U. S. DEPARTMENT OF THE INTERIOR

U. S. GEOLOGICAL SURVEY

DESCRIPTIONS OF SEISMIC ARRAY COMPONENTS: PART 1. DIGGER, DIGIREC, AND MULTIPLEXER.

Compiled by

W. H. K. Lee
MS 977, 345 Middlefield Road
Menlo Park, CA 94025

Open-File Report 92-430-A
July, 1992

INTRODUCTION

In the summer of 1990, funding was available to design and implement two portable seismic arrays for the volcano program. The approach was based on Lee et al. (1989). Several contracts were awarded to commercial companies to design and implement various components needed to build the portable arrays. The purpose of this report to present the first three components – – DIGGER, DIGIREC, and MULTIPLEXER – – in details as submitted by the contractors. Source code on PC/DOS/MS DOS diskette for this report is presented in U. S. Geological Survey Open-File Report 92-430-B.

DIGGER

DIGGER is the prototype of a remote field digitizer for the USGS Digital Seismic Telemetry System. The original hardware design and specifications were made by Gray Jensen of the USGS. Cutler Digital Design of Mountain View was awarded a contract to implement this prototype. The DIGGER report as submitted by Cutler Digital Design is reproduced in its entirety in Appendix 1 (p. 4-88).

DIGIREC

DIGIREC is an electronic circuit board by Cutler Digital Design of Mountain View, California to receive serial data from up to 16 different field digitizer. The original specifications were due to Gray Jensen and Willie Lee. The DIGIREC report as submitted by Cutler Digital Design is reproduced in its entirety in Appendix 2 (p. 89-259).

MULTIPLEXER

MULTIPLEXER is an electronic circuit board by Dean Tottingham of Palo Alto, California to multiplex 64 channels of analog seismic signals for input to a 16-bit A/D board made by Data Translation, Inc. of Marlboro, Massachusetts. The design used an approach originally developed by Ellis (1989). The MULTIPLEXER report as submitted by Dean Tottingham is reproduced in its entirety in Appendix 3 (p. 260-276).

REFERENCES

Ellis, J. O. (1989). Expanding the input multiplexer for the Data Translation, Inc. Model DT2821 analog-to-digital converter, *U. S. Geol. Surv. Open-File Report 89-201.*

Lee, W. H. K., D. M. Tottingham, and J. O. Ellis (1989). Design and implementation of a PC-based seismic data acquisition, processing, and analysis system, *IASPEI Software Library, 1*, 21-46.

APPENDIX 1.

DIGGER

A Remote Field Digitizer

Cutler Digital Design
2215 Sun Mor Avenue
Mountain View, California 94040-3834
(415) 967-7617

DIGGER - Field Digitizer

Introduction

The DIGGER field digitizer was designed to gather data in a remote site, transform the data to a digital format, and transmit the data in a serial format. This circuit was envisioned as one of the necessary parts in the transformation of an existing analog seismic data acquisition system, to a digital seismic data acquisition system.

The requirements for such a device are : 1) modest power consumption, 2) high digitizer resolution, 3) high quality signal conditioning, 4) multichannel capability, 5) retained programmability, 6) resynchronization capability, and 7) serial output with modulation.

Description

DIGGER software operation has two modes. In normal mode, it digitizes and transmits data. In command mode, it allows the user to reprogram its parameters.

Normally, DIGGER samples the appropriate channels (as specified by the parameters) once each sample period. It then transmits the digitized data over an RS-232 serial output line. Occasionally, it will halt data collection and both resynchronize the output modulator and recalibrate the A/D converter. Once each sample period, it checks to see if it has received a serial input string that matches the password..if so, it transfers out of normal mode into command mode.

In command mode, DIGGER accepts commands over the serial input line, changing or reporting on parameter values. DIGGER will automatically revert out of command mode and into normal mode if a certain amount of time without serial activity has passed.

The hardware of DIGGER is composed of 4 separate circuit boards: 1) a processor board, a digitizer board, an analog input board, and a modulator board.

The analog input board accommodates 3 channels of analog data. The amplifiers are set to look at the range of +/- 4.5 millivolts. The analog signals are filtered with a 7 pole Bessel low pass filter.

The digitizer board can handle up to eight inputs. Currently it is constrained to 6 analog signals (from off the board), and a signal that is proportional to the power supply voltage (that is on the board). The multiplexer and digitizer are both controlled by the processor board and can digitize on the order of one sample per millisecond. The resolution of the digitizer is 16 bits. This board also generates from a 12 volt battery the appropriate voltages at adequate current for supplying two of the analog boards

The controller board is a complete computer system with a UART, RAM, EPROM, EEROM, a clock, and input/output control lines. The program that makes the DIGGER work is located in the EPROM on this board. The crystal oscillator that drives the UART and the realtime clock is accurate enough for proper operation, and loose enough so that the timing of separate DIGGERs will drift with respect to one another. There are three base clock rates (selected by a hardware jumper) for the realtime clock, 1200 HZ, 600 HZ and 300 HZ. Higher clock rates increase the power consumption, but give better resolution for unusual sample frequencies. There are six available baud rates (selected by a hardware jumper): 300, 600, 1200, 2400, 4800, and 9600. The actual sample rate is limited by the baud rate and the realtime clock rate, but is set by the software using parameter setting. Choice of which channels are digitized is also made by the software using parameter setting.

The modulator board uses a fax modulation integrated circuit to encode 9600 baud asynchronous transmissions. To insure proper operation, it must be instructed periodically to send a training sequence, in the place of data.

The system is designed to run of a single 12 volt battery.

Development Environment

The Manx C compiler and assembler were used to develop the firmware. Assembly language source code is in the files: LMACROS.H, and DIGGER.ASM.

C-language source code is in the files: \MANXC\HEADER\STDIO.H, DIGGER.C, and STKSIZ.C. The C language code is linked with the DIGGERA.O (the assembly language source), and the two libraries, m.lib and c.lib. (Note: in linking it is important to link the m.lib before the c.lib.)

The code was compiled and linked using the small code and small data model. The ".EXE" file is converted to a ".HEX" file by the Manx program HEX86. This file is suitable for loading EPROM programmers.

To facilitate the development process a program called MAKE developed by Borland International was used. The "makefile" that specifies the different steps (with program options) in compiling the software is included in the appendices.

User Manual

When the DIGGER is first started it recalibrates the digitizer, resets the modulator and then sets its parameters. It first tries to set the parameters according to the non volatile RAM. If there is a problem reading the non volatile RAM (EEROM), it sets the parameters to default values. Then the DIGGER waits for the inactivity timer to expire before going into NORMAL mode. The inactivity timer is reset anytime that serial characters are received by the DIGGER.

In command mode there are various commands:

The MENU Command

The MENU command lists the various commands that are allowed in the COMMAND mode. Here is what is returned if the MENU command is invoked:

```
MENU
Commands:
    STATUS - T - display current status
       SET - S - set various parameter values
      SAVE - V - save current parameter values permanently
   COLLECT - C - exit command mode
     RESET - E - reset a/d and vco
   RESTORE - R - restore parameter values from non volatile memory
      MENU - M - show commands
   DEFAULT - D - set all parameters to default values
```

Commands can be invoked by typing the name of the command and striking the enter key. Notice in the above listing that there is a unique letter after each command name. The command can also be invoked by typing this letter and striking the enter key.

The STATUS Command

The STATUS command displays the values of each parameter:

```
STATUS
Status:

     GOOD non volatile ram
.     300 Hardware interrupt frequency in HZ
     9600 Hardware baud rate
   100.00 Sample Frequency in HZ
       60 Minutes of inactivity before TIMEOUT
       13 START CHARACTER numeric value
       10 STOP CHARACTER numeric value
     1439 Minutes until next resync
     1440 Minutes between resyncs

        3 Channels selected
      channel  Sample Order
         3       sample(1)
         4       sample(2)
         5       sample(3)

          Password is <USGS>
```

The first indicator explains whether there is a functional non volatile RAM in the DIGGER. SAVE and RESTORE commands will only work properly if the status of the non volatile RAM is good.

The value for the hardware interrupt frequency reflects the value of the associated software parameter. The actual setting of this parameter is done with a hardware jumper. For proper operation the software parameter must be set to match the hardware setting.

The value for the hardware baud rate reflects the value of its associated software parameter. The actual setting of this parameter is done with a hardware jumper. For proper operation the software parameter must be set to match the hardware setting.

The frequency parameter indicates how often samples are taken from each data channel. For example, in the above report we see that 100 samples are taken from each of three channels. The sample frequency is 100 HZ (even though the aggregate sampling frequency is 300HZ).

In general, the inactivity timer is specified in minutes. This is the amount of time that must elapse with no serial input for the software to revert from command mode to normal mode. There is one instance when the inactivity period can be specified at 20 seconds. This happens if parameters are set to default values. In this case, the STATUS command reports 0 seconds as the inactivity threshold.

When data are transmitted, they are preceded by a start character, and followed by a stop character. The integer value (between 0 and 255) is also shown in the status.

The automatic resynchronization/recalibration can be set to occur between ten minutes and one week. Both the period, and the amount of time until it happens next are reported in the status.

The number of channels, and their multiplexer assignments are shown. It is possible to digitize the same channel more than once per sample period. The number of channels is limited by the baud rate, the hardware clock rate, and the sample frequency.

The password that allows the transition from normal mode to command mode is also shown in the status display

The SET Command

The SET command allows the changing of the values for 9 parameters. The following appears, once the SET command is invoked:

```
SET
Set Routine:
     1.  Indicate Hardware Clock Rate
     2.  Indicate Hardware Baud Rate
     3.  Set Sample Rate
     4.  Set Inactivity Timeout
     5.  Set Start Character
     6.  Set Stop Character
     7.  Set New Password
     8.  Set Resync Interval
     9.  Set Channels

     0.  Exit the Set Routine

Type the number of your selection:
```

Specifying "1" allows the changing of the hardware clock parameter. It should be noted, this does not change the actual hardware interrupt, only the parameter that is used in software:

```
Type the number of your selection:  1

Note: To change the actual tick rate a jumper in hardware must be moved
      This parameter should be set to match the hardware setting

      300 Hardware interrupt frequency in HZ

Type the hardware clock rate (1200,600,300):
```

The specification may be the full clock rate, or a shortened version. For example, "12" is good enough to specify "1200". If the value typed does not match any of the possibilities, the old value of the parameter is kept.

Specifying "2" allows the changing of the hardware baud rate parameter. It should be noted, this does not change the actual hardware setting, only the parameter that is used in software:

```
Type the number of your selection:  2

Note: To change the actual baud rate a jumper in hardware must be moved
      This parameter should be set to match the hardware setting

      9600 Hardware baud rate

Type the hardware baud rate (9600,4800,2400,1200,600,300):
```

The specification may be the full baud rate, or a shortened version. For example, "9" or "96" is good enough to specify "9600". If the value typed does not match any of the possibilities, the old value of the parameter is kept.

Specifying "3" allows the setting of the sample frequency. The actual frequency set may differ from that entered, since only certain frequencies are possible. The actual frequency if displayed before returning to the main SET menu:

```
Type the number of your selection:  3

  100.00 Sample Frequency in HZ

Type the desired sample rate (HZ):
```

Specifying "4" allows the setting of the inactivity timer:

```
Type the number of your selection:  4

       1 Minutes of inactivity before TIMEOUT

Type number of minutes of inactivity before timeout (1-60) :
```

Specifying "5" allows the setting of the start character:

```
Type the number of your selection:  5

     13 START CHARACTER numeric value

Type numeric value of the start character (0-255):
```

Specifying "6" allows the setting of the stop character:

```
Type the number of your selection:  6

     10 STOP CHARACTER numeric value

Type numeric value of the start character (0-255):
```

Specifying "7" allows the setting of the password:

```
Type the number of your selection:  7

          Password is <USGS>

Type new password

:
```

Specifying "8" allows the setting of the number of minutes between resynchronization/recalibration:

```
Type the number of your selection:  8

    1440 Minutes until next resync
    1440 Minutes between resyncs

Type the length (in minutes) of the resync interval.
    (There are 1440 minutes in a day)

( 10 - 10080 ) :
```

Specifying "9" allows the setting of the number of channels to digitize and the multiplexer assignment of those channels:

```
Type the number of your selection:  9

       3 Channels selected
      channel  Sample Order
         3       sample(1)
         4       sample(2)
         5       sample(3)


How many channels? (1-3) :
```

The SAVE Command

The SAVE command saves the current parameters in the non volatile RAM. If it is successful, the following message returns:

```
SAVE
Saving parameters to NVRAM
Write to NVRAM complete
```

If it is unsuccessful, it will stall for about a minute and then return the following message:

```
SAVE
Saving parameters to NVRAM
FAILURE to write to NVRAM
```

The COLLECT Command

The COLLECT command causes the software to shift immediately from COMMAND mode to NORMAL mode.

The RESET Command

The RESET command forces the immediate resynchronization/recalibration for the DIGGER. This process takes about 3 seconds.

```
RESET
Resetting A/D and VCO
```

The RESTORE Command

The RESTORE command causes an attempt at setting the parameters to the values in the non volatile RAM. If the non volatile RAM is correctly configured, the following is displayed:

```
RESTORE
Status:

     GOOD non volatile ram
.     300 Hardware interrupt frequency in HZ
     9600 Hardware baud rate
   100.00 Sample Frequency in HZ
       60 Minutes of inactivity before TIMEOUT
       13 START CHARACTER numeric value
       10 STOP CHARACTER numeric value
     1439 Minutes until next resync
     1440 Minutes between resyncs

        3 Channels selected
      channel  Sample Order
         3       sample(1)
         4       sample(2)
         5       sample(3)

          Password is <USGS>
```

If there is a problem reading the non volatile RAM, the following is displayed:

```
RESTORE
Could not restore from NV ram
```

The DEFAULT Command

The DEFAULT command sets the parameters to their default values:

```
DEFAULT
Default values set
```

Appendix A - Schematics

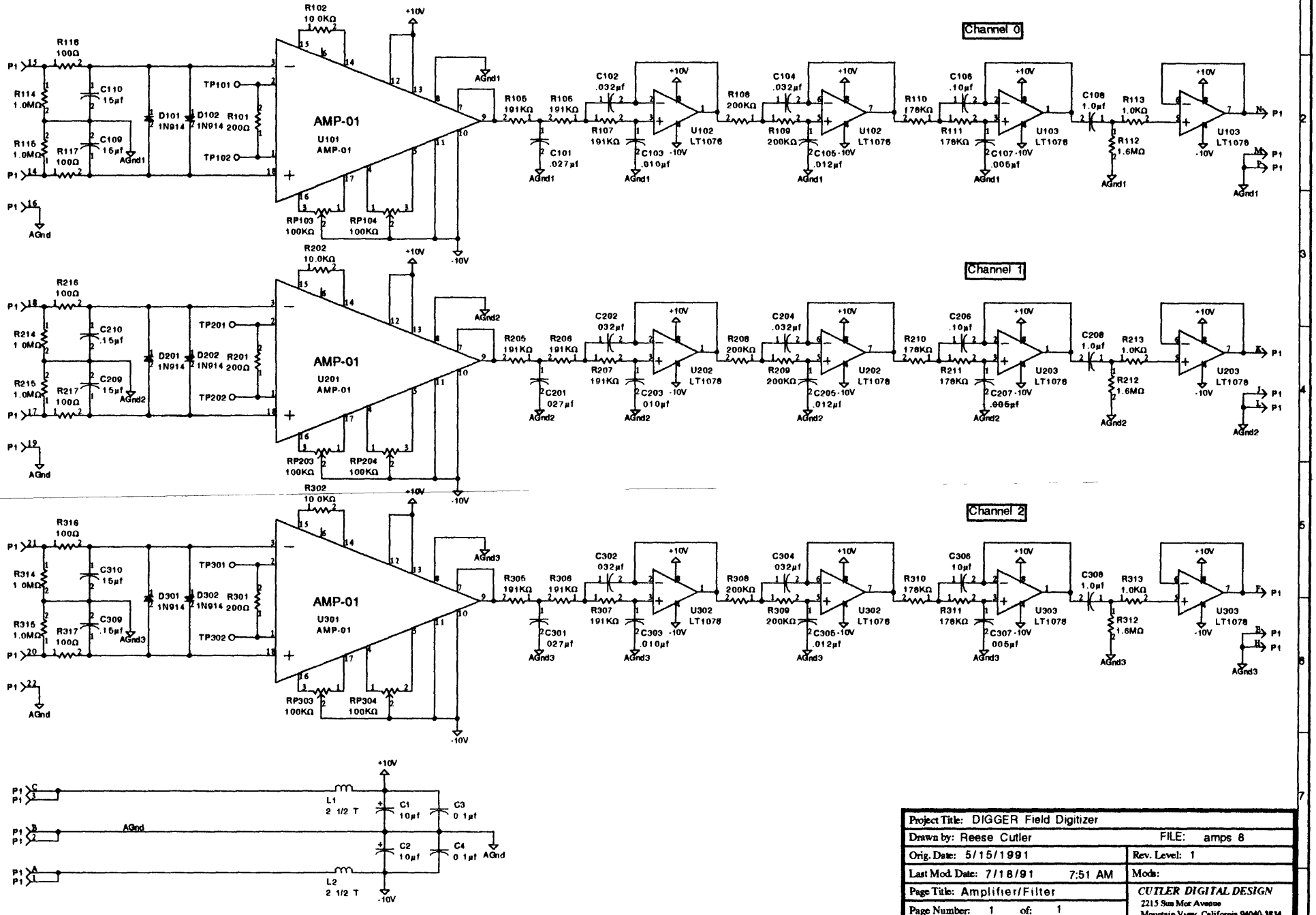
Analog Board Schematic

Multiplexer and Digitizer Board Schematic

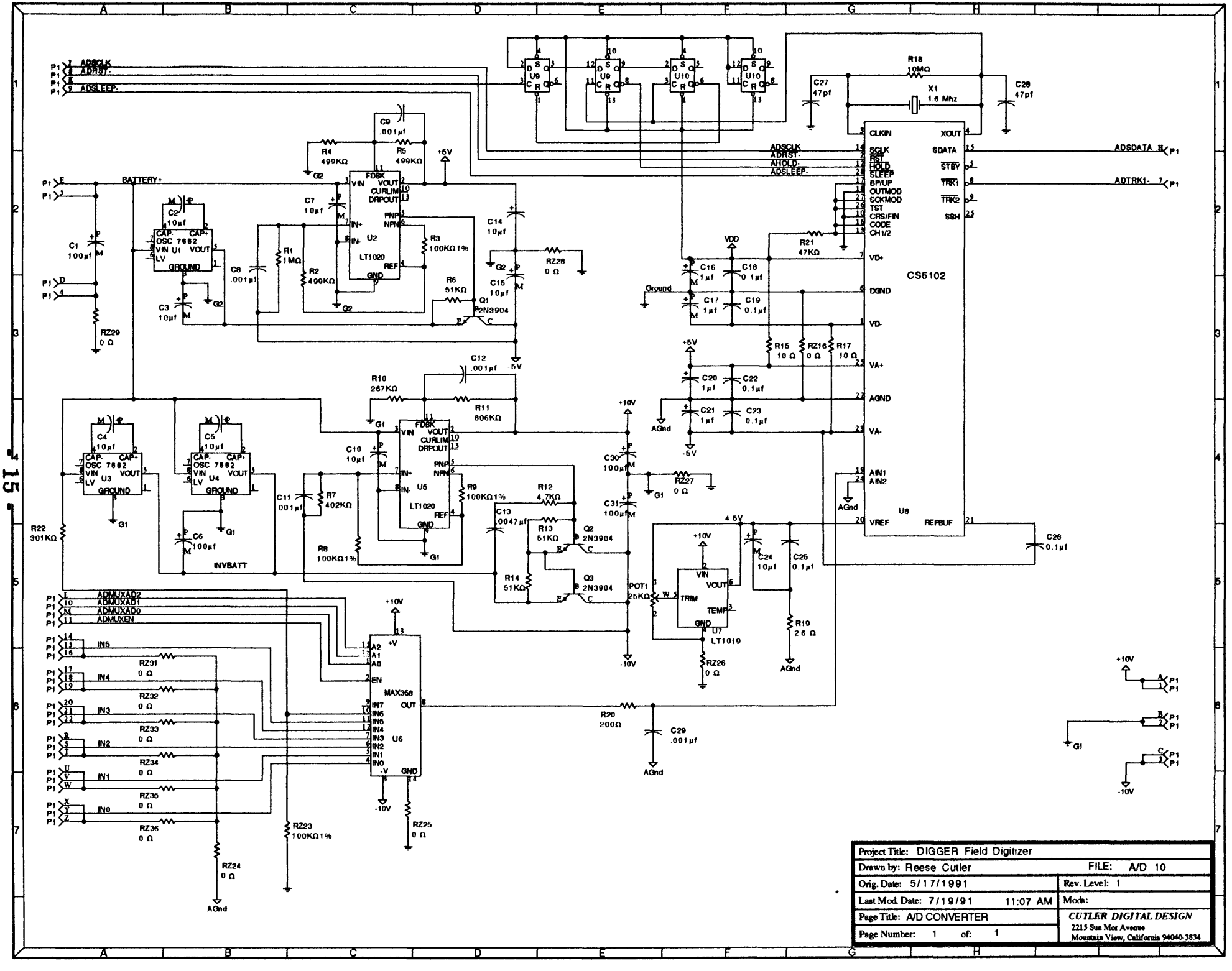
Processor Board Schematic

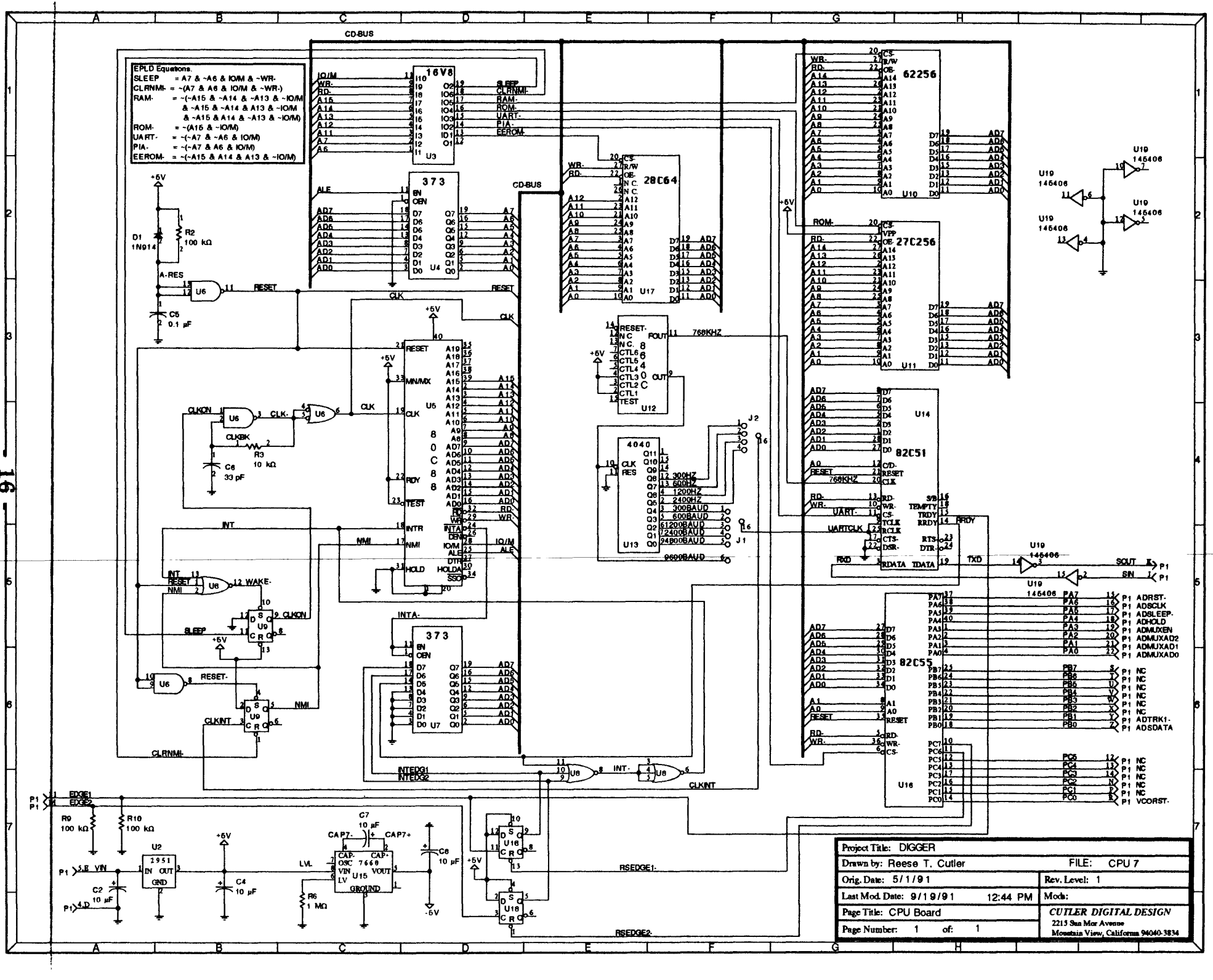
Modulator Board Schematic

Backplane Board Schematic

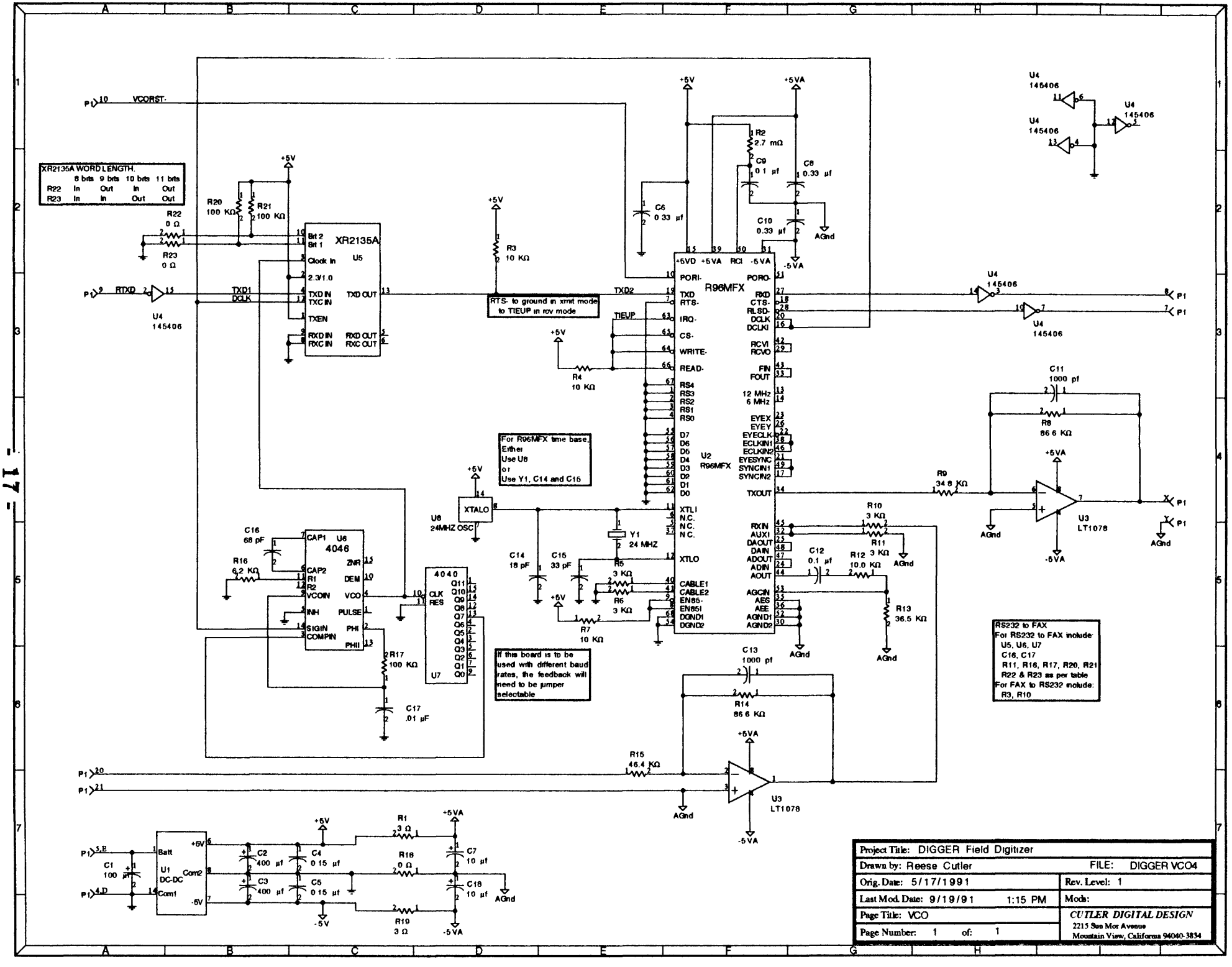


Project Title: DIGGER Field Digitizer		
Drawn by: Reese Cutler	FILE: amps 8	
Orig. Date: 5/15/1991	Rev. Level: 1	
Last Mod. Date: 7/18/91 7:51 AM	Mods:	
Page Title: Amplifier/Filter	CUTLER DIGITAL DESIGN	
Page Number: 1 of: 1	2215 Sun Mor Avenue, Mountain View, California 94040-3834	

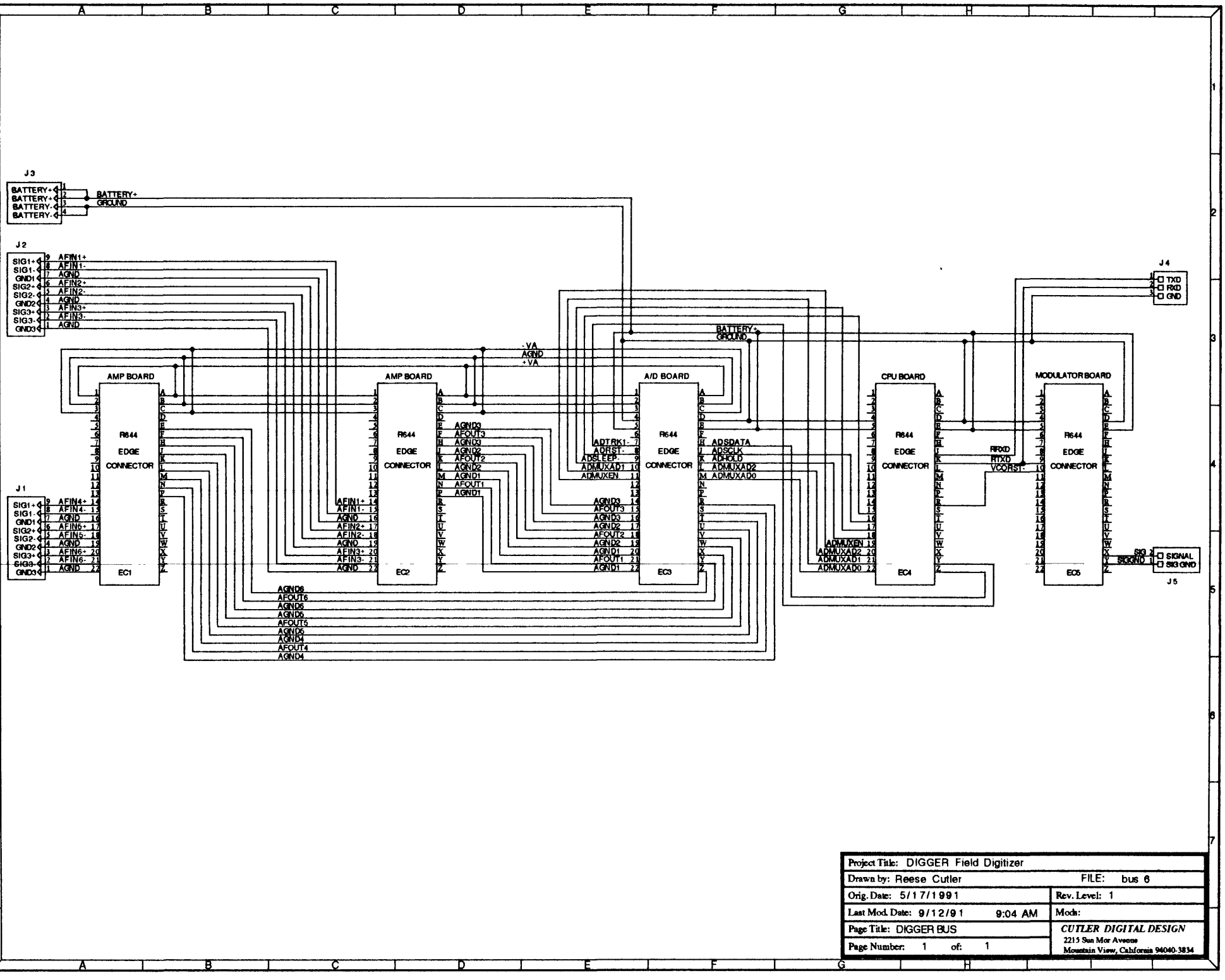

ADSCLK
ADRST-
AHOLD-
ADSLEEP-
BATTERY+
CS5102
ADSDATA R
ADTRK1-
INVBATT
ADMUXAD2
ADMUXAD1
ADMUXAD0
ADMUXEN
IN5
IN4
IN3
IN2
IN1
IN0
MAX358
LT1020
LT1019
2N3904
AGnd
Ground
Project Title: DIGGER Field Digitizer
Drawn by: Reese Cutler
FILE: A/D 10
Orig. Date: 5/17/1991
Rev. Level: 1
Last Mod. Date: 7/19/91 11:07 AM
Mods:
Page Title: A/D CONVERTER
CUTLER DIGITAL DESIGN
2215 Sun Mor Avenue
Mountain View, California 94040-3834
Page Number: 1 of: 1



Project Title: DIGGER	
Drawn by: Reese T. Cutler	FILE: CPU 7
Orig. Date: 5/1/91	Rev. Level: 1
Last Mod. Date: 9/19/91 12:44 PM	Mods:
Page Title: CPU Board	CUTLER DIGITAL DESIGN
Page Number: 1 of: 1	2215 Sun Mor Avenue, Mountain View, California 94040-3834



Project Title: DIGGER Field Digitizer			
Drawn by: Reese Cutler		FILE:	DIGGER VCO4
Orig. Date: 5/17/1991		Rev. Level: 1	
Last Mod. Date: 9/19/91	1:15 PM	Mods:	
Page Title: VCO		CUTLER DIGITAL DESIGN	
Page Number: 1	of: 1	2215 Sun Mor Avenue, Mountain View, California 94040-3834	



Project Title: DIGGER Field Digitizer		
Drawn by: Reese Cutler		FILE: bus 6
Orig. Date: 5/17/1991		Rev. Level: 1
Last Mod. Date: 9/12/91	9:04 AM	Mods:
Page Title: DIGGER BUS		*CUTLER DIGITAL DESIGN*
Page Number: 1	of: 1	2215 Sun Mor Avenue Mountain View, California 94040-3834

Appendix B - MAKEFILE

MAKEFILE

```
digger.hex: digger.exe
      hex86 digger.exe -s32

digger.exe: digger.o diggera.o stksiz.o
      ln -o digger.exe -f digger.f -c f800 -d 0000 -lm -lc -t -v

digger.o: digger.c
      cc digger.c +U

diggera.o:  digger.asm
      as digger.asm -o diggera.o -l

stksiz.o: stksiz.c
      cc stksiz.c
```

