	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TCON2.1 (DFSEL)	I	Digital Filter Sampling Selection		0
		State Meaning	Asserted: Output sampling S3 selected. Negated: Output sampling S2 selected.	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	
TCON2.0 (EDGESEL)	I	Rise-Fall Edge selection		1
		State Meaning	Asserted: Rise edge selection Negated: Fall edge selection	
		Timing	Assertion: Implemented in future. Negation: Implemented in future.	

9.8 Memory map and register definition

The timers have three associated registers for control and configuration, **TMOD**, **TCON** and **TCON2** that are located at the addresses specified in the statement that follows, these registers have a size of 8 bits.

9.9 Extra Information

Not applicable

9.10 Initialization Information

Both Timer 0 and Timer 1 in Mode 0 are a 24-bit Counter. As the count rolls over from all 1s to all 0s, it sets the Timer interrupt flag TF1. The counted input is enabled to the Timer when TR1 = 1 and either GATE = 0 or INT1 = 1. (Setting GATE = 1 allows the Timer to be controlled by external input INT1, to facilitate pulse width measurements.) TR1 is a control bit in the Special Function Register TCON.GATE is in TMOD.

Either Timer 0 and Timer 1 in Mode 3 are a 24-bit Down Counter. As the count rolls over from all 0s to all 1s, it sets the Timer interrupt flag TF1. The counted input is enabled to the Timer when TR1 = 1 and either GATE = 0 or INT1 = 1. (Setting GATE = 1 allows the Timer to be controlled by external input INT1, to facilitate pulse width measurements.) TR1 is a control bit in the Special Function Register TCON.GATE is in TMOD.

Timer 2 is turn on / off something setting bit **TR2** in register **TCON2.**

9.11 Application Information

Starting from the growth of the use electronics embedded on the surface mobile vehicle did necessary the creation and development of several devices for improvement of the acting and safety of those platforms, the necessity of a new device was evident. This device was called Flywheel.

He acts in the improvement of the motor acting optimizing his consumption and efficiency providing stability in his operation.

Through analog sensor the obtained signs are converted in digital signs after have been processed, are sent for an Unit of Control of the motor (Engine Control Unit - ECU) that it makes the necessary corrections in the injection control of the fuel mixture and ignition speed.

It is a auto-adaptive system to monitor and recognizes the changes that happen in the motor and it compensates them automatically acting in the Map Base of Fuel, progress and air flow in ECU.

This document provides all the functions and settings of timers, registers and their associated memory locations.

Further, details will be discussed about the operation of timer 2, (which is a specific application automotive), all your settings and associated records.

10 Baud Rate Block Description

10.1 Introduction

The baud rate module is responsible for the baud rate generation for the serial block. This generation occurs through the clock frequency division. In this implementation it works in two modes of operation, both fixes, according to the specification. The following figure illustrates the block diagram of baud rate module, with it input and output signals.

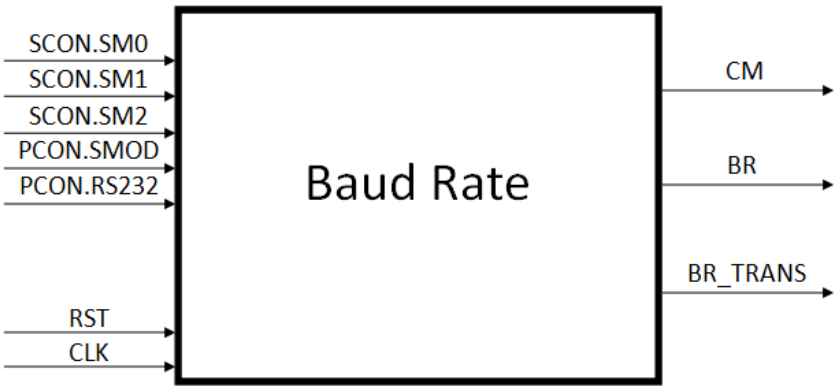


Figure 19 – Baud Rate block diagram

10.2 Overview

The exact value of clock frequency division is determined through two input signals, the SM0 value and the SMOD value. This value can be 2, 32, 64 or other four values pre-determined by RS232 mode.

10.3 Features

The brief description of Baud Rate block is described below.

- Baud rate generation for the serial block
- Machine cycle generation

10.4 Modes of operation

There are two modes of operation in the Baud Rate block. The modes of operation are defined by the SCON register and PCON register.

When the bit SM0 of SCON register is equal to 0 the mode of operation is mode 0. In this mode, the communication occurs in a rate equal at one machine cycle, but with the opposite signal value.

When the bit SM0 of SCON register is equal at 1 the mode of operation is mode 2. This mode can work with six divisors, according with SMOD and RS232 bit value, in PCON register. If the SMOD value is equal to 0, the frequency divisor is equal to 32, else, the frequency can work with five divisors, according with RS232, SM1 and SM2 bits values. If RS232 value is equal to 0, the frequency divisor is equal to 64, else, the SM1 and SM2 bit values are read for determine the transmit rate in bits per second. This values can be 9600, 19200, 57600 or 115200.

10.5 External signal description

N.A. There is not any external signal of the chip received by the baud rate module.

10.6 Detailed signal descriptions

The complete interface description is presented in the table below. The interface description includes both internal ports and external pins.

Table 69 – Interface description

Signal	I/O	Description		Reset
CLK	I	Interface data clock		0
		State Meaning	Asserted: Clock level high Negated: Clock level low	
		Timing	Assertion: Duty cycle 50% Negation: Duty cycle 50%	
RST	I	Synchronous reset		0
		State Meaning	Asserted: Normal operation Negated: Chip in the reset state	
		Timing	Assertion: Can be synchronous asserted Negation: Synchronous to clock	
SM0	I	Bit that determine mode of operation		0
		State Meaning	Asserted: Mode 2 active Negated: Mode 0 active	
		Timing	Assertion: Synchronous with clock Negation: Synchronous with clock	
SM1	I	Bit that determine communication rate in BPS		0
		State Meaning	Asserted: 57600 or 115200 BPS Negated: 9600 or 19200 BPS	
		Timing	Assertion: Synchronous with clock Negation: Synchronous with clock	
SM2	I	Bit that determine communication rate in BPS		0
		State Meaning	Asserted: 19200 or 115200 BPS Negated: 9600 or 57600 BPS	
		Timing	Assertion: Synchronous with clock Negation: Synchronous with clock	
RS232		Bit that determine if serial communication is active or not		0
		State Meaning	Asserted: Mode RS232 active Negated: Mode RS232 not active	
		Timing	Assertion: Synchronous with clock Negation: Synchronous with clock	
SMOD	I	Bit that determine the baud duplicator, in mode 2		0
		State Meaning	Asserted: Divisor equal at 32 Negated: Divisor equal at 64	
		Timing	Assertion: Synchronous with clock Negation: Synchronous with clock	
CM	O	Cycle machine output		0
		State Meaning	Asserted: First half of machine cycle Negated: Second half of machine cycle	
		Timing	Assertion: Synchronous with clock Negation: Synchronous with clock	
BR	O	Output of baud rate value		0
		State Meaning	Asserted: Serial clock level high Negated: Serial clock level low	
		Timing	Assertion: Synchronous with clock Negation: Synchronous with clock	

Signal	I/O	Description		Reset
BR_TRANS	O	Output of baud rate 16 faster times of output BR, in mode 2		0
		State Meaning	Asserted: Serial transition detector level high Negated: Serial transition detector level low	
		Timing	Assertion: Synchronous with clock Negation: Synchronous with clock	

10.7 Memory map and register definition

The memory map for Baud Rate interface registers consists of 8 bit registers with no special requirements. The memory map and registers details are in the following sections.

10.8 Memory map

Memory map for Baud Rate registers.

Table 70 – Memory map

Address/Offset	Register	Access	Reset Value	Section
0x9Fh	SCON	R	0x00h	8.2.1
0x87h	PCON	R	0x00h	8.2.2

10.9 Functional Description

10.9.1 Baud Rate Modes

The Baud Rate block has two modes of operation, which define the value of baud rate in serial block. In the mode 0, the generated baud rate is equal to machine cycle, and in the mode 2, the generated baud rate can be two values, depending of the baud rate duplicator bit value.

Table 71 – The Baud Rate mode of operation

Mode	SM0	SM1	SM2	SMOD	RS232	BR	BR_TRANS	CM
0	0	X	X	X	X	Fosc /2	0	Fosc /2
2	1	X	X	1	0	Fosc/32	Fosc /2	Fosc /2
2	1	X	X	0	0	Fosc /64	Fosc /4	Fosc /2
2	1	0	0	0	1	9600 BPS	153600 BPS	Fosc /2
2	1	0	1	0	1	19200 BPS	307200 BPS	Fosc /2
2	1	1	0	0	1	57600 BPS	921600 BPS	Fosc /2
2	1	1	1	0	1	115200 BPS	1843200 BPS	Fosc /2

10.10 Extra Information

N.A.

10.11 Initialization Information

Before using Baud Rate utilization, the internal registers must be properly initialized and the clock system must be synchronized. An example for configuration can be:

SM0 = 0**,** mode 0 configured

SM1 = 0, in mode 0 don't care

SM2 = 0, in mode 0 don't care

SMOD = 0, in mode 0 don't care

RS232 = 0, in mode 0 don't care

RST = 0 -> 1, reset mode exit

10.12 Application Information

N.A.

