TABLE OF CONTENTS

USER'S GUIDE 1

Section 1: Introduction 2
Section 2: Ordering Information 3
Section 3: Architecture 4
Section 4: Programming Model 7
Section 5: CPU Timing 58
Section 6: Memory Access 68
Section 7 Power Management 79
Section 8: Reset Conditions 92
Section 9: Interrupts 95
Section 10: Parallel I/O 100
Section 11: Programmable Timers 104
Section 12: Serial I/O 119
Section 13: Timed Access Protection 132
Section 14: Real–Time Clock 135
Section 15: Battery Backup 139
Section 16: Instruction Set Details 141
Section 17: Troubleshooting 149

USER’S GUIDE

SECTION 1: INTRODUCTION

The Dallas Semiconductor High–Speed Microcontroller is an 8051–compatible device that provides improved performance and power consumption compared to the original version. It retains instruction set and object code compatibility with the 8051, yet performs the same operations in fewer clock cycles. Consequently, more throughput is possible for the same crystal speed. As an alternative, the High–Speed Microcontroller can be run slowly to save power. The more efficient design allows a much slower crystal speed to get the same results as an original 8051, using much less power.

The fundamental innovation of the High–Speed Microcontroller is the use of only four clocks per instruction cycle compared with twelve for the original 8051. This results in up to 3 times improvement in performance. In addition, the High–Speed Microcontroller is updated with several new peripherals and features while providing all of the standard features of an 80C32. These include 256 bytes of on–chip RAM, 32 I/O ports, three 16–bit timer/counters, and an on–chip UART. All devices provide 256 bytes of RAM for variables and stack. 128 bytes can be reached using direct addressing and 128 using indirect addressing. The RAM area matches the standard 80C32.

In addition to improved efficiency, many members of the High–Speed Microcontroller family can operate at a maximum clock rate of 33 MHz. Combined with the 3 times performance, this allows for a maximum performance equivalent to a 99 MHz 8051. This level of computing power is comparable to many 16–bit processors, but without the added expense and complexity of implementing a 16–bit interface.

A number of peripherals were added to the original 80C32 core when designing the High–Speed Microcontroller family. Some devices have a programmable Watchdog Timer to supervise the system. It will count up to a user programmable interval and then reset the CPU unless cleared by software. Other features such as a second, full–function UART and dual data pointers are available to minimize external hardware and system complexity. The incorporation of 6 external interrupts allows greater flexibility in dealing with external events.

Some members of the High–Speed Microcontroller family incorporate Power Management Modes which allow the device to dynamically vary the internal clock speed from 4 clocks per cycle (default) to 64 or 1024 clocks per cycle. Because power consumption is directly proportional to clock speed, the device can reduce its operating frequency during periods of little or no activity. This greatly reduces power consumption. The switch-back feature allows the device to quickly return to divide by 4 mode upon receipt of an interrupt or serial port activity, allowing the device to respond to external events while in Power Management Mode.

Various memory configurations are available with the High–Speed Microcontroller family. EPROM and Mask programmable ROM versions are available for program memory. Some versions incorporate extended MOVX SRAM on–chip, reducing or eliminating the need for external data memory. This memory can be made nonvolatile in the DS87C530 through the use of an external lithium battery.

SECTION 2: ORDERING INFORMATION

The High–Speed Microcontroller family follows the part numbering convention shown below. Note that all combinations of devices are not currently available. Please refer to individual data sheets for the available versions.

DS80C320–MCG

SPEED:	D	18 MHz
	G	25 MHz
	L	33 MHz
TEMPERATURE:	C	0°C to 70°C
	N	–40°C to +85°C
PACKAGE:	M	PLASTIC DIP
	Q	PLCC
	E	THIN PLASTIC QUAD FLAT PACK (TQFP)
	W	40–PIN WINDOWED CERDIP
	K	52–PIN WINDOWED CERQUAD
OPERATING VOLTAGE:	0	+5V
	3	+3V OR WIDE VOLTAGE
MEMORY TYPE:	0	ROMless
	3	ROM
	7	EPROM

SECTION 3: ARCHITECTURE

The High–Speed Microcontroller is based on the industry standard 80C52. The core is an accumulator based architecture using internal registers for data storage and peripheral control. It executes the standard 8051 instruction set. This section provides a brief description of each architecture feature. Details concerning the programming model, instruction set, and register descriptions are provided in Section 4.

ALU

The ALU is responsible for math functions, comparisons, and general decision making in the High–Speed Microcontroller. The ALU is not explicitly used by software. Instruction decoding prepares the ALU automatically and passes it the appropriate data. The ALU primarily uses two special function registers (SFRs) as the source and destination for all operations. These are the Accumulator and B register. The ALU also provides status information in the Program Status Register. The SFRs are described below.

SPECIAL FUNCTION REGISTERS

All peripherals and operations that are not explicit instructions in the High–Speed Microcontroller are controlled via Special Function Registers (SFRs). All SFRs are described in Section 4. The most commonly used registers that are basic to the architecture are also described below.

Accumulator

The Accumulator is the primary register used in the High–Speed Microcontroller. It is the source and destination of most math, data movement, decisions, and other operations. Although it can be bypassed, most high–speed instructions require the use of the Accumulator (ACC) as one argument.

B Register

The B register is used as the second 8–bit argument in multiply and divide operations. When not used for these purposes, the B register can be used as a general purpose register.

Program Status Word

The Program Status Word holds a selection of bit flags that include the Carry Flag, Auxiliary Carry Flag, General Purpose Flag, Register Bank Select, Overflow Flag, and Parity Flag.

Data Pointer(s)

The Data Pointer is used to designate a memory address for the MOVX instruction. This address can point to a MOVX RAM location, either on– or off–chip, or a memory mapped peripheral. When moving data from one memory area to another or from memory to a memory mapped peripheral, a pointer is needed for both the source and destination. Thus, the High–Speed Microcontroller offers two data pointers. The user selects the active pointer via a dedicated SFR bit.

Stack Pointer

The High–Speed Microcontroller provides a Stack in the scratchpad RAM area discussed below. The Stack Pointer denotes the register location at the top of the Stack, which is the last used value. The user can place the Stack anywhere in scratchpad RAM by setting the Stack Pointer to that location.

I/O Ports

The standard High–Speed Microcontroller offers four 8–bit I/O ports. ROMless versions use Port 0 and Port 2 as address and data busses. In those versions, only two ports are available for general purpose I/O. Each I/O port is a Special Function Register that can be written or read. The I/O port has a latch that retains the value which software writes. In general, during a read operation, software reads the state of the external pin. Each port is represented by an SFR location.

Timer/Counters

Three 16–bit Timer/Counters are available in the High–Speed Microcontroller. Each timer is contained in two SFR locations that can be written or read by software. The timers are controlled by other SFRs described in Section 4.

UARTs

The High–Speed Microcontroller provides one or two UARTs. These are controlled and accessed as SFRs. Each UART has an address that is used to read or write the UART. The same address is used for both read and write operations. The microcontroller distinguishes between a read and a write by the instruction. Each UART is controlled by its own SFR control register.

SCRATCHPAD REGISTERS (RAM)

The High–Speed Core provides 256 bytes of Scratchpad RAM for general purpose data and variable storage. The first 128 bytes are directly available to software. The second 128 are available through indirect addressing discussed below. Selected portions of this RAM have other optional functions.

Stack

The stack is a RAM area that the microcontroller uses to store return address information during Calls and Interrupts. The user can also place variables on the stack when necessary. The Stack Pointer mentioned above designates the RAM location that is the top of the stack. Thus, depending on the value of the Stack Pointer, the stack can be located anywhere in the 256 bytes of RAM. A common location would be in the upper 128 bytes of RAM, as these are accessible through indirect addressing only.

Working Registers

The first thirty two bytes of the Scratchpad RAM can be used as four banks of eight Working Registers for high speed data movement. Using four banks, software can quickly change context by simply changing to a different bank. In addition to the Accumulator, the Working Registers are commonly used as a data source or destination. Some of the Working Registers can also be used as pointers to other RAM locations (indirect addressing).

PROGRAM COUNTER

The Program Counter (PC) is a 16–bit value that designates the next program address to be fetched. On–chip hardware automatically increments the PC value to move to the next ROM location.

ADDRESS/DATA BUS

The High–Speed Microcontroller addresses a 64KB program and 64KB data memory area. In the ROMless versions, all memory is outside. Other versions use a combination of internal and external memory. When external memory is accessed, Ports 0 and 2 are used as a multiplexed address and data bus. Port 2 provides the address MSB. Even versions with internal memory can use the bus on Ports 0 and 2 to access more memory.

WATCHDOG TIMER

The Watchdog Timer provides a supervisory function for applications that cannot afford to run out of control. The Watchdog Timer is a programmable free running timer. If allowed to reach the termination of its count, if enabled, the Watchdog will reset the CPU. Software must prevent this by clearing or resetting the Watchdog prior to its time–out.

POWER MONITOR

Some members of the High–Speed Microcontroller family incorporate a band–gap reference and analog circuitry to monitor the power supply conditions. If V_{CC} begins to drop out of tolerance, the Power Monitor will issue an optional early warning Power–fail Interrupt. If power continues to fall, the Power Monitor will invoke a reset condition. This will remain until power returns to normal operating voltage. The Power Monitor also functions on power up, holding the microcontroller in a reset state until power is stable.

INTERRUPTS

The High–Speed Microcontroller is capable of evaluating a number of interrupt sources simultaneously. Each version of the High–Speed Microcontroller provides a different number of interrupt sources. Each interrupt has an associated interrupt vector, flag, priority, and enable. These interrupts can be globally enabled or disabled.

TIMING CONTROL

The High–Speed Microcontroller provides an on–chip oscillator for use with an external crystal. This can be bypassed by injecting a clock source into the XTAL1 pin. The clock source is used to create machine cycle timing (four clocks), ALE, $\overline{PSEN}$, Watchdog, Timer, and serial baud rate timing. In addition, some devices incorporate an on–chip ring oscillator which can be used to provide an approximately 2–4 MHz clock source.

REAL–TIME CLOCK

The DS87C530 incorporates a real–time clock (RTC), which is accessed via SFR locations. The RTC is divided into hour, minute, second, and subsecond registers, and also incorporates a 65536 day calendar. Alarm registers allow the RTC to issue interrupts at a specific time once per day, or as a recurring alarm every hour, minute, or second. An external watch crystal and lithium power source allow the processor to maintain timekeeping in the absence of V_{CC}.

FEATURE SUMMARY

The High–Speed Microcontroller family offers a combination of features and peripherals as shown in Table 3–1. This User's Guide is designed as a comprehensive guide covering all the features available in the High–Speed Microcontroller family. The designer should investigate the specific data sheet to determine which features are available on a particular device.

PRODUCT FEATURE MATRIX 3–1

FEATURE	DS80C310	DS80C320	DS80C323	DS83C520	DS87C520	DS87C530
Internal Program ROM				16KB Mask ROM	16KB EPROM	16KB EPROM
Internal Scratchpad RAM	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes
Internal MOVX SRAM				1KB SRAM	1KB SRAM	1KB SRAM
Serial Ports	1	2	2	2	2	2
External Interrupts	6	6	6	6	6	6
16–bit Timers	3	3	3	3	3	3
Watchdog Timer		✓	✓	✓	✓	✓
Power–fail/Precision Reset		✓	✓	✓	✓	✓
Power–Fail Interrupt		✓	✓	✓	✓	✓
Data Pointers	2	2	2	2	2	2
Power Management Modes				✓	✓	✓
Ring Oscillator		✓	✓	✓	✓	✓
EMI Reduction Mode				✓	✓	✓
Real–Time Clock						✓
Nonvolatile SRAM						✓
Operating Voltage	4.5V–5.5V	4.5V–5.5V	2.7V–5.5	4.5V–5.5V	4.5V–5.5V	4.5V–5.5V

SECTION 4: PROGRAMMING MODEL

This section provides a programmer's overview of the High–Speed Microcontroller core. It includes information on the memory map, on–chip RAM, Special Function Registers (SFRs), and instruction set. The programming model of the High–Speed Microcontroller is very similar to that of the industry standard 80C52. The memory map is identical. It uses the same instruction set, though instruction timing is improved. Several new SFRs have been added.

MEMORY ORGANIZATION

The High–Speed Microcontroller, like the 8052, uses several distinct memory areas. These are Registers, program memory, and data memory. Registers serve to control on–chip peripherals and as RAM. Note that Registers (on–chip RAM) are separate from data memory. Registers are divided into three categories including directly addressed on–chip RAM, indirectly addressed on–chip RAM, and Special Function Registers. The program and data memory areas are discussed under Memory Map. The Registers are discussed under Register Map.

MEMORY MAP

The High–Speed Microcontroller uses a memory addressing scheme that separates program memory (ROM) from data memory (RAM). Each area is 64KB beginning at address 0000h and ending at FFFFh as shown in Figure 4–1. The program and data segments can overlap since they are accessed in different ways. Program memory is fetched by the microcontroller automatically. These addresses are never written by software. In fact, there are no instructions that allow the ROM area to be written. There is one instruction (MOVC) that is used to explicitly read the program area. This is commonly used to read look–up tables. The data memory area is accessed explicitly using the MOVX instruction. This instruction provides multiple ways of specifying the target address. It is used to access the 64KB of data memory.

The address and data range of devices with on–chip program and data memory overlap the 64K memory space. When on–chip memory is enabled, accessing memory in the on–chip range will cause the device to access internal memory. Memory accesses beyond the internal range will be addressed externally via ports 0 and 2.

The ROMSIZE feature allows software to dynamically configure the maximum address of on–chip program memory. This allows the device to act as a bootstrap loader for an external Flash or nonvolatile SRAM. Secondly, this method can also be used to increase the amount of available program memory from 64KB to 80KB without bank switching. For more information on this feature, please consult Section 6.

Program and data memory can also be increased beyond the 64KB limit using bank switching techniques. This is described in Application Note 81, Memory Expansion with the High–Speed Microcontroller family.

REGISTER MAP

The Register Map is illustrated in Figure 4–2. It is entirely separate from the program and data memory areas mentioned above. A separate class of instructions is used to access the registers. There are 256 potential register location values. In practice, the High–Speed Microcontroller has 256 bytes of Scratchpad RAM and up to 128 Special Function Registers (SFRs). This is possible since the upper 128 Scratchpad RAM locations can only be accessed indirectly. That is, the contents of a Working Register (described below) will designate the RAM location. Thus a direct reference to one of the upper 128 locations must be an SFR access. Direct RAM is reached at locations 0 to 7Fh (0 to 127). SFRs are accessed directly between 80h and FFh (128 to 255). The RAM locations between 128 and 255 can be reached through an indirect reference to those locations.

Scratchpad RAM is available for general purpose data storage. It is commonly used in place of off–chip RAM when the total data contents are small. When off–chip RAM is still needed, the Scratchpad area will still provide the fastest general purpose access. Within the 256 bytes of RAM, there are several special purpose areas. These are described as follows.

Bit Addressable Locations

In addition to direct register access, some individual bits are also accessible. There are individually addressable bits in both the RAM and SFR area. In the Scratchpad RAM area, registers 20h to 2Fh are bit addressable. This provides 128 (16 * 8) individual bits available to software. A bit access is distinguished from a full register access by the type of instruction. Addressing modes

are discussed in Section 5. In the SFR area, any register location ending in a 0 or 8 is bit addressable. Figure 4–3 shows details of the on–chip RAM addressing including the locations of individual RAM bits.

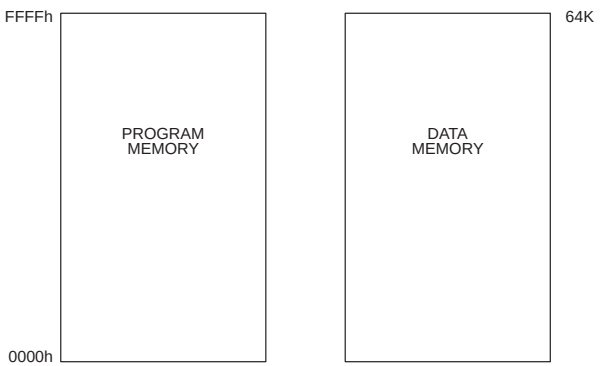
Working Registers

As part of the lower 128 bytes of RAM, there are four banks of eight Working Registers (each). The Working Registers are general purpose RAM locations that can be addressed in a special way. They are designated R0 through R7. Since there are four banks, the currently selected bank will be used by any instruction using R0–R7. This allows software to change context by simply switching banks. This is controlled via the Program Status Word register in the SFR area described below. The Working Registers also allow their contents to be used for indirect addressing of the upper 128 bytes of RAM. Thus an instruction can designate the value stored in R0 (for example) to address the upper RAM. This value might be the result of another calculation.

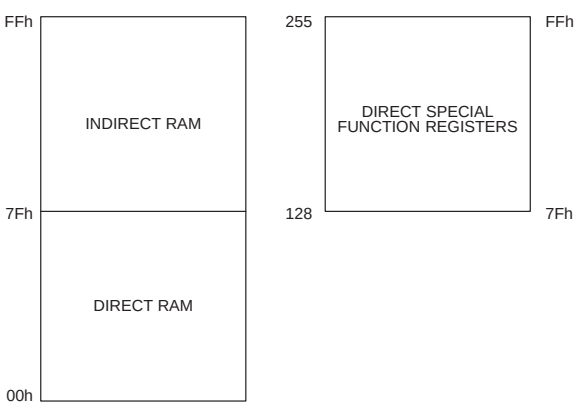
Stack

Another use of the Scratchpad area is for the programmer's stack. This area is selected using the Stack Pointer (SP,81h) SFR. Whenever a call or interrupt is invoked, the return address is placed on the Stack. It also is available to the programmer for variables, etc. Since the Stack can be moved, there is no fixed location within the RAM designated as Stack. The Stack Pointer will default to 07h on reset. The user can then move it as needed. A convenient location would be the upper RAM area (>7Fh) since this is only available indirectly. The SP will point to the last used value. Therefore, the next value placed on the Stack is put at SP + 1. Each PUSH or CALL will increment the SP by the appropriate value. Each POP or RET will decrement as well.

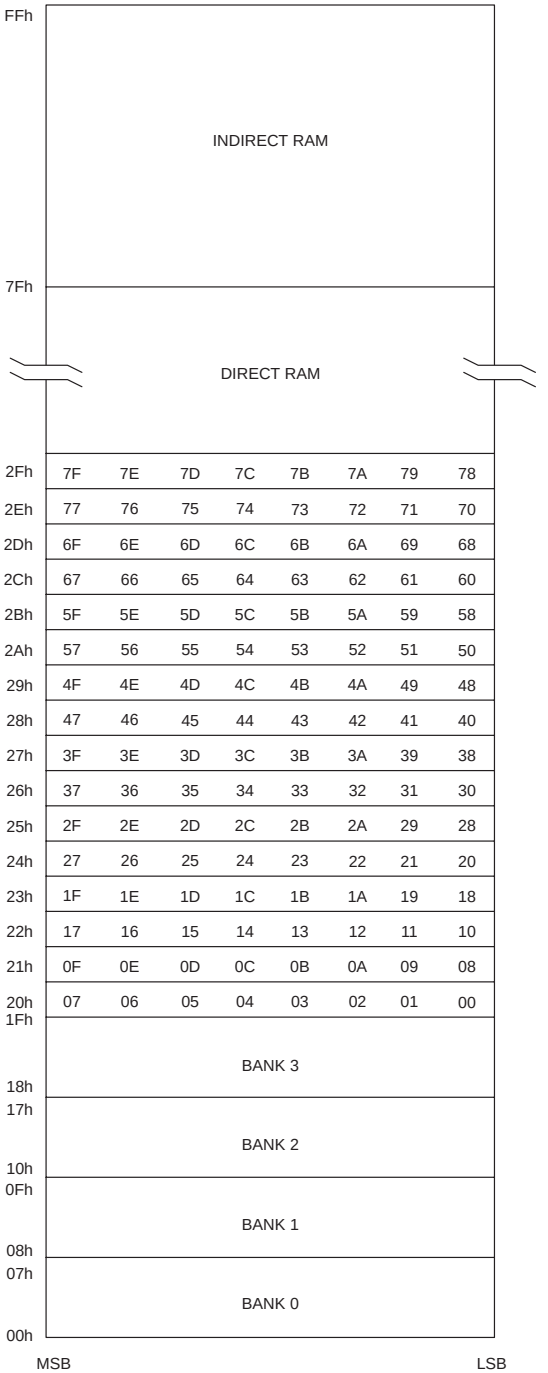
MEMORY MAP Figure 4–1



REGISTER MAP Figure 4–2



SCRATCHPAD REGISTER ADDRESSING Figure 4–3



SPECIAL FUNCTION REGISTERS

The High–Speed Microcontroller, like the 8051, uses Special Function Registers (SFRs) to control peripherals and modes. In many cases, an SFR will control individual functions or report status on individual functions. The SFRs reside in register locations 80h–FFh and are reached using direct addressing. SFRs that end in 0 or 8 are bit addressable.

All standard SFR locations from the original 8051 are duplicated in the High–Speed Microcontroller, with several additions. Following tables illustrate the locations of the SFRs for various devices. Following each table is a description of the default reset conditions of all SFR bits. The following section is a detailed description of each Special Function Register.

DS80C310 SPECIAL FUNCTION REGISTER LOCATIONS Table 4–1

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
SP									81h
DPL									82h
DPH									83h
DPL1									84h
DPH1									85h
DPS	0	0	0	0	0	0	0	SEL	86h
PCON	SMOD_0	SMOD0	–	–	GF1	GF0	STOP	IDLE	87h
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	88h
TMOD	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0	89h
TL0									8Ah
TL1									8Bh
TH0									8Ch
TH1									8Dh
CKCON	–	–	T2M	T1M	T0M	MD2	MD1	MD0	8Eh
P1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	90h
EXIF	IE5	IE4	IE3	IE2	–	–	–	–	91h
SCON0	SM0/FE_0	SM1_0	SM2_0	REN_0	TB8_0	RB8_0	TI_0	RI_0	98h
SBUF0									99h
P2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	A0h
IE	EA	–	ET2	ES0	ET1	EX1	ET0	EX0	A8h
SADDR0									A9h
P3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	B0h
IP	–	–	PT2	PS0	PT1	PX1	PT0	PX0	B8h
SADEN0									B9h
STATUS	0	HIP	LIP	1	1	1	1	1	C5h
T2CON	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$	C8h
T2MOD	–	–	–	–	–	–	T2OE	DCEN	C9h
RCAP2L									CAh
RCAP2H									CBh
TL2									CCh
TH2									CDh
PSW	CY	AC	F0	RS1	RS0	OV	F1	P	D0h
WDCON	–	POR	–	–	–	–	–	–	D8h
ACC									E0h
EIE	–	–	–	–	EX5	EX4	EX3	EX2	E8h
B									F0h
EIP	–	–	–	–	PX5	PX4	PX3	PX2	F8h

DS80C310 SPECIAL FUNCTION REGISTER RESET VALUES Table 4–2

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
SP	0	0	0	0	0	1	1	1	81h
DPL	0	0	0	0	0	0	0	0	82h
DPH	0	0	0	0	0	0	0	0	83h
DPL1	0	0	0	0	0	0	0	0	84h
DPH1	0	0	0	0	0	0	0	0	85h
DPS	0	0	0	0	0	0	0	0	86h
PCON	0	0	–	–	0	0	0	0	87h
TCON	0	0	0	0	0	0	0	0	88h
TMOD	0	0	0	0	0	0	0	0	89h
TL0	0	0	0	0	0	0	0	0	8Ah
TL1	0	0	0	0	0	0	0	0	8Bh
TH0	0	0	0	0	0	0	0	0	8Ch
TH1	0	0	0	0	0	0	0	0	8Dh
CKCON	–	–	0	0	0	0	0	1	8Eh
P1	1	1	1	1	1	1	1	1	90h
EXIF	0	0	0	0	–	–	–	–	91h
SCON0	0	0	0	0	0	0	0	0	98h
SBUF0	0	0	0	0	0	0	0	0	99h
P2	1	1	1	1	1	1	1	1	A0h
IE	0	–	0	0	0	0	0	0	A8h
SADDR0	0	0	0	0	0	0	0	0	A9h
P3	1	1	1	1	1	1	1	1	B0h
IP	–	–	0	0	0	0	0	0	B8h
SADEN0	0	0	0	0	0	0	0	0	B9h
STATUS	0	0	0	1	1	1	1	1	C5h
T2CON	0	0	0	0	0	0	0	0	C8h
T2MOD	–	–	–	–	–	–	0	0	C9h
RCAP2L	0	0	0	0	0	0	0	0	CAh
RCAP2H	0	0	0	0	0	0	0	0	CBh
TL2	0	0	0	0	0	0	0	0	CCh
TH2	0	0	0	0	0	0	0	0	CDh
PSW	0	0	0	0	0	0	0	0	D0h
WDCON	–	SPECIAL	–	–	–	–	–	–	D8h
ACC	0	0	0	0	0	0	0	0	E0h
EIE	–	–	–	–	0	0	0	0	E8h
B	0	0	0	0	0	0	0	0	F0h
EIP	–	–	–	–	0	0	0	0	F8h

DS80C320/DS80C323 SPECIAL FUNCTION REGISTER LOCATIONS Table 4–3

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
SP									81h
DPL									82h
DPH									83h
DPL1									84h
DPH1									85h
DPS	0	0	0	0	0	0	0	SEL	86h
PCON	SMOD_0	SMOD0	–	–	GF1	GF0	STOP	IDLE	87h
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	88h
TMOD	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0	89h
TL0									8Ah
TL1									8Bh
TH0									8Ch
TH1									8Dh
CKCON	WD1	WD0	T2M	T1M	T0M	MD2	MD1	MD0	8Eh
P1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	90h
EXIF	IE5	IE4	IE3	IE2	–	RGMD	RGSL	**BGS**	91h
SCON0	SM0/FE_0	SM1_0	SM2_0	REN_0	TB8_0	RB8_0	TI_0	RI_0	98h
SBUF0									99h
P2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	A0h
IE	EA	ES1	ET2	ES0	ET1	EX1	ET0	EX0	A8h
SADDR0									A9h
SADDR1									AAh
P3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	B0h
IP	–	PS1	PT2	PS0	PT1	PX1	PT0	PX0	B8h
SADEN0									B9h
SADEN1									BAh
SCON1	SM0/FE_1	SM1_1	SM2_1	REN_1	TB8_1	RB8_1	TI_1	RI_1	C0h
SBUF1									C1h
STATUS	PIP	HIP	LIP	1	1	1	1	1	C5h
TA									C7h
T2CON	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$	C8h
T2MOD	–	–	–	–	–	–	T2OE	DCEN	C9h
RCAP2L									CAh
RCAP2H									CBh
TL2									CCh
TH2									CDh
PSW	CY	AC	F0	RS1	RS0	OV	F1	P	D0h
WDCON	SMOD_1	**POR**	EPFI	PFI	**WDIF**	WTRF	**EWT**	**RWT**	D8h
ACC									E0h
EIE	–	–	–	EWDI	EX5	EX4	EX3	EX2	E8h
B									F0h
EIP	–	–	–	PWDI	PX5	PX4	PX3	PX2	F8h

Shaded bits are Timed Access protected.

DS80C320/DS80C323 SPECIAL FUNCTION REGISTER RESET VALUES Table 4–4

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
SP	0	0	0	0	0	1	1	1	81h
DPL	0	0	0	0	0	0	0	0	82h
DPH	0	0	0	0	0	0	0	0	83h
DPL1	0	0	0	0	0	0	0	0	84h
DPH1	0	0	0	0	0	0	0	0	85h
DPS	0	0	0	0	0	0	0	0	86h
PCON	0	0	–	–	0	0	0	0	87h
TCON	0	0	0	0	0	0	0	0	88h
TMOD	0	0	0	0	0	0	0	0	89h
TL0	0	0	0	0	0	0	0	0	8Ah
TL1	0	0	0	0	0	0	0	0	8Bh
TH0	0	0	0	0	0	0	0	0	8Ch
TH1	0	0	0	0	0	0	0	0	8Dh
CKCON	0	0	0	0	0	0	0	1	8Eh
P1	1	1	1	1	1	1	1	1	90h
EXIF	0	0	0	0	–	0	SPECIAL	0	91h
SCON0	0	0	0	0	0	0	0	0	98h
SBUF0	0	0	0	0	0	0	0	0	99h
P2	1	1	1	1	1	1	1	1	A0h
IE	0	0	0	0	0	0	0	0	A8h
SADDR0	0	0	0	0	0	0	0	0	A9h
SADDR1	0	0	0	0	0	0	0	0	AAh
P3	1	1	1	1	1	1	1	1	B0h
IP	–	0	0	0	0	0	0	0	B8h
SADEN0	0	0	0	0	0	0	0	0	B9h
SADEN1	0	0	0	0	0	0	0	0	BAh
SCON1	0	0	0	0	0	0	0	0	C0h
SBUF1	0	0	0	0	0	0	0	0	C1h
STATUS	0	0	0	1	1	1	1	1	C5h
TA	1	1	1	1	1	1	1	1	C7h
T2CON	0	0	0	0	0	0	0	0	C8h
T2MOD	–	–	–	–	–	–	0	0	C9h
RCAP2L	0	0	0	0	0	0	0	0	CAh
RCAP2H	0	0	0	0	0	0	0	0	CBh
TL2	0	0	0	0	0	0	0	0	CCh
TH2	0	0	0	0	0	0	0	0	CDh
PSW	0	0	0	0	0	0	0	0	D0h
WDCON	0	SPECIAL	0	SPECIAL	0	SPECIAL	SPECIAL	0	D8h
ACC	0	0	0	0	0	0	0	0	E0h
EIE	–	–	–	0	0	0	0	0	E8h
B	0	0	0	0	0	0	0	0	F0h
EIP	–	–	–	0	0	0	0	0	F8h

DS83C520/DS87C520 SPECIAL FUNCTION REGISTER LOCATIONS Table 4–7

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
P0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	80h
SP									81h
DPL									82h
DPH									83h
DPL1									84h
DPH1									85h
DPS	0	0	0	0	0	0	0	SEL	86h
PCON	SMOD_0	SMOD0	–	–	GF1	GF0	STOP	IDLE	87h
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	88h
TMOD	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0	89h
TL0									8Ah
TL1									8Bh
TH0									8Ch
TH1									8Dh
CKCON	WD1	WD0	T2M	T1M	T0M	MD2	MD1	MD0	8Eh
P1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	90h
EXIF	IE5	IE4	IE3	IE2	XT/$\overline{RG}$	RGMD	RGSL	**BGS**	91h
SCON0	SM0/FE_0	SM1_0	SM2_0	REN_0	TB8_0	RB8_0	TI_0	RI_0	98h
SBUF0									99h
P2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	A0h
IE	EA	ES1	ET2	ES0	ET1	EX1	ET0	EX0	A8h
SADDR0									A9h
SADDR1									AAh
P3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	B0h
IP	–	PS1	PT2	PS0	PT1	PX1	PT0	PX0	B8h
SADEN0									B9h
SADEN1									BAh
SCON1	SM0/FE_1	SM1_1	SM2_1	REN_1	TB8_1	RB8_1	TI_1	RI_1	C0h
SBUF1									C1h
ROMSIZE	–	–	–	–	–	RMS2	RMS1	RMS0	C2h
PMR	CD1	CD0	SWB	–	XTOFF	ALEOFF	DME1	DME0	C4h
STATUS	PIP	HIP	LIP	XTUP	SPTA1	SPRA1	SPTA0	SPRA0	C5h
TA									C7h
T2CON	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$	C8h
T2MOD	–	–	–	–	–	–	T2OE	DCEN	C9h
RCAP2L									CAh
RCAP2H									CBh
TL2									CCh
TH2									CDh
PSW	CY	AC	F0	RS1	RS0	OV	F1	P	D0h
WDCON	SMOD_1	**POR**	EPFI	PFI	**WDIF**	WTRF	**EWT**	**RWT**	D8h
ACC									E0h
EIE	–	–	–	EWDI	EX5	EX4	EX3	EX2	E8h
B									F0h
EIP	–	–	–	PWDI	PX5	PX4	PX3	PX2	F8h

Shaded bits are Timed Access protected.

DS83C520/DS87C520 SPECIAL FUNCTION REGISTER RESET VALUES Table 4–8

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
P0	1	1	1	1	1	1	1	1	80h
SP	0	0	0	0	0	1	1	1	81h
DPL	0	0	0	0	0	0	0	0	82h
DPH	0	0	0	0	0	0	0	0	83h
DPL1	0	0	0	0	0	0	0	0	84h
DPH1	0	0	0	0	0	0	0	0	85h
DPS	0	0	0	0	0	0	0	0	86h
PCON	0	0	–	–	0	0	0	0	87h
TCON	0	0	0	0	0	0	0	0	88h
TMOD	0	0	0	0	0	0	0	0	89h
TL0	0	0	0	0	0	0	0	0	8Ah
TL1	0	0	0	0	0	0	0	0	8Bh
TH0	0	0	0	0	0	0	0	0	8Ch
TH1	0	0	0	0	0	0	0	0	8Dh
CKCON	0	0	0	0	0	0	0	1	8Eh
P1	1	1	1	1	1	1	1	1	90h
EXIF	0	0	0	0	SPECIAL	SPECIAL	SPECIAL	0	91h
SCON0	0	0	0	0	0	0	0	0	98h
SBUF0	0	0	0	0	0	0	0	0	99h
P2	1	1	1	1	1	1	1	1	A0h
IE	0	0	0	0	0	0	0	0	A8h
SADDR0	0	0	0	0	0	0	0	0	A9h
SADDR1	0	0	0	0	0	0	0	0	AAh
P3	1	1	1	1	1	1	1	1	B0h
IP	–	0	0	0	0	0	0	0	B8h
SADEN0	0	0	0	0	0	0	0	0	B9h
SADEN1	0	0	0	0	0	0	0	0	BAh
SCON1	0	0	0	0	0	0	0	0	C0h
SBUF1	0	0	0	0	0	0	0	0	C1h
ROMSIZE	–	–	–	–	–	1	0	1	C2h
PMR	0	1	0	–	0	0	0	0	C4h
STATUS	0	0	0	SPECIAL	0	0	0	0	C5h
TA	1	1	1	1	1	1	1	1	C7h
T2CON	0	0	0	0	0	0	0	0	C8h
T2MOD	–	–	–	–	–	–	0	0	C9h
RCAP2L	0	0	0	0	0	0	0	0	CAh
RCAP2H	0	0	0	0	0	0	0	0	CBh
TL2	0	0	0	0	0	0	0	0	CCh
TH2	0	0	0	0	0	0	0	0	CDh
PSW	0	0	0	0	0	0	0	0	D0h
WDCON	0	SPECIAL	0	SPECIAL	0	SPECIAL	SPECIAL	0	D8h
ACC	0	0	0	0	0	0	0	0	E0h
EIE	–	–	–	0	0	0	0	0	E8h
B	0	0	0	0	0	0	0	0	F0h
EIP	–	–	–	0	0	0	0	0	F8h

DS87C530 SPECIAL FUNCTION REGISTER LOCATIONS Table 4–9

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
P0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	80h
SP									81h
DPL									82h
DPH									83h
DPL1									84h
DPH1									85h
DPS	0	0	0	0	0	0	0	SEL	86h
PCON	SMOD_0	SMOD0	–	–	GF1	GF0	STOP	IDLE	87h
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	88h
TMOD	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0	89h
TL0									8Ah
TL1									8Bh
TH0									8Ch
TH1									8Dh
CKCON	WD1	WD0	T2M	T1M	T0M	MD2	MD1	MD0	8Eh
P1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	90h
EXIF	IE5	IE4	IE3	IE2	XT/$\overline{RG}$	RGMD	RGSL	**BGS**	91h
TRIM	**E4K**	**X12/$\overline{6}$**	**TRM2**	**$\overline{TRM2}$**	**TRM1**	**$\overline{TRM1}$**	**TRM0**	**$\overline{TRM0}$**	96h
SCON0	SM0/FE_0	SM1_0	SM2_0	REN_0	TB8_0	RB8_0	TI_0	RI_0	98h
SBUF0									99h
P2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	A0h
IE	EA	ES1	ET2	ES0	ET1	EX1	ET0	EX0	A8h
SADDR0									A9h
SADDR1									AAh
P3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	B0h
IP	–	PS1	PT2	PS0	PT1	PX1	PT0	PX0	B8h
SADEN0									B9h
SADEN1									BAh
SCON1	SM0/FE_1	SM1_1	SM2_1	REN_1	TB8_1	RB8_1	TI_1	RI_1	C0h
SBUF1									C1h
ROMSIZE	–	–	–	–	–	**RMS2**	**RMS1**	**RMS0**	C2h
PMR	CD1	CD0	SWB	–	XTOFF	ALEOFF	DME1	DME0	C4h
STATUS	PIP	HIP	LIP	XTUP	SPTA1	SPRA1	SPTA0	SPRA0	C5h
TA									C7h
T2CON	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$	C8h
T2MOD	–	–	–	–	–	–	T2OE	DCEN	C9h
RCAP2L									CAh
RCAP2H									CBh
TL2									CCh
TH2									CDh
PSW	CY	AC	F0	RS1	RS0	OV	F1	P	D0h
WDCON	SMOD_1	**POR**	EPFI	PFI	**WDIF**	WTRF	**EWT**	**RWT**	D8h
ACC									E0h
EIE	–	–	ERTCI	EWDI	EX5	EX4	EX3	EX2	E8h

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
B									F0h
RTASS									F2h
RTAS	0	0							F3h
RTAM	0	0							F4h
RTAH	0	0	0						F5h
EIP	–	–	PRTCI	PWDI	PX5	PX4	PX3	PX2	F8h
RTCC	SSCE	SCE	MCE	HCE	RTCRE	**RTCWE**	RTCIF	**RTCE**	F9h
RTCSS									FAh
RTCS	0	0							FBh
RTCM	0	0							FCh
RTCH									FDh
RTCD0									FEh
RTCD1									FFh

Shaded bits are Timed Access protected.

DS87C530 SPECIAL FUNCTION REGISTER RESET VALUES Table 4–10

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
P0	1	1	1	1	1	1	1	1	80h
SP	0	0	0	0	0	1	1	1	81h
DPL	0	0	0	0	0	0	0	0	82h
DPH	0	0	0	0	0	0	0	0	83h
DPL1	0	0	0	0	0	0	0	0	84h
DPH1	0	0	0	0	0	0	0	0	85h
DPS	0	0	0	0	0	0	0	0	86h
PCON	0	0	–	–	0	0	0	0	87h
TCON	0	0	0	0	0	0	0	0	88h
TMOD	0	0	0	0	0	0	0	0	89h
TL0	0	0	0	0	0	0	0	0	8Ah
TL1	0	0	0	0	0	0	0	0	8Bh
TH0	0	0	0	0	0	0	0	0	8Ch
TH1	0	0	0	0	0	0	0	0	8Dh
CKCON	0	0	0	0	0	0	0	1	8Eh
P1	1	1	1	1	1	1	1	1	90h
EXIF	0	0	0	0	SPECIAL	SPECIAL	SPECIAL	0	91h
TRIM	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	96h
SCON0	0	0	0	0	0	0	0	0	98h
SBUF0	0	0	0	0	0	0	0	0	99h
P2	1	1	1	1	1	1	1	1	A0h
IE	0	0	0	0	0	0	0	0	A8h
SADDR0	0	0	0	0	0	0	0	0	A9h
SADDR1	0	0	0	0	0	0	0	0	AAh
P3	1	1	1	1	1	1	1	1	B0h
IP	–	0	0	0	0	0	0	0	B8h
SADEN0	0	0	0	0	0	0	0	0	B9h
SADEN1	0	0	0	0	0	0	0	0	BAh

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
SCON1	0	0	0	0	0	0	0	0	C0h
SBUF1	0	0	0	0	0	0	0	0	C1h
ROMSIZE	–	–	–	–	–	1	0	1	C2h
PMR	0	1	0	–	0	0	0	0	C4h
STATUS	0	0	0	SPECIAL	0	0	0	0	C5h
TA	1	1	1	1	1	1	1	1	C7h
T2CON	0	0	0	0	0	0	0	0	C8h
T2MOD	–	–	–	–	–	–	0	0	C9h
RCAP2L	0	0	0	0	0	0	0	0	CAh
RCAP2H	0	0	0	0	0	0	0	0	CBh
TL2	0	0	0	0	0	0	0	0	CCh
TH2	0	0	0	0	0	0	0	0	CDh
PSW	0	0	0	0	0	0	0	0	D0h
WDCON	0	SPECIAL	0	SPECIAL	0	SPECIAL	SPECIAL	0	D8h
ACC	0	0	0	0	0	0	0	0	E0h
EIE	–	–	0	0	0	0	0	0	E8h
B	0	0	0	0	0	0	0	0	F0h
RTASS	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	F2h
RTAS	0	0	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	F3h
RTAM	0	0	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	F4h
RTAH	0	0	0	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	F5h
EIP	–	–	0	0	0	0	0	0	F8h
RTCC	SPECIAL	SPECIAL	SPECIAL	SPECIAL	0	0	SPECIAL	SPECIAL	F9h
RTCSS	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	FAh
RTCS	0	0	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	FBh
RTCM	0	0	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	FCh
RTCH	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	FDh
RTCD0	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	FEh
RTCD1	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	FFh

SPECIAL FUNCTION REGISTERS

Most of the unique features of the High Speed Microcontroller family are controlled by bits in special function registers (SFRs) located in unused locations in the 8051 SFR map. This allows for increased functionality while maintaining complete instruction set compatibility.

The description for each bit indicates its read and write access, as well as its state after a power on reset. Bits which are affected by a no–battery reset are also indicated. Note that many bits and registers are unique to specific devices, and their functions will vary between different members of the High Speed Microcontroller Family.

Port 0 (P0)

	7	6	5	4	3	2	1	0
SFR 80h	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
	RW–1	RW–1	RW–1	RW–1	RW–1	RW–1	RW–1	RW–1

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

P0.7–0
Bits 7–0

Port 0. This port functions as a multiplexed address/data bus during external memory access, and as a general purpose I/O port on devices which incorporate internal program memory. During external memory cycles, this port will contain the LSB of the address when ALE is high, and data when ALE is low. When used as a general purpose I/O, this port is open–drain and requires pull–ups. Writing a 1 to any pin of this port will place it in a high impedance mode, which is necessary if the pin is to be used as an input. Pull–ups are not required when used as a memory interface. The Port 0 latch does not exist on devices with no internal program memory, such as the DS80C310 and DS80C320.

Stack Pointer (SP)

	7	6	5	4	3	2	1	0
SFR 81h	SP.7	SP.6	SP.5	SP.4	SP.3	SP.2	SP.1	SP.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–1	RW–1	RW–1

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

SP.7–0
Bits 7–0

Stack Pointer. The stack pointer identifies the location where the stack will begin. The stack pointer is incremented before every PUSH operation. This register defaults to 07h after reset.

Data Pointer Low 0 (DPL)

	7	6	5	4	3	2	1	0
SFR 82h	DPL.7	DPL.6	DPL.5	DPL.4	DPL.3	DPL.2	DPL.1	DPL.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

DPL.7–0
Bits 7–0

Data Pointer Low 0. This register is the low byte of the standard 80C32 16–bit data pointer. DPL and DPH are used to point to non–scratchpad data RAM.

Data Pointer High 0 (DPH)

	7	6	5	4	3	2	1	0
SFR 83h	DPH.7	DPH.6	DPH.5	DPH.4	DPH.3	DPH.2	DPH.1	DPH.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

DPH.7–0
Bits 7–0

Data Pointer High 0. This register is the high byte of the standard 80C32 16–bit data pointer. DPL and DPH are used to point to non–scratchpad data RAM.

Data Pointer Low 1 (DPL1)

	7	6	5	4	3	2	1	0
SFR 84h	DPL1.7	DPL1.6	DPL1.5	DPL1.4	DPL1.3	DPL1.2	DPL1.1	DPL1.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

DPL1.7–0
Bits 7–0

Data Pointer Low 1. This register is the low byte of the auxiliary 16–bit data pointer. When the SEL bit (DPS.0) is set, DPL1 and DPH1 are used in place of DPL and DPH during DPTR operations.

Data Pointer High 1 (DPH1)

	7	6	5	4	3	2	1	0
SFR 85h	DPH1.7	DPH1.6	DPH1.5	DPH1.4	DPH1.3	DPH1.2	DPH1.1	DPH1.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

DPH1.7–0
Bits 7–0

Data Pointer High 1. This register is the high byte of the auxiliary 16–bit data pointer. When the SEL bit (DPS.0) is set, DPL1 and DPH1 are used in place of DPL and DPH during DPTR operations.

Data Pointer Select (DPS)

	7	6	5	4	3	2	1	0
SFR 86h	0	0	0	0	0	0	0	SEL
	R–0	R–0	R–0	R–0	R–0	R–0	R–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

Bits 7–1

Reserved. These bits will read 0.

SEL
Bit 0

Data Pointer Select. This bit selects the active data pointer.
0=Instructions that use the DPTR will use DPL and DPH.
1=Instructions that use the DPTR will use DPL1 and DPH1.

Power Control (PCON)

	7	6	5	4	3	2	1	0
SFR 87h	SMOD_0	SMOD0	–	–	GF1	GF0	STOP	IDLE
	RW–0	RW–0			RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

SMOD_0
Bit 7

Serial Port 0 Baud Rate Doubler Enable. This bit enables/disables the serial baud rate doubling function for Serial Port 0.
0=Serial Port 0 baud rate will be that defined by baud rate generation equation.
1=Serial Port 0 baud rate will be double that defined by baud rate generation equation.

SMOD0
Bit 6

Framing Error Detection Enable. This bit selects function of the SCON0.7 and SCON1.7 bits.
0=SCON0.7 and SCON1.7 control the SM0 function defined for the SCON0 and SCON1 registers.
1=SCON0.7 and SCON1.7 are converted to the Framing Error (FE) flag for the respective Serial Port.

Bits 5–4

Reserved. Read data is indeterminate.

GF1
Bit 3

General Purpose User Flag 1. This is a bit–addressable, general purpose flag for software control.

GF0
Bit 2

General Purpose User Flag 0. This is a bit–addressable, general purpose flag for software control.

STOP
Bit 1

Stop Mode Select. Setting this bit will stop program execution, halt the CPU oscillator, and internal timers, and place the CPU in a low–power mode. This bit will always be read as a 0. Setting this bit while the Idle bit is set will place the device in an undefined state.

IDLE
Bit 0

Idle Mode Select. Setting this bit will stop program execution but leave the CPU oscillator, timers, serial ports, and interrupts active. This bit will always be read as a 0.

Timer/Counter Control (TCON)

	7	6	5	4	3	2	1	0
SFR 88h	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

TF1
Bit 7

Timer 1 Overflow Flag. This bit indicates when Timer 1 overflows its maximum count as defined by the current mode. This bit can be cleared by software and is automatically cleared when the CPU vectors to the Timer 1 interrupt service routine.
0=No Timer 1 overflow has been detected.
1=Timer 1 has overflowed its maximum count.

TR1
Bit 6

Timer 1 Run Control. This bit enables/disables the operation of Timer 1. Halting this timer will preserve the current count in TH1, TL1.
0=Timer 1 is halted.
1=Timer 1 is enabled.

TF0
Bit 5

Timer 0 Overflow Flag. This bit indicates when Timer 0 overflows its maximum count as defined by the current mode. This bit can be cleared by software and is automatically cleared when the CPU vectors to the Timer 0 interrupt service routine or by software.
0=No Timer 0 overflow has been detected.
1=Timer 0 has overflowed its maximum count.

TR0
Bit 4

Timer 0 Run Control. This bit enables/disables the operation of Timer 0. Halting this timer will preserve the current count in TH0, TL0.
0=Timer 0 is halted.
1=Timer 0 is enabled.

IE1
Bit 3

Interrupt 1 Edge Detect. This bit is set when an edge/level of the type defined by IT1 is detected. If IT1=1, this bit will remain set until cleared in software or the start of the External Interrupt 1 service routine. If IT1=0, this bit will inversely reflect the state of the $\overline{\text{INT1}}$ pin.

IT1
Bit 2

Interrupt 1 Type Select. This bit selects whether the $\overline{\text{INT1}}$ pin will detect edge or level triggered interrupts.
0=$\overline{\text{INT1}}$ is level triggered.
1=$\overline{\text{INT1}}$ is edge triggered.

IE0
Bit 1

Interrupt 0 Edge Detect. This bit is set when an edge/level of the type defined by IT0 is detected. If IT0=1, this bit will remain set until cleared in software or the start of the External Interrupt 0 service routine. If IT0=0, this bit will inversely reflect the state of the $\overline{\text{INT0}}$ pin.

IT0
Bit 0

Interrupt 0 Type Select. This bit selects whether the $\overline{\text{INT0}}$ pin will detect edge or level triggered interrupts.
0=$\overline{\text{INT0}}$ is level triggered.
1=$\overline{\text{INT0}}$ is edge triggered.

Timer Mode Control (TMOD)

	7	6	5	4	3	2	1	0
SFR 89h	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

GATE
Bit 7

Timer 1 Gate Control. This bit enables/disables the ability of Timer 1 to increment.
0=Timer 1 will clock when TR1=1, regardless of the state of $\overline{INT1}$.
1=Timer 1 will clock only when TR1=1 and $\overline{INT1}$=1.

C/T
Bit 6

Timer 1 Counter/Timer Select.
0=Timer is incremented by internal clocks.
1=Timer is incremented by pulses on T1 when TR1 (TCON.6) is 1.

M1, M0
Bits 5–4

Timer 1 Mode Select. These bits select the operating mode of Timer 1.

M1	M0	Mode
0	0	Mode 0: 8 bits with 5–bit prescale
0	1	Mode 1: 16 bits
1	0	Mode 2: 8 bits with auto–reload
1	1	Mode 3: Timer 1 is halted, but holds its count.

GATE
Bit 3

Timer 0 Gate Control. This bit enables/disables the ability of Timer 0 to increment.
0=Timer 0 will clock when TR0=1, regardless of the state of $\overline{INT0}$.
1=Timer 0 will clock only when TR0=1 and $\overline{INT0}$=1.

C/$\overline{T}$
Bit 2

Timer 0 Counter/Timer Select.
0=Timer is incremented by internal clocks.
1=Timer is incremented by pulses on T0 when TR0 (TCON.4) is 1.

