Analog Devices Inc./Maxim Integrated - DS80C390-FCR+ Datasheet

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	8051
Core Size	8-Bit
Speed	40MHz
Connectivity	CANbus, EBI/EMI, SIO, UART/USART
Peripherals	Power-Fail Reset, WDT
Number of I/O	32
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	3.85V ~ 5.5V
Data Converters	-
Oscillator Type	External
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	64-LQFP
Supplier Device Package	64-LQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/analog-devices/ds80c390-fcr

AC ELECTRICAL CHARACTERISTICS—(MULTIPLEXED ADDRESS/DATA BUS) (Note 10, Note 11)

PARAMETER	SYMBOL	CONDITIONS	40MHz		VARIABLE CLOCK		UNITS
			MIN	MAX	MIN	MAX	
Oscillator Frequency	$1 / t_{CLCL}$	External oscillator	0	40	0	40	MHz
		External crystal	1	40	1	40	
ALE Pulse Width	t_{LHLL}				$0.375\ t_{MCS} - 5$		ns
Port 0 Instruction Address or $\overline{CE0–4}$ Valid to ALE Low	t_{AVLL}				$0.125\ t_{MCS} - 5$		ns
Address Hold After ALE Low	t_{LLAX1}				$0.125\ t_{MCS} - 5$		ns
ALE Low to Valid Instruction In	t_{LLIV}					$0.625\ t_{MCS} - 20$	ns
ALE Low to $\overline{PSEN}$ Low	t_{LLPL}				$0.125\ t_{MCS} - 5$		ns
$\overline{PSEN}$ Pulse Width	t_{PLPH}				$0.5\ t_{MCS} - 8$		ns
$\overline{PSEN}$ Low to Valid Instruction In	t_{PLIV}					$0.5\ t_{MCS} - 20$	ns
Input Instruction Hold After $\overline{PSEN}$	t_{PXIX}		0		0		ns
Input Instruction Float After $\overline{PSEN}$	t_{PXIZ}					$0.25\ t_{MCS} - 5$	ns
Port 0 Address to Valid Instruction In	t_{AVIV1}					$0.75\ t_{MCS} - 22$	ns
Port 2, 4 Address to Valid Instruction In	t_{AVIV2}					$0.875\ t_{MCS} - 30$	ns
$\overline{PSEN}$ Low to Address Float	t_{PLAZ}			0		0	ns

Note 11: All parameters apply to both commercial and industrial temperature operation unless otherwise noted. The value t_{MCS} is a function of the machine cycle clock in terms of the processor's input clock frequency. These relationships are described in the *Stretch Value Timing* table. All signals characterized with load capacitance of 80pF except Port 0, ALE, $\overline{PSEN}$, $\overline{RD}$, and $\overline{WR}$ with 100pF. Interfacing to memory devices with float times (turn off times) over 25ns can cause bus contention. This does not damage the parts, but causes an increase in operating current. Specifications assume a 50% duty cycle for the oscillator. Port 2 and ALE timing changes in relation to duty cycle variation. Some AC timing characteristic drawings contain references to the CLK signal. This waveform is provided to assist in determining the relative occurrence of events and cannot be used to determine the timing of signals relative to the external clock. AC timing is characterized and guaranteed by design but is not production tested.

Figure 2. Multiplexed 9-Cycle Address/Data $\overline{CE0\text{-}3}$ MOVX Read/Write Operation

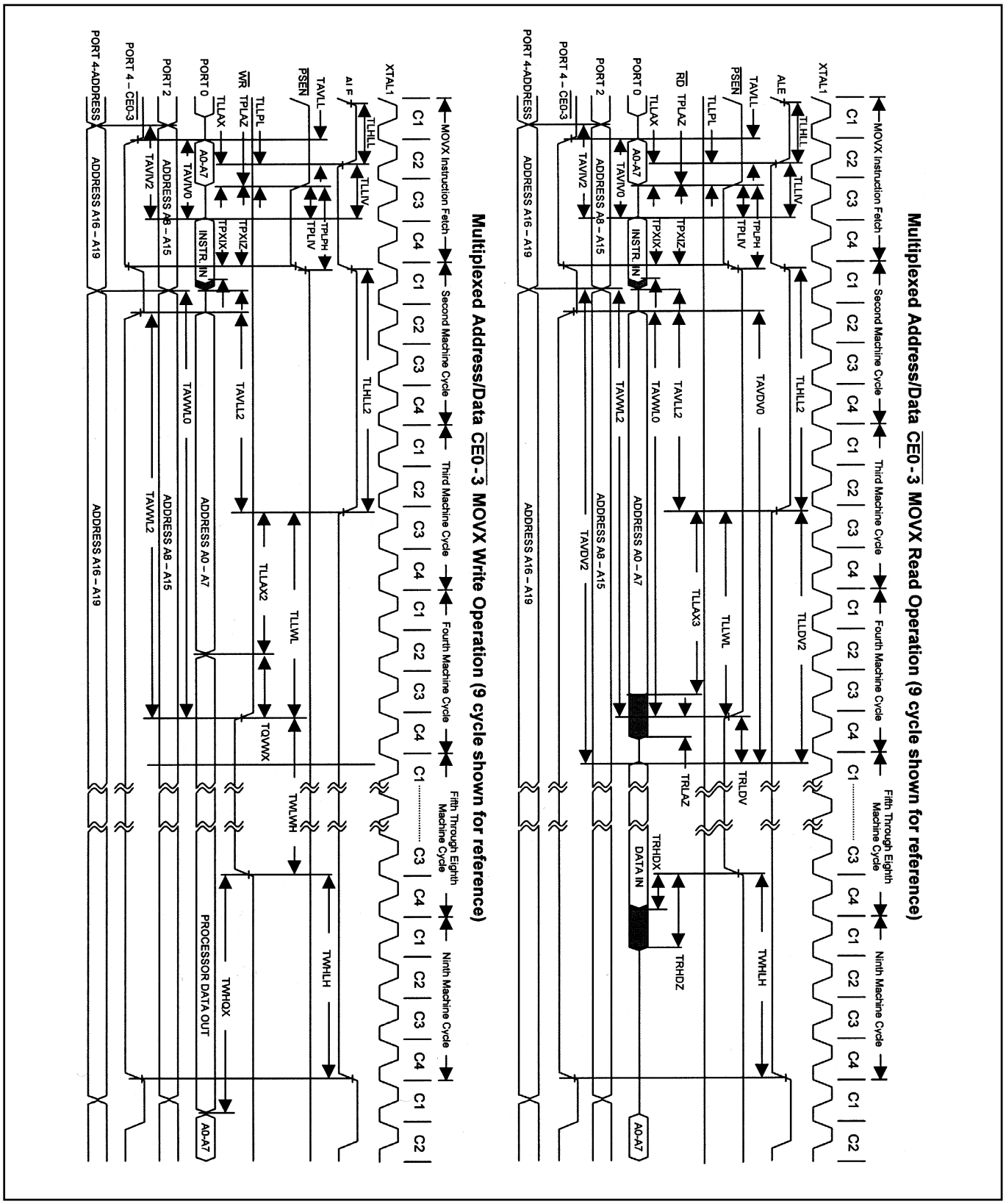


Figure 8. Multiplexed 3-Cycle Data Memory $\overline{CE0\text{-}3}$ Read

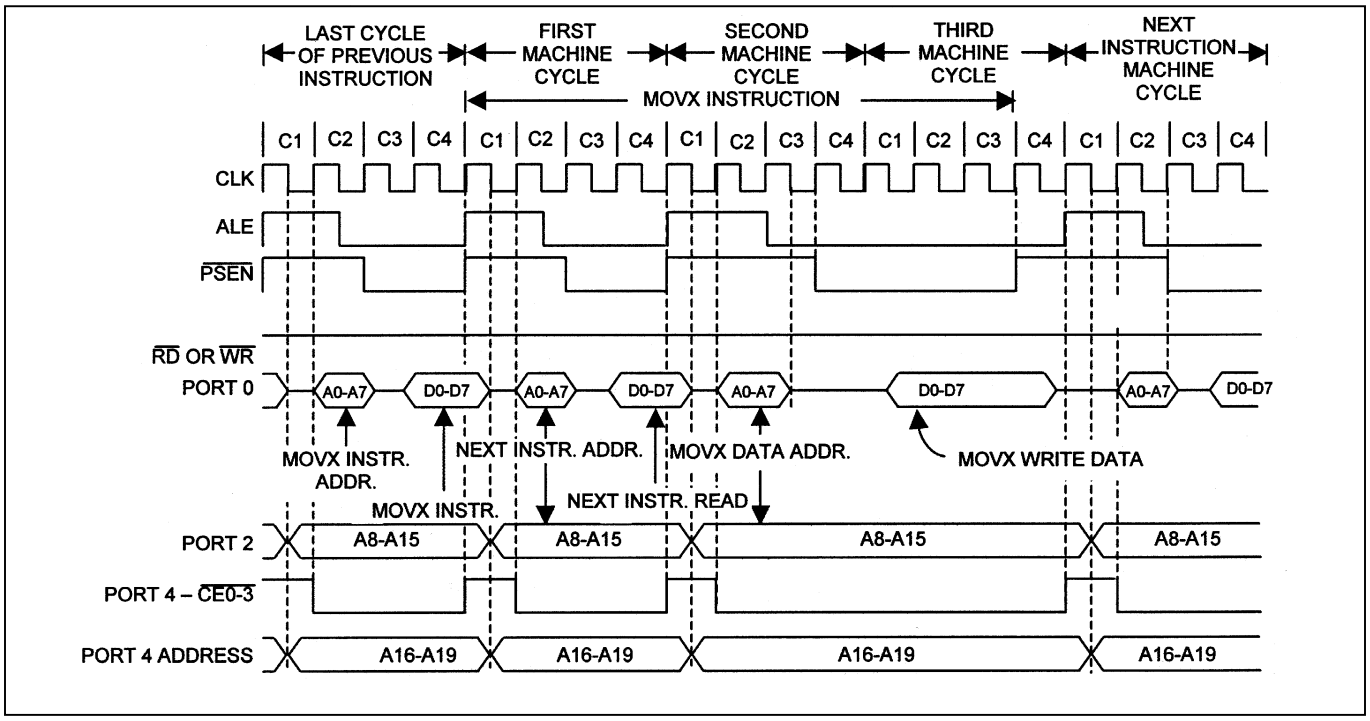
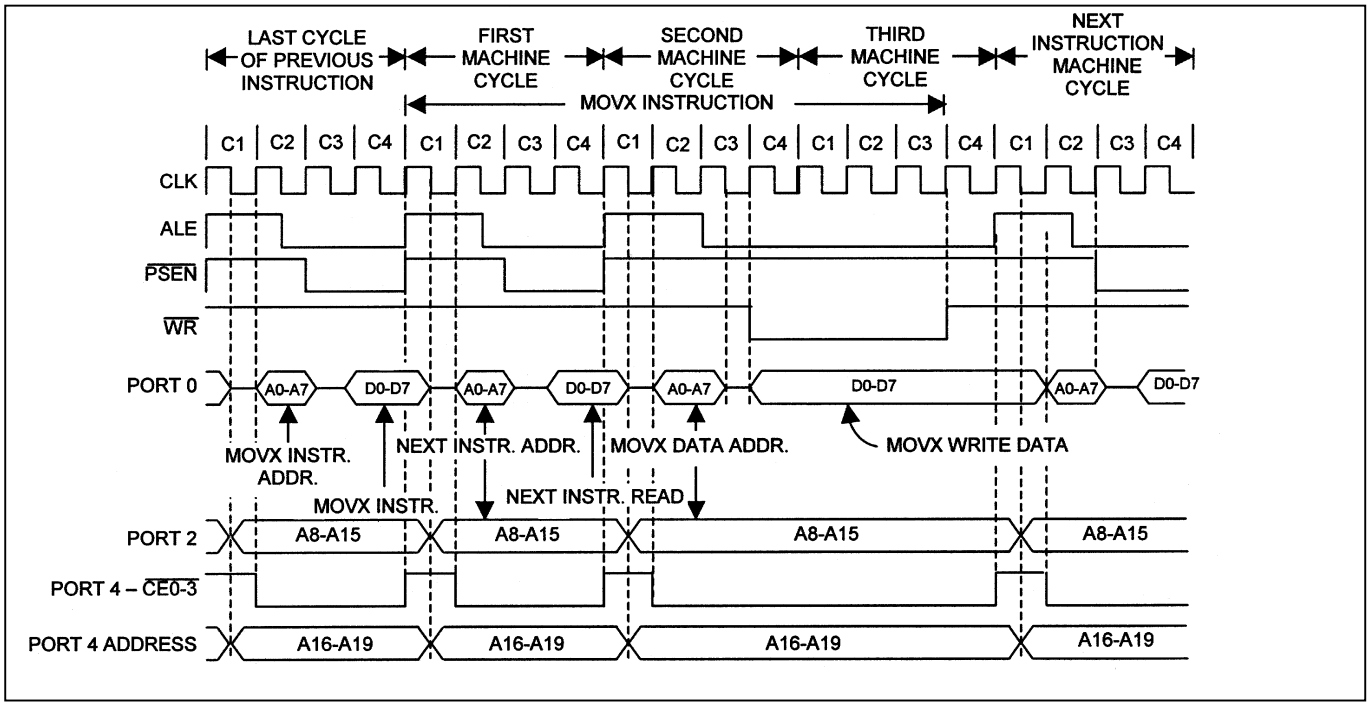