Appendix C - Assembler Source Code

DIGGER.ASM

LMACROS.H

DIGGER.ASM

```
; DIGGER.ASM                          (R. CUTLER,  06/20/91)
; Copyright (C) 1991 by Cutler Digital Design

   ;*************************************************************
   ;***                        INCLUDES
   ;*************************************************************

      include lmacros.h

;*******
; ASSEMBLER FLAGS
;

;NONMI       EQU   1

   ;***  CONSTANTS  ***************************************

ZERO      EQU     00H
BIT0      EQU     1H
BIT1      EQU     2H
BIT2      EQU     4H
BIT3      EQU     8H
BIT4      EQU     10H
BIT5      EQU     20H
BIT6      EQU     40H
BIT7      EQU     80H
XON       EQU     17
XOFF      EQU     19
STARTBYTE EQU      13
OFF    EQU   0
ON     EQU   1

MINUSONE  EQU    FFFFH

ROM       EQU   0F800H     ;***  Beginning of ROM (paragraph)
RAM       EQU   00H        ;***  Beginning of RAM (paragraph)
                           ;***  BOOT address = F800:7FF0H

; IO ADDRESS SPACE

UART  EQU   00H
PIA   EQU   40H
SLEEPAD      EQU   80H
CLRNMI       EQU   0C0H
;      ADDRESS
PARBASE      EQU   040H
SERBASE EQU 000H
SLEEPFLOP EQU      080H
PARAA EQU   PARBASE
PARB  EQU   PARBASE+1
PARC  EQU   PARBASE+2
PARCON       EQU   PARBASE+3
SERCON       EQU   SERBASE+1
SERDAT       EQU   SERBASE
```

```
;     OUTPUT   76543210 TO PARALLEL CONTROL BYTE
;              1        SET PARALLEL MODE
;              0        BIT CHANGE FOR PORT C
;               00       PORT A AND UPPER C MODE ZERO (BASIC I/O)
;               01       PORT A AND UPPER C MODE ONE (STROBED I/O)
;               10       PORT A AND UPPER C MODE TWO (BIDIR I/O)
;               11       PORT A AND UPPER C MODE TWO (BIDIR I/O)
;                 1       PORT A - INPUT
;                 0       PORT A - OUTPUT
;                  1       PORT C(UPPER) - INPUT
;                  0       PORT C(UPPER) - OUTPUT
;                   0       PORT B AND LOWER C MODE ZERO (BASIC I/O)
;                   1       PORT B AND LOWER C MODE ONE (STROBED I/O)
;                    1       PORT B - INPUT
;                    0       PORT B - OUTPUT
;                     1 PORT C(LOWER) - INPUT
;                     0 PORT C(LOWER) - OUTPUT
SETPARMODE      EQU     BIT7
BITWIDC         EQU     ZERO
ABASIC          EQU     ZERO
ASTROBE         EQU     BIT5
ABIDIR          EQU     BIT6
AINPUT          EQU     BIT4
AOUTPUT         EQU     ZERO
CUPIN           EQU     BIT3
CUPOUT          EQU     ZERO
BBASIC          EQU     ZERO
BSTROBE         EQU     BIT2
BINPUT          EQU     BIT1
BOUTPUT         EQU     ZERO
CDOWNIN         EQU     BIT0
CDOWNOUT        EQU     ZERO

; A/D AND MUX CONSTANTS
RESET       EQU   0
NORESET           EQU   128
SCLOCK            EQU   0
NOSCLOCK    EQU   64
ADSLEEP           EQU   0
NOADSLEEP   EQU   32
HOLD        EQU   16
NOHOLD            EQU   0
ENABLEMUX   EQU   8
DISABLEMUX  EQU   0
CHAN7       EQU   7
CHAN6       EQU   6
CHAN5       EQU   5
CHAN4       EQU   4
CHAN3       EQU   3
CHAN2       EQU   2
CHAN1       EQU   1
CHAN0       EQU   0
RESETBIT    EQU   80H
SCLOCKBIT   EQU   40H
ADSLEEPBIT  EQU   20H
HOLDBIT           EQU   10H
MUXBIT            EQU   8
NOTRESETBIT EQU   07FH
NOTSCLOCKBIT      EQU   0BFH
NOTADSLEEPBIT     EQU   0DFH
NOTHOLDBIT  EQU   0EFH
NOTMUXBIT   EQU   0F7H
;     MOV   AL,RESET + NOSCLOCK + NOADSLEEP + NOHOLD + ENABLEMUX + CHAN0
;     OUTPT PARAA,AL
```

```
   ;*****************************************************************
   ;***                          MACROS                           ***
   ;*****************************************************************
SETC7 MACRO
      OUTPT PARCON,0FH
      ENDM

SETC6 MACRO
      OUTPT PARCON,0DH
      ENDM

SETC5 MACRO
      OUTPT PARCON,0BH
      ENDM

SETC4 MACRO
      OUTPT PARCON,09H
      ENDM

SETC3 MACRO
      OUTPT PARCON,07H
      ENDM

SETC2 MACRO
      OUTPT PARCON,05H
      ENDM

SETC1 MACRO
      OUTPT PARCON,03H
      ENDM

SETC0 MACRO
      OUTPT PARCON,01H
      ENDM

RESETC7     MACRO
      OUTPT PARCON,0EH
      ENDM

RESETC6     MACRO
      OUTPT PARCON,0CH
      ENDM

RESETC5     MACRO
      OUTPT PARCON,0AH
      ENDM

RESETC4     MACRO
      OUTPT PARCON,08H
      ENDM

RESETC3     MACRO
      OUTPT PARCON,06H
      ENDM

RESETC2     MACRO
      OUTPT PARCON,04H
      ENDM

RESETC1     MACRO
      OUTPT PARCON,02H
      ENDM

RESETC0     MACRO
      OUTPT PARCON,00H
      ENDM
```

```
RESETON     MACRO
;       INPUT PARAA
        AND   AL,NOTRESETBIT
;       OUTPT PARAA
        ENDM


SCLOCKON MACRO
;       INPUT PARAA
        AND   AL,NOTSCLOCKBIT
;       OUTPT PARAA
        ENDM

ADSLEEPON MACRO
;       INPUT PARAA
        AND   AL,NOTADSLEEPBIT
;       OUTPT PARAA
        ENDM

HOLDON      MACRO
;       INPUT PARAA
        OR    AL,HOLDBIT
;       OUTPT PARAA
        ENDM

MUXON MACRO
;       INPUT PARAA
        OR    AL,MUXBIT
;       OUTPT PARAA
        ENDM

RESETOFF    MACRO
;       INPUT PARAA
        OR    AL,RESETBIT
;       OUTPT PARAA
        ENDM


SCLOCKOFF MACRO
;       INPUT PARAA
        OR    AL,SCLOCKBIT
;       OUTPT PARAA
        ENDM

ADSLEEPOFF MACRO
;       INPUT PARAA
        OR    AL,ADSLEEPBIT
;       OUTPT PARAA
        ENDM

HOLDOFF     MACRO
;       INPUT PARAA
        AND   AL,NOTHOLDBIT
;       OUTPT PARAA
        ENDM

MUXOFF      MACRO
;       INPUT PARAA
        AND   AL,NOTMUXBIT
;       OUTPT PARAA
        ENDM
```

```
   ;***  usage OUTPT  OUTPUT-PORT , VALUE

OUTPT MACRO  ARG1,ARG2  ;***  Allows output of reg., mem. or constants

IF    ARG1 GT 255       ;***  Used by lots of routines
      IFDIF <"DX">,<ARG1>
            MOV     DX,ARG1
      ENDIF
      IFDIF <"AL">,<ARG2>
            MOV     AL,ARG2
      ENDIF
      OUT     DX,AL
ELSE
      IFDIF <"AL">,<ARG2>
            MOV   AL,ARG2
      ENDIF
      OUT   ARG1,AL
ENDIF
        ENDM


   ;***  usage OUTPTL  OUTPUT-PORT(>255) , VALUE

OUTPTL      MACRO  ARG1,ARG2  ;***  Allows output of reg., mem. or constants
      MOV     DX,ARG1
      MOV     AL,ARG2
      OUT     DX,AL
        ENDM


   ;***  usage INPUT  INPUT-PORT

INPUT MACRO  ARG1       ;***  Allows input to reg., mem. or constants
      IF    ARG1 GT 255       ;***  Used by lots of routines
      IFDIF <"DX">,<ARG1>
        MOV     DX,ARG1
      ENDIF
        IN      AL,DX
      ELSE
      IN    AL,ARG1
      ENDIF
        ENDM

PUTSERCON MACRO   ARG1
      IFDIF <ARG1>,<"AL">
      MOV   AL,ARG1
      ENDIF
      OUT   SERCON,AL
      CALL  SERPAS
;     CALL  SERPAS
      ENDM

GETSERCON MACRO
      IN    AL,SERCON
      CALL  SERPAS
;     CALL  SERPAS
      ENDM

PUTSERDAT MACRO   ARG1
      IFDIF <ARG1>,<"AL">
      MOV   AL,ARG1
      ENDIF
      OUT   SERDAT,AL
      CALL  SERPAS
;     CALL  SERPAS
      ENDM
```

```
GETSERDAT MACRO
      IN    AL,SERDAT
      CALL  SERPAS
;     CALL  SERPAS
      ENDM

INTON MACRO          ;***  Enable interrupts
      STI               ;***  (Used by InitIO & GITCH3)
        ENDM


INTOFF      MACRO       ;***  Disable interrupts
      CLI               ;***  (InitIO, DOEDGE & GITCH1)
        ENDM

DISP  MACRO ARG1
      LOCAL DISP1
      LOCAL DISP2
      LOCAL DISP3
      LOCAL DISP4
      PUSH  AX    ;SAVE REGISTERS
      PUSH  BX
      JMP   DISP2 ;JUMP TO CODE BELOW
DISP1:
      DB    ARG1  ; TEXT STRING
      DB    0
DISP2:
      MOV   BX,OFFSET DISP1   ;SET START OF STRING
DISP3:
      MOV   AL,CS:[BX]  ;GET CHAR
      CMP   AL,00H
      JE    DISP4       ; IF 0, THEN DONE
      CALL  PUTCH       ;ELSE SEND IT
      INC   BX          ; UPDATE POINTER
      JMP   DISP3       ; AND LOOP
DISP4:
      POP   BX                ; RESTORE REGISTERS
      POP   AX
      ENDM

DISPCR      MACRO ARG1
      DISP  ARG1
      PUSH  AX
      CALL  CRLF
      POP   AX
      ENDM
```

```
;COMMENTS FROM MANX C SETUP:
;
;       If you want your stack at a fixed place, you may remove the
;       "bss cstack..." statement below and change the "mov sp,cstack..."
;       to load SP with the value you desire.  Note that the setup of
;       SS may need to be changed also.  If the program is small data
;       model, then SS must be equal to DS or pointers to automatic
;       variables won't work.
;
;       Otherwise, stacksize should be set according to your program's
;       requirements.
stacksize equ     2048
;
;       DATASEG is the data segment address (as a paragraph #)
;       this is picked from the -D option of the linker
;
;Dseg equ   040H  ;physical addr 0x400, just above the int vectors

; IT IS A LITTLE TRICKY ASSIGNING LOWER RAM, IT SEEMS THAT THE FIRST WORD
;   IN dataseg IS RESERVED FOR _DORG_.  INSERTING ORG STATMENTS SEEMS TO
;   ENFORCE A TWO BYTE OFFSET.


; INTERRUPT VECTORS:
DATASEG     segment     word public 'data'

      DW    ?
      PUBLIC      VEC01
VEC01 DW    ?
      DW    ?
VEC02 DW    INTNMI
      DW    CODESEG
VEC03 DW    ?
      DW    ?
VEC04 DW    ?
      DW    ?
VEC05 DW    ?
      DW    ?
VEC06 DW    ?
      DW    ?
VEC07 DW    ?
      DW    ?
VEC08 DW    ?
      DW    ?
VEC09 DW    ?
      DW    ?
VEC0A DW    ?
      DW    ?
VEC0B DW    ?
      DW    ?
VEC0C DW    ?
      DW    ?
VEC0D DW    ?
      DW    ?
VEC0E DW    ?
      DW    ?
VEC0F DW    ?
      DW    ?
VEC10 DW    ?
      DW    ?
VEC11 DW    ?
      DW    ?
VEC12 DW    ?
      DW    ?
VEC13 DW    ?
      DW    ?
```

```
VEC14 DW    ?
      DW    ?
VEC15 DW    ?
      DW    ?
VEC16 DW    ?
      DW    ?
VEC17 DW    ?
      DW    ?
VEC18 DW    ?
      DW    ?
VEC19 DW    ?
      DW    ?
VEC1A DW    ?
      DW    ?
VEC1B DW    ?
      DW    ?
VEC1C DW    ?
      DW    ?
VEC1D DW    ?
      DW    ?
VEC1E DW    ?
      DW    ?
VEC1F DW    ?
      DW    ?

      PUBLIC      VEC20
VEC20 DW    INT1
      DW    CODESEG
      PUBLIC      VEC21
VEC21 DW    ?
      DW    ?
      PUBLIC      VEC22
VEC22 DW    ?
      DW    ?
VEC23 DW    ?
      DW    ?
VEC24 DW    ?
      DW    ?
VEC25 DW    ?
      DW    ?
      PUBLIC      VEC26
VEC26 DW    ?
      DW    CODESEG
      PUBLIC      VEC27
VEC27 DW    ?
      DW    CODESEG
VEC28 DW    ?
      DW    ?
VEC29 DW    ?
      DW    ?
VEC2A DW    ?
      DW    ?
VEC2B DW    ?
      DW    ?
VEC2C DW    ?
      DW    ?
VEC2D DW    ?
      DW    ?
VEC2E DW    ?
      DW    ?
VEC2F DW    ?
      DW    ?
VEC30 DW    ?
      DW    ?
VEC31 DW    ?
      DW    ?
```

```
VEC32 DW    ?
      DW    ?
VEC33 DW    ?
      DW    ?
VEC34 DW    ?
      DW    ?
VEC35 DW    ?
      DW    ?
VEC36 DW    ?
      DW    ?
VEC37 DW    ?
      DW    ?
VEC38 DW    ?
      DW    ?
VEC39 DW    ?
      DW    ?
VEC3A DW    ?
      DW    ?
VEC3B DW    ?
      DW    ?
VEC3C DW    ?
      DW    ?
VEC3D DW    ?
      DW    ?
VEC3E DW    ?
      DW    ?
VEC3F DW    ?
      DW    ?
VEC40 DW    INT2
      DW    CODESEG
VEC41 DW    ?
      DW    ?
VEC42 DW    ?
      DW    ?
VEC43 DW    ?
      DW    ?
VEC44 DW    ?
      DW    ?
VEC45 DW    ?
      DW    ?
VEC46 DW    ?
      DW    ?
VEC47 DW    ?
      DW    ?
VEC48 DW    ?
      DW    ?
VEC49 DW    ?
      DW    ?
VEC4A DW    ?
      DW    ?
VEC4B DW    ?
      DW    ?
VEC4C DW    ?
      DW    ?
VEC4D DW    ?
      DW    ?
VEC4E DW    ?
      DW    ?
VEC4F DW    ?
      DW    ?
VEC50 DW    ?
      DW    ?
VEC51 DW    ?
      DW    ?
VEC52 DW    ?
      DW    ?
```

```
VEC53 DW    ?
      DW    ?
VEC54 DW    ?
      DW    ?
VEC55 DW    ?
      DW    ?
VEC56 DW    ?
      DW    ?
VEC57 DW    ?
      DW    ?
VEC58 DW    ?
      DW    ?
VEC59 DW    ?
      DW    ?
VEC5A DW    ?
      DW    ?
VEC5B DW    ?
      DW    ?
VEC5C DW    ?
      DW    ?
VEC5D DW    ?
      DW    ?
VEC5E DW    ?
      DW    ?
VEC5F DW    ?
      DW    ?
VEC60 DW    INT3
      DW    CODESEG
VEC61 DW    ?
      DW    ?
VEC62 DW    ?
      DW    ?
VEC63 DW    ?
      DW    ?
VEC64 DW    ?
      DW    ?
VEC65 DW    ?
      DW    ?
VEC66 DW    ?
      DW    ?
VEC67 DW    ?
      DW    ?
VEC68 DW    ?
      DW    ?
VEC69 DW    ?
      DW    ?
VEC6A DW    ?
      DW    ?
VEC6B DW    ?
      DW    ?
VEC6C DW    ?
      DW    ?
VEC6D DW    ?
      DW    ?
VEC6E DW    ?
      DW    ?
VEC6F DW    ?
      DW    ?
VEC70 DW    ?
      DW    ?
VEC71 DW    ?
      DW    ?
VEC72 DW    ?
      DW    ?
VEC73 DW    ?
      DW    ?
```

```
VEC74 DW    ?
      DW    ?
VEC75 DW    ?
      DW    ?
VEC76 DW    ?
      DW    ?
VEC77 DW    ?
      DW    ?
VEC78 DW    ?
      DW    ?
VEC79 DW    ?
      DW    ?
VEC7A DW    ?
      DW    ?
VEC7B DW    ?
      DW    ?
VEC7C DW    ?
      DW    ?
VEC7D DW    ?
      DW    ?
VEC7E DW    ?
      DW    ?
VEC7F DW    ?
      DW    ?
VEC80 DW    INT4
      DW    CODESEG
VEC81 DW    ?
      DW    ?
VEC82 DW    ?
      DW    ?
VEC83 DW    ?
      DW    ?
VEC84 DW    ?
      DW    ?
VEC85 DW    ?
      DW    ?
VEC86 DW    ?
      DW    ?
VEC87 DW    ?
      DW    ?
VEC88 DW    ?
      DW    ?
VEC89 DW    ?
      DW    ?
VEC8A DW    ?
      DW    ?
VEC8B DW    ?
      DW    ?
VEC8C DW    ?
      DW    ?
VEC8D DW    ?
      DW    ?
VEC8E DW    ?
      DW    ?
VEC8F DW    ?
      DW    ?
VEC90 DW    ?
      DW    ?
VEC91 DW    ?
      DW    ?
VEC92 DW    ?
      DW    ?
VEC93 DW    ?
      DW    ?
VEC94 DW    ?
      DW    ?
```

```
VEC95 DW    ?
      DW    ?
VEC96 DW    ?
      DW    ?
VEC97 DW    ?
      DW    ?
VEC98 DW    ?
      DW    ?
VEC99 DW    ?
      DW    ?
VEC9A DW    ?
      DW    ?
VEC9B DW    ?
      DW    ?
VEC9C DW    ?
      DW    ?
VEC9D DW    ?
      DW    ?
VEC9E DW    ?
      DW    ?
VEC9F DW    ?
      DW    ?
VECA0 DW    INT5
      DW    CODESEG
VECA1 DW    ?
      DW    ?
VECA2 DW    ?
      DW    ?
VECA3 DW    ?
      DW    ?
VECA4 DW    ?
      DW    ?
VECA5 DW    ?
      DW    ?
VECA6 DW    ?
      DW    ?
VECA7 DW    ?
      DW    ?
VECA8 DW    ?
      DW    ?
VECA9 DW    ?
      DW    ?
VECAA DW    ?
      DW    ?
VECAB DW    ?
      DW    ?
VECAC DW    ?
      DW    ?
VECAD DW    ?
      DW    ?
VECAE DW    ?
      DW    ?
VECAF DW    ?
      DW    ?
VECB0 DW    ?
      DW    ?
VECB1 DW    ?
      DW    ?
VECB2 DW    ?
      DW    ?
VECB3 DW    ?
      DW    ?
VECB4 DW    ?
      DW    ?
VECB5 DW    ?
      DW    ?
```

```
VECB6 DW    ?
      DW    ?
VECB7 DW    ?
      DW    ?
VECB8 DW    ?
      DW    ?
VECB9 DW    ?
      DW    ?
VECBA DW    ?
      DW    ?
VECBB DW    ?
      DW    ?
VECBC DW    ?
      DW    ?
VECBD DW    ?
      DW    ?
VECBE DW    ?
      DW    ?
VECBF DW    ?
      DW    ?
VECC0 DW    INT6
      DW    CODESEG
VECC1 DW    ?
      DW    ?
VECC2 DW    ?
      DW    ?
VECC3 DW    ?
      DW    ?
VECC4 DW    ?
      DW    ?
VECC5 DW    ?
      DW    ?
VECC6 DW    ?
      DW    ?
VECC7 DW    ?
      DW    ?
VECC8 DW    ?
      DW    ?
VECC9 DW    ?
      DW    ?
VECCA DW    ?
      DW    ?
VECCB DW    ?
      DW    ?
VECCC DW    ?
      DW    ?
VECCD DW    ?
      DW    ?
VECCE DW    ?
      DW    ?
VECCF DW    ?
      DW    ?
VECD0 DW    ?
      DW    ?
VECD1 DW    ?
      DW    ?
VECD2 DW    ?
      DW    ?
VECD3 DW    ?
      DW    ?
VECD4 DW    ?
      DW    ?
VECD5 DW    ?
      DW    ?
VECD6 DW    ?
      DW    ?
```

```
VECD7 DW    ?
      DW    ?
VECD8 DW    ?
      DW    ?
VECD9 DW    ?
      DW    ?
VECDA DW    ?
      DW    ?
VECDB DW    ?
      DW    ?
VECDC DW    ?
      DW    ?
VECDD DW    ?
      DW    ?
VECDE DW    ?
      DW    ?
VECDF DW    ?
      DW    ?
VECE0 DW    INT7
      DW    CODESEG
VECE1 DW    ?
      DW    ?
VECE2 DW    ?
      DW    ?
VECE3 DW    ?
      DW    ?
VECE4 DW    ?
      DW    ?
VECE5 DW    ?
      DW    ?
VECE6 DW    ?
      DW    ?
VECE7 DW    ?
      DW    ?
VECE8 DW    ?
      DW    ?
VECE9 DW    ?
      DW    ?
VECEA DW    ?
      DW    ?
VECEB DW    ?
      DW    ?
VECEC DW    ?
      DW    ?
VECED DW    ?
      DW    ?
VECEE DW    ?
      DW    ?
VECEF DW    ?
      DW    ?
VECF0 DW    ?
      DW    ?
VECF1 DW    ?
      DW    ?
VECF2 DW    ?
      DW    ?
VECF3 DW    ?
      DW    ?
VECF4 DW    ?
      DW    ?
VECF5 DW    ?
      DW    ?
VECF6 DW    ?
      DW    ?
VECF7 DW    ?
      DW    ?
```

```
VECF8 DW    ?
      DW    ?
VECF9 DW    ?
      DW    ?
VECFA DW    ?
      DW    ?
VECFB DW    ?
      DW    ?
VECFC DW    ?
      DW    ?
VECFD DW    ?
      DW    ?
VECFE DW    ?
      DW    ?
VECFF DW    ?
      DW    ?

;MANX C DECLARATIONS
      public      $MEMRY
$MEMRY      dw    -1
      dw    -1
      public      errno_
errno_      dw    0
      public      _dsval_,_csval_
_dsval_     dw    0
_csval_     dw    0
      public      _mbot_, sbot
_mbot_      dw    0
      dw    0
sbot  dw    0
      dw    0
      extrn _Dorg_:byte,_Dend_:byte
      extrn _Uorg_:byte,_Uend_:byte
      public      cstack
      bss   cstack:byte,stacksize
DATASEG     ends
      extrn _Corg_:byte,_Cend_:byte

ifdef FARPROC
      extrn main_:far, $fltinit:far, TickInt_:far
else
      extrn main_:near, $fltinit:near, TickInt_:near
endif
```

```
;MANX C STARTUP CODE
;_____________________________________________________________________
;_____________________________________________________________________
;
;/////////////////   STARTUP ROUTINE CODE SPACE   \\\\\\\\\\\\\\\\\\\
;_____________________________________________________________________
;_____________________________________________________________________

      public      $begin
$begin      proc  far
; ASSUMES THAT THE NMI IS DISABLED AT THIS POINT
; SOFTWARE COLD RESETS MUST FORCE NMI TO A DUMMY ROUTINE
      cli
      cld
;
;     Compute where initialzed data starts (@ next para after code)
;
      mov   ax,offset _Cend_+15
      mov   cl,4
      shr   ax,cl
      add   ax,seg _Cend_
      mov   ds,ax       ;place where data is in rom
      mov   bx,DATASEG  ;place where data is to go in ram
      mov   es,bx
;
;     Note: For hardware reasons the instruction which loads SS should
;           be immediatly followed by the load of SP.
;
      mov   ss,bx
      mov   sp,offset cstack+stacksize
;
;           copy Init data from rom to ram
      mov   di,0
      mov   cx,offset _Dend_
      inc   cx
      shr   cx,1
      jcxz  nocopy
      mov   si,0
rep   movsw
nocopy:
;
;           clear uninitialized data

;**** don't zero uninitialized stuff
      jmp   noclear
;****
      mov   di,offset _Uorg_
      mov   cx,offset _Uend_
      sub   cx,di
      inc   cx
      shr   cx,1
      jcxz  noclear
      sub   ax,ax
rep   stosw
noclear:
;
      assume      ds:DATASEG,es:DATASEG

      mov   ds,bx             ;set DS, now DS, SS, ES are equal
```

```
;*************************************************************
;
; 8088 INIT CODE
;
;*************************************************************
      ASSUME CS:CODESEG,DS:DATASEG,ES:DATASEG,SS:DATASEG

; ASSUME MASKABLE INTERRUPTS DISABLED, AND NMI DISABLED

; SET NMI TO DRIVE 300HZ CLOCK, AND RESET FLIPFLOP
; SET UP PARALLEL PORT PORT A AS OUTPUT PORT B AS INPUT PORT C AS OUTPUT
      OUTPT  PARCON,<SETPARMODE+ABASIC+AOUTPUT+CUPOUT+BBASIC+BINPUT+CDOWNOUT>

; INITIAL SETTINGS TO A/D
      MOV    AL,NORESET+NOSCLOCK+NOADSLEEP+NOHOLD+ENABLEMUX
      OUT    PARAA,AL

; FREEZE EDGE1 AND EDGE2 INTERRUPTS OFF
      OUTPT PARC,00H

; HOLD VCO RESET
      RESETC0

; RESET SERIAL PORT
      CALL  RESSER

; ENABLE INTERRUPT DRIVE RS232 OUTPUT
      CALL  PUTCHBUFON

; RESET NMI INTERRUPT FLIPFLOP
IFNDEF      NONMI
      OUTPT CLRNMI,00H  ;RESET NMI FLIPFLOP
ENDIF

; ENABLE INTERRUPTS
      INTON



; END 8088 INIT CODE
;

;*************************************************************
;
;     BACK TO MANX C STARTUP CODE
;
;*************************************************************

      mov   di,$MEMRY
      inc   di
      and   di,0fffeH    ;adjust to word boundary
      mov   $MEMRY,di    ;save memory allocation info for sbrk()
      mov   $MEMRY+2,ds
      mov   _mbot_,di
      mov   _mbot_,ds    ; SHOULD THIS BE _mbot_+2?
      mov   sbot,0ffffH ;this is the heap limit for sbrk()
      mov   sbot+2,0fff0h
      mov   _dsval_,ds
      mov   _csval_,cs  ;this is of dubious value in large code
      sti
      call  $fltinit     ;setup floating point software/hardware
      jnc   flt_ok
      hlt                ;program needs 8087 and one wasn't found
flt_ok:
      jmp   main_        ;main shouldn't return in ROM based system
$begin      endp
```

```
      ASSUME CS:CODESEG,DS:DATASEG,ES:DATASEG,SS:DATASEG

procdef     SysInt      <<IntFlg,word>>
      pushf ;save flags on stack
      mov   ax,IntFlg
      or    ax,ax
      jne   si1
      INTOFF
      jmp   si2
si1:  INTON
si2:  pop   ax    ;get flags
      and   ax,0200H ;remove all but interrupt bit
      pret
      pend  SysInt

procdef INTOFF
      INTOFF
      pret
      pend  INTOFF

;ENABLE INTERRUPTS
procdef INTON
      INTON
      pret
      pend  INTON

procdef GetAChar
      CALL  GETCH
      MOV   AH,0H
      pret
      pend  GetAChar


procdef LookAtChar
      MOV   AX,ISBFCNT  ;SEE IF ANY CHARS THERE
      CMP   AX,0
      JE    M1X
      MOV   BX,ISBFOPT
      MOV   AL,[BX]                 ;GET CHAR
      MOV   AH,0
      pret

M1X:  MOV   AX,0FFFFH
      pret
      pend  LookAtChar


procdef     PutAChar    <<pchar,word>>
      mov   ax,pchar
      CALL  PUTCH
      pret
      pend  PutAChar

procdef     PutAWord    <<pword,word>>
      mov   ax,pword
      mov   al,ah
      CALL  PUTCH
      mov   ax,pword
      CALL  PUTCH
      pret
      pend  PutAWord
```

```
procdef     Pause <<pcount,word>>
      mov   ax,pcount
      CALL  PAUSE
      pret
      pend  Pause

procdef     SetTick     <<stcountl,word>,<stcounth,word>>
      mov   NMIFRZ,0FFFFh
      mov   ax,stcounth
      mov   NMICNTH,ax
      mov   ax,stcountl
      mov   NMICNTL,ax
      mov   NMIFRZ,0h
      pret
      pend  SetTick

procdef     GetTick
      MOV   NMIFRZ,0FFFFH
      mov   ax,NMICNTL
      mov   dx,NMICNTH
      MOV   NMIFRZ,0H
      pret
      pend  GetTick

procdef     ResetVCO
      RESETC0           ;hold line low for a while

; WAIT FOR A while
      MOV   AX,10
      CALL  PAUSE

      SETC0       ;remove reset signal
      pret
      pend  ResetVCO

procdef     MEMPoke     <<Maddro,word>,<Mdata,word>>
      push  ax
      push  si
      mov   si,Maddro
      mov   ax,Mdata
      mov   [si],al
      pop   si
      pop   ax
      pret
      pend  MEMPoke

procdef     MEMPeek     <<Maddri,word>>
      push  si
      mov   si,Maddri
      mov   al,[si]
      mov   ah,00h
      pop   si
      pret
      pend  MEMPeek

procdef     IOPoke      <<addro,word>,<data,word>>
      push  ax
      push  dx
      mov   dx,addro
      mov   ax,data
      out   dx,al
      pop   dx
      pop   ax
      pret
      pend  IOPoke
```

```
procdef     IOPeek      <<addri,word>>
      push  dx
      mov   dx,addri
      in    al,dx
      mov   ah,00h
      pop   dx
      pret
      pend  IOPeek

procdef     Sleep
      CALL SLEEP
      pret
      pend  Sleep

procdef     GetAD <<adport,word>>
      MOV   AX,adport
      CALL  GETAD
      pret
      pend  GetAD

procdef     ResetAD
      PUSH  AX
      PUSH  DX
      CALL  RESAD
      POP   DX
      POP   AX
      pret
      pend  ResetAD

;FORCE A HARD RESET
procdef     ColdReset

; MUST FIRST DISABLE NMI BY WAITING FOR IT, AND NOT
; RESETING FLIP FLOP
      JMP   $begin
      pret
      pend  ColdReset
```

```
GETAD:

;GET A/D VALUE FROM PREVIOUS CHANNEL, THEN SET UP NEW CHANNEL AS
; INDICATED BY ENTRY VALUE OF AX

;ASSUME WE START WITH NORESET,NOSCLOCK,NOADSLEEP,NOHOLD,
;   AND ENABLEMUX TO PROPER CHANNEL

;****SAVE REGS
      PUSH  CX
      PUSH  BX

;GALOOP:    MOV   NMIHAP,0H

;DISABLE INTERRUPTS
      INTOFF

;USING CHANNEL IN AX, MAKE NEXT MUX CONTROL WORD
      AND   AL,07H
      OR    AL,NORESET+NOSCLOCK+NOADSLEEP+NOHOLD+ENABLEMUX
      MOV   AH,AL

;     SETC1

;****ASSERT HOLD
      IN    AL,PARAA          ;GET CURRENT MUX SETTING
      HOLDON
      OUT   PARAA,AL    ; AND ASSERT HOLD

      MOV   AL,AH       ;TURN OFF HOLD, AND ENABLE NEX MUX CHANNEL
      OUT   PARAA,AL

;****WAIT FOR TRACK1 RDY
TRKLP:      IN    AL,PARB           ;GET BYTE
      AND   AL,02H            ;STRIP OF OTHER BITS
      JNZ   TRKLP       ;LOOP UNTIL BIT 1 IS A 0
TRKTHERE:

;     MOV   AL,AH       ;TURN OFF HOLD, AND ENABLE NEX MUX CHANNEL
;     OUT   PARAA,AL

;****READ 16 BITS INTO BX
      MOV   CX,010H           ;SET COUNT TO 16
ADLOOP:
      MOV   AL,AH
      SCLOCKON
      OUT   PARAA,AL    ;TURN ON CLOCK
      MOV   AL,AH
      OUT   PARAA,AL    ;TURN OFF CLOCK

      IN    AL,PARB
      ROR   AL,1        ;GET BIT FROM PARC(0) INTO CF
      RCL   BX,1        ;PUT IT IN BX
      LOOP  ADLOOP

;****XFER BX TO AX
      MOV   AX,BX

;ENABLE INTERRUPTS
      INTON

;****RESTORE REG'S
      POP   BX
      POP   CX

      RET
```

```
; RESET A/D, MUX
RESAD:
; INVOKE THE RESET LINE
      MOV   AL,RESET + NOSCLOCK + NOADSLEEP + NOHOLD + ENABLEMUX + CHAN0
      OUTPT PARAA,AL

; WAIT FOR AT LEAST ONE SECOND, THIS INSURES REFERENCE SETTLED
      MOV   AX,300
      CALL  PAUSE

; RESTORE RESET LINE
      MOV   AL,NORESET + NOSCLOCK + NOADSLEEP + NOHOLD + ENABLEMUX + CHAN0
      OUTPT PARAA,AL

; WAIT FOR AT LEAST 2 SECONDS, THIS INSURES A/D CALIBRATED
      MOV   AX,600
      CALL  PAUSE
      RET

; RESSET SERIAL PORT
RESSER:
      PUTSERCON 00H
      PUTSERCON 00H
      PUTSERCON 00H
      PUTSERCON 40H
      PUTSERCON 4EH
;     OUTPT SERCON,0CEH ;1STOP,NOPARITY,8BIT,CLKX16
      PUTSERCON 37H
;     OUTPT SERCON,15H  ;CLEAR ERROR, ENABLE TRANSMITTER
      OUTPT SERCON,37H  ;CLEAR ERROR, ENABLE TRANSMITTER,RTS, DTR
      MOV   AX,OFFSET ISBUF
      MOV   ISBFIPT,AX
      MOV   ISBFOPT,AX
      MOV   AX,0H
      MOV   ISBFCNT,AX
      MOV   BYTE PTR PUTCHBUFFLAG,00H
SERPAS:     RET

DOSER:      PUSH  AX          ;SAVE REGISTER
      GETSERCON
      AND     AL,02H      ;SEE IF HOLDING A CHAR
      JE    SERINY            ;EXIT IF NO CHAR THERE
      PUSH  BX          ;SAVE ANOTHER REGISTER
      MOV   BX,ISBFIPT  ;GET INPUT POINTER
      GETSERDAT
      MOV   [BX],AL           ;STORE CHARACTER THROUGH INPUT POINTER
      INC   BX          ;INCREMENT POINTER
      CMP   BX,OFFSET ISBUF+ISBUFLEN
      JB    SERIN1            ;JUMP IF BX IS BELOW THE END VALUE
      MOV   BX,OFFSET ISBUF
SERIN1:     MOV   ISBFIPT,BX  ;RESTORE INPUT POINTER
      CMP   ISBFCNT,ISBUFLEN  ; FULL BUFFER ALREADY?
      JE    SERIN2            ; IF SO CONTINUE BELOW
      INC   ISBFCNT
      JMP   SERINX
SERIN2:     MOV   BX,ISBFOPT  ;GET OUTPUT POINTER
      INC   BX          ; INCREMENT IT
      CMP   BX,OFFSET ISBUF+ISBUFLEN      ;CHECK FOR WRAP AROUND
      JB    SERIN3            ;JUMP BELOW IF NONE
      MOV   BX,OFFSET ISBUF   ;ELSE LOAD BASE VALUE
SERIN3:     MOV   ISBFOPT,BX  ;RESTORE POINTER
SERINX:     POP   BX          ;RESTORE REGISTER
SERINY:     POP   AX
      RET
```

```
DOPUTCH:
;CLEAR TRDY (EDGE1) FLIP FLOP
      PUSH  AX
      RESETC6
      SETC6
      POP   AX
PCHECK:
;CHECK IF SERIAL OUTPUT PORT IS READY
      PUSH  AX
      GETSERCON
      AND   AL,01H
      JE    DOPX          ; IF NOT, THEN EXIT BELOW

;CHECK IF CHARS IN BUFFER
      MOV   AX,OSBFCNT
      CMP   AX,00H
      JE    DOPX          ;IF NOT, THEN EXIT BELOW

; READY TO SEND ANOTHER CHARACTER!
; UPDATE COUNT
      DEC   AX
      MOV   OSBFCNT,AX

; GET NEXT CHARACTER FROM BUFFER
      PUSH  BX
      MOV   BX,OSBFOPT
      MOV   AL,[BX]
; SEND IT
      PUTSERDAT AL
;UPDATE POINTER INTO BUFFER
      INC   BX
      CMP   BX,OFFSET OSBUF+OSBUFLEN
      JB    DOP2
      MOV   BX,OFFSET OSBUF
DOP2: MOV   OSBFOPT,BX
      POP   BX

; AND EXIT
DOPX:
      POP   AX
      RET

PUTCHBUFON:
; TURN ON PUTCHBUF FLAG
      MOV   BYTE PTR PUTCHBUFFLAG,0FFH

; RESET POINTERS AND COUNT
      PUSH  AX
      MOV   AX,OFFSET OSBUF
      MOV   OSBFIPT,AX
      MOV   OSBFOPT,AX
      MOV   AX,0
      MOV   OSBFCNT,AX

; RESET TRDY (EDGE1) FLIPFLOP
      RESETC6
      SETC6
      POP   AX
      RET
```

```
PUTCHBUFOFF:
      PUSH  AX
      PUSH  BX
      PUSH  CX

;CHECK IF IT IS ALREADY OFF
      MOV   AL,PUTCHBUFFLAG
      CMP   AL,00H
      JE    PBOFFX              ;IF SO EXIT

; DISABLE TRDY (EDGE1) FLIPFLOP
      RESETC6

;TURN OF FLAG
      MOV   BYTE PTR PUTCHBUFFLAG,00H

;GET OUTPUT POINTER AND COUNT
      MOV   BX,OSBFOPT
      MOV   CX,OSBFCNT

;IF COUNT IS ZERO, THEN EXIT
      CMP   CX,00H
      JE    PBOFFX

;LOOP, USING OTHER PUTCH TO SEND REMAINING CHARS
PBOFF1:     MOV   AL,[BX]
      CALL  PUTCH
      INC   BX
      CMP   BX,OFFSET OSBUF+OSBUFLEN
      JB    PBOFF2
      MOV   BX,OFFSET OSBUF
PBOFF2:     LOOP  PBOFF1

PBOFFX:     POP   CX
      POP   BX
      POP   AX
      RET

PUTCHCHECK:
      PUSH  AX
      GETSERCON
;     INPUT SERCON
      AND   AL,01H
      JE    PUTCHCHEKX
      MOV   AX,OSBFCNT
      CMP   AX,00H
      JE    PUTCHCHEKX
      DEC   AX
      MOV   OSBFCNT,AX
      PUSH  BX
      MOV   BX,OSBFOPT
PBCHK1:     MOV   AL,[BX]
      PUTSERDAT AL
;     OUTPT SERDAT,AL
      INC   BX
      CMP   BX,OFFSET OSBUF+OSBUFLEN
      JB    PBCHK2
      MOV   BX,OFFSET OSBUF
PBCHK2:     MOV   OSBFOPT,BX
      POP   BX
PUTCHCHEKX:
      POP   AX
      RET
```

```
PUTCH:
      PUSH    AX
      MOV   AL,PUTCHBUFFLAG
      CMP   AL,00H
      JE    PUTCH1
      INTOFF
      MOV   AX,OSBFCNT
      CMP   AX,OSBUFLEN
      JE    PUTCHX1
      INC   AX
      MOV   OSBFCNT,AX
      POP   AX
      PUSH  AX
      PUSH  BX
      MOV   BX,OSBFIPT
      MOV   [BX],AL
      INC   BX
      CMP   BX,OFFSET OSBUF+OSBUFLEN
      JB    PUTCHX2
      MOV   BX,OFFSET OSBUF
PUTCHX2:
      MOV   OSBFIPT,BX
      POP   BX
PUTCHX1:
      INTON
      POP   AX
      INTOFF
      CALL  PCHECK
      INTON
      RET
PUTCH1:
      GETSERCON
      AND     AL,01H          ;SEE IF HOLDING
      JE      PUTCH1
      POP     AX
      PUTSERDAT AL
      RET

GETNUM:     PUSH  DX
      PUSH  BX
      PUSH  CX
      MOV   CX,10
      MOV   BX,0
GTNMLP:     CALL  GETCH
      CALL  PUTCH
      AND   AX,07FH           ;MASK OUT UNWANTED BITS
      CMP   AL,13       ;IF RETURN, THEN EXIT
      JE    GETNUMX
      CMP   AL,'0'            ;IF LESS THAN '0', TRY AGAIN
      JL    GTNMLP
      CMP   AL,'9'            ;IF GT THAN '9', TRY AGAIN
      JG    GTNMLP
      SUB   AL,'0'            ;TURN INTO NUMBER
      PUSH  AX          ;SAVE DIGIT ON THE STACK
      MOV   AX,BX       ;MULTIPLY CURRENT VALUE BY 10
      MUL   CX
      MOV   BX,AX
      POP   AX          ;ADD DIGIT TO CURRENT VALUE
      ADD   BX,AX
      JMP   GTNMLP            ;AND LOOP
GETNUMX:
      MOV   AX,BX       ;RETURN VALUE IN AX
      POP   CX
      POP   BX
      POP   DX
      RET
```

```
GETCH:      PUSH  BX
GETCH1:
      MOV   AX,ISBFCNT
      CMP   AX,00H
      JE      GETCH2
      MOV   BX,ISBFOPT
      MOV   AL,[BX]                 ;GET CHAR
      PUSH  AX
;     DECREMENT COUNT
      DEC   ISBFCNT
;     UPDATE OUTPUT POINTER
      INTOFF
      INC   BX          ; INCREMENT OUTPUT POINTER
      CMP   BX,OFFSET ISBUF+ISBUFLEN      ;CHECK FOR WRAP AROUND
      JB    GETCH3            ;JUMP BELOW IF NONE
      MOV   BX,OFFSET ISBUF   ;ELSE LOAD BASE VALUE
GETCH3:     MOV   ISBFOPT,BX  ;RESTORE POINTER
      INTON
      POP   AX
      POP   BX
      RET
GETCH2:
      CALL  SLEEP
      JMP   GETCH1
CR:
      MOV   AL,0DH
      CALL  PUTCH
      RET
CRLF:
      CALL  CR
      CALL  LF
      RET
LF:
      MOV     AL,0AH
      CALL    PUTCH
      RET
SPACE:
      MOV     AL,20H
      CALL    PUTCH
      RET
PRWORD:     PUSH  AX          ;PRINTS A WORD IN HEX FORMAT
      PUSH  AX
      MOV   AL,AH
      CALL  PRBYTE
      POP   AX
      CALL  PRBYTE
      POP   AX
      RET
BACKWORD:   PUSH  AX          ;PRINTS A WORD IN HEX FORMAT
      PUSH  AX
      CALL  PRBYTE
      POP   AX
      MOV   AL,AH
      CALL  PRBYTE
      POP   AX
      RET
TIMNIB:     PUSH  AX
      MOV   AL,AH
      CALL  PUTCHNIB
      POP   AX
      RET
```

```
PRBYTESP:
      CALL  PRBYTE
      CALL  SPACE
      RET
DISPDIG:
      PUSH  AX
      PUSH  BX
      PUSH  CX
      PUSH  DX
      MOV   BX,0AH            ;DIVISOR IS 10.
      MOV   CX,01H            ;SET INITIAL COUNT TO 1
      CMP   AX,0H       ;INITIAL ZERO?
      JE    DISPDIGLP3  ; IF SO PRINT ONE ZERO AND EXIT
      MOV   CX,0H       ;ELSE CLEAR COUNT AND ACCUM DIGITS
DISPDIGLP1:
      MOV   DX,0H       ;CLEAR HIGH ORDER BYTE
      DIV   BX          ;
      INC   CX          ; UPDATE COUNT
      PUSH  DX          ; SAVE DIGIT
      CMP   AX,0H       ;DONE YET?
      JNE   DISPDIGLP1
DISPDIGLP2:
      POP   AX
DISPDIGLP3:
      CALL  PUTCHNIB    ;SEND NIBBLE
      DEC   CX          ;DECREASE CX
      JNE   DISPDIGLP2
      POP   DX
      POP   CX
      POP   BX
      POP   AX
      RET
PRBYTE:
      PUSH    CX        ;SAVE CX
      PUSH    AX        ;SAVE AX
      SHR     AL,1
      SHR     AL,1
      SHR     AL,1
      SHR     AL,1
      CALL    PUTCHNIB
      POP     AX
      CALL    PUTCHNIB
      POP     CX
      RET
PRBYTEFILL:
      PUSH    CX        ;SAVE CX
      PUSH    AX        ;SAVE AX
      SHR     AL,1
      SHR     AL,1
      SHR     AL,1
      SHR     AL,1
      CALL    PUTCHNIB
      MOV   AL,AH
      CALL  PUTCH
      POP     AX
      CALL    PUTCHNIB
      POP     CX
      RET
```

```
PUTCHNIB:
        PUSH  AX
        AND     AL,0FH
        CMP     AL,0AH
        JL      PN1
        ADD     AL,07H
PN1:    ADD     AL,30H
        CALL    PUTCH
        POP   AX
        RET

SLEEP:      PUSH  AX
        OUTPT SLEEPFLOP,0H
        POP   AX
        RET

PAUSE:      MOV   NMIFRZ,0FFFFH
        MOV   NMICNTH,0000H
        MOV   NMICNTL,AX
        MOV   NMIFRZ,0H
PAUSEL:     .
IFDEF NONMI
        PUSH  CX
PAUSEL1:
        MOV   CX,2000
PAUSEL2:
        CMP   AX,CX
        LOOP  PAUSEL2
        DEC   AX
        JNZ   PAUSEL1
        POP   CX
ELSE
        TEST  NMICNTL,0FFFFH
        JZ    PAUSEX
        CALL  SLEEP
        JMP   PAUSEL
ENDIF
PAUSEX:     RET
```

```
;*************************************************************
;***                  INTERRUPT CALLS                      ***
;*************************************************************

INT7: CALL  DOSER
      CALL  DOPUTCH
      IRET
INT6: CALL  DOPUTCH
      IRET
INT5: CALL  DOSER
      IRET
INT4: IRET
INT3: CALL  DOSER
      CALL  DOPUTCH
      IRET
INT2: CALL  DOPUTCH
      IRET
INT1: CALL  DOSER
      IRET

INTNMI:
IFDEF NONMI
      IRET
ENDIF
      PUSH  AX
      MOV   NMIHAP,0FFFFH
; RESET NMI INTERRUPT FLIPFLOP
      OUT   CLRNMI,AL
;     OUTPT CLRNMI,00H  ;RESET NMI FLIPFLOP

; CHECK COUNT DOWN TO ZERO COUNTER
      TEST  NMIFRZ,0FFFFH
      JNZ   INTNMIX

      TEST  NMICNTL,0FFFFH
      JNZ   INTDECIT

      TEST  NMICNTH,0FFFFH

; IF ZERO, THEN EXIT
      JZ    INTNMIX
      DEC   NMICNTH
      DEC   NMICNTL
      JMP   INTNMIX

INTDECIT:
; ELSE, DECREMENT BY 1
      DEC   NMICNTL

INTNMIX:
      PUSH  BP
      PUSH  SI
      PUSH  DI
      PUSH  DX
      PUSH  CX
      PUSH  BX
      CALL  TickInt_
      POP   BX
      POP   CX
      POP   DX
      POP   DI
      POP   SI
      POP   BP

      POP   AX
      IRET
```

```
;SAMPLE INTERVAL INTERRUP ROUTINE
SAMPINT:
      PUSH  AX
      PUSH  DX
      PUSH  SI
      PUSH  DI

SMPIX:


      POP   DI
      POP   SI
      POP   DX
      POP   AX
      IRET
```

```
;_____________________________________________________________________
;
;//////////////////////////   DATA SPACE   \\\\\\\\\\\\\\\\\\\\\\\\\\\\
;_____________________________________________________________________
;_____________________________________________________________________

DATASEG     SEGMENT

ISBFIPT             DW    ?
ISBFOPT             DW    ?
ISBFCNT             DW    ?
OSBFIPT             DW    ?
OSBFOPT             DW    ?
OSBFCNT             DW    ?
PUTCHBUFFLAG        DB    ?
LPOINT              DW    ?
LLENGTH             DW    ?

; REAL TIME CLOCK REGISTERS
MILLISECS   DW    ?
SECOND              DW    ?
NMICNTL             DW    ?
NMICNTH             DW    ?
NMIFRZ              DW    ?
NMIHAP              DW    ?

;     IF INTFLAG  = 0   THEN THE STATE OF THE INTERRUPT IS CHANGING
;                 = 1   THEN SLOW INTERRUPTS ARE ON
;                 =255  THEN FAST INTERRUPTS ARE ON

INTFLAG             DB    ?

; A time sync VARIABLE USED TO remember when an edge has been received
EDGEFLAG    DW    ?
; OUTPUT DATA REGISTERS
OUTBYTE             DB    ?
BITCOUNT    DB    ?
CTLFLAG             DB    ?
OUTBUF              DB    ?
BUFFUL              DB    ?
NEWCTL              DB    ?

ISBUFLEN EQU 100H
ISBUF       DB    ISBUFLEN DUP (?)

OSBUFLEN EQU 200H
OSBUF       DB    OSBUFLEN DUP (?)

DATASEG     ENDS
;*********************************************************************
codeseg ends
      end   $begin
```

```
      nlist
; Copyright (C) 1985 by Manx Software Systems, Inc.
; :ts=8
      ifndef      MODEL
MODEL equ   0
      endif
      if    MODEL and 1
      largecode
FARPROC     equ 1
FPTRSIZE equ      4
      else
FPTRSIZE equ      2
      endif
      if    MODEL and 2
LONGPTR equ 1
      endif

;this macro to be used on returning
;restores bp and registers
pret  macro
if havbp
      pop bp
endif
      ret
      endm

internal macro    pname
      public      pname
pname proc
      endm

intrdef     macro pname
      public      pname
ifdef FARPROC
      pname label far
else
      pname label near
endif
      endm

procdef     macro pname, args
      public pname&_
ifdef FARPROC
      _arg  = 6
      pname&_     proc  far
else
      _arg  = 4
      pname&_     proc  near
endif
ifnb <args>
      push bp
      mov bp,sp
      havbp = 1
      decll <args>
else
      havbp = 0
endif
      endm
```

```
entrdef     macro pname, args
      public pname&_
ifdef FARPROC
      _arg  = 6
      pname&_:
else
      _arg  = 4
      pname&_:
endif
ifnb <args>
if    havbp
      push  bp
      mov   bp,sp
else
      error must declare main proc with args, if entry has args
endif
      decll <args>
endif
      endm

;this macro equates 'aname' to arg on stack
decl  macro       aname, type
havtyp      = 0
ifidn <type>,<byte>
      aname equ   byte ptr _arg[bp]
      _arg = _arg + 2
      havtyp = 1
endif
ifidn <type>,<dword>
      aname equ dword ptr _arg[bp]
      _arg = _arg + 4
      havtyp = 1
endif
ifidn <type>,<cdouble>
      aname equ qword ptr _arg[bp]
      _arg = _arg + 8
      havtyp = 1
endif
ifidn <type>, <ptr>
      ifdef LONGPTR
            aname equ dword ptr _arg[bp]
            _arg = _arg + 4
      else
            aname equ   word ptr _arg[bp]
            _arg = _arg + 2
      endif
      havtyp = 1
endif
ifidn <type>, <fptr>
      ifdef FARPROC
            aname equ dword ptr _arg[bp]
            _arg = _arg + 4
      else
            aname equ   word ptr _arg[bp]
            _arg = _arg + 2
      endif
      havtyp = 1
endif
ifidn <type>, <word>
      aname equ   word ptr _arg[bp]
      _arg = _arg + 2
      havtyp = 1
endif
ife   havtyp
      error -- type is unknown.
endif
```

```
      endm

;this macro loads an arg pointer into DEST, with optional SEGment
ldptr macro dest, argname, seg
ifdef LONGPTR
      ifnb <seg>        ;;get segment if specified
            ifidn <seg>,<es>
                  les   dest,argname
            else
                  ifidn <seg>,<ds>
                        lds dest,argname
                  else
                        mov dest, word ptr argname
                        mov seg, word ptr argname[2]
                  endif
            endif
      else
            ifidn <dest>,<si>       ;;si gets seg in ds
                  lds   si, argname
            else
                  ifidn <dest>,<di> ;;or, es:di
                        les   di, argname
                  else
                        garbage error: no seg for long pointer
                  endif
            endif
      endif
else
      mov dest, word ptr argname    ;;get the pointer
ENDIF
      ENDM

decll macro list
      IRP   i,<list>
      decl i
      ENDM
      ENDM

pend  macro pname
pname&_     endp
      endm

retptrm     macro src,seg
mov   ax, word ptr src
ifdef LONGPTR
      mov   dx, word ptr src+2
endif
      endm

retptrr     macro src,seg
mov   ax,src
ifdef LONGPTR
      ifnb <seg>
            mov   dx, seg
      endif
endif
      endm

retnull     macro
ifdef LONGPTR
      sub   dx,dx
endif
      sub   ax,ax
      endm
```

```
pushds      macro
      ifdef LONGPTR
      push  ds
      endif
      endm

popds macro
      ifdef LONGPTR
      pop   ds
      endif
      endm

finish      macro
codeseg     ends
      endm

      list
codeseg     segment     byte public 'code'
      assume      cs:codeseg
```