M1, M0
Bits 1–0

Timer 0 Mode Select. These bits select the operating mode of Timer 0. When timer 0 is in mode 3, TL0 is started/stopped by TR0 and TH0 is started/stopped by TR1. Run control for timer 1 is then provided via the timer 1 mode selection.

M1	M0	Mode
0	0	Mode 0: 8 bits with 5–bit prescale
0	1	Mode 1: 16 bits
1	0	Mode 2: 8 bits with auto–reload
1	1	Mode 3: Timer 0 is two 8 bit counters

Timer 0 LSB (TL0)

	7	6	5	4	3	2	1	0
SFR 8Ah	TL0.7	TL0.6	TL0.5	TL0.4	TL0.3	TL0.2	TL0.1	TL0.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

TL0.7–0 **Timer 0 LSB.** This register contains the least significant byte of Timer 0.
Bits 7–0

Timer 1 LSB (TL1)

	7	6	5	4	3	2	1	0
SFR 8Bh	TL1.7	TL1.6	TL1.5	TL1.4	TL1.3	TL1.2	TL1.1	TL1.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

TL1.7–0 **Timer 1 LSB.** This register contains the least significant byte of Timer 1.
Bits 7–0

Timer 0 MSB (TH0)

	7	6	5	4	3	2	1	0
SFR 8Ch	TH0.7	TH0.6	TH0.5	TH0.4	TH0.3	TH0.2	TH0.1	TH0.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

TH0.7–0 **Timer 0 MSB.** This register contains the most significant byte of Timer 0.
Bits 7–0

Timer 1 MSB (TH1)

	7	6	5	4	3	2	1	0
SFR 8Dh	TH1.7	TH1.6	TH1.5	TH1.4	TH1.3	TH1.2	TH1.1	TH1.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

TH1.7–0 **Timer 1 MSB.** This register contains the most significant byte of Timer 1.
Bits 7–0

Clock Control (CKCON)

	7	6	5	4	3	2	1	0
SFR 8Eh	WD1	WD0	T2M	T1M	T0M	MD2	MD1	MD0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–1

R=Unrestricted Read, W= Unrestricted Write, –n=Value after Reset

WD1, WD0
Bits 7–6

Watchdog Timer Mode Select 1–0. These bits determine the watchdog timer time–out period. The timer divides the crystal frequency by a programmable value as shown below. The divider value is expressed in clock (crystal) cycles. The use of PMM1 or PMM2 will further divide the clock cycle count by either 16 or 256, respectively. Note that the reset time–out is 512 clocks longer than the interrupt, regardless of whether the interrupt is enabled.

WD1	WD0	Interrupt Divider	Reset Divider
0	0	2^{17}	2^{17} + 512
0	1	2^{20}	2^{20} + 512
1	0	2^{23}	2^{23} + 512
1	1	2^{26}	2^{26} + 512

T2M
Bit 5

Timer 2 Clock Select. This bit controls the division of the system clock that drives Timer 2. This bit has no effect when the timer is in baud rate generator or clock output modes. Clearing this bit to 0 maintains 80C32 compatibility. This bit has no effect on instruction cycle timing.
0=Timer 2 uses a divide by 12 of the crystal frequency.
1=Timer 2 uses a divide by 4 of the crystal frequency.

T1M
Bit 4

Timer 1 Clock Select. This bit controls the division of the system clock that drives Timer 1. Clearing this bit to 0 maintains 80C32 compatibility. This bit has no effect on instruction cycle timing.
0=Timer 1 uses a divide by 12 of the crystal frequency.
1=Timer 1 uses a divide by 4 of the crystal frequency.

T0M
Bit 3

Timer 0 Clock Select. This bit controls the division of the system clock that drives Timer 0. Clearing this bit to 0 maintains 80C32 compatibility. This bit has no effect on instruction cycle timing.
0=Timer 0 uses a divide by 12 of the crystal frequency.
1=Timer 0 uses a divide by 4 of the crystal frequency.

MD2, MD1, MD0
Bits 2–0

Stretch MOVX Select 2–0. These bits select the time by which external MOVX cycles are to be stretched. This allows slower memory or peripherals to be accessed without using ports or manual software intervention. The RD or WR strobe will be stretched by the specified interval, which will be transparent to the software except for the increased time to execute the MOVX instruction. All internal MOVX instructions on devices containing MOVX SRAM are performed at the 2 machine cycle rate.

MD2	MD1	MD0	Stretch Value	MOVX Duration
0	0	0	0	2 Machine Cycles
0	0	1	1	3 Machine Cycles (reset default)
0	1	0	2	4 Machine Cycles
0	1	1	3	5 Machine Cycles
1	0	0	4	6 Machine Cycles
1	0	1	5	7 Machine Cycles
1	1	0	6	8 Machine Cycles
1	1	1	7	9 Machine Cycles

Port 1 (P1)

	7	6	5	4	3	2	1	0
SFR 90h	P1.7 $\overline{\text{INT5}}$	P1.6 INT4	P1.5 $\overline{\text{INT3}}$	P1.4 INT2	P1.3 TXD1	P1.2 RXD1	P1.1 T2EX	P1.0 T2
	RW–1	RW–1	RW–1	RW–1	RW–1	RW–1	RW–1	RW–1

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

P1.7–0
Bits 7–0

General Purpose I/O Port 1. This register functions as a general purpose I/O port. In addition, all the pins have an alternative function listed below. P1.2–7 contain functions that are new to the 80C32 architecture. The timer 2 functions on pins P1.1–0 are available on the 80C32, but not the 80C31. Each of the functions is controlled by several other SFRs. The associated Port 1 latch bit must contain a logic one before the pin can be used in its alternate function capacity.

$\overline{\text{INT5}}$
Bit 7

External Interrupt 5. A falling edge on this pin will cause an external interrupt 5 if enabled.

INT4
Bit 6

External Interrupt 4. A rising edge on this pin will cause an external interrupt 4 if enabled.

$\overline{\text{INT3}}$
Bit 5

External Interrupt 3. A falling edge on this pin will cause an external interrupt 3 if enabled.

INT2
Bit 4

External Interrupt 2. A rising edge on this pin will cause an external interrupt 2 if enabled.

TXD1
Bit 3

Serial Port 1 Transmit. This pin transmits the serial port 1 data in serial port modes 1,2,3 and emits the synchronizing clock in serial port mode 0.

RXD1
Bit 2

Serial Port 1 Receive. This pin receives the serial port 1 data in serial port modes 1,2,3 and is a bi–directional data transfer pin in serial port mode 0.

T2EX
Bit 1

Timer 2 Capture/Reload Trigger. A 1 to 0 transition on this pin will cause the value in the T2 registers to be transferred into the capture registers if enabled by EXEN2 (T2CON.3). When in auto–reload mode, a 1 to 0 transition on this pin will reload the timer 2 registers with the value in RCAP2L and RCAP2H if enabled by EXEN2 (T2CON.3).

T2
Bit 0

Timer 2 External Input. A 1 to 0 transition on this pin will cause timer 2 to increment or decrement depending on the timer configuration.

External Interrupt Flag (EXIF)

	7	6	5	4	3	2	1	0
SFR 91h	IE5	IE4	IE3	IE2	XT/$\overline{\text{RG}}$	RGMD	RGSL	BGS
	RW–0	RW–0	RW–0	RW–0	RW–*	R–*	RW–*	RT–0

R=Unrestricted Read, W=Unrestricted Write, T=Timed Access Write Only, –n=Value after Reset, *=See description

IE5
Bit 7

External Interrupt 5 Flag. This bit will be set when a falling edge is detected on $\overline{\text{INT5}}$. This bit must be cleared manually by software. Setting this bit in software will cause an interrupt if enabled.

IE4
Bit 6

External Interrupt 4 Flag. This bit will be set when a rising edge is detected on INT4. This bit must be cleared manually by software. Setting this bit in software will cause an interrupt if enabled.

IE3
Bit 5

External Interrupt 3 Flag. This bit will be set when a falling edge is detected on $\overline{\text{INT3}}$. This bit must be cleared manually by software. Setting this bit in software will cause an interrupt if enabled.

IE2
Bit 4

External Interrupt 2 Flag. This bit will be set when a rising edge is detected on INT2. This bit must be cleared manually by software. Setting this bit in software will cause an interrupt if enabled.

XT/$\overline{\text{RG}}$
Bit 3

Crystal/Ring Source Select. This bit selects the crystal oscillator or ring oscillator as the desired clock source. This bit will be the inverse of RGMD except during the crystal warm–up period when executing a ring oscillator resume from Stop. XTUP (STATUS.4) must be set to 1 and XTOFF (PMR.3) must be cleared to 0 before this bit can be set. Attempts to modify this bit when these conditions are not met will be ignored. This bit must be cleared before XTOFF can be set to 1. This bit is set to 1 after a power–on reset, and unchanged by all other forms of reset. This bit is not used on the DS80C310 or DS80C320 and will be 1 when read.
0=The ring oscillator is selected as the clock source. This setting is unaffected by XTUP (STATUS.4) and XTOFF (PMR.3).
1=The crystal oscillator is selected as the clock source. This setting is invalid unless XTUP=1 and XTOFF=0.

RGMD
Bit 2

Ring Mode Status. This bit indicates the current clock source for the device. This bit is cleared to 0 after a power–on reset, and unchanged by all other forms of reset. The state of this bit will be undefined on devices which do not incorporate a ring oscillator.
0=Device is operating from the external crystal or oscillator.
1=Device is operating from the ring oscillator.

RGSL
Bit 1

Ring Oscillator Select. This bit selects the clock source following a resume from Stop mode. Using the ring oscillator to resume from Stop mode allows almost instantaneous start–up. This bit is cleared to 0 after a power–on reset, and unchanged by all other forms of reset. The state of this bit will be undefined on devices which do not incorporate a ring oscillator.
0=The device will hold operation until the crystal oscillator has warmed–up.
1=The device will begin operating from the ring oscillator, and when the crys-

tal warm–up is complete, will switch to the clock source indicated by the XT/$\overline{RG}$ bit.

BGS
Bit 0

Band–gap Select. This bit enables/disables the band–gap reference during Stop mode. Disabling the band–gap reference provides significant power savings in Stop mode, but sacrifices the ability to perform a power fail interrupt or power–fail reset while stopped. This bit can only be modified with a Timed Access procedure. The state of this bit will be undefined on devices which do not incorporate a band–gap reference.
0=The band–gap reference is disabled in Stop mode but will function during normal operation.
1=The band–gap reference will operate in Stop mode.

RTC Trim Register (TRIM)

	7	6	5	4	3	2	1	0
SFR 96h	E4K	X12/$\overline{6}$	TRM2	$\overline{TRM2}$	TRM1	$\overline{TRM1}$	TRM0	$\overline{TRM0}$
	RT–*	RT–*	RT–*	RT–*	RT–*	RT–*	RT–*	RT–*

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset, *=See description

E4K
Bit 7

External 4096 Hz RTC Signal Enable. This bit enables the output of a 4096 Hz signal on pin P1.7 derived from the RTC. Setting this bit overrides any other function of the pin. It is used for adjusting the frequency of the 32.768 RTC crystal oscillator using the trim bits. This bit is cleared to 0 after any reset, including a no–battery reset.
0=Calibration function disabled. P1.7 pin will function per the normal pin description.
1=4096 Hz signal output on P1.7

X12/$\overline{6}$
Bit 6

RTC Crystal Capacitance Select. This bit selects the internal loading capacitance of the RTC crystal amplifier. This bit is set to 1 after a no–battery reset, and unchanged by all other forms of reset.
0=RTC loading is set for 6 pF crystal.
1=RTC loading is set for 12.5 pF crystal.

TRM2
Bit 5

RTC Trim Bit 2. This bit controls the relative adjustment of the RTC internal capacitance. It is used to calibrate the RTC oscillator frequency. This bit is set to 1 after a no–battery reset, and unchanged by all other forms of reset.

$\overline{TRM2}$
Bit 4

RTC Inverted Trim Bit 2. This bit must always be set to the complement of the TRM2 bit. Incorrectly writing this bit will default bits TRIM.7, TRIM.5–0 to their no–battery reset value. This bit is cleared to 0 after a no–battery reset, and unchanged by all other forms of reset.

TRM1
Bit 3

RTC Trim Bit 1. This bit controls the relative adjustment of the RTC internal capacitance. It is used to calibrate the RTC oscillator frequency. This bit is set to 0 after a no–battery reset, and unchanged by all other forms of reset.

$\overline{TRM1}$
Bit 2

RTC Inverted Trim Bit 1. This bit must always be set to the complement of the TRM1 bit. Incorrectly writing this bit will default bits TRIM.7, TRIM.5–0 to their no–battery reset value. This bit is cleared to 1 after a no–battery reset, and unchanged by all other forms of reset.

TRM0
Bit 1

RTC Trim Bit 0. This bit controls the relative adjustment of the RTC internal capacitance. It is used to calibrate the RTC oscillator frequency. This bit is set to 0 after a no–battery reset, and unchanged by all other forms of reset.

$\overline{\text{TRM0}}$
Bit 0

RTC Inverted Trim Bit 0. This bit must always be set to the complement of the TRM0 bit. Incorrectly writing this bit will default bits TRIM.7, TRIM.5–0 to their no–battery reset value. This bit is cleared to 1 after a no–battery reset, and unchanged by all other forms of reset.

Serial Port 0 Control (SCON0)

	7	6	5	4	3	2	1	0
SFR 98h	SM0/FE_0	SM1_0	SM2_0	REN_0	TB8_0	RB8_0	TI_0	RI_0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

SM0–2
Bits 7–5

Serial Port 0 Mode. These bits control the mode of serial port 0. In addition, the SM0 and SM2_0 bits have secondary functions as shown below.

SM0	SM1	SM2	MODE	FUNCTION	LENGTH	PERIOD
0	0	0	0	Synchronous	8 bits	12 t_{CLK}
0	0	1	0	Synchronous	8 bits	4 t_{CLK}
0	1	x	1	Asynchronous	10 bits	Timer 1 or 2 baud rate equation
1	0	0	2	Asynchronous	11 bits	64 t_{CLK} (SMOD=0) 32 t_{CLK} (SMOD=1)
1	0	1	2	Asynchronous w/ Multiprocessor communication	11 bits	64 t_{CLK} (SMOD=0) 32 t_{CLK} (SMOD=1)
1	1	0	3	Asynchronous	11 bits	Timer 1 or 2 baud rate equation
1	1	1	3	Asynchronous w/ Multiprocessor communication	11 bits	Timer 1 or 2 baud rate equation

SM0/FE_0
Bit 7

Framing Error Flag. When SMOD0 (PCON.6)=0, this bit (SM0) is used to select the mode for serial port 0. When SMOD0 (PCON.6)=1, this bit (FE) will be set upon detection of an invalid stop bit. When used as FE, this bit must be cleared in software. Once the SMOD0 bit is set, modifications to this bit will not affect the serial port mode settings. Although accessed from the same register, internally the data for bits SM0 and FE are stored in different locations.

SM1_0
Bit 6

No alternate function.

SM2_0
Bit 5

Multiple CPU Communications. The function of this bit is dependent on the serial port 0 mode.
Mode 0: Selects 12 t_{CLK} or 4 t_{CLK} period for synchronous serial port 0 data transfers.
Mode 1: When set, reception is ignored (RI_0 is not set) if invalid stop bit received.
Mode 2/3: When this bit is set, multiprocessor communications are enabled

in modes 2 and 3. This will prevent the RI_0 bit from being set, and an interrupt being asserted, if the 9th bit received is not 1.

REN_0
Bit 4

Receive Enable. This bit enables/disables the serial port 0 receiver shift register.
0=Serial port 0 reception disabled.
1=Serial port 0 receiver enabled (modes 1, 2, 3). Initiate synchronous reception (mode 0).

TB8_0
Bit 3

9th Transmission Bit State. This bit defines the state of the 9th transmission bit in serial port 0 modes 2 and 3.

RB8_0
Bit 2

9th Received Bit State. This bit identifies the state of the 9th reception bit of received data in serial port 0 modes 2 and 3. In serial port mode 1, when SM2_0=0, RB8_0 is the state of the stop bit. RB8_0 is not used in mode 0.

TI_0
Bit 1

Transmitter Interrupt Flag. This bit indicates that data in the serial port 0 buffer has been completely shifted out. In serial port mode 0, TI_0 is set at the end of the 8th data bit. In all other modes, this bit is set at the end of the last data bit. This bit must be manually cleared by software.

RI_0
Bit 0

Receiver Interrupt Flag. This bit indicates that a byte of data has been received in the serial port 0 buffer. In serial port mode 0, RI_0 is set at the end of the 8th bit. In serial port mode 1, RI_0 is set after the last sample of the incoming stop bit subject to the state of SM2_0. In modes 2 and 3, RI_0 is set after the last sample of RB8_0. This bit must be manually cleared by software.

Serial Data Buffer 0 (SBUF0)

	7	6	5	4	3	2	1	0
SFR 99h	SBUF0.7	SBUF0.6	SBUF0.5	SBUF0.4	SBUF0.3	SBUF0.2	SBUF0.1	SBUF0.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

SBUF0.7–0
Bits 7–0

Serial Data Buffer 0. Data for serial port 0 is read from or written to this location. The serial transmit and receive buffers are separate registers, but both are addressed at this location.

Port 2 (P2)

	7	6	5	4	3	2	1	0
SFR A0h	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
	RW–1	RW–1	RW–1	RW–1	RW–1	RW–1	RW–1	RW–1

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

P2.7–0
Bits 7–0

Port 2. This port functions as an address bus during external memory access, and as a general purpose I/O port on devices which incorporate internal program memory. During external memory cycles, this port will contain the MSB of the address. The Port 2 latch does not control general purpose I/O pins on the DS80C310 and DS80C320, but is still used to hold the address MSB during register–indirect data memory operations such as MOVX A, @R1.

Interrupt Enable (IE)

	7	6	5	4	3	2	1	0
SFR A8h	EA	ES1	ET2	ES0	ET1	EX1	ET0	EX0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

EA
Bit 7

Global Interrupt Enable. This bit controls the global masking of all interrupts except Power–Fail Interrupt, which is enabled by the EPFI bit (WDCON.5).
0=Disable all interrupt sources. This bit overrides individual interrupt mask settings.
1=Enable all individual interrupt masks. Individual interrupts will occur if enabled.

ES1
Bit 6

Enable Serial Port 1 Interrupt. This bit controls the masking of the serial port 1 interrupt.
0=Disable all serial port 1 interrupts.
1=Enable interrupt requests generated by the RI_1 (SCON1.0) or TI_1 (SCON1.1) flags.

ET2
Bit 5

Enable Timer 2 Interrupt. This bit controls the masking of the Timer 2 interrupt.
0=Disable all Timer 2 interrupts.
1=Enable interrupt requests generated by the TF2 flag (T2CON.7).

ES0
Bit 4

Enable Serial port 0 Interrupt. This bit controls the masking of the serial port 0 interrupt.
0=Disable all serial port 0 interrupts.
1=Enable interrupt requests generated by the RI_0 (SCON0.0) or TI_0 (SCON0.1) flags.

ET1
Bit 3

Enable Timer 1 Interrupt. This bit controls the masking of the Timer 1 interrupt.
0=Disable all Timer 1 interrupts.
1=Enable interrupt requests generated by the TF1 flag (TCON.7).

EX1
Bit 2

Enable External Interrupt 1. This bit controls the masking of external interrupt 1.
0=Disable external interrupt 1.
1=Enable interrupt requests generated by the $\overline{\text{INT1}}$ pin.

ET0
Bit 1

Enable Timer 0 Interrupt. This bit controls the masking of the Timer 0 interrupt.
0=Disable all Timer 0 interrupts.
1=Enable interrupt requests generated by the TF0 flag (TCON.5).

EX0
Bit 0

Enable External Interrupt 0. This bit controls the masking of external interrupt 0.
0=Disable external interrupt 0.
1=Enable interrupt requests generated by the $\overline{\text{INT0}}$ pin.

Slave Address Register 0 (SADDR0)

	7	6	5	4	3	2	1	0
SFR A9h	SADDR0.7	SADDR0.6	SADDR0.5	SADDR0.4	SADDR0.3	SADDR0.2	SADDR0.1	SADDR0.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

SADDR0.7–0
Bits 7–0

Slave Address Register 0. This register is programmed with the given or broadcast address assigned to serial port 0.

Slave Address Register 1 (SADDR1)

	7	6	5	4	3	2	1	0
SFR AAh	SADDR1.7	SADDR1.6	SADDR1.5	SADDR1.4	SADDR1.3	SADDR1.2	SADDR1.1	SADDR1.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

SADDR1.7–0
Bits 7–0

Slave Address Register 1. This register is programmed with the given or broadcast address assigned to serial port 1.

Port 3 (P3)

	7	6	5	4	3	2	1	0
SFR B0h	P3.7 $\overline{RD}$	P3.6 $\overline{WR}$	P3.5 T1	P3.4 T0	P3.3 $\overline{INT1}$	P3.2 $\overline{INT0}$	P3.1 TXD0	P3.0 RXD0
	RW–1	RW–1	RW–1	RW–1	RW–1	RW–1	RW–1	RW–1

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

P3.7–0
Bits 7–0

General Purpose I/O Port 3. This register functions as a general purpose I/O port. In addition, all the pins have an alternative function listed below. Each of the functions is controlled by several other SFRs. The associated Port 1 latch bit must contain a logic one before the pin can be used in its alternate function capacity.

$\overline{RD}$
Bit 7

External Data Memory Read Strobe. This pin provides an active low read strobe to an external memory device.

$\overline{WR}$
Bit 6

External Data Memory Write Strobe. This pin provides an active low write strobe to an external memory device.

T1
Bit 5

Timer/Counter 1 External Input. A 1 to 0 transition on this pin will increment timer 1.

T0
Bit 4

Timer/Counter 0 External Input. A 1 to 0 transition on this pin will increment timer 0.

$\overline{INT1}$
Bit 3

External Interrupt 1. A falling edge/ low level on this pin will cause an external interrupt 1 if enabled.

$\overline{INT0}$
Bit 2

External Interrupt 0. A falling edge/ low level on this pin will cause an external interrupt 0 if enabled.

TXD0
Bit 1

Serial Port 0 Transmit. This pin transmits the serial port 0 data in serial port modes 1,2,3 and emits the synchronizing clock in serial port mode 0.

RXD0
Bit 0

Serial Port 0 Receive. This pin receives the serial port 0 data in serial port modes 1,2,3 and is a bi–directional data transfer pin in serial port mode 0.

Interrupt Priority (IP)

	7	6	5	4	3	2	1	0
SFR B8h	–	PS1	PT2	PS0	PT1	PX1	PT0	PX0
		RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

Bit 7

Reserved. Read data is indeterminate.

PS1
Bit 6

Serial Port 1 Interrupt. This bit controls the priority of the serial port 1 interrupt.
0=Serial port 1 priority is determined by the natural priority order.
1=Serial port 1 is a high priority interrupt.

PT2
Bit 5

Timer 2 Interrupt. This bit controls the priority of the Timer 2 interrupt.
0=Timer 2 priority is determined by the natural priority order.
1=Timer 2 is a high priority interrupt.

PS0
Bit 4

Serial port 0 Interrupt. This bit controls the priority of the serial port 0 interrupt.
0=Serial port 0 priority is determined by the natural priority order.
1=Serial port 0 is a high priority interrupt.

PT1
Bit 3

Timer 1 Interrupt. This bit controls the priority of the Timer 1 interrupt.
0=Timer 1 priority is determined by the natural priority order.
1=Timer 1 is a high priority interrupt.

PX1
Bit 2

External Interrupt 1. This bit controls the priority of external interrupt 1.
0=External interrupt 1 priority is determined by the natural priority order.
1=External interrupt 1 is a high priority interrupt.

PT0
Bit 1

Timer 0 Interrupt. This bit controls the priority of the Timer 0 interrupt.
0=Timer 0 priority is determined by the natural priority order.
1=Timer 0 is a high priority interrupt.

PX0
Bit 0

External Interrupt 0. This bit controls the priority of external interrupt 0.
0=External interrupt 0 priority is determined by the natural priority order.
1=External interrupt 0 is a high priority interrupt.

Slave Address Mask Enable Register 0 (SADEN0)

	7	6	5	4	3	2	1	0
SFR B9h	SADEN0.7	SADEN0.6	SADEN0.5	SADEN0.4	SADEN0.3	SADEN0.2	SADEN0.1	SADEN0.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

SADEN0.7–0
Bits 7–0

Slave Address Mask Enable Register 0. This register functions as a mask when comparing serial port 0 addresses for automatic address recognition. When a bit in this register is set, the corresponding bit location in the SADDR0 register will be exactly compared with the incoming serial port 0 data to determine if a receiver interrupt should be generated. When a bit in this register is cleared, the corresponding bit in the SADDR0 register becomes a don't care and is not compared against the incoming data. All incoming data will generate a receiver interrupt when this register is cleared.

Slave Address Mask Enable Register 1 (SADEN1)

	7	6	5	4	3	2	1	0
SFR BAh	SADEN1.7	SADEN1.6	SADEN1.5	SADEN1.4	SADEN1.3	SADEN1.2	SADEN1.1	SADEN1.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

SADEN1.7–0
Bits 7–0

Slave Address Mask Enable Register 1. This register functions as a mask when comparing serial port 1 addresses for automatic address recognition.

When a bit in this register is set, the corresponding bit location in the SADDR1 register will be exactly compared with the incoming serial port 1 data to determine if a receiver interrupt should be generated. When a bit in this register is cleared, the corresponding bit in the SADDR1 register becomes a don't care and is not compared against the incoming data. All incoming data will generate a receiver interrupt when this register is cleared.

Serial Port 1 Control (SCON1)

	7	6	5	4	3	2	1	0
SFR C0h	SM0/FE_1	SM1_1	SM2_1	REN_1	TB8_1	RB8_1	TI_1	RI_1
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

SM0–2,
Bits 7–5

Serial Port 1 Mode. These bits control the mode of serial port 1 as shown below. In addition, the SM0 and SM2 bits have secondary functions as shown below.

SM0	SM1	SM2	MODE	FUNCTION	LENGTH	PERIOD
0	0	0	0	Synchronous	8 bits	12 t_{CLK}
0	0	1	0	Synchronous	8 bits	4 t_{CLK}
0	1	x	1	Asynchronous	10 bits	Timer 1 baud rate equation
1	0	0	2	Asynchronous	11 bits	64 t_{CLK} (SMOD=0) 32 t_{CLK} (SMOD=1)
1	0	1	2	Asynchronous w/ Multiprocessor communication	11 bits	64 t_{CLK} (SMOD=0) 32 t_{CLK} (SMOD=1)
1	1	0	3	Asynchronous	11 bits	Timer 1 baud rate equation
1	1	1	3	Asynchronous w/ Multiprocessor communication	11 bits	Timer 1 baud rate equation

SM0/FE_1
Bit 7

Framing Error Flag. When SMOD0 (PCON.6)=0, this bit (SM0) is used to select the mode for serial port 1. When SMOD0 (PCON.6)=1, this bit (FE) will be set upon detection of an invalid stop bit. When used as FE, this bit must be cleared in software. Once the SMOD0 bit is set, modifications to this bit will not affect the serial port mode settings. Although accessed from the same register, internally the data for bits SM0 and FE are stored in different locations.

SM1_1
Bit 6

No alternate function.

SM2_1
Bit 5

Multiple CPU Communications. The function of this bit is dependent on the serial port 1 mode.
Mode 0: Selects 12 t_{CLK} or 4 t_{CLK} period for synchronous serial port 1 data transfers.
Mode 1: When this bit is set, reception is ignored (RI_1 is not set) if invalid stop bit received.
Mode 2/3: When this bit is set, multiprocessor communications are enabled

in modes 2 and 3. This will prevent the RI_1 bit from being set, and an interrupt being asserted, if the 9th bit received is not 1.

REN_1
Bit 4

Receive Enable. This bit enables/disables the serial port 1 receiver shift register.
0=Serial port 1 reception disabled.
1=Serial port 1 receiver enabled (modes 1, 2, 3). Initiate synchronous reception (mode 0).

TB8_1
Bit 3

9th Transmission Bit State. This bit defines the state of the 9th transmission bit in serial port 1 modes 2 and 3.

RB8_1
Bit 2

9th Received Bit State. This bit identifies the state of the 9th reception bit of received data in serial port 1 modes 2 and 3. In serial port mode 1, when SM2_1=0, RB8_1 is the state of the stop bit. RB8_1 is not used in mode 0.

TI_1
Bit 1

Transmitter Interrupt Flag. This bit indicates that data in the serial port 1 buffer has been completely shifted out. In serial port mode 0, TI_1 is set at the end of the 8th data bit. In all other modes, this bit is set at the end of the last data bit. This bit must be manually cleared by software.

RI_1
Bit 0

Receiver Interrupt Flag. This bit indicates that a byte of data has been received in the serial port 1 buffer. In serial port mode 1, RI_1 is set at the end of the 8th bit. In serial port mode 1, RI_1 is set after the last sample of the incoming stop bit subject to the state of SM2_1. In modes 2 and 3, RI_1 is set after the last sample of RB8_1. This bit must be manually cleared by software.

Serial Data Buffer 1 (SBUF1)

	7	6	5	4	3	2	1	0
SFR C1h	SBUF1.7	SBUF1.6	SBUF1.5	SBUF1.4	SBUF1.3	SBUF1.2	SBUF1.1	SBUF1.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

SBUF1.7–0
Bits 7–0

Serial Data Buffer 1. Data for serial port 1 is read from or written to this location. The serial transmit and receive buffers are separate registers, but both are addressed at this location.

ROM Size Select (ROMSIZE)

	7	6	5	4	3	2	1	0
SFR C2h	–	–	–	–	–	RS2	RS1	RS0
						RT–1	RT–0	RT–1

R=Unrestricted Read, T=Timed Access Write Only, –n=Value after Reset

Bits 7–3

These bits are reserved. Read data is indeterminate.

ROMSIZE.2–0
Bits 2–0

ROM Size Select 2–0. This register is used to select the maximum on–chip decoded address for ROM. Care must be taken that the memory location of the current program counter will be valid both before and after modification.

These bits can only be modified using a timed access procedure. The $\overline{EA}$ pin will override the function of these bits when asserted, forcing the device to access external program memory only. Configuring this register to a setting that exceeds the maximum amount of internal memory may corrupt device operation. These bits will default on reset to the maximum amount of internal program memory (i.e. 16K for DS87C520).

RS2	RS1	RS0	MAXIMUM ON–CHIP ROM ADDRESS
0	0	0	0KB/Disable on–chip ROM
0	0	1	1KB/03FFh
0	1	0	2KB/07FFh
0	1	1	4KB/0FFFh
1	0	0	8KB/1FFFh
1	0	1	16KB/3FFFh
1	1	0	32KB/7FFFh
1	1	1	64KB/FFFFh

Power Management Register (PMR)

	7	6	5	4	3	2	1	0
SFR C4h	CD1	CD0	SWB	–	XTOFF	ALEOFF	DME1	DME0
	RW–0	RW–1	RW–0		RW*–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset, *=See description

CD1, CD0
Bits 7–6

Clock Divide Control 1–0. These bits select the number of crystal oscillator clocks required to generate one machine cycle. Switching between modes requires a transition through the divide by 4 mode (CD1, CD0=01). For example, to go from 64 to 1024 clocks per cycle the device must first go from 64 to 4 clocks per cycle, and then from 4 to 1024 clocks per cycle. Attempts to perform an invalid transition will be ignored. The setting of these bits will effect the timers and serial ports as shown below.

		OSC CYCLES PER MACH. CYCLE	OSC CYCLES PER TIMER 0/1/2 CLOCK		OSC CYCLES PER TIMER 2 CLK, BAUD RATE GEN.		OSC CYCLES PER SERIAL PORT CLK, MODE 0		OSC CYCLES PER SERIAL PORT CLK, MODE 2	
CD1	CD0		TxM=0	TxM=1	T2M=0	T2M=1	SM2=0	SM2=1	SDMO=0	SMOD=1
0	0	RESERVED								
0	1	4	12	4	2	2	12	4	64	32
1	0	64	192	64	32	32	192	64	1024	512
1	1	1024	3072	1024	512	512	3072	1024	16384	8192

SWB
Bit 5

Switchback Enable. This bit allows an enabled external interrupt or serial port activity to force the Clock Divide Control bits to the divide by 4 state (01). Upon internal acknowledgment of an external interrupt, the device will

switch modes at the start of the jump to the interrupt service routine. Note that this means that an external interrupt must actually be recognized (i.e. be enabled and not masked by higher priority interrupts) for the switchback to occur. For serial port reception, the switch occurs at the start of the instruction following the falling edge of the start bit.

Bit 4

Reserved. When modifying the PMR register, software must write a 0 to this bit. Read data will be indeterminate.

XTOFF
Bit 3

Crystal Oscillator Disable. This bit disables the CPU crystal oscillator. It can only be set to 1 while running from the ring oscillator (XT/$\overline{RG}$=0). Clearing this bit restarts the crystal amplifier, resets the crystal warm–up counter, and after 65,536 external crystal cycles the XTUP bit will be set.
0=Crystal oscillator is enabled
1=Crystal oscillator is disabled

ALEOFF
Bit 2

ALE Disable. This bit disables the expression of the ALE signal on the device pin during all on–board program and data memory accesses. External memory accesses will automatically enable ALE independent of ALEOFF.
0=ALE expression is enabled
1=ALE expression is disabled

DME1, DME0
Bits 1–0

Data Memory Enable 1–0. These bits determine the functional relationship of the first 1024 bytes of data memory. Three memory configurations are supported to allow either external data memory access through the expanded multiplexed address/data bus of Ports 0 and Port 2, internal SRAM data memory access, or read–only access to EPROM programming information. Note these bits are cleared after a reset, so access to the internal SRAM is prohibited until these bits are modified.

DME1	DME0	DATA MEMORY ADDRESS RANGE	MEMORY ACCESS
0	0	0000h – FFFFh	External Data Memory (default)
0	1	0000h – 03FFh 0400h – FFFFh	Internal SRAM Data Memory External Data Memory
1	0	Reserved	Reserved
1	1	0000h – 03FFh 0400h – FFFBh FFFCh FFFDh – FFFFh	Internal SRAM Data Memory Reserved System Control Byte (EPROM Read Only) Reserved

The System Control Byte is a special EPROM location that contains nonvolatile system information. This byte is set during EPROM programming and is not alterable by software. This register can only be read when both Data Memory Enable bits are set. The user must be sure that this location is programmed by the use of a special programming utility supplied with the programming device.

System Control Byte Description (EPROM; FFFCh)

Bits 7–3 — Reserved. These bits will read 1. These bits should be set to 1 during EPROM programming.

LB3, LB2, LB1
Bits 2–0 — **EPROM Program Lock Bit 3–1.** These bits show the status of the firmware security of the on–board EPROM. Bit combinations other than shown are illegal.

LB3	LB2	LB1	EPROM PROTECTION MODE
0	0	0	Unconditional verification, full external operation. Additional EPROM programming allowed without full device erasure.
0	0	1	Verification using encryption, execution of external MOVC instruction on internal program memory is disabled. All other program execution and data memory access allowed. Device must be fully erased before EPROM can be programmed again.
0	1	1	Verification disabled, execution of external MOVC instruction on internal program memory is disabled, and access to internal MOVX data from external program is prohibited. All other program execution and data memory access allowed. Device must be fully erased before EPROM can be programmed again.
1	1	1	Verification disabled, external program execution prohibited. Device must be fully erased before EPROM can be programmed again.

Status Register (STATUS)

	7	6	5	4	3	2	1	0
SFR C5h	PIP	HIP	LIP	XTUP	SPTA1	SPRA1	SPTA0	SPRA0
	R–0	R–0	R–0	R–*	R–0	R–0	R–0	R–0

R=Unrestricted Read, –n=Value after Reset, *=See description

PIP
Bit 7 — **Power Fail Priority Interrupt Status.** When set, this bit indicates that software is currently servicing a power–fail interrupt. It is cleared when the program executes the corresponding RETI instruction. This bit is indeterminate on devices which do not incorporate the power–fail interrupt.

HIP
Bit 6 — **High Priority Interrupt Status.** When set, this bit indicates that software is currently servicing a high priority interrupt. It is cleared when the program executes the corresponding RETI instruction.

LIP
Bit 5 — **Low Priority Interrupt Status.** When set, this bit indicates that software is currently servicing a low priority interrupt. It is cleared when the program executes the corresponding RETI instruction.

XTUP
Bit 4 — **Crystal Oscillator Warm–up Status.** This bit indicates whether the CPU crystal oscillator has completed the 65,536 cycle warm–up and is ready to operate from the external crystal or oscillator. This bit is cleared each time the crystal oscillator is restarted following an exit from Stop mode or the XTOFF bit (PMR.3) is set. While cleared, this bit prevents software from setting the XT/$\overline{\text{RG}}$ bit (EXIF.3) to enable operation from the crystal. Note that XTUP differs from the RGMD bit (EXIF.2) in that XTUP shows the status of

the crystal while RGMD shows the current clock source. This bit is set to 1 following a power–on reset, but is unaffected by other forms of reset.

SPTA1
Bit 3

Serial Port 1 Transmit Activity Monitor. When set, this bit indicates that data is currently being transmitted by serial port 1. It is cleared when the internal hardware sets the TI_1 bit. Do not alter the Clock Divide Control bits (PMR.7–6) while this bit is set or serial port data may be lost.

SPRA1
Bit 2

Serial Port 1 Receive Activity Monitor. When set, this bit indicates that data is currently being received by serial port 1. It is cleared when the internal hardware sets the RI_1 bit. Do not alter the Clock Divide Control bits (PMR.7–6) while this bit is set or serial port data may be lost.

SPTA0
Bit 1

Serial Port 0 Transmit Activity Monitor. When set, this bit indicates that data is currently being transmitted by serial port 0. It is cleared when the internal hardware sets the TI_0 bit. Do not alter the Clock Divide Control bits (PMR.7–6) while this bit is set or serial port data may be lost.

SPRA0
Bit 0

Serial Port 0 Receive Activity Monitor. When set, this bit indicates that data is currently being received by serial port 0. It is cleared when the internal hardware sets the RI_0 bit. Do not alter the Clock Divide Control bits (PMR.7–6) while this bit is set or serial port data may be lost.

Timed Access Register (TA)

	7	6	5	4	3	2	1	0
SFR C7h	TA.7	TA.6	TA.5	TA.4	TA.3	TA.2	TA.1	TA.0
	W–1	W–1	W–1	W–1	W–1	W–1	W–1	W–1

W=Unrestricted Write, –n=Value after Reset

TA.7–0
Bits 7–0

Timed Access. Correctly accessing this register permits modification of timed access protected bits. Write AAh to this register first, followed within 3 cycles by writing 55h. Timed access protected bits can then be modified for a period of 3 cycles measured from the writing of the 55h.

Timer 2 Control (T2CON)

	7	6	5	4	3	2	1	0
SFR C8h	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/RL2
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

TF2
Bit 7

Timer 2 Overflow Flag. This flag will be set when Timer 2 overflows from FFFFh or the count equal to the capture register in down count mode. It must be cleared by software. TF2 will only be set if RCLK and TCLK are both cleared to 0.

EXF2
Bit 6

Timer 2 External Flag. A negative transition on the T2EX pin (P1.1) or timer 2 underflow/overflow will cause this flag to set based on the CP/$\overline{RL2}$ (T2CON.0), EXEN2 (T2CON.3), and DCEN (T2MOD.0) bits. If set by a neg-

ative transition, this flag must be cleared to 0 by software. Setting this bit in software or detection of a negative transition on the T2EX pin will force a timer interrupt if enabled.

CP/$\overline{RL2}$	EXEN2	DCEN	RESULT
1	0	x	Negative transitions on P1.1 will not affect this bit.
1	1	x	Negative transitions on P1.1 will set this bit.
0	0	0	Negative transitions on P1.1 will not affect this bit.
0	1	0	Negative transitions on P1.1 will set this bit.
0	x	1	Bit toggles whenever timer 2 underflows/overflows and can be used as a 17th bit of resolution. In this mode, EXF2 will not cause an interrupt.

RCLK
Bit 5

Receive Clock Flag. This bit determines the serial port 0 timebase when receiving data in serial modes 1 or 3.
0=Timer 1 overflow is used to determine receiver baud rate for serial port 0.
1=Timer 2 overflow is used to determine receiver baud rate for serial port 0. Setting this bit will force timer 2 into baud rate generation mode. The timer will operate from a divide by 2 of the external clock.

TCLK
Bit 4

Transmit Clock Flag. This bit determines the serial port 0 timebase when transmitting data in serial modes 1 or 3.
0=Timer 1 overflow is used to determine transmitter baud rate for serial port 0.
1=Timer 2 overflow is used to determine transmitter baud rate for serial port 0. Setting this bit will force timer 2 into baud rate generation mode. The timer will operate from a divide by 2 of the external clock.

EXEN2
Bit 3

Timer 2 External Enable. This bit enables the capture/ reload function on the T2EX pin if Timer 2 is not generating baud rates for the serial port.
0=Timer 2 will ignore all external events at T2EX.
1=Timer 2 will capture or reload a value if a negative transition is detected on the T2EX pin.

TR2
Bit 2

Timer 2 Run Control. This bit enables/disables the operation of timer 2. Halting this timer will preserve the current count in TH2, TL2.
0=Timer 2 is halted.
1=Timer 2 is enabled.

C/$\overline{T2}$
Bit 1

Counter/Timer Select. This bit determines whether timer 2 will function as a timer or counter. Independent of this bit, timer 2 runs at 2 clocks per tick when used in either baud rate generator or clock output mode.
0=Timer 2 function as a timer. The speed of timer 2 is determined by the T2M bit (CKCON.5).
1=Timer 2 will count negative transitions on the T2 pin (P1.0).

CP/$\overline{RL2}$
Bit 0

Capture/Reload Select. This bit determines whether the capture or reload function will be used for timer 2. If either RCLK or TCLK is set, this bit will not function and the timer will function in an auto–reload mode following each overflow.
0=Auto–reloads will occur when timer 2 overflows or a falling edge is detected on T2EX if EXEN2=1.

1=Timer 2 captures will occur when a falling edge is detected on T2EX if EXEN2=1.

Timer 2 Mode (T2MOD)

	7	6	5	4	3	2	1	0
SFR C9h	–	–	–	–	–	–	T2OE	DCEN
							RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

Bits 7–2 Reserved. Read data will be indeterminate.

T2OE
Bit 1

Timer 2 Output Enable. This bit enables/disables the clock output function of the T2 pin (P1.0).
0=The T2 pin functions as either a standard port pin or as a counter input for timer 2.
1=Timer 2 will drive the T2 pin with a clock output if C/$\overline{T2}$=0. Also, timer 2 rollovers will not cause interrupts.

DCEN
Bit 0

Down Count Enable. This bit, in conjunction with the T2EX pin, controls the direction that timer 2 counts in 16–bit auto–reload mode.

DCEN	T2EX	DIRECTION
1	1	Up
1	0	Down
0	x	Up

Timer 2 Capture LSB (RCAP2L)

	7	6	5	4	3	2	1	0
SFR CAh	RCAP2L.7	RCAP2L.6	RCAP2L.5	RCAP2L.4	RCAP2L.3	RCAP2L.2	RCAP2L.1	RCAP2L.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

RCAP2L.7–0
Bits 7–0

Timer 2 Capture LSB. This register is used to capture the TL2 value when timer 2 is configured in capture mode. RCAP2L is also used as the LSB of a 16–bit reload value when timer 2 is configured in auto–reload mode.

Timer 2 Capture MSB (RCAP2H)

	7	6	5	4	3	2	1	0
SFR CBh	RCAP2H.7	RCAP2H.6	RCAP2H.5	RCAP2H.4	RCAP2H.3	RCAP2H.2	RCAP2H.1	RCAP2H.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

RCAP2H.7–0
Bits 7–0

Timer 2 Capture MSB. This register is used to capture the TH2 value when timer 2 is configured in capture mode. RCAP2H is also used as the MSB of a 16–bit reload value when timer 2 is configured in auto–reload mode.

Timer 2 LSB (TL2)

	7	6	5	4	3	2	1	0
SFR CCh	TL2.7	TL2.6	TL2.5	TL2.4	TL2.3	TL2.2	TL2.1	TL2.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

TL2.7–0
Bits 7–0

Timer 2 LSB. This register contains the least significant byte of Timer 2.

Timer 2 MSB (TH2)

	7	6	5	4	3	2	1	0
SFR CDh	TH2.7	TH2.6	TH2.5	TH2.4	TH2.3	TH2.2	TH2.1	TH2.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

TH2.7–0
Bits 7–0

Timer 2 MSB. This register contains the most significant byte of Timer 2.

Program Status Word (PSW)

	7	6	5	4	3	2	1	0
SFR D0h	CY	AC	F0	RS1	RS0	OV	F1	PARITY
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

CY
Bit 7

Carry Flag. This bit is set when if the last arithmetic operation resulted in a carry (during addition) or a borrow (during subtraction). Otherwise it is cleared to 0 by all arithmetic operations.

AC
Bit 6

Auxiliary Carry Flag. This bit is set to 1 if the last arithmetic operation resulted in a carry into (during addition), or a borrow (during subtraction) from the high order nibble. Otherwise it is cleared to 0 by all arithmetic operations.

F0
Bit 5

User Flag 0. This is a bit–addressable, general purpose flag for software control.

RS1, RS0
Bits 4–3

Register Bank Select 1–0. These bits select which register bank is addressed during register accesses.

RS1	RS0	REGISTER BANK	ADDRESS
0	0	0	00h – 07h
0	1	1	08h – 0Fh
1	0	2	10h – 17h
1	1	3	18h – 1Fh

OV
Bit 2

Overflow Flag. This bit is set to 1 if the last arithmetic operation resulted in a carry (addition), borrow (subtraction), or overflow (multiply or divide). Otherwise it is cleared to 0 by all arithmetic operations.

F1
Bit 1

User Flag 1. This is a bit–addressable, general purpose flag for software control.

PARITY
Bit 0

Parity Flag. This bit is set to 1 if the modulo–2 sum of the eight bits of the accumulator is 1 (odd parity); and cleared to 0 on even parity.

Watchdog Control (WDCON)

	7	6	5	4	3	2	1	0
SFR D8h	SMOD	POR	EPFI	PFI	WDIF	WTRF	EWT	RWT
	RW–0	RT–*	RW–0	RW–*	RT–0	RW–*	RT–*	RT–0

R=Unrestricted Read, W=Unrestricted Write, T=Timed Access Write Only, –n=Value after Reset, *=See description

SMOD
Bit 7

Serial Modification. This bit controls the doubling of the serial port 1 baud rate in modes 1, 2, and 3.
0=Serial port 1 baud rate operates at normal speed
1=Serial port 1 baud rate is doubled.

POR
Bit 6

Power–on Reset Flag. This bit indicates whether the last reset was a power–on reset. This bit is typically interrogated following a reset to determine if the reset was caused by a power–on reset. It must be cleared by a Timed Access write before the next reset of any kind or the software may erroneously determine that another power–on reset has occurred. This bit is set following a power–on reset and unaffected by all other resets. Note: This bit is not Timed Access protected on the DS80C310.
0=Last reset was from a source other than a power–on reset
1=Last reset was a power–on reset.

EPFI
Bit 5

Enable Power fail Interrupt. This bit enables/disables the ability of the internal band–gap reference to generate a power–fail interrupt when V_{CC} falls below approximately 4.5 volts. While in Stop mode, both this bit and the Band–gap Select bit, BGS (EXIF.0), must be set to enable the power–fail interrupt.
0=Power–fail interrupt disabled.
1=Power–fail interrupt enabled during normal operation. Power–fail interrupt enabled in Stop mode if BGS is set.

PFI
Bit 4

Power fail Interrupt Flag. When set, this bit indicates that a power–fail interrupt has occurred. This bit must be cleared in software before exiting the interrupt service routine, or another interrupt will be generated. Setting this bit in software will generate a power–fail interrupt, if enabled.

WDIF
Bit 3

Watchdog Interrupt Flag. This bit, in conjunction with the Watchdog Timer Interrupt Enable bit, EWDI (EIE.4), and Enable Watchdog Timer Reset bit (WDCON.1), indicates if a watchdog timer event has occurred and what action will be taken. This bit must be cleared in software before exiting the interrupt service routine, or another interrupt will be generated. Setting this bit in software will generate a watchdog interrupt if enabled. This bit can only be modified using a Timed Access Procedure.

EWT	EWDI	WDIF	RESULT
x	x	0	No watchdog event has occured.
0	0	1	Watchdog time–out has expired. No interrupt has been generated.
0	1	1	Watchdog interrupt has occured.
1	0	1	Watchdog time–out has expired. No interrupt has been generated. Watchdog timer reset will occur in 512 cycles if RWT is not strobed.
1	1	1	Watchdog interrupt has occurred. Watchdog timer reset will occur in 512 cycles if RWT is not set using a Timed Access procedure.

WTRF
Bit 2

Watchdog Timer Reset Flag. When set, this bit indicates that a watchdog timer reset has occurred. It is typically interrogated to determine if a reset was caused by watchdog timer reset. It is cleared by a power–on reset, but otherwise must be cleared by software before the next reset of any kind or software may erroneously determine that a watchdog timer reset has occurred. Setting this bit in software will not generate a watchdog timer reset. If the EWT bit is cleared, the watchdog timer will have no effect on this bit.

EWT
Bit 1

Enable Watchdog Timer Reset. This bit enables/disables the ability of the watchdog timer to reset the device. This bit has no effect on the ability of the watchdog timer to generate a watchdog interrupt. The time–out period of the watchdog timer is controlled by the Watchdog Timer Mode Select bits (CKCON.7–6). Clearing this bit will disable the ability of the watchdog timer to generate a reset, but have no affect on the timer itself, or its ability to generate a watchdog timer interrupt. This bit can only be modified using a Timed Access Procedure. The default power–on reset state of this bit is 0 on the

ROMless devices. If the device contains EPROM program memory, the default power–on reset state of EWT is determined by the Watchdog Default POR State bit (WDPOR) located in the System Control Byte. If the device contains mask ROM program memory, the default power–on reset state is determined by a mask option.
0=A timeout of the watchdog timer will not cause the device to reset.
1=A timeout of the watchdog timer will cause the device to reset.

