SECTION 9: INTERRUPTS

The High–Speed Microcontroller family utilizes a three–priority interrupt system. The number of interrupts varies according to the specific device. Each source has an independent priority bit, flag, interrupt vector, and enable. In addition, interrupts can be globally enabled (or disabled). The system is compatible with the original 8051 family. All of the original interrupts are available. Several new sources have been added with new associated control and status bits, and new interrupt vectors. Note that the interrupt vector table can extend from 0000h to 006Bh, so existing code may require a relocation of the start address to avoid a conflict with the upper end of the vector table. A summary of all interrupts appears in Table 9–1 below.

INTERRUPT SUMMARY Table 9–1

INTERRUPT	INTERRUPT VECTOR	NATURAL PRIORITY	FLAG	ENABLE	PRIORITY CONTROL
Power–fail Indicator	33h	0	PFI (WDCON.4)	EPFI (WDCON.5)	N/A
External Interrupt 0	03h	1	IE0 (TCON.1)**	EX0 (IE.0)	PX0 (IP.0)
Timer 0 Overflow	0Bh	2	TF0 (TCON.5)*	ET0 (IE.1)	PT0 (IP.1)
External Interrupt 1	13h	3	IE1 (TCON.3)**	EX1 (IE.2)	PX1 (IP.2)
Timer 1 Overflow	1Bh	4	TF1 (TCON.7)*	ET1 (IE.3)	PT1 (IP.3)
Serial Port 0	23h	5	RI_0 (SCON0.0), TI_0 (SCON0.1)	ES0 (IE.4)	PS0 (IP.4)
Timer 2 Overflow	2Bh	6	TF2 (T2CON.7)	ET2 (IE.5)	PT2 (IP.5)
Serial Port 1	3Bh	7	RI_1 (SCON1.0), TI_1 (SCON1.1)	ES1 (IE.6)	PS1 (IP.6)
External Interrupt 2	43h	8	IE2 (EXIF.4)	EX2 (EIE.0)	PX2 (EIP.0)
External Interrupt 3	4Bh	9	IE3 (EXIF.5)	EX3 (EIE.1)	PX3 (EIP.1)
External Interrupt 4	53h	10	IE4 (EXIF.6)	EX4 (EIE.2)	PX4 (EIP.2)
External Interrupt 5	5Bh	11	IE5 (EXIF.7)	EX5 (EIE.3)	PX5 (EIP.3)
Watchdog Interrupt	63h	12	WDIF (WDCON.3)	EWDI (EIE.4)	PWDI (EIP.4)
Real –Time Clock	6Bh	13	RTCIF (RTCC.1)	ERTCI (EIE.5)	PRTCI (EIP.5)

Unless marked these flags must be cleared manually by software.

* Cleared automatically by hardware when the service routine is vectored to.

** If edge triggered, cleared automatically by hardware when the service routine is vectored to. If level triggered, flag follows the state of the pin.

INTERRUPT OVERVIEW

An interrupt allows the software to react to unscheduled or asynchronous events. When an interrupt occurs, the CPU is expected to "service" the interrupt. This service takes the form of an Interrupt Service Routine (ISR). The ISR resides at a predetermined address as shown in Table 9–1. When the interrupt occurs, the CPU will vector to the appropriate location. It will run the code found at this location, staying in an interrupt service state until done with the ISR. Once an ISR has begun, it can be interrupted only by a higher priority interrupt. The ISR is terminated by a return from interrupt instruction (RETI). When an RETI is performed, the processor will return to the instruction that would have been next when the interrupt occurred.

Each interrupt source has an associated vector. This is the address to which the CPU will jump when the interrupt occurs. When the interrupt condition occurs, the processor will also indicate this by setting a flag bit. This

bit is set regardless of whether the interrupt is enabled or not. That is, the flag responds to the condition, not the interrupt. Most flags must be cleared manually by software. However, IE0 and IE1 are cleared automatically by hardware when the service routine is vectored to if the interrupt was edge triggered. In level triggered mode, the flag follows the state of the pin. Flags TF0 and TF1 are always cleared automatically when the service routine is vectored to. Refer to the individual bit descriptions for more details. In order for the processor to acknowledge the interrupt and vector to the ISR, the interrupt must be enabled. Each source has an independent enable as shown in Table 9–1.

Prior to using any source, interrupts must be globally enabled. This is done using the EA bit at location IE.7. Setting this bit to a logic 1 allows individual interrupts to be enabled. Setting it to a logic 0 disables all interrupts regardless of the individual interrupt enables. The only exception is the Power–fail Interrupt. This is subject to its individual enable only. The EA bit has no effect on the Power–fail Interrupt.

