AP-728

APPLICATION NOTE

# Interfacing an Intel386™ EX Microprocessor to an 82527 CAN Controller

**GREG SCOTT**
TECHNICAL MARKETING ENGINEER

January 1996

Order Number: 272790-001

Information in this document is provided in connection with Intel products. Intel assumes no liability whatsoever, including infringement of any patent or copyright, for sale and use of Intel products except as provided in Intel's Terms and Conditions of Sale for such products.

Intel retains the right to make changes to these specifications at any time, without notice. Microcomputer Products may have minor variations to this specification known as errata.

*Other brands and names are the property of their respective owners.

†Since publication of documents referenced in this document, registration of the Pentium, OverDrive and iCOMP trademarks has been issued to Intel Corporation.

Contact your local Intel sales office or your distributor to obtain the latest specifications before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation
P.O. Box 7641
Mt. Prospect, IL 60056-7641

or call 1-800-879-4683

# INTERFACING AN INTEL386™ EX MICROPROCESSOR TO AN 82527 CAN CONTROLLER

## CONTENTS

PAGE

## 1.0 INTRODUCTION

The purpose of this application note is to describe one method of interfacing an Intel 82527 CAN controller to the local bus of an Intel386™ EX microprocessor at 25 MHz. A basic understanding of the Intel386 EX microprocessor is assumed. This description includes a verification that all AC timing specifications are satisfied and a brief description of the 82527 interface modes and clocking structure.

### 1.1 Interface Suggestions

- The 82527 is used in asynchronous mode 3.
- Use the DSACK0# signal to assert the READY# signal.
- The 82527 RESET# pin is tied to a Intel386 EX processor port pin or reset circuit.
- The Intel386 EX processor is operating at 25 MHz.

### 1.2 Top 6 Issues

- The AS and E pins of the 82527 must be tied to $V_{CC}$.
- 82527 RESET# pin must be asserted low for 1 ms minimum.
- Default condition of 82527 for mode 3 sets MCLK = SCLK/2, for best access time, this should be changed to MCLK = SCLK.
- An external PAL (Programmable Array Logic) is required. Note, the delay through the PAL must be considered.
- A buffer must be used between the Intel386 processor and the 82527 data lines.
- The $t_{DVCH}$ timing specification for asynchronous mode 3 is 20 ns. All the 82527s meet this timing specification. This change will be reflected in the next datasheet (release date Q1'96).

## 2.0 CPU INTERFACE MODES

The 82527 supports six CPU interface modes allowing users to connect the 82527 to host-CPUs of various architectures. The CPU interface modes are:

- 8-bit Intel multiplexed (mode 0)
- 16-bit Intel multiplexed (mode 1)
- 8-bit non-Intel multiplexed (mode 2—AS, E, R/W#)
- 8-bit non-multiplexed synchronous (mode 3)
- 8-bit non-multiplexed asynchronous (mode 3)
- Serial (SPI compatible)

When interfacing the Intel386 EX microprocessor to the 82527, the 8-bit, non-multiplexed asynchronous mode (Mode 3) of the 82527 is used.

## 3.0 CLOCKING STRUCTURE

The operation of the 82527 is controlled by two clocks, the system clock (SCLK) and the memory clock (MCLK). The SCLK is derived from the external oscillator, while the MCLK is based off the frequency of the SCLK. The bit timings for all CAN bus communications are based on the frequency of the SCLK, while the MCLK provides clocking for all read and write operations to the 82527 RAM via the Intel386 EX processor/82527 interface.

The frequency of the SCLK may be equal-to or one-half the external oscillator frequency and is defined by the value of the DSC bit in the CPU interface register. The maximum frequency of the SCLK is 10 MHz as specified in the 82527 datasheet (order number 272250). An 8 MHz SCLK frequency is typically sufficient to interface the 82527 to a 1 Mbit/sec CAN bus.

