# 87C521/87C541

CMOS Single-Chip Microcontrollers

PRELIMINARY

## DISTINCTIVE CHARACTERISTICS

- Software and pin-compatible with 80C51, 80C521, and 80C541
- Beneficial for prototyping and initial production
- All 80C521 and 80C541 features retained
- Flashrite™ EPROM programming
- Two-level Program Memory Lock
- 32-Byte Encryption Array
- In-Circuit Test Mode facilitates testing

| | RAM (bytes) | EPROM (bytes) |
|---|---|---|
| 87C521 | 256 | 8K |
| 87C541 | 256 | 16K |

87C521 = User programmable 80C521
87C541 = User programmable 80C541

## GENERAL DESCRIPTION

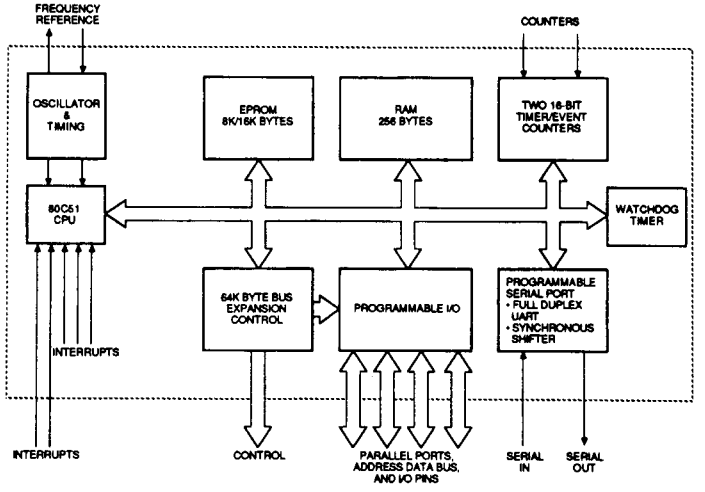
The 87C521 and 87C541 are CMOS EPROM versions of the 80C521 and 80C541, respectively. The 87C521 includes 8K bytes of on-chip EPROM, and the 87C541 includes 16K bytes of EPROM.

These user-programmable products are software- and pin-compatible with their ROM-based counterparts. All of the 80C521 and 80C541 features are retained, including a robust Watchdog Timer, Dual Data Pointers, and Software Reset. For more information consult the 80C521/80C541 Datasheet (order #09136).

The EPROM features on the 87C51 and 87C52T2 have also been retained. A two-level programmable lock structure prevents externally fetched code from accessing internal Program Memory and can disable EPROM verification and programming. A 32-byte Encryption Array can be used to encode the program code bytes during EPROM verification. A Flashrite programming algorithm allows the 8K-byte 87C521 and 16K-byte 87C541 to be programmed in approximately 24 and 48 seconds, respectively.



## BLOCK DIAGRAM



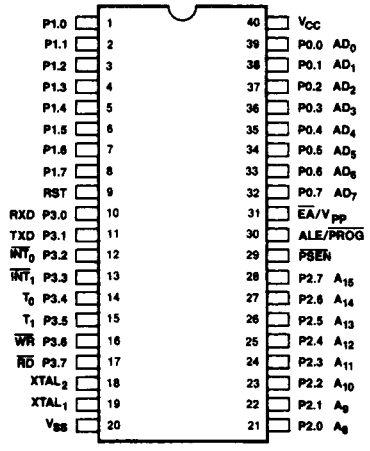

BD007750



Publication # 09744 | Rev. B | Amendment /0
Issue Date: October 1989

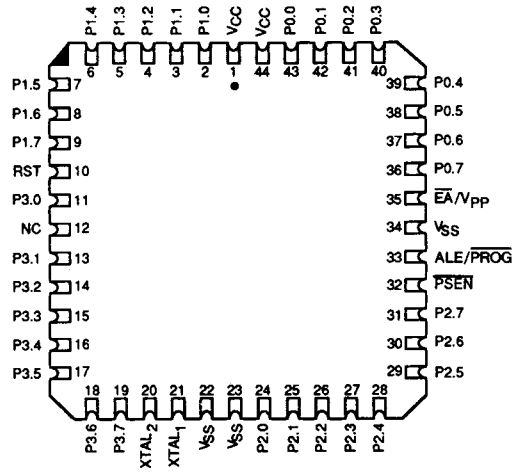
## CONNECTION DIAGRAMS
### Top View

**DIP**



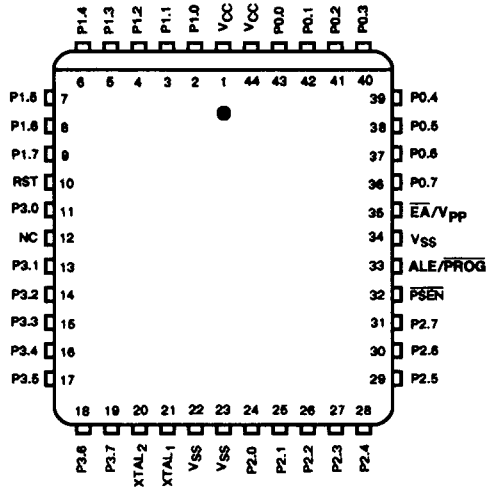

CD005552

**LCC**




CD010872

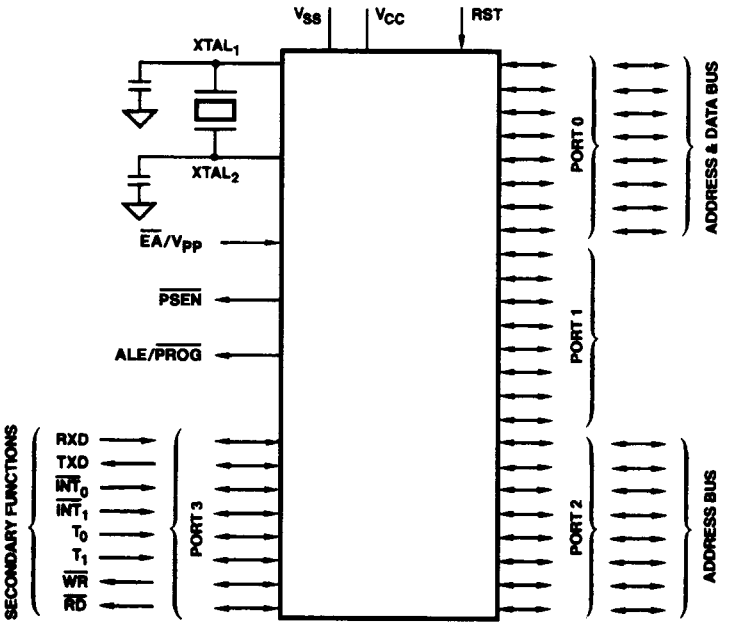
**PLCC**




CD009442

Note: Pin 1 is marked for orientation.

## LOGIC SYMBOL

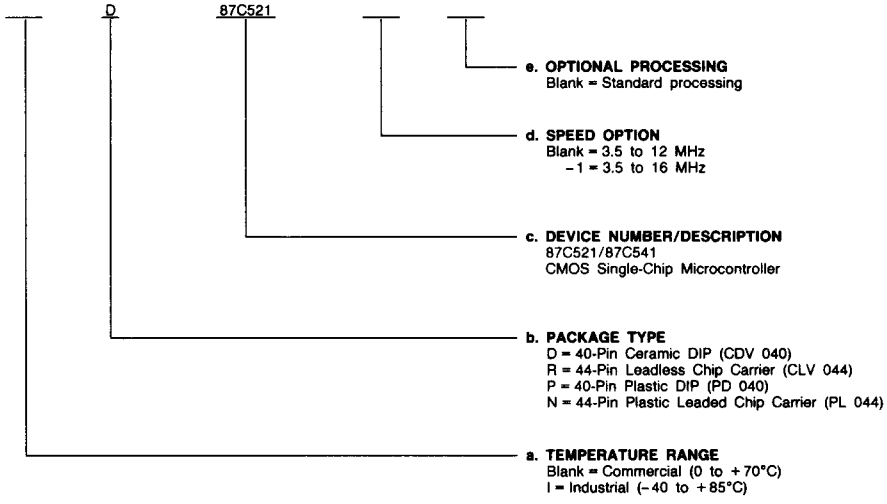



LS001326

## ORDERING INFORMATION

### Commodity Products

AMD commodity products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of: **a. Temperature Range**
**b. Package Type**
**c. Device Number**
**d. Speed Option**
**e. Optional Processing**




| Valid Combinations | |
|---|---|
| D, R, P, N<br>ID, IR, IP, IN | 87C521 |
| | 87C521-1 |
| | 87C541 |
| | 87C541-1 |

**Valid Combinations**

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released valid combinations, and to obtain additional data on AMD's standard military grade products.

## PIN DESCRIPTION

**Port 0 (Bidirectional; Open Drain)**

Port 0 is an open-drain I/O port. Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1s. Port 0 also outputs the code bytes during program verification in the 87C521/87C541. External pullups are required during program verification.

**Port 1 (Bidirectional)**

Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source four LS TTL inputs. Port 1 pins that have 1s written to them are pulled High by the internal pullups and—while in this state—can be used as inputs. As inputs, Port 1 pins that are externally being pulled Low will source current ($I_{IL}$ on the data sheet) because of the internal pullups.

Port 1 also receives the low-order address bytes during program verification.

**Port 2 (Bidirectional)**

Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source four LS TTL inputs. Port 2 pins having 1s written to them are pulled High by the internal pullups and—while in this state—can be used as inputs. As inputs, Port 2 pins externally being pulled Low will source current ($I_{IL}$) because of internal pullups.

Port 2 emits the high-order address byte during fetches from external Program Memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1s. During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function register.

Port 2 also receives the high-order address bits during the programming of the EPROM and during program verification of the EPROM, as well as some control signals.

**Port 3 (Bidirectional)**

Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source four LS TTL inputs. Port 3 pins having 1s written to them are pulled High by the internal pullups and—while in this state—can be used as inputs. As inputs, Port 3 pins externally being pulled Low will source current ($I_{IL}$) because of the pullups. Port 3 also receives some control signals for EPROM programming and program verification.

Port 3 also serves the functions of various special features as listed below:

| Port Pin | Alternate Function |
|---|---|
| $P_{3.0}$ | RxD (Serial Input Port) |
| $P_{3.1}$ | TxD (Serial Output Port) |
| $P_{3.2}$ | $\overline{INT_0}$ (External Interrupt 0) |
| $P_{3.3}$ | $\overline{INT_1}$ (External Interrupt 1) |
| $P_{3.4}$ | $T_0$ (Timer 0 External Input) |
| $P_{3.5}$ | $T_1$ (Timer 1 External Input) |
| $P_{3.6}$ | $\overline{WR}$ (External Data Memory Write Strobe) |
| $P_{3.7}$ | $\overline{RD}$ (External Data Memory Read Strobe) |

**RST Reset (Input; Active High)**

This pin is used to reset the device when held High for two machine cycles while the oscillator is running. A small internal resistor permits power-on reset using only a capacitor connected to $V_{CC}$.