RWT
Bit 0

Reset Watchdog Timer. Setting this bit will reset the watchdog timer count. This bit must be set using a Timed Access procedure before the watchdog timer expires, or a watchdog timer reset and/or interrupt will be generated if enabled. The time–out period is defined by the Watchdog Timer Mode Select bits (CKCON.7–6). This bit will always be 0 when read.

Accumulator (A or ACC)

	7	6	5	4	3	2	1	0
SFR E0h	ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

ACC.7–0
Bits 7–0

Accumulator. This register serves as the accumulator for arithmetic operations. It is functionally identical to the accumulator found in the 80C32.

Extended Interrupt Enable (EIE)

	7	6	5	4	3	2	1	0
SFR E8h	–	–	ERTCI	EWDI	EX5	EX4	EX3	EX2
			RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

Bit 7–6

Reserved. Read data will be indeterminate.

ERTCI
Bit 5

Real Time Clock Interrupt Enable. This bit enables/disables the real–time clock interrupt on the DS87C530. This bit will read 0 on all other devices.
0=Disable the real–time clock interrupt.
1=Enable interrupt requests generated by the real–time clock.

EWDI
Bit 4

Watchdog Interrupt Enable. This bit enables/disables the watchdog interrupt.
0=Disable the watchdog interrupt.
1=Enable interrupt requests generated by the watchdog timer.

EX5
Bit 3

External Interrupt 5 Enable. This bit enables/disables external interrupt 5.
0=Disable external interrupt 5.
1=Enable interrupt requests generated by the $\overline{\text{INT5}}$ pin

EX4
Bit 2

External Interrupt 4 Enable. This bit enables/disables external interrupt 4.
0=Disable external interrupt 4.
1=Enable interrupt requests generated by the INT4 pin

EX3
Bit 1

External Interrupt 3 Enable. This bit enables/disables external interrupt 3.
0=Disable external interrupt 3.
1=Enable interrupt requests generated by the $\overline{\text{INT3}}$ pin.

EX2
Bit 0

External Interrupt 2 Enable. This bit enables/disables external interrupt 2.
0=Disable external interrupt 2.
1=Enable interrupt requests generated by the INT2 pin.

B Register (B)

	7	6	5	4	3	2	1	0
SFR F0h	B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

B.7–0
Bits 7–0

B Register. This register serves as a second accumulator for certain arithmetic operations. It is functionally identical to the B register found in the 80C32.

Real Time Alarm Subsecond Register (RTASS)

	7	6	5	4	3	2	1	0
SFR F2h	RTASS.7	RTASS.6	RTASS.5	RTASS.4	RTASS.3	RTASS.2	RTASS.1	RTASS.0
	RW–*	RW–*	RW–*	RW–*	RW–*	RW–*	RW–*	RW–*

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset, *=See description

RTASS.7–0
Bits 7–0

Real Time Alarm Subsecond. These bits represent the subsecond alarm which will be compared against the RTC Subsecond register (RTCSS;FAh). The ability of a match between the two registers to cause an alarm is controlled by the RTC Subsecond Register Compare Enable bit (RTCC.7). The contents of this register will be indeterminate following a no–battery reset, and unchanged by all other forms of reset.

Real Time Alarm Second Register (RTAS)

	7	6	5	4	3	2	1	0
SFR F3h	0	0	RTAS.5	RTAS.4	RTAS.3	RTAS.2	RTAS.1	RTAS.0
	R–0	R–0	RW–*	RW–*	RW–*	RW–*	RW–*	RW–*

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset, *=See description

Bits 7–6

Reserved. These bits will be 0 when read.

RTAS.5–0
Bits 5–0

Real Time Alarm Second. These bits represent the second alarm which will be compared against the RTC Second register (RTCS;FBh). The ability of a match between the two registers to cause an alarm is controlled by the RTC Second Register Compare Enable bit (RTCC.6). This register should only be loaded with values from 0 to 3Bh (0 to 59 seconds). The contents of this register will be indeterminate following a no–battery reset (except bits 7, 6), and unchanged by all other forms of reset.

Real Time Alarm Minute Register (RTAM)

	7	6	5	4	3	2	1	0
SFR F4h	0	0	RTAM.5	RTAM.4	RTAM.3	RTAM.2	RTAM.1	RTAM.0
	R–0	R–0	RW–*	RW–*	RW–*	RW–*	RW–*	RW–*

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset, *=See description

Bits 7–6 Reserved. These bits will be 0 when read.

RTAM.5–0
Bits 5–0 **Real Time Alarm Minute.** These bits represent the minute alarm which will be compared against the RTC Minute register (RTCM;FCh). The ability of a match between the two registers to cause an alarm is controlled by the RTC Minute Register Compare Enable bit (RTCC.5).). This register should only be loaded with values from 0 to 3Bh (0 to 59 minutes). The contents of this register will be indeterminate following a no–battery reset (except bits 7, 6), and unchanged by all other forms of reset.

Real Time Alarm Hour Register (RTAH)

	7	6	5	4	3	2	1	0
SFR F5h	0	0	0	RTAH.4	RTAH.3	RTAH.2	RTAH.1	RTAH.0
	R–0	R–0	R–0	RW–*	RW–*	RW–*	RW–*	RW–*

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset, *=See description

Bits 7–5 Reserved. These bits will be 0 when read.

RTAH.4–0
Bits 4–0 **Real Time Alarm Hour.** These bits represent the hour alarm which will be compared against the RTC Hour register (RTCH;FDh). The ability of a match between the two registers to cause an alarm is controlled by the RTC Hour Register Compare Enable bit (RTCC.4). This register should only be loaded with values from 0 to 17h (0 to 23 hours). The day of week bits DOW2–0, located in RTCH.7–5 do not have a corresponding alarm feature. The contents of this register will be indeterminate following a no–battery reset (except bits 7, 6, 5), and unchanged by all other forms of reset.

Extended Interrupt Priority (EIP)

	7	6	5	4	3	2	1	0
SFR F8h	–	–	PRTCI	PWDI	PX5	PX4	PX3	PX2
			RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

Bit 7–6 Reserved. Read data will be indeterminate.

PRTCI
Bit 5 **Real Time Clock Interrupt Priority.** This bit controls the priority of the real–time clock interrupt on the DS87C530. This bit will read 0 on all other devices.
0=The real–time clock interrupt is a low priority interrupt.
1=The real–time clock interrupt is a high priority interrupt.

PWDI
Bit 4

Watchdog Interrupt Priority. This bit controls the priority of the watchdog interrupt.
0=The watchdog interrupt is a low priority interrupt.
1=The watchdog interrupt is a high priority interrupt.

PX5
Bit 3

External Interrupt 5 Priority. This bit controls the priority of external interrupt 5.
0=External interrupt 5 is a low priority interrupt.
1=External interrupt 5 is a high priority interrupt.

PX4
Bit 2

External Interrupt 4 Priority. This bit controls the priority of external interrupt 4.
0=External interrupt 4 is a low priority interrupt.
1=External interrupt 4 is a high priority interrupt.

PX3
Bit 1

External Interrupt 3 Priority. This bit controls the priority of external interrupt 3.
0=External interrupt 3 is a low priority interrupt.
1=External interrupt 3 is a high priority interrupt.

PX2
Bit 0

External Interrupt 2 Priority. This bit controls the priority of external interrupt 2.
0=External interrupt 2 is a low priority interrupt.
1=External interrupt 2 is a high priority interrupt.

Real Time Clock Control Register (RTCC)

	7	6	5	4	3	2	1	0
SFR F9h	SSCE	SCE	MCE	HCE	RTCRE	RTCWE	RTCIF	RTCE
	RW–*	RW–*	RW–*	RW–*	RW*–0	RT*–0	R*–*	RT–*

R=Unrestricted Read, W=Unrestricted Write, T=Timed Access Write Only, –n=Value after Reset, *=See description

SSCE
Bit 7

RTC Subsecond Register Compare Enable. This bit enables a match between the Real Time Alarm Subsecond Register (RTASS;F2h) and the Real Time Clock Subsecond Register (RTCSS;FAh) to contribute to the RTC interrupt request. This bit will be indeterminate following a no–battery reset, and is unaffected by all other resets.
0=The subsecond value is a Don't Care when evaluating the RTC alarm. If any other alarm register compare bits are enabled, this will cause one interrupt per subsecond tick (1/256 second) for as long as the other registers match.
1=Include the subseconds along with any other registers when evaluating alarm compare conditions.

SCE
Bit 6

RTC Second Register Compare Enable. This bit enables a match between the Real Time Alarm Second Register (RTAS;F3h) and the Real Time Clock Second Register (RTCS;FBh) to contribute to the RTC interrupt request. This bit will be indeterminate following a no–battery reset, and is unaffected by all other resets.
0=The second value is a Don't Care when evaluating an RTC alarm. If any other alarm register compare bits are enabled, this will cause one interrupt per second as long as the other registers match.
1=Include the second along with any other registers when evaluating alarm compare conditions.

MCE
Bit 5

RTC Minute Register Compare Enable. This bit enables a match between the Real Time Alarm Minute Register (RTAM;F4h) and the Real Time Clock Minute Register (RTCM;FCh) to contribute to the RTC interrupt request. This bit will be indeterminate following a no–battery reset, and is unaffected by all other resets.
0=The minute value is a Don't Care when evaluating an RTC alarm. If any other alarm register compare bits are enabled, this will cause one interrupt per minute as long as the other registers match.
1=Include the minute along with any other registers when evaluating alarm compare conditions.

HCE
Bit 4

RTC Hour Register Compare Enable. This bit enables a match between the Real Time Alarm Hour Register (RTAH;F5h) and the Real Time Clock Hour Register (RTCH;FDh) to contribute to the RTC interrupt request. This bit will be indeterminate following a no–battery reset, and is unaffected by all other resets.
0=The hour value is a Don't Care when evaluating an RTC alarm. If any other alarm register compare bits are enabled, this will cause one interrupt per hour for as long as the other registers match.
1=Include the hour along with any other registers when evaluating alarm compare conditions.

RTCRE
Bit 3

RTC Read Enable. This bit temporarily halts internal updating of the RTC to allow software to read the current time. No loss of time will occur. This bit will be cleared to 0 following any reset. Attempts to set the RTCRE and RTCWE bits simultaneously will be ignored. When this bit is cleared, software must wait 4 machine cycles before setting either the RTCRE or RTCWE bit again.
0=Reads of the RTC clock registers (RTCSS;FAh, RTCS;FBh, RTCM;FCh, RTCH;FDh, RTCD0;FEh, RTCD1;FFh) are prohibited and will return erroneous values.
1=Reads of the RTC clock registers are permitted during a 1 ms window starting from the time the bit is set. Immediately after setting this bit, software must wait 4 machine cycles to allow all time registers to synchronize. This bit should be cleared by the user when the desired reads are complete, although it will clear automatically within 1.95 ms if not cleared in software.

RTCWE
Bit 2

RTC Write Enable. This bit temporarily halts the RTC to allow software to update the current time. No loss of time will occur. This bit can only be modified using a Timed Access procedure. Changing this bit from 1 to 0 will reset the RTCSS register to 00h.This bit will be cleared to 0 following any reset.
0=Writes to the RTC clock registers (RTCSS;FAh, RTCS;FBh, RTCM;FCh, RTCH;FDh, RTCD0;FEh, RTCD1;FFh) are ignored. Attempts to set the RTCRE and RTCWE bits simultaneously will be ignored. When this bit is cleared, software must wait 4 machine cycles before setting either the RTCRE or RTCWE bit again.
1=Writes to the RTC clock registers are permitted during a 1 ms window starting from the time this bit is set. Immediately after setting this bit, software must wait 4 machine cycles to allow all time registers to synchronize. This bit should be cleared by the user when the desired updates are complete, although it will clear automatically after 1.95 ms if not cleared in software.

RTCIF
Bit 1

RTC Interrupt Flag. This bit indicates that a RTC alarm match has been made between all the enabled alarm registers and their corresponding clock registers. This bit will generate an RTC Interrupt if the ERTCI bit (EIE.5) is set, and must be cleared by software following an interrupt. RTC interrupts cannot be generated by setting this bit. Clearing all alarm compare enable bits (RTCC.7–4) will also clear this bit. This bit will be indeterminate following a no–battery reset, and is unaffected by all other resets. This bit cannot be set in software.
0=No RTC interrupts are pending.
1=RTC Interrupt is pending/active.

RTCE
Bit 0

RTC Enable. This bit enables/disables the RTC oscillator, halting the RTC. This bit must be accessed using a Timed Access procedure. This bit will be indeterminate following a no–battery reset, and is unaffected by all other resets. If RTC operation is desired, it must be enabled following battery application.
0=RTC oscillator is disabled.
1=RTC oscillator is enabled.

Real Time Clock Subsecond Register (RTCSS)

	7	6	5	4	3	2	1	0
SFR FAh	RTCSS.7	RTCSS.6	RTCSS.5	RTCSS.4	RTCSS.3	RTCSS.2	RTCSS.1	RTCSS.0
	R*–*	R*–*	R*–*	R*–*	R*–*	R*–*	R*–*	R*–*

R=Unrestricted Read, –n=Value after Reset, *=See description

RTCSS.7–0
Bits 7–0

Real Time Clock Subseconds. This register represents the subsecond value of the RTC. It can be read only when the RTCRE bit is set, and writes are not permitted. It is reset to 00h when the RTCWE bit is cleared. The register counts from 0h to FFh.

Real Time Clock Second Register (RTCS)

	7	6	5	4	3	2	1	0
SFR FBh	0	0	RTCS.5	RTCS.4	RTCS.3	RTCS.2	RTCS.1	RTCS.0
	R*–0	R*–0	R*W*–*	R*W*–*	R*W*–*	R*W*–*	R*W*–*	R*W*–*

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset, *=See description

Bits 7–6

Reserved. These bits will be 0 when read.

RTCS.5–0
Bits 5–0

Real Time Clock Seconds. This register represents the second value of the RTC. This register can be read only when the RTCRE bit is set, and can only be modified when the RTCWE bit is set. Consult the description of the RTCWE bit for the programming protocol for this register. This register counts from 0h to 3Bh (0 to 59 seconds), and any writes to this register outside of that range will generate an inaccurate count.

Real Time Clock Minute Register (RTCM)

	7	6	5	4	3	2	1	0
SFR FCh	0	0	RTCM.5	RTCM.4	RTCM.3	RTCM.2	RTCM.1	RTCM.0
	R*–0	R*–0	R*W*–*	R*W*–*	R*W*–*	R*W*–*	R*W*–*	R*W*–*

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset, *=See description

Bits 7–6 Reserved. These bits will be 0 when read.

RTCM.5–0
Bits 5–0 **Real Time Clock Minutes.** This register represents the minute value of the RTC. This register can be read only when the RTCRE bit is set, and can only be modified when the RTCWE bit is set. Consult the description of the RTCWE bit for the programming protocol for this register. This register counts from 0h to 3Bh (0 to 59 minutes), and any writes to this register outside of that range will generate an inaccurate count.

Real Time Clock Hour Register (RTCH)

	7	6	5	4	3	2	1	0
SFR FDh	DOW2	DOW1	DOW0	RTCH.4	RTCH.3	RTCH.2	RTCH.1	RTCH.0
	R*W*–*	R*W*–*	R*W*–*	R*W*–*	R*W*–*	R*W*–*	R*W*–*	R*W*–*

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset, *=See description

DOW2–0
Bits 7–5 **Real Time Clock Day of the Week.** These bits represent the current day of the week. This register can be read only when the RTCRE bit is set, and can only be modified when the RTCWE bit is set. Consult the description of the RTCWE bit for the programming protocol for this register. This register counts from 1h to 7h, and increments when the hour value of the RTC (RTCH.4–0) rolls over from 17h to 0h. Writing a 0h to these bits will disable the day of week function and the count will remain 0. No alarm corresponds to these bits.

RTCH.4–0
Bits 4–0 **Real Time Clock Hours.** These bits represent the hour value of the RTC. This register can be read only when the RTCRE bit is set, and can only be modified when the RTCWE bit is set. Consult the description of the RTCWE bit for the programming protocol for this register. This register counts from 0h to 17h (0 to 23 hours), and any writes outside of that range will generate an inaccurate count.

Real Time Clock Day Register 0 (RTCD0)

	7	6	5	4	3	2	1	0
SFR FEh	RTCD0.7	RTCD0.6	RTCD0.5	RTCD0.4	RTCD0.3	RTCD0.2	RTCD0.1	RTCD0.0
	R*W*–*	R*W*–*	R*W*–*	R*W*–*	R*W*–*	R*W*–*	R*W*–*	R*W*–*

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset, *=See description

RTCD0.7–0
Bits 7–0

Real Time Clock Day Register 0. This register contains the least significant byte of the 16–bit current day count. This is not an absolute value tied to a specific calendar date, but rather a relative day count defined by the user. This register can be read only when the RTCRE bit is set, and can only be modified when the RTCWE bit is set. Consult the description of the RTCWE bit for the programming protocol for this register. The register counts from 0h to FFh. No alarm corresponds to these bits.

Real Time Clock Day Register 1 (RTCD1)

	7	6	5	4	3	2	1	0
SFR FFh	RTCD1.7	RTCD1.6	RTCD1.5	RTCD1.4	RTCD1.3	RTCD1.2	RTCD1.1	RTCD1.0
	R*W*–*	R*W*–*	R*W*–*	R*W*–*	R*W*–*	R*W*–*	R*W*–*	R*W*–*

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset, *=See description

RTCD1.7–0
Bits 7–0

Real Time Clock Day Register 1. This register contains the most significant byte of the 16–bit current day count. This is not an absolute value tied to a specific calendar date, but rather a relative day count defined by the user. This register can be read only when the RTCRE bit is set, and can only be modified when the RTCWE bit is set. Consult the description of the RTCWE bit for the programming protocol for this register. The register counts from 0h to FFh. A rollover of this register will clear RTCD1 and RTCD0. No alarm corresponds to these bits.

INSTRUCTION TIMING

All instructions in the High–Speed Microcontroller perform the same functions as their 80C32 counterparts. Their effect on bits, flags, and other status functions is identical. However, the timing of each instruction is different. This applies both in absolute terms of nanoseconds for a given crystal, and in relative terms of clocks.

For absolute timing of real–time events, the timing of software loops will need to be calculated using the data provided in Section 16, Instruction Set Details. However, timers default to run at the older 12 clocks per timer increment. Therefore, while software runs at higher speed, timer–based events need no modification.

The relative time of two instructions might be different in the new architecture than it was previously. For example, both the one–byte, two–cycle "MOVX A, @DPTR" instruction and the three–byte, two–cycle "MOV direct, direct" instruction used two cycles. Therefore, they required the same amount of time. In the High–Speed Microcontroller, the MOVX instruction uses two cycles but the "MOV direct, direct" uses three cycles. While both are faster than their original counterparts, they now have different execution times from each other. This is because the High–Speed Microcontroller typically uses one cycle for each byte. This is generally true for all instructions except for MUL, DIV, MOVC, MOVX, and branch type instructions. These require more cycles than the number of bytes involved. The timing of each instruction should be examined for familiarity with the changes. Note that a machine cycle now requires just four clocks, and provides one ALE pulse per cycle. Many instructions require only one cycle, but some require five. In the original architecture, all were one or two cycles except for MUL and DIV.

ADDRESSING MODES

The High–Speed Microcontroller uses the standard 8051 instruction set which is supported by a wide range of third party assemblers and compilers. Like the 8051, the High–Speed Microcontroller uses three memory areas. These are program memory, data memory, and Registers. Both the program and data areas are 64KB each. They extend from 0000h to FFFFh. The register areas are located between 00h and FFh, but do not overlap with the program and data segments. This is because the High–Speed Microcontroller uses different modes of addressing to reach each memory segment. These modes are described below.

Program memory is the area from which all instructions are fetched. It is inherently read only. This is because the 8051 instruction set provides no instructions that write to this area. Read/write access is for data memory and Registers only. No special action is required to fetch from program memory. Each instruction fetch will be performed automatically by the on–chip hardware. In versions that contain on–chip memory, the hardware will decide whether the fetch is on–chip or off–chip based on the address.

Explicit addressing modes are needed for the data memory and register areas. These modes determine which register area is accessed or if off–chip data memory is used.

The High–Speed Microcontroller supports eight addressing modes. They are:

Register Addressing
Direct Addressing
Register Indirect Addressing
Immediate Addressing
Register Indirect Addressing with Displacement
Relative Addressing
Page Addressing
Extended Addressing

Five of the eight are used to address operands. The remainder are used for program control and branching. When writing assembly language instructions that use arguments, the convention is destination, source. Each mode of addressing is summarized below. Note that many instructions (such as ADD) have multiple addressing modes available..

Register Addressing

Register Addressing is used for operands that are located in one of the eight Working Registers (R7–R0). These are the currently selected Working Register bank, which reside in the lower 32 bytes of Scratchpad RAM. A register bank is selected using two bits in the Program Status Word (PSW;D0h). This addressing mode is powerful, since it uses the active bank without knowing which bank is selected. Thus one instruction can have multiple uses by simply switching banks. Register Addressing is also a high–speed instruction, requiring only one machine cycle. Two examples of Register Addressing are provided below.

```
ADD     A, R4       ;Add Accumulator to register R4

INC     R2          ;Increment the value in register R2
```

In the first case, the value in R4 is the source of the operation. In the later, R2 is the destination. These instructions do not consider the absolute address of the register. They will act on whichever bank has been selected.

Any Working Register may also be accessed by Direct Addressing, described below. To do this, the absolute address must be specified.

Direct Addressing

Direct Addressing is the mode used to access the entire lower 128 bytes of Scratchpad RAM and the SFR area. It is commonly used to move the value in one register to another. Two examples are shown below.

```
MOV     72h, 74h    ;Move the value in register 74 to
                    ; register 72.

MOV     90h, 20h    ;Move the value in register 20 to
                    ; the SFR at 90h (Port 1)
```

Note that there is no instruction difference between a RAM access and an SFR access. The SFRs are simply register locations above 7Fh.

Direct Addressing also extends to bit addressing. There is a group of instructions that explicitly use bits. The address information provided to such an instruction is the bit location, rather than the register address. Registers between 20h and 2Fh contain bits that are individually addressable. SFRs that end in 0 or 8 are bit addressable. An example of Direct Bit Addressing is as follows.

```
SETB    00h         ;Set bit 00 in the RAM. This is the
                    ; LSb of the register at address 20h
                    ; as shown in Section 4.

MOV     C, 0B7h     ;Move the contents of bit B7 to the
                    ; Carry flag. Bit B7 is the MSb of
                    ; register B0 (Port 3).
```

Register Indirect Addressing

This mode is used to access the Scratchpad RAM locations above 7Fh. It can also be used to reach the lower RAM (0h – 7Fh) if needed. The address is supplied by the contents of the Working Register specified in the instruction. Thus one instruction can be used to reach many values by altering the contents of the designated Working Register. Note that in general, only R0 and R1 can be used as pointers. An example of Register Indirect Addressing is as follows.

```
ANL     A, @R0      ;Logical AND the Accumulator
                    ; with the contents of the register
                    ; pointed to by the value stored in
                    ; R0.
```

This mode is also used for Stack manipulation. This is because all Stack references are directed by the value in the Stack Pointer register. The Push and Pop instructions use this method of addressing. An example is as follows.

```
PUSH    A           ;Saves the contents of the
                    ; accumulator on the stack.
```

Register Indirect Addressing is used for all off–chip data memory accesses. These involve the MOVX instruction. The pointer registers can be R0, R1, DPTR0 and DPTR1. Both R0 and R1 reside in the Working Register area of the Scratchpad RAM. They can be used to reference a 256 byte area of off–chip data memory. When using this type of addressing, the upper address byte is supplied by the value in the Port 2 latch. This value must be selected by software prior to the MOVX instruction. An example is as follows.

```
MOVX    @R0, A      ;Write the value in the accumulator
                    ; to the address pointed to by R0 in
                    ; the page pointed to by P2.
```

The 16–bit Data pointers (DPTRs) can be used as an absolute off–chip reference. This gives access to the entire 64KB data memory map. An example is as follows.

```
MOVX @DPTR, A   ;Write the value in the accumulator
                ; to the address referenced by the
                ; selected data pointer.
```

Immediate Addressing

Immediate Addressing is used when one of the operands is predetermined and coded into the software. This mode is commonly used to initialize SFRs and to mask particular bits without affecting others. An example is as follows.

```
ORL     A, #40h     ;Logical OR the Accumulator
                    ; with 40h.
```

Register Indirect with Displacement

Register Indirect Addressing with Displacement is used to access data in lookup tables in program memory space. The location is created using a base address with an index. The base address can be either the PC or the DPTR. The index is the accumulator. The result is stored in the accumulator. An example is as follows.

```
MOVC  A, @A+DPTR    ;Load the accumulator with the
                    ; contents of program memory
                    ; pointed to by the contents of
                    ; the DPTR plus the value in
                    ; the accumulator.
```

Relative Addressing

Relative Addressing is used to determine a destination address for Conditional branch. Each of these instructions includes an 8–bit value that contains a two's complement address offset (–127 to +128) which is added to the PC to determine the destination address. This destination is branched to when the tested condition is true. The PC points to the program memory location immediately following the branch instruction when the offset is added. If the tested condition is not true, the next instruction is performed. An example is as follows.

```
JZ      $–20        ;Branch to the location (PC+2)–20
                    ; if the contents of the accumulator
                    ;=0.
```

Page Addressing

Page Addressing is used by the Branching instructions to specify a destination address within the same 2KB block as the next contiguous instruction. The full 16–bit address is calculated by taking the five highest order bits for the next instruction (PC+2) and concatenating them with the lowest order 11 bit field contained in the current instruction. An example is as follows.

```
0870h   ACALL100h       ;Call to the subroutine at
                        ; address 100h plus the
                        ; current page address.
```

In this example, the current page address is 800h, so the destination address is 900h.

Extended Addressing

Extended Addressing is used by the Branching instructions to specify a 16–bit destination address within the 64KB address space. The destination address is fixed in software as an absolute value. An example is as follows.

```
LJMP    0F732h      ; Jump to address 0F732h.
```

PROGRAM STATUS FLAGS

All Program Status Flags are contained in the Program Status Word at SFR location D0h. It contains flags that reflect the status of the CPU and the result of selected operations. The flags are summarized below. The following table shows the instructions that affect each flag.

Bit Description :

PSW.7 Carry	**C** Set when the previous operation resulted in a carry (during addition) or a borrow (during subtraction), otherwise cleared.
PSW.6 Auxiliary Carry	**AC** Set when the previous operation resulted in a carry (during addition) or a borrow (during subtraction) from the high order nibble. Otherwise cleared.
PSW.2 Overflow	**OV** Set when a carry was generated into the high order bit but not a carry out of the high order bit. OV is normally used with 2's complement arithmetic.
PSW.0 Parity	**P** Set to logic 1 to indicate an odd number of ones in the accumulator (odd parity). Cleared for an even number of ones. This produces even parity.

All of these bits are cleared to a logic 0 for all resets.

INSTRUCTIONS THAT AFFECT FLAG SETTINGS Table 4–4

INSTRUCTION	FLAGS			INSTRUCTION	FLAGS		
	C	**OV**	**AC**		**C**	**OV**	**AC**
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C, bit	X		
MUL	0	X		ANL C, $\overline{\text{bit}}$	X		
DIV	0	X		ORL C, bit	X		
DA	X			ORL C, $\overline{\text{bit}}$	X		
RRC	X			MOV C, bit	X		
RLC	X			CJNE	X		
SETB C	1						

X indicates the modification is according to the result of the instruction.

SECTION 5: CPU TIMING

The timing of the High–Speed Microcontroller is the area with the greatest departure from the original 8051 series. This section will briefly explain the timing and also compare it to the original.

OSCILLATOR

The High–Speed Microcontroller provides an on–chip oscillator circuit that can be driven by an external crystal or by an off–chip TTL clock source. The oscillator circuit provides the internal clocking signals to the on–chip CPU and I/O circuits.

Figure 5–1 shows the required connections for a crystal. In most cases, a crystal will be the preferred clock source. For very low power applications, a low frequency ceramic resonator may also be used. The capacitors shown in Figure 5–1 are typical values. If a resonator is used, higher capacitance, such as 47 pF may be needed.

For higher frequency designs, an off–chip clock oscillator may be preferred. This is illustrated in Figure 5–2. When using an off–chip oscillator, the duty cycle becomes important. As nearly as possible, a 50% duty cycle should be supplied.

XTAL1

This pin is the input to an inverting high gain amplifier. It also serves as the input for an off–chip oscillator. Note that when using an off–chip oscillator, XTAL2 is disconnected.

XTAL2

This pin is the output of the crystal amplifier. It can be used to distribute the clock to other devices on the same board. If using a crystal, the loading on this pin should be kept to a minimum, especially capacitive loading.

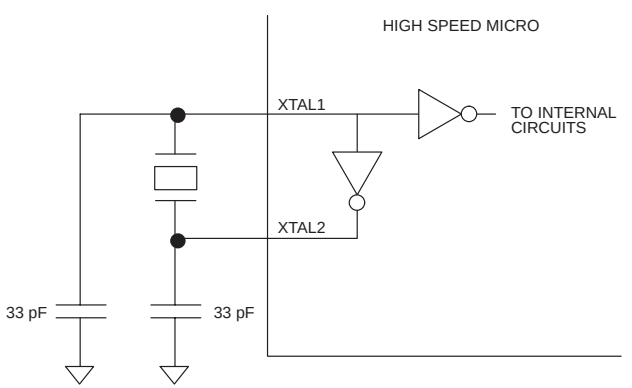
OSCILLATOR CHARACTERISTICS

The High–Speed Microcontroller was designed to operate with a parallel resonant AT cut crystal. The crystal should resonate at the desired frequency in its primary or fundamental mode. The oscillator employs a high gain amplifier to assure a clean waveform at high frequency. Due to the high performance nature of the product, both clock edges are used for internal timing. Therefore, the duty cycle of the clock source is of importance. A crystal circuit will balance itself automatically. Thus crystal users will not need to take extra precautions concerning duty cycle.

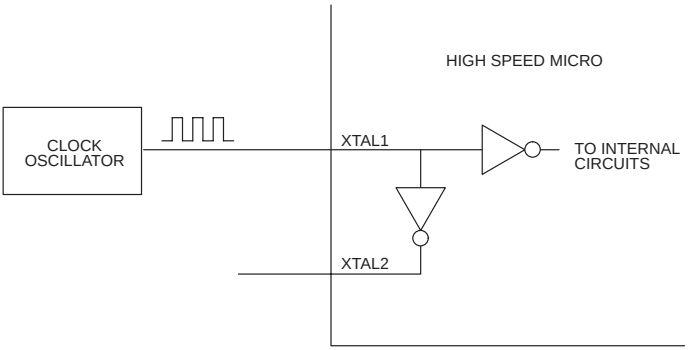
CRYSTAL SELECTION

The High–Speed Microcontroller family was designed to operate with fundamental mode crystals for improved stability. Although most high speed (i.e. greater than 25 MHz) crystals operate from their third overtone, fundamental mode crystals are available from most major crystal suppliers. Designers are cautioned to ensure that high speed crystals being specified for use in their application do operate at the rated frequency in their fundamental mode. The use of a third overtone crystal will typically result in oscillation rates at one–third the desired speed.

CRYSTAL CONNECTION Figure 5–1



CLOCK SOURCE INPUT Figure 5–2



INSTRUCTION TIMING

The clock source, whether crystal or oscillator, supplies the internal functions with a precise time base. The clock is used to create the basic unit of timing called a machine cycle. One machine cycle consists of four clocks when operating in divide by 4 mode. The use of Power Management modes will cause the device to utilize 64 or 1024 external clock cycles per machine cycle. Each clock within a machine cycle produces a state designated with a C. Thus, within a machine cycle there are four states called C1, C2, C3, and C4. Various operations take place during each C state. Within this section and throughout others, an event timing will be identified by its C state. For example, ALE rises at the beginning of the C1 time. Since the clock source is the source of nearly all timing, the electrical specifications are given in terms of clocks. The time of a clock period is referred to as t_{CLCL}.

Most times in the electrical specifications are specified as some number of clocks from the edge of a signal. The signal edges were also derived from the clock source and the C states.

Due to the limited number of edges within a machine cycle, selected events must occur between edges. The High–Speed Microcontroller employs sophisticated circuits to create half and quarter clock events. That is, some events occur between clock edges. Such circuits assure that events occur as precisely as if a clock edge were available. While being generally transparent to the user, these circuits result in the use of fractional clocks in the electrical specifications. For example, a time may be specified as 2.5 t_{CLCL}.

As mentioned above, a machine cycle is the basic timing unit of most functions in the High–Speed Microcontroller. A machine cycle of the High–Speed Microcontroller is the time required to execute a single cycle instruction. Almost half the opcodes (126 of 255) of the 8051 instruction set are implemented in a single machine cycle in the High–Speed Microcontroller. The remaining instructions require multiple machine cycles.

The Power Management Modes implemented on some devices modify the number of clock cycles needed to execute an instruction. Instead of 4 clocks per machine cycle, power management mode 1 (PMM1) and power management mode 2 (PMM2) utilize 64 and 1024 clocks per cycle respectively to conserve power. A full description of the power management modes and their effect on CPU operation is provided in Section 7.

All instructions are coded within an 8–bit field called an opcode. This single byte must be fetched from program memory. The opcode is decoded by the CPU. It determines what action the microcontroller will take and whether more information is needed from memory. If no other memory is needed, then only one byte was required. Thus the instruction is called a one byte instruction. In some cases, more data is needed. These will be two or three byte instructions.

In most cases, the number of memory accesses (bytes) needed by an instruction is equal to the number of machine cycles. Thus single cycle instructions contain one byte, and two cycle instructions have two bytes. This is true except for the special cases mentioned below.

SINGLE CYCLE INSTRUCTIONS

The standard single cycle instruction timing is shown in Figure 5–3. As mentioned above, there are 126 opcodes that are single cycle instructions. An example of a single cycle instruction is as follows:

DEC A 14h

TWO CYCLE INSTRUCTIONS

All two cycle instructions require two cycles because they involve two bytes or require two memory accesses. The first byte is an opcode that instructs the CPU. This is the instruction itself. The second byte is normally an operand or it specifies the location of the operand. For example, the instruction "ANL A, direct" uses two cycles and requires two bytes. Two examples are as follows:

ANL A, direct	55h a7–0
ANL A, #data	54h d7–d0

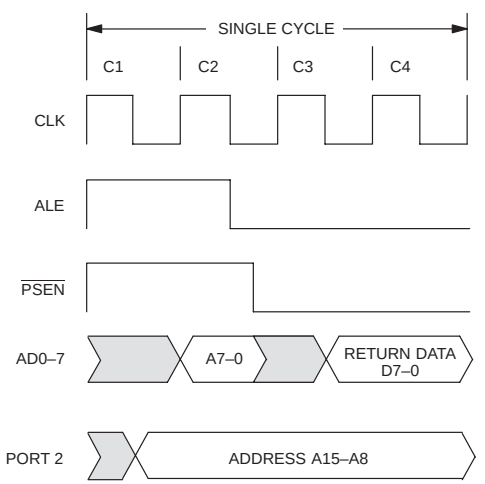
Note that in the first example, the first memory access is the opcode. The second memory access is the location of the operand in the register map. Since the result is stored in an internal register, this operation does not require a memory access. The second example is very similar. Again, the first byte represents the opcode. In this example, the second byte is the operand itself. This byte is used directly by the instruction. The timing for a two cycle instruction is shown in Figure 5–4.

One other type of two cycle instruction requires two cycles but only includes one byte. This is because the second memory access is the result of the instruction. These are the MOVX instructions. An example is as follows:

MOVX @DPTR, A F0h

The second cycle in this instruction is the write to data memory at the address pointed to by the data pointer. Thus this instruction is a two cycle one byte instruction, but requires two memory accesses. The MOVX timing is a special case, since the user can control it with the Stretch MOVX feature. The timing for the Stretch MOVX is discussed in the section on Memory Access.

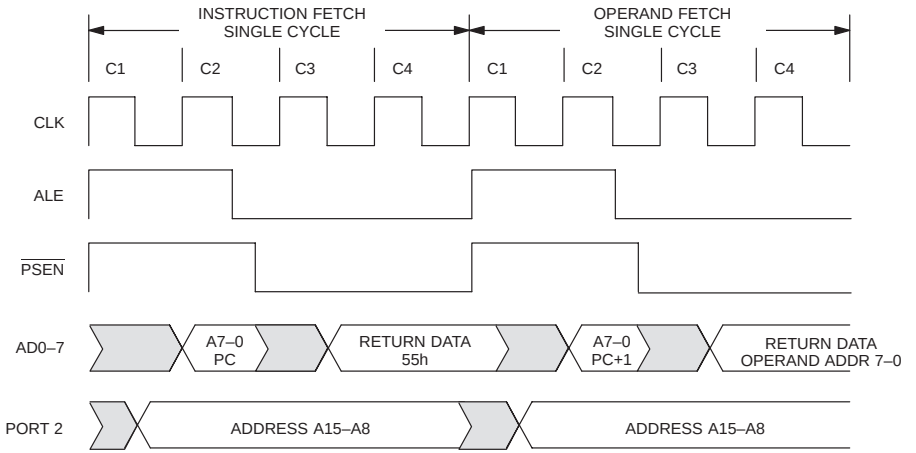
SINGLE CYCLE INSTRUCTION TIMING Figure 5–3



*Shaded areas are held in a weak latch on the port until overdriven.

TWO CYCLE INSTRUCTION TIMING Figure 5–4

Example: ANL A, direct : 55h addr7–0



*Shaded areas are held in a weak latch on the port until overdriven.

THREE CYCLE INSTRUCTIONS

Three cycle instructions come in two varieties. The first requires three memory accesses. These are similar to one and two cycle instructions in that the number of bytes equals the number of cycles.

The second variety is a three cycle instruction that simply requires 12 clocks to perform the function. This may have one or two bytes. Examples of both types are shown below.

ANL direct, #data	53h	(3 bytes)
	a_7–a_0	
	d7–d0	
SJMP rel	80h	(2 bytes)
	a_7–a_0	
INC DPTR	A3h	(1 byte)

In the first example, the first memory fetch is the opcode. The second is the location of the destination register. The third memory fetch is the operand that is used by the instruction. This instruction has three memory accesses, so it requires three machine cycles. The second example has the operand in the first byte and the jump location in the second. It requires three cycles to actually perform the jump. The third example contains simply the opcode, which is one byte. This instruction involves the manipulation of a 16–bit register so it takes longer than 8–bit operations. Figure 5–5 shows the timing of all three types of three cycle instructions.

FOUR CYCLE INSTRUCTIONS

All four cycle instructions require more time than the associated number of bytes. These are all program branching instructions that can move program control to a new location. The four cycle instructions use either 1 or 3 bytes as shown in the following examples. Figure 5–6 shows the timing of both four cycle instructions.

RET	22h
CJNE A, #data, addr	B4h
	d7–d0
	a7–a0

FIVE CYCLE INSTRUCTIONS

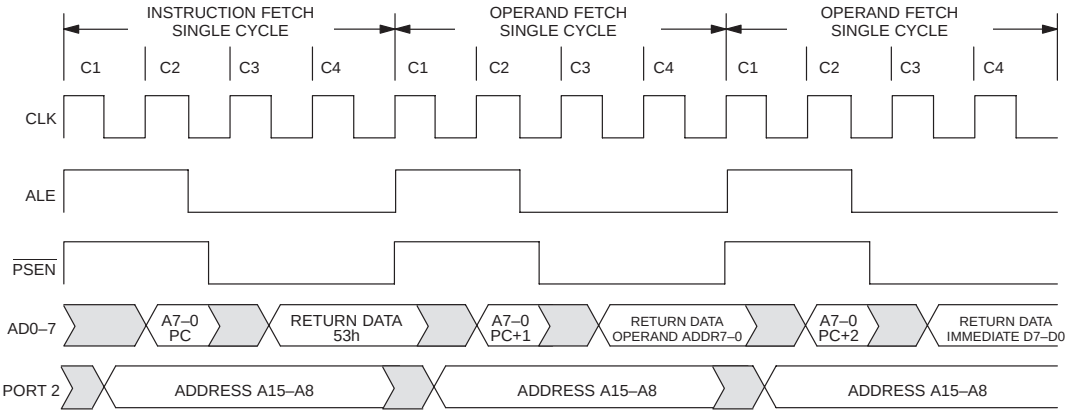
There are only two 5 cycle instructions in the High–Speed Microcontroller. They are the Multiply (MUL) and Divide (DIV). These are shown below. Figure 5–7 shows the timing of 5 cycle instructions.

MUL A, B	A4h
DIV A, B	84h

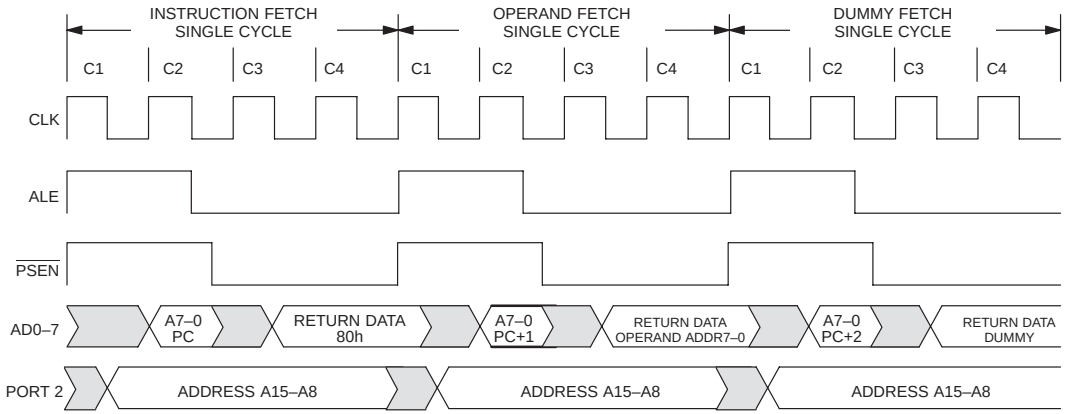
Note that the five cycle instructions require only one byte. They need 5 cycles to accomplish the math function.

THREE CYCLE INSTRUCTION TIMING Figure 5–5

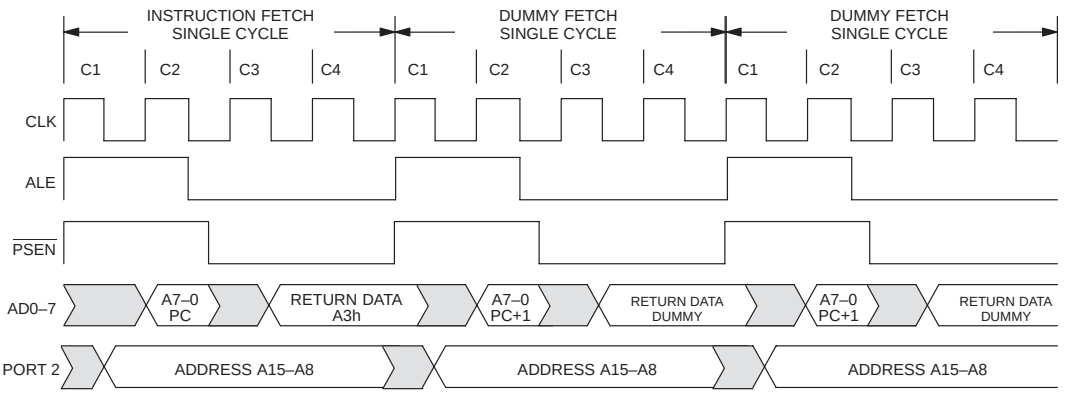
Example 2: ANL direct, #data 53h a7–a0 d7–d0



Example 2: SJMP rel 80h a7–a0



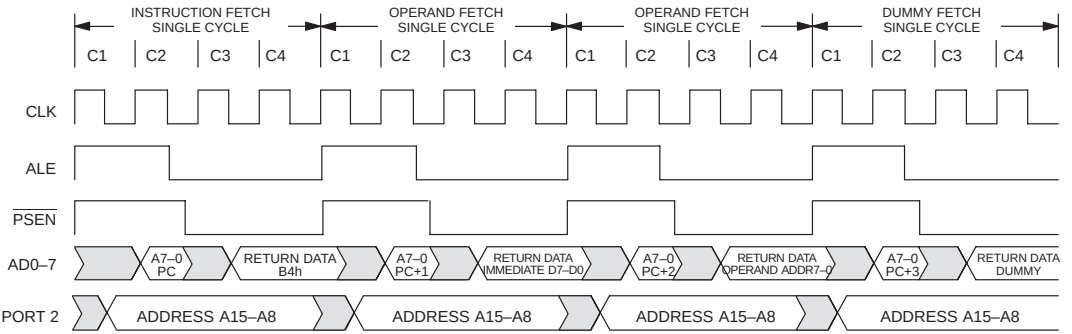
Example 3: INC DPTR A3h



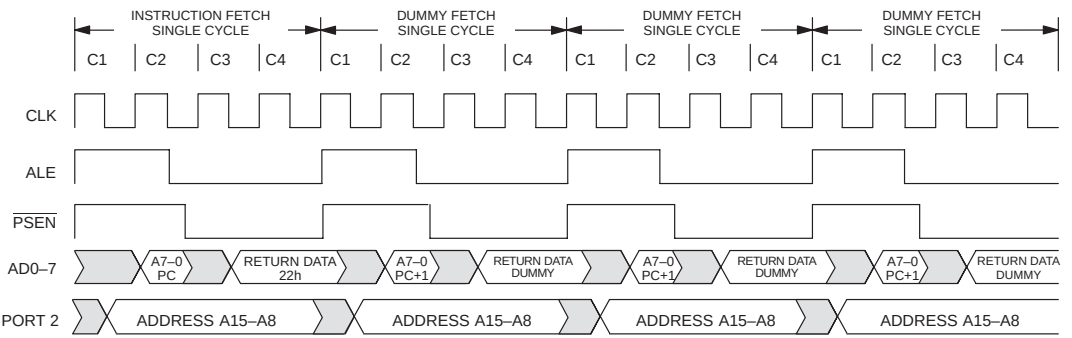
*Shaded areas are held in a weak latch on the port until overdriven.

FOUR CYCLE INSTRUCTION TIMING Figure 5–6

Example 1: CJNE A, #data, addr B4h d7–d0 a7–a0

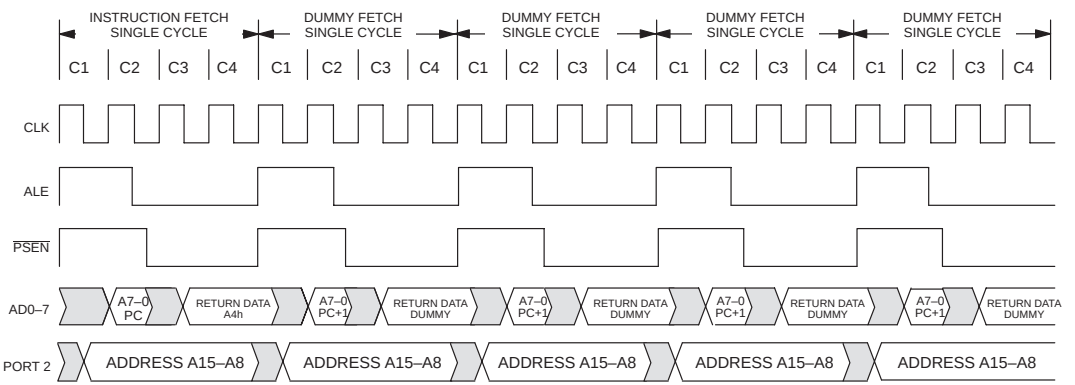


Example 2: RET 22h



FIVE CYCLE INSTRUCTION TIMING Figure 5–7

Example: MUL A,B A4h



*Shaded areas are held in a weak latch on the port until overdriven.

COMPARISON TO THE 8051

The original 8051 had a 12 clock architecture. A machine cycle needed 12 clocks and most instructions were either one or two machine cycles. Thus except for the MUL and DIV instructions, the 8051 used either 12 or 24 clocks for each instruction. Furthermore, each cycle in the 8051 used two memory fetches. In many cases the second fetch was a dummy, and the extra clock cycles were wasted.

The High–Speed Microcontroller uses 4 clocks per cycle. Since a cycle is now aligned with a memory fetch when possible, most instructions have the same number of cycles as bytes. This leads to more "categories" than the original 8051. Where there were primarily one and two cycle instructions before, there are now one, two, three, and four cycle instructions. Multiply and Divide require five cycles. Note however, that regardless of the number of cycles, each instruction is at least 1.5 and most are 2 to 3 times faster than its original counterpart. Table 5–1 shows each instruction, the number of clocks used in the High–Speed Microcontroller and the number used in the 8051 for comparison. The factor by which the High–Speed Microcontroller improves on the 8051 is shown as the Speed Advantage. A Speed Advantage of 3.0 means that the High–Speed Microcontroller performs the same instruction three times faster that the 8051.

Table 5–2 provides a summary by instruction type. Note that many of the instructions provide multiple opcodes. As an example, the ADD A, Rn instruction can act on one of 8 working registers. There are 8 opcodes for this instruction because it can be used on 8 independent locations. Table 5–2 shows totals for both number of instructions and number of opcodes. Averages are provided in the tables. However, the real speed improvement seen in any system will depend on the instruction mix. Programs that use immediate or direct data combined with the accumulator or working registers will be improved the least. These are two cycle, two byte instructions. Moderate performance improvement will be gained by emphasizing short branches and instructions that use only direct and immediate data (no accumulator or working register). These instructions tend to be three cycle instructions. The largest number of improvements come from the single cycle instructions involving only the accumulator and working registers. Also, the two cycle data movement instructions involving the working registers are greatly improved.

INSTRUCTION TIMING COMPARISON Table 5–1

* High–Speed Microcontroller is abbreviated as HSM.

