What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	80C51
Core Size	8-Bit
Speed	40MHz
Connectivity	UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	20
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	2K x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-LCC (J-Lead)
Supplier Device Package	28-PLCC (11.51x11.51)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at89c5115-sisum

Pin Description

Pin Name	Type	Description
VSS	GND	Circuit ground
VCC		**Supply Voltage**
VAREF		Reference Voltage for ADC
VAVCC		Supply Voltage for ADC
VAGND		Reference Ground for ADC
P1.0:7	I/O	**Port 1:** Is an 8-bit bi-directional I/O port with internal pull-ups. Port 1 pins can be used for digital input/output or as analog inputs for the Analog Digital Converter (ADC). Port 1 pins that have 1's written to them are pulled high by the internal pull-up transistors and can be used as inputs in this state. As inputs, Port 1 pins that are being pulled low externally will be the source of current (I_{IL}, See section 'Electrical Characteristic') because of the internal pull-ups. Port 1 pins are assigned to be used as analog inputs via the ADCCF register (in this case the internal pull-ups are disconnected). As a secondary digital function, port 1 contains the Timer 2 external trigger and clock input; the PCA external clock input and the PCA module I/O. P1.0/AN0/T2 Analog input channel 0, External clock input for Timer/counter2. P1.1/AN1/T2EX Analog input channel 1, Trigger input for Timer/counter2. P1.2/AN2/ECI Analog input channel 2, PCA external clock input. P1.3/AN3/CEX0 Analog input channel 3, PCA module 0 Entry of input/PWM output. P1.4/AN4/CEX1 Analog input channel 4, PCA module 1 Entry of input/PWM output. P1.5/AN5 Analog input channel 5, P1.6/AN6 Analog input channel 6, P1.7/AN7 Analog input channel 7, It can drive CMOS inputs without external pull-ups.
P2.0:1	I/O	**Port 2:** Is an 2-bit bi-directional I/O port with internal pull-ups. Port 2 pins that have 1's written to them are pulled high by the internal pull-ups and can be used as inputs in this state. As inputs, Port 2 pins that are being pulled low externally will be a source of current (IIL, on the datasheet) because of the internal pull-ups. In the T89C5115 Port 2 can sink or source 5mA. It can drive CMOS inputs without external pull-ups.

SFR Mapping

Tables 3 through Table 11 show the Special Function Registers (SFRs) of the T89C5115.

Table 2. C51 Core SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
ACC	E0h	Accumulator								
B	F0h	B Register								
PSW	D0h	Program Status Word	CY	AC	F0	RS1	RS0	OV	F1	P
SP	81h	Stack Pointer								
DPL	82h	Data Pointer Low byte LSB of DPTR								
DPH	83h	Data Pointer High byte MSB of DPTR								

Table 3. I/O Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
P1	90h	Port 1								
P2	A0h	Port 2 (x2)								
P3	B0h	Port 3								
P4	C0h	Port 4 (x2)								

Table 4. Timers SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
TH0	8Ch	Timer/Counter 0 High byte								
TL0	8Ah	Timer/Counter 0 Low byte								
TH1	8Dh	Timer/Counter 1 High byte								
TL1	8Bh	Timer/Counter 1 Low byte								
TH2	CDh	Timer/Counter 2 High byte								
TL2	CCh	Timer/Counter 2 Low byte								
TCON	88h	Timer/Counter 0 and 1 control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
TMOD	89h	Timer/Counter 0 and 1 Modes	GATE1	C/T1#	M11	M01	GATE0	C/T0#	M10	M00

Table 4. Timers SFRs (Continued)

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
T2CON	C8h	Timer/Counter 2 control	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2#	CP/RL2#
T2MOD	C9h	Timer/Counter 2 Mode							T2OE	DCEN
RCAP2H	CBh	Timer/Counter 2 Reload/Capture High byte								
RCAP2L	CAh	Timer/Counter 2 Reload/Capture Low byte								
WDTRST	A6h	WatchDog Timer Reset								
WDTPRG	A7h	WatchDog Timer Program						S2	S1	S0

Table 5. Serial I/O Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
SCON	98h	Serial Control	FE/SM0	SM1	SM2	REN	TB8	RB8	TI	RI
SBUF	99h	Serial Data Buffer								
SADEN	B9h	Slave Address Mask								
SADDR	A9h	Slave Address								