Immediately prior to a Watchdog Reset or Software Reset, this pin is pulled High for one state time. The internal pullup can be overdriven by an external driver capable of sinking/sourcing 2.5 mA. (See Figure 6 of the 80C521 Datasheet, order #09136C/0, for possible circuit configurations.)

**ALE/$\overline{PROG}$ Address Latch Enable/$\overline{Program\ Pulse}$ (Input/Output)**

Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. ALE can drive eight LS TTL inputs.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, allowing use for external-timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory. This pin also accepts the program pulse input ($\overline{PROG}$) when programming the EPROM.

**$\overline{PSEN}$ $\overline{Program\ Store\ Enable}$ (Output; Active Low)**

$\overline{PSEN}$ is the read strobe to external Program Memory. $\overline{PSEN}$ can drive eight LS TTL inputs. When the device is executing code from an external program memory, $\overline{PSEN}$ is activated twice each machine cycle—except that two $\overline{PSEN}$ activations are skipped during each access to external Data Memory. $\overline{PSEN}$ is not activated during fetches from internal Program Memory.

**$\overline{EA}$/$V_{PP}$ $\overline{External\ Access\ Enable}$/Programming Voltage (Input; Active Low)**

$\overline{EA}$ must be externally held Low to enable the device to fetch code from external Program Memory locations 0000H to 1FFFH for the 87C521 and 3FFFH for the 87C541. If $\overline{EA}$ is held High, the 87C521/87C541 executes from internal Program Memory unless the program counter exceeds 1FFFH and 3FFFH respectively.

This pin also receives the 12.75-V programming supply voltage during programming of the EPROM.

**$XTAL_1$ Crystal (Input)**

Input to the inverting-oscillator amplifier, and input to the internal clock-generator circuits.

**$XTAL_2$ Crystal (Output)**

Output of the inverting-oscillator amplifier.

**$V_{CC}$ Power Supply**

Power supply during normal, idle, and power-down operations.

**$V_{SS}$ Circuit Ground**

## PROGRAMMING

The 87C521/87C541 can be programmed with the Flashrite algorithm. It differs from other methods in the value used for $V_{PP}$ (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

To program the EPROM, either the internal or external oscillator must be running between 4 and 6 MHz, since the internal bus is used to transfer address and program data to the appropriate internal registers. Table 1 shows the various EPROM programming modes.

**Table 1. EPROM Programming Modes for the 87C521/87C541**

| Mode | RST | $\overline{PSEN}$ | ALE/$\overline{PROG}$ | $\overline{EA}$/$V_{PP}$ | P2.7 | P2.6 | P3.7 | P3.6 |
|---|---|---|---|---|---|---|---|---|
| Program Code | H | L | L* | $V_{PP}$ | H | L | H | H |
| Verify Code | H | L | H | $V_{PPX}$ | L | L | H | H |
| Pgm Encryption Table | H | L | L* | $V_{PP}$ | H | L | H | L |
| Pgm Lock Bit 1 | H | L | L* | $V_{PP}$ | H | H | H | H |
| Pgm Lock Bit 2 | H | L | L* | $V_{PP}$ | H | H | L | L |
| Read Silicon Signature | H | L | H | H | L | L | L | L |

Key: H = Logic High for that pin
L = Logic Low for that pin
$V_{PP}$ = 12.75 V ±0.25 V
$V_{CC}$ = 5 V ±10% during programming and verification
2.0 V < $V_{PPX}$ < 13.0 V

*ALE/$\overline{PROG}$ receives 25 programming pulses while $V_{PP}$ is held at 12.75 V. Each programming pulse is Low for 100 µs (±10% µs) and High for a minimum of 10 µs.

### Programming

The programming configuration for the 87C521 is shown in Figure 1. The address of the EPROM location to be programmed is applied to Ports 1 and 2 as shown in the figure. The programming configuration of the 87C541 is identical except that P2.5 is also used as an address input. The code byte to be programmed into that location is applied to Port 0. Once RST, $\overline{PSEN}$, Port 2, and Port 3 are held to the levels indicated in Figure 1, ALE/$\overline{PROG}$ is pulsed Low 25 times, as shown in Figure 2.

The maximum voltage applied to the $\overline{EA}$/$V_{PP}$ pin must not exceed 13 V at any time as specified for $V_{PP}$. Even a slight spike can cause permanent damage to the device. The $V_{PP}$ source should thus be well-regulated and glitch-free.

When programming, a 0.1-µF capacitor is required across $V_{PP}$ and ground to suppress spurious transients that may damage the device.

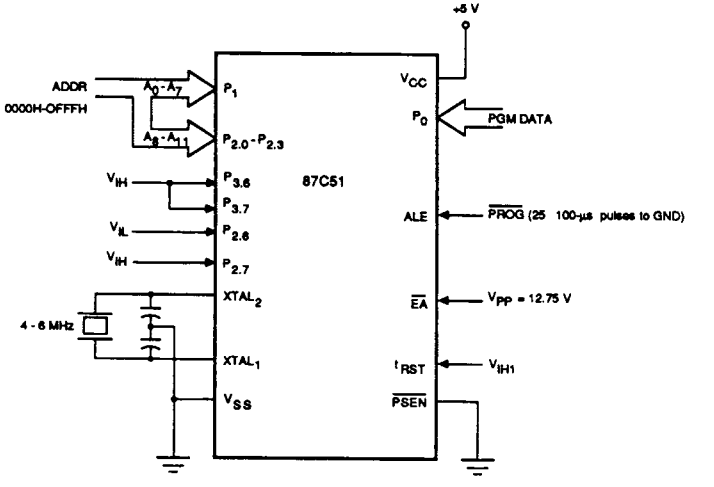



**Figure 1. 87C521 Programming Configuration**

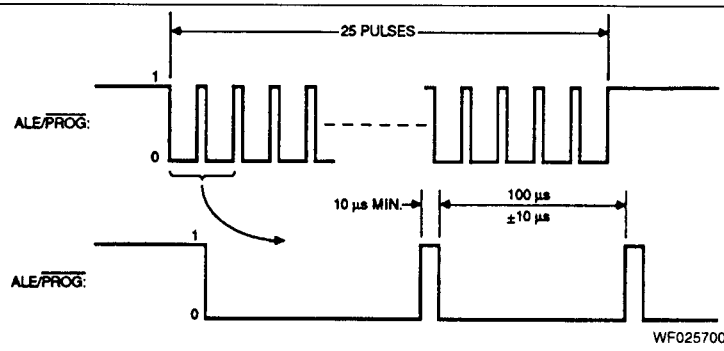



**Figure 2. PROG Waveforms**

### Program Verification

The 87C521/87C541 provides a method of reading the programmed code bytes in the EPROM array for program verification. This function is possible as long as Lock Bit 2 has not been programmed.

For program verification, the address of the Program Memory location to be read is applied to Ports 1 and 2 as shown in Figure 3. Verification of the 87C541 is identical except that P2.5 is also used as an address input. Once RST, PSEN, Port 2, and Port 3 are held to the levels indicated, the contents of the addressed location will be emitted on Port 0. External pullups are required on Port 0 for this operation. The EPROM programming and verification waveforms provide further details.

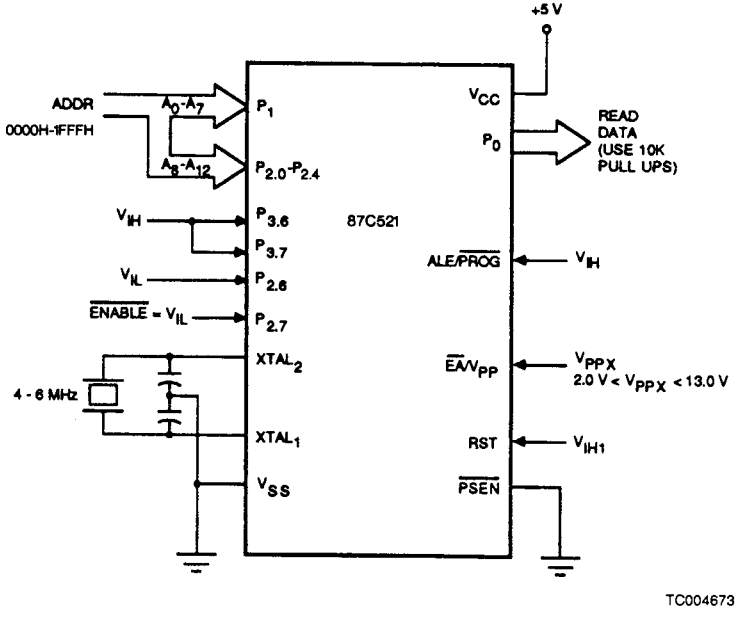



**Figure 3. 87C521 Program Verification**

## Program Encryption Table

The 87C521/87C541 features a 32-byte Encryption Array. It can be programmed by the customer, thus encrypting the program code bytes read during EPROM verification. The EPROM verification procedure is performed as usual except that each code byte comes out logically X-NORed with one of the 32 key bytes.

The key byte used is the one whose address corresponds to the lower 5 bits of the EPROM verification address. Thus, when the EPROM is verified starting with address 0000H, all 32 keys in their correct sequence must be known. Unprogrammed bytes have the value FFH. Thus, if the Encryption Table is left unprogrammed, no encryption will be performed, since any byte X-NORed with FFH leaves that byte unchanged.

To program the Encryption Table, programming is set up as usual, except that P3.6 is held Low, as shown in Table 1. The 25-pulse programming sequence is applied to each address, 00 through 1FH. The programming of these bytes does not affect the standard 4K-byte EPROM array. When the Encryption Table is programmed, the Program Verify operation will produce only encrypted data.

The Encryption Table cannot be directly read. The programming of Lock Bit 1 will disable further Encryption Table programming.

## Security Lock Bits

The 87C521/87C541 contains two Lock Bits that can be programmed to obtain additional security features. P = Programmed and U = Unprogrammed.

| Lock Bit 1 | Lock Bit 2 | Result |
|---|---|---|
| U | U | Normal Operation |
| P | U | • Externally fetched code cannot access internal Program Memory<br>• All further Programming disabled (except Lock Bit 2) |
| U | P | Reserved |
| P | P | • Externally fetched code cannot access internal Program Memory<br>• All further Programming disabled<br>• Program Verification disabled |

To program the Lock Bits, a 100 pulse programming sequence is required using the levels shown in Table 1. After Lock Bit 1 is programmed, further programming of the Code Memory and Encryption Table is disabled. However, Lock Bit 2 may still be programmed, providing the highest level of security available on the 87C521/87C541.