INSTRUCTION	HEX CODE	HSM CLOCK CYCLES	HSM TIME @ 25 MHz	8051 CLOCK CYCLES	8051 TIME @25 MHz	HSM vs. 8051 SPEED ADVANTAGE
ADD A, Rn	28..2F	4	160ns	12	480ns	3
ADD A, direct	25	8	320ns	12	480ns	1.5
ADD A, @Ri	26..27	4	160ns	12	480ns	3
ADD A, #data	24	8	320ns	12	480ns	1.5
ADDC A, Rn	38..3F	4	160ns	12	480ns	3
ADDC A, direct	35	8	320ns	12	480ns	1.5
ADDC A, @Ri	36..37	4	160ns	12	480ns	3
ADDC A, #data	34	8	320ns	12	480ns	1.5
SUBB A, Rn	98..9F	4	160ns	12	480ns	3
SUBB A, direct	95	8	320ns	12	480ns	1.5
SUBB A, @Ri	96..97	4	160ns	12	480ns	3
SUBB A, #data	94	8	320ns	12	480ns	1.5
INC A	04	4	160ns	12	480ns	3
INC Rn	08..0F	4	160ns	12	480ns	3
INC direct	05	8	320ns	12	480ns	1.5
INC @Ri	06..07	4	160ns	12	480ns	3
INC DPTR	A3	12	480ns	24	960ns	2
DEC A	14	4	160ns	12	480ns	3
DEC Rn	18..1F	4	160ns	12	480ns	3
DEC direct	15	8	320ns	12	480ns	1.5

DEC @Ri	16..17	4	160ns	12	480ns	3
MUL AB	A4	20	800ns	48	1.92us	2.4
DIV AB	84	20	800ns	48	1.92us	2.4
DA A	D4	4	160ns	12	480ns	3
ANL A, Rn	58..5F	4	160ns	12	480ns	3
ANL A, direct	55	8	320ns	12	480ns	1.5
ANL A, @Ri	56..57	4	160ns	12	480ns	3
ANL A, #data	54	8	320ns	12	480ns	1.5
ANL direct, A	52	8	320ns	12	480ns	1.5
ANL direct, #data	53	12	480ns	24	960ns	2
ORL A, Rn	48..4F	4	160ns	12	480ns	3
ORL A, direct	45	8	320ns	12	480ns	1.5
ORL A, @Ri	46..47	4	160ns	12	480ns	3
ORL A, #data	44	8	320ns	12	480ns	1.5
ORL direct, A	42	8	320ns	12	480ns	1.5
ORL direct, #data	43	12	480ns	24	960ns	2
XRL A, Rn	68..6F	4	160ns	12	480ns	3
XRL A, direct	65	8	320ns	12	480ns	1.5
XRL A, @Ri	66..67	4	160ns	12	480ns	3
XRL A, #data	64	8	320ns	12	480ns	1.5
XRL direct, A	62	8	320ns	12	480ns	1.5
XRL direct, #data	63	12	480ns	24	960ns	2
CLR A	E4	4	160ns	12	480ns	3
CPL A	F4	4	160ns	12	480ns	3
RL A	23	4	160ns	12	480ns	3
RLC A	33	4	160ns	12	480ns	3
RR A	03	4	160ns	12	480ns	3
RRC A	13	4	160ns	12	480ns	3
SWAP A	C4	4	160ns	12	480ns	3
MOV A, Rn	E8..EF	4	160ns	12	480ns	3
MOV A, direct	E5	8	320ns	12	480ns	1.5
MOV A, @Ri	E6..E7	4	160ns	12	480ns	3
MOV A, #data	74	8	320ns	12	480ns	1.5
MOV Rn, A	F8..FF	4	160ns	12	480ns	3
MOV Rn, direct	A8..AF	8	320ns	24	960ns	3
MOV Rn, #data	78..7F	8	320ns	12	480ns	1.5
MOV direct, A	F5	8	320ns	12	480ns	1.5
MOV direct, Rn	88..8F	8	320ns	24	960ns	3
MOV direct, direct	85	12	480ns	24	960ns	2
MOV direct, @Ri	86..87	8	320ns	24	960ns	3
MOV direct, #data	75	12	480ns	24	960ns	2
MOV @Ri, A	F6..F7	4	160ns	12	480ns	3
MOV @Ri, direct	A6..A7	8	320ns	24	960ns	3
MOV @Ri, #data	76..77	8	320ns	12	480ns	1.5
MOV DPTR, #data16	90	12	480ns	24	960ns	2
MOVC A, @A+DPTR	93	12	480ns	24	960ns	2
MOVC A, @A+PC	83	12	480ns	24	960ns	2
MOVX A, @Ri	E2..E3	8	320ns	24	960ns	3
MOVX A, @DPTR	E0	8	320ns	24	960ns	3
MOVX @Ri, A	F2..F3	8	320ns	24	960ns	3

MOVX @DPTR, A	F0	8	320ns	24	960ns	3
PUSH direct	C0	8	320ns	24	960ns	3
POP direct	D0	8	320ns	24	960ns	3
XCH A, Rn	C8..CF	4	160ns	12	480ns	3
XCH A, direct	C5	8	320ns	12	480ns	1.5
XCH A, @Ri	C6..C7	4	160ns	12	480ns	3
XCHD A, @Ri	D6..D7	4	160ns	12	480ns	3
CLR C	C3	4	160ns	12	480ns	3
CLR bit	C2	8	320ns	12	480ns	1.5
SETB C	D3	4	160ns	12	480ns	3
SETB bit	D2	8	320ns	12	480ns	1.5
CPL C	B3	4	160ns	12	480ns	3
CPL bit	B2	8	320ns	12	480ns	1.5
ANL C, bit	82	8	320ns	24	960ns	3
ANL C, /bit	B0	8	320ns	24	960ns	3
ORL C, bit	2	8	320ns	24	960ns	3
ORL C, /bit	A0	8	320ns	24	960ns	3
MOV C, bit	A2	8	320ns	12	480ns	1.5
MOV bit, C	92	8	320ns	24	960ns	3
ACALL addr 11	Hex code					
Hex codes=11, 31, 51, 71, 91, B1, D1, or F1	Byte 1	12	480ns	24	960ns	2
LCALL addr 16	12	16	640ns	24	960ns	1.5
RET	22	16	640ns	24	960ns	1.5
RETI	32	16	640ns	24	960ns	1.5
AJMP addr 11	Hex code					
Hex code=01, 21, 41, 61, 81, A1, C1, or E1	Byte 1	12	480ns	24	960ns	2
LJMP addr 16	2	16	480ns	24	960ns	1.5
JMP @A+DPTR	73	12	480ns	24	960ns	2
SJMP rel	80	12	480ns	24	960ns	2
JZ rel	60	12	480ns	24	960ns	2
JNZ rel	70	12	480ns	24	960ns	2
JC rel	40	12	480ns	24	960ns	2
JNC rel	50	12	480ns	24	960ns	2
JB bit, rel	20	16	640ns	24	960ns	1.5
JNB bit, rel	30	16	640ns	24	960ns	1.5
JBC bit, rel	10	16	640ns	24	960ns	1.5
CJNE A, direct, rel	B5	16	640ns	24	960ns	1.5
CJNE A, #data, rel	B4	16	640ns	24	960ns	1.5
CJNE Rn, #data, rel	B8..BF	16	640ns	24	960ns	1.5
CJNE @Ri, #data, rel	B6..B7	16	640ns	24	960ns	1.5
DJNZ Rn, rel	D8..DF	12	480ns	24	960ns	2
DJNZ direct, rel	D5	16	640ns	24	960ns	1.5
NOP	00	4	160ns	12	480ns	3

INSTRUCTION SPEED SUMMARY Table 5–2

INSTRUCTION CATEGORY		QUANTITY	SPEED ADVANTAGE
Total Instructions:	One Cycle One Byte	37	3.0
Total Instructions:	Two Cycle One Byte	4	3.0
Total Instructions:	Two Cycle Two Bytes X1.5	27	1.5
Total Instructions:	Two cycle Two Bytes X3.0	11	3.0
Total Instructions:	Three Cycle One Byte	4	2.0
Total Instructions:	Three Cycle Two Bytes	8	2.0
Total Instructions:	Three Cycle Three Bytes	7	2.0
Total Instructions:	Four Cycle One Byte	2	1.5
Total Instructions:	Four Cycle Three Bytes	9	1.5
Total Instructions:	Five Cycle One Byte	2	2.4
Average Across all Instructions		**111**	**2.3**
Total Opcodes:	One Cycle One Byte	126	3.0
Total Opcodes:	Two Cycle One Byte	6	3.0
Total Opcodes:	Two Cycle Two Bytes X1.5	35	1.5
Total Opcodes:	Two Cycle Two Bytes X3.0	27	3.0
Total Opcodes:	Three Cycle One Byte	4	2.0
Total Opcodes:	Three Cycle Two Bytes	29	2.0
Total Opcodes:	Three Cycle Three Bytes	7	2.0
Total Opcodes:	Four Cycle One Byte	2	1.5
Total Opcodes:	Four Cycle Three Bytes	17	1.5
Total Opcodes:	Five Cycle One Byte	2	2.4
Average Across all Instructions		**255**	**2.5**

SECTION 6: MEMORY ACCESS

The High–Speed Microcontroller follows the memory interface convention established for the industry standard 80C51/80C31. Products in the family may vary, so refer to the specific product data sheet for any potential differences. Like the 8051 series, the High–Speed Microcontroller uses two memory segments. These are program memory and data memory. Program memory is read–only and is usually implemented in ROM or EPROM. Data memory is read/write and is commonly implemented in SRAM.

Memory areas can be implemented as either on–chip, off–chip, or a combination. When using devices without internal program memory, or if the maximum address of on–chip program or data memory is exceeded, the device will perform an external memory access using the Expanded memory bus on ports 0 and 2. While serving as a memory bus, port 0 and port 2 do not function as I/O ports, following the standard 8051 convention of addressing external memory. The $\overline{PSEN}$ signal will go active low to serve as a chip enable or output enable when performing a code fetch from external memory. Products with no on–chip program memory such as the DS80C320 will always use the Expanded bus. These devices have no Port 0 latch since the port is dedicated for memory operations. Devices which incorporate on–chip MOVX data memory operate in a similar fashion, except that the $\overline{RD}$ and $\overline{WR}$ signals serve as chip enables when accessing an external SRAM.

Program execution begins at address 0000h. This is the reset vector. Any reset will cause the next program fetch to begin at this location. Subsequent branches and interrupts determine how the memory fetch deviates from sequential addressing. Since all programs begin at 0000h, this will be the beginning address of all program execution. If on–chip program memory is present, program execution will begin at internal location 0000h, otherwise external program memory will be used.

INTERNAL PROGRAM MEMORY

Some members of the High–Speed Microcontroller family incorporate internal EPROM or ROM for program storage. On–chip program memory begins at address 0000h and is contiguous through the amount of on–chip memory. Exceeding the maximum address of on–chip memory will cause the device to perform an external memory access using the Expanded memory bus on ports 0 and 2. For example, if the on–chip program memory is 16KB, then it lies between 0000h and 3FFFh in a contiguous area. Thus a fetch at program memory location 4000h would be directed to the Expanded bus. Restricting memory operations within the on–chip memory allows ports 0 and 2 to be used for general purpose I/O. For more information concerning memory size for a specific device, consult the specific data sheet.

The High–Speed Microcontroller family was designed to be compatible with industry standard 87C51FB programming tools. A number of third–party device programmers are available which support Dallas Semiconductor products. In addition, Dallas Semiconductor manufactures the DS87000 Microcontroller Programmer, specifically designed for EPROM–based members of the High–Speed Microcontroller family.

INTERNAL DATA MEMORY

Some members of the High–Speed Microcontroller family incorporate internal SRAM for additional data storage. This memory is addressed via MOVX commands, and is in addition to the 256 bytes of scratchpad memory. On–chip data memory begins at address 0000h and is contiguous through the amount of on–chip memory. Exceeding the maximum address of on–chip memory will cause the device to perform an external memory access using the Expanded memory bus on ports 0 and 2. For example, if the on–chip program memory is 1KB, then it lies between 0000h and 03FFh in a contiguous area. Thus a MOVX instruction affecting memory location 0400h would be directed to the Expanded bus.

Another advantage of internal data memory is that it guarantees a 2 machine cycle data memory access. This data can be made nonvolatile on the DS87C530 through the use of an external battery. Restricting memory operations within the on–chip memory allows ports 0 and 2 to be used for general purpose I/O. For more information concerning memory size for a specific device, consult the corresponding data sheet.

Upon a power–on reset, the internal data memory area is disabled and transparent to the system map. Any memory access between 0000h and FFFFh will be directed to the Expanded bus. This allows the device to remain drop–in compatible with existing 87C52 designs. To enable the internal SRAM area, software must configure the Data Memory Enable bits DME1, DME0 (PMR.1–0). The three memory configurations shown in Table 6–1 are supported to allow either external data memory access via the expanded bus, internal data

memory access, or read–only access to the EPROM System Control Byte. Note that these bits are cleared after a reset, so access to the internal data memory is prohibited until these bits are modified. The contents of internal data memory are not affected by the changing of the Data Memory Enable bits.

DATA MEMORY ACCESS CONTROL Table 6–1

DME1	DME0	DATA MEMORY ADDRESS RANGE	DATA MEMORY LOCATION
0	0	0000h–FFFFh	External Data Memory (default)
0	1	0000h–03FFh 0400h–FFFFh	Internal Data Memory External Data Memory
1	0	Reserved	Reserved
1	1	0000h–03FFh 0400h–FFFBh FFFCh FFFDh–FFFFh	Internal Data Memory Reserved System Control Byte (Read only) Reserved

ROMSIZE FEATURE

Members of the High–Speed Microcontroller family which incorporate internal program memory allow the system to dynamically vary the on–chip memory size. This permits the device to reconfigure the upper limit of on–chip memory, allowing a portion of the memory to be mapped off–chip. The size of on–chip memory can vary from 0KB to the full range of memory, allowing the device to behave like a device with less on–chip memory.

This feature has two primary uses. In the first instance, it allows the device to act as a bootstrap loader for a Flash memory or nonvolatile SRAM (NVSRAM). The internal program memory can contain a bootstrap loader, which can program the external memory device. Secondly, this method can be used to increase the amount of available program memory from 64KB to 80KB without bank–switching.

The maximum amount of on–chip memory is selected by configuring the ROM Size Select register bits RMS2, RMS1, RMS0 (ROMSIZE.2–0). The modification of the ROMSIZE register must be followed by a 2 machine cycle delay, such as executing two NOP instructions, before jumping to the new address range. Interrupts must be disabled during this operation, because a jump to the interrupt vector during the changing of the memory map can cause erratic results. In addition, modification of the ROMSIZE register must be done from a location that will be valid both before and after the on–chip memory configuration. If off–chip memory access is planned, it is recommended that ports 0 and 2 not be used as general purpose I/O, as their state will be disturbed by the memory operations. The settings for the ROM Size Select register are shown in Table 6–2. Note that the memory configurations shown are not available on all devices.

ROMSIZE REGISTER SETTINGS Table 6–2

RMS2	RMS1	RMS0	Max. On–chip ROM
0	0	0	0KB
0	0	1	1KB
0	1	0	2KB
0	1	1	4KB
1	0	0	8KB
1	0	1	16KB
1	1	0	32KB
1	1	1	64KB

After reset, a device with internal program memory will reset the ROMSIZE bits to their default setting. This will be the maximum amount of on–chip memory for that device. The procedure to reconfigure the amount of on–chip memory is as follows:

1. Jump to a location in program memory that will be unaffected by the change,
2. Disable interrupts by clearing the EA bit (IE.7),
3. Write AAh to the Timed Access Register (TA;C7h),
4. Write 55h to the Timed Access Register (TA;C7h),
5. Modify the ROM Size Select bits (RMS2–RMS0),

6. Delay 2 machine cycles (2 NOP instructions),
7. Enable interrupts by setting the EA bit (IE.7).

If the 0KB of internal program memory setting is selected, extra precautions must be taken. In this case, it will be necessary to duplicate the interrupt vector table in external program memory. This is because the interrupt vector table is located in the lower 1KB of memory, and the device will automatically redirect any fetches from the interrupt vector table to external memory. Care must be exercised when assembling or compiling the program so that all the modules are located at the correct starting address, including the interrupt vector table.

PROGRAM MEMORY INTERCONNECT

The program memory interconnect scheme for the High–Speed Microcontroller family is shown in Figure 6–1. This example uses the DS80C320 and one 32K x 8 EPROM. The Program Store Enable ($\overline{PSEN}$) signal is used to provide an output enable to the EPROM. It can also be used to provide a chip enable, but this produces less favorable timing. The address LSB and data are multiplexed on port 0, and the address MSB is provided on port 2. An external latch, shown in the diagram as a 74F373, is used to latch the lower byte of the address to the memory device. The Address Latch Enable (ALE) signal controls the timing of the latch so that the operation is performed in the proper sequence. The signals and relative timing for a program access are shown in Figure 6–2.

When implementing a high speed memory interface, the F series (or faster) logic should be used. HC logic will have worst case propagation delays that are too long. Specifications for all devices should be checked. More information on memory interface timing can be found in Application Note 57, DS80C320 Memory Interface Timing, and Application Note 85, High Speed Microcontroller Interface Timing.

The first product in the family, the DS80C320, provides an extremely high speed interface to external ROM or EPROM. This assures that the user can use the slowest, and least expensive, memory device for a given crystal speed. The DS80C320 provides very fast slew rates, but controls ringing and overshoot. Fast slew rates allow the maximum possible time for memory access. In most cases, however, these aspects will be transparent to the user. Refer to the electrical specifications for exact timing of each product.

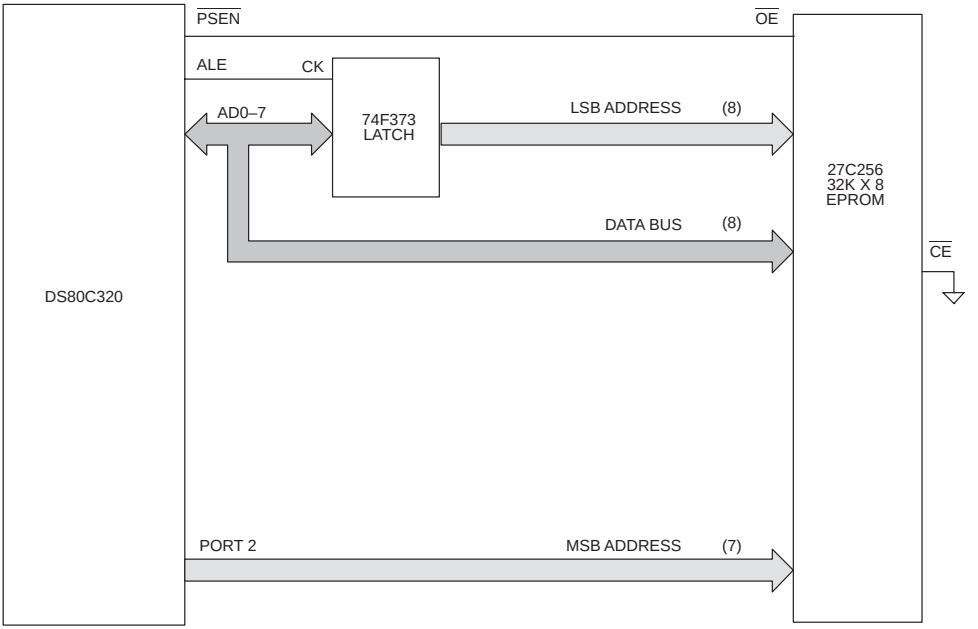
DATA MEMORY INTERCONNECT

As described in Section 4, the High–Speed Microcontroller provides a small amount of RAM mapped as registers for on–chip direct access. This is not considered data memory and does not fall into the memory map. Systems that require more RAM or memory mapped peripherals must use the data memory area. This segment is a 64KB space located between 0000h and FFFFh. It is reached using the MOVX instruction. Any use of this instruction automatically accesses the data area. Although, the original 8051 convention placed all data memory off–chip, many members of the High–Speed Microcontroller family contain some data memory on–chip.

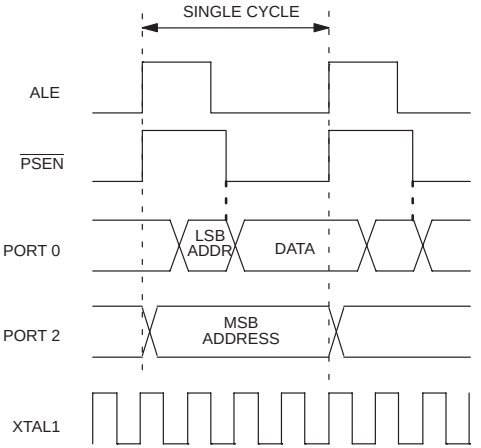
From a software standpoint, the physical location of the data area is not relevant because the same instructions are used. Like the program segment, if software accesses a data address that is above the on–chip data area, this access will automatically be routed to the Expanded bus. Thus data or peripherals that are off–chip can be used in conjunction with on–chip memory by selecting addresses that do not overlap. As an example, if the microcontroller has 1KB of on–chip data memory, then a MOVX instruction at location 0400h will be directed off–chip via the Expanded bus.

The physical connection of off–chip data memory is shown in Figure 6–3. This illustrates a DS80C320 with interfaced with an 8K SRAM. The data memory map begins at address 0000h since the DS80C320 has no on–chip data memory. A similar interconnection scheme would be implemented if a device with internal data memory, such as the DS87C520 would be used. Note that any external memory that overlapped the range of on–chip data memory would not be used.

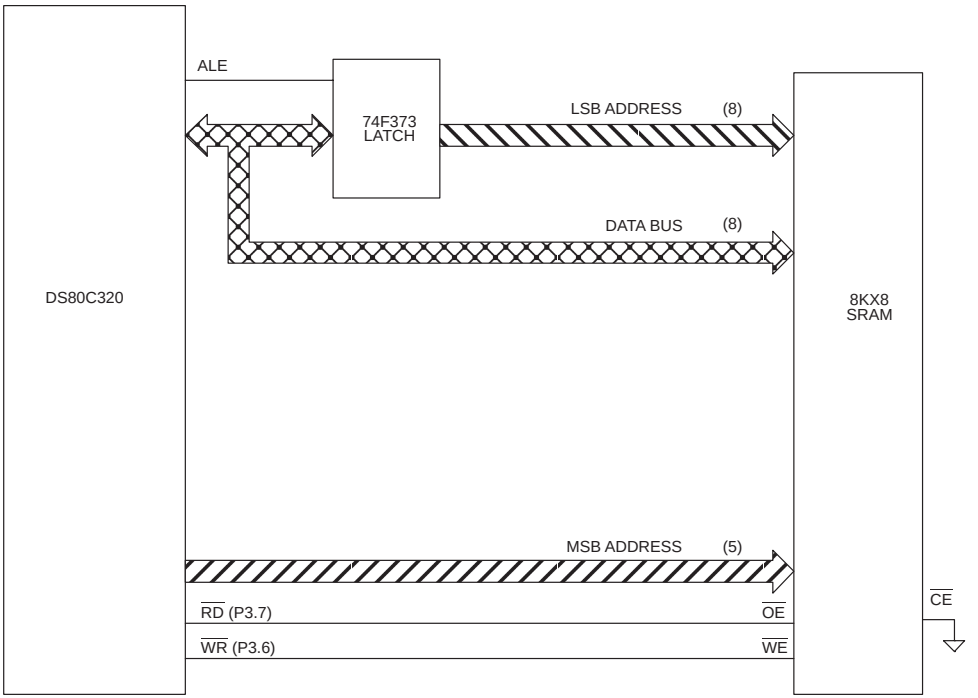
PROGRAM MEMORY INTERFACE Figure 6–1



PROGRAM MEMORY SIGNALS Figure 6–2



DATA MEMORY INTERFACE Figure 6–3



DATA MEMORY ACCESS

As mentioned above, the High–Speed Microcontroller uses the MOVX instruction for data memory access. This includes off–chip RAM and memory mapped peripherals needing read/write access. Several aspects of the MOVX operation have been enhanced as compared to the original 8051. The principal improvements are in the areas of the MOVX timing and the Data Pointer.

The MOVX instruction is used to generate read/write access to off–chip address locations. It has several addressing modes. The MOVX can be used to reach a 256 byte block by using the MOVX @ Ri command. This instruction uses the value in the designated working register to address one of 256 locations. The upper byte of the address is supplied by the value in the Port 2 latch. A second way to access data is the Data Pointer (DPTR). This 16–bit register provides an absolute address for data memory access. 16–bits cover the entire 64KB area. Thus the DPTR serves as a pointer to memory. Using the DPTR, the relevant instruction is MOVX @DPTR.

The original 8051 contained one DPTR. While this provides access to the entire memory area, it is difficult to move data from one address to another. The High–Speed Microcontroller provides two Data Pointers. Thus software can load both a source and a destination address. The MOVX instruction will use the active pointer to direct the off–chip address. The active pointer is selected by the Data Pointer Select (DPS).

The Data Pointers are called DPTR0 and DPTR1. DPTR0 is located at SFR addresses 82h and 83h. These are the locations used by the original 8051. No modification of standard code is needed to use DPTR0. The new DPTR is located at SFR 84h and 85h. The Data Pointer Select bit (DPS) chooses the active pointer and is located at the LSb of the SFR location 86h. No other bits in register 86h have any effect and are set to 0. When DPS is set to 0, the DPTR0 is active. When set to 1, DPTR1 is used.

The user switches between data pointers by toggling the DPS bit (LSb of register 86h). The INC instruction is the fastest way to accomplish this. All DPTR–related instructions use the currently selected DPTR for any activity. Therefore only one instruction is required to switch from a source to a destination address. Using the Dual Data Pointer saves code from needing to save source and destination addresses when doing a block move. Once loaded , the software simply switches between DPTR0 and DPTR1. Sample code listed below illustrates the saving from using the dual DPTR. The relevant register locations are summarized as follows.

DPL	82h	Low byte original DPTR
DPH	83h	High byte original DPTR
DPL1	84h	Low byte new DPTR
DPH1	85h	High byte new DPTR
DPS	86h	DPTR Select (LSb)

The example program listed below was original code written for an 8051 and requires a total of 1869 machine cycles on the DS80C320. This takes 299 µs to execute at 25 MHz. The new code using the Dual DPTR requires only 1097 machine cycles taking 175.5 µs. The Dual DPTR saves 772 machine cycles or 123.5 µs for a 64 byte block move. Since each pass through the loop saves 12 machine cycles when compared to the single DPTR approach, larger blocks gain more efficiency using this feature.

A typical application of the Dual Data Pointer is moving data from an external RAM to a memory mapped display. Another application would be to retrieve data from a stored table, process it using a software algorithm, then store the result in a new table.

64 BYTE BLOCK MOVE WITH DUAL DATA POINTER

```
; SH and SL are high and low byte source address.
; DH and DL are high and low byte of destination address.
; DPS is the data pointer select. Reset condition is DPS=0, DPTR0 is selected.
                                                                    # CYCLES
DPS     EQU 86h                 ; TELL ASSEMBLER ABOUT DPS
MOV     R5, #64                 ; NUMBER OF BYTES TO MOVE           2
MOV     DPTR, #DHDL             ; LOAD DESTINATION ADDRESS          3
INC     DPS                     ; CHANGE ACTIVE DPTR                2
MOV     DPTR, #SHSL             ; LOAD SOURCE ADDRESS               2

MOVE:
; THIS LOOP IS PERFORMED THE NUMBER OF TIMES LOADED INTO R5, IN THIS EXAMPLE 64
MOVX    A, @DPTR                ; READ SOURCE DATA BYTE             2
INC     DPS                     ; CHANGE DPTR TO DESTINATION        2
MOVX    @DPTR, A                ; WRITE DATA TO DESTINATION         2
INC     DPTR                    ; NEXT DESTINATION ADDRESS          3
INC     DPS                     ; CHANGE DATA POINTER TO SOURCE     2
INC     DPTR                    ; NEXT SOURCE ADDRESS               3
DJNZ    R5, MOVE                ; FINISHED WITH TABLE?              3
```

64 BYTE BLOCK MOVE WITHOUT DUAL DATA POINTER

```
; SH and SL are high and low byte source address.
; DH and DL are high and low byte of destination address.
                                                                    # CYCLES
MOV     R5, #64d                ; NUMBER OF BYTES TO MOVE           2
MOV     DPTR, #SHSL             ; LOAD SOURCE ADDRESS               3
MOV     R1, #SL                 ; SAVE LOW BYTE OF SOURCE           2
MOV     R2, #SH                 ; SAVE HIGH BYTE OF SOURCE          2
MOV     R3, #DL                 ; SAVE LOW BYTE OF DESTINATION      2
MOV     R4, #DH                 ; SAVE HIGH BYTE OF DESTINATION     2

MOVE:
; THIS LOOP IS PERFORMED THE NUMBER OF TIMES LOADED INTO R5, IN THIS EXAMPLE 64
MOVX    A, @DPTR                ; READ SOURCE DATA BYTE             2
MOV     R1, DPL                 ; SAVE NEW SOURCE POINTER           2
MOV     R2, DPH                 ;                                   2
MOV     DPL, R3                 ; LOAD NEW DESTINATION              2
MOV     DPH, R4                 ;                                   2
MOVX    @DPTR, A                ; WRITE DATA TO DESTINATION         2
INC     DPTR                    ; NEXT DESTINATION ADDRESS          3
MOV     R3, DPL                 ; SAVE NEW DESTINATION POINTER      2
MOV     R4, DPH                 ;                                   2
MOV     DPL, R1                 ; GET NEW SOURCE POINTER            2
MOV     DPH, R2                 ;                                   2
INC     DPTR                    ; NEXT SOURCE ADDRESS               3
DJNZ    R5, MOVE                ; FINISHED WITH TABLE?              3
```

DATA MEMORY TIMING

Data memory timing refers to the execution of the MOVX instruction. This instruction includes a program fetch memory access, then a read or write memory access. The program fetch for a MOVX instruction is no different from any other instruction. The unique timing occurs for the second memory operation when data is accessed.

As described in Section 5, the High–Speed Microcontroller uses four oscillator clocks for each machine cycle. A machine cycle involves one memory access. Generally, an instruction using two memory accesses would be a two machine cycle instruction (except for branches, MUL, DIV, INC DPTR, MOVC, and MOVX). The MOVX instruction is unique in that the user determines the time allowed for a data memory access. This feature is called the Stretch MOVX instruction.

The High–Speed Microcontroller allows the application software to adjust the speed of data memory access. The microcontroller is capable of performing the MOVX in as little as two machine cycles. Since one machine cycle is used for the program fetch, this leaves one machine cycle to perform the actual data memory access. However, this value can be adjusted as needed so that both fast memory and slow memory or peripherals can be accessed with no glue logic. Even in high–speed systems, it may not be necessary to perform data memory access at full speed. In addition, there are a variety of slower memory mapped peripherals such as LCD displays or UARTs.

When using a MOVX instruction, the user controls the time for which a read or write strobe is kept active. Setup and hold times are also adjusted. Thus the Stretch value will be selected to provide a long enough memory strobe to satisfy the access time of the target device.

The Stretch MOVX is controlled by a value in a special function register described below. This allows the user to select a stretch value between zero and seven. A Stretch of zero will result in a two machine cycle MOVX. This leaves one machine cycle to actually read or write data. A Stretch of seven will result in a MOVX of nine cycles. The time is added to the middle of the memory strobe, creating a very long read or write cycle. The Stretch value can be changed dynamically under software control depending on the type of memory or peripheral to be accessed.

On reset, the Stretch value will default to a one, resulting in a three cycle MOVX. Therefore, data memory access will not be performed at full speed. This is a convenience to existing designs that may not have fast RAM in place. When maximum speed is desired, the software should select a Stretch value of zero. Note that faster RAMs will be needed. When using very slow RAM or peripherals, a larger stretch value can be selected. Note that this affects data memory only and the only way to slow program memory (ROM) access is to use a slower crystal.

Using a Stretch value between one and seven results in a wider read/write strobe allowing more time for memory/peripherals to respond. The microcontroller stretches the read/write strobe and all related timing. The full speed access is shown in Figure 6–4. Note that this is not the reset default case. A three cycle MOVX is shown in Figure 6–5A. This is the reset default condition. To modify the MOVX timing, the Stretch value in the Clock Control register described below must be changed. Figure 6–5B shows the timing for a four cycle MOVX (Stretch=2).

Table 6–1 below shows the resulting strobe widths for each Stretch value. The memory stretch is implemented using the Clock Control Special Function Register at SFR location 8Eh. The stretch value is selected using bits CKCON.2–0. In the table, these bits are referred to as M2 through M0. Note that the Stretch time can be dynamically varied, allowing fast RAMs but slow peripherals. The first stretch allows the use of common 120 ns or 150 ns RAMs without dramatically lengthening the memory access.

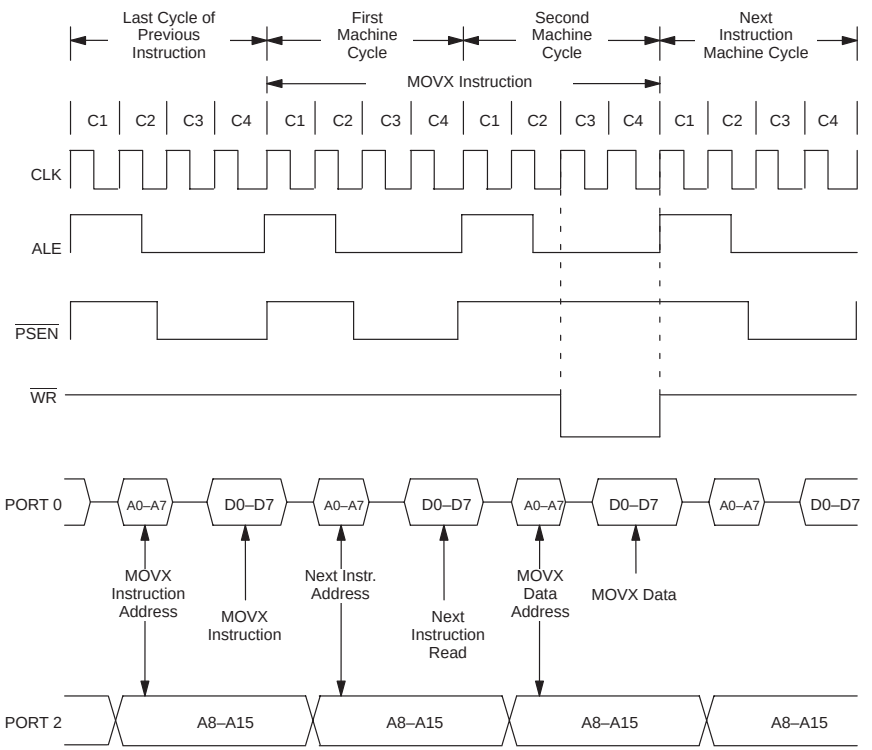
Note that the first Stretch value does not follow the pattern of adding four clocks to the strobe. This is because the first Stretch uses one clock to create additional set–up and one clock to create additional hold time. Systems using a Stretch cycle of zero are presumed to be fast enough or to be running at a slower clock speed. Since the Stretch is based on crystal timing, the resulting pulse widths must be viewed on the basis of the real system timing.

DATA MEMORY CYCLE STRETCH VALUES Table 6–1

CKCON.2–0			MEMORY CYCLES	$\overline{RD}$ OR $\overline{WR}$ STROBE WIDTH		
M2	M1	M0		IN CLOCKS	t @ 25 MHz	t @ 12 MHz
0	0	0	2	2	80 ns	167 ns
0	0	1	3 (default)	4	160 ns	333 ns
0	1	0	4	8	320 ns	667 ns
0	1	1	5	12	480 ns	1000 ns
1	0	0	6	16	640 ns	1333 ns
1	0	1	7	20	800 ns	1667 ns
1	1	0	8	24	960 ns	2000 ns
1	1	1	9	28	1120 ns	2333 ns

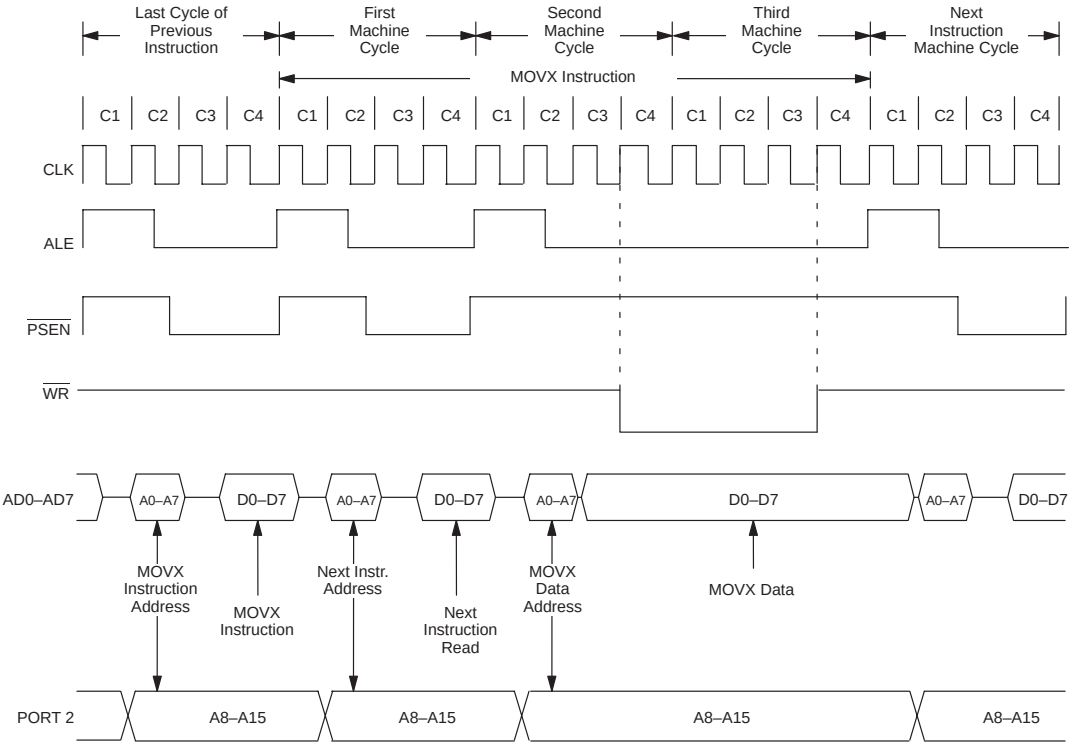
Note: These numbers represent nominal values. Actual timing may vary slightly.

FULL SPEED MOVX INSTRUCTION Figure 6–4



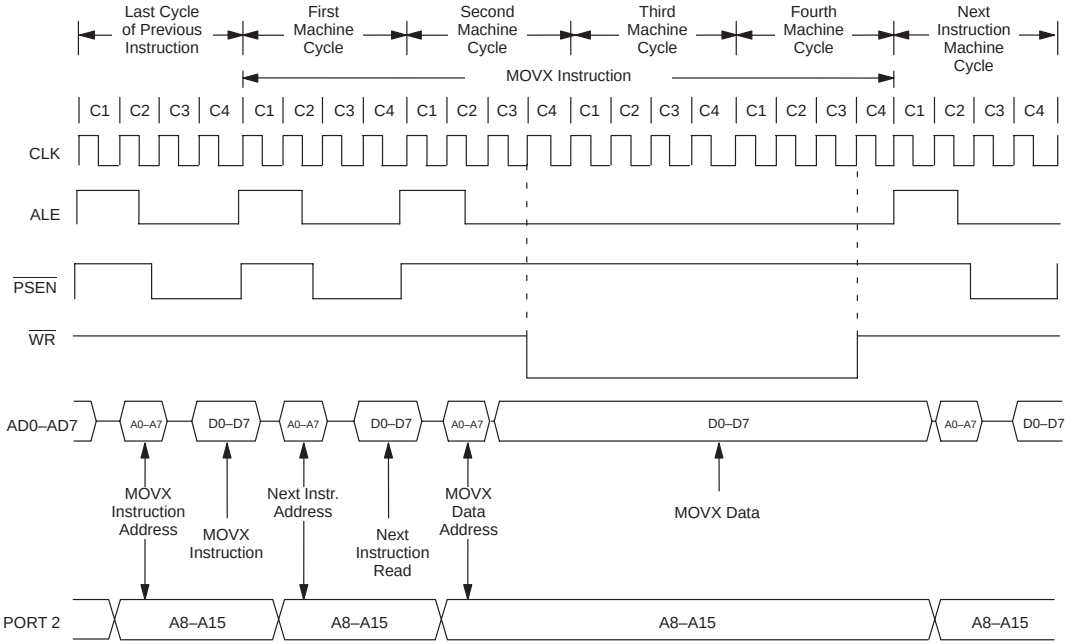
DATA MEMORY WRITE (2 CYCLE, STRETCH=0)

THREE CYCLE MOVX INSTRUCTION Figure 6–5a



THREE CYCLE DATA MEMRORY WRITE (RESET DEFAULT)
STRETCH VALUE=1

FOUR CYCLE MOVX INSTRUCTION Figure 6–5b



FOUR CYCLE DATA MEMORY WRITE
STRETCH VALUE=2

SECTION 7: POWER MANAGEMENT

The High–Speed Microcontroller has several features that relate to power consumption and management. They provide a combination of controlled operation in unreliable power applications and reduced power consumption in portable or battery powered applications. The range of features is shown below with details to follow.

POWER MANAGEMENT
EARLY WARNING POWER FAIL INTERRUPT
POWER FAIL/POWER ON RESET
BAND–GAP SELECT
WATCHDOG WAKE UP FROM IDLE

POWER SAVING
IDLE MODE
STOP MODE
RING WAKE UP FROM STOP
POWER MANAGEMENT MODES

PRECISION VOLTAGE MONITOR

The High–Speed Microcontroller uses a precision band–gap reference and other analog circuits to monitor the state of the power supply during power–up and power–down transitions. Other microcontroller systems would require external circuits to perform these functions. The band–gap reference provides a precise voltage to compare with V_{CC}. When V_{CC} begins to drop, the Power Monitor compares it to its reference. This enables the analog circuits to detect when V_{CC} passes through predetermined thresholds, V_{PFW} and V_{RST}. These are specified in the individual product data sheets.

EARLY WARNING POWER FAIL INTERRUPT

Devices which incorporate the precision voltage reference have the ability to generate a power–fail interrupt and/or reset in response to a low supply voltage. When V_{CC} reaches the V_{PFW} threshold, the microcontroller can generate an power–fail Interrupt. This early warning of supply voltage failure allows the system time to save critical parameters in nonvolatile memory and put external functions in a safe state.

The power–fail interrupt is optional and is enabled using the Enable Power Fail Warning Interrupt (EPFI) bit at WDCON.5. If enabled, V_{CC} dropping below V_{PFW} will cause the device to vector to address 33h. The Power–fail Interrupt status bit ,PFI (WDCON.4), will be set any time V_{CC} transitions below V_{PFW}. This flag is not cleared when V_{CC} is above V_{PFW}, and software should clear it immediately after reading it. As long as the condition exists, PFI will be immediately set again by hardware.

A typical application of the PFI is to place the device into a "safe mode" when a power loss appears imminent. When the interrupt occurs, the code vectors to location 33h. At this time, software can disable the interrupt, save any critical data, clear PFI, then continually poll the status of the power supply via the PFI flag. As long as PFI is set, power is still below V_{PFW}. If power returns to the proper level, PFI will not be set once cleared by software. This indicates a safe operating condition. If power continues to fall, a power–fail reset will be invoked automatically.

POWER–FAIL RESET

Devices which incorporate the power–fail reset will automatically invoke a reset when V_{CC} drops below V_{RST}. This will halt device operation, and place all outputs in their reset state. This state will continue to be held until V_{CC} drops below the voltage necessary to power the port pins. Because V_{RST} is lower than V_{PFW}, the microcontroller has the option to use the power–fail interrupt to place the device into a "safe" state before the device halts operation with a power–fail reset. This feature is automatic on devices which incorporate the power–fail reset feature, and cannot be disabled.

POWER ON RESET

When V_{CC} is applied to a system using the High–Speed Microcontroller, the device will hold itself in reset until power is within tolerance and stable. An internal band–gap reference provides a highly accurate and stable means of detecting power supply levels. It requires no external circuits to accomplish this. As power rises, the processor will stay in a reset state until $V_{CC} > V_{RST}$. As V_{CC} rises above V_{RST}, internal analog circuits will detect this and activate the on–chip crystal oscillator. On–chip hardware will then count 65536 oscillator clocks. During this count, V_{CC} must remain above V_{RST} or the process restarts. If an off–chip clock source is used, then clock counting still begins once $V_{CC} > V_{RST}$. This count period is used to make certain that power is within tolerance, and that the oscillator has time to stabilize. This provides a very controlled and predictable start–up condition.

Once the 65536 count period has elapsed, the reset condition is removed automatically, and software execution will begin at the reset vector location of 0000h. Software will be able to detect the power on reset condition using the Power–On Reset (POR) flag. POR is located at WDCON.6. This bit will be high to indicate that a Power–on Reset has occurred. It should then be cleared by software.

The complete power cycle operation is shown in Figure 7–1. Note that the interrupt threshold is fixed, but the interrupt itself is optional. Reset thresholds are also fixed and the reset operation is transparent. It requires no external components and no action by software to control reset operation.

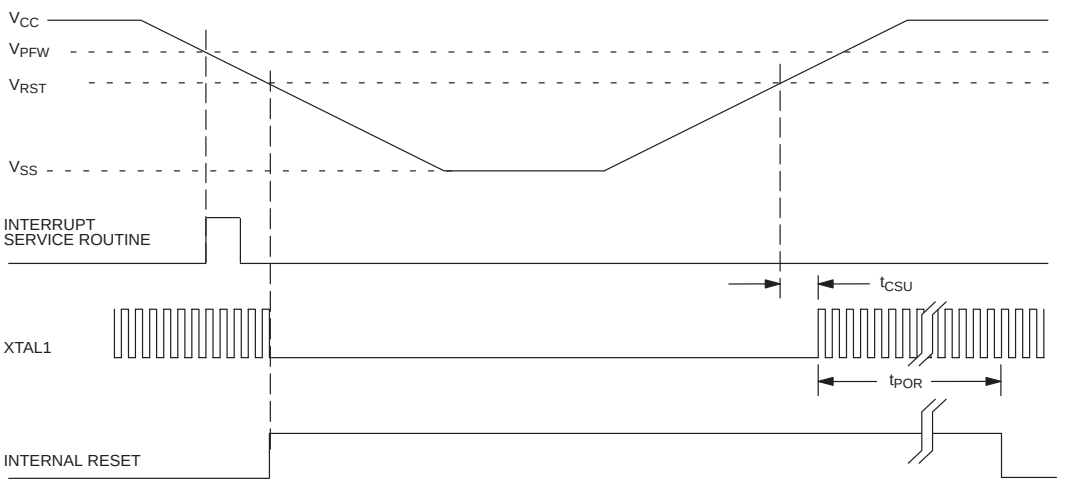
BAND–GAP SELECT

When present, the band–gap reference will provide a precise voltage reference for the power–fail monitor circuitry. The band–gap is normally disabled automatically upon entering Stop mode to provide the lowest power state. Since the band–gap is inactive, there can be no power–fail interrupt and no power–fail reset, similar to a traditional 8051.

If the use of the power–fail features are desired in Stop mode, the BGS bit (EXIF, 91h) may be used. When set to a logic 1 by software, the band–gap reference and associated power monitor circuits will remain active in Stop mode. The price of this feature is higher power supply current requirements. In Stop mode with the band–gap reference disabled (default), the processor draws approximately 1 µA. With the band–gap enabled, it draws approximately 50 µA.

BGS allows the user to decide whether the control circuitry and its associated power consumption are needed. If the application is such that power will not fail while in Stop or if it does not matter that power fails, the BGS should be set to 0 (default). If power can fail at any time and cause problems, the BGS should be set to 1.

POWER CYCLE OPERATION Figure 7–1



WATCHDOG WAKE UP

The Watchdog Wake up is more of an application than a feature. It allows a system to enter the Idle mode for power savings, then to wake up periodically to sample the external world. Idle mode is a low power state described below. Any of the programmable timers can perform this function, but the Watchdog allows a much longer period to be selected. At 12 MHz, the maximum Watchdog time–out is over 5.5 seconds. This contrasts with 0.78 seconds using the 16–bit timers. Software that uses the Watchdog as a wake up alarm should only enable the Watchdog Interrupt and not the Reset. Note that the Watchdog cannot be used to wake the system while in Stop mode since no clocks are running. Stop mode is described below.

POWER MANAGEMENT SUMMARY

The following is a summary of the power management bits and those that are useful or related. They are contained in the register locations WDCON;D8h, EIE;E8h, EXIF;91h, and PCON; 87h.

WDCON.6 POR – Power on Reset. Hardware will set this bit on a power up condition. Software can read it, but must clear it manually. This bit assists software in determining the cause of a reset.

WDCON.5 EPFI – Enable Power–fail Interrupt. Setting this bit to 1 enables the Power–fail interrupt. This will occur when V_{CC} drops to approximately 4.5 volts, and the processor vectors to location 33h. Setting this bit to a 0 turns off the Power–fail Interrupt.

WDCON.4 PFI – Power Fail Interrupt Flag. Hardware will set this bit to a 1 when a Power–fail condition occurs. Software must clear the bit manually. Writing a 1 to this bit will force an interrupt, if enabled.

WDCON.3 WDIF – Watchdog Interrupt Flag. If the Watchdog Interrupt is enabled (EIE.4), hardware will set this bit to indicate that the Watchdog Interrupt has occurred. If the interrupt is not enabled, this bit indicates that the time–out has passed. If the Watchdog Reset is enabled (WDCON.1), the user has 512 clocks to strobe the Watchdog prior to a reset. Software or any reset can clear this flag.

WDCON.2 WTRF – Watchdog Timer Reset Flag. Hardware will set this bit when the Watchdog Timer causes a reset. Software can read it, but must clear it manually. A Power–fail Reset will also clear the bit. This bit assists software in determining the cause of a reset. If EWT=0, the Watchdog Timer will have no affect on this bit.

WDCON.1 EWT – Enable Watchdog Timer Reset. Setting this bit will turn on the Watchdog Timer Reset function. The Interrupt will not occur unless the EWDI bit in the EIE register is set. A reset will occur according to the WD1 and WD0 bits in the CKCON register. Setting this bit to a 0 will disable the reset but leave the timer running.

WDCON.0 RWT – Reset Watchdog Timer. This bit serves as the strobe for the Watchdog function. During the time–out period, software must set the RWT bit if the Watchdog is enabled. Failing to set the RWT will cause a reset when the time–out has elapsed. There is no need to set the RWT bit to a 0 because it is self–clearing.

EIE.4 EWDI – Enable Watchdog Interrupt. Setting this bit in software enables the Watchdog Interrupt.