Table 6. PCA SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
CCON	D8h	PCA Timer/Counter Control	CF	CR		CCF4	CCF3	CCF2	CCF1	CCF0
CMOD	D9h	PCA Timer/Counter Mode	CIDL					CPS1	CPS0	ECF
CL	E9h	PCA Timer/Counter Low byte								
CH	F9h	PCA Timer/Counter High byte								
CCAPM0 CCAPM1	DAh DBh	PCA Timer/Counter Mode 0 PCA Timer/Counter Mode 1		ECOM0 ECOM1	CAPP0 CAPP1	CAPN0 CAPN1	MAT0 MAT1	TOG0 TOG1	PWM0 PWM1	ECCF0 ECCF1
CCAP0H CCAP1H	FAh FBh	PCA Compare Capture Module 0 H PCA Compare Capture Module 1 H	CCAP0H7 CCAP1H7	CCAP0H6 CCAP1H6	CCAP0H5 CCAP1H5	CCAP0H4 CCAP1H4	CCAP0H3 CCAP1H3	CCAP0H2 CCAP1H2	CCAP0H1 CCAP1H1	CCAP0H0 CCAP1H0

Table 10. SFR Mapping

	0/8[1]	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
F8h	**IPL1** **xxxx xx0x**	CH 0000 0000	CCAP0H 0000 0000	CCAP1H 0000 0000					FFh
F0h	**B** **0000 0000**		ADCLK xxx0 0000	ADCON x000 0000	ADDL 0000 0000	ADDH 0000 0000	ADCF 0000 0000	IPH1 xxxx xx0x	F7h
E8h	**IEN1** **xxxx xx0x**	CL 0000 0000	CCAP0L 0000 0000	CCAP1L 0000 0000					EFh
E0h	**ACC** **0000 0000**								E7h
D8h	**CCON** **0000 0000**	CMOD 0xxx x000	CCAPM0 x000 0000	CCAPM1 x000 0000					DFh
D0h	**PSW** **0000 0000**	FCON 0000 0000	EECON xxxx xx00						D7h
C8h	**T2CON** 0000 0000	T2MOD xxxx xx00	RCAP2L 0000 0000	RCAP2H 0000 0000	TL2 0000 0000	TH2 0000 0000			CFh
C0h	**P4** **xxxx xx11**								C7h
B8h	**IPL0** **x000 0000**	SADEN 0000 0000							BFh
B0h	**P3** **1111 1111**							IPH0 x000 0000	B7h
A8h	**IEN0** **0000 0000**	SADDR 0000 0000							AFh
A0h	**P2** **xxxx xx11**		AUXR1[2] xxxx 00x0				WDTRST 1111 1111	WDTPRG xxxx x000	A7h
98h	**SCON** **0000 0000**	SBUF 0000 0000							9Fh
90h	**P1** **1111 1111**								97h
88h	**TCON** **0000 0000**	TMOD 0000 0000	TL0 0000 0000	TL1 0000 0000	TH0 0000 0000	TH1 0000 0000		CKCON 0000 0000	8Fh
80h		SP 0000 0111	DPL 0000 0000	DPH 0000 0000				PCON 00x1 0000	87h
	0/8[1]	1/9	2/A	3/B	4/C	5/D	6/E	7/F	

Reserved

Notes:
1. These registers are bit-addressable.
   Sixteen addresses in the SFR space are both byte-addressable and bit-addressable. The bit-addressable SFRs are those whose address ends in 0 and 8. The bit addresses, in this area, are 0x80 through to 0xFF.
2. AUXR1 bit ENBOOT is initialized with the content of the BLJB bit inverted.

Table 14. Pin Conditions in Special Operating Modes

Mode	Port 1	Port 2	Port 3	Port 4
Reset	High	High	High	High
Idle (internal code)	Data	Data	Data	Data
Idle (external code)	Data	Data	Data	Data
Power-Down(internal code)	Data	Data	Data	Data
Power-Down (external code)	Data	Data	Data	Data