## Silicon Signature Verification

AMD supports silicon signature verification for the 87C521/87C541. The manufacturer code and part code can be read from the device before any programming is done to enable the EPROM Programmer to recognize the device.

To read the silicon signature, the external pins are set up as shown in Figure 4. This procedure is the same as a normal verification except that P3.6 and P3.7 are pulled to a logic Low. The values returned are:

| | | |
|---|---|---|
| Manufacturer Code | Address: 0030H | Code: 01H |
| Part Code: 87C521 | Address: 0031H | Code: 32H |
| Part Code: 87C541 | Address: 0031H | Code: 32H |

Code 01H indicates AMD as the manufacturer. Code 32H indicates that the device type is the 87C521 or 87C541.

## In-Circuit Test Mode

The In-Circuit Test Mode facilitates testing and debugging of systems using the 87C521/87C541 without the device having to be removed from the circuit. The In-Circuit Test Mode is invoked by:

1. Pulling ALE Low while RST is held High and $\overline{PSEN}$ is High.

2. Holding ALE Low as RST is deactivated.

While the device is in In-Circuit Test Mode, the Port 0 pins go into a float state, and the other port pins and ALE and $\overline{PSEN}$ are weakly pulled High. The oscillator circuit remains active. While the 87C521/87C541 is in this mode, an emulator or test CPU can be used to drive the circuit. Normal operation is restored when a Hardware Reset is applied.

## Erasure Characteristics

Light and other forms of electromagnetic radiation can lead to erasure of the EPROM when exposed for extended periods of time.

Wavelengths of light shorter than 4000 angstroms, such as sunlight or indoor fluorescent lighting, can eventually cause inadvertent erasure and, therefore, should not be allowed to expose the EPROM for lengthy durations (approximately one week in sunlight or three years in room-level fluorescent lighting). It is suggested that the window be covered with an opaque label if an application is likely to subject the device to this type of radiation.

It is recommended that ultraviolet light (of 2537 angstroms) be used at a dose of at least 15 W-sec/cm$^2$ when erasing the EPROM. An ultraviolet lamp rated at 12,000 $\mu$W/cm$^2$ held one inch away for 20–30 minutes should be sufficient.

EPROM erasure leaves the Program Memory in an "all ones" state.

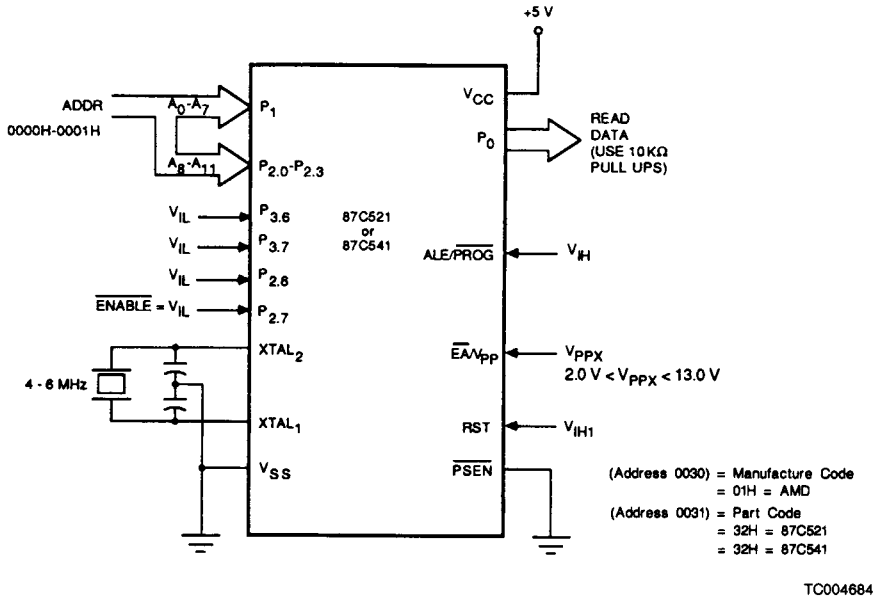



**Figure 4. 87C521/87C541 Silicon Signature Verification Configuration**

## Oscillator Characteristics

$XTAL_1$ and $XTAL_2$ are the input and output, respectively, of an inverting amplifier that is configured for use as an on-chip oscillator (see Figure 5). Either a quartz crystal or ceramic resonator may be used.

To drive the device from an external clock source, $XTAL_1$ should be driven while $XTAL_2$ is left unconnected (see Figure 6). There are no requirements on the duty cycle of the external clock signal since the input to the internal clocking circuitry is through a divide-by-two flip-flop; but minimum and maximum High and Low times specified on the data sheet must be observed.

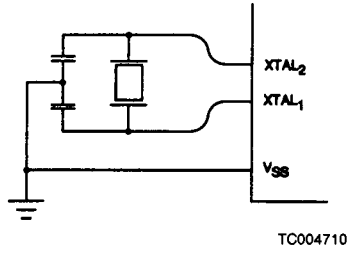



**Figure 5. Crystal Oscillator**

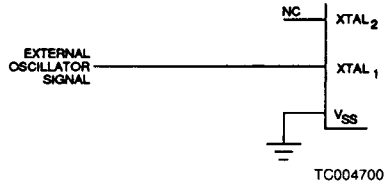



**Figure 6. External Drive Configuration**

## ABSOLUTE MAXIMUM RATINGS

Storage Temperature ..... −65 to +150°C
Voltage on $\overline{EA}$/$V_{PP}$ Pin to $V_{SS}$ ..... −0.5 to +13.0 V
Voltage on $V_{CC}$ to $V_{SS}$ ..... −0.5 to +6.5 V
Voltage on Any Other Pin to $V_{SS}$ ..... −0.5 to +6.5 V
Power Dissipation ..... 200 mW

*Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.*

## OPERATING RANGES

Commercial (C) Devices
Ambient Temperature ($T_A$) ..... 0 to +70°C
Supply Voltage ($V_{CC}$) ..... +4.5 to +5.5 V
Ground ($V_{SS}$) ..... 0 V

Industrial (I) Devices
Ambient Temperature ($T_A$) ..... −40 to +85°C
Supply Voltage ($V_{CC}$) ..... +4.5 to +5.5 V
Ground ($V_{SS}$) ..... 0 V

*Operating ranges define those limits between which the functionality of the device is guaranteed.*

## DC CHARACTERISTICS over operating ranges

| Parameter Symbol | Parameter Description | Test Conditions | Min. | Max. | Unit |
|---|---|---|---|---|---|
| $V_{IL}$ | Input Low Voltage (Except $\overline{EA}$) | | −0.5 | 0.2 $V_{CC}$ − 0.1 | V |
| $V_{IL1}$ | Input Low Voltage ($\overline{EA}$) | | 0 | 0.2 $V_{CC}$ − 0.3 | V |
| $V_{IH}$ | Input High Voltage (Except $XTAL_1$, RST) | | 0.2 $V_{CC}$ + 0.9 | $V_{CC}$ + 0.5 | V |
| $V_{IH1}$ | Input High Voltage to $XTAL_1$, RST | | 0.7 $V_{CC}$ | $V_{CC}$ + 0.5 | V |
| $V_{OL}$ | Output Low Voltage (Ports 1, 2, 3) | $I_{OL}$ = 1.6 mA (Note 1) | | 0.45 | V |
| $V_{OL1}$ | Output Low Voltage (Port 0, ALE, $\overline{PSEN}$) | $I_{OL}$ = 3.2 mA (Note 1) | | 0.45 | V |
| $V_{OH}$ | Output High Voltage (Ports 1, 2, 3), ALE, $\overline{PSEN}$ | $I_{OH}$ = −60 µA, $V_{CC}$ = 5 V ±10% | 2.4 | | V |
| | | $I_{OH}$ = −10 µA | 0.9 $V_{CC}$ | | |
| $V_{OH1}$ | Output High Voltage (Port 0 in External Bus Mode) | $I_{OH}$ = −800 µA, $V_{CC}$ = 5 V ±10% | 2.4 | | V |
| | | $I_{OH}$ = −80 µA (Note 2) | 0.9 $V_{CC}$ | | |
| $I_{IL}$ | Logical 0 Input Current (Ports 1, 2, 3) | $V_{IN}$ = 0.45 V | | −50 | µA |
| $I_{TL}$ | Logical 1-to-0 Transition Current (Ports 1, 2, 3) | (Note 3) | | −650 | µA |
| $I_{LI}$ | Input Leakage Current (Port 0) | $V_{IN}$ = $V_{IL}$ or $V_{IH}$ | | ±10 | µA |
| $I_{CC}$ | Power Supply Current: Active Mode @ 12 MHz (Note 4) | (Note 5) | | Note 4 | mA |
| | Idle Mode @ 12 MHz (Note 4) | | | Note 4 | |
| | Power-Down Mode | | | 50 | µA |
| RRST | Reset Pulldown Resistor | | 50 | 300 | kΩ |
| $C_{IO}$ | Pin Capacitance | Test Freq = 1 MHz, $T_A$ = 25°C | | 10 | pF |



Notes: 1. Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the $V_{OL}$s of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100 pF), the noise pulse on the ALE line may exceed 0.8 V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.

2. Capacitive loading on Ports 0 and 2 may cause the $V_{OH}$ on ALE and $\overline{PSEN}$ to momentarily fall below the 0.9 $V_{CC}$ specification when the address bits are stabilizing.

3. Pins of Ports 1, 2, and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when $V_{IN}$ is approximately 2 V.

4. $I_{CCMAX}$ at other frequencies is given by:
   Active Mode: $I_{CC}$ TYPICAL = 0.94 x Freq + 13.71 $I_{CCMAX}$ = 1.38 x Freq + 20.4
   Idle Mode: $I_{CC}$ TYPICAL = 0.38 x Freq + 5.4 $I_{CCMAX}$ = 0.38 x Freq + 11.9
   where Freq is the external oscillator frequency in MHz. $I_{CCMAX}$ is given in mA.

5. Active Mode $I_{CC}$ is measured with all output pins disconnected; $XTAL_1$ driven with TCLCH, TCHCL = 5 ns, $V_{IL}$ = $V_{SS}$ + 0.5 V, $V_{IH}$ = $V_{CC}$ − 0.5 V; $XTAL_2$ NC; $\overline{EA}$ = RST = Port 0 = $V_{CC}$.
   Idle Mode $I_{CC}$ is measured with all output pins disconnected; $XTAL_1$ driven with TCLCH, TCHCL = 5 ns, $V_{IL}$ = $V_{SS}$ + 0.5 V, $V_{IH}$ = $V_{CC}$ − 0.5 V; $XTAL_2$ = NC; Port 0 = $V_{CC}$; $\overline{EA}$ = RST = $V_{SS}$.
   Power-Down Mode $I_{CC}$ is measured with all output pins disconnected; $\overline{EA}$ = Port 0 = $V_{CC}$; $XTAL_2$ NC; RST = $V_{SS}$.

