A MULTICHANNEL, GENERAL-PURPOSE DATA LOGGER

MICHAEL EDWIN GARDENER

A MULTICHANNEL, GENERAL-PURPOSE DATA LOGGER

by

MICHAEL EDWIN GARDENER

Thesis submitted in partial fulfilment of the requirements for the Diploma in Technology to the Department of Electrical Engineering (light current) at the Cape Technikon.

Institute for Maritime Technology
Part of the Armscor Organisation

Simon's Town
South Africa
January, 1986

ABSTRACT

This thesis describes the implementation of a general-purpose, microprocessor-based Data Logger.

The Hardware allows analog data acquisition from one to thirty two channels with 12 bit resolution and at a data throughput of up to 2KHz.

The data is logged directly to a Buffer memory and from there, at the end of each log, it is dumped to an integral cassette data recorder. The recorded data can be transfered from the logger to a desk-top computer, via the IEEE 488 port, for further processing and display.

All log parameters are user selectable by means of menu prompted keyboard entry and a Real-Time clock (RTC) provides date and time information automatically.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Mr C Rudolph, and Mr D Meyer for their encouragement and assistance during the course of this project.

I would especially like to thank Mr S Koster for his patience and friendly advice in helping me come to grips with the microprocessor development tasks as well as acting as a sounding-board for my design ideas.

I would like to acknowledge the circuits designed by my collegues Mr S Koster and Mr G de Waal used in this project.

I would also like to thank the various members of the Institute for Maritime Technology who assisted me in the preparation of this document with special thanks to Miss T Miles for her word processing effort.

I am indepted to the Institute for Maritime Technology for the financial and material support extended to me during my Diploma in Technology Studies.

TABLE OF CONTENTS

1. INTRODUCTION

As part of an ongoing data gathering exercise situated in Durban, a requirement arose for a self-activating, stand-alone, Data Logger.

A system specification for the logger was drawn up and a survey of commercially available equipment established that none would meet the particular needs of this application. It was therefore decided to build a data logger that would meet the specifications; the design and development of which forms the basis of this thesis.

The main objective of this thesis is to provide a comprehensive document on the Data Logger that will enable the potential user to understand its theory of operation, design limitations and operating instructions. Enough information is included to allow repairs and modifications to be carried out when required.

2. DESIGN CRITERIA

The following prominent criteria were taken into consideration during the design stage of the Data Logger.

1. The design should be based on the Intel 8085 microprocessor family as it is well supported at the Institute for Maritime Technology (IMT) with regard to development system, high-level language (PL/M) and in circuit emulator.

2. The logger should be designed on a modular basis to allow for future expansion or modification and to be easily adaptable to other IMT requirements.

3. The Data Logger should be of a general-purpose, as opposed to dedicated, design to ensure flexibility in operating role.

4. In keeping with the modular concept it was decided that the system should be implemented using the S.A.Bus standard.

   The S.A.Bus is a South African microprocessor bus standard that was first formulated in 1978 and has the following objectives.

   1. Uniform card size and edge connectors.
   2. Interchangeability of system modules between various organizations.
   3. Standard bus line assignment.
   4. Access to relevant information concerning system modules.

   It also has the following advantages.

   1. The S.A. bus favours the 8085 microprocessor family.
   2. Most boards are fully buffered and only require a single +5V supply.
   3. There are at least two commercial concerns that support the S.A. bus with a selection of off-the-shelf boards.

5. The design should incorporate the M.F.E. Digital Cassette Tape Transport which was available at IMT.

6. A tape read-back unit would be required to recover the logged date from the digital cassettes. It was decided that this function should be incorporated into the Data Logger. Because the required Data Logger would be remotely sited and in continuous use, a second identical instrument should be constructed which would fulfil several useful roles:

    1. A back up system for the remotely sited unit.
    2. A unit on which to develop or modify the system hardware/software.
    3. A useful instrument for general-purpose data logging at IMT.
    4. A tape read-back unit.

7. Because at least two units were required it was decided to make use of printed circuit boards (p.c.b's) for all in-house electronics.

8. The system was required to be easy to operate and therefore the design should incorporate a menu-driven operating system with an interactive intelligent display to facilitate entry of system log parameters.

9. A real-time read out of selected data was also required. This was to be implemented in two ways:

    1. A read out of a single logged channel to be displayed on an integral display should be provided.
    2. An interface should be provided to allow data to be displayed on a HP-85 computer in real-time.

10. Battery back up of system parameters as well as clock/calander data should be incorporated so that in the event of a mains power interruption the system could perform a warm-start.

3. SYSTEM DESCRIPTION

The function of the Data Logger is to sample analog signals at a pre-determined scan rate; convert the analog signals to digital data and store this data in a buffer memory. This process is initiated by certain trigger conditions being met and when the required number of scans has been completed the data is dumped from the buffer onto digital cassette tape.

The logger also has a data retrieval or play back mode to allow data stored on tape to be transfered, via the IEEE 488 port, to a computer for processing.

The Data Logger comprises of nine functional blocks (Fig 1):

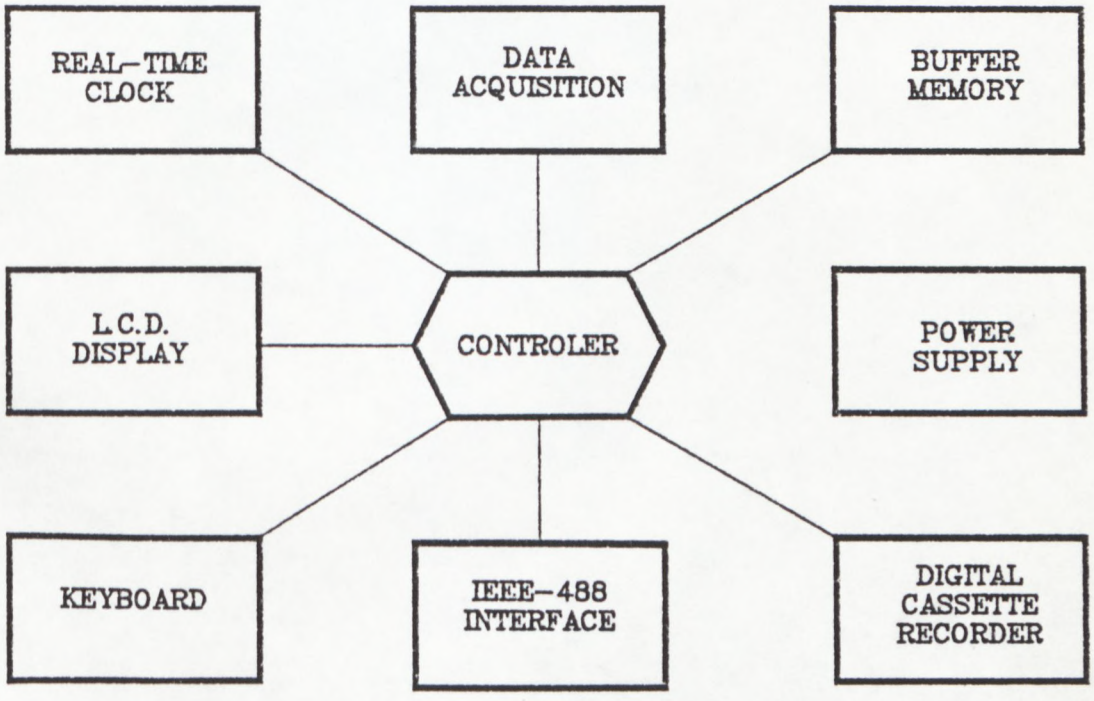


FIG. 1

DATA LOGGER FUNCTIONAL BLOCK DIAGRAM

3.1. Keyboard:

The Keyboard is part of the user/instrument interface and consists of 16 keys which can be divided into three groups.

1. Numeric keys:

There are ten keys which allow the numbers 0 - 9 to be entered as variables.

2. Control keys:

There are three control keys:

1. [←] and [→] which allow the cursor position on the display to be shifted left or right or parameters to be decremented or incremented i.e. channel number.
2. [E] allows the selected parameters to be entered into the program.

3. Soft-keys

There are three soft-keys that have functions which are uniquely defined by the current program and change under program control. The actual function that a particular soft-key will perform at any time is labelled on the lower line of the display.

3.2. Display

The intelligent Liquid Crystal Display (L.C.D.) provides the other half of the user/instrument interface. The device can display two lines of sixteen characters each. Data is usually displayed on the top line with the soft key labels on the lower line.

3.3. Real-Time Clock

The real-time clock (RTC) provides the date and time data for the log. It also provides an alarm clock type interrupt for triggering a log and a periodic interrupt which governs the logger scan rate.

The scan rate is user selectable via the keyboard by entering a number between 1 and 15. This represents scan rates from 30,517 μ sec to 500 m sec with the scan rate doubling as the number is decremented by one. (15 = 500m sec, 14 = 250m sec etc.)

The maximum scan rate is governed by the data acquisition hardware and software, and in practice is limited to scan rate 10 (15,625m sec). This means that each of 32 channels can be scanned once every 15,625m sec and the data for one selected channel displayed on the LCD. The clock chip also provides 50 bytes of battery backed up RAM which is used to store the log parameters. This enables the logger to recover from a mains power interruption with all the log parameters intact and with the correct date and time.

3.4. Data Acquisition

The Data Acquisition board is capable of servicing 32 analog signals on a multiplexed basis.

When a log is initiated each channel, starting with channel #1, is connected in turn via a Sample and Hold amplifier (S & H amp.) to the Anolog to Digital Converter (A/D). The A/D converter takes 35 μ secs to convert the analog signal to a 12 bit digital word. These 12 bits are stored in the Buffer memory as two consecutive 8 bit bytes. (First 8 msb's, then 4 lsb's with 4 trailing zeros.)

The Multiplexer (MUX) is then incremented, by one, to the next channel and the process repeated until all the selected channels have been serviced. The program then waits for a scan interrupt from the RTC to begin the next scan. The program keeps scanning until the required number of scans to complete the log is achieved. The maximum number of scans is governed by the buffer memory size, which is 16K in this instance, and can be determined by

$$\text{Number of Scans}_{(max)} \leq 16384 \div (2 \times \text{Number of channels}).$$

3.5. Buffer Memory

The buffer memory consists of 16k x 8 bit CMOS static RAM. It is used as a means of buffering the data before transferring the log onto digital cassette. This is necessary to increase the storage capacity of the tape as well as making the logger independent of the Digital Cassette transports data transfer rate.

3.6. IEEE-488 Interface

The IEEE-488 interface provides a convenient means of linking the logger to a computer or other instrumentation. Specifically, it can be used to allow the logged data for one pre-selected channel to be displayed on a HP85 computer in real-time. It is also used to transport the data from the digital cassette to the HP85 or HP9816 for processing. (see Appendix E)

For Data Format see Appendix F

3.7. Digital Cassette Recorder

The Digital Cassette recorder is used for the mass storage of logged data. It has three modes of operation selectable from the keyboard:

1. Record Mode

This is the default mode and is selected whenever it is required to save data on the cassette tape. In this mode the logged data stored in the Buffer memory is dumped to tape at the end of each log. This data is preceded by header data which contains the following information. Date, Time, Scan Rate, Number of channels, Number of Scans. The header and data combined, form one record.
(see Appendix F)

2. Play Mode

This mode allows data to be retrieved from the data cassette. When this mode is initially selected the cassette is rewound and then positioned at the beginning of the first record (record #1). The program then allows the operator to select any record either by using the [←], [→] keys to increment or decrement the record number or via the [ENTER] soft-key. When the desired record number is displayed the [READ] soft-key allows the program to find the record and read it into the Buffer memory. If the IEEE ON mode has been selected then data will be read out of the Buffer to the IEEE-488 bus under IEEE listener control.

3. Off Mode

This mode ensures that no data dump takes place at the end of a log and the data is therefore lost.

3.8. Power Supply

This unit converts 220V AC mains to +5V @ 3A, ±15V@ 85mA and -5V@ 500mA. Because of the battery back-up feature a power-cut of several hours can be tolerated although the Data Logger will not be operational during this time.

3.9. Controller

The controller manages all the above functions using the software to determine the course of events.

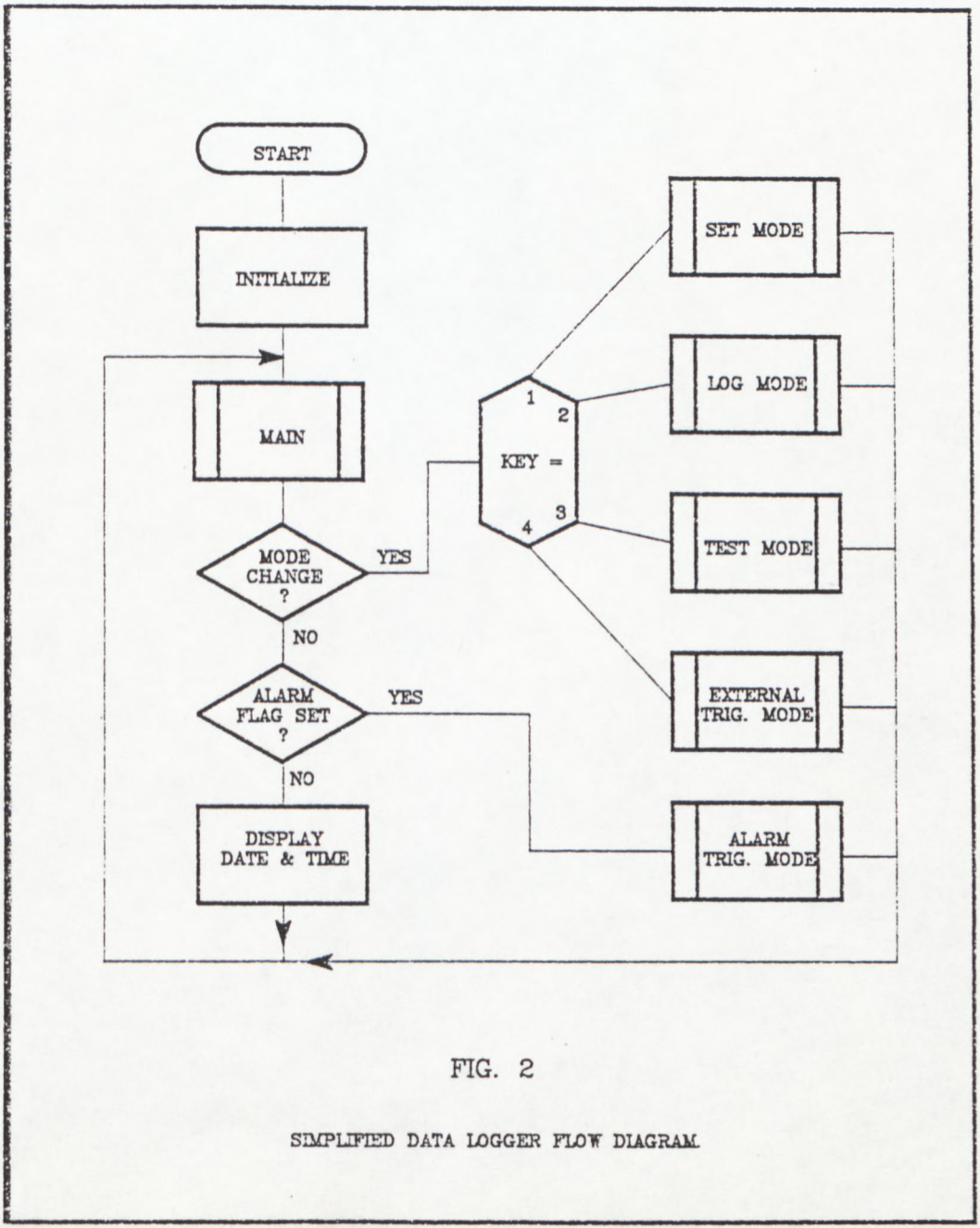


FIG. 2

SIMPLIFIED DATA LOGGER FLOW DIAGRAM

Fig 2 shows a simplified flow diagram of the Data Logger program.

The program first initializes all the hardware modules. It then enters the loop MAIN which checks to see whether a soft-key depression or external trigger contact closure has occurred. If so the program will change mode to service the particular need before returning to the beginning of MAIN. These modes are:

1. Set Mode

   This mode allows either Date and Time, or Log Parameters to be set. The log parameters include:

   1. IEEE mode selection {ON or OFF}
   2. TAPE mode selection {Record, Play or OFF}
   3. Trigger mode selection {Alarm, Manual or External}
   4. Scan Rate
   5. Number of Channels
   6. Number of Scans

2. Log Mode

   The response to this selection depends on the Trigger mode selected above:

   1. Alarm Mode

      In this mode the selected log START TIME is displayed when the [LOG-A] soft-key is depressed.

   2. Manual Mode

      The [LOG-M] soft-key starts a log immediately.

   3. External Mode

      In this mode, if [LOG-X] is depressed, the user is asked if he wishes to override the external mode and if the response is yes, a log is started.

3. Test Mode

   This mode allows either the IEEE Interface or the Data Acquisition board to be tested.

1. IEEE-Test

If selected all the values from 0 - 255 are output consecutively to the IEEE bus. An intelligent listener on the bus could use this data to ascertain whether the IEEE interface is functioning correctly or not.
(see Appendix E)

2. Data acquisition Test

This function allows the Data Logger to be used as a digital volt meter. It enables input signals to be monitored or set up as each channel is individually selectable via the keyboard and displayed on the LCD.

4. External Trigger Mode

If either key [E] or the external trigger contacts are made a log is started. The contacts are connected in parallel with key [E] and must be closed for least 100m sec to be accepted as valid.

If the Alarm Trigger mode has been selected then the alarm flag is checked and if it is set then a log is started. Lastly the current date and time are output to the LCD display before the program loops back to the beginning of MAIN. This loop is executed ad infinitum.

4. SYSTEM SPECIFICATIONS

* Number of input channels: 1 to 32.

* Scan Rate: 2 scans / sec minimum
  64 scans / sec maximum (see text)

* Number of Scans: 1 to 9999

* Log Activation: External contact (X)
  Manual (M)
  Alarm (A)

Analog input: ±5V bipolar (10Vp to p)

Resolution: 12 bit A/D (2,44 mV per bit)

Data Buffer: 16k (expandable to 40k)

Data Storage: Integral M.F.E. Digital Cassette Recorder
(676k bytes both sides)

Data Replay: Transfer from Digital recorder to IEEE-488 interface port.

Date and Time: From battery backed up Real-Time Clock.

Operator Interface: 16 Key matrix keyboard.
2 line by 16 character intelligent alphanumeric liquid crystal display.

Power Supply: 220V A.C.

* These parameters are set via the keyboard.

5. SYSTEM SOFTWARE

At the outset of the project a choice had to be made between Assembler and PL/M as the medium for software development. The high-level language PL/M was chosen because it encourages and in fact enforces good software engineering practices like structured programming, modular program design and top-down design implementation. It also improves programmer productivity and is more reliable, maintainable and portable than Assembler. It can also serve as documentation because it is easy and logical to follow.

Assembler, on the other hand, has few of these advantages but could make more efficient use of the hardware capabilities and perhaps reduce code size. It requires the programmer to have above average programming skills and experience.

The Data Logger system software comprises of 9 PL/M modules:

1 KEYBOARD MODULE
2 DISPLAY MODULE
3 REAL TIME CLOCK MODULE
4 DACQ MODULE
5 IEEE 488 MODULE
6 TAPE MODULE
7 BIN ASCII ASCII BIN MODULE
8 SET LOG PARAMETER MODULE
9 DATA LOGGER MODULE (The main module)

Each module is a collection of procedures with closely related program functions. i.e. DISPLAY MODULE holds all the procedures pertaining to the control of the display hardware.

The first 6 modules hold utility programs which are specific to the hardware they service and they will therefore be discussed along with the hardware under the heading Theory of Operation.

The last 3 modules will be discussed below.

5.1. BIN$ASCII$ASCII$BIN MODULE:

This module consists of two procedures used to convert an ASCII string to a binary word and visa versa.

BINTOASCII: Public

Converts a 16 bit binary word to four ASCII numeric characters and returns to the calling program with them in the string variable CHAR.

USE: CALL BIN$ TO $ASCII (READING, SIZE).

Where READING is a 16 bit binary word and SIZE is 0 TO 4.

If SIZE=2 then the string CHAR is left justified as follows: 0023 becomes 2323.

ASCII$ TO$BIN: Public

In this procedure the number of ASCII numeric characters indicated by SIZE are converted to a 16 bit binary word and stored in the variable BIN$DATA. This value can be accessed by the calling program.

Use CALL ASCIITOBIN (ASCII$ADDRESS, SIZE)

where ASCII$ADDRESS holds the starting address of an ASCII string and SIZE is the number of characters to be converted. (limits 1 - 4)

5.2. SET $ LOG $ PARAMETERS $ MODULE:

This module allows the log parameters to be entered into the program via menu prompted interactive keyboard entry.

LOG$PARAM$2:

Outputs the first two characters of the string CHAR to the LCD and positions the cursor beneath the most significant digit (m.s.d.)

LOG$PARAM $4:

Same as LOG$PARAM$2 but for 4 characters.

ALARM$DISP:

Sets up the LCD to display the Alarm time.

X$ OVERRIDE:

This procedure sets up the LCD to display a question and a menu for the external trigger override function.

LOG$TRIG:

Is entered whenever the [LOG-?] soft key is depressed. It responds differently depending on which choice of trigger mode has previously been selected. i.e. depends on the value of the variable TRIG$TYPE.

If the Alarm mode was selected the actual alarm time is displayed on the LCD until either the alarm interrupt occurs or the [EXIT] soft-key is depressed.

If the External trigger mode was selected then the procedure X$OVERRIDE is called. This displays "X-TRIG OVERRIDE?" on line 1 and "NO YES EXIT" on line 2 of the LCD. Depending on the operators response, either a log will be initiated or the program will return to the calling program.

If the Manual trigger was selected then a log is initiated.

CHAN$DISP$SELECT:

Allows the channel to be displayed or output to the IEEE port to be selected.

SETLOGPARAMETERS:

Supplies the prompts and menus that allow the log parameters to be entered into the program. In order, they are:

1 Number of channels.
2 Scan Rate.
3 Number of Scans.
4 Channel to be displayed.
5 Trigger mode.

5.3. DATA$LOGGER$MODULE:

This is the main module and consists of the following procedures:

TEST:

Outputs the message "TEST MODE" and the menu D-ACQ IEEE EXIT". It then waits for one of the soft-keys to be depressed before calling the required procedure or exiting.

For "D-ACQ" response see DACQ$TEST in DACQ$MODULE.
For "IEEE" response see IEEE488TEST in IEEE488MODULE.

SET$LOGGER$MODE:

Allows the IEEE MODE to be set ON or OFF and the TAPE MODE to RECORD, PLAY or OFF.

It also calls the procedure SETLOGPARAMETERS before returning to the calling program.

SET$MODE:

Allows the user to set either the RTC time and date or the log parameters and calls the procedures SET$DATE$TIME or SETLOGMODE respectively before returning to the calling program.

MAIN:

This is the main procedure and does the following:

Firstly it checks to see if TAPE$MODE$FLAG is set to PLAY (1). If it is then a call to the procedure TAPE$PLAY is made which allows the logger to act as a data cassette replay unit. (see TAPE$MODULE further). If not, then a check is made to see if any one of the three soft-keys has been depressed or if the external contact, which is connected in parallel with the [E] key, has been made (KEY = 0 to 3). The following responses are made to each key.

[SET] - The set mode is entered by a call to the SET$MODE procedure.

[LOG-M] - The log mode is entered via a call to the LOG$TRIG procedure.

[TEST] - The test mode is entered by a call to the TEST procedure.

[E] or External contact - The logger will enter the log mode via the LOG$SCAN procedure

Note: All the above will return to the beginning of MAIN on completion. (see Figure 2)

Next, the main logger menu will be output to the display. "SET LOG-M TEST". In the LOG-M message the M indicates that the manual trigger option has been selected and will change to A for Alarm and X for external.

If the Alarm trigger mode has been selected a check is made to see if an alarm interrupt has occurred. (RST55FLAG set). If it has, then a call to LOG$SCAN is made. If the TAPE$MODE$FLAG is not off (2) then a check is made to see if the TAPE$ERROR$FLAG has been set. If it has, then a call is made to the TAPE$ ERROR procedure. The Real-Time clock is then interrogated by a call to RTC$DATA. This returns the current date and time in the two strings RTC$DATE and RTC$TIME. The day of month is output to line 1 of the display and then the month word. This is after the month number has been converted into a 3 character month word via the look-up-table, DATE$MES. The time is then output to the display.

Lastly the 8085 interrupt register is read to determine whether either RST7.5 (keyboard interrupt) or RST 5.5 (RTC interrupt) has occurred. If the former then the keyboard register is read into the variable KEY and if the latter then RST55FLAG is set to 0FFH. The Program then loops back to the beginning of MAIN.

6. THEORY OF OPERATION

Because the system was designed on a modular basis the theory of operation of each module making up the system will be discussed under the following headings:

1 IMT 8085 C.P.U. board.
2 IMT DATA ACQUISITION board.
3 IMT MFE TAPE board.
4 IMT IEEE-488 INTERFACE board.
5 Microcom 32K Memory board.
6 Data Logger P.S.U.

The discussion will encompas both the hardware and, were applicable, the software modules pertaining to each board.

It should be noted that, with the exception of the 32k memory board, all the above hardware was designed and developed inhouse specifically so that the Data Logger could be implemented.

6.1. IMT 8085 C.P.U. board

6.1.1. General

The IMT 8085 C.P.U. board is in fact a S.A. bus compatable single-board computer.

The design was conceived when it was realize that although IMT had a comprehensive 8085 microprocessor development system there was no easy and cheap solution for the implementation of small microprocessor controllers or systems. It was designed to be S.A. bus compatable so that the use of off-the-shelf boards would allow for system expansion if necessary, i.e. memory, input/output, data acquisition etc.

This single-board computer will enable many microprocessor applications to be tackled that would previously not have been viable due to the design tasks and their related cost and time requirements. It will also enable the designer to concentrate on the application of the computer rather than the hardware implementation details.

6.1.2. Features:

The IMT 8085 C.P.U. board has the following features:

- On board, battery backed up, CMOS Real-Time Clock.
- 50 bytes of battery backed up CMOS static RAM.
- Single +5V supply @ 300mA
- 2k or 4k (link selectable) JEDEC EPROM socket (2716 or 2732 type)
- 2k CMOS static RAM in JEDEC socket (6116 type)
- Two 8 bit I/O ports, one 6 bit port and a 14 bit counter/timer as well as 256 bytes of static RAM. (8155)
- 14 Pin header for strobed input keyboard (up to 8 data lines)
- 14 Pin header for intelligent Liquid Crystal Alphanuerics display (Densitron LMA2C16DC type)
- Power ON and Manual Reset.
- Board designed to be S.A. Bus compatable.
- All bus signals are buffered and will load the external S.A. Bus by a maximum of one LS TTL load.

6.1.3. Hardware:

The two prime design goals were:

- To be S.A. bus compatable.
- To be a complete stand-alone single board computer.

The design meets both these requirements.

With Reference to the 8085 CPU Board circuit diagram (Appendix G) the circuitry will be discussed in detail.

The 8085 (IC3) is the heart of the circuitry and is run at an input clock frequency of 4,19430 MHz. This signal is derived from either the MC 146818 Real-time clock chip (IC 15) or by directly applying the crystal to the X_1, X_2 inputs of the 8085. The board will function at higher frequencies (say 6MHz) but the Intelligent display could not be used without modification as it requires a minimum 450 n sec pulse on 'E' for correct operation. (see data sheet for wait state modification). The following S.A. Bus output control lines are implemented by buffering and/or inverting the standard

8085 control signals: CLK, $\overline{\text{RESET}}$, $\overline{\text{HOLDA}}$, $\overline{\text{INTA}}$ and $\overline{\text{SYNC}}$.

$\overline{M1}$ is achieved by buffering ($\overline{S1.SO}$) and is active for an opcode fetch or $\overline{INTA}$.

$\overline{OW}$, $\overline{IR}$, $\overline{MW}$ and $\overline{MR}$ are derived from the 8085 signals $\overline{WR}$, $\overline{RD}$ and IO/$\overline{M}$ being decoded by a 74LS42 (IC 9). The RD and WR lines have 4k7 pull-up resistors on them to prevent spurious signals from being generated when the microprocessor is in the hold state.

The hold state is initiated by the S.A. bus $\overline{HOLD}$ line being taken low and indicates to the 8085 that another device is requesting the use of the address and data busses. e.g. for DMA. When the 8085 is ready to relinquish control it outputs the $\overline{HOLDA}$ signal. This signal together with RESET OUT and CLK (OUT) from the 8085 is used by a 74LS74 (IC 4) to implement a control signal for putting IC's 5, 6, 7 and 8 into the high impedance state. These IC's are buffers for the data, address and some control lines and are re-activated when $\overline{HOLDA}$ from the Bus is taken high.

IC 8 demultiplexes the low order address lines from the 8085 multiplexed address/data bus and buffers them onto the S.A. Bus.

IC 6 is a bidirectional three state buffer and is used to control data flow to and from the S.A. Bus. Pin 1 of IC 6 is the direction control pin and is normally held high allowing data to flow from the board to the S.A. Bus. The circuitry in the dashed block (IC 17 and IC 13) controls this line and will only allow data to be read from the S.A. Bus for memory addresses 4000H and above and I/O addresses 40H and above. i.e. external memory must be mapped above 4000H and external I/O above 40H. This circuitry was included because the first 16k of the microprocessor address space is absolutly decoded into 2k blocks by IC 12. The first two blocks are anded together by ICs 13 and 17 to form a contiguous 4k block for the 2732 type EPROM. Although use is not made of the full 16k it simplifies the design and still leaves room for 48k of external memory and 192 I/O locations. (See System Memory and I/O Maps. FIG 3 and FIG 4)

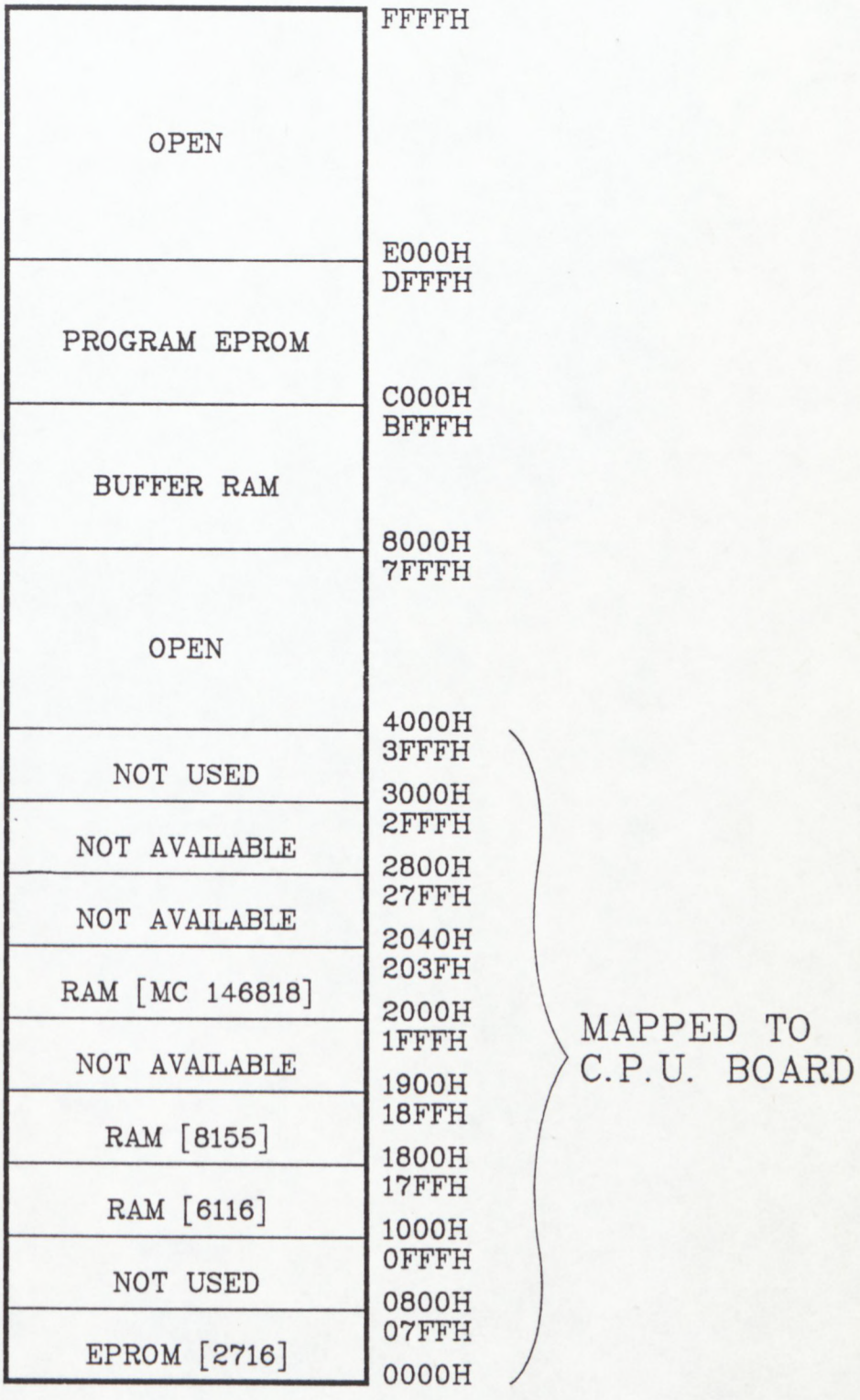


FIG 3

SYSTEM MEMORY MAP

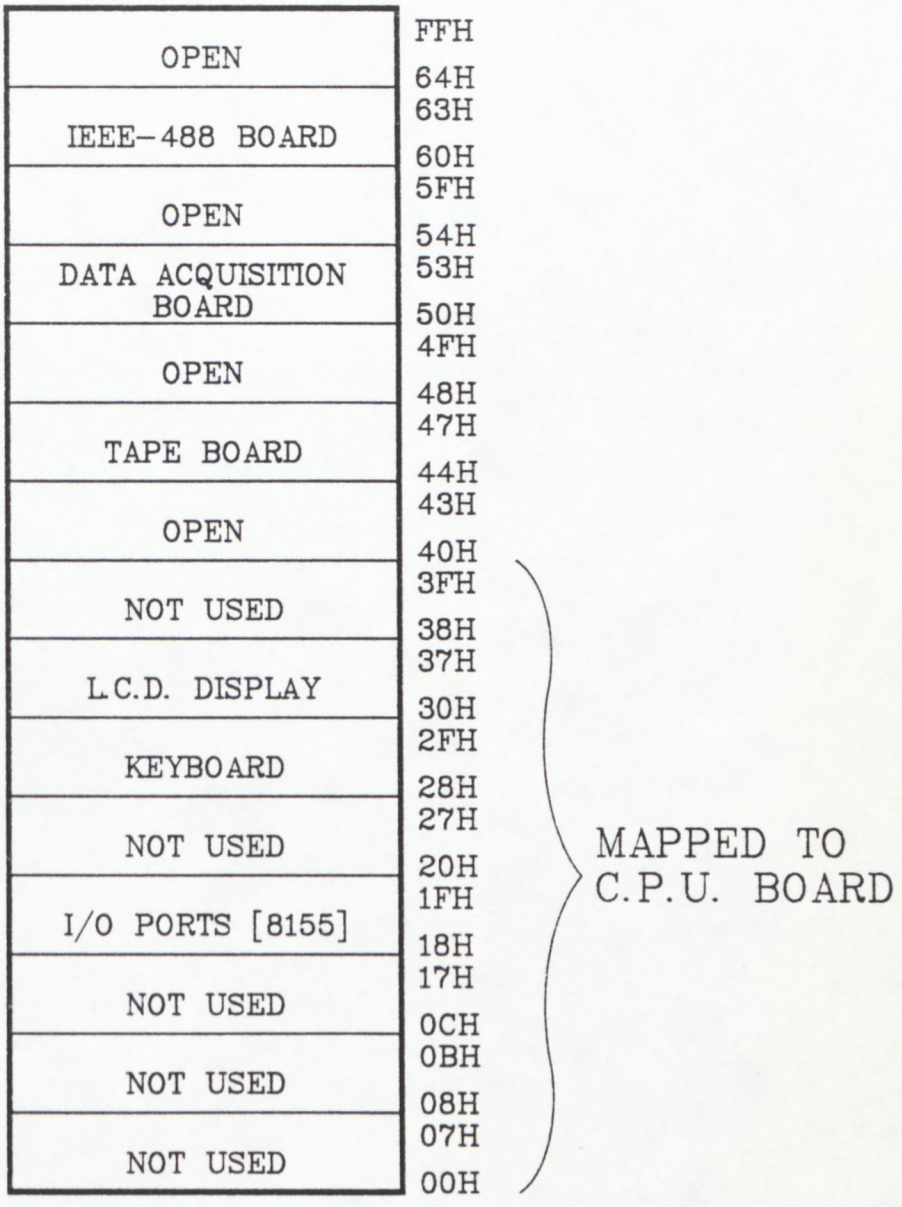


FIG 4

SYSTEM I/O MAP

6.1.4. Memory:

Two 28 pin JEDEC compatible sockets are provided, one each for RAM and EPROM. The address decoding allows for 2k or 4k of on board EPROM of the 2716 or 2732 type (link selectable) and 2k of CMOS static RAM of the 6116 or 5516 type. By cutting the track to pin 23 of IC 11 (RAM) and connecting pin 23 to +5V the RAM socket can be used for an extra 2k of EPROM (2716) if required.

6.1.5. Parallel I/O:

The board makes use of an Intel 8155 RAM/IO/TIMER which has 22 programmable parallel I/O lines i.e. two 8 bit and one 6 bit port. These lines are defined by a software write to the 8155 command/status register. A further two lines are available for the programmable timer function.

I/O ADDRESS	8155 SELECTION	
18H	Internal command / Status Register	
19H	PORT A	8 bit
1AH	PORT B	8 bit
1BH	PORT C	6 bit
1CH	Low order timer count	8 bit
1DH	High order timer count	6 bit +2 mode bits

8155 I/O addressing scheme

The timer is basically a 14-bit down counter but has four different modes of operation allowing the following to be generated under software control.

- A single square wave or programmable pulse.
- A continuous square wave.
- A single pulse on terminal count (rate generator)
- A continuous pulse train or programmable strobe.

There are also 256 byts of static RAM available starting at address 1800H.

6.1.6. Display:

6.1.6.1. Hardware

An intelligent 2 line by 16 character dot matrix, alphanumeric, L.C.D. display by the Densitron Corporation is used to implement the display function. It interfaces directly with the 8085 bus lines, with a minimum of hardware being required for the strobe enable 'E' pin. This signal is derived by the circuitry of IC 14 which combines $\overline{OW}$, $\overline{OR}$ and $\overline{CE}$ to produce a positive going pulse conforming to the enable pulse requirements. Note: This pulse length may not be less than 450n sec for correct operation therefore the 8085 input clock frequency should not be greater than 5MHz. i.e. system clock < 2,5MHz.

The display module has an on board CMOS microprocessor which takes care of all the display update, refresh and control functions.

The top line (line 1) of the display is generally used to display the relevant data and the bottom line (line 2) is used for soft-key function display.

The voltage divider R6, VR1 is used to adjust the viewing angle of the display.

The display connects to the board via a 14 pin ribbon-cable header.

6.1.6.2. Software:

Associated with the display hardware is a PL/M software monitor module, DISPLAY$MODULE. This module has five Public procedures and two Public variables which can be accessed by any external program to facilitate using the display.

LCD$INIT: Public

Initializes the display as follows:

- 8 bit word length.
- Two line display.
- Cursor ON.
- Increment cursor position on entry.
- Clear display.

C$OUT: Public

Outputs one character to the LCD at the current cursor position.

Use: CALL C$OUT ('X')
where X is the ASCII character to be output.
or CALL C$OUT (0F3H)
where the hexidecimal value of the character to be output is used. (see data sheet in Appendix D for font table).

MENU: Public

Outputs three soft-key labels of 5 characters each to line 2 of the display.

Use: CALL MENU (.X, .Y, .Z)
where .X, .Y and .Z are the starting address locations of 3 separate 5 character strings X, Y and Z.

Note: Character 5 of Z is not displayed because it is off screen.

TEXT$OUT: Public

Outputs a string of characters starting at the current cursor position.

Use: CALL TEXT$OUT (.M, S)
where .M is the address of the first character in the message M and S is the size or number of characters to be displayed.
OR CALL TEXT$OUT (.M, LEN (M))
This will display the entire string M.

INST$OUT: Public

Outputs one instruction to the display instruction register.

Use: CALL INST$OUT (I)
where I is an instruction code from the instruction set below.

Instruction set:

Clear display	=	01H
Home	=	02H
Blink Cursor	=	0FH
Cursor ON	=	0EH
Cursor OFF	=	0CH
HOME 2	=	C0H
Shift cursor left	=	10H
Shift cursor right	=	14H
Shift display left	=	18H
Shift display right	=	1CH
DISPLAY OFF	=	08H

(see data sheet further)

The two Public variables are:

. CHAR: This is a 16 character temporary storage location:

. POSITION: This variable holds the current cursor position at all times.

LCD$BUSY:

Is used within the module to determine when the display microprocessor is busy. i.e. processing an instruction. The procedure is called whenever a character or instruction is sent to the display module. The procedure returns to the calling program only once the busy flag is cleared. Before it returns it enters the new cursor position into POSITION.

Lastly, the display data and instruction registers are mapped to I/O locations 31H and 30H respectively.

6.1.7. Keyboard:

6.1.7.1. General

The keyboard is a 16 key x - y matrix type.
The keys have the following functions:

. Ten have the standard numeric functions 0 through 9.

. Three have the dedicated functions:

1. Left shift or decrement [←].

This key allows the cursor to be shifted one position to the left or a number to be decremented by one for each key depression. Which of the two functions it performs depends on the current program.

2. Right shift or increment [→].

This key performs the same as above but in the opposite direction.

3. Enter [E]

This key allows a variable to be entered into the program once it has been correctly keyed in.

. The last three keys are soft-keys.

These keys change roles from time to time under software control. Their designated function at any one time is displayed in the lower line of the L.C.D. as a label of maximum 5 character length. The keys are colour coded for ease of association with the labels which have matching colours on the L.C.D. bezel.

This function allows a great deal of flexibility in the system/operator interface, especially with regard to the input and selection of parameters, and enhances the menu programming facility.

6.1.7.2. Hardware:

The keyboard has associated with it a circuit called Matrix Keyboard Encoder which was designed by Mr S Koster. This circuitry resides on a P.C. board which is mounted on the back of the keyboard and is connected directly to the C.P.U. bus lines via a ribbon cable. There are four data lines which convey the 16 possible key values to the C.P.U. (Allowance has been made on the CPU board for 8 bit data.) Of the remaining four lines two are used for +5V and ground, one for keyboard output enable and the last for the Data Available signal which is connected to RST 7.5 on the 8085.

The circuitry uses a 74C922 16 key encoder chip which has the following built in functions:

. On chip scanning.

. 2 Key roll over.

. Keybounce elimination.

. Tristate outputs.

. Last key register.

When a valid keyboard entry has been made a Data Available signal is generated. This signal stays high until the key is released and is used to generate a positive going pulse on RST 7.5 of the 8085. It also enables an oscillator which drives circuitry to cause a Beep sound for user feedback.

6.1.7.3. Software:

The software drivers for the keyboard circuitry are found in KEYBOARD$MODULE. This module has four Public procedures to facilitate its use.

KEY$INIT: Public

Sets up the RST 7.5 jump vector and the interrupt mask. It also enables the RST 7.5 interrupt.

Closely associated with this procedure is a procedure called RST$75.

RST$75: is an interrupt routine and is invoked when the Data Available strobe (RST 7.5 interrupt) is received from the keyboard encoder circuitry by the 8085 . The procedure returns with the depressed key value in KEY (0 - F Hex) and an ASCII value in KEY$ASCII which is generated by a look-up table. An interrupt flag (FLAG1) is set and the program control is returned to where it left off.

KEY$ENTRY: Public

Waits for a key depression to occur which is indicated by FLAG1 being set. Once this happens FLAG1 is reset and the procedure returns to the calling program with the keys value in KEY.

KEY$IN: Public

Allows a string of up to 16 characters (usually numeric) to be assembled in the external variable CHAR which are usually displayed on the LCD for feed back. It also allows the cursor to be shifted right or left for editing purposes.

This procedure will only return once [E] is received. At this point a correctly entered string of numerals will be available in CHAR and a count variable (I) will hold the number of characters entered.

6.1.8. Real-Time Clock:

6.1.8.1. General:

The time-of-day clock with calander function was implemented using a single CMOS chip (MC 146818). It has the following main features:

. Battery back-up for the clock functions as well as for 50 bytes of static RAM. This RAM is used for all variables and input parameters that need to be protected from power-cuts.

. A Programmable periodic interrupt mode.

. An alarm interrupt mode.

See data sheets in Appendix D further.

The clock is set via menu-prompted keyboard entry.

6.1.8.2. Hardware:

The clock chip interfaces directly with the CPU bus lines. The clock out (pin 21) which is derived from the 4 MHz crystal is used as the microprocessor clock input. An on-board Ni-CAD battery provides the 3, 6 V required for battery back-up. To implement a glitch free change over from +5V to battery it was found necessary to include a comparator circuit which isolates the $\overline{CE}$ line on the RTC chip as power fails. This circuit is set to operate when the +5V rail falls below 4,8V. When the +5V rail falls below 3,6V the battery takes over via a diode switch. This circuitry is hard wired into the system. (IC's 18 and 19)

6.1.8.3. Software

REAL$TIME$CLOCK$MODULE contains the software drivers for the above hardware. They are:

RTC$INIT: Public

Initializes the RST5.5 jump vector and sets the interrupt mask on the 8085. It also sets the initial Periodic interrupt rate at 0,5 sec and leaves it in the disabled state.

RST$55: Public

Is entered via the RST 5.5 interrupt and sets RST55FLAG before returning to where the program left off. The flag is used to indicated that an interrupt on the RST 5.5 line has occurred.

SET$DATE$TIME: Public

Puts the RTC chip in the set mode and allows the date and time or the alarm time to be set. The above is accomplished via menu-prompted keyboard entry and once completed the RTC chip is put into the run mode and the display is cleared before returning to the calling program.

The procedure RTC$INIT$DATA is used by the above to convert ASCII characters into packed BCD which in turn is used to load the relevant RTC registers with the new data.

RTC$DATA: Public

This procedure waits for the RTC internal update cycle to be completed before it reads the data registers and converts them to two 6 character ASCII strings, RTC$DATE and RTC$TIME.

Their format is:

RTC$DATE: DD MM YY
RTC$TIME: HH MM SS

This data is used to output to the LCD or used whenever current date and time data are required.

6.1.9. Specifications:

1. System Clock:- 2,09715 MHz

2. Memory:

On Board: 2k or 4k bytes EPROM; start address 0H (2716 or 2732)
2k bytes static RAM; start address 1000H (6116 type)
256 bytes static RAM; start address 1800H (8155)
50 bytes battery backed up static RAM; start address 2000H (MC 146818).
Expansion: 48k bytes. ROM or RAM starting at 4000H.

3. Input/Output:

Parallel: 22 programmable 1/0 lines.
Serial: The SID and SOD pins can be used for serial 1/0.
Expansion: 192 1/0 locations available starting at 40H

4. Interrupts

8085 NAME	PRIORITY	USE	RESTART ADDRESS
TRAP	1	S.A. BUS $\overline{\text{NMI}}$	24H
RST 7.5	2	Keyboard interrupt	3CH
RST 6.5	3	Not used	34H
RST 5.5	4	RTC interrupt$\overline{}$	2CH
INTR	5	S.A. BUS INTR	Depends on external circuitry

5. Timers: Time of day clock/calander (MC 146818)
Periodic interrupt generator on RST 5.5 (MC 146818)
Alarm clock interrupt on RST 5.5 (MC 146818)
14 bit general purpose timer (8155)

6. Electrical:

Requires 5V dc at 300mA for completely populated board.

6.2. IMT Data Acquisition Board

6.2.1. General:

This board was designed with the following basic requirements in mind.

1. S.A. Bus compatibility.
2. 12 bit resolution.
3. 32 multiplexed analog input channels.
4. Data through put of at least 64 samples/sec.

The AD 574A, Analog to Digital (A/D) converter by Analog Devices was chosen to implement this board. It is a 12 bit (or 8 bit) successive-approximation A/D converter that can easily be interfaced to the S.A. bus. It is configured as a bipolar input device with the following input voltage ranges: -5 to +5V or -10 to +10V which are link selectable.

6.2.2. Data Throughput:

For a successive approximation A/D the input signal is required to remain constant during the conversion time. This implies that it must not change by more than half a least significant bit (LSB) during this time.

The maximum input frequency (f_{max}) for a given precision (n bits) is given by:

$$f_{max} / nbit = \frac{1}{2\pi \times 2^{n} \times \text{Sampling time}}$$ Ref. 4 Page 15

Sampling time for the AD574 is:

12 bit conversion:- 35μ sec
8 bit conversion:- 24μ sec

$$\therefore \quad f_{max} / 12bit = \frac{1}{2\pi \times 2^{12} \times 35 \times 10^{-6}}$$

$$= 1{,}11 \text{ Hz}$$

similarly $f_{max} / 8bit = 25{,}9$ Hz

Because these Figures are rather low a Sample and Hold (S&H) amplifier (AD 582) is included in the design to improve them. f_{max} of the system is now governed by the S&H aperture jitter Figure which is 15n sec for the AD 582.

Now $$f_{max} / n \text{ bit} = \frac{1}{2\pi \times 2^{n} \times \text{Aperture jitter}}$$ Ref. 3 Page 14-34

$$\therefore \quad f_{max} / 12 \text{ bit} = \frac{1}{2\pi \times 12^{12} \times 15 \times 10^{-9}}$$

$$= 2{,}590 \text{ kHz}$$

and $f_{max} / 8$ bit = 41,446 KHz but in practice it is limited to the reciprocal of the sum of A/D conversion time (8 bit = 24μ sec) added to the S&H acquisition time (6μ sec)

$$\therefore f_{max} / 8 \text{ bit} = \frac{1}{(24 + 6) \times 10^{-6}} = 33{,}33 \text{ KHz}$$

If the sampling time is not referenced to an absolute time then the f_{max} is again the reciprocal of the sum of the conversion time (35µ sec for 12 bit) plus the acquisition time of the S&H amp (25µ sec for 12 bit)

$$\text{i.e. } f_{max} / 12 \text{ bit} = \frac{1}{(25 + 25) \times 10^{-6}}$$

$$= 16{,}67 \text{ KHz}$$

Throughput for the Data acquisition board, using overlapped multiplexing is therefore:

12 bit resolution: - 60µ sec.
8 bit resolution: - 30µ sec.

6.2.3. Printed Circuit layout and Power supply requirements:

The layout of a 12 bit data acquisition board is fairly critical if full use is to be made of the available resolution. Consider that if the A/D is configured for a 10V input voltage range then one bit represents 10V divided by 2^{12} or 2,44mV. This means that any noise signals (and power supply ripple) must be kept well below this value if 12 bit accuracy is to be achieved.

Some points to consider during P.C. layout are:

1 Separate digital and analog circuits by "ground" tracks.
2 Reduce edge connector contact resistance effect by using parallel connections.
3 Use thick track and run separate ground lines for each DC supply.
4 De-couple DC supplies at the input to the PCB as well as at each IC. Use a combination of low value ceramic and high value tantalum bead capacitors to achieve this. i.e. 0,1µF ceramic in parallel with 47µF tantalum.
5 Use ground guard's on sensitive input lines.
6 Leave as much ground plane on the the PCB as possible.
7 Connect all unused IC pins to ground.

8 Connect the analog and digital grounds together locally on the board i.e. at A/D converter.

9 Site A/D and S&H as close as possible physically.

10 If unavoidable, analog and digital lines should cross at 90° to minimise interference.

A design for a low noise ±15V tracking regulator supply was found in the National Semiconductor Linear Databook (Ref. 13 Page 1-45) which gave the required regulation and output ripple Figures. This design uses an opamp in a feedback configuration to improved the output ripple.

6.2.2. Hardware:

With reference to the IMT Data Acquisition board circuit diagram (Appendix G). IC1 buffers most of the incoming lines from the S.A. bus. IC 2 buffers the rest and decodes the board address (absolutely). The board occupies eight consecutive port locations (50H to 57H) and the DIL switch S1 and pull-up resistors select the actual board start address.

IC4 (c and d) decode the chip select ($\overline{CS}$) line to the A/D converter (IC5). IC3 (c) has a similar function for IC6 (8255). IC3 (a and b) control the chip enable ($\overline{G}$) of IC7 (74LS245). This ensures that the bi-directional data lines are only selected when (($\overline{IR}$ OR $\overline{OW}$) AND board select from IC2) is true. The DIR pin of IC7 is controlled by $\overline{OW}$.

The analog signals of the Data Acquisition board are multiplexed to the S&H amp by two Multiplexer (MUX) chips (IC9 and 10). The channel to be sampled is selected by IC6 (Port B) under software control. IC4(a) ensures that only one of the MUX IC's is operative at any time.

IC8 is the Sample and Hold (S&H) amplifier.

The sample/hold mode is selected automatically by the status (STS) output of the A/D. When the A/D is triggered, by a software command, the conversion cycle begins by taking the STS line high which puts the S&H amp into the hold mode. It remains in the hold mode until the A/D has completed it's conversion. STS is then taken low which allows the S&H amp to revert to the sample or track mode.