Figure 9. Multiplexed 3-Cycle Data Memory $\overline{CE0\text{-}3}$ Write



ELECTRICAL CHARACTERISTICS—(NONMULTIPLEXED ADDRESS/DATA BUS) (Note 13)

PARAMETER	SYMBOL	CONDITIONS	40MHz		VARIABLE CLOCK		UNITS
			MIN	MAX	MIN	MAX	
Oscillator Frequency	$1 / t_{CLCL}$	External oscillator	0	40	0	40	MHz
		External crystal	1	40	1	40	
$\overline{PSEN}$ Pulse Width	t_{PLPH}				$0.5\ t_{MCS} - 8$		ns
$\overline{PSEN}$ Low to Valid Instruction In	t_{PLIV}					$0.5\ t_{MCS} - 20$	ns
Input Instruction Hold After $\overline{PSEN}$	t_{PXIX}		0		0		ns
Input Instruction Float After $\overline{PSEN}$	t_{PXIZ}					See MOVX Characteristics	ns
Port 1 Address, Port 4 CE to Valid Instruction In	t_{AVIV1}					$0.75\ t_{MCS} - 22$	ns
Port 2, 4 Address to Valid Instruction In	t_{AVIV2}					$0.875\ t_{MCS} - 30$	ns

Note 13: All parameters apply to both commercial and industrial temperature operation unless otherwise noted. The value t_{MCS} is a function of the machine cycle clock in terms of the processor's input clock frequency. These relationships are described in the *Stretch Value Timing* table. All signals characterized with load capacitance of 80pF except Port 0, ALE, $\overline{PSEN}$, $\overline{RD}$, and $\overline{WR}$ with 100pF. Interfacing to memory devices with float times (turn off times) over 25ns can cause bus contention. This does not damage the parts, but causes an increase in operating current. Specifications assume a 50% duty cycle for the oscillator. Port 2 and ALE timing changes in relation to duty cycle variation. Some AC timing characteristic drawings contain references to the CLK signal. This waveform is provided to assist in determining the relative occurrence of events and cannot be used to determine the timing of signals relative to the external clock.