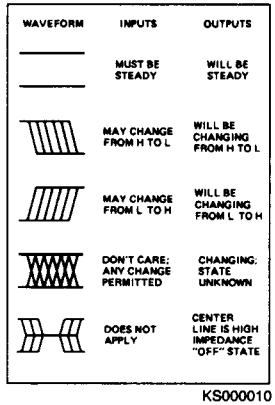
**SWITCHING CHARACTERISTICS** over operating ranges
(Load Capacitance for Port 0, ALE, and $\overline{PSEN}$ = 100 pF, Load Capacitance for All Other Outputs = 80 pF)

| Parameter Symbol | Parameter Description | 16 MHz Osc. | | 12 MHz Osc. | | Variable Oscillator | | Unit |
|---|---|---|---|---|---|---|---|---|
| | | Min. | Max. | Min. | Max. | Min. | Max. | |
| 1/TCLCL | Oscillator Frequency | | | | | 3.5 | 16 | MHz |
| TLHLL | ALE Pulse Width | 85 | | 127 | | 2TCLCL-40 | | ns |
| TAVLL | Address Valid to ALE Low | 7 | | 28 | | TCLCL-55 | | ns |
| TLLAX | Address Hold After ALE Low | 27 | | 48 | | TCLCL-35 | | ns |
| TLLIV | ALE Low to Valid Instr. In | | 150 | | 234 | | 4TCLCL-100 | ns |
| TLLPL | ALE Low to $\overline{PSEN}$ Low | 22 | | 43 | | TCLCL-40 | | ns |
| TPLPH | $\overline{PSEN}$ Pulse Width | 142 | | 205 | | 3TCLCL-45 | | ns |
| TPLIV | $\overline{PSEN}$ Low to Valid Instr. In | | 83 | | 145 | | 3TCLCL-105 | ns |
| TPXIX | Input Instr. Hold After $\overline{PSEN}$ | 0 | | 0 | | 0 | | ns |
| TPXIZ | Input Instr. Float After $\overline{PSEN}$ | | 38 | | 59 | | TCLCL-25 | ns |
| TAVIV | Address to Valid Instr. In | | 208 | | 312 | | 5TCLCL-105 | ns |
| TPLAZ | $\overline{PSEN}$ Low to Address Float | | 10 | | 10 | | 10 | ns |
| TRLRH | $\overline{RD}$ Pulse Width | 275 | | 400 | | 6TCLCL-100 | | ns |
| TWLWH | $\overline{WR}$ Pulse Width | 275 | | 400 | | 6TCLCL-100 | | ns |
| TRLDV | $\overline{RD}$ Low to Valid Data In | | 148 | | 252 | | 5TCLCL-165 | ns |
| TRHDX | Data Hold After $\overline{RD}$ | 0 | | 0 | | 0 | | ns |
| TRHDZ | Data Float After $\overline{RD}$ | | 55 | | 97 | | 2TCLCL-70 | ns |
| TLLDV | ALE Low to Valid Data In | | 350 | | 517 | | 8TCLCL-150 | ns |
| TAVDV | Address to Valid Data In | | 398 | | 585 | | 9TCLCL-165 | ns |
| TLLWL | ALE Low to $\overline{RD}$ or $\overline{WR}$ Low | 137 | 238 | 200 | 300 | 3TCLCL-50 | 3TCLCL+50 | ns |
| TAVWL | Address Valid to $\overline{RD}$ or $\overline{WR}$ Low | 120 | | 203 | | 4TCLCL-130 | | ns |
| TQVWX | Data Valid to $\overline{WR}$ Transition | 2 | | 23 | | TCLCL-60 | | ns |
| TQVWH | Data Valid to $\overline{WR}$ High | 287 | | 433 | | 7TCLCL-150 | | ns |
| TWHQX | Data Hold After $\overline{WR}$ | 12 | | 33 | | TCLCL-50 | | ns |
| TRLAZ | $\overline{RD}$ Low to Address Float | | 0 | | 0 | | 0 | ns |
| TWHLH | $\overline{RD}$ or $\overline{WR}$ High to ALE High | 22 | 103 | 43 | 123 | TCLCL-40 | TCLCL+40 | ns |



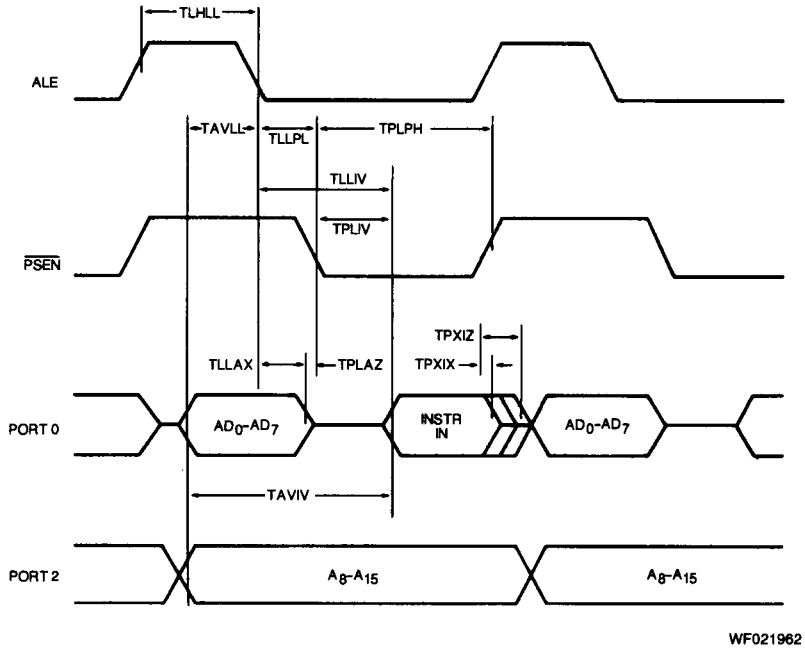
## SWITCHING WAVEFORMS

### KEY TO SWITCHING WAVEFORMS

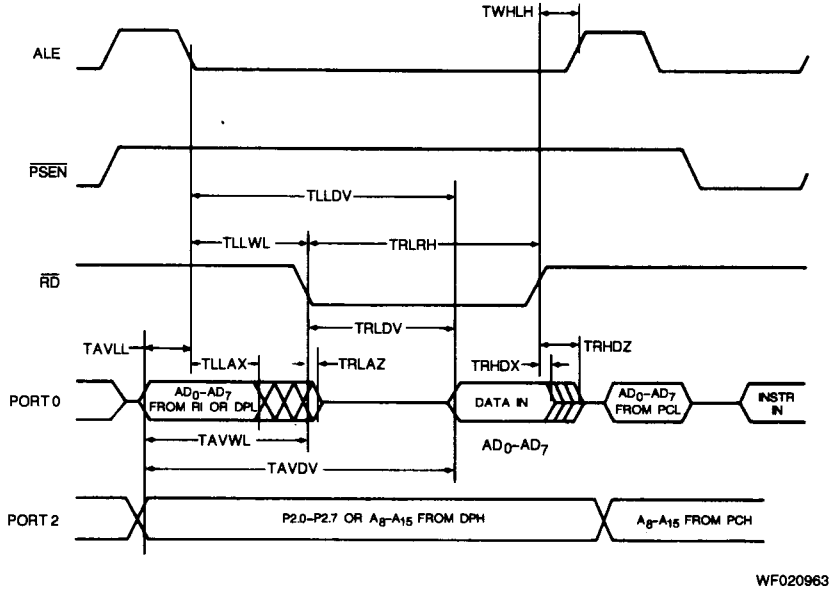



KS000010

## SWITCHING WAVEFORMS

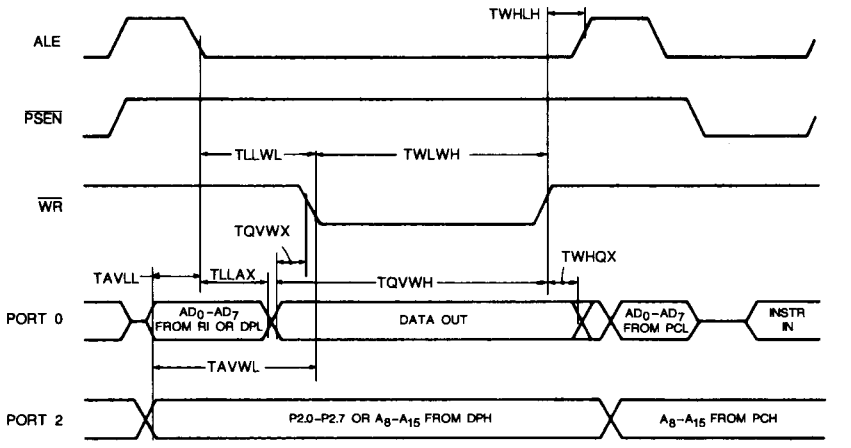



**External Program Memory Read Cycle**



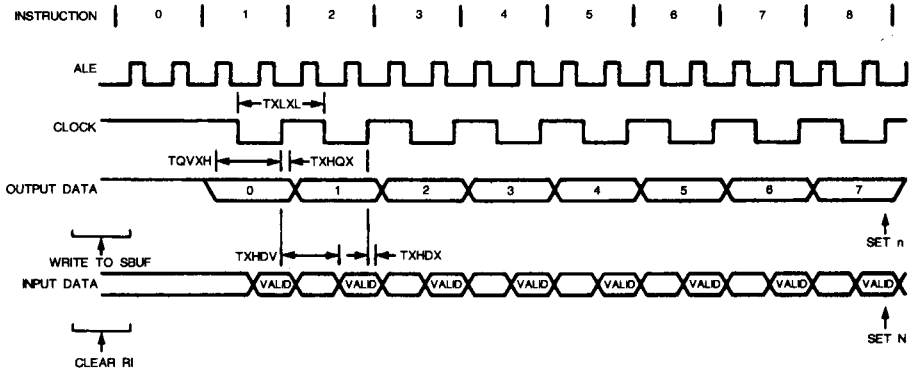

**External Data Memory Read Cycle**

## SWITCHING WAVEFORMS (continued)




WF020934

**External Data Memory Write Cycle**



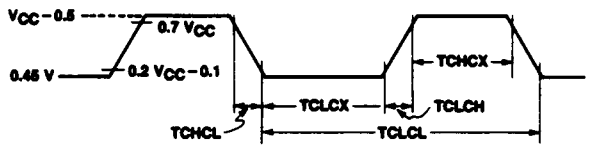

WF020951

**Shift Register Timing Waveforms**

## EXTERNAL CLOCK DRIVE

| Parameter Symbol | Parameter Description | Min. | Max. | Unit |
|---|---|---|---|---|
| 1/TCLCL | Oscillator Frequency | 3.5 | 16 | MHz |
| TCHCX | High Time | 30 | | ns |
| TCLCX | Low Time | 30 | | ns |
| TCLCH | Rise Time | | 20 | ns |
| TCHCL | Fall Time | | 20 | ns |