Provision has been made on the PCB to accomodate different size hold capacitors (C2) and there are also two 8 bit, software configurable, digital input/output ports available for digital data acquisition. (Port A and Port C of IC6).

Analog input connection to the board is made either by the on board terminal blocks or via the 34 way ribbon cable header (P3).

To calibrate the board VR1 and VR3 can either be set to 50Ω (midway) or can be adjusted as per the procedure "Bipolar Operation" on Page 7 of the AD574 Circuit Details (Appendix D). VR3 is used as an offset nul pot for the S&H amp. (IC8).

6.2.3. Software

The software module DACQ$MODULE has been written to facilitate the use of the Data Acquisition board. It has the following procedures:

DACQINIT: Public

Initializes the on-board 8255 (IC6) with all its ports set to the output mode.

MUX$35:

Is used to ensure that the A/D converter has a full 35 micro seconds in which to complete a 12 bit conversion before being read. It also increments the channel pointer, CHAN, and outputs this new value to Port B of IC6 which selects the new MUX channel. The MUX increment is done during this time to speed up system throughout by "overlapping" some of the system settling times and is allowed because the S/H amp is always in the "Hold" mode before this procedure is called.

The procedure returns to the calling program after 35µ sec with the MUX switch incremented by one.

LOG$DISPLAY$PROCEDURE:

This procedure configures the display to output "CHAN.#" on line 1 and a soft-key menu on line 2 which depends on whether the TEST flag is set or not

TEST = 0	".....	.....	EXIT"
TEST = 1	".....	ENTRY	EXIT"

This flag is supplied by the calling program.

LOG$DATA$NORMALIZE:

Normalizes the 12-bit data (DATA$HI and DATA$LO) for a selected channel, to a value in the range 0 - 10000. The value is stored in the 16 bit variable AD$DATA after the LSB has been rounded up and represents milli Volts.

If the IEEE$MODE$FLAG is set then AD$DATA is output (high byte first) to the IEEE-488 port provided that the IEEE listener indicates via the $\overline{\text{NRFD}}$ line that it is ready for data ($\overline{\text{NRFD}}$=1) and that the Data ready buffer (DRB) line is low. AD$DATA is then converted first to a bipolar (0-5000mV) value and then this binary value is converted to four ASCII numerals with sign and output to the LCD with "mV" appended.

LOG$SCAN:

This is the main procedure and is called when the Data Logger is triggered.

First the buffer pointer (BUFF$POINT), which points to the next location in the Buffer memory to be filled, is initialized to zero and then the MUX switch is set to channel #1.

The periodic interrupt function from the Real-Time clock is set to the selected scan rate and enabled.

The current value of Date and Time is read from the RTC to be used in the Data Header.

A wait loop is then entered until the periodic interrupt occurs. When this happens the A/D conversion cycle is started by a dummy output to Port 50H. The time delay procedure (MUX$35) is called and when it returns, two bytes of data are read into tempory storage arrays (DATA$HI and DATA$LO) and then transfered to the Buffer memory (high byte first). BUFF$POINT is then incremented by two. Note that the MUX switch is incremented by one during the MUX$35 call.

When the required number of channels has been converted the MUX switch is reset to channel #1 and LOG$DATA$NORMALIZE is called. This procedure displays the value of the analog channel initially selected, on the LCD in millivolts.

If the required number of scans has not yet been met the program loops back to wait for the next periodic interrupt, otherwise the interrupt is disabled.

If the TAPE$MODE$FLAG is set, a tape dump of all the logged data, preceded by the Data Header (DATE, TIME, etc) is made to the Digital Tape recorder.

If the ALARM TRIGGER mode was active then the alarm is re-enabled before returning to the calling program.

DACQTEST: Public

Allows a specific, selected channel to be displayed and output to the IEEE port. The data is updated as fast as the software can run.

The selection of the channel to be displayed depends on the value in DISP. This value can be modified while the program is running by using the [←], [→] or [ENTRY] Keys. The procedure will run until either the [EXIT] key is pressed or the Data Logger is reset.

6.3. IMT M.F.E. Tape Board:

6.3.1. General:

The Mass storage media chosen for the Data Logger was Digital Cassette. This choice was made for two reasons:

1 A data logger typically has to operate in a harsh environment and therefore floppy disk storage, for instance, would not perform satisfactorily. A digital cassette (same format as the common audio cassette) is robust and will survive transportation fairly well.

2 Two M.F.E. digital cassette tape transports were available.

The MFE 450B tape transport has the following features:

- Robust design and good environmental characteristics.
- High data reliability is achieved by using an optically encoded clock to regulate the transport motor speed. (better than 1×10^{-7} error rate.)
- Transport reliability is indicated by the 15,000 Hours MTBF Figure.
- Storage capacity is 676k bytes using both sides of a 300 foot cassette at 800 BPI.

6.3.2. Hardware:

The MFE 450B type transport was fitted with the MFE 414 parallel option which made interfacing it to the S.A. Bus a fairly simple task. This circuitry is called the IMT MFE Tape Interface and will be discussed with reference to the circuit diagram (see Appendix G).

IC1 is used to buffer the board from the S A Bus. IC2 is a 6 bit binary comparator which is employed as an address decoder. The board takes up four consecutive (fully decoded) input/output locations with the start address being set up on S1 (selected to start at 44H for Data Logger).

When the I/O address and switch value match, IC2 (pin 9) is allowed to be pulled high by a 4k7, resistor. (IR1 pin 7) which causes the $\overline{CS}$ line on IC4 (pin 6) to be taken low by the inverter IC3 (c).

The data lines of IC4 (8255) are buffered from the SA Bus via a 74LS245 (IC5) bi-directional buffer. The direction of data flow is controlled by the SA Bus $\overline{OW}$ line. IC5 chip enable ($\overline{G}$) is only low (enabled) when IC2 (pin 9) is high AND $\overline{OW}$ OR $\overline{IR}$ are active (low). At all other times IC5 is in the high impedance state.

IC4 Port A and Port B are used for data input and output respectively (from the tape recorder's point of view). Port C is used for controlling the tape drive via the control lines CE1 to CE7. The Port C line along with R1, R2 and T1 was designed to drive a LED to be used as a tape transport motion or busy indicator but was not implemented. All the above lines as well as the power lines required by the tape drive were terminated in a 40 way ribbon cable header on the board (P2).

The power required by the tape transport comes from the Data Logger P.S.U. via P3 except for the -12V which is generated on board by a DC to DC converter (CB 3811).

6.3.3. Software:

The procedures used to control the tape transport are to be found in TAPE$MODULE:

TAPE$INIT: Public

Initializes the 8255 on the M.F.E. Tape Board to the required configuration. (Port A = output, Port B = input and Port C = output)

TAPE$STROBE$OUT:

Is used to strobe a control line (CE1 - CE7) from high to low and back to high.

USE: CALL TAPE$STROBE$OUT (CE4)

This would reset the EOT flag in the tape transport.

TAPE$STROBE$IN:

Strobes data into a variable T$DATA.

USE: CALL TAPE$STROBE$IN (CE6)

This would cause the USRT status register to be read into T$DATA.

DELAY:

Produces a delay of approximately 1 milli second.

USE: CALL DELAY (25)

This would cause the program to be delayed by 25m sec before proceeding.

PORTAOUT:

Outputs data to the tape transport USRT.

USE: CALL PORTAOUT (CONTROL, DATA)

This outputs DATA to the control register designated by CONTROL.

TBMT

This procedure waits for the transmit buffer of the USRT to be empty before returning to the calling program. This is achieved by repeatedly reading the USRT's status register until the TBMT bit is high.

RDA:

Reads the USRT status register repeatedly until the Receive Data Available flag is set. It then returns to the calling program.

TAPE$ERROR$MES:

Outputs a message to the L.C.D. which depends on the error condition prevailing at the time. The error condition is determined by reading the USRT status register and the message could be one of the following:

1 "END OF TAPE!"
2 "WRITE PROTECTED!"
3 "LOAD CASSETTE!"

These messages allow the user to quickly determine why the program has aborted.

REWIND:

Checks whether a cassette is loaded and if so rewinds it to the beginning. It then pauses for 360 m sec, as required by the cassette machanics, before returning to the calling program.

POSITION$TAPE:

Allows the tape to be positioned at the beginning of a particular record which is denoted by the value in RECORD.

GETOFFLEADER:

Positions the tape head just after the clear lead-in tape (if called from TAPE$WRITE$INIT) or just before record #1 (if called from TAPE$READ$INIT).

RECORD$DISPLAY

Displays the next record to be read or written to, on the LCD.

TAPE$BLOCK$DUMP: Public

Dumps a block of data, preceded by a Data header onto tape. The header contains: Date, Time, Scan Rate, Number of Channels, Number of Scans (high byte), Number of Scans (low byte). The size of the block is determined by:

Block size = (No. of Chans x No of Scans x 2) + 16 bytes.
i.e. for 27 channels and 240 scans
Block size = (27 x 240 x 2) +16
= 12,976 bytes.

∴ Tape capacity in records is approximately = 676,000 ÷ Block size.

Using the above Figures for Block size:

Tape capacity ≃ 52 records.

The procedure checks for tape errors at all times during and before the dump process and sets the TAPE$ERROR$FLAG if necessary. After all the data has been dumped onto tape an inter-record gap is generated and the variable RECORD is incremented before the procedure returns.

If the procedure is called from TAPE$WRITE$INIT then RECORD = 0 and a dummy block of 256 bytes is initially dumped onto tape. This block is never read but is required by the transport logic for error free operation. Because a USRT is used in the tape drive electronics a SYNC word (AAH) is required at the beginning and end of each record and is provided for in the procedure.

TAPE$BLOCK$READ:

This procedure positions the tape head before the selected record and then reads the header data into the Buffer memory. This data is also used to calculate the amount of data to be read into the Buffer memory. Tape errors are monitored throughout the procedure and the RECORD counter is incremented before returning to the calling program.

RECORD$SELECT:

Allows any record (limit 1 to 99) to be selected via the keyboard and its number to be displayed on the LCD prior to reading it.

The program allows for incrementing or decrementing the record number by one for each depression of the [→] or [←] keys respectively. Any record number can be entered via the [ENTRY] soft key.

TAPE$READ$INIT:

Ensures that a cassette is loaded and then rewinds the tape and positions the tape head at the beginning for record #1.

BUFFTOIEEE$DUMP:

Dumps the data header and then the log data from the Buffer memory to the IEEE-488 port. The data will not be output unless the IEEE listener indicates that it is ready to receive data ($\overline{NRFD}$ = 1).

TAPE$PLAY: Public

Checks keyboard entries to determine whether:

- A record is to be read.
- The present record number is to be incremented or decremented.
- The program is to exit or returned to the calling program.

TAPE$WRITE$INIT:

Initializes the data cassette in the transport by rewinding it, getting off the clear leader and executing a dummy dump. The procedure returns to the calling program with the tape head positioned in front of Record #1 and with the variable RECORD = 1.

TAPE$ERROR: Public

This procedure calls TAPE$ERROR$MES which outputs the relevant error message to the LCD and then waits for a response from the user. The user must respond by removing and/or inserting a cassette depending on the prompt displayed. TAPE$WRITE$INIT is then called and the TAPE$ERROR$FLAG reset before returning to the calling program.

6.4. IMT IEEE-488 Board

6.4.1. General:

This board was designed and developed by Mr G de Waal to primarily interface a discreet logic circuit to the IEEE-488 Bus. For this application the Fairchild 96LS488 LSI circuit was ideally suited as it required no other intelligent hardware to implement.

The writer was involved with the project during the development stage and suggested extending the circuitry to include S.A. Bus compatibility. He also undertook to design the extra circuitry required. As the suggestion was not in conflict with the original design goals and would make the final board more versitile, it was decided to incorporate it.

Mr de Waal left IMT before being able to document his work and this was left to the writer, hence the inclusion of this material in this document.

The Board implements the IEEE-488 bus Talk, Listen and TALK/LISTEN modes (but not the controller mode) and either interfaces to the S.A. Bus as a fully buffered I/O peripheral with minimal software overhead requirements or to discreet logic in a stand-alone configuration (Talk mode only). It does not cater for the Extended Address mode or Parallel Pole mode.

6.4.2. Hardware:

The GPIB or IEEE-488 bus is an internationally recognized bus standard for instrumentation interface which defines mechanical, electrical and functional specifications. It is organized as a byte-serial, bit-parallel communications structure with 16 active bus lines which can be divided into 3 functional groups:

- 8 Data lines ($\overline{DIO}$ to $\overline{DIO}_8$) which require negative logic signals.
- 3 handshake lines for data transfer ($\overline{NRFD}$, $\overline{NDAC}$ and $\overline{DAV}$).
- 5 control lines ($\overline{ATN}$, $\overline{REN}$, $\overline{EOI}$, $\overline{IFC}$ and $\overline{SRQ}$).

With reference to the IMT IEEE-488 board circuit diagram (Appendix G).

The handshake and control lines are implemented by IC1 (96LS488)) and conform to the IEEE-488 standard of 1980. The data lines are buffered by IC3 or IC4 (74ALS245). These are bi-directional bus driver/receivers and capable of sinking 48mA which complies with the IEEE-488 bus specifications.

The 96LS488 can be operated in 14 different modes selectable by D.I.L. switch SW2. Table 1 (page 9) of the 96LS488 data sheet (Appendix D) defines these modes. (Note that the extended address option is not available.)

The IEEE device address is set up by D.I.L. switch SW1 (1 to 5). This allows selection of any of the 31 address codes or unlisten. (see table 2 on page 10 of the 96LS488 data sheet). The remaining switches of SW1 (6 to 8) are used to connect inverted "strobe output" signals to their "ready input" signals. This allows the required handshakes to be automatically generated if the peripheral electronics cannot supply them.

The 96LS488 clock is derived from a crystal and R1, C4 and C5 or the RC network R1, C5. (See data sheet, in Appendix D, page 12 fig 6). The crystal option should be chosen if precise timing is required otherwise the RC oscillator is adaquate. Both should produce a 10MHz clock frequency which gives correct source handshake delays. A "power-on" reset function is provided by R6 and C7 which are connected to the master reset pin (MR) of the 96LS488.

There is space on the board (P2) for 3 LED's which are driven by the following 96LS488 status lines:

- $\overline{R}$/L; this diode is on when the device is in the Remote mode.
- $\overline{LAD}$; this diode is on when the 96LS488 is addressed to listen.
- $\overline{TAD}$; this diode is on when the 96LS488 is addressed to talk.

Also on P2 are two points to be connected to a "remote/local" switch which requests a return to local input (RTL = 0) when depressed.

In the stand alone configuration IC4 buffers the signal lines from IC's 5 and 6 to the DIO bus lines. IC's 5 and 6 are 8 line to 4 line switches and allow the 96LS488 to select either a data or status read from the external electronics by means of the D/S/E line. Only the IEEE TALK mode is implemented with this hardware configuration.

In the stand alone configuration only IC's 1 to 6 but excluding IC3 are used. The board is connected to the external electronics via P3.

In the S A Bus configuration only IC1 (and its peripheral components) and IC3 are required of the above mentioned circuitry. (i.e. not IC's 2, 4, 5 and 6).

IC3 allows for bi-directional data transfer and buffers the data to and from the 8255 (IC9). The direction pin of IC3 is controlled by $\overline{DRB}$ (from the 96LS488) gated with Port C2 (IC9). IC12 is used to buffer the board from the S A Bus. IC11 is a 6 bit binary comparator which is employed as an address decoder. The board takes up four consecutive (fully decoded) input/output locations with the start address being set up on SW3. (Selected to start at 60H for the IMT Data Logger).

When the I/O address and switch values match, IC11 (pin 9) is allowed to be pulled high by a 4k7 resistor (IR2 pin 7) which causes the $\overline{CS}$ line on IC9 (pin 6) to be taken low by the inverter IC13c. The data lines of IC9 (8255) are buffered from the S A Bus via a 74LS245 (IC10) bi-directional buffer. The direction of data flow is controlled by the SA Bus $\overline{OW}$ line. IC10 chip enable ($\overline{G}$) is only low (enabled) when IC11 (pin 9) is high AND $\overline{OW}$ OR $\overline{IR}$ are active (low). At all other times IC 10 is in the high impedance state.

IC9 is a Programmable Peripheral Interface (PPI) and has three 8 bit ports.
In this application IC9 Port A was configured in mode 2 (bi-directional I/O with handshakes) to simplify software design. To work correctly in this mode a hardware modification had to be made because the TXST or STST signals from the 96LS488 were not compatible with the $\overline{ACKA}$ requirements of IC9. The track was cut between IC8 pin 10 and IC9 pin 11 and the IEEE bus line, DAV (IC1 pin 28), was connected to IC9 pin 11.

A further modification was made by connecting, the IEEE bus line, $\overline{NRFD}$ (IC1 pin 26) to IC9 pin 23 (PB5). This line is used to test whether the IEEE bus is clear for data exchange (by the S A Bus).

Port B is used to input status data from the 96LS488 to the S A Bus and the remainder of the Port C lines (PC0 and PC2) are used as output "bit" ports.

Most of the 8255 port lines connect directly to the 96LS488 but the handshake lines require inverting for proper operation. This is accomplished by IC's 7 and 8. The

signal from PC7 ($\overline{OBF}$) is gated with $\overline{D}$/S/E from the 96LS488 by 1C7 (a, b and c) to produce either a TXRDY or STRDY signal for 1C1.

Port C2 is used to control the chip enable pin ($\overline{G}$) of 1C3. This signal is gated with $\overline{DRB}$ from 1C1 by 1C8 (a and d) to control the direction pin (DIR) of 1C3.

6.4.3. Software:

The software drivers for this board are to be found in IEEE488MODULE:

IEEE488INT: Public

Initializes Port A of IC9 to mode 2 and enables IC3 via Port C2.

IEEE488TEST: Public

Outputs 0 to 255 as data to the IEEE-488 bus. It is used by an external computer (i.e. HP85) to determine if the interconnection between itself and the DATA LOGGER has been correctly made and is functional.

After the board has been initialized by IEEE488INIT it is treated as a bi-directional I/O port situated at (the selected start address) 60H. All hand shaking between 96LS488 (1C1) and the 8255 (1C9) is automatic and the 96LS488 or $\overline{NRFD}$ status can be read from Port B (location 61H.)

Status Data: Port B

PB0 = $\overline{LAD}$ from 96LS488
PB1 = $\overline{CLR}$ from 96LS488
PB2 = $\overline{RQS}$ from 96LS488
PB3 = $\overline{DRB}$ from 96LS488
PB4 = $\overline{D}$/S/E from 96LS488
PB5 = $\overline{NRFD}$ from IEEE-BUS

To use: Establish whether the IEEE-488 bus is ready to receive data by waiting for $\overline{DRB}$ (from Port B) to equal 0 and then output a byte of data to Port A.

In the IMT Data Logger the IEEE-488 board is only used for outputting data to the IEEE-Bus. This occurs in the LOG$DATA$NORMALIZE and BUFFTOIEEE$DUMP procedures.

6.5. Microcom 32k Memory board

The Buffer and Program Memory requirements for the Data Logger were met by using an "off-the-shelf" 32k byte S A Bus Memory board from Microcom (see Appendix G).

This board can accomodate a mixture of 2716 EPROM's and 6116 CMOS static RAM and is therefore very useful in this application.

The board start address was chosen to be 8000H which is the start address of the Buffer RAM. The Buffer RAM is 16k bytes long and extends from 8000H - BFFFH.

The CPU restarts at 0H but is immediately vectored to the program located at C000H. The program is approximately 8k long and therefore extends from C000H - DFFFH.

6.6. Data Loger P.S.U.

The power requirements for the Data Logger circuitry are as follows:

+5V at 3A - Tape drive and logic.
-5V at 500mA - Tape drive.
$\pm$15V at 35mA - low noise for Data Acquisition board.

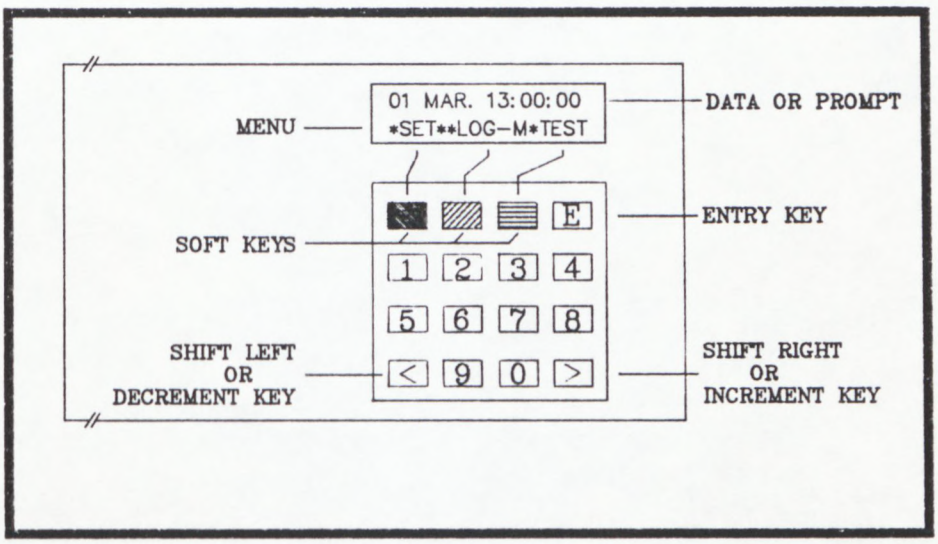
The +5V is supplied by an OLV15 (UGLY) power supply with the power transistor re-located to an external heat sink.

The -5V and $\pm$15V requirements are supplied by the Data Logger P.S.U. board circuitry (see Appendix G).

7. OPERATING INSTRUCTIONS:

The Data Logger is set-up and controlled by the user by means of the 16 key keyboard. (See System description 3.1)

The program displays either an explicit prompt like "Set Channel to be Displayed" or "Load Cassette" and then waits for the user to respond, or a menu like "SET LOG-M TEST" from which the user may select his choice via soft-key entry. The keyboard and display functions are illustrated below (Fig 5).

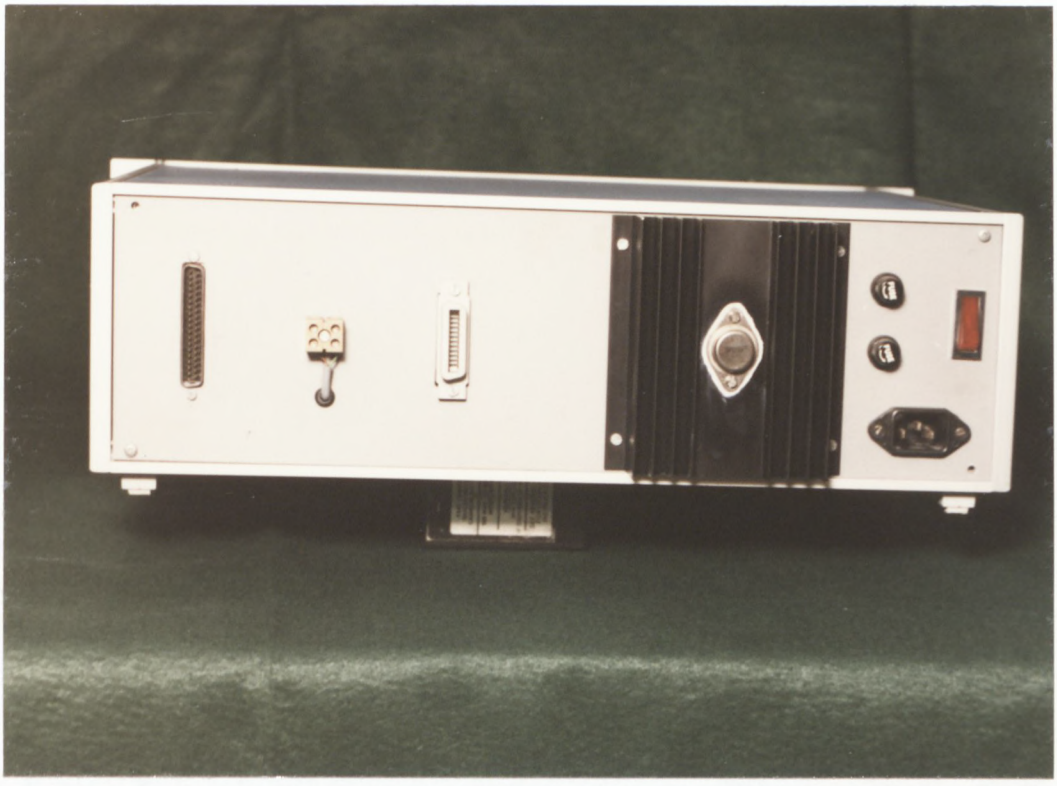


KEYBOARD & DISPLAY FUNCTIONS

FIG 5

The IEEE-488 interface connector, External Trigger connector block and the Data Acquisition connector along with the 220V mains plug, fuses and on/off switch are all located on the Data Logger back pannel. (Fig 6)

The pin allocation for the Data Acquisition connector is tabulated below (Fig 7). The DATA LOGGER IEEE-488 address is set at 10 (see Appendix G for switch settings).



BACK PANEL

FIG 6

CHANNEL	DEE CONNECTOR PIN	CHANNEL	DEE CONNECTOR PIN
1	20	17	28
2	21	18	29
3	22	19	30
4	23	20	31
5	24	21	32
6	25	22	33
7	26	23	34
8	27	24	35
9	1	25	9
10	2	26	10
11	3	27	11
12	4	28	12
13	5	29	13
14	6	30	14
15	7	31	15
16	8	32	16
GND	36	GND	17

DATA ACQUISITION CONNECTOR PIN ALLOCATION

FIG 7

To operate the Data Logger in the log mode the following sequence of steps should be taken.

1 Initial Power up (7.1)
2 Set Date and Time (7.2)
3 Set Logger Parameters (7.3)
4 Data Acquisition Test and Calibration (7.4) if required.
5 IEEE-488 Port Test (7.5) if required.
6 Cassette loading in record mode (7.6), referring to Tape Transport Prompts (7.7) when necessary.

To operate in the Data Playback mode, Initial Power up (7.1) and then Data Playback (7.8) should be followed.

7.1. Initial Power up

On initial power up and switch on the Main menu and a very mixed up date and time should be displayed on the LCD.

```
                ??  ???      ??:??:??
main menu       SET LOG-?  TEST
```

Note: The ?'s represent random characters at this stage.

If the above display is not in evidence then the mains switch should be switched off and then on once more. If the Main menu is still not displayed then the instrument is faulty.

7.2. Set Date and Time

To set the required date and time into the Data Logger the following steps must be followed, From the main menu push [SET] to get

"SET MODE"
TIME LOG EXIT

push [TIME] for

SET DATE
DDMMYY

Key in the required date in the format displayed using the [←] or [→] keys to edit any incorrectly keyed value.

For example: 1st March 1986 is keyed in as

[0], [1], [0], [3], [8], [6].

When you are satisfied that the date is correct push [E] to get

SET TIME
HHMMSS

The required time, in 24 hour clock format, is now keyed in.

example: 1pm is keyed in as [1], [3], [0], [0], [0], [0].

Wait for your time reference to reach the set value before pushing [E]. The program now returns to the main display.

Using the values in the above examples the display would look as follows:

01 MAR. 13:00:00
SET LOG-? TEST

With the time updating each second.

7.3. Set Logger Parameters:

Starting with main display push [SET] to get

"SET MODE"
TIME LOG EXIT

Push [LOG] for

IEEE MODE SET
OFF ON

[OFF] = Program does not output data to the IEEE-488 port.
[ON] = Program outputs the "Channel to be Displayed" data (see below) to the IEEE-488 port.

Note: The program will hang up in the Log or TEST-DACQ modes if no IEEE listener is connected to the Data Logger and [ON] is selected.

Upon selection of [OFF] or [ON] you get

TAPE MODE SET
REC PLAY OFF

[REC] = Program records data on cassette at the end of each log.

[PLAY] = Program enters the Tape Play-Back mode.

[OFF] = Program does not record data.

Upon selection of [REC] or [OFF] you get:

SET
NO OF CHANS: ??

Key in the required number of channels to be logged (limits 1-32), e.g. 9 is entered as [0], [9] and then [E] to get

SET
SCAN RATE: ??

Key in one of the codes tabulated below.

code	scan rate	code	scan rate
15	500m sec	12	62,5m sec
14	250m sec	11	31,25m sec
13	125m sec	10	15,625m sec

e.g. If code 15 was used each selected channel would be scanned once every 500m sec.

Key in [E] to get

SET
NO OF SCANS: ????

Key in the required number of scans for the log (limits 1-9999). e.g. 240 is keyed in as [0], [2], [4], [0] and then [E] to get

SET CHAN. TO BE
DISPLAYED: #??

The real-time logged data of the channel selected here is displayed on the LCD and (if the IEEE mode is "ON") is output to the IEEE-488 bus during the log.

Key in the required channel number (two digits) and then [E] to get.

SET TRIGGER
ALARM EXT MAN

This procedure allows the user to select one of three alternative ways to trigger a log:

1. [ALARM] - When the [ALARM] key is pushed the program displays:

SET TIME ALARM
HHMMSS

Key in the required log start time (24 hour clock) and then push [E] to return to the main display which will look as follows.

01 MAR. 13:00:00
SET LOG-A TEST

[LOG-A] can be used to display the ALARM time as follows:

LOG STARTS AT
12:34 EXIT

Push [EXT] to return to the main display.

Note: A log is started at the Alarm time and every 24 hours thereafter.

2. [EXIT] - This key is pushed to select the external trigger mode. Once it has been selected the program returns to the main display as follows:

01 MAR 13:00:00
SET LOG-X TEST

The program now allows a log to start every time the external contact points are shorted together.

[LOG-X] can be used to manually override the external trigger from the following display:

X-TRIG OVERRIDE?
NO YES EXIT

[NO] or [EXIT] allow the program to return to the main display.
[YES] starts a log and returns to the main display upon completion.

3. [MAN] - When this key is selected the program returns to the main display.

01 MAR 13:00:00
SET LOG-M TEST

In this mode a log is started every time [LOG-M] is pushed.

7.4. Data Acquisition Test and Calibration: Ensure that nothing is connected to the Data Acquisition port. From the main display select [TEST] to get

"TEST MODE"
D-ACQ IEEE EXIT

Push [D-ACQ] to get

CHAN # ?? -5000mV
ENTRY EXIT

Note: If instead of -5000mV the display is blank, then the IEEE mode is ON and no IEEE listener is connected to the IEEE port. Either the IEEE mode must be switched off (use [EXIT] and then 7.3 above) or an IEEE listener must be provided.

Select channel 1 by either using the [←], [→] or [ENTRY] keys. If [ENTRY] is pushed then the following will be displayed.

SET CHAN. TO BE
DISPLAYED: # ??

The key sequence [0], [1] and then [E] will result in the following display.

CHAN # 01 -5000mV
ENTRY EXIT

If a known voltage (between ± 5V) is now applied between the CH1 and ground pins (see Fig 7) on the Data Acquisitions connector this voltage should be displayed (±3mV) on the LCD in place of -5000mV.

Any channel can be selected and the above process repeated to ensure that all the Data Acquisition electronics are working correctly.

If the displayed voltages are incorrect follow the calibration procedure set out below:

Data Acquisition board Calibration:

1. Select Test Mode: [D-ACQ], CHAN # 01.
2. The display should read -5000mV with the input open circuit (no connection to the Data Acquisition connector).
3. Short CH1 input to Analog input ground. (pin 20 to pin 17 of Data Acquisition connector). Display should now read approximately 0000mV.
4. Check -15V rail for -15,000V using a high input impedance DVM (FLUKE 8502A or similar.) and analog ground as reference. Adjust VR1 on the P.S.U. board if necessary.
5. Connect the DVM between analog ground and the "link" between S&H and A/D on the Data Acquisition board.
6. Adjust the S&H pot. (VR3) for 0000mV on the DVM.
7. Remove short on input and attach a stable, adjustable DC voltage source between CH1 input and analog ground (pin 20 and 17 resp.)
8. Set input voltage to -49988V and adjust VR2 for a transition on the Data Logger LCD from -4998 to -5000mV.
9. Set input voltage to +49955V and adjust VR1 for a transition on the LCD from 4994 to 4997mV.
10. Do some random checks with various input voltage for confidence.

Note: The resolution of the A/D converter is 2,44mV per bit for the Data Logger.

To Return to the main display push [EXIT].

7.5. IEEE-488 Port Test

From the main display push [TEST] to get

"TEST MODE"
D-ACQ IEEE EXIT

Push [IEEE]. This program immediately outputs from 0 - 255 to the IEEE-488 port before returning to the main display. This test can be used to establish whether the Data Logger / IEEE-488 computer link is correctly made. (see Appendix E)

7.6. Cassette loading in Record Mode

If a cassette is being loaded for the first time or if "LOAD CASSETTE" is not at present being displayed then the following procedure must be used to load and initialize a cassette.

Follow the Set Logger Parameters procedure (7.3) until

TAPE MODE SET
REC PLAY OFF

With no cassette in the tape transport push [PLAY]. "LOAD CASSETTE" will now be displayed.

Insert a Data cassette (with record tabs intact).

TAPE INITIALIZE
IN PROGRESS

Will now be displayed until initiallization is complete and then the program will return to the main display to await the first log trigger.

7.7. Tape Transport Prompts:

In the record mode the program will attempt to dump data from the Buffer memory to tape at the end of each log. If a tape transport error occurs at this point one of the following tape error messages will be displayed:

"END OF TAPE!" - requires the operator to remove the tape at which time "LOAD CASSETTE!" will be displayed.

"LOAD CASSETTE!" - requires the operator to insert a tape (either new or side B of current tape) into the tape drive.

TAPE INITIALIZE
IN PROGRESS

will now be displayed until initiallization is complete and then the program will return to the main display to await the next log trigger.

"WRITE PROTECTED!" - requires the operator to replace the write protect tab or replace the tape before the program will continue. When the tape is removed from the tape transport "LOAD CASSETTE!" will be displayed.

7.8. Data Playback

The Data Logger must be connected to a computer (HP85 or HP9816) via the IEEE-488 port for data playback. The computer program must be waiting to receive the data before the [READ] key is pushed (see below) to initiate the data transfer sequence.

To select the Tape playback mode the Set Logger Parameters procedure (7.3) should be followed:

From the main display select [SET], [LOG], [ON] to get

TAPE MODE SET
REC PLAY OFF

Insert the pre-recorded Data Cassette and push [PLAY]

TAPE INITIALIZE
IN PROGRESS

will now be displayed until the tape is rewound and then

NEXT RECORD #01
READ ENTER EXIT

is displayed.

The record number can be incremented or decremented by the [←], [→] or [ENTER] KEYS.

When the desired record is displayed as the "Next Record" push [READ]. The data header and then the data will be output to the IEEE-488 port. Once the data transfer is complete the Record number is incremented by one and the display reverts to

NEXT RECORD #02
READ ENTER EXIT

The [EXIT] key returns you to the main display.

8. SERVICING

The Data Logger requires a minimal amount of servicing. This is limited to tape head cleaning at regular intervals (see MFE Manual) and recalibration of the data acquisition circuitry when deemed necessary. (see section 7.4).

9. SOFTWARE PROBLEMS AND ENHANCEMENTS

9.1. Problems

Initially, when the Data Logger was put into service a problem was discovered with the software (version V1.1). The problem was that the program would randomly "hang-up" as a log was triggered. This seemed to be caused by the external trigger interrupt occurring while the program was processing an "IF" statement. The actual cause was never found but would appear to be a peculiarity of the PL/M compiler.

The software was modified (Version V1.2) to get round this problem by masking out all interrupts during the sensitive part of the program. This is the current software version used in the IMT General Purpose Data Logger.

9.2. Enhancements

In the particular application for which the IMT Data Logger was originally designed the external log trigger was provided by a microwave doppler-shift movement detector. This device sometimes produced false triggers which resulted in large amounts of unwanted "data" being logged.

To overcome this problem the software was modified to include a qualify routine to determine whether the external trigger coincided with a signal before dumping to tape. (Version V1.3).

This was achieved by allowing all external triggers to initiate a log and then performing a "qualifying" test on the data in the Buffer memory.

The data of a pre-selected channel is passed through a software low-pass filter (see 9.3) and then the filtered minimum and maximum signal values are determined. The difference between maximum and minimum is then compared to a preset threshold value and only if the threshold value is exceeded is the data dumped to tape.

The above enhancement worked well but, because the external trigger device was range and velocity sensitive, the log trigger was sometimes too early and at other times too late to start a log. This resulted in the wanted signal being truncated. To overcome this problem it was decided to do away with the external trigger device and to design a software threshold trigger which would use the actual data signal to initiate a log. (Version V1.4).

The operation of the software trigger is similar to that of a digital oscilloscope with provision being made for selecting the threshold level (in mV) as well as the pre-trigger length (in samples) from the keyboard.

The software trigger is implemented by allowing the data logger to be continuously in the log mode. The logged data is stored in a revolving buffer memory which is set up to be a specific size determined by the number of scans and the number of channels selected. Data is stored in consecutive memory locations starting at the beginning of the buffer. A buffer-pointer is used to keep track of the next location to be filled and when the buffer is completely full the buffer-pointer is reset to 0.

It can be seen that this method allows the buffer to always contain "history" data equal to one complete log after one initial log period has been completed.

The pre-trigger function allows a selectable amount of this "history" data to be included at the beginning of a log so that the log data contains the complete signal and not only that portion after the threshold trigger has occurred.

A test is done at the end of each scan to determine if the pre-selected threshold on the specified channel has been reached. If it has, then a log is started with the pre-trigger data included at the beginning.

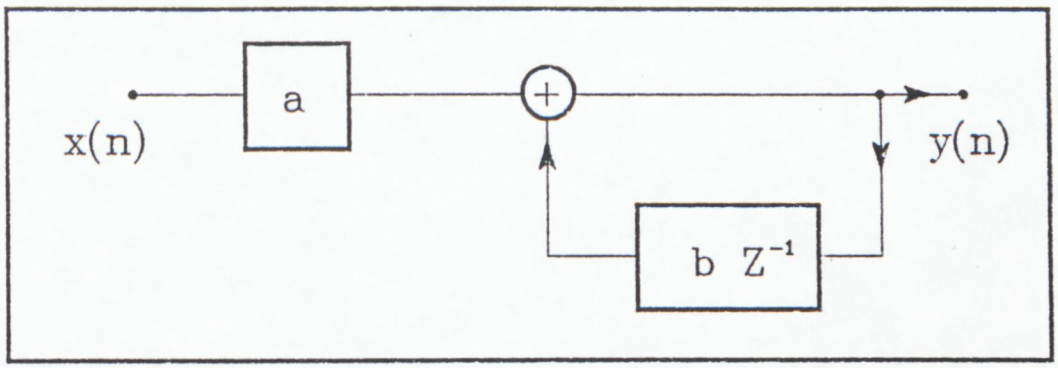
The trigger reference voltage, in this case, is not fixed but is a filtered version of the ambient background signal on the specified channel. A dynamic reference was chosen because the input signal d.c. level drifts due to its sensitive electronics and other background physical effects. To allow for this variation, but still have a constant threshold level, a software filter was designed to smooth the reference voltage.

9.3. Software Filter Design:

The filter is a first order Recursive low-pass filter of the form

$$y(n) = ax(n) + by(n - 1)$$

or diagramatically



The general transfer function for the above is:

$$H(z) = \frac{a}{1 - bz^{-1}}$$

For a - 3dB cut off frequency = fo
and sampling interval = T

$$b = e^{-2\pi\, fo\, T} \qquad \text{for } b < 1 \qquad \ldots\ldots\ldots (1)$$

let $\omega o = 2\pi$ fo and due to the Sampling Theorem $fo \leq \frac{1}{2T}$

or $\omega oT \leq \pi$

substituting in (1) $b = e^{-\omega oT}$

Note: for unity gain a + b = 1

For a required fo of 1×10^{-3} Hz with T = 500 msec.

b = 0,9969
a = 0,0031

but to simplify the software arithmetic the following values for a and b were chosen

b = 0,99; a = 0,01

Working back fo = $1,39 \times 10^{-3}$ Hz

This simplification allowed 16 bit integer arithmetic to be used and gave satisfactory results.

Briefly, the filter allows 1% of the most recent input signal to be summed with 99% of the history signal to provide a filtered output. The output takes up the value of the slowly changing d.c. offsets but filters out rapid fluctuations due to signal or noise.

10. CONCLUSION

A General Purpose Data Logger has been successfully designed, developed and built that meets the original system requirements. In several aspects it is more sophisticated than the original requirement called for but this increases its flexibility in its general purpose role.

Two Data Loggers have been in service for over a year, with one in continuous twenty four hour use. They have proved to be extremely reliable with only one component failure occurring during this time even though the remote unit is sited in a very hostile environment.

The operation of the remote instrument has been successfully undertaken by non-technical personnel who are required to change the cassette tapes when necessary and to occasionally reset the logger after a long power cut or an excessive power line surge or glitch.

The time and effort spent on designing modular electronics and software has been justified by the fact that, to date, six further projects have made use of the 8085 CPU board and at least half of these have also used the IEEE-488 and/or Data Acquisition boards. These projects would most probably not have been undertaken if not for the availability and built in flexibility of the hardware/software.

The experience and knowledge gained by undertaking this project cannot be quantified but will, I am sure, contribute significantly towards future IMT project goals.

REFERENCES

1. Analog Devices (1977) Analog-Digital Conversion Notes.

2. Analog Devices (1980) Designers Guide to High Resolution Products

3. Analog Devices (1984) Data Book Volume I

4. Analog Devices (1984) Analog Dialogue Volume 18, Number 3

5. Hewlett Packard HP85 Computer Manuals

6. Intel (1980) Peripheral Design Handbook

7. Intel Intellect Series II/85 Microcomputer Development System Documentation including ICE 85, PL/M - 80 and U.P.P Handbooks.

8. Larsen, Titus and Titus (1980) 8080/8085 Software Design Books 1 and 2

9. Larsen, Titus and Titus (1981) 8085A Cookbook

10. Mc Craken D.D. (1977) A Guide to PL/M Programming for Microcomputer Applications.

11. M.F.E. (1983) Model 450B Cassette Transport Interface and Installation Manual.

12. Pulse (Nov. 1981) SABUS

13. National Semiconductor (1982) Linear Databook.

APPENDIX A

Software Version V1.2

Program listings

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE DATALOGGER
OBJECT MODULE PLACED IN :F1:LOGMOD.OBJ
COMPILER INVOKED BY:  PLM80 :F1:LOGMOD.SRC DEBUG PRINT(:F1:LOGMOD.LST) DATE(01/05/85) PAGELENGTH(59)


                /*****************************************************************************/
                /*                        DATA LOGGER MAIN MODULE                            */
                /*                                                                           */
                /*****************************************************************************/
  1             DATA$LOGGER:
                DO;
  2     1        DECLARE DEC LITERALLY 'DECLARE';
  3     1        DEC LIT LITERALLY 'LITERALLY';
  4     1        DEC LEN LIT 'LENGTH';
  5     1        DEC CLEAR$DISP LIT '01H';
  6     1        DEC RETURN$HOME LIT '02H';
  7     1        DEC HOME$2 LIT '0C0H';
  8     1        DEC CURSOR$ON LIT '0EH';
  9     1        DEC CURSOR$OFF LIT '0CH';
 10     1        DEC MES$1 (*) BYTE DATA (0F3H,'SET',0F3H);
 11     1        DEC MES$2 (*) BYTE DATA (0F3H,'LOG',0F3H);
 12     1        DEC MES$3 (*) BYTE DATA ('TEST ');
 13     1        DEC MES$4 (*) BYTE DATA ('TIME',0F3H);
 14     1        DEC MES$6 (*) BYTE DATA ('EXIT ');
 15     1        DEC MES$7 (*) BYTE DATA ('"SET MODE"');
 16     1        DEC MES$17 (*) BYTE DATA ('IEEE ');
 17     1        DEC MES$20 (*) BYTE DATA ('D-ACQ');
 18     1        DEC MES$21 (*) BYTE DATA ('"TEST MODE"');
 19     1        DEC MES$22$INIT (*) BYTE DATA ('LOG- ');
 20     1        DEC MES$22 (5) BYTE ;
 21     1        DEC MES$27 (*) BYTE DATA ('TAPE ');
 22     1        DEC MES$29 (*) BYTE DATA ('MODE SET');
 23     1        DEC MES$30 (*) BYTE DATA (' REC ');
 24     1        DEC MES$31 (*) BYTE DATA ('PLAY ');
 25     1        DEC MES$32 (*) BYTE DATA (' OFF ');
 26     1        DEC MES$33 (*) BYTE DATA (' ON  ');
 27     1        DEC BLANK (*) BYTE DATA ('     ');
 28     1        DEC DATE$MES (*) BYTE DATA ('JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC');
 29     1        DEC (FLAG1,KEY) BYTE EXTERNAL;
 30     1        DEC RTC$DATE(6) BYTE EXTERNAL;
 31     1        DEC RTC$TIME(6) BYTE EXTERNAL;
 32     1        DEC (POSITION) BYTE EXTERNAL;
 33     1        DEC MONTH BYTE EXTERNAL;
 34     1        DEC TRIG$TYPE BYTE EXTERNAL;
 35     1        DEC TAPE$MODE$FLAG BYTE PUBLIC AT (2016H);   /* 0=RECORD,1=PLAY,2=OFF */
 36     1        DEC IEEE$MODE$FLAG BYTE PUBLIC AT (2017H);   /* 0=OFF,1=ON */
 37     1        DEC RST$55$FLAG BYTE EXTERNAL;
 38     1        DEC TAPE$ERROR$FLAG BYTE EXTERNAL;
 39     1        DEC (I,T) BYTE;

 40     1        S$MASK: PROCEDURE (MASK) EXTERNAL;
 41     2         DEC MASK BYTE;
 42     2        END S$MASK;

 43     1        R$MASK: PROCEDURE BYTE EXTERNAL;
```

```
 44   2        END R$MASK;

 45   1        KEY$INIT: PROCEDURE EXTERNAL;
 46   2        END KEY$INIT;

 47   1        RTC$INIT: PROCEDURE EXTERNAL;
 48   2        END RTC$INIT;

 49   1       INST$OUT: PROCEDURE (INST) EXTERNAL;
 50   2         DEC INST BYTE;
 51   2       END INST$OUT;

 52   1       C$OUT: PROCEDURE (CHAR) EXTERNAL;
 53   2         DEC CHAR BYTE;
 54   2       END C$OUT;

 55   1       TEXT$OUT: PROCEDURE(P4,SIZE) EXTERNAL;
 56   2         DEC P4 ADDRESS;
 57   2         DEC SIZE BYTE;
 58   2       END TEXT$OUT;

 59   1       LCD$INIT:PROCEDURE EXTERNAL;
 60   2        END LCD$INIT;

 61   1       KEY$ENTRY: PROCEDURE EXTERNAL;
 62   2       END KEY$ENTRY;

 63   1       SET$LOG$PARAMETERS: PROCEDURE EXTERNAL;
 64   2       END SET$LOG$PARAMETERS;

 65   1       TAPE$PLAY: PROCEDURE EXTERNAL;
 66   2       END TAPE$PLAY;

 67   1       SET$DATE$TIME: PROCEDURE(A) EXTERNAL;
 68   2        DEC A BYTE;
 69   2       END SET$DATE$TIME;

 70   1       RTC$DATA: PROCEDURE(A) EXTERNAL;
 71   2        DEC A BYTE;
 72   2       END RTC$DATA;

 73   1       MENU: PROCEDURE(P1,P2,P3) EXTERNAL;
 74   2        DEC (P1,P2,P3)ADDRESS;
 75   2       END MENU;

 76   1       TAPE$INIT:PROCEDURE EXTERNAL;
 77   2       END TAPE$INIT;

 78   1       D$AC0$INIT:PROCEDURE EXTERNAL;
 79   2       END D$AC0$INIT;

 80   1       D$AC0$TEST:PROCEDURE EXTERNAL;
 81   2       END D$AC0$TEST;

 82   1       IEEE$488$INIT:PROCEDURE EXTERNAL;
 83   2       END IEEE$488$INIT;
```

```
 84   1      IEEE$488$TEST:PROCEDURE EXTERNAL;
 85   2      END IEEE$488$TEST;

 86   1      LOG$TRIG: PROCEDURE EXTERNAL;
 87   2      END LOG$TRIG;

 88   1      LOG$SCAN: PROCEDURE EXTERNAL;
 89   2      END LOG$SCAN;

 90   1      TAPE$ERROR: PROCEDURE EXTERNAL;
 91   2      END TAPE$ERROR;

 92   1      TEST: PROCEDURE;
 93   2       CALL INST$OUT(CLEAR$DISP);
 94   2       CALL INST$OUT(82H);                    /* ADVANCE CURSOR TO POSITION 2 */
 95   2       CALL TEXT$OUT(.MES$21,LEN(MES$21));    /* TEST MODE */
 96   2       CALL MENU(.MES$20,.MES$17,.MES$6);     /* D-ACQ  IEEE  EXIT */
 97   2       KEY=0FH;
 98   2       DO WHILE KEY>2;
 99   3        FLAG1=0;
100   3        CALL KEY$ENTRY;
101   3       END;
102   2       DO CASE KEY;
103   3        CALL D$ACQ$TEST;
104   3        CALL IEEE$488$TEST;
105   3        ;
106   3       END;
107   2      END TEST;

108   1      SET$LOGGER$MODE: PROCEDURE;
109   2       CALL INST$OUT(CLEAR$DISP);
110   2       CALL INST$OUT(81H);
111   2       CALL TEXT$OUT(.MES$17,LEN(MES$17));    /* IEEE */
112   2       CALL TEXT$OUT(.MES$29,LEN(MES$29));    /* MODE SET */
113   2       CALL MENU(.MES$32,.MES$33,.BLANK);     /* OFF  ON  */
114   2       KEY=0FH;
115   2       DO WHILE KEY>1;
116   3        FLAG1=0;
117   3        CALL KEY$ENTRY;
118   3       END;
119   2       IEEE$MODE$FLAG=KEY;
120   2       CALL INST$OUT(CLEAR$DISP);
121   2       CALL INST$OUT(81H);
122   2       CALL TEXT$OUT(.MES$27,LEN(MES$27));    /* TAPE */
123   2       CALL TEXT$OUT(.MES$29,LEN(MES$29));    /* MODE SET */
124   2       CALL MENU(.MES$30,.MES$31,.MES$32);    /* REC  PLAY  OFF */
125   2       KEY=0FH;
126   2       DO WHILE KEY>2;
127   3        FLAG1=0;
128   3        CALL KEY$ENTRY;
129   3       END;
130   2       TAPE$MODE$FLAG=KEY;
131   2       IF TAPE$MODE$FLAG<>1 THEN
132   2        CALL SET$LOG$PARAMETERS;
133   2      END SET$LOGGER$MODE;
```

```
134   1       SET$MODE: PROCEDURE;
135   2         DISABLE;
136   2         CALL INST$OUT(CLEAR$DISP);
137   2         CALL INST$OUT(CURSOR$ON);
138   2         CALL INST$OUT(83H);                    /* ADVANCE CURSOR TO POSITION 3 */
139   2         CALL TEXT$OUT(.MES$7,LEN(MES$7));      /* SET MODE */
140   2         CALL MENU(.MES$4,.MES$2,.MES$6);       /* TIME  LOG  EXIT */
141   2         KEY=0FH;
142   2         DO WHILE KEY>2;
143   3         FLAG1=0;
144   3         CALL KEY$ENTRY;
145   3         END;
146   2         DO CASE(KEY);
147   3          CALL SET$DATE$TIME(0);
148   3          CALL SET$LOGGER$MODE;
149   3          ;
150   3         END;
151   2       END SET$MODE;

152   1       MAIN: PROCEDURE;
153   2        DEC (I,TEMP) BYTE;
154   2        DISABLE;
155   2        IF TAPE$MODE$FLAG=1 THEN               /* TAPE PLAY MODE? */
156   2         CALL TAPE$PLAY;
157   2        MES$22(4) = TRIG$TYPE;
158   2        KEY=0FH;
159   2        DO WHILE KEY>3;
160   3        CALL S$MASK(01FH);                     /* MASK OUT ALL! INTERUPTS */
161   3        CALL MENU(.MES$1,.MES$22,.MES$3);      /* SET  LOG-M  TEST */
162   3        IF (TRIG$TYPE='A') AND (RST$55$FLAG=0FFH) THEN /* ALARM MODE AND ALARM FLAG SET ? */
163   3          DO;
164   4           CALL S$MASK(01AH);                   /* ENABLE 7.5/5.5,RESET 7.5 F/F */
165   4           CALL LOG$SCAN;
166   4           RETURN;
167   4          END;
168   3        IF (TAPE$MODE$FLAG < 2) AND (TAPE$ERROR$FLAG > 0) THEN /* TAPE MODE NOT OFF AND */
                                                                    /* TAPE ERROR FLAG SET */
169   3         CALL TAPE$ERROR;
170   3        CALL INST$OUT(RETURN$HOME);
171   3        CALL INST$OUT(CURSOR$OFF);
172   3        CALL RTC$DATA(0);
173   3        DO I = 0 TO 1;
174   4         CALL C$OUT (RTC$DATE(I));             /* O/P DAY OF MONTH */
175   4        END;
176   3        CALL C$OUT(' ');
177   3        TEMP=MONTH;
178   3        IF TEMP>09H THEN                       /* CONVERT MONTH NUMBER TO */
179   3          TEMP=TEMP-6;                         /* A 3 CHARACTER WORD */
180   3        TEMP = (TEMP*3)-3;
181   3        DO I = 0 TO 2;
182   4         CALL C$OUT (DATE$MES(I+TEMP));        /* O/P MONTH WORD */
183   4        END;
184   3        CALL C$OUT ('.');
185   3        CALL C$OUT (' ');
186   3        DO I = 0 TO 5;
```