Registers

Table 15. PCON Register
PCON (S:87h)
Power Control Register

7	6	5	4	3	2	1	0
SMOD1	SMOD0	-	POF	GF1	GF0	PD	IDL

Bit Number	Bit Mnemonic	Description
7	SMOD1	**Serial port Mode bit 1** Set to select double baud rate in mode 1, 2 or 3.
6	SMOD0	**Serial port Mode bit 0** Clear to select SM0 bit in SCON register. Set to select FE bit in SCON register.
5	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.
4	POF	**Power-off Flag** Clear to recognize next reset type. Set by hardware when V_{CC} rises from 0 to its nominal voltage. Can also be set by software.
3	GF1	**General purpose Flag** Cleared by user for general purpose usage. Set by user for general purpose usage.
2	GF0	**General purpose Flag** Cleared by user for general purpose usage. Set by user for general purpose usage.
1	PD	**Power-down Mode bit** Cleared by hardware when reset occurs. Set to enter power-down mode.
0	IDL	**Idle Mode bit** Clear by hardware when interrupt or reset occurs. Set to enter idle mode.

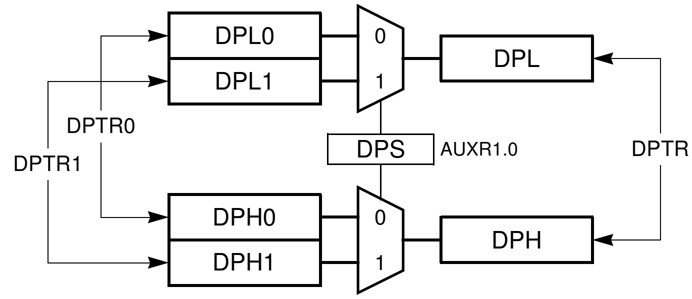
Reset Value = 00X1 0000b
Not bit addressable

Dual Data Pointer

Description

The T89C5115 implements a second data pointer for speeding up code execution and reducing code size in case of intensive usage of external memory accesses.
DPTR0 and DPTR1 are Seen by the CPU as DPTR and are accessed using the SFR addresses 83h and 84h that are the DPH and DPL addresses. The DPS bit in AUXR1 register (See Figure 18) is used to select whether DPTR is the data pointer 0 or the data pointer 1 (See Figure 11).

Figure 11. Dual Data Pointer Implementation



Application

Software can take advantage of the additional data pointers to both increase speed and reduce code size, for example, block operations (copy, compare…) are well served by using one data pointer as a “source” pointer and the other one as a “destination” pointer. Hereafter is an example of block move implementation using the two pointers and coded in assembler. The latest C compiler takes also advantage of this feature by providing enhanced algorithm libraries.
The INC instruction is a short (2 Bytes) and fast (6 machine cycle) way to manipulate the DPS bit in the AUXR1 register. However, note that the INC instruction does not directly force the DPS bit to a particular state, but simply toggles it. In simple routines, such as the block move example, only the fact that DPS is toggled in the proper sequence matters, not its actual value. In other words, the block move routine works the same whether DPS is 0 or 1 on entry.

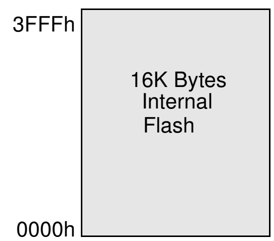
```
; ASCII block move using dual data pointers
; Modifies DPTR0, DPTR1, A and PSW
; Ends when encountering NULL character
; Note: DPS exits opposite to the entry state unless an extra INC AUXR1 is
added

AUXR1EQU0A2h

move:movDPTR,#SOURCE ; address of SOURCE
  incAUXR1 ; switch data pointers
  movDPTR,#DEST ; address of DEST
mv_loop:incAUXR1; switch data pointers
  movxA,@DPTR; get a byte from SOURCE
  incDPTR; increment SOURCE address
  incAUXR1; switch data pointers
  movx@DPTR,A; write the byte to DEST
  incDPTR; increment DEST address
  jnzmv_loop; check for NULL terminator
end_move:
```

Program/Code Memory

The T89C5115 implement 16K Bytes of on-chip program/code memory.

The Flash memory increases EPROM and ROM functionality by in-circuit electrical erasure and programming. Thanks to the internal charge pump, the high voltage needed for programming or erasing Flash cells is generated on-chip using the standard V_{DD} voltage. Thus, the Flash memory can be programmed using only one voltage and allows In-System Programming (ISP). Hardware programming mode is also available using specific programming tool.