Figure 13. Nonmultiplexed External Program Memory Read Cycle

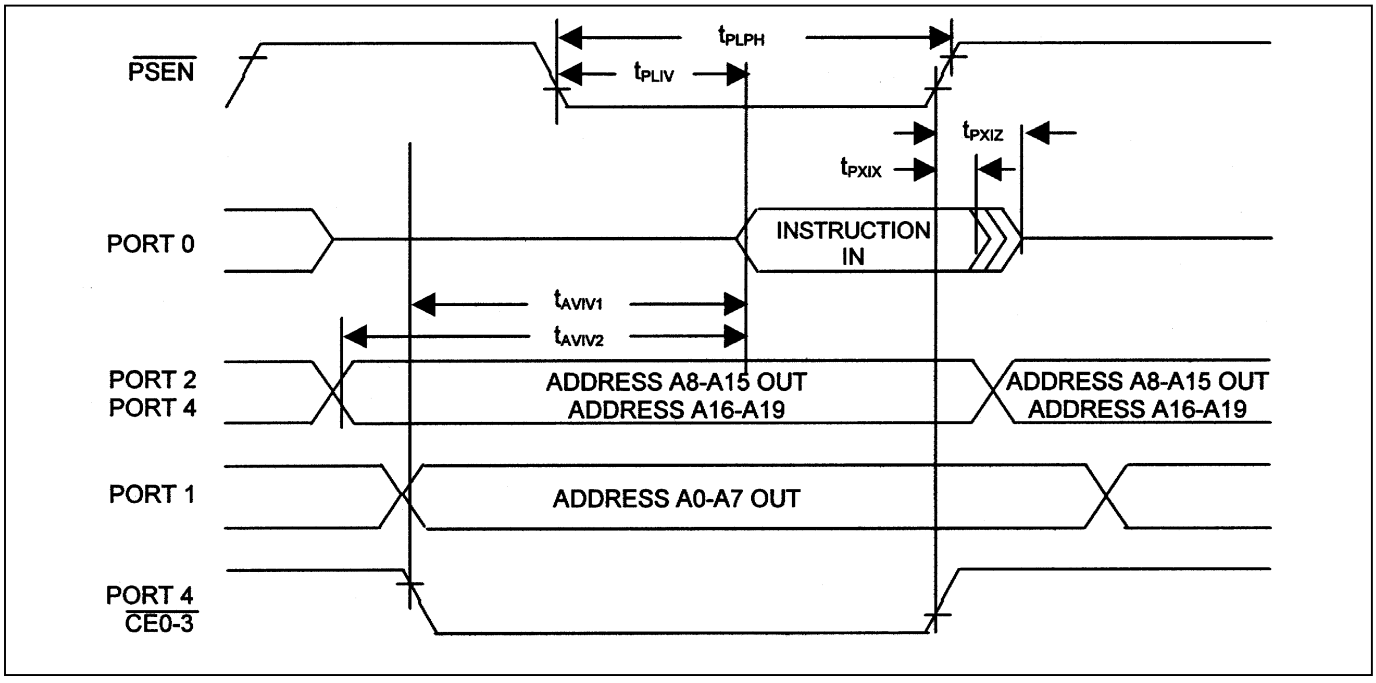


Figure 20. Nonmultiplexed 3-Cycle Data Memory $\overline{CE0\text{-}3}$ Read

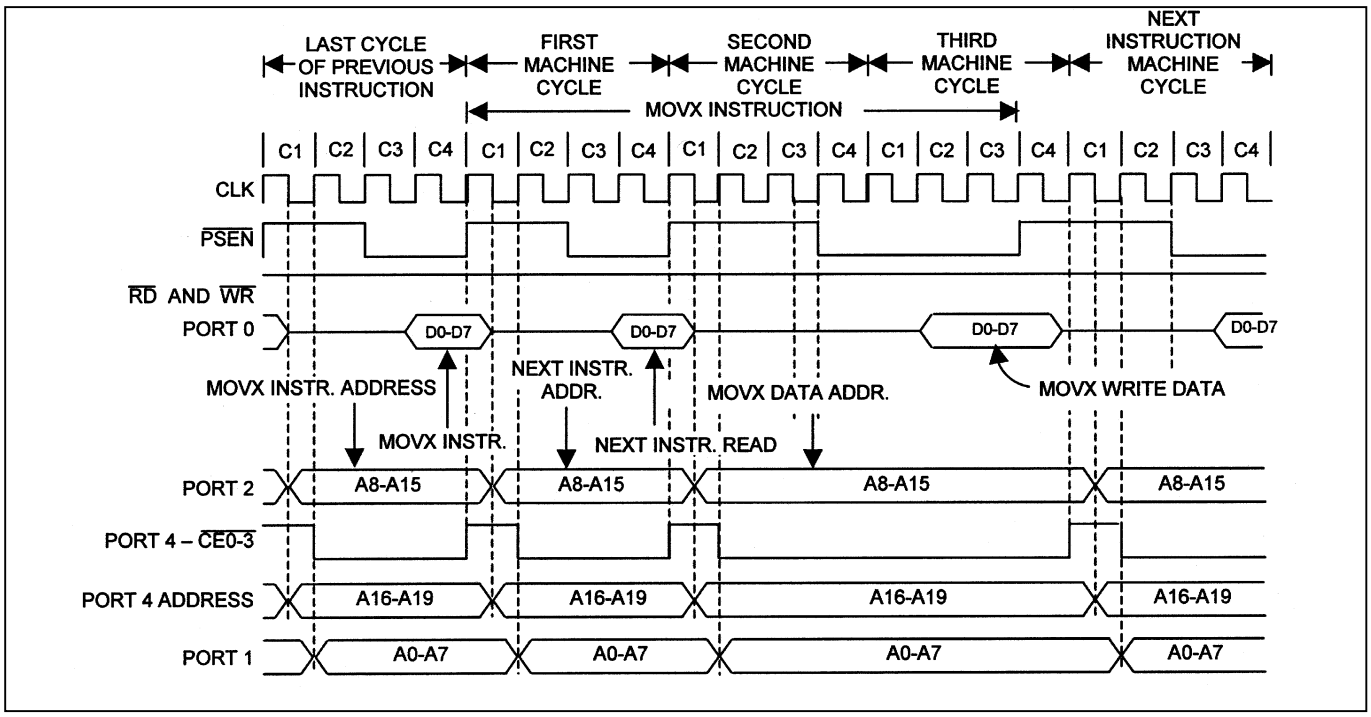


Figure 21. Nonmultiplexed 3-Cycle Data Memory $\overline{CE0\text{-}3}$ Write

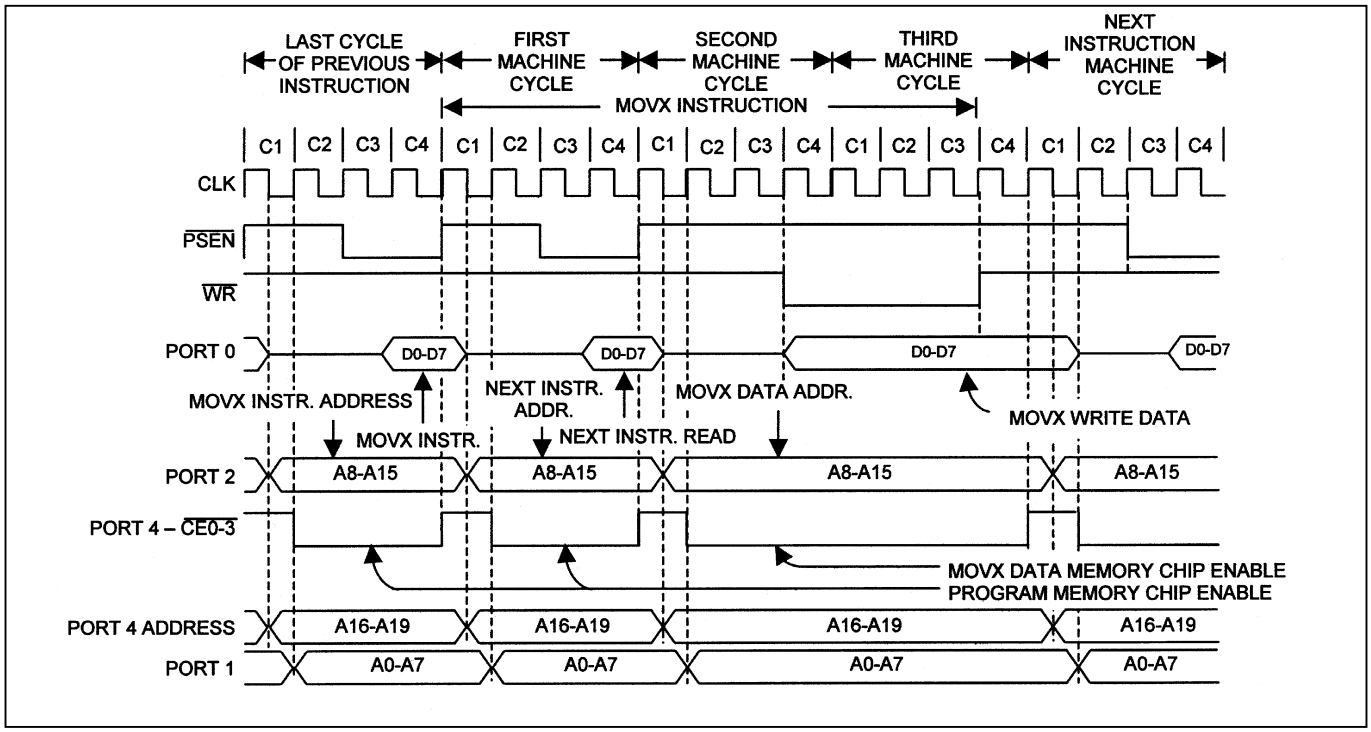


Figure 22. Nonmultiplexed 9-Cycle Data Memory $\overline{\text{PCE0-3}}$ Read or Write