Appendix D - C language Source Code

DIGGER.C

STKSIZ.C

DIGGER.C

```
/*
      DIGGER.C
      REMOTE DIGITIZER/TRANSMITTER Program
      DESCRIPTION: 80C88 "C" LANGUAGE MODULE FOR USE WITH DIGGER.ASM,
      JUNE 21, 1991
      COPYRIGHT CUTLER DIGITAL DESIGN 1991
      Rev.1  09-18-91 Reese T. Cutler << Original program >>
*/

#define version 1

/*    This program has been written to operate in a custom
      8088 computer controller has up to 3 analog inputs and a
      serial I/O line.

      The program calls on various assembly language routines which can be
      found in the assembly language file DIGGER.ASM and which are described
      below.  In addition, some "C" language routines in this program
      can be called by assembly interrupt routines in DIGGER.ASM

      The DIGGER.ASM code has all the hooks necessary for the application
      to work in a standalone romable manner.

*/


/*
      The following assembly language routines are available:

      INTON()                 enables interrupts

      INTOFF()          disables interrupts

      SysInt(int state) sets interrupt (on/off) according to
                        the value in state.  If state is 0, interrupts
                        are disabled, else enabled.  This routine
                        in integer that indicates the state of
                        interrupts, before change: 0 if they were
                        disabled, 512 if they were enabled.

      ColdReset()       restarts the receiver board

      GetAChar()        returns a char in an integer

      PutAChar(int char)      sends a character, waits til done

      PutAWord(int word)      sends a word, waits til done

      LookAtChar()            checks to see if any chars in serial
                        input buffer, if not, returns -1,
                        if so, returns value for char
                        does not remove char from buffer

      Pause(int tickcount)    waits for 'tickcount' ticks, each tick
                        is 33.3 ms

      SetTick(int tickcount)  sets tick counter

long  GetTick()         returns tick counter value, tick counter
                        counts down to zero, then stops until set
                        again.

      Sleep()                 powers down processor until next interrupt
```

```
      MEMPoke(int address, int data)
                        loads a memory location (indicated by address)
                        with a new value (indicated by data)

      MEMPeek(int address)    reads a memory location (indicated by address)

      IOPoke(int address, int data)
                        loads an io location (indicated by address)
                        with a new value (indicated by data)

      IOPeek(int address)     reads an io location (indicated by address)

      ResetAD()         resets a/d, takes 3 seconds

      ResetVCO()        resets VCO

      GetAD(int nextchan)     returns a/d value for current channel,
                        after setting mux for next channel

*/


#include "\manxc\header\stdio.h"


/*********************************************************/
/* logical definitions                                   */
/*********************************************************/
#define     true  1
#define false     0
#define TRUE      1
#define     FALSE 0

/* Parallel port addresses */
#define     PARA  64
#define PARB      65
#define     PARC  66
#define     PARCON      67

/* default ticks per second */
#define     DefaultTickRate         300   /* hardware clock interrupt rate */
#define     DefaultTickLimit  3     /* divisor of hardware clock */
#define     DefaultTimeoutSeconds   20    /* Seconds of inactivity until revert*/
#define     DefaultStartChar  13    /* carriage return */
#define     DefaultStopChar         10    /* line feed */
#define     DefaultResyncRate 1440L /* minutes between resync */
#define     DefaultBaudRate         9600  /* baud rate */

/* special characters */
#define BACKSPACE 8
#define     SPACE       32
#define CARRIAGERETURN  13
#define     LINEFEED    10
#define     BELL        7
#define     DELETE            127
```

```
/* commands */
#define     TIMEOUT     0
#define COMM1     1
#define COMM2     2
#define COMM3     3
#define COMM4     4
#define COMM5     5
#define COMM6     6
#define COMM7     7
#define COMM8     8
#define COMM9     9
#define COMM10    10
#define COMM11    11
#define COMM12    12
#define COMM13    13
#define COMM14    14
#define COMM15    15
#define COMM16    16
#define COMM17    17
#define COMM18    18
#define COMM19    19
#define COMM20    20

#define BADCOMM 21

/* nvram addresses */

#define     NVROffset   0x6000

#define     PasswordAdd 0
#define ValidNVRAdd     80
#define     StopCharAdd 160
#define     StartCharAdd      161
#define     ChanCountAdd      162
#define     FirstChanAdd      163
#define     SecondChanAdd     164
#define     ThirdChanAdd      165
#define     FourthChanAdd     166
#define     FifthChanAdd      167
#define     SixthChanAdd      168
#define     SeventhChanAdd    169
#define     EighthChanAdd     170
#define     TickLimitAdd      171
#define     TickRateAdd 173
#define     baudRateAdd 175
#define     TTCountAdd  177
#define     RSCountAdd  181
#define     NextAdd           185

#define     StartPassWord     "USGS"
#define     ValidMessage      "Digger, (c) Cutler Digital Design, 1991"


/* function predefinitions */
char upper(int c);
long GetTick();
long NVReadLong(int address);
```

```
/* global variables */
char  Password[80]=     {65,65,65,65,65,65,0,0,0,0
                  ,0,0,0,0,0,0,0,0,0,0
                  ,0,0,0,0,0,0,0,0,0,0
                  ,0,0,0,0,0,0,0,0,0,0
                  ,0,0,0,0,0,0,0,0,0,0
                  ,0,0,0,0,0,0,0,0,0,0
                  ,0,0,0,0,0,0,0,0,0,0
                  };
char  tPassword[80]=    {0,0,0,0,0,0,0,0,0,0
                  ,0,0,0,0,0,0,0,0,0,0
                  ,0,0,0,0,0,0,0,0,0,0
                  ,0,0,0,0,0,0,0,0,0,0
                  ,0,0,0,0,0,0,0,0,0,0
                  ,0,0,0,0,0,0,0,0,0,0
                  ,0,0,0,0,0,0,0,0,0,0
                  };
int   pCount=6;
int   channelMap[8]=    {4,5,3,0,0,0,0,0};
int   channelCount=3;
int   NVFlag=FALSE;
int   tick=0,csec=0,second=0,minute=0,hour=0,day=0,year=0;
int   SampleCount=0,SampleDec=0;
int   data[8]={0,1,2,3,4,5,6,7};
int   TickLimit=DefaultTickLimit;
int   TickRate=DefaultTickRate;
long  TimeoutTickCount=DefaultTimeoutSeconds*DefaultTickRate;
int   StartChar=DefaultStartChar;
int   StopChar=DefaultStopChar;
long  resetCount=(DefaultResyncRate *(DefaultTickRate/DefaultTickLimit)*60);
long  resetCounter=(DefaultResyncRate *(DefaultTickRate/DefaultTickLimit)*60);
int   baudRate=DefaultBaudRate;
```

```
main()   /*************************************************************/
{
char InputLine[80];
char  myChar;
int   index;

PrintString("\nU. S. Geological Survey Field Digitizer\n");
PrintString("(c) Cutler Digital Design 1991\n\n");

ResetVCO();
ResetAD();

/* Set up Password */
if(!RestoreFromNVR())
      SetDefault();

GetAD(channelMap[channelCount-1]);

ShowStatus();


/* Are there any input commands?  Return after 20 seconds of inactivity */
DoCommands();

SampleCount=0;
while (true)
      {
      if (SampleCount>0)
            {
            INTOFF();
            SampleDec+=1;
            INTON();
            resetCounter-=1;
            if (resetCounter == 0L)
                  {
                  ResetVCO();
                  ResetAD();
                  tick=0;
                  SampleCount=0;
                  SampleDec=0;
                  resetCounter=resetCount;
                  }
            for (index=0 ; index < channelCount ; index++)
                  data[index]=GetAD(channelMap[index]);

            PutAChar(StartChar);

            for (index=0 ; index < channelCount ; index++)
                  PutAWord(data[index]);

            PutAChar(StopChar);

            if (LookAtChar()!=-1)
                  {
                  CheckForPassword();
                  }
            }
      else
            Sleep();
      }

} /*******  End main() ***************************************************/
```

```
/*************************************************************************
/
/      Title:      CheckForPassword()
/
*************************************************************************/
CheckForPassword()
{
int myChar;
char InputLine[80];
int index;

for ( index=0 ; index < pCount-1 ;index++)
      {
      tPassword[index]=tPassword[index+1];
      }
tPassword[pCount-1]=upper(GetAChar());

if(!strcmp(Password,tPassword))
      {
      for (index=0;index<=pCount;index++)
            tPassword[index]=0;

      /* Are there any input commands?  Return after 20 seconds of inactivity */
      DoCommands();
/*
      CR();
      PrintString("waiting for <cr> to continue ");
      GetLine(InputLine,80);
      PrintString(InputLine);
*/
      GetAD(channelMap[channelCount-1]);
      INTOFF();
      SampleCount=0;
      SampleDec=0;
      INTON();
      }
}

/*************************************************************************
/
/      Title:      DoCommands()
/
*************************************************************************/
DoCommands()
{

SetTick(TimeoutTickCount);
PrintString("\nEnter a command (type MENU for selection)\n");

while (GetTick()!=0)
      ExecuteCommand(GetCommand());
}

/*************************************************************************
/
/      Title:      PrintTime()
/
*************************************************************************/
/*
PrintTime()
{
char tm[80];
sprintf(tm,"%02d:%02d:%02d\n\r",hour,minute,second);
PrintString(tm);
}
*/
```

```
/*************************************************************************
/
/      Title:      GetCommand()
/
*************************************************************************/
GetCommand()
{
int command;
char line[80];

if (GetLineT(line))
      {
      if (line[0]==0)
            command=BADCOMM;
      else if (!strcmp(line,"T"))
            command=COMM1;
      else if (!strcmp(line,"S"))
            command=COMM2;
      else if (!strcmp(line,"V"))
            command=COMM3;
      else if (!strcmp(line,"C"))
            command=COMM4;
      else if (!strcmp(line,"E"))
            command=COMM5;
      else if (!strcmp(line,"R"))
            command=COMM6;
      else if (!strcmp(line,"M"))
            command=COMM10;
      else if (!strcmp(line,"HELP"))
            command=COMM10;
      else if (!strcmp(line,"?"))
            command=COMM10;
      else if (!strcmp(line,"D"))
            command=COMM11;
      else if (!strcmp(line,"STATUS"))
            command=COMM1;
      else if (!strcmp(line,"SET"))
            command=COMM2;
      else if (!strcmp(line,"SAVE"))
            command=COMM3;
      else if (!strcmp(line,"COLLECT"))
            command=COMM4;
      else if (!strcmp(line,"RESET"))
            command=COMM5;
      else if (!strcmp(line,"RESTORE"))
            command=COMM6;
      else if (!strcmp(line,"MENU"))
            command=COMM10;
      else if (!strcmp(line,"DEFAULT"))
            command=COMM11;
      else if (!strcmp(line,"COMM12"))
            command=COMM12;
      else if (!strcmp(line,"COMM13"))
            command=COMM13;
      else if (!strcmp(line,"COMM14"))
            command=COMM14;

      else
            command=BADCOMM;
      }
else
      command=TIMEOUT;

return(command);
}
```

```
/*************************************************************************
/
/      Title:     ExecuteCommand(int command)
/
*************************************************************************/
ExecuteCommand(int command)
{
long templong;
char  line[80];
int temp;

if (command != TIMEOUT)
      {
      templong=GetTick();
      SetTick(TimeoutTickCount);
      switch(command)
            {
            case COMM1:
                  ShowStatus();
                  break;
            case COMM2:
                  DoSet();
                  break;
            case COMM3:
                  SaveToNVR();
                  break;
            case COMM4:
                  SetTick(0L);
                  break;
            case COMM5:
                  PrintString("\nResetting A/D and VCO \n");
                  ResetVCO();
                  ResetAD();
                  SetTick(TimeoutTickCount);
                  break;
            case COMM6:
                  if (RestoreFromNVR())
                        ShowStatus();
                  else
                        PrintString("\n\nCould not restore from NV ram\n");
                  break;
            case COMM10:
                  DoMenu();
                  break;
            case COMM11:
                  SetDefault();
                  PrintString("\nDefault values set\n");
                  break;
            case COMM12:
                  PrintString("Command 12\n");
                  break;
            case COMM13:
                  PrintString("Command 13\n");
                  break;
            case COMM14:
                  PrintString("Command 14\n");
                  break;

            case BADCOMM:
                  PrintString("\nCommand not recognized, type MENU for list\n");
                  break;
            default:
                  break;
            }
      }
}
```

```
/*************************************************************************
/
/      Title:     GetLineT(char line[])
/
*************************************************************************/
GetLineT(char line[])
{
int gotLine;
int charcount;

line[0]=0;
gotLine=FALSE;
charcount=0;
SetTick(TimeoutTickCount);

while (!gotLine)
      {
      if (LookAtChar() == -1)
            Sleep();
      else
            {
            gotLine = ServeLine(line,80,&charcount);
            if (! gotLine) SetTick(TimeoutTickCount);
            }
      if (GetTick() == 0) gotLine = TRUE;
      }


return( GetTick() > 0 );

}

/*************************************************************************
/
/      Title:     DoMenu()
/
*************************************************************************/
DoMenu()
{
PrintString("\nCommands:\n");
PrintString("      STATUS - T - display current status\n");
PrintString("         SET - S - set various parameter values\n");
PrintString("        SAVE - V - save current parameter values permanently\n");
PrintString("     COLLECT - C - exit command mode\n");
PrintString("       RESET - E - reset a/d and vco\n");
PrintString("     RESTORE - R - restore parameter values from non volatile
memory\n");
PrintString("        MENU - M - show commands\n");
PrintString("     DEFAULT - D - set all parameters to default values\n");
PrintString("\n\n");
}
```

```
/*************************************************************************
/
/      Title:     DoSet()
/
*************************************************************************/
DoSet()
{
int OpType;

OpType = 1;
SetTick(TimeoutTickCount);
while ( (OpType != 0 ) && (GetTick() > 0))
      {
      CR();
      PrintString("Set Routine:\n");
      PrintString("    1.  Indicate Hardware Clock Rate\n");
      PrintString("    2.  Indicate Hardware Baud Rate\n");
      PrintString("    3.  Set Sample Rate\n");
      PrintString("    4.  Set Inactivity Timeout\n");
      PrintString("    5.  Set Start Character\n");
      PrintString("    6.  Set Stop Character\n");
      PrintString("    7.  Set New Password\n");
      PrintString("    8.  Set Resync Interval\n");
      PrintString("    9.  Set Channels\n");
      CR();
      PrintString("    0.  Exit the Set Routine\n");
      CR();
      if (GetNumTD(&OpType, "Type the number of your selection:  "))
            {
            CR();
            CR();
            switch(OpType)
                  {case 1:
                        DoSetClock();
                        break;
                  case 2:
                        DoSetBaudRate();
                        break;
                  case 3:
                        DoSetFrequency();
                        break;
                  case 4:
                        DoSetTimeout();
                        break;
                  case 5:
                        DoSetStartChar();
                        break;
                  case 6:
                        DoSetStopChar();
                        break;
                  case 7:
                        DoSetPassword();
                        break;
                  case 8:
                        DoSetResync();
                        break;
                  case 9:
                        DoSetChannels();
                        break;
                  default:
                        PrintString("\nBACK TO COMMAND MODE\n\n");
                        break;}}
      else
            PrintString("\nTimeout during Set Routine\n");
      }
}
```

```
/*************************************************************************
/
/       Title:      NVRead(int address)
/
*************************************************************************/
NVRead(int address)
{
return( MEMPeek( (address & 0x1fff) + NVROffset) );
}

/*************************************************************************
/
/       Title:      NVReadWord(int address)
/
*************************************************************************/
NVReadWord(int address)
{
unsigned temp;

temp=NVRead(address+1);
temp=temp <<8;
temp=temp | NVRead(address);
return( temp );
}

/*************************************************************************
/
/       Title:      long NVReadLong(int address)
/
*************************************************************************/
long NVReadLong(int address)
{
unsigned long temp;

temp=NVReadWord(address+2);
temp=(temp<<16) | (NVReadWord(address) & 0x0000ffff);
return(temp);
}

/*************************************************************************
/
/       Title:      NVWrite(int a1, int data)
/
*************************************************************************/
NVWrite(int a1, int data)
{
int limit;
int address;

data = data&0xff;
address=(a1 & 0x1fff)+ NVROffset;
limit=10000;
MEMPoke(address,data);
while ((limit-- >= 0) && (data != MEMPeek(address))) ;
}
```

```
/*************************************************************************
/
/       Title:      NVWriteWord(int a1, int data)
/
*************************************************************************/
NVWriteWord(int a1, int data)
{

NVWrite(a1,data);
NVWrite(a1+1,data>>8);
}

/*************************************************************************
/
/       Title:      NVWriteLong(int a1, long data)
/
*************************************************************************/
NVWriteLong(int a1, long data)
{
int temp;
long templ;

templ=data & 0x0000ffff;
temp=templ;
NVWriteWord(a1,temp);

templ=(data>>16) & 0x0000ffff;
temp=templ;
NVWriteWord(a1+2,temp);
}

/*************************************************************************
/
/       Title:      GetNumTD(val, msg)
/
*************************************************************************/
GetNumTD(val, msg)
int *val;
char msg[];
{
char line[80];

PrintString(msg);
if (GetLineT(line))
        {
        if(line[0]==0)
                *val=0;
        else
                if (sscanf(line,"%d",val) != 1 ) *val=0;
        return(true);
        }
else
        {
        return(FALSE);
        }
}
```

```
/*************************************************************************
/
/      Title:     GetNumTL(val, msg)
/
*************************************************************************/
GetNumTL(val, msg)
int *val;
char msg[];
{
char line[80];

PrintString(msg);
if (GetLineT(line))
      {
      if(line[0]==0)
            *val=0;
      else
            if (sscanf(line,"%ld",val) != 1) *val=0;
      return(true);
      }
else
      {
      return(FALSE);
      }
}

/*************************************************************************
/
/      Title:     GetNumTX(val, msg)
/
*************************************************************************/
GetNumTX(val, msg)
int *val;
char msg[];
{
char line[80];

PrintString(msg);
if (GetLineT(line))
      {
      if(line[0]==0)
            *val=0;
      else
            if (sscanf(line,"%x",val) != 1) *val=0;
      return(true);
      }
else
      {
      return(FALSE);
      }
}
```

```
/*************************************************************************
/
/      Title:      ServeLine(line,size,count)
/
*************************************************************************/
ServeLine(line,size,count)
char line[];
int size;
int *count;

{
char mychar;
int mychari;
int eol;

mychar=upper(GetAChar());
mychari=mychar;
if ( (mychari==CARRIAGERETURN) || (mychari == LINEFEED)
            || (*count >= (size-1)))
       {
       line[*count]=0;
       eol=TRUE;
       }
else if ((mychari == BACKSPACE) || (mychari == DELETE) )
       {
       eol = FALSE;
       if (*count > 0)
             {
             (*count)--;
             PutAChar(BACKSPACE);
             PutAChar(SPACE);
             PutAChar(BACKSPACE);
             }
       else
             {
             PutAChar(BELL);
             }
       }
else
       {
       PutAChar(mychari);
       line[(*count)++]=mychar;
       eol = FALSE;
       }
return (eol);
}

/*************************************************************************
/
/      Title:      PrintString(ch)
/
*************************************************************************/
PrintString(ch)
char ch[];
{
int count;
int mychar;

count = 0;
while(ch[count]!=0)
       {
       mychar=ch[count++];
       PutAChar(mychar);
       if (mychar=='\n') PutAChar('\r');
       }
}
```

```
/*************************************************************************
/
/      Title:      extern TickInt()
/
*************************************************************************/
extern TickInt()
{
int index;
int IntState;

tick+=1;
SampleCount-=SampleDec;
SampleDec=0;
if (tick>=TickLimit)
      {
      tick=0;
      SampleCount+=1;
      csec+=1;
      if (csec>99)
            {
            csec=0;
            second+=1;
            if (second >59)
                  {
                  second=0;
                  minute+=1;
                  if (minute>59)
                        {
                        minute=0;
                        hour+=1;
                        }
                  }
            }
      }
}

/*************************************************************************
/
/      Title:      char upper(int c)
/
*************************************************************************/
char upper(int c)
{
if ( (c>='a') && (c<='z') )
      return(c-32);
else
      return(c);
}
/*************************************************************************
/
/      Title:      WriteValidMsg()
/
*************************************************************************/
WriteValidMsg()
{
char line[80];
char  c;
int count,nvrcount;

strcpy(line,ValidMessage);
count=0;
nvrcount=ValidNVRAdd;
while ( (c = line[count++]) != 0 )
      NVWrite(nvrcount++,c);
NVWrite(nvrcount,0);
}
```

```
/*************************************************************************
/
/      Title:      CheckNVRam()
/
*************************************************************************/
CheckNVRam()
{
char line[80];
char  c;
int count,nvrcount;

strcpy(line,ValidMessage);
count=0;
nvrcount=ValidNVRAdd;
NVFlag=FALSE;
while ( (c = line[count++]) != 0 )
      if ( ! (NVFlag = (NVRead(nvrcount++) == c) ) ) break;
return(NVFlag);
}

/*************************************************************************
/
/      Title:      ShowStatus()
/
*************************************************************************/
ShowStatus()
{
char line[80];

/* Header */
PrintString("\nStatus:\n\n");

/* Status of NV Ram */
ShowNVStatus();

/* Ticks per second */
ShowTickRate();

/* baud rate */
ShowBaudRate();

/* Ticks per sample period */
ShowTickLimit();

/* Show timeout */
ShowTimeout();

/* Show Start Character */
ShowStartChar();

/* Show Stop Character */
ShowStopChar();

/* Show the Reset Count */
ShowResetCount();

/* show all the channel selections */
ShowChannels();

CR();

/* Show the Password */
ShowPassword();

CR();
}
```

```
/*************************************************************************
/
/       Title:      ShowNVStatus()
/
*************************************************************************/
ShowNVStatus()
{
/* Status of NV Ram */
if (NVFlag)
      PrintString("    GOOD non volatile ram.\n");
else
      PrintString("     BAD non volatile ram.\n");
}

/*************************************************************************
/
/       Title:      ShowTickRate()
/
*************************************************************************/
ShowTickRate()
{
char line[80];

/* Ticks per second */
sprintf(line,"%8d Hardware interrupt frequency in HZ\n",TickRate);
PrintString(line);
}

/*************************************************************************
/
/       Title:      ShowBaudRate()
/
*************************************************************************/
ShowBaudRate()
{
char line[80];

/* baud rate */
sprintf(line,"%8d Hardware baud rate\n",baudRate);
PrintString(line);
}

/*************************************************************************
/
/       Title:      ShowTickLimit()
/
*************************************************************************/
ShowTickLimit()
{
char line[80];
float freq,freq1;
int temp1,temp2;

freq=TickRate;
freq=freq/TickLimit;
sprintf(line,"%8.2f Sample Frequency in HZ\n",freq);
PrintString(line);
}
```

```
/*************************************************************************
/
/       Title:      ShowTimeout()
/
*************************************************************************/
ShowTimeout()
{
char line[80];
long temp;

temp=TimeoutTickCount/TickRate;
temp=temp/60;
/* Ticks per sample period */
sprintf(line,"%8ld Minutes of inactivity before TIMEOUT\n",temp);
PrintString(line);
}

/*************************************************************************
/
/       Title:      ShowStartChar()
/
*************************************************************************/
ShowStartChar()
{
char line[80];

/* Start Character */
sprintf(line,"%8d START CHARACTER numeric value\n",StartChar);
PrintString(line);
}

/*************************************************************************
/
/       Title:      ShowStopChar()
/
*************************************************************************/
ShowStopChar()
{
char line[80];

/* Stop Character */
sprintf(line,"%8d STOP CHARACTER numeric value\n",StopChar);
PrintString(line);
}
```

```
/*************************************************************************
/
/      Title:      ShowResetCount()
/
*************************************************************************/
ShowResetCount()
{
char line[80];
long temp;

/* Reset count */
temp=resetCounter;
temp=temp*TickLimit;
temp=temp/60;
temp=temp/TickRate;
sprintf(line,"%81d Minutes until next resync\n",temp);
PrintString(line);
temp=resetCount;
temp=temp*TickLimit;
temp=temp/60;
temp=temp/TickRate;
sprintf(line,"%81d Minutes between resyncs\n",temp);
PrintString(line);
}

/*************************************************************************
/
/      Title:      ShowChannels()
/
*************************************************************************/
ShowChannels()
{
char line[80];
int i;

/* Channel count */
sprintf(line,"\n%8d Channels selected\n",channelCount);
PrintString(line);

PrintString("      channel  Sample Order\n");
sprintf(line,"%10d      sample(%d)\n",channelMap[channelCount-1],1);
PrintString(line);
for (i=0 ; i< channelCount-1; i++)
      {
      sprintf(line,"%10d      sample(%d)\n",channelMap[i],i+2);
      PrintString(line);
      }
}

/*************************************************************************
/
/      Title:      ShowPassword()
/
*************************************************************************/
ShowPassword()
{

/* Password */
PrintString("         Password is <");
PrintString(Password);
PrintString(">\n");

}
```

```
/*************************************************************************
/
/       Title:      CR()
/
*************************************************************************/
CR()
{
PrintString("\n");
}

/*************************************************************************
/
/       Title:      SaveToNVR()
/
*************************************************************************/
SaveToNVR()
{

PrintString("\nSaving parameters to NVRAM\n");
/* identifying message */
WriteValidMsg();

/*is NVRam good? */
CheckNVRam();

if (NVFlag)
      {
      /* save Password */
      WritePassword();

      NVWriteWord(TickRateAdd,TickRate);
      NVWriteWord(baudRateAdd,baudRate);
      NVWriteWord(TickLimitAdd,TickLimit);
      NVWriteLong(TTCountAdd,TimeoutTickCount);
      NVWriteLong(RSCountAdd,resetCount);
      NVWrite(StartCharAdd,StartChar);
      NVWrite(StopCharAdd,StopChar);
      NVWrite(ChanCountAdd,channelCount);
      NVWrite(FirstChanAdd,channelMap[0]);
      NVWrite(FirstChanAdd+1,channelMap[1]);
      NVWrite(FirstChanAdd+2,channelMap[2]);
      NVWrite(FirstChanAdd+3,channelMap[3]);
      NVWrite(FirstChanAdd+4,channelMap[4]);
      NVWrite(FirstChanAdd+5,channelMap[5]);
      NVWrite(FirstChanAdd+6,channelMap[6]);
      NVWrite(FirstChanAdd+7,channelMap[7]);

      PrintString("Write to NVRAM complete\n");
      }
else
      {
      PrintString("FAILURE to write to NVRAM\n");
      }

}
```

```
/*************************************************************************
/
/      Title:      RestoreFromNVR()
/
*************************************************************************/
RestoreFromNVR()
{/*is NVRam good? */
CheckNVRam();

if (NVFlag)
      {
      ReadPassword();
      TickRate=NVReadWord(TickRateAdd);
      baudRate=NVReadWord(baudRateAdd);
      TickLimit=NVReadWord(TickLimitAdd);
      TimeoutTickCount=NVReadLong(TTCountAdd);
      resetCount=NVReadLong(RSCountAdd);
      resetCounter=resetCount;
      StartChar=NVRead(StartCharAdd);
      StopChar=NVRead(StopCharAdd);
      channelCount=NVRead(ChanCountAdd);
      channelMap[0]=NVRead(FirstChanAdd);
      channelMap[1]=NVRead(FirstChanAdd+1);
      channelMap[2]=NVRead(FirstChanAdd+2);
      channelMap[3]=NVRead(FirstChanAdd+3);
      channelMap[4]=NVRead(FirstChanAdd+4);
      channelMap[5]=NVRead(FirstChanAdd+5);
      channelMap[6]=NVRead(FirstChanAdd+6);
      channelMap[7]=NVRead(FirstChanAdd+7);
      return(TRUE);
      }
else
      {
      return(FALSE);
      }
}

/*************************************************************************
/
/      Title:      SetDefault()
/
*************************************************************************/
SetDefault()

{
DefaultPassword();

TickRate=DefaultTickRate;
baudRate=DefaultBaudRate;
TickLimit=DefaultTickLimit;
TimeoutTickCount=DefaultTimeoutSeconds;
TimeoutTickCount*=DefaultTickRate;
resetCount=(DefaultResyncRate *(DefaultTickRate/DefaultTickLimit)*60);
resetCounter=resetCount;
StartChar=DefaultStartChar;
StopChar=DefaultStopChar;
channelCount=3;
channelMap[0]=4;
channelMap[1]=5;
channelMap[2]=3;
channelMap[3]=0;
channelMap[4]=0;
channelMap[5]=0;
channelMap[6]=0;
channelMap[7]=0;
}
```

```
/*************************************************************************
/
/       Title:      DefaultPassword()
/
*************************************************************************/
DefaultPassword()
{
strcpy(Password,StartPassWord);
pCount=0;
while (Password[pCount])
       {
       Password[pCount]=upper(Password[pCount]);
       tPassword[pCount++]=0;
       }
tPassword[pCount]=0;
}

/*************************************************************************
/
/       Title:      ReadPassword()
/
*************************************************************************/
ReadPassword()
{
int    count;

count=0;
while ((Password[count]=NVRead(PasswordAdd+count)) != 0) count++;
pCount=0;
while (Password[pCount])
       {
       Password[pCount]=upper(Password[pCount]);
       tPassword[pCount++]=0;
       }
tPassword[pCount]=0;
}

/*************************************************************************
/
/       Title:      WritePassword()
/
*************************************************************************/
WritePassword()
{
char line[80];
char  c;
int count,nvrcount;

count=0;
nvrcount=PasswordAdd;

while ( (c = Password[count++]) != 0 )
      NVWrite(nvrcount++,c);
NVWrite(nvrcount,0);
}
```

```
/*************************************************************************
/
/      Title:     DoSetClock()
/
*************************************************************************/
DoSetClock()
{
int temp;
int oldTickRate;

PrintString("Note: To change the actual tick rate a jumper in hardware must be
moved\n");
PrintString("      This parameter should be set to match the hardware
setting\n\n");
oldTickRate=TickRate;
ShowTickRate();
if (GetNumTD(&temp, "\nType the hardware clock rate (1200,600,300): "))
      {
      switch(temp)
            {
            case 1:
            case 12:
            case 120:
            case 1200:
                  TickRate=1200;
                  break;
            case 3:
            case 30:
            case 300:
                  TickRate=300;
                  break;
            case 6:
            case 60:
            case 600:
                  TickRate=600;
                  break;
            default:
                  break;
            }
/* adjust frequency, this should also adjust Resync,Inactivity and pcount */
      FreqAdjust(oldTickRate);
      CR();
      CR();
      ShowTickRate();
      }
}
```

```
/*************************************************************************
/
/      Title:     DoSetBaudRate()
/
*************************************************************************/
DoSetBaudRate()
{
int temp;

PrintString("Note: To change the actual baud rate a jumper in hardware must be
moved\n");
PrintString("      This parameter should be set to match the hardware
setting\n\n");
ShowBaudRate();
if (GetNumTD(&temp, "\nType the hardware baud rate (9600,4800,2400,1200,600,300):
"))
      {
      switch(temp)
            {
            case 1:
            case 12:
            case 120:
            case 1200:
                  baudRate=1200;
                  break;
            case 2:
            case 24:
            case 240:
            case 2400:
                  baudRate=2400;
                  break;
            case 3:
            case 30:
            case 300:
                  baudRate=300;
                  break;
            case 4:
            case 48:
            case 480:
            case 4800:
                  baudRate=4800;
                  break;
            case 6:
            case 60:
            case 600:
                  baudRate=600;
                  break;
            case 9:
            case 96:
            case 960:
            case 9600:
                  baudRate=9600;
                  break;
            default:
                  break;
            }
      ChannelCountCheck();
      CR();
      CR();
      ShowBaudRate();
      }
}
```

```
/*************************************************************************
/
/      Title:     DoSetFrequency()
/
*************************************************************************/
DoSetFrequency()
{
int temp;
int oldTickLimit;

oldTickLimit=TickLimit;
ShowTickLimit();
if (GetNumTD(&temp, "\nType the desired sample rate (HZ): "))
      {
      if (temp != 0)
            {
            if (temp<1) temp=1;
            TickLimit=TickRate/temp;
            if (TickLimit<1) TickLimit=1;
            FreqChangeCleanup(TickRate,oldTickLimit);
            CR();
            CR();
            ShowTickLimit();
            }
      }
}

/*************************************************************************
/
/      Title:     DoSetTimeout()
/
*************************************************************************/
DoSetTimeout()
{
int temp;
long templong;
long templong1;

ShowTimeout();
if (GetNumTD(&temp, "\nType number of minutes of inactivity before timeout (1-60) :
"))
      {
      if (temp != 0)
            {
            if (temp<1) temp=1;
            else if (temp>60) temp=60;
            templong=temp;
            templong1=TickRate;
            TimeoutTickCount=templong1*templong*60;
            CR();
            CR();
            ShowTimeout();
            }
      }
}
```

```
/*************************************************************************
/
/      Title:     DoSetStartChar()
/
*************************************************************************/
DoSetStartChar()
{
int temp;

ShowStartChar();
if (GetNumTD(&temp, "\nType numeric value of the start character (0-255): "))
      {
      if (temp<0) temp=0;
      else if (temp>255) temp=255;
      StartChar=temp;
      CR();
      CR();
      ShowStartChar();
      }
}

/*************************************************************************
/
/      Title:     DoSetStopChar()
/
*************************************************************************/
DoSetStopChar()
{
int temp;

ShowStopChar();
if (GetNumTD(&temp, "\nType numeric value of the start character (0-255): "))
      {
      if (temp<0) temp=0;
      else if (temp>255) temp=255;
      StopChar=temp;
      CR();
      CR();
      ShowStopChar();
      }
}
```

```
/*************************************************************************
/
/      Title:     DoSetPassword()
/
*************************************************************************/
DoSetPassword()
{
char line[80];

ShowPassword();
PrintString("\nType new password\n\n:");
if (GetLineT(line))
      {
      if (line[0] != 0)
            {
            strcpy(Password,line);
            pCount=0;
            while (Password[pCount])
                  {
                  Password[pCount]=upper(Password[pCount]);
                  tPassword[pCount++]=0;
                  }
            tPassword[pCount]=0;
            CR();
            CR();
            ShowPassword();
            }
      }
}

/*************************************************************************
/
/      Title:     DoSetResync()
/
*************************************************************************/
DoSetResync()
{
int temp;

ShowResetCount();
PrintString("\nType the length (in minutes) of the resync interval.\n");
PrintString("    (There are 1440 minutes in a day)");
if (GetNumTD(&temp, "\n\n( 10 - 10080 ) : "))
      {
      if (temp<10)  temp=10;
      else if (temp>10080) temp=10080;
      resetCount=temp;
      resetCount=(resetCount * 60); /* change to secs */
      resetCount=resetCount * TickRate;
      resetCount=resetCount / TickLimit;
      resetCounter=resetCount;
      CR();
      CR();
      ShowResetCount();
      }
}
```

```
/*************************************************************************
/
/      Title:     DoSetChannels()
/
*************************************************************************/
DoSetChannels()
{
char line[80];
int temp;
int chancnt,chan[8];
int getOut;
int maxCount;

maxCount=MaxChannelCount();
ShowChannels();


sprintf(line,"\n\n How many channels? (1-%d) :",maxCount);

if (GetNumTD(&chancnt,line))
      {
      if (chancnt != 0)
            {
            if (chancnt>maxCount) chancnt=maxCount;
            if (chancnt<1) chancnt=1;
            for (temp=0;temp<chancnt;temp++)
                  {
                  if(!GetNumTD(&chan[temp],"\nChannel : "))
                        break;
                  if (getOut = (chan[temp]==0))
                        break;
                  }
            if (GetTick()!=0)
                  {
                  if (!getOut)
                        {
                        channelCount=chancnt;
                        for (temp=0 ; temp <chancnt-1; temp++)
                              channelMap[temp]=chan[temp+1];
                        channelMap[chancnt-1]=chan[0];
                        }
                  }
            }
      if (GetTick()!=0)
            {
            CR();
            CR();
            ShowChannels();
            }
      }
}
```

```
/*************************************************************************
/
/      Title:     FreqAdjust(int oldRate)
/
*************************************************************************/
FreqAdjust(int oldRate)
{
int oldTickLimit;

oldTickLimit=TickLimit;

switch(oldRate)
      {
      case 1200:
            switch(TickRate)
                  {
                  case 1200:
                        break;
                  case 600:
                        TimeoutTickCount=TimeoutTickCount/2;
                        TickLimit=TickLimit/2;
                        if (TickLimit<1) TickLimit=1;
                        break;
                  case 300:
                        TimeoutTickCount=TimeoutTickCount/4;
                        TickLimit=TickLimit/4;
                        if (TickLimit<1) TickLimit=1;
                        break;
                  }
            break;
      case 600:
            switch(TickRate)
                  {
                  case 1200:
                        TimeoutTickCount=TimeoutTickCount*2;
                        TickLimit=TickLimit*2;
                        break;
                  case 600:
                        break;
                  case 300:
                        TimeoutTickCount=TimeoutTickCount/2;
                        TickLimit=TickLimit/2;
                        if (TickLimit<1) TickLimit=1;
                        break;
                  }
            break;
      case 300:
            switch(TickRate)
                  {
                  case 1200:
                        TimeoutTickCount=TimeoutTickCount*4;
                        TickLimit=TickLimit*4;
                        break;
                  case 600:
                        TimeoutTickCount=TimeoutTickCount*2;
                        TickLimit=TickLimit*2;
                        break;
                  case 300:
                        break;
                  }
            break;
      }


FreqChangeCleanup(oldRate,oldTickLimit);
}
```

```
/*************************************************************************
/
/       Title:      MaxChannelCount()
/
*************************************************************************/
MaxChannelCount()
{
long templ;
int temp;

templ=baudRate;
templ=templ/10;
templ=templ*TickLimit;
templ=templ/TickRate;
templ=templ - 3 ; /* a free char, a start char and a stop char, 3 total */
templ=templ/2;
temp=templ;
if (temp<1) temp=1;
if (temp>8) temp=8;
return(temp);
}

/*************************************************************************
/
/       Title:      FreqChangeCleanup(int oldRate, int oldLimit)
/
*************************************************************************/
FreqChangeCleanup(int oldRate, int oldLimit)
{
char line[80];

resetCount=resetCount / TickLimit;
resetCount=resetCount * oldLimit ;
resetCount=resetCount / oldRate  ;
resetCount=resetCount * TickRate ;
resetCounter=resetCount;

ChannelCountCheck();
}

/*************************************************************************
/
/       Title:      ChannelCountCheck()
/
*************************************************************************/
ChannelCountCheck()
{
int MCC;

MCC=MaxChannelCount();
if(channelCount>MCC)
        {
        channelMap[MCC-1]=channelMap[channelCount-1];
        channelCount=MCC;
        }
}
```

STKSIZ.C

```
#if sizeof(char *) == 4
int _STKSIZ = 8192/16;  /* (in paragraphs) (large data model) */
int _STKRED = 1024;           /* size of RED zone (in bytes) */
#else
int _STKSIZ = 4096/16;  /* (in paragraphs) */
int _STKRED = 2048;           /* size of RED zone (in bytes) */
#endif

int _HEAPSIZ = 4096/16; /* (in paragraphs) */
int _STKLOW = 0;        /* default is stack above heap (small only) */

/*
NOTE: The RED zone is used for stack saftey checking.  With stack above heap,
the heap will not be allowed to get within STKRED bytes of the current value
of the SP, if limit checking is enabled (see the cc +b option), SP isn't
allowed any closer than STKRED bytes to the top of the heap.  If stack below
heap and limit checking is enabled, SP isn't allowed any closer than STKRED
bytes to _Utop.  Minimum stack size = 2*_STKRED.  Minimum value for STKRED
should be about 256 bytes.  This allows some margin to issue DOS calls and
allow interrupt handlers to execute.  (Some people think that this should
be > 1k.) */
```

APPENDIX 2. DIGIREC

Dear Willie, May 7, 1991

Enclosed is the documentation for the "DIGIREC" project. I call the circuit that goes in the PC to receive RS232 data "DIGIREC". I have included documentation for three different aspects of the development: 1) hardware design, 2) firmware design and 3) software design (xdetect). I have included source code where I felt it was appropriate, hence the amount of documentation is rather large.

I am also including a floppy disk with the backup of the Xdetect directory, and a floppy disk with a backup of the hardware and firmware development.

Reese

Hardware Description of DIGIREC

Introduction

The DIGIREC circuit was envisioned as one of the necessary parts in the transformation of an existing analog seismic data acquisition system, to a digital seismic data acquisition system.

There exists a system which uses an IBM PC clone, and a standard A/D converter board, to monitor analog signals coming from seismic instrumentation, record and analyze seismic events. This system is limited in its performance by the resolution of the analog seismic data, which is in turn limited to the techniques used in available analog telemetry. Such limitations can be reduced, if data are digitized before telemetry, and transferred digitally. Platforms to digitize and transmit (using standard asynchronous serial formats) exist.

What is needed is a electronic circuit that can receive serial data, from up to 16 different digitizers (this means from as many as 16 serial ports), justify these data to a common time mark, and finally cause these data to be transferred into the IBM PC clone through direct memory access (DMA). DIGIREC was designed and built to meet this need.

DIGIREC is a microprocessor based controller. It is composed of the following functional blocks: 1) microprocessor, 2) memory, 3) input/ output address decoding, 4) RS232 level translators, 5) serial ports, 6) electrically alterable programmable read only memory (EAPROM), 7) reset circuit, and 8) the interface to the bus of an IBM PC clone.

The bus interface is divided into the following functional blocks: 1) the bus address, data and control interface, 2) an input register, 3) an output register, 4) interrupt channel selection and interrupt operation, 5) direct memory access channel selection, and direct memory access operation, and 6) the direct memory access output register.

The circuit for DIGIREC is provided in a 4 page schematic. Programmable logic devices are used to simplify the design. On the first page of the schematic, a programmable logic device is represented as a "black box". In subsequent, pages programmable logic devices are represented by the combinations of gates that are programmed into them. The equations for all programmable logic devices are included with this documentation.

Functional Hardware Blocks

Microprocessor (page 1 of the schematic)

The microprocessor chosen for DIGIREC is the V40 (U2), which is manufactured by NEC. It is a hybrid with the equivalent of an 8088 microprocessor integrated with extra memory control, timing, serial interface, and a super-set of assembly language instructions.

The microprocessor is driven by a 14.7456 MHz oscillator. This frequency is near the maximum operating frequency of the V40 device, and divides down to convenient frequencies for use with serial ports.

For greater accuracy, the oscillator (U4) is a temperature compensated crystal oscillator (TCXO). To minimize supply voltage effects on the oscillator, a separate supply (U1) is provided for the oscillator. To minimize loading effects on the oscillator, its output is buffered (U3) by an inverter. There is no hardware adjustment for the frequency of the TCXO, and the trimming input is held at a constant voltage by a resistor bridge (R1 and R2). Frequency adjustment is done in firmware.

An octal latch (U5) is used to catch the lower address lines from the multiplexed address/data lines of the microprocessor.

Memory (page 1 of the schematic)

DIGIREC has a 32K X 8 RAM (U11), and a 32K X 8 EPROM (U10). The RAM provides enough storage space for almost 2 seconds of serial data (sampled at 100 samples per second on 64 channels). The EPROM is large enough to allow speed optimization through in line coding.

I/O Address decoding (page 1 of the schematic)

A one of eight decoder (U15) is used to select between the 8 dual UARTs. A programmable logic device (U6) is used to provide the read and write lines necessary to use the input, output and DMA registers that interface to the IBM PC like bus. Note: this programmable logic device also provides a tri-state buffer that is used in interfacing to the EAPROM (see below).

RS232 level translators (page 2 of the schematic)

The sixteen serial data lines enter the circuit through a DB37 connect. Since RS232 logic levels are different than standard TTL or CMOS levels, these 16 lines must be converted using U30, U31, U32 and U33.

Note: the DB37 connector is also used to input a sample clock. Either an external clock is used (pin 36), or the internal clock is fed back into the board (jumper between pins 36 and 37).

Serial ports (page 2 of the schematic)

The sixteen serial input ports are handled by eight dual UARTs (U21, U22, U23, U24, U25, U26, U27 and U28). In this application, the serial outputs of the devices are not used. The microprocessor supplies most of the control logic, as well as the appropriate clock, derived from the 14.7456 MHz system clock. These dual UARTs have parallel input and output ports, as part of the package. The output port of U28 is used to provide a firmware generated 100 Hz clock. The output port of U26 is used to control DMA operation and the selection of DMA port. The inputs and the outputs of U24 are used to control the EAPROM. The input port of U23 is used to indicate when a DMA operation is complete. The output port of U22 is used to control and select the interrupt lines.

EAPROM (page 2 of the schematic)

Non volatile memory is used to record the actual frequency of the crystal oscillator. The microprocessor interfaces to the EAPROM (U29) using a single , bi-directional serial data line, and a clock line. These controls accomplished using the input and output ports of U24, and by using a tri-state gate from a programmable logic device (U6).

Reset circuit (page 3 of the schematic)

DIGIREC is reset by one of three things: 1) on power up, 2) if the PC bus reset line is asserted, and 3) the software in the PC performs a read operation at the I/O address that is 4 greater than DIGIREC's base address. The logic to accomplish this is shown at the bottom of page 3 of the schematics.

Bus Interface (page 3 and 4 of the schematic)

Address and data and control interface (page 3 of the schematic)

Address and read/write control come from the PC bus, and are gated in U12 to provide the appropriate control. Bi-directional buffers (U19 and U20) are used for normal data transfer. A tri-state gate (also U12) is used to indicate to the PC bus when 16 bit operations are in effect.

Input register (page 3 of the schematic)

U7 and U13 are each 8 bit registers, that allow data to be output by DIGIREC, and input by the PC bus. When this register is being read by the PC bus, data are asserted from these devices onto an internal bus; this bus is gated onto the PC bus by the bi-directional buffers (U19 and U20).

Output register (page 3 of the schematic)

U8 and U14 are each 8 bit registers, that allow data to be input to DIGIREC, and output by the PC bus. When this register is being written by the PC bus, PC bus data are asserted onto an internal bus by the bi-directional buffers (U19 and U20); these data are then loaded in U8 and U14.

Interrupt control and selection (page 4 of the schematic)

The hardware allows dynamic selection between 5 different interrupt lines on the PC bus (IRQ10, IRQ15, IRQ3, IRQ5 and IRQ7). This is accomplished by control coming from the output port of U22, and a programmable logic device (U35). The interrupt lines that are not in use, are left floating. The interrupt condition is asserted, when enabled, and a manual interrupt signal is invoked, or a terminal count condition is detected during DMA operation.

DMA control and selection (page 4 of the schematic)

The hardware allows dynamic selection between 3 different DMA channels on the PC bus (5, 6 and 7). This is accomplished by control coming from the output port of U26, and a programmable logic device (U16). The DMA lines that are not in use, are left floating. The DMA logic provides status information to the microprocessor (through the input port on U23), and when enabled invokes an interrupt.

DMA register (page 4 of the schematic)

The DMA register (U17 and U18) holds data that are about to be transferred to the PC bus. When the high order byte is loaded by the microprocessor, the next DMA operation is enabled.

Future

Future versions of DIGIREC may need to use a faster microprocessor. The algorithm used to collect and buffer the serial data is fairly straight forward, but it uses much of the processor time. Faster processors will be necessary if the functionality of the system is to be increased. (E.G. event detection, based on the incoming data is not possible with the current processor).

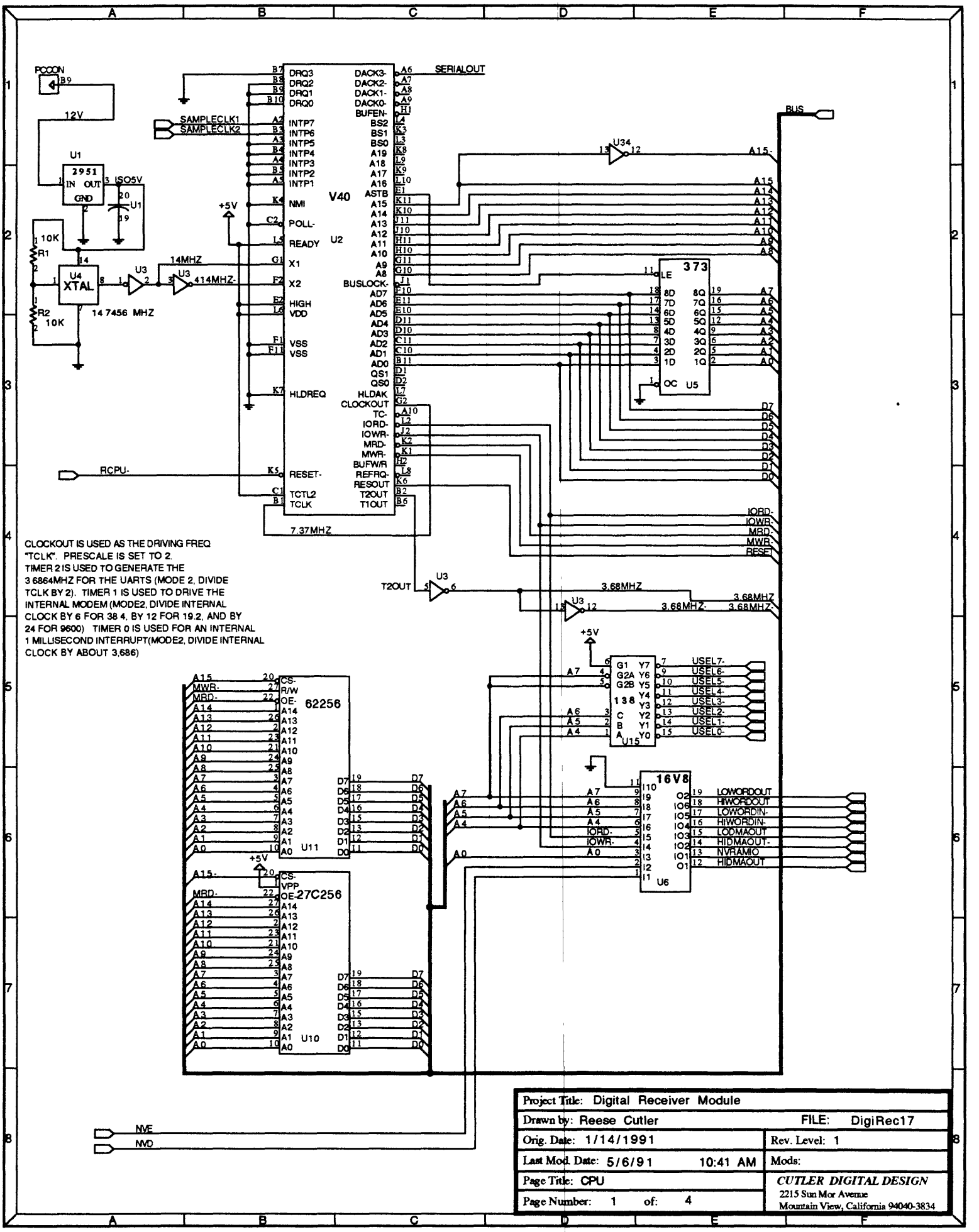
Currently, the PC bus address is set inside a programmable logic device. The only real hardware change to recommend at this time is the inclusion of a dip switch and appropriate logic to allow field modification of this address.

Conclusion

Happily, in the tests done so far, DIGIREC works well. Data are collected, buffered, and properly time justified. Furthermore, the firmware and software exist to make DIGIREC a successful part of a digital seismic data acquisition system.

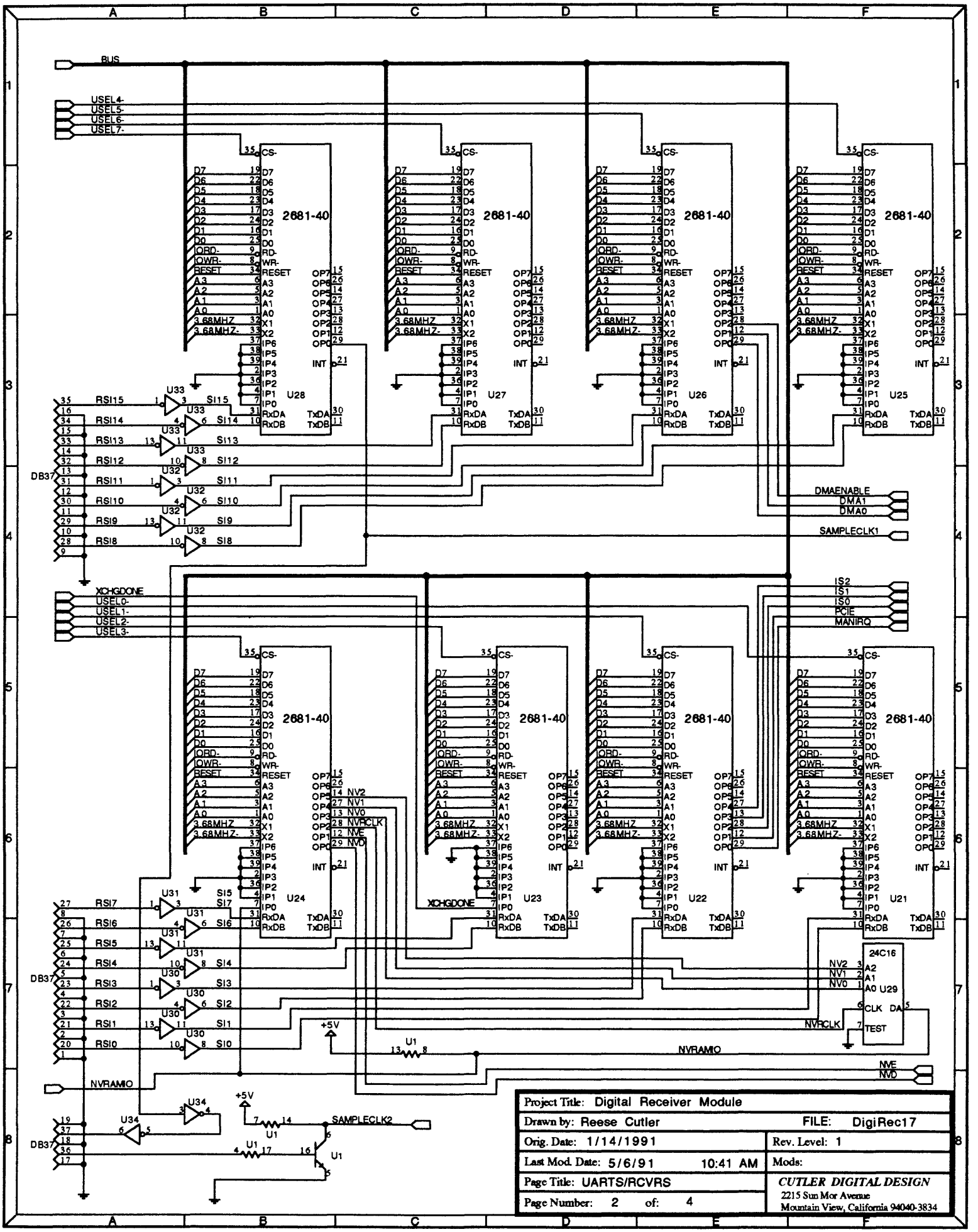
APPENDIX A

SCHEMATICS

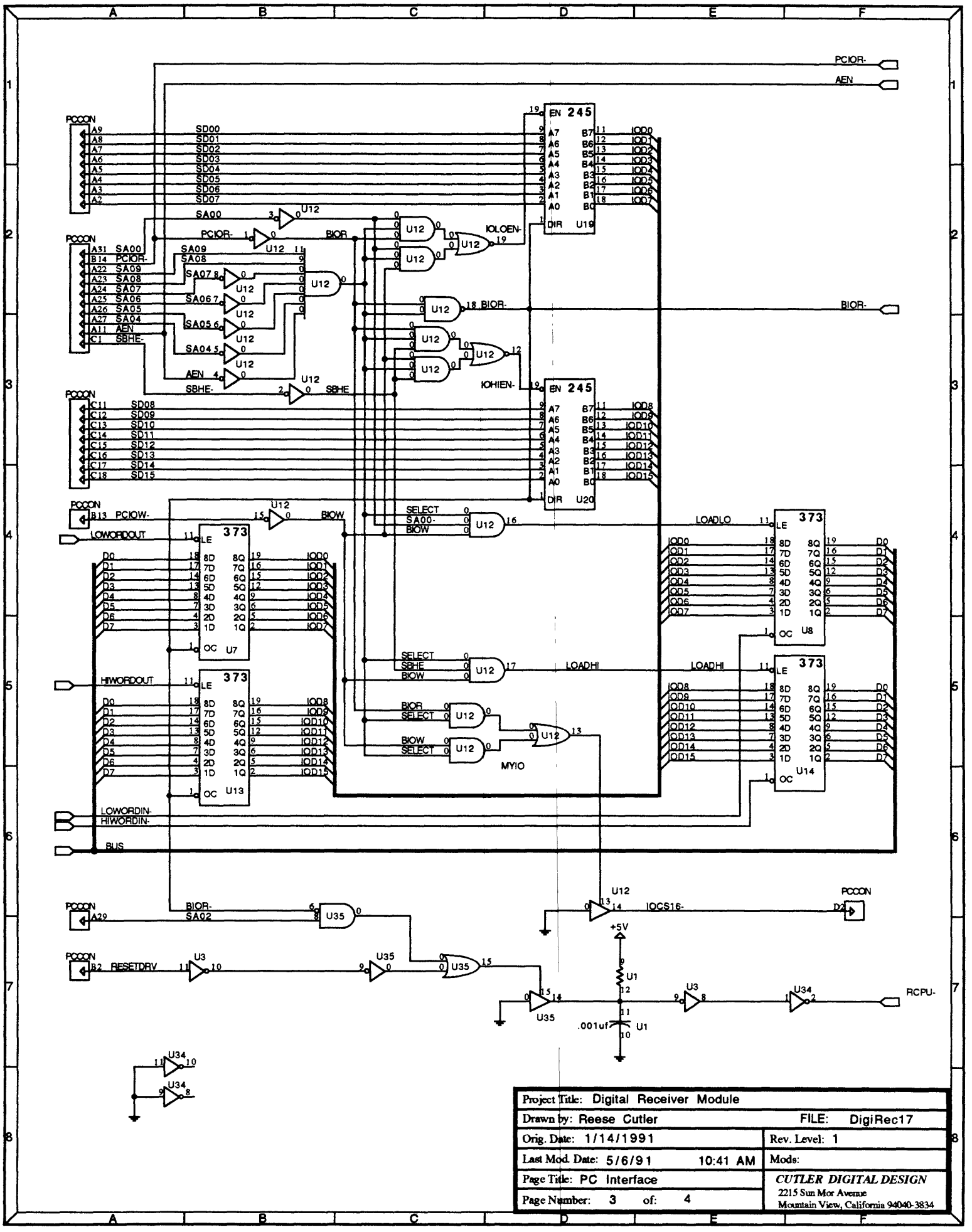


CLOCKOUT IS USED AS THE DRIVING FREQ "TCLK". PRESCALE IS SET TO 2. TIMER 2 IS USED TO GENERATE THE 3 6864MHZ FOR THE UARTS (MODE 2, DIVIDE TCLK BY 2). TIMER 1 IS USED TO DRIVE THE INTERNAL MODEM (MODE2, DIVIDE INTERNAL CLOCK BY 6 FOR 38 4, BY 12 FOR 19.2, AND BY 24 FOR 9600) TIMER 0 IS USED FOR AN INTERNAL 1 MILLISECOND INTERRUPT(MODE2, DIVIDE INTERNAL CLOCK BY ABOUT 3,686)

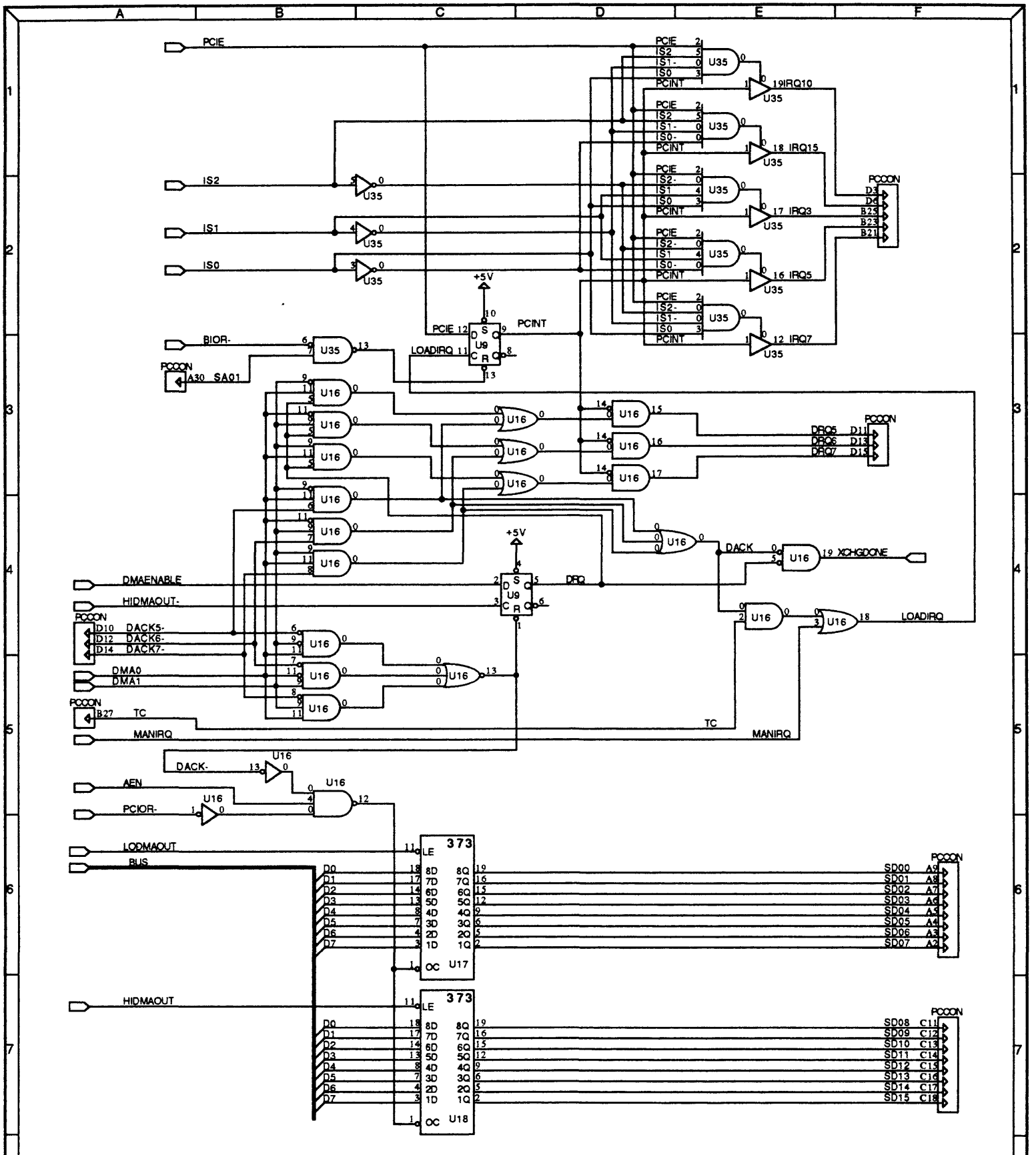
Project Title: Digital Receiver Module	
Drawn by: Reese Cutler	FILE: DigiRec17
Orig. Date: 1/14/1991	Rev. Level: 1
Last Mod. Date: 5/6/91 10:41 AM	Mods:
Page Title: CPU	*CUTLER DIGITAL DESIGN*
Page Number: 1 of: 4	2215 Sun Mor Avenue Mountain View, California 94040-3834



Project Title: Digital Receiver Module		
Drawn by: Reese Cutler	FILE: DigiRec17	
Orig. Date: 1/14/1991	Rev. Level: 1	
Last Mod. Date: 5/6/91 10:41 AM	Mods:	
Page Title: UARTS/RCVRS	*CUTLER DIGITAL DESIGN* 2215 Sun Mor Avenue Mountain View, California 94040-3834	
Page Number: 2 of: 4		



Project Title: Digital Receiver Module		
Drawn by: Reese Cutler	FILE: DigiRec17	
Orig. Date: 1/14/1991		Rev. Level: 1
Last Mod. Date: 5/6/91	10:41 AM	Mods:
Page Title: PC Interface		*CUTLER DIGITAL DESIGN*
Page Number: 3 of: 4		2215 Sun Mor Avenue Mountain View, California 94040-3834



Project Title: Digital Receiver Module		
Drawn by: Reese Cutler	FILE: DigiRec17	
Orig. Date: 1/14/1991	Rev. Level: 1	
Last Mod. Date: 5/6/91 10:41 AM	Mods:	
Page Title: DMA & PC INTERRUPTS	*CUTLER DIGITAL DESIGN*	
Page Number: 4 of: 4	2215 Sun Mor Avenue Mountain View, California 94040-3834	

APPENDIX B

PROGRAMMABLE LOGIC DEVICE EQUATIONS

```
title    REGISTER LOAD AND ENABLE
pattern  U6
revision A
author   RT Cutler
company  CUTLER DIGITAL DESIGN
date     02/28/91


chip U6 gal16v8

; pin 1     2     3  4     5     6  7  8  9  10
      NVRDI NVREN A0 /IOWR /IORD A4 A5 A6 A7 gnd

; pin 11 12       13      14        15       16        17        18        19
20
      nc HIDMAOUT NVRDATA /HIDMAOTE LODMAOUT /HIWORDIN /LOWORDIN HIWORDOUT
LOWORDOUT vcc

; @define s
;@define SELECT "/AEN * /SA04 * /SA05 * /SA06 * /SA07 * SA08 * SA09"


equations
NVRDATA=NVRDI
NVRDATA.trst=NVREN

HIDMAOUT = A7*A6*/A5*/A4*A0*IOWR

LODMAOUT = A7*A6*/A5*/A4*/A0*IOWR

HIDMAOTE = A7*A6*/A5*/A4*A0*IOWR

HIWORDIN = A7*A6*A5*A4*IORD

LOWORDIN = A7*A6*A5*/A4*IORD

HIWORDOUT = A7*A6*A5*A4*IOWR

LOWORDOUT = A7*A6*A5*/A4*IOWR


; end of U6
```

```
title    PC BUS ADDRESS DECODE, ETC
pattern  U12
revision A
author   RT Cutler
company  CUTLER DIGITAL DESIGN
date     02/04/91


chip U12 gal16v8

; pin 1      2     3    4    5    6    7    8    9    10
      /PCIOR /SBHE SA00 AEN SA04 SA05 SA06 SA07 SA08 gnd

; pin 11   12      13   14      15     16     17     18    19      20
      SA09 /IOHIEN MYIO /IOCS16 /PCIOW LOADLO LOADHI /BIOR /IOLOEN vcc

; @define s
@define SELECT "/AEN * /SA04 * /SA05 * /SA06 * /SA07 * SA08 * SA09"


equations
IOHIEN = SELECT * SBHE * PCIOR
      +SELECT * SBHE * PCIOW

MYIO =      SELECT * PCIOR
      +SELECT * PCIOW
/IOCS16 = gnd
 IOCS16.trst = MYIO

LOADLO = SELECT * /SA00 * PCIOW

LOADHI = SELECT * SBHE * PCIOW

BIOR = PCIOR * SELECT

IOLOEN = SELECT * PCIOR * /SA00
      +SELECT * PCIOW * /SA00

; end of U12
```

```
title    DMA GLUE
pattern  U16
revision A
author   RT Cutler
company  CUTLER DIGITAL DESIGN
date     02/04/91


chip U16 gal16v8

; pin 1      2  3      4   5   6      7      8      9    10
      /PCIOR TC MANIRQ AEN DRQ /DACK5 /DACK6 /DACK7 DMA1 gnd

; pin 11   12     13    14   15   16   17   18      19       20
      DMA0 /DMAEN /DACK INT  DRQ5 DRQ6 DRQ7 LOADIRQ XCHGDONE vcc



equations
LOADIRQ = DACK5 * /DMA1 * DMA0 * TC
        + DACK6 * DMA1 * /DMA0 * TC
        + DACK7 * DMA1 * DMA0 * TC
      + MANIRQ

DMAEN = DACK5 * /DMA1 * DMA0 * PCIOR * AEN
     + DACK6 * DMA1 * /DMA0 * PCIOR * AEN
     + DACK7 * DMA1 * DMA0 * PCIOR * AEN

XCHGDONE= /DRQ * /DACK5 * /DMA1 * DMA0
      + /DRQ * /DACK6 * DMA1 * /DMA0
      + /DRQ * /DACK7 * DMA1 * DMA0

DACK = DACK5 * /DMA1 * DMA0
     + DACK6 * DMA1 * /DMA0
     + DACK7 * DMA1 * DMA0

DRQ5 = DRQ   * /DMA1 * DMA0 * /INT
     + DACK5 * /DMA1 * DMA0 * /INT

DRQ6 = DRQ   * DMA1 * /DMA0 * /INT
     + DACK6 * DMA1 * /DMA0 * /INT

DRQ7 = DRQ   * DMA1 * DMA0 * /INT
     + DACK7 * DMA1 * DMA0 * /INT


; end of U16
```

```
title    IRQ LOGIC
pattern  U35
revision A
author   RT Cutler
company  CUTLER DIGITAL DESIGN
date     02/22/91


chip U16 gal16v8

; pin 1   2    3   4   5   6     7    8    9        10
      IRQ PCIE IS0 IS1 IS2 /BIOR SA01 SA02 /PCRESET gnd

; pin 11 12   13   14 15  16   17   18    19    20
      NC IRQ7 /RSI /R REN IRQ5 IRQ3 IRQ15 IRQ10  vcc



equations
IRQ10 = IRQ
IRQ10.trst = PCIE * IS2 * /IS1 * IS0

IRQ15 = IRQ
IRQ15.trst = PCIE * IS2 * /IS1 * /IS0

IRQ3  = IRQ
IRQ3.trst  = PCIE * /IS2 * IS1 * IS0

IRQ5  = IRQ
IRQ5.trst  = PCIE * /IS2 * IS1 * /IS0

IRQ7  = IRQ
IRQ7.trst  = PCIE * /IS2 * /IS1 * IS0

RSI = BIOR * SA01

REN = PCRESET
    + BIOR*SA02
R = vcc
R.trst=REN

; end of U35
```

Firmware Description of DIGIREC

Introduction

The DIGIREC circuit was envisioned as one of the necessary parts in the transformation of an existing analog seismic data acquisition system, to a digital seismic data acquisition system.