Figure 12. Program/Code Memory Organization



Flash Memory Architecture

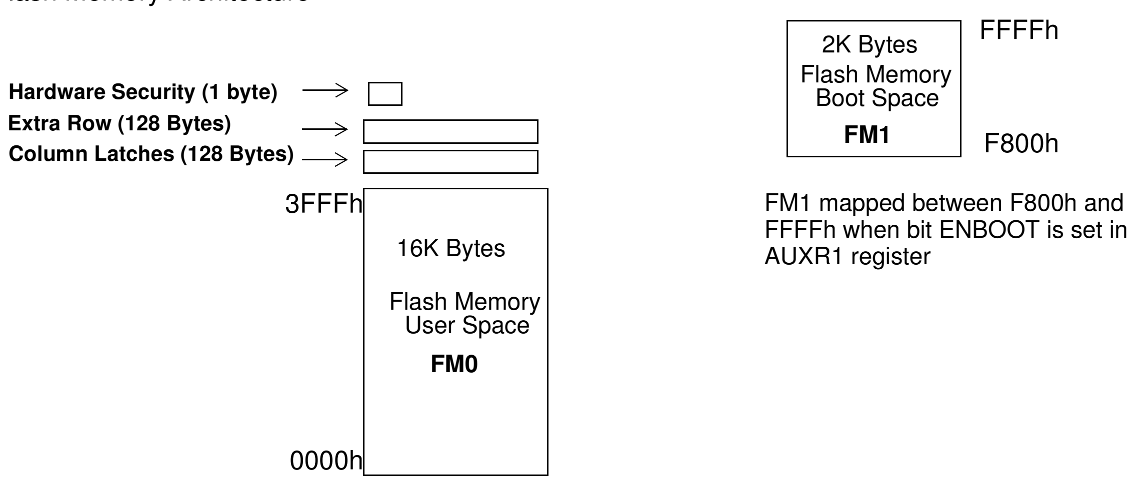
T89C5115 features two on-chip Flash memories:

- Flash memory FM0:
  containing 16K Bytes of program memory (user space) organized into 128 bytes pages,
- Flash memory FM1:
  2K Bytes for boot loader and Application Programming Interfaces (API).

The FM0 can be program by both parallel programming and Serial ISP whereas FM1 supports only parallel programming by programmers. The ISP mode is detailed in the 'In-System Programming' section.

All Read/Write access operations on Flash memory by user application are managed by a set of API described in the 'In-System Programming' section.

Figure 13. Flash Memory Architecture



FM0 Memory Architecture

The Flash memory is made up of 4 blocks (See Figure 13):

1. The memory array (user space) 16K Bytes
2. The Extra Row
3. The Hardware security bits
4. The column latch registers

User Space

This space is composed of a 16K Bytes Flash memory organized in 128 pages of 128 Bytes. It contains the user's application code.

Extra Row (XRow)

This row is a part of FM0 and has a size of 128 Bytes. The extra row may contain information for boot loader usage.

Hardware Security Byte

The Hardware security Byte space is a part of FM0 and has a size of 1 byte.
The 4 MSB can be read/written by software, the 4 LSB can only be read by software and written by hardware in parallel mode.

Column Latches

The column latches, also part of FM0, have a size of full page (128 Bytes).
The column latches are the entrance buffers of the three previous memory locations (user array, XROW and Hardware security byte).

Cross Flash Memory Access Description

The FM0 memory can be programmed as describe on Table 20. Programming FM0 from FM0 is impossible.

The FM1 memory can be program only by parallel programming.

Table 20 show all software Flash access allowed.

Table 20. Cross Flash Memory Access

		Action	FM0 (user Flash)	FM1 (boot Flash)
Code executing from	FM0 (user Flash)	Read	ok	-
		Load column latch	ok	-
		Write	-	-
	FM1 (boot Flash)	Read	ok	ok
		Load column latch	ok	-
		Write	ok	-

Sharing Instructions

Table 26. Instructions shared

Action	RAM	ERAM	EEPROM DATA	Boot FLASH	FM0	Hardware Byte	XROW
Read	MOV	MOVX	MOVX	MOVC	MOVC	MOVC	MOVC
Write	MOV	MOVX	MOVX	-	by cl	by cl	by cl

