intel®

AP-723

APPLICATION NOTE

Interfacing an Intel 82527 Serial Communications Controller to a 68322

GREG SCOTT
TECHNICAL MARKETING ENGINEER

October 1995

Order Number: 272763-001

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 INTEL 82527 SERIAL COMMUNICATIONS CONTROLLER TO A 68322

CONTENTS

PAGE

1.0 INTRODUCTION

The purpose of this application note is to describe one method of interfacing a Motorola 68332 microcontroller at 16.78 MHz with an Intel 82527 CAN controller. 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.
- The 82527 matches 68332 microcontroller pin for pin on: CS#, R/W#, DSACK0#, D7–D0, and A7–A0.
- The 82527 INT# pin is tied to the 68332 microcontroller IRQ# pin.
- The 82527 RESET# pin is tied to a 68332 port pin or reset circuit.
- The 68332 is operating at 16.78 MHz.

1.2 Top Four 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 times, this should be changed to MCLK = SCLK.
- Single read operations are supported, the double read operation is not needed.

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 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 68332 microcontroller to the 82527, 8-bit non-multiplexed asynchronous mode (Mode 3) 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 68332/ 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 data sheet (Order Number 272250-005). 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 synchronous 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 68332 microcontroller. 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 68332 microcontroller will insert wait states until the DSACK0# 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 ASYNCHRONOUS

Figure 4-1 demonstrates a minimal hardware interface between the 68332 microcontroller and the 82527.

The CS# signal is generated by the 68332 microcontroller. Both devices are clocked by Quartz crystals. Consult the crystal manufacturer specifications for proper load capacitance. For the lowest access time of the 82527 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 (location 02H). For more information on this register refer to the architectural overview (Order Number 272410-002). The RESET# signal for the 82527 may be generated using a port pin on the 68332 microcontroller, 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.

The CS# signal may be generated by a PAL decoding the upper address lines. If this method is used, the Address and R/W# signal must be valid for a minimum of 3 ns before CS# is generated.