```
 187   4          IF (I = 2 )OR (I = 4 )THEN
 188   4           DO;
 189   5            CALL C$OUT(':');
 190   5            CALL C$OUT (RTC$TIME(I));              /* O/P TIME */
 191   5           END;
                   ELSE
 192   4            CALL C$OUT(RTC$TIME(I));
 193   4          END;
 194   3          T = R$MASK;
 195   3          IF (T AND 40H) = 40H THEN                /* ON RST 7.5 */
 196   3           KEY=INPUT(028H) AND 0FH;                /* READ KEYBOARD */
 197   3          IF (T AND 10H) = 10H THEN                /* ON RST 5.5 */
 198   3           RST$55$FLAG = 0FFH;                     /* SET RST 5.5 FLAG */
 199   3         END;
 200   2         CALL S$MASK(01AH);                       /* ENABLE 7.5/5.5,RESET 7.5 F/F */
 201   2         DO CASE(KEY);
 202   3          CALL SET$MODE;
 203   3          CALL LOG$TRIG;
 204   3          CALL TEST;
 205   3          CALL LOG$SCAN;                          /* EXTERNAL TRIGGER PARALLED WITH KEY #3 [E] */
 206   3         END;
 207   2        END MAIN;


                                    /*  MAIN PROGRAM  */
 208   1        CALL LCD$INIT;
 209   1        CALL KEY$INIT;
 210   1        CALL RTC$INIT;
 211   1        CALL TAPE$INIT;
 212   1        CALL D$ACQ$INIT;
 213   1        CALL IEEE$488$INIT;
 214   1        TAPE$ERROR$FLAG=0;
 215   1        RST$55$FLAG=0;
 216   1        DO I=0 TO 4;
 217   2         MES$22(I) = MES$22$INIT(I);
 218   2        END;
 219   1        DO WHILE 1;
 220   2         CALL MAIN;
 221   2         CALL S$MASK(01AH);                       /* ENABLE 7.5/5.5,RESET 7.5 F/F */
 222   2        END;
 223   1        END DATA$LOGGER;


MODULE INFORMATION:

     CODE AREA SIZE     = 03E0H    992D
     VARIABLE AREA SIZE = 0009H      9D
     MAXIMUM STACK SIZE = 000AH     10D
     259 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE KEYBOARDMODULE
OBJECT MODULE PLACED IN :F1:KEYMOD.OBJ
COMPILER INVOKED BY:  PLM80 :F1:KEYMOD.SRC DEBUG PRINT(:F1:KEYMOD.LST) DATE(14/01/85) PAGELENGTH(59)


                /*************************************************************************/
                /*                          KEYBOARD    MODULE                           */
                /*                                                                       */
                /*************************************************************************/
  1             KEYBOARD$MODULE:
                DO;
  2     1        DECLARE KEY$BOARD LITERALLY '028H';       /* KEYBOARD PORT ADDRESS */
  3     1        DECLARE SL$CURSOR LITERALLY '10H';
  4     1        DECLARE SR$CURSOR LITERALLY '14H';
  5     1        DECLARE KEY$ASCII BYTE PUBLIC;            /* DESIGNATED ASCII VALUE OF LAST KEY DEPRESSED */
  6     1        DECLARE FLAG1 BYTE PUBLIC;                /* RST 7.5 INTERRUPT FLAG (SET=FFH,CLEAR=0H) */
  7     1        DECLARE KEY BYTE PUBLIC;                  /* BINARY VALUE OF LAST KEY DEPRESSED */
                                                           /* PROVIDED INTERRUPTS ARE ENABLED */
  8     1        DECLARE CHAR (16) BYTE EXTERNAL;

  9     1        S$MASK: PROCEDURE (MASK) EXTERNAL;
 10     2         DECLARE MASK BYTE;
 11     2        END S$MASK;

 12     1        KEY$INIT: PROCEDURE PUBLIC;
 13     2         DECLARE JUMP BYTE AT (03CH);
 14     2         DECLARE VECTOR ADDRESS AT (03DH);
 15     2         JUMP = 0C3H;                             /* JMP OP CODE */
 16     2         VECTOR = .RST$75;                        /* INIT. RST7.5 JUMP VECTOR */
 17     2         CALL S$MASK(01BH);                       /* SET INTERRUPT MASK: ENABLE RST7.5,RESET F/F */
 18     2        END KEY$INIT;

 19     1       INST$OUT: PROCEDURE (INSTRUCTION) EXTERNAL;
 20     2         DECLARE INSTRUCTION BYTE;
 21     2       END INST$OUT;

 22     1       C$OUT: PROCEDURE (CHARACTER) EXTERNAL;
 23     2         DECLARE CHARACTER BYTE;
 24     2       END C$OUT;

 25     1       KEY$ENTRY: PROCEDURE PUBLIC;
 26     2        ENABLE;
 27     2        DO WHILE FLAG1 = 0H;                      /*  WAIT FOR RST7.5 INTERRUPT */
 28     3        END;                                      /* ie KEY DEPRESSION DETECTED */
 29     2        FLAG1 = 0H;                               /* RESET INTERRUPT FLAG */
 30     2       END KEY$ENTRY;

 31     1       RST$75: PROCEDURE;
 32     2        DISABLE;
 33     2        KEY = INPUT (KEY$BOARD) AND 0FH;          /* MASK OFF 4 MSB'S OF I/P VALUE */
 34     2        DO CASE (KEY);
 35     3         KEY$ASCII = 'G';                         /* ALLOCATE AN ASCII VALUE TO "KEY$ASCII" */
 36     3         KEY$ASCII = 'B';                         /* DEPENDING ON VALUE OF "KEY" */
 37     3         KEY$ASCII = 'Y';
 38     3         KEY$ASCII = 'E';
```

```
 39   3          KEY$ASCII = '1';
 40   3          KEY$ASCII = '2';
 41   3          KEY$ASCII = '3';
 42   3          KEY$ASCII = '4';
 43   3          KEY$ASCII = '5';
 44   3          KEY$ASCII = '6';
 45   3          KEY$ASCII = '7';
 46   3          KEY$ASCII = '8';
 47   3          KEY$ASCII = 'C';
 48   3          KEY$ASCII = '9';
 49   3          KEY$ASCII = '0';
 50   3          KEY$ASCII = 'F';
 51   3         END;
 52   2         FLAG1 = 0FFH;                          /* SET KEY INTERRUPT FLAG */
 53   2        END RST$75;

 54   1        KEY$IN: PROCEDURE PUBLIC;
 55   2          DECLARE I BYTE;
 56   2          I,FLAG1= 0;
 57   2          CALL KEY$ENTRY;                      /* WAIT FOR KEY DEPRESSION */
 58   2          DO WHILE KEY$ASCII <> 'E';           /* ENTER ? ,ASCII 'E',[E] */
 59   3           IF (KEY$ASCII > 029H) AND (KEY$ASCII < 040H) THEN  /* ASCII '0' TO '9' ? */
 60   3            DO;
 61   4             CHAR(I) = KEY$ASCII;              /* PUT ASCII VALUE OF "KEY" INTO STRING "CHAR" */
 62   4             CALL C$OUT(CHAR(I));              /* O/P ABOVE CHARACTER TO DISPLAY */
 63   4            END;
                   ELSE
 64   3            DO;
 65   4             IF KEY$ASCII = 'C' THEN          /* ASCII 'C'? , [<] */
 66   4              DO;
 67   5               I= I-2;
 68   5               CALL INST$OUT(SL$CURSOR);       /* BACKSPACE CURSOR */
 69   5              END;
                    ELSE
 70   4             IF KEY$ASCII='F' THEN            /* ASCII 'F'? [>] */
 71   4              CALL INST$OUT(SR$CURSOR);       /* ADVANCE CURSOR */
                    ELSE
 72   4               CALL C$OUT('?');               /* FOR ANY OTHER VALUE O/P '?' TO DISPLAY */
 73   4            END;
 74   3            I=I+1;
 75   3            CALL KEY$ENTRY;                  /* WAIT FOR NEXT KEY DEPRESSION */
 76   3          END;
 77   2         END KEY$IN;

 78   1        END KEYBOARD$MODULE;
```

MODULE INFORMATION:

```
     CODE AREA SIZE     = 0159H    345D
     VARIABLE AREA SIZE = 0004H      4D
     MAXIMUM STACK SIZE = 0002H      2D
     95 LINES READ
     0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE BINASCIIASCIIBINMODULE
OBJECT MODULE PLACED IN :F1:BASMOD.OBJ
COMPILER INVOKED BY:  PLM80 :F1:BASMOD.SRC DEBUG PRINT(:F1:BASMOD.LST) DATE(14/01/85) PAGELENGTH(59)


                /*************************************************************************/
                /*                  BINARY TO ASCII / ASCII TO BINARY                    */
                /*                              MODULE                                   */
                /*************************************************************************/
  1             BIN$ASCII$ASCII$BIN$MODULE:
                DO;
  2    1         DECLARE CHAR (16) BYTE EXTERNAL;
  3    1         DECLARE BIN$DATA ADDRESS PUBLIC;

  4    1        BIN$TO$ASCII: PROCEDURE(READING,SIZE) PUBLIC;/* CONVERTS BINARY TO 4 CHARACTER ASCII */
  5    2          DECLARE (BCD,I,SIZE) BYTE;
  6    2          DECLARE READING ADDRESS;                  /* "READING" = 0 TO 9999 (DECIMAL) MAX. */
  7    2          DECLARE TENS (*) ADDRESS DATA(1000,100,10,1);
  8    2          DO I =0 TO 3;
  9    3           BCD=0;
 10    3           DO WHILE READING >= TENS(I);             /* BINARY TO BCD CONVERSION */
 11    4            READING = READING-TENS(I);
 12    4            BCD=BCD+1;
 13    4           END;
 14    3           CHAR(I)=(BCD OR 30H);                    /* BCD TO ASCII CONVERSION */
 15    3          END;
 16    2          IF SIZE = 2 THEN                         /* LEFT JUSTIFY STRING (0023 > 2323)  */
 17    2           DO;
 18    3            CHAR(0)=CHAR(2);
 19    3            CHAR(1)=CHAR(3);
 20    3           END;
 21    2        END BIN$TO$ASCII;

 22    1        ASCII$TO$BIN: PROCEDURE(ASCII$ADR,SIZE) PUBLIC;/* CONVERTS ASCII TO BINARY "SIZE" */
 23    2          DECLARE (ASCII$ADR,BIN$DATA$ADR) ADDRESS;/* INDICATES NUMBER OF ASCII CHARACTERS (1-4) */
 24    2          DECLARE (SIZE,I) BYTE;
 25    2          DECLARE (ASCII$ARRAY BASED ASCII$ADR)(4) BYTE;/* START ADDRESS OF ASCII STRING */
 26    2          BIN$DATA=0;
 27    2          DO I =0 TO (SIZE-1);
                   /* BIN$DATA = BIN$DATA*10 + (ASCII$ARRAY(I) AND 0FH */
 28    3           BIN$DATA = SHL(BIN$DATA,1) + SHL(BIN$DATA,3) + (ASCII$ARRAY(I) AND 0FH);
 29    3          END;                                      /* "BIN$DATA" HOLDS RESULT IN BINARY */
 30    2        END ASCII$TO$BIN;

 31    1        END BIN$ASCII$ASCII$BIN$MODULE;


MODULE INFORMATION:

     CODE AREA SIZE     = 00C5H    197D
     VARIABLE AREA SIZE = 000DH     13D
     MAXIMUM STACK SIZE = 0002H      2D
     40 LINES READ
     0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE DISPLAYMODULE
OBJECT MODULE PLACED IN :F1:DSPMOD.OBJ
COMPILER INVOKED BY:  PLM80 :F1:DSPMOD.SRC DEBUG PRINT(:F1:DSPMOD.LST) DATE(14/01/85) PAGELENGTH(59)


                /*************************************************************************/
                /*                          DISPLAY     MODULE                           */
                /*                  (DENSITRON 2 LINE INTELIGENT LCD )                   */
                /*************************************************************************/
  1             DISPLAY$MODULE:
                DO;
  2     1        DECLARE CLEAR$DISP LITERALLY '01H';
  3     1        DECLARE RETURN$HOME LITERALLY '02H';      /* MOVE CURSOR TO LINE 1, POSITION 1 */
  4     1        DECLARE INST$REG LITERALLY '030H';
  5     1        DECLARE D$REG LITERALLY '031H';
  6     1        DECLARE ENTRY$MODE$INC LITERALLY '06H';
  7     1        DECLARE CURSOR$ON LITERALLY '0EH';
  8     1        DECLARE CURSOR$OFF LITERALLY '0CH';
  9     1        DECLARE TWO$LINE LITERALLY '038H';
 10     1        DECLARE WORD$LENGTH LITERALLY '030H';
 11     1        DECLARE BLINK LITERALLY '0FH';
 12     1        DECLARE SL$CURSOR LITERALLY '10H';
 13     1        DECLARE SR$CURSOR LITERALLY '14H';
 14     1        DECLARE HOME$2 LITERALLY '0C0H';          /* MOVE CURSOR TO LINE 2, POSITION 1 */
 15     1        DECLARE CHAR (16) BYTE PUBLIC;            /* TEMP. STRING STORAGE */
 16     1        DECLARE (POSITION) BYTE PUBLIC;           /* CURRENT CURSOR POSITION */

 17     1       LCD$BUSY:PROCEDURE;
 18     2         DECLARE FLAG BYTE;
 19     2         DISABLE;
 20     2         FLAG= 0FFH;
 21     2         DO WHILE FLAG > 0H;                     /* WAIT WHILE LCD PROCESSOR BUSY */
 22     3          FLAG = INPUT(INST$REG) AND 080H;
 23     3         END;
 24     2         POSITION = INPUT(INST$REG) AND 07FH;    /* UPDATE (CURSOR) POSITION */
 25     2         ENABLE;
 26     2       END LCD$BUSY;

 27     1       INST$OUT: PROCEDURE (INST) PUBLIC;
 28     2         DECLARE INST BYTE;
 29     2         OUTPUT(INST$REG) = INST;                /* O/P 1 INSTRUCTION TO LCD PROCESSOR */
 30     2         CALL LCD$BUSY;
 31     2       END INST$OUT;

 32     1       LCD$INIT:PROCEDURE PUBLIC;
 33     2        DECLARE I BYTE;
 34     2        DECLARE LCD$INITIAL$CONDITIONS(*) BYTE DATA (WORD$LENGTH,WORD$LENGTH,TWO$LINE,
                                                             CURSOR$ON,ENTRY$MODE$INC,CLEAR$DISP);
 35     2         DO I=0 TO LAST(LCD$INITIAL$CONDITIONS);
 36     3          CALL INST$OUT (LCD$INITIAL$CONDITIONS(I));
 37     3         END;
 38     2       END LCD$INIT;

 39     1       C$OUT: PROCEDURE (CHAR) PUBLIC;           /* O/P 1 CHARACTER TO LCD AT PRESENT */
 40     2       DECLARE CHAR BYTE;                        /* CURSOR POSITION & INC "POSITION" */
```

```
 41   2        OUTPUT(D$REG) = CHAR;
 42   2        CALL LCD$BUSY;
 43   2      END C$OUT;

 44   1      TEXT$OUT: PROCEDURE(P4,SIZE) PUBLIC;      /* O/P MESSAGE , WHOSE */
 45   2        DECLARE P4 ADDRESS;                     /* STARTING ADDRESS IS PASSED TO P4, */
 46   2        DECLARE (SIZE,I) BYTE;                  /* OF LENGTH "SIZE" */
 47   2        DECLARE (TEXT$4 BASED P4)(1)BYTE;
 48   2        DO I=0 TO (SIZE-1);
 49   3         CALL C$OUT(TEXT$4(I));
 50   3        END;
 51   2      END TEXT$OUT;

 52   1      MENU: PROCEDURE(P1,P2,P3) PUBLIC;      /* DISPLAYS 3 SOFT-KEY MESSAGES , OF 5  */
 53   2       DECLARE (P1,P2,P3)ADDRESS;            /* CHARACTERS EACH , ON LINE 2 */
 54   2       DECLARE (TEXT1 BASED P1)(5) BYTE;     /* STARTING ADDRESSES PASSED TO P1,P2 & P3 RESP. */
 55   2       DECLARE (TEXT2 BASED P2)(5) BYTE;     /* (LAST MESSAGE,LAST CHARACTER NOT DISPLAYD!) */
 56   2       DECLARE (TEXT3 BASED P3)(5) BYTE;
 57   2       DECLARE (I,N) BYTE;
 58   2       CALL INST$OUT(HOME$2);
 59   2       I,N=0;
 60   2       DO N=0 TO 2;
 61   3        DO I=0 TO 4;
 62   4         DO CASE(N);
 63   5          CALL C$OUT(TEXT1(I));
 64   5          CALL C$OUT(TEXT2(I));
 65   5          CALL C$OUT(TEXT3(I));
 66   5         END;
 67   4        END;
 68   3       CALL C$OUT(0F3H);
 69   3       END;
 70   2      END MENU;

 71   1      END DISPLAY$MODULE;


MODULE INFORMATION:

     CODE AREA SIZE     = 0127H    295D
     VARIABLE AREA SIZE = 0021H     33D
     MAXIMUM STACK SIZE = 0004H      4D
     84 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE REALTIMECLOCKMODULE
OBJECT MODULE PLACED IN :F1:RTCMOD.OBJ
COMPILER INVOKED BY:  PLM80 :F1:RTCMOD.SRC DEBUG PRINT(:F1:RTCMOD.LST) DATE(14/01/85) PAGELENGTH(59)


                /*****************************************************************************/
                /*                      REAL-TIME CLOCK  MODULE                              */
                /*                           (MC146818)                                      */
                /*****************************************************************************/
  1             REAL$TIME$CLOCK$MODULE:
                DO;
  2    1         DECLARE CLEAR$DISP LITERALLY '01H';
  3    1         DECLARE DISP$CURSOR$ON LITERALLY '0EH';
  4    1         DECLARE BLINK LITERALLY '0FH';
  5    1         DECLARE HOME$2 LITERALLY '0C0H';
  6    1         DECLARE MES$14(*) BYTE DATA ('ALARM');
  7    1         DECLARE DATE$MENU (*) BYTE DATA ('SET DATE DDMMYY');
  8    1         DECLARE TIME$MENU (*) BYTE DATA ('SET TIME HHMMSS');
  9    1         DECLARE (FLAG1,KEY) BYTE EXTERNAL;
 10    1         DECLARE RTC$DATE(6) BYTE PUBLIC;
 11    1         DECLARE RTC$TIME(6) BYTE PUBLIC;
 12    1         DECLARE (I,N) BYTE;
 13    1         DECLARE CHAR (16) BYTE EXTERNAL;
 14    1         DECLARE POSITION BYTE EXTERNAL;
 15    1         DECLARE TEMP BYTE;
 16    1         DECLARE SECONDS BYTE AT (2000H);
 17    1         DECLARE MINUTES BYTE AT (2002H);
 18    1         DECLARE HOURS BYTE AT (2004H);
 19    1         DECLARE DAY$OF$WEEK BYTE AT (2006H);
 20    1         DECLARE DAY$OF$MONTH BYTE AT (2007H);
 21    1         DECLARE MONTH BYTE PUBLIC AT (2008H);
 22    1         DECLARE YEAR BYTE AT (2009H);
 23    1         DECLARE SECONDS$ALARM BYTE AT (2001H);
 24    1         DECLARE MINUTES$ALARM BYTE AT (2003H);
 25    1         DECLARE HOURS$ALARM BYTE AT (2005H);
 26    1         DECLARE REG$A BYTE PUBLIC AT (200AH);
 27    1         DECLARE REG$B BYTE PUBLIC AT (200BH);
 28    1         DECLARE REG$C BYTE AT (200CH);
 29    1         DECLARE REG$D BYTE AT (200DH);
 30    1         DECLARE RST$55$FLAG BYTE PUBLIC;        /* CLOCK INTERRUPT FLAG */
 31    1         DECLARE SCAN$RATE BYTE EXTERNAL;

 32    1        INST$OUT: PROCEDURE (INST) EXTERNAL;
 33    2         DECLARE INST BYTE;
 34    2        END INST$OUT;

 35    1        S$MASK: PROCEDURE (MASK) EXTERNAL;
 36    2         DECLARE MASK BYTE;
 37    2        END S$MASK;

 38    1        TEXT$OUT: PROCEDURE (P4,SIZE) EXTERNAL;
 39    2         DECLARE P4 ADDRESS;
 40    2         DECLARE SIZE BYTE;
 41    2        END TEXT$OUT;
```

```
42   1      KEY$IN: PROCEDURE EXTERNAL;
43   2      END KEY$IN;

44   1      RST$55: PROCEDURE PUBLIC;
45   2       DECLARE C BYTE;
46   2       C = REG$C;                                  /* CLEAR IRQ */
47   2       RST$55$FLAG=0FFH;
48   2       ENABLE;
49   2      END RST$55;

50   1      RTC$INIT: PROCEDURE PUBLIC;
51   2        DECLARE JUMP$55 BYTE AT (02CH);
52   2        DECLARE VECTOR$55 ADDRESS AT (02DH);      /* INITIALIZE RST 5.5 JUMP VECTOR */
53   2        JUMP$55 = 0C3H;                           /* JUMP OP CODE */
54   2        VECTOR$55 = .RST$55;
55   2        REG$A = SCAN$RATE AND 0FH;                /* INIT SCAN RATE (PIE RATE) */
56   2        REG$B = 02H;                              /* PIE DISABLE */
57   2        CALL S$MASK(01AH); /* SET INTERRUPT MASK (ENABLE 7.5 & 5.5,RESET 7.5 F/F)  */
58   2      END RTC$INIT;

59   1      RTC$INIT$DATA:PROCEDURE(A);                 /* SETS RTC INITIAL DATA */
60   2       DECLARE A BYTE;
61   2       DO I=0 TO 4 BY 2;
62   3        TEMP = (CHAR(I+1) AND 0FH) OR SHL((CHAR(I) AND 0FH),4);/* CONVERT 2 ASCII CHARS. */
                                                                    /* TO PACKED BCD */
63   3        IF A=1 THEN                               /* IF A=1 THEN SET ALARM */
64   3         DO CASE (I+N);
65   4         ;
66   4          HOURS$ALARM = TEMP;
67   4         ;
68   4          MINUTES$ALARM = TEMP;
69   4         ;
70   4          SECONDS$ALARM = TEMP;
71   4         END;
                ELSE
72   3         DO CASE (I+N);
73   4          DAY$OF$MONTH = TEMP;
74   4          HOURS = TEMP;
75   4          MONTH = TEMP;
76   4          MINUTES = TEMP;
77   4          YEAR = TEMP;
78   4          SECONDS = TEMP;
79   4         END;
80   3       END;
81   2      END RTC$INIT$DATA;

82   1      SET$DATE$TIME: PROCEDURE(A) PUBLIC;           /* A=1 FOR ALARM SET */
83   2       DECLARE A BYTE;
84   2       IF A=0 THEN
85   2        DO;
86   3         REG$B = 10000010B;                        /* SET,BCD,24 HR */
87   3         REG$A = 00H;                              /* INITIALIZE REG A */
88   3        END;
89   2       DO N = 0 TO 1;
90   3         IF A=1 THEN N=1;                          /*SET ONLY TIME IN ALARM MODE!! */
92   3         CALL INST$OUT(CLEAR$DISP);
```

```
 93   3          DO CASE A;
 94   4           CALL INST$OUT(84H);                    /* POSITION CURSOR */
 95   4           CALL INST$OUT(81H);                    /*        "        */
 96   4          END;
 97   3          DO CASE N;
 98   4           CALL TEXT$OUT(.DATE$MENU,9);           /* SET DATE */
 99   4           CALL TEXT$OUT(.TIME$MENU,9);           /* SET TIME */
100   4          END;
101   3          DO CASE A;
102   4           ;
103   4           CALL TEXT$OUT(.MES$14,5);              /* ALARM */
104   4          END;
105   3          CALL INST$OUT(HOME$2 +5);              /* MOVE CURSOR TO LINE 2 POSITION 5 */
106   3          DO CASE N;
107   4           CALL TEXT$OUT(.DATE$MENU(9),6);        /* DDMMYY */
108   4           CALL TEXT$OUT(.TIME$MENU(9),6);        /* HHMMSS */
109   4          END;
110   3         CALL INST$OUT((POSITION-6)OR 80H);       /* MOVE CURSOR -6 PLACES */
111   3         CALL INST$OUT(BLINK);                    /* BLINK ON */
112   3         CALL KEY$IN;
113   3         CALL RTC$INIT$DATA(A);
114   3        END;
115   2        REG$B = 00000010B;                        /* GET OUT OF CLOCK SET MODE */
116   2        CALL INST$OUT(DISP$CURSOR$ON);            /* BLINK OFF */
117   2        CALL INST$OUT(CLEAR$DISP);
118   2       END SET$DATE$TIME;

119   1       RTC$DATA:PROCEDURE(A) PUBLIC;
120   2        DECLARE (I,N,A) BYTE;
121   2        DECLARE ASCII (6) BYTE;
122   2        DISABLE;
123   2        DO WHILE (REG$A AND 80H)<>0;
124   3        END;
125   2        DO N =0 TO 1;
126   3         DO I =0 TO 4 BY 2;
127   4          IF A=1 THEN
128   4          DO CASE (I+N);
129   5           TEMP = 0;
130   5           TEMP = HOURS$ALARM;
131   5           TEMP = 0;
132   5           TEMP = MINUTES$ALARM;
133   5           TEMP = 0;
134   5           TEMP = SECONDS$ALARM;
135   5          END;
               ELSE
136   4          DO CASE (I+N);
137   5          TEMP = DAY$OF$MONTH;
138   5          TEMP = HOURS;
139   5          TEMP = MONTH;
140   5          TEMP = MINUTES;
141   5          TEMP = YEAR;
142   5          TEMP = SECONDS;
143   5         END;
144   4         ASCII(I) = SHR(TEMP,4) OR 30H;      /* CONVERTS PACKED BCD TO 2 ASCII CHARACTERS */
145   4         ASCII(I+1) = (TEMP AND 0FH) OR 30H;                    /*  "  */
146   4        END;
```

```
 147   3          DO CASE N;
 148   4            DO I = 0 TO 5;
 149   5              RTC$DATE(I) = ASCII(I);
 150   5            END;
 151   4            DO I = 0 TO 5;
 152   5              RTC$TIME(I) = ASCII(I);
 153   5            END;
 154   4          END;
 155   3        END;
 156   2        ENABLE;
 157   2      END RTC$DATA;

 158   1      END REAL$TIME$CLOCK$MODULE;


MODULE INFORMATION:

     CODE AREA SIZE     = 03D3H    979D
     VARIABLE AREA SIZE = 001CH     28D
     MAXIMUM STACK SIZE = 0004H      4D
     175 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE SETLOGPARAMETERMODULE
OBJECT MODULE PLACED IN :F1:SLPMOD.OBJ
COMPILER INVOKED BY:  PLM80 :F1:SLPMOD.SRC DEBUG PRINT(:F1:SLPMOD.LST) DATE(14/01/85) PAGELENGTH(59)


                /*****************************************************************/
                /*                    SET LOG PARAMETER MODULE                   */
                /*                                                               */
                /*****************************************************************/
  1             SET$LOG$PARAMETER$MODULE:
                DO;
  2    1         DECLARE DEC LITERALLY 'DECLARE';
  3    1         DEC LIT LITERALLY 'LITERALLY';
  4    1         DEC LEN LIT 'LENGTH';
  5    1         DEC CLEAR$DISP LIT '01H';
  6    1         DEC RETURN$HOME LIT '02H';
  7    1         DEC HOME$2 LIT '0C0H';
  8    1         DEC CURSOR$ON LIT '0EH';
  9    1         DEC MES$1 (*) BYTE DATA (0F3H,'SET',0F3H);
 10    1         DEC MES$5 (*) BYTE DATA (0F3H,'MAN',0F3H);
 11    1         DEC MES$6 (*) BYTE DATA ('EXIT');
 12    1         DEC MES$8 (*) BYTE DATA (' NO OF CHANS:');
 13    1         DEC MES$9 (*) BYTE DATA ('  SCAN RATE:');
 14    1         DEC MES$10(*) BYTE DATA ('NO OF SCANS:');
 15    1         DEC MES$11(*) BYTE DATA ('SET CHAN. TO BE');
 16    1         DEC MES$12(*) BYTE DATA ('DISPLAYED:');
 17    1         DEC MES$13(*) BYTE DATA ('  SET TRIGGER');
 18    1         DEC MES$14(*) BYTE DATA ('ALARM');
 19    1         DEC MES$15(*) BYTE DATA (0F3H,'EXT',0F3H);
 20    1         DEC MES$18(*) BYTE DATA ('LOG STARTS AT');
 21    1         DEC MES$24(*) BYTE DATA (' NO  ');
 22    1         DEC MES$25(*) BYTE DATA (' YES ');
 23    1         DEC MES$26(*) BYTE DATA ('X-TRIG OVERRIDE?');
 24    1         DEC (FLAG1,KEY) BYTE EXTERNAL;
 25    1         DEC CHAR (16) BYTE EXTERNAL;
 26    1         DEC (POSITION) BYTE EXTERNAL;
 27    1         DEC TRIG$TYPE BYTE PUBLIC AT (2010H);       /* A=ALARM,X=EXTERNAL,M=MANUAL */
 28    1         DEC NO$OF$CHANS BYTE PUBLIC AT (2011H);
 29    1         DEC SCAN$RATE BYTE PUBLIC AT (2012H);       /* 15=500ms,14=250ms,13=125ms etc */
 30    1         DEC NO$OF$SCANS ADDRESS PUBLIC AT (2013H);
 31    1         DEC (DISP) BYTE PUBLIC AT (2015H);          /* CHANNEL TO BE DISPLAYED */

 32    1         DEC BIN$DATA ADDRESS EXTERNAL ;
 33    1         DEC RTC$TIME(6) BYTE EXTERNAL;
 34    1         DEC RST$55$FLAG BYTE EXTERNAL;
 35    1         DEC REG$B BYTE EXTERNAL;

 36    1        INST$OUT: PROCEDURE (INST) EXTERNAL;
 37    2          DEC INST BYTE;
 38    2        END INST$OUT;

 39    1        C$OUT: PROCEDURE (CHAR) EXTERNAL;
 40    2          DEC CHAR BYTE;
 41    2        END C$OUT;
```

```
42   1       TEXT$OUT: PROCEDURE(P4,SIZE) EXTERNAL;
43   2         DEC P4 ADDRESS;
44   2         DEC SIZE BYTE;
45   2       END TEXT$OUT;

46   1       BIN$TO$ASCII: PROCEDURE(READING,SIZE) EXTERNAL;
47   2         DEC SIZE BYTE;
48   2         DEC READING ADDRESS;
49   2       END BIN$TO$ASCII;

50   1       ASCII$TO$BIN: PROCEDURE(ASCII$ADR,SIZE) EXTERNAL;
51   2         DEC ASCII$ADR ADDRESS;
52   2         DEC SIZE BYTE;
53   2       END ASCII$TO$BIN;

54   1       KEY$ENTRY: PROCEDURE EXTERNAL;
55   2       END KEY$ENTRY;

56   1       KEY$IN: PROCEDURE EXTERNAL;
57   2        END KEY$IN;

58   1       SET$DATE$TIME: PROCEDURE(A) EXTERNAL;
59   2        DEC A BYTE;
60   2       END SET$DATE$TIME;

61   1       LOG$PARAM$2: PROCEDURE;
62   2         CALL TEXT$OUT(.CHAR,2);
63   2         CALL INST$OUT((POSITION-2) OR 80H);
64   2         CALL KEY$IN;
65   2       END LOG$PARAM$2;

66   1       LOG$PARAM$4: PROCEDURE;
67   2         CALL TEXT$OUT(.CHAR,4);
68   2         CALL INST$OUT((POSITION-4) OR 80H);
69   2         CALL KEY$IN;
70   2       END LOG$PARAM$4;

71   1       MENU: PROCEDURE(P1,P2,P3) EXTERNAL;
72   2        DEC (P1,P2,P3)ADDRESS;
73   2       END MENU;

74   1       RTC$DATA: PROCEDURE(A) EXTERNAL;
75   2        DEC A BYTE;
76   2       END RTC$DATA;

77   1       LOG$SCAN: PROCEDURE EXTERNAL;
78   2       END LOG$SCAN;

79   1       ALARM$DISP: PROCEDURE ;                    /* DISPLAY ALARM TIME */
80   2         CALL INST$OUT(CLEAR$DISP);
81   2         CALL TEXT$OUT(.MES$18,LEN(MES$18));      /* LOG STARTS AT */
82   2         CALL RTC$DATA(1);
83   2         CALL INST$OUT(HOME$2);
84   2         CALL TEXT$OUT(.RTC$TIME(0),2);
85   2         CALL C$OUT(':');
86   2         CALL TEXT$OUT(.RTC$TIME(2),2);
```

```
 87   2        CALL INST$OUT((POSITION +7) OR 80H);
 88   2        CALL TEXT$OUT(.MES$6,4);                    /* EXIT */
 89   2      END ALARM$DISP;

 90   1      X$OVERIDE:PROCEDURE;
 91   2        CALL INST$OUT(CLEAR$DISP);
 92   2        CALL TEXT$OUT(.MES$26,LEN(MES$26));         /* X-TRIG OVERRIDE? */
 93   2        CALL MENU(.MES$24,.MES$25,.MES$6);          /* NO  YES  EXIT */
 94   2      END X$OVERIDE;

 95   1      LOG$TRIG:PROCEDURE PUBLIC;
 96   2       KEY=0FH;      /* CLEAR KEY */
 97   2       IF TRIG$TYPE = 'A' THEN                     /* ALARM TRIGGER? */
 98   2        DO;
 99   3         CALL ALARM$DISP;
100   3         DO WHILE KEY()2;                          /* EXIT? */
101   4          ENABLE;
102   4          DO WHILE (FLAG1=0) AND (RST$55$FLAG=0);/* WAIT FOR KEY OR ALARM INTERRUPT */
103   5          END;
104   4          IF RST$55$FLAG=0FFH THEN
105   4           CALL LOG$SCAN;
106   4         END;
107   3        END;
108   2       IF TRIG$TYPE = 'X' THEN                     /* EXTERNAL TRIGGER? */
109   2        DO;
110   3         CALL X$OVERIDE;                           /* X-TRIG OVERRIDE */
111   3         DO WHILE KEY)3;
112   4          FLAG1=0;
113   4          CALL KEY$ENTRY;
114   4         END;
115   3         DO CASE KEY;
116   4           ;                                       /* NO */
117   4           CALL LOG$SCAN;                          /* YES */
118   4           ;                                       /* EXIT */
119   4           CALL LOG$SCAN;                          /* EXTERNAL START */
120   4         END;
121   3        END;
122   2       IF TRIG$TYPE ='M' THEN                      /* MANUAL TRIGGER? */
123   2        CALL LOG$SCAN;
124   2      END LOG$TRIG;

125   1      CHAN$DISP$SELECT: PROCEDURE PUBLIC;
126   2        CALL INST$OUT(CLEAR$DISP);
127   2        CALL INST$OUT(CURSOR$ON);
128   2        CALL TEXT$OUT(.MES$11,LEN(MES$11));         /* SET CHAN. TO BE */
129   2        CALL INST$OUT(HOME$2);
130   2        CALL TEXT$OUT(.MES$12,LEN(MES$12));         /* DISPLAYED */
131   2        CALL C$OUT(' ');
132   2        CALL C$OUT('#');
133   2        CALL BIN$TO$ASCII(DISP,2);
134   2        CALL LOG$PARAM$2;
135   2        CALL ASCII$TO$BIN(.CHAR,2);
136   2        DISP=BIN$DATA;
137   2      END CHAN$DISP$SELECT;

138   1      SET$LOG$PARAMETERS: PROCEDURE PUBLIC;
```

```
139   2        DEC I BYTE;
140   2        DO I=0 TO 2;
141   3         CALL INST$OUT(CLEAR$DISP);
142   3         CALL INST$OUT(85H);                    /* CURSOR POSITION=5 */
143   3         CALL TEXT$OUT(.MES$1,LEN(MES$1));      /* SET */
144   3         CALL INST$OUT(HOME$2);
145   3         DO CASE I;
146   4          DO;
147   5           CALL TEXT$OUT(.MES$8,LEN(MES$8));    /* NO OF CHANS */
148   5           CALL BIN$TO$ASCII(NO$OF$CHANS,2);
149   5           CALL LOG$PARAM$2;
150   5           CALL ASCII$TO$BIN(.CHAR,2);
151   5           NO$OF$CHANS=BIN$DATA;
152   5          END;
153   4          DO;
154   5           CALL TEXT$OUT(.MES$9,LEN(MES$9));    /* SCAN RATE */
155   5           CALL BIN$TO$ASCII(SCAN$RATE,2);
156   5           CALL LOG$PARAM$2;
157   5           CALL ASCII$TO$BIN(.CHAR,2);
158   5           SCAN$RATE=BIN$DATA;
159   5          END;
160   4          DO;
161   5           CALL TEXT$OUT(.MES$10,LEN(MES$10));  /* NO OF SCANS */
162   5           CALL BIN$TO$ASCII(NO$OF$SCANS,4);
163   5           CALL LOG$PARAM$4;
164   5           CALL ASCII$TO$BIN(.CHAR,4);-
165   5           NO$OF$SCANS=BIN$DATA;
166   5          END;
167   4         END;
168   3        END;
169   2        CALL CHAN$DISP$SELECT;                   /* SET CHANNEL TO BE DISPLAYED */
170   2        CALL INST$OUT(CLEAR$DISP);
171   2        CALL INST$OUT(82H);
172   2        CALL TEXT$OUT(.MES$13,LEN(MES$13));      /* SET TRIGGER */
173   2        CALL MENU(.MES$14,.MES$15,.MES$5);       /* ALARM  EXT  MAN */
174   2        KEY = 0FH;
175   2        DO WHILE KEY>2;
176   3         FLAG1=0;
177   3         CALL KEY$ENTRY;
178   3        END;
179   2        DO CASE (KEY);
180   3         DO;
181   4          TRIG$TYPE=('A');                      /* ALARM TRIGGER */
182   4          CALL SET$DATE$TIME(1);                /* SET ALARM TIME */
183   4          RST$55$FLAG=0;
184   4          REG$B=22H;                            /* ENABLE ALARM */
185   4         END;
186   3         TRIG$TYPE=('X');                       /* EXTERNAL TRIGGER */
187   3         TRIG$TYPE=('M');                       /* MANUAL TRIGGER */
188   3        END;
189   2      END SET$LOG$PARAMETERS;

190   1      END SET$LOG$PARAMETER$MODULE;
```

MODULE INFORMATION:

```
     CODE AREA SIZE     = 0325H    805D
     VARIABLE AREA SIZE = 0001H      1D
     MAXIMUM STACK SIZE = 0006H      6D
     215 LINES READ
     0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE IEEE488MODULE
OBJECT MODULE PLACED IN :F1:IEEE.OBJ
COMPILER INVOKED BY:  PLM80 :F1:IEEE.SRC DEBUG PRINT(:F1:IEEE.LST) DATE(14/01/85) PAGELENGTH(59)



                /*****************************************************************************/
                /*                          IEEE 488 MODULE                                  */
                /*                        TALK/LISTEN MODE                                   */
                /*****************************************************************************/
  1             IEEE$488$MODULE:
                DO;
  2   1          DECLARE DEC LITERALLY 'DECLARE';

  3   1          IEEE$488$INIT: PROCEDURE PUBLIC;
  4   2           OUTPUT(63H)=0C2H;                    /* INIT 8255 */
  5   2           OUTPUT(62H)=01H;                     /* INIT 74LS245 VIA PORT C */
  6   2          END IEEE$488$INIT;

  7   1          IEEE$488$TEST: PROCEDURE PUBLIC;     /* OUTPUTS FROM 0 TO 255 CONSECUTIVELY */
  8   2           DEC I BYTE;                         /* TO IEEE 488 BUS */
  9   2           DO I=0 TO 255H;
 10   3             OUTPUT (60H)=NOT I;               /* INVERT DUE TO IEEE 488 NEGITIVE LOGIC */
 11   3           END;
 12   2          END IEEE$488$TEST;

 13   1         END IEEE$488$MODULE;


MODULE INFORMATION:

     CODE AREA SIZE     = 0028H     40D
     VARIABLE AREA SIZE = 0001H      1D
     MAXIMUM STACK SIZE = 0002H      2D
     21 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE DACOMODULE
OBJECT MODULE PLACED IN :F1:DACMOD.OBJ
COMPILER INVOKED BY:  PLM80 :F1:DACMOD.SRC DEBUG PRINT(:F1:DACMOD.LST) DATE(14/01/85) PAGELENGTH(59)


                /*************************************************************************/
                /*                    DATA ACQUISITION & LOGGING MODULE                  */
                /*    DISPLAYES BIPOLAR OUTPUT IN mVOLTS (+/- 5V FULL SCALE)             */
                /*************************************************************************/
  1             D$AC0$MODULE:
                DO;
  2     1        DECLARE DEC LITERALLY 'DECLARE';
  3     1        DEC LIT LITERALLY 'LITERALLY';
  4     1        DEC CLEAR$DISP LIT '01H';
  5     1        DEC RETURN$HOME LIT '02H';
  6     1        DEC CURSOR$OFF LIT '0CH';
  7     1        DEC HOME$2 LIT '0C0H';
  8     1        DEC (I,B,N) BYTE;
  9     1        DEC TEMP$LO ADDRESS;
 10     1        DEC TEMP$MID ADDRESS;
 11     1        DEC TEMP$HI ADDRESS;
 12     1        DEC DATA$HI(32) BYTE; /* TEMP STORAGE OF LOGGED DATA FOR ONE SCAN (HI BYTE) */
 13     1        DEC DATA$LO(32) BYTE; /*                          "                (LO BYTE) */
 14     1        DEC AD$DATA ADDRESS;
 15     1        DEC CHAR(16) BYTE EXTERNAL;
 16     1        DEC CHAN BYTE;                              /* CHANNEL COUNTER FOR MUX */
 17     1        DEC NO$OF$SCANS ADDRESS EXTERNAL;
 18     1        DEC NO$OF$CHANS BYTE EXTERNAL;
 19     1        DEC SCAN$RATE BYTE EXTERNAL;
 20     1        DEC BLANK (*) BYTE DATA('      ');
 21     1        DEC MES$6 (*) BYTE DATA('EXIT');
 22     1        DEC MES$16 (*) BYTE DATA('ENTRY');
 23     1        DEC MES$19 (*) BYTE DATA(7FH,'    ',7EH);
 24     1        DEC MES$23 (*) BYTE DATA ('CHAN.#');
 25     1        DEC MES$24 (*) BYTE DATA (' ( ) ');
 26     1        DEC FLAG1 BYTE EXTERNAL;
 27     1        DEC RST$55$FLAG BYTE EXTERNAL;
 28     1        DEC DISP BYTE EXTERNAL;
 29     1        DEC KEY BYTE EXTERNAL;
 30     1        DEC POSITION BYTE EXTERNAL;
 31     1        DEC REG$A BYTE EXTERNAL;
 32     1        DEC REG$B BYTE EXTERNAL;
 33     1        DEC BUFFER (16384) BYTE PUBLIC AT (8000H);
 34     1        DEC BUFF$POINT ADDRESS;
 35     1        DEC TAPE$MODE$FLAG BYTE EXTERNAL;
 36     1        DEC IEEE$MODE$FLAG BYTE EXTERNAL;
 37     1        DEC TRIG$TYPE BYTE EXTERNAL;

 38     1        INST$OUT: PROCEDURE(INSTRUCTION) EXTERNAL;
 39     2          DEC INSTRUCTION BYTE;
 40     2        END INST$OUT;

 41     1        C$OUT: PROCEDURE (CHAR) EXTERNAL;
 42     2          DEC CHAR BYTE;
 43     2        END C$OUT;
```

```
44   1       TEXT$OUT: PROCEDURE(P4,SIZE)EXTERNAL;
45   2         DEC P4 ADDRESS;
46   2         DEC SIZE BYTE;
47   2       END TEXT$OUT;
                 .
48   1       MENU: PROCEDURE(P1,P2,P3)EXTERNAL;
49   2         DEC (P1,P2,P3)ADDRESS;
50   2       END MENU;

51   1       BIN$TO$ASCII:PROCEDURE (READING,SIZE) EXTERNAL;
52   2         DEC SIZE BYTE;
53   2         DEC READING ADDRESS;
54   2       END BIN$TO$ASCII;

55   1       CHAN$DISP$SELECT:PROCEDURE EXTERNAL;
56   2       END CHAN$DISP$SELECT;

57   1       TAPE$BLOCK$DUMP:PROCEDURE EXTERNAL;
58   2       END TAPE$BLOCK$DUMP;

59   1       RTC$DATA:PROCEDURE (A) EXTERNAL;
60   2         DEC A BYTE;
61   2       END RTC$DATA;

62   1       D$AC0$INIT: PROCEDURE PUBLIC;
63   2        OUTPUT(57H)=80H;                    /* INIT. DATA ACQUISITION BOARD 8255 */
64   2       END D$AC0$INIT;

65   1       MUX$35: PROCEDURE;                   /* INC. CHAN PLUS 35 MICRO SEC DELAY*/
66   2         DEC A BYTE;
67   2           CHAN = CHAN+1;                   /* INC. CHANNEL COUNTER */
68   2           OUTPUT(55H)=CHAN;                /* INC. MUX */
69   2           A=A+1;                           /* DUMMY TASK */
70   2           A=A+1;                           /*      "       */
71   2       END MUX$35;

72   1       LOG$DISP:PROCEDURE (DISP,TEST);      /* DISPLAYS LOG MESSAGE */
73   2        DEC (DISP,TEST) BYTE;
74   2        CALL INST$OUT(CLEAR$DISP);
75   2        CALL INST$OUT(CURSOR$OFF);
76   2        CALL TEXT$OUT(.MES$23,LENGTH(MES$23));/* CHAN.# */
77   2        CALL BIN$TO$ASCII(DISP,2);
78   2        CALL TEXT$OUT(.CHAR,2);
79   2        DO CASE TEST;
80   3         CALL MENU(.BLANK,.BLANK,.MES$6);    /*              EXIT */
81   3         CALL MENU(.BLANK,.MES$16,.MES$6);   /*       ENTRY  EXIT */
82   3        END;
83   2       END LOG$DISP;

84   1       LOG$DATA$NORMALIZE:PROCEDURE;
85   2         CALL INST$OUT(089H);
86   2         TEMP$MID = (((DATA$HI(DISP-1) AND 0FH)*1000)/8)*5;     /* NOLMALIZE DATA FOR 10V I/P */
87   2         TEMP$HI = (((SHR(DATA$HI(DISP-1),4))*1000)/8)*5;
88   2         TEMP$LO = (((SHR(DATA$LO(DISP-1),4))*1000)/8)*5;
89   2         AD$DATA = TEMP$HI + SHR(TEMP$MID,4)+ SHR(TEMP$LO,8);   /* AD$DATA RANGE 0-10000 mV */
```

```
 90   2          DISABLE;
 91   2          IF (SHL(LOW(TEMP$MID),4) + LOW(TEMP$LO)) >80H THEN
 92   2            AD$DATA = AD$DATA + 1;              /* ROUND UP LSB */
 93   2          IF IEEE$MODE$FLAG=1 THEN              /* IF IEEE MODE SELECTED THEN O/P TO IEEE 488 BUS */
 94   2           DO;
 95   3            ENABLE;
 96   3            B=0H;
 97   3            DO WHILE B  <> 20H;                 /* WAIT FOR NRFD=1 AND DRB=0 FROM IEEE BUS*/
 98   4              B=(INPUT(061H) AND 028H);         /* INDICATING IEEE LISTNER READY FOR DATA */
 99   4              IF KEY=2 THEN B=20H;
101   4            END;
102   3            OUTPUT(60H)=NOT(HIGH(AD$DATA));     /* OUTPUT HI-BYTE TO IEEE */
103   3            B=0H;
104   3            DO WHILE B  <> 20H;                 /* WAIT FOR NRFD=1 AND DRB=0 FROM IEEE BUS*/
105   4              B=(INPUT(061H) AND 028H);
106   4              IF KEY=2 THEN B=20H;
108   4            END;
109   3            OUTPUT(60H)=NOT(LOW(AD$DATA));      /* OUTPUT LO-BYTE TO IEEE */
110   3            DISABLE;
111   3          END;
112   2          IF AD$DATA >= 5000 THEN               /* O/P LOGGED VALUE (BI-POLAR) OF SELLECTED */
113   2            DO:                                 /* CHANNEL (IN mV,PLUSS SIGN) TO DISPLAY */
114   3              CALL C$OUT('+');
115   3              AD$DATA = AD$DATA - 5000;
116   3            END;
              ELSE
117   2            DO;
118   3              CALL C$OUT('-');
119   3              AD$DATA = 5000 - AD$DATA;
120   3            END;
121   2          CALL BIN$TO$ASCII(AD$DATA,4);
122   2          ENABLE;
123   2          CALL TEXT$OUT(.CHAR,4);
124   2          CALL C$OUT('m');
125   2          CALL C$OUT('V');
126   2        END LOG$DATA$NORMALIZE;

127   1        LOG$SCAN:PROCEDURE PUBLIC;
128   2         KEY=0FH;
129   2         BUFF$POINT=0;
130   2         OUTPUT(55H)=0;                         /* SELECT MUX CHAN #1*/
131   2         CHAN=0;                                /* RESET CHANNEL COUNTER */
132   2         CALL LOG$DISP(DISP,0);
133   2         REG$A = SCAN$RATE AND 0FH;             /* SET SCAN INTERUPT RATE */
134   2         DISABLE;
135   2         REG$B = 42H;                           /* ENABLE PIE FROM REAL-TIME CLOCK */
136   2         CALL RTC$DATA(0);                      /* GET DATE AND TIME */
137   2         DO N=0 TO (NO$OF$SCANS - 1);
138   3          IF KEY<>2 THEN
139   3           DO;
140   4            RST$55$FLAG=0;
141   4            DO WHILE RST$55$FLAG=0;               /* WAIT FOR RST5.5 INTERUPT */
142   5             ENABLE;
143   5            END;
144   4            DISABLE;
145   4             DO I=0 TO (NO$OF$CHANS - 1);
```

```
146   5             OUTPUT (50H)=0;                    /* START A/D CONVERSION (12 BIT) */
147   5             CALL MUX$35;
148   5             DATA$HI(I) = INPUT(50H);           /* I/P TWO BYTES OF DATA */
149   5             DATA$LO(I) = INPUT(51H);           /* FROM A/D CONVERTER. */
150   5             BUFFER(BUFF$POINT)=DATA$HI(I);     /* STORE TWO BYTES OF DATA */
151   5             BUFFER(BUFF$POINT+1)=DATA$LO(I);   /* IN "BUFFER". */
152   5             BUFF$POINT=BUFF$POINT+2;           /* INCREMENT BUFFER POINTER */
153   5            END;
154   4           CHAN = 0;                            /* RESET CHANNEL COUNTER */
155   4           OUTPUT(55H)=0;                       /* RESET MUX TO CHANNEL #1 */
156   4           ENABLE;
157   4           CALL LOG$DATA$NORMALIZE;             /* DISPLAY (AND O/P) SELECTED CHAN. */
158   4          END;
159   3        END;
160   2        REG$B = 02H;                            /* DISABLE PIE */
161   2        DISABLE;
162   2        IF (TAPE$MODE$FLAG=0) AND (KEY<>2) THEN
163   2         CALL TAPE$BLOCK$DUMP;                  /* DUMP BUFFER TO TAPE */
164   2        RST$55$FLAG=0;
165   2        IF TRIG$TYPE='A' THEN
166   2         REG$B=22H;                             /* RE-ENABLE ALARM */
167   2       END LOG$SCAN;

168   1       D$ACQ$TEST:PROCEDURE PUBLIC;
169   2         KEY,FLAG1=0;
170   2         OUTPUT(55H)=DISP-1;                    /* O/P SELECTED CHANNEL TO MUX */
171   2         CALL LOG$DISP(DISP,1);
172   2         DO WHILE KEY<>2;                       /*DO WHILE NOT EXIT*/
173   3          DISABLE;
174   3          OUTPUT(50H)=0;                        /*START A/D CONVERSION (12 BIT) */
175   3          CALL MUX$35;
176   3          ENABLE;
177   3          OUTPUT(55H)=DISP-1;                   /* O/P SELECTED CHANNEL TO MUX */
178   3          DATA$HI(DISP-1)=INPUT(50H);           /* I/P TWO BYTES OF DATA */
179   3          DATA$LO(DISP-1)=INPUT(51H);           /* FROM A/D CONVERTER */
180   3          CALL LOG$DATA$NORMALIZE;              /* DISPLAY (AND O/P) SELECTED CHANNEL */
181   3          DISABLE;
182   3          IF FLAG1=0FFH THEN
183   3           DO;
184   4             IF KEY=1 THEN CALL CHAN$DISP$SELECT;/* ENTER NEW CHANNEL TO BE DISPLAYED (DISP) */
186   4             IF KEY=12 THEN DISP=DISP-1;         /* DECREMENT DISP */
188   4             IF KEY=15 THEN DISP=DISP+1;         /* INCREMENT DISP */
190   4             OUTPUT(55H)=DISP-1;                 /* O/P NEW CHANNEL TO MUX */
191   4             CALL LOG$DISP(DISP,1);
192   4             FLAG1=0;
193   4            END;
194   3          END;
195   2       END D$ACQ$TEST;