The DIGIREC controller is a small computer, and functions only with a program (firmware) in ROM. This program has been written using a combination of assembly language and the higher level language C.

It is this program that services the 16 UARTs, generates a 100 Hz clock, gathers a new data sample on the positive transition of the sample clock, and transfers buffers of stored data, through the mechanism of direct memory access (DMA) to the PC bus.

Development Tools

The Manx C compiler and assembler were used to develop the firmware. Assembly language source code is in the files: DIGIREC.EQU, LMACROS.H, DIGIREC.ASM, SERIN.MAC, SERIN0.ASM, SERIN8.ASM, and SERINX.ASM. The assembly language code is divided into this many parts to take advantage of the macro feature of the assembler. The macro feature does use up memory, and the assembler runs out of space, if the code is not separated and assembled in parts.

C-language source code is in the files: DIGIREC.C, DRCOM.H, STKSIZ.C.

Program Description

There are three separate parts to the DIGIREC firmware. The first part services each of the UARTs. The second part performs data acquisition on the rising edge of the sample clock. The third part is a loop which services any DMA operation in progress, and does any command processing, based on the state of an input (from the PC bus) register.

One millisecond Interrupt Operation

Once a millisecond, the normal program of the controller is interrupted. The interrupt program first prepares a counter to generate the next interrupt, one millisecond hence.

It services an internal sample clock line. One out of each ten milliseconds, it causes a clock line to go high. The other nine milliseconds, that clock line is held low.

It checks to see that it has not interrupted itself. When the 1 millisecond interrupt code is entered, while still in progress, it does some housekeeping, and leaves. If it has not interrupted itself, the code proceeds.

Finally, it passes through threaded code that services each of the 16 UARTs. Look at the macro in SERIN.MAC, to see how the code maintains a state variable for each UART. It determines, with each new character, where in the sequence of 8 message bytes, that character belongs, and into which of 3 8 byte buffers, it belongs. The threading of code is done to reduce the processor overhead.

The UARTs are not serviced under individual interrupt control, because of the inherent inefficiency of 8088 interrupts.

Note: by dynamically adjusting the counter that causes the 1 millisecond interrupt, the precision of the internal sample clock setting is much greater than the accuracy. The clock accuracy is +/- 1 part per million.

Sample Clock Interrupt Operation

The sample clock is provided through the external DB37 connector that also carries the 16 RS232 lines. Either an external clock line is generated, or a jumper on the connector is installed to use an internally generated sample clock. When the rising edge of the sample clock is detected, sample clock interrupt operation begins.

With each sample clock interrupt, data from the 16 serial ports is transferred into a large holding buffer. Data are kept in this buffer, until a DMA operation is requested, to transfer them to the PC bus. The buffer is large enough to hold the data of up to 2 seconds, though with proper operation there should be no more than 1/100th of a seconds data in the buffer at any one time.

The code is **not** designed to accommodate a sample clock rate greater than 100 Hz.

It will be possible to change the firmware, so that only the internal sample rate clock is used, independent of any jumpers, or clock signals at the DB37 connector.

With this version of the firmware, the occurrence of missing data (a sample interval during which no new data are completely received), cause the previous data to be transferred into the holding buffer. It will be possible to designate a fixed "missing data" value to be used instead.

Main Loop Operation

The main loop is written in C, and has two functions. The first is to repeatedly call the DMA service routine. The second is to check the input register for new data from the PC bus, and invoke the command processor when necessary.

Data from the serial ports are received and buffered as described above. These data must be transferred to the PC bus, upon request. The mechanism to perform this transfer is called Direct Memory Access (DMA). DIGIREC was designed to perform the DMA operation using a polling technique. There is a loop in the program that repeatedly checks to see if another DMA operation is pending, and if so, services it.

The second job of this loop is the recognition and processing of commands, from the PC bus. Commands are recognized as new values in the high order byte of the word-long input register.

Commands are invoked whenever a new word is written from the PC bus into the input register. The high order byte is the command, and the low order byte is either a parameter, or a sub-command.

In response to the command, so action is taken. When done, the command value is written to the output register. If there is a specific return value from the operation, it is returned in the low order byte of the output register.

Here is the current list of available commands:

NOP - This command is used to clear the command register when commands are repeated. Repeated commands will not be recognized, unless the command register is cleared first.

GENERAL - The general command is a group of commands that have no parameters. In place of a parameter, there is the sub-command. The commands are:

CAPTURE causes subsequent data samples to be written into the capture buffer, rather than the normal holding buffer. The number of times that this happens is specified by the least significant word in the "SHIFT_IN_BYTE" shift register (see below);
RESET causes DIGIREC to reset
DISABLE_INT disables the interrupt feature.
ENABLE_INT enables the interrupt feature.

SET_DMA enables, or disables the DMA feature. The parameter allows four values, specifying one of the 3 allowed DMA channels or the state of no DMA.

SHIFT_IN_BYTE shifts the parameter into the least significant byte of a 6 byte shift register. This register is used as a parameter for some instructions.

WRITE_NVR causes a write operation into the EAPROM. The byte shift register holds the address, and the parameter is the value to be written. The return value is the status of the write operation.

READ_NVR causes a read of the EAPROM, with the address taken from the byte shift register. The data byte is transferred back to the PC bus with another instruction, GET_NVR. The return value is the status of the read operation.

GET_NVR causes the return of the most recent result of a READ_NVR command.

SET_BAUD_RATE takes the baud rate value from the byte shift register, saves it in the data structure of the UART specified by the parameter, and resets the UART accordingly.

GET_BAUD_RATE uses the parameter to select a UART and returns the current setting for its baud rate.

SET_PARITY takes the parity value from the byte shift register, saves it in the data structure of the UART specified by the parameter, and resets the UART accordingly.

GET_PARITY uses the parameter to select a UART and returns the current setting for its parity.

SET_SYNC takes the synchronization character value from the byte shift register and saves it in the data structure for the UART specified by the parameter, and sets

SET_CLOCK takes the 3 words in the byte shift register as parameters to reset the internal 100 Hz sample clock.

DISABLE_UART disables the UART specified by the parameter.

GET_CAPTURE returns the data value in the capture buffer addressed by the parameter.

SET_POSITION reassigns the position of each data stream in the holding buffer. There are 16 serial channels, each with 4 data points, per sample period. Hence there are 64 possible data streams. Any of these streams can be assigned position 0 – 63 in the holding buffer. Any of these streams can be eliminated by assigning it to position 64. SET_POSITION causes the stream indicated by the parameter to be assigned the position indicated by the low order byte of the byte shift register. Parameter value 0 is the first data word in the first UART, parameter value 4 is the first data word in the second UART, etc.

GET_POSITION returns the position value for the stream specified by the parameter. Parameter value 0 is the first data word in the first UART, parameter value 4 is the first data word in the second UART, etc.

SET_INT uses the parameter to determine which interrupt line to use (lines 3, 5, 7, 10, and 15 are available). If none of the available interrupt lines is specified, then the no line is enabled.

Conclusion

The major concern, while developing this firmware was performance. It was unclear that all 16 serial ports would be serviced properly, with data demultiplexed without loss. Happily, this was all accomplished without any loss of flexibility. In fact, there is a great deal more flexibility than is currently being used. The seismic software does not currently take full advantage of the data stream positioning, synchronization character, and missing data features. It will also be possible to store more information in the EAPROM.

The current configuration is set to supply last data points, rather than a common missing data value for empty sample periods. There is no protection provided for holding buffer overflow, though in the observed behavior of the system, there has never been a need. Finally, the firmware is currently set to accept the sample clock from an external input only (this does allow the internal sample clock to be strapped at the DB37 connector to the external clock input).

