intel®

AP-724

APPLICATION NOTE

Interfacing an MCS® 51 Microcontroller to an 82527 CAN Controller

GREG SCOTT
TECHNICAL MARKETING ENGINEER

October 1995

Order Number: 272764-001

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 MCS® 51 MICROCONTROLLER TO AN 82527 CAN CONTROLLER

CONTENTS

1.0 INTRODUCTION

The purpose of this application note is to describe two methods of interfacing an Intel MCS 51 microcontroller with an Intel 82527 CAN controller. This description includes a demonstration that all AC timing specifications are satisfied and a brief description of the 82527 interface modes and clocking structure. Also present are two hardware schematics.

1.1 Interface Suggestions

- The 82527 should be used in mode 0.
- The 82527 matches MCS 51 controller pin for pin on: ALE, WR#, RD#, and AD7–AD0.
- The 82527 INT# pin is tied to the MCS 51 controller INT0# pin.
- The 82527 RESET# pin is tied to a MCS 51 controller port pin.
- The CS# signal may be generated by decoding the upper address bits from the MCS 51 controller.

1.2 Top Five Issues

- The double read operation is required for reading low speed registers.
- The CS# signal must be valid in 12.5 ns of a valid address
- The CS# signal may be generated with a PAL when all 8-bit of the upper address are needed, or with an inverter when one bit of the upper address is free.
- Reset time for the 82527 is 1 ms.
- Default condition of 82527 for mode 0 sets MCLK = SCLK/2, this must be set to MCLK = SCLK following a reset.

2.0 82527 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 multiplexed (mode 0)
- 16-bit multiplexed (mode 1)
- 8-bit 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 MCS 51 controller to the 82527, 8-bit multiplexed mode (Mode 0) of the 82527 is used.

3.0 82527 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 CPU (host)/ 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.

The 82527 contains two types of registers in the RAM: high-speed registers (locations 02H, 04H, and 05H) and normal or low-speed registers (all registers except 02H, 04H, and 05H). Read and write operations to the low speed registers occur over a synchronous internal bus which is clocked by the MCLK. High-speed registers 02H, 04H and 05H are decoupled from the internal bus, allowing them to be accessed more quickly by the host CPU. High-speed read registers 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. Interfacing the MCS 51 microcontroller operating at 16 MHz to the 82527 requires the use of the double-read operation. Please refer to the 82527 Architectural Overview (Order Number 272410) for additional information regarding the double-read operation.

4.0 INTERFACING SCHEMATICS FOR MODE 0

Figure 4-1 demonstrates an MCS 51 controller with an 8-bit multiplexed external address/data bus interfacing to the 82527. Scheme 1 uses minimal hardware and two separate crystals. Scheme 2 decodes the upper address when all upper address bits are needed and uses one crystal.

In interface scheme 1 the CS# signal is derived by inverting the address line, A15. Both devices are clocked by Quartz crystals, consult the crystal manufacturer specifications for proper load capacitance. The 82527 can use either a 16 MHz or an 8 MHz crystal. 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 (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 an available port pin on the MCS 51 controller, this signal may also be derived by an RC network. The reset pin on the 82527 must be below V_{IL} for a minimum of 1 ms after V_{CC} is in the operation range to guarantee a proper device reset.