196   1       END D$ACQ$MODULE;
```

MODULE INFORMATION:

CODE AREA SIZE = 038DH 909D

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE TAPEMODULE
OBJECT MODULE PLACED IN :F1:TAPMOD.OBJ
COMPILER INVOKED BY:  PLM80 :F1:TAPMOD.SRC DEBUG PRINT(:F1:TAPMOD.LST) DATE(16/01/85) PAGELENGTH(59)


                /*****************************************************************************/
                /*                          MFE 450 TAPE MODULE                              */
                /*                                                                           */
                /*****************************************************************************/

   1            TAPE$MODULE:
                DO;
   2    1        DECLARE DEC LITERALLY 'DECLARE';
   3    1        DEC LIT LITERALLY 'LITERALLY';
   4    1        DEC PORT$A LIT '044H';
   5    1        DEC PORT$B LIT '045H';
   6    1        DEC PORT$C LIT '046H';
   7    1        DEC PORT$CON$REG LIT '047H';
   8    1        DEC IDLE LIT '0FFH';
   9    1        DEC CE1 LIT '0FDH';                          /* TRANSPORT CONTROL */
  10    1        DEC CE2 LIT '0FBH';                          /* USART SYNC WORD */
  11    1        DEC CE3 LIT '0F7H';                          /* WRITE DATA */
  12    1        DEC CE4 LIT '0EFH';                          /* RESET */
  13    1        DEC CE5 LIT '0DFH';                          /* TRANSPORT STATUS */
  14    1        DEC CE6 LIT '0BFH';                          /* USART STATUS */
  15    1        DEC CE7 LIT '07FH';                          /* READ DATA */
  16    1        DEC WRITE$MODE LIT '08H';
  17    1        DEC FWD LIT '001H';
  18    1        DEC REV LIT '002H';
  19    1        DEC SEARCH$SPEED LIT '010H';
  20    1        DEC IPS$80 LIT '030H';
  21    1        DEC IPS$120 LIT '004H';
  22    1        DEC I ADDRESS;
  23    1        DEC (N,K,T$DATA) BYTE;
  24    1        DEC BUFFER (16384)BYTE EXTERNAL;
  25    1        DEC BUFF$SIZE ADDRESS;
  26    1        DEC BUFF$POINT ADDRESS;
  27    1        DEC SYNC LIT '0AAH';                         /* SYNC WORD=AAH */
  28    1        DEC (KEY,FLAG1) BYTE EXTERNAL;
  29    1        DEC CLEAR$DISP LIT '01H';
  30    1        DEC CURSOR$OFF LIT '0CH';
  31    1        DEC CURSOR$ON LIT '0EH';
  32    1        DEC CHAR(16) BYTE EXTERNAL;
  33    1        DEC HOME$2 LIT '0C0H';
  34    1        DEC POSITION BYTE EXTERNAL;
  35    1        DEC LAST$RECORD BYTE;
  36    1        DEC RECORD BYTE AT (2018H);
  37    1        DEC BIN$DATA ADDRESS EXTERNAL;
  38    1        DEC BLANK (*) BYTE DATA ('      ');
  39    1        DEC MES$1 (*) BYTE DATA (0F3H,'SET',0F3H);
  40    1        DEC MES$6 (*) BYTE DATA ('EXIT');
  41    1        DEC MES$16 (*) BYTE DATA ('ENTRY');
  42    1        DEC MES$35 (*) BYTE DATA ('NEXT RECORD = ');
  43    1        DEC MES$36 (*) BYTE DATA ('READ ');
  44    1        DEC MES$37 (*) BYTE DATA ('SET RECORD TO BE');
```

```
 45  1      DEC MES$38 (*) BYTE DATA ('READ: # ');
 46  1      DEC MES$39 (*) BYTE DATA (' LOAD CASSETTE!');
 47  1      DEC MES$40 (*) BYTE DATA ('  END OF TAPE!');
 48  1      DEC MES$41 (*) BYTE DATA ('WRITE PROTECTED!');
 49  1      DEC MES$42 (*) BYTE DATA ('TAPE INITILIZE');
 50  1      DEC MES$43 (*) BYTE DATA ('  IN PROGRESS');
 51  1      DEC RTC$DATE(6) BYTE EXTERNAL;
 52  1      DEC RTC$TIME(6) BYTE EXTERNAL;
 53  1      DEC NO$OF$SCANS ADDRESS EXTERNAL;
 54  1      DEC NO$OF$CHANS BYTE EXTERNAL;
 55  1      DEC SCAN$RATE BYTE EXTERNAL;
 56  1      DEC TAPE$ERROR$FLAG BYTE PUBLIC;
 57  1      DEC TAPE$MODE$FLAG BYTE EXTERNAL;
 58  1      DEC IEEE$MODE$FLAG BYTE EXTERNAL;

 59  1     INST$OUT: PROCEDURE(INSTRUCTION) EXTERNAL;
 60  2      DEC INSTRUCTION BYTE;
 61  2     END INST$OUT;

 62  1     TEXT$OUT: PROCEDURE(P4,SIZE) EXTERNAL;
 63  2      DEC SIZE BYTE;
 64  2      DEC P4 ADDRESS;
 65  2     END TEXT$OUT;

 66  1     BIN$TO$ASCII: PROCEDURE(READING,SIZE) EXTERNAL;
 67  2      DEC SIZE BYTE;
 68  2      DEC READING ADDRESS;
 69  2     END BIN$TO$ASCII;

 70  1     ASCII$TO$BIN: PROCEDURE(ASCII$ADR,SIZE) EXTERNAL;
 71  2      DEC SIZE BYTE;
 72  2      DEC ASCII$ADR ADDRESS;
 73  2     END ASCII$TO$BIN;

 74  1     MENU: PROCEDURE(P1,P2,P3) EXTERNAL;
 75  2      DEC (P1,P2,P3) ADDRESS;
 76  2     END MENU;

 77  1     KEY$IN: PROCEDURE EXTERNAL;
 78  2     END KEY$IN;

 79  1     KEY$ENTRY: PROCEDURE EXTERNAL;
 80  2     END KEY$ENTRY;

 81  1     TAPE$INIT: PROCEDURE PUBLIC;
 82  2      OUTPUT (47H)=82H;                    /* INIT MFE TAPE BOARD 8255 */
 83  2     END TAPE$INIT;

 84  1     TAPE$STROBE$OUT: PROCEDURE(INST);     /* STROBES CONTROL LINE PASSED TO "INST" */
 85  2       DEC INST BYTE;                      /* FROM HI TO LO TO HI */
 86  2       OUTPUT(PORT$C) = INST;
 87  2       OUTPUT(PORT$C) = IDLE;
 88  2     END TAPE$STROBE$OUT;

 89  1     TAPE$STROBE$IN: PROCEDURE(INST);      /* STROBES DATA INTO T$DATA */
 90  2       DEC INST BYTE:
```

```
 91   2        OUTPUT(PORT$C) = INST;                  /* CONTROL LINE PASSED TO "INST" */
 92   2        T$DATA = INPUT(PORT$B);
 93   2        OUTPUT(PORT$C) = IDLE;
 94   2      END TAPE$STROBE$IN;

 95   1      DELAY:PROCEDURE(MIL$SEC);                 /* DELAYS (1ms * VALUE OF MIL$SEC) */
 96   2        DEC MIL$SEC ADDRESS;
 97   2        DO I = 0 TO MIL$SEC;
 98   3         CALL TIME(7);/* ~1 mSEC*/
 99   3        END;
100   2      END DELAY;

101   1      PORT$A$OUT:PROCEDURE(CONTROLL,OUT$DATA);  /* O/P DATA TO USART */
102   2        DEC (CONTROLL,OUT$DATA) BYTE;
103   2        OUTPUT (PORT$A) = OUT$DATA;
104   2        CALL TAPE$STROBE$OUT(CONTROLL);
105   2      END PORT$A$OUT;

106   1      TBMT:PROCEDURE;
107   2        CALL TAPE$STROBE$IN(CE6);               /* READ USRT STATUS */
108   2        DO WHILE (T$DATA AND 01H)=00H;          /* WAIT FOR TRANSMIT BUFFER TO BE EMTY */
109   3         CALL TAPE$STROBE$IN(CE6);
110   3        END;
111   2      END TBMT;

112   1      RDA:PROCEDURE;
113   2        CALL TAPE$STROBE$IN(CE6);               /* READ USRT STATUS */
114   2        DO WHILE (T$DATA AND 080H)=00H;         /*WAIT FOR RECEIVE DATA  AVAILABLE */
115   3         CALL TAPE$STROBE$IN(CE6);
116   3        END;
117   2      END RDA;

118   1      TAPE$ERROR$MES: PROCEDURE;                /* O/P MESSAGE DEPENDING ON ERROR TYPE */
119   2       CALL INST$OUT(CLEAR$DISP);
120   2       IF (T$DATA AND 80H)=80H THEN
121   2        DO;
122   3         CALL TEXT$OUT(.MES$39,LENGTH(MES$39)); /* LOAD CASSETTE! */
123   3         CALL MENU(.MES$1,.BLANK,.BLANK);       /* SET */
124   3         RETURN;
125   3        END;
126   2       IF (T$DATA AND 20H)=20H THEN
127   2        DO;
128   3         CALL TEXT$OUT(.MES$40,LENGTH(MES$40)); /* END OF TAPE! */
129   3         RETURN;
130   3        END;
131   2       IF (T$DATA AND 01H)=01H THEN
132   2        CALL TEXT$OUT(.MES$41,LENGTH(MES$41));  /* WRITE PROTECTED! */
133   2      END TAPE$ERROR$MES;

134   1      REWIND: PROCEDURE;
135   2       CALL TAPE$STROBE$OUT(CE4);               /* RESET EOT */
136   2       CALL TAPE$STROBE$IN(CE5);                /* RESET OPTICAL */
137   2       CALL PORT$A$OUT(CE1,(SEARCH$SPEED OR REV));
138   2       DO WHILE (T$DATA AND 0A0H) = 00H;        /* TAPE LOADED,NO EOT? */
139   3        CALL TAPE$STROBE$IN(CE5);               /* TRANSPOT STATUS */
140   3       END;
```

```
141   2       CALL PORT$A$OUT(CE1,00H);                 /* STOP */
142   2       CALL DELAY(360);                          /* WAIT 360mSEC */
143   2      END REWIND;

144   1      POSITION$TAPE: PROCEDURE;                  /* POSITION TAPE AT BEGINNING OF GAP */
145   2       DO I = 1 TO K;                            /* DENOTED BY "RECORD" */
146   3        DO WHILE (T$DATA AND 0A2H) = 02H;
147   4         CALL TAPE$STROBE$IN(CE5);               /* DO WHILE GAP PRESENT */
148   4        END;
149   3        DO WHILE (T$DATA AND 0A2H) = 00H;
150   4         CALL TAPE$STROBE$IN(CE5);               /* DO WHILE DATA PRESENT */
151   4        END;
152   3       END;
153   2      END POSITION$TAPE;

154   1      GET$OFF$LEADER: PROCEDURE(READ$FLAG);
155   2       DEC READ$FLAG BYTE;
156   2       CALL TAPE$STROBE$OUT(CE4);                /* RESET EOT */
157   2       CALL TAPE$STROBE$IN(CE5);                 /* RESET OPTICAL */
158   2       DO CASE READ$FLAG;
159   3        CALL PORT$A$OUT(CE1, (SEARCH$SPEED OR FWD OR WRITE$MODE));/* START TAPE TRANSPORT FOWARD */
160   3        CALL PORT$A$OUT(CE1,FWD);
161   3       END;
162   2       DO WHILE (T$DATA AND 0E0H) = 00H;         /* CLEAR LEADER PRESENT? */
163   3        CALL TAPE$STROBE$IN(CE5);                /* TRANSPORT STATUS */
164   3       END;
165   2       DO CASE READ$FLAG;
166   3        CALL PORT$A$OUT(CE1,08H);                /* STOP */
167   3        DO;
168   4          CALL POSITION$TAPE;
169   4          CALL PORT$A$OUT(CE1,00H);              /* STOP */
170   4        END;
171   3       END;
172   2       CALL DELAY(360);                          /*WAIT 360mSEC*/
173   2       CALL PORT$A$OUT(CE1,00H);                 /* WRITE MODE OFF */
174   2      END GET$OFF$LEADER;

175   1      RECORD$DISP: PROCEDURE;
176   2       CALL INST$OUT(CLEAR$DISP);
177   2       CALL TEXT$OUT(.MES$35,LENGTH(MES$35));    /* NEXT RECORD= */
178   2       DISABLE;
179   2       CALL BIN$TO$ASCII(RECORD,2);
180   2       ENABLE;
181   2       CALL TEXT$OUT(.CHAR,2);
182   2       CALL MENU(.MES$36,.MES$16,.MES$6);        /* READ  ENTRY  EXIT */
183   2      END RECORD$DISP;

184   1      TAPE$BLOCK$DUMP: PROCEDURE PUBLIC;
185   2       IF RECORD=0 THEN
186   2        BUFF$SIZE=0FFH;
              ELSE
187   2        BUFF$SIZE=NO$OF$SCANS * NO$OF$CHANS *2;
188   2       CALL TAPE$STROBE$OUT(CE4);                /* RESET EOT & USRT */
189   2       CALL TAPE$STROBE$IN(CE5);                 /* RESET OPTICAL */
190   2       CALL TAPE$STROBE$IN(CE5);                 /* TRANSPORT STATUS */
191   2       BUFF$POINT=0;
```

```
192   2       IF (T$DATA AND 0A1H)=00H THEN           /* TAPE ERROR? */
193   2          CALL PORT$A$OUT(CE1,(FWD OR WRITE$MODE));
              ELSE
194   2          DO;
195   3          CALL PORT$A$OUT(CE1,0);              /* STOP */
196   3          TAPE$ERROR$FLAG=0FFH;
197   3          RETURN;
198   3          END;
199   2       CALL DELAY(50);
200   2       CALL TBMT;                              /* WAIT TILL TRANS BUFF = EMTY */
201   2       CALL PORT$A$OUT(CE3,00H);
202   2       CALL TBMT;
203   2       CALL PORT$A$OUT(CE3,SYNC);              /* O/P SYNC WORD */
204   2       CALL TBMT;                              /* PREAMBLE STARTS HERE */
205   2          DO I=0 TO 5;
206   3           CALL PORT$A$OUT(CE3,RTC$DATE(I));  /* O/P DATE */
207   3           CALL TBMT;
208   3          END;
209   2          DO I=0 TO 5;
210   3           CALL PORT$A$OUT(CE3,RTC$TIME(I));  /* O/P TIME */
211   3           CALL TBMT;
212   3          END;
213   2          CALL PORT$A$OUT(CE3,SCAN$RATE);     /* O/P SCAN RATE */
214   2          CALL TBMT;
215   2          CALL PORT$A$OUT(CE3,NO$OF$CHANS);   /* O/P NUMBER OF CHANNELS */
216   2          CALL TBMT;
217   2          CALL PORT$A$OUT(CE3,HIGH(NO$OF$SCANS));/* O/P NUMBER OF SCANS (HI BYTE) */
218   2          CALL TBMT;
219   2          CALL PORT$A$OUT(CE3,LOW(NO$OF$SCANS));/* O/P NUMBER OF SCANS (LO BYTE) */
220   2          CALL TBMT;
221   2          DO I=0 TO BUFF$SIZE;                /* DUMP LOGGED DATA TO TAPE FROM BUFFER */
222   3            CALL PORT$A$OUT(CE3,BUFFER(I));
223   3            CALL TAPE$STROBE$IN(CE5);         /*TRANSPORT STATUS*/
224   3            IF (T$DATA AND 0A1H)=00H THEN     /* TAPE ERROR? */
225   3               CALL TBMT;
                   ELSE
226   3             DO;
227   4              CALL PORT$A$OUT(CE1,0);         /* STOP */
228   4              TAPE$ERROR$FLAG=0FFH;
229   4              RETURN;
230   4             END;
231   3          END;
232   2       CALL PORT$A$OUT(CE3,SYNC);              /* SYNC WORD */
233   2       CALL TBMT;
234   2       CALL PORT$A$OUT(CE3,00H);
235   2       CALL TAPE$STROBE$IN(CE5);               /* TRANSPORT STATUS */
236   2       DO WHILE (T$DATA AND 0A3H)=00H;         /* WAIT FOR GAP */
237   3        CALL TAPE$STROBE$IN(CE5);              /* TRANSPORT STATUS */
238   3       END;
239   2       CALL DELAY(50);                         /* WAIT 50mS */
240   2       CALL PORT$A$OUT(CE1,80H);               /* STOP */
241   2       CALL DELAY(360);                        /* WAIT 360mSEC */
242   2       CALL PORT$A$OUT(CE1,00H);               /* WRITE MODE OFF */
243   2       RECORD=RECORD+1;
244   2     END TAPE$BLOCK$DUMP;
```

```
245   1       TAPE$BLOCK$READ: PROCEDURE;
246   2        DISABLE;
247   2        CALL TAPE$STROBE$OUT(CE4);                /* RESET EOT & USRT */
248   2        CALL TAPE$STROBE$IN(CE5);                 /* RESET OPTICAL */
249   2        CALL TAPE$STROBE$IN(CE5);                 /* TRANSPORT STATUS */
250   2        IF (T$DATA AND 0A0H)=00H THEN             /* TAPE ERROR? */
251   2          DO;
252   3           IF RECORD (= LAST$RECORD THEN          /* POSITION TAPE AT GAP IMMEDIATLY */
253   3             DO;                                  /* BEFORE RECORD TO BE READ */
254   4              K=(LAST$RECORD - RECORD)+1;
255   4              CALL PORT$A$OUT(CE1,REV);
256   4              CALL POSITION$TAPE;
257   4              DO WHILE (T$DATA AND 0A2H)=02H;
258   5               CALL TAPE$STROBE$IN(CE5);         /* DO WHILE GAP PRESENT */
259   5              END;
260   4              CALL DELAY(50);
261   4              CALL PORT$A$OUT(CE1,0H);
262   4              CALL DELAY(360);
263   4              CALL PORT$A$OUT(CE1,FWD);
264   4              DO WHILE (T$DATA AND 0A2H) = 00H;
265   5               CALL TAPE$STROBE$IN(CE5);         /* DO WHILE DATA PRESENT */
266   5              END;
267   4             END;
                   ELSE
268   3             DO;
269   4              K=RECORD -(LAST$RECORD+1);
270   4              CALL PORT$A$OUT(CE1,(FWD));
271   4              IF K > 0 THEN
272   4               CALL POSITION$TAPE;
273   4             END;
274   3          END;
                 ELSE
275   2          DO;
276   3           CALL PORT$A$OUT(CE1,0);               /* STOP */
277   3           TAPE$ERROR$FLAG=0FFH;
278   3           RETURN;
279   3          END;
280   2        CALL TAPE$STROBE$OUT(CE4);                /* RESET USRT (START SYNC WORD SEARCH) */
281   2        CALL PORT$A$OUT(CE2,SYNC);                /* SET SYNC WORD */
282   2        CALL RDA;
283   2        CALL TAPE$STROBE$IN(CE7);
284   2        DO I = 0 TO 15;                           /* READ PREAMBLE */
285   3         CALL RDA;
286   3         CALL TAPE$STROBE$IN(CE7);
287   3        BUFFER(I) = T$DATA;
288   3        END;
289   2        BUFF$POINT=16;
290   2        NO$OF$CHANS=BUFFER(13);
291   2        NO$OF$SCANS=BUFFER(14);
292   2        NO$OF$SCANS=SHL(NO$OF$SCANS,8) + BUFFER(15); /* FORM 16 BIT WORD */
293   2        DO N=0 TO NO$OF$SCANS-1;
294   3         DO I = 0 TO SHL(NO$OF$CHANS,1)-1;        /* (NO$OF$CHANS * 2) - 1 */
295   4          CALL RDA;                               /* DATA FOR EACH CHAN = 2 BYTES (12 BITS) */
296   4          CALL TAPE$STROBE$IN(CE7);               /* TRANSFER DATA FROM TAPE TO BUFFER */
297   4          BUFFER(BUFF$POINT) = T$DATA;
298   4          CALL TAPE$STROBE$IN(CE5);               /* TRANSPORT STATUS */
```

```
299   4          IF (T$DATA AND 0A0H)=00H THEN          /* TAPE ERROR? */
300   4            BUFF$POINT=BUFF$POINT+1;
                  ELSE
301   4            DO;
302   5             CALL PORT$A$OUT(CE1,0);             /* STOP */
303   5             TAPE$ERROR$FLAG=0FFH;
304   5             RETURN;
305   5            END;
306   4          END;
307   3         END;
308   2         CALL TAPE$STROBE$IN(CE5);               /* TRANSPORT STATUS */
309   2         DO WHILE (T$DATA AND 0A2H)=00H;         /* WAIT FOR GAP */
310   3          CALL TAPE$STROBE$IN(CE5);              /* TRANSPORT STATUS */
311   3         END;
312   2         CALL DELAY(50);
313   2         CALL PORT$A$OUT(CE1,00H);               /* STOP */
314   2         CALL DELAY(360);                        /*WAIT 360mSEC*/
315   2         LAST$RECORD=RECORD;                     /* UPDATE LAST$RECORD */
316   2         RECORD=RECORD+1;
317   2         CALL RECORD$DISP;
318   2         ENABLE;
319   2        END TAPE$BLOCK$READ;

320   1        RECORD$SELECT: PROCEDURE;               /* SELECT DESIRED RECORD  (1-99) */
321   2         CALL INST$OUT(CLEAR$DISP);
322   2         CALL INST$OUT(CURSOR$ON);
323   2         CALL TEXT$OUT(.MES$37,LENGTH(MES$37));  /* SET RECORD TO BE */
324   2         CALL INST$OUT(HOME$2);
325   2         CALL TEXT$OUT(.MES$38,LENGTH(MES$38));  /* READ # */
326   2         CALL BIN$TO$ASCII(RECORD,2);
327   2         CALL TEXT$OUT(.CHAR,2);
328   2         CALL INST$OUT((POSITION-2) OR 80H);
329   2         CALL KEY$IN;
330   2         CALL ASCII$TO$BIN(.CHAR,2);
331   2         RECORD=BIN$DATA;
332   2         CALL INST$OUT(CURSOR$OFF);
333   2        END RECORD$SELECT;

334   1        TAPE$READ$INIT: PROCEDURE;
335   2         CALL TAPE$STROBE$IN(CE5);               /* TRANSPORT STATUS */
336   2         IF (T$DATA AND 80H)<>0 THEN             /* TAPE LOADED? */
337   2          DO;
338   3           TAPE$ERROR$FLAG=0FFH;
339   3           RETURN;
340   3          END;
341   2         ELSE;
342   2         CALL INST$OUT(CLEAR$DISP);
343   2         CALL TEXT$OUT(.MES$42,LENGTH(MES$42));  /* TAPE INITIALIZE */
344   2         CALL INST$OUT(HOME$2);
345   2         CALL TEXT$OUT(.MES$43,LENGTH(MES$43));  /* IN PROGRESS */
346   2         CALL REWIND;                            /* TAPE IS REWOUND AND POSITIONED AT */
347   2         K=1;                                    /* BEGINNING OF GAP BEFORE RECORD #1 */
348   2         CALL GET$OFF$LEADER(1);
349   2         LAST$RECORD=0;
350   2         RECORD=01H;
351   2        END TAPE$READ$INIT;
```

```
352   1       BUFF$TO$IEEE$DUMP: PROCEDURE;
353   2        DEC TEMP BYTE;
354   2        DO I = 0 TO 15;
355   3         OUTPUT(60H)=NOT(BUFFER(I));          /* DUMP PREAMBLE TO IEEE 488 BUS */
356   3        END;
357   2        TEMP=0;
358   2        DO WHILE TEMP<>20H;
359   3         TEMP=(INPUT(61H) AND 28H);          /* WAIT FOR IEEE 488 BUS = READY FOR DATA */
360   3        END;
361   2        BUFF$POINT=16;
362   2        DO N=0 TO NO$OF$SCANS-1;
363   3         DO I = 0 TO SHL(NO$OF$CHANS,1)-1;
364   4          OUTPUT(60H)=NOT(BUFFER(BUFF$POINT)); /* DUMP DATA FROM BUFFER TO IEEE 488 BUS */
365   4          BUFF$POINT=BUFF$POINT+1;            /* INVERT DUE TO IEEE 488  NEGITIVE LOGIC */
366   4         END;
367   3        END;
368   2       END BUFF$TO$IEEE$DUMP;

369   1       TAPE$PLAY: PROCEDURE PUBLIC;           /* TAPE REPLAY MODE */
370   2        CALL TAPE$READ$INIT;
371   2        KEY=0;
372   2        FLAG1=0;
373   2        CALL RECORD$DISP;
374   2        DO WHILE (KEY<>2) AND (TAPE$ERROR$FLAG=0);/* DO WHILE NOT EXIT OR TAPE ERROR */
375   3         CALL KEY$ENTRY;
376   3         IF KEY=0 THEN
377   3          DO;
378   4           CALL TAPE$BLOCK$READ;             /* READ BLOCK OF DATA */
379   4           IF IEEE$MODE$FLAG=1 THEN
380   4            CALL BUFF$TO$IEEE$DUMP;
381   4          END;
                 ELSE
382   3          DO;
383   4           IF KEY=1 THEN CALL RECORD$SELECT;  /* SELLECT RECORD */
385   4           IF KEY=12 THEN RECORD=RECORD-1;    /* DECREMENT RECORD */
387   4           IF KEY=15 THEN RECORD=RECORD+1;    /* INCREMENT RECORD */
389   4           CALL RECORD$DISP;
390   4          END;
391   3        END;
392   2        TAPE$MODE$FLAG=0;                     /*RETURN TO RECORD MODE */
393   2       END TAPE$PLAY;

394   1       TAPE$WRITE$INIT: PROCEDURE;
395   2        CALL INST$OUT(CLEAR$DISP);
396   2        CALL TEXT$OUT(.MES$42,LENGTH(MES$42)); /* TAPE INITIALIZE */
397   2        CALL INST$OUT(HOME$2);
398   2        CALL TEXT$OUT(.MES$43,LENGTH(MES$43)); /* IN PROGRESS */
399   2        CALL REWIND;
400   2        CALL GET$OFF$LEADER(0);
401   2        RECORD=0;
402   2        CALL TAPE$BLOCK$DUMP;                 /* INITIAL DUMMY DUMP */
403   2       END TAPE$WRITE$INIT;

404   1       TAPE$ERROR:PROCEDURE PUBLIC;
405   2         CALL TAPE$ERROR$MES;
```

```
406   2        DO WHILE (T$DATA AND 80H)=0;
407   3         CALL TAPE$STROBE$IN(CE5);              /* DO WHILE CASSETTE LOADED */
408   3        END;
409   2        CALL TAPE$ERROR$MES;
410   2.       KEY=9;
411   2        DO WHILE (KEY>0) AND (TAPE$ERROR$FLAG=0FFH);
412   3         CALL TAPE$STROBE$IN(CE5);              /* TRANSPORT STATUS */
413   3         DISABLE;
414   3         IF (T$DATA AND 80H)=0 THEN             /* NEW CASSETTE LOADED? */
415   3          DO;
416   4           TAPE$ERROR$FLAG=0;
417   4           CALL TAPE$WRITE$INIT;
418   4          END;
419   3          ENABLE;
420   3          CALL DELAY(1);
421   3        END;
422   2      END TAPE$ERROR;

423   1      END TAPE$MODULE;


MODULE INFORMATION:

     CODE AREA SIZE     = 0809H   2057D
     VARIABLE AREA SIZE = 0013H     19D
     MAXIMUM STACK SIZE = 0008H      8D
     461 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

APPENDIX B

Software Version V1.3

Program listings

Note: Only those modules which differ from Version V1.2 are included.

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE TAPEMODULE
OBJECT MODULE PLACED IN :F1:TAPMOD.OBJ
COMPILER INVOKED BY:  PLM80 :F1:TAPMOD.SRC DEBUG PRINT(:F1:TAPMOD.LST) DATE(03/06/85) PAGELENGTH(59)


                /*************************************************************************/
                /*                        MFE 450 TAPE MODULE                            */
                /*                                                                       */
                /*************************************************************************/

  1             TAPE$MODULE:
                DO;
  2    1         DECLARE DEC LITERALLY 'DECLARE';
  3    1         DEC LIT LITERALLY 'LITERALLY';
  4    1         DEC PORT$A LIT '044H';
  5    1         DEC PORT$B LIT '045H';
  6    1         DEC PORT$C LIT '046H';
  7    1         DEC PORT$CON$REG LIT '047H';
  8    1         DEC IDLE LIT '0FFH';
  9    1         DEC CE1 LIT '0FDH';                         /* TRANSPORT CONTROL */
 10    1         DEC CE2 LIT '0FBH';                         /* USART SYNC WORD */
 11    1         DEC CE3 LIT '0F7H';                         /* WRITE DATA */
 12    1         DEC CE4 LIT '0EFH';                         /* RESET */
 13    1         DEC CE5 LIT '0DFH';                         /* TRANSPORT STATUS */
 14    1         DEC CE6 LIT '0BFH';                         /* USART STATUS */
 15    1         DEC CE7 LIT '07FH';                         /* READ DATA */
 16    1         DEC WRITE$MODE LIT '08H';
 17    1         DEC FWD LIT '001H';
 18    1         DEC REV LIT '002H';
 19    1         DEC SEARCH$SPEED LIT '010H';
 20    1         DEC IPS$80 LIT '030H';
 21    1         DEC IPS$120 LIT '004H';
 22    1         DEC I ADDRESS;
 23    1         DEC (N,K,T$DATA) BYTE;
 24    1         DEC BUFFER (16384)BYTE EXTERNAL;
 25    1         DEC BUFF$SIZE ADDRESS;
 26    1         DEC BUFF$POINT ADDRESS;
 27    1         DEC SYNC LIT '0AAH';                        /* SYNC WORD=AAH */
 28    1         DEC (KEY,FLAG1) BYTE EXTERNAL;
 29    1         DEC CLEAR$DISP LIT '01H';
 30    1         DEC CURSOR$OFF LIT '0CH';
 31    1         DEC CURSOR$ON LIT '0EH';
 32    1         DEC CHAR(16) BYTE EXTERNAL;
 33    1         DEC HOME$2 LIT '0C0H';
 34    1         DEC POSITION BYTE EXTERNAL;
 35    1         DEC LAST$RECORD BYTE;
 36    1         DEC RECORD BYTE AT (2018H);
 37    1         DEC BIN$DATA ADDRESS EXTERNAL;
 38    1         DEC BLANK (*) BYTE DATA ('      ');
 39    1         DEC MES$1 (*) BYTE DATA (0F3H,'SET ',0F3H);
 40    1         DEC MES$6 (*) BYTE DATA ('EXIT');
 41    1         DEC MES$16 (*) BYTE DATA ('ENTRY');
 42    1         DEC MES$35 (*) BYTE DATA ('NEXT RECORD = ');
 43    1         DEC MES$36 (*) BYTE DATA ('READ ');
 44    1         DEC MES$37 (*) BYTE DATA ('SET RECORD TO BE');
```

```
45   1        DEC MES$38 (*) BYTE DATA ('READ: # ');
46   1        DEC MES$39 (*) BYTE DATA (' LOAD CASSETTE!');
47   1        DEC MES$40 (*) BYTE DATA ('  END OF TAPE!');
48   1        DEC MES$41 (*) BYTE DATA ('WRITE PROTECTED!');
49   1        DEC MES$42 (*) BYTE DATA ('READ INITILIZE');
50   1        DEC MES$43 (*) BYTE DATA ('  IN PROGRESS');
51   1        DEC MES$44 (*) BYTE DATA ('TAPE BLOCK READ');
52   1        DEC MES$45 (*) BYTE DATA ('TAPE BLOCK WRITE');
53   1        DEC MES$46 (*) BYTE DATA (' IEEE TRANSFER');
54   1        DEC MES$47 (*) BYTE DATA ('WRITE INITIALIZE');
55   1        DEC RTC$DATE(6) BYTE EXTERNAL;
56   1        DEC RTC$TIME(6) BYTE EXTERNAL;
57   1        DEC NO$OF$SCANS ADDRESS EXTERNAL;
58   1        DEC NO$OF$CHANS BYTE EXTERNAL;
59   1        DEC SCAN$RATE BYTE EXTERNAL;
60   1        DEC TAPE$ERROR$FLAG BYTE PUBLIC;
61   1        DEC TAPE$MODE$FLAG BYTE EXTERNAL;
62   1        DEC IEEE$MODE$FLAG BYTE EXTERNAL;

63   1       INST$OUT: PROCEDURE(INSTRUCTION) EXTERNAL;
64   2        DEC INSTRUCTION BYTE;
65   2       END INST$OUT;

66   1       TEXT$OUT: PROCEDURE(P4,SIZE) EXTERNAL;
67   2        DEC SIZE BYTE;
68   2        DEC P4 ADDRESS;
69   2       END TEXT$OUT;

70   1       BIN$TO$ASCII: PROCEDURE(READING,SIZE) EXTERNAL;
71   2        DEC SIZE BYTE;
72   2        DEC READING ADDRESS;
73   2       END BIN$TO$ASCII;

74   1       ASCII$TO$BIN: PROCEDURE(ASCII$ADR,SIZE) EXTERNAL;
75   2        DEC SIZE BYTE;
76   2        DEC ASCII$ADR ADDRESS;
77   2       END ASCII$TO$BIN;

78   1       MENU: PROCEDURE(P1,P2,P3) EXTERNAL;
79   2        DEC (P1,P2,P3) ADDRESS;
80   2       END MENU;

81   1       KEY$IN: PROCEDURE EXTERNAL;
82   2       END KEY$IN;

83   1       KEY$ENTRY: PROCEDURE EXTERNAL;
84   2       END KEY$ENTRY;

85   1       TAPE$INIT: PROCEDURE PUBLIC;
86   2        OUTPUT (47H)=82H;                          /* INIT MFE TAPE BOARD 8255 */
87   2       END TAPE$INIT;

88   1       TAPE$STROBE$OUT: PROCEDURE(INST);          /* STROBES CONTROL LINE PASSED TO "INST" */
89   2         DEC INST BYTE;                           /* FROM HI TO LO TO HI */
90   2         OUTPUT(PORT$C) = INST;
91   2         OUTPUT(PORT$C) = IDLE;
```

```
 92   2      END TAPE$STROBE$OUT;

 93   1      TAPE$STROBE$IN: PROCEDURE(INST);          /* STROBES DATA INTO T$DATA */
 94   2        DEC INST BYTE;
 95   2        OUTPUT(PORT$C) = INST;                  /* CONTROL LINE PASSED TO "INST" */
 96   2        T$DATA = INPUT(PORT$B);
 97   2        OUTPUT(PORT$C) = IDLE;
 98   2      END TAPE$STROBE$IN;

 99   1      DELAY:PROCEDURE(MIL$SEC);                 /* DELAYS (1ms * VALUE OF MIL$SEC) */
100   2        DEC MIL$SEC ADDRESS;
101   2        DO I = 0 TO MIL$SEC;
102   3         CALL TIME(7);/* ~1 mSEC*/
103   3        END;
104   2      END DELAY;

105   1      PORT$A$OUT:PROCEDURE(CONTROLL,OUT$DATA);  /* O/P DATA TO USART */
106   2        DEC (CONTROLL,OUT$DATA) BYTE;
107   2        OUTPUT (PORT$A) = OUT$DATA;
108   2        CALL TAPE$STROBE$OUT(CONTROLL);
109   2      END PORT$A$OUT;

110   1      TBMT:PROCEDURE;
111   2        CALL TAPE$STROBE$IN(CE6);               /* READ USRT STATUS */
112   2        DO WHILE (T$DATA AND 01H)=00H;          /* WAIT FOR TRANSMIT BUFFER TO BE EMTY */
113   3         CALL TAPE$STROBE$IN(CE6);
114   3        END;
115   2      END TBMT;

116   1      RDA:PROCEDURE;
117   2        CALL TAPE$STROBE$IN(CE6);               /* READ USRT STATUS */
118   2        DO WHILE (T$DATA AND 080H)=00H;         /*WAIT FOR RECEIVE DATA  AVAILABLE */
119   3         CALL TAPE$STROBE$IN(CE6);
120   3        END;
121   2      END RDA;

122   1      TAPE$ERROR$MES: PROCEDURE;                /* O/P MESSAGE DEPENDING ON ERROR TYPE */
123   2       CALL INST$OUT(CLEAR$DISP);
124   2       IF (T$DATA AND 80H)=80H THEN
125   2        DO;
126   3         CALL TEXT$OUT(.MES$39,LENGTH(MES$39)); /* LOAD CASSETTE! */
127   3         CALL MENU(.MES$1,.BLANK,.BLANK);       /* SET */
128   3         RETURN;
129   3        END;
130   2       IF (T$DATA AND 20H)=20H THEN
131   2        DO;
132   3         CALL TEXT$OUT(.MES$40,LENGTH(MES$40)); /* END OF TAPE! */
133   3         RETURN;
134   3        END;
135   2       IF (T$DATA AND 01H)=01H THEN
136   2        CALL TEXT$OUT(.MES$41,LENGTH(MES$41));  /* WRITE PROTECTED! */
137   2      END TAPE$ERROR$MES;

138   1      REWIND: PROCEDURE;
139   2       CALL TAPE$STROBE$OUT(CE4);               /* RESET EOT */
140   2       CALL TAPE$STROBE$IN(CE5);                /* RESET OPTICAL */
```

```
141   2       CALL PORT$A$OUT(CE1,(SEARCH$SPEED OR REV));
142   2       DO WHILE (T$DATA AND 0A0H) = 00H;          /* TAPE LOADED,NO EOT? */
143   3         CALL TAPE$STROBE$IN(CE5);                /* TRANSPOT STATUS */
144   3       END;
145   2       CALL PORT$A$OUT(CE1,00H);                  /* STOP */
146   2       CALL DELAY(360);                           /* WAIT 360mSEC */
147   2     END REWIND;

148   1     POSITION$TAPE: PROCEDURE;                    /* POSITION TAPE AT BEGINNING OF GAP */
149   2       DO I = 1 TO K;                             /* DENOTED BY "RECORD" */
150   3         DO WHILE (T$DATA AND 0A2H) = 02H;
151   4           CALL TAPE$STROBE$IN(CE5);              /* DO WHILE GAP PRESENT */
152   4         END;
153   3         DO WHILE (T$DATA AND 0A2H) = 00H;
154   4           CALL TAPE$STROBE$IN(CE5);              /* DO WHILE DATA PRESENT */
155   4         END;
156   3       END;
157   2     END POSITION$TAPE;

158   1     GET$OFF$LEADER: PROCEDURE(READ$FLAG);
159   2       DEC READ$FLAG BYTE;
160   2       CALL TAPE$STROBE$OUT(CE4);                 /* RESET EOT */
161   2       CALL TAPE$STROBE$IN(CE5);                  /* RESET OPTICAL */
162   2       DO CASE READ$FLAG;
163   3         CALL PORT$A$OUT(CE1, (SEARCH$SPEED OR FWD OR WRITE$MODE));/* START TAPE TRANSPORT FOWARD */
164   3         CALL PORT$A$OUT(CE1,FWD);
165   3       END;
166   2       DO WHILE (T$DATA AND 0E0H) = 00H;          /* CLEAR LEADER PRESENT? */
167   3         CALL TAPE$STROBE$IN(CE5);                /* TRANSPORT STATUS */
168   3       END;
169   2       DO CASE READ$FLAG;
170   3         CALL PORT$A$OUT(CE1,08H);                /* STOP */
171   3         DO;
172   4           CALL POSITION$TAPE;
173   4           CALL PORT$A$OUT(CE1,00H);              /* STOP */
174   4         END;
175   3       END;
176   2       CALL DELAY(360);                           /*WAIT 360mSEC*/
177   2       CALL PORT$A$OUT(CE1,00H);                  /* WRITE MODE OFF */
178   2     END GET$OFF$LEADER;

179   1     RECORD$DISP: PROCEDURE;
180   2       CALL INST$OUT(CLEAR$DISP);
181   2       CALL TEXT$OUT(.MES$35,LENGTH(MES$35));     /* NEXT RECORD= */
182   2       DISABLE;
183   2       CALL BIN$TO$ASCII(RECORD,2);
184   2       ENABLE;
185   2       CALL TEXT$OUT(.CHAR,2);
186   2       CALL MENU(.MES$36,.MES$16,.MES$6);         /* READ  ENTRY  EXIT */
187   2     END RECORD$DISP;

188   1     TAPE$BLOCK$DUMP: PROCEDURE PUBLIC;
189   2       IF RECORD=0 THEN
190   2         BUFF$SIZE=0FFH;
              ELSE
191   2         BUFF$SIZE=NO$OF$SCANS * NO$OF$CHANS *2;
```

```
192   2       CALL INST$OUT(CLEAR$DISP);
193   2       CALL TEXT$OUT(.MES$45,LENGTH(MES$45));  /* TAPE BLOCK WRITE */
194   2       CALL INST$OUT(HOME$2);
195   2       CALL TEXT$OUT(.MES$43,LENGTH(MES$43));  /* IN PROGRESS */
196   2       CALL TAPE$STROBE$OUT(CE4);               /* RESET EOT & USRT */
197   2       CALL TAPE$STROBE$IN(CE5);                /* RESET OPTICAL */
198   2       CALL TAPE$STROBE$IN(CE5);                /* TRANSPORT STATUS */
199   2       BUFF$POINT=0;
200   2       IF (T$DATA AND 0A1H)=00H THEN            /* TAPE ERROR? */
201   2          CALL PORT$A$OUT(CE1,(FWD OR WRITE$MODE));
                ELSE
202   2          DO;
203   3          CALL PORT$A$OUT(CE1,0);               /* STOP */
204   3          TAPE$ERROR$FLAG=0FFH;
205   3          RETURN;
206   3          END;
207   2       CALL DELAY(50);
208   2       CALL TBMT;                               /* WAIT TILL TRANS BUFF = EMTY */
209   2       CALL PORT$A$OUT(CE3,00H);
210   2       CALL TBMT;
211   2       CALL PORT$A$OUT(CE3,SYNC);               /* O/P SYNC WORD */
212   2       CALL TBMT;                               /* PREAMBLE STARTS HERE */
213   2          DO I=0 TO 5;
214   3           CALL PORT$A$OUT(CE3,RTC$DATE(I));    /* O/P DATE */
215   3           CALL TBMT;
216   3          END;
217   2          DO I=0 TO 5;
218   3           CALL PORT$A$OUT(CE3,RTC$TIME(I));    /* O/P TIME */
219   3           CALL TBMT;
220   3          END;
221   2          CALL PORT$A$OUT(CE3,SCAN$RATE);       /* O/P SCAN RATE */
222   2          CALL TBMT;
223   2          CALL PORT$A$OUT(CE3,NO$OF$CHANS);     /* O/P NUMBER OF CHANNELS */
224   2          CALL TBMT;
225   2          CALL PORT$A$OUT(CE3,HIGH(NO$OF$SCANS));/* O/P NUMBER OF SCANS (HI BYTE) */
226   2          CALL TBMT;
227   2          CALL PORT$A$OUT(CE3,LOW(NO$OF$SCANS));/* O/P NUMBER OF SCANS (LO BYTE) */
228   2          CALL TBMT;
229   2          DO I=0 TO BUFF$SIZE;                  /* DUMP LOGGED DATA TO TAPE FROM BUFFER */
230   3            CALL PORT$A$OUT(CE3,BUFFER(I));
231   3            CALL TAPE$STROBE$IN(CE5);           /*TRANSPORT STATUS*/
232   3            IF (T$DATA AND 0A1H)=00H THEN       /* TAPE ERROR? */
233   3               CALL TBMT;
                   ELSE
234   3             DO;
235   4              CALL PORT$A$OUT(CE1,0);           /* STOP */
236   4              TAPE$ERROR$FLAG=0FFH;
237   4              RETURN;
238   4             END;
239   3          END;
240   2       CALL PORT$A$OUT(CE3,SYNC);               /* SYNC WORD */
241   2       CALL TBMT;
242   2       CALL PORT$A$OUT(CE3,00H);
243   2       CALL TAPE$STROBE$IN(CE5);                /* TRANSPORT STATUS */
244   2       DO WHILE (T$DATA AND 0A3H)=00H;          /* WAIT FOR GAP */
245   3        CALL TAPE$STROBE$IN(CE5);               /* TRANSPORT STATUS */
```

```
246   3       END;
247   2       CALL DELAY(50);                         /* WAIT 50mS */
248   2       CALL PORT$A$OUT(CE1,80H);               /* STOP */
249   2       CALL DELAY(360);                        /* WAIT 360mSEC */
250   2       CALL PORT$A$OUT(CE1,00H);               /* WRITE MODE OFF */
251   2       RECORD=RECORD+1;
252   2      END TAPE$BLOCK$DUMP;

253   1      TAPE$BLOCK$READ: PROCEDURE;
254   2       DISABLE;
255   2       CALL INST$OUT(CLEAR$DISP);
256   2       CALL TEXT$OUT(.MES$44,LENGTH(MES$44));  /* TAPE BLOCK READ */
257   2       CALL INST$OUT(HOME$2);
258   2       CALL TEXT$OUT(.MES$43,LENGTH(MES$43));  /* IN PROGRESS */
259   2       CALL TAPE$STROBE$OUT(CE4);              /* RESET EOT & USRT */
260   2       CALL TAPE$STROBE$IN(CE5);               /* RESET OPTICAL */
261   2       CALL TAPE$STROBE$IN(CE5);               /* TRANSPORT STATUS */
262   2       IF (T$DATA AND 0A0H)=00H THEN           /* TAPE ERROR? */
263   2         DO;
264   3          IF RECORD <= LAST$RECORD THEN        /* POSITION TAPE AT GAP IMMEDIATLY */
265   3            DO;                                /* BEFORE RECORD TO BE READ */
266   4             K=(LAST$RECORD - RECORD)+1;
267   4             CALL PORT$A$OUT(CE1,REV);
268   4             CALL POSITION$TAPE;
269   4             DO WHILE (T$DATA AND 0A2H)=02H;
270   5              CALL TAPE$STROBE$IN(CE5);       /* DO WHILE GAP PRESENT */
271   5             END;
272   4             CALL DELAY(50);
273   4             CALL PORT$A$OUT(CE1,0H);
274   4             CALL DELAY(360);
275   4             CALL PORT$A$OUT(CE1,FWD);
276   4             DO WHILE (T$DATA AND 0A2H) = 00H;
277   5              CALL TAPE$STROBE$IN(CE5);       /* DO WHILE DATA PRESENT */
278   5             END;
279   4            END;
              ELSE
280   3            DO;
281   4             K=RECORD -(LAST$RECORD+1);
282   4             CALL PORT$A$OUT(CE1,(FWD));
283   4             IF K > 0 THEN
284   4              CALL POSITION$TAPE;
285   4            END;
286   3         END;
              ELSE
287   2         DO;
288   3          CALL PORT$A$OUT(CE1,0);             /* STOP */
289   3          TAPE$ERROR$FLAG=0FFH;
290   3          RETURN;
291   3         END;
292   2       CALL TAPE$STROBE$OUT(CE4);              /* RESET USRT (START SYNC WORD SEARCH) */
293   2       CALL PORT$A$OUT(CE2,SYNC);              /* SET SYNC WORD */
294   2       CALL RDA;
295   2       CALL TAPE$STROBE$IN(CE7);
296   2       DO I = 0 TO 15;                         /* READ PREAMBLE */
297   3        CALL RDA;
298   3        CALL TAPE$STROBE$IN(CE7);
```

```
299   3        BUFFER(I) = T$DATA;
300   3        END;
301   2        BUFF$POINT=16;
302   2        NO$OF$CHANS=BUFFER(13);
303   2        NO$OF$SCANS=BUFFER(14);
304   2        NO$OF$SCANS=SHL(NO$OF$SCANS,8) + BUFFER(15); /* FORM 16 BIT WORD */
305   2        DO N=0 TO NO$OF$SCANS-1;
306   3         DO I = 0 TO SHL(NO$OF$CHANS,1)-1;        /* (NO$OF$CHANS * 2) - 1 */
307   4          CALL RDA;                               /* DATA FOR EACH CHAN = 2 BYTES (12 BITS) */
308   4          CALL TAPE$STROBE$IN(CE7);               /* TRANSFER DATA FROM TAPE TO BUFFER */
309   4          BUFFER(BUFF$POINT) = T$DATA;
310   4          CALL TAPE$STROBE$IN(CE5);               /* TRANSPORT STATUS */
311   4          IF (T$DATA AND 0A0H)=00H THEN           /* TAPE ERROR? */
312   4            BUFF$POINT=BUFF$POINT+1;
                  ELSE
313   4            DO;
314   5             CALL PORT$A$OUT(CE1,0);              /* STOP */
315   5             TAPE$ERROR$FLAG=0FFH;
316   5             RETURN;
317   5            END;
318   4         END;
319   3        END;
320   2        CALL TAPE$STROBE$IN(CE5);                /* TRANSPORT STATUS */
321   2        DO WHILE (T$DATA AND 0A2H)=00H;          /* WAIT FOR GAP */
322   3         CALL TAPE$STROBE$IN(CE5);               /* TRANSPORT STATUS */
323   3        END;
324   2        CALL DELAY(50);
325   2        CALL PORT$A$OUT(CE1,00H);                /* STOP */
326   2        CALL DELAY(360);                         /*WAIT 360mSEC*/
327   2        LAST$RECORD=RECORD;                      /* UPDATE LAST$RECORD */
328   2        RECORD=RECORD+1;
329   2        CALL RECORD$DISP;
330   2        ENABLE;
331   2       END TAPE$BLOCK$READ;

332   1       RECORD$SELECT: PROCEDURE;                 /* SELECT DESIRED RECORD  (1-99) */
333   2        CALL INST$OUT(CLEAR$DISP);
334   2        CALL INST$OUT(CURSOR$ON);
335   2        CALL TEXT$OUT(.MES$37,LENGTH(MES$37));   /* SET RECORD TO BE */
336   2        CALL INST$OUT(HOME$2);
337   2        CALL TEXT$OUT(.MES$38,LENGTH(MES$38));   /* READ # */
338   2        CALL BIN$TO$ASCII(RECORD,2);
339   2        CALL TEXT$OUT(.CHAR,2);
340   2        CALL INST$OUT((POSITION-2) OR 80H);
341   2        CALL KEY$IN;
342   2        CALL ASCII$TO$BIN(.CHAR,2);
343   2        RECORD=BIN$DATA;
344   2        CALL INST$OUT(CURSOR$OFF);
345   2       END RECORD$SELECT;

346   1       TAPE$READ$INIT: PROCEDURE;
347   2        CALL TAPE$STROBE$IN(CE5);                /* TRANSPORT STATUS */
348   2        IF (T$DATA AND 80H)<>0 THEN              /* TAPE LOADED? */
349   2         DO;
350   3          TAPE$ERROR$FLAG=0FFH;
351   3          RETURN;
```

```
352   3         END;
353   2        ELSE;
354   2        CALL INST$OUT(CLEAR$DISP);
355   2        CALL TEXT$OUT(.MES$42,LENGTH(MES$42));  /* TAPE INITIALIZE */
356   2        CALL INST$OUT(HOME$2);
357   2        CALL TEXT$OUT(.MES$43,LENGTH(MES$43));  /* IN PROGRESS */
358   2        CALL REWIND;                            /* TAPE IS REWOUND AND POSITIONED AT */
359   2        K=1;                                    /* BEGINNING OF GAP BEFORE RECORD #1 */
360   2        CALL GET$OFF$LEADER(1);
361   2        LAST$RECORD=0;
362   2        RECORD=01H;
363   2       END TAPE$READ$INIT;

364   1       BUFF$TO$IEEE$DUMP: PROCEDURE;
365   2        DEC TEMP BYTE;
366   2        CALL INST$OUT(CLEAR$DISP);
367   2        CALL TEXT$OUT(.MES$46,LENGTH(MES$46));  /* IEEE TRANSFER */
368   2        CALL INST$OUT(HOME$2);
369   2        CALL TEXT$OUT(.MES$43,LENGTH(MES$43));  /* IN PROGRESS */
370   2        DO I = 0 TO 15;
371   3         OUTPUT(60H)=NOT(BUFFER(I));            /* DUMP PREAMBLE TO IEEE 488 BUS */
372   3        END;
373   2        TEMP=0;
374   2        DO WHILE TEMP<>20H;
375   3         TEMP=(INPUT(61H) AND 28H);             /* WAIT FOR IEEE 488 BUS = READY FOR DATA */
376   3        END;
377   2        BUFF$POINT=16;
378   2        DO N=0 TO NO$OF$SCANS-1;
379   3         DO I = 0 TO SHL(NO$OF$CHANS,1)-1;
380   4          OUTPUT(60H)=NOT(BUFFER(BUFF$POINT));  /* DUMP DATA FROM BUFFER TO IEEE 488 BUS */
381   4          BUFF$POINT=BUFF$POINT+1;              /* INVERT DUE TO IEEE 488  NEGITIVE LOGIC */
382   4         END;
383   3        END;
384   2       END BUFF$TO$IEEE$DUMP;

385   1       TAPE$PLAY: PROCEDURE PUBLIC;            /* TAPE REPLAY MODE */
386   2        CALL TAPE$READ$INIT;
387   2        KEY=0;
388   2        FLAG1=0;
389   2        CALL RECORD$DISP;
390   2        DO WHILE (KEY<>2) AND (TAPE$ERROR$FLAG=0);/* DO WHILE NOT EXIT OR TAPE ERROR */
391   3         CALL KEY$ENTRY;
392   3         IF KEY=0 THEN
393   3          DO;
394   4           CALL TAPE$BLOCK$READ;                /* READ BLOCK OF DATA */
395   4           IF IEEE$MODE$FLAG=1 THEN
396   4            CALL BUFF$TO$IEEE$DUMP;
397   4          END;
                 ELSE
398   3          DO;
399   4           IF KEY=1 THEN CALL RECORD$SELECT;    /* SELLECT RECORD */
401   4           IF KEY=12 THEN RECORD=RECORD-1;      /* DECREMENT RECORD */
403   4           IF KEY=15 THEN RECORD=RECORD+1;      /* INCREMENT RECORD */
405   4           CALL RECORD$DISP;
406   4          END;
407   3        END;
```

```
 408   2       TAPE$MODE$FLAG=0;                       /*RETURN TO RECORD MODE */
 409   2      END TAPE$PLAY;

 410   1      TAPE$WRITE$INIT: PROCEDURE;
 411   2       CALL INST$OUT(CLEAR$DISP);
 412   2       CALL TEXT$OUT(.MES$47,LENGTH(MES$47));  /* WRITE INITIALIZE */
 413   2       CALL INST$OUT(HOME$2);
 414   2       CALL TEXT$OUT(.MES$43,LENGTH(MES$43));  /* IN PROGRESS */
 415   2       CALL REWIND;
 416   2       CALL GET$OFF$LEADER(0);
 417   2       RECORD=0;
 418   2       CALL TAPE$BLOCK$DUMP;                   /* INITIAL DUMMY DUMP */
 419   2      END TAPE$WRITE$INIT;

 420   1      TAPE$ERROR:PROCEDURE PUBLIC;
 421   2        CALL TAPE$ERROR$MES;
 422   2        DO WHILE (T$DATA AND 80H)=0;
 423   3         CALL TAPE$STROBE$IN(CE5);             /* DO WHILE CASSETTE LOADED */
 424   3        END;
 425   2        CALL TAPE$ERROR$MES;
 426   2        KEY=9;
 427   2        DO WHILE (KEY>0) AND (TAPE$ERROR$FLAG=0FFH);
 428   3         CALL TAPE$STROBE$IN(CE5);             /* TRANSPORT STATUS */
 429   3         DISABLE;
 430   3         IF (T$DATA AND 80H)=0 THEN            /* NEW CASSETTE LOADED? */
 431   3          DO;
 432   4           TAPE$ERROR$FLAG=0;
 433   4           CALL TAPE$WRITE$INIT;
 434   4          END;
 435   3          ENABLE;
 436   3          CALL DELAY(1);
 437   3        END;
 438   2      END TAPE$ERROR;

 439   1      END TAPE$MODULE;


MODULE INFORMATION:

     CODE AREA SIZE     = 0894H   2196D
     VARIABLE AREA SIZE = 0013H     19D
     MAXIMUM STACK SIZE = 0008H      8D
     477 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE DACQMODULE
OBJECT MODULE PLACED IN :F1:DACMOD.OBJ
COMPILER INVOKED BY:  PLM80 :F1:DACMOD.SRC DEBUG PRINT(:F1:DACMOD.LST) DATE(05/06/85) PAGELENGTH(59)


                /*************************************************************************/
                /*                    DATA ACQUISITION & LOGGING MODULE                  */
                /*    DISPLAYES BIPOLAR OUTPUT IN mVOLTS (+/- 5V FULL SCALE)             */
                /*************************************************************************/
  1             D$ACQ$MODULE:
                DO;
  2    1         DECLARE DEC LITERALLY 'DECLARE';
  3    1         DEC LIT LITERALLY 'LITERALLY';
  4    1         DEC CLEAR$DISP LIT '01H';
  5    1         DEC RETURN$HOME LIT '02H';
  6    1         DEC CURSOR$OFF LIT '0CH';
  7    1         DEC HOME$2 LIT '0C0H';
  8    1         DEC (I,B,N) BYTE;
  9    1         DEC TEMP$LO ADDRESS;
 10    1         DEC TEMP$MID ADDRESS;
 11    1         DEC TEMP$HI ADDRESS;
 12    1         DEC DATA$HI(32) BYTE; /* TEMP STORAGE OF LOGGED DATA FOR ONE SCAN (HI BYTE) */
 13    1         DEC DATA$LO(32) BYTE; /*                          "                (LO BYTE) */
 14    1         DEC AD$DATA ADDRESS;
 15    1         DEC CHAR(16) BYTE EXTERNAL;
 16    1         DEC CHAN BYTE;                              /* CHANNEL COUNTER FOR MUX */
 17    1         DEC NO$OF$SCANS ADDRESS EXTERNAL;
 18    1         DEC NO$OF$CHANS BYTE EXTERNAL;
 19    1         DEC SCAN$RATE BYTE EXTERNAL;
 20    1         DEC BLANK (*) BYTE DATA('      ');
 21    1         DEC MES$6 (*) BYTE DATA('EXIT');
 22    1         DEC MES$16 (*) BYTE DATA('ENTRY');
 23    1         DEC MES$19 (*) BYTE DATA(7FH,'    ',7EH);
 24    1         DEC MES$23 (*) BYTE DATA ('CHAN.#');
 25    1         DEC MES$24 (*) BYTE DATA (' ( ) ');
 26    1         DEC FLAG1 BYTE EXTERNAL;
 27    1         DEC RST$55$FLAG BYTE EXTERNAL;
 28    1         DEC DISP BYTE EXTERNAL;
 29    1         DEC KEY BYTE EXTERNAL;
 30    1         DEC POSITION BYTE EXTERNAL;
 31    1         DEC REG$A BYTE EXTERNAL;
 32    1         DEC REG$B BYTE EXTERNAL;
 33    1         DEC BUFFER (16384) BYTE PUBLIC AT (8000H);
 34    1         DEC BUFF$POINT ADDRESS;
 35    1         DEC TAPE$MODE$FLAG BYTE EXTERNAL;
 36    1         DEC IEEE$MODE$FLAG BYTE EXTERNAL;
 37    1         DEC TRIG$TYPE BYTE EXTERNAL;

 38    1        INST$OUT: PROCEDURE(INSTRUCTION) EXTERNAL;
 39    2          DEC INSTRUCTION BYTE;
 40    2        END INST$OUT;

 41    1        C$OUT: PROCEDURE (CHAR) EXTERNAL;
 42    2          DEC CHAR BYTE;
 43    2        END C$OUT;
```

```
44   1      TEXT$OUT: PROCEDURE(P4,SIZE)EXTERNAL;
45   2        DEC P4 ADDRESS;
46   2        DEC SIZE BYTE;
47   2      END TEXT$OUT;

48   1      MENU: PROCEDURE(P1,P2,P3)EXTERNAL;
49   2        DEC (P1,P2,P3)ADDRESS;
50   2      END MENU;

51   1      BIN$TO$ASCII:PROCEDURE (READING,SIZE) EXTERNAL;
52   2        DEC SIZE BYTE;
53   2        DEC READING ADDRESS;
54   2      END BIN$TO$ASCII;

55   1      CHAN$DISP$SELECT:PROCEDURE EXTERNAL;
56   2      END CHAN$DISP$SELECT;

57   1      TAPE$BLOCK$DUMP:PROCEDURE EXTERNAL;
58   2      END TAPE$BLOCK$DUMP;

59   1      RTC$DATA:PROCEDURE (A) EXTERNAL;
60   2        DEC A BYTE;
61   2      END RTC$DATA;

62   1      D$ACQ$INIT: PROCEDURE PUBLIC;
63   2       OUTPUT(57H)=80H;                    /* INIT. DATA ACQUISITION BOARD 8255 */
64   2      END D$ACQ$INIT;

65   1      MUX$35: PROCEDURE;                  /* INC. CHAN PLUS 35 MICRO SEC DELAY*/
66   2        DEC A BYTE;
67   2          CHAN = CHAN+1;                  /* INC. CHANNEL COUNTER */
68   2          OUTPUT(55H)=CHAN;               /* INC. MUX */
69   2          A=A+1;                          /* DUMMY TASK */
70   2          A=A+1;                          /*      "       */
71   2      END MUX$35;

72   1      LOG$DISP:PROCEDURE (DISP,TEST);     /* DISPLAYS LOG MESAGE */
73   2       DEC (DISP,TEST) BYTE;
74   2       CALL INST$OUT(CLEAR$DISP);
75   2       CALL INST$OUT(CURSOR$OFF);
76   2       CALL TEXT$OUT(.MES$23,LENGTH(MES$23));/* CHAN.# */
77   2       CALL BIN$TO$ASCII(DISP,2);
78   2       CALL TEXT$OUT(.CHAR,2);
79   2       DO CASE TEST;
80   3         CALL MENU(.BLANK,.BLANK,.MES$6);  /*              EXIT */
81   3         CALL MENU(.BLANK,.MES$16,.MES$6); /*      ENTRY   EXIT */
82   3       END;
83   2      END LOG$DISP;

84   1      LOG$DATA$NORMALIZE:PROCEDURE;
85   2        CALL INST$OUT(089H);
86   2        TEMP$MID = (((DATA$HI(DISP-1) AND 0FH)*1000)/8)*5;     /* NOLMALIZE DATA FOR 10V I/P */
87   2        TEMP$HI = (((SHR(DATA$HI(DISP-1),4))*1000)/8)*5;
88   2        TEMP$LO = (((SHR(DATA$LO(DISP-1),4))*1000)/8)*5;
89   2        AD$DATA = TEMP$HI + SHR(TEMP$MID,4)+ SHR(TEMP$LO,8);   /* AD$DATA RANGE 0-10000 mV */
```

```
 90   2        DISABLE;
 91   2        IF (SHL(LOW(TEMP$MID),4) + LOW(TEMP$LO)) >80H THEN
 92   2          AD$DATA = AD$DATA + 1;              /* ROUND UP LSB */
 93   2        IF IEEE$MODE$FLAG=1 THEN              /* IF IEEE MODE SELECTED THEN O/P TO IEEE 488 BUS */
 94   2         DO;
 95   3          ENABLE;
 96   3          B=0H;
 97   3          DO WHILE B  <> 20H;                 /* WAIT FOR NRFD=1 AND DRB=0 FROM IEEE BUS*/
 98   4            B=(INPUT(061H) AND 028H);         /* INDICATING IEEE LISTNER READY FOR DATA */
 99   4            IF KEY=2 THEN B=20H;
101   4          END;
102   3          OUTPUT(60H)=NOT(HIGH(AD$DATA));     /* OUTPUT HI-BYTE TO IEEE */
103   3          B=0H;
104   3          DO WHILE B  <> 20H;                 /* WAIT FOR NRFD=1 AND DRB=0 FROM IEEE BUS*/
105   4            B=(INPUT(061H) AND 028H);
106   4            IF KEY=2 THEN B=20H;
108   4          END;
109   3          OUTPUT(60H)=NOT(LOW(AD$DATA));      /* OUTPUT LO-BYTE TO IEEE */
110   3          DISABLE;
111   3        END;
112   2        IF AD$DATA >= 5000 THEN               /* O/P LOGGED VALUE (BI-POLAR) OF SELLECTED */
113   2          DO;                                 /* CHANNEL (IN mV,PLUSS SIGN) TO DISPLAY */
114   3            CALL C$OUT('+');
115   3            AD$DATA = AD$DATA - 5000;
116   3          END;
          ELSE
117   2          DO;
118   3            CALL C$OUT('-');
119   3            AD$DATA = 5000 - AD$DATA;
120   3          END;
121   2        CALL BIN$TO$ASCII(AD$DATA,4);
122   2        ENABLE;
123   2        CALL TEXT$OUT(.CHAR,4);
124   2        CALL C$OUT('m');
125   2        CALL C$OUT('V');
126   2      END LOG$DATA$NORMALIZE;

127   1      DATA$QUALIFY: PROCEDURE;
128   2       DEC THRESHOLD LIT '61';                 /* APPROX 3mG OR 150mV */
129   2       DEC (MAX,MIN,FILT$DATA) ADDRESS;
130   2       BUFF$POINT = (DISP-1)*2;                /* INIT BUFF POINT */
131   2       AD$DATA = (BUFFER(BUFF$POINT)*16)+SHR(BUFFER(BUFF$POINT+1),4);/* RIGHT JUSTIFY DATA */
132   2       MIN,MAX,FILT$DATA = AD$DATA;            /* INIT VARIABLES */
133   2       DO I=0 TO (NO$OF$SCANS - 1);
134   3        AD$DATA = (BUFFER(BUFF$POINT)*16)+SHR(BUFFER(BUFF$POINT+1),4);/* RIGHT JUSTIFY DATA */
135   3        FILT$DATA = (AD$DATA * 3 / 10)+(FILT$DATA * 7 / 10);/* R/C FILTER CALC */
136   3        BUFF$POINT = BUFF$POINT + (NO$OF$CHANS * 2);/* INC. BUFF POINT */
137   3        IF FILT$DATA > MAX THEN MAX = FILT$DATA;
139   3        IF FILT$DATA < MIN THEN MIN = FILT$DATA;
141   3       END;
142   2       IF MAX - MIN > THRESHOLD THEN
143   2        CALL TAPE$BLOCK$DUMP;                 /* IF THRESHOLD EXCEDED THEN DATA CAN BE DUMPED */
144   2      END DATA$QUALIFY;

145   1      LOG$SCAN:PROCEDURE PUBLIC;
146   2       KEY=0FH;
```

```
147   2       BUFF$POINT=0;
148   2       OUTPUT(55H)=0;                          /* SELECT MUX CHAN #1*/
149   2       CHAN=0;                                 /* RESET CHANNEL COUNTER */
150   2       CALL LOG$DISP(DISP,0);
151   2       REG$A = SCAN$RATE AND 0FH;              /* SET SCAN INTERUPT RATE */
152   2       DISABLE;
153   2       REG$B = 42H;                            /* ENABLE PIE FROM REAL-TIME CLOCK */
154   2       CALL RTC$DATA(0);                       /* GET DATE AND TIME */
155   2       DO N=0 TO (NO$OF$SCANS - 1);
156   3        IF KEY<>2 THEN
157   3         DO;
158   4          RST$55$FLAG=0;
159   4          DO WHILE RST$55$FLAG=0;                /* WAIT FOR RST5.5 INTERUPT */
160   5           ENABLE;
161   5          END;
162   4          DISABLE;
163   4           DO I=0 TO (NO$OF$CHANS - 1);
164   5            OUTPUT (50H)=0;                     /* START A/D CONVERSION (12 BIT) */
165   5            CALL MUX$35;
166   5            DATA$HI(I) = INPUT(50H);            /* I/P TWO BYTES OF DATA */
167   5            DATA$LO(I) = INPUT(51H);            /* FROM A/D CONVERTER. */
168   5            BUFFER(BUFF$POINT)=DATA$HI(I);      /* STORE TWO BYTES OF DATA */
169   5            BUFFER(BUFF$POINT+1)=DATA$LO(I);    /* IN "BUFFER". */
170   5            BUFF$POINT=BUFF$POINT+2;            /* INCREMENT BUFFER POINTER */
171   5           END;
172   4          CHAN = 0;                             /* RESET CHANNEL COUNTER */
173   4          OUTPUT(55H)=0;                        /* RESET MUX TO CHANNEL #1 */
174   4          ENABLE;
175   4          CALL LOG$DATA$NORMALIZE;              /* DISPLAY (AND O/P) SELECTED CHAN. */
176   4         END;
177   3       END;
178   2       REG$B = 02H;                            /* DISABLE PIE */
179   2       DISABLE;
180   2       IF (TAPE$MODE$FLAG=0) AND (KEY<>2) THEN
181   2        CALL DATA$QUALIFY;                     /* IS THIS DATA ACCEPTABLE ?  */
182   2       RST$55$FLAG=0;
183   2       IF TRIG$TYPE='A' THEN
184   2        REG$B=22H;                             /* RE-ENABLE ALARM */
185   2      END LOG$SCAN;

186   1      D$ACQ$TEST:PROCEDURE PUBLIC;
187   2        KEY,FLAG1=0;
188   2        OUTPUT(55H)=DISP-1;                    /* O/P SELECTED CHANNEL TO MUX */
189   2        CALL LOG$DISP(DISP,1);
190   2        DO WHILE KEY<>2;                       /*DO WHILE NOT EXIT*/
191   3         DISABLE;
192   3         OUTPUT(50H)=0;                        /*START A/D CONVERSION (12 BIT) */
193   3         CALL MUX$35;
194   3         ENABLE;
195   3         OUTPUT(55H)=DISP-1;                   /* O/P SELECTED CHANNEL TO MUX */
196   3         DATA$HI(DISP-1)=INPUT(50H);           /* I/P TWO BYTES OF DATA */
197   3         DATA$LO(DISP-1)=INPUT(51H);           /* FROM A/D CONVERTER */
198   3         CALL LOG$DATA$NORMALIZE;              /* DISPLAY (AND O/P) SELECTED CHANNEL */
199   3         DISABLE;
200   3         IF FLAG1=0FFH THEN
201   3          DO;
```

```
 202   4            IF KEY=1 THEN CALL CHAN$DISP$SELECT;/* ENTER NEW CHANNEL TO BE DISPLAYED (DISP) */
 204   4            IF KEY=12 THEN DISP=DISP-1;         /* DECREMENT DISP */
 206   4            IF KEY=15 THEN DISP=DISP+1;         /* INCREMENT DISP */
 208   4            OUTPUT(55H)=DISP-1;                 /* O/P NEW CHANNEL TO MUX */
 209   4            CALL LOG$DISP(DISP,1);
 210   4            FLAG1=0;
 211   4          END;
 212   3        END;
 213   2      END D$ACQ$TEST;

 214   1      END D$ACQ$MODULE;


MODULE INFORMATION:

     CODE AREA SIZE     = 0468H   1128D
     VARIABLE AREA SIZE = 0057H     87D
     MAXIMUM STACK SIZE = 0008H      8D
     229 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

APPENDIX C

Software Version V1.4

Program listings

Note: Only those modules which differ from Version V1.2 are included.

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE DATALOGGER
OBJECT MODULE PLACED IN :F1:LOGMOD.OBJ
COMPILER INVOKED BY:  PLM80 :F1:LOGMOD.SRC DEBUG PRINT(:F1:LOGMOD.LST) DATE(08/08/85) PAGELENGTH(59)


                /*************************************************************************/
                /*                      DATA LOGGER MAIN MODULE                          */
                /*                                                                       */
                /*************************************************************************/
  1             DATA$LOGGER:
                DO;
  2     1        DECLARE DEC LITERALLY 'DECLARE';
  3     1        DEC LIT LITERALLY 'LITERALLY';
  4     1        DEC LEN LIT 'LENGTH';
  5     1        DEC CLEAR$DISP LIT '01H';
  6     1        DEC RETURN$HOME LIT '02H';
  7     1        DEC HOME$2 LIT '0C0H';
  8     1        DEC CURSOR$ON LIT '0EH';
  9     1        DEC CURSOR$OFF LIT '0CH';
 10     1        DEC (POSITION) BYTE EXTERNAL;
 11     1        DEC RST$55$FLAG BYTE EXTERNAL;
 12     1        DEC TAPE$ERROR$FLAG BYTE EXTERNAL;

 13     1        KEY$INIT: PROCEDURE EXTERNAL;
 14     2        END KEY$INIT;

 15     1        RTC$INIT: PROCEDURE EXTERNAL;
 16     2        END RTC$INIT;

 17     1       LCD$INIT:PROCEDURE EXTERNAL;
 18     2        END LCD$INIT;

 19     1       TAPE$INIT:PROCEDURE EXTERNAL;
 20     2       END TAPE$INIT;

 21     1       D$ACQ$INIT:PROCEDURE EXTERNAL;
 22     2       END D$ACQ$INIT;

 23     1       LOG$SCAN: PROCEDURE EXTERNAL;
 24     2       END LOG$SCAN;


                                    /*  MAIN PROGRAM  */
 25     1       CALL LCD$INIT;
 26     1       CALL KEY$INIT;
 27     1       CALL RTC$INIT;
 28     1       CALL TAPE$INIT;
 29     1       CALL D$ACQ$INIT;
 30     1       TAPE$ERROR$FLAG=0;
 31     1       RST$55$FLAG=0;
 32     1       DO WHILE 1;
 33     2        CALL LOG$SCAN;
 34     2       END;
 35     1       END DATA$LOGGER;
```

```
MODULE INFORMATION:

     CODE AREA SIZE     = 0024H     36D
     VARIABLE AREA SIZE = 0000H      0D
     MAXIMUM STACK SIZE = 0002H      2D
     50 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE REALTIMECLOCKMODULE
OBJECT MODULE PLACED IN :F1:RTCMOD.OBJ
COMPILER INVOKED BY:  PLM80 :F1:RTCMOD.SRC DEBUG PRINT(:F1:RTCMOD.LST) DATE(13/08/85) PAGELENGTH(59)
```

```
                /*************************************************************************/
                /*                    REAL-TIME CLOCK  MODULE                            */
                /*                         (MC146818)                                    */
                /*************************************************************************/
  1             REAL$TIME$CLOCK$MODULE:
                DO;
  2     1        DECLARE CLEAR$DISP LITERALLY '01H';
  3     1        DECLARE DISP$CURSOR$ON LITERALLY '0EH';
  4     1        DECLARE BLINK LITERALLY '0FH';
  5     1        DECLARE HOME$2 LITERALLY '0C0H';
  6     1        DECLARE MES$14(*) BYTE DATA ('ALARM');
  7     1        DECLARE DATE$MENU (*) BYTE DATA ('SET DATE DDMMYY');
  8     1        DECLARE TIME$MENU (*) BYTE DATA ('SET TIME HHMMSS');
  9     1        DECLARE (FLAG1,KEY) BYTE EXTERNAL;
 10     1        DECLARE RTC$DATE(6) BYTE PUBLIC;
 11     1        DECLARE RTC$TIME(6) BYTE PUBLIC;
 12     1        DECLARE (I,N) BYTE;
 13     1        DECLARE CHAR (16) BYTE EXTERNAL;
 14     1        DECLARE POSITION BYTE EXTERNAL;
 15     1        DECLARE TEMP BYTE;
 16     1        DECLARE SECONDS BYTE AT (2000H);
 17     1        DECLARE MINUTES BYTE AT (2002H);
 18     1        DECLARE HOURS BYTE AT (2004H);
 19     1        DECLARE DAY$OF$WEEK BYTE AT (2006H);
 20     1        DECLARE DAY$OF$MONTH BYTE AT (2007H);
 21     1        DECLARE MONTH BYTE PUBLIC AT (2008H);
 22     1        DECLARE YEAR BYTE AT (2009H);
 23     1        DECLARE SECONDS$ALARM BYTE AT (2001H);
 24     1        DECLARE MINUTES$ALARM BYTE AT (2003H);
 25     1        DECLARE HOURS$ALARM BYTE AT (2005H);
 26     1        DECLARE REG$A BYTE PUBLIC AT (200AH);
 27     1        DECLARE REG$B BYTE PUBLIC AT (200BH);
 28     1        DECLARE REG$C BYTE AT (200CH);
 29     1        DECLARE REG$D BYTE AT (200DH);
 30     1        DECLARE RST$55$FLAG BYTE PUBLIC;        /* CLOCK INTERRUPT FLAG */
 31     1        DECLARE SCAN$RATE BYTE EXTERNAL;

 32     1       INST$OUT: PROCEDURE (INST) EXTERNAL;
 33     2        DECLARE INST BYTE;
 34     2       END INST$OUT;

 35     1       S$MASK: PROCEDURE (MASK) EXTERNAL;
 36     2        DECLARE MASK BYTE;
 37     2       END S$MASK;

 38     1       TEXT$OUT: PROCEDURE (P4,SIZE) EXTERNAL;
 39     2        DECLARE P4 ADDRESS;
 40     2        DECLARE SIZE BYTE;
 41     2       END TEXT$OUT;
```

```
 42   1      KEY$IN: PROCEDURE EXTERNAL;
 43   2      END KEY$IN;

 44   1      SET$LOG$PARAMETERS: PROCEDURE EXTERNAL;
 45   2      END SET$LOG$PARAMETERS;

 46   1      RST$55: PROCEDURE PUBLIC;
 47   2       DECLARE C BYTE;
 48   2       C = REG$C;                                  /* CLEAR IRQ */
 49   2       RST$55$FLAG=0FFH;
 50   2       ENABLE;
 51   2      END RST$55;

 52   1      RTC$INIT: PROCEDURE PUBLIC;
 53   2        DECLARE JUMP$55 BYTE AT (02CH);
 54   2        DECLARE VECTOR$55 ADDRESS AT (02DH);      /* INITIALIZE RST 5.5 JUMP VECTOR */
 55   2        JUMP$55 = 0C3H;                           /* JUMP OP CODE */
 56   2        VECTOR$55 = .RST$55;
 57   2        REG$A = SCAN$RATE AND 0FH;                /* INIT SCAN RATE (PIE RATE) */
 58   2        REG$B = 02H;                              /* PIE DISABLE */
 59   2        CALL S$MASK(01AH); /* SET INTERRUPT MASK (ENABLE 7.5 & 5.5,RESET 7.5 F/F)  */
 60   2      END RTC$INIT;

 61   1      RTC$INIT$DATA: PROCEDURE;                    /* SETS RTC INITIAL DATA */
 62   2       DO I=0 TO 4 BY 2;
 63   3        TEMP = (CHAR(I+1) AND 0FH) OR SHL((CHAR(I) AND 0FH),4);/* CONVERT 2 ASCII CHARS. */
                                                                      /* TO PACKED BCD */
 64   3        DO CASE (I+N);
 65   4         DAY$OF$MONTH = TEMP;
 66   4         HOURS = TEMP;
 67   4         MONTH = TEMP;
 68   4         MINUTES = TEMP;
 69   4         YEAR = TEMP;
 70   4         SECONDS = TEMP;
 71   4        END;
 72   3       END;
 73   2      END RTC$INIT$DATA;

 74   1      SET$DATE$TIME: PROCEDURE PUBLIC;
 75   2       REG$B = 10000010B;                          /* SET,BCD,24 HR */
 76   2       REG$A = 00H;                                /* INITIALIZE REG A */
 77   2       DO N = 0 TO 1;
 78   3        IF YEAR=56H THEN N=0;
 80   3        DO CASE N;                                 /* ALLOWS DEFAULT TO PREVIOUS SETTING */
 81   4         DO I=0 TO 5;
 82   5          CHAR(I) = RTC$DATE(I);
 83   5         END;
 84   4         DO I=0 TO 5;
 85   5          CHAR(I) = RTC$TIME(I);
 86   5         END;
 87   4        END;
 88   3        CALL INST$OUT(CLEAR$DISP);
 89   3        CALL INST$OUT(84H);                        /* POSITION CURSOR */
 90   3        DO CASE N;
 91   4         CALL TEXT$OUT(.DATE$MENU,9);              /* SET DATE */
 92   4         CALL TEXT$OUT(.TIME$MENU,9);              /* SET TIME */
```

```
 93   4          END;
 94   3          CALL INST$OUT(HOME$2 +5);                  /* MOVE CURSOR TO LINE 2 POSITION 5 */
 95   3          DO CASE N;
 96   4            CALL TEXT$OUT(.DATE$MENU(9),6);          /* DDMMYY */
 97   4            CALL TEXT$OUT(.TIME$MENU(9),6);          /* HHMMSS */
 98   4          END;
 99   3          CALL INST$OUT((POSITION-6)OR 80H);         /* MOVE CURSOR -6 PLACES */
100   3          CALL INST$OUT(BLINK);                      /* BLINK ON */
101   3          CALL KEY$IN;
102   3          CALL RTC$INIT$DATA;
103   3          IF YEAR=56H THEN                           /* TEST FOR PASSWORD( ****56) */
104   3            CALL SET$LOG$PARAMETERS;
105   3         END;
106   2         REG$B = 00000010B;                          /* GET OUT OF CLOCK SET MODE */
107   2         CALL INST$OUT(DISP$CURSOR$ON);              /* BLINK OFF */
108   2         CALL INST$OUT(CLEAR$DISP);
109   2        END SET$DATE$TIME;

110   1        RTC$DATA:PROCEDURE PUBLIC;
111   2         DECLARE (I,N) BYTE;
112   2         DECLARE ASCII (6) BYTE;
113   2         DISABLE;
114   2         DO WHILE (REG$A AND 80H)<>0;
115   3         END;
116   2         DO N =0 TO 1;
117   3          DO I =0 TO 4 BY 2;
118   4           DO CASE (I+N);
119   5            TEMP = DAY$OF$MONTH;
120   5            TEMP = HOURS;
121   5            TEMP = MONTH;
122   5            TEMP = MINUTES;
123   5            TEMP = YEAR;
124   5            TEMP = SECONDS;
125   5           END;
126   4           ASCII(I) = SHR(TEMP,4) OR 30H;      /* CONVERTS PACKED BCD TO 2 ASCII CHARACTERS */
127   4           ASCII(I+1) = (TEMP AND 0FH) OR 30H;                    /*  "  */
128   4          END;
129   3          DO CASE N;
130   4           DO I = 0 TO 5;
131   5            RTC$DATE(I) = ASCII(I);
132   5           END;
133   4           DO I = 0 TO 5;
134   5            RTC$TIME(I) = ASCII(I);
135   5           END;
136   4          END;
137   3         END;
138   2         ENABLE;
139   2        END RTC$DATA;

140   1        END REAL$TIME$CLOCK$MODULE;
```