Note: by cl : using Column Latch

Table 27. Read MOVX A, @DPTR

EEE bit in EECON Register	FPS in FCON Register	ENBOOT	ERAM	EEPROM DATA	Flash Column Latch
0	0	X	OK		
0	1	X	OK		
1	0	X		OK	
1	1	X	OK		

Table 28. Write MOVX @DPTR,A

EEE bit in EECON Register	FPS bit in FCON Register	ENBOOT	ERAM	EEPROM Data	Flash Column Latch
0	0	X	OK		
0	1	X			OK
1	0	X		OK	
1	1	X			OK

Here is an example of how to use given addresses to address different slaves:

```
Slave A:SADDR1111 0001b
    SADEN1111 1010b
    Given1111 0X0Xb

Slave B:SADDR1111 0011b
    SADEN1111 1001b
    Given1111 0XX1b

Slave C:SADDR1111 0011b
    SADEN1111 1101b
    Given1111 00X1b
```

The SADEN byte is selected so that each slave may be addressed separately.
For slave A, bit 0 (the LSB) is a don't-care bit; for slaves B and C, bit 0 is a 1. To communicate with slave A only, the master must send an address where bit 0 is clear (e.g. `1111 0000b`).
For slave A, bit 1 is a 0; for slaves B and C, bit 1 is a don't care bit. To communicate with slaves A and B, but not slave C, the master must send an address with bits 0 and 1 both set (e.g. `1111 0011b`).
To communicate with slaves A, B and C, the master must send an address with bit 0 set, bit 1 clear, and bit 2 clear (e.g. `1111 0001b`).

Broadcast Address

A broadcast address is formed from the logical OR of the SADDR and SADEN registers with zeros defined as don't-care bits, e.g.:

```
    SADDR 0101 0110b
    SADEN 1111 1100b
    SADDR OR SADEN1111 111Xb
```

The use of don't-care bits provides flexibility in defining the broadcast address, however in most applications, a broadcast address is FFh. The following is an example of using broadcast addresses:

```
Slave A:SADDR1111 0001b
    SADEN1111 1010b
    Given1111 1X11b,

Slave B:SADDR1111 0011b
    SADEN1111 1001b
    Given1111 1X11B,

Slave C:SADDR=1111 0010b
    SADEN1111 1101b
    Given1111 1111b
```

For slaves A and B, bit 2 is a don't care bit; for slave C, bit 2 is set. To communicate with all of the slaves, the master must send an address FFh. To communicate with slaves A and B, but not slave C, the master can send and address FBh.

Timers/Counters

The T89C5115 implements two general-purpose, 16-bit Timers/Counters. Such are identified as Timer 0 and Timer 1, and can be independently configured to operate in a variety of modes as a Timer or an event Counter. When operating as a Timer, the Timer/Counter runs for a programmed length of time, then issues an interrupt request. When operating as a Counter, the Timer/Counter counts negative transitions on an external pin. After a preset number of counts, the Counter issues an interrupt request.
The various operating modes of each Timer/Counter are described in the following sections.

Timer/Counter Operations

A basic operation is Timer registers THx and TLx (x = 0, 1) connected in cascade to form a 16-bit Timer. Setting the run control bit (TRx) in TCON register (See Figure 37) turns the Timer on by allowing the selected input to increment TLx. When TLx overflows it increments THx; when THx overflows it sets the Timer overflow flag (TFx) in TCON register. Setting the TRx does not clear the THx and TLx Timer registers. Timer registers can be accessed to obtain the current count or to enter preset values. They can be read at any time but TRx bit must be cleared to preset their values, otherwise the behavior of the Timer/Counter is unpredictable.

The C/Tx# control bit selects Timer operation or Counter operation by selecting the divided-down peripheral clock or external pin Tx as the source for the counted signal. TRx bit must be cleared when changing the mode of operation, otherwise the behavior of the Timer/Counter is unpredictable.
For Timer operation (C/Tx# = 0), the Timer register counts the divided-down peripheral clock. The Timer register is incremented once every peripheral cycle (6 peripheral clock periods). The Timer clock rate is $f_{PER}/6$, i.e. $f_{OSC}/12$ in standard mode or $f_{OSC}/6$ in X2 Mode.