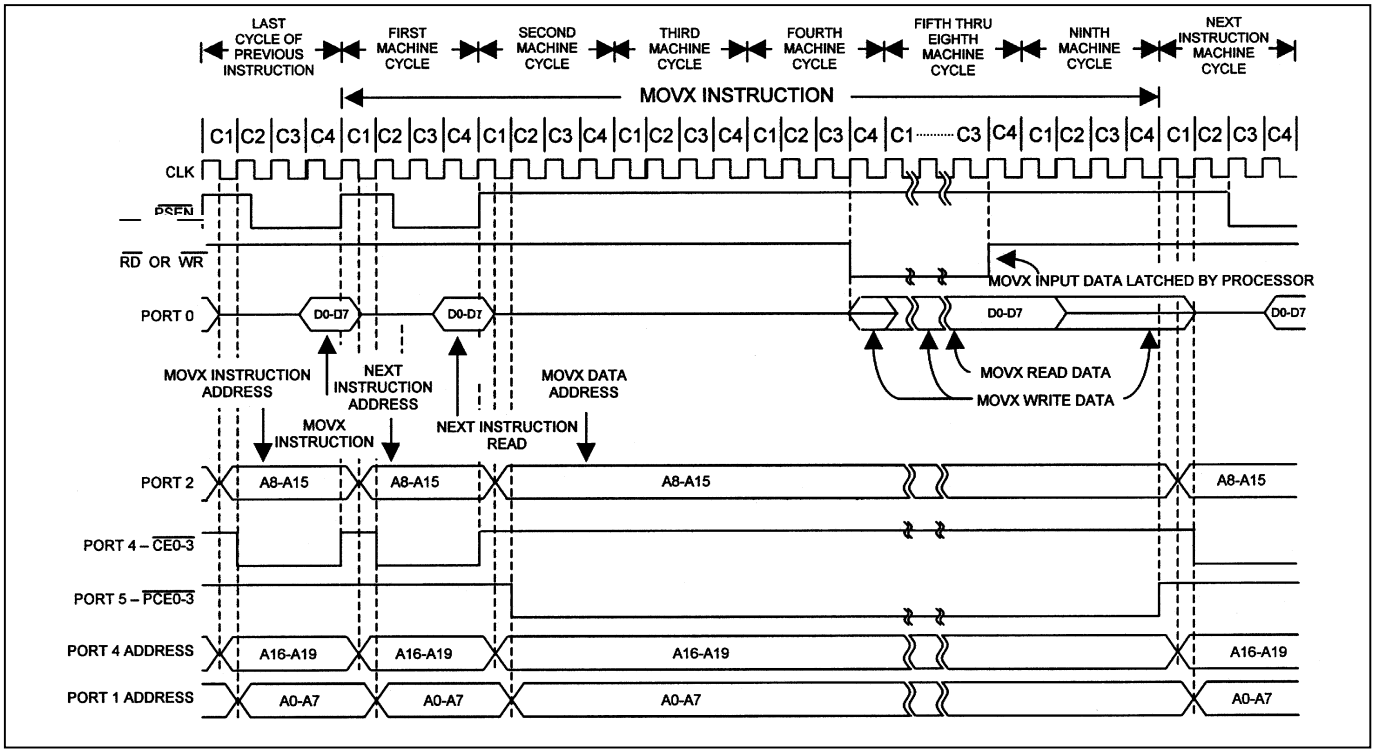


Figure 23. Nonmultiplexed 9-Cycle Data Memory $\overline{\text{CE0-3}}$ Read

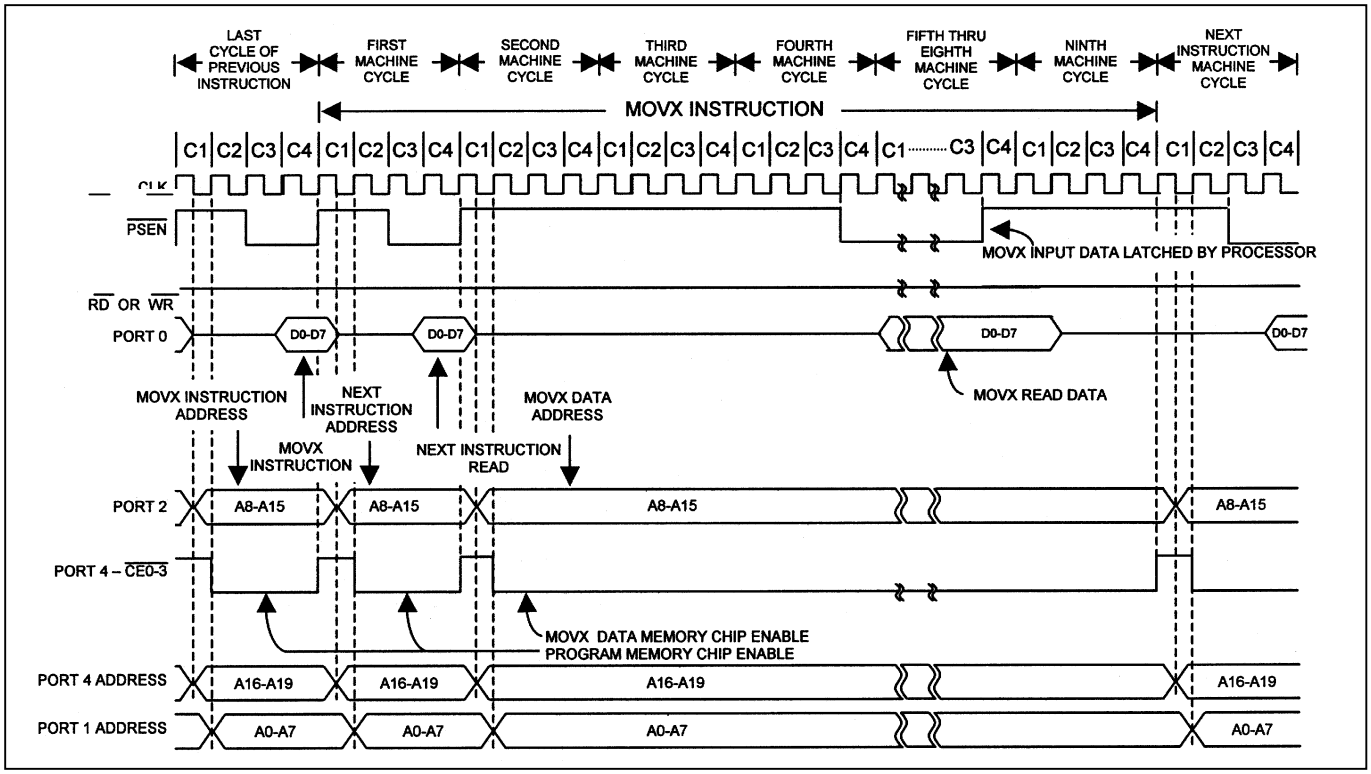
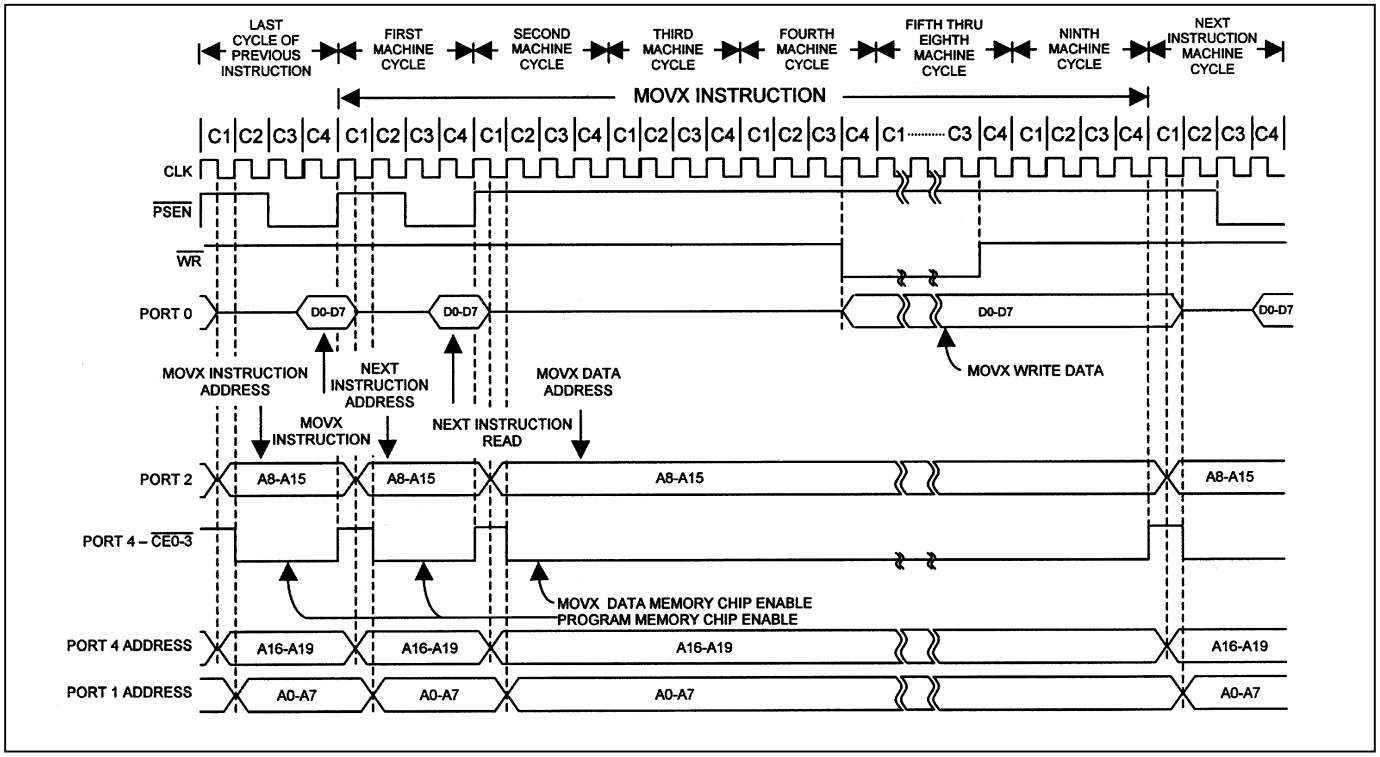


Figure 24. Nonmultiplexed 9-Cycle Data Memory $\overline{CE0\text{-}3}$ Write



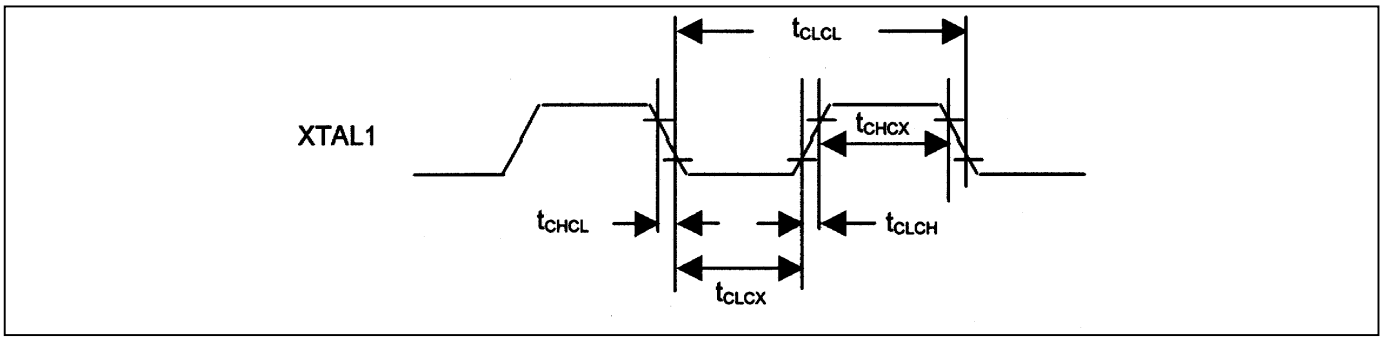
t_{MCS} TIME PERIODS

SYSTEM CLOCK SELECTION			t_{MCS}
4X/$\overline{2X}$	CD1	CD0	
1	0	0	1 t_{CLCL}
0	0	0	2 t_{CLCL}
X	1	0	4 t_{CLCL}
X	1	1	1024 t_{CLCL}

EXTERNAL CLOCK CHARACTERISTICS

PARAMETER	SYMBOL	MIN	MAX	UNITS
Clock High Time	t_{CHCX}	8		ns
Clock Low Time	t_{CLCX}	8		ns
Clock Rise Time	t_{CLCH}		4	ns
Clock Fall Time	t_{CHCL}		4	ns

Figure 25. External Clock Drive



SERIAL PORT MODE 0 TIMING CHARACTERISTICS

PARAMETER	SYMBOL	CONDITIONS	TYP	UNITS
Serial Port Clock Cycle Time	t_{XLXL}	SM2 = 0:2 clocks per cycle	12 t_{CLCL}	ns
		SM2 = 1:4 clocks per cycle	4 t_{CLCL}	
Output Data Setup to Clock Rising	t_{QVXH}	SM2 = 0:12 clocks per cycle	10 t_{CLCL}	ns
		SM2 = 1:4 clocks per cycle	3 t_{CLCL}	
Output Data Hold from Clock Rising	t_{XHQX}	M2 = 0:12 clocks per cycle	2 t_{CLCL}	ns
		SM2 = 1:4 clocks per cycle	t_{CLCL}	
Input Data Hold After Clock Rising	t_{XHDX}	SM2 = 0:12 clocks per cycle	t_{CLCL}	ns
		SM2 = 1:4 clocks per cycle	0	
Clock Rising Edge to Input Data Valid	t_{XHDV}	SM2 = 0:12 clocks per cycle	11 t_{CLCL}	ns
		SM2 = 1:4 clocks per cycle	2 t_{CLCL}	

Because the device runs the standard 8051 instruction set, in general, software written for existing 80C32-based systems will work on the DS80C390. The primary exceptions are related to timing-critical issues, since the high-performance core of the microcontroller executes instructions much faster than the original. Memory interfacing is performed identically to the standard 80C32. The high-speed nature of the DS80C390 core slightly changes the interface timing, and designers are advised to consult the timing diagrams in this data sheet for more information.

The DS80C390 provides the same timer/counter resources, full duplex serial port, 256 bytes of scratchpad RAM and I/O ports as the standard 80C32. Timers default to a 12 clocks-per-machine cycle operation to keep timing compatible with original 8051 systems, but can be programmed to run at the faster four clocks-per-machine cycle if desired. New hardware functions are accessed using special function registers that do not overlap with standard 80C32 locations.

This data sheet provides only a summary and overview of the DS80C390. Detailed descriptions are available in the *High-Speed Microcontroller User's Guide: DS80C390 Supplement*. This data sheet assumes a familiarity with the architecture of the standard 80C32. In addition to the basic features of that device, the DS80C390 incorporates many new features.

PERFORMANCE OVERVIEW

The DS80C390's higher performance comes not just from increasing the clock frequency but also from a more efficient design. This updated core removes the dummy memory cycles that are present in a standard, 12 clocks-per-machine cycle 8051. In the DS80C390, the same machine cycle takes 4 clocks. Thus the fastest instruction, one machine cycle, executes three times faster for the same crystal frequency. The majority of instructions on the DS80C390 see the full 3-to-1 speed improvement, while a few execute between 1.5 and 2.4 times faster. Regardless of specific performance improvements, all instructions are faster than the original 8051.

Improvement of individual programs depends on the actual mix of instructions used. Speed-sensitive applications should make the most use of instructions that are three times faster. However, the large number of 3-to-1 improved op codes makes dramatic speed improvements likely for any arbitrary combination of instructions. These architecture improvements and the submicron CMOS design produce a peak instruction cycle in 100ns (10 MIPS). The dual data pointer feature also allows the user to eliminate wasted instructions when moving blocks of memory.