MODULE INFORMATION:

CODE AREA SIZE = 0348H 840D

```
     VARIABLE AREA SIZE = 0019H     25D
     MAXIMUM STACK SIZE = 0004H      4D
     156 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE SETLOGPARAMETERMODULE
OBJECT MODULE PLACED IN :F1:SLPMOD.OBJ
COMPILER INVOKED BY:  PLM80 :F1:SLPMOD.SRC DEBUG PRINT(:F1:SLPMOD.LST) DATE(23/08/85) PAGELENGTH(59)


                /*************************************************************************/
                /*                    SET LOG PARAMETER MODULE                           */
                /*                                                                       */
                /*************************************************************************/
  1             SET$LOG$PARAMETER$MODULE:
                DO;
  2    1         DECLARE DEC LITERALLY 'DECLARE';
  3    1         DEC LIT LITERALLY 'LITERALLY';
  4    1         DEC LEN LIT 'LENGTH';
  5    1         DEC CLEAR$DISP LIT '01H';
  6    1         DEC RETURN$HOME LIT '02H';
  7    1         DEC HOME$2 LIT '0C0H';
  8    1         DEC CURSOR$ON LIT '0EH';
  9    1         DEC MES$11(*) BYTE DATA ('SET CHAN. TO BE');
 10    1         DEC MES$12(*) BYTE DATA ('DISPLAYED:');
 11    1         DEC MES$18(*) BYTE DATA ('SET THRESHOLD');
 12    1         DEC MES$24(*) BYTE DATA ('[mV]');
 13    1         DEC MES$25(*) BYTE DATA ('[SCANS]');
 14    1         DEC MES$26(*) BYTE DATA ('SET PRE-TRIGGER');
 15    1         DEC CHAR (16) BYTE EXTERNAL;
 16    1         DEC POSITION BYTE EXTERNAL;
 17    1         DEC PRE$TRIG BYTE PUBLIC AT (2010H);        /* MAX 255 SCANS */
 18    1         DEC NO$OF$CHANS BYTE PUBLIC AT (2011H);     /* MAX 32 CHANNALS */
 19    1         DEC SCAN$RATE BYTE PUBLIC AT (2012H);       /* 15=500ms,14=250ms,13=125ms etc */
 20    1         DEC NO$OF$SCANS ADDRESS PUBLIC AT (2013H);  /* MAX 64K SCANS */
 21    1         DEC DISP BYTE PUBLIC AT (2015H);            /* CHANNEL TO BE DISPLAYED */
 22    1         DEC THRESHOLD ADDRESS PUBLIC AT (2016H);    /* BI-POLAR TRIGGER LEVEL IN mV */
 23    1         DEC BIN$DATA ADDRESS EXTERNAL ;

 24    1        INST$OUT: PROCEDURE (INST) EXTERNAL;
 25    2          DEC INST BYTE;
 26    2        END INST$OUT;

 27    1        C$OUT: PROCEDURE (CHAR) EXTERNAL;
 28    2          DEC CHAR BYTE;
 29    2        END C$OUT;

 30    1        TEXT$OUT: PROCEDURE(P4,SIZE) EXTERNAL;
 31    2          DEC P4 ADDRESS;
 32    2          DEC SIZE BYTE;
 33    2        END TEXT$OUT;

 34    1        BIN$TO$ASCII: PROCEDURE(READING,SIZE) EXTERNAL;
 35    2          DEC SIZE BYTE;
 36    2          DEC READING ADDRESS;
 37    2        END BIN$TO$ASCII;

 38    1        ASCII$TO$BIN: PROCEDURE(ASCII$ADR,SIZE) EXTERNAL;
 39    2          DEC ASCII$ADR ADDRESS;
 40    2          DEC SIZE BYTE;
```

```
41   2      END ASCII$TO$BIN;

42   1      KEY$ENTRY: PROCEDURE EXTERNAL;
43   2      END KEY$ENTRY;

44   1      KEY$IN: PROCEDURE EXTERNAL;
45   2       END KEY$IN;

46   1      LOG$PARAM$2: PROCEDURE;
47   2        CALL TEXT$OUT(.CHAR,2);
48   2        CALL INST$OUT((POSITION-2) OR 80H);
49   2        CALL KEY$IN;
50   2      END LOG$PARAM$2;

51   1      LOG$PARAM$4: PROCEDURE;
52   2        CALL TEXT$OUT(.CHAR,4);
53   2        CALL INST$OUT((POSITION-4) OR 80H);
54   2        CALL KEY$IN;
55   2      END LOG$PARAM$4;

56   1      MENU: PROCEDURE(P1,P2,P3) EXTERNAL;
57   2       DEC (P1,P2,P3)ADDRESS;
58   2      END MENU;

59   1      CHAN$DISP$SELECT: PROCEDURE PUBLIC;
60   2        CALL INST$OUT(CLEAR$DISP);
61   2        CALL INST$OUT(CURSOR$ON);
62   2        CALL TEXT$OUT(.MES$11,LEN(MES$11));     /* SET CHAN. TO BE */
63   2        CALL INST$OUT(HOME$2);
64   2        CALL TEXT$OUT(.MES$12,LEN(MES$12));     /* DISPLAYED */
65   2        CALL C$OUT(' ');
66   2        CALL C$OUT('#');
67   2        CALL BIN$TO$ASCII(DISP,2);
68   2        CALL LOG$PARAM$2;
69   2        CALL ASCII$TO$BIN(.CHAR,2);
70   2        DISP=BIN$DATA;
71   2      END CHAN$DISP$SELECT;

72   1      SET$LOG$PARAMETERS: PROCEDURE PUBLIC;
73   2         CALL INST$OUT(CLEAR$DISP);
74   2         CALL INST$OUT(RETURN$HOME);
75   2         CALL TEXT$OUT(.MES$18,LEN(MES$18));    /* SET THRESHOLD */
76   2         CALL INST$OUT(0CAH);
77   2         CALL TEXT$OUT(.MES$24,LEN(MES$24));    /* [mV] */
78   2         CALL INST$OUT(0C5H);
79   2         CALL BIN$TO$ASCII(THRESHOLD,4);
80   2         CALL LOG$PARAM$4;
81   2         CALL ASCII$TO$BIN(.CHAR,4);
82   2         THRESHOLD=(BIN$DATA);                   /* BI-POLAR TRIGGER LEVEL IN mV */

83   2         CALL INST$OUT(CLEAR$DISP);
84   2         CALL INST$OUT(RETURN$HOME);
85   2         CALL TEXT$OUT(.MES$26,LEN(MES$26));    /* SET PRE-TRIGGER */
86   2         CALL INST$OUT(0C8H);
87   2         CALL TEXT$OUT(.MES$25,LEN(MES$25));    /* [SCANS] */
88   2         CALL INST$OUT(0C5H);
```

```
 89   2          CALL BIN$TO$ASCII(PRE$TRIG,2);
 90   2          CALL LOG$PARAM$2;
 91   2          CALL ASCII$TO$BIN(.CHAR,2);
 92   2          PRE$TRIG=BIN$DATA;

 93   2        CALL CHAN$DISP$SELECT;                /* SET CHANNEL TO BE DISPLAYED */
 94   2      END SET$LOG$PARAMETERS;

 95   1      END SET$LOG$PARAMETER$MODULE;


MODULE INFORMATION:

     CODE AREA SIZE     = 013BH    315D
     VARIABLE AREA SIZE = 0000H      0D
     MAXIMUM STACK SIZE = 0006H      6D
     115 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE DAC0MODULE
OBJECT MODULE PLACED IN :F1:DACMOD.OBJ
COMPILER INVOKED BY:  PLM80 :F1:DACMOD.SRC DEBUG PRINT(:F1:DACMOD.LST) DATE(16/08/85) PAGELENGTH(59)


                /*************************************************************************/
                /*                  DATA ACQUISITION & LOGGING MODULE                    */
                /*    DISPLAYES BIPOLAR OUTPUT IN mVOLTS (+/- 5V FULL SCALE)             */
                /*************************************************************************/
  1             D$AC0$MODULE:
                DO;
  2   1          DECLARE DEC LITERALLY 'DECLARE';
  3   1          DEC LIT LITERALLY 'LITERALLY';
  4   1          DEC CLEAR$DISP LIT '01H';
  5   1          DEC RETURN$HOME LIT '02H';
  6   1          DEC CURSOR$OFF LIT '0CH';
  7   1          DEC HOME$2 LIT '0C0H';
  8   1          DEC (I,B,N,T) BYTE;
  9   1          DEC TEMP$LO ADDRESS;
 10   1          DEC TEMP$MID ADDRESS;
 11   1          DEC TEMP$HI ADDRESS;
 12   1          DEC DATA$HI(32) BYTE; /* TEMP STORAGE OF LOGGED DATA FOR ONE SCAN (HI BYTE) */
 13   1          DEC DATA$LO(32) BYTE; /*                          "                (LO BYTE) */
 14   1          DEC AD$DATA ADDRESS;
 15   1          DEC CHAR(16) BYTE EXTERNAL;
 16   1          DEC CHAN BYTE;                              /* CHANNEL COUNTER FOR MUX */
 17   1          DEC NO$OF$SCANS ADDRESS EXTERNAL;
 18   1          DEC NO$OF$CHANS BYTE EXTERNAL;
 19   1          DEC SCAN$RATE BYTE EXTERNAL;
 20   1          DEC BLANK (*) BYTE DATA('      ');
 21   1          DEC MES$1 (*) BYTE DATA(0F3H,'SET',0F3H);
 22   1          DEC MES$3 (*) BYTE DATA('TEST ');
 23   1          DEC MES$6 (*) BYTE DATA('EXIT');
 24   1          DEC MES$16 (*) BYTE DATA('ENTRY');
 25   1          DEC MES$19 (*) BYTE DATA(7FH,'    ',7EH);
 26   1          DEC MES$22 (*) BYTE DATA('LOG-T');
 27   1          DEC MES$23 (*) BYTE DATA ('CHAN.#');
 28   1          DEC MES$24 (*) BYTE DATA (' ( ) ');
 29   1          DEC FLAG1 BYTE EXTERNAL;
 30   1          DEC RST$55$FLAG BYTE EXTERNAL;
 31   1          DEC DISP BYTE EXTERNAL;
 32   1          DEC KEY BYTE EXTERNAL;
 33   1          DEC POSITION BYTE EXTERNAL;
 34   1          DEC REG$A BYTE EXTERNAL;
 35   1          DEC REG$B BYTE EXTERNAL;
 36   1          DEC BUFFER (16384) BYTE PUBLIC AT (8000H);
 37   1          DEC BUFF$POINT ADDRESS PUBLIC;
 38   1          DEC BUFF$SIZE ADDRESS PUBLIC;
 39   1          DEC THRESHOLD ADDRESS EXTERNAL;             /* TRIGGER THRESHOLD IN mV */
 40   1          DEC FILT$DATA ADDRESS;
 41   1          DEC LAST$BUFF$POINT ADDRESS;
 42   1          DEC DATE$MES (*) BYTE DATA ('JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC');
 43   1          DEC RTC$DATE (6) BYTE EXTERNAL;
 44   1          DEC RTC$TIME (6) BYTE EXTERNAL;
 45   1          DEC MONTH BYTE EXTERNAL;
```

```
46   1        DEC FILT$DATA$FLAG BYTE;
47   1        DEC TAPE$ERROR$FLAG BYTE EXTERNAL;
48   1        DEC PRE$TRIG BYTE EXTERNAL;

49   1       INST$OUT: PROCEDURE(INSTRUCTION) EXTERNAL;
50   2         DEC INSTRUCTION BYTE;
51   2       END INST$OUT;

52   1       C$OUT: PROCEDURE (CHAR) EXTERNAL;
53   2         DEC CHAR BYTE;
54   2       END C$OUT;

55   1       TEXT$OUT: PROCEDURE(P4,SIZE)EXTERNAL;
56   2         DEC P4 ADDRESS;
57   2         DEC SIZE BYTE;
58   2       END TEXT$OUT;

59   1       MENU: PROCEDURE(P1,P2,P3)EXTERNAL;
60   2         DEC (P1,P2,P3)ADDRESS;
61   2       END MENU;

62   1       BIN$TO$ASCII:PROCEDURE (READING,SIZE) EXTERNAL;
63   2         DEC SIZE BYTE;
64   2         DEC READING ADDRESS;
65   2       END BIN$TO$ASCII;

66   1       CHAN$DISP$SELECT:PROCEDURE EXTERNAL;
67   2       END CHAN$DISP$SELECT;

68   1       TAPE$BLOCK$DUMP:PROCEDURE EXTERNAL;
69   2       END TAPE$BLOCK$DUMP;

70   1       TAPE$ERROR: PROCEDURE EXTERNAL;
71   2       END TAPE$ERROR;

72   1       RTC$DATA:PROCEDURE (A) EXTERNAL;
73   2         DEC A BYTE;
74   2       END RTC$DATA;

75   1       SET$DATE$TIME: PROCEDURE EXTERNAL;
76   2        DEC A BYTE;
77   2       END SET$DATE$TIME;

78   1       D$AC0$INIT: PROCEDURE PUBLIC;
79   2        OUTPUT(57H)=80H;                    /* INIT. DATA ACQUISITION BOARD 8255 */
80   2       END D$AC0$INIT;

81   1       MUX$35: PROCEDURE;                   /* INC. CHAN PLUS 35 MICRO SEC DELAY*/
82   2         DEC A BYTE;
83   2           CHAN = CHAN+1;                   /* INC. CHANNEL COUNTER */
84   2           OUTPUT(55H)=CHAN;                /* INC. MUX */
85   2           A=A+1;                           /* DUMMY TASK */
86   2           A=A+1;                           /*      "       */
87   2       END MUX$35;

88   1       LOG$DISP:PROCEDURE (DISP,TEST);      /* DISPLAYS LOG MESAGE */
```

```
 89   2       DEC (DISP,TEST) BYTE;
 90   2       CALL INST$OUT(CLEAR$DISP);
 91   2       CALL INST$OUT(CURSOR$OFF);
 92   2       CALL TEXT$OUT(.MES$23,LENGTH(MES$23));/* CHAN.# */
 93   2       CALL BIN$TO$ASCII(DISP,2);
 94   2       CALL TEXT$OUT(.CHAR,2);
 95   2       DO CASE TEST;
 96   3         CALL MENU(.BLANK,.BLANK,.MES$6);      /*              EXIT */
 97   3         CALL MENU(.BLANK,.MES$16,.MES$6);     /*       ENTRY  EXIT */
 98   3       END;
 99   2     END LOG$DISP;

100   1     LOG$DATA$NORMALIZE:PROCEDURE;
101   2       CALL INST$OUT(089H);
102   2       TEMP$MID = (((DATA$HI(DISP-1) AND 0FH)*1000)/8)*5;     /* NOLMALIZE DATA FOR 10V I/P */
103   2       TEMP$HI = (((SHR(DATA$HI(DISP-1),4))*1000)/8)*5;
104   2       TEMP$LO = (((SHR(DATA$LO(DISP-1),4))*1000)/8)*5;
105   2       AD$DATA = TEMP$HI + SHR(TEMP$MID,4)+ SHR(TEMP$LO,8);   /* AD$DATA RANGE 0-10000 mV */
106   2       DISABLE;
107   2       IF (SHL(LOW(TEMP$MID),4) + LOW(TEMP$LO)) >80H THEN
108   2         AD$DATA = AD$DATA + 1;              /* ROUND UP LSB */
109   2       IF AD$DATA >= 5000 THEN               /* O/P LOGGED VALUE (BI-POLAR) OF SELLECTED */
110   2         DO;                                 /* CHANNEL (IN mV,PLUSS SIGN) TO DISPLAY */
111   3           CALL C$OUT('+');
112   3           AD$DATA = AD$DATA - 5000;
113   3         END;
              ELSE
114   2         DO;
115   3           CALL C$OUT('-');
116   3           AD$DATA = 5000 - AD$DATA;
117   3         END;
118   2       CALL BIN$TO$ASCII(AD$DATA,4);
119   2       ENABLE;
120   2       CALL TEXT$OUT(.CHAR,4);
121   2       CALL C$OUT('m');
122   2       CALL C$OUT('V');
123   2     END LOG$DATA$NORMALIZE;

124   1     D$AC0$TEST:PROCEDURE PUBLIC;
125   2       REG$B = 02H;                          /* DISABLE PIE (FROM RTC) */
126   2       KEY,FLAG1=0;
127   2       OUTPUT(55H)=DISP-1;                   /* O/P SELECTED CHANNEL TO MUX */
128   2       CALL LOG$DISP(DISP,1);
129   2       DO WHILE KEY<>2;                      /*DO WHILE NOT EXIT*/
130   3         DISABLE;
131   3         OUTPUT(50H)=0;                      /*START A/D CONVERSION (12 BIT) */
132   3         CALL MUX$35;
133   3         ENABLE;
134   3         OUTPUT(55H)=DISP-1;                 /* O/P SELECTED CHANNEL TO MUX */
135   3         DATA$HI(DISP-1)=INPUT(50H);         /* I/P TWO BYTES OF DATA */
136   3         DATA$LO(DISP-1)=INPUT(51H);         /* FROM A/D CONVERTER */
137   3         CALL LOG$DATA$NORMALIZE;            /* DISPLAY (AND O/P) SELECTED CHANNEL */
138   3         DISABLE;
139   3         IF FLAG1=0FFH THEN
140   3           DO;
141   4             IF KEY=1 THEN CALL CHAN$DISP$SELECT;/* ENTER NEW CHANNEL TO BE DISPLAYED (DISP) */
```

```
143   4            IF KEY=12 THEN DISP=DISP-1;          /* DECREMENT DISP */
145   4            IF KEY=15 THEN DISP=DISP+1;          /* INCREMENT DISP */
147   4            OUTPUT(55H)=DISP-1;                  /* O/P NEW CHANNEL TO MUX */
148   4            CALL LOG$DISP(DISP,1);
149   4            FLAG1=0;
150   4          END;
151   3        END;
152   2      END D$AC0$TEST;

153   1      THRESH$CHECK: PROCEDURE;
154   2       TEMP$HI=LAST$BUFF$POINT + ((DISP-1)*2);  /* POSITION DATA POINTER */
155   2       AD$DATA = (BUFFER(TEMP$HI)*16)+SHR(BUFFER(TEMP$HI+1),4);/* RIGHT JUSTIFY DATA */
156   2       IF FILT$DATA$FLAG=0H THEN
157   2        DO;
158   3         FILT$DATA = AD$DATA;
159   3         FILT$DATA$FLAG=0FFH;
160   3        END;
               ELSE
161   2        DO;
162   3         TEMP$LO=(FILT$DATA*9);                  /* FILTER CALC. [Y=X*0,01 + Y*0,99] */
163   3         FILT$DATA=((AD$DATA/10)+TEMP$LO+(TEMP$LO / 10))/10;/* 9% MAX ROUNDING ERROR */
164   3          IF FILT$DATA >= AD$DATA THEN
165   3           DO;
166   4            IF (FILT$DATA-AD$DATA)>T THEN
167   4             DO;
168   5              KEY=1;
169   5              RETURN;
170   5             END;
171   4           END;
                  ELSE
172   3            IF (AD$DATA-FILT$DATA)>T THEN
173   3             DO;
174   4              KEY=1;
175   4              RETURN;
176   4             END;
              LAST$BUFF$POINT=BUFF$POINT;
178   3        END;
179   2      END THRESH$CHECK;

180   1      TRIG$SCAN:PROCEDURE;
181   2       KEY=0FH;
182   2       CALL LOG$DISP(DISP,0);
183   2       DO N=0 TO ((NO$OF$SCANS - 1)- PRE$TRIG);
184   3        IF KEY<>2 THEN
185   3         DO;
186   4          RST$55$FLAG=0;
187   4          DO WHILE RST$55$FLAG=0;              /* WAIT FOR RST5.5 INTERUPT */
188   5           ENABLE;
189   5          END;
190   4          DISABLE;
191   4           DO I=0 TO (NO$OF$CHANS - 1);
192   5            OUTPUT (50H)=0;                    /* START A/D CONVERSION (12 BIT) */
193   5            CALL MUX$35;
194   5            DATA$HI(I) = INPUT(50H);           /* I/P TWO BYTES OF DATA */
195   5            DATA$LO(I) = INPUT(51H);           /* FROM A/D CONVERTER. */
196   5            BUFFER(BUFF$POINT)=DATA$HI(I);     /* STORE TWO BYTES OF DATA */
```

```
197   5            BUFFER(BUFF$POINT+1)=DATA$LO(I);     /* IN "BUFFER". */
198   5            BUFF$POINT=BUFF$POINT+2;             /* INCREMENT BUFFER POINTER */
199   5           END;
200   4         IF BUFF$POINT =BUFF$SIZE THEN
201   4           BUFF$POINT=0;
202   4           CHAN = 0;                              /* RESET CHANNEL COUNTER */
203   4           OUTPUT(55H)=0;                         /* RESET MUX TO CHANNEL #1 */
204   4           ENABLE;
205   4           CALL LOG$DATA$NORMALIZE;               /* DISPLAY (AND O/P) SELECTED CHAN. */
206   4         END;
207   3       END;
208   2       REG$B = 02H;                               /* DISABLE PIE */
209   2       DISABLE;
210   2       IF KEY <>2 THEN
211   2         CALL TAPE$BLOCK$DUMP;
212   2       RST$55$FLAG=0;
213   2      END TRIG$SCAN;

214   1      DISP$DATE$TIME: PROCEDURE;
215   2       DEC TEMP BYTE;
216   2       CALL INST$OUT(RETURN$HOME);
217   2       CALL RTC$DATA(0);
218   2       DO I=0 TO 1;
219   3        CALL C$OUT (RTC$DATE(I));               /* O/P DAY OF MONTH */
220   3       END;
221   2       CALL C$OUT (' ');
222   2       TEMP=MONTH;
223   2       IF TEMP>09H THEN                         /* CONVERT MONTH NUMBER TO */
224   2        TEMP=TEMP-6;                            /* A 3 CHARACTER WORD */
225   2       TEMP=(TEMP*3)-3;
226   2       DO I=0 TO 2;
227   3        CALL C$OUT (DATE$MES(I+TEMP));          /* O/P MONTH WORD */
228   3       END;
229   2       CALL C$OUT ('.');
230   2       CALL C$OUT (' ');
231   2       DO I=0 TO 5;
232   3       IF (I=2) OR (I=4) THEN
233   3        DO;
234   4         CALL C$OUT (':');
235   4         CALL C$OUT (RTC$TIME(I));              /* O/P TIME */
236   4        END;
             ELSE
237   3         CALL C$OUT (RTC$TIME(I));
238   3        END;
239   2      END DISP$DATE$TIME;

240   1      LOG$SCAN:PROCEDURE PUBLIC;
241   2       IF TAPE$ERROR$FLAG > 0 THEN
242   2        CALL TAPE$ERROR;                        /* TAPE ERROR FLAG SET ? */
243   2       CALL MENU(.MES$1,.MES$22,.MES$3);        /* SET LOG-T TEST */
244   2       FILT$DATA$FLAG=0H;
245   2       NO$OF$SCANS = 240;
246   2       NO$OF$CHANS = 27;  \
247   2       SCAN$RATE = 15;
248   2       BUFF$SIZE = (NO$OF$SCANS*NO$OF$CHANS*2);
249   2       T = (THRESHOLD*4)/10;                    /* CONVERTS mV TO bits[250mV=100bits] */
```

```
 250   2       KEY=0FH;
 251   2       BUFF$POINT=0;
 252   2       LAST$BUFF$POINT=0;
 253   2       OUTPUT(55H)=0;                          /* SELECT MUX CHAN #1*/
 254   2       CHAN=0;                                 /* RESET CHANNEL COUNTER */
 255   2       REG$A = SCAN$RATE AND 0FH;              /* SET SCAN INTERUPT RATE */
 256   2       DISABLE;
 257   2       REG$B = 42H;                            /* ENABLE PIE FROM REAL-TIME CLOCK */
 258   2       DO WHILE KEY >2;
 259   3         RST$55$FLAG=0;
 260   3         DO WHILE RST$55$FLAG=0;                 /* WAIT FOR RST5.5 INTERUPT */
 261   4          ENABLE;
 262   4         END;
 263   3         DISABLE;
 264   3         DO I=0 TO (NO$OF$CHANS - 1);
 265   4            OUTPUT (50H)=0;                      /* START A/D CONVERSION (12 BIT) */
 266   4            CALL MUX$35;
 267   4            DATA$HI(I) = INPUT(50H);             /* I/P TWO BYTES OF DATA */
 268   4            DATA$LO(I) = INPUT(51H);             /* FROM A/D CONVERTER. */
 269   4            BUFFER(BUFF$POINT)=DATA$HI(I);       /* STORE TWO BYTES OF DATA */
 270   4            BUFFER(BUFF$POINT+1)=DATA$LO(I);     /* IN "BUFFER". */
 271   4            BUFF$POINT=BUFF$POINT+2;             /* INCREMENT BUFFER POINTER */
 272   4         END;
 273   3         IF BUFF$POINT = BUFF$SIZE THEN
 274   3           BUFF$POINT=0;
 275   3         CHAN = 0;                               /* RESET CHANNEL COUNTER */
 276   3         OUTPUT(55H)=0;                          /* RESET MUX TO CHANNEL #1 */
 277   3         CALL THRESH$CHECK;                      /* CHECK IF THRESHOLD IS EXCEEDED */
 278   3         IF KEY <> 1 THEN
 279   3          CALL DISP$DATE$TIME;
 280   3         END;
 281   2         DO CASE (KEY);
 282   3          CALL SET$DATE$TIME;
 283   3          CALL TRIG$SCAN;
 284   3          CALL D$ACQ$TEST;
 285   3        END;
 286   2      END LOG$SCAN;

 287   1      END D$ACQ$MODULE;


MODULE INFORMATION:

     CODE AREA SIZE     = 061AH   1562D
     VARIABLE AREA SIZE = 005BH     91D
     MAXIMUM STACK SIZE = 000AH     10D
     313 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE TAPEMODULE
OBJECT MODULE PLACED IN :F1:TAPMOD.OBJ
COMPILER INVOKED BY:  PLM80 :F1:TAPMOD.SRC DEBUG PRINT(:F1:TAPMOD.LST) DATE(22/08/85) PAGELENGTH(59)


                /*************************************************************************/
                /*                        MFE 450 TAPE MODULE                            */
                /*                                                                       */
                /*************************************************************************/

  1             TAPE$MODULE:
                DO;
  2    1         DECLARE DEC LITERALLY 'DECLARE';
  3    1         DEC LIT LITERALLY 'LITERALLY';
  4    1         DEC PORT$A LIT '044H';
  5    1         DEC PORT$B LIT '045H';
  6    1         DEC PORT$C LIT '046H';
  7    1         DEC PORT$CON$REG LIT '047H';
  8    1         DEC IDLE LIT '0FFH';
  9    1         DEC CE1 LIT '0FDH';                     /* TRANSPORT CONTROL */
 10    1         DEC CE2 LIT '0FBH';                     /* USART SYNC WORD */
 11    1         DEC CE3 LIT '0F7H';                     /* WRITE DATA */
 12    1         DEC CE4 LIT '0EFH';                     /* RESET */
 13    1         DEC CE5 LIT '0DFH';                     /* TRANSPORT STATUS */
 14    1         DEC CE6 LIT '0BFH';                     /* USART STATUS */
 15    1         DEC CE7 LIT '07FH';                     /* READ DATA */
 16    1         DEC WRITE$MODE LIT '08H';
 17    1         DEC FWD LIT '001H';
 18    1         DEC REV LIT '002H';
 19    1         DEC SEARCH$SPEED LIT '010H';
 20    1         DEC IPS$80 LIT '030H';
 21    1         DEC IPS$120 LIT '004H';
 22    1         DEC I ADDRESS;
 23    1         DEC (N,K,T$DATA) BYTE;
 24    1         DEC BUFFER (16384) BYTE EXTERNAL;
 25    1         DEC BUFF$SIZE ADDRESS EXTERNAL;         /* BUFFER MEMORY NEEDED ONE LOG */
 26    1         DEC BUFF$POINT ADDRESS EXTERNAL;
 27    1         DEC SYNC LIT '0AAH';                    /* SYNC WORD=AAH */
 28    1         DEC (KEY,FLAG1) BYTE EXTERNAL;
 29    1         DEC CLEAR$DISP LIT '01H';
 30    1         DEC CURSOR$OFF LIT '0CH';
 31    1         DEC CURSOR$ON LIT '0EH';
 32    1         DEC CHAR(16) BYTE EXTERNAL;
 33    1         DEC HOME$2 LIT '0C0H';
 34    1         DEC POSITION BYTE EXTERNAL;
 35    1         DEC LAST$RECORD BYTE;
 36    1         DEC RECORD BYTE AT (2018H);
 37    1         DEC BIN$DATA ADDRESS EXTERNAL;
 38    1         DEC BLANK (*) BYTE DATA ('      ');
 39    1         DEC MES$1 (*) BYTE DATA (0F3H,'SET',0F3H);
 40    1         DEC MES$35 (*) BYTE DATA ('RECORD # ');
 41    1         DEC MES$39 (*) BYTE DATA (' LOAD CASSETTE!');
 42    1         DEC MES$40 (*) BYTE DATA ('  END OF TAPE!');
 43    1         DEC MES$41 (*) BYTE DATA ('WRITE PROTECTED!');
 44    1         DEC MES$43 (*) BYTE DATA ('  IN PROGRESS');
```

```
45   1       DEC MES$45 (*) BYTE DATA ('TAPE BLOCK WRITE');
46   1       DEC MES$47 (*) BYTE DATA ('WRITE INITIALIZE');
47   1       DEC RTC$DATE(6) BYTE EXTERNAL;
48   1       DEC RTC$TIME(6) BYTE EXTERNAL;
49   1       DEC NO$OF$SCANS ADDRESS EXTERNAL;
50   1       DEC NO$OF$CHANS BYTE EXTERNAL;
51   1       DEC SCAN$RATE BYTE EXTERNAL;
52   1       DEC TAPE$ERROR$FLAG BYTE PUBLIC;
53   1       DEC PRE$TRIG BYTE EXTERNAL;                    /* PRE-TRIGGER VALUE IN SCANS */

54   1      INST$OUT: PROCEDURE(INSTRUCTION) EXTERNAL;
55   2       DEC INSTRUCTION BYTE;
56   2      END INST$OUT;

57   1      TEXT$OUT: PROCEDURE(P4,SIZE) EXTERNAL;
58   2       DEC SIZE BYTE;
59   2       DEC P4 ADDRESS;
60   2      END TEXT$OUT;

61   1      BIN$TO$ASCII: PROCEDURE(READING,SIZE) EXTERNAL;
62   2       DEC SIZE BYTE;
63   2       DEC READING ADDRESS;
64   2      END BIN$TO$ASCII;

65   1      MENU: PROCEDURE(P1,P2,P3) EXTERNAL;
66   2       DEC (P1,P2,P3) ADDRESS;
67   2      END MENU;

68   1      TAPE$INIT: PROCEDURE PUBLIC;
69   2       OUTPUT (47H)=82H;                             /* INIT MFE TAPE BOARD 8255 */
70   2      END TAPE$INIT;

71   1      TAPE$STROBE$OUT: PROCEDURE(INST);             /* STROBES CONTROL LINE PASSED TO "INST" */
72   2        DEC INST BYTE;                              /* FROM HI TO LO TO HI */
73   2        OUTPUT(PORT$C) = INST;
74   2        OUTPUT(PORT$C) = IDLE;
75   2      END TAPE$STROBE$OUT;

76   1      TAPE$STROBE$IN: PROCEDURE(INST);               /* STROBES DATA INTO T$DATA */
77   2        DEC INST BYTE;
78   2        OUTPUT(PORT$C) = INST;                       /* CONTROL LINE PASSED TO "INST" */
79   2        T$DATA = INPUT(PORT$B);
80   2        OUTPUT(PORT$C) = IDLE;
81   2      END TAPE$STROBE$IN;

82   1      DELAY:PROCEDURE(MIL$SEC);                      /* DELAYS (1ms * VALUE OF MIL$SEC) */
83   2        DEC MIL$SEC ADDRESS;
84   2        DO I = 0 TO MIL$SEC;
85   3         CALL TIME(7);/* ~1 mSEC*/
86   3        END;
87   2      END DELAY;

88   1      PORT$A$OUT:PROCEDURE(CONTROLL,OUT$DATA);       /* O/P DATA TO USART */
89   2        DEC (CONTROLL,OUT$DATA) BYTE;
90   2        OUTPUT (PORT$A) = OUT$DATA;
91   2        CALL TAPE$STROBE$OUT(CONTROLL);
```

```
 92   2      END PORT$A$OUT;

 93   1      TBMT:PROCEDURE;
 94   2        CALL TAPE$STROBE$IN(CE6);                /* READ USRT STATUS */
 95   2        DO WHILE (T$DATA AND 01H)=00H;           /* WAIT FOR TRANSMIT BUFFER TO BE EMTY */
 96   3         CALL TAPE$STROBE$IN(CE6);
 97   3        END;
 98   2      END TBMT;

 99   1      TAPE$ERROR$MES: PROCEDURE;                 /* O/P MESSAGE DEPENDING ON ERROR TYPE */
100   2       CALL INST$OUT(CLEAR$DISP);
101   2       IF (T$DATA AND 80H)=80H THEN
102   2        DO;
103   3         CALL TEXT$OUT(.MES$39,LENGTH(MES$39)); /* LOAD CASSETTE! */
104   3         CALL MENU(.MES$1,.BLANK,.BLANK);       /* SET */
105   3         RETURN;
106   3        END;
107   2       IF (T$DATA AND 20H)=20H THEN
108   2        DO;
109   3         CALL TEXT$OUT(.MES$40,LENGTH(MES$40)); /* END OF TAPE! */
110   3         RETURN;
111   3        END;
112   2       IF (T$DATA AND 01H)=01H THEN
113   2        CALL TEXT$OUT(.MES$41,LENGTH(MES$41));  /* WRITE PROTECTED! */
114   2      END TAPE$ERROR$MES;

115   1      REWIND: PROCEDURE;
116   2       CALL TAPE$STROBE$OUT(CE4);                /* RESET EOT */
117   2       CALL TAPE$STROBE$IN(CE5);                 /* RESET OPTICAL */
118   2       CALL PORT$A$OUT(CE1,(SEARCH$SPEED OR REV));
119   2       DO WHILE (T$DATA AND 0A0H) = 00H;         /* TAPE LOADED,NO EOT? */
120   3        CALL TAPE$STROBE$IN(CE5);                /* TRANSPOT STATUS */
121   3       END;
122   2       CALL PORT$A$OUT(CE1,00H);                 /* STOP */
123   2       CALL DELAY(360);                          /* WAIT 360mSEC */
124   2      END REWIND;

125   1      GET$OFF$LEADER: PROCEDURE;
126   2       CALL TAPE$STROBE$OUT(CE4);                /* RESET EOT */
127   2       CALL TAPE$STROBE$IN(CE5);                 /* RESET OPTICAL */
128   2       CALL PORT$A$OUT(CE1, (SEARCH$SPEED OR FWD OR WRITE$MODE));/* START TAPE TRANSPORT FOWARD */
129   2       DO WHILE (T$DATA AND 0E0H) = 00H;         /* CLEAR LEADER PRESENT? */
130   3        CALL TAPE$STROBE$IN(CE5);                /* TRANSPORT STATUS */
131   3       END;
132   2       CALL PORT$A$OUT(CE1,08H);                /* STOP */
133   2       CALL DELAY(360);                          /*WAIT 360mSEC*/
134   2       CALL PORT$A$OUT(CE1,00H);                 /* WRITE MODE OFF */
135   2      END GET$OFF$LEADER;

136   1      TAPE$BLOCK$DUMP: PROCEDURE PUBLIC;
137   2       IF RECORD=0 THEN
138   2        BUFF$SIZE=0FFH;
139   2       CALL INST$OUT(CLEAR$DISP);
140   2       CALL TEXT$OUT(.MES$45,LENGTH(MES$45));  /* TAPE BLOCK WRITE */
141   2       CALL INST$OUT(0C2H);
142   2       CALL TEXT$OUT(.MES$35,LENGTH(MES$35));  /* RECORD # */
```

```
143   2       CALL BIN$TO$ASCII(RECORD,2);
144   2       CALL TEXT$OUT(.CHAR,2);
145   2       CALL TAPE$STROBE$OUT(CE4);                /* RESET EOT & USRT */
146   2       CALL TAPE$STROBE$IN(CE5);                 /* RESET OPTICAL */
147   2       CALL TAPE$STROBE$IN(CE5);                 /* TRANSPORT STATUS */
148   2       IF (T$DATA AND 0A1H)=00H THEN             /* TAPE ERROR? */
149   2          CALL PORT$A$OUT(CE1,(FWD OR WRITE$MODE));
                ELSE
150   2            DO;
151   3             CALL PORT$A$OUT(CE1,0);             /* STOP */
152   3             TAPE$ERROR$FLAG=0FFH;
153   3             RETURN;
154   3            END;
155   2       CALL DELAY(50);
156   2       CALL TBMT;                                /* WAIT TILL TRANS BUFF = EMTY */
157   2       CALL PORT$A$OUT(CE3,00H);
158   2       CALL TBMT;
159   2       CALL PORT$A$OUT(CE3,SYNC);                /* O/P SYNC WORD */
160   2       CALL TBMT;                                /* PREAMBLE STARTS HERE */
161   2          DO I=0 TO 5;
162   3           CALL PORT$A$OUT(CE3,RTC$DATE(I));     /* O/P DATE */
163   3           CALL TBMT;
164   3          END;
165   2          DO I=0 TO 5;
166   3           CALL PORT$A$OUT(CE3,RTC$TIME(I));     /* O/P TIME */
167   3           CALL TBMT;
168   3          END;
169   2          CALL PORT$A$OUT(CE3,SCAN$RATE);        /* O/P SCAN RATE */
170   2          CALL TBMT;
171   2          CALL PORT$A$OUT(CE3,NO$OF$CHANS);      /* O/P NUMBER OF CHANNELS */
172   2          CALL TBMT;
173   2          CALL PORT$A$OUT(CE3,HIGH(NO$OF$SCANS));/* O/P NUMBER OF SCANS (HI BYTE) */
174   2          CALL TBMT;
175   2          CALL PORT$A$OUT(CE3,LOW(NO$OF$SCANS));/* O/P NUMBER OF SCANS (LO BYTE) */
176   2          CALL TBMT;
177   2          DO I=0 TO BUFF$SIZE-1;                 /* DUMP LOGGED DATA TO TAPE FROM BUFFER */
178   3            CALL PORT$A$OUT(CE3,BUFFER(BUFF$POINT));
179   3            IF BUFF$POINT = (BUFF$SIZE-1) THEN
180   3              BUFF$POINT=0;
                   ELSE
181   3              BUFF$POINT=BUFF$POINT+1;
182   3            CALL TAPE$STROBE$IN(CE5);            /*TRANSPORT STATUS*/
183   3            IF (T$DATA AND 0A1H)=00H THEN        /* TAPE ERROR? */
184   3               CALL TBMT;
                   ELSE
185   3                DO;
186   4                 CALL PORT$A$OUT(CE1,0);         /* STOP */
187   4                 TAPE$ERROR$FLAG=0FFH;
188   4                 RETURN;
189   4                END;
190   3          END;
191   2       CALL PORT$A$OUT(CE3,SYNC);                /* SYNC WORD */
192   2       CALL TBMT;
193   2       CALL PORT$A$OUT(CE3,00H);
194   2       CALL TAPE$STROBE$IN(CE5);                 /* TRANSPORT STATUS */
195   2       DO WHILE (T$DATA AND 0A3H)=00H;           /* WAIT FOR GAP */
```

```
 196   3          CALL TAPE$STROBE$IN(CE5);              /* TRANSPORT STATUS */
 197   3        END;
 198   2        CALL DELAY(50);                          /* WAIT 50mS */
 199   2        CALL PORT$A$OUT(CE1,80H);                /* STOP */
 200   2        CALL DELAY(360);                         /* WAIT 360mSEC */
 201   2        CALL PORT$A$OUT(CE1,00H);                /* WRITE MODE OFF */
 202   2        RECORD=RECORD+1;
 203   2      END TAPE$BLOCK$DUMP;

 204   1      TAPE$WRITE$INIT: PROCEDURE PUBLIC;
 205   2        CALL INST$OUT(CLEAR$DISP);
 206   2        CALL TEXT$OUT(.MES$47,LENGTH(MES$47));  /* WRITE INITIALIZE */
 207   2        CALL INST$OUT(HOME$2);
 208   2        CALL TEXT$OUT(.MES$43,LENGTH(MES$43));  /* IN PROGRESS */
 209   2        CALL REWIND;
 210   2        CALL GET$OFF$LEADER;
 211   2        RECORD=0;
 212   2        CALL TAPE$BLOCK$DUMP;                    /* INITIAL DUMMY DUMP */
 213   2      END TAPE$WRITE$INIT;

 214   1      TAPE$ERROR:PROCEDURE PUBLIC;
 215   2          CALL TAPE$ERROR$MES;
 216   2          DO WHILE (T$DATA AND 80H)=0;
 217   3            CALL TAPE$STROBE$IN(CE5);            /* DO WHILE CASSETTE LOADED */
 218   3          END;
 219   2          CALL TAPE$ERROR$MES;
 220   2          KEY=9;
 221   2          DO WHILE (KEY>0) AND (TAPE$ERROR$FLAG=0FFH);
 222   3            CALL TAPE$STROBE$IN(CE5);            /* TRANSPORT STATUS */
 223   3            DISABLE;
 224   3            IF (T$DATA AND 80H)=0 THEN           /* NEW CASSETTE LOADED? */
 225   3             DO;
 226   4               TAPE$ERROR$FLAG=0;
 227   4               CALL TAPE$WRITE$INIT;
 228   4             END;
 229   3             ENABLE;
 230   3             CALL DELAY(1);
 231   3          END;
 232   2      END TAPE$ERROR;

 233   1      END TAPE$MODULE;
```