11 Interruption Module Description

11.1 Introduction

This block is responsible to generate an interrupt request to CPU. It evaluates the priority of different interrupts sources which can occur at same time and decides what is the interrupt that should be executed by CPU.

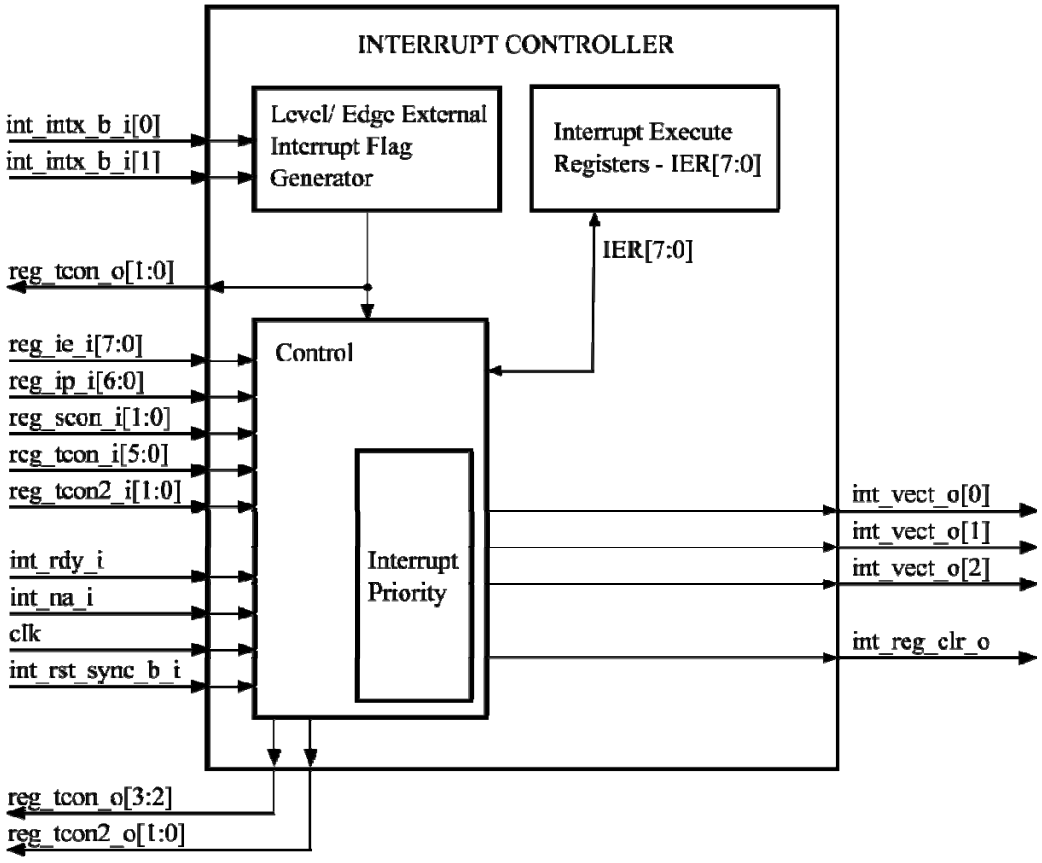


Figure 20: Interrupt Controller Block Diagram

11.2 Overview

The Interrupt Controller module evaluates and decides whether an interrupt request must be generated to CPU.

This module can monitor up to 8 interrupt sources. These sources are Timer 0, Timer 1, Timer 2 , Serial Communication Port (transmit and receive), External Pin 0, External Pin 1 and Transceiver.

Interrupt sources can be individually configurable through IE and IP registers.

11.3 Features

The EMC08 provides 8 interrupt sources and 7 vectors. The Figure shows how the IE and IP registers and the polling sequence work to determine which if any interrupt will be serviced.

11.4 Level/Edge External Interrupt Flag Generator

The block verify if the IE is set (EX0 and EX1), whether the EX0 or EX1is 1 the generator set the pin 3.3 and 3.2 as input (P3EN).

Other function of this block is check if the external interrupt is generate by level or edge, see section 11.13.

11.5 Control

The control block verify all input and the IER to decide if the interrupt will be generate.

Table 72: Port Description

int_rdy_i	int_na_i	Description
0	0	Interrupt accept
0	1	Interrupt not accept
1	0	Interrupt done, RETI
1	1	-

If the interrupt is not accepted the core block set the input int_na_i and when the core block finish a interrupt routine the input int_rdy_i is set.

11.6 IER - Interrupt Execute Registers

This registers show which interrupt is executing by core, the control block can read and write this register each machine cycle.

Bit	7	6	5	4	3	2	1	0
R W	RTXRX	RS	RT2	RT1	RX1	RT0	RX0	-
Reset	0	0	0	0	0	0	0	0

Field	Description
RTXRX	Executing Tranceiver interrupt bit.
RS	Executing Serial interrupt bit.
RT2	Executing Timer 2 interrupt bit.
RT1	Executing Timer 1 interrupt bit.
RX1	Executing External Interrupt bit.
RT0	Executing Timer 0 interrupt bit.
RX0	Executing External Interrupt 0 bit.

11.7 Priority Level Structure

Each interrupt source can also be individually programmed to one of two priority levels by setting or clearing a bit in Special Function Register IP. A low-priority interrupt can itself be interrupted by a high-priority interrupt but not by another low-priority interrupt. A high-priority interrupt can't be interrupted by any other interrupt source.

If two requests of different priority levels are received simultaneously, the request of higher priority level is serviced. If requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence, as follows:

Table 73: Priority Level

Source	Priority within level
/INT0	1
TF0	2
/INT1	3
TF1	4
TF2	5
RI + TI	6
TXRX	7

Note that the priority within level structure is only used to resolve simultaneous requests of the same priority level.

11.8 Modes of Operation

The Interrupt Controller module is always powered on. There is not low power consumption modes.

11.9 External Signal Description

Interrupt Controller module can be externally triggered, through external pins /INT0 and /INT1.

11.10 Detailed Signal Descriptions

The complete interface description is presented in the Table 6. The interface description includes both internal port and external pins:

Table 74: Interface Description

Signal	I/O	Description		Reset
int_int_b_i[0]	I	External Interrupt input 0		?
		State Meaning	Asserted: A transition from high to low or a low level of /INT0 triggers set correspondent external interrupt flag. Negated: A transition from low to high has no effect. In level mode, high level clears correspondent interrupt flag.	
		Timing	Assertion: an input high or low should be hold for at least one machine cycle to ensure sampling Negated: an input low to high should be hold for at least one machine cycle to ensure sampling	
int_int_b_i[1]	I	External Interrupt input 1		?
		State Meaning	Asserted: A transition from high to low or a low level of /INT1 triggers set correspondent external interrupt flag. Negated: A transition from low to high has no effect. In level mode, high level clears correspondent interrupt flag.	

Signal	I/O	Description		Reset
		Timing	Assertion: an input high or low should be hold for at least one machine cycle to ensure sampling Negated: an input low to high should be hold for at least one machine cycle to ensure sampling	
clk	I	Register Interrupt Clock		?
		State Meaning	Asserted: n/a Negated: n/a	
		Timing	Assertion: Duty Cycle 50 % Negation: Duty Cycle 50 %	
int_rst_sync_b_i	I	Synchronous reset		?
		State Meaning	Asserted: Chip in reset state Negated: Normal Operation	
		Timing	Assertion: Can be asynchronously asserted Negation: May occur at any time, synchronous to internal clock	
int_na_i	I	Interrupt Not Accepted		?
		State Meaning	Asserted: High level indicates a not accepted interrupt by core Negated: Low level indicates a accepted interrupt by core	
		Timing	Assertion: an input high should be hold for at least one machine cycle to ensure sampling Negated: an input low should be hold for at least one machine cycle to ensure sampling	
reg_ie_i[7:0]	I	Interrupt Enable Register Bus - IE[7:0]		0
		State Meaning	Asserted: Data Registers Negated: Data Registers	
		Timing	Assertion: Synchronous with internal clock Negation: Synchronous with internal clock	
reg_ip_i[6:0]	I	Interrupt Priority Register Bus – IP[6:0]		0
		State Meaning	Asserted: Data Registers Negated: Data Registers	
		Timing	Assertion: Synchronous with internal clock Negation: Synchronous with internal clock	
reg_scon_i[1:0]	I	Interrupt Flag Register – SCON[1:0]		0
		State Meaning	Asserted: Data Registers Negated: Data Registers	
		Timing	Assertion: Synchronous with internal clock Negation: Synchronous with internal clock	
reg_tcon_i[5:0]	I	Interrupt Flag Register – TCON[5:0]		0
		State Meaning	Asserted: Data Registers Negated: Data Registers	

Signal	I/O	Description		Reset
		Timing	Assertion: Synchronous with internal clock Negation: Synchronous with internal clock	
reg_tcon2_i[1:0]	I	Interrupt Flag Register - TCON2[1:0]		0
		State Meaning	Asserted: Data Registers Negated: Data Registers	
		Timing	Assertion: Synchronous with internal clock Negation: Synchronous with internal clock	
int_rdy_i	I	Interrupt Ready		?
		State Meaning	Asserted: High level indicates interrupt routine finished by core Negated: Low level indicates an interrupt routine still executed by core	
		Timing	Assertion: an input high should be hold for at least one machine cycle to ensure sampling Negated: an input low should be hold for at least one machine cycle to ensure sampling	
int_vect_o[2:0]	O	Interrupt Address Vector – int_vect[2:0]		0
		State Meaning	Asserted: Interrupt Address Vector Negated: Interrupt Address Vector	
		Timing	Assertion: Synchronous with internal clock Negation: Synchronous with internal clock	
reg_tcon_o[3:0]	O	Interrupt Flag Register – TCON[3:0]		0
		State Meaning	Asserted: Data Registers Negated: Data Registers	
		Timing	Assertion: Synchronous with internal clock Negation: Synchronous with internal clock	
reg_tcon2_o[1:0]	O	Interrupt Flag Register – TCON2[1:0]		0
		State Meaning	Asserted: Data Registers Negated: Data Registers	
		Timing	Assertion: Synchronous with internal clock Negation: Synchronous with internal clock	
int_reg_clr_o	O	Interrupt Register Clear		0
		State Meaning	Asserted: Set when has new registers values Negated: Clear when has no new register values	
		Timing	Assertion: Synchronous with internal clock Negation: Synchronous with internal clock	

11.11 Memory Map and Register Definition

The memory map for Interrupt Controller module registers consists of 8 bit registers with no special requirements. The memory map and registers details are in the following sections.

