4 8032 Differences

The 8051 is the very basic micro-controller. In this chapter we present one of the first improved versions or variants, namely the 8032/8052 micro-controller, with an enhanced internal memory and an additional timer. An explanation of the new special function registers associated with the new internal peripheral is also given.

4.1 8032 Extras

The 8032 microcontroller is the 8051's "big brother." It is a slightly more powerful microcontroller, sporting a number of additional features which the developer may make use of.

Hex Byte Address	Notes	Notes	Hex Byte Address
FF Upper 128 bytes 80	(8032 ONLY) Accessible by Indirect Addressing only	SFR area Acces-sible by Direct Address-ing only	FF 80
7F Lower 128 bytes 00	Accessible by Direct and Indi-rect Ad-dressing.		

Table 4-1 8032 Total Internal RAM organisation

- 256 bytes of Internal RAM (compared to 128 in the standard 8051). The lower 128 bytes are accessible using either Direct or Indirect addressing modes. The additional upper 128 bytes can only be accessed using the Indirect addressing mode.
- A third 16-bit timer (Timer 2), capable of a number of new operating modes, interrupts and 16-bit reloads.
- The serial port can now make use of either Timer 1 or Timer 2 to generate the baud rates.
- Additional SFRs to support the functionality offered by the third timer. These SFRs still reside in the 80h-FFh area accessible only by Direct Addressing to differentiate them from the Indirectly addressable internal RAM used for program stack and/or variables.

Table 4-1 shows the internal memory differences that there are between the 8051 and 8032. The remainder of this chapter will explain these additional features offered by the 8032, and how they are used within user programs.

4.2 256 Bytes of Internal RAM

The standard 8051 microcontroller contains 128 bytes of Internal RAM that are available to the developer as working memory for variables and/or for the operating stack. Instructions that refer to internal ram addresses in the range of 00h through 7Fh refer to the basic 8051.

Addresses which are accessible using direct addressing, in the range of 80h through FFh refer to Special Function Registers (SFRs).

Although the 8032 has 256 bytes of Internal RAM, the above mentioned method of referencing them remains true. Using Direct Addressing, any address between 00h and 7Fh refers to Internal RAM whereas any address in the range of 80h through FFh refers to SFRs.

The 8032's additional Internal RAM may only be accessed using Indirect Addressing. Indirect addressing always refers to Internal RAM and never to an SFR.

Thus, to read the value contained in Internal RAM address 90h, we would need to code something along the following lines:

```
MOV R0, #90h        ;Set the indirect address to 90h
MOV A, @R0          ;Read the contents of Internal RAM pointed to by R0
```

The above code first assigns the value 90h to the register R0. It subsequently reads, indirectly, the contents of the address contained in (pointed by) R0 (90h). Thus, after these two instructions have executed, the Accumulator will contain the value of Internal RAM address 90h.

Hex Byte Address	Hex Bit Address								Notes
FF 80	Additional Indirectly Addressable General Purpose RAM								Can be used as a STACK Area by loading SP with 7FH or higher
7F 30	Directly and Indirectly Addressable General Purpose RAM								
2F	7F	7E	7D	7C	7B	7A	79	78	Bit Addressable Section
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	
1F 18	Register Bank 3 (R0 – R7)								Bank is selected using RS0 and RS1 in the PSW register. See SFRs.
17 10	Register Bank 2 (R0 – R7)								
0F 08	Register Bank 1 (R0 – R7)								
07 00	Register Bank 0 (R0 – R7)								

Table 4-2 8032 Internal RAM organisation

It is very important to understand that the above code just mentioned, is not the same as the following:

```
MOV A, 90h ;Reads the contents of SFR 90h (P1)
```

This instruction uses direct addressing; recall that direct addressing reads Internal RAM when the address is in the range of 00h through 7Fh, and reads an SFR when the address is in the range of 80h through FFh. Thus in the case of this second example, the move instruction reads the value of SFR 90h, which happens to be P1 (I/O Port 1).

The importance of using the correct addressing mode cannot be over-emphasised. It should however be noted here, that when using a compiler to compile a C source code program into machine code, the compiler automatically would use the correct addressing form. The compiler would know where the variable or SFR is located and could therefore decide which type of addressing mode is required to access that variable or SFR.

4.3 Additional Timer 2

An important addition for the 8032 is the availability of a third timer, referred to as Timer 2. We shall now deal with the SFRs connected with this timer, as well as the various modes of operation for this extra timer.