MODULE INFORMATION:

```
     CODE AREA SIZE     = 03E8H   1000D
     VARIABLE AREA SIZE = 000DH     13D
     MAXIMUM STACK SIZE = 0008H      8D
     259 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

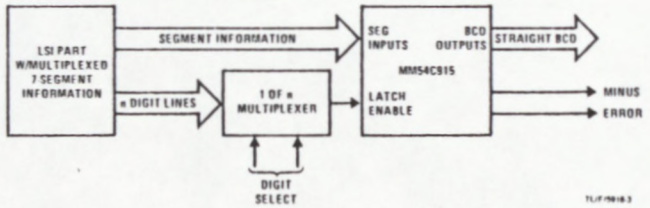
APPENDIX D

DATA SHEETS

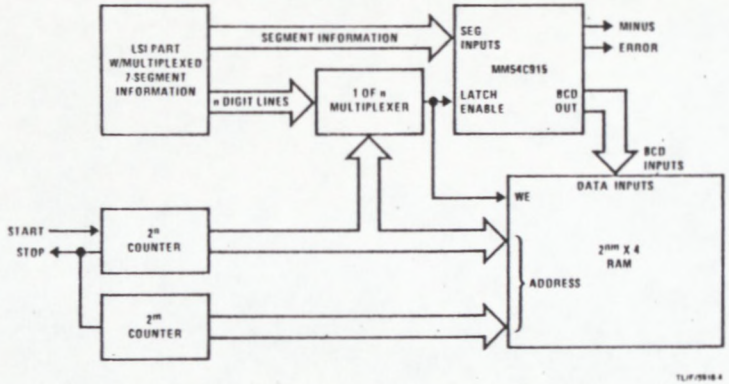
1. Keyboard encoder 74C922
2. Densitron Dot Matrix Display LMA IC24-DC
3. Real-Time Clock MC146818
4. 12-bit A/D Converter AD 574A
5. Sample and Hold AD582
6. 16 Channel Multiplexer AD7506
7. GPIB (IEEE-488) Circuit 96LS488

Typical Applications

Multiplex 7-Segment to Straight BCD



Memory Expansion from 7-Segment Outputs



National Semiconductor

MM54C922/MM74C922 16-Key Encoder
MM54C923/MM74C923 20-Key Encoder

General Description

These CMOS key encoders provide all the necessary logic to fully encode an array of SPST switches. The keyboard scan can be implemented by either an external clock or external capacitor. These encoders also have on-chip pull-up devices which permit switches with up to 50 kΩ on resistance to be used. No diodes in the switch array are needed to eliminate ghost switches. The internal debounce circuit needs only a single external capacitor and can be defeated by omitting the capacitor. A Data Available output goes to a high level when a valid keyboard entry has been made. The Data Available output returns to a low level when the entered key is released, even if another key is depressed. The Data Available will return high to indicate acceptance of the new key after a normal debounce period; this two key roll over is provided between any two switches.

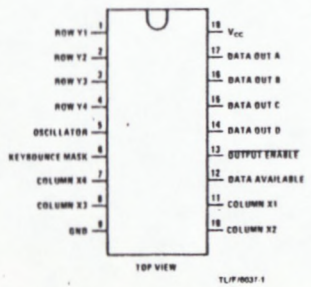
An internal register remembers the last key pressed even after the key is released. The TRI-STATE® outputs provide for easy expansion and bus operation and are LPTTL compatible.

Features

- 50 kΩ maximum switch on resistance
- On or off chip clock
- On chip row pull-up devices
- 2 key roll-over
- Keybounce elimination with single capacitor
- Last key register at outputs
- TRI-STATE outputs LPTTL compatible
- Wide supply range 3V to 15V
- Low power consumption

Connection Diagrams

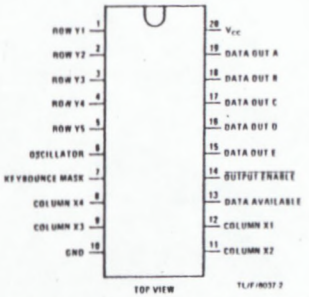
Dual-In-Line Package



Order Number MM54C922J or MM74C922J
See NS Package J18A

Order Number MM54C922N or MM74C922N
See NS Package N18A

Dual-In-Line Package



Order Number MM54C923J or MM74C923J
See NS Package J20A

Order Number MM54C923N or MM74C923N
See NS Package N20A

Absolute Maximum Ratings (Note 1)

Voltage at Any Pin	V_{CC} − 0.3V to V_{CC} + 0.3V	Package Dissipation	500 mW
Operating Temperature Range		Operating V_{CC} Range	3V to 15V
MM54C922, MM54C923	−55°C to +125°C	V_{CC}	18V
MM74C922, MM74C923	−40°C to +85°C	Lead Temperature (Soldering, 10 seconds)	300°C
Storage Temperature Range	−65°C to +150°C		

DC Electrical Characteristics Min/Max limits apply across temperature range unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
CMOS TO CMOS						
V_{T+}	Positive-Going Threshold Voltage at Osc and KBM Inputs	V_{CC} = 5V, $I_{IN} \geq 0.7$ mA	3	3.6	4.3	V
		V_{CC} = 10V, $I_{IN} \geq 1.4$ mA	6	6.8	8.6	V
		V_{CC} = 15V, $I_{IN} \geq 2.1$ mA	9	10	12.9	V
V_{T-}	Negative-Going Threshold Voltage at Osc and KBM Inputs	V_{CC} = 5V, $I_{IN} \geq 0.7$ mA	0.7	1.4	2	V
		V_{CC} = 10V, $I_{IN} \geq 1.4$ mA	1.4	3.2	4	V
		V_{CC} = 15V, $I_{IN} \geq 2.1$ mA	2.1	5	6	V
$V_{IN(1)}$	Logical "1" Input Voltage, Except Osc and KBM Inputs	V_{CC} = 5V,	3.5	4.5		V
		V_{CC} = 10V,	8	9		V
		V_{CC} = 15V,	12.5	13.5		V
$V_{IN(0)}$	Logical "0" Input Voltage, Except Osc and KBM Inputs	V_{CC} = 5V,		0.5	1.5	V
		V_{CC} = 10V,		1	2	V
		V_{CC} = 15V,		1.5	2.5	V
I_{rp}	Row Pull-Up Current at Y1, Y2, Y3, Y4 and Y5 Inputs	V_{CC} = 5V, V_{IN} = 0.1 V_{CC}		−2	−5	μA
		V_{CC} = 10V		−10	−20	μA
		V_{CC} = 15V		−22	−45	μA
$V_{OUT(1)}$	Logical "1" Output Voltage	V_{CC} = 5V, I_O = −10μA	4.5			V
		V_{CC} = 10V, I_O = −10μA	9			V
		V_{CC} = 15V, I_O = −10μA	13.5			V
$V_{OUT(0)}$	Logical "0" Output Voltage	V_{CC} = 5V, I_O = 10μA			0.5	V
		V_{CC} = 10V, I_O = 10μA			1	V
		V_{CC} = 15V, I_O = 10μA			1.5	V
R_{on}	Column "ON" Resistance at X1, X2, X3 and X4 Outputs	V_{CC} = 5V, V_O = 0.5V		500	1400	Ω
		V_{CC} = 10V, V_O = 1V		300	700	Ω
		V_{CC} = 15V, V_O = 1.5V		200	500	Ω
I_{CC}	Supply Current Osc at 0V, (one Y low)	V_{CC} = 5V,		0.55	1.1	mA
		V_{CC} = 10V		1.1	1.9	mA
		V_{CC} = 15V		1.7	2.6	mA
$I_{IN(1)}$	Logical "1" Input Current at $\overline{\text{Output Enable}}$	V_{CC} = 15V, V_{IN} = 15V		0.005	1.0	μA
$I_{IN(0)}$	Logical "0" Input Current at $\overline{\text{Output Enable}}$	V_{CC} = 15V, V_{IN} = 0V	−1.0	−0.005		μA
CMOS/LPTTL INTERFACE						
$V_{IN(1)}$	Logical "1" Input Voltage, Except Osc and KBM Inputs	54C, V_{CC} = 4.5V	V_{CC} −1.5			V
		74C, V_{CC} = 4.75V	V_{CC} −1.5			V
$V_{IN(0)}$	Logical "0" Input Voltage, Except Osc and KBM Inputs	54C, V_{CC} = 4.5V			0.8	V
		74C, V_{CC} = 4.75V			0.8	V
$V_{OUT(1)}$	Logical "1" Output Voltage	54C, V_{CC} = 4.5V, I_O = −360μA	2.4			V
		74C, V_{CC} = 4.75V, I_O = −360μA	2.4			V
$V_{OUT(0)}$	Logical "0" Output Voltage	54C, V_{CC} = 4.5V, I_O = 360μA			0.4	V
		74C, V_{CC} = 4.75V, I_O = 360μA			0.4	V

DC Electrical Characteristics (Cont'd.)

Min/Max limits apply across temperature range unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
OUTPUT DRIVE (See 54C/74C Family Characteristics Data Sheet) (Short Circuit Current)						
I_{SOURCE}	Output Source Current (P-Channel)	V_{CC} = 5V, V_{OUT} = 0V, T_A = 25°C	−1.75	−3.3		mA
I_{SOURCE}	Output Source Current (P-Channel)	V_{CC} = 10V, V_{OUT} = 0V, T_A = 25°C	−8	−15		mA
I_{SINK}	Output Sink Current (N-Channel)	V_{CC} = 5V, V_{OUT} = V_{CC}, T_A = 25°C	1.75	3.6		mA
I_{SINK}	Output Sink Current (N-Channel)	V_{CC} = 10V, V_{OUT} = V_{CC}, T_A = 25°C	8	16		mA

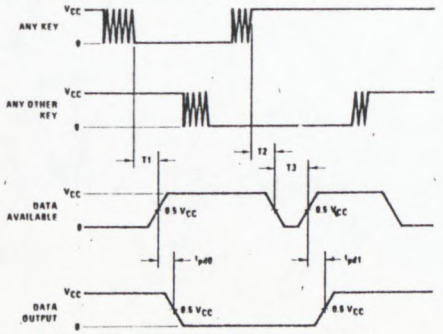
AC Electrical Characteristics T_A = 25°C, C_L = 50 pF, unless otherwise noted

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
t_{pd0},t_{pd1}	Propagation Delay Time to Logical "0" or Logical "1" from D.A.	C_L = 50 pF, *(Figure 1)*				
		V_{CC} = 5V		60	150	ns
		V_{CC} = 10V		35	80	ns
		V_{CC} = 15V		25	60	ns
t_{0H},t_{1H}	Propagation Delay Time from Logical "0" or Logical "1" into High Impedance State	R_L = 10k, C_L = 10pF *(Figure 2)*				
		V_{CC} = 5V R_L = 10k		80	200	ns
		V_{CC} = 10V C_L = 10 pF		65	150	ns
		V_{CC} = 15V		50	110	ns
t_{H0},t_{H1}	Propagation Delay Time from High Impedance State to a Logical "0" or Logical "1"	R_L = 10k, C_L = 50 pF, *(Figure 2)*				
		V_{CC} = 5V R_L = 10k		100	250	ns
		V_{CC} = 10V C_L = 50 pF		55	125	ns
		V_{CC} = 15V		40	90	ns
C_{IN}	Input Capacitance	Any Input, (Note 2)		5	7.5	pF
C_{OUT}	TRI-STATE Output Capacitance	Any Output, (Note 2)		10		pF

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. Except for "Operating Temperature Range" they are not meant to imply that the devices should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: Capacitance is guaranteed by periodic testing.

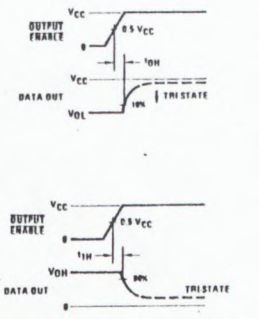
Switching Time Waveforms



T1 ≈ T2 ≈ RC, T3 ≈ 0.7 RC where R ≈ 10k and C is external capacitor at KBM input.

TL/F/6037-3

FIGURE 1

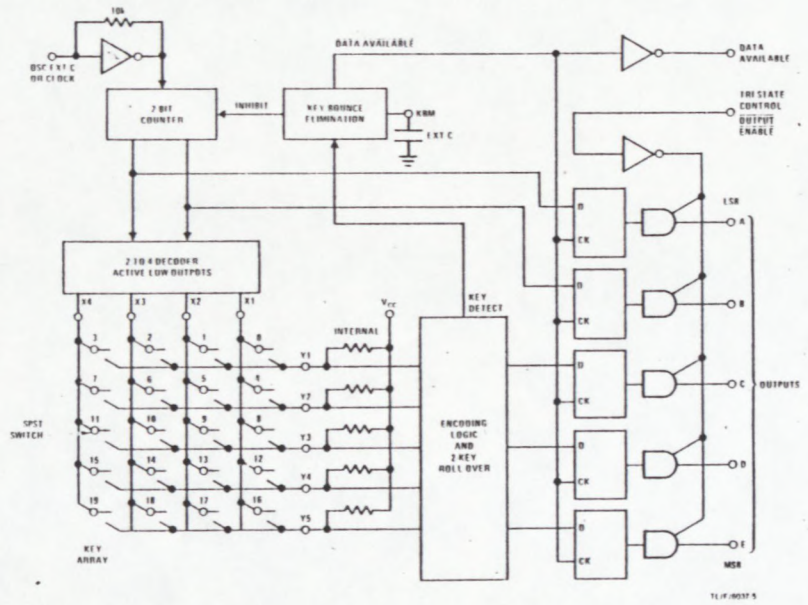


TL/F/6037-4

FIGURE 2

MM54C922/MM74C922, MM54C923/MM74C923

Block Diagram



TL/F/8037-5

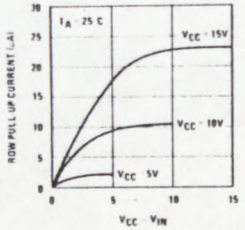
Truth Table

SWITCH POSITION		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		Y1,X1	Y1,X2	Y1,X3	Y1,X4	Y2,X1	Y2,X2	Y2,X3	Y2,X4	Y3,X1	Y3,X2	Y3,X3	Y3,X4	Y4,X1	Y4,X2	Y4,X3	Y4,X4	Y5*,X1	Y5*,X2	Y5*,X3	Y5*,X4
DATA OUTPUT	D	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	A	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	B	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
	C	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
	D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
	F*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

*Omit for MM54C922/MM74C922

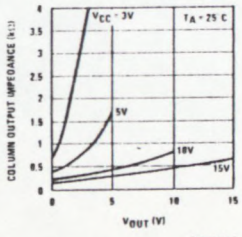
Typical Performance Characteristics

Typical I_{rp} vs V_{IN} at Any Y Input



TL/F/8037-6

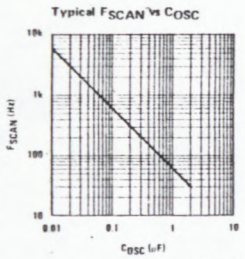
Typical R_{on} vs V_{OUT} at Any X Output



TL/F/8037-7

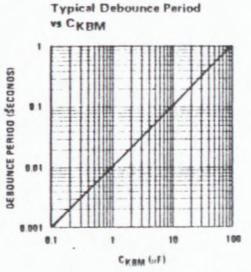
Typical Performance Characteristics (Cont'd.)

Typical F_{SCAN} vs C_{OSC}



TL/F/8037-8

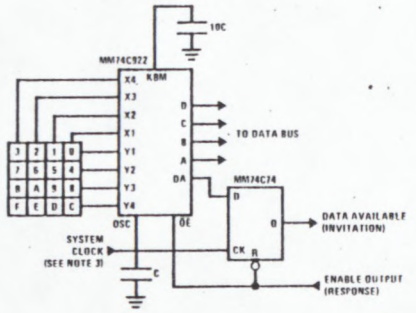
Typical Debounce Period vs C_{KBM}



TL/F/8037-9

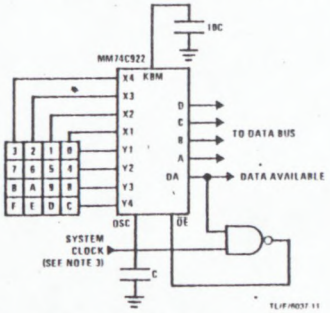
Typical Applications

Synchronous Handshake (MM74C922)



TL/F/8037-10

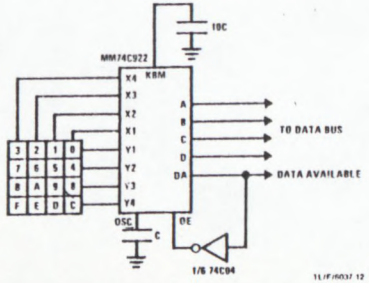
Synchronous Data Entry Onto Bus (MM74C922)



TL/F/8037-11

Outputs are enabled when valid entry is made and go into TRI-STATE when key is released.

Asynchronous Data Entry Onto Bus (MM74C922)



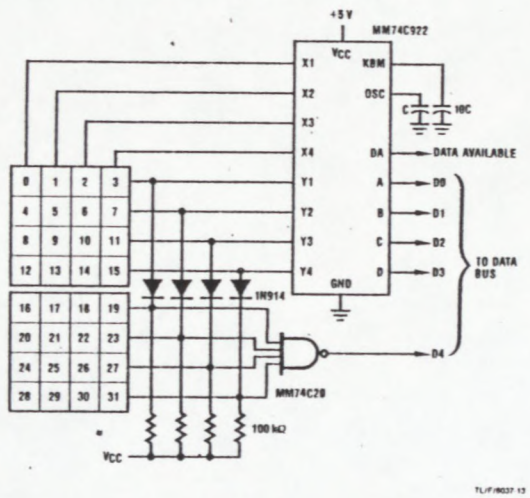
TL/F/8037-12

Outputs are in TRI-STATE until key is pressed, then data is placed on bus. When key is released, outputs return to TRI-STATE.

Note 3: The keyboard may be synchronously scanned by omitting the capacitor at osc. and driving osc. directly if the system clock rate is lower than 10 kHz.

Typical Application (Cont'd.)

Expansion to 32 Key Encoder (MM74C922)



TL/F/9037-13

Theory of Operation

The MM74C922/MM74C923 Keyboard Encoders implement all the logic necessary to interface a 16 or 20 SPST key switch matrix to a digital system. The encoder will convert a key switch closer to a 4(MM74C922) or 5(MM74C923) bit nibble. The designer can control both the keyboard scan rate and the key debounce period by altering the oscillator capacitor, C_{OSE}, and the key bounce mask capacitor, C_{MSK}. Thus, the MM74C922/MM47C923's performance can be optimized for many keyboards.

The keyboard encoders connect to a switch matrix that is 4 rows by 4 columns (MM74C922) or 5 rows by 4 columns (MM74C923). When no keys are depressed, the row inputs are pulled high by internal pull-ups and the column outputs sequentially output a logic "0". These outputs are open drain and are therefore low for 25% of the time and otherwise off. The column scan rate is controlled by the oscillator input, which consists of a Schmitt trigger oscillator, a 2-bit counter, and a 2-4-bit decoder.

When a key is depressed, key 0, for example, nothing will happen when the X1 input is off, since Y1 will remain high. When the X1 column is scanned, X1 goes low and Y1 will go low. This disables the counter and keeps X1 low. Y1 going low also initiates the key bounce circuit timing and locks out the other Y inputs. The key code to be outputted is a combination of the frozen counter value and the decoded Y inputs. Once the key bounce circuit times out, the data is latched, and the Data Available (DAV) output goes high.

If, during the key closure the switch bounces, Y1 input will go high again, restarting the scan and resetting the key bounce circuitry. The key may bounce several times, but as soon as the switch stays low for a debounce period, the closure is assumed valid and the data is latched.

A key may also bounce when it is released. To ensure that the encoder does not recognize this bounce as another key closure, the debounce circuit must time out before another closure is recognized.

The two key roll over feature can be illustrated by assuming a key is depressed, and then a second key is depressed. Since all scanning has stopped, and all other Y inputs are disabled, the second key is not recognized until the first key is lifted and the key bounce circuitry has reset.

The output latches feed TRI-STATE, which are enabled when the Output Enable ($\overline{OE}$) input is taken low.

National Semiconductor

MM54C932/MM74C932 Phase Comparator

General Description

The MM74C932/MM54C932 consists of two independent output phase comparator circuits. The two phase comparators have a common signal input and a common comparator input. The signal input can be directly coupled for a large voltage signal, or capacitively coupled to the self-biasing amplifier at the signal input for a small voltage signal.

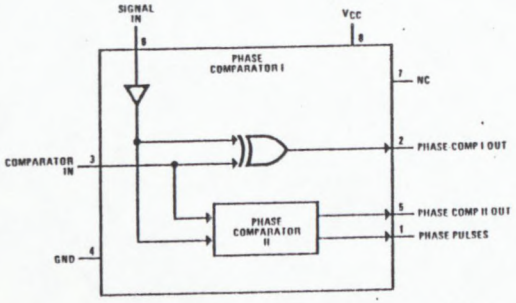
Phase comparator I, an exclusive-OR gate, provides a digital error signal (phase comp. I out) and maintains 90° phase shifts at the VCO center frequency. Between signal input and comparator input (both at 50% duty cycle), it may lock onto the signal input frequencies that are close to harmonics of the VCO center frequency.

Phase comparator II is an edge-controlled digital memory network. It provides a digital error signal (phase comp. II out) and lock in signal (phase pulses) to indicate a locked condition and maintains a 0° phase shift between signal input and comparator input.

Features

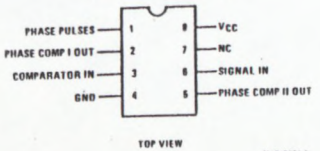
- Wide supply voltage range
- Convenient mini-DIP package
- TRI-STATE® phase-comparator output (comparator II)
- 200 mV input voltage (signal in) sensitivity (typical)

Block and Connection Diagrams



TL/F/5921-1

Dual-In-Line Package



TOP VIEW

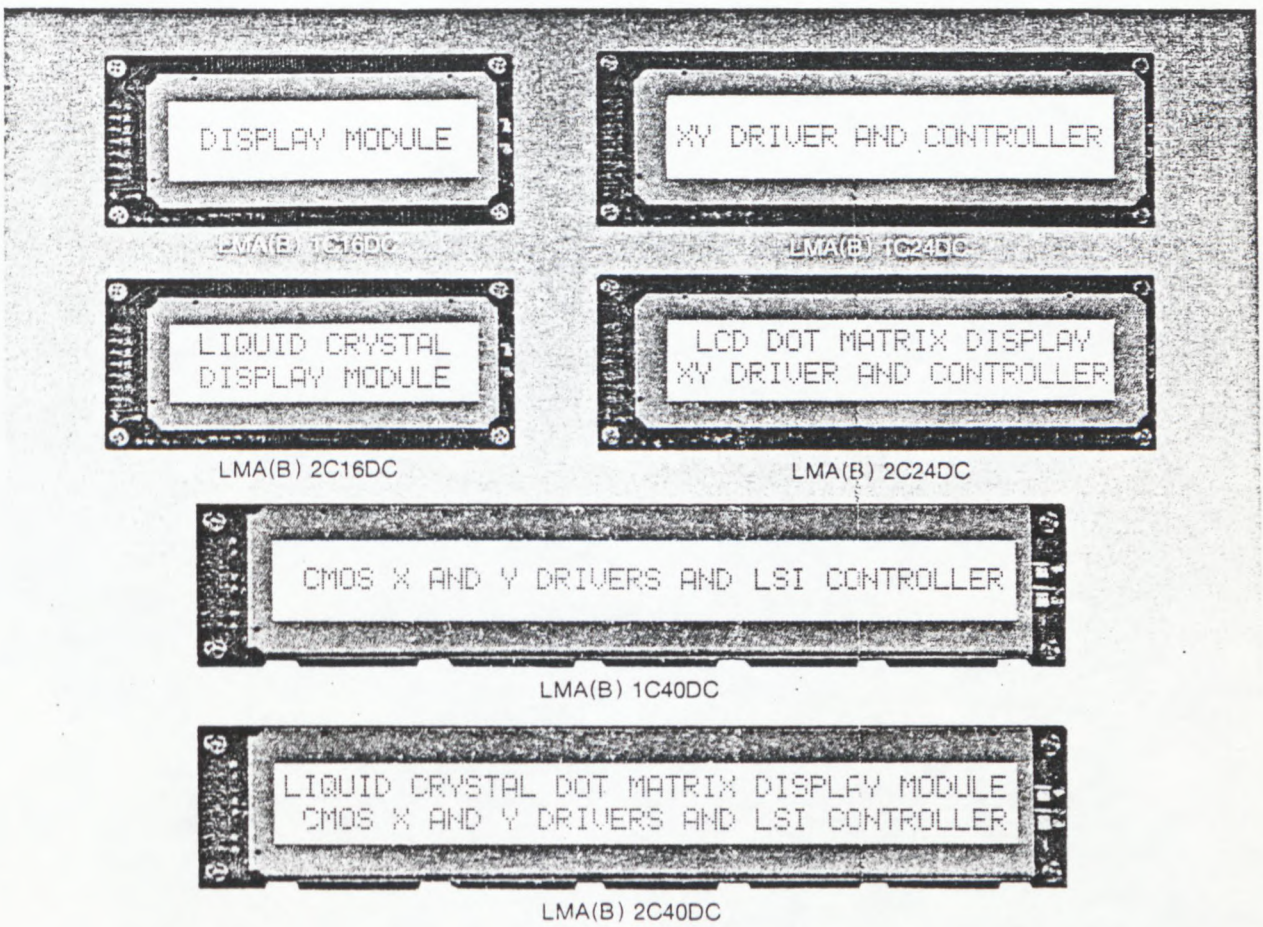
TL/F/5921-2

Order Number MM54C932N or MM74C932N
See NS Package N08E

DENSITRON CORPORATION

INFORMATION DISPLAY

DOT MATRIX LCD MODULES WITH EL BACKLIGHTING CHARACTER DISPLAY
LMA & LMB SERIES



LMA(B) 2C16DC

LMA(B) 2C24DC

LMA(B) 1C40DC

LMA(B) 2C40DC

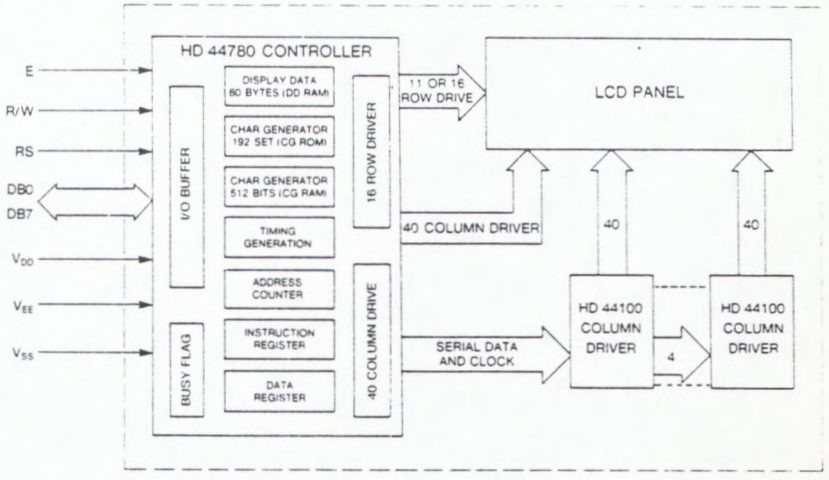
FEATURES:

- SIX MODELS — 16X1, 16X2, 24X1, 24X2, 40X1, 40X2 ARE AVAILABLE
- 5V, 2mA, SINGLE POWER SUPPLY
- EASY READABILITY, DOT MATRIX CHARACTERS WITH CURSOR
- SUNLIGHT AND NIGHT READABILITY WITH E-L BACKLIGHT
- WIDE, ADJUSTABLE VIEWING ANGLE
- EXTREMELY SMALL AND LIGHT WEIGHT
- TTL AND 5V CMOS COMPATIBLE
- INTERFACE DIRECTLY TO ANY 4 OR 8-BIT DATA BUS
- "POWERFUL" INSTRUCTIONS SAVE LINES OF CODING
- DISPLAY "BLANK," "WINK" OR "WINK LIMITED AREA"
- FULLY ASCII COMPATIBLE
- 192-CHARACTER GENERATOR ROM (96 Alphanumerics/Symbols, 64 Kata Kana, 32 Euro/Greek/Symbols)
- 8-USER PROGRAMMABLE RAM LOCATIONS FOR CUSTOM SYMBOLS
- 80-CHARACTER MEMORY ALLOWS EASY "SCROLLING" OR GENERAL PURPOSE STORAGE
- CURSOR "WINK," "REVERSE VIDEO," OR OFF
- AUTOMATIC DISPLAY SHIFT—SIMPLE TRANSMISSION OF "STRINGS"
- CAN SCROLL LEFT, RIGHT OR ALTERNATE WITH ENTIRE LINE REPLACEMENT
- 5X7 OR 5X10 FORMATS, DISPLAYS "TRUE" LOWER CASE DESCENDERS

APPLICATIONS

- TELECOMMUNICATIONS
- MEDICAL INSTRUMENTS
- HAND-HELD TERMINALS
- ELECTRONIC TYPEWRITERS
- POS TERMINALS
- TEST INSTRUMENTS
- LIGHT METERS
- CHEMICAL ANALYZERS
- WORD PROCESSORS
- NAVIGATION EQUIPMENT

TYPICAL MODULE BLOCK DIAGRAM



GENERAL DESCRIPTION:

These intelligent, compact modules are complete display systems with minimum external requirements. Their cost effectiveness, versatility and simplicity now make information display easy.

The modules' low power is due to an exceptional, on-board, flat-pack CMOS microprocessor and driver. Operation is this simple: connect to any 4 or 8-bit data bus (or a single memory chip), switch one control line, and apply a single, non-critical "enable" pulse. Instantly the desired character appears on the LCD. Shifting is automatic.

The modules' compactness is impressive. The 16-character display requires just 3⅜ x 1 x ½ inch. Mounting is simple and only 14 connections join the display. Electro-luminescent backlighting, integral to the display, is a low cost option. Holding frame is brushed stainless steel.

The display is virtually burden-free to the host processor. Internal registers store up to 80 characters and all display update and refresh is internal. Software is greatly eased by powerful, single step instructions which eliminate many lines of conventional graphics code.

Featured are a single, low power 5V supply, compatibility with TTL or 5V CMOS, and direct interface to 4 or 8-bit processors. A 192-character font provides 32 special codes, abbreviations and symbols. A further enhancement is an 8-character, fully user programmable, custom character capability. A uniquely customized, field-changeable, specialized graphic representation is now possible.

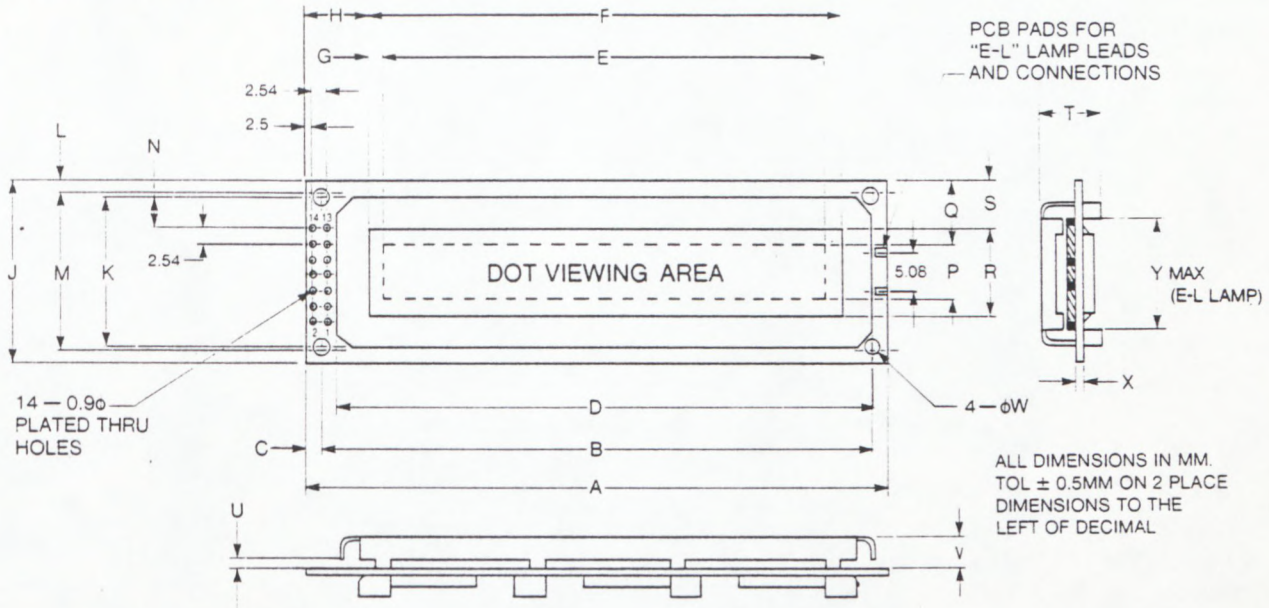
The simplicity and convenience of the module's operation means that now, for the first time, only the designer's imagination will be challenged. Display applications are truly limitless.

Available upon request are a series of broad based application notes to speed your system implementation. Our staff of experienced engineers are eager to assist and provide comprehensive technical support. We welcome your call.

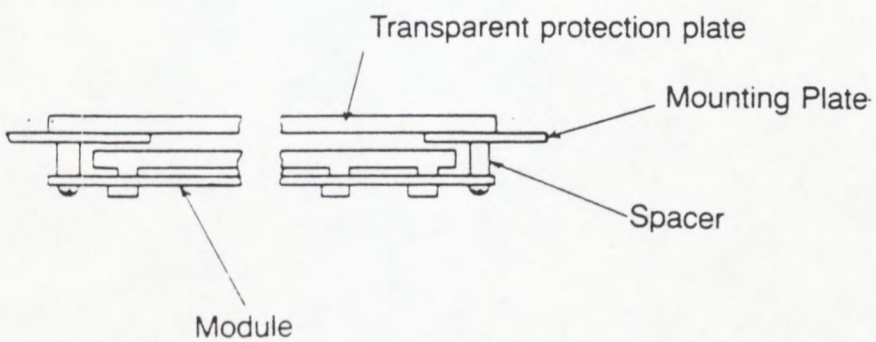
ORDERING INFORMATION

Characters per Line X Number of Lines	Module with Reflective Polarizer without EL Lamp	Module with Transflective Polarizer and EL Lamp	(EL Lamp Part Number : see page 15)	Applicable DC/AC Inverter : see page 16
16×1	LMA1C16-DC	LMB1C16-DCL	(ELS 16)	DAS 5V3
24×1	LMA1C24-DC	LMB1C24-DCL	(ELS 24)	
40×1	LMA1C40-DC	LMB1C40-DCL	(ELS 40)	
16×2	LMA2C16-DC	LMB2C16-DCL	(ELS 16)	
24×2	LMA2C24-DC	LMB2C24-DCL	(ELS 24)	
40×2	LMA2C40-DC	LMB2C40-DCL	(ELS 40)	

MECHANICAL DIMENSIONS



TYPICAL MODULE MOUNTING

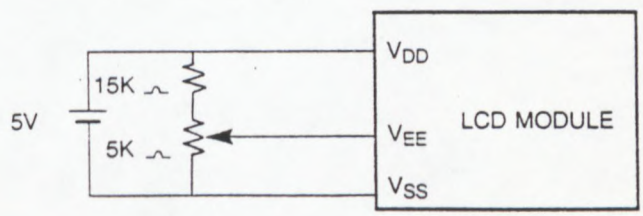


PART NO.	A	B	C	D	E	F	G	H	J	K	L	M
LMA(B)1C16-DC	85.0	80.0	2.5	73.5	59.4	63.5	2.05	13.25	36.0	31.0	2.5	28.5
LMA(B)1C24-DC	118.0	113.0	2.5	108.5	89.4	93.5	2.05	13.25	36.0	31.0	2.5	28.5
LMA(B)1C40-DC	182.0	175.0	3.5	163.5	149.4	154.5	2.85	16.30	33.5	26.5	3.5	32.5
LMA(B)2C16-DC	85.0	80.0	2.5	73.5	59.4	63.5	2.05	13.25	36.0	31.0	2.5	28.5
LMA(B)2C24-DC	118.0	113.0	2.5	108.5	89.4	93.5	2.05	13.25	36.0	31.0	2.5	28.5
LMA(B)2C40-DC	182.0	175.0	3.5	163.5	149.4	154.5	2.85	16.30	33.5	26.5	3.5	32.5

PART NO.	N	P	Q	R	S	T	U	V	W	X	Y
LMA(B)1C16-DC	7.88	8.7	13.65	15.8	10.1	9.3	1.4	4.7	2.5	1.6	20.0
LMA(B)1C24-DC	7.88	8.7	13.65	15.8	10.1	9.3	1.4	4.7	2.5	1.6	20.0
LMA(B)1C40-DC	7.00	8.7	13.25	15.8	8.5	9.3	1.4	4.7	3.5	1.6	22.5
LMA(B)2C16-DC	7.88	12.3	11.85	15.8	10.1	9.3	1.4	4.7	2.5	1.6	20.0
LMA(B)2C24-DC	7.88	12.3	11.85	15.8	10.1	9.3	1.4	4.7	2.5	1.6	20.0
LMA(B)2C40-DC	7.00	12.3	10.60	15.8	8.5	9.3	1.4	4.7	3.5	1.6	22.5

ELECTRO-OPTICAL CHARACTERISTICS $V_{DD} = 5.0 \pm 0.25V$, $T_a = 25°C$						
ITEM	SYMBOL	CONDITION	MIN.	TYP.	MAX.	UNIT
Input "High" Voltage	V_{IH}		2.2	—	V_{CC}	V
Input "Low" Voltage	V_{IL}		−0.3	—	0.6	V
Output "High" Voltage	V_{OH}	$-I_{OH} = 0.205mA$	2.4	—	—	V
Output "Low" Voltage	V_{OL}	$I_{OL} = 1.6mA$	—	—	0.4	V
Power Supply Current	I_{DD}	$V_{DD} = 5.0V$	—	0.5	2.0	mA
1 Line Module Drive Voltage (1/8 Duty) 5x7 Font + Cursor	$V_{DD} - V_{EE}$	$T_a = 0°C$	4.2	4.3	4.4	V
		$T_a = 25°C$	3.8	3.9	4.0	V
		$T_a = 50°C$	3.4	3.5	3.6	V
1 Line Module Drive Voltage (1/11 Duty) 5x10 Font + Cursor	$V_{DD} - V_{EE}$	$T_a = 0°C$	4.5	4.6	4.7	V
		$T_a = 25°C$	4.1	4.2	4.3	V
		$T_a = 50°C$	3.6	3.7	3.8	V
2 Line Module Drive Voltage (1/16 Duty) 5x7 Font + Cursor	$V_{DD} - V_{EE}$	$T_a = 0°C$	4.8	4.9	5.0	V
		$T_a = 25°C$	4.3	4.4	4.5	V
		$T_a = 50°C$	3.8	3.9	4.0	V
Viewing Angle	$\phi 1 - \phi 2$	K = 1.4	20	—	—	deg.
Contrast Ratio	K	$\phi = 20°$, $\theta = 0°$	3	—	—	
Rise Time	t_r	$\phi = 20°$	—	150	250	ms
Fall Time	t_f	$\phi = 20°$	—	150	250	ms

POWER SUPPLY



PIN FUNCTION		
PIN NAME	I/O	FUNCTION
V_{SS}		Ground; 0V
V_{DD}		+ 5V
V_{EE}		Power supply for LC driving
RS	I	Signal to select registers "0": Instruction register (for write) Busy flag; address counter (for read) "1": Data register (for read and write)
R/W	I	Signal to select read (R) and write (W) "0": write MPU → LCD Module "1": read MPU ← LCD Module
E	I	Operation start signal for data read or write
DB0 ~ DB3	I/O	Data bus of lower order 4 lines having bidirectional tri-state. Used for data transfer between the MPU and the module. These four are not used during 4-bit operation
DB4 ~ DB7	I/O	Data bus of higher order 4 lines having bidirectional tri-state. Used for data transfer between the MPU and the module. DB7 can be used as a BUSY flag.

NOTE: In the module, the data can be sent in either 4-bit 2 sequence operation or 8-bit single-operation so that it can interface to both 4 and 8 bit MPU's.

(1) When interface data is 4 bits long, data is transferred using only lines DB4 ~ DB7 and DB0 ~ DB3 are not used. Data transfer between the module and the MPU is complete when the 4-bit data is transferred twice. Data of the higher order 4 bits (the contents of DB4 ~ DB7 when the interface data is 8 bits long) is transferred first, and the lower order 4 bits (the contents of DB0 ~ DB3 when the interface data is 8 bits long) follows.

(2) When the interface data is 8 bits long, data is transferred using all 8 data lines of DB0 ~ DB7.

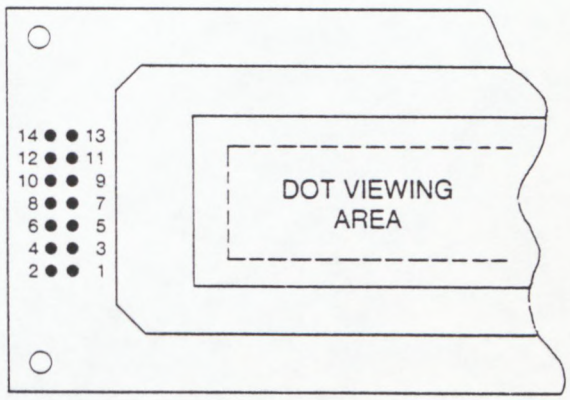
ABSOLUTE MAXIMUM RATINGS

ITEM	SYMBOL	MIN.	MAX.	UNIT
Logic circuit power supply voltage	V_{DD} - V_{SS}	0	7.0	V
LC driver circuit supply voltage	V_{DD} - V_{EE}	0	13.5	V
Input voltage	V_I	V_{SS}	V_{DD}	V
Operating temperature	t_a	0	50	°C
Storage temperature	t_{stg}	−20	70	°C

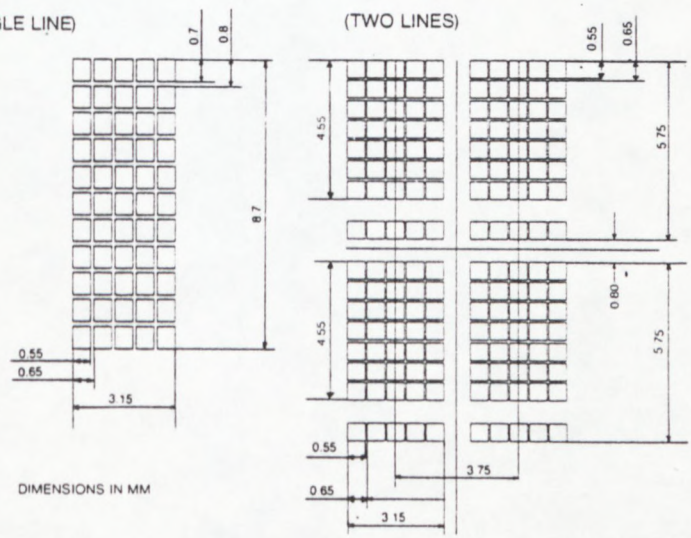
PIN CONNECTION

PIN NO.	SIGNAL	LEVEL
1	V_{SS}	GROUND
2	V_{DD}	+ 5.0V
3	V_{EE}	≃ + 0.7V
4	RS	L = INST, H = CHAR
5	R/W	L = W, H = R
6	E	LATCH ON FALL
7	DB0	POSITIVE LOGIC
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	

(Top View)

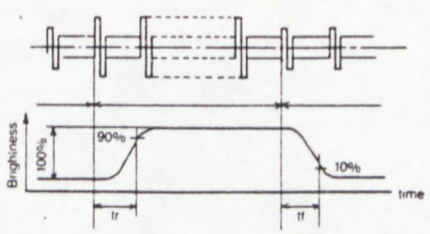


Display Pattern

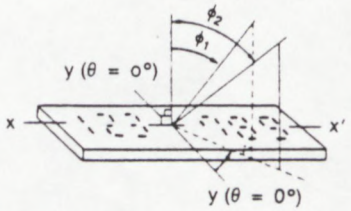


DIMENSIONS IN MM

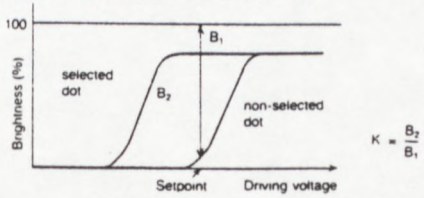
Definition of optical response



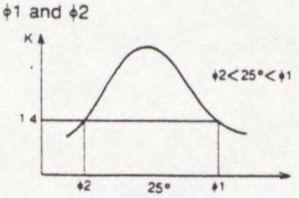
Definition of angle ∅ and θ



Definition of contrast 'K'



ϕ1 and ϕ2

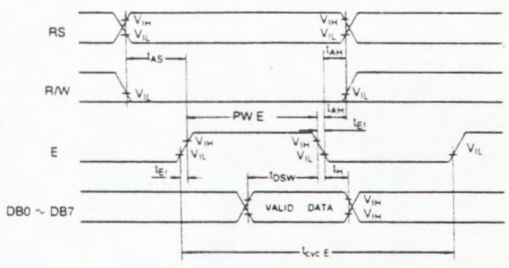


FONT TABLE

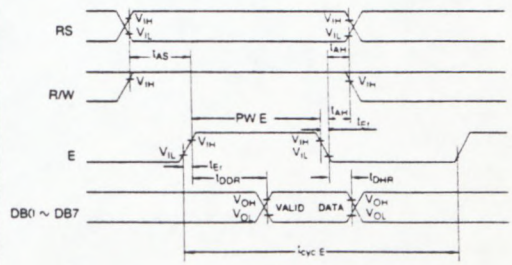

UPPER 4 BIT HEXADECIMAL
LOWER 4 BIT HEXADECIMAL
Higher 4 bit
Lower 4 bit
0 2 3 4 5 6 7 A B C D E F
0000 0010 0011 0100 0101 0110 0111 1010 1011 1100 1101 1110 1111
0 xxxx0000 CG RAM (1)
1 xxxx0001 (2)
2 xxxx0010 (3)
3 xxxx0011 (4)
4 xxxx0100 (5)
5 xxxx0101 (6)
6 xxxx0110 (7)
7 xxxx0111 (8)
8 xxxx1000 (1)
9 xxxx1001 (2)
A xxxx1010 (3)
B xxxx1011 (4)
C xxxx1100 (5)
D xxxx1101 (6)
E xxxx1110 (7)
F xxxx1111 (8)

TIMING CHARACTERISTICS

WRITE OPERATION					
ITEM	SYMBOL	MIN.	TYP.	MAX.	UNIT
Enable Cycle Time	$t_{cyc\ E}$	1.0	—	—	μs
Enable Pulse Width	PW E	450	—	—	ns
Enable Rise/Fall Time	t_{Er}, t_{Ef}	—	—	25	ns
Address Set-up Time	t_{AS}	140	—	—	ns
Address Hold Time	t_{AH}	10	—	—	ns
Data Start-up Time	t_{DSW}	195	—	—	ns
Data Hold Time	t_H	10	—	—	ns



READ OPERATION					
ITEM	SYMBOL	MIN.	TYP.	MAX.	UNIT
Enable Cycle Time	$t_{cyc\ E}$	1.0	—	—	μs
Enable Pulse Width	PW E	450	--	—	ns
Enable Rise/Fall Time	t_{Er}, t_{Ef}	—	—	25	ns
Address Set-up Time	t_{AS}	140	—	—	ns
Address Hold Time	t_{AH}	10	—	—	ns
Data Delay Time	t_{DDR}	—	—	320	ns
Data Hold Time	t_{DHR}	20	—	—	ns

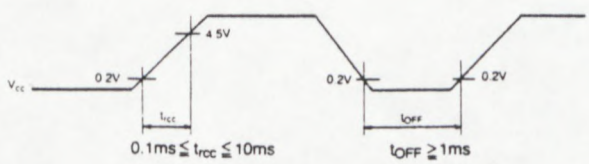


RESET FUNCTION

The module automatically performs initialization (reset) when power is turned on (using internal reset circuit). The following instructions are executed in initialization:

(1) Clear display
The busy flag is kept in the busy state (BF = 1) until initialization ends. The time is 15ms.
(2) Function set..... DL = 1: 8 bits wide interface data
N = 0 : 1-line display
F = 0 : 5×7 dot character font
(3) Display ON/OFF control ..... D = 0 : Display OFF
C = 0 : Cursor OFF
B = 0 : Blink OFF
(4) Entry mode set..... I/O = 1: +1 (increment)
S = 0 : No shift
(5) DD RAM is selected

Because initialization may not be performed completely depending on the rise time of the power supply when it is turned on, pay attention to the following time relationship.



t_{OFF} stipulates the time of power OFF for power supply instantaneous dip or when power supply repeats ON and OFF.

NOTE: When the above power supply condition is not satisfied, the internal reset circuit does not operate normally. In this case, perform the needed initialization by sending instructions from the MPU after turning the power ON. When initialization by the internal reset circuit is not performed normally, whether the interface data is 4 bits long or 8 bits long is not clear. Therefore designate 8-bit data length by sending the function set instruction twice and then perform the required initialization.

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	1	*	*	*	*
0	0	0	0	1	1	*	*	*	*

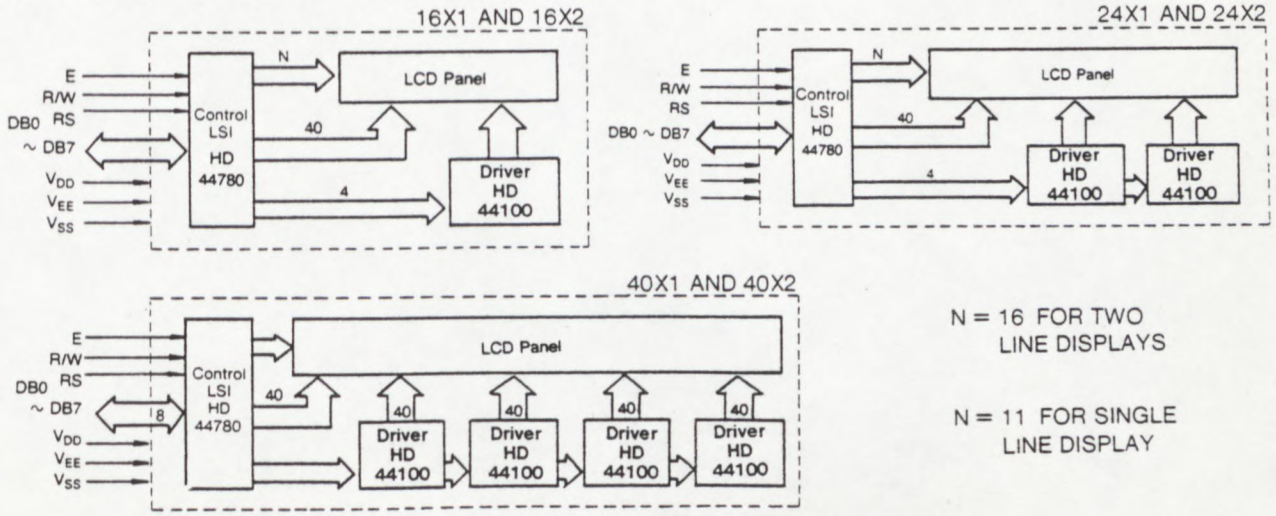
When this instruction is sent, the module enters the 8-bit data length mode without fail.

INSTRUCTIONS

Instruction	Code										Description	Execution Time (when f_{cp} or f_{osc} is 250KHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears all display and returns the cursor to the home position (Address 0)	82 μs ~ 1.64ms
Return Home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (Address 0). Also returns the display being shifted to the original position. DD RAM contents remain unchanged.	40 μs ~ 1.6ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and specifies or not to shift the display. These operations are performed during data write and read.	40 μs