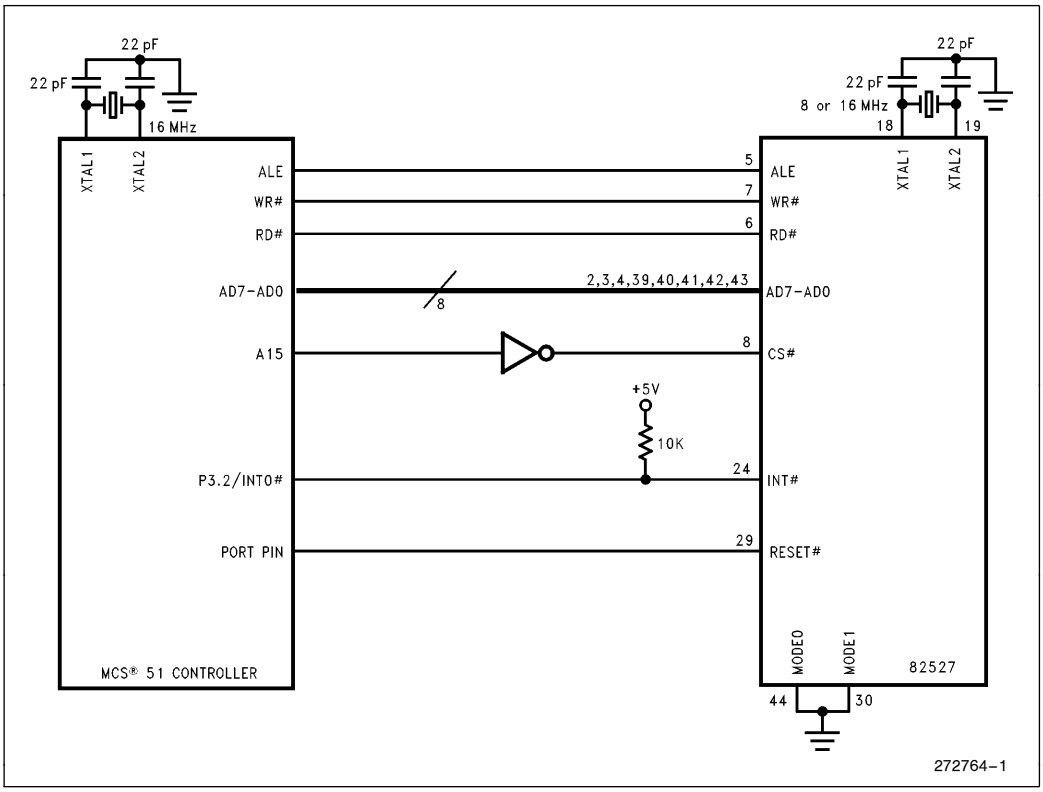

Figure 4-1. Interface Scheme 1

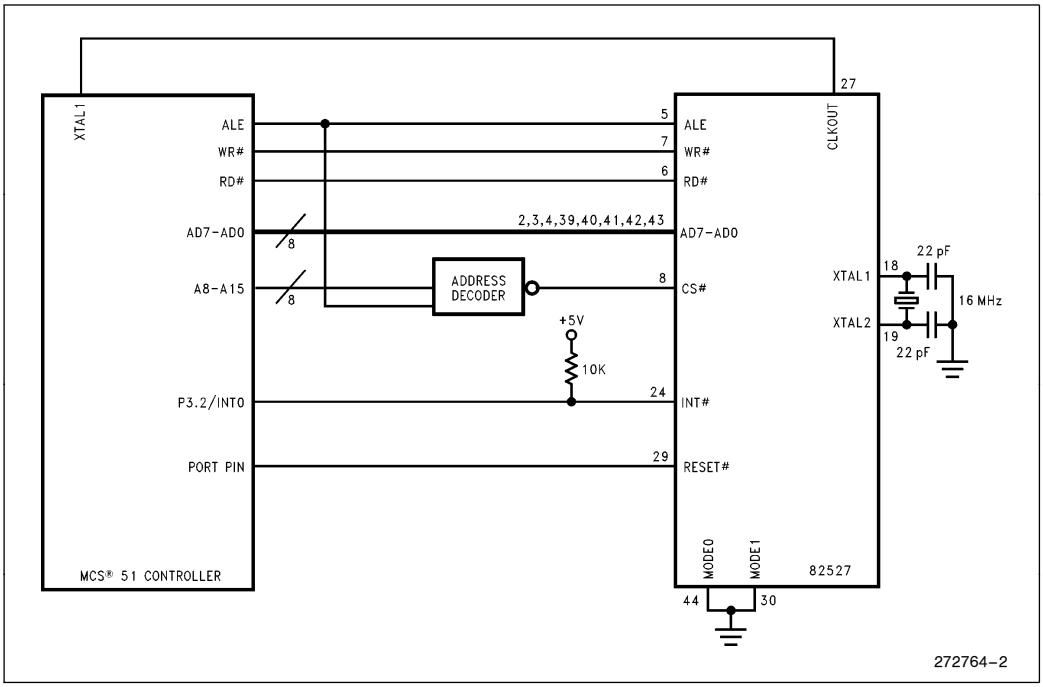

Figure 4-2. Interface Scheme 2

In interface scheme 2 the 82527 is clocked by a Quartz crystal and the MCS 51 controller uses the CLKOUT pin on the 82527 as a clock source. The default frequency for the CLKOUT pin in mode 0 is XTAL (i.e., 16 MHz).

The CS# signal is generated by a PAL decoding the upper address lines. The CS# signal must be generated 12.5 ns after a valid address is driven on the bus. The 82527 requires the CS# signal to be valid 10 ns before ALE goes low (t_{CLLL} = 10 ns min). The MCS 51 controller sends a valid address 22.5 ns prior to ALE falling (t_{AVLL} = 22.5 ns). The equation to calculate the time to generate the 82527 chip select is:

$$\text{CS\# generation} < t_{AVLL} - t_{CLLL}$$
$$\text{CS\# Generation} < 12.5 \text{ ns}$$