4.3.1 New SFRs for 8032's third timer (T2)

In addition to the 8051's standard 21 SFRs, the 8032 adds an additional 5 SFRs related to the 8032's third timer as shown shaded in Table 4-3a. All of the original 8051 SFRs shown in Tables 4-3a and 4-3b function exactly as they do in the 8051 – the 8032 simply adds new SFRs, it doesn't change the definition of the standard SFRs. The five new SFRs are in the range of C8h to CDh (SFR C9h is not defined), plus some additional bits shown shaded.

Note that the TL2/TH2 register pair and the RCAP2L/RCAP2H register pair occupy consecutive memory addresses, low byte first, contrary to the registers available for Timer 0 and Timer 1. This means that in KEIL C, we can load the whole 16-bit Timer 2 counter registers TL2 and TH2 by using the SFR16 data type.

Hex Byte Address	Hex Bit Address								Symbol
FF – F9	Not implemented on chip								-
* F8 *	Not implemented on chip								-
F7 – F1	Not implemented on chip								-
* F0 *	F7	F6	F5	F4	F3	F2	F1	F0	B
EF – E9	Not implemented on chip								-
* E8 *	Not implemented on chip								-
E7 – E1	Not implemented on chip								-
* E0 *	E7	E6	E5	E4	E3	E2	E1	E0	ACC
DF – D9	Not implemented on chip								-
* D8 *	Not implemented on chip								-
D7 – D1	Not implemented on chip								-
* D0 *	D7	D6	D5	D4	D3	D2	D1	D0	PSW
CF – CE	Not implemented on chip								-
CD									TH2
CC									TL2
CB									RCAP2H
CA									RCAP2L
C9	Not implemented on chip								-
C8	CF	CE	CD	CC	CB	CA	C9	C8	T2CON
C7 – C1	Not implemented on chip								-
* C0 *	Not implemented on chip								-
BF – B9	Not implemented on chip								-
* B8 *	-	-	BD	BC	BB	BA	B9	B8	IP
B7 – B1	Not implemented on chip								-
* B0 *	B7	B6	B5	B4	B3	B2	B1	B0	P3
AF – A9	Not implemented on chip								-
* A8 *	AF	-	AD	AC	AB	AA	A9	A8	IE
A7 – A1	Not implemented on chip								-
* A0 *	A7	A6	A5	A4	A3	A2	A1	A0	P2

Table 4-3a 8032 Special Function Registers (SFRs)-DIRECT addressing ONLY

Hex Byte Address	Hex Bit Address								Symbol
9F – 9A	Not implemented on chip								-
99									SBUF
* 98 *	9F	9E	9D	9C	9B	9A	99	98	SCON
97 – 91	Not implemented on chip								-
* 90 *	97	96	95	94	93	92	91	90	P1
8F –8E	Not implemented on chip								-
8D									TH1
8C									TH0
8B									TL1
8A									TL0
89									TMOD
* 88 *	8F	8E	8D	8C	8B	8A	89	88	TCON
87									PCON
86 – 84	Not implemented on chip								-
83									DPH
82									DPL
81									SP
* 80 *	87	86	85	84	83	82	81	80	P0

Table 4-3b 8032 Special Function Registers (SFRs)-DIRECT addressing ONLY

The procedure would be as follows:

- We first declare a variable of type SFR16, say using

  SFR16 T2REG = 0xCC;

- We then simply write

  T2REG = 0x1234; //This would load 34H in TL2 and 12H in TH2.

4.3.2 T2CON SFR (C8H)

Bit-addressable				
Bit	**Name**	**Alternate Names**	**Bit Hex Address**	**Explanation of Function**
7	TF2	T2CON.7	CF	Timer 2 overflow. This bit is set when T2 overflows. When T2 interrupt is enabled, this bit will cause the interrupt to be triggered. This bit will not be set if either TCLK or RCLK bits are set
6	EXF2	T2CON.6	CE	Timer 2 External flag. Set by a reload or capture caused b a 1-0 transition on T2EX (P1.1), but only when EXEN2 is set. When T2 interrupt is enabled, this bit will also trigger the interrupt.
5	RCLK	T2CON.5	CD	Timer 2 Receiver Clock. When this bit is set, Timer 2 will be used to determine the serial port receive baud rate. When cleared, Timer 1 will be used as the baud rate generator.
4	TCLK	T2CON.4	CC	Timer 2 Transmitter Clock. When this bit is set, Timer 2 will be used to determine the serial port transmitter baud rate. When cleared, Timer 1 will be used as the baud rate generator.
3	EXEN2	T2CON.3	CB	Timer 2 External enable. When set, a 1-0 transition on T2EX (P1.1) will cause a capture or a reload to occur.
2	TR2	T2CON.2	CA	Timer 2 run. When set, timer 2 will start. Timer 2 will stop when this bit is cleared.
1	C/T2	T2CON.1	C9	Timer 2 Counter/Interval timer. If cleared, Timer 2 is an interval counter. If set, Timer 2 is incremented by 1-0 transitions on T2 (P1.0).
0	CP/RL2C	T2CON.0	C8	Timer 2 Capture/Reload. If cleared, auto reload occurs on timer 2 overflow, or T2EX 1-0 transition if EXEN2 is set. If set, a capture will occur on a 1-0 transition of T2EX, if EXEN2 is set.