INTERRUPT SOURCES

Various combinations of interrupt sources are available on different members of the High–Speed Microcontroller family. These are broken into several categories : External, Timer–based, Serial Communication, real–time clock and Power Monitor. Each type is described below. Interrupt sources are sampled once per machine cycle. If the source goes active after the sample, it will not be registered until the next cycle.

External Interrupts

The High–Speed Microcontroller has 6 external interrupt sources. These include the standard 2 interrupts of the 8051 architecture and 4 new sources. The original interrupts are $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$. These are active low, but can be programmed to be edge– or level–sensitive. The detection mode is controlled by bits IT0 and IT1, respectively. When ITx=0, the interrupt is triggered by a logic 0 on the appropriate interrupt pin. The interrupt condition remains in force as long as the pin is low. When ITx=1, the interrupt is pseudo edge–triggered. This means that if on successive samples, the pin is high then low, the interrupt is activated.

Since the external interrupts are sampled, the pin driver of an edge–triggered interrupt should hold the both the high, then the low condition for at least one machine cycles (each) to insure detection. This means maximum sampling frequency on any interrupt pin is 1/8 of the main oscillator frequency.

It is important to note that level–sensitive interrupts are not latched. If the interrupt is level–sensitive, the condition must be present until the processor can respond to the interrupt. This is most important if other interrupts are being used with a higher or equal priority. If the device is currently processing another interrupt, the condition must be present until the present interrupt is complete. This is because the level–sensitive interrupt will not be sampled until the RETI instruction is executed.

The remaining 4 external interrupts are similar in nature, with two differences. First, INT2 and INT4 are active high instead of active low. Second, all of the four new interrupts are edge–detect only. They do not have level–detect modes. All associated bits and flags operate the same and have the same polarity as the original two. A logic 1 indicates the presence of a condition, not the logic state of the pin.

If the Power Management Modes are utilized, the designer must remember that edge triggered interrupts must be high and low for one machine cycle before being recognized. This means that in PMM1 it will require 128 external clock cycles to recognize a level sensitive interrupt. Similarly, in PMM2 it will require 2048 external clock cycles to recognize a level sensitive interrupt. As a result, the interrupt latency for these interrupts will be slightly longer in PMM1or PMM2.

Timer Interrupts

The High–Speed Microcontroller incorporates three 16–bit programmable timers, each of which can generate an interrupt. In addition, some members of the family incorporate a programmable watchdog timer. The three programmable timers operate in the same manner as the 80C52. Each timer has an independent interrupt enable, flag, vector, and priority. The Watchdog Timer also has its own interrupt enable, flag, and priority.

Timers 0, 1, and 2 will set their respective flags when the timer overflows from a full condition, depending on its mode. This flag will be set regardless of the interrupt enable state. If the interrupt is enabled, this event will also cause a jump to the corresponding interrupt vector. For timers 0 and 1, the flags are cleared when the processor jumps to the interrupt vector. Thus these flags are not available for use by the interrupt service routine (ISR), but are available outside of the ISR and in

applications that don't acknowledge the interrupt (i.e., jump to the vector). If the interrupt is not acknowledged, then software must manually clear the flag bit. In timer 2, jumping to the interrupt vector does not clear the flag, so software must always clear it manually. Timer 0 and 1 flag bits reside in the TCON register. Timer 2 flag bit resides in the T2CON register. The interrupt enables and priorities for timers 0, 1, and 2 reside in the IE and IP registers respectively.

The Watchdog Interrupt usually has a different connotation than the Timer interrupts. Unless the Watchdog is being used as a very long timer, the interrupt means the software has failed to reset the counter and may be lost. The ISR can attempt to determine the system state. If the Watchdog is not cleared, the CPU will be reset in 512 clocks if EWT=1. Like other sources, the Watchdog Timer has a flag bit, an enable, and a priority. It also has its own vector. These are summarized in table 9–1.

Serial Communication Interrupts

Each UART is capable of generating an interrupt. The UART has its own interrupt enable, vector, and priority. The UART differs from other sources as it has two flags. These are used by the ISR to determine whether the interrupt comes from a received word or a transmitted one. Unlike the timers, the UART flags are not altered when the interrupt is serviced. Software must change them manually.

When a UART finishes the transmission of a word, an interrupt will be generated (if enabled). Likewise, the UART will generate an interrupt when a word is completely received. The CPU will not be notified until the word is completely received or transmitted.

Real–Time Clock

The DS87C530 real–time clock (RTC) has the ability to assert an RTC interrupt if enabled. The alarm can be programmed for a specific time once per day, or can be a recurring alarm once per hour, minute, second, or subsecond. This interrupt has the lowest priority of all interrupts, but can be used to bring the device out of Stop mode if desired. More information on this interrupt can be found in Section 14.