WF020910

**External Clock Drive Waveform**

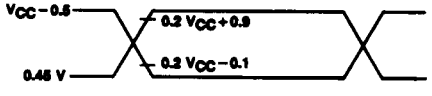
## SERIAL PORT TIMING — SHIFT REGISTER MODE

(Test Conditions: $T_A$ = 0 to +70°C; $V_{CC}$ = 5 V ±10%; $V_{SS}$ = 0 V; Load Capacitance = 80 pF)

| Parameter Symbol | Parameter Description | 16 MHz Osc. | | Variable Oscillator | | Unit |
|---|---|---|---|---|---|---|
| | | Min. | Max. | Min. | Max. | |
| TXLXL | Serial Port Clock Cycle Time | 750 | | 12TCLCL | | ns |
| TQVXH | Output Data Setup to Clock Rising Edge | 492 | | 10TCLCL–133 | | ns |
| TXHQX | Output Data Hold after Clock Rising Edge | 8 | | 2TCLCL–117 | | ns |
| TXHDX | Input Data Hold after Clock Rising Edge | 0 | | 0 | | ns |
| TXHDV | Clock Rising Edge to Input Data Valid | | 700 | | 10TCLCL–133 | ns |



### AC Testing




WF020900

AC inputs during testing are driven at $V_{CC}$ – 0.5 for a logic 1 and 0.45 V for a logic 0. Timing measurements are made at $V_{IH}$ min. for a logic 1 and $V_{IL}$ max. for a logic 0.

**Input/Output Waveform**


WF020940

For timing purposes a port pin is no longer floating when a 100-mV change from load voltage occurs, and begins to float when a 100-mV change from the loaded $V_{OH}/V_{OL}$ level occurs. $I_{OL}/I_{OH} \geq \pm 20$ mA.

**Float Waveform**

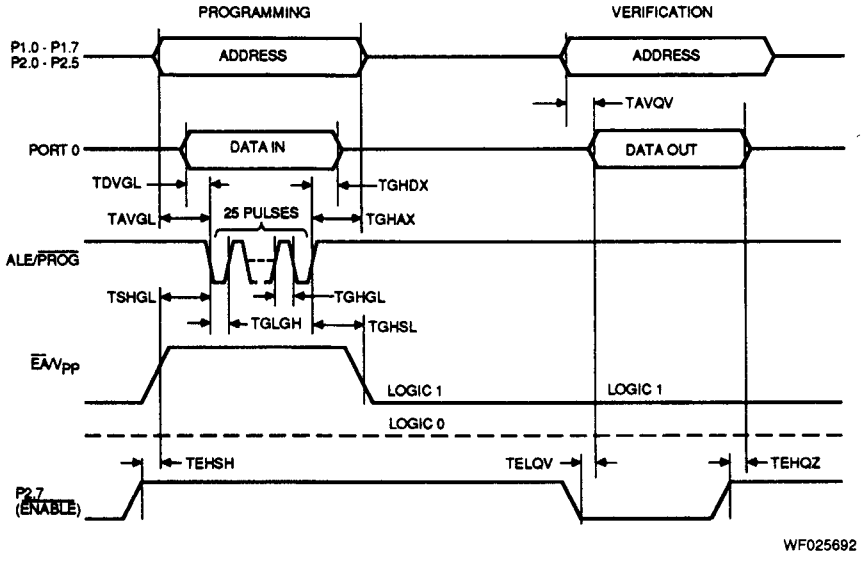
## EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

($T_A$ = +21 to +27°C)

| Parameter Symbol | Parameter Description | Min. | Max. | Unit |
|---|---|---|---|---|
| $V_{PP}$ | Programming Supply Voltage | 12.5 | 13.0 | V |
| $I_{PP}$ | Programming Supply Current | | 50 | mA |
| 1/TCLCL | Oscillator Frequency | 4 | 6 | MHz |
| TAVGL | Address Setup to $\overline{PROG}$ | 48TCLCL | | |
| TGHAX | Address Hold After $\overline{PROG}$ | 48TCLCL | | |
| TDVGL | Data Setup to $\overline{PROG}$ | 48TCLCL | | |
| TGHDX | Data Hold After $\overline{PROG}$ | 48TCLCL | | |
| TEHSH | $P_{2.7}$ ($\overline{ENABLE}$) High to $V_{PP}$ | 48TCLCL | | |
| TSHGL | $V_{PP}$ Setup to $\overline{PROG}$ | 10 | | µs |
| TGHSL | $V_{PP}$ Hold After $\overline{PROG}$ | 10 | | µs |
| TGLGH | $\overline{PROG}$ Width | 90 | 110 | µs |
| TAVQV | Address to Data Valid | | 48TCLCL | |
| TELQV | $\overline{ENABLE}$ to Data Valid | | 48TCLCL | |
| TEHQZ | Data Float After $\overline{ENABLE}$ | 0 | 48TCLCL | |
| TGHGL | $\overline{PROG}$ High to $\overline{PROG}$ Low | 10 | | µs |



## EPROM PROGRAMMING AND VERIFICATION WAVEFORMS




WF025692

For Programming conditions, see Figures 1 and 2.
For Verification conditions, see Figure 3.

# Software Routines

## DUAL DATA POINTER ROUTINES

The Dual Data Pointer feature enhances the manipulation of external memory by providing an easy way to use two separate 16-bit pointers with external memory and to selectively switch between them. This can increase execution speed of many functions considerably while at the same time reducing the number of required instructions. For instance, in block-move operations in external RAM, Dual Data Pointers can show more than 100% speed improvement using less than 65% of the original code space.

The following registers are associated with the Dual Data Pointers.

| | | |
|---|---|---|
| Data Pointer Low | (DPL) | DPTR0 (Original Data Pointer) |
| Data Pointer High | (DPH) | |
| Data Pointer Low 1 | (DPL1) | DPTR1 (New Data Pointer) |
| Data Pointer High 1 | (DPH1) | |
| Data Pointer Selection | (DPS) | |

The six instructions that refer to "DPTR" now refer to the data pointer that is currently enabled, either DPTR0 or DPTR1. DPS is used to selectively enable the data pointers.

```
INC     DPTR          ; Increment Data Pointer
MOV     DPTR,#data16  ; Loads DPTR with 16-bit constant
MOVC    A,@A+DPTR     ; Move code byte relative to DPTR to Acc
MOVX    A,@DPTR       ; Move external RAM to Acc
MOVX    @DPTR,A       ; Move Acc to external RAM
JMP     @A + DPTR     ; Jump indirect relative to DPTR
```

For complete information on the Dual Data Pointer feature, consult the 80C521/80C321 Data Sheet.

### Block Move in External RAM

Data Pointers are used extensively in the 8051 Family when a block of data is moved from a source area to a destination area in external RAM. The following examples illustrate the speed improvement and code space efficiency gained by using the Dual Data Pointer feature.

The first example shows a 32-byte block move executed by a traditional, single data pointer 8051 Family member. Contrast this with the second example which shows a 32-byte block move executed using the Dual Data Pointers.

With Dual Data Pointers, one data pointer can be assigned to the source address and the other to the destination address. The code then switches between the two data pointers without having to save and restore a data pointer. The speed improvement of this 32-byte block move is 115% and uses less than 57% of the original code space.

***32-Byte Block Move with a Single Data Pointer***

```
; SH and SL are the High and Low source addresses
; DH and DL are the High and Low destination addresses
; Register R5 contains the number of bytes to be moved

;                              Bytes/Cycles
;
        MOV   R5,#32        ;  2  1  - 32 bytes to move
        MOV   DPTR,#SHSL    ;  3  2  - Source    address
        MOV   R1,#SL        ;  2  1  - Initialize source address
        MOV   R2,#SH        ;  2  1
        MOV   R3,#DL        ;  2  1  - Initialize dest. address
        MOV   R4,#DH        ;  2  1

LOOP:   MOVX  A,@DPTR       ;  1  2  - Read byte from source
        MOV   R1,DPL        ;  2  2  - Save source pointer
        MOV   R2,DPH        ;  2  2
        MOV   DPL,R3        ;  2  2  - Load dest. pointer
        MOV   DPH,R4        ;  2  2
        MOVX  @DPTR,A       ;  1  2  - Write byte to dest.
        INC   DPTR          ;  1  2  - Next dest. pointer
        MOV   R3,DPL        ;  2  2  - Save dest. pointer
        MOV   R4,DPH        ;  2  2
        MOV   DPL,R1        ;  2  2  - Load source pointer
        MOV   DPH,R2        ;  2  2
        INC   DPTR          ;  1  2  - Next source pointer
        DJNZ  R5,LOOP       ;  2  2  - Loop till R5=0
```

***32-Byte Block Move with Dual Data Pointers***

```
; SH and SL are the High and Low Source addresses
; DH and DL are the High and Low Destination addresses
; Register R5 contains the number of bytes to move
; DPS = 01 at start (DPTR1 selected)

;                              Bytes/Cycles
;
        MOV   R5,#32        ;  2  1  - 32 bytes to move
        MOV   DPTR,#DHDL    ;  3  2  - DPTR1 = Dest. address
        INC   DPS           ;  2  1  - Switch to DPTR0
        MOV   DPTR,#SHSL    ;  3  2  - DPTR0 = Source address

LOOP:   MOVX  A,@DPTR       ;  1  2  - Read byte from source
        INC   DPS           ;  2  1  - Switch to DPTR1
        MOVX  @DPTR,A       ;  1  2  - Write byte to dest.
        INC   DPTR          ;  1  2  - Next dest. pointer
        INC   DPS           ;  2  1  - Switch to DPTR0
        INC   DPTR          ;  1  2  - Next source pointer
        DJNZ  R5,LOOP       ;  2  2  - Loop till R5=0
```

Suggestion: The fastest way to switch data pointers is to increment the DPS register. Since Bits 7–1 of this register are defined to be zero, the increment (or decrement) operation simply alternates the contents of DPS between 00H and 01H.