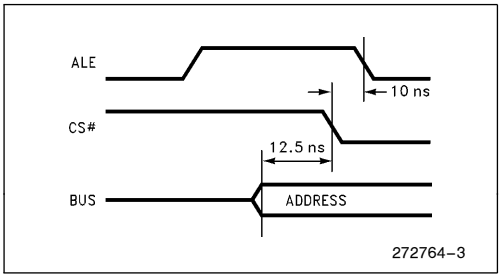

Figure 4-3. 82527 CS# Setup Timing Requirements

5.0 TIMING CONSIDERATIONS

Table 5-1. Critical Timing

Symbol	Parameter	8XC51FX(4)	@16 MHz	82527 @ 8 MHz(2)
Timings the MCS®51 Controller Must Satisfy (Automotive Specifications(3))				
t_{AVLL}	Address Valid to Falling ALE	Tosc − 40 min	22.5	7.5 min
t_{LLAX}	Address Hold After ALE Falling	Tosc − 30 min	32.5	10 min
t_{LHLL}	ALE High Period	2Tosc − 40 min	85	30 min
t_{LLRL}	ALE Falling to RD# Falling	3Tosc − 50 min	137.5	20 min
t_{QVWH}	Data Valid to WR# Rising	7Tosc − 150 min	287.5	27 min
t_{WHQX}	Data Hold After WR# Rising	Tosc − 50 min	12.5	12.5 min
t_{WLWH}	WR# Low Period	6Tosc − 100 min	275	30 min
t_{WHLH}	WR# Rising to ALE Rising	Tosc − 40 min	22.5	8 min
t_{RLRH}	RD# Low Period	6Tosc − 100 min	275	40 min
Timings the 82527 Must Satisfy				
t_{RLDV}(1)	RD# Falling to Data Valid	5Tosc − 165 max	147.5	55 max
t_{RHDZ}	RD# Rising to 82527 Data Float	2Tosc − 60 max	65	45 max
t_{AVDV}	Address Valid to Data Valid	9Tosc − 165 max	397.5	317.5 max
Timings the CS# Generator Must Satisfy				
t_{CLLL}	CS# Falling to ALE Falling		0	10 min
t_{WHCH}	WR# Rising to CS# Rising		0	0 min

NOTES:
1. This is the time for reading the high-speed registers (02h, 04h, and 05h).
2. The 82527 timings are based on an 8 MHz MCLK
3. The timings comparison uses the Automotive datasheets Order Number 231792 (Automotive Products databook) for the MCS 51 controller and Order Number 272250 for the 82527 device.
4. Tosc = 62.5 ns, all timings are in ns.

5.1 Read Timings

Since double reads are required, all register accesses have $t_{RLDV} < 55$ ns which meets the MCS 51 controller requirements ($t_{RLDV} = 147.5$ ns). A double-read operation is implemented by first addressing the desired low speed register then addressing the high-speed read register (location 04h). Note that a double-read operation is only required for reading a low-speed register. For more information on the double-read operation refer to the 82527 architectural overview (Order Number 272410). The condition of a "read cycle with a previous write cycle" as specified in the 82527 datasheet never occurs using the MCS 51 microcontroller, therefore should be ignored.