Firmware Description of DIGIREC

Appendix - Digirec Firmware Listing

```
/*
      DIGIREC.C
      PC DIGITAL DATA RECEIVER Program
      DESCRIPTION: 80C88 "C" LANGUAGE MODULE FOR USE WITH DIGIREC.ASM,
                        SERIN0.ASM,SERIN8.ASM AND SERINX.ASM
      APRIL 30, 1991
      COPYRIGHT CUTLER DIGITAL DESIGN 1991
      Rev.1  04-30-91 Reese T. Cutler << Original program >>
*/

#define version 1

/*    This program has been written to operate in a custom
      NEC V40 computer controller has 16 serial input ports and
      which has a 16 bit interface to the standard pc bus.

      The program calls on various assembly language routines which can be
      found in the assembly language file DIGIREC.ASM and which are described
      below.  In addition, some "C" language routines in this program
      can be called by assembly interrupt routines in DIGIREC.ASM

      The DIGIREC.ASM code (with SERIN0.ASM, SERIN8.ASM and SERINX.ASM)
      has all the hooks necessary for the application
      to work in a standalone romable manner.

*/


/*
      The following assembly language routines are available:

      IOin(ioaddress)           returns a byte value in an integer
      int ioaddress;            corresponding to io byte register at
                          address "ioaddress"

      IOout(ioaddress,data)   loads byte value from "data" into io byte
      int   ioaddress;   register at address "ioaddress"
      int   data;

      INTON()                   enables interrupts

      INTOFF()            disables interrupts

      ColdReset()         restarts the receiver board

      DISABLEU0()         DISABLES UART (DISABLED) WITH CUR PARAMS
      DISABLEU1()
      DISABLEU2()
      DISABLEU3()
      DISABLEU4()
      DISABLEU5()
      DISABLEU6()
      DISABLEU7()
      DISABLEU8()
      DISABLEU9()
      DISABLEUA()
      DISABLEUB()
      DISABLEUC()
      DISABLEUD()
      DISABLEUE()
      DISABLEUF()
```

```
      ENABLEU0()        ENABLES UART WITH CUR PARAMS
      ENABLEU1()
      ENABLEU2()
      ENABLEU3()
      ENABLEU4()
      ENABLEU5()
      ENABLEU6()
      ENABLEU7()
      ENABLEU8()
      ENABLEU9()
      ENABLEUA()
      ENABLEUB()
      ENABLEUC()
      ENABLEUD()
      ENABLEUE()
      ENABLEUF()

      DODMA()                 SEND ANOTHER WORD BY DMA IF APPROPRIATE

      GetPoint()        GET THE POINTER TO BE BEGINNING OF THE DMA
                        BUFFER SPACE.

*/


/* Command interpreter commands */
/* The PC invokes commands by writing a new (and different) command word.

   Results are returned in the status word (same I/O address as
   the command word).

   The high order byte of the command word is called the command byte,
   the low order byte is called the parameter byte.  The parameter byte must
   be changed before, or at the same time as the command byte.

   Upon completion, the command byte is placed in the high order byte of
   the status word, and varying values in the low order byte of the status
   word as specified below.

   The PC knows that its command has been accepted, when it sees
   the current command byte appear in the high order byte of the status word.

   The "nop" command is provided to allow spacing between repeat usage
   of the same command.

   The "general" command pulls together commands that have no parameter
   and no return value.  The parameter byte is used as a sub command.

*/

#include    "drcom.h"
```

```
#define     WORDS_IN_BUFFER   0x3000


/***  LOGICAL DEFINITIONS  ***/
#define true      1
#define false     0
#define SETREG    15
#define RESETREG  14
#define INREG     13

#define port0     0
#define port1     16
#define     INT_PORT    16
      #define     LOAD_IRQ    1
      #define     PCIE        2
      #define     INT_SEL_BITS      0x1C
#define port2     32
#define port3     48
#define NVRAM_PORT 48
#define port4     64
#define port5     80
#define     DMA_PORT    80
      #define     DMA_BIT     4
#define port6     96
#define port7     112

#define     LoDmaAdd    192
#define HiDmaAdd  193

#define LoConAdd  224
#define HiConAdd  240


struct uartdata
      {
      int   state;
      int   start_char;
      int   ststate;          /* internal, do not use */
      int   empbuf[4];
      int   shb0[4];
      int   sbh1[4];
      int   sbh2[4];
      int   bufcount;         /* internal, do not use */
      int   empptr;
      int   curptr;
      int   prevptr;          /* internal, do not use */
      int   baud_rate;
      int   parity;
      int   first_on;
      int   second_on;
      int   third_on;
      int   fourth_on;
      int   first_offset;
      int   second_offset;
      int   third_offset;
      int   fourth_offset;
      };

extern struct uartdata STATE0[16];
```

```
extern int  PERIOD;
extern int  BASECNT0;
extern int  BASECNT1;
extern int  CNTINC;
extern      int   CAPFLAG;
extern      int   CAPDATA[128];
extern      int   BUFFER[WORDS_IN_BUFFER];
extern      int   SBCNT;
extern      int   SBIPTR;
extern      int   SBOPTR;
extern      int   SBBPTR;
extern      int   SBEPTR;
extern      int   SBINC;
extern      int   SBMXCNT;
extern      int   DMACNT;
extern      int   DMAMXCNT;
extern      int   DMAPTR;
extern      int   msflag;

main()   /*****************************************************************/
{
int   Command, Subcommand, Parameter, Result;
int   CommandMem, SubcommandMem;
unsigned int      byte1,byte2,byte3,byte4,byte5,byte6;
int   capturebuf[64];
int   i,j,k;
int off,chn,urt;
int   interrupt;
int   nvr,sb0,sb1,sb2,recog0,recog1,recog2;

/* MAKE THINKS LOOK LIKE A RESET JUST TOOK PLACE */
IOout(LoConAdd,RESET);
IOout(HiConAdd,GENERAL);

/* WAIT FOR NOP COMMAND */
while ((CommandMem=IOin(HiConAdd)) != NOP);

/*input the parameter byte */
Parameter = IOin(LoConAdd)&0xff;

/* reflect the new command and parameter at output */
IOout(LoConAdd,Parameter);
IOout(HiConAdd,CommandMem);

while (true)
      {
      if((Command=IOin(HiConAdd)) != CommandMem)
            {
            CommandMem=Command;
            Parameter=IOin(LoConAdd)&0xff;
            Subcommand=Parameter;
            Result=Parameter;
            switch (Command)
                  {
                  case NOP:
                        Result=0;
                        break;
```

```
case GENERAL:
      switch(Subcommand)
            {
            case CAPTURE:
                  i=(byte2<<8)+byte1;
                  CAPFLAG=i;
                  break;
            case RESET:
                  ColdReset();
                  break;
            case DISABLE_INT:
                  IOout(INT_PORT+RESETREG,PCIE);
                  IOout(INT_PORT+SETREG,LOAD_IRQ);
                  IOout(INT_PORT+RESETREG,LOAD_IRQ);
                  break;
            case ENABLE_INT:
                  IOout(INT_PORT+SETREG,PCIE);
                  break;
            default:
                  break;
            }
      break;
case SHIFT_IN_BYTE:
      byte6=byte5;
      byte5=byte4;
      byte4=byte3;
      byte3=byte2;
      byte2=byte1;
      byte1=Parameter;
      Result=Parameter;
      break;
case GET_NVR:
      Result=nvr;
      break;
case READ_NVR:
      Result=read_nvram( ( (byte2<<8) | byte1), &nvr);
      break;
case WRITE_NVR:
      Result=write_nvram( ( (byte2<<8) | byte1), Parameter);
      break;
```

```
case SET_BAUD_RATE:
      Parameter=Parameter & 15;
      switch(byte1)
            {
            case B300:
                  STATE0[Parameter].baud_rate=B300_VAL;
                  break;
            case B600:
                  STATE0[Parameter].baud_rate=B600_VAL;
                  break;
            case B1200:
                  STATE0[Parameter].baud_rate=B1200_VAL;
                  break;
            case B2400:
                  STATE0[Parameter].baud_rate=B2400_VAL;
                  break;
            case B4800:
                  STATE0[Parameter].baud_rate=B4800_VAL;
                  break;
            case B9600:
                  STATE0[Parameter].baud_rate=B9600_VAL;
                  break;
            default:
                  break;
            }
      Enable_Uart(Parameter);
      Result=Parameter;
      break;
case SET_SYNC:
      Parameter=Parameter & 15;
      STATE0[Parameter].start_char=byte1;
      Result=Parameter;
      break;
case SET_CLOCK:

      reset_clock((byte1+(byte2<<8)),
                  (byte3+(byte4<<8)),
                  (byte5+(byte6<<8)));
      Result=Parameter;
      break;
```

```
case GET_BAUD_RATE:
      Parameter=Parameter & 15;
      i=STATE0[Parameter].baud_rate;
      switch(i)
            {
            case B300_VAL:
                  Result=B300_VAL;
                  break;
            case B600_VAL:
                  Result=B600_VAL;
                  break;
            case B1200_VAL:
                  Result=B1200_VAL;
                  break;
            case B2400_VAL:
                  Result=B2400_VAL;
                  break;
            case B4800_VAL:
                  Result=B4800_VAL;
                  break;
            case B9600_VAL:
                  Result=B9600_VAL;
                  break;
            default:
                  Result=BBAD;
                  break;
            }
      break;
case SET_PARITY:
      Parameter=Parameter & 15;
      switch(byte1)
            {
            case NO_PARITY:
                  STATE0[Parameter].parity=NO_PARITY_VAL;
                  break;
            case B600:
                  STATE0[Parameter].parity=EVEN_PARITY_VAL;
                  break;
            case B1200:
                  STATE0[Parameter].parity=ODD_PARITY_VAL;
                  break;
            default:
                  break;
            }
      Enable_Uart(Parameter);
      Result=Parameter;
      break;
case GET_PARITY:
      Parameter=Parameter & 15;
      i=STATE0[Parameter].parity;
      switch(i)
            {
            case NO_PARITY_VAL:
                  Result=NO_PARITY;
                  break;
            case EVEN_PARITY_VAL:
                  Result=EVEN_PARITY;
                  break;
            case ODD_PARITY_VAL:
                  Result=ODD_PARITY;
                  break;
            default:
                  Result=BAD_PARITY;
                  break;
            }
      break;
```

```
case DISABLE_UART:
      Parameter=Parameter & 15;
      Disable_Uart(Parameter);
      break;
case GET_CAPTURE:
      while (CAPFLAG != 0) ;
      i=Parameter/2;
      if ((Parameter & 1)==1)
            Result=CAPDATA[i]>>8;
      else
            Result=CAPDATA[i];
      break;
case SET_POSITION:
      /* Parameter has the channel # (0-63)
         byte1 has the offset(0-63,any other then no data) */
      urt=(Parameter>>2)&0xf;
      chn=Parameter&3;
      if (byte1>63)
            off=128;
      else
            off=(byte1<<1);
      switch(chn)
            {
            case 0:
                  STATE0[urt].first_offset=off;
                  break;
            case 1:
                  STATE0[urt].second_offset=off;
                  break;
            case 2:
                  STATE0[urt].third_offset=off;
                  break;
            case 3:
                  STATE0[urt].fourth_offset=off;
                  break;
            }
      break;
case GET_POSITION:
      /* Parameter has the channel # (0-63) */
      urt=(Parameter>>2)&0xf;
      chn=Parameter&3;
      switch(chn)
            {
            case 0:
                  Result=STATE0[urt].first_offset;
                  break;
            case 1:
                  Result=STATE0[urt].second_offset;
                  break;
            case 2:
                  Result=STATE0[urt].third_offset;
                  break;
            case 3:
                  Result=STATE0[urt].fourth_offset;
                  break;
            }
      break;
```

```
case SET_INT:
      /* Parameter has the intnumber */
      switch(Parameter)
            {
            case INT_10:
                  IOout(INT_PORT+RESETREG,INT_SEL_BITS);
                  IOout(INT_PORT+SETREG,(INT_10_VAL*4));
                  Result=INT_10;
                  break;
            case INT_15:
                  IOout(INT_PORT+RESETREG,INT_SEL_BITS);
                  IOout(INT_PORT+SETREG,(INT_15_VAL*4));
                  Result=INT_15;
                  break;
            case INT_3:
                  IOout(INT_PORT+RESETREG,INT_SEL_BITS);
                  IOout(INT_PORT+SETREG,(INT_3_VAL*4));
                  Result=INT_3;
                  break;
            case INT_5:
                  IOout(INT_PORT+RESETREG,INT_SEL_BITS);
                  IOout(INT_PORT+SETREG,(INT_5_VAL*4));
                  Result=INT_5;
                  break;
            case INT_7:
                  IOout(INT_PORT+RESETREG,INT_SEL_BITS);
                  IOout(INT_PORT+SETREG,(INT_7_VAL*4));
                  Result=INT_7;
                  break;
            default:
                  IOout(INT_PORT+RESETREG,INT_SEL_BITS);
                  Result=NO_INT;
                  break;
            }
      break;
```

```
                case SET_DMA:
                        DMACNT=-1;  /* FREEZE DMA OPERATION */
                        Parameter=Parameter & 3; /*strip off hi bits*/
                        IOout(DMA_PORT+RESETREG,7);
                        IOout(HiDmaAdd,0);
                        switch(Parameter)
                                {
                                case ENABLE_DMA_CHANNEL_7:
                                        IOout(DMA_PORT+SETREG,7);
                                        Result=reset_buffer();
                                        break;
                                case ENABLE_DMA_CHANNEL_6:
                                        IOout(DMA_PORT+SETREG,6);
                                        Result=reset_buffer();
                                        break;
                                case ENABLE_DMA_CHANNEL_5:
                                        IOout(DMA_PORT+SETREG,5);
                                        Result=reset_buffer();
                                        break;
                                case DISABLE_DMA:
                                default:
                                        Result=get_count();
                                        break;
                                }
                        break;
                default:
                        break;
                }
            IOout(LoConAdd,Result);
            IOout(HiConAdd,Command);
            }

      DODMA();

      }
} /*******  End main() ***********************************************/

reset_clock(lowcount,hicount,increment)
int lowcount;
int hicount;
int increment;
{
INTOFF();

BASECNT0=lowcount;
BASECNT1=hicount;
CNTINC=increment;

INTON();
}
```

```
reset_buffer()
{
int buffer_count;
int i,j,len,bcplus,len2;
int ptr;

len=get_count();

buffer_count=((WORDS_IN_BUFFER-64)/len);
bcplus=buffer_count+1;
ptr = GetPoint() + (len * 2 * buffer_count);
len2= len *2;

INTOFF();

SBCNT=0;
DMAPTR=SBIPTR=SBOPTR=SBBPTR=GetPoint();
SBEPTR=ptr;
SBMXCNT=bcplus;
SBINC=len2;
DMAMXCNT=len;
DMACNT=0;

INTON();

return(len);

}

get_count()
{
int i,j,len;

len=0;
for (i=0;i<16;i++)   /* for each of 16 uarts */
      {
      j=STATE0[i].first_offset;
      if ((j!=128) && (len<j)) len=j;
      j=STATE0[i].second_offset;
      if ((j!=128) && (len<j)) len=j;
      j=STATE0[i].third_offset;
      if ((j!=128) && (len<j)) len=j;
      j=STATE0[i].fourth_offset;
      if ((j!=128) && (len<j)) len=j;
      }
len=(len>>1)+1;
if (len<1) len=1;
if (len>64) len=64;
return(len);
}
```

```
/*********************************************************************
*
*      Enable_Uart()
*
*********************************************************************/
Enable_Uart(p)
int p;
{
switch(p)
      {
      case 0:
            ENABLEU0();
            break;
      case 1:
            ENABLEU1();
            break;
      case 2:
            ENABLEU2();
            break;
      case 3:
            ENABLEU3();
            break;
      case 4:
            ENABLEU4();
            break;
      case 5:
            ENABLEU5();
            break;
      case 6:
            ENABLEU6();
            break;
      case 7:
            ENABLEU7();
            break;
      case 8:
            ENABLEU8();
            break;
      case 9:
            ENABLEU9();
            break;
      case 10:
            ENABLEUA();
            break;
      case 11:
            ENABLEUB();
            break;
      case 12:
            ENABLEUC();
            break;
      case 13:
            ENABLEUD();
            break;
      case 14:
            ENABLEUE();
            break;
      case 15:
            ENABLEUF();
            break;
      default:
            break;
      }
}
```

```
/*********************************************************************
*
*     Disable_Uart()
*
*********************************************************************/
Disable_Uart(p)
int p;
{
switch(p)
      {
      case 0:
            DISABLEU0();
            break;
      case 1:
            DISABLEU1();
            break;
      case 2:
            DISABLEU2();
            break;
      case 3:
            DISABLEU3();
            break;
      case 4:
            DISABLEU4();
            break;
      case 5:
            DISABLEU5();
            break;
      case 6:
            DISABLEU6();
            break;
      case 7:
            DISABLEU7();
            break;
      case 8:
            DISABLEU8();
            break;
      case 9:
            DISABLEU9();
            break;
      case 10:
            DISABLEUA();
            break;
      case 11:
            DISABLEUB();
            break;
      case 12:
            DISABLEUC();
            break;
      case 13:
            DISABLEUD();
            break;
      case 14:
            DISABLEUE();
            break;
      case 15:
            DISABLEUF();
            break;
      default:
            break;
      }
}
```

```
/*************************************************************************
 *
 *                read_nvram
 *
 *************************************************************************
 */
read_nvram(address,data)
int address;
int *data;
{
int   temp;
int   temp1;

IOout(port2+RESETREG,0xff);
IOout(port2+SETREG,address);

/* set up dummy write, to load address */

/* remove for lower density nvram
   if(nvr_send_start(0,address) == 0)
            {
            nvr_stop();
            return(4);
            }

   if(nvr_send_byte(address) != 0)
            {
            nvr_stop();
            return(5);
            }

*/

/* set up read */
   if(nvr_send_start(1,address) == 0)
            {
            nvr_stop();
            return(6);
            }
   *data=nvr_read_byte();

/* conclude interaction */
   nvr_stop();

IOout(port4+RESETREG,0xff);
IOout(port4+SETREG,*data);

/* and exit */
   return(0);
}
```

```
/*************************************************************************
 *
 *                write_nvram
 *
 *************************************************************************
 */
write_nvram(address,data)
int address;
{
int   temp;
int   temp1;

/* set up write, with full address */
   if(nvr_send_start(0,address) == 0)
            {
            nvr_stop();
            return(1);
            }

/* remove for lower density nvram
   if(nvr_send_byte(address) != 0)
            {
            nvr_stop();
            return(2);
            }
*/

/* now write data */
   if(nvr_send_byte(data) != 0)
            {
            nvr_stop();
            return(3);
            }

/* conclude interaction */
   nvr_stop();

/* and exit */
   return(0);
}
```

```
nvr_send_start(rw,address)
int rw;
int address;
{
int   ack,slab;
int mscount;

IOout(port0+SETREG,0xff);
IOout(port0+RESETREG,0xff);
mscount=0;

INTOFF();
/* make slave address byte */
/* adjust for lower density nvram
slab= 0xa0 | ( (address>>7) & 0xe );
*/
slab= (address << 1) & 126;

if (rw!=0) slab |= 1 ;

/*send successful read request */
do
      {
      /* issue start */
      nvr_start();

      /* send byte, and get acknowledge bit */
      ack=nvr_send_byte(slab);

      /* do time check for timeout */
      mscount=mscount+get_tick();
      }
while((mscount<20250) && (ack!=0));

INTON();
/* if timeout condition, the return 0 as error flag */
if (mscount == 20250) return(0);
return(1);

}

get_tick()
{
int temp;
/* this routine may miss ticks, but will not produce any
      false ticks */
if (msflag!=0)
      {
      msflag=0;
      return(1);
      }
return(0);
}
```

```
nvr_send_byte(address)
int address;
{
nvr_send_bit(address & 0x80);
nvr_send_bit(address & 0x40);
nvr_send_bit(address & 0x20);
nvr_send_bit(address & 0x10);
nvr_send_bit(address & 0x08);
nvr_send_bit(address & 0x04);
nvr_send_bit(address & 0x02);
nvr_send_bit(address & 0x01);

return(nvr_get_bit());
}

nvr_read_byte()
{
int i,temp;

temp=0;
for(i=0;i<8; i++)
      {
      temp = (temp<<1) | nvr_get_bit();
      }
nvr_send_bit(1);  /* skip acknowledge */
return(temp);
}

nvr_get_bit()
{
int temp;
IOout(NVRAM_PORT+RESETREG,2); /*free up data line */

IOout(NVRAM_PORT+SETREG,4);   /* turn on clock */

temp=IOin(NVRAM_PORT+INREG);  /* read data line */

IOout(NVRAM_PORT+RESETREG,4); /* turn off clock */

if ( (temp& 1)!=0) return(1); else return(0);
}

nvr_send_bit(bit)
int bit;
{
int temp;

if(bit==0)
      IOout(NVRAM_PORT+SETREG,2);
else
      IOout(NVRAM_PORT+RESETREG,2);

IOout(NVRAM_PORT+SETREG,4);   /* turn on clock */

/* the following line is just for a pause, may not be necessary */
temp=IOin(NVRAM_PORT+INREG);  /* read data line */

IOout(NVRAM_PORT+RESETREG,4); /* turn off clock */
}
```

```
nvr_start()
{
IOout(NVRAM_PORT+RESETREG,1); /*be sure datum to gate is low */

IOout(NVRAM_PORT+RESETREG,4); /* be sure clock is low */

IOout(NVRAM_PORT+RESETREG,2); /* data high, by disabling tristate driver */

IOout(NVRAM_PORT+SETREG,4);   /* clock high */

IOout(NVRAM_PORT+SETREG,2);   /* data lo */

IOout(NVRAM_PORT+RESETREG,4); /* clock lo */

}

nvr_stop()
{
IOout(NVRAM_PORT+RESETREG,1); /*be sure datum to gate is low */

IOout(NVRAM_PORT+RESETREG,4); /* be sure clock is low */

IOout(NVRAM_PORT+SETREG,2);   /* be sure datum is low (enable tristate) */

IOout(NVRAM_PORT+SETREG,4);   /* clock high */

IOout(NVRAM_PORT+RESETREG,2); /* datum hi */

IOout(NVRAM_PORT+RESETREG,4); /* clock lo */

}
```

```
/*
      DRCOM.H
      Defines for the different commands issued by a PC like
      computer through the io port (usually 300) to control the
      digital data receiver

      APRIL 30, 1991
      COPYRIGHT CUTLER DIGITAL DESIGN 1991
*/



/* Command interpreter commands */
/* The PC invokes commands by writing a new (and different) command word.

   Results are returned in the status word (same I/O address as
   the command word).

   The high order byte of the command word is called the command byte,
   the low order byte is called the parameter byte.  The parameter byte must
   be changed before, or at the same time as the command byte.

   Upon completion, the command byte is placed in the high order byte of
   the status word, and varying values in the low order byte of the status
   word as specified below.

   The PC knows that its command has been accepted, when it sees
   the current command byte appear in the high order byte of the status word.

   The "nop" command is provided to allow spacing between repeat usage
   of the same command.

   The "general" command pulls together commands that have no parameter
   and no return value.  The parameter byte is used as a sub command.

*/

#define     NOP          0            /* dummy command, does nothing,
                                     return = 0 */
#define     GENERAL            1            /* collection of commands that have
                                        no parameters,
                                     return = subcommand */
      /* general sub commands */
      #define     RESET        0
      #define     CAPTURE            8     /* CAPTURE A DATA SAMPLE */
                        /* 16 bit count in byte1(lo) and byte 2(hi) */
      #define     DISABLE_INT 9
      #define     ENABLE_INT  10

#define     SET_DMA            2
      #define     ENABLE_DMA_CHANNEL_7    3
      #define     ENABLE_DMA_CHANNEL_6    2
      #define     ENABLE_DMA_CHANNEL_5    1
      #define     DISABLE_DMA                     0
                                    /* return=NUMBER OF DATA CHANNELS (1-64)*/

#define     SHIFT_IN_BYTE      3            /* auxilliary parameter shift regist
                                     has 4 bytes, this command shifts
                                     in a new low order byte.
                                     return=parameter byte. */
```

```
#define     WRITE_NVR   4           /* write the byte in Parameter
                                    into the nvram location indicated
                                    by the address in byte shift
                                    register, the return value for this
                                    operation is 0 for unsuccessful,
                                    non-zero, for successful */

#define     READ_NVR    5           /* read the byte
                                    from the nvram location indicated
                                    by the address in byte shift
                                    register, and store it.
                                    The return value for this
                                    operation is 0 for unsuccessful,
                                    non-zero, for successful.
                                    Data that has been read is available
                                    using the GET_NVR command */

#define     GET_NVR         6           /* return whatever is in nvr storage */

#define     SET_BAUD_RATE   9           /* set the baud rate of the uart
                                    specified in parameter to the
                                    rate specified in the auxiliary
                                    shift register
                                    return=parameter & 00001111B */

#define     GET_BAUD_RATE   10          /* get the baud rate of the uart
                                    specified in parameter.
                                    return=baud rate */

      #define     B300  0
      #define     B600  1
      #define     B1200 2
      #define     B2400 3
      #define     B4800 4
      #define     B9600 5
      #define     BBAD  6

      #define     B300_VAL    0X44
      #define     B600_VAL    0X55
      #define     B1200_VAL   0X66
      #define     B2400_VAL   0X88
      #define     B4800_VAL   0X99
      #define     B9600_VAL   0XBB


#define     SET_PARITY  11    /* channel in parameter, parity value in
                              byte1,   return= parameter */

#define     GET_PARITY  12    /* channel in parameter, return = parity */

      #define     NO_PARITY   0
      #define     EVEN_PARITY 1
      #define     ODD_PARITY  2
      #define     BAD_PARITY  3

      #define     NO_PARITY_VAL     19
      #define     EVEN_PARITY_VAL   3
      #define     ODD_PARITY_VAL    7


#define     DISABLE_UART      13
#define     SET_POSITION      14
#define     GET_POSITION      15
```

```
#define     SET_INT           16          /* parameter holds int choice
                                  returns parameter */


      #define     INT_10            0
      #define     INT_15            1
      #define     INT_3       2
      #define     INT_5       3
      #define     INT_7       4
      #define     NO_INT            5

      #define     INT_10_VAL  5
      #define     INT_15_VAL  4
      #define     INT_3_VAL   3
      #define     INT_5_VAL   2
      #define     INT_7_VAL   1


#define     SET_SYNC    17
#define     SET_CLOCK   20
#define     GET_CAPTURE 100
```

```
; DIGIREC.ASM                          (R. CUTLER,  04/30/91)
; Copyright (C) 1991 by Cutler Digital Design

   ;*************************************************************
   ;***                        INCLUDES
   ;*************************************************************

      include lmacros.h
      INCLUDE     DIGIREC.EQU


   ;*************************************************************
   ;***                        MACROS                         ***
   ;*************************************************************

   ;***  usage OUTPT  OUTPUT-PORT , VALUE

OUTPT MACRO  ARG1,ARG2  ;***  Allows output of reg., mem. or constants
      IF    ARG1 GT 255       ;***  Used by lots of routines
      IFDIF <"DX">,<ARG1>
      MOV     DX,ARG1
      ENDIF
      IFDIF <"AL">,<ARG2>
      MOV     AL,ARG2
      ENDIF
      OUT     DX,AL
      ELSE
      IFDIF <"AL">,<ARG2>
      MOV   AL,ARG2
      ENDIF
      OUT   ARG1,AL
      ENDIF
        ENDM


   ;***  usage OUTPTL  OUTPUT-PORT(>255) , VALUE

OUTPTL      MACRO  ARG1,ARG2  ;***  Allows output of reg., mem. or constants
      MOV     DX,ARG1
      MOV     AL,ARG2
      OUT     DX,AL
        ENDM


   ;***  usage INPUT  INPUT-PORT

INPUT MACRO  ARG1       ;***  Allows input to reg., mem. or constants
      IF    ARG1 GT 255       ;***  Used by lots of routines
      IFDIF <"DX">,<ARG1>
        MOV     DX,ARG1
      ENDIF
        IN      AL,DX
      ELSE
      IN    AL,ARG1
      ENDIF
        ENDM


INTON MACRO          ;***  Enable interrupts
      STI               ;***  (Used by InitIO & GITCH3)
        ENDM


INTOFF      MACRO       ;***  Disable interrupts
      CLI               ;***  (InitIO, DOEDGE & GITCH1)
        ENDM
```

```
;*********************************
;
; DATA STRUCTURE USED INT THE SERIN ROUTINES
;   CALLED ONCE FOR EACH SERIAL PORT
;   THESE VARIABLES ARE DECLARED IN SERIN0.ASM AND SERIN8.ASM
;   BY CALLING THE MACRO DEFINED IN SERIN.MAC
;
;*********************************
UDAT  MACRO UN
DATASEG     SEGMENT     WORD PUBLIC 'data'
      EXTRN STATE&UN&_:WORD          ;HOLDS THE ADDRESS OF THE PORTION OF
                                     ; CODE TO EXECUTE ONCE AN INPUT
                                     ; CHARACTER IS FOUND FOR THIS CHANNEL
                                     ; DURING THE MILLISECOND INTERRUPT
      EXTRN STCHAR&UN&_:BYTE         ;SYNC CHARACTER, USED TO RECOGNIZE WHEN
                                     ; A NEW DATA FRAME (8 CHARS, 7 WITH DATA)
                                     ; IS ABOUT TO ARRIVE
      EXTRN STSTATE&UN&_:WORD        ;MEMORY OF WHAT THE STARTING STATE IS
                                     ; CURRENTLY.  THIS IS NECESSARY, SINCE
                                     ; STATE INFORMATION INCLUDES THE CHOICE
                                     ; OF WHICH BUFFER DATA ARE SUPPOSED TO
                                     ; GO TO.  WHEN AN RECEIVER ERROR IS DETECTED
                                     ; THE CODE WILL RESET TO THE BEGINNING
                                     ; OF THE BUFFER
      EXTRN EMPBUF&UN&_:WORD         ;MISSING DATA BUFFER
      EXTRN SHB0&UN&_:WORD                ;DATA BUFFER 0
      EXTRN SHB1&UN&_:WORD                ;DATA BUFFER 1
      EXTRN SHB2&UN&_:WORD                ;DATA BUFFER 2
      EXTRN IBFCNT&UN&_:WORD         ;INPUT BUFFER COUNT, EITHER 0,2,OR 4
                                     ; IS USED TO DECIDE WHICH INPUT BUFFER
                                     ; TO USE, WHEN THE SAMPLE INTERVAL IS
                                     ; OVER, AND DATA ARE TRANSFERRED FROM
                                     ; INPUT BUFFERS, TO THE LARGE ARRAY
                                     ; OF HOLDING BUFFERS
      EXTRN EMPPTR&UN&_:WORD         ;POINTER TO MISSING DATA
      EXTRN CURPTR&UN&_:WORD         ;POINTER TO MOST RECENTLY TRANSFERRED
                                     ; FILLED BUFFER
      EXTRN PREVPTR&UN&_:WORD        ;POINTER TO OLDEST RESIDENT INPUT
                                     ; BUFFER
      EXTRN BRATE&UN&_:BYTE               ;BAUD RATE INDICATOR FOR UART
      EXTRN PARITY&UN&_:BYTE         ;PARITY INDICATOR FOR UART
      EXTRN FRSTCH&UN&_:WORD         ;FLAG TO INDICATE WHETHER THE FIRST
                                     ; DATUM IN THE BUFFER IS ENABLED
                                     ; NOT CURRENTLY USED
      EXTRN SCNDCH&UN&_:WORD         ;FLAG TO INDICATE WHETHER THE SECOND
                                     ; DATUM IN THE BUFFER IS ENABLED
                                     ; NOT CURRENTLY USED
      EXTRN THRDCH&UN&_:WORD         ;FLAG TO INDICATE WHETHER THE THIRD
                                     ; DATUM IN THE BUFFER IS ENABLED
                                     ; NOT CURRENTLY USED
      EXTRN FRTHCH&UN&_:WORD         ;FLAG TO INDICATE WHETHER THE FOURTH
                                     ; DATUM IN THE BUFFER IS ENABLED
                                     ; NOT CURRENTLY USED
      EXTRN FRSTOFF&UN&_:WORD        ;OFFSET OF THE FIRST DATUM IN
                                     ; THE HOLDING BUFFERS.  SINCE THERE
                                     ; ARE AT MOST 64 DATA POINTS, A VALUE
                                     ; OF 64*2 CAUSES THIS DATA POINT TO
                                     ; BE STORED BEYOND THE HOLDING BUFFER
                                     ; AND EFFECTIVELY DISABLES IT.
      EXTRN SCNDOFF&UN&_:WORD        ;OFFSET OF THE SECOND DATUM IN
                                     ; THE HOLDING BUFFERS.  SINCE THERE
                                     ; ARE AT MOST 64 DATA POINTS, A VALUE
                                     ; OF 64*2 CAUSES THIS DATA POINT TO
                                     ; BE STORED BEYOND THE HOLDING BUFFER
                                     ; AND EFFECTIVELY DISABLES IT.
```

```
      EXTRN THRDOFF&UN&_:WORD ;OFFSET OF THE THIRD DATUM IN
                              ; THE HOLDING BUFFERS.  SINCE THERE
                              ; ARE AT MOST 64 DATA POINTS, A VALUE
                              ; OF 64*2 CAUSES THIS DATA POINT TO
                              ; BE STORED BEYOND THE HOLDING BUFFER
                              ; AND EFFECTIVELY DISABLES IT.
      EXTRN FRTHOFF&UN&_:WORD ;OFFSET OF THE FOURTH DATUM IN
                              ; THE HOLDING BUFFERS.  SINCE THERE
                              ; ARE AT MOST 64 DATA POINTS, A VALUE
                              ; OF 64*2 CAUSES THIS DATA POINT TO
                              ; BE STORED BEYOND THE HOLDING BUFFER
                              ; AND EFFECTIVELY DISABLES IT.
DATASEG     ENDS
      ENDM

;*********************************
;
; INIT DATA STRUCTURE FOR UART &UNUM
;
;*********************************
INITU MACRO UNUM
      EXTRN SI00&UNUM:NEAR
      MOV   STATE&UNUM&_,OFFSET SI00&UNUM ;ADDRESS OF CODE TO LOOK FOR
                                  ; SYNC CHARACTER DURING
                                  ; MSECOND INTERRUPT PROCESSING

      MOV   STSTATE&UNUM&_,OFFSET SI00&UNUM     ;SET RESET ADDRESS TO THE SAME
      MOV   STCHAR&UNUM&_,STARTBYTE       ;STANDARD START BYTE

      MOV   BRATE&UNUM&_,R9600            ;SET STANDARD BAUD RATE

      MOV   PARITY&UNUM&_,ODDP            ;SET TO ODD PARITY
      MOV   FRSTOFF&UNUM&_,(0&UNUM&H * 8) ;INIT OFFSET FIRST CHANNEL
      MOV   SCNDOFF&UNUM&_,(0&UNUM&H * 8)+2     ;INIT OFFSET SECOND CHANNEL
      MOV   THRDOFF&UNUM&_,(0&UNUM&H * 8)+4     ;INIT OFFSET THIRD CHANNEL
      MOV   FRTHOFF&UNUM&_,(0&UNUM&H * 8)+6     ;INIT OFFSET FOURTH CHANNEL
      MOV   EMPBUF&UNUM&_,08000H          ;INIT MISSING DATA BUFFER
      MOV   EMPBUF&UNUM&_+2,08000H
      MOV   EMPBUF&UNUM&_+4,08000H
      MOV   EMPBUF&UNUM&_+6,08000H
      MOV   IBFCNT&UNUM&_,0               ;INIT INPUT BUFFER COUNT
      MOV   EMPPTR&UNUM&_,OFFSET EMPBUF&UNUM&_  ; INIT THE POINTER TO
                                      ;  MISSING DATA BUFFER

      ENDM
```

```
;************************************
;
; ENABLE UART &UNUM1 (SEE DATA SHEETS FOR 2681 DUAL UART FOR DETAILS)
;
;************************************
ENABLEU     MACRO UNUM1
      PUBLIC      ENABLEU&UNUM1&_
ENABLEU&UNUM1&_:  PUSH  AX
      MOV   AL,00111010B
      OUT   UART&UNUM1+UCOMM,AL     ;RESET TRANSMIT & DISABLE TX AND RX
      MOV   AL,00101010B
      OUT   UART&UNUM1+UCOMM,AL     ;RESET RECVR & DISABLE TX AND RX
      MOV   AL,00011010B
      OUT   UART&UNUM1+UCOMM,AL     ;RESET MR PTR &DISABLE TX AND RX
      MOV   AL,PARITY&UNUM1&_
      OUT   UART&UNUM1+UMODE,AL     ;SET PARITY
      MOV   AL,00000111B
      OUT   UART&UNUM1+UMODE,AL     ;SET MODE AND STOP BITS
      MOV   AL,BRATE&UNUM1&_
      OUT   UART&UNUM1+CSR,AL ;SET BAUD RATE (ACR MUST BE SET ELSEWHERE)
      MOV   AL,1
      OUT   UART&UNUM1+UCOMM,AL     ;TURN UART ON
      POP   AX
      RET
      ENDM

;************************************
;
; RESET DUART &DNUM (SEE DATA SHEETS FOR 2681 DUAL UART FOR DETAILS)
;
;************************************
INITD MACRO DNUM
      MOV   AL,0
      OUT   DUART&DNUM+OPCR,AL      ;PURE OUTPUT CONTROL
      OUT   DUART&DNUM+ACR,AL ;DEFAULT
      OUT   DUART&DNUM+IMR,AL ;DEFAULT
      MOV   AL,0FFH
      OUT   DUART&DNUM+RESET,AL     ;CLEAR
      ENDM

;************************************
;
;     DISABLE RCVER OF UART&UNUM3 (SEE DATA SHEETS FOR 2681 DUAL UART)
;
;************************************
DISABLEU MACRO    UNUM3
      PUBLIC      DISABLEU&UNUM3&_
DISABLEU&UNUM3&_: MOV   AL,2
      OUT   UART&UNUM3+UCOMM,AL
      RET
      ENDM
```

```
;*************************************
;
;     GET DATA FROM INPUT BUFFERS INTO
;           CAPTURE  BUFFER
;
;*************************************
GETDATA     MACRO U
      LOCAL DOWN
      MOV   SI,IBFCNT&U&_      ; GET INPUT BUFFER COUNT, WHICH IS IN FACT AN
                               ;  OFFSET INTO A POINTER TABLE WHOSE BASE IS
                               ;  EMPPTR&U&_
      MOV   SI,EMPPTR&U&_[SI];GET THE ACTUAL DATA
                              ; POINTER IN SI
; THE FOLLOWING LINE IS INCLUDED IF NO MISSING DATA IS WANTED
      MOV   EMPPTR&U&_,SI     ;SAVE THIS POINTER FOR FUTURE MISSING DATA

; GET THE FIRST DATUM, ASSUME DX IS ALREADY SET TO THE DESTINATION
      MOV   DI,DX
      ADD   DI,FRSTOFF&U&_
      MOVS  CAPDATA_,EMPBUF&U&_

; GET THE SECOND DATUM, ASSUME DX IS ALREADY SET TO THE DESTINATION
      MOV   DI,DX
      ADD   DI,SCNDOFF&U&_
      MOVS  CAPDATA_,EMPBUF&U&_

; GET THE THIRD DATUM, ASSUME DX IS ALREADY SET TO THE DESTINATION
      MOV   DI,DX
      ADD   DI,THRDOFF&U&_
      MOVS  CAPDATA_,EMPBUF&U&_

; GET THE FOURTH DATUM, ASSUME DX IS ALREADY SET TO THE DESTINATION
      MOV   DI,DX
      ADD   DI,FRTHOFF&U&_
      MOVS  CAPDATA_,EMPBUF&U&_

;ADJUST THE INPUT BUFFER COUNT
; INTERRUPT SHOULD ALREADY BE DISABLED, SO THERE WILL BE NO INTERFERENCE
; FROM THE MILLISECOND INTERRUPT ROUTINE
      SUB   IBFCNT&U&_,2
      JNS   DOWN
      MOV   IBFCNT&U&_,0
DOWN:
      ENDM
```

```
   ;************************************************************
   ;***                        EQUATES, AND DECLARATIONS
   ;************************************************************

;COMMENTS FROM MANX C SETUP:
;
;      If you want your stack at a fixed place, you may remove the
;      "bss cstack..." statement below and change the "mov sp,cstack..."
;      to load SP with the value you desire.  Note that the setup of
;      SS may need to be changed also.  If the program is small data
;      model, then SS must be equal to DS or pointers to automatic
;      variables won't work.
;
;      Otherwise, stacksize should be set according to your program's
;      requirements.
stacksize equ     2048
;
;      DATASEG is the data segment address (as a paragraph #)
;      this is picked from the -D option of the linker
;
;Dseg equ   040H  ;physical addr 0x400, just above the int vectors

; IT IS A LITTLE TRICKY ASSIGNING LOWER RAM, IT SEEMS THAT THE FIRST BYTE
;   IN dataseg IS RESERVED FOR _DORG_.  INSERTING ORG STATMENTS SEEMS TO
;   ENFORCE A TWO BYTE OFFSET.

; INTERRUPT VECTORS:
DATASEG     segment     word public 'data'

      DW    ?
      PUBLIC      VEC01
VEC01 DW    ?
      DW    ?
VEC02 DW    ?
      DW    ?
VEC03 DW    ?
      DW    ?
VEC04 DW    ?
      DW    ?
VEC05 DW    ?
      DW    ?
VEC06 DW    ?
      DW    ?
VEC07 DW    ?
      DW    ?
VEC08 DW    ?
      DW    ?
VEC09 DW    ?
      DW    ?
VEC0A DW    ?
      DW    ?
VEC0B DW    ?
      DW    ?
VEC0C DW    ?
      DW    ?
VEC0D DW    ?
      DW    ?
VEC0E DW    ?
      DW    ?
VEC0F DW    ?
      DW    ?
VEC10 DW    ?
      DW    ?
VEC11 DW    ?
      DW    ?
VEC12 DW    ?
```

```
        DW      ?
VEC13   DW      ?
        DW      ?
VEC14   DW      ?
        DW      ?
VEC15   DW      ?
        DW      ?
VEC16   DW      ?
        DW      ?
VEC17   DW      ?
        DW      ?
VEC18   DW      ?
        DW      ?
VEC19   DW      ?
        DW      ?
VEC1A   DW      ?
        DW      ?
VEC1B   DW      ?
        DW      ?
VEC1C   DW      ?
        DW      ?
VEC1D   DW      ?
        DW      ?
VEC1E   DW      ?
        DW      ?
VEC1F   DW      ?
        DW      ?

        PUBLIC          VECT0
VECT0   DW      OFFSET MSINT
        DW      CODESEG
        PUBLIC          VECSER
VECSER          DW      ?
        DW      ?
        PUBLIC          VECT2
VECT2   DW      ?
        DW      ?
VECFR0          DW      ?
        DW      ?
VECFR1          DW      ?
        DW      ?
VECFR2          DW      ?
        DW      ?
        PUBLIC          VECXCLK
VECXCLK         DW      OFFSET SAMPINT
        DW      CODESEG
        PUBLIC          VECCLK
VECCLK          DW      OFFSET SAMPINT
        DW      CODESEG
```

```
;MANX C DECLARATIONS
        public      $MEMRY
$MEMRY      dw      -1
        dw      -1
        public      errno_
errno_      dw      0
        public      _dsval_,_csval_
_dsval_     dw      0
_csval_     dw      0
        public      _mbot_, sbot
_mbot_      dw      0
        dw      0
sbot    dw      0
        dw      0
        extrn _Dorg_:byte,_Dend_:byte
        extrn _Uorg_:byte,_Uend_:byte
        public      cstack
        bss     cstack:byte,stacksize
DATASEG     ends
        extrn _Corg_:byte,_Cend_:byte

ifdef FARPROC
        extrn main_:far, $fltinit:far
else
        extrn main_:near, $fltinit:near
endif
```

```
;MANX C STARTUP CODE
;_____________________________________________________________________
;_____________________________________________________________________
;
;/////////////////   STARTUP ROUTINE CODE SPACE   \\\\\\\\\\\\\\\\\\\
;_____________________________________________________________________
;_____________________________________________________________________

      public      $begin
$begin      proc  far
      cli
      cld
; initialize stack
      mov   bx,DATASEG  ;place where data is to go in ram
      mov   ds,bx
      mov   es,bx
;
;     Note: For hardware reasons the instruction which loads SS should
;           be immediatly followed by the load of SP.
;
      mov   ss,bx
      mov   sp,offset cstack+stacksize

;************************************************************
;
; V40 INIT CODE
;
;************************************************************
      ASSUME CS:CODESEG,DS:DATASEG,ES:DATASEG,SS:DATASEG



;P1BASE
; SET UP ON CHIP PERIPHERAL CONNECTION REGISTER (OPCN)
;
; BITS: 7 6 5 4 |  3 2 | 1 0 |
;       - - - - | IRSW | PF  |
;     IRSW = 00   INTP1 PIN IS INT1, INTP2 PIN IS INT2
;        = 01 SCU INT IS INT1, INTP2 PIN IS INT2
;        = 10 INTP1 PIN IS INT1, TOUT2 IS INT2
;        = 11 SCU INT IS INT1, TOUT2 IS INT2
;
;     PF   = 00 DMARQ3, DMAAK3-, INTAK-
;        = 01 DMARQ3, DMAAK3-, TOUT1
;        = 10 RxD, TxD, INTAK-
;        = 11 RxD, TxD, SRDY-
;
      OUTPTL      OPCN,BIT3+BIT2+BIT0

; SET UP ON CHIP PERIPHERAL SELECTION REGISTER (OPSEL)
; BITS: 7 6 5 4 |  3 |  2 |  1 |  0 |
;       - - - - | SS | TS | IS | DS |
;
;     SS = 0 DISABLE SERIAL CONTROL UNIT
;     SS = 1 ENABLE SERIAL CONTROL UNIT
;     TS = 0 DISABLE TIMER CONTROL UNIT
;     TS = 1 ENABLE TIMER CONTROL UNIT
;     IS = 0 DISABLE INTERRUPT CONTROL UNIT
;     IS = 1 ENABLE INTERRUPT CONTROL UNIT
;     DS = 0 DISABLE DMA CONTROL UNIT
;     DS = 1 ENABLE DMA CONTROL UNIT
;     OUTPTL      OPSEL,BIT2+BIT1+BIT0    ;ENABLE DMA,INTERRUPT AND TIMER UNITS
      OUTPTL      OPSEL,BIT2+BIT1         ;FOR NOW, ONLY TIMER UNIT AND ICU
```

```
;       SET OPHA TO 00H
        OUTPTL      OPHA,IPBASE

;       SET UP INTERNAL PERIPHERAL ADDRESSES
        OUTPTL      DULA,DMABASE
        OUTPTL      IULA,IUBASE
        OUTPTL      TULA,TUBASE
        OUTPTL      SULA,SUBASE



; TIMER CLOCK SELECTION REGISTER SETUP
; BITS: 7 6 5 |  4  |  3  |  2  |  1  |  0  |
;       - - - | CS2 | CS1 | CS0 |     PS    |
;
;       PS = 00 PRESCALE BY 2
;       PS = 01 PRESCALE BY 4
;       PS = 10 PRESCALE BY 8
;       PS = 11 PRESCALE BY 16
;
;       CSN = 0     CLOCK TO TIMER N = INTERNAL CLOCK
;       CSN = 1 CLOCK TO TIMER N = TCLK PIN
        OUTPTL      TCKS,00010000B          ;PRESCALE=2, ONLY T2 DRIVEN EXTERNALLY

; SET UP REFRESH CONTROL REGISTER
; BITS: |  7  |  6  |  5  |  4  |  3  |  2  |  1  |  0  |
;       | RE  |  -  |  -  |             RTM             |
;
;       RE = 0 DISABLES REFRESH
        OUTPTL      RFC,0H

; SET UP MEMORY BOUNDARY REGISTER
; BITS: |  7  |  6  |  5  |  4  |  3  |  2  |  1  |  0  |
;       |  -  |       LMB       |  -  |       UMB       |
;
;       LMB AND UMB BLOCK SIZE
;       000   32KB
;       001   64KB
;       010   96KB
;       011   128KB
;       100   192KB
;       101   256KB
;       110   384KB
;       111   512KB
        OUTPTL      WMB,0H              ;SET EACH SIZE AT 32KB

; SET UP WAIT CYCLES FOR DIFFERENT PARTS OF MEM AND I/O (WCY1)
; BITS: |  7  |  6  |  5  |  4  |  3  |  2  |  1  |  0  |
;       |    IOW    |    UMW    |    MMW    |    LMW    |
;
;       XXW = 00    0 WAIT STATES
;             01    1 WAIT STATE
;             10    2 WAIT STATES
;             11    3 WAIT STATES
        OUTPTL      WCY1,11000000B    ; SET RAM/ROM TO 0 WAIT STATES
                            ;     I/O TO 3 WAIT STATES
                            ; MAYBE CHANGE LATER
```

```
; SET UP WAIT CYCLES FOR DMA AND REFRESH (WCY2)
; BITS: |  7  |  6  |  5  |  4  |  3  |  2  |  1  |  0  |
;       |  -  |  -  |  -  |  -  |   DMAW    |    RFW    |
;
;     XXW = 00     0 WAIT STATES
;           01     1 WAIT STATE
;           10     2 WAIT STATES
;           11     3 WAIT STATES
      OUTPTL       WCY2,00000000B    ;SET EVERYTHING TO 0 WAIT STATES
                         ; TUNE THIS LATER IF USED

; SET UP INTERRUPT REGISTER
; INITIALIZE INTERRUPT CONTROLLER
      OUTPT INTREG0,00010011B ; INITIALIZE,EDGE TRIG,SINGLE MODE
      OUTPT INTREG1,00100000B ;VECTOR TABLE STARTS AT ADD 80H
      OUTPT INTREG1,00000011B ;SELF FI MODE

      MOV   AL,0FFH           ;SET MASK VECTOR
      CALL  MASKINT

; THE FOLLOWING CLOCK MUST BE STARTED BEFORE INITIALIZING UARTS, FOR
;  PROPER INIT:

; SET UP TIMER 2, IT USES TCLK (7.37mhz) AND DIVIDES IT BY TWO
;   SEE V40 TECHNICAL LITERATURE
      MOV   AL,10110100B      ;REG2,LOWER BYTE THEN HIGHER,MODE2,BINARY
      OUT   CLKCON,AL
      MOV   AL,2
      OUT   CLK2,AL
      MOV   AL,0
      OUT   CLK2,AL

; INITIALIZE UART DATA STRUCTURES
      INITU 0
      INITU 1
      INITU 2
      INITU 3
      INITU 4
      INITU 5
      INITU 6
      INITU 7
      INITU 8
      INITU 9
      INITU A
      INITU B
      INITU C
      INITU D
      INITU E
      INITU F

; INITIALIZE DUART DATA STRUCTURES
      INITD 0
      INITD 1
      INITD 2
      INITD 3
      INITD 4
      INITD 5
      INITD 6
      INITD 7
```

```
; ENABLE UARTS
      CALL  ENABLEU0_
      CALL  ENABLEU1_
      CALL  ENABLEU2_
      CALL  ENABLEU3_
      CALL  ENABLEU4_
      CALL  ENABLEU5_
      CALL  ENABLEU6_
      CALL  ENABLEU7_
      CALL  ENABLEU8_
      CALL  ENABLEU9_
      CALL  ENABLEUA_
      CALL  ENABLEUB_
      CALL  ENABLEUC_
      CALL  ENABLEUD_
      CALL  ENABLEUE_
      CALL  ENABLEUF_

;  INIT DMA TO PC OPERATIONS
;
      MOV   DMAPTR_,OFFSET BUFFER_
      MOV   DMACNT_,0FFFFH
      MOV   CAPFLAG_,0H
      MOV   AL,DMAON+DMA0+DMA1
      OUT   DMAPORT+RESET,AL
      OUT   DMAREG+1,AL
;  INIT SAMPLE BUFFERING STRUCTURE, SET TO SAMPLE INTO ONLY 2 BUFFER
      MOV   AX,OFFSET BUFFER_
      MOV   SBIPTR_,AX
      MOV   SBOPTR_,AX
      MOV   SBBPTR_,AX
      ADD   AX,256
      MOV   SBEPTR_,AX
      MOV   SBINC_,128
      MOV   SBCNT_,0
      MOV   DMAMXCNT_,64
      MOV   SBMXCNT_,2


; SET UP TIMER 1, IT USES INTERNAL CLOCK (3.68 mhz) AND DIVIDES IT BY 24
;                           TO GET THE APPROP FREQ FOR 9600
      MOV   AL,01110100B      ;REG2,LOWER BYTE THEN HIGHER,MODE2,BINARY
      OUT   CLKCON,AL
      MOV   AL,24
      OUT   CLK1,AL
      MOV   AL,0
      OUT   CLK1,AL

; SET UP TIMER 0, IT USES INTRNL CLK (3.68 mhz) AND DIVIDES TO 1MS
      MOV   AL,00110100B      ;REG2,LOWER BYTE THEN HIGHER,MODE2,BINARY
      OUT   CLKCON,AL
      MOV   AL,LOW 3686
      OUT   CLK0,AL
      MOV   AL,HIGH 3686
      OUT   CLK0,AL

;SET UP MILLISECOND INTERRUPT VECTOR
      MOV   AX,CODESEG
      MOV   VECT0+2,AX
      MOV   AX,OFFSET MSINT
      MOV   VECT0,AX
```

```
;SET UP MILLISECOND ADJUSTMENT REGISTERS
      MOV   BASECNT0_,3686
      MOV   BASECNT1_,3687
      MOV   REMDCNT,0
      MOV   CNTINC_,26214           ;((65536.0*4.0)/10.0)
      MOV   PERIOD_,10
      MOV   MCOUNT_,10  ;RESET MCOUNT

; CLEAR THE MS INTERRUPT FLAG
      MOV   MSINTFLG,0

;ENABLE T0 INTERRUPT
      MOV   AL,00000001B
      CALL  UNMASKINT
      MOV   AL,01000000B
      CALL  UNMASKINT
; END V40 INIT CODE
;

;*****************************************************************
;
;     BACK TO MANX C STARTUP CODE
;
;*****************************************************************

      cli
      cld
;
;     Compute where initialzed data starts (@ next para after code)
;
      mov   ax,offset _Cend_+15
      mov   cl,4
      shr   ax,cl
      add   ax,seg _Cend_
      mov   ds,ax       ;place where data is in rom
      mov   bx,DATASEG  ;place where data is to go in ram
      mov   es,bx
;
;     Note: For hardware reasons the instruction which loads SS should
;           be immediatly followed by the load of SP.
;
      mov   ss,bx
      mov   sp,offset cstack+stacksize
;
;           copy Init data from rom to ram
      mov   di,0
      mov   cx,offset _Dend_
      inc   cx
      shr   cx,1
      jcxz  nocopy
      mov   si,0
rep   movsw
nocopy:
```

```
;
;           clear uninitialized data

;**** don't zero uninitialized stuff
      jmp   noclear
;****
      mov   di,offset _Uorg_
      mov   cx,offset _Uend_
      sub   cx,di
      inc   cx
      shr   cx,1
      jcxz  noclear
      sub   ax,ax
rep   stosw
noclear:
;
      assume      ds:DATASEG,es:DATASEG

      mov   ds,bx             ;set DS, now DS, SS, ES are equal
      mov   di,$MEMRY
      inc   di
      and   di,0fffeH   ;adjust to word boundary
      mov   $MEMRY,di   ;save memory allocation info for sbrk()
      mov   $MEMRY+2,ds
      mov   _mbot_,di
      mov   _mbot_,ds
      mov   sbot,0ffffH ;this is the heap limit for sbrk()
      mov   sbot+2,0fff0h
      mov   _dsval_,ds
      mov   _csval_,cs  ;this is of dubious value in large code
      sti
      call  $fltinit    ;setup floating point software/hardware
      jnc   flt_ok
      hlt               ;program needs 8087 and one wasn't found
flt_ok:

      jmp   main_       ;main shouldn't return in ROM based system
$begin      endp


      ASSUME CS:CODESEG,DS:DATASEG,ES:DATASEG,SS:DATASEG
```

```
;*************************************************************
;
;     DEFINE THE ENABLE/DISABLE UART ROUTINES
;
;*************************************************************
      ENABLEU     0
      ENABLEU     1
      ENABLEU     2
      ENABLEU     3
      ENABLEU     4
      ENABLEU     5
      ENABLEU     6
      ENABLEU     7
      ENABLEU     8
      ENABLEU     9
      ENABLEU     A
      ENABLEU     B
      ENABLEU     C
      ENABLEU     D
      ENABLEU     E
      ENABLEU     F

      DISABLEU    0
      DISABLEU    1
      DISABLEU    2
      DISABLEU    3
      DISABLEU    4
      DISABLEU    5
      DISABLEU    6
      DISABLEU    7
      DISABLEU    8
      DISABLEU    9
      DISABLEU    A
      DISABLEU    B
      DISABLEU    C
      DISABLEU    D
      DISABLEU    E
      DISABLEU    F

;*****************************************************************
;
;           OTHER SERVICE ROUTINES
;
;*****************************************************************
UNMASKINT: XOR    AL,0FFH           ;COMPLEMENT AL
      AND   AL,INTMASK
      JMP   SHORT MASKINT1
MASKINT: OR AL,INTMASK
MASKINT1: MOV     INTMASK,AL
      OUTPT INTREG1,AL
      RET

;*****************************************************************
;***                      procdef's                            ***
;***                                                           ***
;*****************************************************************

; PERFORM INPUT
procdef     IOin  <<IOIport,word>>
      MOV   DX,IOIport
      IN    AL,DX
      MOV   AH,0
      pret
      pend  IOin

; PERFORM OUTPUT
```

```
procdef     IOout <<IOOport,word>,<ODATA,word>>
      MOV   AX,ODATA
      MOV   DX,IOOport
      OUT   DX,AL
      pret
      pend  IOout

;GET ADDRESS OF START OF DMA (HOLDING) BUFFERS
procdef     GetPoint
      MOV   AX,OFFSET BUFFER_
      pret
      pend  GetPoint

;TEMPORARILY DISABLE INTERRUPTS
procdef INTOFF
      INTOFF
      pret
      pend  INTOFF

;ENABLE INTERRUPTS
procdef INTON
      INTON
      pret
      pend  INTON

;FORCE A HARD RESET
procdef     ColdReset
      JMP   $begin
      pret
      pend  ColdReset
```

```
;PROCESS A DMA TRANSFER TO PC, IF NECESSARY
procdef     DODMA
      PUSH  AX
      PUSH  BX
      CMP   DMACNT_,0FFFFH
      JE    DODMAX
DODMA1:     CMP   DMACNT_,0
      JNE   DODMA2            ;IF DMACNT_ IS NOT ZERO, WE SEE IF DMA IS RDY

; ANY BUFFERS THERE?
      CMP   SBCNT_,0    ;CHECK SBCNT_
      JE    DODMAX            ;IF ZERO, EXIT
;BUFFERS THERE!

;TIME TO LOAD UP A NEW DMA BUFFER
    ;DECREASE BUFFER COUNT
      DEC   SBCNT_
    ;LOAD NEW DMA POINTER FROM OUTPUT BUFFER POINTER
      MOV   AX,SBOPTR_
      MOV   DMAPTR_,AX
    ;ADVANCE OUTPUT BUFFER POINTER AND WRAP IT IF NECESSARY
      MOV   AX,SBOPTR_  ;GET POINTER IN AX
      ADD   AX,SBINC_   ;ADJUST IT
      CMP   AX,SBEPTR_  ;CMPARE WITH MAX+1
      JNE   DODMA4            ;JUMP IF OKAY
      MOV   AX,SBBPTR_  ;NOT OKAY, SO WRAP
DODMA4:     MOV   SBOPTR_,AX  ; NOW STORE POINTER
    ;RESET THE DMA COUNT TO THE BUFFER SIZE
      MOV   AX,DMAMXCNT_
      MOV   DMACNT_,AX


DODMA2:
; IS DMA READY?
      IN    AL,DUART2+INREG
      AND   AL,01H            ;LOOK AT DMA DONE BIT
      JNE   DODMA3            ; IF NON ZERO, THEN READY, SO JMP TO RDY CODE
DODMAX:
      POP   BX
      POP   AX
      pret

DODMA3:
; GREAT, READY FOR DMA
      MOV   BX,DMAPTR_  ;GET POINTER TO DATA
      MOV   AX,[BX]           ;GET ACTUAL DATA
      OUT   DMAREG,AX   ;DMA DONE, NOW UPDATE
      DEC   DMACNT_           ;DECREASE COUNT
      ADD   DMAPTR_,2   ;ADVANCE POINTER
      JMP   DODMA1            ; AND LOOP BACK TO TRY AGAIN

      pend  DODMA
```

```
   ;*****************************************************************
   ;***                INTERRUPT CALLS                            ***
   ;*****************************************************************



;SAMPLE INTERVAL INTERRUP ROUTINE
SAMPINT:
      PUSH  AX
      PUSH  DX
      PUSH  SI
      PUSH  DI

;DUART6 SHOWS THE BUFFER COUNT ON ITS OUTPUT (FIRST 8 BITS)
      MOV   DX,DUART6+RESET
      MOV   AL,255
      OUT   DX,AL
      MOV   DX,DUART6+SET
      MOV   AX,SBCNT_
      OUT   DX,AL

;CHECK TO SEE IF THE CURRENT SAMPLED BUFFER IS TO GO INTO THE
; DMA HOLDING BUFFERS,  OR INTO THE CAPTURE BUFFER
; * IN NORMAL OPERATION, THE CAPTURE BUFFER IS NOT USED.
      CMP   CAPFLAG_,0
      JE    SAMPINTA
      JMP   SAMPINTC
;NOT CAPTURE, BUT RATHER, DMA SAMPLE
SAMPINTA:

;LOAD BUFFER POINTER (FOR COLLECTING THIS SAMPLE) INTO DX
      MOV   DX,SBIPTR_

;ADJUST INPUT POINTER FOR NEXT TIME
      MOV   AX,DX
      ADD   AX,SBINC_    ;INCREMENT IT
      CMP   AX,SBEPTR_   ;IS IT TOO BIG?
      JNE   SMPI0
;     POINTER MUST WRAP
      MOV   AX,SBBPTR_
;     POINTER IS NOW OKAY, SO STORE IT
SMPI0:      MOV   SBIPTR_,AX

;ADJUST BUFFER COUNT
      MOV   AX,SBCNT_    ;GET COUNT
      INC   AX           ;INCREMENT IT
      CMP   AX,SBMXCNT_ ;COMPARE TO MAX
      JNE   SMPI1        ;JUMP IF NO PROBLEM

   ;COUNT TOO HI, SO, ADVANCE OUTPUT POINTER
      MOV   AX,SBOPTR_   ;GET POINTER IN AX
      ADD   AX,SBINC_    ;ADJUST IT
      CMP   AX,SBEPTR_   ;CMPARE WITH MAX+1
      JNE   SMPI2        ;JUMP IF OKAY
      MOV   AX,SBBPTR_   ;NOT OKAY, SO WRAP
SMPI2:      MOV   SBOPTR_,AX  ; NOW STORE POINTER
      JMP   SMPI3        ; DO NOT CHANGE BUFFER COUNT

SMPI1:      MOV   SBCNT_,AX
```

```
SMPI3:      CLD

      GETDATA     0     ;ASSUMES DX IS BASE PTR, USES DI,SI
      GETDATA     1
      GETDATA     2
      GETDATA     3
      GETDATA     4
      GETDATA     5
      GETDATA     6
      GETDATA     7
      GETDATA     8
      GETDATA     9
      GETDATA     A
      GETDATA     B
      GETDATA     C
      GETDATA     D
      GETDATA     E
      GETDATA     F

SMPI4:

SMPIX:


      POP   DI
      POP   SI
      POP   DX
      POP   AX
      IRET

; GET DATA INTO CAPTURE BUFFER, AND DECREMENT CAPTURE BUFFER COUNT
SAMPINTC:
      DEC   CAPFLAG_

SAMPINTE:
      MOV   DX,OFFSET CAPDATA_
      JMP   SMPI3
```

```
;*********************************************************
;
;     BEGINNING OF MILLISECOND INTERRUPT ROUTINE
;           THE FILES SERIN0,SERIN8 AND SERINX MUST BE
;           LINK RIGHT AFTER DIGIREC, TO COMPLETE THE
;           MSINT CODE PROPERLY
;
;*********************************************************
MSINT:      PUSH  AX
      INTON
;SET MILLISECOND FLAG
      MOV   AX,0FFFFH
      MOV   msflag_,AX

;ADJUST SUB-MICROSECOND COUNTER
; DECIDE WHICH COUNT FOR NEXT MILLISECOND
      MOV   AX,CNTINC_
      ADD   REMDCNT,AX
      JNC   MSINTL
      MOV   AX,BASECNT1_
      JMP   CHGCNT

MSINTL:     MOV   AX,BASECNT0_
CHGCNT:
      ;SEND OUT NEW COUNT, FIRST LOW BYTE, THEN HIGH
      OUT   CLK0,AL
      MOV   AL,AH
      OUT   CLK0,AL

MSINT3:
;CHECK TO SEE IF A SAMPLE INTERVAL IS OVER (MCOUNT_ BECOMES 1)
;DECREMENT THE MILLISECOND COUNTER
      DEC   MCOUNT_
      JE    MSINT1
      CMP   MCOUNT_,1
      JNE   MSINT2
      MOV   AL,0FFH
      OUT   DUART7+SET,AL
      JMP   MSINT2

MSINT1:
      MOV   AX,PERIOD_
      MOV   MCOUNT_,AX
      MOV   AL,0FFH
      OUT   DUART7+RESET,AL

MSINT2:
;MAKE SURE A CONCURRENT UART SERVICE IS NOT IN PROGRESS
      TEST  MSINTFLG,0FFH
      JE    MSINT4
      POP   AX
      IRET
MSINT4:
; CHECK UARTS FOR INPUT
      PUSH  BX
      PUSH  BP
      PUSH  CX
      MOV   MSINTFLG,0FFH
      MOV   BP,DMAPTR_
      MOV   CX,DMACNT_

;*********************************************************
;
;   END OF CODE IN THIS FILE
;     MSINT CODE CONTINUES IN SERIN0.MAC, SERIN8.MAC AND SERINX.MAC
;*********************************************************
```

```
;*****************************************
;      DEFINE UART DATA STRUCTURES
;
       UDAT  0
       UDAT  1
       UDAT  2
       UDAT  3
       UDAT  4
       UDAT  5
       UDAT  6
       UDAT  7
       UDAT  8
       UDAT  9
       UDAT  A
       UDAT  B
       UDAT  C
       UDAT  D
       UDAT  E
       UDAT  F

;_____________________________________________________________________
;_____________________________________________________________________
;
;//////////////////////////   DATA SPACE   \\\\\\\\\\\\\\\\\\\\\\\\\\\\
;_____________________________________________________________________
;_____________________________________________________________________

DATASEG      SEGMENT
       BSS    INTMASK:BYTE,1

       PUBLIC       MCOUNT_
       BSS    MCOUNT_:WORD,2
       BSS    SAMPFLG:BYTE,1
       BSS    DUMMY1:BYTE,1
       BSS    DUMMY2:BYTE,1

       PUBLIC       BASECNT0_
       PUBLIC       BASECNT1_
       PUBLIC       CNTINC_
                    ;THE PROCESSOR CLOCK IS A NOMINAL 14.7456
                    ; THIS IS DIVIDED BY 4, BEFORE REACHING THE
                    ; COUNTER THAT PRODUCES THE MILLISECOND INTERRUPT
                    ;THE MILLISECOND INTERRUPT COUNTER IS LOADED WITH
                    ; ONE OF TWO CONSTANTS, EACH MILLISECOND, IN ORDER
                    ; TO PRODUCE AN AVERAGE MILLISECOND INTERVAL OF
                    ; HIGH PRECISION.
                    ; BASECNT1_ AND BASECNT2_ ARE THE TWO CONSTANTS THAT
                    ; ARE USED.  CNTINC_ IS THE INCREMENT THAT IS
                    ; USED TO DYNAMICALLY DECIDE WHICH OF TWO CONSTANTS
                    ; SHOULD BE USED NEXT
       BSS    BASECNT0_:WORD,2
       BSS    BASECNT1_:WORD,2
       BSS    REMDCNT:WORD,2
       BSS    CNTINC_:WORD,2
       PUBLIC       PERIOD_
       BSS    PERIOD_:WORD,2
```

```
        PUBLIC      MSINTFLG            ;MSINTFLG IS USED TO STOP THE
                                      ; MSINT ROUTINE FROM SERVICING UARTS
                                      ; IF IT IS ALREADY DOING IT.  E.G.
                                      ; A SECOND MILLISECOND INTERRUPT CAN
                                      ; COME IN, EVEN IF UARTS ARE STILL
                                      ; BEING SERVICED FROM AN EARLIER INT.
                                      ; THIS FLAG MAKES IT POSSIBLE TO
                                      ; SERVICE THE MS COUNTER, WITHOUT
                                      ; RE ENTERING THE UART SERVICE CODE.
        BSS   MSINTFLG:BYTE,2
        PUBLIC      CAPFLAG_
        BSS   CAPFLAG_:WORD,2
        PUBLIC      CAPDATA_
        BSS   CAPDATA_:WORD,256
        PUBLIC      SBMXCNT_
        PUBLIC      SBINC_
        PUBLIC      SBEPTR_
        PUBLIC      SBBPTR_
        PUBLIC      SBOPTR_
        PUBLIC      SBIPTR_
        PUBLIC      SBCNT_
        PUBLIC      DMAMXCNT_
        PUBLIC      DMACNT_
        PUBLIC      DMAPTR_
        BSS   DMAPTR_:WORD,2
        BSS   DMACNT_:WORD,2
        BSS   DMAMXCNT_:WORD,2
        BSS   SBCNT_:WORD,2
        BSS   SBIPTR_:WORD,2
        BSS   SBOPTR_:WORD,2
        BSS   SBBPTR_:WORD,2
        BSS   SBEPTR_:WORD,2
        BSS   SBINC_:WORD,2
        BSS   SBMXCNT_:WORD,2
        PUBLIC      msflag_
        BSS   msflag_:WORD,2

BUFFERSIZE_ EQU   06000H
        PUBLIC      BUFFER_
        BSS   BUFFER_:WORD,BUFFERSIZE_
DATASEG     ENDS


;*****************************************************************
codeseg ends
        end   $begin
```

```
FILE: lmacros.h
      nlist
; Copyright (C) 1985 by Manx Software Systems, Inc.
; :ts=8
      ifndef      MODEL
MODEL equ   0
      endif
      if    MODEL and 1
      largecode
FARPROC     equ 1
FPTRSIZE equ      4
      else
FPTRSIZE equ      2
      endif
      if    MODEL and 2
LONGPTR equ 1
      endif

;this macro to be used on returning
;restores bp and registers
pret  macro
if havbp
      pop bp
endif
      ret
      endm

internal macro    pname
      public      pname
pname proc
      endm

intrdef     macro pname
      public      pname
ifdef FARPROC
      pname label far
else
      pname label near
endif
      endm

procdef     macro pname, args
      public pname&_
ifdef FARPROC
      _arg  = 6
      pname&_     proc  far
else
      _arg  = 4
      pname&_     proc  near
endif
ifnb <args>
      push bp
      mov bp,sp
      havbp = 1
      decll <args>
else
      havbp = 0
endif
      endm
```

```
entrdef     macro pname, args
      public pname&_
ifdef FARPROC
      _arg  = 6
      pname&_:
else
      _arg  = 4
      pname&_:
endif
ifnb <args>
if    havbp
      push  bp
      mov   bp,sp
else
      error must declare main proc with args, if entry has args
endif
      decll <args>
endif
      endm

;this macro equates 'aname' to arg on stack
decl  macro       aname, type
;;'byte' or anything else
havtyp      = 0
ifidn <type>,<byte>
      aname equ   byte ptr _arg[bp]
      _arg = _arg + 2
      havtyp = 1
endif
ifidn <type>,<dword>
      aname equ dword ptr _arg[bp]
      _arg = _arg + 4
      havtyp = 1
endif
ifidn <type>,<cdouble>
      aname equ qword ptr _arg[bp]
      _arg = _arg + 8
      havtyp = 1
endif
ifidn <type>, <ptr>
      ifdef LONGPTR
            aname equ dword ptr _arg[bp]
            _arg = _arg + 4
      else
            aname equ   word ptr _arg[bp]
            _arg = _arg + 2
      endif
      havtyp = 1
endif
ifidn <type>, <fptr>
      ifdef FARPROC
            aname equ dword ptr _arg[bp]
            _arg = _arg + 4
      else
            aname equ   word ptr _arg[bp]
            _arg = _arg + 2
      endif
      havtyp = 1
endif
ifidn <type>, <word>
      aname equ   word ptr _arg[bp]
      _arg = _arg + 2
      havtyp = 1
endif
```

```
ife   havtyp
      error -- type is unknown.
endif
      endm

;this macro loads an arg pointer into DEST, with optional SEGment
ldptr macro dest, argname, seg
ifdef LONGPTR
      ifnb <seg>        ;;get segment if specified
            ifidn <seg>,<es>
                  les   dest,argname
            else
                  ifidn <seg>,<ds>
                        lds dest,argname
                  else
                        mov dest, word ptr argname
                        mov seg, word ptr argname[2]
                  endif
            endif
      else
            ifidn <dest>,<si>       ;;si gets seg in ds
                  lds   si, argname
            else
                  ifidn <dest>,<di> ;;or, es:di
                        les   di, argname
                  else
                        garbage error: no seg for long pointer
                  endif
            endif
      endif
else
      mov dest, word ptr argname    ;;get the pointer
ENDIF
      ENDM

decll macro list
      IRP   i,<list>
      decl i
      ENDM
      ENDM

pend  macro pname
pname&_     endp
      endm

retptrm     macro src,seg
mov   ax, word ptr src
ifdef LONGPTR
      mov   dx, word ptr src+2
endif
      endm

retptrr     macro src,seg
mov   ax,src
ifdef LONGPTR
      ifnb <seg>
            mov   dx, seg
      endif
endif
      endm
```

```
retnull     macro
ifdef LONGPTR
      sub   dx,dx
endif
      sub   ax,ax
      endm

pushds      macro
      ifdef LONGPTR
      push  ds
      endif
      endm

popds macro
      ifdef LONGPTR
      pop   ds
      endif
      endm

finish      macro
codeseg     ends
      endm
```

```
File:digirec.equ
      list
codeseg     segment     byte public 'code'
      assume      cs:codeseg

; V40 and local equates

   ;***  CONSTANTS  ****************************************

ZERO      EQU      00H
BIT0      EQU      1H
BIT1      EQU      2H
BIT2      EQU      4H
BIT3      EQU      8H
BIT4      EQU      10H
BIT5      EQU      20H
BIT6      EQU      40H
BIT7      EQU      80H
XON       EQU      17
XOFF      EQU      19
STARTBYTE EQU       13
;STARTBYTE  EQU    "¦"
OFF    EQU    0
ON     EQU    1

MINUSONE  EQU     FFFFH

ROM       EQU    0F800H     ;***  Beginning of ROM (paragraph)
RAM       EQU    00H        ;***  Beginning of RAM (paragraph)
                            ;***  BOOT address = F800:7FF0H

TCKS   EQU    0FFF0H
RFC    EQU    TCKS+2
WMB    EQU    TCKS+4
WCY1   EQU    TCKS+5
WCY2   EQU    TCKS+6
SULA   EQU    TCKS+8
TULA   EQU    TCKS+9
IULA   EQU    TCKS+10
DULA   EQU    TCKS+11
OPHA   EQU    TCKS+12
OPSEL  EQU    TCKS+13
OPCN   EQU    TCKS+14
```

```
; IO ADDRESS DEFINITIONS
DMAREG      EQU   0C0H

DUART0      EQU   00H    ; BASE ADDRESS OF DUART0
UART0 EQU   DUART0       ; UART0
UART1 EQU   DUART0+8; UART1

DUART1      EQU   10H    ; BASE ADDRESS OF DUART0
UART2 EQU   DUART1       ; UART2
UART3 EQU   DUART1+8; UART3

DUART2      EQU   20H    ; BASE ADDRESS OF DUART0
UART4 EQU   DUART2       ; UART4
UART5 EQU   DUART2+8; UART5

DUART3      EQU   30H    ; BASE ADDRESS OF DUART0
UART6 EQU   DUART3       ; UART6
UART7 EQU   DUART3+8; UART7

DUART4      EQU   40H    ; BASE ADDRESS OF DUART0
UART8 EQU   DUART4       ; UART8
UART9 EQU   DUART4+8; UART9

DUART5      EQU   50H    ; BASE ADDRESS OF DUART0
DMAPORT     EQU   DUART5
      DMAON EQU   4
      DMA1  EQU   2
      DMA0  EQU   1

UARTA EQU   DUART5  ; UARTA
UARTB EQU   DUART5+8; UARTB

DUART6      EQU   60H    ; BASE ADDRESS OF DUART0
UARTC EQU   DUART6       ; UARTC
UARTD EQU   DUART6+8; UARTD

DUART7      EQU   70H    ; BASE ADDRESS OF DUART0
UARTE EQU   DUART7       ; UARTE
UARTF EQU   DUART7+8; UARTF

UMODE EQU   0      ; MODE REGISTER
USTAT EQU   1      ; STATUS REG, CLOCK SELECT
UCOMM EQU   2      ; COMMAND REG, WRITE ONLY
UBUFF EQU   3      ; DATA REGISTER
ACR   EQU   4      ; AUX CONTROL REG
IPCR  EQU   4      ; INPUT CHANGE REG
IMR   EQU   5      ; INTERRUPT MODE REGISTER
OPCR  EQU   13     ;OUTPUT CONFIGURE REGISTER
INREG EQU   13     ;INPUT REG
RESET EQU   14     ;RESET OUTPUT REGISTER (OPPOSITE POLARITY FROM DOCS)
SET   EQU   15     ;SET OUTPUT REGISTER (OPPOSITE POLARITY FROM DOCS)
CSR   EQU   1      ;CLOCK SELECT REGISTER FOR UART

;BITS IN UART STATUS WORD
RRDY  EQU   1
UERRORS     EQU   11110000B

; SERIAL INPUT PROCESSING EQUATES:
INIT  EQU   0
MID   EQU   1
FULL  EQU   1
EMPTY EQU   0
SBUFLN      EQU   7
```

```
;BAUD RATE CONSTANTS (TO GO IN CSR)
R9600 EQU   10111011B
R4800 EQU   10011001B
R2400 EQU   10001000B
R1200 EQU   01100110B
R600  EQU   01010101B
R300  EQU   01000100B

; PARITY CONSTANTS
NONE  EQU   00010011B
EVENP EQU   00000011B
ODDP  EQU   00000111B

;PC INTERRUPT BITS
LOADIRQ     EQU   1
PCIE  EQU   2
IS0   EQU   4
IS1   EQU   8
IS2   EQU   16

;DMA EQUATES
DMABASE     EQU   80H    ; DMA UNIT BASE
IUBASE      EQU   90H    ; INTERRUPT UNIT BASE
TUBASE      EQU   94H    ; TIMER UNIT BASE
SUBASE      EQU   98H    ; SERIAL UNIT BASE (NOT USED)

DMARES      EQU   DMABASE
DMACHN      EQU   DMABASE+1
DMACNTL     EQU   DMABASE+2
DMACNTH     EQU   DMABASE+3
DMAADDL     EQU   DMABASE+4
DMAADDM     EQU   DMABASE+5
DMAADDH     EQU   DMABASE+6
DMADEVL     EQU   DMABASE+8
DMADEVH     EQU   DMABASE+9
DMAMODE     EQU   DMABASE+10
DMASTAT     EQU   DMABASE+11
DMAMASK     EQU   DMABASE+15
```

```
;  DMA MODE BYTE DEFINITION
; BITS: 7 6 5 4   3 2 1 0
;       0 0                     DEMAND mode
;       0 1                     single mode
;       1 0                     BLOCK  mode
;       1 1                     CASCADE mode
;           0                   address increment
;           1                   address decrement
;             0                 no auto init
;             1                 AUTO INIT
;                 0 0           verify transfer
;                 0 1           write to memory
;                 1 0           read from memory
;                 1 1           illegal
;                       0 0     channel 0
;                       0 1     channel 1
;                       1 0     channel 2
;                       1 1     channel 3
DEMANDMD  EQU    00H
SINGLEMD  EQU    40H
BLOCKMD   EQU    80H
CASCADMD  EQU    C0H
ADDINC    EQU    00H
ADDDEC    EQU    20H
AUTOIN    EQU    10H
NOAUTOIN  EQU    00H
VERIFY    EQU    00H
TOMEM     EQU    4H
FROMEM    EQU    8H
CH0       EQU    00H
CH1       EQU    01H
CH2       EQU    02H
CH3       EQU    03H
SETON     EQU    0H
SETOFF    EQU    4H
BASEONLY  EQU    4H

AMASKBIT  EQU    02H                    ;ATOD ON CHANNEL 1

INTREG0 EQU     IUBASE
INTREG1 EQU     INTREG0+1

CLK0    EQU     TUBASE
CLK1    EQU     TUBASE+1
CLK2    EQU     TUBASE+2
CLKCON  EQU     TUBASE+3

IPBASE      EQU   0H
```

```
FILE: serin.mac

SERIN MACRO U
      LOCAL CDDONE
;*****************************************
;
; INLINE SERIAL INPUT CODE FOR CHANNEL &U
;
;*****************************************
      ASSUME      CS:CODESEG,DS:DATASEG,ES:DATASET,SS:DATASEG
SERIN0&U:
      IN    AL,UART&U+USTAT   ;GET PORT&U STATUS

      TEST  AL,RRDY
      JNE   SRN&U
      JMP   SERINX&U          ;IF DATA NOT READY, FALL THRU THIS SECTION
SRN&U:
; RRDY:
      TEST  AL,UERRORS
      JE    SERIN1&U          ;IF NO ERRORS JUMP DOWN TO NEXT PROCESSING STP

; ERRORS:
      IN    AL,UART&U+UBUFF   ;CLEAR ERROR BYTE
      MOV   AX,STSTATE&U&_
      MOV   STATE&U&_,AX      ;SET STATE TO INIT
      MOV   AL,01000000B            ;RESET ERROR FLAGS
      OUT   UART&U+UCOMM,AL
      JMP   SERIN0&U          ; AND FALL THRU THIS SECTION

; NO ERRORS:
SERIN1&U:
      MOV   BX,STATE&U&_
      JMP   BX
      PUBLIC      SI00&U
SI00&U:
; INIT STATE0
      IN    AL,UART&U+UBUFF   ;GET INPUT BYTE
      CMP   AL,STCHAR&U&_
      JE    SI10&U
      JMP   SERIN0&U          ;IF IT IS NOT THE START CHARACTER, FALL THRU
;CHAR IS START CHAR:
SI10&U:
      MOV   STATE&U&_,OFFSET SD00&U
                              ;CHANGE THE STATE TO GATHER 1ST BYTE,BUF0
      JMP   SERIN0&U          ;AND FALL THRU

; DATA STATE00
SD00&U:
      IN    AL,UART&U+UBUFF   ;GET INPUT BYTE
      MOV   BYTE PTR SHB0&U&_+1,AL  ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SD10&U
      JMP   SERIN0&U          ; FALL THRU
```

```
; DATA STATE10
SD10&U:
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      MOV   BYTE PTR SHB0&U&_+0,AL  ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SD20&U
      JMP   SERIN0&U           ; FALL THRU

; DATA STATE20
SD20&U:
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      MOV   BYTE PTR SHB0&U&_+3,AL  ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SD30&U
      JMP   SERIN0&U           ; FALL THRU

; DATA STATE30
SD30&U:
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      MOV   BYTE PTR SHB0&U&_+2,AL  ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SD40&U
      JMP   SERIN0&U           ; FALL THRU

; DATA STATE40
SD40&U:
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      MOV   BYTE PTR SHB0&U&_+5,AL  ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SD50&U
      JMP   SERIN0&U           ; FALL THRU

; DATA STATE50
SD50&U:
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      MOV   BYTE PTR SHB0&U&_+4,AL  ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SD60&U
      JMP   SERIN0&U           ; FALL THRU

; DATA STATE60
SD60&U:
      IN    AL,UART&U+UBUFF          ;GET INPUT BYTE
      CBW
      MOV   SHB0&U&_+6,AX            ;STORE BYTE THERE

      MOV   STATE&U&_,OFFSET SI01&U
      MOV   STSTATE&U&_,OFFSET SI01&U
      CLI
      MOV   AX,OFFSET SHB0&U&_
      XCHG  CURPTR&U&_,AX
      XCHG  PREVPTR&U&_,AX
      CMP   IBFCNT&U&_,0
      JNE   SE70&U
      MOV   IBFCNT&U&_,2
      STI
      JMP   SERIN0&U           ; FALL THRU
SE70&U:
      MOV   IBFCNT&U&_,4
      STI
      JMP   SERIN0&U           ; FALL THRU
```

```
SI01&U:
; INIT STATE1
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      CMP   AL,STCHAR&U&_
      JE    SI11&U
      JMP   SERIN0&U           ;IF IT IS NOT THE START CHARACTER, FALL THRU
;CHAR IS START CHAR:
SI11&U:
      MOV   STATE&U&_,OFFSET SD01&U
                               ;CHANGE THE STATE TO GATHER 1ST BYTE,BUF1
      JMP   SERIN0&U           ;AND FALL THRU

; DATA STATE01
SD01&U:
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      MOV   BYTE PTR SHB1&U&_+1,AL  ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SD11&U
      JMP   SERIN0&U           ; FALL THRU

; DATA STATE11
SD11&U:
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      MOV   BYTE PTR SHB1&U&_+0,AL  ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SD21&U
      JMP   SERIN0&U           ; FALL THRU

; DATA STATE21
SD21&U:
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      MOV   BYTE PTR SHB1&U&_+3,AL  ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SD31&U
      JMP   SERIN0&U           ; FALL THRU

; DATA STATE31
SD31&U:
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      MOV   BYTE PTR SHB1&U&_+2,AL  ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SD41&U
      JMP   SERIN0&U           ; FALL THRU

; DATA STATE41
SD41&U:
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      MOV   BYTE PTR SHB1&U&_+5,AL  ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SD51&U
      JMP   SERIN0&U           ; FALL THRU

; DATA STATE51
SD51&U:
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      MOV   BYTE PTR SHB1&U&_+4,AL  ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SD61&U
      JMP   SERIN0&U           ; FALL THRU
```

```
; DATA STATE61
SD61&U:
      IN    AL,UART&U+UBUFF          ;GET INPUT BYTE
      CBW
      MOV   SHB1&U&_+6,AX            ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SI02&U
      MOV   STSTATE&U&_,OFFSET SI02&U
      CLI
      MOV   AX,OFFSET SHB1&U&_
      XCHG  CURPTR&U&_,AX
      XCHG  PREVPTR&U&_,AX
      CMP   IBFCNT&U&_,0
      JNE   SE71&U
      MOV   IBFCNT&U&_,2
      STI
      JMP   SERIN0&U           ; FALL THRU
SE71&U:
      MOV   IBFCNT&U&_,4
      STI
      JMP   SERIN0&U           ; FALL THRU

SI02&U:
; INIT STATE2
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      CMP   AL,STCHAR&U&_
      JE    SI12&U
      JMP   SERIN0&U           ;IF IT IS NOT THE START CHARACTER, FALL THRU
;CHAR IS START CHAR:
SI12&U:
      MOV   STATE&U&_,OFFSET SD02&U
                               ;CHANGE THE STATE TO GATHER 1ST BYTE,BUF2
      JMP   SERIN0&U           ;AND FALL THRU

; DATA STATE02
SD02&U:
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      MOV   BYTE PTR SHB2&U&_+1,AL  ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SD12&U
      JMP   SERIN0&U           ; FALL THRU

; DATA STATE12
SD12&U:
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      MOV   BYTE PTR SHB2&U&_+0,AL  ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SD22&U
      JMP   SERIN0&U           ; FALL THRU

; DATA STATE22
SD22&U:
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      MOV   BYTE PTR SHB2&U&_+3,AL  ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SD32&U
      JMP   SERIN0&U           ; FALL THRU

; DATA STATE32
SD32&U:
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      MOV   BYTE PTR SHB2&U&_+2,AL  ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SD42&U
      JMP   SERIN0&U           ; FALL THRU
```

```
; DATA STATE42
SD42&U:
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      MOV   BYTE PTR SHB2&U&_+5,AL  ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SD52&U
      JMP   SERIN0&U           ; FALL THRU

; DATA STATE52
SD52&U:
      IN    AL,UART&U+UBUFF    ;GET INPUT BYTE
      MOV   BYTE PTR SHB2&U&_+4,AL  ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SD62&U
      JMP   SERIN0&U           ; FALL THRU

; DATA STATE62
SD62&U:
      IN    AL,UART&U+UBUFF          ;GET INPUT BYTE
      CBW
      MOV   SHB2&U&_+6,AX            ;STORE BYTE THERE
      MOV   STATE&U&_,OFFSET SI00&U
      MOV   STSTATE&U&_,OFFSET SI00&U
      CLI
      MOV   AX,OFFSET SHB2&U&_
      XCHG  CURPTR&U&_,AX
      XCHG  PREVPTR&U&_,AX
      CMP   IBFCNT&U&_,0
      JNE   SE72&U
      MOV   IBFCNT&U&_,2
      STI
      JMP   SERIN0&U           ; FALL THRU
SE72&U:
      MOV   IBFCNT&U&_,4
      STI
      JMP   SERIN0&U           ; FALL THRU

SERINX&U:
;     CMP   CX,0
;     JLE   CDDONE
;     IN    AL,DUART2+INREG
;     JE    CDDONE
;;    MOV   AX,[BP]
;     MOV   AX,BP
;     OUT   DMAREG,AX
;     DEC   CX
;     ADD   BP,2
CDDONE:
```

```
DATASEG     segment     word public 'data'

      PUBLIC      STATE&U&_
      PUBLIC      STCHAR&U&_
      PUBLIC      STSTATE&U&_
      PUBLIC      EMPBUF&U&_
      PUBLIC      SHB0&U&_
      PUBLIC      SHB1&U&_
      PUBLIC      SHB2&U&_
      PUBLIC      IBFCNT&U&_
      PUBLIC      EMPPTR&U&_
      PUBLIC      CURPTR&U&_
      PUBLIC      PREVPTR&U&_
      PUBLIC      BRATE&U&_
      PUBLIC      PARITY&U&_
      PUBLIC      FRSTCH&U&_
      PUBLIC      SCNDCH&U&_
      PUBLIC      THRDCH&U&_
      PUBLIC      FRTHCH&U&_
      PUBLIC      FRSTOFF&U&_
      PUBLIC      SCNDOFF&U&_
      PUBLIC      THRDOFF&U&_
      PUBLIC      FRTHOFF&U&_

      BSS   STATE&U&_:WORD,2  ;STATE OF INPUT BUFFER FOR UART&U
      BSS   STCHAR&U&_:BYTE,2 ;RECOGNITION CHARACTER TO INDICATE BEGINNING
                  ; OF A DATA STRING
      BSS   STSTATE&U&_:WORD,2      ;STATE OF INPUT BUFFER FOR UART&U
      BSS   EMPBUF&U&_:WORD,SBUFLN+1      ;INPUT BUFFER FOR SER&U
      BSS   SHB0&U&_:WORD,SBUFLN+1 ;HOLDING BUFFER, EXTRA BYTE
      BSS   SHB1&U&_:WORD,SBUFLN+1 ;HOLDING BUFFER, EXTRA BYTE
      BSS   SHB2&U&_:WORD,SBUFLN+1 ;HOLDING BUFFER, EXTRA BYTE
      BSS   IBFCNT&U&_:WORD,2 ;LOAD FLAG
      BSS   EMPPTR&U&_:WORD,2 ;POINTER TO EMPTY BUFFER
      BSS   CURPTR&U&_:WORD,2 ;LOAD FLAG
      BSS   PREVPTR&U&_:WORD,2      ;INPUT BUFFER COUNTDOWN COUNTER
      BSS   BRATE&U&_:BYTE,2  ;BAUD RATE
      BSS   PARITY&U&_:BYTE,2 ;PARITY
      BSS   FRSTCH&U&_:BYTE,2 ;STATE OF FIRST CHANNEL (ON OR OFF
      BSS   SCNDCH&U&_:BYTE,2 ;STATE OF SECOND CHANNEL (ON OR OFF
      BSS   THRDCH&U&_:BYTE,2 ;STATE OF THIRD CHANNEL (ON OR OFF
      BSS   FRTHCH&U&_:BYTE,2 ;STATE OF FOURTH CHANNEL (ON OR OFF
      BSS   FRSTOFF&U&_:BYTE,2      ;OFFSET OF FIRST CHANNEL (ON OR OFF
      BSS   SCNDOFF&U&_:BYTE,2      ;OFFSET OF SECOND CHANNEL (ON OR OFF
      BSS   THRDOFF&U&_:BYTE,2      ;OFFSET OF THIRD CHANNEL (ON OR OFF
      BSS   FRTHOFF&U&_:BYTE,2      ;OFFSET OF FOURTH CHANNEL (ON OR OFF
dataseg     ENDS

      ENDM
```

```
;*********************************************************
;
;     FILE:SERIN0.ASM          (R. CULTER 04/30/91)
;           COPYRIGHT BY CUTLER DIGITAL DESIGN, 1991
;
; TO BE ASSEMBLED AND LINKED BETWEEN DIGIRECA.O AND SERIN8.O
;
;*********************************************************

;****************************************
;
; INLINE SERIAL INPUT CODE FOR CHANNELS 0-7
;
;****************************************
      INCLUDE     DIGIREC.EQU
      INCLUDE     SERIN.MAC

      SERIN 0
      SERIN 1
      SERIN 2
      SERIN 3
      SERIN 4
      SERIN 5
      SERIN 6
      SERIN 7


;*********************************************************
;
;     FILE:SERIN8.ASM          (R. CULTER 04/30/91)
;           COPYRIGHT BY CUTLER DIGITAL DESIGN, 1991
;
; TO BE ASSEMBLED AND LINKED BETWEEN SERIN0.O AND SERINX.O
;
;*********************************************************

;****************************************
;
; INLINE SERIAL INPUT CODE FOR CHANNELS 8-15
;
;****************************************
      INCLUDE     DIGIREC.EQU
      INCLUDE     SERIN.MAC

      SERIN 8
      SERIN 9
      SERIN A
      SERIN B
      SERIN C
      SERIN D
      SERIN E
      SERIN F
```

```
;*********************************************************
;
;     FILE:SERINX.ASM          (R. CULTER 04/30/91)
;          COPYRIGHT BY CUTLER DIGITAL DESIGN, 1991
;
; TO BE ASSEMBLED AND LINKED AFTER SERIN8.O
;
;*********************************************************

; V40 and local equates
     INCLUDE     DIGIREC.EQU
;**************************************
;
;     EXIT FROM MILLISECOND INTERRUPT
;
;**************************************
DATASEG     SEGMENT     WORD PUBLIC 'data'

      EXTRN MSINTFLG:BYTE
      EXTRN DMAPTR_:WORD
      EXTRN DMACNT_:WORD

DATASEG     ENDS
      ASSUME      DS:DATASEG

      MOV   MSINTFLG,0
      MOV   DMAPTR_,BP
      MOV   DMACNT_,CX
MSX:
      POP   CX
      POP   BP
      POP   BX
      POP   AX
      IRET
```

```
file: stksiz.c

#if sizeof(char *) == 4
int _STKSIZ = 8192/16;  /* (in paragraphs) (large data model) */
int _STKRED = 1024;           /* size of RED zone (in bytes) */
#else
int _STKSIZ = 4096/16;  /* (in paragraphs) */
int _STKRED = 2048;           /* size of RED zone (in bytes) */
#endif

int _HEAPSIZ = 4096/16; /* (in paragraphs) */
int _STKLOW = 0;        /* default is stack above heap (small only) */

/*
NOTE: The RED zone is used for stack saftey checking.  With stack above heap,
the heap will not be allowed to get within STKRED bytes of the current value
of the SP, if limit checking is enabled (see the cc +b option), SP isn't
allowed any closer than STKRED bytes to the top of the heap.  If stack below
heap and limit checking is enabled, SP isn't allowed any closer than STKRED
bytes to _Utop.  Minimum stack size = 2*_STKRED.  Minimum value for STKRED
should be about 256 bytes.  This allows some margin to issue DOS calls and
allow interrupt handlers to execute.  (Some people think that this should
be > 1k.) */
```

Additions to Xdetect for DIGIREC

Introduction

The DIGIREC circuit was envisioned as one of the necessary parts in the transformation of an existing analog seismic data acquisition system, to a digital seismic data acquisition system. The existing software is called XDETECT, and runs under MS DOS. Completing the development of DIGIREC involved fitting it into the XDETECT environment.