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of all display (D), Cursor ON/OFF (C), and blink of cursor position character (B).	40 μs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing DD RM contents.	40 μs
Function Set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), number of display lines (L), and character font (F).	40 μs
Set CG RAM Address	0	0	0	1	A_{CG}						Sets the CG RAM address. CG RAM data is sent and received after this setting.	40 μs
Set DD RAM Address	0	0	1	A_{DD}							Sets the DD RAM address. DD RAM data is sent and received after this setting.	40 μs
Read Busy Flag & Address	0	1	BF	AC							Reads Busy flag (BF) indicating internal operation is being performed and reads address counter contents	40 μs
Write Data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM	40 μs
Read Data to CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM	40 μs
	I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift S/C = 1: Display shift S/C = 0: Cursor move R/L = 1: Shift to the right R/L = 0: Shift to the left DL = 1: 8 bits DL = 0: 4 bits N = 1: 2 lines N = 0: 1 line F = 1: 5 x 10 dots F = 0: 5 x 7 dots BF = 1: Internally operating BF = 0: Can accept instruction										DD RAM: Display data RAM CG RAM: Character generator RAM A_{CG}: CG RAM address A_{DD}: DD RAM address Corresponds to cursor address. AC: Address counter used for both of DD and CG RAM address.	Execution time changes when frequency changes. (Example) When f_{cp} or f_{OSC} is 270 KHz: $40\mu s \times \frac{250}{270} = 37\mu s$	

*Don't care

MODULE BLOCK DIAGRAMS



N = 16 FOR TWO LINE DISPLAYS

N = 11 FOR SINGLE LINE DISPLAY

INSTRUCTION DESCRIPTION

• OUTLINE

Two registers of the HD44780, the Instruction Register (IR) and the Data Register (DR) only can be controlled by MPU directly. Control information is temporarily stored in these registers, prior to internal operation start, to allow interface to various types of MPUs which operate in different speeds from HD44780 internal operation or to allow interface to peripheral control ICs. The HD44780 internal operation is determined by signals sent from the MPU, these signals including register selection signals (RS), read/write signals (R/W) and data bus signals ($DB_0 \sim DB_7$), are called instructions in this paragraph. Table on page 10 shows the instructions and the execution time of the instructions. Details are explained in the following sections. The instructions can be divided into the following 4 types:

(1) Instructions that designate the HD44780 functions such as display format, data length, etc.
(2) Instructions that give internal RAM addresses.
(3) Instructions that perform data transfer with internal RAM.
(4) Other instructions

In the normal use, instructions of category (3), which sends display data, is used most frequently. However, since the HD44780 internal RAM addresses are configured to be automatically incremented (or decremented) by +1 after each data write, MPU program load is lessened. Especially, display shift is performed concurrently with display data write, and this enables the user to develop systems with minimum time and maximum efficiency of programming. When an instruction is being executed (during internal operation), the busy flag DB_7 is active high. This must be monitored when high speed operation is planned (≈50KHz).

• CLEAR DISPLAY

	RS	R/W	DB_7							DB_0
Code	0	0	0	0	0	0	0	0	0	1

Writes space code "20" (hexadecimal) into all the DD RAM addresses. The cursor returns to Address 0 (A_{dd} = "80") and display, if it has been shifted, returns to the original position. In other words, display disappears and the cursor goes to the left edge of the display (the first line if 2 lines are displayed).

• RETURN HOME

	RS	R/W	DB_7							DB_0
Code	0	0	0	0	0	0	0	0	1	*

*(Don't Care)

Returns the cursor to Address 0 (A_{DD} = "80") and display, if it has been shifted, to the original position. The DD RAM contents remain unchanged.

• ENTRY MODE SET

	RS	R/W	DB_7							DB_0
Code	0	0	0	0	0	0	0	1	I/D	S

I/D: Increments (I/D = 1) or decrements (I/D = 0) the DD RAM address by one upon writing into or reading from the DD RAM a character code. The cursor moves to the right when incremented by one. The same applies to writing and reading of CG RAM.

S: Shifts the entire display to either the right or the left when S is 1; to the left when I/D = 1 and to the right when I/D = 0. Therefore, the cursor looks as if it stood still with the display only moved. Display is not shifted when reading from the DD RAM. Display is not shifted when S = 0.

• DISPLAY ON/OFF CONTROL

	RS	R/W	DB_7							DB_0
Code	0	0	0	0	0	0	1	D	C	B

D: Display is turned ON when D = 1 and OFF when D = 0. When display is turned off due to D = 0, the display data remains in the DD RAM and it can be displayed immediately by setting D = 1.

C: The cursor is displayed when C = 1 and not displayed when C = 0. Even if the cursor disappears, function of I/D, etc. does not change during display data write. The cursor is displayed using 5 dots in the 8th lines when the 5 × 7 dot character font is selected and in the 11th line when 5 × 10 dot character font is selected.

B: The character residing at the cursor position blinks when B = 1. The blink is done by switching between all the black dots and display characters at 0.4 second interval. The cursor and the blink can be set concurrently.

(a) Cursor Display Example

(b) Blink Display Example

• CURSOR OR DISPLAY SHIFT

	RS	R/W	DB_7							DB_0
Code	0	0	0	0	0	1	S/C	R/L	*	*

*(Don't Care)

Shifts the cursor position or display to the right and the left without writing or reading the display data. This function is used for correction or search of display.

S/C	R/L	
0	0	Shifts the cursor position to the left. (AC is decremented by one.)
0	1	Shifts the cursor position to the right. (AC is incremented by one.)
1	0	Shifts the entire display to the left. The corsor follows the display shift.
1	1	Shifts the entire display to the right. The cursor follows the display shift.

• FUNCTION SET

	RS	R/W	DB_7							DB_0
Code	0	0	0	0	1	DL	N	F	*	*

*(Don't Care)

DL: Sets interface data length. Data is sent or received in 8 bit length ($DB_7 \sim DB_0$) when DL = 1 and 4 bit length ($DB_7 \sim DB_4$) when DL = 0. When 4 bit length is selected, data must be sent or received twice.
N: Sets number of display lines.
F: Sets character font.

N	F	No. of Display Lines	Character Font	Duty Facotr	Remarks
0	0	1	5×7 dots	1/8	
0	1	1	5×10 dots	1/11	
1	*	2	5×7 dots	1/16	Cannot display 2 lines with 5×10 dot character font.

*(Don't Care)

• SET CG RAM ADDRESS

	RS	R/W	DB_7							DB_0
Code	0	0	0	1	A	A	A	A	A	A

← Higher Order Bits Lower Order Bits →

Sets the CG RAM address in a binary number of AAAAAA to the address counter, and data is written or read from the MPU related to the CG RAM after this.

• SET DD RAM ADDRESS

	RS	R/W	DB_7							DB_0
Code	0	0	1	A_N	A	A	A	A	A	A

← Higher Order Bits Lower Order Bits →

Sets the DD RAM address in a binary number of A_NAAAAAA to the address counter, and data is written or read from the MPU related to the DD RAM after this. However, when N = 0 (1-line display), A_NAAAAAA is "00"–"4F" (hexadecimal). When N = 1 (2-line display), A_NAAAAAA is "00"–"27" (hexadecimal) for the first line, and "40"–"67" (hexadecimal) for the second line.

• READ BUSY FLAG AND ADDRESS

	RS	R/W	DB_7							DB_0
Code	0	1	BF	A	A	A	A	A	A	A

← Higher Order Bits Lower Order Bits →

Reads Busy Flag (BF) that indicates the system is internally operating on an instruction received before. When BF = 1, it indicates that internal operation is going on and the next instruction is not accepted until BF is set to "0." Check the BF status before the next write operation. At the same time, this value of the address counter expressed in a binary number of AAAAAA. The address counter is used by both of CG and DD RAM address, and its value is determined by the previous instruction. Address contents are those of CG RAM and DD RAM previously shown.

• WRITE DATA TO CG OR DD RAM

	RS	R/W	DB_7							DB_0
Code	1	0	D	D	D	D	D	D	D	D

← Higher Order Bits Lower Order Bits →

Writes binary 8 bit data DDDDDDDD to the CG or the DD RAM. Whether the CG or the DD RAM is to be written is determined by the previous designation (CG RAM address setting or DD RAM address setting). After write, the address is automatically incremented or decremented by one according to entry mode. Display shift also follows the entry mode.

• READ DATA FROM CG OR DD RAM

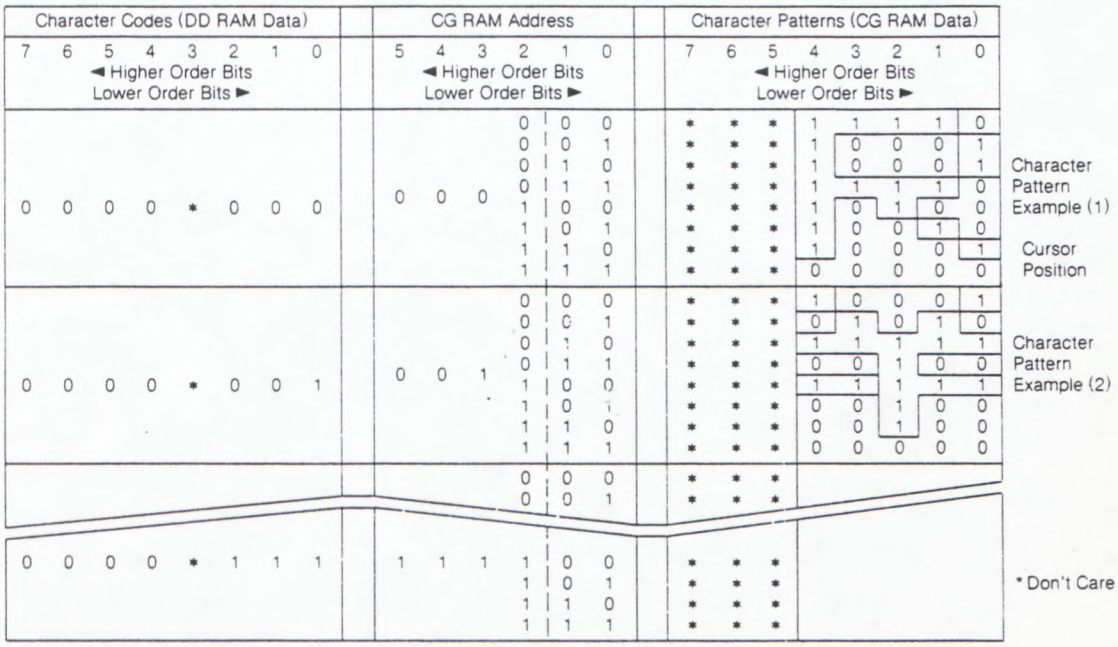
	RS	R/W	DB_7							DB_0
Code	1	1	D	D	D	D	D	D	D	D

← Higher Order Bits Lower Order Bits →

Reads binary 8 bit data DDDDDDDD from the CG or the DD RAM. Whether the CG RAM or the DD RAM is to be read is determined by the previous designation. *Prior to inputting this read instruction, either the CG RAM address set instruction or the DD RAM address set instruction must be executed. If it is not done, the first read data becomes invalid, and data of the next address is read normally from the second read.* After read, the address is automatically incremented or decremented by one according to the entry mode. However, display shift is not performed regardless of entry mode types.

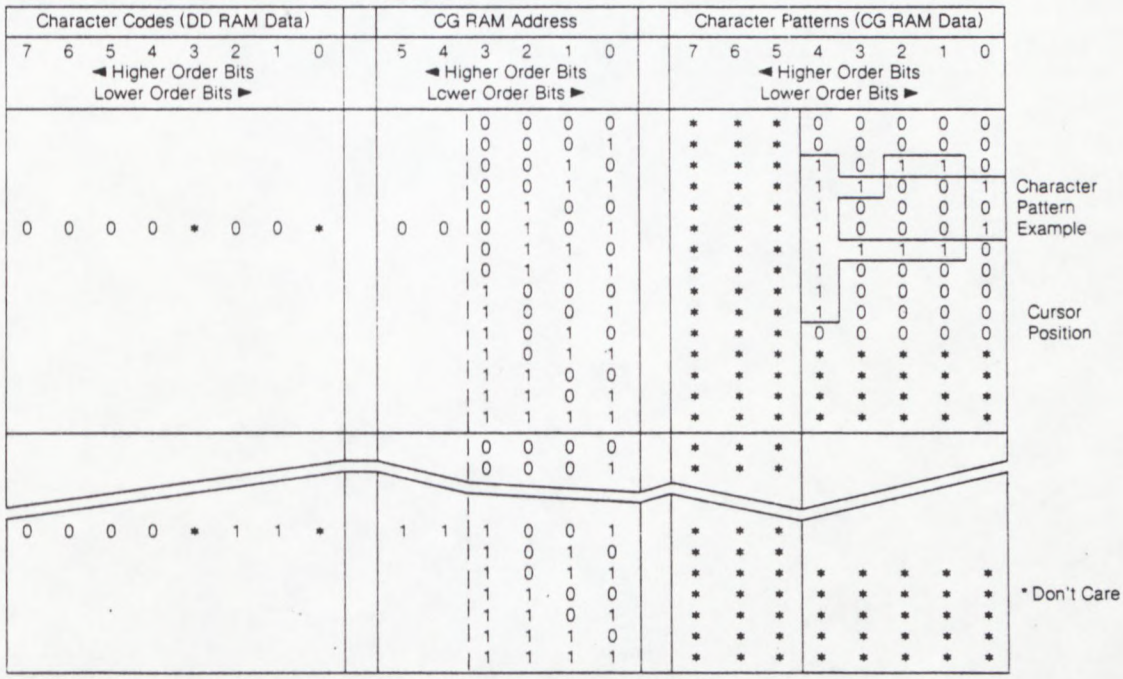
RELATION BETWEEN CG RAM ADDRESSES AND CHARACTER CODES (DD RAM) AND CHARACTER PATTERNS (CG RAM DATA)

- FOR 5×7 DOT CHARACTER PATTERNS



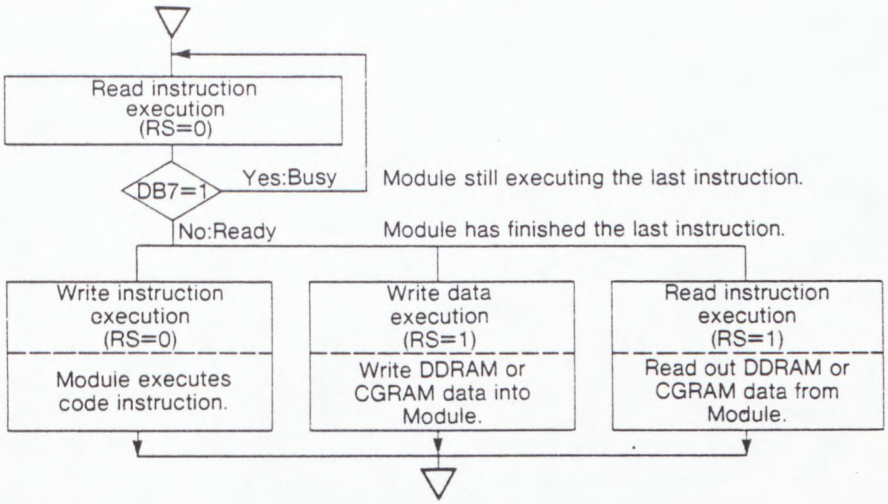
Note 1: Character code bits 0 ~ 2 correspond to CG RAM address bits 3 ~ 5 (3 bits: 8 types).
2: CG RAM address bits 0 ~ 2 designate character pattern line position. The 8th line is the cursor position and display is performed in logical OR with cursor.
3: Character pattern row positions correspond to CG RAM data bits 0 ~ 4, as shown in the figure (Bit 4 being at the left end). Since CG RAM data bits 5 ~ 7 are not used for display, they can be used as general data RAM.
4: CG RAM character patterns are selected when character code bits 4 ~ 7 are all "0." However, since character code bit 3 is a don't care bit, "R" display in the character pattern example, for example, are selected by character code "00" (hexadecimal) or "08" (hexadecimal).
5: "1" for CG RAM data corresponds to selection for display and "0" for non-selection.

- FOR 5×10 DOT CHARACTER PATTERNS



Note 1: Character code bits 1, 2 correspond to CG RAM address bits 4, 5 (2 bits: 4 types).
2: CG RAM address bits 0 ~ 3 designate character pattern line position. The 11th line is the cursor position and display is performed in logical OR with cursor.
Since the 12th ~ 16th lines are not used for display, they can be used as general data RAM.
3: Character pattern row positions are the same as the positions for 5×7 dot character patterns.
4: CG RAM character patterns are selected when character code bits 4 ~ 7 are all "0." However, since character code bit 0 and 3 are don't care bits, "P" display in the character pattern example, for example, are selected by character code "00," "01," "08" and "09" (hexadecimal).
5: "1" for CG RAM data corresponds to selection for display and "0" for non-selection.

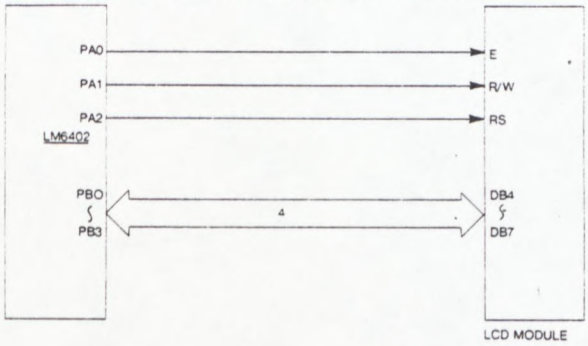
INTERFACING WITH MICROPROCESSORS



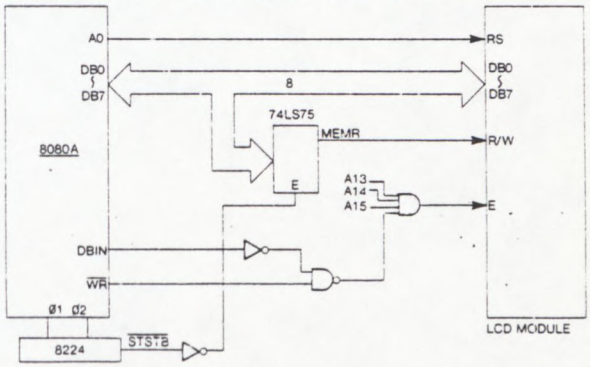
Note • In this flowchart, Read and Write instructions correspond with external RAM reference instruction or I/O instruction.

• By providing sufficient instruction execution time, it is unnecessary to monitor the "Busy line."

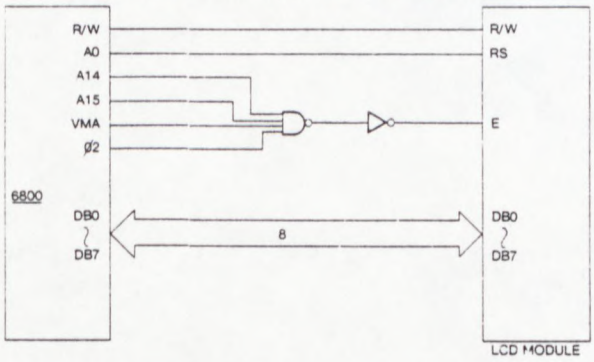
INTERFACE WITH HMCS 40



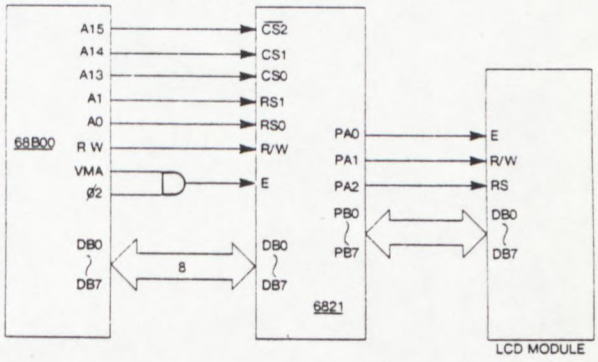
INTERFACE WITH 8080A



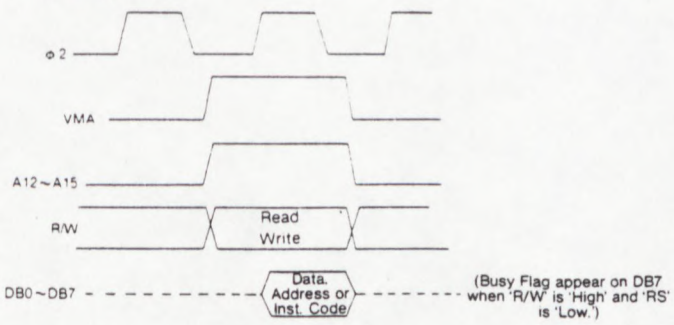
INTERFACE WITH 68B00/PIA



INTERFACE WITH 6800



TIMING CHART



- **6800**
  - LCD module is treated as I/O or RAM device.
  - As Ao is connected to RS. Therefore Ao's level controls the instruction register or the data register.
  - The address of the module as I/O device in the address area of the 6800 are:

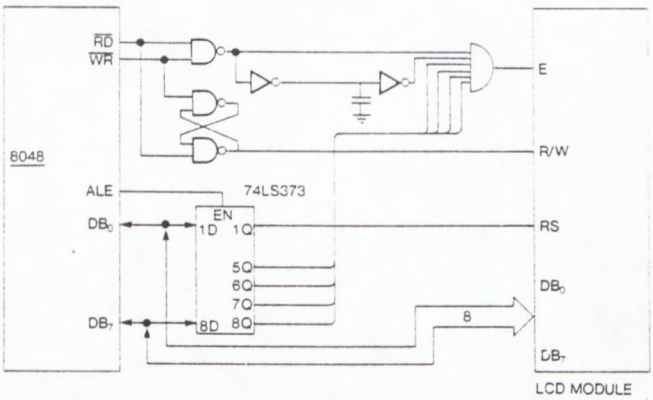
Write function	# 'FXXØ' (R/W = 0)
Write CGRAM/DDRAM data	# 'FXX1' (R/W = 0)
Read Busy flag/address	# 'FXXØ' (R/W = 1)
Read CGRAM/DDRAM data	# 'FXX1' (R/W = 1)

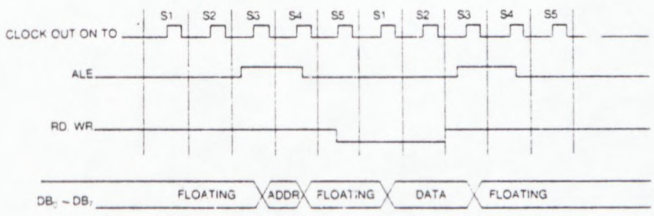
Note: The lead bit 'F' may be 'C', 'D', or 'E'

- **68B00**

The 68B00 is a high speed version of the 6800. The 02 signal is too narrow to properly drive the LCD module. A simple interface is the insertion of the 6821 PIA between the CPU and module. This widens the pulse sufficiently.

INTERFACE WITH 8048 (8049)

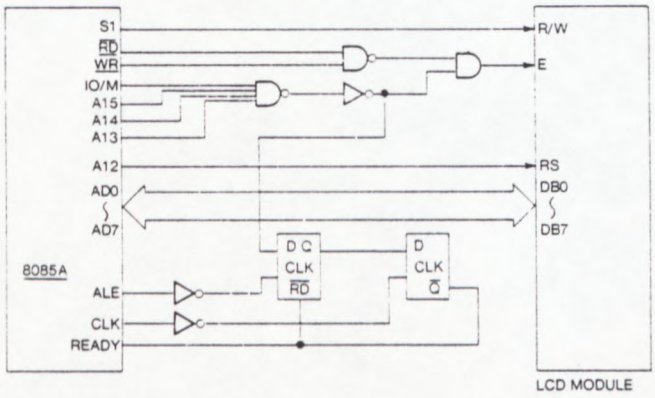


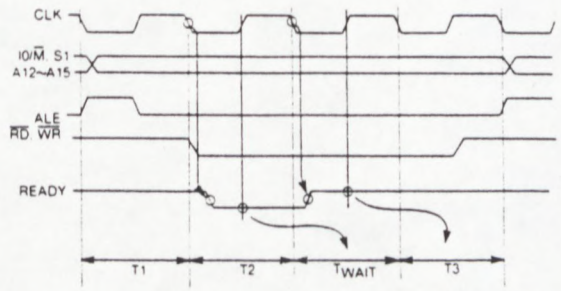


- The module is treated as external RAM.
- As the 8048 doesn't have an equivalent signal to R/W, this signal is made from RD and WR.
- 'NAND' of RD and WR is regarded as 'E' signal. Set-up-time between the leading edge of R/W and the leading edge of 'E' signal should be min. 140nsec, where the leading edge of R/W should precede that of 'E' signal. To keep this set-up-time (min. 140nsec.), the value of 'C' and 'R' should be determined.
- In addressing of the module as external RAM in the address area of 8048, the address on data bus (DB0 ~ DB7) are latched by ALE signal at the execution of external RAM reference instruction (MOVXA, @ Rr: MOVX @ Rr, A). In this case, Ao corresponds to RS of the module, and 'AND' of A4 ~ A7 and 'OR' signal of RD, WR has chip select function.

Write function	# 'FØ' (R/W = 0)
Write CGRAM/DDRAM data	# 'F1' (R/W = 0)
Read busy flag/address	# 'FØ' (R/W = 1)
Read CGRAM/DDRAM data	# 'F1' (R/W = 1)

INTERFACE WITH 8085A

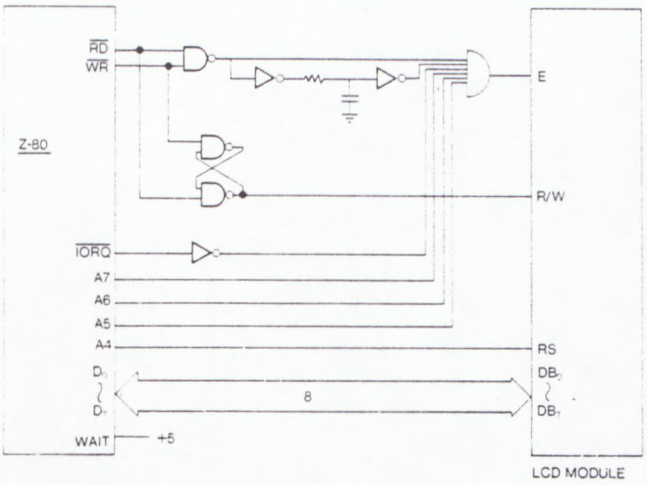




- LCD module is treated as I/O or RAM device.
- RD, WR signal of the 8085A are min. 400nsec. and insufficient to meet the needs of the 'E' signal of the module, min. 450nsec. RD, WR pulse width should be widened with providing T_{WAIT} cycle.
- The address of the module as I/O device in the address area of the 8085A are:

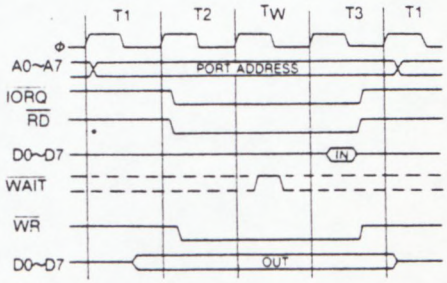
Write function	# 'EX' (R/W = 0)
Write CGRAM/DDRAM data	# 'FX' (R/W = 0)
Read busy flag/address	# 'EX' (R/W = 1)
Read CGRAM/DDRAM data	# 'FX' (R/W = 1)

INTERFACE WITH Z-80

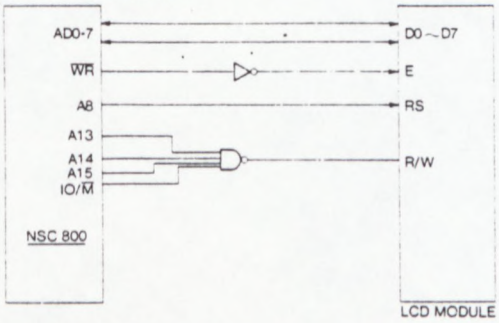


- The module is treated as I/O device.
- In I/O cycle, 1 WAIT STATE (TW) is automatically inserted, and OR signal (E) of $\overline{RD}$, $\overline{WR}$ can be sufficient min. 450nsec.
  When the module is treated as RAM device, 1 WAIT STATE (TW) should be provided with WAIT signal.
- The method of making R/W, 'E' signal of the module from $\overline{RD}$, $\overline{WR}$ is same as that of 8048.
- The address of the module as I/O device in the address area of the Z-80 are:

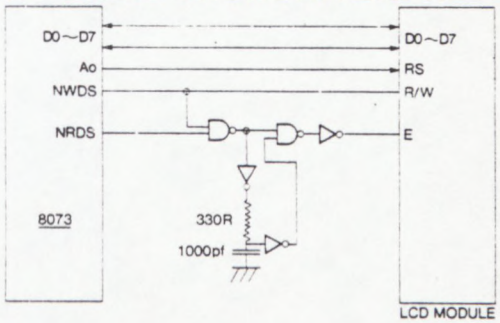
Write function	# 'EX' (R/W = 0)
Write CGRAM/DDRAM data	# 'FX' (R/W = 0)
Read busy flag/address	# 'EX' (R/W = 1)
Read CGRA/DDRAM data	# 'FX' (R/W = 1)



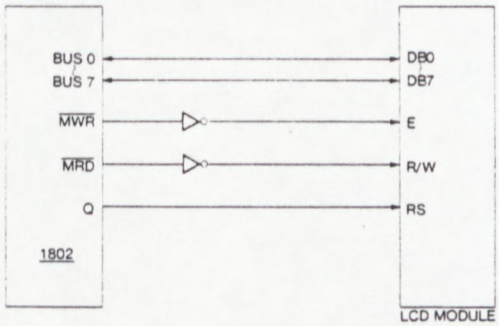
INTERFACE WITH NSC 800 CMOS MICRO



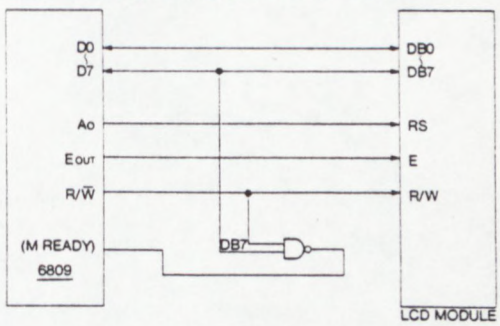
INTERFACE WITH NSC 8073 TINY BASIC MICROINTERPRETER



INTERFACE WITH HUGHES/RCA 1802 CMOS MICRO



INTERFACE WITH MOTOROLA HIGH PERFORMANCE 6809



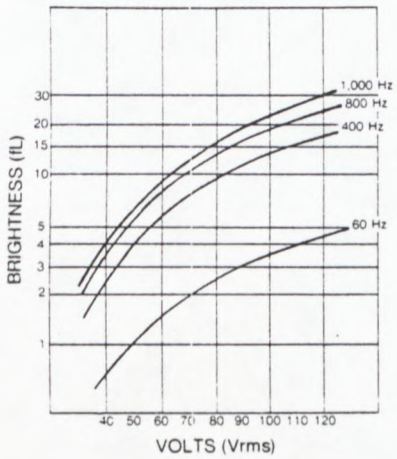
DENSITRON CORPORATION

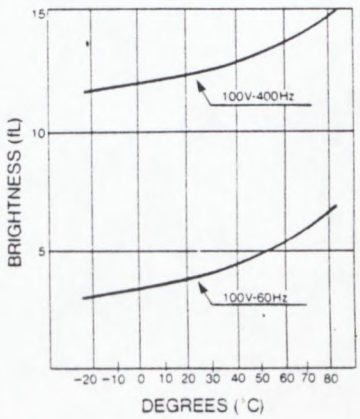
EL LAMPS MODEL ELS FOR LMB SERIES

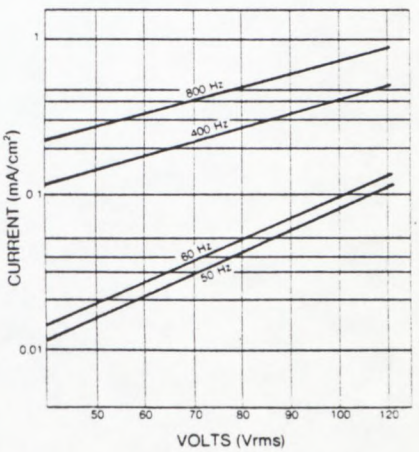
Ordering Number:

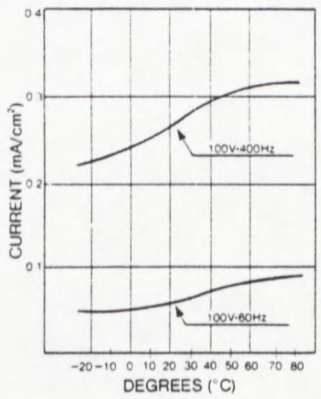
ELS16 for LMB1C16-DC, LMB2C16-DC
ELS24 for LMB1C24-DC, LMB2C24-DC
ELS40 for LMB1C40-DC, LMB2C40-DC

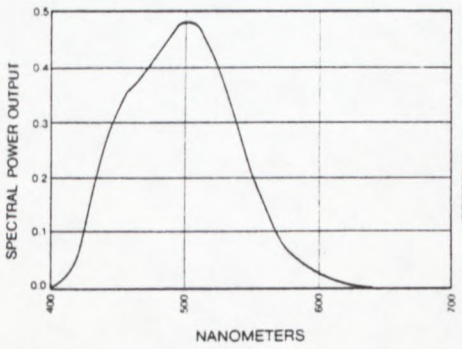
E-L backlighting is provided as a standard option. The blue-green lamp has 50cd/m2 (14fL) of initial brightness and a half-life of 7,500 hours under a constant voltage/frequency condition of 100V/400Hz. Constant power DC to AC inverters are commercially available which will extend the half-life. The lamp color can be shifted to green, yellow or light red by the addition of a filter. However, once the lamp is put behind the LCD display, the brightness observed by the viewer is reduced to approximately 15% of what the lamp emits resulting in about 7.5cd/m2 (2fL) which is more than sufficient for normal night time viewing and may need to be reduced for almost completely dark ambients.

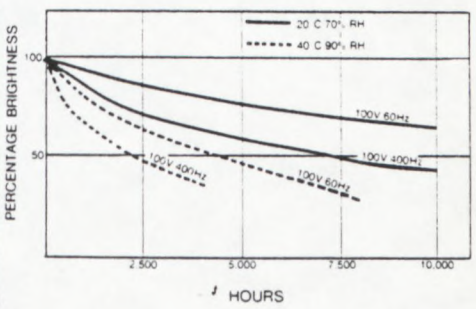


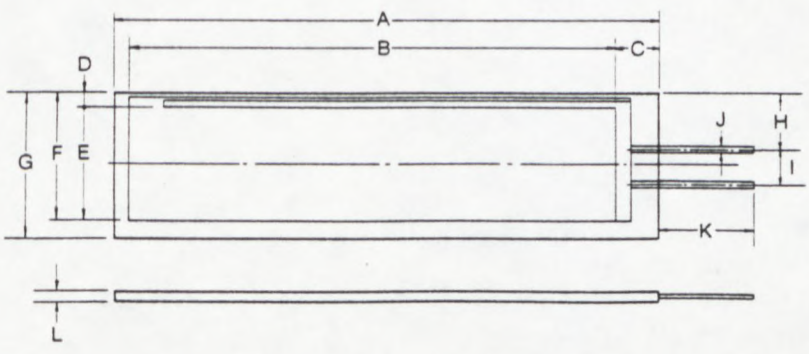












	ELS16	ELS24	ELS40
A	MAX 75.00	MAX 107.00	MAX 167.00
B	MIN 65.00	MIN 97.00	MIN 157.00
C	MAX 6.00	MAX 6.50	MAX 8.00
D	MAX 2.50	MAX 2.50	MAX 2.50
E	MIN 15.80	MIN 15.80	MIN 15.80
F	MIN 16.30	MIN 17.30	MIN 18.00
G	$20.00^{-0}_{-0.5}$	$20.00^{-0}_{-0.7}$	$21.50^{-0.5}_{-1.0}$
H	6.50±1.0	6.50±1.0	8.20±1.0
I	5.00	5.00	5.00
J	2 x 0.75[1]	2 x 0.75[1]	2 x 0.75[1]
K	10.00±1.0	6.00±1.0	16.00±1.0
L	MAX 1.30	MAX 1.30	MAX 1.30

DENSITRON CORPORATION

DC TO AC INVERTER
MODEL DAS 5V3

Ordering Number: DAS5V3

This inverter is a compact encapsulated module used to generate the AC drive signal necessary to excite the E-L lamp. It should be operated from your unregulated DC line due to generated switching noise and will draw relatively low input currents.

- **MAXIMUM RATINGS:**

Input Voltage: 6.0 VDC
Storage Temp.: −20 to +70°C

- **OPERATING RANGE:**

Input Voltage: 2.5 to 5.5 VDC
Ambient Temp.: −10 to +60°C

- **MATERIALS:**

Case: ABS Resin
Fill: Epoxy Compound
Color: Black or Grey

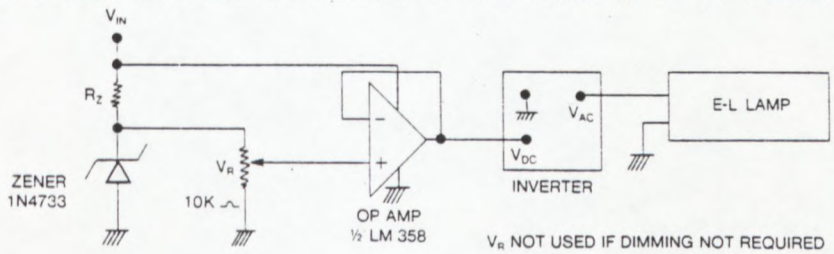
- **TYPICAL OPERATING CONDITIONS:**

Input Voltage: 5.0 VDC

Lamp Load	16x1 or 2	24x1 or 2	40x1 or 2	Units
Input Current	29	31	35	mA
Output Voltage	84	91	82	VRMS
Output Freq.	525	500	415	Hz
Lamp Brightness	74 (21)	53 (15)	45 (13)	cd/m² (ft-L)

Overvoltage Input Current: 60 MA Max

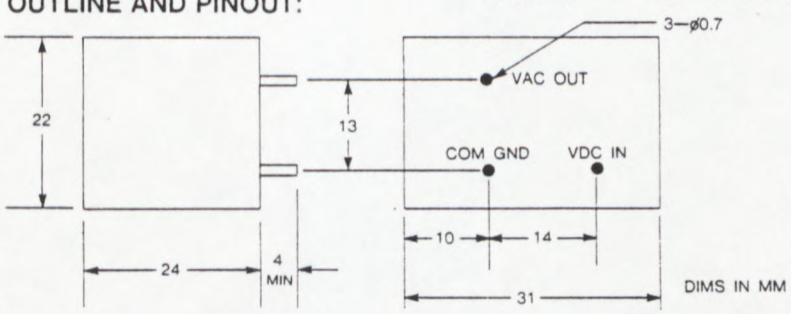
- **OPERATION OF INVERTER FROM LARGER INPUT VOLTAGES WITH VARIABLE LAMP DIMMING**

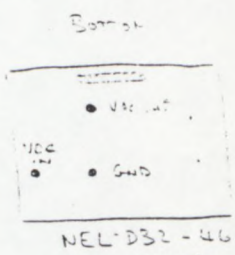


V_R NOT USED IF DIMMING NOT REQUIRED

V_{IN}	R_Z
9.0VDC	220 Ω, ¼ w
12.0VDC	360 Ω, ¼ w
24.0VDC	1K Ω, ½ w
28.0VDC	1.2K Ω, 1 w

- **DIMENSIONAL OUTLINE AND PINOUT:**





DENSITRON CORPORATION
Daini-Hasegawa Bldg., 4-F,
1-12-4, Hamamatsucho, Minato-ku, Tokyo
105 Japan Cable: "DENTROROSE" Tokyo
Phone: (03) 437-3721 Telex: No. J26914

HD146818

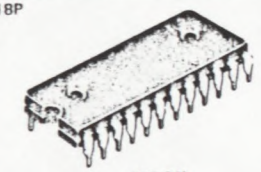
RTC (Real Time Clock Plus RAM)

The HD146818 is a HMCS6800 peripheral CMOS device which combines three unique features: a complete time-of-day clock with alarm and one hundred calendar, a programmable periodic interrupt and square-wave generator, and 50 bytes of Low-power static RAM.

This device includes HD6801, HD6301 multiplexed bus interface circuit and 8085's multiplexed bus interface as well, so it can be directly connected to HD6801, HD6301 and 8085.

The Real-Time Clock plus RAM has two distinct uses. First, it is designed as battery powered CMOS part including all the common battery backed-up functions such as RAM, time, and calender. Secondly, the HD146818 may be used with a CMOS microprocessor to relieve the software of timekeeping work-load and to extend the available RAM of an MPU such as the HD6301.

HD146818P



(DP-24)

HD146818FP

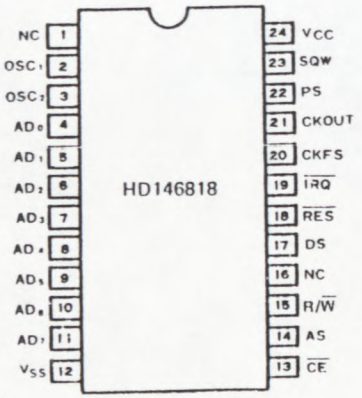
(FP-24)

The Flat Package product is under development.

■ FEATURES

- Time-of-Day Clock and Calendar
  - Counts Seconds, Minutes, and Hours of the Day
  - Counts Days of Week, Date, Month, and Year
- Binary or BCD Representation of Time, Calendar, and Alarm
- 12- or 24 Hour Clock with AM and PM in 12-Hour Mode
- Automatic End of Month Recognition
- Automatic Leap Year Compensation
- Interfaced with Software as 64 RAM Locations
  - 14 Bytes of Clock and Control Register
  - 50 Bytes of General Purpose RAM
- Three Interrupt are Separately Software Maskable and Testable
  - Time-of-Day Alarm, Once-per-Second to Once per-Day
  - Periodic Rates from 30.5μs to 500ms
  - End-of-Clock Update Cycle
- Programmable Square-Wave Output Signal
- Three Time Base Input Options
  - 4.194304 MHz
  - 1.048576 MHz
  - 32.768 kHz
- Clock Output May be used as Microprocessor Clock Input
  - At Time Base Frequency ÷4 or ÷1
- Multiplexed Bus Interface Circuit of HD6801, HD6301 and 8085
- Low-Power, High-Speed, High-Density CMOS
- Battery Backed-up Operation
- Motorola MC146818 Compatible

■ PIN ARRANGEMENT



(Top View)

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC} *	-0.3 ~ +7.0	V
Input Voltage	V_{in} *	-0.3 ~ +7.0	V
Operating Temperature	T_{opr}	0 ~ +70	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum rating are exceeded. Normal operation should be under recomended operating condition. If these conditions are exceeded, it could affect reliability of LSI.

HITACHI

■ RECOMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	V_{CC} *	4.5	5.0	5.25	V
Input Voltage	V_{IL} *	-0.3	–	0.7	V
	V_{IH} *	V_{CC}-1.0	–	V_{CC}	V
Operating Temperature	T_{opr}	0	25	70	°C

* With respect to V_{SS} (SYSTEM GND)

(NOTE) Refer to Battery Backed-up Electrical characteristics.

■ ELECTRICAL CHARACTERISTICS

• DC CHARACTERISTICS (V_{CC} = 4.5 ~ 5.25V, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	AD_0 ~ AD_7, $\overline{CE}$, AS, R/$\overline{W}$, DS, CKFS, PS	V_{IH}		V_{CC}-2.0	–	V_{CC}	V
	$\overline{RES}$			V_{CC}-1.0	–	V_{CC}	
	OSC_1			V_{CC}-1.0	–	V_{CC}	
Input "Low" Voltage	AD_0 ~ AD_7, $\overline{CE}$, AS, R/$\overline{W}$, DS, CKFS, PS	V_{IL}		-0.3	–	0.7	V
	$\overline{RES}$			-0.3	–	0.8	
	OSC_1			-0.3	–	0.8	
Input Leakage Current	OSC_1, $\overline{CE}$, AS, R/$\overline{W}$, DS, $\overline{RES}$, CKFS, PS	$\|I_{in}\|$		–	–	2.5	μA
Three-state (off state) Input Current	AD_0 ~ AD_7	$\|I_{TSI}\|$		–	–	10	μA
Output Leakage Current	$\overline{IRQ}$	I_{LOH}		–	–	10	μA
Output "High" Voltage	AD_0 ~ AD_7	V_{OH}	I_{OH} = -1.6 mA	4.1	–	–	V
	SQW, CKOUT						
	AD_0 ~ AD_7		I_{OH} < -10 μA	V_{CC}-0.1	–	–	V
	SQW, CKOUT						
Output "Low" Voltage	AD_0 ~ AD_7	V_{OL}	I_{OL} = 1.6 mA	–	–	0.5	V
	CKOUT		I_{OL} = 1.6 mA				
	$\overline{IRQ}$, SQW		I_{OL} = 1.6 mA				
Input Capacitance	AD_0 ~ AD_7	C_{in}	V_{in} = 0V, Ta = 25°C, f = 1 MHz	–	–	12.5	pF
	All inputs except AD_0 ~ AD_7			–	–	12.5	pF
Output Capacitance	SQW, CKOUT, $\overline{IRQ}$	C_{out}		–	–	12.5	pF
Supply Current (MPU Read/Write operating)	Crystal Oscillation	I_{CC} *	V_{CC} = 5.0V, SQW: disable, CKOUT = f_{OSC} (No Load), t_{cyc} = 1 μs, Circuit: Fig. 11, Parameter: Table 1	–	–	10	mA
	f_{OSC} = 4 MHz			–	–	7	
	f_{OSC} = 1 MHz			–	–	5	
Supply Current ** (MPU not operating)	f_{OSC} = 4 MHz			–	–	5	mA
	f_{OSC} = 1 MHz			–	–	2	
	f_{OSC} = 32 kHz			–	300	500	μA
Supply Current (MPU Read/Write operating)	External Clock	I_{CC} *	V_{CC} = 5.0V, SQW: disable, CKOUT = f_{OSC} (No Load), OSC_2: open, t_{cyc} = 1 μs, Circuit: Fig. 17	–	–	10	mA
	f_{OSC} = 4 MHz			–	–	7	
	f_{OSC} = 1 MHz			–	–	5	
Supply Current ** (MPU not operating)	f_{OSC} = 4 MHz			–	–	4	mA
	f_{OSC} = 1 MHz			–	–	1	
	f_{OSC} = 32 kHz			–	60	100	μA

* Supply current of HD146818 is defined as the value when the time-base frequency to be used is programmed into Register A. When power is turned on, these bits are unfixed, so there is a case that current more than the above specification may flow. Please never fail to set the time-base frequency after turning on power supply.

** $V_{IH\ min} = V_{CC}$-0.2V
$V_{IL\ max} = V_{SS}$+0.2V

HITACHI

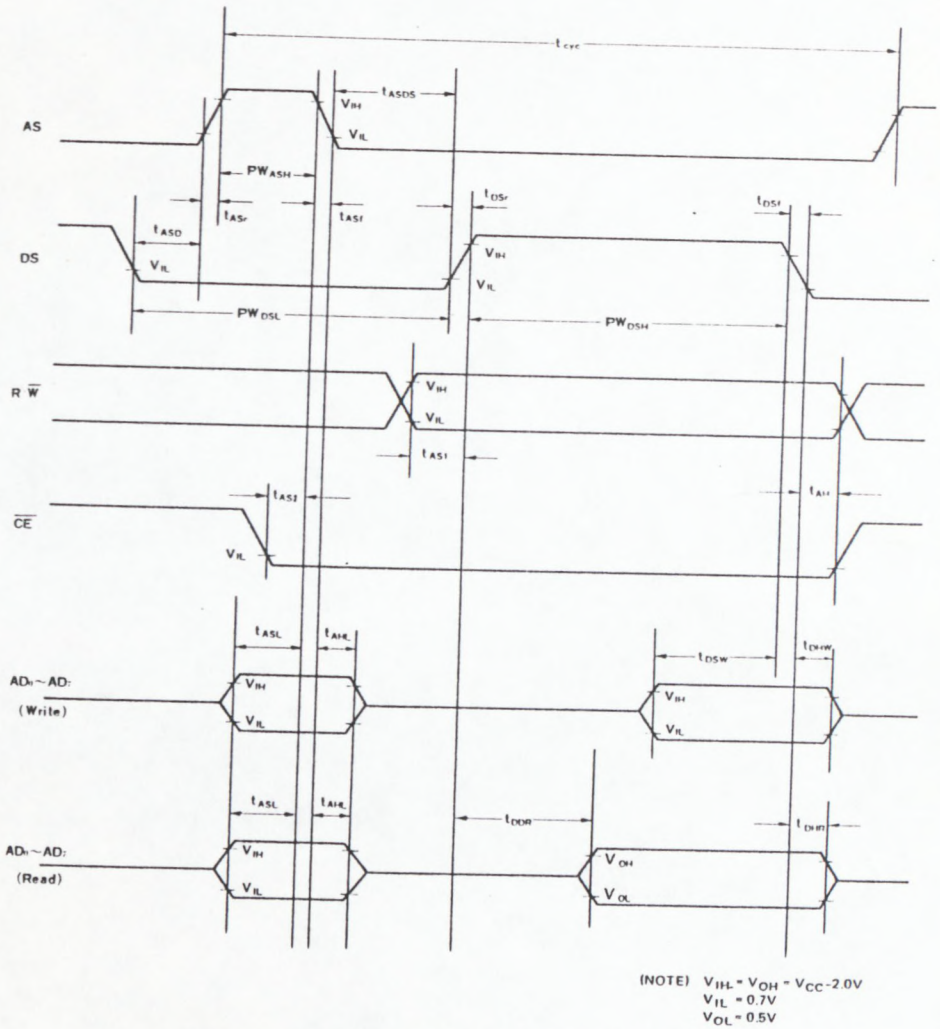
- AC CHARACTERISTICS (V_{CC} = 4.5 ~ 5.25V, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

BUS TIMING

Item	Symbol	min	typ	max	Unit
Cycle Time	t_{cyc}	953	–	–	ns
Pulse Width, AS/ALE "High"	PW_{ASH}	100	–	–	ns
AS Rise Time	t_{ASr}	–	–	30	ns
AS Fall Time	t_{ASf}	–	–	30	ns
Delay Time DS/E to AS/ALE Rise	t_{ASD}	40	–	–	ns
DS Rise Time	t_{DSr}	–	–	30	ns
DS Fall Time	t_{DSf}	–	–	30	ns
Pulse Width, DS/E Low or $\overline{RD}$/$\overline{WR}$ "High"	PW_{DSH}	325	–	–	ns
Pulse Width, DS/E High or $\overline{RD}$/$\overline{WR}$ "Low"	PW_{DSL}	300	–	–	ns
Delay Time, AS/ALE to DS/E Rise	t_{ASDS}	90	–	–	ns
Address Setup Time (R/$\overline{W}$)	t_{AS1}	15	–	–	ns
Address Setup Time ($\overline{CE}$)	t_{AS2}	55	–	–	ns
Address Hold Time (R/$\overline{W}$, $\overline{CE}$)	t_{AH}	10	–	–	ns
Muxed Address Valid Time to AS/ALE Fall	t_{ASL}	50	–	–	ns
Muxed Address Hold Time	t_{AHL}	20	–	–	ns
Peripheral Data Setup Time	t_{DSW}	195	–	–	ns
Write Data Hold Time	t_{DHW}	0	–	–	ns
Peripheral Output Data Delay Time From DS/E or $\overline{RD}$	t_{DDR}	–	–	220	ns
Read Data Hold Time	t_{DHR}	10	–	–	ns

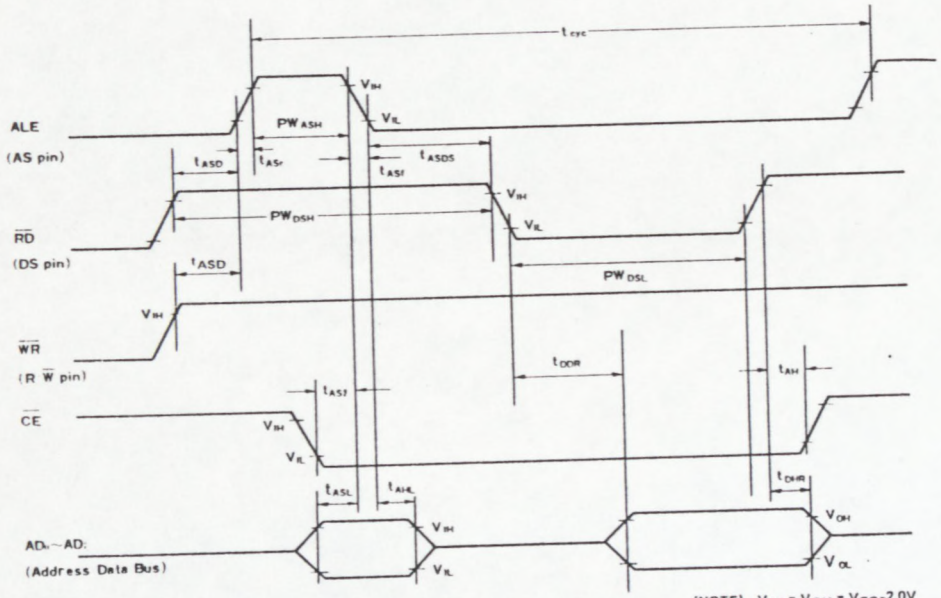
CONTROL SIGNAL TIMING

Item		Symbol	min	typ	max	Unit