11.12 Functional Description

11.12.1 Modes of Operation

The interrupt flags are sampled at every machine cycle. The samples are polled during the following machine cycle. If one of the flags was in a set condition at preceding cycle, the polling cycle will find it and the core block will generate an LCALL to the appropriate service routine, provided this hardware-generated LCALL is not blocked by any of the following conditions:

- An interrupt of equal or higher priority level is already in progress.
- The current (polling) cycle is not the final cycle in the execution of the instruction in progress.
- The instruction in progress is RETI or any write to the IE or IP registers.

Any of these three conditions will block the generation of the LCALL to the interrupt service routine. Condition 2 ensures that the instruction in progress will be completed before vectoring to any service routine. Condition 3 ensures that if the instruction in progress is RETI or any access to IE or IP, then at least one more instruction will be executed before any interrupt is vectored to.

The polling cycle is repeated with each machine cycle and the values polled are the value that were present at previous machine cycle. Note then that if an interrupt flag is active but not being responded to for one of the above conditions, and is not still active when the blocking condition is removed, the denied interrupt will not be serviced. In other words, the fact that the interrupt flag was once active but not serviced is not remembered. Every polling cycle is new.

Thus the processor acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate service in routine. In some cases it also clears the flag that generated the interrupt, and in other cases it doesn't. It never clears the Serial Port or Timer 2 flags. This has to be done in the user's software. It clears an external interrupt flag (IEO or IE1) only if it was transition-activated. The hardware-generated LCALL pushes the contents of the Program Counter onto the stack (but it does not save the PSW) and reloads the PC with an address that depends on the source of the interrupt being vectored to, as shown bellow.

Table 75: Interrupt Address

Source	Vector Address
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
TF2	0023H
RI + TI	002BH
TXRX	0033H

Execution proceeds from that location until the RETI instruction is encountered. The RETI instruction informs the processor that this interrupt routine is no longer in progress, then pops the

top two bytes from the stack and reloads the program Counter. Execution of the interrupted program continues from where it left off.

Note that a simple RET instruction would also have returned execution to the interrupted program, but it would have left the interrupt control system thinking an interrupt was still in progress.

11.13 External Interrupts

The external sources can be programmed to be level-activated or transition-activated by setting or clearing bit IT1 or IT0 in register TCON. If ITx = 0, external interrupt x is triggered by a detected low at the INTx pin. If ITx = 1, external interrupt x is edge-triggered. In this mode if successive samples of the INTx pin show a high in one cycle and a low in the next cycle, interrupt request flag IEx in TCON is set. Flag bit IEx then requests the interrupt.

Since the external interrupt pins are sampled once each machine cycle, an input high or low should hold for at least one machine cycle to ensure sampling. If the external interrupt is transition-activated, the external source has to hold the request pin high for at least one machine cycle, and then hold it low for at least one machine cycle to ensure that the transition is seen so that interrupt request flag IEx will be set. IEx will be automatically cleared by the CPU when the service routine is called.

If the external interrupt is level-activated, the external source has to hold the request active until the requested interrupt is actually generated. Then it has to deactivate the request before the interrupt service routine is complete or else another interrupt will be generated.

11.14 Initialization Information

The reset is synchronous and active low. This signal clears all bits in the internal register, IER[7:0] and sets the output int_vect[2:0]_o to logic level 0.

12 Ports Block Description

12.1 Introduction

The EMC08 microcontroller 8 bit is composed of several blocks like to see in the figure 1, one being the block ports that will be introduced in this document. The block ports consistency of the 4 ports of the 8 bits bidirectional (P0-P3) and one unidirectional port (P4) 8 bits too, like show in Figure 21. The port 3 (P3) is multifunctional and it can have several configurations.

All ports can be configured as inputs or outputs. Accordingly, in total of 32 input/output pins and 8 outputs more enabling the microcontroller to be connected to peripheral devices are available for use. Pin configuration, i.e. whether it is to be configured as an input (1) or an output (0), depends on its logic state defined for the P0EN-P4EN registers. In order to configure a pin microcontroller as an input, it is necessary to apply zero logic (0) to appropriate I/O port bit. In this case, voltage level on appropriate pin will be 0.