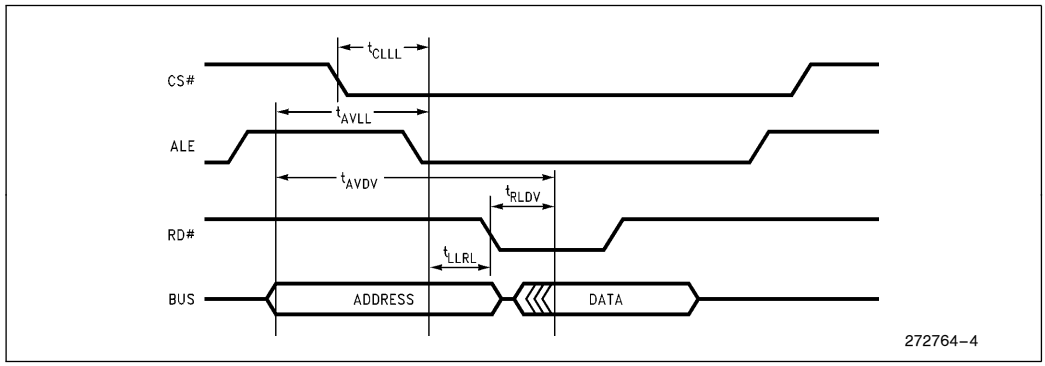

Figure 5-1. Bus Timing Diagram for a Read Cycle

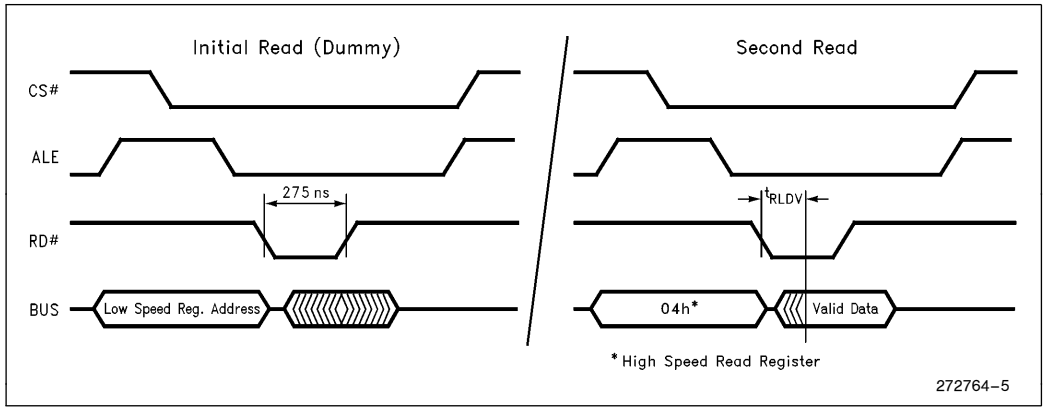

Figure 5-2. Bus Timing Diagram for a Double Read Operation

5.2 Write Timings

The timing comparisons for a write operation are valid. No special operations are needed. The condition of write cycle with a pending write as specified in the 82527 datasheet is not valid.

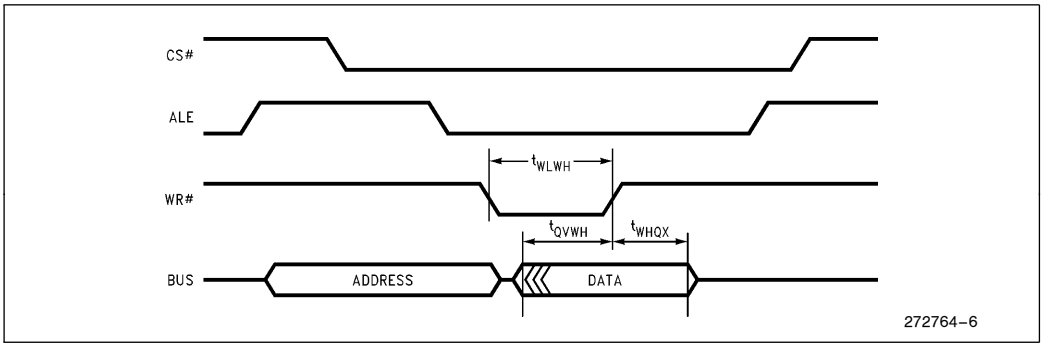

Figure 5-3. Bus Timing Diagram for a Write Cycle

6.0 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).

- 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 and tested using the BSO compiler for the MCS® 96 controller.

```
#define CAN 0x8000

main()
{
   int t, x;
   unsigned char *cr, *cir, *bcr, *bt0;
   unsigned char *btl, *contr0, *contrl;
   unsigned char *gm, *mcr, *arb;

   cir = (unsigned char*)CAN + 2;
   *cir = 0l;

   cr = (unsigned char*)CAN;
   *cr = 0x4l || *cr;

   bcr = (unsigned char*)CAN + 0x2f;
   *bcr = 0x48;

   bt0 = (unsigned char*)CAN + 0x3f;
   btl = (unsigned char*)CAN + 0x4f;
   *bt0 = 0x4l;
   *btl = 0x67;

   *cr = 0l;
```

```
/* Defines the starting address
   of the 82527 chip. */

/* Initializes the counter and
   pointer variables. */

/* Set the CPU Interface Register
   to 40: SCLK = XTAL/2,
   MCLK = SCLK, and disables
   the CLKOUT signal. */

/* Sets the CCE (Change
   Configuration Enable) bit in the
   Control Register. */

/* 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)*/

/* Defines the CAN bus frequency
   as 250 kBits/s and the sampling
   mode. */

/* Clears the CCE bit, preventing
   write access of configuration
   registers */
```

```
   for (t = 0x10; t <= 0xF0; t = t + 0x10)
   {
      contr0 = t + (unsigned char*)CAN;
      contr1 = contr0 +1;
      *contr0 = 0x55;
      *contr1 = 0x55;
   }                                          /* This loop resets Control
                                                 Register 0 and 1. */

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


   mcr = (unsigned char*)CAN + 0x16;
   *mcr = 0x8c;
   mcr = (unsigned char*)CAN + 0x26;
   *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 = (unsigned char*)CAN + t + x;
         *arb = 0xc8;
      }
   }                                          /* Loads $C8 into the arbitration
                                                 registers */


   *cr = 00;                                  /* Takes the 82527 out of the
                                                 initialization mode. */

   for(t = 1;t <=4; t++)
      t = 2;
}
```