At this time there is a version of XDETECT that uses the DIGIREC board, rather than a A/D board, as the source of its data. There are no known problems, though it has not been exercised fully, and there are many features that have not been implemented.

What follows is a brief description of the software that now exists to link DIGIREC to XDETECT, and the source listings.

Development Tools

As with the original Xdetect, the new program was produced using the Microsoft C (version 5.1) compiler, and the corresponding assembler and linker.

Program Description

Modifications to XDETECT

There are two major areas where XDETECT was changed. 1) All access to data comes through MDEMUX, rather than MDT28XX. 2) There are two new global flags that are set by start up switches, to dynamically decide parity for the serial ports, and to dynamically correct the frequency of the DIGIREC sample clock.

Modifications to MDEMUX

The major change to MDEMUX involves the addition of an internal flag that is used by all data access software to determine the source of all data. All routines that originally accessed the MDT28XX routines, now switch between MDT28XX routines or MDIGIREC routines, based on the value of the internal flag. It is not intended that his flag will change during execution, other than at startup time.

In future implementations, it should be possible to have both DIGIREC and DT28XX as concurrent sources of data.

Modifications to other modules

Throughout the various modules that comprise XDETECT there were references to MDT28XX. All of these have been redirected to MDEMUX routines.

In a few cases, constants were assumed, or taken from the MDT28XX.H file. These also have been redirected to MDEMUX calls or constants.

MDIGIREC

MDIGIREC.C is a collection of routines that mimic the MDT28XX routines, but use the digital receiver board DIGIREC, instead of an A/D board. The names of the routines match those in MDT28XX. But the calls to support routines do not use the Data Translation library of routines. In their place, a new set of routines have been written, customized to the DIGIREC board features. These routines are collected in MDIGISUP.C.

MDIGISUP

These are support routines for MDIGIREC.C. They include extended memory allocation, data transfer throughout the greater address space, DMA and interrupt processing, and DIGIREC specific functions.

MDIGIASM

These assembly language routines are provided to invoke the proper C routine at interrupt time, and to perform certain functions efficiently.

MERROR

DIGIREC error messages have been added to the existing error processing structure.

Future Directions

BETTER CONTROL OF DATA STREAMS: It is possible to have an particular remote instrument appear in any position in the data record that is captured each sample period. This will be valuable, for efficient storage management. Ultimately, the user should be able to define the storage map in the input file to XDETECT. Currently, it is fixed.

BETTER CONTROL OF INDIVIDUAL UARTS. It is possible to set baud rate, parity and synchronization character individually for each UART. This will be important when there are different remote digitizers feeding into the same system. Currently, all UARTs are set to the same value.

MISSING DATA. Currently, a missing data point is filled with the previous value for missing data. It is possible to have a fixed value that is used instead. This fixed value can be different for each of the 64 streams of data.

SUDS STRUCTURE. The current software has piggybacked onto the SUDS structure for the DT28XX boards. A new board selector value is used to identify DIGIREC, but otherwise no changes were made. This is not optimal, rather a new structure should be defined that truly reflects the nature of the DIGIREC.

CONCURRENT USE OF DIGIREC AND DT28XX. Technically, it is possible to operate the DIGIREC board and an A/D board at the same time. The software implications, however, are complex.

ADDITIONAL USE OF DIGIREC EAPROM. The non-volatile memory on the DIGIREC is currently used to store the clock correction factor. There is room to store other configuration information as well.

Conclusion

DIGIREC has been successfully incorporated into the XDETECT environment. Progress in the development of the whole system is now waiting on feed back from field use.

Additions to Xdetect for DIGIREC

Appendix - Xdetect Source Listings

```
/* FILE: xdetect.h                                    (R. Cutler 04/30/91)
                                                  (D. Tottingham  01/26/91)

This is an include file of the defines, data structure definitions and
external data declarations used by the xdetect program.

*/

#ifndef _XDETECT_
#define _XDETECT_

/*************************************************************************
                                INCLUDES

*************************************************************************/
#include "mconst.h"


/*************************************************************************
                                DEFINES

*************************************************************************/
#define REVISION_DATE           "04/30/91"
#define REVISION_NAME           "XDETECT2001"
#define VERSIONNUM              2.001

/* Input file */
#define INPUT_FILENAME          "XDETECT.INP"

/* Enable Flags */
#define DEBUG_ENABLED           FALSE
#define TIME_SET        FALSE

/* Keyboard Constants */
#define CNTRL_B                 0x3002
#define CNTRL_F                 0x2106
#define CNTRL_HOME              0x7700
#define CNTRL_L                 0x260c
#define CNTRL_PGDOWN            0x7600
#define CNTRL_PGUP              0x8400
#define CNTRL_Q                 0x1011
#define CNTRL_R                 0x1312
#define CNTRL_T                 0x1414
#define F1                      0x3b00
#define DOWN_ARROW              0x5000
#define UP_ARROW                0x4800


/*************************************************************************
                                GLOBALS

*************************************************************************/
PUBLIC FLAG Debug_enabled;
PUBLIC FLAG Time_set;
PUBLIC FLAG Global_parity;

#endif
```

```
/* FILE: xdetect.c                                    (R. Cutler 04/30/91)
                                                   (D. Tottingham  01/26/91)

This is the driver for xdetect, the multi-channel event detection and data
collection program.  The program has been written and compiled medium model.

EXTERNAL FUNCTIONS CALLED:

dm_dump_configuration ()          dump the DT2821 configuration
dm_get_channel_size ()            get channel block size
dm_get_number_of_ext_buffers ()   get number of external buffers
dm_get_scan_count ()              get scan count
dm_initialize ()                  initialize the queues
dm_initialize_collect_params ()   initialize parameters for data acq
dm_initialize_collection ()       initialize the data acquisition unit
dm_initialize_params ()           initialize parameters for demux
dm_initialize_time ()             initialize the time
dm_reset ()                       reset this module and release all storage
dm_start_collection ()            start continuous data acquisition
dm_stop_collection ()             stop continuous data acquisition
dsp_calibration_detected ()       has a calibration been detected?
dsp_calibration_done ()           are we still saving the calibration?
dsp_get_dsp_status ()             get dsp_enabled
dsp_domain_check ()               look for a calibration signal
dsp_initialize ()                 initialize the dsp module
dsp_initialize_params ()          initialize key parameters
e_initialize ()                   initialize event parameters
e_get_bell_status ()              get bell_enabled
e_ring_bell ()                    ring event alert bell if enabled
e_toggle_bell_status ()           toggle bell_enabled flag
er_abort ()                       display an error message then quit
f_check_pathname ()               check pathname for validity
f_initialize_index ()             initialize the index file
f_initialize_params ()            initialize key parameters
fr_initialize ()                  initialize free run parameters
fr_toggle_freerun ()              toggle free run event
h_initialize_params ()            initialize a screen header
h_toggle_status ()                toggle a status bit
l_display_location ()             write location to stream
l_get_location_status ()          get location_enabled
l_initialize ()                   initialize location module
l_locate_event ()                 locate the event
l_reset_location ()               reset the location flag
l_toggle_location_status ()       toggle location_enabled flag
o_check_pathname ()               check log pathname for validity
o_initialize_params ()            initialize log pathname
o_open_logfile ()                 open a log file for output
o_write_logfile ()                write string to log file stream
p_initialize ()                   initialize parser
p_parse_commands ()               parse commands in input stream
pk_initialize ()                  initialize pick module
pk_pick_arrivals ()               pick arrivals using trigger information
pk_pick_magnitudes ()             pick magnitudes
pk_reset_picks ()                 reset pick queues
r_check_time ()                   compare time with reboot time
r_get_reboot_status ()            get reboot_enabled
r_get_reboot_time ()              get reboot time
r_initialize_reboot ()            initialize reboot structure
r_toggle_reboot_status ()         toggle reboot_enabled flag
s_initialize ()                   initialize the screen module
s_initialize_screen ()            initialize the graphics screen
s_display_ids ()                  display station id, channel, and trigger status
s_display_traces ()               display normalized traces on the screen
s_display_triggers ()             display triggers on the screen
s_reset_screen ()                 reset the monochrome screen to text
s_select_display ()               select the display mode
```

```
s_set_channelrange ()          set the channel range in the block display
st_initialize_params ()        initialize station queue
st_initialize_stations ()      initialize station list
t_display_triggers ()          write event information to stream
t_get_trigger_status ()        get trigger_enabled
t_initialize_event ()          initialize the event structure
t_initialize_trigger ()        initialize the trigger structure
t_reset_trigger ()             reset trigger variables
t_toggle_trigger_status ()     toggle trigger_enabled flag
t_trigger_check ()             look for new triggers
u_build_timeline ()            convert abs. time to an ascii string
u_ispow2 ()                    determine whether i_num is a power of two
u_tzset_GMT ()                 set timezone to GMT
u_timestamp ()                 get timestamp
```

HISTORY:

(04/30/91) o Rewritten to have all access to data acquisition through
V2.001 mdemux.c. This will allow different data input modules (in particular the digital receiver board - digirec) to be used without requiring universal changes to code.
All references to dt_ routines have been replaced with dm_ routines. The one dt_ data structure, DT_28XX, has been renamed DM_INFO, and move to mdemux.h.
(Reese T. Cutler)

(01/26/91) o Fixed bug in pwrlolv1.asm which was causing intermittent
V1.99 'Invalid PowrSTOR address' errors. The function putSTOR would call chkAddr to determine whether buffer overlapped a segment boundary. To determine this overlap, the chkAddr function added the length of the buffer to the base address and then looked at the overflow flag. Using the overflow flag meant that any buffer overlapping the middle of the segment was in error. The carry bit is set up for just this purpose. If the result exceeds 0xffff, the carry flag is set. So I changed a JNO instruction to a JNC instruction.

(12/26/90) o Created the mfreerun module to handle/manage free run events.
V1.98 The FreerunBlockTime parameter was added to the input file to allow the user to block free run events into smaller, manageable events.

o Created the mevent module to act as a sophisticated event manager. As of now, the event manager just manages the event alert bell.

o Replaced the x_set_bell_status(), x_set_reboot_status(), x_set_locate_status(), and x_set_trigger_status() routines in xdetect.c with appropriate routines in the mevent, mreboot, mlocate, and mtrigger modules respectively. Eliminated the bell_enabled, reboot_enabled, location_enabled, and trigger_enabled flags from xdetect.c.

o Miscellaneous cleanup.

(12/02/90) o Fixed bug in dm_initialize(). Due to the fact that PowerSTOR
V1.97 allocates memory in 16K byte chunks, changed the way that dm_initialize() calculates the number of 16K byte chunks in pre-trigger queue.

(11/02/90) V1.96
- Made changes to mdemux.c to support larger pre-trigger queues. In particular, added code to move the pre-trigger queue to extended memory using the PowerSTOR subroutine library.
- Fixed a bug in the rms residual calculation.
- Changed the way the mscreen module allocates memory on the far heap. We now assume that 32 channels (MAX_DISPLAY) will be displayed. What would happen is that we would initially allocate storage for 32 channels in LIST mode, deallocate the storage when the user switched to BLOCK mode, and then try to reallocate the storage when the user switched back to LIST mode. Unfortunately, while the user was in BLOCK mode, the additional storage used in LIST mode would be taken up by other processing thereby fragmenting the far heap. XDETECT would then crash because it could not reallocate the storage. This solution will make do until we come up with a better way to manage the far heap.

(09/29/90) V1.95
- Added rms residual calculation to hs_locate_event(). Added rms residual to l_display_location().
- Added functionality to dt_set_channel() and dt_set_ChannelGain() to make sure that the gain is 1, 2, 4, or 8.
- Added functionality to u_ispow2() and u_log2() to make sure that they only operate on positive, non-zero numbers.

(09/08/90) V1.95
- Fixed bug in u_get_diskfree(). The default drive was not being recognized.
- Fixed bug in s_display_traces(). Negative data values were being scaled (amplified/attenuated) incorrectly resulting in spurious DC offsets.
- Error messages are now routed to the log file. Log file path now defaults to the current working directory. Alternate log file path names (defined in the input file) are now ignored, and the appropriate warning message is displayed. This is due to the fact that warning and/or error messages can be generated before reaching the log file path name definition in the input file resulting in two separate log files, one in the current working directory and the other in the log file path defined in the input file.

(03/30/90) V1.94
- Added the STAverageWindow and LTAverageWindow parameters to the input file. These parameters set the number of samples within the short-term average and long-term average (within the trigger algorithm) respectively.
- Eliminated the NumberOfExtBuffers parameter from the input file. The mdt28xx module adds buffers until it gets an error from ATLAB.
- ChannelBlocksize now defaults to 32768 / # of channels in station list.
- Added timestamp to entries in log file.
- Added error numbers to error messages.

(12/29/89) V1.93

o Renamed msuds.c msudsini.c. msudsini.h includes suds.h which defines all suds structures. suds.h is centrally located in \msc51\include.

o Finished integration of mdsp module into xdetect. Added CalibrationRecording, BandRecording, and BandLimits keywords to mlexer.h and mlexer.c. Added the following functionality to mparse.c:

```
CalibrationRecording= [ON | OFF];
BandRecording= [ON | OFF][; | , BandLimits= { (a, b),...,(a, b) };
```

Removed DSPEnabled keyword. If calibration recording is on and band recording is on, mparse sets dsp_enabled to TRUE in xdetect.c via x_set_dsp_status().

Added H_RECORD_CALIBRATION bit field to mscrnhdr.h. When a calibration is detected and is being saved, mscrnhdr displays "Recording Calibration" on the status line on top of the screen.

Added dsp_set_band() and dsp_initialize_params() to mdsp module. dsp_set_band() accepts a band specification (high and low freq. cutoffs) and puts is into a new BAND structure in the array of BAND structures. dsp_initialize_params() initializes the BAND structure array and operation parameters.

Modified f_write_buffers() so that the event index is only incremented for events not calibrations.

Cleaned up mdsp module. Generated function documentation, alphabetized public mdsp functions, and rewrote small pieces of inadequate code. Added error checking interfacing with the merror module. Wrote init_tms320() to look for a TMS32xxx board.

o Changed keywords and functionality of the following:

```
BellEnabled= [TRUE | FALSE]          to  EventAlertBell= [ON | OFF]
MonitoringEnabled= [TRUE | FALSE]    to  Autotriggering= [ON | OFF]
LocationEnabled= [TRUE | FALSE]      to  AutoLocation= [ON | OFF]

RebootEnabled= [TRUE | FALSE] and
RebootTime= [xx:xx:xx]               to
     Autoreboot= [ON | OFF] [; | , Time= [xx:xx:xx] ]
```

o Changed status line text in mscrnhdr.c as follows:

```
"Monitoring Enabled"   to   "Autotriggering"
"Monitoring Disabled"  to   "Asleep"
```

o Fixed coda duration based magnitude problem. Instead of looking for the first sample in which signal/noise < 2.0, look for first buffer in which maximum signal/noise < 2.0. Modified process_coda() in mpick. Eliminated signal calculation in triggered_on_channel() in mtrigger.

o Added XDETECT revision date and number to log file.

o Added memory_dump flag to ER_MESSAGE handler in merror.h. Added appropriate memory dump flags to each error message in merror.c. For every error, er_abort checks the memory dump flag in the error's ER_MESSAGE structure. If it is set, dumps of both the near and far heap are directed to dumpheap.@@@ for future debugging.

(11/24/89) V1.92
- o Reduced minimum noise (NOISE_INIT) from 30 to 1 digital counts in mtrigger.h to get a more accurate magnitude determination.
- o Removed _dos_getdiskfree() and _dos_getdrive() from mdosfunc library and changed mdosfunc.h accordingly. Replaced DISKFREE_T with struct diskfree_t in u_get_diskspace() in mutils.c.
- o Replaced mevt and mevtdrv modules with mdsp module and improved the DSP interface. Replaced x_set_fft_status() with x_set_dsp_status() and fft_enabled with dsp_enabled. Added dsp control code to process_buffer(). Removed fft code from post_trigger_analysis(). Added MaxCalibrationTime to input file.
- o Added f_get_pathname() to mfile module.

(09/15/89) V1.91
- o Added picker module (mpick.c). This picker has two primary functions: pick arrivals and pick coda durations. The pick arrival function simply takes a linked list of SUDS_TRIGGER structures from the trigger module and converts them into a linked list of SUDS_FEATURE (obs_phase = 50) structures. The pick coda duration function determines the coda duration for each channel that has an arrival pick based on the signal-to-noise ratio.
- o Replaced the hypocenter location function with a halfspace function. The mlocate module is still the driver for the location algorithm, but it now supports the SUDS_ORIGIN structure and maintains a linked list of the last ten locations. This latter ability will be essential for plotting epicenter maps in real-time.
- o Added the mlatlon module to handle all converts to/from latlon from/to xy.
- o Added a vertical scroll feature to the trace display screen which allows the user to view selected 16 channel chunks in detail. You can enter the scroll mode by pressing either CNTRL-PGUP or CNTRL-PGDN. CNTRL-PGUP and CNTRL-PGDN display the previous 16 channels and the next 16 channels respectively. Hitting CNTRL-HOME gets you back to the channels selected in the input file. While you are in scroll mode, a scroll bar appears on the left side of the screen indicating which channels you are viewing with respect to the entire channel list.
- o Added a help screen. Press F1 to display the help screen and any key to return to the trace display screen.

(08/24/89) V1.90

- Changed file format to SUDS (Seismic Unified Data System) which was developed by Peter Wards. Developed to be modular in structure, SUDS is flexible enough to handle as many or as few channels as the user needs. In addition, modules can be easily added and modified without damaging the integrity of the file structure. SUDS is now really the backbone of xdetect.

- Added write_to_buffer() routine to the mfile module to buffer SUDS structures to disk. Compared to the unbuffered approach, this method gives us a 100% speed improvement.

- Modified s_display_traces() to scroll when buffers are less than 512 samples in size. We can use this function later on to let the user specify a decimation factor in real-time!!

- Changed the input file slightly.
  a. Changed the station definition as such:
     Ch= , StName= , Component= [, Trigger=][, Display=][, Gain=];
     The brackets indicate optional parameters. The order of the first two parameters is critical, and each parameter MUST be separated by a comma!!! Default trigger is ON, default display is OFF, and default gain is either ChannelGain or 4 (CHANNEL_GAIN) if ChannelGain is not defined.

  b. Added NetworkName. Specifies the network operating the stations defined in the station list. XDETECT can only support one network!!!

  c. Added NetworkNodeId. Allows you to have multiple computers monitor a single network. This is simply an id that is pasted to every output file to uniquely identify it from other files generated by other computers in the network. The first character of the NetworkNodeId is used as the network code and is appended to the filetype (i.e. .WVx where x is the network code).

  d. Added ChannelGain. The gain parameter in the station definition is only valid on channels 0-15, and these are only valid if less than 17 channels are defined. Otherwise, all gains are set to ChannelGain. If ChannelGain is not given, then the default channel gain is used (CHANNEL_GAIN = 4).

  e. Changed ChannelBufferSize to ChannelBlocksize.

- Added support for a station file, a file of SUDS_STATIONCOMP structures. Each station has a large amount of information that should be written to every event file. Due to the volume and static nature of this data, forcing the user to enter all this information for each station in the input file seems meaningless. By supporting a separate, binary file of this station data, xdetect only expects a network name, station name, and component information for each station. xdetect then finds the corresponding SUDS_STATIONCOMP structure in the station file for each station defined in the input file. If the station is not found, xdetect creates a dummy SUDS_STATIONCOMP and fills it with default/dummy values. Created STPARSE so that users can define SUDS_STATIONCOMP structures in ascii and then parse them into binary suds structures (see stparse.c).

- Changed PreEventTime, MinEventTime, MaxEventTime, and TriggerTimeLimit to floats in order to support < 1 second intervals!!!

- Rewrote merror to make it easier to add/remove error messages from the error message table. Improved error handling for ATLAB: Check to see if card is in machine and if atldrv is installed. Make sure that low frequency and high frequency bounds are not crossed.

- Made HYPO20 into a library. Support MS Fortran V5.0.

- Timer module now supports four independent timers.

(08/01/89) V1.81

- Revamped the MTRIGGER module. Set up two queues, one for channel information and one for trigger information. Made changes to MQUEUE to handle a queue of anything (see Q_TYPE union). Eliminated the clumsy array handling and replaced it with linked list manipulation. Channels can now have multiple triggers. The oldest trigger remains active until it expires at which time the next oldest trigger becomes active.

- Added a display queue to hold channels which are to be displayed. Can handle up to 32 channels/screen. Added a display parameter to the station definition in the input file to allow the user to select which 32 channels he wants to display. Tested software and demonstrated 64 channel capability.

- Changed screen layout. Display with plotmod so we don't have to clear the screen for every buffer. Added real-time amplify/attenuate feature as implemented in XPLAY.

(07/28/89) V1.80

- Fixed the 32Kbyte limitation for internal buffers by changing the ATLAB device driver. Basically, internal buffers do not have to meet DMA requirements because they are never accessed by DMA. However, because ATLAB allows you to have a buffer transfer list either in base memory or extended memory, Data Translation designed AL_DECLARE_BUFFER assuming that you will always have a local BTL. I commented out two lines from aldthr.icl so this test never occurs.

- Renamed the MQUEUE module MDEMUX and moved all the queue manipulation routines to MQUEUE. Set up a queue of internal mux buffers (my own BTL) each capable of holding 32Kbytes. By doing this, I expanded the virtual buffer size to 64Kbytes (ATLAB limitation).

- Changed the screen layout slightly in order to support all three monitors (Hercules, EGA, and VGA). Now we support all three monitors. Created three .LNK files, one for each monitor.

(07/20/89) V1.71

- Migrated code to Microsoft C V5.1. The .EXE file is now down to 133Kbytes (from 233Kb), and execution speed has nearly doubled.

- Added code to XDETECT.C to time acquisition cycle and to compute an approximate maximum digitization frequency. XDETECT displays this approximate maximum digitization frequency on the screen just before exiting the program.

(07/14/89) V1.70 o Source code has been completely reorganized into object oriented modules. Each module has marked its public functions with a unique 1 or 2 letter code. For the most part, the code is simply the first letter of the module name (i.e. t_ prefixes all mtrigger routines). See filemap.xxx for the code-module mapping.

o Fixed the Control-Q problem. In Version 1.6 and earlier, a user could quit the program at any time simply by pressing CNTRL-Q. This caused problems when the program was recording an event (either triggered or in free run). Now, a CNTRL-Q is interpreted as a flag, and the flag is not checked until after xdetect has finished recording the event.

o Added Control-B. This enables/disables the warning bell. Added BellEnabled token to control file.

o Eliminated the ascii .IXM files and replaced them with a single binary INDEX.@@@ file. To minimize problems in the future, this file is OFF LIMITS to post-processing programs. The format of the file is currently:

```
day             (2 bytes)
month           (2 bytes)
year            (2 bytes)
index high word (2 bytes)
index low word  (2 bytes)
```

o The screen header has been changed as follows:

a. The event counter is reset at the beginning of each session.
b. Changed "Event Not Detected" to "Monitoring Enabled"
c. Changed "Event DETECTED" to "Recording Event"
d. Changed "Trigger Disabled" to "Monitoring Disabled"

o The channel numbers are now displayed in decimal instead of hexadecimal.

o Replaced the clumsy input file handler with a sophisticated parser/lexer. This allows full syntax checking. In order to use this parser, the input file syntax had to be changed slightly which makes it incompatible with previous versions. The changes are slight and are as follows:

a. Any non-alphanumeric character string must be placed in double quotes (i.e. StationId = "CH_1"; the underscore makes the string non-alphanumeric). This rule affects StationID, PathName, and LogPathName.
b. The station definition must have the following syntax:
   StationId= , Ch= [, Gain=][, Trigger=];
   The brackets indicate optional parameters.
   The order of the first two parameters is critical, and each parameter MUST be separated by a comma!!!
c. Any non-alphabetic character constant must be placed in single quotes (i.e. NetworkCode= '0'; the 0 is non-alphabetic). This rule only affects NetworkCode.

```
        o Fixed the greenwich mean time problem.  In Version 1.6 and earlier,
          xdetect assumed that users were operating the system under local
          time.  Unfortunately, Microsoft coded their run-time library time
          routines in such a way that daylight savings time is automatically
          set/reset at undocumented, predefined dates in the Spring/Fall.
          xdetect now assumes that users are operating the system under GMT
          without daylight savings time.  Simply trick the Microsoft RTL into
          thinking that the system is at GMT by setting the global variables
          timezone = 0, daylight = 0, and tzname[0] = "GMT".

          o Changed TriggerEnabled to MonitorEnabled in control file.

          o Changed Control-T to Control-M, added Control-H to toggle HYPO,
            added Control-R to toggle REBOOT, and eliminated Control-P.

*/


/*****************************************************************************
                              INCLUDE FILES

*****************************************************************************/
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>
#include <atldefs.h>

#include "mconst.h"
#include "mdemux.h"
#include "mdsp.h"
#include "merror.h"
#include "mevent.h"
#include "mfile.h"
#include "mfreerun.h"
#include "mlocate.h"
#include "mlog.h"
#include "mparse.h"
#include "mpick.h"
#include "mreboot.h"
#include "mscreen.h"
#include "mscrnhdr.h"
#include "mstation.h"
#include "mtrigger.h"
#include "mutils.h"
#include "xdetect.h"
#include "timer.h"


/*****************************************************************************
                              GLOBAL DATA

*****************************************************************************/
FLAG Debug_enabled = DEBUG_ENABLED;             /* debug on/off */
FLAG Time_set = TIME_SET;
FLAG Global_parity = -1;
PRIVATE FLAG quit_enabled = FALSE;              /* quit flag */
PRIVATE unsigned int display_type;

PRIVATE char out_str[85];
double min_time, max_time, avg_E;
```

```
/*****************************************************************************
                          FUNCTION DECLARATIONS

*****************************************************************************/
PRIVATE void quit ();


/*===========================================================================*
 *                          initialize_defaults                              *
 *===========================================================================*/
/* Initialize defaults parameters.                                           */

PRIVATE
void initialize_defaults ()
{
   u_tzset_GMT ();
   st_initialize_params();
   dm_initialize_params();
   dsp_initialize_params();
   dm_initialize_collect_params();
   e_initialize ();
   f_initialize_params();
   fr_initialize();
   h_initialize_params();
   l_initialize();
   o_initialize_params();
   p_initialize();
   pk_initialize();
   r_initialize_reboot();
   s_initialize();
   t_initialize_event();
   t_initialize_trigger();
}

/*===========================================================================*
 *                              process_input                                *
 *===========================================================================*/
/* Process the input file.                                                   */

PRIVATE
void process_input (argc, argv)
int argc;
char * argv[];
{
   static char input_fn [MAX_FILENAME_LENGTH];
   FILE * inp_stream;
   FLAG filename_found;

   /* Search for command line options and the filename */
   filename_found = FALSE;
   while (--argc) {
      switch (argv[argc][0]) {
         case '/':
            switch (toupper(argv[argc][1])) {
               case 'D':
                  Debug_enabled = TRUE;
                  break;
               /* room for other slash options */

            case 'N':
                  Debug_enabled = FALSE;
                  break;
            case 'T':
                  Time_set = TRUE;
                  break;
```

```
            case 'E':
                  Global_parity=1;
                  break;
            case 'O':
                  Global_parity=-1;
                  break;
            case '0':
                  Global_parity=0;
                  break;
            }
            break;
         default:
            strcpy (&input_fn[2], argv[argc]);
            filename_found = TRUE;
      }
   }

   if (!filename_found) {
      printf ("Enter control file [<CR> = %s]  >> ", INPUT_FILENAME);

      input_fn[0] = MAX_FILENAME_LENGTH - 5;
      cgets (input_fn);
      printf ("\n\n");
      if (! strcmp (&input_fn[2], "\0"))
         strcpy (&input_fn[2], INPUT_FILENAME);
   }

   o_write_logfile ("<<<<<<<<<<<<<<<<<<<<<   N E W   S E S S I O N
>>>>>>>>>>>>>>>>>>>>>\n");
   sprintf (out_str, "XDETECT Version %5.2f, Last Revision %s\n", VERSIONNUM,
REVISION_DATE);
   o_write_logfile (out_str);
   sprintf (out_str, "INPUT FILE:  %s\n\n", &input_fn[2]);
   o_write_logfile (out_str);

   /* If the input file exists, process the contents */
   if (inp_stream = fopen (&input_fn[2], "r")) {
      p_parse_commands (inp_stream);
      fclose (inp_stream);
   } else er_abort (X_INVALID_INPUTFILE);
}
```

```
/*=============================================================================*
 *                              display_setup                                  *
 *=============================================================================*/
/* Display setup information.                                                  */

PRIVATE
void display_setup ()
{
   /* Print out some key flags and values */
   printf ("\nBell      DSP       Location  Trigger   Reboot    Reboot time   Clk
source\n");
   o_write_logfile ("Bell      DSP       Location  Trigger   Reboot    Reboot time
Clk source\n");
   sprintf (out_str, "%-10s%-10s%-10s%-10s%-10s%-8s      %-14s\n\n",
           ((e_get_bell_status())       ? "ENABLED"  : "DISABLED"),
           ((dsp_get_dsp_status())      ? "ENABLED"  : "DISABLED"),
           ((l_get_location_status())   ? "ENABLED"  : "DISABLED"),
           ((t_get_trigger_status())    ? "ENABLED"  : "DISABLED"),
           ((r_get_reboot_status())     ? "ENABLED"  : "DISABLED"),
           r_get_reboot_time(),
           ((dm_get_clock_source() == 'i') ? "INTERNAL" : "EXTERNAL"));

   printf (out_str);
   o_write_logfile (out_str);
   sprintf (out_str, "Number of external buffers = %d\n\n",
dm_get_number_of_ext_buffers());
   printf (out_str);
   o_write_logfile (out_str);
}

/*=============================================================================*
 *                              check_pathnames                                *
 *=============================================================================*/
/* Check pathnames.                                                            */

PRIVATE
void check_pathnames ()
{
   o_check_pathname();
   f_check_pathname();
}

/*=============================================================================*
 *                              initialize_system                              *
 *=============================================================================*/
/* Initialize system parameters.                                               */

PRIVATE
void initialize_system ()
{
   /* Initialize the station list */
   st_initialize_stations ();

   /* Initialize the DSP */
   dsp_initialize();

   /* Initialize the data acquisition unit */
   dm_initialize_collection ();

   /* Allocate internal and external multiplexed buffers */
   dm_initialize_buffers();

   /* Initialize the demux queue.  Setup the PowerSTOR */
   dm_initialize ();
```

```
  /* Start data acquisition */
   dm_start_collection ();

   /* Display the current setup */
   display_setup ();

   /* Initialize the screen and screen header */
   s_initialize_screen ();
   dm_initialize_time ();

   /* Initialize the index file */
   f_initialize_index ();

   /* Initialize the ids */
   display_type = HOME;
   s_select_display (HOME);
   s_display_ids ();
}

/*=============================================================================*
 *                          post_trigger_analysis                              *
 *=============================================================================*/

PRIVATE
void post_trigger_analysis ()
{
   FILE * log_stream;

   /* Display triggers on screen. */
   if (Debug_enabled)
      t_display_triggers (stdout);
   if (LOG_FILE) {
      log_stream = o_open_logfile ();
      t_display_triggers (log_stream);
      fclose (log_stream);
   }
}

/*=============================================================================*
 *                              process_buffer                                 *
 *=============================================================================*/
/* Main processing loop.                                                       */

PRIVATE
void process_buffer ()
{
   FILE * log_stream;
   double dt;

   /* Do we have a buffer to process */
   if (! dm_buffer_full())
      return;

   /* Main processing loop */
   set_starttime(TIMER_0);
   dm_get_new_buffers();
   s_display_traces (0, 0);
```

```
if (dsp_calibration_detected()) {
   if (dsp_calibration_done()) {
     f_write_buffers (END_FILE, F_CALIBRATION);
     h_toggle_status (H_RECORD_CALIBRATION);
   }
   else f_write_buffers (CONTINUE_FILE, F_CALIBRATION);

} else if (t_event_detected()) {
   if (t_event_done ()) {
      /* Event is over.  Locate the event */
      if (l_locate_event()) {
         if (LOG_FILE) {
            log_stream = o_open_logfile ();
            l_display_location (log_stream);
            fclose (log_stream);
         }
         if (Debug_enabled) l_display_location (stdout);
      }
      f_write_buffers (END_FILE, F_EARTHQUAKE);
      l_reset_location ();
      pk_reset_picks ();
      t_reset_trigger ();
      h_toggle_status (H_RECORD_EVENT);
      if (Debug_enabled)
         printf ("Data Acquisition continued...\n");

   } else {
      f_write_buffers (CONTINUE_FILE, F_EARTHQUAKE);
      pk_pick_magnitudes ();
   }

} else if (fr_continue_freerun()) {

} else if (quit_enabled) {
     quit();

} else if (r_check_time()) {

} else if (dsp_domain_check()) {
   if (Debug_enabled)
      printf ("Calibration found and being saved...\n");
   h_toggle_status (H_RECORD_CALIBRATION);
   f_write_buffers (START_FILE, F_CALIBRATION);

} else if (t_trigger_check ()) {
   e_ring_bell ();

   /* Setup to collect eq data */
   h_toggle_status (H_RECORD_EVENT);

   /* Pick the p-arrivals and magnitudes */
   pk_pick_arrivals ();
   pk_pick_magnitudes ();

   /* Do some post-trigger processing at this point */
   post_trigger_analysis ();

   /* Write pre-trigger data to disk */
   f_write_buffers (START_FILE, F_EARTHQUAKE);
}
```

```
   set_stoptime(TIMER_0);
   dt = diff_time(TIMER_0);
   if (dt < min_time) min_time = dt;
   if (dt > max_time) max_time = dt;
   if (t_event_detected() /* || free_run_flag */)
      avg_E += (dt - avg_E) / dm_get_number_of_ext_buffers();
}

/*=============================================================================*
 *                                  quit                                       *
 *=============================================================================*/
/* Reset screen to text.  Quit xdetect.                                        */

PRIVATE
void quit ()
{
   double avg_time, max_sps;

   dm_stop_collection ();
   s_reset_screen ();
   dm_reset();

   printf ("\n\nMinimum time              = %lf sec.\n", min_time);
   printf ("Maximum time              = %lf sec.\n", max_time);
   if (avg_E) {
      max_sps = dm_get_channel_size() / avg_E;
      printf ("Average event time         = %lf sec.\n", avg_E);
      printf ("Maximum digitization rate  = %lf sps.\n", max_sps);
   }

   exit ( 1);
}
```

```
/*=============================================================*
 *                          m a i n                            *
 *=============================================================*/
main (argc, argv)
int argc;
char * argv[];
{
   FILE * log_stream;
   char * timeline;
   int c;

   s_reset_screen ();

   /* Opening banner */
   printf ("\n\n%s (%s)    Tottingham\n", REVISION_NAME, REVISION_DATE);
   printf ("\nMulti-Channel Event Detection and Collection Program\n\n");

   initialize_defaults ();

   process_input (argc, argv);

   check_pathnames ();

   initialize_system ();

   /* Display the current unit configuration */
   if (Debug_enabled)
      dm_dump_configuration (stdout);
   if (LOG_FILE) {
      log_stream = o_open_logfile();
      dm_dump_configuration (log_stream);
      fclose (log_stream);
   }

   min_time = 10000;  /* just a big number */
   max_time = avg_E = 0;

   while (1) {
      if (c = keystat ()) {
         switch (c) {
            case CNTRL_B:
               e_toggle_bell_status ();

               if (Debug_enabled)
                  printf ("Bell %s \n",
                           ((e_get_bell_status()) ? "ENABLED" : "DISABLED"));
               break;
            case CNTRL_F:
               fr_toggle_freerun();
               break;
            case CNTRL_L:
               l_toggle_location_status ();
               if (Debug_enabled)
                  printf ("Location %s \n",
                           ((l_get_location_status()) ? "ENABLED" : "DISABLED"));
               break;
            case CNTRL_HOME:
               if (display_type != HOME) {
                  s_select_display (HOME);
                  s_display_ids ();
                  s_display_traces (0, 0);
                  s_display_triggers ();
                  display_type = HOME;
               }
               break;
```

```
case CNTRL_PGDOWN:
   if (display_type == BLOCK &&
       !s_set_channelrange (MAX_BLOCK, MAX_BLOCK))
      continue;
   else if (display_type != BLOCK) {
      s_select_display (BLOCK);
      display_type = BLOCK;
      s_set_channelrange (0, MAX_BLOCK);
   }
   s_display_ids ();
   s_display_traces (0, 0);
   s_display_triggers ();
   break;
case CNTRL_PGUP:
   if (display_type == BLOCK &&
       !s_set_channelrange (-MAX_BLOCK, MAX_BLOCK))
      continue;
   else if (display_type != BLOCK) {
      s_select_display (BLOCK);
      display_type = BLOCK;
      s_set_channelrange (0, MAX_BLOCK);
   }
   s_display_ids ();
   s_display_traces (0, 0);
   s_display_triggers ();
   break;
case CNTRL_Q:
   quit_enabled = TRUE;
   break;
case CNTRL_R:
   r_toggle_reboot_status ();

   if (Debug_enabled)
      printf ("Reboot %s \n",
              ((r_get_reboot_status()) ? "ENABLED" : "DISABLED"));
   break;
case CNTRL_T:
   t_toggle_trigger_status ();

   h_toggle_status (H_AUTOTRIGGER_ENABLED);
   if (Debug_enabled)
      printf ("Autotriggering %s \n",
              ((t_get_trigger_status()) ? "ENABLED" : "DISABLED"));
   break;
case DOWN_ARROW:
   s_display_traces (0, 1);
   break;
case F1:
   s_view_help ();
   while (! keystat())
      process_buffer ();
   s_clear ();
   s_select_screen (SCREEN_1);
   h_update ();
   s_display_ids ();
   s_display_traces (0, 0);
   s_display_triggers ();
   if (display_type == BLOCK)
      s_display_bar ();
   break;
```

```
            case UP_ARROW:
               s_display_traces (0, -1);
               break;
            default:
               break;
         }
      }
      process_buffer ();
   }
}
```

```
/* FILE: mdemux.h                                    (R. Cutler 04/30/91)
                                                  (D. Tottingham  10/06/90)

This is an include file of defines, data structure definitions and
external declarations that are common in the mdemux module.

*/

#ifndef _MDEMUX_
#define _MDEMUX_

/*****************************************************************************
                                  INCLUDES
*****************************************************************************/
#include "mconst.h"
#include "mqueue.h"


/*****************************************************************************
                                  DEFINES
*****************************************************************************/
#define PRE_EVENT_TIME                  11      /* 11 seconds */

#define NEW_BUFFER                      3
#define AVAILABLE_BUFFER                2
#define ARCHIVED_BUFFER                 1

#define     DIGIREC_BOARD                 1
#define     DT2821_BOARD                  2

/*****************************************************************************
                          STRUCTURE DEFINITIONS


*****************************************************************************/
typedef struct {
        int far * channel_list;
        int far * gain_list;
        SUDS_STRUCTTAG structtag;
        SUDS_ATODINFO  info;
} DM_INFO;
```

```
/*****************************************************************************
                        EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

*****************************************************************************/
PUBLIC FLAG         dm_buffer_full ();
PUBLIC int          dm_data_mid_point ();
PUBLIC float              dm_data_input_range();
PUBLIC float              dm_data_counts();
PUBLIC void         dm_dump_configuration (FILE *);
PUBLIC unsigned int       dm_get_channel_gain (unsigned int);
PUBLIC unsigned long      dm_get_channel_size ();
PUBLIC char         dm_get_clock_source ();
PUBLIC DM_INFO far *      dm_get_collect_info();
PUBLIC double             dm_get_current_time ();
PUBLIC double             dm_get_digitization_rate ();
PUBLIC Q_BUFFER far *     dm_get_first_buffer ();
PUBLIC Q_BUFFER far *     dm_get_head_buffer ();
PUBLIC void         dm_get_new_buffers ();
PUBLIC Q_BUFFER far *     dm_get_next_buffer ();
PUBLIC int          dm_get_number_of_ext_buffers ();
PUBLIC double             dm_get_pre_event_time ();
PUBLIC int          dm_get_scan_count ();
PUBLIC Q_BUFFER far *     dm_get_tail_buffer ();
PUBLIC void         dm_initialize ();
PUBLIC void         dm_initialize_params ();
PUBLIC void         dm_initialize_time ();
PUBLIC void         dm_reset ();
PUBLIC void               dm_set_PreEventTime (double);

PUBLIC void         dm_initialize_buffers ();
PUBLIC void         dm_initialize_collection ();
PUBLIC void         dm_initialize_collect_params ();
PUBLIC void         dm_initialize_params ();
PUBLIC void         dm_initialize_time ();
PUBLIC void         dm_reset ();
PUBLIC void         dm_set_channel ();
PUBLIC void         dm_set_ChannelBlocksize ();
PUBLIC void         dm_set_ChannelGain ();
PUBLIC void         dm_set_ClockSource();
PUBLIC void         dm_set_DigitizationRate ();
PUBLIC void         dm_set_TriggerSource ();
PUBLIC void         dm_set_PreEventTime ();
PUBLIC void         dm_start_collection ();
PUBLIC void         dm_stop_collection ();

#endif
```

```
/* FILE: mdemux.c                                        (R. Cutler 04/30/91)
                                                      (D. Tottingham  12/02/90)

This is a collection of C functions that manage the demux data queue for
xdetect.  All functions have been written and compiled medium model. The
following functions are included:

dm_buffer_full ()        return buffer status
dm_data_mid_point ()           give mid point in data values
dm_data_input_range()          give input range
dm_data_counts()         give number of counts that span range
dm_dump_configuration ()       dump the DT2821, or Digirec configuration
dm_get_channel_gain ()         get channel gain
dm_get_channel_size ()         get channel size
dm_get_clock_source ()         get clock source
dm_get_collect_info()          get pointer to information about collect
dm_get_current_time ()           get current time
dm_get_digitization_rate ()    get digitization rate
dm_get_first_buffer ()           get first new demux buffer from demux queue
dm_get_head_buffer ()            get buffer at head of demux queue
dm_get_new_buffers ()            get buffer(s) from ADC and demultiplex
dm_get_next_buffer ()            get next buffer from demux queue
dm_get_number_of_ext_buffers ()      get number of external buffers
dm_get_pre_event_time ()         get pre event time
dm_get_scan_count ()           get scan count
dm_get_tail_buffer ()            get buffer at tail of demux queue
dm_initialize ()                 initialize the demux queues
dm_initialize_buffers ()       initialize buffers
dm_initialize_collection ()    initialize collection routines
dm_initialize_collect_params ()      initialize parameters of collect software
dm_initialize_params ()          initialize parameters
dm_initialize_time ()            initialize the time
dm_reset ()                      reset this module and release all storage
dm_set_channel ()        put channel and gain in lists
dm_set_ChannelBlocksize ()     set channel block size
dm_set_ChannelGain ()          set channel gain
dm_set_ClockSource ()          set clock source
dm_set_DigitizationRate ()     set digitization rate
dm_set_TriggerSource ()        set trigger source
dm_set_PreEventTime ()           set PreEventTime constant
dm_start_collection ()         start A/D conversion, or Digirec sampling
dm_stop_collection ()          stop A/D conversion, or Digirec sampling
```

```
EXTERNAL FUNCTIONS CALLED:


dr_buffer_full ()       return buffer status
dr_dump_configuration ()      dump the Digirec configuration
dr_get_atodinfo ()            return Digirec information structure
dr_get_channel_gain ()        get channel gain
dr_get_channel_size ()          get channel block size
dr_get_clock_source ()        get clock source
dr_get_digitization_rate ()     get digitization rate
dr_get_first_buffer ()          get first new demux buffer from demux queue
dr_get_new_buffers ()           get new mux buffer(s) from DT2821
dr_get_next_buffer ()           get buffer at tail of mux queue
dr_get_number_of_ext_buffers () get number of external buffers
dr_get_scan_count ()            get scan count
dr_initialize_collection ()   initialize Digirec converter
dr_initialize_buffers ()      initialize buffers
dr_initialize_params ()       initialize parameters of collect software
dr_set_channel ()       put channel and gain in lists
dr_set_ChannelBlocksize ()    set channel block size
dr_set_ChannelGain ()         set channel gain
dr_set_ClockSource            set clock source
dr_set_DigitizationRate ()    set digitization rate
dr_set_TriggerSource ()       set trigger source
dr_start_collection ()        start collecting data from serial ports
dr_stop_collection ()         stop collecting data from serial ports

dt_buffer_full ()       return buffer status
dt_dump_configuration ()      dump the DT2821, or Digirec configuration
dt_get_atodinfo ()            return DT2821 structure
dt_get_channel_gain ()        get channel gain
dt_get_channel_size ()          get channel block size
dt_get_clock_source ()        get clock source
dt_get_digitization_rate ()     get digitization rate
dt_get_first_buffer ()          get first new demux buffer from demux queue
dt_get_new_buffers ()           get new mux buffer(s) from DT2821
dt_get_next_buffer ()           get buffer at tail of mux queue
dt_get_number_of_ext_buffers () get number of external buffers
dt_get_scan_count ()            get scan count
dt_initialize_ADC ()          initialize A/D converter
dt_initialize_buffers ()      initialize buffers
dt_initialize_params ()       initialize parameters of collect software
dt_set_channel ()       put channel and gain in lists
dt_set_ChannelBlocksize ()    set channel block size
dt_set_ChannelGain ()         set channel gain
dt_set_ClockSource            set clock source
dt_set_DigitizationRate ()    set digitization rate
dt_set_TriggerSource ()       set trigger source
dt_start_ADC ()               start A/D conversion
dt_stop_ADC ()                stop A/D conversion

er_abort ()                     display an error message then quit
o_write_logfile ()              write string to log file stream
q_dequeue ()                    dequeue a data link from a data queue
q_enqueue ()                    enqueue a data link on a data queue
q_initialize ()                 initialize a data queue
suds_initialize ()              initialize a suds structure
suds_initialize_tag ()          initialize a suds structtag
u_convert_time ()               convert a struct timeb into abs. time

HISTORY:
   none

*/
```

```
/*****************************************************************************
                              INCLUDE FILES

*****************************************************************************/
#include <malloc.h>
#include <stdio.h>
#include <sys\types.h>
#include <sys\timeb.h>
#include <time.h>

#include "mconst.h"
#include "mdemux.h"
#include "mdigirec.h"
#include "mdt28xx.h"
#include "merror.h"
#include "mlog.h"
#include "mqueue.h"
#include "msudsini.h"
#include "mutils.h"
#include "powrstor.h"
#include "xdetect.h"


/*****************************************************************************
                                 GLOBALS

*****************************************************************************/

PRIVATE void dm_get_new_collect_buffers ();
PRIVATE Q_BUFFER far * dm_get_next_collect_buffer();

PRIVATE Q_QUEUE demux_queue;
PRIVATE Q_LINK * head_ptr, * first_ptr;
PRIVATE double current_time, startup_time;
PRIVATE double pre_event_time;
PRIVATE int CollectType=DIGIREC_BOARD;

/*===========================================================================*
 *                              adjust_queue                                 *
 *===========================================================================*/
/* Adjust the pre-trigger queue and return the used ST_heap_ptr.             */

PRIVATE
struct STORheap far * adjust_queue (buffer_time)
double buffer_time;
{
   struct STORheap far * sheap_ptr;
   STORstatus stat;
   Q_TYPE type;

   sheap_ptr = NULL;
   if ((current_time - startup_time) > (pre_event_time + buffer_time)) {
      if (q_dequeue (&demux_queue, &type)) {
         sheap_ptr = type.buffer->ST_heap_ptr;
         _ffree (type.buffer);
      }
   }
   return (sheap_ptr);
}
```

```
/*=============================================================================*
 *                        create_available_buffers                             *
 *=============================================================================*/
/* Convert NEW buffers into AVAILABLE buffers.                                 */

PRIVATE
void create_available_buffers ()
{
   Q_LINK * hptr;

   for (hptr = demux_queue.head; hptr != NULL; hptr = hptr->next)
      if (hptr->type.buffer->buffer_status == NEW_BUFFER) {
         hptr->type.buffer->buffer_status = AVAILABLE_BUFFER;
      }
}

/*=============================================================================*
 *                          get_available_buffer                               *
 *=============================================================================*/
/* Find an available data buffer.                                              */

PRIVATE
int far * get_available_buffer ()
{
   Q_LINK * hptr;

   for (hptr = demux_queue.head;
        hptr != NULL && hptr->type.buffer->buffer_status != NEW_BUFFER;
        hptr = hptr->next)
      if (hptr->type.buffer->buffer_status == AVAILABLE_BUFFER) {
         hptr->type.buffer->buffer_status = ARCHIVED_BUFFER;
         return (hptr->type.buffer->data);
      }
   return (NULL);
}

/*=============================================================================*
 *                                  get_data                                   *
 *=============================================================================*/
/* If data is not in base memory, get it from the $heap.                       */

PRIVATE
void get_data (buffer)
Q_BUFFER far * buffer;
{
   STORstatus stat;
   unsigned int actual_buffer_size;

   if (buffer == NULL || buffer->buffer_status != ARCHIVED_BUFFER)
      return;

   actual_buffer_size = buffer->buffer_size * sizeof(unsigned);
   if ((buffer->data = get_available_buffer()) == NULL) {
      buffer->data = (int far *) _fmalloc ( actual_buffer_size);
      if (buffer->data == NULL) er_abort (DM_NO_STORAGE);
   }

   buffer->buffer_status = AVAILABLE_BUFFER;
   stat = getSTOR(*(buffer->ST_heap_ptr), 0L, buffer->data, actual_buffer_size);
   if (stat != Okay) er_abort (DM_POWRSTOR + stat);
}
```

```
/*=============================================================================*
 *                          dm_get_current_time                                *
 *=============================================================================*/
/* Get current time.                                                           */

PUBLIC
double dm_get_current_time ()
{
   return (current_time);
}

/*=============================================================================*
 *                          dm_get_first_buffer                                *
 *=============================================================================*/
/* Get first new demux buffer from demux queue.                                */

PUBLIC
Q_BUFFER far * dm_get_first_buffer ()
{
   head_ptr = first_ptr;
   if (head_ptr != NULL) {
      get_data (head_ptr->type.buffer);
      return (head_ptr->type.buffer);
   } else return (NULL);
}

/*=============================================================================*
 *                          dm_get_head_buffer                                 *
 *=============================================================================*/
/* Get buffer at head of queue.                                                */

PUBLIC
Q_BUFFER far * dm_get_head_buffer ()
{
   head_ptr = demux_queue.head;
   if (head_ptr != NULL) {
      get_data (head_ptr->type.buffer);
      return (head_ptr->type.buffer);
   } else return (NULL);
}

/*=============================================================================*
 *                          dm_get_new_buffers                                 *
 *=============================================================================*/
/* Get buffer from ADC.  Update buffer_start_time.  Demultiplex the raw
   data and save link in data queue.                                           */

PUBLIC
void dm_get_new_buffers ()
{
   Q_BUFFER far * mux, far * demux;
   Q_TYPE type;
   STORstatus stat;
   unsigned int actual_buffer_size;

   /* Wait for a buffer to complete */
   dm_get_new_collect_buffers ();
   first_ptr = NULL;

   /* Free up the new demux data buffers */
   create_available_buffers();
```

```
   while ( (mux = dm_get_next_collect_buffer()) != NULL) {
      demux = (Q_BUFFER far *) _fmalloc (sizeof (Q_BUFFER));
      if (demux == NULL) er_abort (DM_NO_STORAGE);

      suds_initialize (MUXDATA, &demux->info);
      suds_initialize_tag (MUXDATA, &demux->structtag);

      demux->buffer_status = NEW_BUFFER;
      demux->buffer_size = mux->buffer_size;
      demux->structtag = mux->structtag;
      demux->info = mux->info;
      demux->info.begintime = current_time;
      demux->info.loctime = 0;

      actual_buffer_size = mux->buffer_size * sizeof(unsigned);
      demux->data = get_available_buffer();
      if (demux->data == NULL) {
         demux->data = (int far *) _fmalloc ( actual_buffer_size);
         if (demux->data == NULL) er_abort (DM_NO_STORAGE);
      }

      demux->ST_heap_ptr = adjust_queue (mux->seconds_in_buffer);
      if (demux->ST_heap_ptr == NULL) {
         demux->ST_heap_ptr
                  = (struct STORheap far *) _fmalloc ( sizeof(struct STORheap));
         if (demux->ST_heap_ptr == NULL) er_abort (DM_NO_STORAGE);
         stat = newSheap( demux->ST_heap_ptr);
         if (stat != Okay) er_abort (DM_POWRSTOR + stat);
         stat = setSheap( *(demux->ST_heap_ptr),
                          (unsigned long) (actual_buffer_size));
         if (stat != Okay) er_abort (DM_POWRSTOR + stat);
      }

      /* Demultiplex the mux buffer */
      to_demux (mux->data, demux->data, mux->buffer_size *
                sizeof(unsigned), mux->info.numchans);

      /* Copy the demultiplexed data into the Sheap */
      stat = putSTOR( *(demux->ST_heap_ptr), demux->data, 0L, actual_buffer_size);
      if (stat != Okay) er_abort (DM_POWRSTOR + stat);

      /* Put the demux buffer in the queue */
      type.buffer = demux;
      q_enqueue (&demux_queue, type);

      current_time += mux->seconds_in_buffer;
      if (first_ptr == NULL) first_ptr = demux_queue.tail;
   }
}

/*=============================================================================*
 *                          dm_get_next_buffer                                 *
 *=============================================================================*/
/* Get next buffer from queue.                                                 */

PUBLIC
Q_BUFFER far * dm_get_next_buffer ()
{
   head_ptr = head_ptr->next;
   if (head_ptr != NULL) {
      get_data (head_ptr->type.buffer);
      return (head_ptr->type.buffer);
   } else return (NULL);
}
```

```
/*=============================================================================*
 *                          dm_get_pre_event_time                              *
 *=============================================================================*/
/* Get pre event time.                                                         */

PUBLIC
double dm_get_pre_event_time ()
{
   return (pre_event_time);
}

/*=============================================================================*
 *                           dm_get_tail_buffer                                *
 *=============================================================================*/
/* Get buffer at tail of queue.                                                */

PUBLIC
Q_BUFFER far * dm_get_tail_buffer ()
{
   if (demux_queue.tail != NULL) {
      get_data (demux_queue.tail->type.buffer);
      return (demux_queue.tail->type.buffer);
   } else return (NULL);
}
```

```
/*=============================================================================*
 *                                dm_initialize                                *
 *=============================================================================*/
/* Initialize the queues and calculate some key values.                        */

PUBLIC
void dm_initialize ()
{
   static union INITelement init_storage[2];
   STORstatus stat;
   int low_limit, high_limit;
   unsigned int needed_size, chunks_in_block, blocks_in_queue;
   unsigned long samples_in_queue, bytes_in_block, channel_size;

   /* Dedicate an area of extended memory to PowerSTOR. Upper and
      lower limits are specified in 16K chunks from the bottom of
      base memory.  Note that there are 8 16K chunks in each ATLAB
      buffer (128K bytes) and 64 16K chunks to the 1M byte line.
    */
   low_limit = 64;
   high_limit = -(dm_get_number_of_ext_buffers() * 8 + 1);

   /* Calculate the amount of PowerSTOR needed to hold the pre-trigger queue */
   channel_size = dm_get_channel_size();
   bytes_in_block = channel_size * dm_get_scan_count() * sizeof(unsigned);
   chunks_in_block = bytes_in_block / SMALLEST_CHUNK;
   if (bytes_in_block % SMALLEST_CHUNK) chunks_in_block++;
   samples_in_queue = pre_event_time * dm_get_digitization_rate();
   blocks_in_queue = (samples_in_queue / channel_size) + 1;
   if (samples_in_queue % channel_size) blocks_in_queue++;
   needed_size = chunks_in_block * blocks_in_queue + POWRSTOR_USAGE;
   if (needed_size < 4) needed_size = 4;

/** debug **/
if (Debug_enabled) {
   printf ("PowerSTOR stats: \n");
   printf (" low_limit, high_limit = %d, %d\n", low_limit, high_limit);
   printf (" chunks_in_block, blocks_in_queue, needed_size = %u, %u, %u\n",
           chunks_in_block, blocks_in_queue, needed_size);
}

   /* Fill the init_storage array */
/**/
   init_storage[0].EXTmem.memType = ExtMem;
   init_storage[0].EXTmem.LoLimit = low_limit;
   init_storage[0].EXTmem.HiLimit = high_limit;
/**/
   init_storage[1].Nulmem.memType = NulMem;
/**
   init_storage[0].diskF.memType = DskFile;
   init_storage[0].diskF.MaxSize = needed_size;
   strcpy(init_storage[0].diskF.namestring, "d:file.tmp");
**/

   /* Initialize PowerSTOR */
   stat = InitPwr( needed_size, init_storage);
   if (stat != Okay) er_abort (DM_POWRSTOR + stat);

   q_initialize (&demux_queue);
   first_ptr = head_ptr = NULL;
}
```

```
/*=============================================================================*
 *                            dm_initialize_params                             *
 *=============================================================================*/
/* Initialize parameters.                                                      */

PUBLIC
void dm_initialize_params ()
{
   pre_event_time = PRE_EVENT_TIME;
}

/*=============================================================================*
 *                             dm_initialize_time                              *
 *=============================================================================*/
/* Initialize the time.                                                        */

PUBLIC
void dm_initialize_time ()
{
   struct timeb current_timeb;

   ftime (&(current_timeb));
   startup_time = current_time = u_convert_time (current_timeb);
}

/*=============================================================================*
 *                                  dm_reset                                   *
 *=============================================================================*/
/* Reset this module and release all storage.                                  */

PUBLIC
void dm_reset ()
{
   Q_TYPE type;

   while (q_dequeue (&demux_queue, &type)) {
      if (type.buffer->buffer_status != ARCHIVED_BUFFER)
         _ffree (type.buffer->data);
      _ffree (type.buffer->ST_heap_ptr);
      _ffree (type.buffer);
   }
   QuitPwr();
}

/*=============================================================================*
 *                            dm_set_PreEventTime                              *
 *=============================================================================*/
/* Set PreEventTime constant.                                                  */

PUBLIC
void dm_set_PreEventTime (PreEventTime)
double PreEventTime;
{
      pre_event_time = PreEventTime;
}
```

```
/*=============================================================================*
 *                              dm_buffer_full                                 *
 *=============================================================================*/
/*                                                                             */

PUBLIC
FLAG dm_buffer_full ()
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            return(dr_buffer_full());
            break;
      case DT2821_BOARD:
            return(dt_buffer_full());
            break;
      }
}

/*=============================================================================*
 *                              dm_dump_configuration                          *
 *=============================================================================*/
/*                                                                             */

PUBLIC
void dm_dump_configuration (stream)
FILE * stream;
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            dr_dump_configuration(stream);
            break;
      case DT2821_BOARD:
            dt_dump_configuration(stream);
            break;
      }
}

/*=============================================================================*
 *                              dm_get_channel_gain                            *
 *=============================================================================*/
/*                                                                             */

PUBLIC
unsigned int dm_get_channel_gain (channel)
unsigned int channel;
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            return(dr_get_channel_gain(channel));
            break;
      case DT2821_BOARD:
            return(dt_get_channel_gain(channel));
            break;
      }
}
```

```
/*=============================================================================*
 *                              dm_get_channel_size                            *
 *=============================================================================*/
/*                                                                             */

PUBLIC
unsigned long dm_get_channel_size ()
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            return(dr_get_channel_size());
            break;
      case DT2821_BOARD:
            return(dt_get_channel_size());
            break;
      }
}

/*=============================================================================*
 *                              dm_get_clock_source                            *
 *=============================================================================*/
/*                                                                             */

PUBLIC
char dm_get_clock_source ()
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            return(dr_get_clock_source());
            break;
      case DT2821_BOARD:
            return(dt_get_clock_source());
            break;
      }
}

/*=============================================================================*
 *                              dm_get_digitization_rate                       *
 *=============================================================================*/
/*                                                                             */

PUBLIC
double dm_get_digitization_rate ()
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            return(dr_get_digitization_rate());
            break;
      case DT2821_BOARD:
            return(dt_get_digitization_rate());
            break;
      }
}
```

```
/*=============================================================================*
 *                              dm_get_number_of_ext_buffers                   *
 *=============================================================================*/
/*                                                                             */

PUBLIC
int dm_get_number_of_ext_buffers ()
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            return(dr_get_number_of_ext_buffers());
            break;
      case DT2821_BOARD:
            return(dt_get_number_of_ext_buffers());
            break;
      }
}

/*=============================================================================*
 *                              dm_get_scan_count                              *
 *=============================================================================*/
/*                                                                             */

PUBLIC
int dm_get_scan_count ()
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            return(dr_get_scan_count());
            break;
      case DT2821_BOARD:
            return(dt_get_scan_count());
            break;
      }
}

/*=============================================================================*
 *                              dm_initialize_buffers                          *
 *=============================================================================*/
/*                                                                             */

PUBLIC
void dm_initialize_buffers ()
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            dr_initialize_buffers();
            break;
      case DT2821_BOARD:
            dt_initialize_buffers();
            break;
      }
}
```

```
/*=============================================================================*
 *                              dm_initialize_collection                       *
 *=============================================================================*/
/*                                                                             */

PUBLIC
void dm_initialize_collection ()
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            dr_initialize_collection();
            break;
      case DT2821_BOARD:
            dt_initialize_ADC();
            break;
      }
}

/*=============================================================================*
 *                              dm_initialize_collect_params                   *
 *=============================================================================*/
/*                                                                             */

PUBLIC
void dm_initialize_collect_params ()
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            dr_initialize_params();
            break;
      case DT2821_BOARD:
            dt_initialize_params();
            break;
      }
}

/*=============================================================================*
 *                              dm_set_channel                                 *
 *=============================================================================*/
/*                                                                             */

PUBLIC
void dm_set_channel (channel,gain)
unsigned int channel;
unsigned int gain;
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            dr_set_channel(channel,gain);
            break;
      case DT2821_BOARD:
            dt_set_channel(channel,gain);
            break;
      }
}
```

```
/*=============================================================================*
 *                              dm_set_ChannelBlocksize                        *
 *=============================================================================*/
/*                                                                             */

PUBLIC
void dm_set_ChannelBlocksize (blocksize)
unsigned long blocksize;
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            dr_set_ChannelBlocksize(blocksize);
            break;
      case DT2821_BOARD:
            dt_set_ChannelBlocksize(blocksize);
            break;
      }
}

/*=============================================================================*
 *                              dm_set_ChannelGain                             *
 *=============================================================================*/
/*                                                                             */

PUBLIC
void dm_set_ChannelGain (gain)
unsigned int gain;
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            dr_set_ChannelGain(gain);
            break;
      case DT2821_BOARD:
            dt_set_ChannelGain(gain);
            break;
      }
}

/*=============================================================================*
 *                              dm_set_ClockSource                             *
 *=============================================================================*/
/*                                                                             */

PUBLIC
void dm_set_ClockSource(timing_source)
int timing_source;
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            dr_set_ClockSource(timing_source);
            break;
      case DT2821_BOARD:
            dt_set_ClockSource(timing_source);
            break;
      }
}
```

```
/*=============================================================================*
 *                              dm_set_DigitizationRate                        *
 *=============================================================================*/
/*                                                                             */

PUBLIC
void dm_set_DigitizationRate (rate)
double rate;
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            dr_set_DigitizationRate(rate);
            break;
      case DT2821_BOARD:
            dt_set_DigitizationRate(rate);
            break;
      }
}

/*=============================================================================*
 *                              dm_set_TriggerSource                           *
 *=============================================================================*/
/*                                                                             */

PUBLIC
void dm_set_TriggerSource (trigger_source)
int trigger_source;
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            dr_set_TriggerSource(trigger_source);
            break;
      case DT2821_BOARD:
            dt_set_TriggerSource(trigger_source);
            break;
      }
}

/*=============================================================================*
 *                              dm_start_collection                            *
 *=============================================================================*/
/*                                                                             */

PUBLIC
void dm_start_collection ()
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            dr_start_collection();
            break;
      case DT2821_BOARD:
            dt_start_ADC();
            break;
      }
}
```

```
/*=============================================================================*
 *                              dm_stop_collection                             *
 *=============================================================================*/
/*                                                                             */

PUBLIC
void dm_stop_collection ()
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            dr_stop_collection();
            break;
      case DT2821_BOARD:
            dt_stop_ADC();
            break;
      }
}

/*=============================================================================*
 *                              dm_data_mid_point                              *
 *=============================================================================*/
/*                                                                             */

PUBLIC
dm_data_mid_point ()
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            return(DIGIREC_MID_POINT);
            break;
      case DT2821_BOARD:
            return(ADC_MID_POINT);
            break;
      }
}

/*=============================================================================*
 *                              dm_data_input_range                            *
 *=============================================================================*/
/*                                                                             */

PUBLIC
float dm_data_input_range ()
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            return(DIGIREC_INPUT_RANGE);
            break;
      case DT2821_BOARD:
            return(ADC_INPUT_RANGE);
            break;
      }
}
```

```
/*=============================================================================*
 *                              dm_data_counts                                 *
 *=============================================================================*/
/*                                                                             */

PUBLIC
float dm_data_counts ()
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            return(DIGIREC_COUNTS);
            break;
      case DT2821_BOARD:
            return(ADC_COUNTS);
            break;
      }
}

/*=============================================================================*
 *                              dm_get_collect_info                            *
 *=============================================================================*/
/*                                                                             */

PUBLIC
DM_INFO far * dm_get_collect_info()
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            return(dr_get_collect_info());
            break;
      case DT2821_BOARD:
            return(dt_get_atodinfo());
            break;
      }
}

/*=============================================================================*
 *                              dm_get_new_collect_buffers                     *
 *=============================================================================*/
/*                                                                             */

PRIVATE
void dm_get_new_collect_buffers ()
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            dr_get_new_buffers();
            break;
      case DT2821_BOARD:
            dt_get_new_buffers();
            break;
      }
}
```

```
/*=============================================================================*
 *                              dm_get_next_collect_buffer                     *
 *=============================================================================*/
/*                                                                             */

PRIVATE
Q_BUFFER far * dm_get_next_collect_buffer()
{
switch(CollectType)
      {
      case DIGIREC_BOARD:
            return(dr_get_next_buffer());
            break;
      case DT2821_BOARD:
            return(dt_get_next_buffer());
            break;
      }
}
```

```
/* FILE: mdigirec.h                                (R. Cutler 04/30/91)

This is an include file of the defines, data structure definitions and
external data declarations for using the mdigirec module.

*/

#ifndef _MDIGIREC_
#define _MDIGIREC_

/************************************************************************
                                INCLUDES

************************************************************************/
#include "mconst.h"
#include "mqueue.h"
#include "mdemux.h"


/************************************************************************
                                DEFINES

************************************************************************/
/* Constants */
#define     DR_OKAY                 0

#define DIGIREC_INPUT_RANGE         5.00            /* in volts, 20 = +/- 10V */
#define DIGIREC_MID_POINT           2048             /* 10 bit */
#define     DIGIREC_COUNTS          4096.0
#define DIGIREC                 4096        /*identifier for digirec board */

#define     DR_BASE_ADDRESS         0x0300
#define     DR_DMA_CHANNEL          6
#define     DR_INT_CHANNEL          10

#define BOARD_CLOCK_FREQUENCY   4e6
#define HIGH_BYTE_ENABLE        0x0001
#define MAX_CHANNEL             128
#define MAX_GAIN                8

/* Defaults */
#define CHANNEL_GAIN            4
#define CLOCK_SOURCE            INTERNAL_CLOCK
#define DIGITIZATION_RATE       100.0           /* Samples/sec/channel */
#define TIME_OUT                10
#define TRIGGER_SOURCE          INTERNAL_TRIGGER

/* Buffer Constants */
#define MAX_BUFFER_SIZE         16384L


/* bit definitions for the A/D timing source */

#define INTERNAL_CLOCK        0
#define EXTERNAL_CLOCK        1
#define INTERNAL_TRIGGER      0
#define EXTERNAL_TRIGGER      2

/************************************************************************
 *                STRUCTURE DEFINITIONS
 ************************************************************************/
```

```
/*****************************************************************************
                          EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

*****************************************************************************/
PUBLIC FLAG dr_buffer_full ();
PUBLIC void dr_dump_configuration (FILE *);
PUBLIC DM_INFO far * dr_get_collect_info ();
PUBLIC unsigned long dr_get_buffer_size ();
PUBLIC unsigned int  dr_get_channel_gain (unsigned int);
PUBLIC unsigned long dr_get_channel_size ();
PUBLIC char dr_get_clock_source ();
PUBLIC double dr_get_digitization_rate ();
PUBLIC void dr_get_new_buffers ();
PUBLIC Q_BUFFER far * dr_get_next_buffer ();
PUBLIC int dr_get_number_of_ext_buffers ();
PUBLIC int dr_get_scan_count ();
PUBLIC char dr_get_trigger_source ();
PUBLIC void dr_initialize_buffers ();
PUBLIC void dr_initialize_collection();
PUBLIC void dr_initialize_params();
PUBLIC void dr_set_channel (unsigned int, unsigned int);
PUBLIC void dr_set_ChannelBlocksize (unsigned long);
PUBLIC void dr_set_ChannelGain (unsigned int);
PUBLIC void dr_set_ClockSource (int);
PUBLIC void dr_set_DigitizationRate (double);
PUBLIC void dr_set_TriggerSource (int);
PUBLIC void dr_start_collection ();
PUBLIC void dr_stop_collection ();

#endif
```

```
/* FILE: mdigirec.c                                    (R. Cutler 04/30/91)
                                                    (D. Tottingham  09/29/90)

This is a collection of C functions that manage the DIGIREC board for xdetect.
All functions have been written and compiled medium model.  The following
functions are included:

dr_buffer_full ()               return buffer status
dr_dump_configuration ()        dump the DT2821 configuration
dr_get_collect_info ()          return DM_INFO structure
dr_get_buffer_size ()           get buffer size
dr_get_channel_size ()          get channel block size
dr_get_channel_gain ()          get channel gain
dr_get_clock_source ()          get clock source
dr_get_digitization_rate ()     get digitization rate
dr_get_new_buffers ()           get new buffer(s) from DT2821
dr_get_next_buffer ()           get next buffer from mux queue
dr_get_number_of_ext_buffers () get number of external buffers
dr_get_scan_count ()            get scan count
dr_get_trigger_source ()        get trigger source
dr_initialize_buffers ()        initialize buffers
dr_initialize_collection ()     initialize the digirec
dr_initialize_params()          initialize parameters
dr_set_channel ()               put channel and gain in channel and gain lists
dr_set_ChannelBlocksize ()      set channel block size
dr_set_ChannelGain ()           set the channel gain
dr_set_ClockSource ()           set clock source
dr_set_DigitizationRate ()      set the digitization rate
dr_set_TriggerSource ()         set trigger source
dr_start_collection ()          start continuous data acquisition
dr_stop_collection ()           stop continuous data acquisition

EXTERNAL FUNCTIONS CALLED:

er_abort ()                     display an error message then quit
o_write_logfile ()              write string to log file stream
suds_initialize ()              initialize a suds structure
suds_initialize_tag ()          initialize a suds structtag

HISTORY:

V2.001 (04/30/91) changed to allow other data acquisition systems, the
                  information data structure was renamed from DT_28XX
                  to DM_INFO, and move to mdemux.h
                  (Reese T. Cutler)


*/
```

```
/*****************************************************************************
                              INCLUDE FILES
*****************************************************************************/
#include <conio.h>
#include <malloc.h>
#include <math.h>
#include <stdio.h>
#include <dos.h>

#include "mconst.h"
#include "mdemux.h"
#include "mdigirec.h"
#include "mdigisup.h"
#include "merror.h"
#include "mfile.h"
#include "mlog.h"
#include "mqueue.h"
#include "msudsini.h"
#include "mstation.h"
#include "mutils.h"
#include "xdetect.h"


/*****************************************************************************
                                 GLOBALS
*****************************************************************************/
PRIVATE DM_INFO adc_info, far * a = &adc_info;

PRIVATE DR_CONFIGURATION config, far * c = &config;

PRIVATE int status;
PRIVATE Q_QUEUE mux_queue;
PRIVATE Q_LINK * next_ptr;
PRIVATE double digitization_rate;
PRIVATE unsigned long buffer_size;
PRIVATE unsigned long channel_blocksize;
PRIVATE unsigned int scan_count;
PRIVATE unsigned int channel_gain;
```

```
/*=============================================================================*
 *                          allocate_int_buffers                               *
 *=============================================================================*/
/* Allocate "internal" buffer queue for digirec.  All internal buffers
   reside in the base 640K of user RAM.                                        */

PRIVATE
void allocate_int_buffers ()
{
   static unsigned int far int_buffer_number;
   unsigned int actual_blocksize, actual_buffersize;
   double seconds_in_buffer;
   unsigned long offset;
   Q_TYPE type;
   Q_BUFFER far * mux;
   char * netwname;

   actual_buffersize = (buffer_size <= MAX_BUFFER_SIZE) ? buffer_size :
MAX_BUFFER_SIZE;
   actual_blocksize = actual_buffersize / scan_count;
   seconds_in_buffer = actual_blocksize / digitization_rate;

   q_initialize (&mux_queue);
   netwname = st_get_netwname ();

   for (offset = 0; offset < buffer_size; offset += MAX_BUFFER_SIZE) {
      mux = (Q_BUFFER far *) _fmalloc (sizeof (Q_BUFFER));
      if (mux == NULL) er_abort (DR_INT_MEM);

      mux->data = (int far *) _fmalloc ( actual_buffersize * sizeof(unsigned));
      if (mux->data == NULL) er_abort (DR_INT_MEM);
      mux->buffer_size = actual_buffersize;
      mux->seconds_in_buffer = seconds_in_buffer;
      suds_initialize (MUXDATA, &mux->info);
      suds_initialize_tag (MUXDATA, &mux->structtag);

      mux->structtag.len_data = actual_buffersize * sizeof(unsigned);
      u_strncpy (mux->info.netname, ((char far *)netwname), 4);
      mux->info.blocksize = actual_blocksize;
      mux->info.numchans = scan_count;
      mux->info.dig_rate = digitization_rate;
      mux->info.typedata = 's';
      type.buffer = mux;
      q_enqueue (&mux_queue, type);
   }
}

/*=============================================================================*
 *                          allocate_ext_buffers                               *
 *=============================================================================*/
/* Allocate an "external" buffers for digirec.  All external buffers reside
   ABOVE the base 1000K of user RAM.                                           */

PRIVATE
void allocate_ext_buffers ()
{
   static unsigned far ext_buffer_number;
int xbcnt;

dr_init_xm(1024);
xbcnt=dr_init_xm_buff(buffer_size);
a->info.extended_bufs = dr_128K_blocks_used();
if (Debug_enabled) printf("\nexternal buffer count is %d\n",xbcnt);
if (Debug_enabled) printf("#of 128K external blocks used %d\n",
      a->info.extended_bufs);
}
```

```
/*=============================================================================*
 *                              dr_buffer_full                                 *
 *=============================================================================*/
/* Return buffer status.                                                       */

PUBLIC
FLAG dr_buffer_full ()
{
   static unsigned int far buffer_number, far count_remaining;
   return (dr_full_status());
}

/*=============================================================================*
 *                           dr_dump_configuration                             *
 *=============================================================================*/
/* Dump the DT2821 configuration to a stream.                                  */

PUBLIC
void dr_dump_configuration (stream)
FILE * stream;
{
int i;

if (c->device_id != DIGIREC)
      fprintf(stream, "No device defined.\r\n");
else
      {
      fprintf (stream, "Device id is ");
      fprintf (stream, "DIGIREC");

      fprintf(stream, ", ");

      fprintf(stream, "single channel DMA");
      fprintf(stream, ".");

      fprintf(stream, "\r\n");
      fprintf(stream, "Base I/O address is %4x hex.\r\n",c->base_address);
      fprintf(stream, "%d Digital channels.\r\n",c->channel_count);
      }
}

/*=============================================================================*
 *                             dr_get_buffer_size                              *
 *=============================================================================*/
/* Get buffer size.                                                            */

PUBLIC
unsigned long dr_get_buffer_size ()
{
   return (buffer_size);
}

/*=============================================================================*
 *                            dr_get_channel_gain                              *
 *=============================================================================*/
/* Get channel gain.                                                           */

PUBLIC
unsigned int dr_get_channel_gain (channel)
unsigned int channel;
{
   return ((channel < c->channel_count) ? a->gain_list[channel] : channel_gain);
}
```

```
/*=============================================================================*
 *                          dr_get_channel_size                                *
 *=============================================================================*/
/* Get channel block size.                                                     */

PUBLIC
unsigned long dr_get_channel_size ()
{
   return (channel_blocksize);
}

/*=============================================================================*
 *                           dr_get_clock_source                               *
 *=============================================================================*/
/* Get clock source.                                                           */

PUBLIC
char dr_get_clock_source ()
{
   return (a->info.timing_source);
}

/*=============================================================================*
 *                           dr_get_collect_info
 *=============================================================================*/
/* Return DM_INFO structure.                                                   */

PUBLIC
DM_INFO far * dr_get_collect_info ()
{
   return (a);
}

/*=============================================================================*
 *                        dr_get_digitization_rate                             *
 *=============================================================================*/
/* Get digitization rate.                                                      */

PUBLIC
double dr_get_digitization_rate ()
{
   return (digitization_rate);
}
```

```
/*=============================================================================*
 *                            dr_get_new_buffers                               *
 *=============================================================================*/
/* Wait for buffer to complete.  Copy external buffer into user
   accessible mux queue.                                                       */

PUBLIC
void dr_get_new_buffers ()
{
   unsigned long offset, longxmem;
   Q_LINK * head;

   /* Wait for a buffer to complete */

   offset = 0;
   longxmem=dr_get_dma_buffer();
   if ( longxmem != 0 )

      {
      head = next_ptr = mux_queue.head;
      while (head != NULL) {
      getxmem(longxmem+offset,fpta(head->type.buffer->data),
            (int) head->type.buffer->buffer_size);
        offset += head->type.buffer->buffer_size;
         head = head->next;
      }
   }
   else er_abort (DR_DMA_BLOCKED);

}

/*=============================================================================*
 *                            dr_get_next_buffer                               *
 *=============================================================================*/
/* Get next buffer from mux queue.                                             */

PUBLIC
Q_BUFFER far * dr_get_next_buffer ()
{
   Q_LINK * this_ptr;

   this_ptr = next_ptr;
   if (next_ptr != NULL)
      next_ptr = next_ptr->next;

   return (this_ptr->type.buffer);
}

/*=============================================================================*
 *                         dr_get_number_of_ext_buffers                        *
 *=============================================================================*/
/* Get number of external buffers.                                             */

PUBLIC
int dr_get_number_of_ext_buffers ()
{
   return (a->info.extended_bufs);
}
```

```
/*=============================================================================*
 *                              dr_get_scan_count                              *
 *=============================================================================*/
/* Get scan count.                                                             */

PUBLIC
int dr_get_scan_count ()
{
   return (scan_count);
}

/*=============================================================================*
 *                            dr_get_trigger_source                            *
 *=============================================================================*/
/* Get trigger source.                                                         */

PUBLIC
char dr_get_trigger_source ()
{
   return (a->info.trigger_source);
}

/*=============================================================================*
 *                            dr_initialize_buffers                            *
 *=============================================================================*/
/* Initialize buffers.                                                         */

PUBLIC
void dr_initialize_buffers ()
{
if (Debug_enabled) printf("entered dr initialize buffers\n");

/* scan_count is the number of stations producing 16 bit data */
/* channel_blocksize is the number of scans in a dma buffer, it must
      be a power of 2 */
/* buffer_size is the number of 16 bit words in each dma buffer */

   buffer_size = channel_blocksize * scan_count;
   buffer_size = (! buffer_size) ? SEGMENT_SIZE : buffer_size;
   channel_blocksize = buffer_size / scan_count;


   /* Do some error checking first */

   if (! u_ispow2( channel_blocksize))
      er_abort (DR_CBS_NP2);
   else if (buffer_size > SEGMENT_SIZE)
      er_abort (DR_BS_TOO_BIG);

   allocate_int_buffers ();

   allocate_ext_buffers ();

}
```

```
/*=============================================================================*
 *                          dr_initialize_collection                           *
 *=============================================================================*/
/* Initialize the DIGIREC                          */

PUBLIC
void dr_initialize_collection ()
{
   unsigned int actual_scancount;
   int errnum;
   int i,j,k;

if (Debug_enabled) printf("got to DR_INITIALIZE_COLLECTION\n");
   /* Do some error checking */
if (Debug_enabled) printf("scan_count is %d\n",scan_count);
   if (!scan_count)
      er_abort (DR_SCAN_COUNT_0);
   else if (! u_ispow2( (unsigned long)scan_count))
      er_abort (DR_SCAN_COUNT_NP2);

   actual_scancount = (scan_count > c->channel_count) ? c->channel_count :
                      scan_count;

dr_set_io_add(c->base_address);

dr_reset();

k=0;
for (i=0; i<16 ; i++)
      {
      for(j=0;j<3; j++)
            {
            dr_set_position(i,j,ScanCheck(k));
            k+=1;
            }
      dr_set_position(i,3,ScanCheck(48+i));
      }
if (Debug_enabled) printf("Base I/O address is %4x hex.\r\n",c->base_address);

/**debug**/
if (Debug_enabled)
   printf ("Channel digitization rate = %10.41f\n", digitization_rate);

}
```

```
/*=============================================================================*
 *                          dr_initialize_params                               *
 *=============================================================================*/
/* Initialize parameters.                                                      */

PUBLIC
void dr_initialize_params()
{

   /* Get the current configuration */
      c->base_address=DR_BASE_ADDRESS;
            dr_set_io_add(c->base_address);
      c->dma_channel=DR_DMA_CHANNEL;
            dr_set_dma_chan(c->dma_channel);
      c->int_channel=DR_INT_CHANNEL;
            dr_set_int_chan(c->int_channel);
      c->device_id = DIGIREC;
      c->scan_count = 0;
      c->channel_count=64;

   digitization_rate = DIGITIZATION_RATE;
   channel_blocksize = 0;
   channel_gain = CHANNEL_GAIN;
   scan_count = 0;

   suds_initialize (ATODINFO, &a->info);
   suds_initialize_tag (ATODINFO, &a->structtag);

      a->info.base_address = c->base_address;
      a->info.device_id = c->device_id;
      a->info.device_flags = 0;

   a->info.trigger_source = 'i';
   a->info.timing_source = 'i';
   a->info.external_mux = 1;

   a->channel_list = (int far *) _fmalloc (c->channel_count * sizeof(int));
   if (a->channel_list == NULL) er_abort (DR_INT_MEM);

   a->gain_list = (int far *) _fmalloc (c->channel_count * sizeof(int));
   if (a->gain_list == NULL) er_abort (DR_INT_MEM);

}
```

```
/*=============================================================================*
 *                                dr_set_channel                               *
 *=============================================================================*/
/* Put channel number in channel list and gain number in gain list.            */

PUBLIC
void dr_set_channel (channel, gain)
unsigned int channel;
unsigned int gain;
{
static FLAG first_time = TRUE;
unsigned int i, max_channel;

if (channel != scan_count) er_abort (DR_BAD_CHAN);

if (scan_count < c->channel_count)
      {
      a->channel_list[scan_count] = channel;
      a->gain_list[scan_count] = (gain == 0) ? channel_gain : gain;
      if (a->gain_list[scan_count] > MAX_GAIN ||
            !u_ispow2((unsigned long) a->gain_list[scan_count]) )
            {
            er_abort (DR_BAD_GAIN);
            }
      }
else
      {
      er_abort(DR_SCAN_COUNT_BIG);
      }
   scan_count++;
}

/*=============================================================================*
 *                          dr_set_ChannelBlocksize                            *
 *=============================================================================*/
/* Set the channel block size.                                                 */

PUBLIC
void dr_set_ChannelBlocksize (blocksize)
unsigned long blocksize;
{
   channel_blocksize = blocksize;
}

/*=============================================================================*
 *                             dr_set_ChannelGain                              *
 *=============================================================================*/
/* Set the channel gain.                                                       */

PUBLIC
void dr_set_ChannelGain (gain)
unsigned int gain;
{
   int i;

   if (gain > MAX_GAIN || !u_ispow2((unsigned long) gain))
      er_abort (DR_BAD_GAIN);
   channel_gain = gain;
   for (i = 0; i < scan_count && i < c->channel_count; i++)
      a->gain_list[i] = gain;

}
```

```
/*=============================================================================*
 *                              dr_set_ClockSource                             *
 *=============================================================================*/
/* Set clock source.                                                           */

PUBLIC
void dr_set_ClockSource (timing_source)
int timing_source;
{

   a->info.timing_source = (timing_source == INTERNAL_CLOCK) ? 'i' : 'e';

}

/*=============================================================================*
 *                            dr_set_DigitizationRate                          *
 *=============================================================================*/
/* Set the digitization rate.                                                  */

PUBLIC
void dr_set_DigitizationRate (rate)
double rate;
{
   digitization_rate = rate;
}

/*=============================================================================*
 *                              dr_set_TriggerSource                           *
 *=============================================================================*/
/* Set trigger source.                                                         */

PUBLIC
void dr_set_TriggerSource (trigger_source)
int trigger_source;
{

   a->info.trigger_source = (trigger_source == INTERNAL_TRIGGER) ? 'i' : 'e';

}

/*=============================================================================*
 *                              dr_start_collection                            *
 *=============================================================================*/
/* Start continuous data acquisition.                                          */

PUBLIC
void dr_start_collection ()
{

dr_start_acq();

}

/*=============================================================================*
 *                              dr_stop_collection                             *
 *=============================================================================*/
/* Stop continuous data acquisition.  Reset the DIGIREC.                       */

PUBLIC
void dr_stop_collection ()
{
dr_stop_acq();
}
```

```
/*=============================================================*
 *                          ScanCheck                          |
 *=============================================================*/
ScanCheck(i)
int i;
{
if (i<scan_count) return(i);
else return(64);
}
```

```
/* FILE: mdigisup.h                                    (R. Cutler 04/30/91)

This is an include file of the defines, data structure definitions and
external data declarations for using the digiserv module.

*/

#ifndef _MDIGISUP_
#define _MDIGISUP_

/*****************************************************************************
                                  INCLUDES

*****************************************************************************/
#include "drcom.h"
#include "mconst.h"
#include "mqueue.h"
#include "mdemux.h"



/*****************************************************************************
                                  DEFINES

*****************************************************************************/
/* Constants */

/*****************************************************************************
 *                STRUCTURE DEFINITIONS
 *****************************************************************************/

typedef struct xbuf {
            long length;
            long pointer;
            struct xbuf far * next;
            } X_BUF;

typedef struct drbuf {
            unsigned int type;
            unsigned int length;
            unsigned int status;
            struct drbuf * next;
            struct drbuf * last;
            long address;
            } DR_BUF;

typedef struct {
            int base_address;
            int device_id;
            int reserved_1;
            int scan_count;
            int channel_count;
            int dma_channel;
            int int_channel;
            int reserved[9];
            } DR_CONFIGURATION;
```

```
/*************************************************************************
                        EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

*************************************************************************/

PUBLIC      unsigned int      dr_128K_blocks_used();
PUBLIC      int         dr_dma_status();
PUBLIC      int         dr_full_status();
PUBLIC      unsigned long     dr_get_dma_buffer();
PUBLIC      int         dr_init_xm(unsigned int blkcnt);
PUBLIC      unsigned int      dr_init_xm_buff(unsigned long buffersize);
PUBLIC      void        dr_irq_setup();
PUBLIC      void        dr_irq_restore();
PUBLIC      int         dr_isr();
PUBLIC      int         dr_issue_command (int comm, int parm);
PUBLIC      int         dr_nvread(int add, int *da);
PUBLIC      int         dr_nvwrite(int add, int da);
PUBLIC      int         dr_set_baud_rate(int comm, int parm);
PUBLIC      int         dr_set_dma_chan(int chan);
PUBLIC      int         dr_set_int_chan(int chan);
PUBLIC      int         dr_set_io_add(int add);
PUBLIC      int         dr_set_parity(int comm, int parm);
PUBLIC      void        dr_set_position(int chan, int wrd, int pos);
PUBLIC      int         dr_start_acq();
PUBLIC      int         dr_start_dma();
PUBLIC      int         dr_stop_acq();
PUBLIC      int         dr_stop_dma();
PUBLIC      int         dr_reset ();
PUBLIC      long        fpta(int far * point);
PUBLIC                  printgrab(int far data[]);
PUBLIC                  showhead();
#endif
```

```
/*
      DRCOM.H
      Defines for the different commands issued by a PC like
      computer through the io port (usually 300) to control the
      digital data receiver

      FEBRUARY, 1991
      COPYRIGHT CUTLER DIGITAL DESIGN 1991
*/


/* Command interpreter commands */
/* The PC invokes commands by writing a new (and different) command word.

   Results are returned in the status word (same I/O address as
   the command word).

   The high order byte of the command word is called the command byte,
   the low order byte is called the parameter byte.  The parameter byte must
   be changed before, or at the same time as the command byte.

   Upon completion, the command byte is placed in the high order byte of
   the status word, and varying values in the low order byte of the status
   word as specified below.

   The PC knows that its command has been accepted, when it sees
   the current command byte appear in the high order byte of the status word.

   The "nop" command is provided to allow spacing between repeat usage
   of the same command.

   The "general" command pulls together commands that have no parameter
   and no return value.  The parameter byte is used as a sub command.

*/

#define     NOP          0           /* dummy command, does nothing,
                                    return = 0 */
#define     GENERAL            1           /* collection of commands that have
                                    no parameters,
                                    return = subcommand */
      /* general sub commands */
      #define     RESET       0
      #define     CAPTURE           8     /* CAPTURE A DATA SAMPLE */
                        /* 16 bit count in bytel(lo) and byte 2(hi) */
      #define     DISABLE_INT 9
      #define     ENABLE_INT  10

#define     SET_DMA           2
      #define     ENABLE_DMA_CHANNEL_7    3
      #define     ENABLE_DMA_CHANNEL_6    2
      #define     ENABLE_DMA_CHANNEL_5    1
      #define     DISABLE_DMA                     0
                                    /* return=NUMBER OF DATA CHANNELS (1-64)*/

#define     SHIFT_IN_BYTE     3           /* auxilliary parameter shift regist
                                    has 4 bytes, this command shifts
                                    in a new low order byte.
                                    return=parameter byte. */
```

```
#define     WRITE_NVR   4          /* write the byte in Parameter
                                   into the nvram location indicated
                                   by the address in byte shift
                                   register, the return value for this
                                   operation is 0 for unsuccessful,
                                   non-zero, for successful */

#define     READ_NVR    5          /* read the byte
                                   from the nvram location indicated
                                   by the address in byte shift
                                   register, and store it.
                                   The return value for this
                                   operation is 0 for unsuccessful,
                                   non-zero, for successful.
                                   Data that has been read is available
                                   using the GET_NVR command */

#define     GET_NVR         6          /* return whatever is in nvr storage */

#define     SET_BAUD_RATE   9          /* set the baud rate of the uart
                                   specified in parameter to the
                                   rate specified in the auxiliary
                                   shift register
                                   return=parameter & 00001111B */

#define     GET_BAUD_RATE   10         /* get the baud rate of the uart
                                   specified in parameter.
                                   return=baud rate */

      #define     B300  0
      #define     B600  1
      #define     B1200 2
      #define     B2400 3
      #define     B4800 4
      #define     B9600 5
      #define     BBAD  6

      #define     B300_VAL    0X44
      #define     B600_VAL    0X55
      #define     B1200_VAL   0X66
      #define     B2400_VAL   0X88
      #define     B4800_VAL   0X99
      #define     B9600_VAL   0XBB


#define     SET_PARITY  11    /* channel in parameter, parity value in
                           byte1,   return= parameter */

#define     GET_PARITY  12    /* channel in parameter, return = parity */

      #define     NO_PARITY   0
      #define     EVEN_PARITY 1
      #define     ODD_PARITY  2
      #define     BAD_PARITY  3

      #define     NO_PARITY_VAL     19
      #define     EVEN_PARITY_VAL   3
      #define     ODD_PARITY_VAL    7
```

```
#define     DISABLE_UART      13
#define     SET_POSITION      14
#define     GET_POSITION      15

#define     SET_INT           16          /* parameter holds int choice
                                  returns parameter */


      #define     INT_10            0
      #define     INT_15            1
      #define     INT_3       2
      #define     INT_5       3
      #define     INT_7       4
      #define     NO_INT            5

      #define     INT_10_VAL  5
      #define     INT_15_VAL  4
      #define     INT_3_VAL   3
      #define     INT_5_VAL   2
      #define     INT_7_VAL   1


#define     SET_SYNC    17
#define     SET_CLOCK   20
#define     GET_CAPTURE 100
```

```
/* file: MDIGISUP.C                                    (R. Cutler 04/30/91)
*/

/*****************************************************************************
                              INCLUDE FILES

*****************************************************************************/
#include    <stdio.h>
#include    <conio.h>
#include    <string.h>
#include    <ctype.h>
#include    <dos.h>
#include    <malloc.h>
#include    <time.h>

#include    "drcom.h"
#include    "mdigisup.h"
#include    "merror.h"
#include    "xdetect.h"

/*===========================================================================*
 *                              define                                       *
 *===========================================================================*/
/* local defines                                                             */
#define     DR_OKAY                 1
#define     DMA_OKAY            0
#define     OKAY                0
#define     BAD                 1
#define     FALSE               0
#define TRUE                    1

/* INT 21 call parameters */
#define     GET_IRQ_VEC 0x35
#define     SET_IRQ_VEC 0x25

/* Interrupt controller constant and registers */
#define     INTA1 0x21
#define     INTB1 0xa1


/*===========================================================================*
 *                              function pre definitions                     *
 *===========================================================================*/
void far irq_entry();
void(far * ie_ptr)();
```

```
/*=============================================================================*
 *                              global variables                               *
 *=============================================================================*/

/* arrays for dynamic choice of interrupts and dma channels */
static int pagereg[8]={ 0x87, 0x83, 0x81, 0x00, 0x82, 0x8b, 0x89, 0x8a};
static int addr[8]   ={ 0x00, 0x02, 0x04, 0x06, 0xc0, 0xc4, 0xc8, 0xcc};
static int cntreg[8] ={ 0x01, 0x03, 0x05, 0x07, 0xc2, 0xc6, 0xca, 0xce};
static int chenb[8]  ={ 0, 1, 2, 3, 0, 1, 2, 3};
static int chdsb[8]  ={ 4, 5, 6, 7, 4, 5, 6, 7};
static int statmsk[8]={ 1, 2, 4, 8, 1, 2, 4, 8};
static int inmode[8] ={ 0x44, 0x45, 0x46, 0x47, 0x44, 0x45, 0x46, 0x47};
static int outmode[8]={ 0x48, 0x49, 0x4a, 0x4b, 0x48, 0x49, 0x4a, 0x4b};
static int statreg[8]={  0x8,  0x8,  0x8,  0x8, 0xd0, 0xd0, 0xd0, 0xd0};
static int maskreg[8]={  0xa,  0xa,  0xa,  0xa, 0xd4, 0xd4, 0xd4, 0xd4};
static int modereg[8]={  0xb,  0xb,  0xb,  0xb, 0xd6, 0xd6, 0xd6, 0xd6};
static int byteff[8] ={  0xc,  0xc,  0xc,  0xc, 0xd8, 0xd8, 0xd8, 0xd8};
static char irq[16]  ={ 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f,
                        0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77 };
static char enable_irq[16]
                      ={ 0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f,
                         0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f };
static int  ireg[16] ={INTA1,INTA1,INTA1,INTA1,INTA1,INTA1,INTA1,INTA1,
                       INTB1,INTB1,INTB1,INTB1,INTB1,INTB1,INTB1,INTB1};


static unsigned int     lastcomm;
unsigned int far  digirec;
static unsigned int     intchan=10;
static unsigned int     intflag=NO_INT;
static unsigned int     dmachan=6;
static      unsigned int      dmaflag=DISABLE_DMA;

union REGS  inregs,outregs;
struct SREGS      segregs;
unsigned    irqseg_sav,irqoff_sav;

/*=============================================================================*
 *                External Memory Variables                                    *
 *=============================================================================*/
PRIVATE unsigned long xm_lo_add,xm_hi_add;
PRIVATE     unsigned int xm_block_count, xm_pages_used;
PRIVATE     unsigned long _xmalloc(unsigned long byte_count);

PRIVATE     X_BUF far * ehead;
PRIVATE     X_BUF far * etail;
PRIVATE     X_BUF far * fhead;
PRIVATE     X_BUF far * ftail;
PRIVATE     X_BUF far * process;
PRIVATE     X_BUF far * collect;

PRIVATE     int   dma_err = DMA_OKAY;
```

```
/*=============================================================================*
 *                              collect_to_full                                *
 *=============================================================================*/
PRIVATE
collect_to_full()
{
if (collect != NULL)
      {
      if (ftail==NULL)
            {
            fhead=collect;
            }
      else
            {
            ftail->next=collect;
            }
      ftail=collect;
      ftail->next=NULL;
      collect=NULL;
      return(TRUE);
      }
else
      return(FALSE);
}

/*=============================================================================*
 *                                compute_time_nums                            *
 *=============================================================================*/
PRIVATE
int compute_time_nums(freq,lowl,lowh,highl,highh,incrementl,incrementh)
double freq;
int   *lowl, *lowh, *highl, *highh, *incrementl, *incrementh;
{
int   low;
int   high;
int   increment;
double r,s;

r=250.0 * freq;
low=(int)(r);
high=low+1;
s= r - (double) low ;
increment=(int)((65536.0*s)+0.5);
*lowl=low&255;
*lowh=low >> 8;
*highl=high&255;
*highh=high >> 8;
*incrementl=increment&255;
*incrementh=increment >> 8;

}

/*=============================================================================*
 *                                dr_128K_blocks_used                          *
 *=============================================================================*/
PUBLIC
unsigned int dr_128K_blocks_used()
{
return(xm_pages_used);
}
```

```
/*=============================================================================*
 *                                dr_dma_status                                *
 *=============================================================================*/
PUBLIC
int dr_dma_status()
{
int stat;

return( statmsk[dmachan] & inp(statreg[dmachan]) );

}

/*=============================================================================*
 *                                dr_full_status                               *
 *=============================================================================*/
PUBLIC
int   dr_full_status()
{

return(fhead!=NULL);
}

/*=============================================================================*
 *                                dr_get_dma_buffer                            *
 *=============================================================================*/
PUBLIC
unsigned long dr_get_dma_buffer()
{
int q;
int intflag;

/* wait for next buffer, return valid buffer address, or
      return 0 for over run or timeout */

while ((fhead==NULL))
      {
      if(dma_err!=DMA_OKAY) return(0L);
      }

/* if process is not NULL, then, transfer it to empty list before getting
   a new process buffer */
intflag=SysInt(0);      /* turn off interrupt */
process_to_empty();
SysInt(intflag);  /*restore interrupt */

intflag=SysInt(0);      /* turn off interrupt */
full_to_process();
SysInt(intflag);

return(process->pointer);
}
```

```
/*=============================================================================*
 *                                  dr_init_xm                                 *
 *=============================================================================*/
PUBLIC
int dr_init_xm(blk_cnt)
unsigned int blk_cnt;
{
union REGS irg,org;
unsigned long temp;

/* get # of 1K blocks of extended memory (beyond 1M) */
irg.h.ah=0x88;
xm_block_count=int86(0x15,&irg,&org);

if (xm_block_count<blk_cnt) er_abort(DR_EXT_MEM);

xm_lo_add=0x100000;
temp=(long)xm_block_count;
xm_hi_add=(temp * 1024)+xm_lo_add;
temp=(long)blk_cnt;
xm_lo_add=xm_hi_add - (temp * 1024);
xm_pages_used=0;
return(DR_OKAY);
}

/*=============================================================================*
 *          dr_init_xm_buff                                                    *
 *=============================================================================*/
PUBLIC
unsigned int dr_init_xm_buff(bs)
unsigned long bs;
{
unsigned long xadd;
unsigned int count,c1;
PRIVATE     X_BUF far * temp;

ehead=NULL;
etail=NULL;
fhead=NULL;
ftail=NULL;
collect=NULL;
count = 0;

while ( (xadd=_xmalloc(bs*2)) != 0L)
      {
      if ((process = (X_BUF far *) _fmalloc(sizeof(X_BUF)))== NULL)
            er_abort(DR_INT_MEM);
      process->length=bs;
      process->pointer=xadd;
      count += 1;
      process_to_empty();
      }
process = NULL;

return(count);
}
```

```
/*=============================================================================*
 *                                  dr_irq_restore                             *
 *=============================================================================*/
PUBLIC
void dr_irq_restore()
{
  unsigned i;

/* disable current interrupt */
    i = inp(ireg[intchan]);
    outp(ireg[intchan],(i | ~enable_irq[intchan]));

/* restore previous vector */
    inregs.h.ah = SET_IRQ_VEC;
    inregs.h.al = irq[intchan];
    inregs.x.dx = irqoff_sav;
    segregs.ds = irqseg_sav;
    intdosx(&inregs,&outregs,&segregs);
}

/*=============================================================================*
 *                                  dr_irq_setup                               *
 *=============================================================================*/
PUBLIC
void dr_irq_setup()
{
  unsigned i;
/* set up to get existing irq vector */
    inregs.h.ah = GET_IRQ_VEC;
    inregs.h.al = irq[intchan];
/* do it */
    intdosx(&inregs,&outregs,&segregs);
/* save values */
    irqseg_sav = segregs.es;
    irqoff_sav = outregs.x.bx;

/* set up to load a new irq vector */
    ie_ptr = irq_entry;
    inregs.h.ah = SET_IRQ_VEC;
    inregs.h.al = irq[intchan];
    inregs.x.dx = FP_OFF(ie_ptr);
    segregs.ds = FP_SEG(ie_ptr);
/* do it */
    intdosx(&inregs,&outregs,&segregs);

/* enable new interrupt */
    i = inp(ireg[intchan]);
    outp(ireg[intchan],(i & enable_irq[intchan]));
}
```

```
/*=============================================================================*
 *                          dr_isr                                             *
 *=============================================================================*/
PUBLIC
int dr_isr()          /* IRQ serv routine called from irq_entry in irq.asm */
{
int i;

      /* SERVICE INTERRUPT */

/* move collect buffer to a full list */
if (collect_to_full())
      {
      if (dma_err == DMA_OKAY)
            dma_err = dr_start_dma();
      }
else
      {
      dma_err = DR_DMA_BLOCKED;
      }
}


/*=============================================================================*
 *                                    dr_issue_command                         *
 *=============================================================================*/
PUBLIC
int dr_issue_command(comm,parm)
int   comm;
int   parm;
{
if (comm==lastcomm) dr_send_command( NOP,0);
return(dr_send_command(comm,parm));

}

/*=============================================================================*
 *                          dr_nvread                                          *
 *=============================================================================*/
PUBLIC
int dr_nvread(add,da)
int add;
int *da;
{
int status;
int lim;
int add1;

add1=255&(add>>8);
add=255&add;
lim=-1;
status=1;
while ((lim<10000) && (status != 0))
      {
      dr_issue_command(SHIFT_IN_BYTE,add1);
      dr_issue_command(SHIFT_IN_BYTE,add);
      status=dr_issue_command(READ_NVR,0);
      *da=dr_issue_command(GET_NVR,0);
      lim+=1;
      }
return(lim<10000);
}
```

```
/*=============================================================================*
 *                          dr_nvwrite                                         *
 *=============================================================================*/
PUBLIC
int dr_nvwrite(add,da)
int add;
int da;
{
int stat;
int rda;
int count=0;

da=da&255;
stat=nv_try_write(add,da);

while ((stat != 0) && (count<50))
      {
      stat=nv_try_write(add,da);
      count += 1;
      }
return(count<50);
}

/*=============================================================================*
 *                                    dr_reset                                 *
 *=============================================================================*/
PUBLIC
int   dr_reset()
{
int returnval;
int  recchar,j,a,d,i,b1,b2,b3,b4,b5,b6;
int c0,c1,c2,c3,c4,c5,c6;
long l1;
int parity;
double freq;
int adds[10];

recchar=33;
adds[0]=0;
adds[1]=1;
adds[2]=2;
adds[3]=3;
adds[4]=4;
adds[5]=5;
adds[6]=6;

freq=14.7456*1.1;
if (Time_set)
      {
      printf("CLOCK FREQUENCY CORRECTION\n\n");
      reset_digirec();
      if(!dr_sync_after_reset()) er_abort(DR_NO_DIGIREC);
      printf("\nType the actual frequency of the\n  oscillator in mHz (e.g.
14.7456):\n");
      scanf("%lf",&freq);
      compute_time_nums(freq,&b1,&b2,&b3,&b4,&b5,&b6);
      if (Debug_enabled)
      printf("b0=%02x, b1=%02x, b2=%02x, b3=%02x, b4=%02x, b5=%02x, b6=%02x\n",
            recchar,b1,b2,b3,b4,b5,b6);
      c0=c1=c2=c3=c4=c5=c6=0;
      if ( !dr_nvwrite(adds[0],recchar) ) printf("nvw error b0\n");
      if ( !dr_nvwrite(adds[1],b1) ) printf("nvw error b1\n");
      if ( !dr_nvwrite(adds[2],b2) ) printf("nvw error b2\n");
      if ( !dr_nvwrite(adds[3],b3) ) printf("nvw error b3\n");
      if ( !dr_nvwrite(adds[4],b4) ) printf("nvw error b4\n");
      if ( !dr_nvwrite(adds[5],b5) ) printf("nvw error b5\n");
```

```
      if ( !dr_nvwrite(adds[6],b6) ) printf("nvw error b6\n");
      if (i=dr_nvread(adds[0],&c0) == 9) printf("nvr error %d c0\n",i);
      if (i=dr_nvread(adds[1],&c1) == 9) printf("nvr error %d c1\n",i);
      if (i=dr_nvread(adds[2],&c2) == 9) printf("nvr error %d c2\n",i);
      if (i=dr_nvread(adds[3],&c3) == 9) printf("nvr error %d c3\n",i);
      if (i=dr_nvread(adds[4],&c4) == 9) printf("nvr error %d c4\n",i);
      if (i=dr_nvread(adds[5],&c5) == 9) printf("nvr error %d c5\n",i);
      if (i=dr_nvread(adds[6],&c6) == 9) printf("nvr error %d c6\n",i);
      if (Debug_enabled)
            printf("c0=%02x, c1=%02x, c2=%02x, c3=%02x, c4=%02x, c5=%02x,
c6=%02x\n",
                  c0,c1,c2,c3,c4,c5,c6);
      printf("Press any key to stop\n");
      while(!kbhit());
      getch();
      exit(0);
      }

reset_digirec();
if(!dr_sync_after_reset()) er_abort(DR_NO_DIGIREC);

if (Global_parity == 0)
      parity=NO_PARITY;
else if (Global_parity == -1)
      parity=ODD_PARITY;
else
      parity=EVEN_PARITY;

for ( i=0 ; i<16 ; i++)
      {
      dr_set_sync_char(i,33);
      dr_set_parity(i,parity);
      }

/* retrieve 6 frequency bytes from nvram  and recognition character */
i=dr_nvread(adds[0],&c0);
c1 = dr_nvread(adds[1],&b1);
c2 = dr_nvread(adds[2],&b2);
c3 = dr_nvread(adds[3],&b3);
c4 = dr_nvread(adds[4],&b4);
c5 = dr_nvread(adds[5],&b5);
c6 = dr_nvread(adds[6],&b6);

/* if we do not trust nvram data, compute standard frequency bytes */
if ((!i) || (c0!=recchar) || !c1 || !c2 || !c3 || !c4 || !c5 || !c6)
      {
      freq=14.7456;
      compute_time_nums(freq,&b1,&b2,&b3,&b4,&b5,&b6);
      }

/* adjust digirec clock */
dr_set_clock(b1,b2,b3,b4,b5,b6);

return(TRUE);
}
```

```
/*=============================================================================*
 *                                  dr_send_command                            *
 *=============================================================================*/
PRIVATE
int dr_send_command(comm,parm)
int   comm;
int   parm;
{
unsigned int status;

lastcomm=comm;
output_word(digirec,((comm<<8)+(parm&0xff)));
status=input_word(digirec);
while((((status>>8)&0xff) != comm) && !kbhit())
      {
      output_word(digirec,((comm<<8)+(parm&0xff)));
      status=input_word(digirec);
      }
return(status&0xff);
}

/*=============================================================================*
 *                                  dr_set_baud_rate                           *
 *=============================================================================*/
dr_set_baud_rate(comm,parm)
int comm;
int parm;
{

if ((comm> -1)&&(comm<16))
      {
      switch(parm)
            {
            case B300:
            case B600:
            case B1200:
            case B2400:
            case B4800:
            case B9600:
                  break;
            default:
                  return(FALSE);
            }
      dr_issue_command(SHIFT_IN_BYTE,parm);
      dr_issue_command(SET_BAUD_RATE,comm);
      return(TRUE);
      }
else
      return(FALSE);
}
```

```
/*=============================================================================*
 *                                  dr_set_clock                               *
 *=============================================================================*/
dr_set_clock(b1,b2,b3,b4,b5,b6)
int b1,b2,b3,b4,b5,b6;
{
int lowcount,highcount,incrmt;

      dr_issue_command(SHIFT_IN_BYTE,b6);
      dr_issue_command(SHIFT_IN_BYTE,b5);
      dr_issue_command(SHIFT_IN_BYTE,b4);
      dr_issue_command(SHIFT_IN_BYTE,b3);
      dr_issue_command(SHIFT_IN_BYTE,b2);
      dr_issue_command(SHIFT_IN_BYTE,b1);
      dr_issue_command(SET_CLOCK,0);
      return(TRUE);
}

/*=============================================================================*
 *                                  dr_set_dma_chan                            *
 *=============================================================================*/
PUBLIC
int dr_set_dma_chan(chan)
int chan;
{

dmachan=chan;

switch(chan)
      {
      case 5:
            dmaflag=ENABLE_DMA_CHANNEL_5;
            break;
      case 6:
            dmaflag=ENABLE_DMA_CHANNEL_6;
            break;
      case 7:
            dmaflag=ENABLE_DMA_CHANNEL_7;
            break;
      default:
            dmaflag=DISABLE_DMA;
            return(FALSE);
            break;
      }

return(TRUE);
}
```

```
/*=============================================================================*
 *                                  dr_set_int_chan                            *
 *=============================================================================*/
PUBLIC
int dr_set_int_chan(chan)
int chan;
{

intchan=chan;
/* check interrupt port */
switch(intchan)
      {
      case 10:
            intflag=INT_10;
            break;
      case 15:
            intflag=INT_15;
            break;
      default:
            intflag=NO_INT;
            break;
      }

}

/*=============================================================================*
 *                                  dr_set_io_add                              *
 *=============================================================================*/
PUBLIC
int dr_set_io_add(add)
int add;
{
digirec=add;
}

/*=============================================================================*
 *                                  dr_set_parity                              *
 *=============================================================================*/
PUBLIC
int dr_set_parity(comm,parm)
int comm;
int parm;
{

if ((comm> -1)&&(comm<16))
      {
      switch(parm)
            {
            case NO_PARITY:
            case EVEN_PARITY:
            case ODD_PARITY:
                  break;
            default:
                  return(FALSE);
            }
      dr_issue_command(SHIFT_IN_BYTE,parm);
      dr_issue_command(SET_PARITY,comm);
      return(TRUE);
      }
else
      return(FALSE);
}
```

```
/*=============================================================================*
 *                                  dr_set_position                            *
 *=============================================================================*/
PUBLIC
void dr_set_position (chan,wrd,pos)
int chan;
int wrd;
int pos;
{
int   num,i,j;
if ((pos>63)||(pos<0)) pos=64;
dr_issue_command(SHIFT_IN_BYTE,pos);
dr_issue_command(SET_POSITION,((chan*4)+wrd));
}

/*=============================================================================*
 *                                  dr_set_sync_char                           *
 *=============================================================================*/
dr_set_sync_char(comm,parm)
int comm;
int parm;
{

if ((comm> -1)&&(comm<16))
      {
      dr_issue_command(SHIFT_IN_BYTE,parm);
      dr_issue_command(SET_SYNC,comm);
      return(TRUE);
      }
else
      return(FALSE);
}

/*=============================================================================*
 *                                  dr_start_acq                               *
 *=============================================================================*/
PUBLIC
dr_start_acq()
{
int errflag,i,count,instrcnt;
long temp;

if (dmaflag == DISABLE_DMA) er_abort(DR_DMA_WRONG_CHANNEL);
if (intflag == NO_INT) er_abort(DR_INT_WRONG_CHANNEL);

/* turn off digirec dma */
instrcnt=dr_issue_command(SET_DMA,DISABLE_DMA);

/* enable digirec to cause interrupt */
dr_issue_command(GENERAL,DISABLE_INT); /*THIS CLEARS INTERRUPT FLAG */
dr_issue_command(SET_INT,intflag);
dr_issue_command(GENERAL,ENABLE_INT);

/* enable interrupt at the pc */
dr_irq_setup();

/* set up pc dma */
if( (dma_err = dr_start_dma()) != DMA_OKAY)
      {
      dr_stop_acq();
      er_abort(dma_err);
      }

/* turn on digirec dma */
dr_issue_command(SET_DMA,dmaflag);
}
```

```
/*=============================================================================*
 *                                  dr_start_dma
 *   memadd is a 32 bit address (only 24 bits usable)
 *   count is up to 64K (0 == 64K)
 *=============================================================================*/
PUBLIC
int dr_start_dma()
{
long memadd;
unsigned int count;
int adjustedcount;

/* allow only channels 5,6 and 7 for now */
if (dmaflag==DISABLE_DMA) return(DR_DMA_WRONG_CHANNEL);

/* get new xm buffer */
if (ehead==NULL) return(DR_DMA_BLOCKED);
empty_to_collect();

memadd=collect->pointer;
count=collect->length;

/* temporarily disable channel */

outp(maskreg[dmachan],chdsb[dmachan]);


/* set up page register */
outp(pagereg[dmachan], memadd>>16);

/* reset byte flip/flop */
outp(byteff[dmachan],0);

/* load address */
outp(addr[dmachan],memadd>>1);      /* low byte first */
outp(addr[dmachan],memadd>>9);      /* high byte second */

/* load adjusted count */
adjustedcount=count-1;
outp(cntreg[dmachan],adjustedcount);
outp(cntreg[dmachan],adjustedcount>>8);

/* set direction */
outp(modereg[dmachan],inmode[dmachan]);

/* and enable dma */
outp(maskreg[dmachan],chenb[dmachan]);

return(DMA_OKAY);
}
```

```
/*=============================================================================*
 *                                  dr_stop_acq                                *
 *=============================================================================*/
dr_stop_acq()
{
/* turn off digirec dma */
dr_issue_command(SET_DMA,DISABLE_DMA);

/* turn of digirec interrupt */
dr_issue_command(SET_INT,NO_INT);

/* restore pc interrupt vectors */
dr_irq_restore();

/* disable dma */
dr_stop_dma();

return(TRUE);
}

/*=============================================================================*
 *                                  dr_stop_dma                                *
 *=============================================================================*/
PUBLIC
int dr_stop_dma()
{

/* allow only channels 5,6 and 7 for now */
if (dmaflag == DISABLE_DMA) return(DR_DMA_WRONG_CHANNEL);

outp(maskreg[dmachan],chdsb[dmachan]);
return(DMA_OKAY);
}

/*=============================================================================*
 *                                dr_sync_after_reset                          *
 *=============================================================================*/
PRIVATE
int dr_sync_after_reset()
{
int notimeout;
long timeout_count;

timeout_count=10000L;

/* send a nop until you see a nop return */
do    {
      output_word(digirec,(NOP<<8));
      timeout_count -= 1L;
      notimeout= (timeout_count != 0L);
      }
while ((notimeout) && ((input_word(digirec)>>8) != NOP));

lastcomm=NOP;
return (notimeout);
}
```

```
/*=============================================================================*
 *                              empty_to_collect                               *
 *=============================================================================*/
PRIVATE
empty_to_collect()
{
if (ehead != NULL)
      {
      collect = ehead;
      ehead=ehead->next;
      collect->next=NULL;
      if (ehead == NULL)
            etail = NULL;
      return(TRUE);
      }
else
      return(FALSE);
}

/*=============================================================================*
 *                                  fpta (far pointer to absolute)             *
 *=============================================================================*/
PUBLIC
long fpta (point)
int far *point;
{

return( ( ( (long)point & 0xffff0000L) >> 12 ) + ( (long)point & 0xffff) );
}

/*=============================================================================*
 *                              full_to_process                                *
 *=============================================================================*/
PRIVATE
full_to_process()
{
if (fhead != NULL)
      {
      process = fhead;
      fhead=fhead->next;
      process->next = NULL;
      if (fhead == NULL)
            ftail = NULL;
      return(TRUE);
      }
else
      return(FALSE);
}

/*=============================================================================*
 *                          nv_try_write                                       *
 *=============================================================================*/
nv_try_write(add,da)
int add;
int da;
{
dr_issue_command(SHIFT_IN_BYTE,(add>>8));
dr_issue_command(SHIFT_IN_BYTE,add);

return(dr_issue_command(WRITE_NVR,da));

}
```

```
/*=============================================================================*
 *                              process_to_empty                               *
 *=============================================================================*/
PRIVATE
process_to_empty()
{
if (process != NULL)
      {
      if (etail==NULL)
            {
            ehead=process;
            }
      else
            {
            etail->next=process;
            }
      etail=process;
      etail->next=NULL;
      process=NULL;
      return(TRUE);
      }
else
      return(FALSE);
}

/*=============================================================================*
 *                                  reset_digirec                              *
 *=============================================================================*/
PRIVATE
int   reset_digirec()
{
input_word(digirec+4);
}
```

```
/*=============================================================================*
 *                                  _xmalloc                                   *
 *=============================================================================*/
PRIVATE
unsigned long _xmalloc(byte_count)
unsigned long byte_count;
{
long temp,page_bottom;

/* if count is greater than 128K, exit with 0 start address */
if (byte_count > 0x00020000L) return (0L);

/* if count is greater than remaining words in current 128K page,
      go to next lower page */
page_bottom=xm_hi_add & 0xfffe0000L;
if ((xm_hi_add - page_bottom)<byte_count)
      {
      page_bottom -= byte_count;
      /* if not enough room left, send back 0 */
      if (page_bottom < xm_lo_add)
            return(0L);
      else /* cross page boundary and allocate buffer */
            {
            xm_hi_add = page_bottom;
            xm_pages_used += 1;
            return(xm_hi_add);
            }
      }
else
      {
      xm_hi_add -= byte_count;
      if (xm_hi_add < xm_lo_add)
            {
            xm_hi_add += byte_count;
            return(0L);
            }
      else
            return(xm_hi_add);
      }
}
```

```
;*************************************************************************
;
; FILE: mdigiasm.asm                          (R. Cutler 04/30/91)
;
;     The following c callable routines in assembly language:
;
;     irq_entry() interrupt routine to handle digirec interrupt
;                  does some interrupt housekeeping and calls
;                  the c routine "dr_isr"
;
;     input_word(int address)
;                 routine in input a full word from an i/o port
;                  the address is normally in the range of 0-3FF
;                  the return value is the contents of the i/o port
;
;     outpu_word(int address, int data)
;                 routinge to output a full word to an i/o port
;                  the address is normally in the range of 0-3FF
;                  the return value is indeterminate (currently
;                  it reflects the value of data written to i/o)
;
;     getxmem(long source_add, long destination_add, int word_count)
;                 routine to transfer data words (16 bits) between
;                  memory locations, including extended memory
;
;     SysInt(int flag)
;                 routine to set or clear the interrupt flag
;                  if flag is 0200H, interrupts are enabled
;                  if flag is 0000H, interrupts are disabled
;                  the state of the system interrupt flag, before
;                  being altered, is the return value for this routine.
;
;*************************************************************************

;define as medium model:
M       equ 0

            include model.inc

DGROUP      GROUP _DATA
_TEXT       segment     byte public 'CODE'
            assume      cs:_TEXT,ds:_DATA
```

```
;*****************************************************************
;
;           _irq_entry
;
;*****************************************************************
       PUBLIC  _irq_entry

       EXTRN   _dr_isr:near

FAR_BSS     segment word
      extrn _digirec:word
FAR_BSS     ends


            assume      cs:_TEXT,ds:DGROUP

_irq_entry proc far

      push  bp
      push  ax
      push  bx
      push  cx
      push  dx
      push  ds
      push  es
      push  di
      push  si
      push  sp

; enable other interrupts
      sti

; set up for far data
      push  es
      mov   ax,FAR_BSS
      mov   es,ax
      assume es:FAR_BSS


; restore ds
      pop   es

            assume      cs:_TEXT,ds:DGROUP
;inform both interrupt controllers
;
      mov   al,20h            ;non specific eoi for both controllers
      out   20h,al            ;send to first controller
      out   0a0h,al           ;send to second controller

;insure that ds is pointing to the right segment
      mov   ax,DGROUP
      mov   ds,ax

; call c routine to handle data
      push  cs
      call  _dr_isr

; set up for far data
      mov   ax,FAR_BSS
      mov   es,ax
      assume es:FAR_BSS
```

```
;clear digirec interrupt
      mov   dx,es:_digirec    ;base address of digirec
      inc   dx          ;increment by 2 to interrupt off flag
      inc   dx
      in    ax,dx       ;clear by reading, no valid data

; restore registers

      pop   sp
      pop   si
      pop   di
      pop   es
      pop   ds
      pop   dx
      pop   cx
      pop   bx
      pop   ax
      pop   bp
       iret
_irq_entry endp


;*****************************************************************
;
;           _input_word
;
;*****************************************************************
            public      _input_word

_DATA       segment word
p1          struc                   ; stack structure
            dw    ?                 ; old bp
            dw    return_size dup (?)     ; return address
iaddress    dw    ?                 ;input address
p1          ends
_DATA       ends

            assume      cs:_TEXT,ds:DGROUP
PROCEDURE   _input_word
            push  bp                ; save registers
            mov   bp,sp
            push  dx

            mov   dx,[bp].iaddress  ;get address
            in    ax,dx

            pop   dx                ; restore registers
            pop   bp
            ret
_input_word       endp
```

```
;*****************************************************************
;
;           _output_word
;
;*****************************************************************
            public      _output_word

_DATA       segment word
p2          struc                   ; stack structure
            dw    ?                 ; old bp
            dw    return_size dup (?)     ; return address
oaddress    dw    ?                 ;output address
data        dw    ?                 ;data
p2          ends
_DATA       ends

            assume      cs:_TEXT,ds:DGROUP
PROCEDURE   _output_word
            push  bp                ; save registers
            mov   bp,sp
            push  dx

            mov   dx,[bp].oaddress  ;get address
            mov   ax,[bp].data            ;get data
            out   dx,ax

            pop   dx                ; restore registers
            pop   bp
            ret
_output_word            endp

;*****************************************************************
;
;           _getxmem
;
;*****************************************************************
            public      _getxmem

_DATA       segment word
p3          struc                   ;stack structure
            dw    ?                 ;old bp
            dw    return_size dup (?)     ;return address
souradd           dw    ?
souraddhi   dw    ?
destadd           dw    ?
destaddhi   dw    ?
wdcnt       dw    ?
p3          ends

p4          struc
dummy       dw    8 dup (?)
sptr        dw    ?
slo         dw    ?
shi         db    ?
sac         db    ?
sresv       dw    ?
dptr        dw    ?
dlo         dw    ?
dhi         db    ?
dac         db    ?
p4          ends


_mygdt            dw    48 dup (?)

_DATA       ends
```

```
            assume      cs:_TEXT,ds:DGROUP
PROCEDURE   _getxmem
            push  bp                ; save registers
            mov   bp,sp
            push  ds
            push  es
            push  bx
            push  cx
            push  si
            push  di
; set up extra segment
            mov   ax,seg _mygdt
            mov   es,ax
; zero dummy
            mov   ax,0
            mov   bx,offset _mygdt
            mov   es:[bx].dummy,ax
            mov   es:[bx].dummy+2,ax
            mov   es:[bx].dummy+4,ax
            mov   es:[bx].dummy+6,ax
; load memory segment - 1 and access code
            mov   ax,0ffffh
            mov   es:[bx].sptr,ax
            mov   es:[bx].dptr,ax
            mov   al,093h
            mov   es:[bx].sac,al
            mov   es:[bx].dac,al

; load source address
            mov   ax,[bp].souradd
            mov   es:[bx].slo,ax
            mov   ax,[bp].souraddhi
            mov   es:[bx].shi,al

; load destination address
            mov   ax,[bp].destadd
            mov   es:[bx].dlo,ax
            mov   ax,[bp].destaddhi
            mov   es:[bx].dhi,al

; set up source index
            mov   si,bx

; set up word count
            mov   cx,[bp].wdcnt

; set up command
            mov   ah,087h           ;move block command

; issue interrupt 15
            int   015h

            mov   al,0
            xchg  al,ah
;pop and return
gtxmx:
            pop   di
            pop   si
            pop   cx
            pop   bx
            pop   es
            pop   ds
            pop   bp
            ret
_getxmem    endp
```

```
;*****************************************************************
;
;           _SysInt
;
;*****************************************************************
            PUBLIC      _SysInt

_DATA       segment word
SysIntp     struc                   ; stack structure
            dw    ?                 ; old bp
            dw    return_size dup (?)     ; return address
intflag           dw    ?                 ;first word argument
            dw    ?                 ;second word argument
SysIntp     ends


_DATA       ends

            assume      cs:_TEXT,ds:DGROUP
PROCEDURE   _SysInt
            push  bp
            mov   bp,sp

;save flags for use in setting return value
            pushf

            mov   ax,[bp].intflag
            or    ax,ax
            jne   si1
            cli
            jmp   si2
si1:        sti
si2:
; set return value to interrupt flag, before adjusting it
            pop   ax
            and   ax,0200h

            pop   bp
            ret
_SysInt     endp

_TEXT       ends
            end
```

```
# FILE: xdetect                                        (R. Cutler   04/30/91)
#
# This is a make file for xdetect.c.  The file is set up to make a medium
# model version.
#
# HISTORY:
#       none
#
# For max. optimization:
# cl /Ox
# link /PAC /F /E
#
# For Codeview:
# cl /Zi /Od
# link /CO /m

# Video selection.  MONO = Hercules graphics, EGA = EGA graphics,
#                   VGA  = EGA graphics on VGA
#
# Note that the video selection may also be defined from the make command line,
# overriding the value set here.  An example would be:
#
#               > make video=EGA xdetect
#
# *NO* spaces are allowed between the = sign and the video selection string,
# and the selection string *MUST* be in uppercase.

video=MONO

.c.obj:
        cl /c /D$(video) /Ox /AM /FPc $*.c

.asm.obj:
        masm /Dmedium /V /Z /MX $*;

.for.obj:
        fl /c /AM /FPc $*.for

xdetect.obj:    xdetect.c mconst.h mdemux.h mdsp.h mdt28xx.h mfreerun.h\
                merror.h mfile.h mlocate.h mlog.h mparse.h mreboot.h\
                mscreen.h mscrnhdr.h mtrigger.h mutils.h xdetect.h timer.h

mbase36.obj:    mbase36.c mbase36.h mconst.h

mboot.obj:      mboot.asm

mdmux.obj:      mdmux.asm

mdemux.obj:     mdemux.c mconst.h mdemux.h mdigirec.h mdigisup.h\
            mdt28xx.h merror.h mlog.h mqueue.h\
                mscrnhdr.h msudsini.h mutils.h powrstor.h xdetect.h

mdigiasm.obj:     mdigiasm.asm model.inc

mdigirec.obj:   mdigirec.c mconst.h mdemux.h mdigirec.h mdigisup.h\
            merror.h mfile.h mlog.h mqueue.h mstation.h msudsini.h\
            mutils.h xdetect.h

mdigisup.obj:     mdigisup.c drcom.h mdigisup.h merror.h xdetect.h

mdsp.obj:       mdsp.c mconst.h mdemux.h mdsp.h merror.h mfile.h\
                mlog.h mscreen.h mutils.h xdetect.h

mdt28xx.obj:    mdt28xx.c mconst.h mdemux.h mdt28xx.h merror.h mfile.h\
                mlog.h mqueue.h mstation.h msudsini.h mutils.h xdetect.h
```

```
merror.obj:     merror.c mconst.h mdemux.h mdt28xx.h merror.h mlog.h mscreen.h

mevent.obj:     mevent.c mconst.h mevent.h

mfile.obj:      mfile.c mbase36.h mconst.h mdosfunc.h mdemux.h mdsp.h mdt28xx.h\
                merror.h mfile.h mlocate.h mlog.h mpick.h mscrnhdr.h mstation.h\
                msudsini.h mtrigger.h mutils.h xdetect.h

mfreerun.obj:   mfreerun.c mconst.h mdemux.h mfile.h mfreerun.h mscrnhdr.h

mhalfsp.obj:    mhalfsp.c mconst.h merror.h mhalfsp.h mlatlon.h mpick.h\
                mstation.h xdetect.h

mlatlon.obj:    mlatlon.c mconst.h mlatlon.h

mlexer.obj:     mlexer.c mconst.h mlexer.h

mlocate.obj:    mlocate.c mconst.h merror.h mhalfsp.h mlocate.h mqueue.h\
                msudsini.h mutils.h

mlog.obj:       mlog.c mconst.h merror.h mlog.h mutils.h

mparse.obj:     mparse.c mconst.h mdemux.h mdsp.h mdt28xx.h mfile.h mfreerun.h\
                mhalfsp.h mlexer.h mlocate.h mlog.h mparse.h mreboot.h\
                mscrnhdr.h mstation.h mtrigger.h xdetect.h

mpick.obj:      mpick.c mconst.h mdemux.h merror.h mqueue.h\
                msudsini.h mstation.h mtrigger.h mutils.h

mqueue.obj:     mqueue.c mconst.h merror.h mqueue.h

mreboot.obj:    mreboot.c mconst.h mdemux.h mreboot.h

mscreen.obj:    mscreen.c mconst.h mdemux.h mdsp.h mdt28xx.h merror.h mhelp.h\
                mlog.h mscreen.h mstation.h mtrigger.h mutils.h mvideo.h

mscrnhdr.obj:   mscrnhdr.c mconst.h mscrnhdr.h mutils.h mvideo.h xdetect.h

mstation.obj:   mstation.c mconst.h mdosfunc.h mdt28xx.h merror.h mlatlon.h\
                mqueue.h mstation.h msudsini.h mtrigger.h mutils.h

msudsini.obj:   msudsini.c mconst.h msudsini.h mutils.h

mtrigger.obj:   mtrigger.c mconst.h mdemux.h merror.h mqueue.h\
                mscreen.h mscrnhdr.h msudsini.h mtrigger.h mutils.h

mutils.obj:     mutils.c mconst.h merror.h mutils.h

powrstor.obj:   powrstor.c powrstor.h

pwrlolvl.obj:   pwrlolvl.asm

timer.obj:      timer.c mconst.h

xdetect.exe:    xdetect.obj mbase36.obj mboot.obj mdmux.obj mdemux.obj\
            mdigiasm.obj mdigirec.obj mdigisup.obj mdsp.obj mdt28xx.obj\
                merror.obj mfile.obj mfreerun.obj\
                mhalfsp.obj mlatlon.obj mlexer.obj mlocate.obj mlog.obj\
                mparse.obj mpick.obj mreboot.obj mqueue.obj mscreen.obj\
                mscrnhdr.obj mstation.obj msudsini.obj mtrigger.obj\
                mutils.obj powrstor.obj pwrlolvl.obj timer.obj
        link /nod /PAC /F /E @$(video).lnk
```