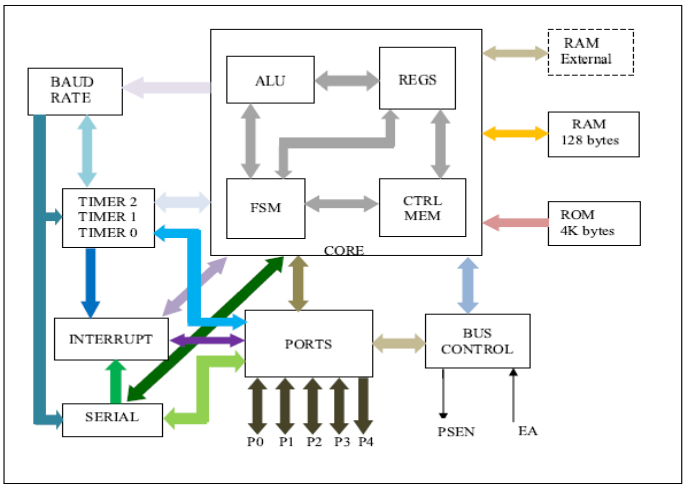


Figure 21 – Digital top block diagram

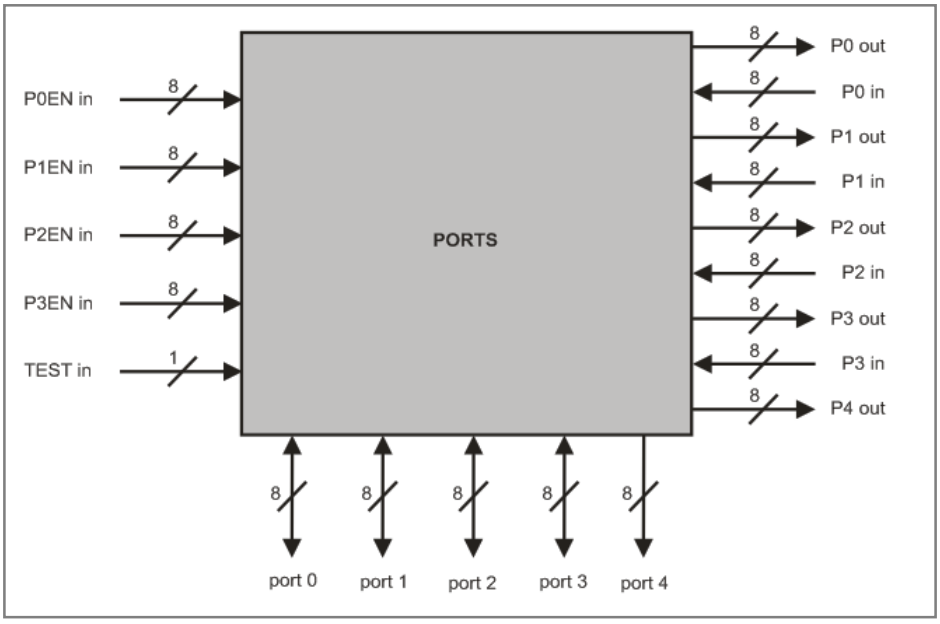


Figure 22– Ports block diagram

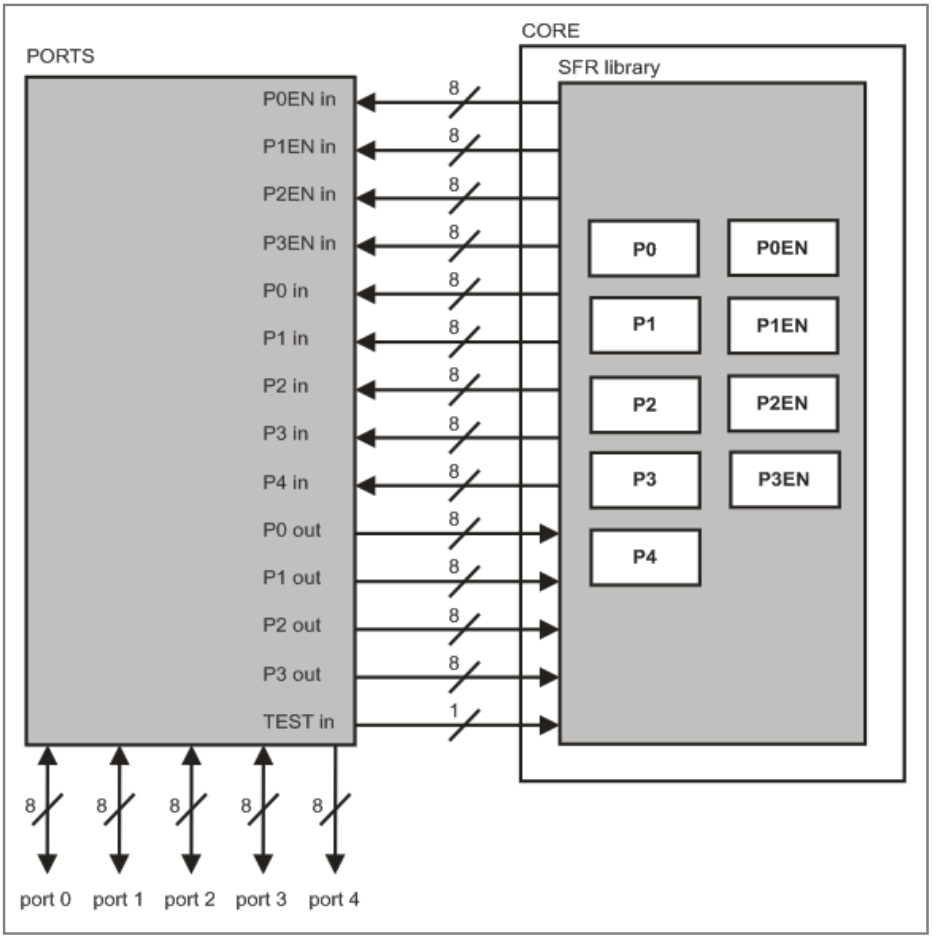


Figure 23 – Relation between the PORTS block and CORE block

12.2 Overview

12.2.1 Functionality:

Setting: The block has the functionality to configure ports each pin input/output (I/O) to be used by different blocks in the Tx/Rx data information or addresses.

Control: Ports Block don't has control functionality, it is solely responsible for configuring the ports as I/O, their decision inputs or outputs will be the responsibility of other blocks (interrupts, Tx, Rx serial access memory).

The figure below is showed the several ports with their several functionalities.

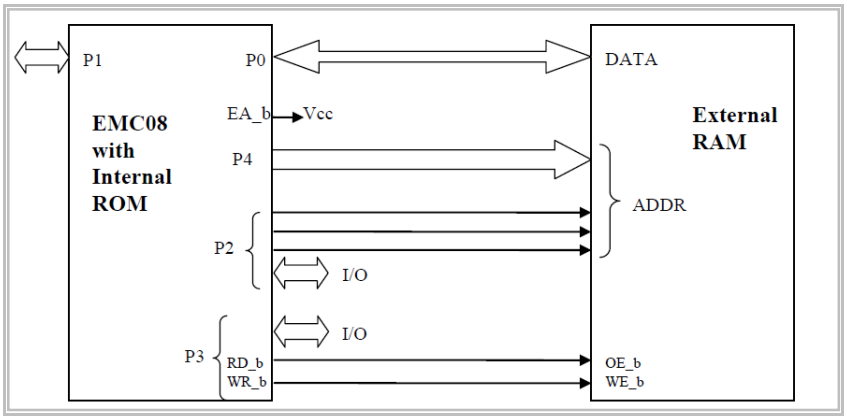


Figure 24 – Ports block diagram

12.3 Features

Ports Block is connected right the blocks: interrupts, serial and Bus Control (access to external memory). Each of these blocks will be responsible for setting the bits of records that each port has, making possible the flow of data.

It can be seen in Table 5 records the settings of P [3:0] EN for each port to behave as I/O (input or output).

- Ports Block is a module completely passive, them it is dependent on the changes performed by other modules system.
- It has the functionality to be prepared to configure the ports so that events involving the flow of data (Tx/Rx) for these ports will assist.
- No functionality block ports making any kind of access control and/or data type.

12.4 Important Considerations

- When needed, access to the external memory, input buffers on port 0 and output drives to ports 0, 2, 4 are used:
  - In this case, the data bus in port 0 is bidirectional;
  - The port 2 has an output byte in the most significative address byte of the external memory when the address is 16 bits. In other words, the port 2 pins continue emitting the contents of SFR.
- The drive from the port 0 can be:
  - The bus data from external memory;
  - Or the I/O general purpose.
- The port 2 can be changed to:
  - The high address bus;
  - Or the I/O general purpose.
- The output drivers of Ports 0 can be switchable to DATA BUS or General Purpose I/O and Port 2 can be switchable to HIGHER ADDRESS BUS to General Purpose I/O by an internal CONTROL signal for its external memory access.
  - The port 4 is the address bus down. During the access to external memory the rest of special registers P0/P2 remain unchanged.

- Each port of the I/O can be independently used as input to the output by P [3:0] configured for records.
- (Ports 0 and Ports 2 can't be used for I/O general purpose when it is used as address or data bus).

12.5 Modes of operation

In the EMC08 SOW the Ports Block in study say that this block has two configurations modes and anything can be input or output. These configurations are the following:

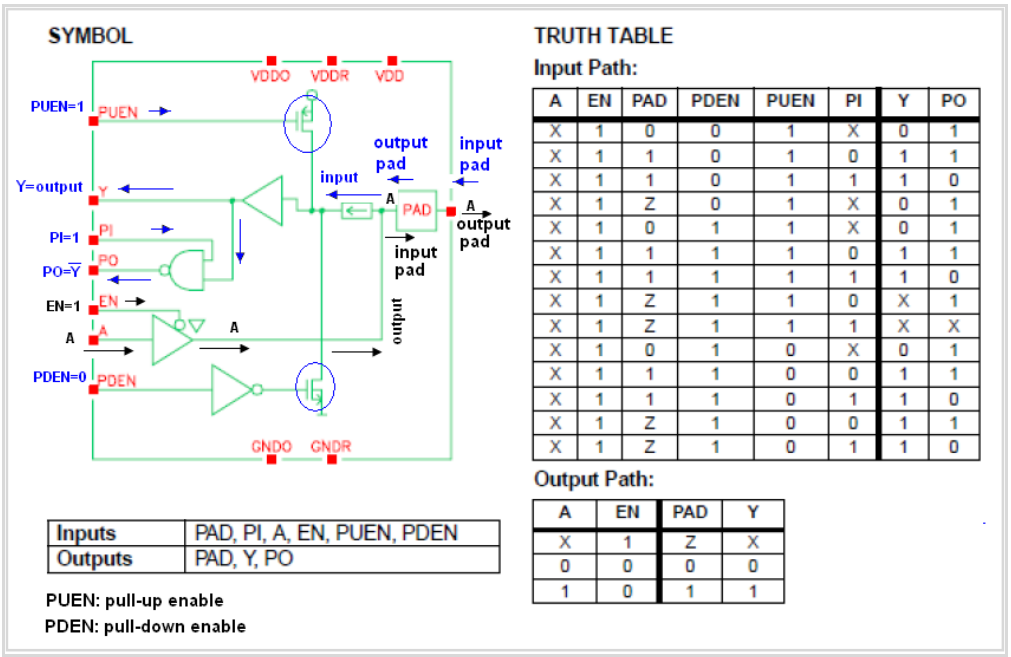
12.6 I/O PADs Configurations

- **Considering that each port is as follows:**

Figure 25 – Parts ports block

- **Port configuration.**

In the figure 6 are showed the diagram functional of a I/O buffer pins on each of the ports.



A	EN	PAD	PDEN	PUEN	PI	Y	PO
X	1	0	0	1	X	0	1
X	1	1	0	1	0	1	1
X	1	1	0	1	1	1	0
X	1	Z	0	1	X	0	1
X	1	0	1	1	X	0	1
X	1	1	1	1	0	1	1
X	1	1	1	1	1	1	0
X	1	Z	1	1	0	X	1
X	1	Z	1	1	1	X	X
X	1	0	1	0	X	0	1
X	1	1	1	0	0	1	1
X	1	1	1	0	1	1	0
X	1	Z	1	0	0	1	1
X	1	Z	1	0	1	1	0

A	EN	PAD	Y
X	1	Z	X
0	0	0	0
1	0	1	1

Figure 26 – I/O Circuits of the PADs

12.7 Circuit Configuration

Input/Output (I/O) pin:

Figure above illustrates a simplified schematic of all circuits within the microcontroller connected to one of its pins. It refers to all the pins except those of the P0 port which do not have pull-up resistors built-in.

How does work the input port microcontroller?

When PUEN (Pull-up is in enable = 1) then the transistor is biased and allows the power circuit, and enables the output Y. In another circuit would be in open circuit.

When PDEN (Pull-down is not in enable = 0) for the presence of the denied buffer transistor is polarized and closes the circuit.

The output Y can be used as denied that activating the input (PI = 1).

If PI = 0, only if Y is the output only.

Input pin

A logic one (1) is applied to a bit of the P register. The output FE transistor is turned off and the appropriate pin remains connected to the power supply voltage over a pull-up resistor of high resistance.