INSTRUCTION SET SUMMARY

All instructions perform exactly the same functions as their 8051 counterparts. Their effect on bits, flags, and other status functions is identical. However, the timing of instructions is different, both in absolute and relative number of clocks. The absolute timing of software loops can be calculated using a table in the *High-Speed Microcontroller User's Guide: DS80C390 Supplement*. However, counter/timers default to run at the traditional 12 clocks per increment. In this way, timer-based events occur at the standard intervals with software executing at higher speed. Timers optionally can run at the faster four clocks per increment to take advantage of faster processor operation.

The relative time of two DS80C390 instructions might differ from the traditional 8051. For example, in the original architecture the "MOVX A, @DPTR" instruction and the "MOV direct, direct" instruction required the same amount of time: two machine cycles or 24 oscillator cycles. In the DS80C390, the MOVX instruction takes as little as two machine cycles, or eight oscillator cycles, but the "MOV direct, direct" uses three machine cycles, or 12 oscillator cycles. While both are faster than their original counterparts, they now have different execution times. This is because the device usually uses one instruction cycle for each instruction byte. Examine the timing of each instruction 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. Refer to the *High-Speed Microcontroller User's Guide: DS80C390 Supplement* for details and individual instruction timing.

SPECIAL FUNCTION REGISTERS (SFRs)

Special function registers (SFRs) control most special features of the microcontroller, allowing the device to have many new features but use the same instruction set as the 8051. When writing software to use a new feature, an equate statement defines the SFR to an assembler or compiler. This is the only change needed to access the new function. The DS80C390 duplicates the SFRs contained in the standard 80C52. Table 1 shows the register addresses and bit locations. Many are standard 80C52 registers. The *High-Speed Microcontroller User's Guide: DS80C390 Supplement* contains a full description of all SFRs.

Table 1. SFR Locations (continued)

REGISTER	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0	ADDRESS
C0M14C	MSRDY	ETI	ERI	INTRQ	EXTRQ	MTRQ	ROW/TIH	DTUP	BEh
C0M15C	MSRDY	ETI	ERI	INTRQ	EXTRQ	MTRQ	ROW/TIH	DTUP	BFh
SCON1	SM0/FE_1	SM1_1	SM2_1	REN_1	TB8_1	RB8_1	TI_1	RI_1	C0h
SBUF1									C1h
PMR	CD1	CD0	SWB	CTM	4X/$\overline{2X}$	ALEOFF	—	—	C4h
STATUS	PIP	HIP	LIP	—	SPTA1	SPRA1	SPTA0	SPRA0	C5h
MCON	IDM1	IDM0	CMA	—	PDCE3	PDCE2	PDCE1	PDCE0	C6h
TA									C7h
T2CON	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$	C8h
T2MOD	—	—	—	D13T1	D13T2	—-	T2OE	DCEN	C9h
RCAP2L									CAh
RCAP2H									CBh
TL2									CCh
TH2									CDh
COR	IRDACK	C1BPR7	C1BPR6	C0BPR7	C0BPR6	COD1	COD0	CLKOE	CEh
PSW	CY	AC	F0	RS1	RS0	OV	F1	P	D0h
MCNT0	$\overline{LSHIFT}$	CSE	SCB	MAS4	MAS3	MAS2	MAS1	MAS0	D1h
MCNT1	MST	MOF	—	CLM	—	—	—	—	D2h
MA									D3h
MB									D4h
MC									D5h
C1RMS0									D6h
C1RMS1									D7h
WDCON	SMOD_1	POR	EPFI	PFI	WDIF	WTRF	EWT	RWT	D8h
C1TMA0									DEh
C1TMA1									DFh
ACC									E0h
C1C	ERIE	STIE	PDE	SIESTA	CRST	AUTOB	ERCS	SWINT	E3h
C1S	BSS	CECE	WKS	RXS	TXS	ER2	ER1	ER0	E4h
C1IR	INTIN7	INTIN6	INTIN5	INTIN4	INTIN3	INTIN2	INTIN1	INTIN0	E5h
C1TE									E6h
C1RE									E7h
EIE	CANBIE	C0IE	C1IE	EWDI	EX5	EX4	EX3	EX2	E8h
MXAX									EAh
C1M1C	MSRDY	ETI	ERI	INTRQ	EXTRQ	MTRQ	ROW/TIH	DTUP	EBh
C1M2C	MSRDY	ETI	ERI	INTRQ	EXTRQ	MTRQ	ROW/TIH	DTUP	ECh
C1M3C	MSRDY	ETI	ERI	INTRQ	EXTRQ	MTRQ	ROW/TIH	DTUP	EDh
C1M4C	MSRDY	ETI	ERI	INTRQ	EXTRQ	MTRQ	ROW/TIH	DTUP	EEh
C1M5C	MSRDY	ETI	ERI	INTRQ	EXTRQ	MTRQ	ROW/TIH	DTUP	EFh
B									F0h
C1M6C	MSRDY	ETI	ERI	INTRQ	EXTRQ	MTRQ	ROW/TIH	DTUP	F3h
C1M7C	MSRDY	ETI	ERI	INTRQ	EXTRQ	MTRQ	ROW/TIH	DTUP	F4h
C1M8C	MSRDY	ETI	ERI	INTRQ	EXTRQ	MTRQ	ROW/TIH	DTUP	F5h
C1M9C	MSRDY	ETI	ERI	INTRQ	EXTRQ	MTRQ	ROW/TIH	DTUP	F6h
C1M10C	MSRDY	ETI	ERI	INTRQ	EXTRQ	MTRQ	ROW/TIH	DTUP	F7h
EIP	CANBIP	C0IP	C1IP	PWDI	PX5	PX4	PX3	PX2	F8h
C1M11C	MSRDY	ETI	ERI	INTRQ	EXTRQ	MTRQ	ROW/TIH	DTUP	FBh
C1M12C	MSRDY	ETI	ERI	INTRQ	EXTRQ	MTRQ	ROW/TIH	DTUP	FCh
C1M13C	MSRDY	ETI	ERI	INTRQ	EXTRQ	MTRQ	ROW/TIH	DTUP	FDh
C1M14C	MSRDY	ETI	ERI	INTRQ	EXTRQ	MTRQ	ROW/TIH	DTUP	FEh
C1M15C	MSRDY	ETI	ERI	INTRQ	EXTRQ	MTRQ	ROW/TIH	DTUP	FFh

Note: Shaded bits are timed-access protected.

40-BIT ACCUMULATOR

The accelerator also incorporates an automatic accumulator function, permitting the implementation of multiply-and-accumulate and divide-and-accumulate functions without any additional delay. Each time the accelerator is used for a multiply or divide operation, the result is transparently added to a 40-bit accumulator. This can greatly increase speed of DSP and other high-level math operations.

The accumulator can be accessed anytime the multiply/accumulate status flag (MCNT1;D2h) is cleared. The accumulator is initialized by performing five writes to the multiplier C register (MC;D5h), LSB first. The 40-bit accumulator can be read by performing five reads of the multiplier C register, MSB first.

MEMORY ADDRESSING

The DS80C390 incorporates three internal memory areas:

- 256 bytes of scratchpad (or direct) RAM
- 4kB of SRAM configurable as various combinations of MOVX data memory, stack memory, and MOVC program memory
- 512 bytes of RAM reserved for the CAN message centers.

Up to 4MB of external memory is addressed via a multiplexed or demultiplexed 20-bit address bus/8-bit data bus and four chip-enable (active during program memory access) or four peripheral-enable (active during data memory access) signals. Three different addressing modes are supported, as selected by the AM1, AM0 bits in the ACON SFR.

16-Bit Address Mode

Memory is accessed by 16-bit address mode similarly to the traditional 8051. It is op-code compatible with the 8051 microprocessor and identical to the byte and cycle count of the Dallas Semiconductor High-Speed Microcontroller family. A device operating in this mode can access up to 64kB of program and data memory. The device defaults to this mode following any reset.

22-Bit Paged-Address Mode

The 22-bit paged-address mode retains binary-code compatibility with the 8051 instruction set, but adds one machine cycle to the ACALL, LCALL, RET, and RETI instructions with respect to Dallas Semiconductor's High-Speed Microcontroller family timing. This is transparent to standard 8051 compilers. Interrupt latency is also increased by one machine cycle. In this mode, interrupt vectors are fetched from 0000xxh.

22-Bit Contiguous Address Mode

The 22-bit contiguous addressing mode uses a full 22-bit program counter, and all modified branching instructions automatically save and restore the entire program counter. The 22-bit branching instructions such as ACALL, AJMP, LCALL, LJMP, MOV DPTR, RET, and RETI instructions require an assembler, compiler, and linker that specifically supports these features. The INC DPTR is lengthened by one cycle but remains byte-count-compatible with the standard 8051 instruction set.

Internally, the device uses a 22-bit program counter. The lowest order 22 bits are used for memory addressing, with a special 23rd bit used to map the 4kB SRAM above the 4MB memory space in bootstrap loader applications. Address bits 16–23 for the 22-bit addressing modes are generated through additional SFRs dependent on the type of instruction as shown in Table 4.

Table 4. Extended Address Generation

INSTRUCTION	ADDRESS BITS 23–16	ADDRESS BITS 15–8	ADDRESS BITS 7–0
MOVX instructions using DPTR	DPX;93h	DPH;83h	DPL;82h
MOVX instructions using DPTR1	DPX1;95h	DPH1;85h	DPL1;84h
MOVX instructions using @Ri	MXAX;EAh	P2;A0h	Ri
Addressing program memory in 22-bit paged mode	AP;9Ch	—	—
10-bit stack pointer mode	—	ESP;9Bh	SP;81h

INTERNAL MOVX SRAM

The DS80C390 contains 4kB of SRAM that can be configured as user accessible MOVX memory, program memory, or optional stack memory. The specific configuration and locations are governed by the internal data memory configuration bits (IDM1, IDM0) in the memory control register (MCON;C6h). Note that when the SA bit (ACON.2) is set, the first 1kB of the MOVX data memory is reserved for use by the 10-bit expanded stack. Internal memory accesses will not generate $\overline{WR}$, $\overline{RD}$, or $\overline{PSEN}$ strobes.

The DS80C390 can configure its 4kB of internal SRAM as combined program and data memory. This allows the application software to execute self-modifiable code. The technique loads the 4kB SRAM with bootstrap loader software, and then modifies the IDM1 and IDM0 bits to map the 4kB starting at memory location 40000h. This allows the system to run the bootstrap loader without disturbing the 4MB external memory bus, making the device in-system reprogrammable for flash or NV RAM.