APPENDIX 3. MULTIPLEXER

DI-64x4-V1 Multiplexer Technical Reference Manual
Version 1.0

by

Dean Tottingham

June 29, 1991

TABLE OF CONTENTS

1. INTRODUCTION

Data Translation DT2800 series boards work well for those applications that require only a few high speed A/D channels. In fact, these boards support aggregate sampling rates between 50 and 150 Ksamples/second. For those applications whose analog signals require only low speed A/D channels, much of the board's bandwidth is wasted because the number of available A/D channels is totally insufficient.

The DT2827 supports four differential inputs at 100 Ksamples/second. For applications that require only 50 samples/second/channel, 99.8% of the available bandwidth is wasted. To support those applications that require far more input channels, the DI-64X4-V1 multiplexer board expands the number of input channels on the DT2827 from four to 64 and reduces the available bandwidth to 93.6% (which is still rather lousy).

This document describes how the DI-64x4-V1 multiplexer board works. It provides sufficient information for one to program and troubleshoot this board. It is not meant as a user's guide!

2. DESIGN GOALS

The DI-64x4-V1 multiplexer design is based on the following design goals:

- o 64 differential input channels
- o High signal-to-noise ratio to support DT2827's 16-bit dynamic range.
- o Sophisticated clocking hardware to eliminate 16th channel problem inherent in the USGS 128 X 16 multiplexer design.
- o Command oriented communication protocol between the application program and the multiplexer board.
- o Commands and supporting hardware to permit:
    1. Programmable number of banks
    2. Board self-identification
    3. Automatic erratic bank switching correction
- o No additional cabling between DT2827 and multiplexer board.

3. THEORY OF OPERATION

3.1 Communication Protocol

The DT2800 series boards have a single 16-bit digital I/O channel which is used to send commands to and read/write data from/to the multiplexer.

The 16-bit I/O channel is actually two 8-bit channels, DIO0 and DIO1, and the most significant byte (MSB) is DIO1. The individual bits within DIO0 and DIO1 are DIO0-0, DIO0-1,..., DIO0-7 and DIO1-0, DIO1-1,..., DIO1-7 respectively. The most significant bits (msb) of DIO0 and DIO1 are DIO0-7 and DIO1-7 respectively. Figure 1.0 shows DIO0 and DIO1 from the multiplexer's perspective. DIO1 is used as a read/write data port, and DIO0 is used as a write only command/control port.

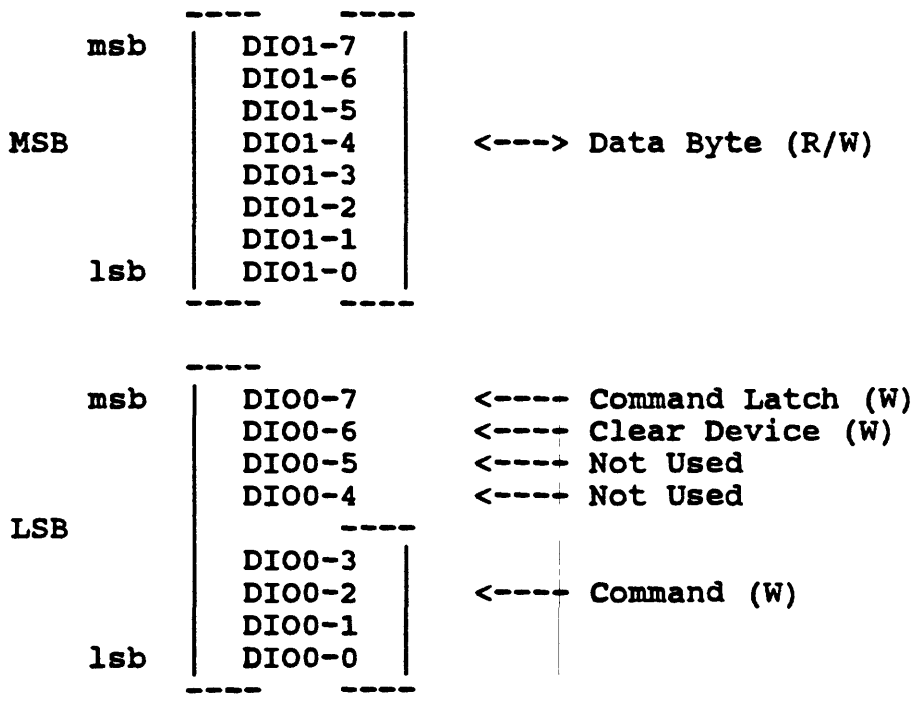


Figure 1.0

DIO0 and DIO1 may be independently written but can only be read as a pair. Unfortunately, during a read, the command byte within DIO0 is trashed (DIO0 floats). To circumvent this problem, the command nibble must be latched into the command register (U8 on the schematic) on the multiplexer board. To do this, the command latch bit (DIO0-7) must be pulled LOW and then pulled HIGH. During steady state conditions, the command latch bit is tied to a pull-up resistor which keeps it HIGH during read cycles.

Besides DIO0-7, the DIO0-6 bit also has special meaning. Pulling DIO0-6 low resets the multiplexer board. Because of this, care must be taken to ensure that this bit remain HIGH unless otherwise instructed by the user. During steady state conditions, the clear bit is tied to a pull-up resistor which keeps it HIGH during read cycles.

Some commands read a data byte from the multiplexer while other commands write a data byte to the multiplexer. The following pseudocode samples illustrate how one might instruct the multiplexer to initiate a read or write request.

- **Initiating a read request:**

  1. Latch the read command into the command register:

     a. Put the read command in DIO0's lower nibble,
        set DIO0-6 (clear bit) to 1, and
        set DIO0-7 (latch bit) to 0. (atomic)

     b. Set DIO0-6 (clear bit) to 1 and
        set DIO0-7 (latch bit) to 1. (atomic)

  2. Read 16-bit DIO port. Result is in DIO1.

- **Initiating a write request:**

  1. Latch the write command into the command register and place the data (to be written) in DIO1.

     a. Put the write command in DIO0's lower nibble,
        put the data (to be written) in DIO1,
        set DIO0-6 (clear bit) to 1, and
        set DIO0-7 (latch bit) to 0.

     b. Set DIO0-6 (clear bit) to 1 and
        set DIO0-7 (latch bit) to 1.

  2. Clear the command register to latch the data byte into the data register (U6 on the schematic).

     a. Clear DIO0's lower nibble,
        put the data (to be written) in DIO1,
        set DIO0-6 (clear bit) to 1, and
        set DIO0-7 (latch bit) to 0.

     b. Set DIO0-6 (clear bit) to 1 and
        set DIO0-7 (latch bit) to 1.

3.2 Command Decoding

As discussed in section 3.1, the multiplexer board is command driven. Some commands allow one to read from DIO1 while other commands allow one to write to DIO1. From the multiplexer's perspective, DIO1 is a bus; some commands instruct it to write to the bus while other commands instruct it to read from the bus.

Currently, the multiplexer board supports three commands. Each command listed below controls a specific hardware component on the multiplexer board:

Opcode	Name	Description
01	WRITE_NBANKS	Sets the number of banks that will be digitized.
02	READ_MUXID	Gets the multiplexer type.
03	READ_CURRENT_BANK	Gets the bank that is currently being digitized.

As described in section 3.1, a command is latched into the command register (74LS175) (U8) as a result of initiating a read or write request. The least significant two bits of the latched command (3Q and 4Q) are fed to the inputs of a 2-to-4 decoder (74LS139) (U9). The 2-to-4 decoder is enabled by the most significant two bits of the latched command (1Q and 2Q) which should both be LOW for these commands. The decoder is only enabled when these bits are both LOW which is ensured by the OR gate (74LS32) (U19). 1Q and 2Q feed the OR gate, and the output of the OR gate is fed into /2G of the decoder. The other decoder (the 74LS139 is a dual 2-to-4 decoder) is disabled by tying /1G to +5V.

The output of the 2-to-4 decoder are three signals which control representative hardware components associated with each command. The /2Y1 output of the 2-to-4 decoder is active as a result of initiating the WRITE_NBANKS command. It latches of the number of banks into the data register (U6). The /2Y2 output of the 2-to-4 decoder is active as a result of initiating the READ_MUXID command. It enables the tri-state buffer (74LS244) (U7) which writes the multiplexer id into the lower nibble of DIO1. The /2Y3 output of the 2-to-4 decoder is active as a result of initiating the READ_CURRENT_BANK command. It enables the tri-state buffer (74LS244) (U7) which writes the current bank being digitized into the lower nibble of DIO1.

3.3 Clock Generation

The clock generation circuit keeps the address counter synchronized with the DT2800 series board. It generates the clock pulse which stimulates the address counter to advance the current bank pointer.

The timing characteristics of the clock generation circuit are tightly coupled with the dynamics of address generation (channel pointer) circuitry on the DT2800 series board. The two key components of this circuit on the DT2800 are U27, a 74F161 (4-bit binary counter), and U18, a 74HCT85 (4-bit magnitude comparitor). The counter's /LD input signal (U27, pin 9) is tied to the comparitor's A=B signal through an inverter. As the counter counts, the magnitude comparitor is constantly comparing the output of the counter with the maximum value (which is one less than the number of channels being digitized per bank.) When the counter reaches this magic value, the magnitude comparitor sets its A=B output HIGH, and, on the next clock, the counter is reset to zero. A=B seems to be the signal we need to clock the multiplexer's counter. Almost.

Unfortunately, there is a slight twist to this otherwise perfect scheme. The A/D converter is not in step with the current output of the counter. In fact, the channel currently being digitized is NOT the one currently pointed to by the output of the counter. Therefore, we need to delay the effect of the A=B signal on clocking the multiplexer's counter until the digitization engine starts digitizing the zeroth channel of the current bank. As it turns out, the digitization engine does output the pointer to the channel currently being digitized. Conditioning the A=B trigger on the negative edge of the least significant bit of the current channel pointer (U29, pin 2) would solve this problem. Note that the actual A=B pulse may be long gone by the time we get a negative edge on this signal so we need to store the A=B pulse until we see the negative edge.

The pair of flip-flops in U3 are used to implement this circuit. The negative edge of the A=B pulse (or positive edge of /A=B) sets the first flip-flop (U3B). The output of the first flip-flop feeds the D input of the second flip-flop (U3A). A positive edge of the pulse on pin 2 of U29 (lsb of the current channel pointer and called CLK on the multiplexer schematics) will clock the second flip-flop which will generate a positive edge on the output (Q) of the second flip-flop. Since this output is connected to the clock input to the multiplexer's counter, it will trigger the counter to advance the current bank pointer. The inverted output (/Q) of the second flip-flop is fed back to the /CLR line of the first flip-flop which effectively clears the first flip-flop and, on the next pulse on CLK,

will also clear the second flip-flop in preparation for the next A=B, etc.

Note that U1 (74LS244) is used to buffer /A=B, CLK, and DIO0-6 (CLR).

3.4 Address Generation

The address generation circuit closely resembles the address generation circuit on the DT2800 series board as described in section 3.3.

The two key components of this circuit are U4, a 74LS161 (4-bit binary counter), and U5, a 74LS85 (4-bit magnitude comparitor). The counter's /LD input signal (U27, pin 9) is tied to the comparitor's A=B signal through an inverter. As the counter counts, the magnitude comparitor is constantly comparing the output of the counter with the value stored in the data register (which is one less than the number of banks). When the counter reaches this magic value, the magnitude comparitor sets its A=B output HIGH, and, on the next clock, the counter is reset to zero.

3.5 Multiplexer Operation

The DT2827 A/D board is capable of digitizing up to 4 differential signals concurrently. Unbeknownst to the A/D board, the multiplexer circuit allows it to digitize up to 64 differential channels concurrently. To do this, the multiplexer maintains 16 banks of 4 differential channels/bank and permits the A/D board to "see" only one bank at any given time. When the A/D board is finished digitizing the last channel in the current bank, the multiplexer feeds it the next bank of channels. And the saga continues.

Banks are mapped to channels in the following manner:

Bank	Channel List	
	First	Last
0	0	3
1	4	7
2	8	11
3	12	15
4	16	19
5	20	23
6	24	27
7	28	31
8	32	35
9	36	39
10	40	43
11	44	47
12	48	51
13	52	55
14	56	59
15	60	63

Each of the four differential inputs feeding the A/D board has a corresponding 16 X 1 differential multiplexer circuit on the multiplexer board. Although they are not available in chip form, a 16 X 1 differential multiplexer can be made out of two 8 X 1 differential multiplexer chips (DG507A). As one might recall, the DG507A chip has two outputs, D1 and D2, and an ENABLE input. By tying the most significant bit of bank address (ADR 3) to the ENABLE input of one DG507A and an inverted ADR 3 to the ENABLE input of the other DG507A, only one multiplexer within the pair is active at any given time. Because of this, the corresponding DG507A outputs can be tied together (D1 to D1 and D2 to D2), and the result is a single 16 X 1 differential multiplexer.

Analog signals are fed to the multiplexer board through eight 16 pin headers. Each header contains eight differential channels. Channels are mapped to headers as follows:

Header	Channel List First	Last
J1	0	7
J2	8	15
J3	16	23
J4	24	31
J5	32	39
J6	40	47
J7	48	55
J8	56	64

3.6 DC-DC Converter

The DC-DC converter circuit on the multiplexer board provides +/- 15V outputs at +/- 125 mA from a +5V input. This circuit is based on Maxim's MAX743 DC-DC converter chip.

The MAX743 typically provides 75% to 85% efficiency over most of the load range. It operates with current-mode feedback at 200kHz, so it can be used with small, lightweight external components. This chip is inherently reliable due to its internal power transistors and monolithic construction. Thermal shutdown prevents overheating.

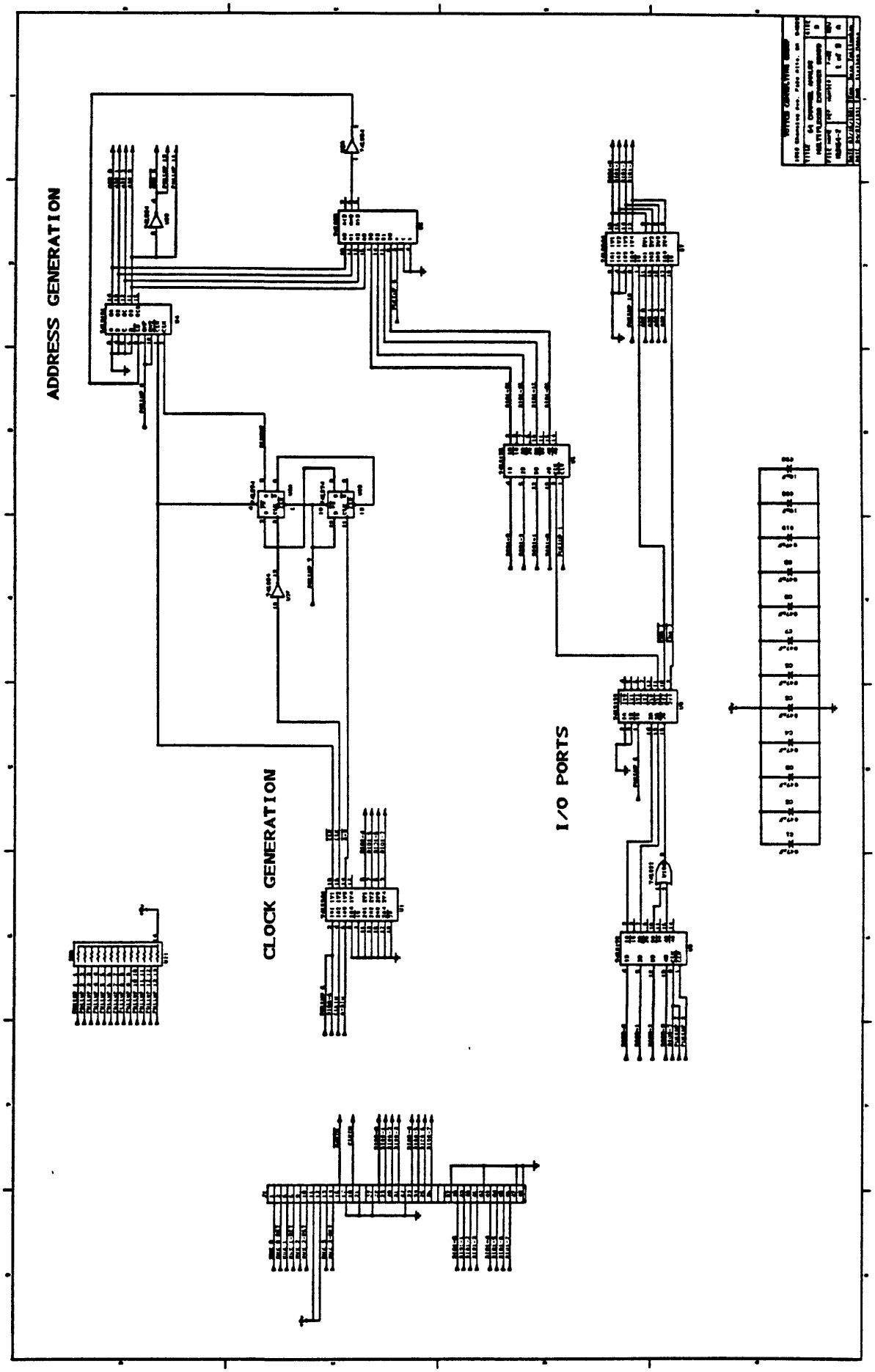
Details regarding the DC-DC converter design based on this chip can be obtained from Maxim Integrated Products.

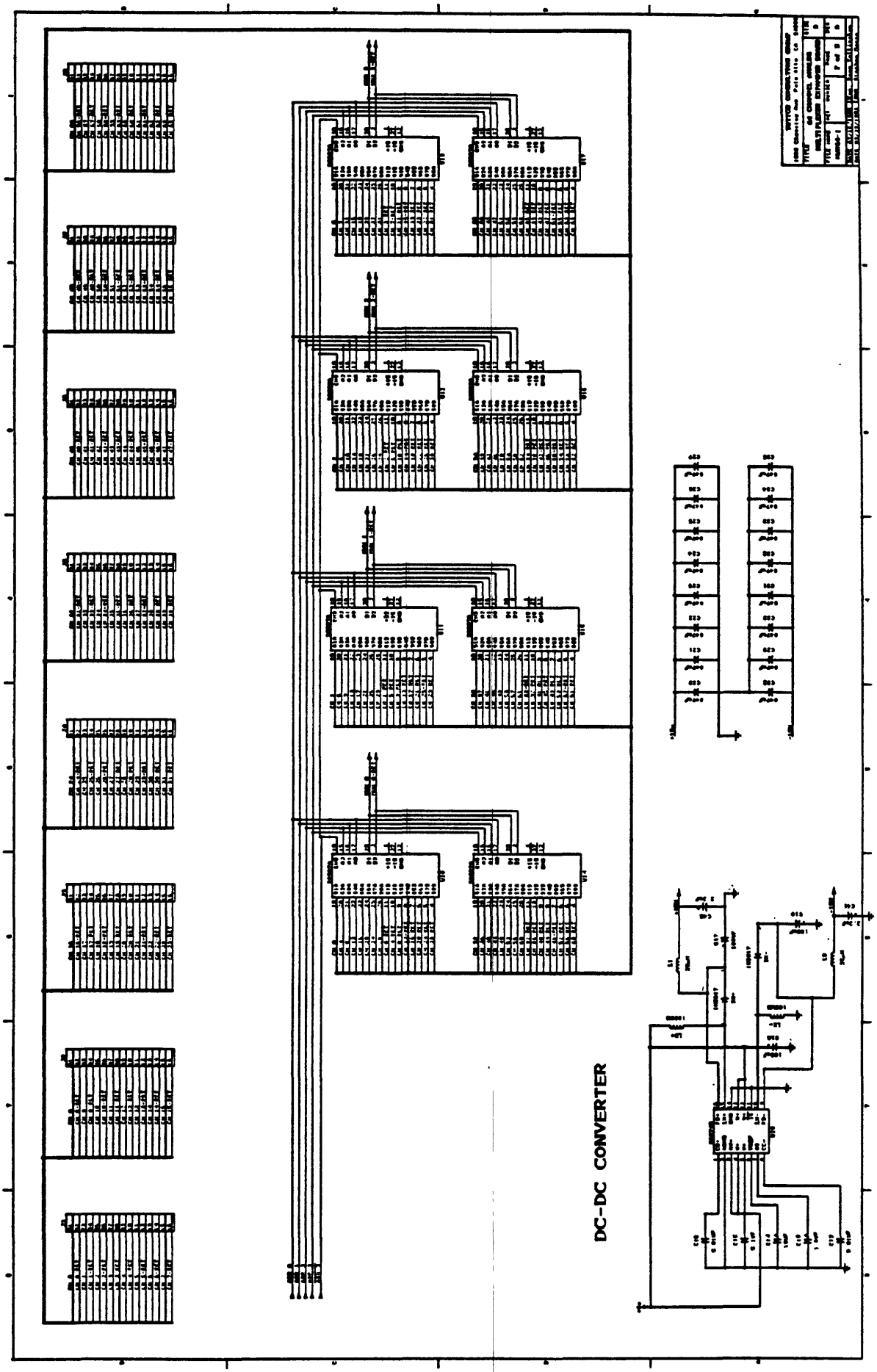
4. DT2827 Hardware Interface

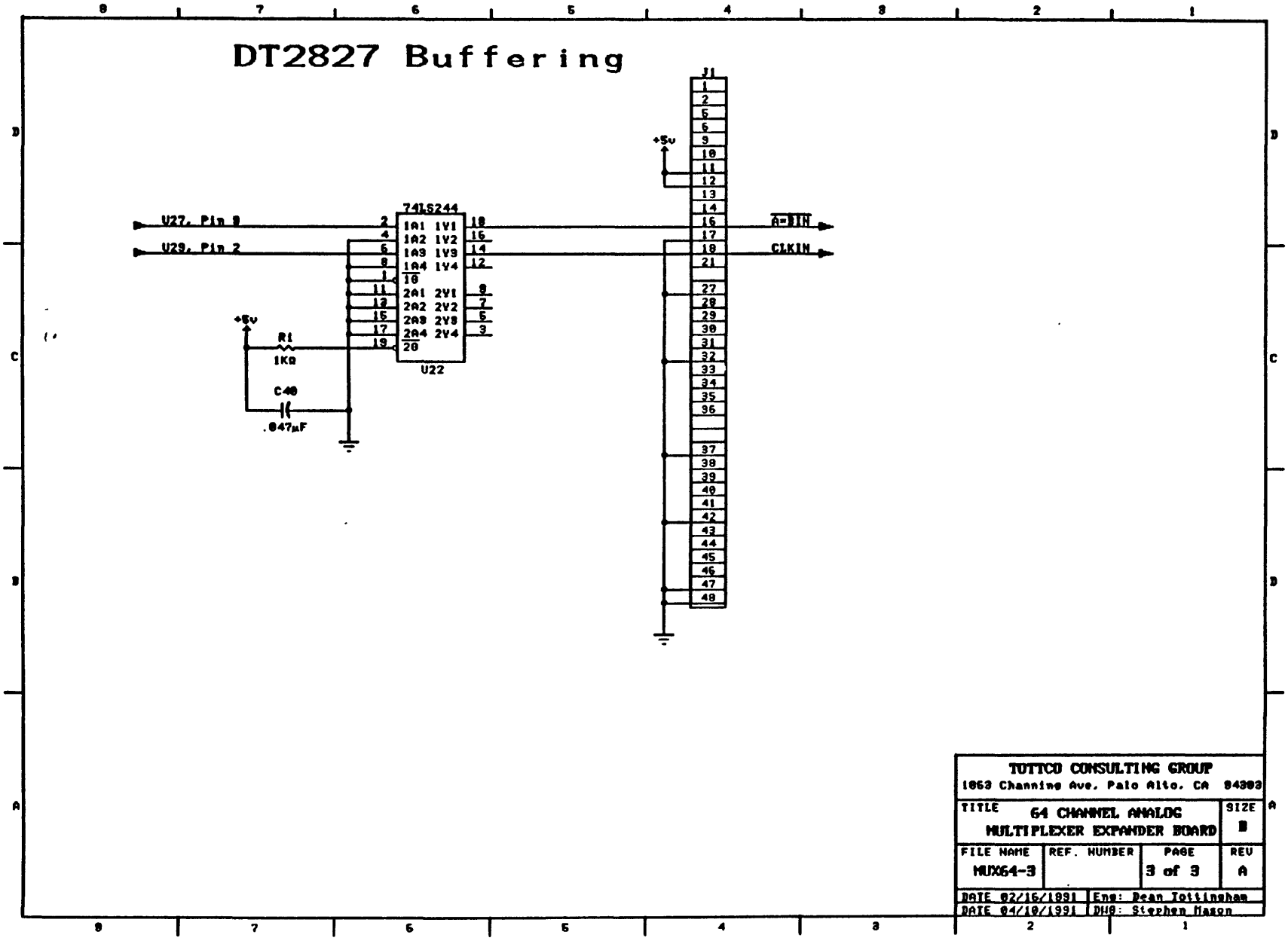
The multiplexer board needs several signals and +5V from the DT2827 board. Luckily, the 50 pin connector though which the DT2827 meets the outside world has several unused pins.

Pins 11 and 12 are used to feed +5V from the AT bus to the multiplexer board. /A=B (U27-pin 9) and CLKIN (U29-pin 2) are buffered through a 74LS244 before they are shipped out of pin 16 and pin 18 respectively.

APPENDIX A: Schematics


ADDRESS GENERATION
CLOCK GENERATION
I/O PORTS


DC-DC CONVERTER



TUTTCO CONSULTING GROUP 1063 Channing Ave. Palo Alto. CA 94303			
TITLE: 64 CHANNEL ANALOG MULTIPLEXER EXPANDER BOARD			SIZE: B
FILE NAME: MUX64-3	REF. NUMBER:	PAGE: 3 of 3	REV: A
DATE 02/16/1991	Eng: Dean Tottingham		
DATE 04/10/1991	DHG: Stephen Mason		

APPENDIX B: Parts List

#	Item	Part No.	Manufacturer	Qnty
1	IC	74LS04		1
2	IC	74LS32		1
3	IC	74LS74		1
4	IC	74LS85		1
5	IC	74LS139		1
6	IC	74LS161		1
7	IC	74LS175		2
8	IC	74LS244		3
9	IC	DG507A	Analog Devices	8
10	IC	761-1-R1K		1
11	KIT	MAX743CPEKIT	Maxim	1
12	Capacitor	0.047uF, ceramic		26
13	Capacitor	10uF, tanalum		2
14	PCB	DI-64x4-V1 PCB	TCG	1
15	16-pin connector, male		Dupont	8
16	50-pin connector, male		Dupont	1
17	28-pin IC socket		AMP	8
18	20-pin IC socket		AMP	3
19	16-pin IC socket		AMP	5
20	14-pin IC socket		AMP	5
21	0.5" spacers			8
22	Spacer fasteners			8