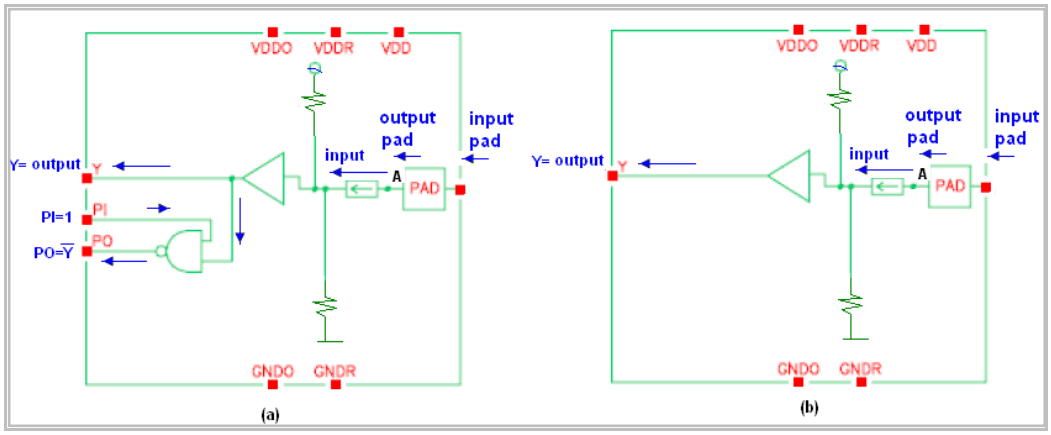


Figure 27 – Circuit for the Input PADs configurations

Table 76 – Inputs and Outputs for the INPUT PADs

Inputs	EN = 0 PUEN = 1 PDEN = 0 If PI = 1 them figure (a) If PI = 0 them figure (b)	Output	If PI = 1 them output = Y and PO If PI = 0 them output = Y

How does work the output port microcontroller?

If EN = 0 in this case the output would be high impedance (Z).

If EN = 1 in the case allows the pitch of the A signal that is sent to the PAD.

Logic zero (0) is applied to a bit of the P register. The output EN transistor is turned on, thus connecting the appropriate pin to ground.

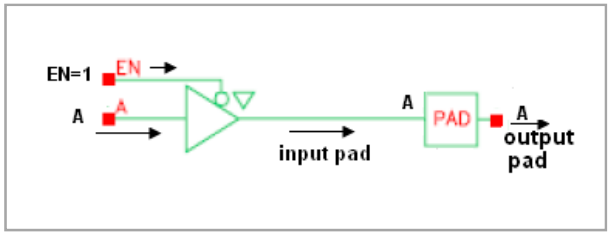


Figure 28 – Circuit for the output configurations

Table 77 – Inputs and Outputs for the OUTPUT PADs

Inputs	EN = 1 PUEN = 0 PDEN = 1 A(Data)	**Outputs**	A (data)

The Ports Block will configure the PAD cells of each pin through pin EN of this cell. When the Ports Block configures the pin EN with value equal to 0 (zero) the PAD cell will be configured to receive data coming from the Ports Block through pin A. When the Ports Block configures the pin EN with value equal to 1 (one) the PAD cell will be configured to receive data from the external environment and send these data to the Ports Block through pin Y.

12.8 Mode Test

This operation mode is related with the test of Ports block. When the TEST in Port is set to 1, the test mode is being activated. At this moment the p0 and p1 ports are set as output and p2 port as input. To set the TEST in port with value equal to zero disables the block test mode.

12.9 External signal description

Four ports of the EMC08 are bidirectional (P0-P3). Each of these ports consists of a latch (Special Function Registers P0 through P3), an output driver and an input buffer. Port 4 is the output of the address bus. The I/O ports (P0-P3) are bit configured by SRF registers (P[3:0]EN) and the P3.1 has an special bit configuration in PCON register (P3SEL) to select the output source from TXD (Serial) or P3.1 register.

Hence, each port can be configured as input or output, them this module can be considered as a sub-module of the ports block like show in the figure 9.

The figure 9 shows an example of the ports module configuration, with pad 0 as input and pad 7 as output for port 0. It indicates the right values for EN, PUEN, PDEN, PI, etc. As well as, the figure 7 and 8 and tables 5 and 6 show the same information.

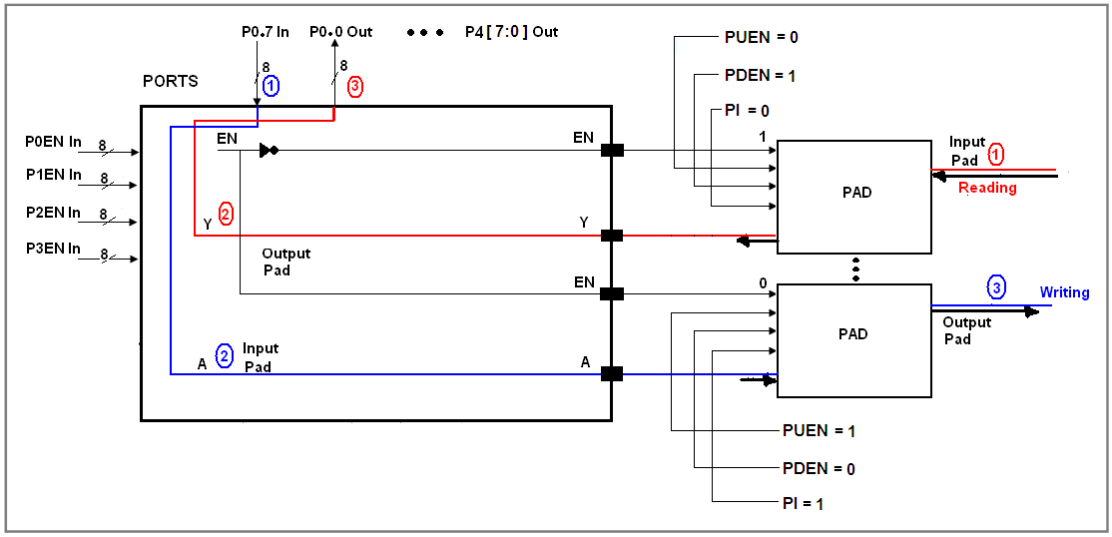


Figure 29 – Example for the Input/Output Configurations Port0

12.10 Functional Description

The ports module will utilize for everything modules for the transmission or reception information govern for his.

The single functionality for the module ports is to configure e port like input or output, that going to work depended of the P0EN, P1EN, P2EN and P3EN registers.

12.11 Extra Information

There is not information extra additional with respect this module.

12.12 Initialization Information

The ports will be initialized on writing mode after receive a reset stimuli.

12.13 Application Information

The ports module does not control any information that pass through it, the usage of this information depend of others modules that are being used this module.

13 Serial Block Description

13.1 Introduction

The serial block of the Microcontroller EMC08 provides control and register through of the Special Function Register SCON and SBUF, for the signals transmission and reception (TB8 and RB8) in the internal communication with other blocks, as ports (P3EN, P3.0 and P3.1), interrupts (TI and RI), baud rate (BR) and core. The top level diagram is shown below:

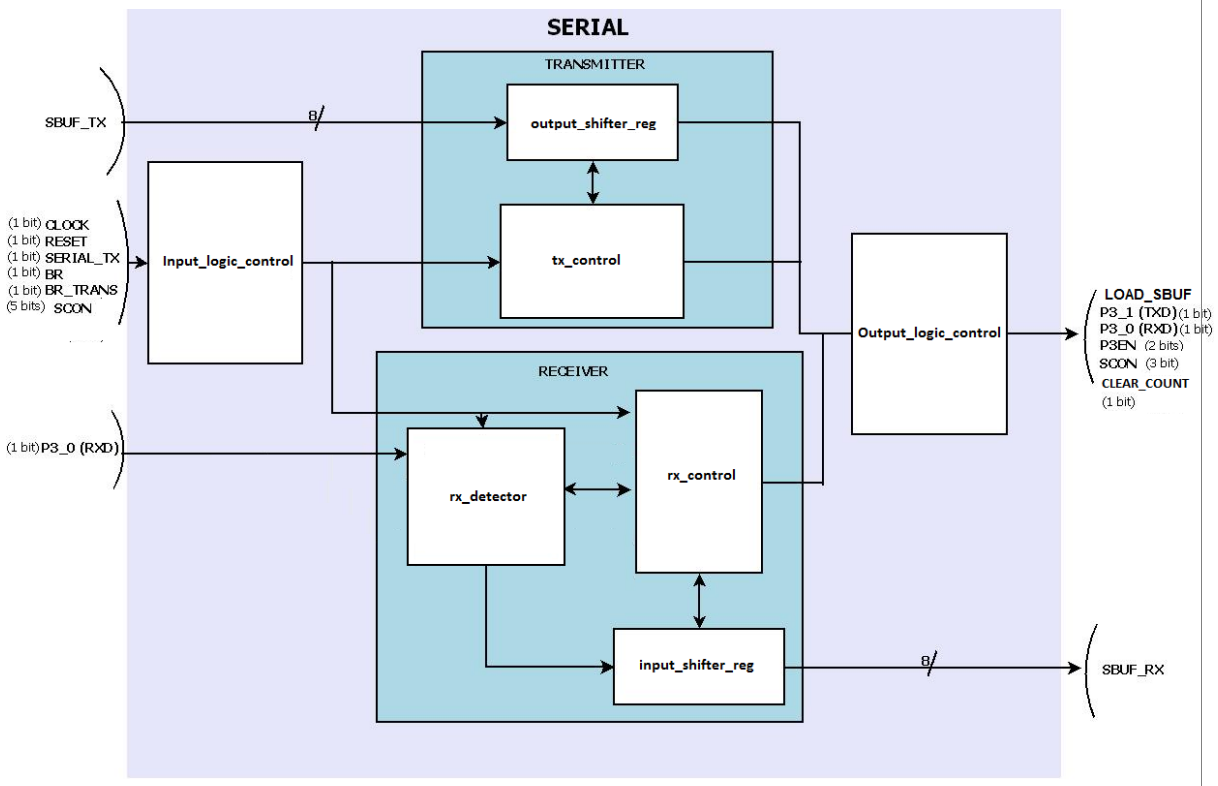


Figure 1 - Block Serial Diagram

13.2 Overview

The Serial block is responsible for receive and transmit data through the serial ports. It has the synchronous and asynchronous mode and can work with 8 (synchronous) or 9 (asynchronous) data bits.