The frequency of the MCLK may be equal-to or one-half the frequency of the SCLK, and is defined by the value of the DMC bit in the CPU interface register. The maximum frequency of the MCLK is 8 MHz, as specified in the 82527 datasheet (order number 272250). The default condition of the CPU interface following a reset is 61h. This default condition configures the SCLK to XTAL/2 and the MCLK to SCLK/2. In mode 3, the default condition for CLKOUT is XTAL/2. In all other modes the default condition is CLKOUT = XTAL.

The 82527 contains two types of registers: High-Speed Registers, HSRs (locations 02H, 04H, and 05H) and normal or Low-Speed Registers, LSRs (all registers except 02H, 04H, and 05H). Read and write operations to the LSRs occur over a syncronous internal bus which is clocked by the MCLK. HSRs 02H, 04H and 05H are decoupled from the internal bus, allowing them to be accessed more quickly by the Intel386 EX microprocessor. HSRs 04H and 05H are implemented for the double-read operation. The double-read operation is used for interfacing the 82527 with faster CPUs that do not allow for long access time. The Intel386 EX microprocessor will insert wait states until the READY# signal is received; therefore, double-read operations are not required. For a more detailed explanation of the double read operation, please refer to the 82527 Architectural Overview (order number 272410).

## 4.0 INTERFACING SCHEMATICS FOR MODE 3 ASYNCRONOUS

The schematic in Figure 4-1 demonstrates a minimal hardware interface between the Intel386 EX microprocessor and the 82527.