For Counter operation (C/Tx# = 1), the Timer register counts the negative transitions on the Tx external input pin. The external input is sampled every peripheral cycles. When the sample is high in one cycle and low in the next one, the Counter is incremented. Since it takes 2 cycles (12 peripheral clock periods) to recognize a negative transition, the maximum count rate is $f_{PER}/12$, i.e. $f_{OSC}/24$ in standard mode or $f_{OSC}/12$ in X2 Mode. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one full peripheral cycle.

Timer 0

Timer 0 functions as either a Timer or event Counter in four modes of operation. Figure 24 through Figure 27 show the logical configuration of each mode.

Timer 0 is controlled by the four lower bits of TMOD register (See Figure 38) and bits 0, 1, 4 and 5 of TCON register (See Figure 37). TMOD register selects the method of Timer gating (GATE0), Timer or Counter operation (T/C0#) and mode of operation (M10 and M00). TCON register provides Timer 0 control functions: overflow flag (TF0), run control bit (TR0), interrupt flag (IE0) and interrupt type control bit (IT0).
For normal Timer operation (GATE0 = 0), setting TR0 allows TL0 to be incremented by the selected input. Setting GATE0 and TR0 allows external pin INT0# to control Timer operation.

Timer 0 overflow (count rolls over from all 1s to all 0s) sets TF0 flag generating an interrupt request.

It is important to stop Timer/Counter before changing mode.

- For normal Timer operation (GATE1= 0), setting TR1 allows TL1 to be incremented by the selected input. Setting GATE1 and TR1 allows external pin INT1# to control Timer operation.
- Timer 1 overflow (count rolls over from all 1s to all 0s) sets the TF1 flag generating an interrupt request.
- When Timer 0 is in mode 3, it uses Timer 1's overflow flag (TF1) and run control bit (TR1). For this situation, use Timer 1 only for applications that do not require an interrupt (such as a Baud Rate Generator for the Serial Port) and switch Timer 1 in and out of mode 3 to turn it off and on.
- It is important to stop Timer/Counter before changing mode.

Mode 0 (13-bit Timer)

Mode 0 configures Timer 1 as a 13-bit Timer, which is set up as an 8-bit Timer (TH1 register) with a modulo-32 prescaler implemented with the lower 5 bits of the TL1 register (See Figure 24). The upper 3 bits of TL1 register are ignored. Prescaler overflow increments TH1 register.

Mode 1 (16-bit Timer)

Mode 1 configures Timer 1 as a 16-bit Timer with TH1 and TL1 registers connected in cascade (See Figure 25). The selected input increments TL1 register.

Mode 2 (8-bit Timer with Auto-Reload)

Mode 2 configures Timer 1 as an 8-bit Timer (TL1 register) with automatic reload from TH1 register on overflow (See Figure 26). TL1 overflow sets TF1 flag in TCON register and reloads TL1 with the contents of TH1, which is preset by software. The reload leaves TH1 unchanged.

Mode 3 (Halt)

Placing Timer 1 in mode 3 causes it to halt and hold its count. This can be used to halt Timer 1 when TR1 run control bit is not available i.e. when Timer 0 is in mode 3.

Interrupt

Each Timer handles one interrupt source that is the timer overflow flag TF0 or TF1. This flag is set every time an overflow occurs. Flags are cleared when vectoring to the Timer interrupt routine. Interrupts are enabled by setting ETx bit in IEN0 register. This assumes interrupts are globally enabled by setting EA bit in IEN0 register.