The serial blocks have a special register called SCON which controls the serials modes and interrupts. Added, there is a serial buffer register (SBUF) which transmits and receives registers are physically separately.

It is possible to set seven different baud rates depending on the operation modes. All of them are fixed rates according with the machine cycle. The Baud Rate module will provide these rates.

Inside the Serial block, there are two main blocks, called Transmitter and Receiver which controls the main functions of the serial transmission as shown in the Figure 1.

13.3 Serial Features

The brief descriptions of main block serial functionalities are described below:

- Mode 0: 8 bits are transmitted and received synchronous on communication half duplex
- Mode 2: 9 bits, transmitted and received asynchronous on communication full duplex
- Special Buffer Register SBUF, TXD/RXD by accesses a physically separate receive/transmit register.

13.4 Modes of operation

The serial port can operate in two different modes, Mode 0 and Mode 2. Some modifications according to specifications were done. In the previous architecture based on the 8051, there were 4 modes of operation. Because the special specification, the EMC project does not need the Mode 1 and Mode 3, then these modules were excluded. This way, just one bit is necessary to set the operation mode. As conversion and project decision this bit will be SM0 bit in the SCON register (will be shown in the next sections). Added to this decision the Multiprocessor operation mode (SM2 bit) will not be implemented as a project decision, so this bit was excluded to. The bits SM1 and SM2 will not affect the Serial Mode and will be ignore.

13.4.1 Mode 0

Serial data enters and exits through RXD. TXD outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The baud rate is fixed according to the oscillator frequency.

13.4.2 Mode 1

RESERVED.

13.4.3 Mode 2

11 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit (TB8 in SCON) can be assigned the value of 0 or 1. Or, for example, the parity bit (P, in the PSW) could be moved into TB8. On the receive side, the 9th data bit goes into RB8 in Special Function Register SCON, while the stop bit is ignored. The baud rate is programmable to either 1/32 or 1/64 the oscillator frequency. Added, it is possible to use other rates generated by the Baud Rate module to use with serial Mode 2. The news rates are 9600bps, 19200bps, 57600bps and 115200bps.

In all modes, transmission is initiated by any instruction that uses SBUF as a destination register. Reception is initiated in Mode 0 by the condition RI = 0 and REN = 1. Reception is initiated in the other modes by the incoming start bit if REN = 1.

13.5 Signal description

13.5.1 External signals

The communication interface between the block serial and exterior is through the pin RESET and the signal TXD/RXD enable (P3EN) by the pin P3.0 for RXD and P3.1 for TXD. It must be provided 16 external pins only for serial communication in one side (Two ports of).

SERIAL_TX is a direct wire to the Serial block. It is a flag coming from Core FSM to start the transmission. It flag is 1 when some data is beginning to write in SBUF_TX.

13.6 Functional Description

The Serial block is responsible for transmit and receive data in a serial form according to the baud rate generated by the Baud Rate block (referred in other documentation). The transmission is Full-Duplex and half-duplex depending on the operational mode, where the register SBUF (Serial Buffer) is divided physically in two registers, the transmitter and the receiver registers to be used as a data buffer. The data, depending if the operation is transmit or receive, goes to different registers.

There are two operational modes in Serial. One mode intended to synchronous transmission (Mode 0), and other for asynchronous transmission (Mode 2). In the synchronous mode, the port P3.0 is used to both functionalities of data receiver and transmitter (RXD), while the port P3.0 is used as output port for the transmitted shifted reference signal (TXD). Added, in Mode 0, just 8 data bits are transmitted as a shifter operation.

In the operational Mode 2, the port P3.0 is used as an input port to the receiver buffer (RXD), while the port P3.1 is used as output port to the transmitter register (TXD). In the Serial Mode 2, 11 bits are used. 2 bits are used for start and stop bits (1^{st} and 11^{th}), 1 bit (10^{th}) as a special data bit (meaning the 9^{th} data bit) and 8 data bit (2^{nd} to 9^{th}).

The Serial configuration is done through the SCON special register. This register is constantly monitored, and according to the bits values, the serial mode is set. For more details of this register, see the previous sections. Besides the configuration, the register SCON has the interruption serial bits, and the 9^{th} received and transmit bit for Mode 2.

In the Serial operation, there are two interrupt flags, the TI and RI. The TI (transmitter interrupt), is set by hardware after the end of a transmission and can only be cleared by software. The RI (reception interrupt), is set by hardware after the end of a reception operation, and can only be cleared by software.

13.7 Internal Blocks

For the description of a more detailed functionality of the serial block in the microcontroller EMC08, this section is subdivided in receive block and transmission block, as it was shown in the figure 1.

The block RECEIVER contains a “rx_detector” block which detects transitions 1-to-0 that can starts the receiver. Included in the “rx_detector” block there is a bit detector that samples the data and validates it. In order to receive the data according to the baud rate, there is an “input_shifter_reg” that controls it in both modes 0 and 2.

The transmission block contains “serial_tx” block, which makes the control of the register SBUF for the transmission of the data received by CORE (from user). In order to shift out the transmitted data there is an “output_shift_register” that controls the data out. Added, the interface with the ports and external data are done with “input and output logic controls” blocks.

13.7.1 Receive Block

In the Receive block, the serial data enters through rx_data, and depending the operation mode goes to “rx_detector (mode 2) or “output_shifter_reg” (mode 0). In the mode 0, 8 data bits are received (LSB first) according to the baud rate. In mode 2, 11 bit are received, where 2 are start and stop bits (bits 0 and 10), 1 are the 9^{th} bit (special bit) and 8 data bits (1^{st} to 8^{th}). The follow subsections describes more detailed the functionality.

13.7.1.1 MODE 0

Reception is initiated by the condition REN = 1 and RI = 0. The RX Control unit writes the bits 11111110 to the receive shift register (input_shifter_reg), and after activates the reception. Additionally the ports are configured P3.0 and P3.1 to work in agreement with the mode 0 of reception.

As data bits come in from the right, 1s shift out to the left. When the 0 that was initially loaded into the right most position arrives at the left most position in the shift register, it flags the RX Control block to do one last shift and load SBUF.

13.7.1.2 MODE 2

In the MODE2 reception is initiated by a detection of 1 to 0 transition at RXD. For this purpose RXD is sampled at a rate of 16 times whatever baud rate has been established (the signal “br_trans” do it). This way there a counter to control the data received. When a transition is detected, the divide by 16 counter is immediately reset, and 1FFH (111111111) is written to the input shift register.

At the 7^{th}, 8^{th} and 9^{th} counter states of each bit time, the bit detector samples the value of RXD. The value accepted is the value that was seen in at least 2 of the 3 samples. If the value accepted during the first bit time is not 0, the receive circuits are reset and the input goes back to looking for another 1 to 0 transition. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register, it flags the RX Control Block to do one last shift, load SBUF and RB8, and set RI. After, the unit goes back to looking for a 1 to 0 transition al the RXD input.

The figures below Figure 2 and Figure 3 shows more details of the serial receiver internal block. Figure 2 shows the interconnections among the internal blocks, while the Figure 3 shows the “rx_control” FSM in a general view.