Table 4-4 T2CON (C8H) SFR

The operation of Timer 2 (T2) is controlled almost entirely by the T2CON SFR shown in Table 4-4, at address C8h. Note that since this SFR has an address which is divisible by 8, then it is Bit-addressable.

4.3.3 Timer 2 as a baud-rate generator

Timer 2 may be used as a baud rate generator. This is accomplished by setting either RCLK (T2CON.5) or TCLK (T2CON.4). With the standard 8051, Timer 1 is the only timer which may be used to determine the baud rate of the serial port. Additionally for the standard 8051 the receive and transmit baud rate must be the same.

With the 8032, however, we may configure the serial port to receive at one baud rate and transmit at another baud rate. For example, if RCLK is set and TCLK is cleared, serial data will be received at the baud rate determined by Timer 2 whereas the baud rate of transmitted data will be determined by Timer 1.

Determining the auto-reload values of Timer 1 for a specific baud rate was discussed in section 2.12.2. Timer 2 can similarly be programmed, the only difference is that in the case of Timer 2, the auto-reload value is placed in RCAP2H and RCAP2L, and the value is a 16-bit value rather than an 8-bit value.

The baud rates (in serial modes 1 and 3) are determined by Timer 2's overflow rate as follows:

$$\text{Baud Rate} = (\text{Timer 2 Overflow Rate})/ 16$$

The Timer can be configured for either timer or counter operation. The timer operation is a little different for Timer 2 when it is being used as a baud rate generator. Normally, as a timer it would increment every machine cycle (oscillator frequency/12). However, when being used as a baud rate generator, it increments at every state time (oscillator frequency/2) and the equations for determining the variable baud rate (serial modes 1 or 3), using Timer 2 become:

$$\text{Baud Rate} = (\text{Osc. Frequency})/(32 [65536 – (\text{RCAP2H,RCAP2L})]) \quad \textbf{… Equation 4-1}$$

or

$$(\text{RCAP2H,RCAP2L}) = [2097152 – (\text{Osc. Freq.})/(\text{Baud Rate})] / 32 \quad \textbf{… Equation 4-2}$$

Where (RCAP2H,RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

Thus to get a baudrate of 345600 baud with an 11.0592 MHz clock, using equation 4-2 we would need to load (RCAP2H,RCAP2L) with 65535. Thus the intial Timer 2 registers TH2 and TL2 as well as the reload registers RCAP2H and RCAP2L would all be loaded with 255 or FFH. An example using Timer 2 as the baud rate generator is given in Appendix F.

Note that when Timer 2 is used as a baud rate generator (having either TCLK or RCLK set), the Timer 2 Overflow Flag (TF2) will not be set, therefore the Timer 2 interrupt does not have to be disabled. Thus when Timer 2 is being used as a baud rate generator, T2EX can still be used as an extra external interrupt if required.

4.3.4 Timer 2 in auto-reload mode

The first mode in which Timer 2 may be used is Auto-Reload. The auto-reload mode functions just like Timer 0 and Timer 1 in auto-reload mode, except that the Timer 2 auto-reload mode performs a full 16-bit reload (recall that Timer 0 and Timer 1 only have 8-bit reload capability). When a reload occurs, the value of TH2 will be reloaded with the value contained in RCAP2H and the value of TL2 will be reloaded with the value contained in RCAP2L.