EXIF.0 BGS – Band–Gap Select. Setting this bit to a 1 will allow the use of the Band–gap voltage reference while in Stop mode. Since this function uses as much as 50 µA, the band–gap is optional in Stop mode. Setting this bit to a 0 will turn off the Band–gap while in Stop mode. When BGS=0, no Power–fail interrupt or Power–fail Reset will be available in Stop mode.

PCON.1 STOP. When this bit is set, the program stops execution, clocks are stopped, and the CPU enters power–down mode.

PCON.0 IDLE. Program execution halts leaving timers, serial ports, and clocks running.

EXIF.2 RGMD – Ring Oscillator Mode. Hardware will set this status bit to a 1 when the clock source is the Ring Oscillator. Hardware will set this status bit to a 0 when the crystal is the clock source. Refer to RGSL below for operation of the Ring Oscillator.

EXIF.1 RGSL – Ring Oscillator Select. When set to a 1 by software, the High–Speed Microcontroller will use a Ring Oscillator to come out of Stop mode without waiting for crystal start–up. This allows an instantaneous start–up when coming out of Stop mode. It is useful if software needs to perform a short task, then return to Stop. It is also useful if software must respond quickly to an external event. After the crystal has performed 65,536 cycles, hardware will switch to the crystal as its clock source. The RGMD status bit reports on this changeover. When RGSL is set to a 0, the High–Speed Microcontroller will delay software execution until after the 65,536 clock crystal startup time. RGSL is only cleared by a power–on reset and is not altered by other forms of reset.

POWER SAVING

The High–Speed Microcontroller is implemented using full CMOS circuitry for low power operation. It is fully static so the clock speed can be run down to DC. Like other CMOS, the power consumption is also a function of operating frequency. Although the High–Speed Microcontroller is designed for maximum performance, it also provides improved power versus work relationships compared with standard 8051 devices. These topics are discussed in detail below.

The High–Speed Microcontroller provides two modes (other than operating) that allow power conservation. They are similar, but have different merits and drawbacks. These modes are Idle and Stop. In the original 8051, the Stop mode is called Power Down. These modes are invoked in the same manner as the original 8051 series.

IDLE MODE

Idle mode suspends all CPU processing by holding the program counter in a static state. No program values are fetched and no processing occurs. This saves considerable power versus full operation. The virtue of Idle mode is that it uses half the power of the operating state, yet reacts instantly to any interrupt conditions. All clocks remain active so the timers, Watchdog, Serial Port, and Power Monitor functions are all working. Since all clocks are running, the CPU can exit the Idle state using any of the interrupt sources.

Software can invoke the Idle mode by setting the IDLE bit in the PCON register at location 87h. The bit is located at PCON.0. The instruction that executes this step will be the last instruction prior to freezing the program counter. Once in Idle, all resources are preserved. There are two ways to exit the Idle mode. First, any interrupt (that is enabled) will cause an exit. This will result in a jump to the appropriate interrupt vector. The IDLE bit in the PCON register will be cleared automatically. On returning from this vector using the RETI instruction, the next address will be the one immediately after the instruction that invoked the Idle state.

The Idle mode can also be removed using a reset. Any of the three reset sources can do this. On receiving the reset stimulus, the CPU will be placed in a reset state and the Idle condition cleared. When the reset stimulus is removed, software will begin execution as for any reset. Since all clocks are active, there will be no delay after the reset stimulus is removed. Note that if enabled, the Watchdog Timer continues to run during Idle and must be supported.

STOP MODE

Stop mode is the lowest power state that the High–Speed Microcontroller can enter. This is achieved by stopping all on–chip clocks, resulting in a fully static condition. No processing is possible, timers are stopped, and no serial communication is possible. Processor operation will halt on the instruction that sets the STOP bit. The internal amplifier that excites the external crystal will be disabled, halting crystal oscillation in Stop mode. Table 7–1 shows the state of the processor pins in Idle and Stop modes.

Stop mode can be exited in two ways. First, like the 8052 microcontrollers, a non–clocked interrupt such as the external interrupts or the power–fail interrupt can be used. Clocked interrupts such as the watchdog timer, internal timers, and serial ports will not operate in Stop mode. Note that the band–gap reference must be enabled in order to use the power–fail interrupt to exit Stop mode, which will increase Stop mode current. Processor operation will resume with the fetching of the interrupt vector associated with the interrupt that caused the exit from Stop mode. When the interrupt service routine is complete, an RETI will return the program to the instruction immediately following the one that invoked the Stop mode.

A second method of exiting Stop mode is with a reset. The watchdog timer reset is not available as a reset source because no timers are running in Stop mode. An external reset via the RST pin will unconditionally exit the device from Stop mode. If the BGS bit is set, the device will provide a reset while in Stop mode if V_{CC} should drop below the V_{RST} level. If the BGS bit is 0, then a dip in power below V_{RST} will not cause a reset. For example, if V_{CC} should drop to a level of V_{RST}–0.5V, then return to the full level, no reset will be generated. For this reason, use of the band–gap reference is recommended if a brownout condition is possible in Stop mode. If power fails completely (V_{CC}=0V), then a power on reset will still be performed when V_{CC} is reapplied regardless of the state of the BGS bit. Processor operation will resume execution from address 0000h like any other reset.

CRYSTAL RESUME FROM STOP MODE

If the microcontroller does not contain a ring oscillator, or if the RGSL bit is 0, a device exiting Stop mode must restart operation using the external crystal as a clock source. The device will experience a power–on reset delay of 65536 external clock cycles to allow the crystal to begin oscillation and the frequency to stabilize. Once this delay is complete, software will begin execution from either address 0000h or the appropriate interrupt vector, depending on the stimulus to exit Stop mode. The same 65536 external clock cycle delay is performed if an external crystal oscillator is used instead of an external crystal.

PIN STATES IN POWER SAVING MODES Table 7–1

DEVICE	MODE	ALE	PSEN	P0 (AD0–7)	P1	P2	P3
DS80C310 DS80C320	Idle or Stop	1	1	Latched[1]	Port data[2]	Latched[3]	Port data[2]
All Others Internal program execution	Idle or Stop	1	1	Port data[2]	Port data[2]	Port data[2]	Port data[2]
All Others External program execution	Idle	1	1	Latched[1]	Port data[2]	Latched[3]	Port data[2]
All Others External program execution	Stop	1	1	Port data[2]	Port data[2]	Port data[4]	Port data[2]

[1] Port exhibits opcode following instruction that sets the Stop bit. Port 0 is operating in true bidirectional mode, and will drive both a logic 1 and a logic 0.
[2] Port reflects data stored in corresponding Port SFR. Port 0 functions as an open–drain output in this mode.
[3] Port exhibits address MSB of opcode following instruction that sets the Stop bit.
[4] Port reflects data stored in corresponding Port SFR. In this mode, the port uses weak pullups. If a bit in the P2 SFR is a 1, the corresponding device pin will transition slowly to a high when the reset state is entered.

RING OSCILLATOR WAKE UP FROM STOP

A typical low power application is to keep the processor in Stop mode most of the time. Periodically, the system will wake up (using an external interrupt), take a reading of some condition, then return to sleep. The duration of full power operation is as short as possible. One disadvantage to this method is that the clock must be restarted prior to performing a meaningful operation. This start–up period is a waste of time and power since no work can be performed. The High–Speed Microcontroller provides an alternative.

If the Ring Select (RGSL) is enabled, the High–Speed Microcontroller can exit Stop mode running from an internal Ring Oscillator. Upon receipt of an interrupt , this oscillator can start instantaneously, allowing software execution to begin immediately while the oscillator is stabilizing. Once 65,536 clock cycles have been detected, the CPU will automatically switch to the normal oscillator as its clock source. However, if the required interrupt response is very short, the software can re–enter Stop mode before the crystal is even stable. In this case, Stop mode can be invoked and both oscillators will be stopped.

SPEED REDUCTION

The High–Speed Microcontroller is a fully CMOS 8051 compatible microcontroller. It can use significantly less power than other 8051 versions because it is more efficient. As an average, software will run 2.5 times faster on the High–Speed Microcontroller than on other 8051 derivatives. Thus the same job can be accomplished by slowing down the crystal by a factor of 2.5. For example, an existing 8051 design that runs at 12 MHz can run at approximately 4.8 MHz on the High–Speed Microcontroller. At this reduced speed, the High–Speed Microcontroller will have lower power consumption than an 8051, yet perform the same job.

Using the 2.5X factor, Table 7–2 shows the approximate speed at which the High–Speed Microcontroller can accomplish the same work as an 8051. The exact improvement will vary depending on the actual instruction mix. Available crystal speeds must also be considered. Refer to Section 16 for information on instruction timing.

CRYSTAL VS MIPS COMPARISON Table 7–2

ORIGINAL 8051 CRYSTAL SPEED	MIPS	HIGH–SPEED MICROCONTROLLER CRYSTAL SPEED
3.57 MHz	0.3	1.4 MHz
7.37 MHz	0.6	2.9 MHz
11.0592 MHz	0.9	4.4 MHz
14.318 MHz	1.2	5.7 MHz
16 MHz	1.3	6.4 MHz
20 MHz	1.6	8 MHz
24 MHz	2.0	9.6 MHz
33 MHz	2.7	13.2 MHz
40 MHz	3.3	16 MHz

POWER MANAGEMENT MODES

Power consumption in CMOS microcontrollers is a function of operating frequency. The Power Management Mode (PMM) feature, available with some members of the High–Speed Microcontroller family, allows software to dynamically match operating frequency and current consumption with the need for processing power. Instead of the default 4 clocks per machine cycle, power management mode 1 (PMM1) and power management mode 2 (PMM2) utilize 64 and 1024 clocks per cycle respectively to conserve power.

A number of special features have been added to enhance the function of the power management modes. The switchback feature allows the device to almost instantaneously return to divide by 4 mode upon acknowledgment of an external interrupt or a falling edge on a serial port receiver pin. The advantages of this become apparent when one calculates the increased interrupt service time of a device operating in PMM. In addition, a device operating in PMM would normally be unable to sample an incoming serial transmission to properly receive it. The switchback feature, explained below, allows a device to return to divide by 4 operation in time to receive incoming serial port data and process interrupts with no loss in performance.

The DS87C520 and DS87C530 incorporate a Status register (STATUS;C5h) to prevent the device from accidentally reducing the clock rate during the servicing of an external interrupt or serial port activity. This register can be interrogated to determine if a high priority, low priority, or power fail interrupt is in progress, or if serial port activity is occurring. Based on this information the software can delay or reject a planned change in the clock divider rate.

In addition, the DS87C520 and DS87C530 has the capability to operate from the internal ring oscillator during normal operation, not only during the crystal warm–up period. Table 7–3 summarizes the new control bits associated with the power management features.

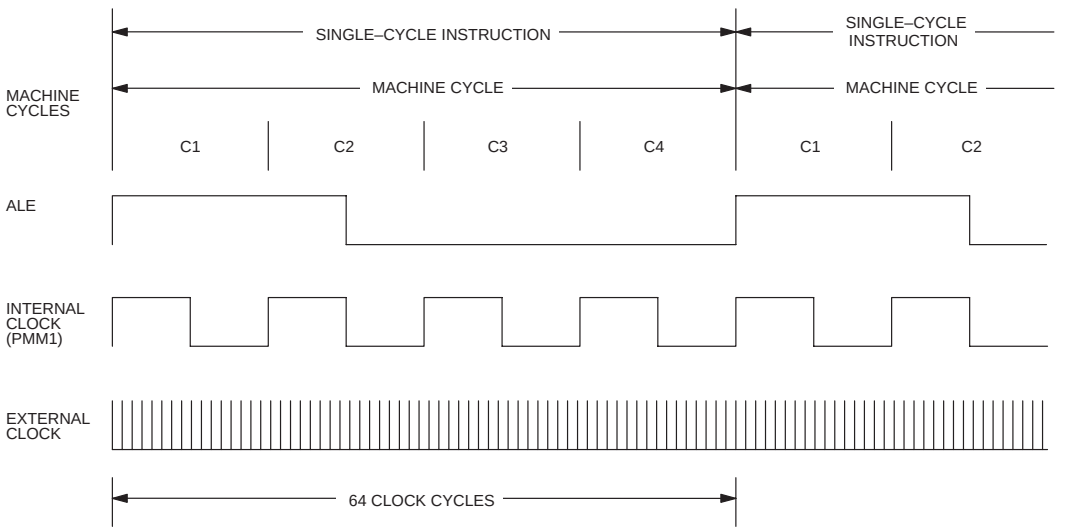
POWER MANAGEMENT AND STATUS BIT SUMMARY Table 7–3

BIT NAME	LOCATION	FUNCTION	RESET STATE	READ/WRITE ACCESS
CD1, CD0	PMR.7–6	Clock Divider Control CD1 CD0 Osc Cycles per Machine Cycle 0 0 Reserved 0 1 4 (Reset Default) 1 0 64 (PMM1) 1 1 1024 (PMM2)	0,1	Write: 0,1 anytime; 1,0 & 1,1 only when previously in 0,1 state. Unrestricted read.
SWB	PMR.5	Switchback Enable 0=Interrupts and serial port activity will not affect clock divider control bits 1=Enabled Interrupts and serial port activity will cause a switchback	0	Unrestricted
PIP	STATUS.7	Power Fail Interrupt Status 0=No power fail interrupt in progress 1=Power fail interrupt in progress	0	Read Only
HIP	STATUS.6	High Priority Interrupt Status 0=No high priority interrupt in progress 1=High priority interrupt in progress	0	Read Only
LIP	STATUS.5	Low Priority Interrupt Status 0=No low priority interrupt in progress 1=Low priority interrupt in progress	0	Read Only
SPTA1	STATUS.3	Serial Port 1 Transmitter Activity Status 0=Serial port 1 transmitter inactive 1=Serial port 1 transmitter active	0	Read Only
SPRA1	STATUS.2	Serial Port 1 Receiver Activity Status 0=Serial port 1 receiver inactive 1=Serial port 1 receiver active	0	Read Only
SPTA0	STATUS.1	Serial Port 0 Transmitter Activity Status 0=Serial port 0 transmitter inactive 1=Serial port 0 transmitter active	0	Read Only
SPRA0	STATUS.0	Serial Port 0 Receiver Activity Status 0=Serial port 0 receiver inactive 1=Serial port 0 receiver active	0	Read Only

POWER MANAGEMENT MODE TIMING

The two power management modes reduce power consumption by internally dividing the clock signal to the device, causing it to operate at a reduced speed. When PMM is invoked, the external crystal will continue to operate at full speed. The difference is that the device uses 16 (PMM1) or 256 (PMM2) external clocks to generate each internal clock cycle (C1, C2, C3 or C4) as opposed to 1 clock per internal clock cycle in divide by 4 mode. This translates to 64 or 1024 external clocks per machine cycle in PMM1 or PMM2, respectively. Relative timing relationships of all signals when the device is operating in PMM1 or PMM2 will remain the same as the 4 cycle timing. Note that all internal functions, on–board timers (including serial port baud rate generation), watchdog timer, and software timing loops will also run at the reduced speed. Most applications will not find it necessary to attend to this much detail, but the information is provided for calculating critical timings. Figure 7–2 demonstrates the internal timing relationships during PMM1.

INTERNAL TIMING RELATIONSHIPS IN PMM1 Figure 7–2



PMM1 and PMM2 are entered and exited by setting the Clock Rate Divider bits (PMR.7–6). In addition, it is possible use the switchback feature to effect a return to the divide by 4 mode from either power management mode. This allows both hardware and software to cause an exit from PMM. Entry to or exit from either PMM must be by the divide by 4 mode. This means that to switch from divide by 64 to divide by 1024 and vice versa, one must first switch back to divide by 4 mode. Attempts to execute an illegal speed change will be ignored and the bits will remain unchanged. It is the responsibility of the software to test for serial port activity before attempting to change speed, as a modification of the clock divider bits during a serial port operation will corrupt the data.

PMM AND PERIPHERAL FUNCTIONS

Timers 0, 1, and 2 will default on reset to a 12 clock per cycle operation to remain compatible with the original 8051 timing. The timers can be individually configured to run at machine cycle timing (divide by 4) by setting the relevant bits in the Clock Control Register (CKCON;8Eh). Because the timers derive their time base from the internal clock, timers 0, 1, and 2 operate at reduced clock rates during PMM. This will also affect the operation of the serial ports in PMM. In general, it is not possible to generate standard baud rates while in PMM, and the user is advised to avoid PMM or use the switchback feature if serial port operation is desired. Table 7–4 shows the effect of the clock divider value on timer operation.

EFFECT OF CLOCK MODES ON TIMER OPERATION Table 7–4

CD1	CD0	OSC. CYCLES PER MACHINE CYCLE	OSC. CYCLES PER TIMER 0/1/2 CLOCK		OSC. CYCLES PER TIMER 2 CLOCK, BAUD RATE GEN.		OSC. CYCLES PER SERIAL PORT CLOCK MODE 0		OSC. CYCLES PER SERIAL PORT CLOCK MODE 2	
			TxM=1	TxM=0	T2M=1	T2M=0	SM2=0	SM2=1	SMOD=0	SMOD=1
0	0	Reserved								
0	1	4	12	4	2	2	12	4	64	32
1	0	64 (PMM1)	192	64	32	32	192	64	1024	512
1	1	1024 (PMM2)	3072	1024	512	512	3072	1024	16,384	8192

SWITCHBACK

The switchback feature solves one of the most vexing dilemmas faced by power–conscious systems. Many applications are unable to use the Stop and Idle modes because they require constant computation. Traditionally, system designers could not reduce the operating speed below that required to process the fastest event. This meant that system architects would be forced to operate their systems at the highest rate of speed even when it was not required.

The switchback feature allows a system to operate at a relatively slow speed, and burst to a faster mode when required by an external event. When this feature is enabled by setting the Switchback Enable bit, SWB, (PMR.5), a qualified interrupt or serial port reception or transmission will cause the device to return to divide by 4 mode. A qualified interrupt is defined as an interrupt which has occurred and been acknowledged. This means that an interrupt must be enabled and also not blocked by a higher priority interrupt. After the event is complete, software can manually return the device to the appropriate PMM. The following sources can trigger a switchback:

external interrupt 0/1/2/3/4/5,
serial start bit detected, Serial Port 0/1,
transmit buffer loaded, Serial Port 0/1,
watchdog timer reset,
power–on reset,
external reset.

In the case of a serial port–initiated switchback, the switchback is not generated by the associated interrupt. This is because a device operating in PMM will not be able to correctly receive a byte of data to generate an interrupt. Instead, a switchback is generated by a serial port reception on the falling edge associated with the start bit, if the associated receiver enable bit (SCON0.4 or SCON1.4) is set. For serial port transmissions, a switchback is generated when the serial port buffer (SBUF0;99h or SBUF1;C1h) is loaded. This ensures the device will be operating in divide by 4 mode when the data is transmitted, and eliminates the need for a write to the CD1, CD0 bits to exit PMM before transmitting. The switchback feature is unaffected by the state of the serial port interrupt flags (RI_0, TI_0, RI_1, TI_1).

The timing of the switchback is dependent on the source. Interrupt–initiated switchbacks will occur at the start of the first C1 cycle following the event initiating the switchback. In PMM, each internal Cx cycle is 16 external clock cycles for PMM1 and 256 cycles for PMM2. If the current instruction in progress is a write to the IE, IP, EIE, or EIP registers, interrupt processing will be delayed until the completion of the following instruction. Serial transmit–initiated switchbacks occur at the start of the instruction following the MOV that loads SBUF0 or SBUF1. Serial reception–initiated switchbacks occur during the Cx cycle in which the falling edge was detected.

There are a few points that must be considered when using a serial port reception to generate a switchback. Under normal circumstances, noise on the line or an aborted transmission would cause the serial port to time–out and the data to be ignored. This presents a problem if the switchback is used, however, because a switchback would occur but there is no indication to the system that one has occurred. If PMM and serial port switchback functions are used in a noisy environment, the user is advised to periodically check if the device has accidentally exited PMM.

A similar problem can occur if multiprocessor communication protocols are used in conjunction with PMM. The High–Speed Microcontroller family supports both the use of the SM2 flag (SCON0.5 or SCON1.5), and the slave address recognition registers (SADDR0;A9h, SADDR1;AAh, SADEN0;B9h, SADEN1;BAh) for multiprocessor communications. The problem is that an invalid address which should be ignored by a particular processor will still generate a switchback. As a result it is not recommended to use a multiprocessor communication scheme in conjunction with PMM. If the system power considerations will allow for an occasional erroneous switchback, a polling scheme can be used to place the device back into PMM.

CLOCK SOURCE SELECTION

The High–Speed Microcontroller family supports three different clock sources for operation. As with most microcontrollers, the device can be clocked from an external crystal using the on–board crystal amplifier, or a clock source can supplied by an external oscillator. In addition, some members of the High–Speed Microcontroller family incorporate an on–board ring oscillator to provide a quick resumption from Stop mode. The ring oscillator is a low power digital oscillator internal to the microcontroller. When enabled, it provides an approximately 2–4 MHz clock source for device operation without external components. The ring oscillator is not as stable as an external crystal, and software should refrain from performing timing dependent operations, including serial port activity, while operating from the ring oscillator.

The ring oscillator provides many advantages to the designers of microcontroller–based systems. One is that it allows Dallas Semiconductor microcontrollers to perform a fast resume from Stop mode, eliminating the crystal warm–up delay when restarting the device. As an added feature, the DS87C520 and DS87C530 will also support extended operation from the ring oscillator, not only during the crystal warm–up period when resuming from Stop. All devices in the High–Speed Microcontroller family must begin operation following a power–on reset from an external clock source, either an external crystal or oscillator. Software can then disable the crystal and run from the lower power ring oscillator. The control and status bits which support the new and/or enhanced features are shown in Table 7–5.

CLOCK CONTROL AND STATUS BIT SUMMARY Table 7–5

BIT NAME	LOCATION	FUNCTION	RESET	WRITE ACCESS
XT/RG	EXIF.3	Crystal/Ring Clock Source Select. This bit is not present on the 80C320. 1=Select crystal or external clock as clock source, 0=Select ring oscillator as clock source	1	0 anytime; 1 when XTUP=1 & XTOFF=0
RGMD	EXIF.2	Ring Oscillator Mode Status. 1=Ring oscillator is current clock source, 0=Crystal or external clock is current clock source.	0	None
RGSL	EXIF.1	Ring Oscillator Select, Stop Mode. 1=Ring oscillator will be the clock source when resuming from Stop mode, 0=Crystal or external clock will be the clock source when resuming from Stop mode Note: Upon completion of crystal warm up period, DS80C320 devices will switch to crystal. DS87C520 and DS87C530 devices will switch to clock source designated by XT/$\overline{\text{RG}}$ bit		Unrestricted
XTOFF	PMR.3	Crystal Oscillator Disable. Disables crystal operation after ring mode has been selected. This bit is not present on the 80C320. 1=Crystal amplifier is disabled. 0=Crystal amplifier is enabled. Check XTUP for status.	0	0 anytime; 1 when XT/$\overline{\text{RG}}$ =0
XTUP	STATUS.4	Crystal Oscillator Warm Up Status. This bit is not present on the 80C320. 1=Oscillator warm up complete. 0=Oscillator warm up still in progress, crystal not available.	1	None

RING OSCILLATOR RESUME FROM STOP

To achieve the minimum power consumption during periods of processor inactivity, software can place the device into Stop mode. Such systems will typically resume operation using an external interrupt, perform some activity, and then return to Stop mode. Traditional designs which rely upon an external crystal as the clock source must incur the startup delay of the crystal when resuming from Stop mode. This is a waste of time and power as no work can be performed until the crystal has stabilized.

Although the ring oscillator provides an approximately 2–4 MHz clock source for device operation, it is not as stable as an external crystal. As a result, high accuracy timing operations should be avoided while running from the ring oscillator. This includes using the timers for pulse measurement, and the use of the serial ports in asynchronous modes (1, 2, 3). Serial ports operating in mode 0 are unaffected by the stability of the clock source as a separate synchronizing clock is generated.

If the Ring Oscillator Select bit, RGSL (EXIF.1) is set, the device will resume operation immediately using the internal ring oscillator as the clock source. The device will continue to run from the ring oscillator until the crystal warm–up period of 65,536 clock cycles (measured from the external source) has completed. At this time the device will switch to the clock source active before it entered Stop mode and continue operation. This allows software execution to begin immediately upon resuming from Stop mode. The current clock source is indicated by the Ring Oscillator Mode bit, RGMD (EXIF.2).

In Stop mode, enabled interrupts become true edge triggered interrupts, compared with the sampled edge detection used during normal operation. This means that external interrupts are more sensitive to noise in Stop mode than during normal operation. Applications should be carefully designed to ensure that noise will not cause an erroneous exit from Stop mode.

SWITCHING BETWEEN CLOCK SOURCES

DS87C520 and DS87C530 incorporate the ability to run the device from the ring oscillator after the crystal warm–up period has elapsed. Immediately following a reset (including initial power–up), all devices must operate from an external crystal or oscillator. At this point, software may switch to the ring oscillator by clearing the XT/$\overline{\text{RG}}$ bit (EXIF.3). If there is no expectation that the crystal oscillator will be needed soon, the crystal oscillator can be disabled by setting the Crystal Oscillator Disable Bit, XTOFF (PMR.3). Note that switching to the ring oscillator does not automatically disable the crystal amplifier, and thus it is possible to be operating the device from the ring oscillator and have the external crystal amplifier operating at the same time. In some cases this may be desired to take advantage of the low–frequency, low–power feature of the ring oscillator but still have the capability of quickly switching back to the external crystal to perform timing or serial port operations.

Switching from the ring oscillator to the crystal oscillator is more involved due to the startup delays inherent in the external crystal. To prevent an accidental disabling of the device, the XTUP bit must be set by internal hardware (indicating an enabled, stable crystal) before setting the XT/$\overline{\text{RG}}$ bit. The procedure to switch to the crystal oscillator when running from the ring oscillator is as follows:

1. Clear the Crystal Oscillator Disable Bit, XTOFF (PMR.3) to restart the crystal oscillator and start the crystal warm–up period.
2. Wait for the Crystal Oscillator Warm Up Status bit, XTUP (STATUS.4) to be set, indicating that the external crystal warm up period is complete. This will take 65,536 external clock cycles.
3. Set the Crystal Oscillator/Ring Oscillator Select Bit, XT/$\overline{\text{RG}}$ (EXIF.3) to select the crystal as the clock source.

SECTION 8: RESET CONDITIONS

The High–Speed Microcontroller provides several ways to place the CPU in a reset state. It also offers the means for software to determine the cause of a reset. The reset state of most processor bits is not dependent on the type of reset, but selected bits do depend on the reset source. The reset sources and the reset state are described below.

RESET SOURCES

The High–Speed Microcontroller has three ways of entering a reset state. Each reset source is described below. They are:

Power–on/Power Fail Reset
Watchdog Timer Reset
External Reset

Power–on/Fail Reset

Members of the High–Speed Microcontroller family incorporate an internal voltage reference which holds the CPU in the power–on reset state while V_{CC} is out of tolerance. Once V_{CC} has risen above the threshold, the microcontroller will restart the oscillation of the external crystal and count 65536 clock cycles. The processor will then begin software execution at location 0000h.

The voltage at which the reset state is entered depends on the specific device. If the device does not contain a precision voltage reference, the power–on reset threshold may be anywhere between 0.8V and V_{CCMIN}. If the device incorporates a precision voltage reference, the threshold will be as specified by the V_{RST} parameter in the data sheet. This helps the system maintain reliable operation by only permitting processor operation when voltage is in a know good state.

The processor will exit the reset condition automatically once the above conditions are met. This happens automatically, needing no external components or action. Execution begins at the standard reset vector address of 0000h. Software can determine that a Power–on Reset has occurred using the Power–on Reset flag (POR). It is located at WDCON.6. Since all resets cause a vector to location 0000h, the POR flag allows software to acknowledge that power failure was the reason for a reset.

Software should clear the POR bit after reading it. When a reset occurs, software will be able to determine if a power cycle was the cause. In this way, processing may take a different course for each of the three resets if applicable. When power fails (drops below V_{RST}), the power monitor will invoke the reset state again. This reset condition will remain while power is below the threshold. When power returns above the reset threshold, a full power–on reset will be performed. Thus a brownout that causes V_{CC} to drop below V_{RST} appears the same as a power up.

Watchdog Timer Reset

The Watchdog Timer is a free running timer with a programmable interval. Software can clear the timer at any time, causing the interval to begin again. The Watchdog supervises CPU operation by requiring software to clear it before the time–out expires. If the timer is enabled and software fails to clear it before this interval expires, the CPU is placed into a reset state. The reset state will be maintained for two machine cycles. Once the reset is removed, the software will resume execution at 0000h.

The Watchdog Timer is fully described in Section 11. Software can determine that a Watchdog time–out was the reason for the reset by using the Watchdog Timer Reset flag (WTRF). WTRF is located at WDCON.2. Hardware will set this bit to a logic 1 when the Watchdog times out without being cleared by software if EWT=1. If a Watchdog Timer reset occurs, software should clear this flag manually. This allows software to detect the event if it occurs again.

External Reset

If the RST input is taken to a logic 1, the CPU will be forced into a reset state. This will not occur instantaneously, as the condition must be detected and then clocked into the microcontroller. It requires a minimum of two machine cycles to detect and invoke the reset state. Thus the reset is a synchronous operation and the crystal must be running to cause an external reset.

Once the reset state is invoked, it will be maintained as long as RST=1. When the RST is removed, the CPU will exit the reset state within two machine cycles and begin execution at address 0000h. All registers will default to their power–on reset state. There is no flag to indicate that an external reset was applied. However, since the other two sources have associated flags, the RST pin is the default source when neither POR or WTRF is set.

If a RST is applied while the processor is in the Stop mode, the scenario changes slightly. As mentioned above, the reset is synchronous and requires a clock to

be running. Since the Stop mode stops all clocks, the RST will first cause the oscillator to begin running and force the program counter to 0000h. Rather than a two machine cycle delay as described above, the processor will apply the full power–on delay (65536 clocks) to allow the oscillator to stabilize.

RESET STATE

Regardless of the source of the reset, the state of the microcontroller is the same while in reset. When in reset, the oscillator is running, but no program execution is allowed. When the reset source is external, the user must remove the reset stimulus. When power is applied to the device, the power–on delay removes the stimulus automatically.

Resets do not affect the Scratchpad RAM. Thus any data stored in RAM will be preserved. The contents of internal MOVX data memory will also remain unaffected by a reset. Note that if the power supply dips below approximately 2V, the RAM contents may be lost. The minimum voltage required for RAM data retention in not specified. Since it is impossible to determine if the power was lower than 2V prior to the power–on reset, RAM must be assumed lost when POR is set.

The reset state of SFR bits are described in Section 4. Bits which are marked SPECIAL have conditions which can affect their reset state. Consult the individual bit descriptions for more information. Note that the stack pointer will also be reset. Thus the stack is effectively lost during a reset even though the RAM contents are not altered. Interrupts and Timers are disabled. The state of the Watchdog Timer is dependent on the specific device in use. Note that the Watchdog time out defaults to its shortest interval on any reset. I/O Ports are taken to a weak high state (FFh). This leaves each port pin configured with the data latch set to a 1. Ports do not go to the 1 state instantly when a reset is applied, but will be taken high within two machine cycles of asserting a reset. When the reset stimulus is removed, program execution begins at address 0000h.

NO–BATTERY RESET

The battery backup feature of the DS87C530 introduces a new type of reset condition. Most SFR bits are automatically reset to their default state upon a power–on reset. The external backup battery feature makes some bits non–volatile, however, and these battery–backed bits will not change state when a power–on reset is applied. Upon the loss or initial connection of battery power these bits will default to the state shown in Table 8–1. Any bits not listed below are either unchanged or set to their default state by a power–on reset.

NO–BATTERY RESET DEFAULT Table 8–1

BIT NAME	LOCATION	NO–BATTERY RESET STATE
E4K	TRIM.7	0
X12/6	TRIM.6	1
TRM2	TRIM.5	1
TRM2	TRIM.4	0
TRM1	TRIM.3	0
TRM1	TRIM.2	1
TRM0	TRIM.1	0
TRM0	TRIM.0	1
RTASS.7–0	RTASS.7–0	Indeterminate
RTAS.7–0	RTAS.7–0	Indeterminate
RTAM.7–0	RTAM.7–0	Indeterminate
RTAH.7–0	RTAH.7–0	Indeterminate
SSCE	RTCC.7	Indeterminate
SCE	RTCC.6	Indeterminate
MCE	RTCC.5	Indeterminate
HCE	RTCC.4	Indeterminate
RTCIF	RTCC.1	Indeterminate
RTCE	RTCC.0	Indeterminate
RTCSS.7–0	RTCSS.7–0	Indeterminate
RTCS.7–0	RTCS.7–0	Indeterminate
RTCM.7–0	RTCM.7–0	Indeterminate
RTCH.7–0	RTCH.7–0	Indeterminate
RTCD0.7–0	RTCD1.7–0	Indeterminate
RTCD1.7–0	RTCD1.7–0	Indeterminate

IN–SYSTEM DISABLE MODE

The High Speed Microcontroller Family supports in–circuit debugging of designs. The In–System Disable (ISD) feature allows the device to be tristated for in–circuit emulation or board testing. During ISD mode, the device pins will take on the following states:

DEVICE PIN	STATE DURING ISD
Port 0, 1, 2, 3 RST, $\overline{EA}$	True Tristate
ALE, $\overline{PSEN}$	Weak Pull–up (~10KΩ)
XTAL1, XTAL2	Oscillator remains active

The following procedure is used to enter ISD mode:

1. Assert reset by pulling RST high,
2. Pull ALE low and pull $\overline{PSEN}$ high,
3. Verify that P2.7, P2.6, P2.5 are not being driven low,
4. Release RST,
5. Hold ALE low and $\overline{PSEN}$ high for at least 2 machine cycles,
6. Device is now in ISD mode. Release ALE and $\overline{PSEN}$ if desired.

Note that pins P2.7, P2.6, P2.5 should not be driven low when RST is released. This will place the device into a reserved test mode. Because these pins have a weak pull–up during reset, they can be left floating. The test mode is only sampled on the falling edge of RST, and once RST is released their state will not effect device operation. In a similar manner, the $\overline{PSEN}$ and RST pins can be released once ISD mode is invoked, and their state will not effect device operation. The RST pin will also be in a tristate mode, but asserting it in ISD mode will return the device to normal operation.

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.

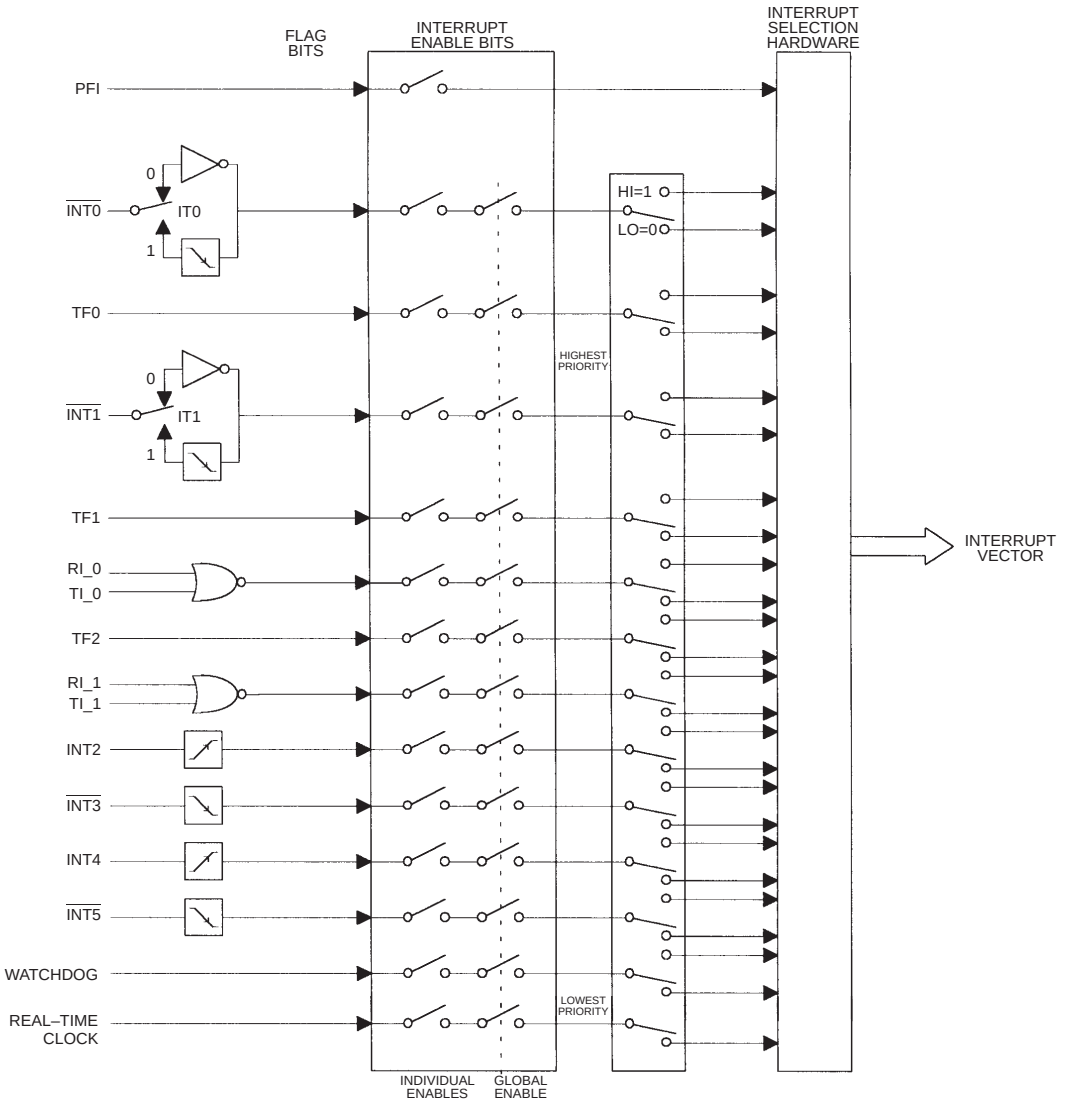
The individual interrupt sources and associated enable and priority bits are shown in Figure 9–1. While the final selection of the appropriate interrupt vector address is referred to as a polling process, this function is actually performed in a single machine cycle using combinatorial logic.

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



SECTION 10: PARALLEL I/O

The High–Speed Microcontroller method of implementing I/O ports follows the standard 8051 convention. This provides backward compatibility with existing designs. All drive capabilities exceed or equal the original 80C32, and voltage levels are compatible. The transitions between strong and weak drives are similar but not identical. Differences are to accommodate higher speed timing and the associated demands on slew rates. As with any new technology, the High–Speed Microcontroller should be evaluated in a system to see how subtle differences affect operation.

The primary High–Speed Microcontroller configuration provides four 8–bit I/O ports. From a software perspective, each port appears as Special Function Register (SFR) with a unique address. Each port register is addressable as a byte or 8 individual bit locations. The CPU distinguishes between a bit access and a byte access by the instruction type. Except for the special cases mentioned below, the register and port pins have identical states. Thus reading or writing a port is the same as reading or writing the SFR for that port.

The microcontroller will distinguish between port and bus operations automatically. If a memory fetch is decoded and requires external memory, Port 0 and 2 will be driven as a bus with the associated timing and drive strengths. If either port SFR is accessed, the port pins will revert to the characteristics described above. This includes a strong pull–down, a strong pull–up for transitions, and a weak pull–up for static conditions.

ROMless versions of the High–Speed Microcontroller dedicate Port 0 and 2 as the memory interface bus. The Port 0 latch does not exist on ROMless devices. The functions of these ports are described in more detail in the specific sections.

PORT 0

General Purpose I/O

Devices which have internal program memory have the ability to use Port 0 as a general purpose I/O. Data written to the port latch serves to set both level and direction of the data on the pin. ROMless devices do not contain a Port 0 latch, because at no time can it be manipulated as a port. When used as an I/O port, it functions as an open–drain output. More detail on the functions of these pins is provided under the description of output and input functions in this section.

Even if internal memory is present, the use of Port 0 as general purpose I/O pins is not recommended if the device will be used to access external memory. This is because the state of the pins will be disturbed during the memory access. In addition, the pull–ups needed to maintain a high state during the use as general purpose I/O will interfere with the complementary drivers employed when the device operates as an expanded memory bus.

Multiplexed Address/Data Bus AD0–7

When used to address expanded memory, Port 0 functions as a multiplexed address/data bus. Port 0 must function as the address/data bus on ROMless devices. Port 0 pins have extremely strong drivers that allow the bus to move 100 pF loads with the timing shown in the electrical specifications. Special circuit protection allows these pins to achieve the maximum slew rate without ringing, eliminating excessive noise or interface problems. Users that compare the High–Speed Microcontroller family to 80C32 devices will find improved drive capability. This power is available for dynamic switching only, and should not be used to drive heavy DC loads such as LEDs.

When used as an address bus, the AD0–7 pins will provide true drive capability for both logic levels. No pull–ups are needed. In fact, pull–ups will degrade the memory interface timing. Members of the High–Speed Microcontroller family employ a two–state drive system on AD0–7. That is, the pin is driven hard for a period to allow the greatest possible setup or access time. Then the pin states are held in a weak latch until forced to the next state or overwritten by an external device. This assures a smooth transition between logic states and also allows a longer hold time. In general, the data is held (hold time) on AD0–7 until another device overwrites the bus. This latch effect is generally transparent to the user.

PORT 2

General Purpose I/O

Devices which have internal program memory have the ability to use Port 2 for a general purpose I/O. Data written to the port latch serves to set both level and direction of the data on the pin. When used as an I/O port, it has complementary outputs that will drive both high and low logic levels. More detail on the functions of these pins is provided under the description of output and input functions in this section.

Even if internal memory is present, the use of Port 2 as general purpose I/O pins is not recommended if the device will be used to access external memory. This is because the state of the pins will be disturbed during the memory access. It is still possible, however, to use the Port 2 latch to hold the upper address byte for Register Indirect Addressing instructions.

Most Significant Address Byte, A8–15

When used to address expanded memory, Port 2 functions as the most significant byte of the address bus. Port 2 must function as the address bus on ROMless devices. When serving as a bus, Port 2 will be driven with strong drivers at all times except immediately after the rising edge of $\overline{PSEN}$. See Figure 5–3 and 5–4 for more details.

PORTS 1 AND 3

Ports 1 and 3 are general purpose I/O ports with optional special functions associated with each pin. Enabling the special function automatically converts the I/O pin to that function. To insure proper operation, each alternate function pin should be programmed to a logic 1. For example, enabling the UART converts P3.0 and P3.1 to the serial I/O functions.

The drive characteristics of these pins do not change when the pin is configured for general I/O or as the special function associated with that pin. The exceptions are pins P3.6 and P3.7, which employ the current–limited transition drivers described later when used as $\overline{RD}$ and $\overline{WR}$ signals. The drive characteristics of Port 1 and Port 3 are the same as for Port 2 (non–bus mode). That is, the logic 0 is created by a strong pull–down. The logic 1 is created by a strong transition pull–up that changes to a weak pull–up.

Using one or more I/O pins of a port as special function pins will not effect the remaining port pins. An extreme example is as follows. P3.6 has the alternate function of $\overline{WR}$ and P3.7 of $\overline{RD}$. These strobes are used for expanded data memory access. If a system used only the $\overline{RD}$ signal, then P3.6 would still be available as an I/O port. This is not a practical suggestion, but it illustrates how the special functions are independent.

A more practical application is the optional use of an interrupt. If $\overline{INT0}$ (P3.2) is enabled, then an externally imposed logic 0 will cause an interrupt. By then disabling the $\overline{INT0}$, P3.2 can be used as a general purpose I/O pin. This allows the $\overline{INT0}$ to be used to "wake–up" the system, but does not eliminate another use of the pin.

OUTPUT FUNCTIONS

Although 8051 I/O ports appear to be true I/O, their output characteristics are dependent on the individual port and pin conditions. When software writes a logic 0 to the port for output, the port is pulled to ground. When software writes a logic 1 to the port for output, Ports 1,2, or 3 will drive weak pull–ups (after the strong transition from 0 to 1). Port 0 will go tri–state. Thus as long as the port is not heavily loaded, true logic values will be output. DC drive capability is provided in the electrical specifications. Note that the DC current available from an I/O port pin is a function of the permissible voltage drop.

Transition current is available to help move the port pin from a 0 to a 1. Since the logic 0 driver is strong, no additional drive current is needed in the 1 to 0 direction. The transition current is applied when the port latch is changed from a logic 0 to a logic 1. Simply writing a logic 1 where a 1 was already in place does not change the strength of the pull–up. This transition current is applied for a one half of a machine cycle. The absolute current is not guaranteed, but is approximately 2 mA at 5V.

When serving as an I/O port, the drive will vary as follows. For a logic 0, the port will invoke a strong pull–down. For a logic 1, the port will invoke a strong pull–up for two oscillator cycles to assist with the logic transition. Then, the port will revert to a weak pull–up. This weak pull–up will be maintained until the port transitions from a 1 to a 0. The weak pull–up can be overdriven by external circuits. This allows the output 1 state to serve as the input state as well.

Substantial DC current is available in both the high and low levels. However, the power dissipation limitations make it inadvisable to heavily load multiple pins. In general, sink and source currents should not exceed 10 mA total per port (8 bits) and 25 mA total per package.

CURRENT–LIMITED TRANSITIONS

The High–Speed Microcontroller family incorporates special circuitry to limit the current consumed by the device when the expanded memory bus is used. These signals employ current–limited drivers which "step" the transition from a logic 0 to a logic 1 to reduce ringing and electromagnetic interference. When expanded memory operations are in progress, the following pins will exhibit the current–limiting feature:

Port 0
Port 2
$\overline{\text{PSEN}}$ (During program memory accesses)
ALE
$\overline{\text{RD}}$ (During data memory read cycles)
$\overline{\text{WR}}$ (During data memory write cycles)

INPUT FUNCTIONS

The input state of the I/O ports is the same as that of the output logic 1. That is, the pin is pulled weakly to a logic 1. This 1 state is easily overcome by external components. Thus, after software writes a 1 to the port pin, the port is configured for input. When the port is read by software, the state of the pin will be read. The only exception is the read–modify–write instructions described below. If the external circuit is driving a logic 1, then the pin will be a logic 1. If the external circuit is driving a 0, then it will overcome the internal pull–up. Thus the pin will be the same as the driven logic state. Note that the port latch is not altered by a read operation. Therefore, if a logic 0 is driven onto a port pin from an external source, then removed, the pin will revert to the weak pull–up as determined by the internal latch.

READ–MODIFY–WRITE INSTRUCTIONS

The normal read instructions will read the pin state without regard to the output data latch. The only exception is the read–modify–write category of instructions. They are listed as follows.

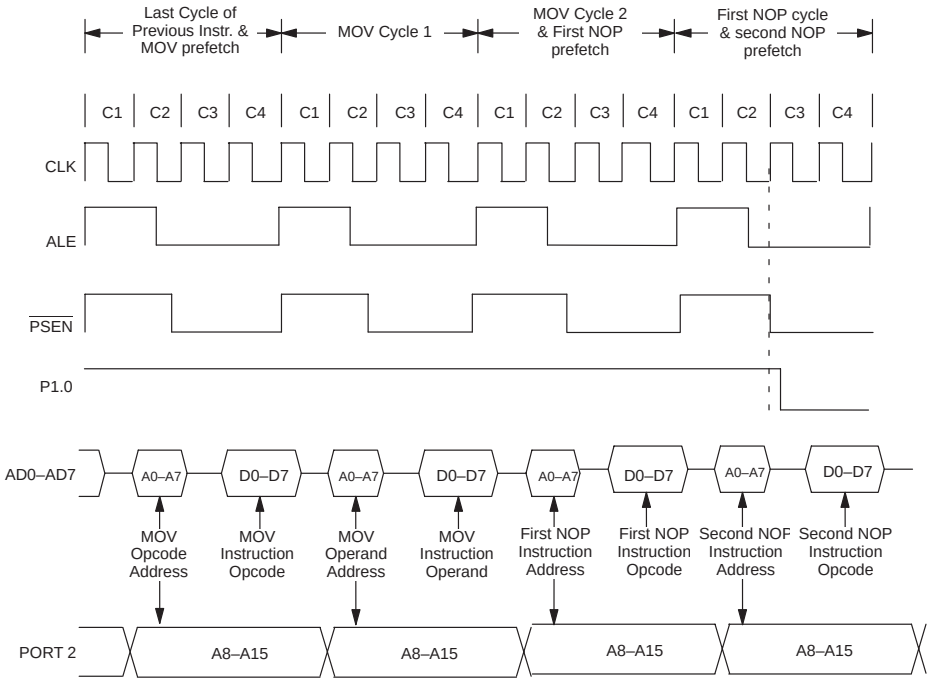
INSTRUCTION	DESCRIPTION
ANL	Logical AND
ORL	Logical OR
XRL	Logical Exclusive OR (XOR)
JBC	Branch if bit set then clear bit
CPL	Complement bit
INC	Increment
DEC	Decrement
DJNZ	Decrement and branch if not zero
MOV PX.n, C	Move the carry bit to bit n of port X
CLR PX.n	Clear bit n of port X
SETB PX.n	Set bit n of port X

The read–modify–write instructions read the state of the latch, then write back the result to the latch. Thus the operation takes place using the value that was originally written to the SFR, without regard to the pin state. The last three instructions listed above are read–modify–write because they read the entire port latch, then write back the changed value. In this case, only one bit will be changed as specified by the instruction.

I/O PORT TIMING

Figure 10–1 shows when port pins change in relationship to instruction timing. The example shown uses a MOV command to change P1.0 from a logic 1 to a logic 0. This diagram is presented to aid the designer in determining the timing relationship for very critical designs. Most designers will not need to consider this much detail.

I/O PORT TIMING FOR MOV INSTRUCTION Figure 10–1



OPTIONAL FUNCTIONS

Every port pin on the High–Speed Microcontroller has an optional special function. These functions are individually selectable. They can also be turned on and off dynamically to suit the application. The optional function for each port pin is described briefly below. More information about each optional function is available in the section dealing with that function.