Power–fail Interrupt

Some devices have the ability to generate an interrupt when V_{CC} drops below a predetermined level. These devices compare V_{CC} against an internal reference. If V_{CC} drops below the level V_{PFW}, an interrupt will result (if enabled). Note that the Power–fail Interrupt has the highest priority. The priority level cannot be altered by the user, but the interrupt can be disabled if not needed. The level of V_{PFW} is provided in the data sheet specifications associated with each product. Note that the EPFI bit enables the Power–fail Interrupt. This bit is not subject to the global interrupt enable (EA). The Power–fail interrupt is a level–sensitive interrupt and will remain set as long as V_{CC} remains below V_{PFW}.

Simulated Interrupts

Software can simulate any interrupt source by setting the corresponding flag bit. This forces an interrupt condition which will be acknowledged if enabled and is otherwise indistinguishable from the real thing. Thus an interrupt flag bit should never be set to a logic 1 by software inadvertently. Once an interrupt has been acknowledged, software cannot prevent or end the interrupt by clearing its flag. However, if for some reason the interrupt acknowledge is delayed, software may clear the flag and thereby prevent the interrupt from occurring. One exception is the real–time clock interrupt flag, RTCIF, which cannot be set in software.

INTERRUPT PRIORITIES

The High–Speed Microcontroller has three interrupt priority levels. These are highest, high, and low. The Power–fail Interrupt is the only source that has highest priority and this level is fixed. The remaining sources are individually programmable to either high or low. Low priority is the default. A low priority interrupt can be interrupted by a high (or highest) priority interrupt. A high priority interrupt can only be interrupted by the Power–fail interrupt.

When an interrupt occurs and is serviced, its priority determines if its ISR can be interrupted. No interrupt source of equal or lesser priority can interrupt another source. That is, an incoming interrupt must be of a higher priority than the one currently being serviced to have priority.

If two interrupt sources of equal priority levels are requested simultaneously, the natural priority is used to arbitrate. The natural priority is given in Table 9–1. Note that natural priority is only used to resolve simultaneous requests. Once an interrupt of a given priority is invoked, only a source that is programmed with a higher priority can intercede.

INTERRUPT ACKNOWLEDGE CYCLE

The process of acknowledging an interrupt requires multiple machine cycles that begin with the setting of the associated flag. For edge triggered external interrupts and internal interrupt sources, the interrupt flags are set automatically by hardware. For level sensitive external interrupts, the flags are actually under control of the external signal, and the flag will rise and fall with the pin level. Each interrupt flag is sampled once per machine cycle. Later in the same machine cycle, the samples are polled by hardware. If the sample indicates a pending interrupt and the interrupt is enabled, then on the next machine cycle it will be acknowledged by the hardware forcing an LCALL to the appropriate vector address. This LCALL will occur unless blocked by one of the following conditions.

1. An interrupt of equal or greater priority has already been invoked and the RETI instruction has not been issued to terminate it.
2. The current machine cycle is not the final cycle in the execution of the current instruction.
3. The instruction in progress is an RETI or an access to IP, IE, EIP, or EIE.

The individual interrupt sources and associated enable and priority bits are shown in Figure 9–1. While the final selection of the appropriate interrupt vector address is referred to as a polling process, this function is actually performed in a single machine cycle using combinatorial logic.

INTERRUPT LATENCY

Interrupt response will require a varying amount of time depending on the state of the microcontroller when the interrupt occurs. If the microcontroller is performing an ISR with equal or greater priority, the new interrupt will not be invoked. In other cases, the response time depends on the current instruction. The fastest possible response to an interrupt is 5 machine cycles. This includes one cycle for detecting the interrupt and four cycles to perform the LCALL that is inherent in the interrupt request. The maximum response time (if no other interrupt is in service) occurs if the microcontroller is performing an RETI instruction, and then executes a MUL or DIV as the next instruction. From the time an interrupt source is activated (not detected), the longest reaction time is 13 machine cycles. This includes 1 cycle to detect the interrupt, 3 cycles to finish the RETI, 5 to perform the MUL or DIV, then 4 for the LCALL to the ISR.

The maximum latency of 13 machine cycle is 52 clocks (13*4). Note that the maximum interrupt latency of an 8051 is 96 clocks (8 machine cycles @ 12 clocks per machine cycle). The maximum latency for the High–Speed Microcontroller at 25 MHz is about 2 µs. The use of Power Management modes can further increase the interrupt latency.

INTERRUPT FUNCTIONAL DESCRIPTION Figure 9–1