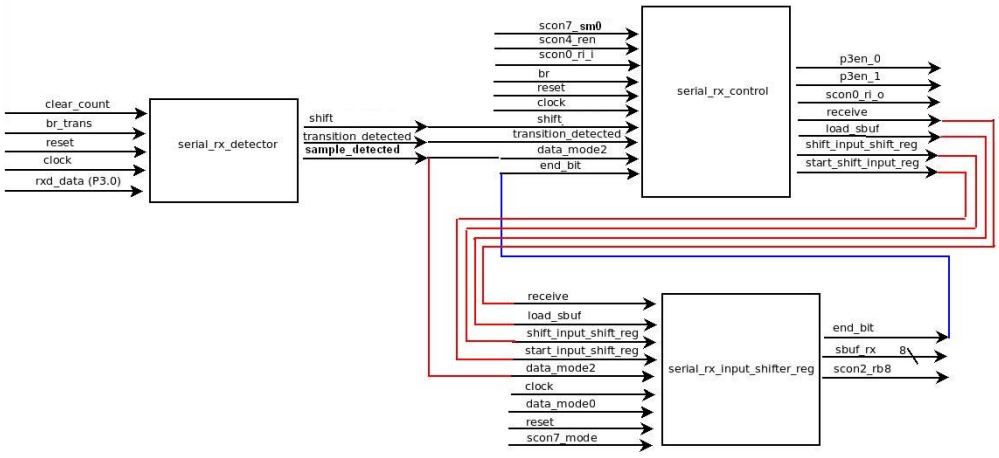


Figure 30 - Reception Serial Diagram

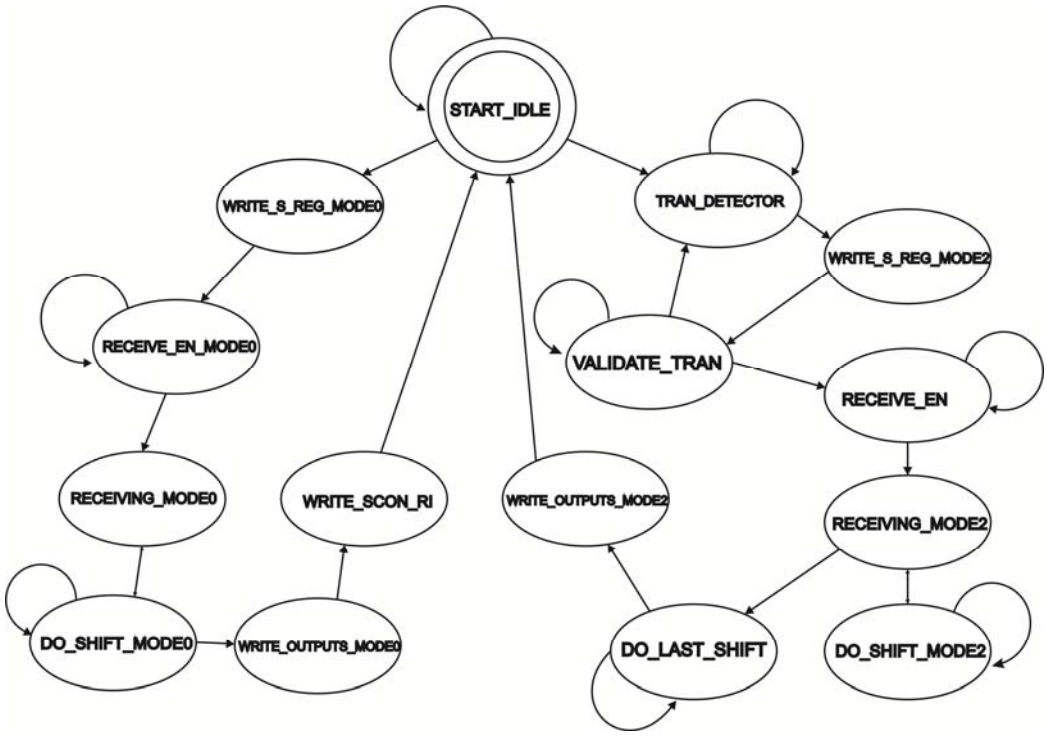


Figure 31 - FSM Reception Serial Block (rx_control)

13.7.2 Transmission Block

13.7.2.1 MODE 0

In this mode, TXD outputs the shift clock. 8 bits ate transmitted (LSB first) by the RXD port, since this mode is half-duplex. Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" (serial_tx signal) load a 1 into the 9th position of the transmit shift register and tells the "tx_control" block to commence a transmission. After it, the signal SEND will be activated.

SEND enables the output of the shift register (output_shifter_reg) to transmit the data through the P3.0 (RXD), and also enables SHIFT CLOCK to be transmitted in P3.1 (TXD). The contents of the transmit shift register are shifted to the right one position.

As data bits shift out to the right, zeroes come in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position is just to the left of the MSB, and all positions to the left of that contain zeroes.

This condition flags the "tx_control" block to do one last shift and then deactivate SEND and set TI.

13.7.2.2 MODE 2

On transmit, the 9th data bit (TB8) can be assigned the value of 0 or 1. On receive the 9th data bit goes into RB8 in SCON. The baud rate is programmable to either 1/32 or 1/64 the oscillator frequency in Mode2. Added the baud rates can assume other fixed values of 9600bps, 19200bps, 57600bps and 115200bps.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" ("serial_tx" signal) loads TB8 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested.

The transmission begins with activation of SEND, which puts the start bit at TXD. After, DATA is activated, which enables the output bit of the transmit shift register to TXD. The first shift clocks a 1 (the stop bit) into the 9th bit position of the shift register. Thereafter, only zeroes are clocked in. Thus, as data bits shift out to the right, zeroes are clocked in from the left. When TB8 is at the output position of the shift register, then the stop bit is just to the left of TB8, and all positions to the left of that contain zeroes. This condition flags the TX Control unit to do one last shift and then deactivate SEND and set TI.

The figures below Figure 4 and Figure 5 shows more details of the serial transmitter internal block. Figure 4 shows the interconnections among the internal blocks, while the Figure 5 shows the "tx_control" FSM in a general view.

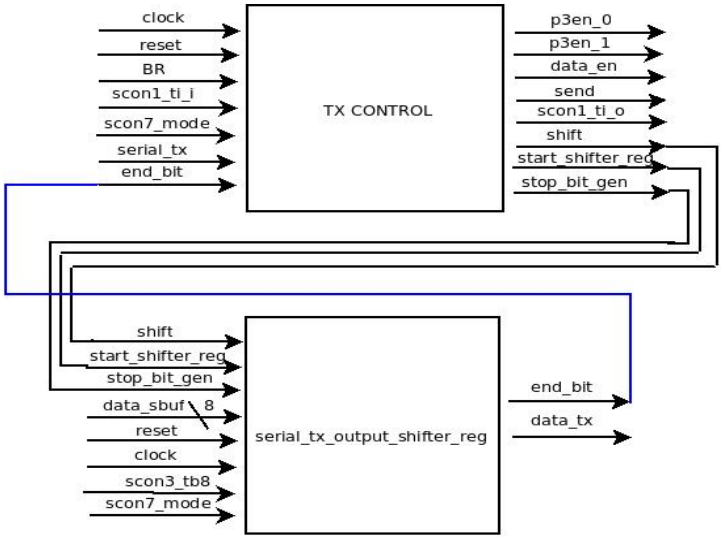


Figure 32 - Transmission Serial Block

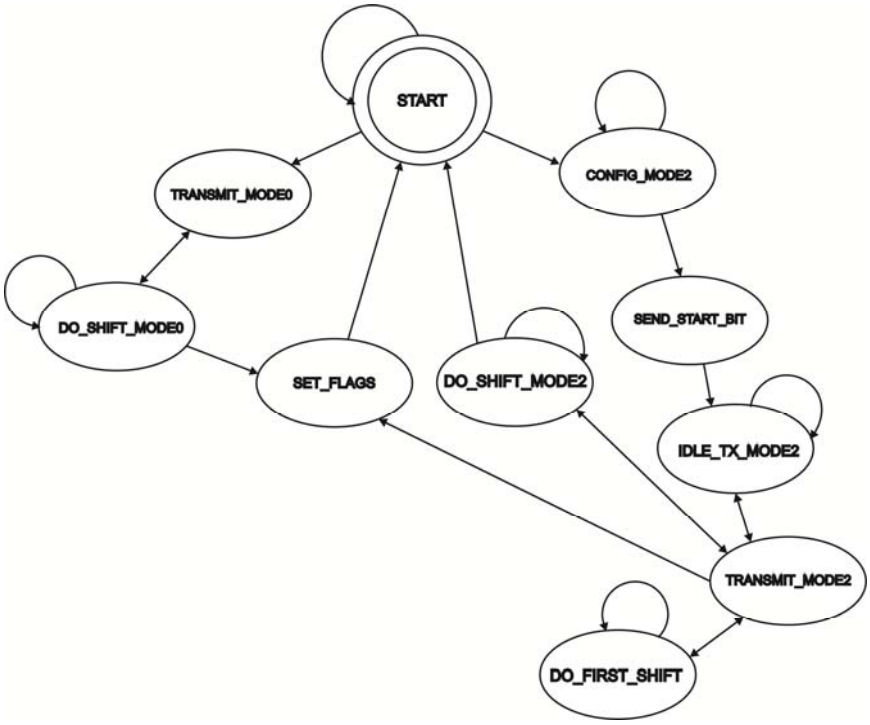


Figure 33 - FSM Transmission Serial Block

13.8 Functional Timing Diagrams

The next diagrams are representative and need to be changed according to the architecture that will be defined in Core block. After it, this section will be updated, but these pictures can be used as a guideline.

13.8.1 Mode Synchronous (mode 0)

This mode works synchronously and have a reference signal in TXD as reference to the receive side.