P0.0	AD0.0 Multiplexed Address/data bus
P2.0	A8 MSB Address bus
P0.1	AD0.1 Multiplexed Address/data bus
P2.1	A9 MSB Address bus
P0.2	AD0.2 Multiplexed Address/data bus
P2.2	A10 MSB Address bus
P0.3	AD0.3 Multiplexed Address/data bus
P2.3	A11 MSB Address bus
P0.4	AD0.4 Multiplexed Address/data bus
P2.4	A12 MSB Address bus
P0.5	AD0.5 Multiplexed Address/data bus
P2.5	A13 MSB Address bus
P0.6	AD0.6 Multiplexed Address/data bus
P2.6	A14 MSB Address bus
P0.7	AD0.7 Multiplexed Address/data bus
P2.7	A15 MSB Address bus
P1.0	T2 Timer 2 output pulse
P3.0	RXD0 Serial Receive UART0
P1.1	T2EX Timer 2 capture/reload input
P3.1	TXD0 Serial Transmit UART0
P1.2	RXD1 Serial Receive UART1
P3.2	$\overline{\text{INT0}}$ External Interrupt 0 active low
P1.3	TXD1 Serial Transmit UART1
P3.3	$\overline{\text{INT1}}$ External Interrupt 1 active low
P1.4	INT2 External Interrupt 2 rising edge active
P3.4	T0 Timer 0 input
P1.5	$\overline{\text{INT3}}$ External Int. 3 falling edge active
P3.5	T1 Timer 1 input
P1.6	INT4 External Interrupt 4 rising edge active
P3.6	$\overline{\text{WR}}$ Write strobe
P1.7	$\overline{\text{INT5}}$ External Int. 5 falling edge active
P3.7	$\overline{\text{RD}}$ Read strobe

SECTION 11: PROGRAMMABLE TIMERS

All members of the High–Speed Microcontroller family incorporate three 16–bit programmable timers and some also have a Watchdog Timer with a programmable interval. Because the Watchdog Timer is significantly different from the other timers, it is described separately. The 16–bit Timers are referred to simply as timers.

In most modes, the timers can be used as either counters of external events or timers. When functioning as a counter, 1 to 0 transitions on a port pin are monitored and counted. When functioning as timers, they effectively count oscillator cycles. The time base for the timer function is the main oscillator clock divided by either 4 or 12. This selection is described below. Because each clock pulse must be sampled high for one machine cycle and low for one machine cycle to be recognized, this sets the maximum sampling frequency on any timer input at 1/8 of the main oscillator frequency.

The three timers are compatible with the 80C32. That is, they offer the same controls and I/O functions that were available in the 80C32. As mentioned above, the actual timing of these functions is user selectable to be compatible with the machine cycle of the older generation of 8051 family (12 clocks per tick) or the new generation (4 clocks per tick). The timing for each of the three timers can be selected independently and can be changed dynamically. Each timer has 4 primary modes as discussed below.

The Watchdog Timer Reset provides CPU monitoring by requiring software to clear the timer before the user selected interval expires. If the Timer is not cleared, the CPU will be reset by the Watchdog. The Watchdog function is optional and is described below. Since the High–Speed Microcontroller timers have a variety of features, the following summary table shows the capabilities.

Timer 0
13–bit Timer/counter
16–bit Timer/counter
8–bit Timer w/ Auto–reload
Two 8–bit Timer/counters
External control pulse timer/counter

Timer 1
13–bit Timer/counter
16–bit Timer/counter
8–bit Timer w/ Auto–reload
External control pulse timer/counter
Baud rate generator

Timer 2
16–bit Timer/counter
16–bit Timer with capture
16–bit Auto–reload Timer/counter
16–bit up/down auto–reload Timer/counter
Baud rate generator
Timer output clock generator

16–BIT TIMERS

Timers 0 and 1 are nearly identical. Timer 2 has several additional features such as up/down counting, capture values and an optional output pin that make it unique. Timers 0 and 1 are described first. Timer 2 is described separately. As mentioned above, the time base for each timer can be varied and this is also discussed in more detail below.

Timer 0 and 1 both have four operating modes. They are 13–bit timer/counter, 16–bit timer/counter, 8–bit timer/counter with auto–reload, and two 8–bit timers. The latter mode is available to Timer 0 only. These modes are controlled by the TMOD register. Each timer can also serve as a counter of external pulses (1 to 0 transition) on the corresponding Tn pin. This selection is controlled by the TMOD register. One other option is to gate the timer/counter using an external control signal. This allows the timer to measure the pulse width of external signals. Timers 0 and 1 are enabled using the TCON register which is also the location of their flags. The registers are described below. Following this is a detailed explanation of the four operating modes.

Each timer consists of a 16–bit register in two bytes. These are called TL0, TH0, TL1, and TH1. As shown, each timer is broken into low and high bytes. Software can read or write any of these locations at any time.

TMOD REGISTER SUMMARY

TIMER MODE CONTROL **TMOD; 89h**

TMOD.7 GATE – Timer 1 GATE control. When GATE=1, Timer 1 will clock only when $\overline{INT1}$ and TR1 =1. When GATE=0, Timer 1 will clock only when TR1=1 irrespective of $\overline{INT1}$.

TMOD.6 C/$\overline{T}$ – Counter/Timer select. When C/$\overline{T}$ is set to a 0, Timer 1 is incremented by internal clocks. When C/$\overline{T}$ is set to a 1, Timer 1 counts based on the T1 (P3.5) pin.

TMOD.5 M1 – Timer 1 Mode select bit 1.

TMOD.4 M0 – Timer 1 Mode select bit 0.

M1	M0	Mode
0	0	Mode 0 : 8–bits with 5–bit prescale
0	1	Mode 1 : 16–bits
1	0	Mode 2 : 8–bits with auto–reload
1	1	Mode 3 : Timer 1 stopped

TMOD.3 GATE – Timer 0 GATE control. When GATE=1, Timer 0 will clock only when $\overline{INT0}$ and TR0 =1. When GATE=0, Timer 0 will clock only when TR0=1 irrespective of $\overline{INT0}$.

TMOD.2 C/$\overline{T}$ – Counter/Timer select. When C/$\overline{T}$ is set to a 0, Timer 0 is incremented by internal clocks. When C/$\overline{T}$ is set to a 1, Timer 0 counts based on the T0 (P3.4) pin.

TMOD.1 M1 – Timer 0 Mode select bit 1.

TMOD.0 M0 – Timer 0 Mode select bit 0.

M1	M0	Mode
0	0	Mode 0 : 8–bits with 5–bit prescale
0	1	Mode 1 : 16–bits
1	0	Mode 2 : 8–bits with auto–reload
1	1	Mode 3 : Timer 0 is two 8–bit timers

TCON REGISTER SUMMARY

TIMER/COUNTER CONTROL	TCON; 88h
TCON.7	TF1 – Timer 1 overflow flag. Set to one when Timer 1 overflows from FFh, and cleared when the processor vectors to the interrupt service routine.
TCON.6	TR1 – Timer 1 run control. Turns on Timer 1 when this bit is set.
TCON.5	TF0 – Timer 0 overflow flag. Set to one when Timer 0 overflows from FFh, and cleared when the processor vectors to the interrupt service routine.
TCON.4	TR0 – Timer 0 run control. Turns on Timer 0 when this bit is set to a one.
TCON.3	IE1 – Interrupt 1 edge detect. Set by hardware when an edge/level is detected on $\overline{INT1}$.
TCON.2	IT1 – Interrupt 1 type select. $\overline{INT1}$ detects a falling edge when this bit is set to a 1. $\overline{INT1}$ detects a low level when this bit is a 0.
TCON.1	IE0 – Interrupt 0 edge detect. Set by hardware when an edge/level is detected on $\overline{INT0}$.
TCON.0	IT0 – Interrupt 0 type select. $\overline{INT0}$ detects a falling edge when this bit is set to a 1. $\overline{INT0}$ detects a low level when this bit is a 0.

MODE 0

Mode 0 configures either Timer 0 or Timer 1 for operation as a 13–bit Timer/Counter. As shown in figure 11–1, bits M1=0 and M0=0 of the TMOD register select this operating mode.

When using Timer 0, TL0 uses only bits 0–4. These bits serve as the 5 LSbs of the 13–bit timer. TH0 provides the 8 MSbs of the 13–bit timer. Bit 4 of TL0 is used as a ripple out to TH0 bit 0, thereby completely bypassing bits 5 through 7 of TL0. Once the timer is started using the TR0 (TCON.4) timer enable, the timer will count as long as GATE (TMOD.3) is 0 or GATE is 1 and pin $\overline{INT0}$ is 1. It will count oscillator cycles if C/$\overline{T}$ (TMOD.2) is set to a logic 0 and 1 to 0 transitions on T0 (P3.4) if C/$\overline{T}$ is set to a 1. When the 13–bit count reaches 1FFFh (all ones), the next count will cause it to roll over to 0000h. The TF0 (TCON.5) flag will be set and an interrupt will occur if enabled. The upper three bits of TL0 will be indeterminate.

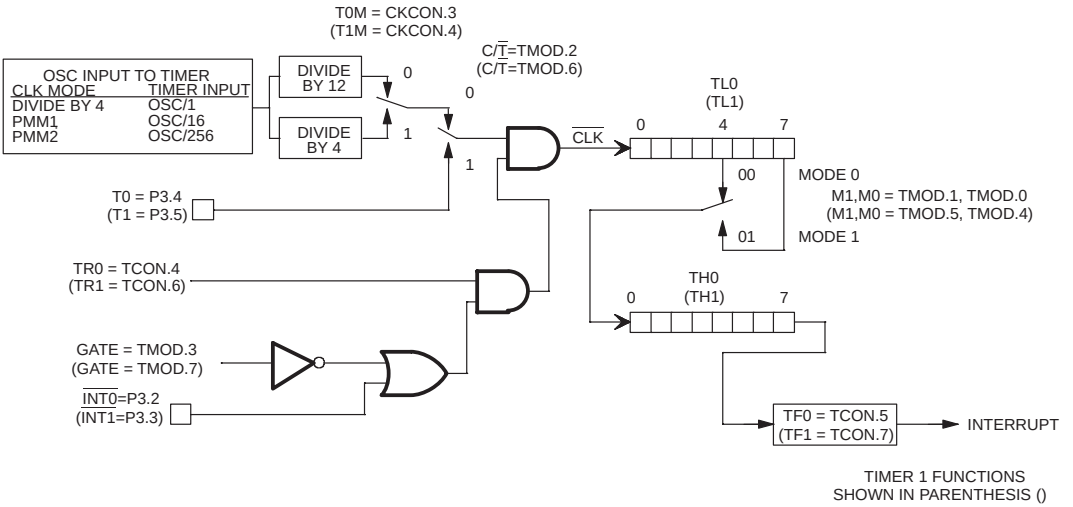
Note that when used as a timer, the time base may be either oscillator cycles/12 or oscillator cycles/4 as selected by bits TnM (n=0 or 1) of the CKCON register. This feature is described in more detail below.

Mode 0 operates identically when Timer 1 is used. The same information applies to TL1 and TH1, which form the 13–bit register. TR1 (TCON.6), $\overline{INT1}$ (P3.3), T1 (P3.5), and the relevant C/$\overline{T}$ (TMOD.6) and GATE (TMOD.7) bits have the same functions.

MODE 1

Mode 1 configures the timer for 16–bit operation as either a timer or counter. Figure 11–1 shows that bits M1=0 and M0=1 of the TMOD register select this operating mode. For Timer n, all of the TLn and THn registers are used. For example, if Timer 1 is configured in mode 1, then TL1 holds the LSB and TH1 holds the MSB. Rollover occurs when the timer reaches FFFFh. An interrupt will also occur if enabled and the relevant TFn flag is set. Timebase selection, counter/timer selection, and the gate function operate as described in mode 0.

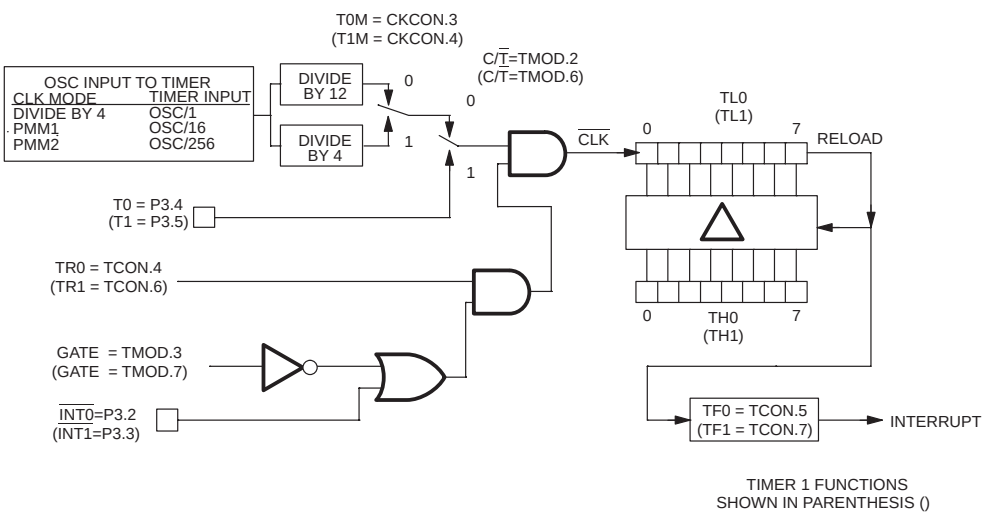
TIMER/COUNTER 0 AND 1 MODES 0 AND 1 Figure 11–1



MODE 2

This mode configures the timer as an 8–bit timer/counter with automatic reload of the start value. This configuration is shown in Figure 11–2, and is selected when bits M1and M0 of the TCON register are set to 1 and 0 respectively. When configured in Mode 2, the timer uses TLn to count and THn to store the reload value. Software must initialize both TLn and THn with the same starting value for the first count to be correct. Once the TLn reaches FFh, it will be automatically loaded with the value in THn. The THn value remains unchanged unless modified by software. Mode 2 is commonly used to generate baud rates since it runs without continued software intervention. As in modes 0 and 1, mode 2 allows counting of either oscillator cycles (crystal/12 or crystal/4) or pulses on pin Tn (C/T=1) when counting is enabled by TRn and the proper setting of GATE and INTn pins.

TIMER/COUNTER 0 AND 1 MODE 2 Figure 11–2

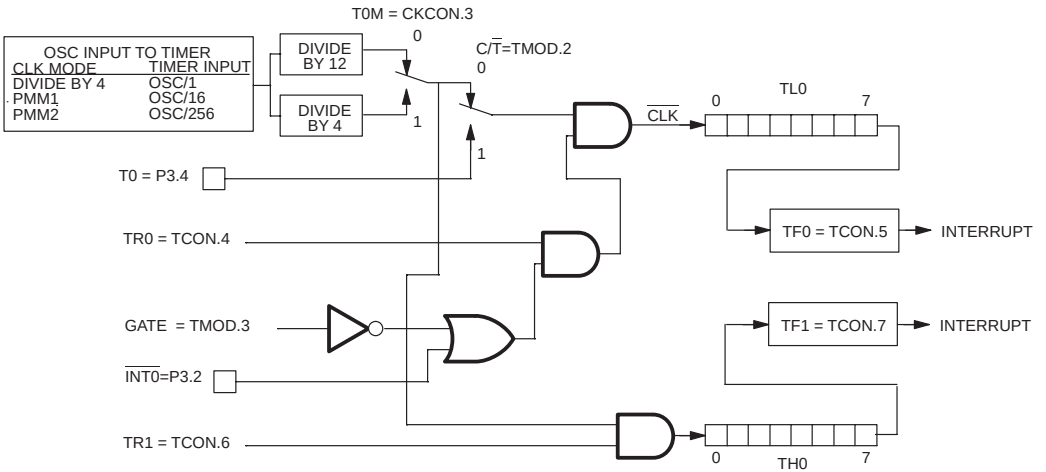


MODE 3

This mode provides an 8–bit timer/counter and a second 8–bit timer as indicated in Figure 11–3. In Mode 3, TL0 is an 8–bit timer/counter controlled by the normal Timer 0 bits (TR0=TCON.4 and TF0=TCON.5). TL0 can be used to count oscillator cycles (crystal/12 or crystal/4) or 1 to 0 transitions on pin T0 as determined by C/$\overline{T}$ (TMOD.2). As in the other modes, the GATE function can use $\overline{INT0}$ to give external run control of the timer to an outside signal.

TH0 becomes an independent 8–bit Timer in Mode 3, however it can only count oscillator cycles (divided by 12 or 4) as shown in the figure. In this mode, some of Timer 1's control signals are used to manipulate TH0. That is, TR1 (TCON.6) and TF1 (TCON.7) become the relevant control and flag bits associated with TH0.

TIMER/COUNTER 0 MODE 3 Figure 11–3



In Mode 3, Timer 1 stops counting and holds its value. Thus Timer 1 has no practical application while in Mode 3.

As mentioned above, when Timer 0 is in Mode 3, it uses some of Timer 1's resources (i.e., TR1 and TF1). Timer 1 can still be used in Modes 0, 1, and 2 in this situation, but its flexibility becomes somewhat limited. While it maintains its basic functionality, its inputs and outputs are no longer available. Therefore when Timer 0 is in Mode 3, Timer 1 can only count oscillator cycles, and it does not have an interrupt or flag. With these limitations, baud rate generation is its most practical application, but other time–base functions may be achieved by reading the registers.

TIMER 2

Like Timers 0 and 1, Timer 2 is a full function timer/counter, however it has several additional capabilities that make it more useful. Timer 2 has independent control registers in T2CON and T2MOD, and is based on count registers TL2 and TH2. All of these registers are described in detail below.

T2CON REGISTER SUMMARY

TIMER TWO CONTROL	T2CON; C8h
T2CON.7	TF2 – Timer 2 Overflow Flag. Hardware will set TF2 when the Timer 2 overflows from FFFFh or from the count equal to the capture register in down count mode. It must be cleared to 0 by software. TF2 will only be set to a 1 if RCLK and TCLK are both cleared to a 0.
T2CON.6	EXF2 – Timer 2 External Flag. Hardware will set EXF2 when a reload or capture is caused by a falling transition on the T2EX pin (P1.1). EXEN2 must be set for this function. This flag must be cleared to 0 by software. Writing a one to this bit will force a timer interrupt if enabled.
T2CON.5	RCLK – Receive Clock Flag. This bit determines whether Timer 1 or 2 is used for Serial Port 0 timing of received data in Serial Modes 1 or 3. RCLK =1 causes Timer 2 overflow to be used as the receive clock. RCLK=0 causes Timer 1 overflow to be used as the receive clock.
T2CON.4	TCLK – Transmit Clock Flag. This bit determines whether Timer 1 or 2 is used for Serial Port 0 timing of Transmit data in Serial Modes 1 or 3. TCLK=1 causes Timer 2 overflow to be used as the transmit clock. TCLK=0 causes Timer 1 overflow to be used as the transmit clock.
T2CON.3	EXEN2 – Timer 2 External Enable. Setting this bit to a 1 allows a capture or reload to occur as a result of a falling transition on T2EX (P1.1), if Timer 2 is not generating baud rates for the serial port. EXEN2=0 causes Timer 2 to ignore all external events at T2EX.
T2CON.2	TR2 – Timer 2 run. Setting this bit to a 1 starts Timer 2. Setting it to a 0 stops Timer 2.
T2CON.1	C/$\overline{T2}$ – Counter/Timer Select. Setting this bit to a 0 selects a timer function for Timer 2. Setting it to a 1 selects a counter of falling transitions on T2 (P1.0). Timer 2 runs at 4 clocks per tick or 12 clocks per tick as programmed by CKCON.5. This bit will be overridden and Timer 2 directed to use a divide by 2 clock if either the baud–rate generator or clock output mode is used.
T2CON.0	CP/$\overline{RL2}$ – Capture/Reload Flag. When this bit is set to 1, Timer 2 captures will occur on 1 to 0 transitions of T2EX (P1.1) if EXEN2=1. When this bit is set to 0, auto–reloads will occur when Timer 2 overflows or when 1 to 0 transitions occur on T2EX if EXEN2=1. If either RCLK or TCLK is set to a 1 this bit will not function and the timer will function in an auto–reload mode following each overflow.

T2MOD REGISTER SUMMARY

TIMER TWO MODE CONTROL	**T2MOD; C9h**
T2MOD.7–2	Reserved
T2MOD.1	T2OE – Timer 2 Output Enable. Setting this bit to a 1 enables the Timer 2 to drive the T2 (P1.0) pin with a clock output. When T2OE=0, the T2 (P1.0) pin is used as either an input for Timer 2 or a standard port pin.
T2MOD.0	DCEN – Down Count Enable. When this bit is set to 1, the Timer 2 function counts up or down when in 16–bit auto–reload mode depending on T2EX (P1.1). When DCEN is set to a 0, the Timer 2 counts up only.

TIMER 2 CAPTURE REGISTERS SUMMARY

LEAST SIGNIFICANT BYTE CAPTURE OF TIMER 2	**RCAP2L; CAh**
RCAP2L.7–0	This register is used to capture the TL2 value when Timer 2 is configured in capture mode. RCAP2L is also used as the LSB of a 16–bit reload value when Timer 2 is configured in auto–reload mode.
MOST SIGNIFICANT BYTE CAPTURE OF TIMER 2	**RCAP2H; CBh**
RCAP2H.7–0	This register is used to capture the TH2 value when Timer 2 is configured in capture mode. RCAP2H is also used as the MSB of a 16–bit reload value when Timer 2 is configured in auto–reload mode.

TIMER 2 MODES

As is seen in the register descriptions, Timer 2 has several abilities not found in Timers 0 and 1. However, it does not offer the 13–bit and dual 8–bit modes. Thus it runs in 16–bit mode at all times. Also note that instead of offering an 8–bit auto–reload mode, Timer 2 has a 16–bit auto–reload mode. This mode uses the Timer Capture registers to hold the reload values. The modes available on Timer 2 are described below.

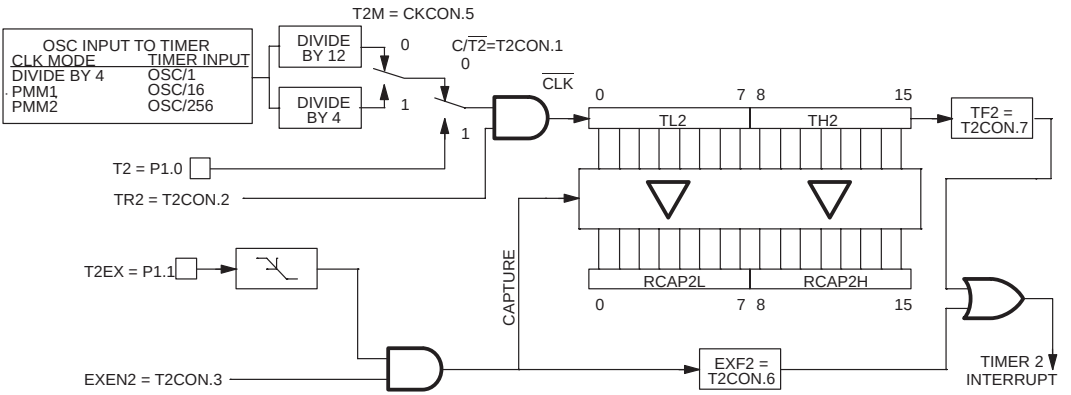
16–bit Timer/Counter

In this mode, Timer 2 performs a simple timer or counter function where it behaves similarly to mode 1 of Timers 0 and 1, but uses 16 instead of 8 bits. This mode, along with the optional capture mode described below, is illustrated in Figure 11–4. The 16–bit count values are found in TL2 and TH2 Special Function Registers (addresses 0CCh and 0CDh respectively). The selection of whether a Timer or Counter function is performed is made using the bit C/$\overline{T2}$ (T2CON.1). When C/$\overline{T2}$ is set to a logic 1, Timer 2 behaves as a counter where it counts 1 to 0 transitions at the T2 (P1.0) pin. When C/$\overline{T2}$ is set to a logic 0, Timer 2 functions as a timer where it counts the oscillator cycles divided by either 12 or 4 as determined by bit T2M (T2CON.5). Timing or counting is enabled by setting bit TR2 (T2CON.2) to 1, and disabled by setting it to 0. When the counter rolls over from FFFFh to 0000h, the TF2 flag (T2CON.7) is set and will cause an interrupt if Timer 2's interrupt is enabled.

16–bit Timer with Capture

A diagram of Timer 2's Capture Mode is shown in Figure 11–4. In this mode, the timer performs basically the same 16–bit timer/counter function described above. However, a 1 to 0 transition on T2EX (pin P1.1) causes the value in Timer 2 to be transferred into the capture registers if enabled by EXEN2 (T2CON.3). The capture registers, RCAP2L and RCAP2H, correspond to TL2 and TH2 respectively. The capture function is enabled by the CP/$\overline{RL2}$ (T2CON.0) bit. When set to a logic 1, the timer is in capture mode as described. When set to a logic 0, the timer is in auto–reload mode described later. As was possible with Timers 0 and 1, the timebase for Timer 2 may be selected to be oscillator cycles divided by either 12 or 4 when in this mode.

TIMER/COUNTER 2 WITH OPTIONAL CAPTURE Figure 11–4



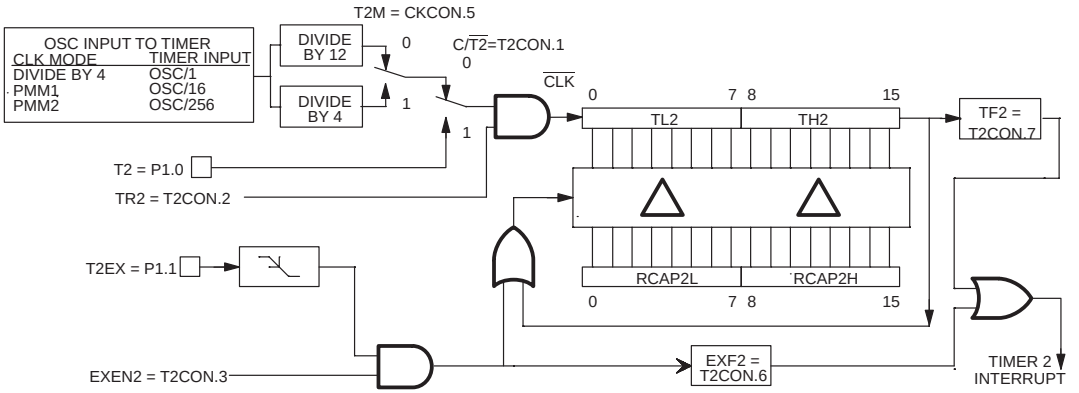
16–bit Auto–reload Timer/Counter

This mode is illustrated in Figure 11–5a. When Timer 2 reaches an overflow state, i.e., rolls over from FFFFh to 0000, it will set the TF2 Flag. This flag can generate an interrupt if enabled. In addition, the timer will restore its starting value and begin timing (or counting) again. The starting value is preloaded by software into the capture registers RCAP2L and RCAP2H. These registers cannot be used for capture functions while also performing auto–reload, so these modes are mutually exclusive. Auto–reload is invoked by the CP/$\overline{RL2}$ (T2CON.0) bit. When set to a logic 0, the timer is in auto–reload mode. When CP/$\overline{RL2}$ is set to a logic 1, the timer is in capture mode described above. If the oscillator timebase is used (C/$\overline{T2}$=T2CON.1=0), the timer's input may be selected to be oscillator cycles divided by either 12 or 4 as determined by T2M (CKCON.5). Otherwise, pulses on pin T2 (P1.0) are counted when C/$\overline{T2}$=1. As in other modes, Counting or timing is enabled or disabled with TR2 (T2CON.2).

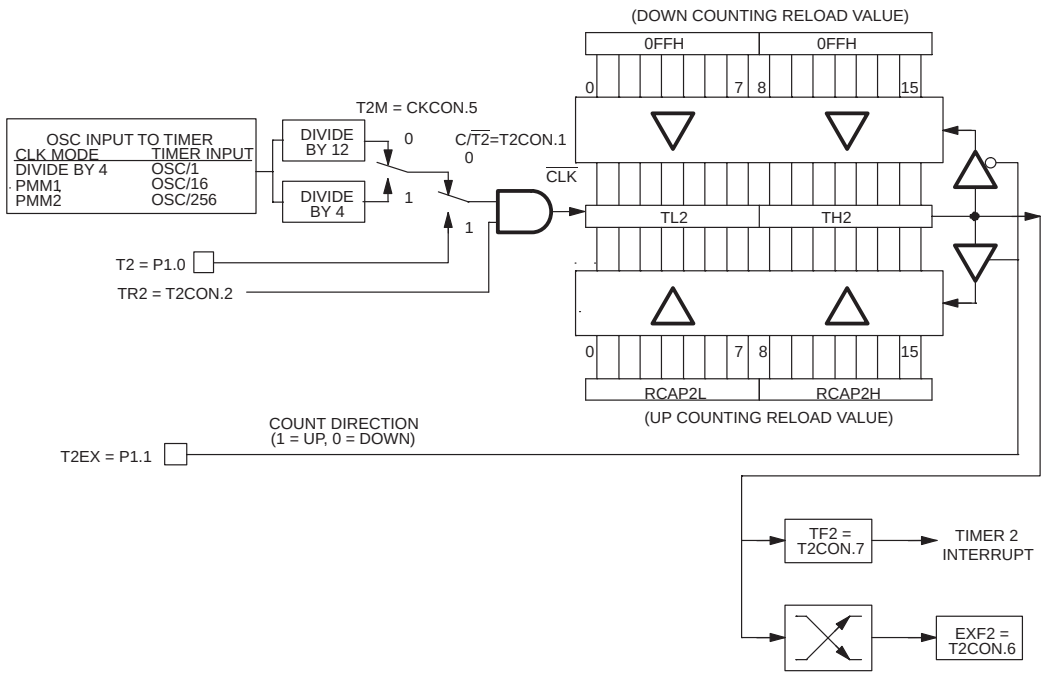
When in auto–reload mode, Timer 2 can also be forced to reload with the T2EX (P1.1) pin. A 1 to 0 transition will force a reload if enabled by the EXEN2 (T2CON.3) bit. If EXEN2 is set to a logic 1, then a 1 to 0 transition on T2EX will cause a reload. Otherwise, the T2EX pin will be ignored.

TIMER/COUNTER 2 AUTO RELOAD MODE Figure 11–5

(a) DCEN=0



(a) DCEN=1



Up/Down Count Auto–reload Timer/Counter

The up/down auto–reload counter option is selected by the DCEN (T2MOD.0) bit, and is illustrated in Figure 11–5b. When DCEN is set to a logic 1, Timer 2 will count up or down as controlled by the state of pin T2EX (P1.1). T2EX will cause upward counting when a logic 1 is applied and down counting when a logic 0 is applied. When DCEN=0, Timer 2 only counts up.

When an upward counting overflow occurs, the value in RCAP2L and RCAP2H will load into T2L and T2H. In the down count direction, an underflow occurs when T2L and T2H match the values in RCAP2L and RCAP2H respectively. When an underflow occurs, FFFFh is loaded into T2L and T2H and counting continues.

Note that in this mode, the overflow/underflow output of the timer is provided to an edge detection circuit as well as to the TF2 bit (T2CON.7). This edge detection circuit toggles the EXF2 bit (T2CON.6) on every overflow or underflow. Therefore, the EXF2 bit behaves as a seventeenth bit of the counter, and may be used as such.

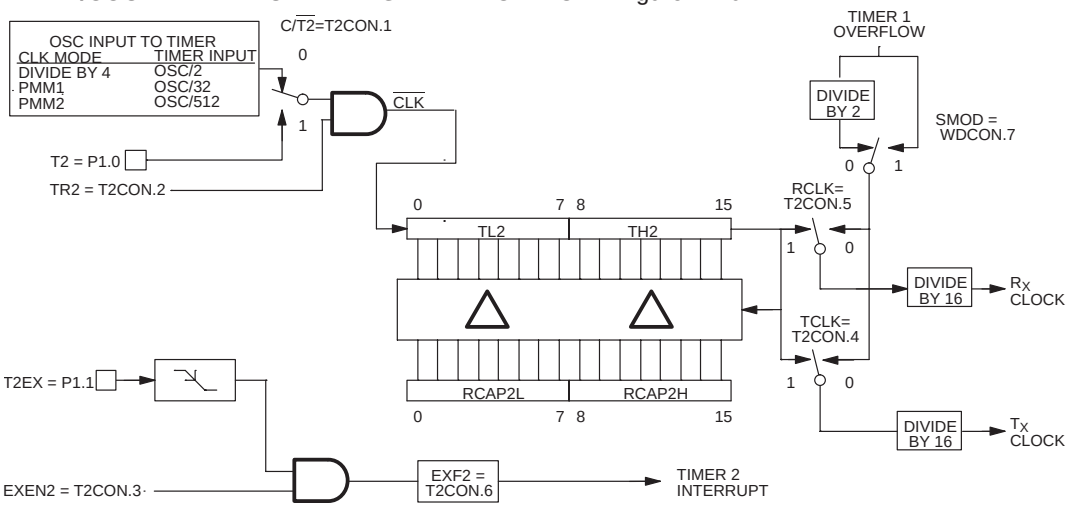
Baud Rate Generator

Timer 2 can be used to generate baud rates for Serial Port 0 in serial modes 1 or 3. Baud rate generator mode is invoked by setting either the RCLK or TCLK bit in the T2CON register to a logic 1, as illustrated in Figure 11–6. In this mode, the timer continues to function in auto–reload mode, but instead of setting the interrupt flag T2F (T2CON.7) and potentially causing an interrupt, the overflow generates the shift clock for the serial port function. As in normal auto–reload mode, an overflow causes RCAP2L and RCAP2H to be transferred into T2L and T2H respectively. Note that when RCLK or TCLK is set to 1, the Timer 2 is forced into 16–bit auto–reload mode regardless of the CP/$\overline{RL2}$ bit.

As explained above, the timer itself cannot set the T2F interrupt flag and therefore cannot generate an interrupt. However if EXEN2 (T2CON.3) is set to 1, a 1 to 0 transition on the T2EX (P1.1) pin will cause the EXF2 (T2CON.6) interrupt flag to be set. If enabled, this will cause a Timer 2 Interrupt to occur. Therefore in this mode, the T2EX pin may be used as an additional external interrupt if desired.

Another feature of the baud rate generator mode is that the crystal derived timebase for the timer is the crystal frequency divided by 2. No other crystal divider selection is possible. If a different timebase is desired, bit C/$\overline{T2}$ (T2CON.1) may be set to a 1 sourcing the timebase from an external clock source supplied by the user on pin T2 (P1.0). Software should not access TL2 or TH2 while the timer is running (TR2=1) in baud rate generator mode. In this mode the timer is clocking so fast that a software read of or write to the TL2 and TH2 registers may corrupt the timer. The RCAP registers may be read, but not modified, while TR2=1. Stop the timer (TR2=0) to modify these registers.

TIMER/COUNTER 2 BAUD RATE GENERATOR MODE Figure 11–6



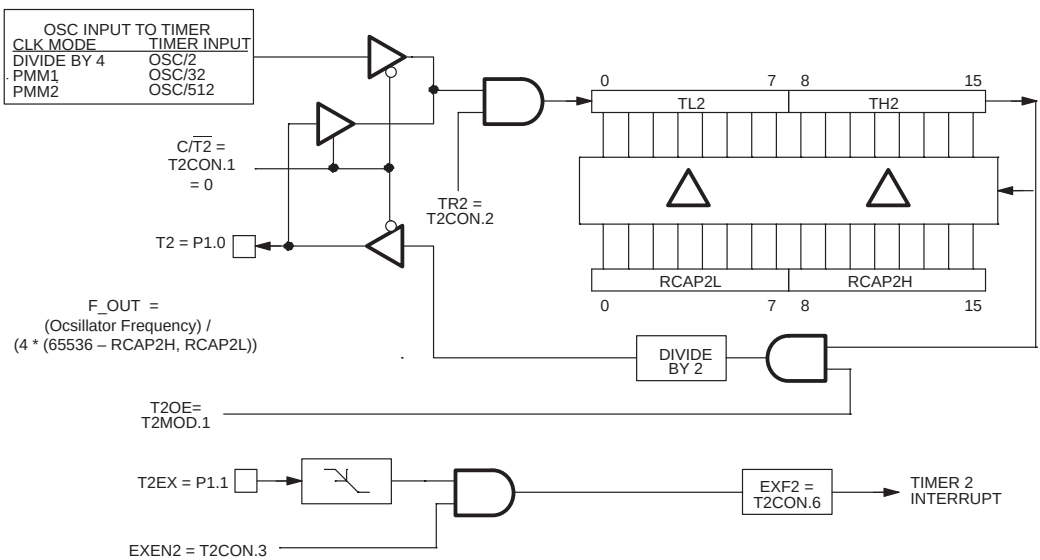
Timer Output Clock Generator

Timer 2 can also be configured to drive a clock output on port pin P1.0 (T2) as shown in Figure 11–7. To configure Timer 2 for this mode, first it must be set to 16–bit auto–reload timer mode (CP/$\overline{RL2}$=0, C/$\overline{T2}$=0). Next, the T2OE (T2MOD.1) bit must be set to a logic 1. TR2 (T2CON.2) must also be set to a logic 1 to enable the timer.

This mode will produce a 50% duty cycle square wave output. The frequency of the square wave is given by the formula in the figure. Each timer overflow causes an edge transition on the pin, i.e., the state of the pin toggles.

Note that this mode has two somewhat unique features in common with the baud rate generation mode. First, the timebase is the crystal frequency divided by 2, and no other divider selection is possible. Second, the timer itself will not generate an interrupt, but if needed, an additional external interrupt may be caused using T2EX as described above. Because of the two mode's similarities, the timer can be used to generate both an external clock and a baud rate clock simultaneously. Once the clock out mode is established, either TCLK or RCLK is set to 1, and the RCAP2 registers are loaded, the timer will provide a clock to both functions.

TIMER/COUNTER 2 CLOCK OUT MODE Figure 11–7



TIME–BASE SELECTION

The High–Speed Microcontroller allows the user to select either 4 or 12 clocks as the time–base for each timer independently. When using the 16–bit Timer/Counters in timer mode, the timer/counter counts the oscillator cycles divided by a predetermined number. In the standard 8051, the 8051 timers count the oscillator divided by 12, which is the standard 8051 machine cycle timing. The High–Speed Microcontroller allows the option of setting the timers to operate from a divide by 4 of the input clock to allow higher precision timing and faster baud rates. This selection has no effect on CPU timing, only on the timers. Following a reset, the timers default to 12 clocks as the time–base to remain drop–in compatible with the original 8051.

The 4 or 12 clock decision is independent for each timer and the default is 12 clocks per timer tick. As an example, a user might select both the baud rate generator timer and one other timer to run at 12 clocks per timer tick with the third timer at 4 clocks per tick. This allows one timer to measure higher speed events or to gain better resolution. The control bits for the time–base selection are located in the Clock Control register (CKCON;8Eh). Timer 2 will function at 2 clocks per tick

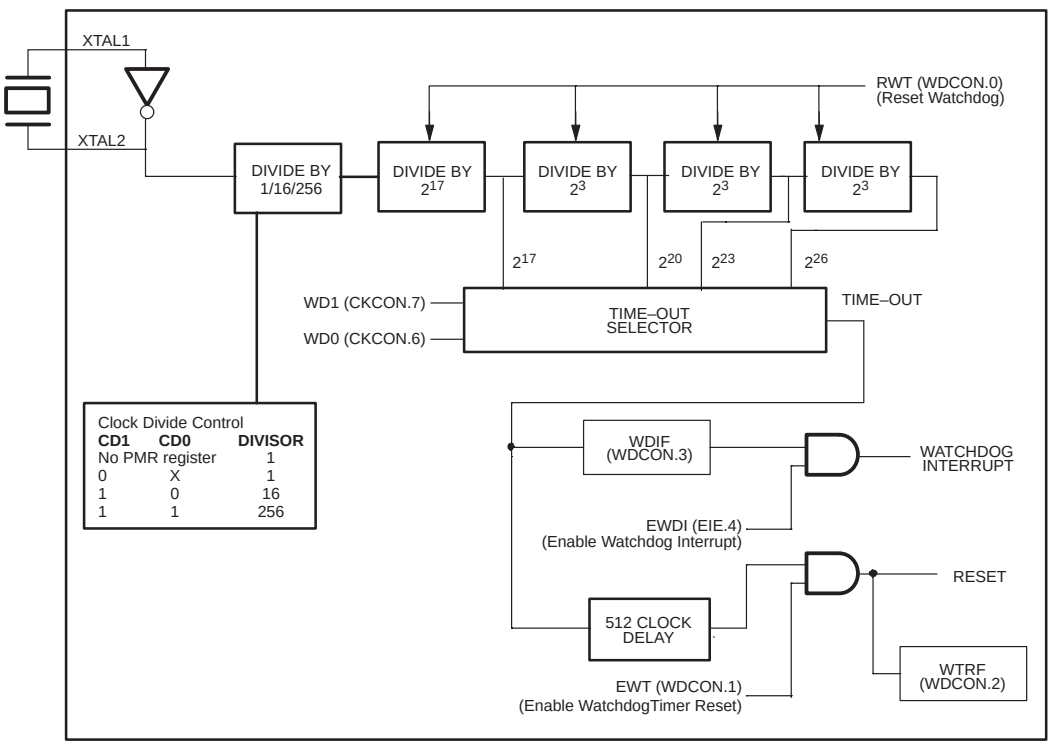
when set for baud rate generation or clock output as described above.

The use of Power Management modes will affect the input clock to the timer as shown in the illustrations. In general, they will divide the input clock by either 16 or 256 for PMM1 and PMM2, respectively. Timer 2, when operating in Baud Rate Generator or Clock Out mode normally uses the input clock frequency divided by 2, but when PMM1 and PMM2 are used, it will operate from a time–base of the input clock divided by 32 and 512, respectively.

WATCHDOG TIMER

The Watchdog Timer is a user programmable clock counter that can serve as a time–base generator, an event timer, or a system supervisor. As can be seen in the diagram of Figure 11–8, the timer is driven by the main system clock that is supplied to a series of dividers. The divider output is selectable, and determines the interval between time–outs. When the time–out is reached, an interrupt flag will be set, and if enabled, a reset will occur. The interrupt flag will cause an interrupt to occur if its individual enable bit is set and the global interrupt enable is set. The reset and interrupt are completely discrete functions that may be acknowledged or ignored, together or separately for various applications.

WATCHDOG TIMER Figure 11–8



The Watchdog Timer Reset function works as follows. After initializing the correct time–out interval (discussed below), software first restarts the Watchdog using RWT (WDCON.0) and then enables the reset mode by setting the Enable Watchdog Timer Reset (EWT=WDCON.1) bit. At any time prior to reaching its user selected terminal value, software can set the Reset Watchdog Timer (RWT=WDCON.0) bit. If RWT is set before the time–out is reached, the timer will start over. If the time–out is reached without RWT being set, the Watchdog will reset

the CPU. Hardware will automatically clear RWT after software sets it. When the reset occurs, the Watchdog Timer Reset Flag (WTRF=WDCON.2) will automatically be set to indicate the cause of the reset, however software must clear this bit manually.

The Watchdog Timer is a free running timer. When used as a simple timer with both the reset and interrupt functions disabled (EWT=0 and EWDI=0), the timer will continue to set the Watchdog Interrupt flag each time the timer completes the selected timer interval as programmed by WD1 (CKCON.7) and WD0 (CKCON.6). Restarting the timer using the RWT (WDCON.0) bit, allows software to use the timer in a polled time–out mode. The WDIF bit is cleared by software or any reset.

The Watchdog Interrupt is also available for applications that do not need a true Watchdog Reset but simply a very long timer. The interrupt is enabled using the Enable Watchdog Timer Interrupt (EWDI=EIE.4) bit. When the time–out occurs, the Watchdog Timer will set the WDIF bit (WDCON.3), and an interrupt will occur if the global interrupt enable (EA=IE.7) is set. Note that WDIF is set 512 clocks before a potential Watchdog Reset. The Watchdog Interrupt Flag will indicate the source of the interrupt, and must be cleared by software.

Using the Watchdog Interrupt during software development can allow the user to select ideal watchdog reset locations. Code is first developed without enabling the Watchdog Interrupt or Reset functions. Once the program is complete, the Watchdog Interrupt function is enabled to identify the required locations in code to set the RWT (WDCON.0) bit. Incrementally adding instructions to reset the Watchdog Timer prior to each address location (identified by the Watchdog Interrupt) will allow the code to eventually run without receiving a Watchdog Interrupt. At this point the Watchdog Timer Reset can be enabled without the potential of generating unwanted resets. At the same time the Watchdog Interrupt may also be disabled. Proper use of the Watchdog Interrupt with the Watchdog Reset allows interrupt software to survey the system for errant conditions.

When using the Watchdog Timer as a system monitor, the Watchdog Reset function should be used. If the Interrupt function were used, the purpose of the watchdog would be defeated. For example, assume the system is executing errant code prior to the Watchdog Interrupt. The interrupt would temporarily force the system back into control by vectoring the CPU to the interrupt service routine. Restarting the Watchdog and exiting by an RETI or RET, would return the processor to the lost position prior to the interrupt. By using the Watchdog Reset function, the processor is restarted from the beginning of the program, and therefore placed into a known state.

The Watchdog has four time–out selections based on the input crystal frequency as shown in the figure. The selections are a preselected number of clocks. Therefore, the actual time–out interval is dependent on the crystal frequency. Shown below are the four time–outs with some example periods for different crystal speeds. Note that the time period shown is for the interrupt event. The reset, when enabled, will occur 512 clocks later regardless of whether the interrupt is used. Therefore the actual Watchdog time–out period is the number shown below plus 512 clocks. Watchdog generated resets will last for two machine cycles.

WD1	WD0	WATCHDOG INTERVAL	NUMBER OF CLOCKS	TIME AT 1.8432 MHz	TIME AT 11.0592 MHz	TIME AT 16 MHz	TIME AT 20 MHz	TIME AT 25 MHz
0	0	2^{17}	131072	71.11 ms	11.85 ms	8.19 ms	6.55 ms	5.24 ms
0	1	2^{20}	1048576	568.89 ms	94.81 ms	65.54 ms	52.43 ms	41.94 ms
1	0	2^{23}	8388608	4551.11 ms	758.52 ms	524.29 ms	419.43 ms	335.54 ms
1	1	2^{26}	67108864	36408.88 ms	6068.15 ms	4194.30 ms	3355.44 ms	2684.35 ms

The Watchdog time–out selection is made using bits WD1 (CKCON.7) and WD0 (CKCON.6) as shown in the figure. The time–out selections possible are shown in the bit descriptions that follow. The watchdog timeout period is affected by the use of Power Management modes. The slower clock rate, either divide by 64 or divide by 1024 is used as the input source for the watchdog timer. This allows the watchdog period to remain synchronized with device operation.

As discussed above, the Watchdog Timer has several SFR bits that contribute to its operation. It can be enabled to function as either a reset source, interrupt source, software polled timer or any combination of the three. Both the reset and interrupt have status flags. The Watchdog also has a bit that restarts the timer. A summary table showing the bit locations is below. A description follows.

BIT NAME	DESCRIPTION	REGISTER LOCATION	BIT POSITION
EWT	Enable Watchdog Timer Reset	WDCON – D8h	WDCON.1
RWT	Reset Watchdog Timer	WDCON – D8h	WDCON.0
WD1	Watchdog interval 1	CKCON – 8Eh	CKCON.7
WD0	Watchdog interval 0	CKCON – 8Eh	CKCON.6
WTRF	Watchdog Timer Reset Flag	WDCON – D8h	WDCON.2
EWDI	Enable Watchdog Timer Interrupt	EIE – E8h	EIE.4
WDIF	Watchdog Interrupt Flag	WDCON – D8h	WDCON.3

The Watchdog Timer is a free running timer. It will be disabled by a Power–fail Reset. A Watchdog time–out reset will not disable the Watchdog Timer, but will restart the timer. In general, software should set the Watchdog to whichever state is desired, just to be certain of its state. Control bits that support Watchdog operation are described below.

WDCON REGISTER SUMMARY

WATCHDOG CONTROL **WDCON; D8h**

WDCON.3 WDIF – Watchdog Interrupt Flag. If the Watchdog Interrupt is enabled (EIE.4), hardware will set this bit to indicate that the Watchdog Interrupt has occurred. If the interrupt is not enabled, this bit indicates that the time–out has passed. If the Watchdog Reset is enabled (WDCON.1), the user has 512 clocks to strobe the Watchdog prior to a reset. Software or any reset can clear this flag.

WDCON.2 WTRF – Watchdog Timer Reset Flag. Hardware will set this bit when the Watchdog Timer causes a reset. Software can read it, but must clear it manually. A Power–fail Reset will also clear the bit. This bit assists software in determining the cause of a reset. If EWT=0, the Watchdog Timer will have no affect on this bit.

WDCON.1 EWT – Enable Watchdog Timer Reset. Setting this bit will turn on the Watchdog Timer Reset function. The interrupt will not occur unless the EWDI bit in the EIE register is set. A reset will occur according to the WD1 and WD0 bits in the CKCON register. Setting this bit to a 0 will disable the reset but leave the timer running.

WDCON.0 RWT – Reset Watchdog Timer. This bit serves as the strobe for the Watchdog function. During the time–out period, software must set the RWT bit if the Watchdog is enabled. Failing to set the RWT will cause a reset when the

time–out has elapsed. There is no need to set the RWT bit to a 0 because it is self–clearing.

Read/write access: All bits have unrestricted read access. POR, EWT, WDIF, and RWT require a Timed Access write. The remaining bits have unrestricted write access.

CLOCK CONTROL **CKCON; 8Eh**

CKCON.7 WD1 – Watchdog Timer mode select bit 1. See table below for operation.

CKCON.6 WD0 – Watchdog Timer mode select bit 0. See table below for operation.

The WD select bits determine the time–out period of the Watchdog Timer. The timer divides the crystal frequency by a programmable value as shown below. The divider value is expressed in number of clock (crystal) cycles. Note that the reset time–out is 512 clocks longer than the interrupt, regardless of whether the interrupt is enabled.