**32-Byte Block Move Efficiency**

| | Single Data Pointer | Dual Data Pointers |
|---|---|---|
| Instructions | 19 | 11 |
| Bytes | 35 | 20 |
| Cycles | 839 | 390 |
| Time (µs) @16 MHz | 629.25 | 292.5 |

**N-Byte Block Move Efficiency** (Where N < 256)

| | Single Data Pointer | Dual Data Pointers |
|---|---|---|
| Instructions | 19 | 11 |
| Bytes | 35 | 20 |
| Cycles | 26N + 6 | 12N + 6 |
| Time (µs) @16 MHz | 0.75 (Cycles) | 0.75 (Cycles) |

## Higher Performance Interrupt Routines

When a frequently occurring interrupt uses a data pointer, the overhead required to store and reload it from the main program can be significant. The performance of interrupt-driven systems can be improved by using the Dual Data Pointer feature to assign a data pointer to a frequently called, time-critical interrupt routine.

In the following code, the Main routine uses only DPTR0. The Interrupt routine stores a byte from the Serial Port into an external RAM buffer for later processing. DPTR1 is dedicated for its use.

```
RESET:      SJMP     START

START:      MOV      DPTR,#MAIN      ; Main routine data pointer
            INC      DPS             ; Switch to DPTR1
            MOV      DPTR,#INT       ; Interrupt data pointer
                                     ; initialization
            INC      DPS             ; Switch back to DPTR0
            MOV      IE,#90H         ; Enable Serial Port Int.

; Main routine is using DPTR0
;           ...      .....
;           ...      .....
;                    ----->>> Interrupt occurs
; Program continue
;           ...      .....
;           ...      .....
;           ...      .....
;           ...      .....

; Interrupt routine begins at the Serial Port Vector Address

VECTOR:     INC      DPS             ; Switch to DPTR1
            MOV      A,SBUF          ; Read from Serial Port
            MOVX     @DPTR,A         ; Store byte in RAM Buffer
            INC      DPTR            ; Next Dest. Address
            INC      DPS             ; Switch to DPTR0
            RETI                     ; Return from Interrupt
```

## Full Duplex Transmit/Receive Buffering

Full Duplex Serial Port operation involves simultaneously transmitting and receiving data. Typically a separate transmit buffer and a receive buffer are assigned in the external memory. When a receive interrupt occurs, the data received in the serial port receive register is saved in the external receive buffer. When data is ready to be transmitted, the data from the external transmit buffer is loaded into the transmit register of the serial port. With two data pointers available, one can be assigned to the transmit buffer and the other to the receive buffer. Thus, the interrupt overhead can be reduced.

```
; Initialize
            MOV       DPS,#00H        ; Select DPTR0
            MOV       DPTR,#XMTBUF    ; Transmit RAM buffer address
            INC       DPS             ; Switch Data Pointers
            MOV       DPTR,#RCVBUF    ; Receive RAM buffer address

; Serial Port Interrupt Routine

INT_BEGIN:  JB        RI,RECEIVE      ; Receive a Byte
            JB        TI,TRANSMIT     ; Transmit a Byte
            SJMP      ERROR           ; Error - neither bit set

TRANSMIT:   CLR       TI              ; Clear Flag
            MOV       DPS,#00H        ; Select DPTR0
            MOVX      A,@DPTR         ; Load data from memory
            MOV       C,P             ; Move Parity bit to carry bit
            CPL       C               ; Set ODD Parity
            MOV       A.7,C           ; Append to bit 7 in Acc
            MOV       SBUF,A          ; Load data to transmit
            INC       DPTR            ; Next Byte
            RETI

RECEIVE:    CLR       RI              ; Clear Flag
            MOV       A,SBUF          ; Load received byte to Acc
            JNB       P,ERROR         ; Jump if Parity error
            ANL       A,#7FH          ; Mask off Parity bit
            MOV       DPS,#01H        ; Select DPTR1
            MOVX      @DPTR,A         ; Store byte in memory
            INC       DPTR            ; Next byte
            RETI

ERROR:      ...       .....           ; Error Handler
            RETI
```

## Tree Structure Manipulation

The Dual Data Pointers can be useful in applications involving data structures containing pointer references, such as trees. For instance in a tree search algorithm, the node currently being searched and its parent may have their addresses stored in the Dual Data Pointers. Even though other required pointers will necessarily be pushed onto the stack, most operations will involve only the two most recently used data pointers. Thus the search algorithm will execute more quickly.

In Figure 8-1, note that DPTR1 can be used to step through another link at node "Dave", as soon as DPTR0 is through accessing all of the links in leaf-node "Jim". The pointer for node "Randy" is located on the stack at this point.

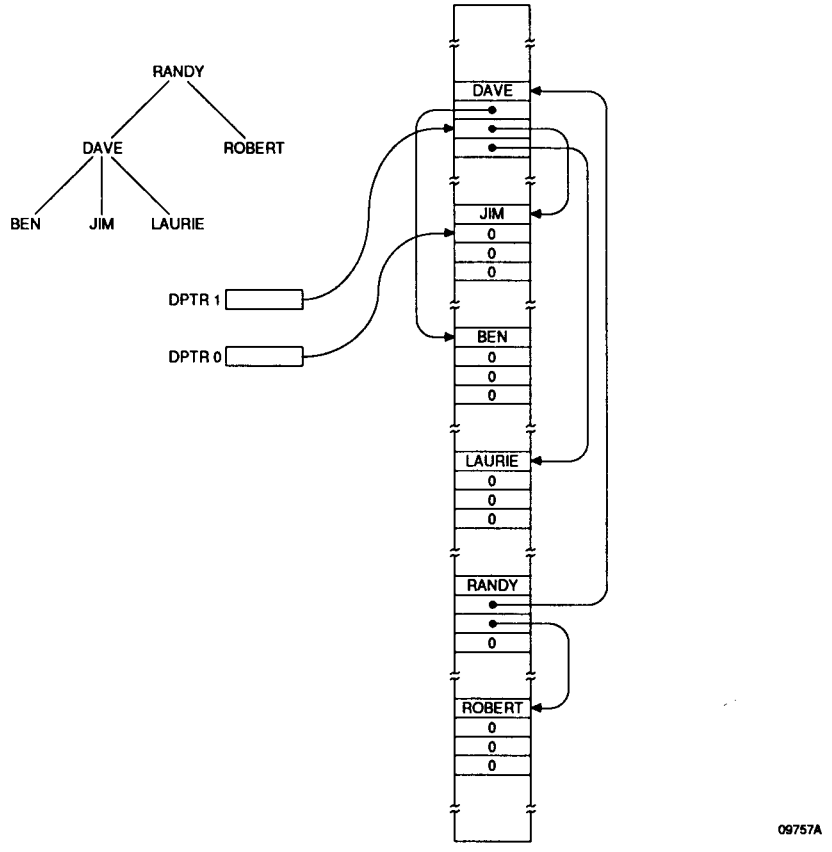



**Figure 8-1. Tree Structure in External Memory**

## ROM Table Access

Use of the Dual Data Pointers need not be limited to manipulations in external RAM. For instance, one or both data pointers can be assigned to ROM tables in program memory space. Table access is then performed with the MOVC instruction. In this way, the base address of a ROM table can reside in one of the data pointers, improving the effective access time.

## Creating an External Stack

For applications that require large amounts of data to be stored on a stack, the internal RAM space may not be sufficient to contain it. This is especially true if the internal RAM is already being used extensively.

With Dual Data Pointers, one data pointer can be assigned specifically to an external stack space in external RAM. The following code provides Push and Pop subroutines using DPTR1 as a stack pointer. Two examples are shown. In the first example the external stack may be up to 64K bytes in length. The second example executes more quickly, but the external stack is limited to 256 bytes.

***Example 1 — 64K byte External Stack Space***

```
; Both Routines Push/Pop bytes from/to the Accumulator

PUSH:       INC     DPS             ; Switch to DPTR1
            INC     DPTR            ; Increment DPTR1
            MOV     @DPTR,A         ; Move Accumulator to Stack
            INC     DPS             ; Switch back to DPTR0
            RET

POP:        INC     DPS             ; Switch to DPTR1
            MOV     A,@DPTR         ; Move Stack byte to Acc
            CJNE    DPL1,#00H,LOW   ;
            DEC     DPH1            ;

LOW:        DEC     DPL1            ; Decrement DPTR1
            INC     DPS             ; Switch back to DPTR0
            RET
```

***Example 2 — 256 Byte External Stack Space***

```
PUSH:       INC     DPS             ; Switch to DPTR1
            INC     DPL1            ; Increment DPTR1
            MOV     @DPTR,A         ; Move Accumulator to Stack
            INC     DPS             ; Switch back to DPTR0
            RET

POP:        INC     DPS             ; Switch to DPTR1
            MOV     A,@DPTR         ; Move Stack byte to Acc
            DEC     DPL1            ; Decrement DPTR1
            INC     DPS             ; Switch back to DPTR0
            RET
```

## WATCHDOG TIMER ROUTINES

The Watchdog Timer (WDT) is a specially designed timer that will reset the chip upon reaching a pre-programmed time interval. Once started it cannot be disabled, except by a reset. It allows safe recovery from problems resulting from electrostatic discharge, external noise, unexpected input conditions or external events, and programming anomalies. Two registers are associated with the Watchdog Timer:

Watchdog Selection (WDS)
Watchdog Key (WDK)

WDS is used to set up the programmed time intervals and indicates the cause of the last reset — a Watchdog or Software Reset versus a Hardware or Power-on Reset. Sixteen time intervals are programmable varying from 128 µs to 4 s (at 12 MHz).

WDK is used to enable the Watchdog Timer as well as clear it. When the Watchdog Timer is cleared, its present count is set to zero, but it continues to increment. For complete information on the Watchdog Timer, consult the *80C521/80C321 Data Sheet.*

### WDT Enable, Clear, and Reset Cause

The following example shows a method of setting up the Watchdog time value to 16.384 ms assuming a 12 MHz clock. The Watchdog Timer is then enabled.

```
; Enable Watchdog Timer
            MOV     WDS,#07H        ; Set up 16.384 msec
                                    ;
            MOV     WDK,#A5H        ; Write first key value
            MOV     WDK,#5AH        ; Write second key value
                                    ; Watchdog timer is 'enabled'
```

Once the Watchdog Timer is enabled, a "clear" sequence should be performed at intervals not exceeding the 16.384 ms time value. The enabling sequence may be used to clear the Watchdog Timer.

```
; Clear Watchdog Timer
            MOV       WDK,#A5H        ; Write first key value
            MOV       WDK,#5AH        ; Write second key value
                                      ; Watchdog Timer is 'cleared'
                                      ; but continues to increment.
```

To test whether the last reset was caused by a Watchdog or Software Reset the following code may be used. If the Reset Cause bit is set, then a Watchdog or Software Reset has occurred.

```
; Reset Cause Identification
            MOV       A,WDS           ; Read Watchdog Selection reg.
            JB        A.7,WDRST       ; Jump if Reset Cause bit is
                                      ; set, else continue
WDRST:      ...       .....           ; Notify external circuitry
```

The security of the Watchdog Timer is not adversely affected by interrupts that may occur in between the writing of the 'A5' and '5A' values to the WDK Register. Thus, if necessary, the user may include clear operations within both a main routine and the interrupt routines. Furthermore, the user need not disable interrupts during the enable/clear operations.