Table 5. Internal MOVX SRAM Configuration

IDM1	IDM0	CMA	MEMORY		
			MOVX DATA	CAN MESSAGE	SHARED PROGRAM/DATA
0	0	0	00F000h–00FFFFh	00EE00h–00EFFFh	—
0	0	1	00F000h–00FFFFh	401000h–4011FFh	—
0	1	0	000000h–000FFFh	00EE00h–00EFFFh	—
0	1	1	000000h–000FFFh	401000h–4011FFh	—
1	0	0	400000h–400FFFh	00EE00h–00EFFFh	—
1	0	1	400000h–400FFFh	401000h–4011FFh	—
1	1	0	—	00EE00h–00EFFFh	400000h–400FFFh*
1	1	1	—	401000h–4011FFh	400000h–400FFFh*

**10-bit expanded stack is not available in shared program/data memory mode.*

EXTERNAL MEMORY ADDRESSING

The enabling and mapping of the chip-enable signals is done through the Port 4 control register (P4CNT;92h) and memory control register (MCON; 96h). Table 7 shows which chip-enable and address line signals are active on Port 4. Following reset, the device will be configured with P4.7–P4.4 as address lines and P4.3–P4.0 configured as $\overline{CE3\text{-}0}$, with the first program fetch being performed from 00000h with $\overline{CE0}$ active. The following tables illustrate which memory ranges are controlled by each chip enable as a function of which address lines are enabled.

Table 6. External Memory Addressing Pin Assignments

ADDRESS/DATA BUS	$\overline{CE3}$–$\overline{CE0}$	$\overline{PCE3}$–$\overline{PCE0}$	ADDR 19–16	ADDR 15–8	ADDR 7–0	DATA BUS
Multiplexed	P4.3–P4.0	P5.7–P5.4	P4.7–P4.4	P2	P0	P0
Demultiplexed	P4.3–P4.0	P5.7–P5.4	P4.7–P4.4	P2	P1	P0

Table 7. Extended Address and Chip-Enable Generation

P4CNT.5–3	PORT 4 PIN FUNCTION				P4CNT.2–0	PORT 4 PIN FUNCTION			
	P4.7	P4.6	P4.5	P4.4		P4.3	P4.2	P4.1	P4.0
000	I/O	I/O	I/O	I/O	000	I/O	I/O	I/O	I/O
100	I/O	I/O	I/O	A16	100	I/O	I/O	I/O	$\overline{CE0}$
101	I/O	I/O	A17	A16	101	I/O	I/O	$\overline{CE1}$	$\overline{CE0}$
110	I/O	A18	A17	A16	110	I/O	$\overline{CE2}$	$\overline{CE1}$	$\overline{CE0}$
111(default)	A19	A18	A17	A16	111(default)	$\overline{CE3}$	$\overline{CE2}$	$\overline{CE1}$	$\overline{CE0}$

Table 8. Program Memory Chip-Enable Boundaries

P4CNT.5–3	$\overline{CE0}$	$\overline{CE1}$	$\overline{CE2}$	$\overline{CE3}$
000	0h–7FFFh	8000h–FFFFh	10000h–17FFFh	18000h–1FFFFh
100	0h–1FFFFh	20000h–3FFFFh	40000h–5FFFFh	60000h–7FFFFh
101	0h–3FFFFh	40000h–7FFFFh	80000h–BFFFFh	C0000h–FFFFFh
110	0h–7FFFFh	80000h–FFFFFh	100000h–17FFFFh	180000h–1FFFFFh
111(default)	0–FFFFFh	100000h–1FFFFFh	200000h–2FFFFFh	300000h–3FFFFFh

The DS80C390 incorporates a feature allowing PCE and CE signals to be combined. This is useful when incorporating modifiable code memory as part of a bootstrap loader or for in-system reprogrammability. Setting the $\overline{PDCE3–0}$ (MCON.3–0) bits causes the corresponding chip-enable signal to function for both MOVC and MOVX operations. Write access to combined program and data memory blocks is controlled by the $\overline{WR}$ signal, and read access is controlled by the $\overline{PSEN}$ signal. This feature is especially useful if the design achieves in-system reprogrammability via external flash memory, in which a single device is accessed through both MOVC instructions (program fetch) and MOVX write operations (updates to code memory). In this case, the internal SRAM is placed in the program/data configuration and loaded with a small bootstrap loader program stored in the external flash memory. The device then executes the internal bootstrap loader routine to modify/update the program memory located in the external flash memory.

STRETCH MEMORY CYCLES

The DS80C390 allows user-application software to select the number of machine cycles it takes to execute a MOVX instruction, allowing access to both fast and slow off-chip data memory and/or peripherals without glue logic. High-speed systems often include memory-mapped peripherals such as LCDs or UARTs with slow access times, so it may not be necessary or desirable to access external devices at full speed. The microprocessor can perform a MOVX instruction in as little as two machine cycles or as many as twelve machine cycles. Accesses to internal MOVX SRAM always use two cycles. Note that stretch cycle settings affect external MOVX memory operations only and that there is no way to slow the accesses to program memory other than to use a slower crystal (or external clock).

External MOVX timing is governed by the selection of 0 to 7 stretch cycles, controlled by the MD2–MD0 SFR bits in the clock-control register (CKCON.2–0). A stretch of zero results in a 2-machine cycle MOVX instruction. A stretch of seven results in a MOVX of 12 machine cycles. Software can dynamically change the stretch value depending on the particular memory or peripheral being accessed. The default of one stretch cycle allows the use of commonly available SRAMs without dramatically lengthening the memory access times.

Stretch cycle settings affect external MOVX timing in three gradations. Changing the stretch value from 0 to 1 adds an additional clock cycle each to the data setup and hold times. When a stretch value of 4 or above is selected, the interface timing changes dramatically to allow for very slow peripherals. First, the ALE signal is lengthened by 1 machine cycle. This increases the address setup time into the peripheral by this amount. Next, the address is held on the bus for one additional machine cycle increasing the address hold time by this amount. The $\overline{WR}$ and $\overline{RD}$ signals are then lengthened by a machine cycle. Finally, during a MOVX write the data is held on the bus for one additional machine cycle, thereby increasing the data hold time by this amount. For every stretch value greater than 4, the setup and hold times remain constant, and only the width of the read or write signal is increased. These three gradations are reflected in the *AC Electrical Characteristics*, where the eight MOVX timing specifications are represented by only three timing diagrams.

The reset default of one stretch cycle results in a three-cycle MOVX for any external access. Therefore, the default off-chip RAM access is not at full speed. This is a convenience to existing designs that use slower RAM. When maximum speed is desired, software should select a stretch value of zero. When using very slow RAM or peripherals, the application software can select a larger stretch value.

The specific timing of MOVX instructions as a function of stretch settings is provided in the *Electrical Specifications* section of this data sheet. As an example, Table 9 shows the read and write strobe widths corresponding to each stretch value.

POWER MANAGEMENT MODE (PMM) AND SWITCHBACK

Power consumption in PMM is less than in idle mode, and approximately one quarter of that consumed in divide-by-four mode. While PMM and Idle modes leave the power-hungry internal timers running, PMM runs all clocked functions such as timers at the rate of crystal divided by 1024, rather than crystal divided by 4. Even though instruction execution continues in PMM (albeit at a reduced speed), it still consumes less power than idle mode. As a result there is little reason to use idle mode in new designs.

When enabled, the switchback feature allows serial ports and interrupts to automatically switch back from divide by 1024 (PMM) to divide-by-4 (standard speed) operation. This feature makes it very convenient to use the PMM in real-time applications. Software can simply set the CD1 and CD0 clock control bits to the 4 clocks-per-cycle mode to exit PMM. However, the microcontroller provides hardware alternatives for automatic Switchback to standard speed (divide-by-4) operation.

Setting the SFR bit SWB (PMR.5) to 1 enables the switchback feature. Once it is enabled, and when PMM is selected, two possible events can cause an automatic switchback to divide-by-4 mode. First, if an interrupt occurs and is acknowledged, the system clock reverts from PMM to divide-by-4 mode. For example, if $\overline{\text{INT0}}$ is enabled and the CPU is not servicing a higher priority interrupt, then switchback occurs on $\overline{\text{INT0}}$. However, if $\overline{\text{INT0}}$ is not enabled or the CPU is servicing a higher priority interrupt, then activity on $\overline{\text{INT0}}$ does not cause switchback to occur.

A switchback can also occur when an enabled UART detects the start bit indicating the beginning of an incoming serial character or when the SBUF register is loaded initiating a serial transmission. Note that a serial character's start bit does not generate an interrupt. The interrupt occurs only on reception of a complete serial word. The automatic switchback on detection of a start bit allows timer hardware to return to divide-by-4 operation (and the correct baud rate) in time for a proper serial reception or transmission. So with switchback enabled and a serial port enabled, the automatic switch to divide-by-4 operation occurs in time to receive or transmit a complete serial character as if nothing special had happened.

STATUS

The status register (STATUS;C5h) provides information about interrupt and serial port activity to assist in determining if it is possible to enter PMM. The microprocessor supports three levels of interrupt priority: power-fail, high, and low. The PIP (power-fail priority interrupt status; STATUS.7), HIP (high-priority interrupt status; STATUS.6), and LIP (low-priority interrupt status; STATUS.5) status bits, when set to logic 1, indicate the corresponding level is in service.

Software should not rely on a lower-priority level interrupt source to remove PMM (switchback) when a higher level is in service. Check the current priority service level before entering PMM. If the current service level locks out a desired switchback source, then it would be advisable to wait until this condition clears before entering PMM. Alternately, software can prevent an undesired exit from PMM by intentionally entering a low priority interrupt service level before entering PMM. This will prevent other low priority interrupts from causing a switchback.

Entering PMM during an ongoing serial port transmission or reception can corrupt the serial port activity. To prevent this, a hardware lockout feature ignores changes to the clock divisor bits while the serial ports are active. Serial port activity can be monitored via the serial port activity bits located in the status register.

IDLE MODE

Setting the IDLE bit (PCON.0) invokes the idle mode. Idle leaves internal clocks, serial ports, and timers running. Power consumption drops because memory is not being accessed and instructions are not being executed. Since clocks are running, the idle power consumption is a function of crystal frequency. It should be approximately one-half of the operational power at a given frequency. The CPU can exit idle mode with any interrupt or a reset. Because PMM consumes less power than idle mode, as well as leaving timers and CPU operating, idle mode is no longer recommended for new designs, and is included for backward software compatibility only.