Figure 28. Timer Interrupt System

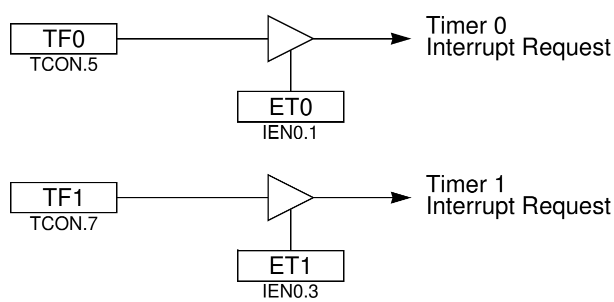


Table 40. TL0 Register
TL0 (S:8Ah)
Timer 0 Low Byte Register

7	6	5	4	3	2	1	0

Bit Number	Bit Mnemonic	Description
7:0		**Low Byte of Timer 0**

Reset Value = 0000 0000b

Table 41. TH1 Register
TH1 (S:8Dh)
Timer 1 High Byte Register

7	6	5	4	3	2	1	0

Bit Number	Bit Mnemonic	Description
7:0		**High Byte of Timer 1**

Reset Value = 0000 0000b

Table 42. TL1 Register
TL1 (S:8Bh)
Timer 1 Low Byte Register

7	6	5	4	3	2	1	0

Bit Number	Bit Mnemonic	Description
7:0		**Low Byte of Timer 1**

Reset Value = 0000 0000b

Table 46. TL2 Register
TL2 (S:CCh)
Timer 2 Low Byte Register

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

Bit Number	Bit Mnemonic	Description
7 - 0		Low Byte of Timer 2

Reset Value = 0000 0000b
Not bit addressable

Table 47. RCAP2H Register
RCAP2H (S:CBh)
Timer 2 Reload/Capture High Byte Register

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

Bit Number	Bit Mnemonic	Description
7 - 0		High Byte of Timer 2 Reload/Capture.

Reset Value = 0000 0000b
Not bit addressable

Table 48. RCAP2L Register
RCAP2L (S:CAh) Timer 2 Reload/Capture Low Byte Register

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

Bit Number	Bit Mnemonic	Description
7 - 0		Low Byte of Timer 2 Reload/Capture.

Reset Value = 0000 0000b
Not bit addressable

Watchdog Programming

The three lower bits (S0, S1, S2) located into WDTPRG register permit to program the WDT duration.

Table 49. Machine Cycle Count

S2	S1	S0	Machine Cycle Count
0	0	0	2^{14} - 1
0	0	1	2^{15} - 1
0	1	0	2^{16} - 1
0	1	1	2^{17} - 1
1	0	0	2^{18} - 1
1	0	1	2^{19} - 1
1	1	0	2^{20} - 1
1	1	1	2^{21} - 1

To compute WD Timeout, the following formula is applied:

$$FTime-Out = \frac{F_{osc}}{6 \times 2^{WDX2 \wedge X2}(2^{14} \times 2^{Svalue})}$$

Note: Svalue represents the decimal value of (S2 S1 S0)

Find Hereafter computed Timeout values for f_{OSC}XTAL = 12 MHz in X1 mode

Table 50. Timeout Computation

S2	S1	S0	f_{OSC}=12 MHz	f_{OSC}=16MHz	f_{OSC}=20 MHz
0	0	0	16.38 ms	12.28 ms	9.82 ms
0	0	1	32.77 ms	24.57 ms	19.66 ms
0	1	0	65.54 ms	49.14 ms	39.32 ms
0	1	1	131.07 ms	98.28 ms	78.64 ms
1	0	0	262.14 ms	196.56 ms	157.28 ms
1	0	1	524.29 ms	393.12 ms	314.56 ms
1	1	0	1.05 s	786.24 ms	629.12 ms
1	1	1	2.10 s	1.57 s	1.25 s

Programmable Counter Array (PCA)

The PCA provides more timing capabilities with less CPU intervention than the standard timer/counters. Its advantages include reduced software overhead and improved accuracy. The PCA consists of a dedicated timer/counter which serves as the time base for an array of two compare/capture modules. Its clock input can be programmed to count any of the following signals:

- PCA clock frequency/6 (See “clock” section)
- PCA clock frequency/2
- Timer 0 overflow
- External input on ECI (P1.2)

Each compare/capture modules can be programmed in any one of the following modes:

- Rising and/or falling edge capture,
- Software timer
- High-speed output
- Pulse width modulator

When the compare/capture modules are programmed in capture mode, software timer, or high speed output mode, an interrupt can be generated when the module executes its function. Both modules and the PCA timer overflow share one interrupt vector.

The PCA timer/counter and compare/capture modules share Port 1 for external I/Os. These pins are listed below. If the port is not used for the PCA, it can still be used for standard I/O.

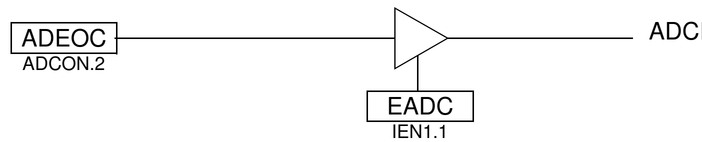
PCA Component	External I/O Pin
16-bit Counter	P1.2/ECI
16-bit Module 0	P1.3/CEX0
16-bit Module 1	P1.4/CEX1

PCA Timer

The PCA timer is a common time base for both modules (See Figure 9). The timer count source is determined from the CPS1 and CPS0 bits in the **CMOD SFR** (See Table 8) and can be programmed to run at:

- 1/6 the PCA clock frequency.
- 1/2 the PCA clock frequency.
- The Timer 0 overflow.
- The input on the ECI pin (P1.2).