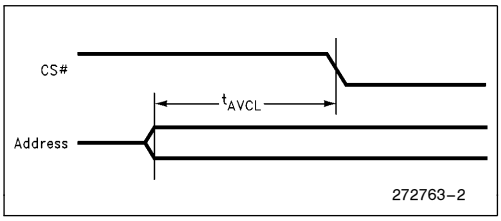

Figure 4-3. 82527 CS# Setup Timing Requirements

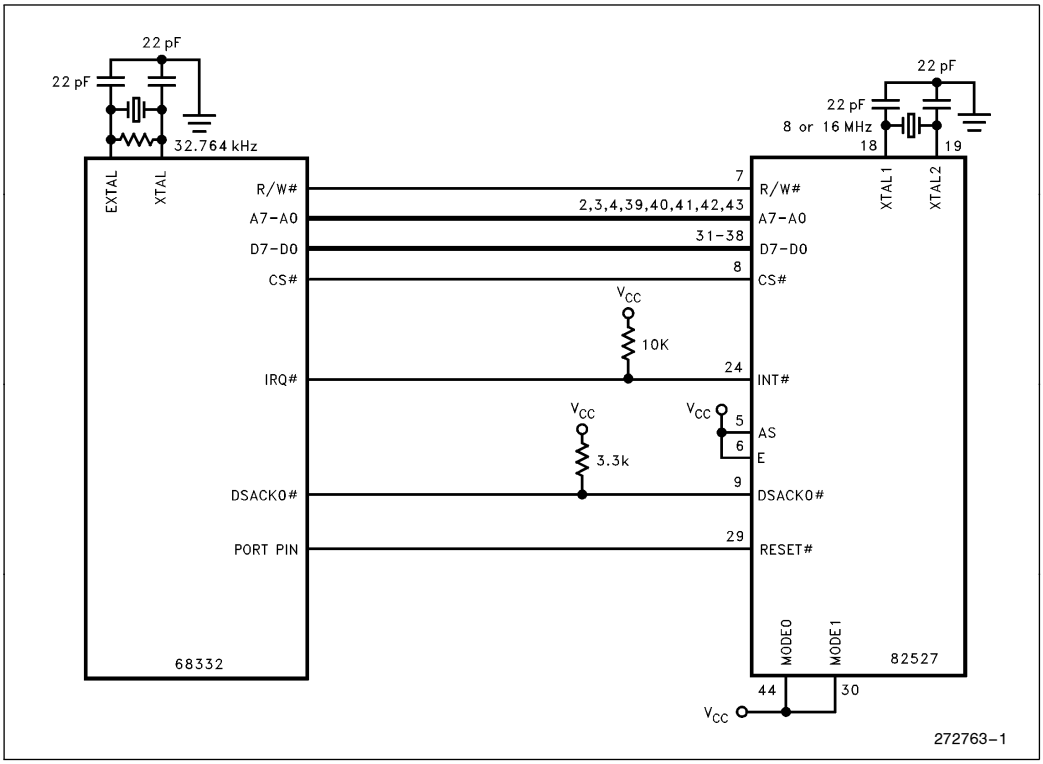

Figure 4-1. Interface Scheme

5.0 TIMING CONSIDERATIONS

Symbol	Parameters	82527 @ 8 MHz(1)	68332 @ 16.78 MHz(1)
t_{AVCL}	Address or R/W# Valid to CS# Low Setup	3 min	15 min
t_{CLDV}	CS# Low to Data Valid	(HSR) 55 max (LSR) 287.5 max	
t_{KLDV}	DSACK0# Low to Output Data Valid	(HSR) 23 max (LSR) <0 max	50 max
t_{CHDV}	82527 Input Data Hold after CS# High	15 min	15 min
t_{CHDH}	82527 Output Data Hold after CS# High	0 min	0 min
t_{CHDZ}	CS# High to Data Float	35 max	55 max
t_{CHKH}(2)	CS# High to DSACK0# $=$ 2.4V	55 max	80 max
t_{CHKZ}(2)	CS# High to DSACK0# Float	100 max	
t_{CHCL}	CS# Width between Successive Cycles	25 min	40 min
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}	CPU Write Data Valid to CS# High	32 min	74 min
t_{CLKL}	CS# Low to DSACK0# Low for Write Access without Previous Write	67 max	
t_{CLKL}	End of Previous Write (CS# High) to DSACK0# Low for a Write Cycle	395 max	

NOTES:
1. The Timing comparisons are based on the 82527 datasheet (Order Number 272250) and the MC68332 User's Manual #MC68332UM/AD rev 1.
2. The 82527 internally pulls the DSACK0# pin to 2.4V. An external pull-up may be used to drive the signal to 5V.
3. The condition of a ''read cycle with a previous write'' or a ''write cycle with a pending write'' is possible with the 68332 microcontroller.
4. A ''read cycle with a previous write cycle'' and a ''write cycle with a previous write cycle'' will cause the 68332 microcontroller to insert wait states, increasing the access time.

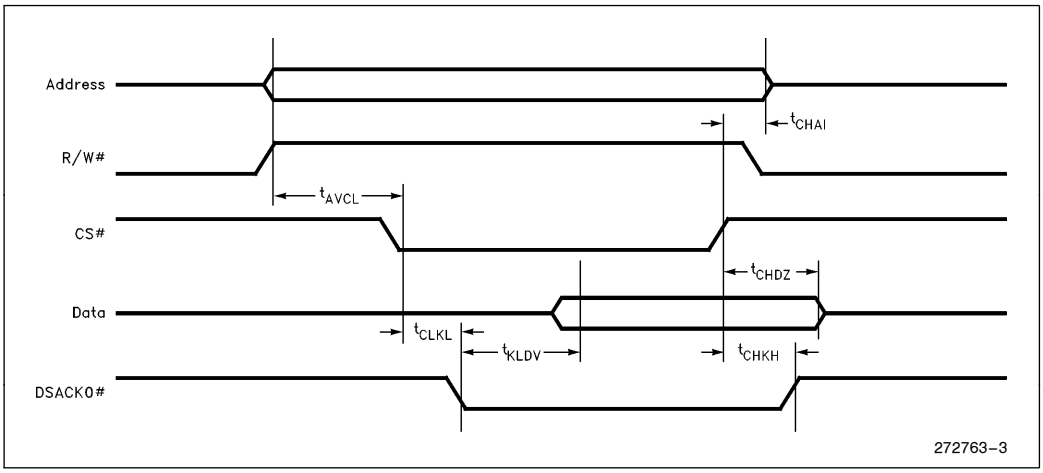

Figure 5-1. Bus Timing Diagram for a Read Cycle

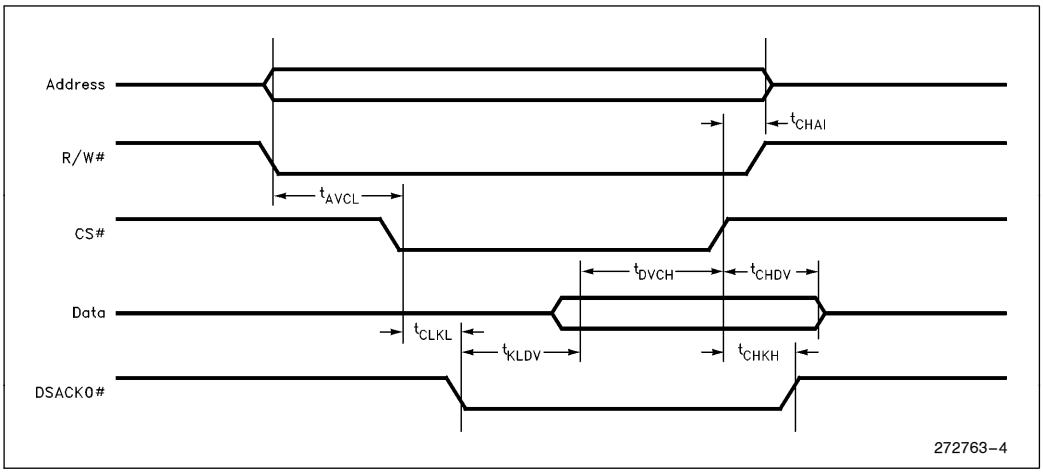

Figure 5-2. Bus Timing Diagram for a Write Cycle

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;
}
```