WD1	WD0	Interrupt Divider	Reset Divider
0	0	2^{17}	$2^{17} + 512$
0	1	2^{20}	$2^{20} + 512$
1	0	2^{23}	$2^{23} + 512$
1	1	2^{26}	$2^{26} + 512$

The default Watchdog time–out is the shortest one (WD1=WD0=0). Software can change this value easily, so this should cause no inconvenience. However, the EWT, WDIF, and RWT bits are protected under the Timed Access procedure. This prevents software from accidentally enabling or disabling the Watchdog. Most importantly, it prevents errant code from accidentally clearing and restarting the Watchdog. More details are discussed in the section on Timed Access.

SECTION 12: SERIAL I/O

The High–Speed Microcontroller serial communication is compatible with the 80C32. This includes framing error detection and automatic address recognition. The High–Speed Microcontroller provides two fully independent UARTs (serial ports) for simultaneous communication over two channels. The UARTs can be operated in identical or different modes and communication speeds. In this documentation, all descriptions apply to both UARTs unless stated otherwise.

Each serial port is capable of both synchronous and asynchronous modes. In the synchronous mode, the microcontroller generates the clock and operates in a half–duplex mode. In the asynchronous mode, full duplex operation is available. Receive data is buffered in a holding register. This allows the UART to receive an incoming word before software has read the previous value. Each UART has an associated control register (SCON0, SCON1) and each has a transmit/receive register (SBUF0, SBUF1). The SFR locations are: SCON0, 98h; SBUF0, 99h; SCON1, C0h; SBUF1, C1h.

The SBUF location provides access to both transmit and receive registers. Reads are directed to the receive buffer and writes to the transmit buffer automatically.

SERIAL MODE SUMMARY

Each port provides four operating modes. These offer different communication protocols and baud rates. These modes are summarized briefly as follows. Detailed descriptions are provided later in this section.

The use of Power management modes, if supported, will affect the internal clock rate and baud rate as shown in Table 7–4. The following descriptions assume that Power Management modes are not in use.

MODE 0

This mode provides synchronous communication with external devices. It is commonly used to communicate with serial peripherals. Serial I/O occurs on the RXD pin. The shift clock is provided on the TXD pin. Note that whether transmitting or receiving, the serial clock is generated by the High–Speed Microcontroller. Thus any device on the serial port in Mode 0 must accept the microcontroller as the master.

The baud rate in Mode 0 is a function of the oscillator input. It will be the clock input divided by either 12 or 4. This is selected by the SM2 bit (SCON0.5 or SCON1.5) as described below. When set to a logic 0, the serial port runs at a divide by 12. When set to a logic 1, the serial port runs at a divide by 4. With the exception of the additional new divide by 4 of the oscillator (supported by SM2), Mode 0 operation is identical to the 80C32.

MODE 1

This mode provides standard full duplex asynchronous communication. A total of 10 bits is transmitted including 1 start bit, 8 data bits, and 1 stop bit. The received stop bit is stored in bit location RB8 in the relevant SCON register.

In Mode 1, the baud rate is a function of timer overflow. This makes the baud rate programmable by the user. Mode 1 has a difference for the two UARTs. Serial Port 0 can use either Timer 1 or 2 to generate baud rates. Serial Port 1 can use only Timer 1. Note that if both serial ports use the same timer, they will be running at the same baud rate. If they use different timers (or different modes), they can run at different rates. Baud rates are discussed in more detail below. Mode 1 operation is identical to the standard 80C32 when Timers 1 or 2 use the default divide by 12 of the oscillator.

MODE 2

This mode is an asynchronous mode that transmits a total of 11 bits. These include 1 start bit, 8 data bits, a programmable ninth bit, and 1 stop bit. The ninth bit is determined by the value in TB8 (SCON0.3 or SCON1.3) for transmission. When the ninth bit is received, it is stored in RB8 (SCON0.2 or SCON1.2). The ninth bit can be a parity value by moving the P bit (PSW.0) to TB8.

The baud rate for Mode 2 is a function of the oscillator frequency. It is either the oscillator input divided by 32 or 64 as programmed by the SMOD bit in the PCON register. Mode 2 operation is identical to the standard 80C3 2.

MODE 3

This mode has the same functionality as Mode 2, but generates baud rates like Mode 1. That is, this mode transmits 11 bits, but generates baud rates via the timers. Like Mode 1, either Timer 1 or 2 can be used for Serial Por t 0 and Timer 1 can be used for Serial Port 1. Mode 3 operation is identical to the standard 80C32 when Timers 1 or 2 use the default divide by 12 of the oscillator.

SERIAL PORT INITIALIZATION

In order to use the UART function(s), the serial port must be initialized. This involves selecting the mode and time base, then initializing the baud rate generator if necessary. Serial communication is then available. Once the baud rate generator is running, the UART can receive data.

In Mode 0, the High–Speed Microcontroller must provide the clock. Serial reception is initiated by setting the RI bit to a logic 0 and REN to a logic 1. This will generate a clock on the TXD pin and shift in the 8 bits on the RXD pin. In the other modes, setting the REN bit to a logic 1 will allow serial reception. The external device must actually initiate it by sending a start bit. In any mode, serial transmission is initiated by writing to either the SBUF0 or SBUF1 location.

Most of the serial port controls are provided by the SCON0 and SCON1 registers. For convenience, these are provided below. In addition, other control bits that influence the Serial Port operation are also summarized below.

SERIAL I/O MODES

MODE	SYNCH/ASYNCH	BAUD CLOCK†	DATA BITS	START/STOP	9TH BIT FUNCTION
0	Synch	4 or 12 t_{CLK}	8	None	None
1	Asynch	Timer 1 or 2*	8	1 start, 1 stop	None
2	Asynch	32 or 64 t_{CLK}	9	1 start, 1 stop	0, 1, parity
3	Asynch	Timer 1 or 2*	9	1 start, 1 stop	0, 1, parity

* Timer 2 available for Serial Port 0 only.
† The use of PMM1 or PMM2 will affect the baud clock.

SERIAL PORT CONTROL ZERO **SCON0; 98h**

This is the standard 80C32 Serial Port. The new Serial Port is designated Serial Port 1 and is documented below.

SCON0.7 SM0/FE_0 – Serial Port 0 Mode bit 0 or Framing Error Flag. PCON.6 (SMOD0) determines whether this bit functions as SM0 or FE. The operation of SM0 (SMOD0=0) is described in the table below. When SMOD0=1, the serial port will set FE to indicate an invalid stop bit. When used as FE, this bit must be cleared in software.

SCON0.6 SM1_0 – Serial Port 0 mode select 1. The operation of SM1 is described in the table below.

SCON0.5 SM2_0 – Multiple MCU communication. Setting this bit to a one enables multiprocessor communication in Modes 2 or 3. If the ninth bit is 0, the RI_0 will not be set. In Mode 1, setting the SM2_0 bit to a one causes the RI_0 bit not to be set if a valid stop bit is not received. In the High–Speed Microcontroller, SM2_0 also has a new function. In mode 0, the SM2_0 bit controls whether the serial port clock runs at a divide by 4 or a divide by 12 of the oscillator when not in PMM. When set to a logic 0, the serial port runs at a divide by 12. When set to a logic one, the serial port runs at a divide by 4. This results in much faster synchronous serial communication.

SCON0.4 REN_0 – Receive Enable. When set to a 1, the receive shift register will be enabled.

SCON0.3 — TB8_0 – Set/clear to define the state of the ninth transmission data bit in modes 2 and 3.

SCON0.2 — RB8_0 – Indicates the state of an incoming ninth bit when in modes 2 and 3. In mode 1, when SM2=0, RB8_0 is the state of the stop bit received. RB8_0 is not used in mode 0.

SCON0.1 — TI_0 – Flag that indicates the transmitted word has been completely shifted out. In mode 0, TI_0 is set at the end of the eighth data bit. In all other modes, this bit is set at the end of the last data bit. It must be cleared manually by software.

SCON0.0 — RI_0 – Flag that indicates a serial word has been received. In mode 0, RI_0 is set at the end of the eighth bit. In mode 1, it is set after the last sample of the incoming stop bit subject to the state of SM2_0. In modes 2 and 3, RI_0 is set after the last sample of RB8_0. It must be cleared manually by software.

SM0/FE_0	SM1_0	Mode	Function	Length	Period
0	0	0	Sync	8 bits	4/12 t_{CLK} (see SM2)
0	1	1	Asynch	10 bits	Timer 1 or 2
1	0	2	Asynch	11 bits	64/32 t_{CLK}
1	1	3	Asynch	11 bits	Timer 1 or 2

Initialization: SCON is set to 00h on a reset.

Read/Write Access: Unrestricted.

SERIAL PORT CONTROL ONE — **SCON1; C0h**

Serial Port 1 performs identically to the standard Serial Port 0 on an 80C32 with one exception. The baud rate generation from Timer 2 is not available in Modes 1 and 3. Timer 1 is used. The port is located at P1.3 and P1.2 for TXD1 and RXD1 respectively.

SCON1.7 — SM0/FE_1 – Serial Port 1 Mode bit 0 or Framing Error Flag. PCON.6 (SMOD0) determines whether this bit functions as SM0 or FE. The operation of SM0 (SMOD0=0) is described in the table below. When SMOD0=1, the serial port will set FE to indicate an invalid stop bit. When used as FE, this bit must be cleared in software.

SCON1.6 — SM1_1 – Serial Port 1 mode select 1. The operation of SM1_1 is described in the table below.

SCON1.5 — SM2_1 – Multiple MCU communication. Setting this bit to a one enables multiprocessor communication in Modes 2 or 3. If the ninth bit is 0, the RI_1 will not be set. In Mode 1, setting the SM2_1 bit to a one causes the RI_1 bit not to be set if a valid stop bit is not received. In the High–Speed Microcontroller, SM2_1 also has a new function. In mode 0, the SM2_1 bit controls whether the serial port clock runs at a divide by 4 or a divide by 12 of the oscillator when not in PMM. When set to a logic 0, the serial port runs at a divide by 12. When set to a logic one, the serial port runs at a divide by 4. This results in much faster synchronous serial communication.

SCON1.4 REN_1 – Receive Enable. When set to a 1, the receive shift register will be enabled.

SCON1.3 TB8_1 – Set/clear to define the state of the ninth transmission data bit in modes 2 and 3.

SCON1.2 RB8_1 – Indicates the state of an incoming ninth bit when in modes 2 and 3. In mode 1, when SM2=0, RB8 is the state of the stop bit received. RB8 is not used in mode 0.

SCON1.1 TI_1 – Flag that indicates the transmitted word has been completely shifted out. In mode 0, TI is set at the end of the eighth data bit. In all other modes, this bit is set at the end of the last data bit. It must be cleared manually by software.

SCON1.0 RI_1 – Flag that indicates a serial word has been received. In mode 0, RI_1 is set at the end of the eighth bit. In mode 1, it is set after the last sample of the incoming stop bit subject to the state of SM2_1. In modes 2 and 3, RI_1 is set after the last sample of RB8_1. It must be cleared manually by software.

SM0	SM1	Mode	Function	Length	Period
0	0	0	Sync	8 bits	4/12 t_{CLK} (see SM2)
0	1	1	Asynch	10 bits	Timer 1
1	0	2	Asynch	11 bits	64/32 t_{CLK}
1	1	3	Asynch	11 bits	Timer 1

Initialization: SCON1 is set to 00h on a reset.

Read/Write Access: Unrestricted.

POWER CONTROL **PCON; 87h**

PCON.7 SMOD_0. Doubles the serial baud rate in modes 1, 2, and 3 for Serial Port 0 (the standard port) when SMOD=1.

PCON.6 SMOD0 – Framing Error Detection Enable. When SMOD0 is set to a one, SCON0.7 and SCON1.7 are converted to the FE flag for the respective serial port. When SMOD0 is 0, then SCON0.7 and SCON1.7 are the SM0 function as defined for the serial port.

WATCHDOG CONTROL **WDCON; D8h**

WDCON.7 SMOD_1 – Serial Modification. When set to a logic 1, this bit doubles the baud rate of Serial Port 1. It works identically to PCON.7.

TIMER TWO CONTROL **T2CON; C8h**

T2CON.5 RCLK – Receive Clock Flag. This bit determines whether Timer 1 or 2 is used for Serial Port 0 timing of received data in Serial Modes 1 or 3. RCLK =1 causes Timer 2 overflow to be used as the receive clock. RCLK=0 causes Timer 1 overflow to be used as the receive clock.

T2CON.4 TCLK – Transmit Clock Flag. This bit determines whether Timer 1 or 2 is used for Serial Port 0 timing of Transmit data in Serial Modes 1 or 3. TCLK=1 causes Timer 2 overflow to be used as the transmit clock. TCLK=0 causes Timer 1 overflow to be used as the transmit clock.

BAUD RATES

Each mode has a baud rate generator associated with it. This generator is generally the same for each UART. Several of the baud rate generation techniques have options and these options are independent for the two UARTs. The baud rate descriptions given below are separated by mode.

Mode 0

Baud rates for this mode are driven directly from the crystal speed divided by either 12 or 4. Mode 0 is synchronous so that the shift clock output frequency will be the baud rate. The formula is simply as follows:

$$\text{Mode 0 baud rate} = \frac{\text{Oscillator Frequency}}{12}$$

or

$$\text{Mode 0 baud rate} = \frac{\text{Oscillator Frequency}}{4}$$

The default case is divide by 12. The user can select the divider using the SM2 bit in the associated SCON register. For Serial Port 0, the SM2_0 bit is SCON0.5. For Serial Port 1, the SM2_0 bit is SCON1.5. When SM2 is set to a logic 0, the baud rate is generated using a divide by 12 of the oscillator input. When SM2 is set to a logic 1, the baud rate is generated using divide by 4. Note that this use of SM2 differs from a standard 80C32. In that device, SM2 had no valid use when the UART was in Mode 0. Since it was generally set to a 0, for the divide by 12, there is no compatibility problem.

Mode 2

In this asynchronous mode, baud rates are also generated from the oscillator input. This mode works identically to the original 8051 family. The baud rate is given by the following formula.

$$\text{Mode 2 baud rate} = \frac{2^{\text{SMOD_x}}}{64} * \text{Oscillator Frequency}$$

The result of this formula generates a baud rate of either 1/32 * oscillator frequency or 1/64 * oscillator frequency. In the formula, the numerator is expressed as two to the power of SMOD, where SMOD is either a 0 or 1. When SMOD=0, the numerator is a 1 and when SMOD=1, the numerator is a 2.

SMOD is a bit that effectively doubles the baud rate when set to a logic 1. For Serial Port 0, SMOD_0 resides at PCON.7. This is the original location in the 8051 family. For Serial port 1, SMOD_1 resides in WDCON.7. The SMOD bits are set to a logic 0 on reset, which gives the lower speed baud rate.

If the application determines that Mode 0 or 2 must be used, then the oscillator or crystal frequency must be selected to generate the correct baud rates since each mode offers two selections for a given frequency.

Mode 1 or 3

These asynchronous modes are commonly used for communication with PCs, modems, and other similar interfaces. The baud rates are programmable using the oscillator input and 16–bit Timer 2 or 8–bit Timer 1. The respective timer is placed in auto–reload mode. Each time the timer reaches its rollover condition (FFFFh – Timer 2 or FFh – Timer 1), a clock is sent to the baud rate circuit. This clock is then divided by 16 to generate the exact baud rate. For Serial Port 0, either Timer 1 or 2 can be used to generate baud rates. Note that there are differences between the timers when used as baud rate generators. Serial Port 1 can use Timer 1 as a baud rate generator. Thus in Mode 1 or 3, the two serial ports can run at the same frequency if Timer 1 is used for both, but different frequencies if both timers are used.

Also note that the user can determine the speed at which Timer 1 runs (4 clocks or 12 clocks). In most cases, 12 clocks will be used for baud rate generation. Timer 2 runs from a 2 clock scheme when used for baud rate generation. This is compatible with the 80C32.

The baud rates for Mode 1 or 3 are given by these formulas.

Serial Port 0 or 1

$$\text{Mode 1, 3 baud rate} = \frac{2^{\text{SMOD_x}}}{32} * \text{Timer 1 Overflow}$$

Serial Port 0

$$\text{Mode 1, 3 baud rate} = \frac{\text{Timer 2 Overflow}}{16}$$

To use Timer 1 as the baud rate generator, it is commonly put into the 8–bit auto–reload mode. In this way,

the CPU is not involved in baud rate generation. Note that the timer interrupt should not be enabled. In the 8–bit auto–reload mode (Timer 1 Mode 2), the reload value is stored in TH1. Thus the combination of crystal frequency and TH1 determine the baud rate. The complete formula is as follows.

$$\text{Mode 1, 3 baud rate} = \frac{2^{SMOD_x}}{32} * \frac{\text{Oscillator Frequency}}{12 * (256 - TH1)}$$

Note that the 12 in the denominator can be changed to a 4 as determined by the Timer selection (T1M; CKCON.4). This formula provides the derived baud rate for a given TH1 and crystal. Most users already know what baud rate is desired and want the timer reload value. Thus the equation solves as follows, when T1M=0.

$$TH1 = 256 - \frac{2^{SMOD_x} * \text{Oscillator Frequency}}{384 * \text{Baud Rate}}$$

Note that the most common application is to use Timer 1 in 8–bit auto–reload mode as a timer. It can actually be used in any mode and can also be configured as a counter.

To use Timer 2 as baud rate generator for Serial Port 0, the Timer is configured in auto–reload mode. Then either TCLK or RCLK bit (or both) are set to a logic 1. TCLK=1 selects Timer 2 as the baud rate generator for the transmitter and RCLK=1 selects Timer 2 for the receiver. Thus Serial Port 0 can have the transmit and receive operating at different baud rates by choosing Timer 1 for one data direction and Timer 2 for the other. Setting either RCLK or TCLK to a logic 1 selects Timer 2 for baud rate generation. RCLK and TCLK reside in T2CON.4 and TCON.5 respectively.

When using Timer 2 to generate baud rates, the formula will be as follows. Note that the reload value is a 16–bit number as compared with Timer 1, which uses only 8 bits.

$$\text{Mode 1, 3 baud rate} = \frac{\text{Oscillator Frequency}}{32 * (65536 - RCAP2H, RCAP2L)}$$

Note that the 32 in the denominator is a result of the timer being run at a divide by 2, combined with the divide by 16 applied to timer overflows as mentioned above. Timer 2 normally runs at a divide by either 12 or 4 in auto–reload mode. Setting RCLK or TCLK causes the divide by 2 operation.

This formula provides the derived baud rate for a given RCAP2H, RCAP2L and crystal. Most users already know what baud rate is desired and want the timer reload value. Thus the equation solves as follows.

$$RCAP2H,\ RCAP2L = 65536 - \frac{\text{Oscillator Frequency}}{32 * \text{Baud Rate}}$$

The Timer 2 interrupt is automatically disabled when either RCLK or TCLK is set. Also, the TF2 (TCON.7) flag will not be set on a timer rollover. The manual reload pin, T2EX (P1.1), will not cause a reload either.

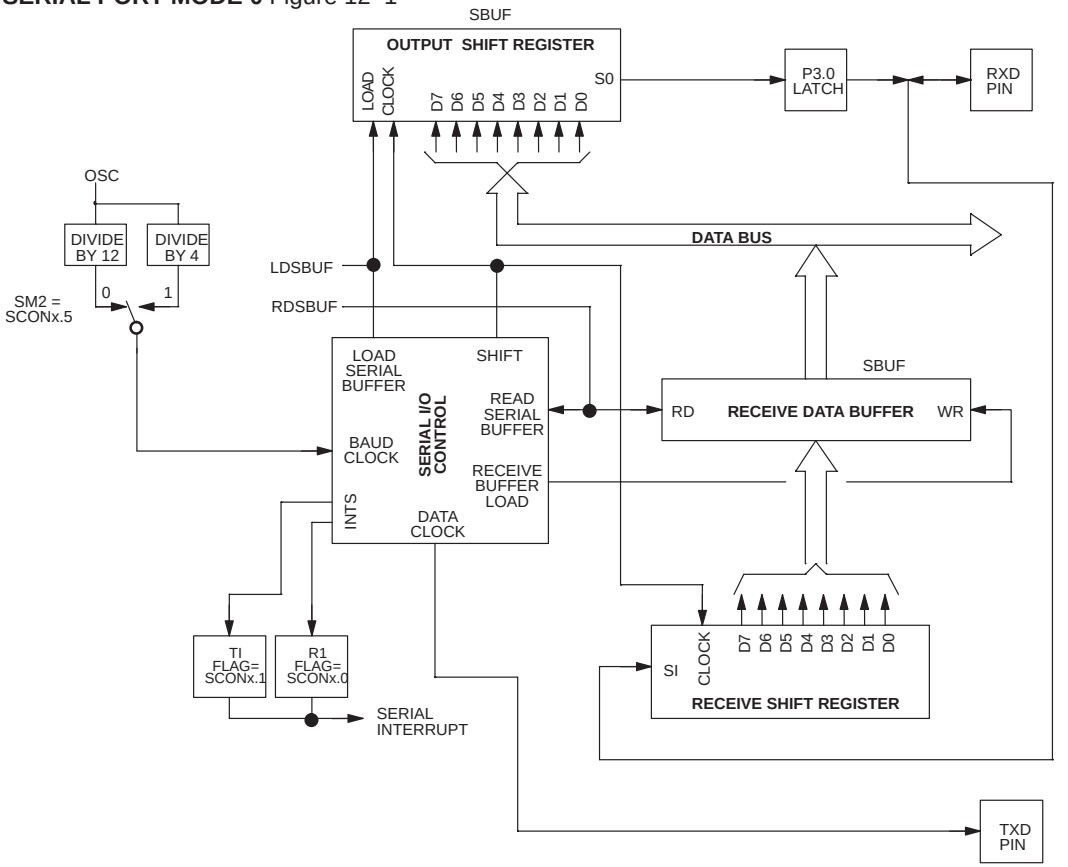
SERIAL I/O DESCRIPTION

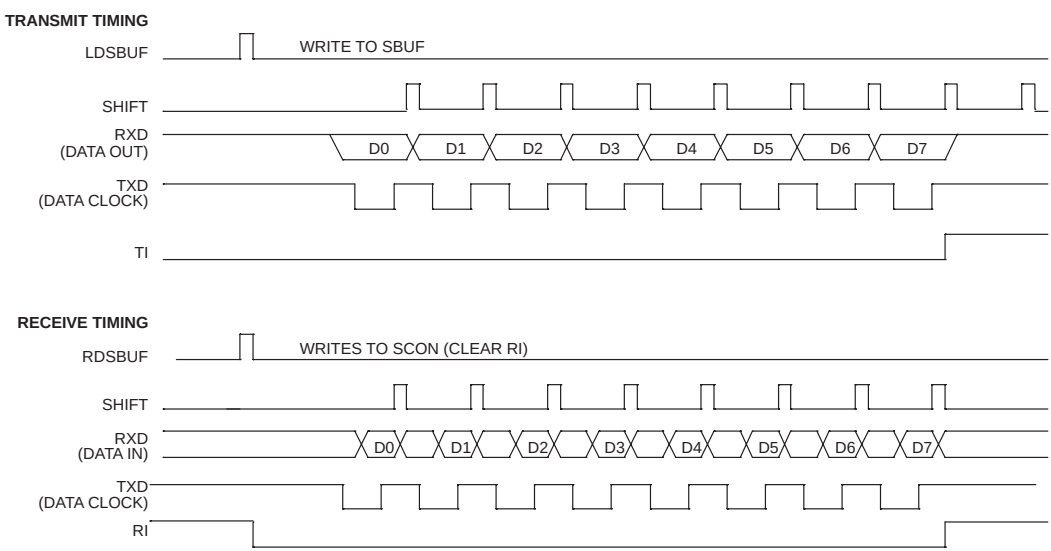
A detailed description of each serial mode is given below. A description of framing error detection and multiprocessor communication follows this section.

Mode 0

This mode is used to communicate in synchronous, half–duplex format with devices that accept the High–Speed Microcontroller as a master. A functional block diagram and basic timing of this mode are shown in Figure 12–1. As can be seen, there is one bidirectional data line (RXD) and one shift clock line (TXD) used for communication. The shift clock is used to shift data into and out of the microcontroller and the remote device. Mode 0 requires that the microcontroller is the master because the microcontroller generates the serial shift clocks for both directions. As described above, the shift clock may be selected to be either divide by 12 or divide by 4 of the oscillator as determined by the SM2 (SCON0.5 or SCON1.5) bit.

SERIAL PORT MODE 0 Figure 12–1





The RXD signal is used for both transmission and reception. TXD provides the shift clock. Data bits enter and exit LSb first. The baud rate is equal to the shift clock frequency. This can be either oscillator divided by 4 or oscillator divided by 12. The relevant UART will begin transmitting when any instruction writes to SBUF0 or SBUF1 (hex address 99 or C1). The internal shift register will then begin to shift data out. The clock will be activated and will transfer data until the 8–bit value is complete. Data will be presented one oscillator cycle prior to the falling edge of the shift clock (TXD), and an external device can latch the data using the rising edge.

The UART will begin to receive data when the REN bit in the SCON register (SCON0.4 or SCON1.4) is set to a logic 1 and the RI bit (SCON0.0 SCON1.0) is set to a logic 0. This condition tells the UART that there is data to be shifted in. The shift clock (TXD) will activate and the UART will latch incoming data on the rising edge. The external device should therefore present data on the falling edge. This process will continue until 8 bits have been received. The RI bit will automatically be set to a logic 1, one machine cycle following the last rising edge of the shift clock on TXD. This will cause reception to stop until the SBUF has been read, and the RI bit cleared. When RI is cleared, another byte will be shifted in.

Mode 1

This mode is asynchronous, full duplex, using a total of 10 bits. The 10 bits consist of a start bit (logic 0), 8 data bits, and 1 stop bit (logic 1) as illustrated in Figure 12–2. The data is transferred LSb first. As described above, the baud rates for Mode 1 are generated by either a divide by 16 of Timer 1 rollover, a divide by 16 of the Timer 2 rollover, or a divide by 16 of (Timer 1 rollover)/2. The UART will begin transmission 5 oscillator cycles after the first rollover of the divide by 16 counter following a software write to SBUF. Transmission takes place on the TXD pin. It begins by the start bit being placed on the pin. Data is then shifted out onto the pin, LSb first. The stop bit follows. The TI bit is set two oscillator cycles after the stop bit is placed on the pin. All bits are shifted out at the rate determined by the baud rate generator.

Once the baud rate generator is active, reception can begin at any time. The REN bit (SCON0.4 or SCON1.4) must be set to a logic 1 to allow reception. The falling edge of a start bit on the RXD pin will begin the reception process. Data will be shifted in at the selected baud rate. At the middle of the stop bit time, certain conditions must be met to load SBUF with the received data:

1. RI must=0, and either
2. If SM2=0, the state of the stop bit doesn't matter, or
3. If SM2=1, the state of the stop bit must=1.

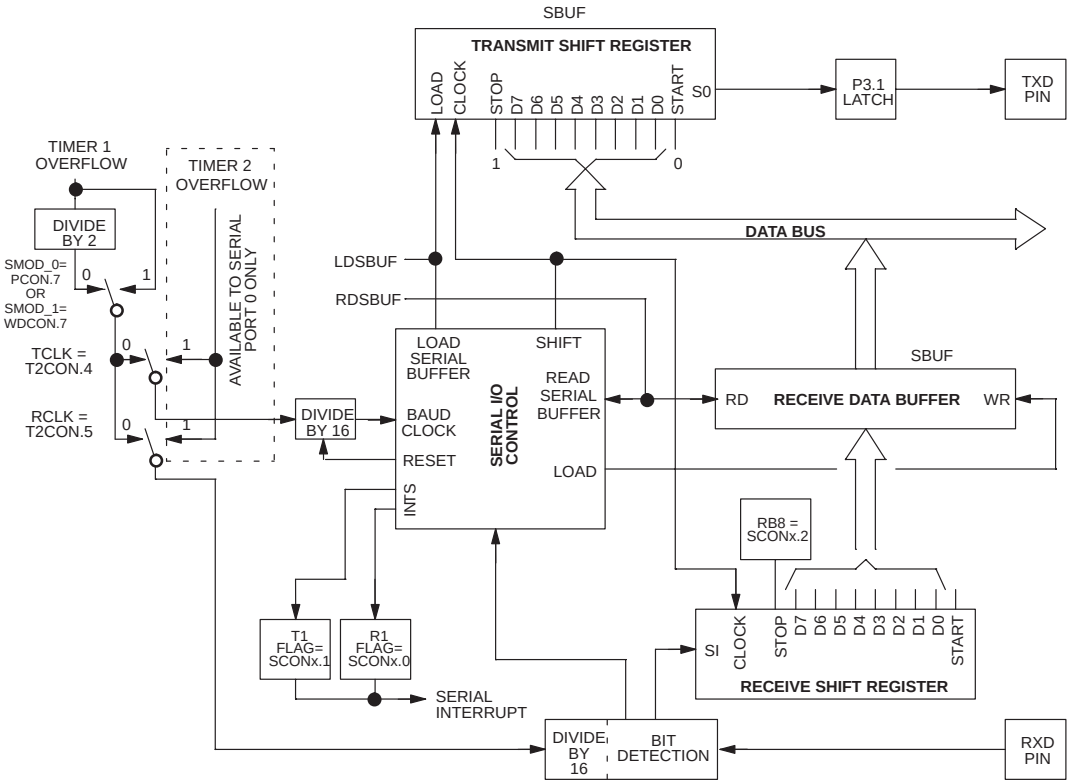
If these conditions are true, then SBUF (hex address 99h or C1h) will be loaded with the received byte, the RB8 bit (SCON0.2 or SCON1.2) will be loaded with the stop bit, and the RI bit (SCON0.0 or SCON1.0) will be set. If these conditions are false, then the received data will be lost (SBUF and RB8 not loaded) and RI will not be set. Regardless of the receive word status, after the middle of the stop bit time, the receiver will go back to looking for a 1 to 0 transition on the RXD pin.

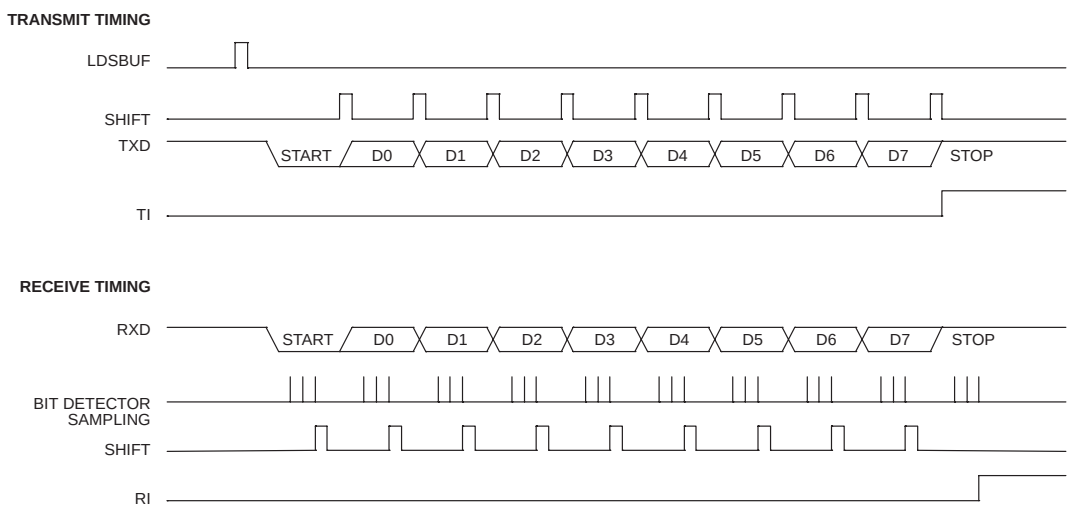
Each data bit received is sampled on the 7th, 8th and 9th clock used by the divide by 16 counter. Using majority voting, two equal samples out of the three, determines the logic level for each received bit. If the start bit was determined to be invalid (=1), then the receiver goes back to looking for a 1 to 0 transition on the RXD pin in order to start the reception of data.

Mode 2

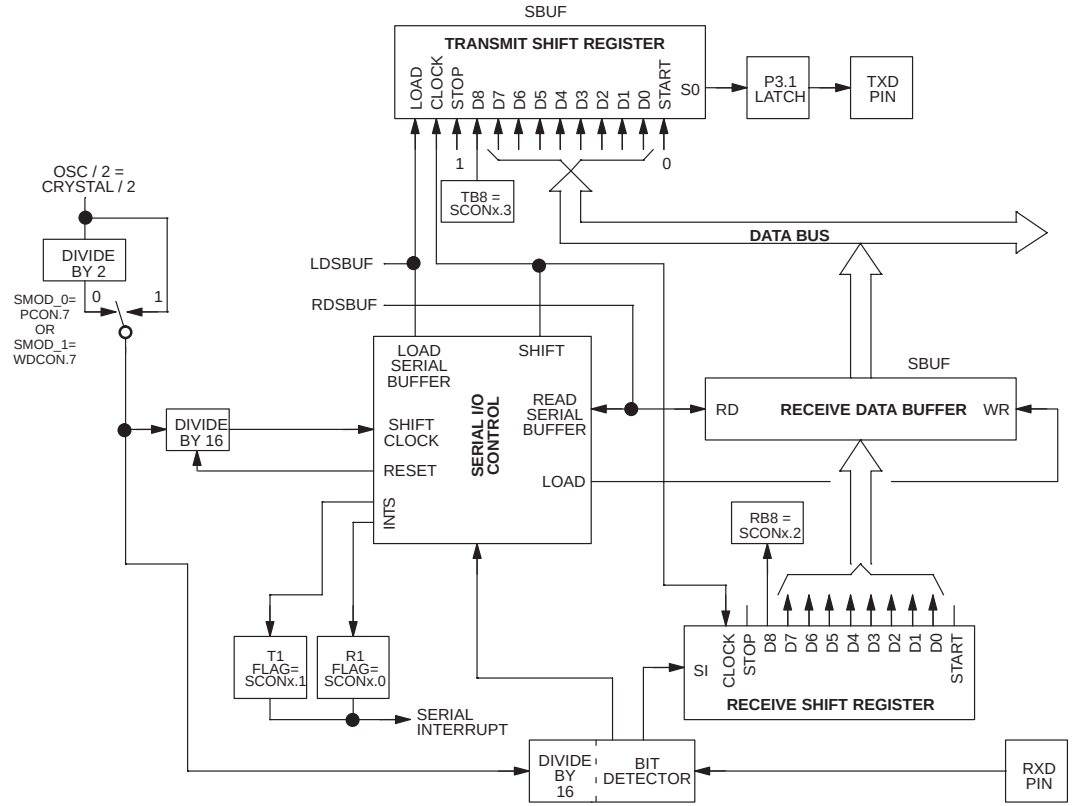
This mode uses a total of 11 bits in asynchronous full duplex communication as illustrated in Figure 12–3. The 11 bits consist of one start bit (a logic 0), 8 data bits, a programmable 9th bit, and one stop bit (a logic 1). Like Mode 1, the transmissions occur on the TXD signal pin and receptions on RXD. For transmission purposes, the 9th bit can be stuffed as a logic 0 or 1. A common use is to put the parity bit in this location. The 9th bit is transferred from the TB8 bit position in the SCON register (SCON0.3 or SCON1.3) during the write to SBUF. Baud rates are generated as a fixed function of the crystal frequency as described above. Like Mode 1, Mode 2's transmission begins 5 oscillator cycles after the first rollover of the divide by 16 counter following a software write to SBUF. It begins by the start bit being placed on the TXD pin. The data is then shifted out onto the pin LSb first, followed by the 9th bit, and finally the stop bit. The TI bit (SCON0.1 or SCON1.1) is set when the stop bit is placed on the pin.

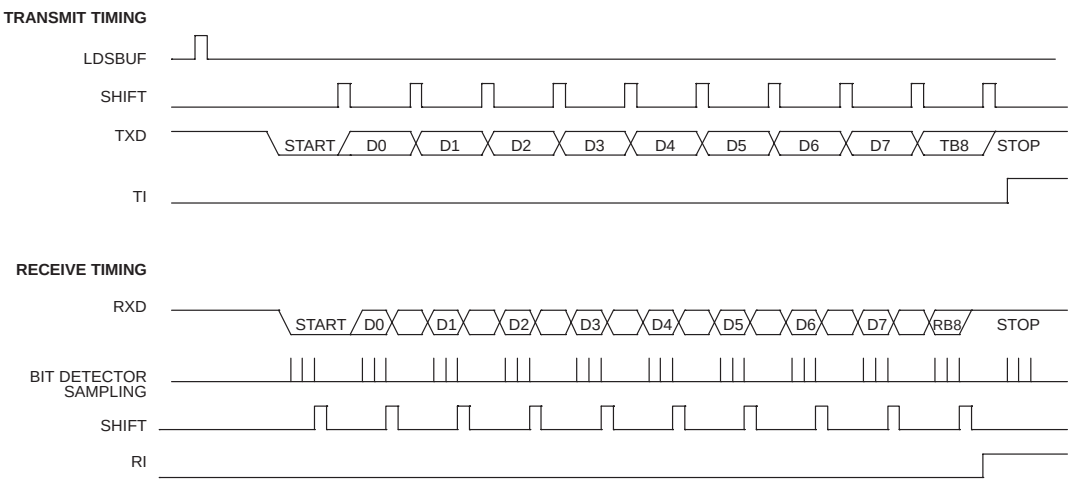
SERIAL PORT MODE 1 Figure 12–2





SERIAL PORT MODE 2 Figure 12–3





Reception begins when a falling edge is detected as part of the incoming start bit on the RXD pin. The RXD pin is then sampled according to the baud rate speed. The 9th bit is placed in the RB8 bit location in SCON (SCON0.2 or SCON1.2). When a stop bit has been received, the data value will be transferred to the SBUF receive register (hex address 99 or C1). The RI bit (SCON0.0 or SCON1.0) will be set to indicate that a byte has been received. At this time, the UART can receive another byte.

Once the baud rate generator is active, reception can begin at any time. The REN bit (SCON0.4 or SCON1.4) must be set to a logic 1 to allow reception. The falling edge of a start bit on the RXD pin will begin the reception process. Data must be shifted in at the selected baud rate. At the middle of the 9th bit time, certain conditions must be met to load SBUF with the received data.

1. RI must=0, and either
2. If SM2=0, the state of the 9th bit doesn't matter, or
3. If SM2=1, the state of the 9th bit must=1.

If these conditions are true, then SBUF will be loaded with the received byte, RB8 will be loaded with the 9th bit, and RI will be set. If these conditions are false, then the received data will be lost (SBUF and RB8 not loaded) and RI will not be set. Regardless of the receive word status, after the middle of the stop bit time, the receiver will go back to looking for a 1 to 0 transition on RXD.

Data is sampled in a similar fashion to Mode 1 with the majority voting on three consecutive samples. Mode 2 uses the sample divide by 16 counter with either the oscillator divided by 2 or 4.

Mode3

This mode has the same operation as Mode 2, except for the baud rate source. As shown in Figure 12–4, Mode 3 can use Timer 1 or 2 for Serial Port 0 and Timer 1 for Serial Port 1. The bit shifting and protocol are the same.

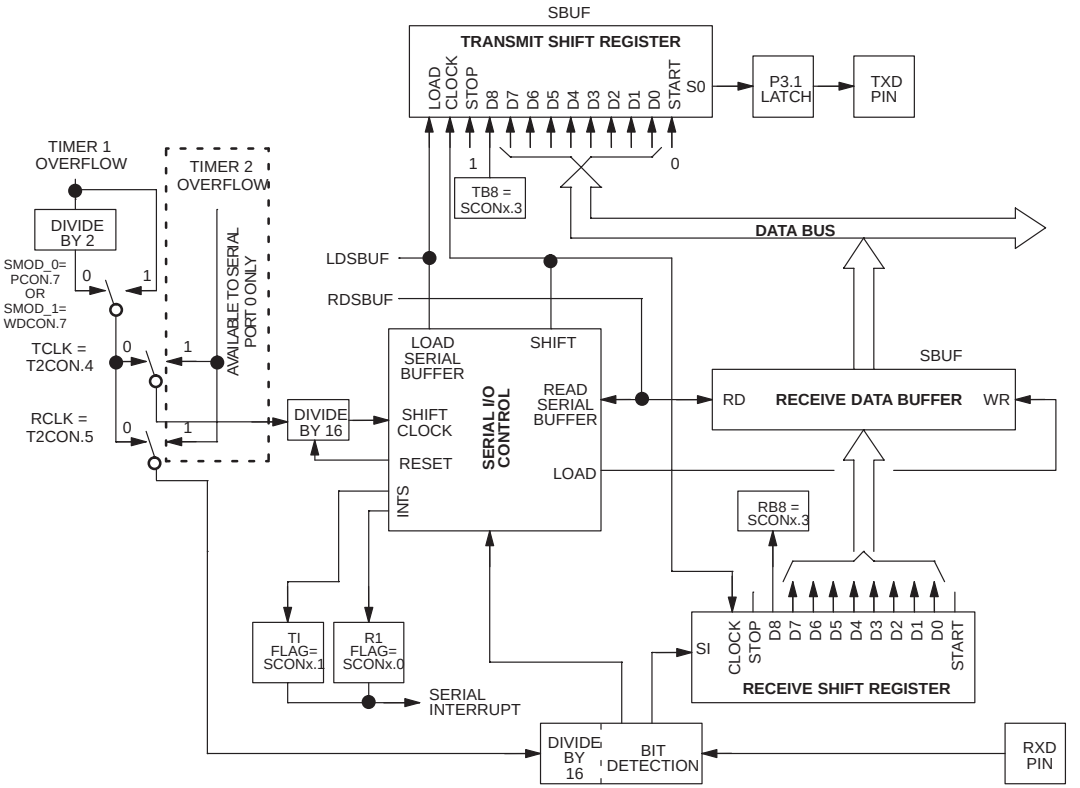
FRAMING ERROR DETECTION

A framing error occurs when a valid stop bit is not detected. This results in the possible improper reception of the serial word. The UART can detect a framing error and notify the software. Typical causes of framing errors are noise and contention. The Framing Error condition is reported in the SCON register for the corresponding UART.

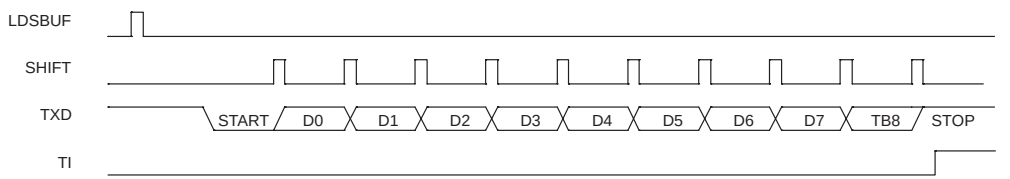
The Framing Error bit, FE, is located in SCON0.7 or SCON1.7. Note that this bit normally serves as SM0 and is described as SM0/FE_0 or SM0/FE_1 in the register description. Framing Error information is made accessible by the Framing Error Detection Enable bit. It is SMOD0 located at PCON.6. When SMOD0 is set to a logic 1, the framing error information is shown in SM0/FE (SCON0.7 or SCON1.7). When SMOD0 is set to a logic 0, the SM0 function is accessible. The information for bits SM0 and FE is actually stored in different registers. Changing SMOD0 only changes which register is accessed; not the contents of either.

The FE bit will be set to a 1 when a framing error occurs. It must be cleared by software. Note that the SMOD0 state must be 1 while reading or writing the FE bit. Also note that receiving a properly framed serial word will not clear the FE bit. This must be done in software.

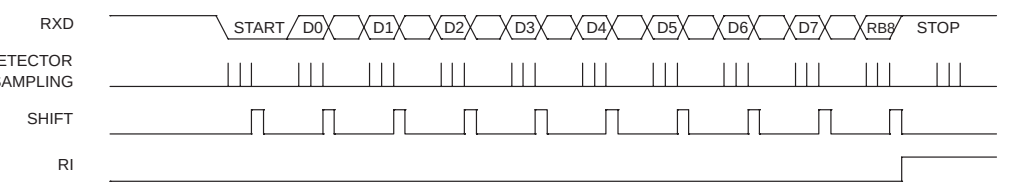
SERIAL PORT MODE 3 Figure 12–4



TRANSMIT TIMING



RECEIVE TIMING



MULTIPROCESSOR COMMUNICATION

Multiprocessor communication mode makes special use of the 9th data bit in Modes 2 and 3. In the original 8051, the 9th bit was restricted to a 0 or 1 condition, but had no special purpose. In the 80C32 and the High–Speed Microcontroller, it can be used to signify that the incoming byte is an address. This allows the processor to be interrupted only if the correct address appears. The Receive Interrupt, if enabled, will only occur when a recognized address is received.

When a serial word is received with the 9th bit set, the byte will be assumed to be an address. The address will be compared to an internally stored address. If it matches, a receive interrupt will occur. The internal address is derived from the contents of two registers. The first register specifies an absolute address. This is the user specified address of the device. The second register tells the comparator which address bit(s) to actually use in the comparison. This allows broadcast transmissions that reach groups of microcontrollers or all microcontrollers on a serial port. The user defines this protocol.

There are two Special Function Registers that support multiprocessor communication for each UART. These are independent, so that different addresses can be used in each. The registers are SADDR0 or SADDR1 (hex address A9 or A10) and SADEN0 or SADEN1 (hex address B9 or B10). The SADDR register specifies the individual processor's address. The SADEN identifies address bits that should be ignored in matching addresses.

Software will write an 8–bit address to the SADDR register. This is the microcontroller's individual address. Any bit in SADEN that contains a logic 0 will cause the corresponding bit in SADDR to be ignored in comparison. Thus logic 0 bits in SADEN create don't care bit states for address comparisons.

When an address is received, each address bit that is not masked by a don't care will be compared to the SADDR. The microcontroller will interrupt on any address that matches this comparison. Any address that meets this comparison is called a Given Address.

The following example shows how one address can be directed to an individual processor, or two out of three.

```
Micro 1

     SADDR             11110000
     SADEN             11111010
     -------------------------
     Given             11110x0x

Micro 2

     SADDR             11110001
     SADEN             11111001
     -------------------------
     Given             11110xx1

Micro 3

     SADDR             11110010
     SADEN             11111010
     -------------------------
     Given             11110x1x
```

Note that an address of 11110000 will reach only microcontroller 1. An address of 11110001 will reach both microcontroller 1 and microcontroller 2. An address of 11110010 will reach only microcontroller 3.

The microcontroller will also match on any address that corresponds to the Broadcast Address. This is the logical OR of the SADDR and SADEN registers, with any 0s defined as don't cares. In most cases, the Broadcast Address will be FFh.

The multiprocessor communication is always enabled. However, the SADEN registers default to 00h, which means all address bits are don't care, so all match. Thus if no multiprocessor communication is used, these registers can be ignored.

The Timer 2 interrupt is automatically disabled when either RCLK or TCLK is set. Also, the TF2 (TCON.7) flag will not be set on a timer rollover. The manual reload pin, T2EX (P1.1), will not cause a reload either.

SECTION 13: TIMED ACCESS PROTECTION

The High–Speed Microcontroller uses a protection feature called Timed Access to prevent accidental writes to critical SFR bits. These bits could cause a system failure or prevent the Watchdog Timer from doing its job if improperly written. The Timed Access involves opening a timing window during which the protected bit can be modified. If the window is opened correctly, it remains open long enough to alter one protected bit. This section explains which bits are protected, why, and how to use the Timed Access feature.

PROTECTED BITS

Bits which are protected by the Timed Access feature are shown below. Only critical function bits which are unique to the High–Speed Microcontroller family are protected, assuring code compatibility with the original 80C51 or 80C52. A full description of the function of each bit is provided in Section 4.

EXIF.0	BGS	Band–gap Select
WDCON.6	POR	Power–on Reset Flag
WDCON.1	EWT	Watchdog Reset Enable
WDCON.0	RWT	Reset Watchdog Timer
WDCON.3	WDIF	Watchdog Interrupt Flag
TRIM.7	E4K	4096 Hz RTC Output
TRIM.6	X12/$\overline{6}$	12pF/6pF Crystal Select
TRIM.5	TRM2	Capacitance Trim Bit 2
TRIM.4	$\overline{\text{TRM2}}$	Inverse Capacitance Trim Bit 2
TRIM.3	TRM1	Capacitance Trim Bit 1
TRIM.2	$\overline{\text{TRM1}}$	Inverse Capacitance Trim Bit 1
TRIM.1	TRM0	Capacitance Trim Bit 0
TRIM.0	$\overline{\text{TRM0}}$	Inverse Capacitance Trim Bit 0
ROMSIZE.2	RMS2	ROM Size Select Bit 2
ROMSIZE.1	RMS1	ROM Size Select Bit 1
ROMSIZE.0	RMS0	ROM Size Select Bit 0
RTCC.2	RTCWE	RTC Write Enable
RTCC.0	RTCE	RTC Enable

PROTECTION SCHEME

Each bit mentioned above is protected against an accidental write by requiring the software to perform a procedure before writing the bit. Timed Access requires the software to write two specific values to the Timed Access register during two consecutive instruction cycles. The values AAh, then 55h, must be written in consecutive instructions to the TA register at SFR location C7h. If the writes are performed correctly, the write access window will open for three machine cycles. During this window, the software may modify a protected bit. The suggested code to open a Timed Access window is:

```
MOV    0C7h, #0AAh
MOV    0C7h, #55h
```

The procedure to modify a Timed Access protected bit begins by writing the value AAh to the Time Access register (TA;C7h). The value 55h must then be written to the Timed Access register within three machine cycles of writing AAh. This opens a three machine cycle window, after the write of 55h, during which any Timed Access protected bits may be modified. Failure to complete any of the required steps will also require the procedure to begin again, starting with the write of AAh to the Timed Access register. Attempts to modify Timed Access protected bits after the window has closed will be ignored. This is regardless of whether any bits were modified. Figure 13–1 illustrates a number of examples of correct and incorrect use of the Timed Access procedure.