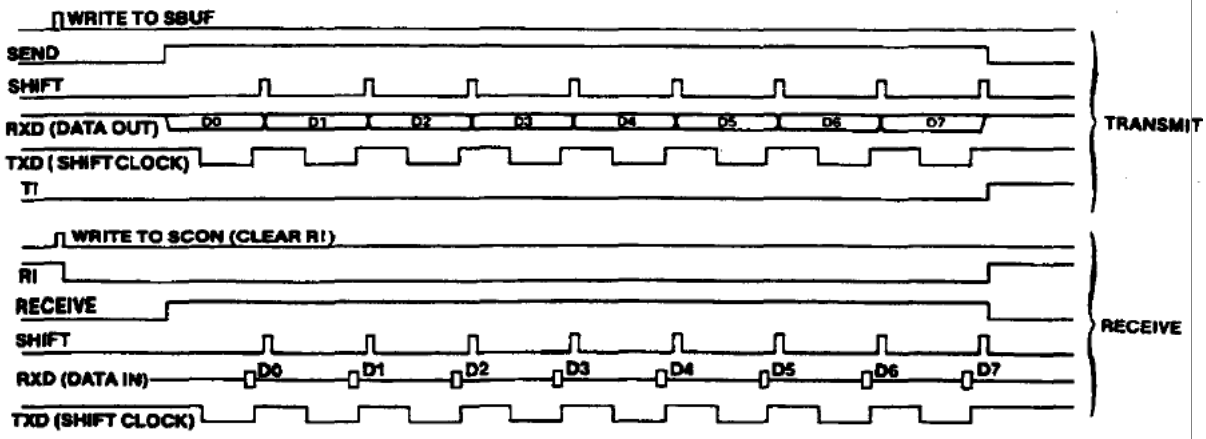


Figure 34 – Timing functionality in mode 0

13.8.2 Mode Asynchronous (mode 2)

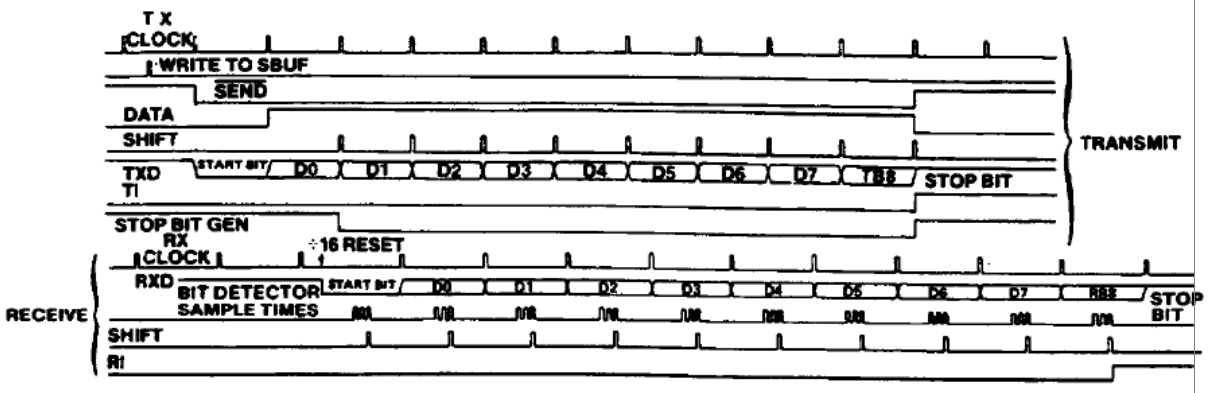


Figure 35 – Timing functionality in mode 2

13.9 Initialization Information

Before using Serial utilization, the internal registers must be properly initialized and the clock system must be synchronized. An example for configuration can be:

SCON SM0 = 0**,** mode 0 configured

P3EN_0 and P3EN_1 = 0 (output as default)

P3.0 and P3.1 = 0 (output as default)

SCON REN = 0 (no reception active)

SCON TB8, RB8 = don’t care (used only in mode 2)

SCON RI and TI = 1 (cleared by software only)

SBUF_TX and SBUF_RX = 8 ´b00000000 (initial value)

CLEAR_COUNT = 0 (do nothing, since the reset sync the signals)

RST = 0 -> 1, reset mode exit

14 Failure Analysis Information

14.1 Latch Divergence Environment

To be defined.

14.2 Microprobe Accessibility

To be defined.

14.3 Packaging of Bare Die

To be defined.

14.4 Logical-to-Physical Bit Map Equations

Not Applicable. The memories used in the system are IPs. The address mapping functions are transparent.

14.5 Top-Level Cell Names

To be defined.

14.6 Top-Level Power/Ground Port Names

To be defined.

14.7 Subcircuit Power/Ground Port Names

To be defined.

14.8 Bond Pad Coordinates

To be defined.

14.9 Name Correspondence of Top-Level Ports

To be defined.

15 Initialization Information

To initialization in Free Run Mode (the customer mode), the correspondent pin TEST_MODE must be set in low level. When system is power on, the power on reset analog block provide a reliable start up of the digital core. The circuit asserts the reset signal after a fixed delay triggered, that is grater then 250 ns due to memory initialization time. After this, Core module becomes to read ROM (internal or external, depending on EA pin) starting by address 0000h.

Core block provides reset signal to all other digital blocks after it initialization. Every block starts immediately after core and can be configured using Special Function Registers.

16 Application Information

The EMC08 can have three automotive functionalities:

a-) Automotive Power Train Solution, which provides two input engine sensors of MAP and Flywheel Tooth signals used to interact with the microcontroller to main control of the fuel and igniter driver engine system.

b-) Innovative Engine Start Security Key by RF transmitter based in digital wireless acting in the immobilizer car system.

c-) Innovative Engine Diagnostic Wireless Solution, to maintenance plan and car system analysis.

The figures below exemplifies an automotive system application and shows it block diagram.

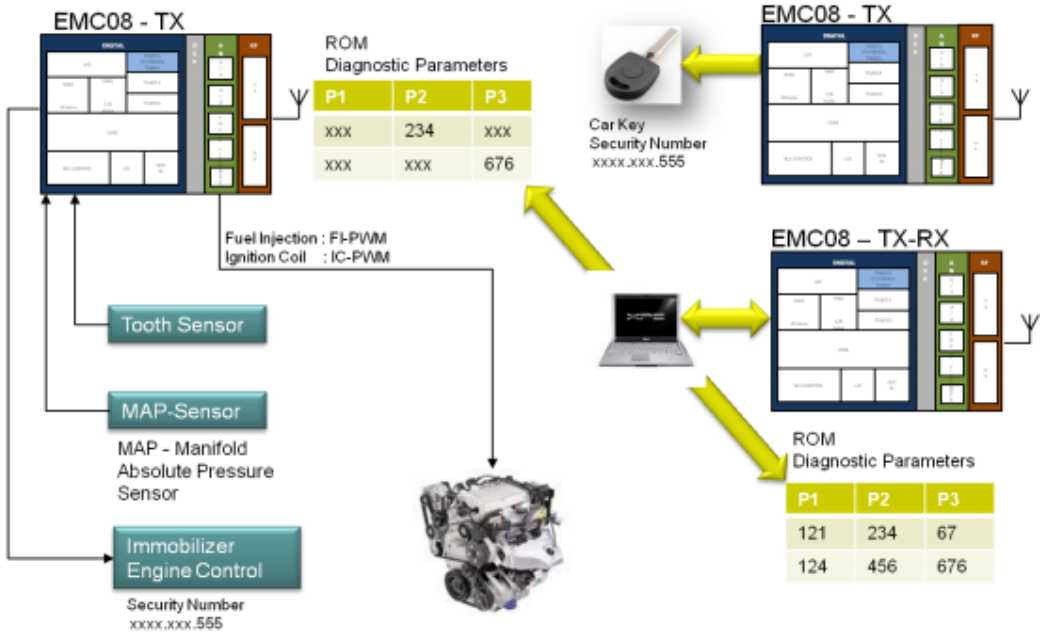
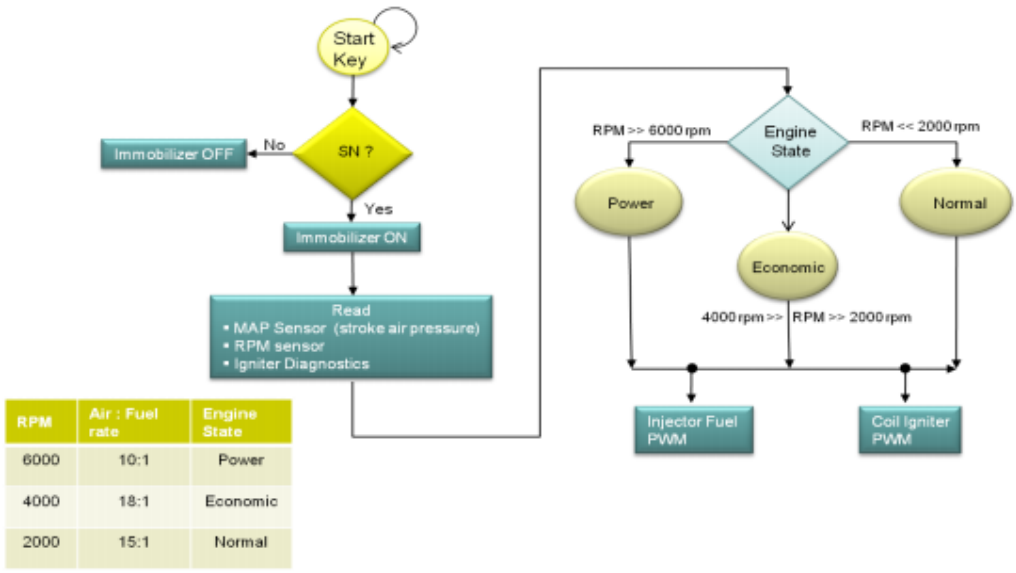


Figure 36 –EMC08 System Application



RPM	Air : Fuel rate	Engine State
6000	10:1	Power
4000	18:1	Economic
2000	15:1	Normal

Figure 37 –Automotive Application Diagram