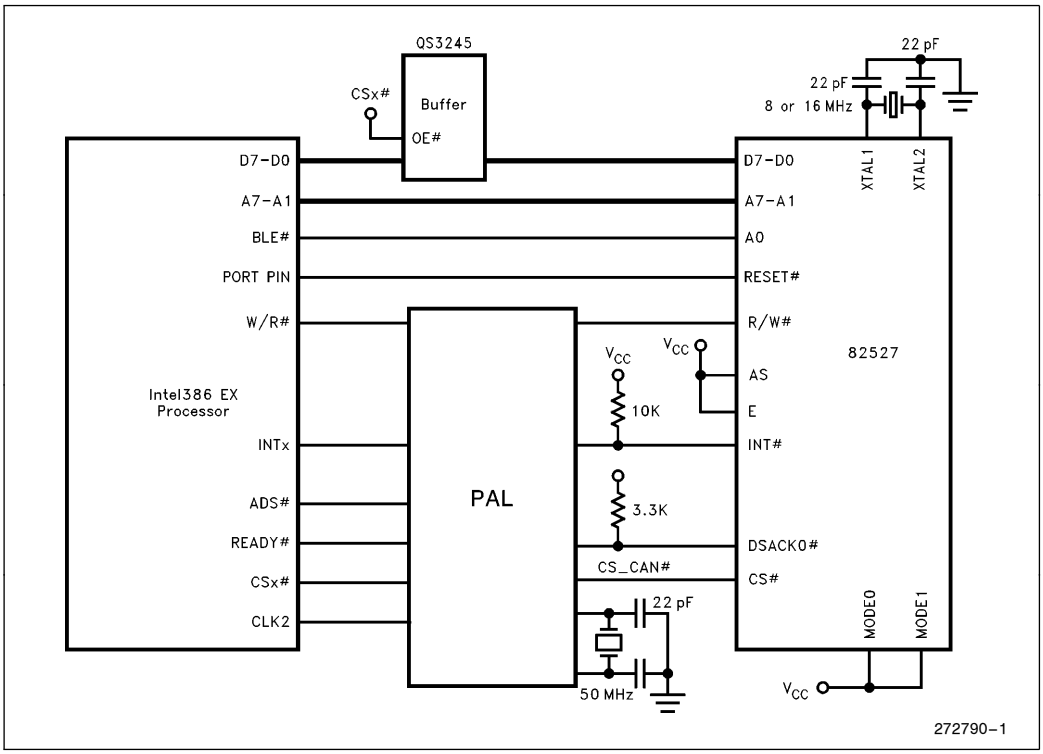


**Figure 4-1. Interface Scheme**

For the lowest access time of the 82527 use either a 16 MHz or an 8 MHz crystal. Consult the crystal manufacturer specifications for proper load capacitance. If the 16 MHz crystal is used then the SCLK = XTAL/2. If the 8 MHz crystal is used then the SCLK = XTAL. The SCLK is programmed by writing to the CPU Interface Register (location 02H). For more information on this register refer to the architectural overview (order number 272410). The RESET# signal for the 82527 may be generated using a port pin on the Intel386 EX Microprocessor, or may be derived by an RC network. The RESET# pin on the 82527 must be asserted to $V_{IL}$ or less for a minimum of 1 ms after $V_{CC}$ is in the operation range to guarantee a proper device reset. The DSACK0# uses a 3.3 kΩ resistor to pull it high. The 82527 internally pulls DSACK0# to 2.4V, then floats the pin. A buffer is used to prevent bus contention during a read cycle.

## 5.0 PAL DESCRIPTION

A PAL is required to satisfy the control and timing requirements of the 82527 and the Intel386 microprocessor. An internal schematic is shown in Figure 5-1. The PAL provides the following.

## 5.1 The CLK2 and PH1 Signals

The PAL provides the 50 MHz clock signal (CLK2) to the Intel386 microprocessor. The CLK2 signal is divided down to 25 MHz creating the PH1 signal. The PH1 signal is used to synchronize DSACK0# with the Intel386 microprocessor bus cycle.

## 5.2 The Interrupt Signal to the Intel386 Microprocessor

The Intel386 microprocessor utilizes a positive transition/level interrupt. The 82527's interrupt is an open drain output which provides an active low interrupt. The PAL inverts the interrupt signal from the 82527 matching the Intel386 microprocessor positive level interrupt. Any available Intel386 microprocessor INTx inputs may be used.

## 5.3 The Intel386 Microprocessor W/R# Signal

The W/R# signal from the Intel386 microprocessor must be interveted to match the 82527 mode 3 R/W# input state requirements.

## 5.4 The READY# Signal

The Intel386 microprocessor READY# signal is based on the DSACK0# signal for the 82527. In mode 3, the 82527 uses the DSACK0# signal to indicate data available during external memory cycles. For the Intel386 microprocessor, the bus cycle is terminated by the READY# signal. The Intel386 microprocessor requires the READY# signal to be asserted at a specific time in the bus cycle. To insure this timing is met, a flip-flop latching on the falling edge of CLK2 is used. The flip-flop latches the result of DSACK0# gated with PH1 and CS_CAN#. The PH1 signal insures that READY# meets the set up and hold timing requirements of the Intel386 microprocessor. The 82527 maintains DSACK0# after CS_CAN# goes high. This results in a second READY# signal being generated on the following PH1. To ensure this does not occur, the DSACK0# signal is also gated with CS_CAN#. The READY# output must be tri-stateable, allowing other devices to assert READY#. The DSACK0# signal on 82527 is an open drain output, consequently an external pull-up resistor is required.

## 5.5 The CS# Signal to the 82527

The CS# signal for the 82527 (CS_CAN#) is generated by the PAL from one of the Intel386 microprocessor's CS# signals (CSx#). In order to meet the timing requirements of both the devices, the CS_CAN# signal is generated differently for a read and a write cycle. For a read cycle, the CS_CAN# signal is directly derived from the CSx# signal. For a write cycle, the 82527 requires that valid data is held on the data lines after CS_CAN# goes high. The Intel386 microprocessor does not hold the data valid long enough to meet the 82527 data hold specification. Therefore, CSx# may not be used directly for CS_CAN#. The PAL generates the CS_CAN# signal by using a state machine, the state diagram is shown in Figure 5-2. The state machine produces the CS_CAN# signal when CSx# and ADS# are low. The CS_CAN# signal is then terminated by the DSACK0# signal. This allows the data to be held valid after CS_CAN# goes high.