Figure 41. ADC interrupt structure



Routine Examples

```
1. Configure P1.2 and P1.3 in ADC channels
// configure channel P1.2 and P1.3 for ADC
 ADCF = 0Ch

// Enable the ADC
 ADCON = 20h
2. Start a standard conversion
// The variable 'channel' contains the channel to convert
// The variable 'value_converted' is an unsigned int
// Clear the field SCH[2:0]
 ADCON &= F8h
// Select channel
 ADCON |= channel
// Start conversion in standard mode
 ADCON |= 08h
// Wait flag End of conversion
 while((ADCON & 01h)!= 01h)
// Clear the End of conversion flag
 ADCON &= EFh
// read the value
 value_converted = (ADDH << 2)+(ADDL)

3. Start a precision conversion (need interrupt ADC)
// The variable 'channel' contains the channel to convert
// Enable ADC
 EADC = 1
// clear the field SCH[2:0]
 ADCON &= F8h
// Select the channel
 ADCON |= channel
// Start conversion in precision mode
 ADCON |= 48h
```

Note: To enable the ADC interrupt: EA = 1

Each of the interrupt sources can be individually enabled or disabled by setting or clearing a bit in the Interrupt Enable register. This register also contains a global disable bit which must be cleared to disable all the interrupts at the same time.

Each interrupt source can also be individually programmed to one of four priority levels by setting or clearing a bit in the Interrupt Priority registers. The Table below shows the bit values and priority levels associated with each combination.

Table 66. Priority Level bit Values

IPH.x	IPL.x	Interrupt Level Priority
0	0	0 (Lowest)
0	1	1
1	0	2
1	1	3 (Highest)

A low-priority interrupt can be interrupted by a high priority interrupt but not by another low-priority interrupt. A high-priority interrupt cannot be interrupted by any other interrupt source.

If two interrupt requests of different priority levels are received simultaneously, the request of the higher priority level is serviced. If interrupt 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, See Table 67.

Table 67. Interrupt Priority Within Level

Interrupt Name	Interrupt Address Vector	Priority Number
External interrupt (INT0)	0003h	1
Timer0 (TF0)	000Bh	2
External interrupt (INT1)	0013h	3
Timer 1 (TF1)	001Bh	4
PCA (CF or CCFn)	0033h	5
UART (RI or TI)	0023h	6
Timer 2 (TF2)	002Bh	7
ADC (ADCI)	0043h	9

Registers

Figure 43. IEN0 Register
IEN0 (S:A8h)
Interrupt Enable Register

7	6	5	4	3	2	1	0
EA	EC	ET2	ES	ET1	EX1	ET0	EX0

Bit Number	Bit Mnemonic	Description
7	EA	**Enable All Interrupt bit** Clear to disable all interrupts. Set to enable all interrupts. If EA=1, each interrupt source is individually enabled or disabled by setting or clearing its interrupt enable bit.
6	EC	**PCA Interrupt Enable** Clear to disable the PCA interrupt. Set to enable the PCA interrupt.
5	ET2	**Timer 2 Overflow Interrupt Enable bit** Clear to disable Timer 2 overflow interrupt. Set to enable Timer 2 overflow interrupt.
4	ES	**Serial port Enable bit** Clear to disable serial port interrupt. Set to enable serial port interrupt.
3	ET1	**Timer 1 Overflow Interrupt Enable bit** Clear to disable timer 1 overflow interrupt. Set to enable timer 1 overflow interrupt.
2	EX1	**External Interrupt 1 Enable bit** Clear to disable external interrupt 1. Set to enable external interrupt 1.
1	ET0	**Timer 0 Overflow Interrupt Enable bit** Clear to disable timer 0 overflow interrupt. Set to enable timer 0 overflow interrupt.
0	EX0	**External Interrupt 0 Enable bit** Clear to disable external interrupt 0. Set to enable external interrupt 0.

Reset Value = 0000 0000b
bit addressable