TIMED ACCESS EXAMPLES Figure 13–1

three machine cycles	three machine cycles	two machine cycles
MOV 0C7h, #0AAh	MOV 0C7h, #55h	SETB EWT

three machine cycles	three machine cycles	one machine cycle	two machine cycles
MOV 0C7h, #0AAh	MOV 0C7h, #55h	NOP	SETB EWT

three machine cycles	three machine cycles	three machine cycles
MOV 0C7h, #0AAh	MOV 0C7h, #55h	MOV WDCON, #02h

VALID TIMED ACCESS PROCEDURES

three machine cycles	one machine cycle	three machine cycles	two machine cycles
MOV 0C7h, #0AAh	NOP	MOV 0C7h, #55h	SETB EWT

*Second write to TA register does not occur within 3 cycles of first write.

three machine cycles	three machine cycles	one machine cycle	three machine cycles
MOV 0C7h, #0AAh	MOV 0C7h, #55h	NOP	MOV WDCON, #02h

*Modification of protected bit did not occur with 3 cycles of second write to TA register.

three machine cycles	three machine cycles	two machine cycles	two machine cycles
MOV 0C7h, #0AAh	MOV 0C7h, #55h	SETB EWT	SETB RWT

*Modification of second protected bit did not complete within 3 cycles of second write to TA register.

INVALID TIMED ACCESS PROCEDURES

TIMED ACCESS PROTECTS WATCHDOG

Any microcontroller–based system can be faced with environmental conditions that are beyond its designed abilities. These include external signal transients due to component failure, fluctuating power conditions, massive electrostatic discharge (ESD), and other unexpected system events. When a microcontroller is exposed to such conditions, program execution can become corrupted. Members of the High–Speed Microcontroller family which incorporate a Watchdog Timer can initiate a reset to recover from these conditions. The primary function of the Timed Access feature is to protect against accidental disabling of the watchdog timer by an "out–of–control" device. This will allow the watchdog timer to reset the system in the event of program execution failure.

The following hypothetical example demonstrates how a single bit change can corrupt program execution. The Timed Access procedure protects against an accidental write to the EWT bit by the errant code, allowing the watchdog timer reset function to reset the device. While this is a purely fictitious example, it illustrates how the watchdog timer and Timed Access feature make the High–Speed Microcontroller minimize the effect of accidental code corruption. *Note: Timed Access is not optional and must be supported if the protected bits are used. This example simply helps explain the category of problem that the Timed Access prevents.*

EXAMPLE: A TRANSIENT CAUSES THE WATCHDOG TO BE DISABLED

```
        TABLE_READ:
C2D2    90 0A 00        MOV     DPTR, 0A00H     ;LOAD TABLE POINTER
C2D5    79 FF           MOV     R1, #0FFH       ;LOAD COUNTER
C2D7    78 90           MOV     R0, #90H        ;DESTINATION POINTER

                        LOOP:
C2D9    E0              MOVX    A, @DPTR        ;READ DATA BYTE
C2DA    F6              MOV     @R0, A          ;STORE IT IN RAM
C2DB    06              INC     R0              ;NEXT TABLE LOCATION
C2DC    A3              INC     DPTR            ;NEXT DATA VALUE
C2DD    D9 C2 D9        DJNZ    R1, LOOP        ;NEXT BYTE OR DONE ?
```

A transient occurs while the opcode is being fetched for the first instruction. The transient causes one bit of the opcode in the first instruction to be read as a 0 instead of 1. The resulting program is what the microcontroller would actually execute:

```
        TABLE_READ:

C2D2    80 0A 00        SJMP    0BH             ;RELATIVE JUMP BY 10 LOCATIONS
C2D5    79 FF           MOV     R1, #0FFH       ;LOAD COUNTER
C2D7    78 90           MOV     R0, #90H        ;DESTINATION POINTER

                        LOOP:
C2D9    E0              MOVX    A, @DPTR        ;READ DATA BYTE
C2DA    F6              MOV     @R0, A          ;STORE IT IN RAM
C2DB    06              INC     R0              ;NEXT TABLE LOCATION
C2DC    A3              INC     DPTR            ;NEXT DATA VALUE
C2DD    D9 C2 D9        DJNZ    R1, LOOP        ;NEXT BYTE OR DONE ?
```

The resulting jump is to address C2DE. This is not even a real opcode, but would be treated as such. The resulting fetch is the value C2 D9. This is the opcode for CLR D9h. The bit addressable location D9h corresponds to the EWT – Enable Watchdog Timer. If the Timed Access procedure did not prevent it, this errant instruction would disable the Watchdog. Note that now, the program execution is completely lost. Real opcodes are being replaced by operands, data and garbage. In the High–Speed Microcontroller, the Watchdog will recover from this state as soon as it times out since it could not have been disabled in this way.

In the High–Speed Microcontroller it is very hard to contrive a situation that will accidentally disable the Watchdog. Note, the Timed Access prevents accidentally writing a bit. It can not prevent accidentally calling the correct code that writes a bit. This is much more unlikely however.

SECTION 14: REAL–TIME CLOCK

The DS87C530 incorporates a real–time clock (RTC) onto the High–Speed Microcontroller family core. This allows the device to perform real–time related functions such as data logging and time–stamping without an external timer. In addition, the RTC includes an alarm function which can execute a software interrupt or resume operation from Stop mode at a specified time. The RTC features are controlled by 12 new SFRs. These registers, as well as two new interrupt control bits are shown in Table 14–1.

REAL–TIME CLOCK CONTROL AND STATUS BIT SUMMARY Table 14–1

BIT NAME	LOCATION	FUNCTION	RANGE	RESET	READ/WRITE ACCESS
ERTCI	EIE.5	RTC Interrupt Enable		0	Unrestricted
PRTCI	EIP.5	RTC Interrupt Priority		0	Unrestricted
RTASS.7–0	RTASS	RTC Alarm Subsecond	0–FFh	Unchanged	Unrestricted
RTAS.5–0	RTAS	RTC Alarm Second	0–3Bh	Unchanged	Unrestricted
RTAM.5–0	RTAM	RTC Alarm Minute	0–3Bh	Unchanged	Unrestricted
RTAH.4–0	RTAH	RTC Alarm Hour	0–17H	Unchanged	Unrestricted
RTCSS.7–0	RTCSS	RTC Subsecond	0–FFh	Unchanged	Read: only if RTCRE=1. Cannot be written. Cleared when RTCWE 1–>0
RTCS.5–0	RTCS	RTC Second	0–3Bh	Unchanged	Read: only if RTCRE=1. Write: only if RTCWE=1. 1.95 ms Read/Write window
RTCM.5–0	RTCM	RTC Minute	0–3Bh	Unchanged	
RTCH.4–0	RTCH.4–0	RTC Hour	0–17h	Unchanged	
DOW.2–0	RTCH.7–5	RTC Day of Week	0–7h	Unchanged	
RTCD1.7–0 RTCD0.7–0	RTCD1, (MSB) RTCD0, (LSB)	RTC Day	0–FFFFh	Unchanged	
SRCE	RTCC.7	RTC Subsecond Compare Enable		Unchanged	Unrestricted
SCE	RTCC.6	RTC Second Compare Enable		Unchanged	Unrestricted
MCE	RTCC.5	RTC Minute Compare Enable		Unchanged	Unrestricted
HCE	RTCC.4	RTC Hour Compare Enable		Unchanged	Unrestricted
RTCRE	RTCC.3	RTC Read Enable		0	Unrestricted
RTCWE	RTCC.2	RTC Write Enable		0	Read: Unrestricted Write: Timed Access
RTCIF	RTCC.1	RTC Interrupt Flag		Unchanged	Unrestricted
RTCE	RTCC.0	RTC Enable		Unchanged	Read: Unrestricted Write: Timed Access
E4K	TRIM.7	External 4096 Hz RTC Signal Enable		0	
X12/$\overline{6}$	TRIM.6	RTC Crystal Capacitance Select		Unchanged	
TRM2–0	TRIM.5 TRIM.3 TRIM.1	RTC Trim Bit 2–0		Unchanged	Read: Unrestricted Write: Timed Access

The RTC control and status registers can be subdivided into 4 groups: RTC time registers (RTCSS;FAh, RTCS;FBh, RTCM;FCh, RTCH;FDh, RTCD0;FEh, RTCD1;FFh), RTC alarm registers (RTASS;F2h, RTAS;F3h, RTAM;F4h, RTAH;F5h), RTC calibration (TRIM;96h), and RTC control (RTCC;F9h).

STARTING AND STOPPING THE RTC

Operation of the RTC is enabled by setting the RTC Enable bit, RTCE (RTCC.0) to 1. This will start the RTC crystal amplifier, and begin clocking the RTC. Like all crystal oscillators, the RTC crystal oscillator has a crystal warm–up period. Software should allow a minimum of 1 second between setting the RTCE bit to 1 and initializing the time. This allows the clock to be guaranteed stable when timekeeping begins. Although it may be desired to program the RTC time registers and then start the oscillator, this sequence is not recommended because of the delay incurred by the RTC crystal warm–up period.

There are two situations where the RTC will be started. The first is the case where the RTC has been intentionally halted following normal operation. When the RTCE bit is set, the time registers will continue their count from the last setting when the clock was stopped. The RTC time value will be inaccurate, although the settings of the RTC alarm registers and the RTCC register will remain intact.

The second case is following the application of battery power. Most of the registers associated with the RTC are non–volatile, so that they will maintain their state while V_{CC} is removed. When battery power is applied to the device, however, the battery backed registers and bits associated with the RTC will be in an indeterminate state and will need to be reinitialized. This includes the RTC Interrupt Flag, RTCIF (RTCC.1), which should be cleared before setting the RTC Interrupt Enable bit (EIE.5).

The RTC can be halted by clearing the RTCE bit to 0. This will immediately halt the RTC and will freeze all the time registers at their current value and preserve all the RTC settings. If RTC functions are not desired, this can be used to reduce the power consumption of the device while in battery backed mode.

SETTING AND READING THE RTC TIME REGISTERS

Access to the RTC time registers (RTCSS, RTCS, RTCM, RTCH, RTCD0, RTCD1) is enabled by the RTCRE (RTCC.3) and RTCWE (RTCC.2) bits. Both user software and the internal clock directly write and read the RTC time. To prevent the possibility of both user software and the internal timer accessing the same register simultaneously, the DS87C530 incorporates a register locking mechanism. Updates to the RTC time registers by the internal timer are temporarily suspended for up to 1.95 ms during software read or write operations. If a subsecond timer tick should occur during the 1.95 ms window, it will be processed immediately as soon as either the RTCWE or RTCRE bit is cleared. Because the subsecond timer tick interval is 3.906 ms, the 1.95 ms window allows sufficient time to complete any operations and process suspended timer ticks before the next timer tick occurs. In this way, no timer ticks can be lost, and accessing the time registers will not affect the accuracy of the RTC. To allow any pending timer ticks to propagate through the RTC circuitry, software must wait 4 machine cycles after setting the RTCWE or RTCRE bits before accessing any of the RTC time registers. The first timer tick following the clearing of the RTWCE bit will be approximately 1.95 ms. All following timer ticks will be 3.90625 ms.

Reading the current time from any or all of the RTC time registers is accomplished by the following procedure:

1. Disable all interrupts by clearing the EA bit (IE.7),
2. Set the RTCRE bit (RTCC.3),
3. Wait 4 machine cycles,
4. Read the appropriate register(s) within 1 ms of RTCRE being set,
5. Clear the RTCRE bit (RTCC.3),
6. Enable interrupts by setting the EA bit (IE.7).

The time on the DS87C530 is set by writing to the Clock Registers. The Second, Minute, Hour, Day of the Week, and Day Count can be set by writing to the respective registers. It is not possible to set the Subsecond Real Time Clock Register (RTCS; FAh). This register is automatically reset to 00h when the RTCWE bit is cleared, either through software or the automatic time–out of the 1.95 ms write window. Writing an invalid time to these

registers (loading the RTCM register with 3Dh or 61 minutes, for example) will result in an inaccurate count by the RTC. It is the responsibility of the software to ensure that only valid times are written to these registers.

The procedure for setting an RTC time register is as follows:

1. Disable all interrupts by clearing the EA bit (IE.7),
2. Perform a Timed Access procedure,
3. Set the RTCWE bit (RTCC.2),
4. Wait 4 machine cycles,
5. Write the appropriate register(s) within 1.95 ms of RTCWE being set,
6. Perform a Timed Access procedure,
7. Clear the RTCWE bit (RTCC.2),
8. Enable interrupts by setting the EA bit (IE.7).

USING THE RTC ALARM

The RTC alarm function is used to generate an interrupt when the RTC value matches selected alarm register values. An alarm can be triggered by a match on one or more of the following alarm registers: Subsecond (RTASS;F2h), Second (RTAS; F3h), Minute (RTAM; F4h), and Hour (RTAH; F5h). Note that there is no alarm register associated with the RTC Day Count or Day of Week registers. If an alarm is desired on a specific date, an alarm can be executed once a day and user software can compare the current date against the Day Register. It is not necessary to set the RTC Write Enable bit, RTCWE, when setting the alarm registers.

The alarm can be set to occur on a match with any or all of the alarm registers. An alarm can occur on a unique time of day, or a recurring alarm can be programmed every subsecond, second, minute, or hour. Alarms can occur synchronously, when the clock rolls over to match the alarm condition, or asynchronously, if the alarm registers are set to a value that matches the current time. Note that only one alarm may occur per subsecond tick. This means that if a synchronous alarm has already occurred during the current subsecond, software cannot cause an asynchronous alarm in the same subsecond.

The specific alarm registers to be compared are selected by setting or clearing the corresponding compare enable bits (RTCC.7–4). Any compare bit which is cleared will result in that register being treated as a Don't Care when evaluating alarm conditions. Clearing all the compare enable bits will disable the ability of the RTC to cause an interrupt, and will immediately clear the RTC Interrupt Flag (RTCC.1). Unlike some interrupts, the RTC flag is not cleared by exiting the RTC interrupt service routine and must be explicitly cleared in software.

The general procedure for setting the RTC alarm registers to cause a RTC interrupt is as follows:

1. Clear the ERTCI Enable bit (EIE.5),
2. Clear all RTC Alarm Compare enable bits (ANL RTCC, #0Fh),
3. Write one or more RTC Alarm registers,
4. Set the desired RTC Alarm Compare enable bits,
5. Set the ERTCI Enable bit (EIE.5).

Setting the alarm to cause an interrupt once during a 24–hour period is done by setting all the alarm registers to the desired value and enabling all compare bits. A recurring alarm is enabled by clearing the compare enable bits associated with one or more alarm registers. For example, to specify an alarm to occur once a minute, the SSCE and SCE bits would be set. In general, a recurring alarm is set using the next lower time increment than the desired interrupt period. For example, if an alarm was desired once an hour, on the hour, a compare on the Real Time Alarm Minute Register would be performed, because the Real Time Clock Minute Register will match the corresponding alarm register only once an hour. The RTASS, RTAS, and RTAM registers would be cleared to 00h, and the SSCE, SCE, and MCE bits would all be set to 1 to match on the time xx:00:00:00. Writing an invalid time to these registers (loading the RTAM register with 3Dh or 61 minutes, for example) will never cause a match by the RTC. It is the responsibility of the software to ensure that only valid times are written to these registers.

It is important to remember that any RTC register whose corresponding compare enable bit is cleared to 0 will always be treated as a match. The alarm registers are interrogated once per subsecond tick to check for an alarm condition. If the SSCE bit was set to a don't care (cleared to 0) in the above example, a match (and interrupt) would occur during every subsecond of the minute in which the Real Time Alarm Minute register matched.

If an alarm occurs while in data retention state ($V_{CC} < V_{BAT}$), the RTCIF flag will be set and the interrupt will remain pending. When power is reapplied to the device, the device will execute an RTC interrupt as soon as interrupts are enabled.

USING THE DAY OF THE WEEK BITS

The DS87C530 contains 3 Day of the Week bits, DOW.2–0 located in the upper 3 bits of the Real Time Clock Hour register (RTCH;FDh). These allow the processor to count from 1 to 7. The day of the week bits will increment anytime the hour register changes from 17h to 00h, indicating a new day. When the day of the week register reaches a count of 111b, it will roll over to 001b.

If the day of the week feature is not needed, writing 000b to the bits will disable the ability of an hour register rollover to change the day of the week. The bits will remain at 000b. This is very convenient from a software standpoint, as it is not necessary to zero out the high order bits when determining the hour from the RTC Hour register.

CHOOSING AN RTC CRYSTAL

The RTC clock source is provided by an external 32.768 kHz crystal attached to the RTCX1 and RTCX2 leads of the DS87C530. The device can be programmed to operate with a crystal rated for either a 6 pF or 12.5 pF load capacitance. The crystal selection is determined by the RTC Crystal Capacitance Select bit (TRIM.6). The default state of this bit after a no–battery reset is for a 12.5 pF crystal.

In general, a lower capacitance crystal will consume less power, but will be more susceptible to noise. Unlike the processor crystal inputs (X1, X2), the RTC crystal does not require external load capacitors. Placing load capacitors on the RTC crystal input pins will cause the RTC to keep incorrect time. To prevent system noise from affecting the RTC, the RTCX1 and RTCX2 pins should be guard–ringed with the GND2 signal.

CALIBRATING THE RTC OSCILLATOR

Although the DS87C530 RTC accuracy is guaranteed for +/– 2 minutes per month, users may occasionally require greater accuracy. The RTC incorporates the ability to adjust the internal capacitance of the crystal amplifier via the RTC Trim Bits (TRM2–0 and $\overline{\text{TRM2–0}}$). This allows the user to more accurately match the capacitance of the crystal amplifier to the crystal. Note that under most circumstances no adjustment of the RTC crystal capacitance is necessary, as it will default to a minimum accuracy of +/– 2 minutes per month.

All of the crystal capacitance controls are located in the RTC Trim register (TRIM;96h). Setting the E4K bit will enable the output of a 4096 Hz signal on P1.7. This signal is derived from a divide by 8 of the 32.768 kHz crystal. Because this is directly generated from the RTC, it can be used to determine the actual frequency of the RTC. By adjusting the value of the TRMx bits, the internal capacitance of the RTC can be varied, slightly slowing or speeding up the RTC frequency. The combination of TRMx bits (TRIM.5–0) that causes the output on pin P1.7 to most closely approximate 4096 Hz will provide the most accurate setting of RTC capacitance.

As a precaution against accidental corruption of the oscillator trim bit settings, the $\overline{\text{TRMx}}$ bits must be programmed in the same instruction to the inverse of their respective TRMx bits. For example, if a trim bit setting of 5 (101) was desired, the $\overline{\text{TRMx}}$ bits should be set to 2 (010). An illegal combination will automatically reset the TRIM register to 0x100101b. This will disable the E4K signal on P1.7, but leave the X12/$\overline{6}$ bit unmodified.

More information on calibrating the RTC oscillator for improved accuracy is located in Application Note 79, Using the DS87C530 Real Time Clock.

SECTION 15: BATTERY BACKUP

The DS87C530 incorporates a feature which can maintain timekeeping and on–chip SRAM contents in the absence of V_{CC}. An external energy source such as a lithium battery or 0.47 F super cap can be connected to the V_{BAT} pin. The nominal battery voltage should be 3V. For proper operation, the battery voltage must always be at least a diode drop (0.7V) below V_{CC}, and is recommended to be below V_{RST}.

The DS87C530 will automatically enter data retention mode when $V_{CC} < V_{BAT}$. When in data retention mode, the RTC and SRAM contents are powered from the energy source connected to the V_{BAT} pin and electrically isolated from the rest of the device. This means that writes to battery backed SFRs and SRAM are ignored and reads will return erroneous data while in data retention mode. The DS87C530 Data Sheet contains a functional diagram of the internal battery switching circuitry.

The data retention switch voltage, the point at which the device switches into data retention mode, is a function of the battery voltage, not an absolute reference. Care must be taken when selecting a battery so that its voltage will stay below V_{CC} during normal operation to prevent an unplanned lockout of the RTC and SRAM. Although it is unlikely that such a situation would occur, it could become an issue if a relatively high voltage battery is used. For example, suppose a 4.5 V battery is used with a device operating at a V_{CC} of 5.0 V. During normal operation, V_{CC} will be above V_{BAT}, so no problem will occur. Suppose that a loss of power occurs, and V_{CC} begins to drop. Under normal circumstances, the device will continue to operate until it reaches V_{RST} (4.0V to 4.25V), at which time device operation will halt. If V_{BAT} is higher than V_{RST}, however, RTC and SRAM access will be prohibited before the device enters reset. This means that there may be a short period of time before reset when the device is operating but could read erroneous data from the RTC or SRAM or fail to write to them. One solution would be to use the power–fail interrupt to halt reads or writes to the RTC or SRAM when V_{CC} is dropping. The best approach is to carefully select battery voltages to avoid the problem entirely.

SELECTING A BATTERY

There are a number of battery chemistries and brands that are suitable for use with battery–backed members of the High–Speed Microcontroller Family. The use of lithium chemistry batteries, such as Lithium Manganese Dioxide, is preferred as their nominal voltage is approximately 3.0V. Coin cells are particularly suited for use with the High–Speed Microcontroller Family because of their capacity, low profile, and small diameter. Many are available with PC mount tabs attached for automated assembly. Table 15–1 shows a list of some common batteries and their capacities. This list is by no means exhaustive, and the inclusion or exclusion of any vendor from this list is in no way a comment on the suitability of a specific battery in a customer's application.

SUGGESTED BATTERIES FOR USE WITH DS87C530 Table 15–1

MANUFACTURER	MODEL NUMBER	TYPE	NOMINAL VOLTAGE	CAPACITY
Panasonic PH: (201) 348–5266	CR1620	Lithium/Manganese Dioxide	3 V	70 mAh
	CR1616	Lithium/Manganese Dioxide	3 V	50 mAh
	CR1220	Lithium/Manganese Dioxide	3 V	35 mAh
	BR1616	Lithium/poly–carbon monofluoride	3 V	48 mAh
	BR1225	Lithium/poly–carbon monofluoride	3 V	38 mAh

Battery life can be calculated by dividing the rated battery capacity by the I_{BAT} current specified on the device specific data sheet. Note that this determines the minimum battery life; while V_{CC} is applied to the device, it draws negligible current from the battery, and so battery life will be lengthened accordingly. Backup current is a function of temperature, and therefore battery life is dependent on the operating environment.

The registers shown in Table 15–2 are battery–backed, and one or more bits will be indeterminate following a no–battery reset. They should be initialized as part of a no–battery reset procedure.

BATTERY BACKED SFRS Table 15–2

REGISTER NAME	LOCATION
TRIM	96h
RTASS	F2h
RTAS	F3h
RTAM	F4h
RTAH	F5h
RTCC	F9h
RTCSS	FAh
RTCS	FBh
RTCM	FCh
RTCH	FDh
RTCD0	FEh
RTCD1	FFh

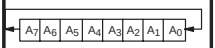
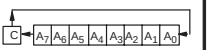
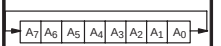
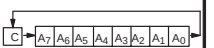
LITHIUM BATTERY CONSIDERATIONS

Lithium primary batteries (non–rechargeable) can fail and/or rupture if subjected to reverse current from the device they are powering. The battery–switching circuitry inside the DS87C530 was designed to reduce or eliminate the need for external hardware required to meet battery safety regulations. As shown in the DS87C530 Data Sheet, a current limiting resistor is always in series with a switching field–effect transistor, regardless of whether the DS87C530 is drawing current from V_{CC} or V_{BAT} pins. This satisfies the two mechanism requirement of most safety codes.

SECTION 16: INSTRUCTION SET DETAILS

	MNEMONIC	INSTRUCTION CODE								HEX	BYTE	CYCLE	EXPLANATION
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0				
ARITHMETIC OPERATION	ADD A, Rn	0	0	1	0	1	n_2	n_1	n_0	28-2F	1	1	(A) = (A) + (Rn)
	ADD A, direct	0 a_7	0 a_6	1 a_5	0 a_4	0 a_3	1 a_2	0 a_1	1 a_0	25 Byte 2	2	2	(A) = (A) + (direct)
	ADD A, @Ri	0	0	1	0	0	1	1	i	26-27	1	1	(A) = (A) + ((Ri))
	ADD A, #data	0 d_7	0 d_6	1 d_5	0 d_4	0 d_3	1 d_2	0 d_1	0 d_0	24 Byte 2	2	2	(A) = (A) + #data
	ADDC A, Rn	0	0	1	1	1	n_2	n_1	n_0	38-3F	1	1	(A) = (A)+(C)+(Rn)
	ADDCA,direct	0 a_7	0 a_6	1 a_5	1 a_4	0 a_3	1 a_2	0 a_1	1 a_0	35 Byte 2	2	2	(A) = (A)+(C)+(direct)
	ADDC A, @Ri	0	0	1	1	0	1	1	i	36-37	1	1	(A) = (A)+(C)+((Ri))
	ADDC A,#data	0 d_7	0 d_6	1 d_5	1 d_4	0 d_3	1 d_2	0 d_1	0 d_0	34 Byte 2	2	2	(A) = (A)+(C)+#data
	SUBB A, Rn	1	0	0	1	1	n_2	n_1	n_0	98-9F	1	1	(A) = (A)-(C)-(Rn)
	SUBB A,direct	1 a_7	0 a_6	0 a_5	1 a_4	0 a_3	1 a_2	0 a_1	1 a_0	95 Byte 2	2	2	(A) = (A)-(C)-(direct)
	SUBB A, @Ri	1	0	0	1	0	1	1	i	96-97	1	1	(A) = (A)-(C)-((Ri))
	SUBB A,#data	1 d_7	0 d_6	0 d_5	1 d_4	0 d_3	1 d_2	0 d_1	0 d_0	94 Byte 2	2	2	(A) = (A)-(C)-#data
	INC A	0	0	0	0	0	1	0	0	04	1	1	(A) = (A) + 1
	INC Rn	0	0	0	0	1	n_2	n_1	n_0	08-0F	1	1	(Rn) = (Rn) + 1
	INC direct	0 a_7	0 a_6	0 a_5	0 a_4	0 a_3	1 a_2	0 a_1	1 a_0	05 Byte 2	2	2	(direct) = (direct)+1
	INC @Ri	0	0	0	0	0	1	1	i	06-07	1	1	((Ri)) = ((Ri)) + 1
	INC DPTR	1	0	1	0	0	0	1	1	A3	1	3	(DPTR)=(DPTR)+1
	DEC A	0	0	0	1	0	1	0	0	14	1	1	(A) = (A) - 1
	DEC Rn	0	0	0	1	1	n_2	n_1	n_0	18-1F	1	1	(Rn) = (Rn) - 1
	DEC direct	0 a_7	0 a_6	0 a_5	1 a_4	0 a_3	1 a_2	0 a_1	1 a_0	15 Byte 2	2	2	(direct) = (direct)-1
	DEC @Ri	0	0	0	1	0	1	1	i	16-17	1	1	((Ri)) = ((Ri)) - 1
	MUL AB	1	0	1	0	0	1	0	0	A4	1	5	$(B_{15–8})$, $(A_{7–0})$ = (A) X (B)
	DIV AB	1	0	0	0	0	1	0	0	84	1	5	$(A_{15–8})$, $(A_{7–0})$ = (A) ÷ (B)

	MNEMONIC	INSTRUCTION CODE								HEX	BYTE	CYCLE	EXPLANATION
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0				
ARITHMETIC OPER.	DA A	1	1	0	1	0	1	0	0	D4	1	1	Contents of Accumulator are BCD, IF [[(A_{3-0}) > 9] OR [(AC) = 1]] THEN (A_{3-0}) = (A_{3-0}) + 6 AND IF [[(A_{7-4}) > 9] OR [(C) = 1]] THEN (A_{7-4}) = (A_{7-4}) + 6
LOGICAL OPERATION	ANL A, Rn	0	1	0	1	1	n_2	n1	n_0	58-5F	1	1	(A) = (A) AND (Rn)
	ANL A, direct	0 a_7	1 a_6	0 a_5	1 a_4	0 a_3	1 a_2	0 a_1	1 a_0	55 Byte 2	2	2	(A) = (A) AND (direct)
	ANL A, @Ri	0	1	0	1	0	1	1	i	56-57	1	1	(A) = (A) AND ((Ri))
	ANL A, #data	0 d_7	1 d_6	0 d_5	1 d_4	0 d_3	1 d_2	0 d_1	0 d_0	54 Byte 2	2	2	(A)=(A) AND #data
	ANL direct, A	0 a_7	1 a_6	0 a_5	1 a_4	0 a_3	0 a_2	1 a_1	0 a_0	52 Byte 2	2	2	(direct) = (direct) AND A
	ANL direct, #data	0 a_7 d_7	1 a_6 d_6	0 a_5 d_5	1 a_4 d_4	0 a_3 d_3	0 a_2 d_2	1 a_1 d_1	1 a_0 d_0	53 Byte 2 Byte 3	3	3	(direct) = (direct) AND #data
	ORL A, Rn	0	1	0	0	1	n_2	n_1	n_0	48-4F	1	1	(A) = (A) OR (Rn)
	ORL A, direct	0 a_7	1 a_6	0 a_5	0 a_4	0 a_3	1 a_2	0 a_1	1 a_0	45 Byte 2	2	2	(A) = (A) OR (direct)
	ORL A, @Ri	0	1	0	0	0	1	1	i	46-47	1	1	(A) = (A) OR ((Ri))
	ORL A, #data	0 d_7	1 d_6	0 d_5	0 d_4	0 d_3	1 d_2	0 d_1	0 d_0	44 Byte 2	2	2	(A) = (A) OR #data
	ORL direct, A	0 a_7	1 a_6	0 a_5	0 a_4	0 a_3	0 a_2	1 a_1	0 a_0	42 Byte 2	2	2	(direct) = (direct) OR (A)
	ORL direct,#data	0 a_7 d_7	1 a_6 d_6	0 a_5 d_5	0 a_4 d_4	0 a_3 d_3	0 a_2 d_2	1 a_1 d_1	1 a_0 d_0	43 Byte 2 Byte 3	3	3	(direct) = (direct) OR #data
	XRL A, Rn	0	1	1	0	1	n_2	n_1	n_0	68-6F	1	1	(A) = (A) XOR (Rn)
	XRL A, direct	0 a_7	1 a_6	1 a_5	0 a_4	0 a_3	1 a_2	0 a_1	1 a_0	65 Byte 2	2	2	(A) = (A) XOR (direct)
	XRL A, @ Ri	0	1	1	0	0	1	1	i	66-67	1	1	(A) = (A) XOR ((Ri))
	XRL A, #data	0 d_7	1 d6	1 d5	0 d_4	0 d_3	1 d_2	0 d_1	0 d_0	64 Byte 2	2	2	(direct) = (A) XOR #data
	XRL direct, A	0 a_7	1 a_6	1 a_5	0 a_4	0 a_3	0 a_2	1 a_1	0 a_0	62 Byte 2	2	2	(direct) = (direct) XOR (A)
	XRL direct, #data	0 a_7 d_7	1 a_6 d_6	1 a_5 d_5	0 a_4 d_4	0 a_3 d_3	0 a_2 d_2	1 a_1 d_1	1 a_0 d_0	63 Byte 2 Byte 3	3	3	(direct) = (direct) XOR #data
	CLR A	1	1	1	0	0	1	0	0	E4	1	1	(A) = 0
	CPL A	1	1	1	1	0	1	0	0	F4	1	1	(A) = ($\overline{A}$)

	MNEMONIC	INSTRUCTION CODE								HEX	BYTE	CYCLE	EXPLANATION
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0				
LOGICAL OPERATION	RL A	0	0	1	0	0	0	1	1	23	1	1	 The contents of the accumulator are rotated left by one bit.
	RLC A	0	0	1	1	0	0	1	1	33	1	1	 The contents of the accumulator are rotated left by one bit.
	RR A	0	0	0	0	0	0	1	1	03	1	1	 The contents of the accumulator are rotated right by one bit.
	RRC A	0	0	0	1	0	0	1	1	13	1	1	 The contents of the accumulator are rotated right by one bit.
	SWAP A	1	1	0	0	0	1	0	0	C4	1	1	$(A_{3-0}) \leftrightarrows (A_{7-4})$
DATA TRANSFER	MOV A, Rn	1	1	1	0	1	n_2	n_1	n_0	E8-EF	1	1	(A) = (Rn)
	MOV A, direct	1 a_7	1 a_6	1 a_3	0 a_4	0 a_3	1 a_2	0 a_1	1 a_0	E5 Byte 2	2	2	(A) = (direct)
	MOV A, @Ri	1	1	1	0	0	1	1	i	E6-E7	1	1	(A) = ((Ri))
	MOV A, #data	0 d_7	1 d_6	1 d_5	1 d_4	0 d_3	1 d_2	0 d_1	0 d_0	74 Byte 2	2	2	(A) = #data
	MOV Rn, A	1	1	1	1	1	n_2	n_1	n_0	F8-FF	1	1	(Rn) = (A)
	MOV Rn, direct	1 a_7	0 a_6	1 a_5	0 a_4	1 a_3	n_2 a_2	n_1 a_1	n_0 a_0	A8-AF Byte 2	2	2	(Rn) = (direct)
	MOV Rn, #data	0 d_7	1 d_6	1 d_5	1 d_4	1 d_3	n_2 d_2	n_1 d_1	n_0 d_0	78-7F Byte 2	2	2	(Rn) = #data
	MOV direct, A	1 a_7	1 a_6	1 a_5	1 a_4	0 a_3	1 a_2	0 a_1	1 a_0	F5 Byte 2	2	2	(direct) = (A)
	MOV direct, Rn	1 a_7	0 a_6	0 a_5	0 a_4	1 a_3	n_2 a_2	n_1 a_1	n_0 a_0	88-8F Byte 2	2	2	(direct) = (Rn)
	MOV direct1, direct2	1 a_7 a_7	0 a_6 a_6	0 a_5 a_5	0 a_4 a_4	0 a_3 a_3	1 a_2 a_2	0 a_1 a_1	1 a_0 a_0	85 Byte 2 Byte 3	3	3	(direct1) = (direct2) (source) (destination)
	MOV direct, @Ri	1 a_7	0 a_6	0 a_5	0 a4	0 a_3	1 a_2	1 a_1	i a_0	86-87 Byte 2	2	2	(direct) = ((Ri))

	MNEMONIC	INSTRUCTION CODE								HEX	BYTE	CYCLE	EXPLANATION
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0				
DATA TRANSFER	MOV direct, #data	0 a_7 d_7	1 a_6 d_6	1 a_5 d_5	1 a_4 d_4	0 a_3 d_3	1 a_2 d_2	0 a_1 d_1	1 a_0 d_0	75 Byte 2 Byte 3	3	3	(direct) = #data
	MOV @Ri, A	1	1	1	1	0	1	1	i	F6-F7	1	1	((Ri)) = A
	MOV @Ri, direct	1 a_7	0 a_6	1 a_5	0 a_4	0 a_3	1 a_2	1 a_1	i a_0	A6-A7 Byte 2	2	2	((Ri)) = (direct)
	MOV @Ri, #data	0 d_7	1 d_6	1 d_5	1 d_4	0 d_3	1 d_2	1 d1	i d_0	76-77 Byte 2	2	2	((Ri)) = #data
	MOV DPTR, #data16	1 d_7 d_7	0 d_6 d_6	0 d_5 d_5	1 d4 d_4	0 d_3 d_3	0 d_2 d_2	0 d_1 d_1	0 d_0 d_0	90 Byte 2 Byte 3	3	3	(DPTR) = #$data_{15-0}$ (DPH) = #$data_{15-8}$ (DPL) = #$data_{7-0}$
	MOVC A, @A + DPTR	1	0	0	1	0	0	1	1	93	1	3	(A)=((A) + (DPTR))
	MOVC A, @A + PC	1	0	0	0	0	0	1	1	83	1	3	(A) = ((A) + (PC))
	MOVX A, @Ri	1	1	1	0	0	0	1	i	E2-E3	1	2–9	(A) = ((Ri))
	MOVX @DPTR	1	1	1	0	0	0	0	0	E0	1	2–9	(A) = ((DPTR))
	MOVX @Ri, A	1	1	1	1	0	0	1	i	F2-F3	1	2–9	((Ri)) = (A)
	MOVX @DPTR,A	1	1	1	1	0	0	0	0	F0	1	2–9	((DPTR)) = (A)
	PUSH direct	1 a_7	1 a_6	0 a_5	0 a_4	0 a_3	0 a_2	0 a_1	0 a_0	C0 Byte 2	2	2	(SP) = (SP) + 1 ((SP)) = (direct)
	POP direct	1 a_7	1 a_6	0 a_5	1 a_4	0 a_3	0 a_2	0 a_1	0 a_0	D0 Byte 2	2	2	(direct) = ((SP)) (SP) = (SP) - 1
	XCH A, Rn	1	1	0	0	1	n_2	n_1	n_0	C8-CF	1	1	(A) = (Rn)
	XCH A, direct	1 a_7	1 a_6	0 a_5	0 a_4	0 a_3	1 a_2	0 a_1	1 a_0	C5 Byte 2	2	2	(A) = (direct)
	XCH A, @Ri	1	1	0	0	0	1	1	i	C6-C7	1	1	(A) = ((Ri))
	XCHD A, @Ri	1	1	0	1	0	1	1	i	D6-D7	1	1	(A_{3-0}) = ((Ri_{3-0}))

	MNEMONIC	INSTRUCTION CODE								HEX	BYTE	CYCLE	EXPLANATION
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0				
BOOLEAN VARIABLE MANIPULATION	CLR C	1	1	0	0	0	0	1	1	C3	1	1	(C) = 0
	CLR bit	1 b_7	1 b_6	0 b_5	0 b_4	0 b_3	0 b_2	1 b_1	0 b_0	C2 Byte 2	2	2	(bit) = 0
	SETB C	1	1	0	1	0	0	1	1	D3	1	1	(C) = 1
	SETB bit	1 b_7	1 b_6	0 b_5	1 b_4	0 b_3	0 b_2	1 b_1	0 b_0	D2 Byte 2	2	2	(bit) = 1
	CPL C	1	0	1	1	0	0	1	1	B3	1	1	(C) = ($\overline{C}$)
	CPL bit	1 b_7	0 b_6	1 b_5	1 b_4	0 b_3	0 b_2	1 b_1	0 b_0	B2 Byte 2	2	2	(bit) = ($\overline{bit}$)
	ANL C, bit	1 b_7	0 b_6	0 b_5	0 b_4	0 b_3	0 b_2	1 b_1	0 b_0	82 Byte 2	2	2	(C) = (C) AND (bit)
	ANL C, $\overline{bit}$	1 b_7	0 b_6	1 b_5	1 b_4	0 b_3	0 b_2	0 b_1	0 b_0	B0 Byte 2	2	2	(C) = (C) AND ($\overline{bit}$)
	ORL C, bit	0 b_7	1 b6	1 b_5	1 b_4	0 b_3	0 b_2	1 b_1	0 b_0	72 Byte 2	2	2	(C) = (C) OR (bit)
	ORL C, $\overline{bit}$	1 b_7	0 b_6	1 b_5	0 b_4	0 b_3	0 b_2	0 b_1	0 b_0	A0 Byte 2	2	2	(C) = (C) OR ($\overline{bit}$)
	MOV C, bit	1 b_7	0 b_6	1 b_5	0 b_4	0 b_3	0 b_2	1 b_1	0 b_0	A2 Byte 2	2	2	(C) = (bit)
	MOV bit, C	1 b_7	0 b_6	0 b_5	1 b_4	0 b_3	0 b_2	1 b_1	0 b_0	92 Byte 2	2	2	(bit) = (C)

	MNEMONIC	INSTRUCTION CODE								HEX	BYTE	CYCLE	EXPLANATION
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0				
PROGRAM BRANCHING	ACALL addr 11	a_{10} a_7	a_9 a_6	a_8 a_5	1 a_4	0 a_3	0 a_2	0 a_1	1 a_0	Byte 1 Byte 2	2	3	(PC) = (PC) + 2 (SP) = (SP) + 1 ((SP)) = (PC_{7-0}) (SP) = (SP) + 1 ((SP)) = (PC_{15-8}) (PC)=page address
	LCALL addr 16	0 a_{15} a_7	0 a_{14} a_6	0 a_{13} a_5	1 a_{12} a_4	0 a_{11} a_3	0 a_{10} a_2	1 a_9 a_1	0 a_8 a_0	12 Byte 2 Byte 3	3	4	(PC) = (PC) + 3 (SP) = (SP) + 1 ((SP)) = (PC_{7-0}) (SP) = (SP) + 1 ((SP)) = (PC_{15-8}) (PC) = $addr_{15-0}$
	RET	0	0	1	0	0	0	1	0	22	1	4	(PC_{15-8}) = ((SP)) (SP) = (SP) - 1 (PC_{7-0}) = ((SP)) (SP) = (SP) - 1
	RETI	0	0	1	1	0	0	1	0	32	1	4	(PC_{15-8}) = ((SP)) (SP) = (SP) - 1 (PC_{7-0}) = ((SP)) (SP) = (SP) - 1
	AJMP addr 11	a_{10} a_7	a_9 a_6	a_8 a_5	0 a_4	0 a_3	0 a_2	0 a_1	1 a_0	Byte 1 Byte 2	2	3	(PC) = (PC) + 2 (PC_{10-0}) = page addr
	LJMP addr 16	0 a_{15} a_7	0 a_{14} a_6	0 a_{13} a_5	0 a_{12} a_4	0 a_{11} a_3	0 a_{10} a_2	1 a_9 a_1	0 a_8 a_0	02 Byte 2 Byte 3	3	4	(PC) = addr15-0
	SJMP rel	1 r_7	0 r_6	0 r_5	0 r_4	0 r_3	0 r_2	0 r_1	0 r_0	80 Byte 2	2	3	(PC) = (PC) + 2 (PC) = (PC) + rel
	JMP @A + DPTR	0	1	1	1	0	0	1	1	73	1	3	(PC) = (A) + (DPTR)
	JZ rel	0 r_7	1 r_6	1 r_5	0 r_4	0 r_3	0 r_2	0 r_1	0 r_0	60 Byte 2	2	3	(PC) = (PC) + 2 IF (A) = 0 THEN (PC) = (PC) + rel
	JNZ rel	0 r_7	1 r_6	1 r_5	1 r_4	0 r_3	0 r_2	0 r_1	0 r_0	70 Byte 2	2	3	(PC) = (PC) + 2 IF (A) $\neq$ 0 THEN (PC) = (PC) + rel

	MNEMONIC	INSTRUCTION CODE								HEX	BYTE	CYCLE	EXPLANATION
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0				
PROGRAM BRANCHING	JC rel	0 r_7	1 r_6	0 r_5	0 r_4	0 r_3	0 r_2	0 r_1	0 r_0	40 Byte 2	2	3	(PC) = (PC) + 2 IF (C) = 1 THEN (PC) = (PC) + rel
	JNC rel	0 r_7	1 r_6	0 r_5	1 r_4	0 r_3	0 r_2	0 r_1	0 r_0	50 Byte 2	2	3	(PC) = (PC) + 2 IF (C) $\neq$ 0 THEN (PC) = (PC) + rel
	JB bit, rel	0 b_7 r_7	0 b_6 r_6	1 b_5 r_5	0 b_4 r_4	0 b_3 r_3	0 b_2 r_2	0 b_1 r_1	0 b_0 r_0	20 Byte 2 Byte 3	3	4	(PC) = (PC) + 3 IF (bit) = 1 THEN (PC) = (PC) + rel
	JNB bit, rel	0 b_7 r_7	0 b_6 r_6	1 b_5 r_5	1 b_4 r_4	0 b_3 r_3	0 b_2 r_2	0 b_1 r_1	0 b_0 r_0	30 Byte 2 Byte 3	3	4	(PC) = (PC) + 3 IF (bit) = 0 THEN (PC) = (PC) + rel
	JBC bit, direct rel	0 b_7 r_7	0 b_6 r_6	0 b_5 r_5	1 b_4 r_4	0 b_3 r_3	0 b_2 r_2	0 b_1 r_1	0 b_0 r_0	10 Byte 2 Byte 3	3	4	(PC) = (PC) + 3 IF (bit) = 1 THEN (bit) = 0 (PC) = (PC) + rel
	CJNE A, direct rel	1 a7 r_7	0 a6 r_6	1 a5 r_5	1 ad r_4	0 a3 r_3	1 a2 r_2	0 a1 r_1	1 a0 r_0	B5 Byte 2 Byte 3	3	4	(PC) = (PC) + 3 IF (direct) < (A) THEN (PC) = (PC) + rel and (C) = 0 OR IF (direct) > (A) THEN (PC) = (PC) + rel and (C) = 1
	CJNE A, #data rel	1 d7 r_7	0 d6 r_6	1 d5 r_5	1 d4 r_4	0 d3 r_3	1 d2 r_2	0 d1 r_1	0 d0 r_0	B4 Byte 2 Byte 3	3	4	(PC) = (PC) + 3 IF #data < (A) THEN (PC) = (PC) + rel and (C) = 0 OR IF #data > (A) THEN (PC) = (PC) + rel and (C) = 1
	CJNE Rn, #data rel	1 d7 r_7	0 d6 r_6	1 d5 r_5	1 d4 r_4	1 d3 r_3	n_2 d2 r_2	n_1 d1 r_1	n_0 d0 r_0	B8-BF Byte 2 Byte 3	3	4	(PC) = (PC) + 3 IF #data < (Rn) THEN (PC) = (PC) + rel and (C) = 0 OR IF #data > (Rn) THEN (PC) = (PC) + rel and (C) = 1
	CJNE @Ri, #data rel	1 d7 r_7	0 d6 r_6	1 d5 r_5	1 d4 r_4	0 d3 r_3	1 d2 r_2	1 d1 r_1	i d0 r_0	B6-B7 Byte 2 Byte 3	3	4	(PC) = (PC) + 3 IF #data < ((Ri)) THEN (PC) = (PC) + rel and (C) = 0 OR IF #data > ((Ri)) THEN (PC) = (PC) + rel and (C) = 1

	MNEMONIC	INSTRUCTION CODE								HEX	BYTE	CYCLE	EXPLANATION
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0				
PROGRAM BRANCHING	DJNZ Rn, rel	1 r_7	1 r_6	0 r_5	1 r_4	1 r_3	n_2 r_2	n_1 r_1	n_0 r_0	D8-Df Byte 2	2	3	(PC) = (PC) + 2 (Rn) = (Rn) - 1 IF (Rn) 0 THEN (PC) = (PC) + rel
	DJNZ direct rel	1 a_7 r_7	1 a_6 r_6	0 a_5 r_5	1 a_4 r_4	0 a_3 r_3	1 a_2 r_2	0 a_1 r_1	1 a_0 r_0	D5 Byte 2 Byte 3	3	4	(PC) = (PC) + 3 (direct) = (direct) - 1 IF (direct)$\neq$0 THEN (PC) = (PC) + rel
	NOP	0	0	0	0	0	0	0	0	00	1	1	(PC) = (PC) + 1

SECTION 17: TROUBLESHOOTING

DEVICE OPERATES AT 1/3 OF CRYSTAL SPEED

The High Speed Microcontroller Family operates from the primary or fundamental mode of the external crystal. Many off–the–shelf high–frequency crystals are specified to operate from their third overtone. When used with a High–Speed Microcontroller, these crystals will resonate in their primary mode, which will appear to be 1/3 of the rated crystal speed. Make sure that any crystals used will operate at their rated speed in primary mode.

DEVICE RESETS FOR NO REASON

During the debugging process, it may be necessary to isolate the cause of an unexpected device reset. Because resets are initiated by a limited number of sources, it is relatively easy to determine their source by interrogating a few bits. These bits should be interrogated early in the code following a reset to determine its source. As a debug tool, software could set the state of one or more port pins to indicate the type of reset to the designer. Note that power supply problems or glitches will appear as unplanned power–on resets.

SOURCE	POR BIT WDCON.6	WTRF BIT WDCON.3
Power–on reset	1	0
Watchdog reset	0	1
External reset	0	0

ACCESS TO INTERNAL MOVX SRAM IS UNSUCCESSFUL

The internal MOVX SRAM available on some members of the High Speed Microcontroller Family is disabled after any reset. To enable the on–chip SRAM, the software should configure the Data Memory Enable bits (PMR.1–0) as needed.

When V_{CC} drops below V_{BAT}, access to the SRAM is disabled to prevent corruption of the data. If the battery voltage is greater than V_{RST}, this means that the processor may continue to operate while SRAM access is denied. Make sure that the battery voltage remains below the minimum V_{RST}.

REAL–TIME CLOCK DOES NOT OPERATE OR KEEP ACCURATE TIME.

The state of the real–time clock (RTC) used on the DS87C530 is undefined following a no–battery reset or battery attach. The RTC oscillator must be enabled by setting the RTCE bit (RTCC.0) for the RTC to work.

The RTC is guaranteed to a minimum accuracy of +/– 2 minutes per month over the rated temperature and voltage specifications. If the time is found to be less accurate than this, it is most likely due to the selection of crystal. Make sure that the RTC crystal is 32.768 kHz, and either 12.5 pF or 6 pF capacitance. The $12/\overline{6}$ bit (TRIM.6) setting should correspond to the crystal in use. Unlike other crystals, external load capacitors should not be used with the RTC. These will seriously distort the accuracy of the clock. Additional information on design considerations with the RTC can be found in Applications Note 79, Using the DS87C530 Real Time Clock.

SERIAL PORT DOES NOT WORK

The serial port is not a complicated peripheral, but there are many elements that need to be initialized. The following checklist is provided to help in debugging.

1. Have the appropriate port latch bits (P3.0, P3.1, P1.2, or P1.3) been set to 1 to enable the serial port functions?
2. Has the correct timebase been selected? (4 clocks per tick or 12 clocks per tick)
3. Is the appropriate timer reload value loaded?
4. Is the appropriate timer mode selected?
5. Is the appropriate timer running by setting TR0, TR1, or TR2 bits?
6. Is the correct serial port mode selected?
7. If desired, is the serial port doubler bit, SMOD, set? (PCON.7 or WDCON.7)
8. If desired, is the receive enable bit (REN_0 or REN_1) set?
9. Is the serial port interrupt enabled?
10. Is the global interrupt enable bit set?

HIGH–SPEED MICROCONTROLLER DOES NOT WORK IN EXISTING 8051 DESIGN

Although the High Speed Microcontroller Family was designed as a drop in replacement for the 8051 family, occasionally a developer may notice problems when inserting into an existing design. Often these problems are related to slow memory interfaces which cannot keep up with the increased throughput of the faster microcontroller. In addition, software timing loops will run faster, possibly changing program operation. These and other effects are described in Application Note 56 (The DS80C320 as a Drop–In Replacement for the 8032). Application Note 57 (DS80C320 Memory Interface Timing) discusses memory interface timing for the DS80C320. Application Note 89 (High–Speed Micro Memory Interface Timing) discusses interfacing other members of the High–Speed Microcontroller family to external memory.