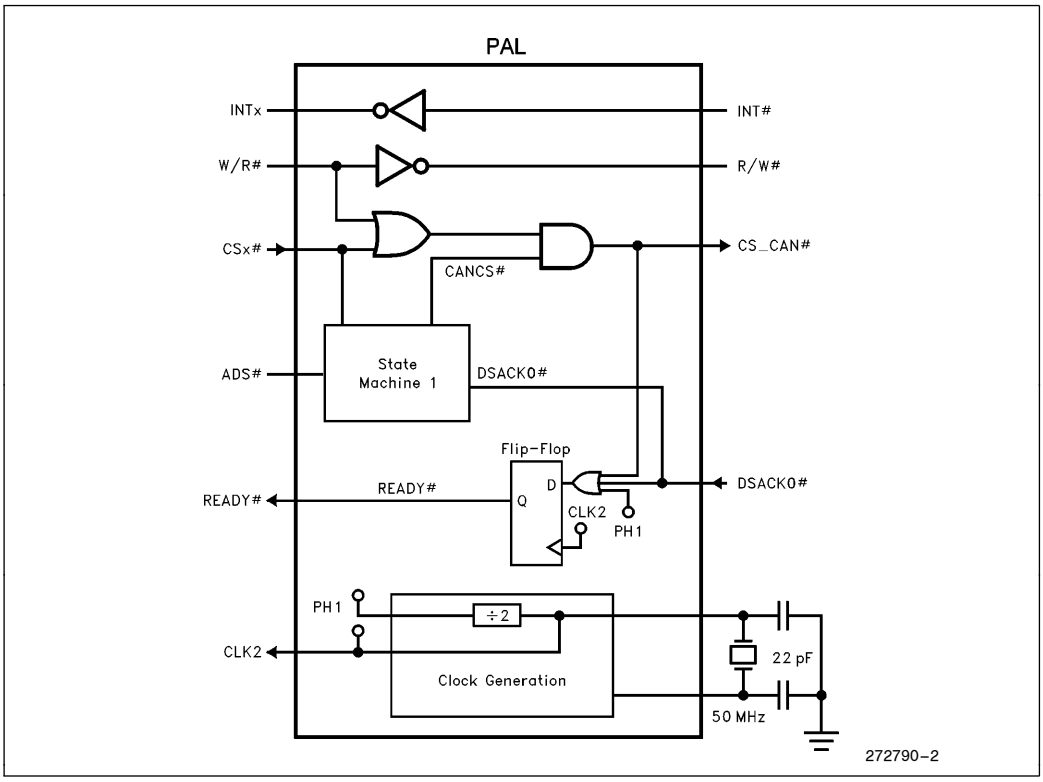


**Figure 5-1. PAL Scheme**

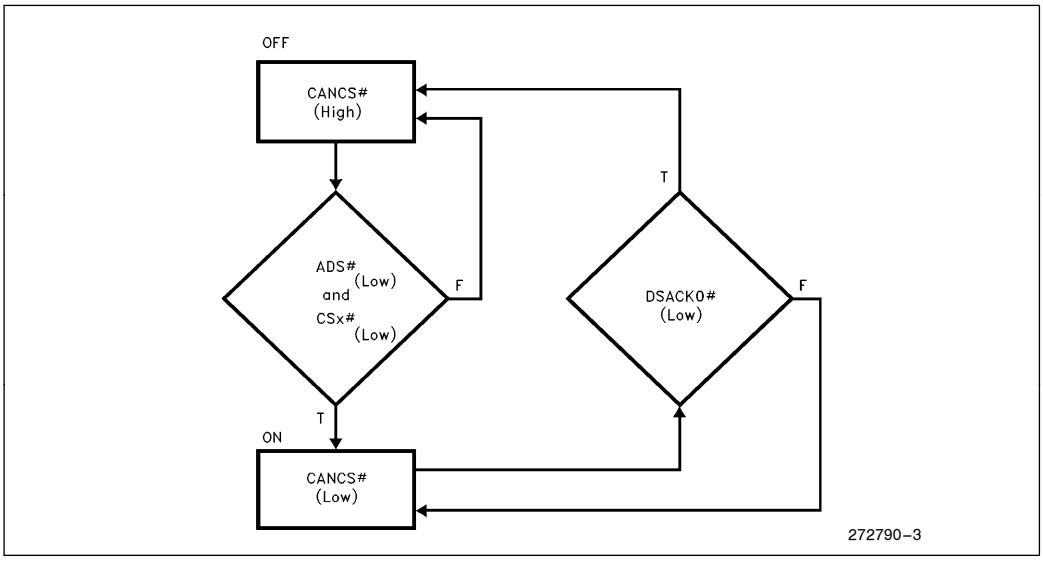


**Figure 5-2. State Machine 1**

## 6.0 TIMING CONSIDERATIONS

| Symbol | Parameters | 82527 @ 8 MHz (1) | Intel386™ EX CPU @ 25 MHz (1) |
|---|---|---|---|
| $t_{AVCL}$ | Address or R/W# Valid to CS# Low Setup | 3 min | (Note 4) |
| $t_{KLDV}$ | DSACK0# Low to Output Data Valid | (HSR) 23 max<br>(LSR) < 0 max | 50 max |
| $t_{CHDV}$ | 82527 Input Data Hold after CS# High | 15 min | 0 min |
| $t_{CHDH}$ | 82527 Output Data Hold after CS# High | 0 min | < 0 min |
| $t_{CHDZ}$ | CS# High to Output Data Float | 35 max | 5 max |
| $t_{CHAI}$ | CS# High to Address Invalid | 7 min | 15 min |
| $t_{CHRI}$ | CS# High to R/W# Invalid | 5 min | 15 min |
| $t_{DVCH}$(3) | CPU Write Data Valid to CS# High | 20 min | 20 min |

**NOTES:**
1. The timing comparisons are in nanoseconds and based on the 82527 datasheet (Order number 272250) and the Intel386 EX Microprocessor datasheet (Order number 272420).
2. The 82527 internally pulls the DSACK0# pin to 2.4V. An external pull-up must be used to drive the signal to 5V.
3. $t_{DVCH}$ is currently 32 ns, $t_{DVCH}$ will be changed to 20 ns, this change will be reflected in the next revision of the 82527 datasheet in Q1'96.
4. $t_{AVCL}$ specification is for the 82527; this timing is dependent on the propagation delay through the PAL.