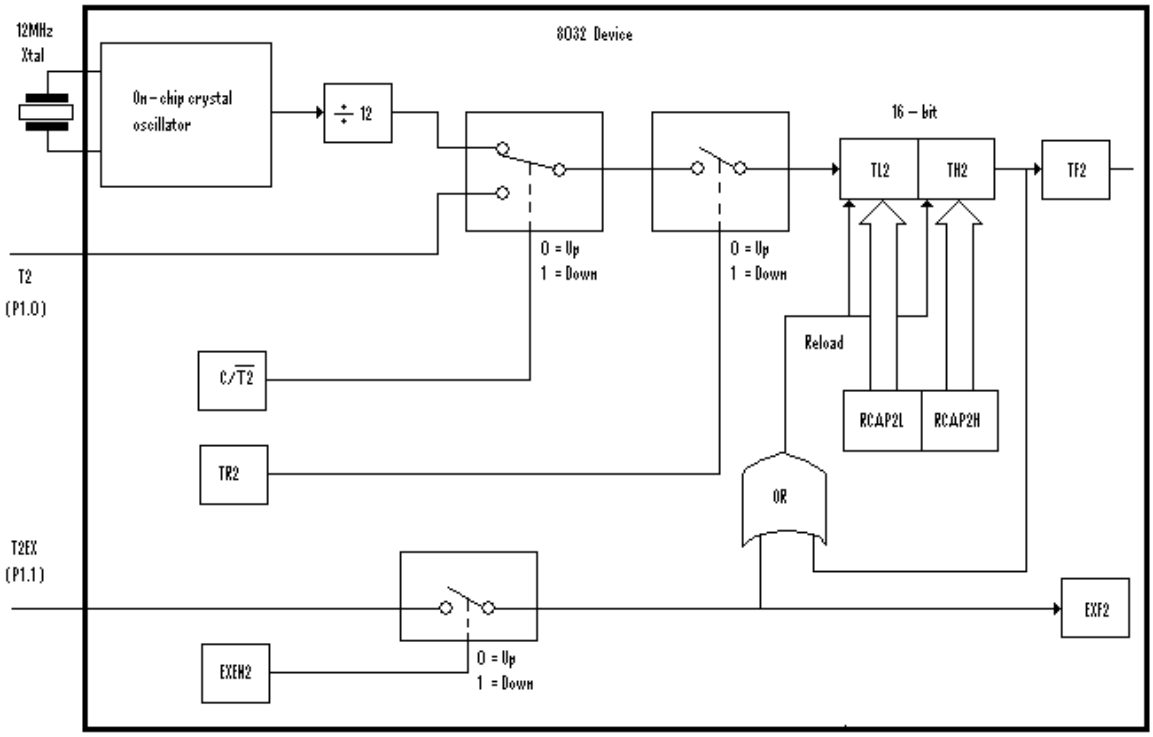

Figure 4-1 Timer 2 Auto-reload Mode

To operate Timer 2 in auto-reload mode, the CP/RL2 bit (T2CON.0) must be cleared. In this mode, Timer 2 (TH2/TL2) will be reloaded with the reload value (RCAP2H/RCAP2L) whenever it overflows from FFFFh back to 0000h. An overflow of Timer 2 will cause the TF2 bit to be set, which will cause an interrupt to be triggered, if Timer 2 interrupt is enabled. Note that TF2 will not be set on an overflow condition if either RCLK or TCLK (T2CON.5 or T2CON.4) are set, which is the case if Timer 2 is being used as a baud rate generator.

Additionally, by also setting EXEN2 (T2CON.3), a reload will also occur whenever a 1-0 transition is detected on T2EX (P1.1). A reload which occurs as a result of such a transition will cause the EXF2 (T2CON.6) flag to be set, triggering a Timer 2 interrupt if the said interrupt has been enabled.

4.3.5 Timer 2 in Capture mode

A new mode specific to Timer 2 is called "Capture Mode." As the name implies, this mode captures the value of Timer 2 (TH2 and TL2) into the capture SFRs (RCAP2H and RCAP2L). To put Timer 2 in capture mode, CP/RL2 (T2CON.0) must be set, as must be EXEN2 (T2CON.3).