Once the 'A5' is written to WDK, the interrupt routine can only affect the Watchdog Timer in three ways: 1) it can go ahead and enable/clear the Watchdog Timer with a '5A'. (The subsequent '5A' written by the main routine will then have no effect); 2) it can write another 'A5'. This affects neither the Watchdog Timer nor the main routine; or 3) it can cause a Software Reset by writing a value other than 'A5' or '5A'. Any routine, though, can be written to generate the Software Reset.

## Power-Down Operation

While the Watchdog Timer is enabled, the Power-Down mode is disabled. The user's code may still attempt to enable the power-down operation (by writing a value 1 to the PD bit in the PCON register), however, the PD bit will remain at 0, and the power-down operation will not take place. If the WDT has *not* been enabled, the power-down operation can proceed normally.

To enter Power-Down mode when the WDT is enabled, the WDT must first be disabled via a Hardware Reset, Software Reset, or Watchdog Reset. The easiest is the Software Reset. This can be accomplished by writing an 'A5' to the Watchdog Key (WDK) register followed by a value other than 'A5' or '5A'. This generates an immediate reset, equivalent to a Hardware Reset except that the Reset-Cause bit is set.

The code below uses the Reset-Cause bit and the Internal RAM (which is not modified by a reset). If the Reset-Cause bit is set, and a special Power-Down-Status byte in internal RAM contains '88H', then the Power-Down mode will be entered by the program code.

```
; WDS = 7 sets up a Watchdog time of 16.384 msec @ 12 MHz.
; 'A5' followed by '5A' written to WDK enables the WDT.
; RAM location 50H is Power Down Status
;       00 implies Power-Down has not been requested.
;       88 implies Power-Down has been requested.

RESET:      MOV     A,WDS           ; Read Reset cause bit in WDS
            JB      A.7,WDRST       ; Jump if reset caused by WDT
            LJMP    MAIN            ; Go on to the Main Routine

WDRST:      MOV     R0,#50H         ; Address Power Down Status
            CJNE    @R0,#88H,MAIN   ; If Power-Down was not
                                    ; requested, then jump and
                                    ; continue normally
            MOV     PCON,#02H       ; else enter Power-Down Mode

MAIN:       MOV     50H,#00H        ; Clear Power Down Status
            MOV     WDS,#07H        ; Set up time value for WDT
            MOV     WDK,#A5H        ; Write first key value
            MOV     WDK,#5AH        ; Write second key value
;           ...     .....           ; WDT is now enabled.
;
; Main Routine Continues..
;
; In Main Routine whenever Power-Down is required, execute:

            MOV     50H,#88H        ; Request Power Down operation
            MOV     WDK,#A5H        ; Write first key value
            MOV     WDK,#11H        ; Software Reset generated -
            NOP                     ; Execution begins at RESET
                                    ; in 3 machine cycles.
```

## Testing the Watchdog Timer

Two methods can be used to verify that the WDT is enabled after the enabling sequence has been written (rather than simply waiting for the WDT to reset to occur). Method I can be used as a precautionary measure after the enabling sequence or at various points within the code. It may also be used to confirm the time interval programmed into the WDT for applications that occasionally use different Watchdog time intervals. Method II can be used as a debugging test during program development.

***Method I***

```
            MOV     WDS,#07H        ; Set the Watchdog time to
                                    ; 16.384 ms @12 MHz
            MOV     WDK,#A5H        ; Write first key value
            MOV     WDK,#5AH        ; Write second key value
                                    ; WDT should now be enabled
            MOV     WDS,#00H        ; Attempt to rewrite contents
                                    ; of the WDS Programmed Time
            MOV     A,WDS           ; Read contents of WDS into Acc
            CJNE    A,#07,ERROR     ; If contents are not 07, then
                                    ; jump to ERROR.
            ...     .....           ; The WDT is enabled and the
                                    ; ACC now holds the programmed
                                    ; time value that the WDT is
                                    ; currently using.

ERROR:      ...     .....           ; Watchdog Timer never received
                                    ; the correct 'A5-5A' sequence
```

***Method II***

```
            MOV     WDS,#07H        ; Set the Watchdog time to
                                    ; 16.384 ms @12 MHz
            MOV     WDK,#A5H        ; Write first key value
            MOV     WDK,#5AH        ; Write second key value
                                    ; WDT should now be enabled

WAIT:       MOV     A,WDS           ;
            JNB     A.5,WAIT        ; Wait 4.096 ms for the TV bit
                                    ; to be set
            ...     .....           ; WDT enabled and incrementing
```

## Using the Watchdog Timer as a Standard Timer

The Timer Verification (TV) bit in the WDS register can be used to implement certain types of timer functions through polling. Once the WDT is enabled, the TV bit will toggle every 4.096 ms (at 12 MHz) until either the WDT overflows, or the WDT is cleared. (The TV bit is initially a 0 after any reset.) When the WDT overflows, a WDT Reset occurs clearing the TV bit. When the WDT is cleared, the TV bit is cleared, but begins toggling again at the same rate. If bits PT3–PT0 are set to '0101' or less, then a WDT Reset will occur before the TV bit toggles.

The following code uses the MAIN polling loop of an application to watch for the TV bit to toggle. It uses the TV bit to output a 25% duty-cycle pulse on Port Pin 1.7 with a period of 1.049 s at 12 MHz.

```
; R6   If 0, then Pulse is Low
;      If 1, then Pulse is High
; LTIME = Low Time, the number of 4.096 ms units equaling
;         786 ms = 192
; HTIME = High Time, the number of 4.096 ms units equaling
;         262 ms = 64
; OLD_TV = A Direct RAM byte whose bit 0 location contains
;          the last read value of TV
; R7 Contains number of TV toggles left to go before P1.7
;    switches

INIT:       MOV     WDS,#0FH        ; Set the Watchdog time to 4 S
                                    ; at 12 MHz (safest value)
            CLR     P1.7            ; Set Port Pin to 0
            MOV     R6,00H          ; Pulse is Low
            MOV     R7,LTIME        ; Load Low Time
            MOV     WDK,#A5H        ;
            MOV     WDK,#5AH        ; WDT is now enabled. TV begins
                                    ; toggling
            MOV     R6,00H          ; Pulse is Low
            MOV     R7,LTIME        ; Load Low Time
            MOV     OLD_TV,#00H     ; Old TV bit equals 0 (TV's
                                    ; reset value)

MAIN:       ...     ....
            MOV     A,WDS           ;
            MOV     C,A.5           ; Move TV bit to Carry
            MOV     A,OLD_TV        ; Move Old TV bit to ACC.0
            ADDC    A,#00           ; Add TV bit (in Carry) to Old
                                    ; TV bit
            JB      A.0,TOGGLE      ; If A.0 = 1, then the TV bit
                                    ; has toggled

CONTINUE:   ...     ....
;           ...     ....

TOGGLE:     INC     OLD_TV          ; Toggle Old TV bit in OLD_TV
                                    ; byte
            DJNZ    R7,CONTINUE     ; If R7 is not 0, then it is
                                    ; not time to toggle P1.7 yet
            CPL     P1.7            ; Toggle Port Pin
            MOV     WDK,#A5H        ;
            MOV     WDK,#5AH        ; Clear WDT, TV starts again
            CJNE    R6,#00,GO_LOW   ; If R6 is 0, then load HTIME
                                    ; else load LTIME
            MOV     R7,HTIME        ; Load High Time
            INC     R6              ; Pulse is High now
            SJMP    CONTINUE        ;

GO_LOW:     MOV     R7,LTIME        ; Load Low Time
            DEC     R6              ; Pulse is Low now
            SJMP    CONTINUE        ;
```

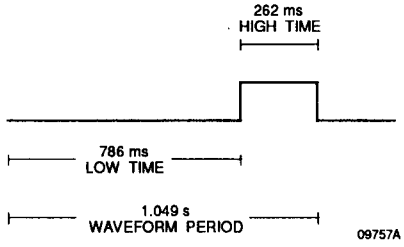



**Figure 8-2. P1.7 Output — 25% Duty Cycle**

## SOFTWARE RESET ROUTINES

A Software Reset may be accomplished through the Watchdog Timer. This "software generated" Watchdog Reset occurs regardless of whether or not the Watchdog Timer was previously enabled. If the Watchdog Timer was enabled, it will be disabled following the reset. The Software Reset is functionally equivalent to the Watchdog Reset.

Two write operations are required to initiate a Software Reset to greatly reduce the chance of unintentional Software Reset generation. More information is available in the *80C521/80C321 Data Sheet.*

### Using Software Reset

Whether or not the Watchdog Timer is being used, the Software Reset feature of the Watchdog Timer may be used to increase the reliability of the program code. For instance, the detection of an unusual hardware error can be followed by a jump to the following code which will always cause a Software Reset.

```
CLR     EA              ; Disable all interrupts.
                        ; Optional
MOV     FLAG,#88H       ; Optional
MOV     WDK,#A5H        ; Write first key value
MOV     WDK,#11H        ; Write a non-A5, non-5A value.
                        ; Software Reset has now been
                        ; generated via the WDT.
NOP                     ; Optional
```

If the Watchdog Timer is cleared within an interrupt routine, that interrupt should be disabled before executing a Software Reset sequence. If the interrupt occurs between the two writes to WDK, and then clears the Watchdog Timer, a Software Reset will not be generated.

To distinguish between a Watchdog Reset and a Software Reset (or separate causes of a Software Reset), a flag value may be written to internal RAM. This flag can be used in combination with the Reset-Cause bit to distinguish between the reset types. An example of this method is shown in the "Power Down Operation" software routine.

After the value '11H' is written to WDK, execution begins at 0000H in three machine cycles. One machine cycle of normal execution takes place after the '11H' is written. Thus, the NOP can be included for safety. Since all registers are initialized during reset, and all external operations take two machine cycles, the only operation that could possibly affect operation after the Software Reset would be a one-cycle write to internal RAM.