Figures 6-1 and 6-2 illustrate a read and write bus cycle, respectively. Note that the PH1 signal is the inverse of the PH2 signal illustrated in the Intel386 EX microprocessor datasheet. The case of a "read cycle with a previous write cycle" or a "write cycle with a previous write cycle" is not possible; therefore, it should be ignored.

### 6.1 Read Cycles

During the read cycle, the CS__CAN# signal is generated directly from the CSx# signal of the Intel386 EX microprocessor. This causes the timing spec $t_{CHAI}$ to be violated. However, this spec is not critical during a normal read cycle. Violation of the $t_{CHAI}$ spec may cause erroneous data to be placed onto the data lines after CS__CAN goes high. However, the Intel386 EX microprocessor will have captured the data and completed its read cycle before CS__CAN# goes high. Therefore, if erroneous data is placed on the data lines it will not effect the read cycle for the Intel386 EX microprocessor.

The 82527 will drive the data lines up to 35 ns after CS# goes high, the Intel386 processor expects the data lines to float approximately 5 ns after CS# goes high. The data buffer is used to tri-state the 82527's data lines to prevent bus contention. The buffer is turned on and off by CSx# signal from the Intel386 processor. The buffer must be bi-directional and must meet the 5 ns requirement of the Intel386 EX microprocessor.