The SCON0 register provides control for serial port 0 while its I/O buffer is SBUF0. The registers SCON1 and SBUF1 provide the same functions for the second serial port. A full description of the use and operation of both serial ports can be found in the *High-Speed Microcontroller User's Guide: DS80C390 Supplement*.

WATCHDOG TIMER

The watchdog is a free-running, programmable timer that can set a flag, cause an interrupt, and/or reset the microcontroller if allowed to reach a preselected timeout. It can be restarted by software.

A typical application uses the watchdog timer as a reset source to prevent software from losing control. The watchdog timer is initialized, selecting the timeout period and enabling the reset and/or interrupt functions. After enabling the reset function, software must then restart the timer before its expiration or the hardware will reset the CPU. In this way, if the code execution goes awry and software does not reset the watchdog as scheduled, the processor is put in a known good state: reset.

Software can select one of four timeout values as controlled by the WD1 and WD0 bits. Timeout values are precise since they are a function of the crystal frequency. When the watchdog times out, it sets the watchdog timer-reset flag (WTRF = WDCON.2), which generates a reset if enabled by the enable watchdog-timer reset (EWT = WDCON.1) bit. Both the enable watchdog-timer reset and the reset watchdog timer control bits are protected by timed-access circuitry. This prevents errant software from accidentally clearing or disabling the watchdog.

The watchdog interrupt is useful for systems that do not require a reset circuit. It set the WDIF (watchdog interrupt) flag 512 clocks before setting the reset flag. Software can optionally enable this interrupt source, which is independent of the watchdog-reset function. The interrupt is commonly used during the debug process to determine where watchdog-reset commands must be located in the application software. The interrupt also can serve as a convenient time base generator or can wake up the processor from power-saving modes.

The clock control (CKCON) and the watchdog control (WDCON) SFRs control the watchdog timer. CKCON.7 and CKCON.6 (WD1 and WD0, respectively) select the watchdog timeout period. Of course, the 4X/$\overline{2X}$ (PMR.3) and CD1:0 (PMR.7:6) system clock-control bits also affect the timeout period. Table 12 shows the timeout selection.

Table 12. Watchdog Timeout Values

4X/$\overline{2X}$	CD1:0	WATCHDOG INTERRUPT TIMEOUT				WATCHDOG RESET TIMEOUT			
		WD1:0 = 00	WD1:0 = 01	WD1:0 = 10	WD1:0 = 11	WD1:0 = 00	WD1:0 = 01	WD1:0 = 10	WD1:0 = 11
1	00	2^{15}	2^{18}	2^{21}	2^{24}	2^{15}+512	2^{18}+512	2^{21}+512	2^{24}+512
0	00	2^{16}	2^{19}	2^{22}	2^{25}	2^{16}+512	2^{19}+512	2^{22}+512	2^{25}+512
x	01	2^{17}	2^{20}	2^{23}	2^{26}	2^{17}+512	2^{20}+512	2^{23}+512	2^{26}+512
x	10	2^{17}	2^{20}	2^{23}	2^{26}	2^{17}+512	2^{20}+512	2^{23}+512	2^{26}+512
x	11	2^{25}	2^{28}	2^{31}	2^{34}	2^{25}+512	2^{28}+512	2^{31}+512	2^{34}+512

Table 12 demonstrates that for a 33MHz crystal frequency, the watchdog timer can produce timeout periods from 3.97ms (2^{17} x 1/33MHz) to over 2 seconds (2.034 = 2^{26} x 1/33MHz) with the default setting of CD1:0 (=10). This wide variation in timeout periods allows very flexible system implementation.

In a typical initialization, the user selects one of the possible counter values to determine the timeout. Once the counter chain has completed a full count, hardware sets the interrupt flag (WDIF = WDCON.3). Regardless of whether the software makes use of this flag, there are then 512 clocks left until the reset flag (WTRF = WDCON.2) is set. Software can enable (1) or disable (0) the reset using the enable watchdog-timer-reset (EWT = WDCON.1) bit.

Table 13. Interrupt Summary

NAME	DESCRIPTION	VECTOR	NATURAL PRIORITY	FLAG BIT	ENABLE BIT	PRIORITY CONTROL BIT
PFI	Power-Fail Interrupt	33h	0	PFI (WDCON.4)	EPFI (WDCON.5)	N/A
INT0	External Interrupt 0	03h	1	IE0 (TCON.1)**	EX0 (IE.0)	PX0 (IP.0)
TF0	Timer 0	0Bh	2	TF0 (TCON.5)*	ET0 (IE.1)	PT0 (IP.1)
INT1	External Interrupt 1	13h	3	IE1 (TCON.3)**	EX1 (IE.2)	PX1 (IP.2)
TF1	Timer 1	1Bh	4	TF1 (TCON.7)*	ET1 (IE.3)	PT1 (IP.3)
SCON0	TI0 or RI0 from Serial Port 0	23h	5	RI_0 (SCON0.0); TI_0 (SCON0.1)	ES0 (IE.4)	PS0 (IP.4)
TF2	Timer 2	2Bh	6	TF2 (T2CON.7)	ET2 (IE.5)	PT2 (IP.7)
SCON1	TI1 or RI1 from Serial Port 1	3Bh	7	RI_1 (SCON1.0); TI_1 (SCON1.1)	ES1 (IE.6)	PS1 (IP.6)
INT2	External Interrupt 2	43h	8	IE2 (EXIF.4)	EX2 (EIE.0)	PX2 (EIP.0)
INT3	External Interrupt 3	4Bh	9	IE3 (EXIF.5)	EX3 (EIE.1)	PX3 (EIP.1)
INT4	External Interrupt 4	53h	10	IE4 (EXIF.6)	EX4 (EIE.2)	PX4 (EIP.2)
INT5	External Interrupt 5	5Bh	11	IE5 (EXIF.7)	EX5 (EIE.3)	PX5 (EIP.3)
C0I	CAN0 Interrupt	6Bh	12	various	C0IE (EIE.6)	C0IP (EIP.6)
C1I	CAN1 Interrupt	73h	13	various	C1IE (EIE.5)	C1IP (EIP.5)
WDTI	Watchdog Timer	63h	14	WDIF (WDCON.3)	EWDI (EIE.4)	PWDI (EIP.4)
CANBUS	CAN0/1 Bus Activity	7Bh	15	various	CANBIE (EIE.7)	CANBIP (EIP.7)

Unless marked, all flags must be cleared by the application software.
**Cleared automatically by hardware when the service routine is entered.*
***If edge-triggered, flag is cleared automatically by hardware when the service routine is entered. If level-triggered, flag follows the state of the interrupt pin.*

CONTROLLER AREA NETWORK (CAN) MODULE

The DS80C390 incorporates two CAN controllers that are fully compliant with the CAN 2.0B specification. CAN is a highly robust, high-performance communication protocol for serial communications. Popular in a wide range of applications including automotive, medical, heating, ventilation, and industrial control, the CAN architecture allows for the construction of sophisticated networks with a minimum of external hardware.

The CAN controllers support the use of 11-bit standard or 29-bit extended acceptance identifiers for up to 15 messages, with the standard 8-byte data field, in each message. Fourteen of the 15 message centers are programmable in either transmit or receive modes, with the 15th designated as a FIFO-buffered, receive-only message center to help prevent data overruns. All message centers support two separate 8-bit media masks and media arbitration fields for incoming message verification. This feature supports the use of higher-level protocols, which make use of the first and/or second byte of data as a part of the acceptance layer for storing incoming messages. Each message center can also be programmed independently to test incoming data with or without the use of the global masks.

Global controls and status registers in each CAN unit allow the microcontroller to evaluate error messages, generate interrupts, locate and validate new data, establish the CAN bus timing, establish identification mask bits, and verify the source of individual messages. Each message center is individually equipped with the necessary status and control bits to establish direction, identification mode (standard or extended), data field size, data status, automatic remote frame request and acknowledgment, and perform masked or non-masked identification acceptance testing.

COMMUNICATING WITH THE CAN MODULE

The microcontroller interface to the CAN modules is divided into two groups of registers. All the global CAN status and control bits as well as the individual message center control/status registers are located in the SFR map. The remaining registers associated with the message centers (data identification, identification/arbitration masks, format, and data) are located in MOVX data space. The CMA bit (MCON.5) allows the message centers to be mapped to either 00EE00h–00EEFFh (CMA = 0) or 401000h–4011FFh (CMA = 1), reducing the possibility of a memory conflict with application software. Note that setting the CMA bit employs a special 23rd address bit that is only used for addressing CAN MOVX memory. The DS80C390's internal architecture requires that the device be in one of the two 22-bit addressing modes when the CMA bit is set to correctly use the 23rd bit and access the CAN MOVX memory. A special lockout feature prevents the accidental software corruption of the control, status, and mask registers while a CAN operation is in progress. Each CAN processor uses 15 message centers. Each message center is composed of four specific areas, including the following:

1) Four arbitration registers (C0MxAR0–3 and C1MxAR0–3) that store either the 11-bit or 29-bit arbitration value. These registers are located in the MOVX memory map.
2) A format register (C0MxF and C1MxF) that informs the CAN processor as to the direction (transmit or receive), the number of data bytes in the message, the identification format (standard or extended), and the optional use of the identification mask or media mask during message evaluation. This register is located in the MOVX memory map.
3) Eight data bytes for storage of 0 to 8 bytes of data (C0MxD0–7 and C1MxD0–7), which are located in the MOVX memory map.
4) Message control registers (C0MxC and C1MxC), which are located in the SFR memory for fast access.

Each of the message centers is identical with the exception of message center 15. Message center 15 has been designed as a receive-only center, and is also buffered through the use of a two-message FIFO to help prevent message loss in a message-overrun situation. The receipt of a third message before either of the first two are read will overwrite the second message, leaving the first message undisturbed.

Modification of the CAN registers located in MOVX memory is protected through the SWINT bits, with one bit protecting each respective CAN module. Consult the description of this bit in the *High-Speed Microcontroller User's Guide: DS80C390 Supplement* for more information. Each CAN module contains a block of control/status/mask registers, 14 functionally identical message centers, plus a 15th message center that is receive-only and incorporates a buffered FIFO. The following tables describe the organization of the message centers located in MOVX space.