## Improving Reliability with Software Reset

For additional reliability, the following instruction sequence may be placed in any unused ROM program space:

```
            NOP       ; First unused ROM location
            NOP
;
            MOV       WDK,#A5H
            MOV       WDK,#00H        ; Software Reset generated
            NOP
            NOP
;
            MOV       WDK,#A5H
            MOV       WDK,#00H        ; Software Reset generated
            NOP
            NOP
;
;           ...       ....            ; Continue repeating the 4-instruction
;           ...       ....            ; sequence
;

SOFTRESET:  MOV       WDK,#A5H
            MOV       WDK,#00H        ; Software Reset generated
            NOP
            NOP
            SJMP      SOFTRESET       ; Last unused ROM location
```

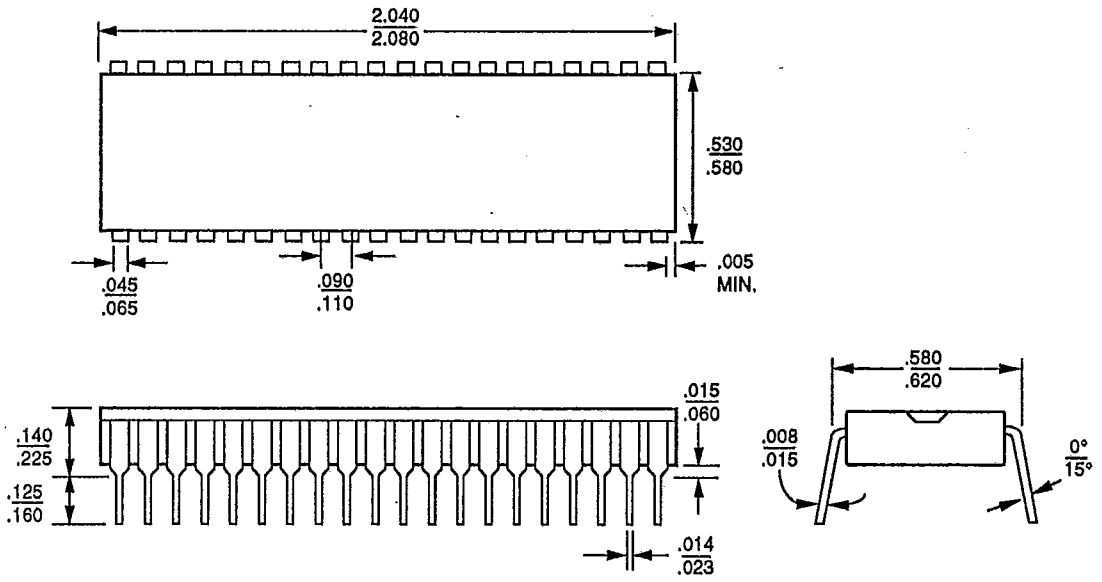
If the program counter branches to any byte of this code (other than the second byte of the SJMP instruction), a Software Reset will be quickly generated. The NOP instructions are used to force the program counter to adjust itself to an instruction boundary.

CHAPTER 11 T-90-20

# Package Outlines

## PHYSICAL DIMENSIONS*

### Plastic Dual-In-Line Package (PD)
### PD 040



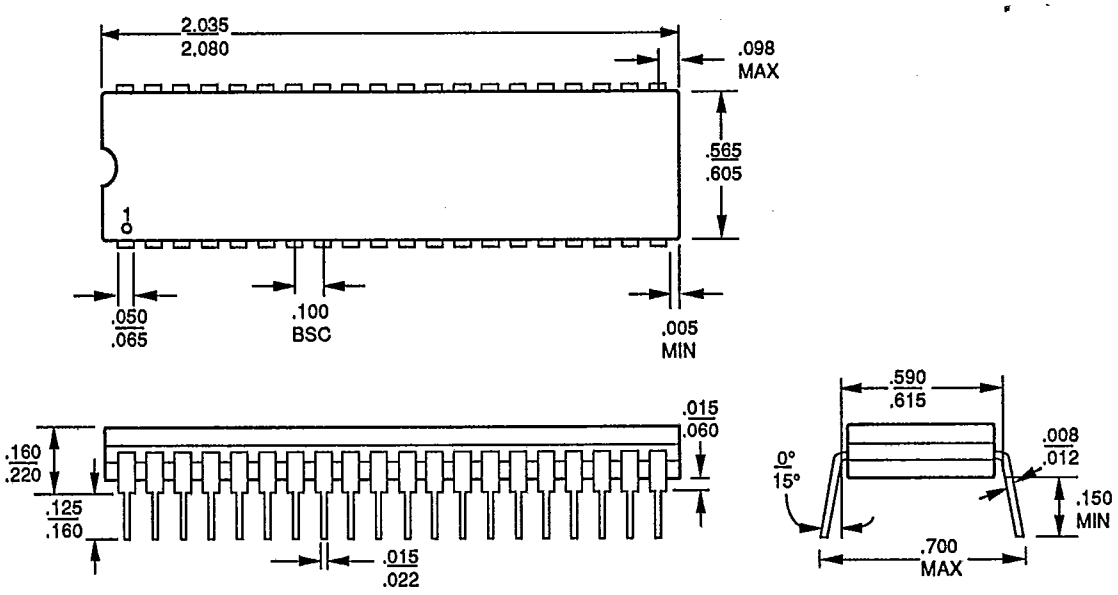

PID# 06823B

* For reference only.

NOTE: Package dimensions are given in inches. To convert to millimeters, multiply by 25.4.

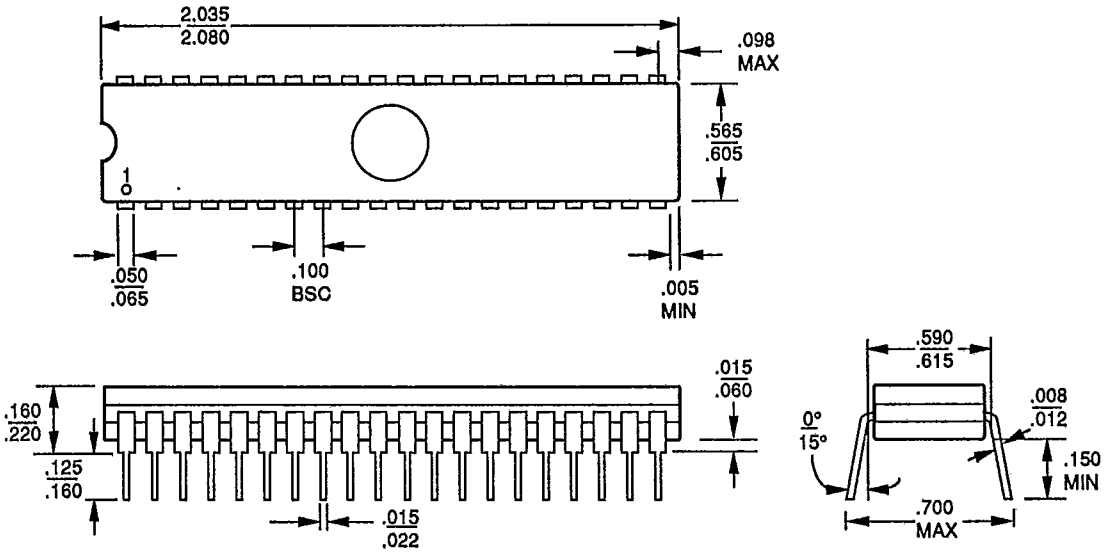
## Ceramic Hermetic Dual-In-Line Packages (CD/CDV)

### CD 040




06824C

### CDV 040



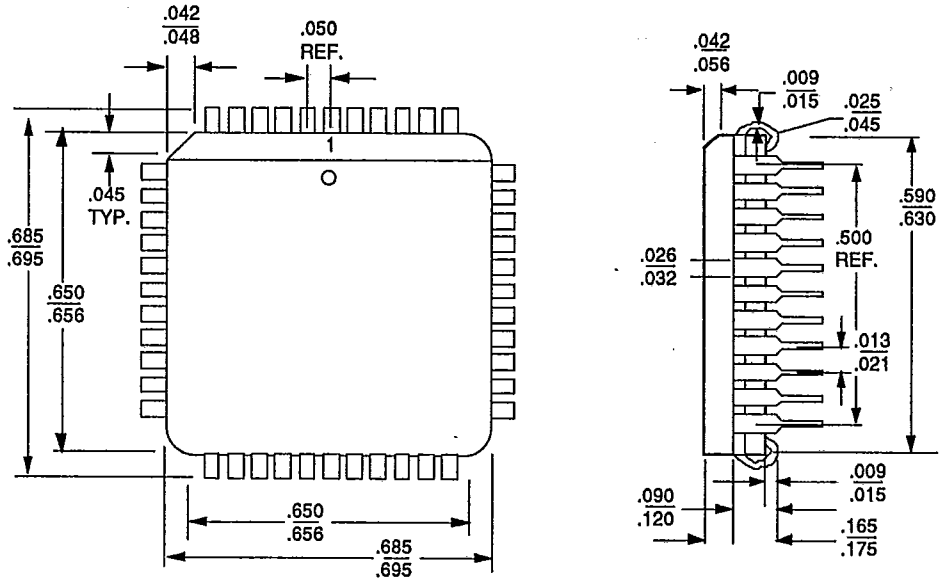

07880C

NOTE: Package dimensions are given in inches. To convert to millimeters, multiply by 25.4.

## Plastic Leaded Chip Carriers (PL)

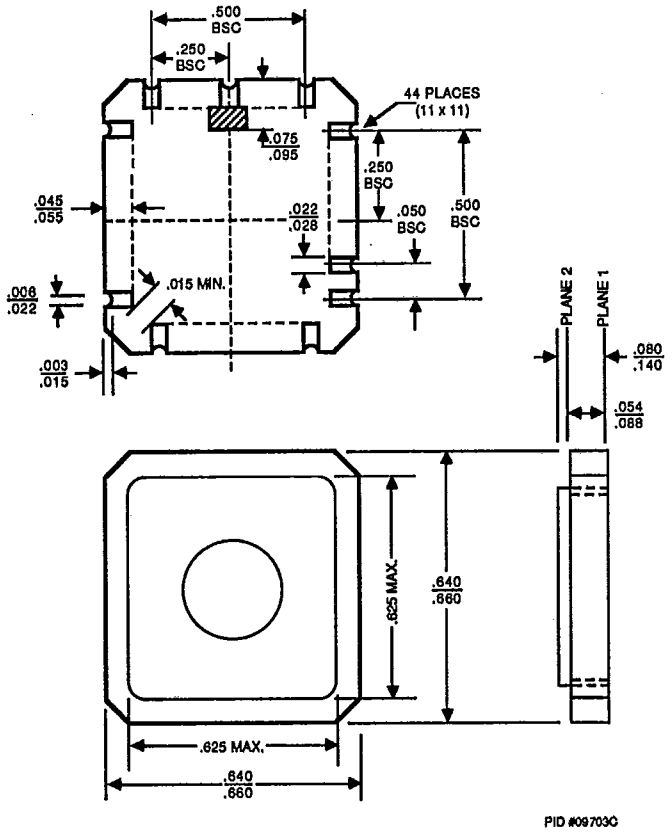
## PL 044

T-90-20




NOTE: Package dimensions are given in inches. To convert to millimeters, multiply by 25.4.

## Ceramic Leadless Chip Carrier (CLV)
## CLV 044




NOTE: Package dimensions are given in inches. To convert to millimeters, multiply by 25.4.