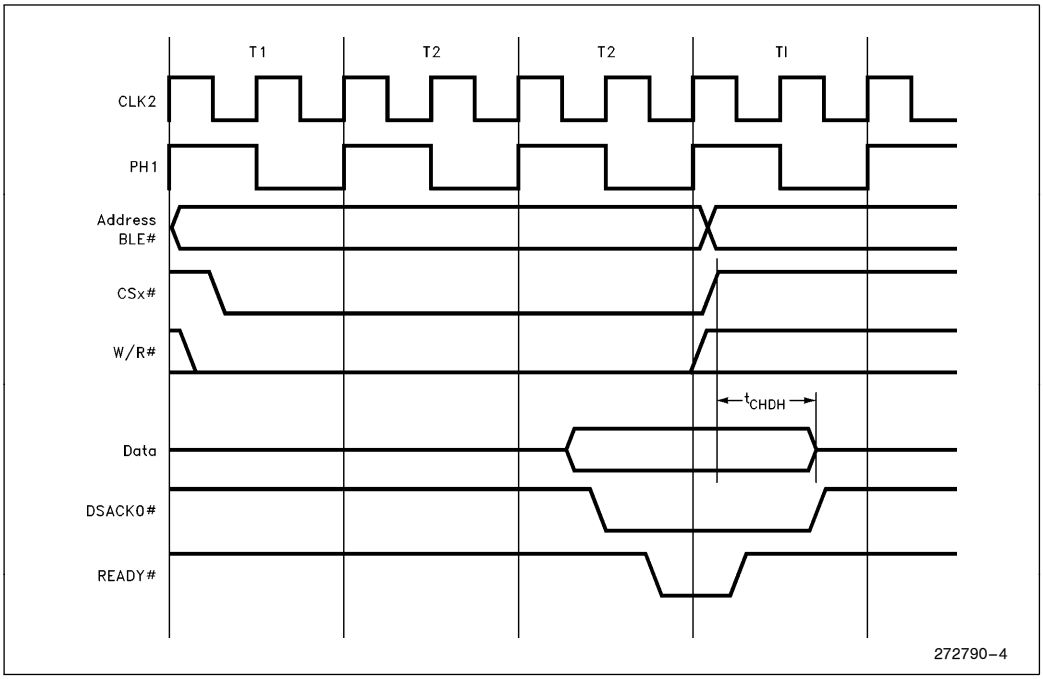


**Figure 6-1. Read Timing Diagram**

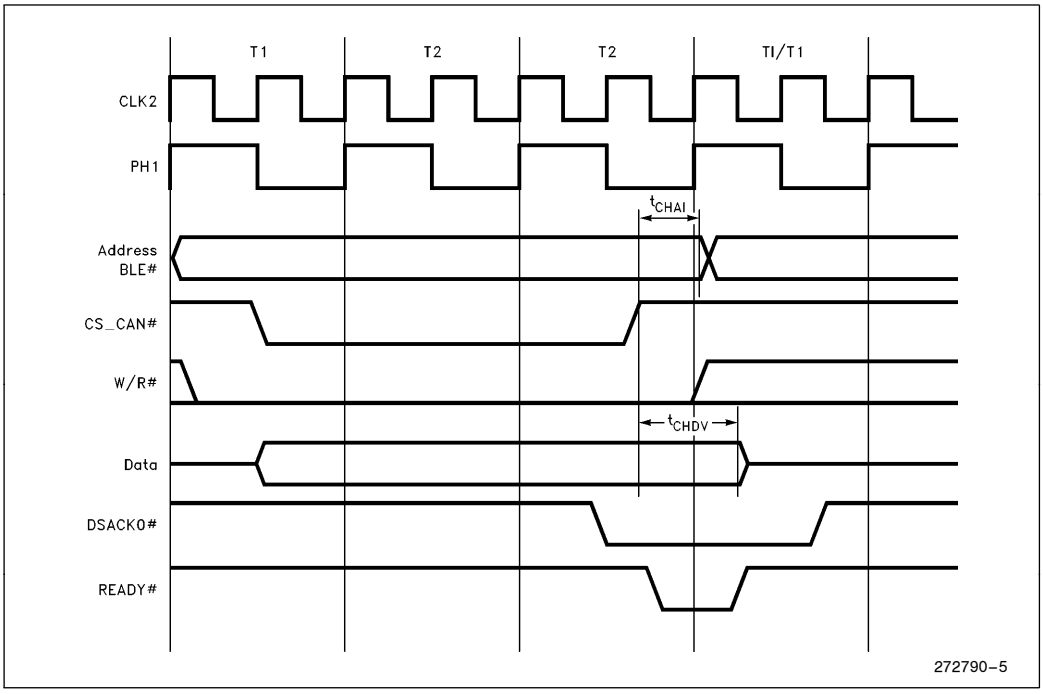


**Figure 6-2. Write Timing Diagrams**

## 6.2 Write Timings

The timing specifications for a write cycle meets the requirements of both the Intel386 EX microprocessor and the 82527; therefore, no special functions are required. The buffer will not have an effect on write cycles.