MOVX MESSAGE CENTERS FOR CAN 1

CAN 1 CONTROL/STATUS/MASK REGISTERS

REGISTER	7	6	5	4	3	2	1	0	MOVX DATA ADDRESS[1]
C1MID0	MID07	MID06	MID05	MID04	MID03	MID02	MID01	MID00	xxxx00h
C1MA0	M0AA7	M0AA6	M0AA5	M0AA4	M0AA3	M0AA2	M0AA1	M0AA0	xxxx01h
C1MID1	MID17	MID16	MID15	MID14	MID13	MID12	MID11	MID10	xxxx02h
C1MA1	M1AA7	M1AA6	M1AA5	M1AA4	M1AA3	M1AA2	M1AA1	M1AA0	xxxx03h
C1BT0	SJW1	SJW0	BPR5	BPR4	BPR3	BPR2	BPR1	BPR0	xxxx04h
C1BT1	SMP	TSEG26	TSEG25	TSEG24	TSEG13	TSEG12	TSEG11	TSEG10	xxxx05h
C1SGM0	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	xxxx06h
C1SGM1	ID20	ID19	ID18	0	0	0	0	0	xxxx07h
C1EGM0	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	xxxx08h
C1EGM1	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13	xxxx09h
C1EGM2	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	xxxx0Ah
C1EGM3	ID4	ID3	ID2	ID1	ID0	0	0	0	xxxx0Bh
C1M15M0	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	xxxx0Ch
C1M15M1	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13	xxxx0Dh
C1M15M2	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	xxxx0Eh
C1M15M3	ID4	ID3	ID2	ID1	ID0	0	0	0	xxxx0Fh

CAN 1 MESSAGE CENTER 1

	Reserved								xxxx10h–11h
C1M1AR0	CAN 1 MESSAGE 1 ARBITRATION REGISTER 0								xxxx12h
C1M1AR1	CAN 1 MESSAGE 1 ARBITRATION REGISTER 1								xxxx13h
C1M1AR2	CAN 1 MESSAGE 1 ARBITRATION REGISTER 2								xxxx14h
C1M1AR3	CAN 1 MESSAGE 1 ARBITRATION REGISTER 3							WTOE	xxxx15h
C1M1F	DTBYC3	DTBYC2	DTBYC1	DTBYC0	T/$\overline{R}$	EX/$\overline{ST}$	MEME	MDME	xxxx16h
C1M1D0–7	CAN 1 MESSAGE 1 DATA BYTES 0–7								xxxx17h–1Eh
	Reserved								xxxx1Fh

CAN 1 MESSAGE CENTERS 2–14

	MESSAGE CENTER 2 REGISTERS (similar to Message Center 1)	xxxx20h–2Fh
	MESSAGE CENTER 3 REGISTERS (similar to Message Center 1)	xxxx30h–3Fh
	MESSAGE CENTER 4 REGISTERS (similar to Message Center 1)	xxxx40h–4Fh
	MESSAGE CENTER 5 REGISTERS (similar to Message Center 1)	xxxx50h–5Fh
	MESSAGE CENTER 6 REGISTERS (similar to Message Center 1)	xxxx60h–6Fh
	MESSAGE CENTER 7 REGISTERS (similar to Message Center 1)	xxxx70h–7Fh
	MESSAGE CENTER 8 REGISTERS (similar to Message Center 1)	xxxx80h–8Fh
	MESSAGE CENTER 9 REGISTERS (similar to Message Center 1)	xxxx90h–9Fh
	MESSAGE CENTER 10 REGISTERS (similar to Message Center 1)	xxxxA0h–AFh
	MESSAGE CENTER 11 REGISTERS (similar to Message Center 1)	xxxxB0h–BFh
	MESSAGE CENTER 12 REGISTERS (similar to Message Center 1)	xxxxC0h–CFh
	MESSAGE CENTER 13 REGISTERS (similar to Message Center 1)	xxxxD0h–DFh
	MESSAGE CENTER 14 REGISTERS (similar to Message Center 1)	xxxxE0h–EFh

CAN 1 MESSAGE CENTER 15

—	Reserved								xxxxF0h–F1h
C1M15AR0	CAN 1 MESSAGE 15 ARBITRATION REGISTER 0								xxxxF2h
C1M15AR1	CAN 1 MESSAGE 15 ARBITRATION REGISTER 1								xxxxF3h
C1M15AR2	CAN 1 MESSAGE 15 ARBITRATION REGISTER 2								xxxxF4h
C1M15AR3	CAN 1 MESSAGE 15 ARBITRATION REGISTER 3							WTOE	xxxxF5h
C1M15F	DTBYC3	DTBYC2	DTBYC1	DTBYC0	0	EX/$\overline{ST}$	MEME	MDME	xxxxF6h
C1M15D0–C1M15D7	CAN 1 MESSAGE 15 DATA BYTE 0–7								xxxxF7h–FEh
	Reserved								xxxxFFh

[1]The first two bytes of the CAN 1 MOVX memory address are dependent on the setting of the CMA bit (MCON.5) CMA = 0, xxxx = 00EF; CMA = 1, xxxx = 4011.

CAN INTERRUPTS

The DS80C390 supports three interrupts associated with the CAN controllers. One interrupt is dedicated to each CAN controller, providing receive/transmit acknowledgments from each of its 15 message centers. The remaining interrupt, the CAN bus activity interrupt, is used to detect CAN bus activity on the C0RX or C1RX pins.

The message center interrupts are enabled/disabled by individual ETI (transmit) and ERI (receive) enable bits in the corresponding message control register (located in SFR memory) for each message center. All the message center interrupts of each CAN module are ORed together into their respective CAN interrupt. The successful transmission or receipt of a message sets the INTRQ bit in the corresponding message control register (located in SFR memory). This bit can only be cleared through software. In addition, the global interrupt-enable bit (IE.7) and the specific CAN interrupt-enable bit, EIE.6 (CAN0) or EIE.5 (CAN1), must be correctly set to acknowledge a message center interrupt.

Interrupt assertion of error and status conditions associated with the CAN modules is controlled by the ERIE and STIE bits located in the CAN control registers, C0C and C1C.

ARBITRATION AND MASKING

After a CAN module has ascertained that an incoming message is bit-error-free, the identification field of that message is then compared against one or more arbitration values to determine if they will be loaded into a message center. Each enabled message center (see the MSRDY bit in the *CAN Message Control Register*) is tested in order from 1 to 15. The first message center to successfully pass the test receives the incoming message and ends the testing. Using masking registers allows the use of more complex identification schemes, as tests can be made based on bit patterns rather than an exact match between all bits in the identification field and arbitration values. Each CAN processor also incorporates a set of five masks to allow messages with different IDs to be grouped and successfully loaded into a message center. Note that some of these masks are optional as per the bits shown in the *Arbitration/Masking Feature Summary* table (Table 14).

There are several possible arbitration tests, varying according to which message center is involved. If all the enabled tests succeed, the message is loaded into the respective message center. The most basic test, performed on all messages, compares either 11 (CAN 2.0A) or 29 (CAN 2.0B) bits of the identification field to the appropriate arbitration register, based on the EX/$\overline{ST}$ bit in the CAN 0/1 format register. The MEME bit (C0MxF.1 or C1MxF.1) controls whether the arbitration and ID registers are compared directly or through a mask register. A special set of arbitration registers dedicated to message center 15 allows added flexibility in filtering this location.

If desired, further arbitration can be performed by comparing the first two bytes of the data field in each message against two 8-bit media arbitration register bytes. The MDME bit in the CAN message center format registers (C0MxF.0 or C1MxF.0) either disables (MDME = 0) arbitration, or enables (MDME = 1) arbitration using the media ID mask registers 0–1.

If the 11-bit or 29-bit arbitration and the optional media-byte arbitration are successful, the message is loaded into the respective message center. The format register also allows the microcontroller to program each message center to function in a receive or transmit mode through the T/$\overline{R}$ bit, and to use from 0 to 8 data bytes within the data field of a message. Note that message center 15 can only be used in a receive mode. To avoid a priority inversion, the DS80C390 CAN processors are configured to reload the transmit buffer with the message of the highest priority (lowest message center number) whenever an arbitration is lost or an error condition occurs.

REVISION HISTORY

REVISION	DESCRIPTION
062299	Initial preliminary release.
090799	Clarifies that unused/unimplemented bits in the CAN MOVX SRAM read 0. Corrected the t_{MCS} time period table. Corrected multiplexed 2-cycle date memory $\overline{CEO-3}$ read figure to show $\overline{RD}$ and $\overline{WR}$ inactive.
110199	Corrected P5.2 and P5.3 pin descriptions. Corrected description of sequence to activate the crystal frequency multiplier. Corrected references to PQFP to read LQFP. Added $\overline{RSTOL}$ timing information.
032904	Official release (removed "preliminary" status). Abs max soldering temp now references JEDEC standard. AC and DC specifications updated to reflect final characterization data. Clarified DC characteristics Note 6 concerning port 4 and 5. Removed Figure 1. *Typical I_{CC} vs. Frequency.* Added t_{LLAX3} specification (identical to t_{LLAX2}). Clarified that t_{RLAZ} is held weak latch until overdriven by external memory. Removed t_{PXIZ}, t_{PHAV}, t_{PHWL}, and t_{PHRL} from nonmultiplexed address/data bus table. Corrected PSEN trace in Figure 10 to not show assertion during MOVX write. Corrected Table 3 to show unnecessary steps during 16/16 divide. Supplied approximate oscillator-fail detection frequency. Removed text references to Stop mode current. Corrected location of PT2 in Table 14.
022305	In *Absolute Maximum Ratings* section (page 2): Removed "A" from IPC/JEDEC J-STD-020A specification to support lead-free devices. In *DC Electrical Characteristics* table (page 2): Changed V_{PFW} MIN to 4.10V from 4.20V Changed V_{PFW} MAX to 4.60V from 4.55V Changed V_{RST} MIN to 3.85V from 3.95V Changed V_{RST} MAX to 4.35V from 4.3V Changed V_{IH2} MIN reference to 0.7 x V_{CC} from 0.7 x V_{DD} Added Note 10 In *AC Electrical Characteristics* table (page 3): Added note to (now) Note 11 that AC timing is characterized and guaranteed by design but is not production tested.
060805	Added lead-free part numbers to *Ordering Information* table.
110905	Added new paragraph to page 33 stating "Software must ensure that the input value for the normalize operation is not zero or the function will not complete. Compilers such as the one from Keil Software have updated their libraries and compensate for this condition." Table 3: clarified text under "Normalize" function. Changed "Configure MCNTO register as required." to "Load MCNT0 with 00h."

Maxim Integrated Products, 120 San Gabriel Drive, Sunnyvale, CA 94086 408-737-7600