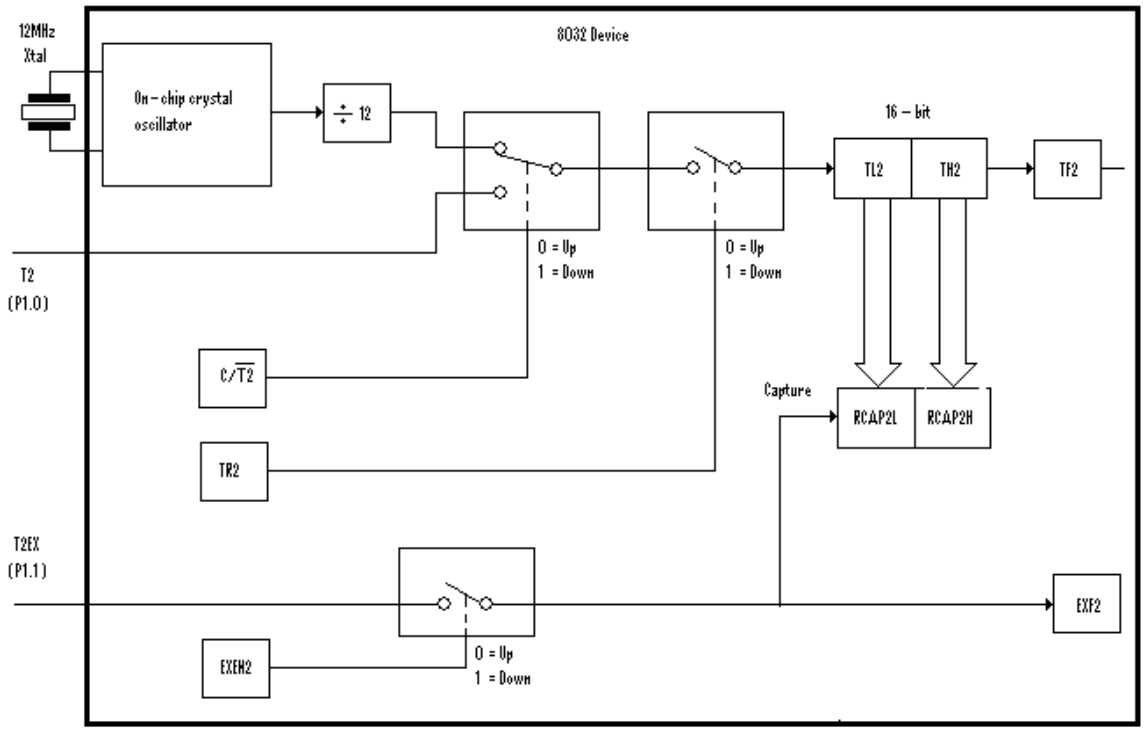

Figure 4-2 Timer 2 in 16-bit capture mode

When configured as mentioned above, a capture will occur whenever a 1-0 transition is detected on T2EX (P1.1). At the moment the transition is detected, the current values of TH2 and TL2 will be copied into RCAP2H and RCAP2L, respectively. At the same time, the EXF2 (T2CON.6) bit will be set, which will trigger an interrupt if Timer 2 interrupt is enabled.

NOTE 1: Even in capture mode, an overflow of Timer 2 will result in TF2 being set and an interrupt being triggered.

NOTE 2: Capture mode is an efficient way to measure the time between events. At the moment that an event occurs, the current value of Timer 2 will be copied into RCAP2H/L. However, Timer 2 will not stop and an interrupt will be triggered. Thus our interrupt routine may copy the value of RCAP2H/L to a temporary holding variable without having to stop Timer 2. When another capture occurs, our interrupt can take the difference between the two values to determine the elapsed time. Again, the main advantage is that we do not have to stop Timer 2 to read its value, as is the case with Timer 0 and Timer 1, where there is the possibility of reading the wrong value if the timer count happens to be close to a roll-over .

4.3.6 Timer 2 Interrupt

As is the case with the other two timers, Timer 2 can be configured to trigger an interrupt. In fact, as can be seen in Table 4-5 a number of situations can trigger a Timer 2 interrupt.

To enable Timer 2 interrupt, set ET2 (IE.5) and it should be noted that this bit of IE is only valid on an 8032 or other devices of the 8051 family which have a Tmer 2 on board. Similarly, the priority of Timer 2 interrupt can be configured using PT2 (IP.5). As always, we have to make sure to also set the EA (IE.7) bit when enabling any interrupt. This will ensure that the controller would recognize the interrupt.

Interrupt Name	Interrupt Number	Flag	Interrupt Hex Vector Address
External 0	0	IE0	0003
Timer 0	1	TF0	000B
External 1	2	IE1	0013
Timer 1	3	TF1	001B
Serial	4	RI or TI	0023
Timer 2	5	TF2 or EXF2	002B

Table 4-5 8032 Interrupt Vector Table location

Once Timer 2 interrupt has been enabled, a Timer 2 interrupt will be triggered whenever TF2 (T2CON.7) or EXF2 (T2CON.6) are set. The Timer 2 Interrupt routine must be placed at 002Bh in code memory.

NOTE: Like the Serial Interrupt, Timer 2 interrupt does not automatically clear the interrupt flag that triggered the interrupt. Since there are two conditions that can trigger a Timer 2 interrupt, either TF2 or EXF2 being set, the microcontroller does not reset these flags automatically when jumping to the ISR. Therefore we have to add some code in the interrupt routine which determines the source of the interrupt and act accordingly. It is possible (and even probable!) that we will want to do one thing when the timer overflows and something completely different when a capture or reload is triggered by an external event. Thus it is imperative to always clear TF2 and/or EXF2 in the Timer 2 ISR. Failing to do so will cause the interrupt to be triggered repeatedly until the bits are cleared.