## 7.0 C-PROGRAM FOR INITIALIZING THE 82527

This program initializes the 82527 CAN controller. For specific details on the functionality of the 82527 registers please refer to the Architectural Overview (Order Number 272410).

- Assumes the 82527 is mapped in I/O space.
- Disables CLKOUT.
- Sets SCLK = XTAL/2 and MCLK = SCLK.
- Sets CAN bus rate to 250 kBits/s.
- Sets message 1 to transmit.
- Sets message 2 to receive.
- Assumes a transceiver is used.

**Note:** This code was compiled using the QuickC for Windows and tested using a modified Intel386 EX microprocessor Eval board.

```
#include <stdio.h>
#include <conio.h>
#define CAN 0xl00                                   /* Defines the starting
                                                       address of the 82527
                                                       chip.*/

void int_527 (void)
{
        int t, x;
        int cr, cir, bcr, bt0, btl, contr0, contrl;
        int gm, mcr, arb;                           /* Initializes the
                                                       counter and pointer
                                                       variables.*/

        cir = CAN + 2;
        outp(cir,40);                               /* Set the CPU Interface
                                                       Register to 40: SCLK =
                                                       XTAL/2, MCLK = SCLK, and
                                                       disables the CLKOUT
                                                       signal.*/

        cr = CAN;
        outp(cr,0x4l | inp(cr));                    /* Sets the CCE (Change
                                                       Configuration Enable)
                                                       bit in the Control
                                                       Register.*/

        bcr = CAN + 0x2f;
        outp(bcr,0x48);                             /* Sets the Bit
                                                       Configuration Register
                                                       to 48. This bypasses the
                                                       input comparator, sets
                                                       logical one as
                                                       recessive, and disables
                                                       the TXl driver. DcR0 and
                                                       DcRl are don't cares
                                                       (set to 0 in this
                                                       case)*/

        bt0 = CAN + 0x3f;
        btl = CAN + 0x4f;
        outp(bt0,0x4l);
        outp(btl,0x67);                             /* Defines the CAN bus
                                                       frequency as 250 kBits/s
                                                       and the sampling mode.*/
```

```
        outp(cr,01);                                    /* Clears the CCE bit,
                                                        preventing write access
                                                        of configuration
                                                        registers.*/

        for (t = 0x10; t <= 0xF0; t = t + 0x10)
        {
                contr0 = t + CAN;
                contrl = contr0 + 1;
                outp(contr0,0x55);
                outp(contrl,0x55);
        }                                               /* This loop resets
                                                        Control Register 0
                                                        and 1.*/

        for (t = 0x06; t <= 0x0b; t++)
        {
                gm = CAN + t;
                outp(gm,0xff);
        }                                               /* This loop sets the
                                                        Global Mask (Standard
                                                        and Extended) to must
                                                        match.*/

        mcr = CAN + 0x16;
        outp(mcr,0x8c);
        mcr + CAN + 0x26;
        outp(mcr,0x84);                                 /* Sets the Message
                                                        Configure Registers for
                                                        message 1 and 2. This
                                                        sets message 1 to
                                                        transmit eight bytes
                                                        using an extend
                                                        identifier and sets
                                                        message 2 to receive
                                                        eight bytes using an
                                                        extended identifier.*/

        for (t = 0x10; t <= 0x20; t = t + 0x10)
        {
                for (x = 2; x <= 5; x++)
                {
                        arb = CAN + t + x;
                        outp(arb,0xc8);
                }
        }                                               /* Loads $C8 into the
                                                        arbitration registers*/

        outp(cr,00);                                    /* Takes the 82527 out
                                                        of the initialization
                                                        mode.*/
}
```