Oscillator Startup	1 MHz, 4 MHz	t_{RC}	–	–	100	ms
	32 kHz		–	–	1000	
Reset Pulse Width		t_{RWL}	5.0	–	–	μs
Reset Delay Time		t_{RLH}	5.0	–	–	μs
Power Sense Pulse Width		t_{PWL}	5.0	–	–	μs
Power Sense Delay Time		t_{PLH}	5.0	–	–	μs
$\overline{IRQ}$ Release from DS		t_{IRDS}	–	–	2.0	μs
$\overline{IRQ}$ Release from $\overline{RES}$		t_{IRR}	–	–	2.0	μs
VRT Bit Delay		t_{VRTD}	–	–	2.0	μs



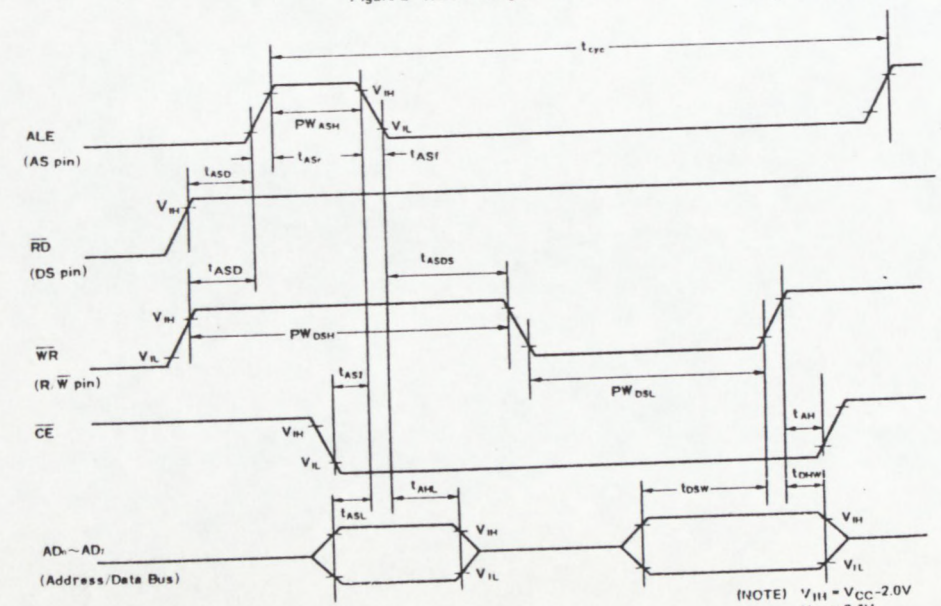
(NOTE) $V_{IHL} = V_{OH} = V_{CC}-2.0V$
$V_{IL} = 0.7V$
$V_{OL} = 0.5V$

Figure 1 Bus Read, Write Timing (6801 Family)



(NOTE) $V_{IH} = V_{OH} = V_{CC}-2.0V$
$V_{IL} = 0.7V, V_{OL} = 0.5V$

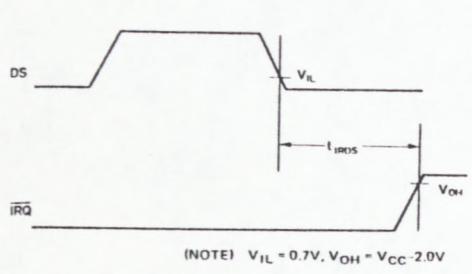
Figure 2 Read Timing (8085 Family)



(NOTE) $V_{IH} = V_{CC}-2.0V$
$V_{IL} = 0.7V$

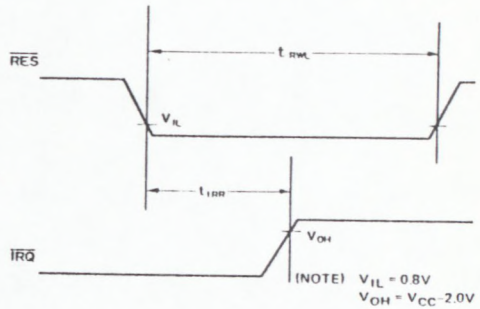
Figure 3 Write Timing (8085 Family)

HITACHI



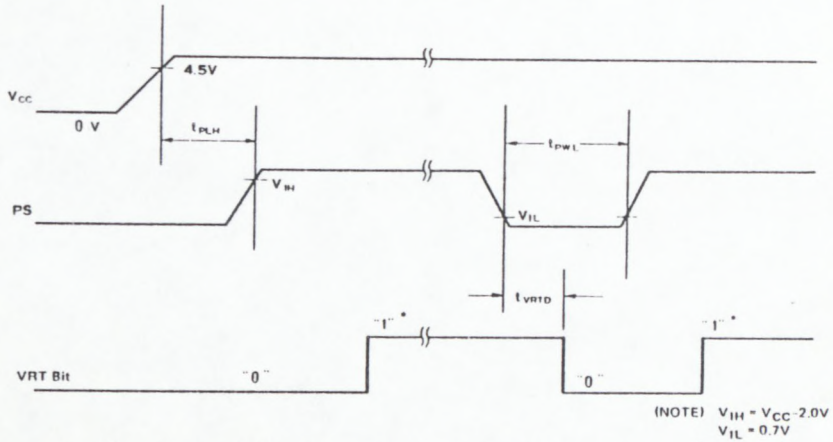
(NOTE) $V_{IL} = 0.7V, V_{OH} = V_{CC}-2.0V$

Figure 4 $\overline{IRQ}$ Release Delay (from DS)



(NOTE) $V_{IL} = 0.8V$
$V_{OH} = V_{CC}-2.0V$

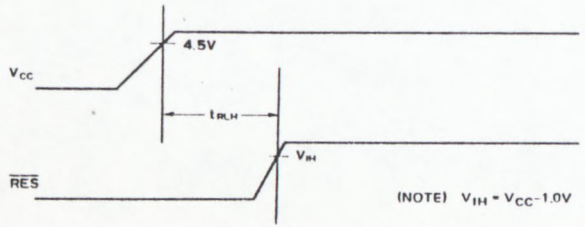
Figure 5 $\overline{IRQ}$ Release Delay (from $\overline{RES}$)



(NOTE) $V_{IH} = V_{CC}-2.0V$
$V_{IL} = 0.7V$

* The VRT bit is set to a "1" by reading control register #D. There is no additional way to clear the VRT bit.

Figure 6 VRT Bit Clear Timing



(NOTE) $V_{IH} = V_{CC}-1.0V$

Figure 7 $\overline{RES}$ Release Delay

HITACHI

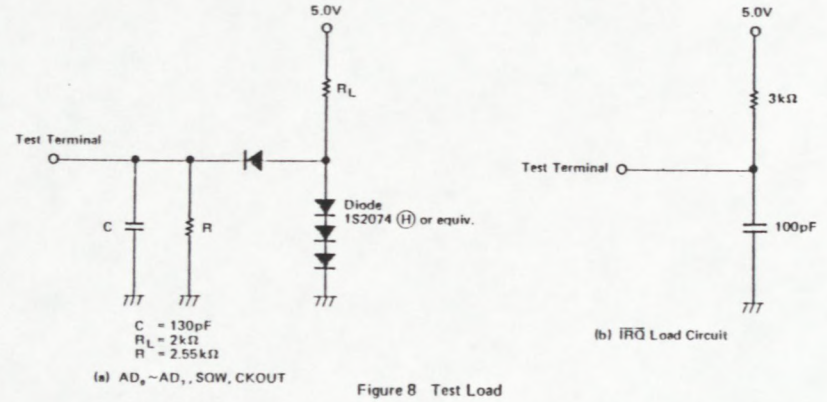


Figure 8 Test Load

- **BATTERY BACKED-UP OPERATION**
- **DEFINITION OF BATTERY BACKED-UP OPERATION**

Active functions
(1) Clock function
(2) Retention of RAM data
(3) $\overline{RES}$, $\overline{IRQ}$, CKFS, CKOUT, PS, SQW functions

Inactive functions
(1) Data bus read/write operation

- **BATTERY BACKED-UP ELECTRICAL CHARACTERISTICS (V_{SS} = 0V, T_a = 0 ~ +70°C, unless otherwise noted.)**

Item		Symbol	Test Condition		min	typ	max	Unit
Supply Voltage		V_{CCL}			2.7	–	4.5	V
Supply Current	Crystal Oscillation	I_{CCL}*	V_{CCL} = 3.0V, SQW : disable, CKOUT: fosc (No load)	4MHz	–	–	600	μA
				1MHz	–	–	350	μA
				32kHz	–	50	100	μA
	External Clock		V_{CCL} = 3.0V, SQW : disable, CKOUT: fosc (No load)	4MHz	–	–	500	μA
				1MHz	–	–	150	μA
				32kHz	–	30	70	μA
Battery Backed-up Transit Setup Time		t_{CE}	Fig. 9		0	–	–	ns
Operation Recovery Time		t_R			t_{cyc}	–	–	ns
Supply Voltage Fall Time		t_{Pf}			300	–	–	μs
Supply Voltage Rise Time		t_{Pr}			300	–	–	μs
Input "High" Voltage		V_{IHL}	V_{CCL} = 2.7V ~ 3.5V	$\overline{CE}$, PS, CKFS	0.7 x V_{CCL}	–	V_{CCL}	V
			V_{CCL} = 3.5V ~ 4.5V		2.5	–	V_{CCL}	V
				$\overline{RES}$	0.8 x V_{CCL}	–	V_{CCL}	V
				OSC_1	0.8 x V_{CCL}	–	V_{CCL}	V
Input "Low" Voltage		V_{ILL}		CKFS, PS	-0.3	–	0.5	V
				$\overline{RES}$	-0.3	–	0.5	V
				OSC_1	-0.3	–	0.5	V
Output "High" Voltage		V_{OHL}	I_{OH} = -800μA	SQW, CKOUT	0.8 x V_{CCL}	–	–	V
Output "Low" Voltage		V_{OLL}	I_{OL} = 800μA	SQW, CKOUT	–	–	0.5	V
				$\overline{IRQ}$	–	–	0.5	V

* The time-base frequency to be used needs to be chosen in Register A.

HITACHI

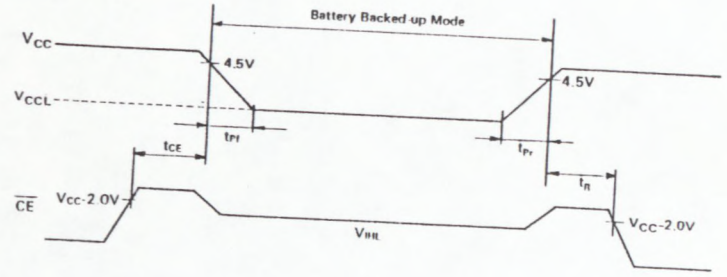


Figure 9 Battery Backed-up Timing

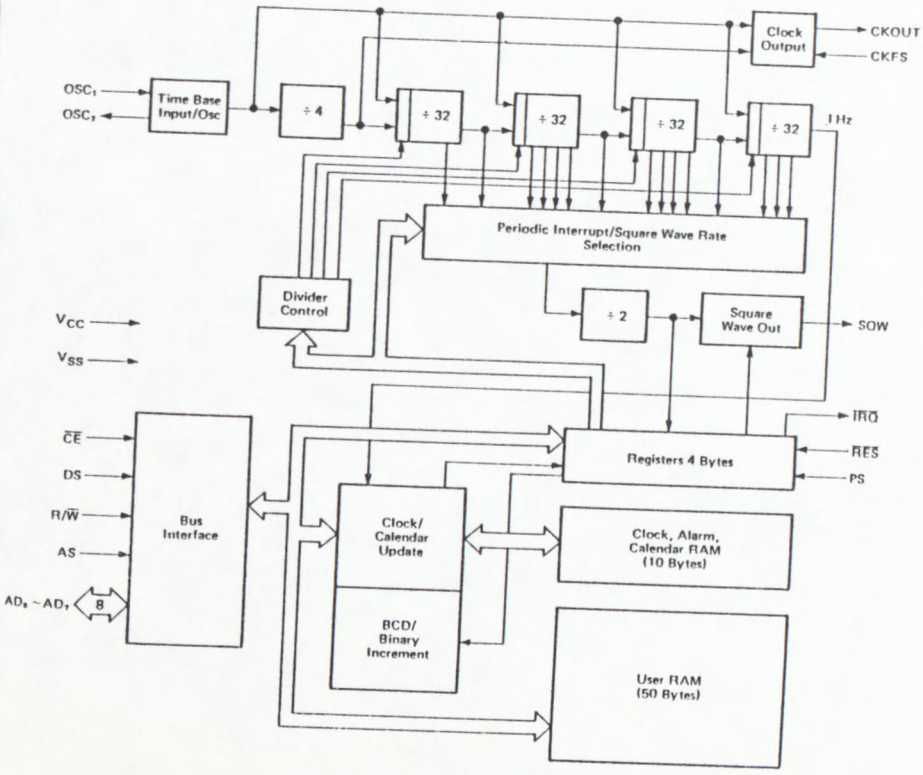


Figure 10 Block Diagram

HITACHI

■ CRYSTAL OSCILLATION CIRCUIT

The on-chip oscillator is designed for a parallel resonant crystal at 4.194304 MHz or 1.048576 MHz or 32.768 kHz frequencies. The crystal connections are shown in Figure 11.

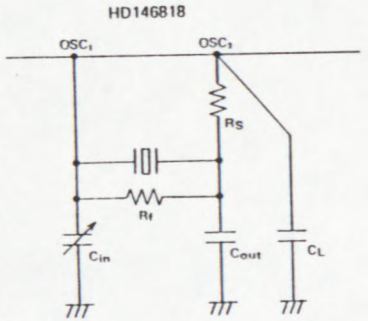


Figure 11 Crystal Oscillator Connection

Table 1 Oscillator Circuit Parameters

Parameter \ fosc	4.194304 MHz	1.048576 MHz	32.768 kHz
R_S	–	–	150 kΩ
R_f	150 kΩ	150 kΩ	5.6 MΩ
C_{in}	22 pF	33 pF	15 pF
C_{out}	22 pF	33 pF	33 pF
C_L	–	–	33 pF
CI	80 Ω (max)	700 Ω (max)	40 kΩ (max)

(NOTE) 1. R_S, C_L are used for 32.768 kHz only.
2. Capacitance (C_{in}) should be adjusted to accurate frequenc
Parameters listed above are applied to the supply curre
measurement (See table of DC CHARACTERISTICS).
3. CI: Crystal Impedance

■ NOTE FOR BOARD DESIGN OF THE OSCILLATION CIRCUIT

In designing the board, the following notes should be taken when the crystal oscillator is used.

(1) Crystal oscillator, load capacity C_{in}, C_{out}, C_L and R_f, R_S must be placed near the LSI as much as possible.
(Normal oscillation may be disturbed when external noise is induced to pin 2 and 3.)

(2) Pin 3 signal line should be wired apart from pin 4 sign line as much as possible. Don't wire them in parallel, normal oscillation may be disturbed when this signal feedbacked to OSC_1.

(3) A signal line or a power source line must not cross or near the oscillation circuit line as shown in the right figu to prevent the induction from these lines and perform th correct oscillation. The resistance among OSC_1, OSC_2 a other pins should be over 10MΩ.

The following design must be avoided.

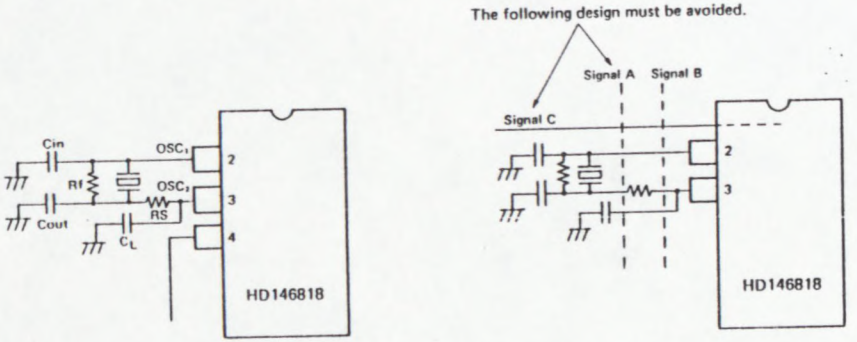


Figure 12 Note for Board Design of the Oscillation Circuit

■ INTERFACE CIRCUIT FOR HD6801, HD6301 AND 8085 PROCESSOR

HD146818 has a new interface circuit which permits the HD146818 to be directly interfaced with many type of multiplexed bus microprocessor such as HD6801, HD6301 and 8085 etc.

Figure 13 shows the bus control circuit. This circuit automatically selects the processor type by using AS/ASE to latch the state of DS/$\overline{RD}$ pin. Since DS is always "Low" and $\overline{RD}$ is always "High during AS/ALE, the latch automatically indicates which processor type is connected.

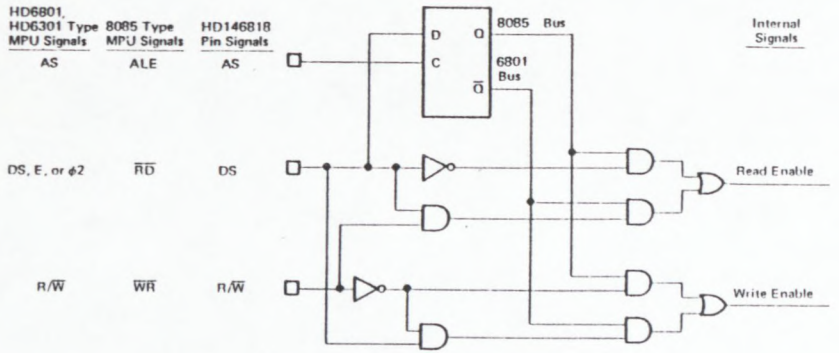


Figure 13 Functional Diagram of the Bus Control Circuit

■ ADDRESS MAP

Figure 14 shows the address map of the HD146818. The memory consists of 50 general purpose RAM bytes, 10 RAM bytes which normally contain the time, calendar, and alarm data, and four control and status bytes. All 64 bytes are directly readable and writable by the processor program except Registers C and D which are read only. Bit 7 of Register A and the seconds byte are also read only. Bit 7, of the second byte, always reads "0". The contents of the four control and status registers are described in the Register section.

• Time, Calendar, and Alarm Locations

The processor program obtains time and calendar information by reading the appropriate locations. The program may initialize the time, calendar, and alarm by writing to these RAM locations. The contents of the 10 time, calendar, and alarm byte may be either binary or binary-coded decimal (BCD).

Before initializing the internal registers, the SET bit in Register B should be set to a "1" to prevent time/calendar updates from occurring. The program initializes the 10 locations in the selected format (binary or BCD), then indicates the format in the data mode (DM) bit of Register B. All 10 time, calendar, and alarm bytes must use the same data mode, either binary or BCD. The SET bit may now be cleared to allow updates. Once initialized the real-time clock makes all updates in the selected data mode. The data mode cannot be changed without reinitializing the 10 data bytes.

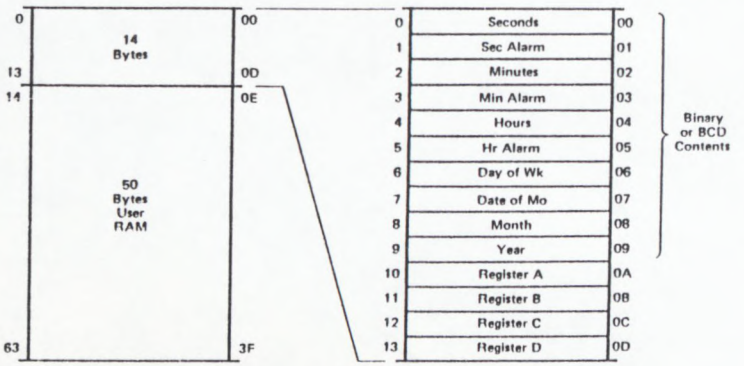


Figure 14 Address Map

Table 2 shows the binary and BCD formats of the 10 time, calendar, and alarm locations. The 24/12 bit in Register B establishes whether the hour locations represent 1-to-12 or 0-to-23. The 24/12 bit cannot be changed without reinitializing the hour locations. When the 12-hour format is selected the high-order bit of the hours byte represents PM when it is a "1".

The time, calendar, and alarm bytes are not always accessable by the processor program. Once-per-second the 10 bytes are switched to the update logic to be advanced by one second and to check for an alarm condition. If any of the 10 bytes are read at this time, the data outputs are undefined. The update lockout time is 248 μs at the 4.194304 MHz and 1.048567 MHz time bases and 1948 μs for the 32.768 kHz time base. The Update Cycle section shows how to accommodate the update cycle in the processor program.

The three alarm bytes may be used in two ways. Wh the program inserts an alarm time in the appropriate hou minutes, and seconds alarm locations, the alarm interrupt initiated at the specified time each day if the alarm enable b is "1". The alternate usage is to insert a "don't care" state one or more of three alarm bytes. The "don't care" code any hexadecimal byte from C0 to FF. That is, the two mo significant bits of each byte, when set to "1", create a "don care" situation. An alarm interrupt each hour is created wi a "don't care" code in the hours alarm location. Similarly, a alarm is generated every minute with "don't care" codes the hours and minutes alarm bytes. The "don't care" cod in all three alarm bytes create an interrupt every second.

Table 2 Time, Calendar, and Alarm Data Modes

Address Location	Function	Decimal Range	Range		Example*	
			Binary Data Mode	BCD Data Mode	Binary Data Mode	BCD Data Mode
0	Seconds	0 ~ 59	$00 ~ $3B	$00 ~ $59	15	21
1	Seconds Alarm	0 ~ 59	$00 ~ $3B	$00 ~ $59	15	21
2	Minutes	0 ~ 59	$00 ~ $3B	$00 ~ $59	3A	58
3	Minutes Alarm	0 ~ 59	$00 ~ $3B	$00 ~ $59	3A	58
4	Hours (12 Hour Mode)	1 ~ 12	$01~$0C (AM) and $81 ~ $8C (PM)	$01 ~ $12 (AM) and $81 ~ $92 (PM)	05	05
	Hours (24 Hour Mode)	0 ~ 23	$00 ~ $17	$00 ~ $23	05	05
5	Hours Alarm (12 Hour Mode)	1 ~ 12	$01 ~ $0C (AM) and $81 ~ $8C (PM)	$01 ~ $12 (AM) and $81 ~ $92 (PM)	05	05
	Hours Alarm (24 Hour Mode)	0 ~ 23	$00 ~ $17	$00 ~ $23	05	05
6	Day of the Week Sunday = 1	1 ~ 7	$01 ~ $07	$01 ~ $07	05	05
7	Day of the Month	1 ~ 31	$01 ~ $1F	$01 ~ $31	0F	15
8	Month	1 ~ 12	$01 ~ $0C	$01 ~ $12	02	02
9	Year	0 ~ 99**	$00 ~ $63	$00 ~ $99	4F	79

* Example: 5:58:21 Thursday 15th February 1979

** Set the lower two digits of year in AD. If this number is multiple of 4, update applied to leap year is excuted.

• Static CMOS RAM

The 50 general purpose RAM bytes are not dedicated within the HD146818. They can be used by the processor program, and are fully available during the update cycle.

When time and calendar information must use battery back-up, very frequently there is other non-volatile data that must be retained when main power is removed. The 50 user RAM bytes serve the need for low-power CMOS battery-backed storage, and extend the RAM available to the program.

When further CMOS RAM is needed, additional HD146818s may be included in the system. The time/calendar functions may be disabled by holding the dividers, in Register A, in the reset state by setting the SET bit in Register B or by removing the oscillator. Holding the dividers in reset prevents interrupts or SQW output from operating while setting the SET bit allows these functions to occur. With the dividers clear, the available user RAM is extended to 59 bytes. Bit 7 of Register A, Registers C and D, and the high-order Bit of the seconds byte cannot effectively be used as general purpose RAM.

■ INTERRUPTS

The RTC plus RAM includes three separate fully automat sources of interrupts to the processor. The alarm interru may be programmed to occur at rates from once-per-secon to one-a-day. The periodic interrupt may be selected for rat from half-a-second to 30.517 μs. The update-ended interru may be used to indicate to the program that an up-date cyc is completed. Each of these independent interrupt condition are described in greater detail in other sections.

The processor program selects which interrupts, if any, wishes to receive. Three bits in Register B enable the thr interrupts. Writing a "1" to a interrupt-enable bit permi that interrupt to be initiated when the event occurs. A "0" the interrupt-enable bit prohibits the $\overline{IRQ}$ pin from bein asserted due to the interrupt cause.

If an interrupt flag is already set when the interrupt becom enabled, the $\overline{IRQ}$ pin is immediately activated, though th interrupt initiating the event may have occurred much earlie Thus, there are cases where the program should clear su earlier initiated interrupts before first enabling new interrupts.

When an interrupt event occurs a flag bit is set to a "1" in Register C. Each of the three interrupt sources have separate flag bits in Register C, which are set independent of the state of the corresponding enable bits in Register B. The flag bit may be used with or without enabling the corresponding enable bits.

In the software scanned case, the program does not enable the interrupt. The "interrupt" flag bit becomes a status bit, which the software interrogates, when it wishes. When the software detects that the flag is set, it is an indication to software that the "interrupt" event occurred since the bit was last read.

However, there is one precaution. The flag bits in Register C are cleared (record of the interrupt event is erased) when Register C is read. Double latching is included with Register C so the bits which are set are stable throughout the read cycle. All bits which are high when read by the program are cleared, and new interrupts (on any bits) are held until after the read cycle. One, two, or three flag bits may be found to be set when Register C is read. The program should inspect all utilized flag bits every time Register C is read to insure that no interrupts are lost.

The second flag bit usage method is with fully enabled interrupts. When an interrupt-flag bit is set and the corresponding interrupt-enable bit is also set, the $\overline{IRQ}$ pin is asserted "Low". $\overline{IRQ}$ is asserted as long as at least one of the three interrupt sources has its flag and enable bits both set. The IRQF bit in Register C is a "1" whenever the $\overline{IRQ}$ pin is being driven "Low".

The processor program can determine that the RTC initiated the interrupt by reading Register C. A "1" in bit 7 (IRQF bit) indicates that one of more interrupts have been initiated by the part. The act of reading Register C clears all the then-active flag bits, plus the IRQF bit. When the program finds IRQF set, it should look at each of the individual flag bits in the same byte which have the corresponding interrupt-mask bits set and service each interrupt which is set. Again, more than one interrupt-flag bit may be set.

■ DIVIDER STAGES

The HD146818 has 22 binary-divider stages following the time base as shown in Figure 10. The output of the dividers is a 1 Hz signal to the update-cycle logic. The dividers are controller by three divider bus (DV2, DV1, and DV0) in Register A.

• Divider Control

The divider-control bits have three uses, as shown in Table 3. Three usable operating time bases may be selected (4.194304 MHz, 1.048576 MHz, or 32.768 kHz). The divider chain may be held reset, which allows precision setting of the time. When the divider is changed from reset to an operating time base, the first update cycle is one second later. The divider-control bits are also used to facilitate testing the HD146818.

Table 3 Divider Configurations

Time-Base Frequency	Divider Bits Register A			Operation Mode	Divider Reset	Bypass First N-Divider Bits
	DV2	DV1	DV0			
4.194304 MHz	0	0	0	Yes	–	N = 0
1.048576 MHz	0	0	1	Yes	–	N = 2
32.768 kHz	0	1	0	Yes	–	N = 7
Any	1	1	0	No	Yes	–
Any	1	1	1	No	Yes	–

(NOTE) Other combinations of divider bits are used for test purposes only.

• Square-Wave Output Selection

Fifteen of the 22 divider taps are made available to a 1-of-15 selector as shown in Figure 10. The first purpose of selecting a divider tap is to generate a square-wave output signal in the SQW pin. Four bits in Register A establish the square-wave frequency as listed in Table 4. The SQW frequency selection shares the 1-of-15 selector with periodic interrupts.

Once the frequency is selected, the output of the SQW pin may be turned on and off under program control with the square-wave enable (SQWE) bit in Register B. Altering the divider, square-wave output selection bits, or the SQW output-enable bit may generate an asymetrical waveform at the time of execution. The square-wave output pin has a number of potential uses. For example, it can serve as a frequency standard for external use, a frequency synthesizer, or could be used to generate one or more audio tones under program control.

• Periodic Interrupt Selection

The periodic interrupt allows the $\overline{IRQ}$ pin to be triggered from once every 500 ms to once every 30.517 μs. The periodic interrupt is separate from the alarm interrupt which may be output from once-per-second to once-per-day.

Table 4 shows that the periodic interrupt rate is selected with the same Register A bits which select the square-wave frequency. Changing one also changes the other. But each function may be separately enabled so that a program could switch between the two features or use both. The SQW pin is enabled by the SQWE bit. Similarly the periodic interrupt is enabled by the PIE bit in Register B.

Periodic interrupt is usable by practically all real-time systems. It can be used to scan for all forms of input from contact closures to serial receive bits or tyes. It can be used in multiplexing displays or with software counters to measure inputs, create output intervals, or await the next needed software function.

Table 4 Periodic Interrupt Rate and Square Wave Output Frequency

Rate Select Control Register 1				4.194304 or 1.048576 MHz Time Base		32.768 kHz Time Base	
RS3	RS2	RS1	RS0	Periodic Interrupt Rate t_{PI}	SQW Output Frequency	Periodic Interrupt Rate t_{PI}	SQW Output Frequency
0	0	0	0	None	None	None	None
0	0	0	1	30.517 μs	32.768 kHz	3.90625 ms	256 Hz
0	0	1	0	61.035 μs	16.384 kHz	7.8125 ms	128 Hz
0	0	1	1	122.070 μs	8.192 kHz	122.070 μs	8.192 kHz
0	1	0	0	244.141 μs	4.096 kHz	244.141 μs	4.096 kHz
0	1	0	1	488.281 μs	2.048 kHz	488.281 μs	2.048 kHz
0	1	1	0	976.562 μs	1.024 kHz	976.562 μs	1.024 kHz
0	1	1	1	1.953125 ms	512 Hz	1.953125 ms	512 Hz
1	0	0	0	3.90625 ms	256 Hz	3.90625 ms	256 Hz
1	0	0	1	7.8125 ms	128 Hz	7.8125 ms	128 Hz
1	0	1	0	15.625 ms	64 Hz	15.625 ms	64 Hz
1	0	1	1	31.25 ms	32 Hz	31.25 ms	32 Hz
1	1	0	0	62.5 ms	16 Hz	62.5 ms	16 Hz
1	1	0	1	125 ms	8 Hz	125 ms	8 Hz
1	1	1	0	250 ms	4 Hz	250 ms	4 Hz
1	1	1	1	500 ms	2 Hz	500 ms	2 Hz

- **Initialization of the Time and the Start Sequence**

The first update of the time occurs about 500ms later after the SET bit of control register B is reset. So keep followings in mind when initializing and adjusting the time.

Procedure of time initialization

(1) Set the SET bit of control register B. (SET = "1")
(2) Set "1" into all the DV0, 1, 2 bits of control register A. (DV0 = DV1 = DV2 = "1")
(3) Set the time and calendar to each RAM.
(4) Set the frequency in use into DV0, 1 and DV2.
(5) Reset the SET bit. (SET = "0")

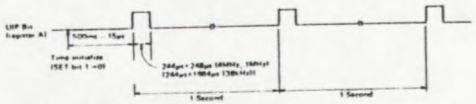


Figure 15 Time Initialization and the First Update

Restriction on Time-of-day and Calendar Initialization

There is a case in HD146818 (RTC) that update is not executed correctly if time of day and calendar shown below are initialized. Therefore, initialize the RTC without using time of day shown below.

Calendar, Time of day & Status after Update	Examples
If 29th 23:59:59 in all the months is initialized, update to 1st in the next month is executed. (Jan. – Dec. However except for Feb. 29th in leap year)	Mar. 29th → Apr. 1st
If 30th 23:59:59 in Apr., June, Sept., and Nov. is initialized, update to 31st in each month is executed.	Apr. 30th → Apr. 31st
If Feb. 28th 23:59:59 (not in leap year) is initialized, update to Feb. 29th is executed.	Feb. 28th, 1983 → Feb. 29th, 1983
If Feb. 28th 23:59:58 (in leap year) is initialized, update to Mar. 1st is executed.	Feb. 28th, 1984 → Mar. 1st, 1984

■ UPDATE CYCLE

The HD146818 executes an update cycle once-per-second assuming one of the proper time bases is in place, the divider is not clear, and the SET bit in Register B is clear. The SET bit in the "1" state permits the program to initialize the time and calendar bytes by stopping an existing update and preventing a new one from occurring.

The primary function of the update cycle is to increment the seconds byte, check for overflow, increment the minutes byte when appropriate and so forth through to the year in the century byte. The update cycle also compares each alarm byte with the corresponding time byte and issues an alarm if a match or if a "don't care" code (11XXXXXX) is present in all three positions.

With a 4.194304 MHz or 1.048576 MHz time base the update cycle takes 248 μs while a 32.768 kHz time base update cycle takes 1984 μs. During the update cycle, the time, calendar, and alarm bytes are not accessable by the processor program. The HD146818 protects the program from reading transitional data. This protection is provided by switching the time, calendar, and alarm portion of the RAM off the microprocessor bus during the entire update cycle. If the processor reads these RAM locations before the update is complete the output will be undefined. The update in progress (UIP) status bit is set during the interval.

A program which randomly accesses the time and date information finds data unavailable statistically once every 4032 attempts. Three methods of accommodating nonavailability during update are usable by the program. In discussing the three methods it is assumed that at random points user programs are able to call a subroutine to obtain the time of day.

The first method of avoiding the update cycle uses the update-ended interrupt. If enabled, an interrupt occurs after every update cycle which indicates that over 999 ms are available to read valid time and date information. During this time a display could be updated or the information could be transfered to continuously available RAM. Before leaving the interrupt service routine, the IRQF bit in Register C should be cleared.

The second method uses the update-in-progress bit (UIP) in Register A to determine if the update cycle is in progress or not. The UIP bit will pulse once-per-second. Statistically, the UIP bit will indicate that time and date information is unavailable once every 2032 attempts. After the UIP bit goes "1", the update cycle begins 244 μs later. Therefore, if a "0" is read on the UIP bit, the user has at least 244 μs before the time/calendar data will be changed. If a "1" is read in the UIP bit, the time/calendar data may not be valid. The user should avoid interrupt service routines that would cause the time needed to read valid time/calendar data to exceed 244 μs.

The third method uses a periodic interrupt to determine if an update cycle is in progress. The UIP bit in Register A is set "1" between the setting of the PF bit in Register C (see Figure 16) Periodic interrupts that occur at a rate of greater than $t_{BUC} + t_{UC}$ allow valid time and date information to be read at each occurrence of the periodic interrupt. The reads should be completed within $(t_{PI} \div 2) + t_{BUC}$ to insure that data is not read during the update cycle.

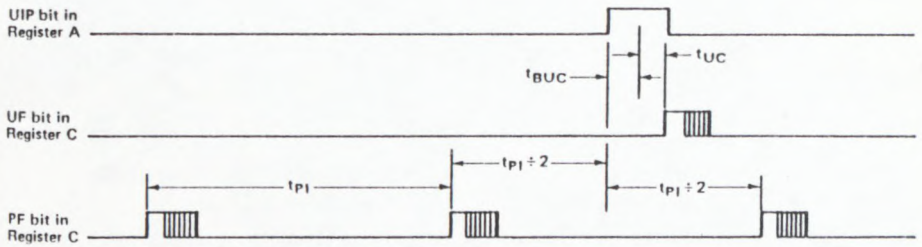
■ POWER-DOWN CONSIDERATIONS

In most systems, the HD146818 must continue to keep time when system power is removed. In such systems, a conversion from system power to an alternate power supply, usually a battery, must be made. During the transition from system to battery power, the designer of a battery backed-up RTC system must protect data integrity, minimize power consumption, and ensure hardware reliability according to the specification described in the section regarding Battery Backed-up operation.

The chip enable ($\overline{CE}$) pin controls all bus inputs (R/$\overline{W}$, DS, AS, $AD_0 \sim AD_7$). $\overline{CE}$, when negated, disallows any unintended modification of the RTC data by the bus. $\overline{CE}$ also reduces power consumption by reducing the number of transitions seen internally.

Power consumption may be further reduced by removing resistive and capacitive loads from the clock out (CKOUT) pin and the squarewave (SQW) pin.

During and after the power source conversion, the V_{IN} maximum specification must never be exceeded. Failure to meet the V_{IN} maximum specification can cause a virtual SCR to appear which may result in excessive current drain and destruction of the part.



t_{PI} = Periodic Interrupt Time Interval (500 ms, 250 ms, 125 ms, 62.5 ms, etc.)
t_{UC} = Update Cycle Time (248 μs or 1984 μs)
t_{BUC} = Delay Time Before Update Cycle (244 μs)

Figure 16 Update-Ended and Periodic Interrupt Relationship

■ SIGNAL DESCRIPTIONS

The block diagram in Figure 10, shows the pin connection with the major internal functions of the HD146818 Real-Time Clock plus RAM. The following paragraphs describe the function of each pin.

• V_{CC}, V_{SS}

DC power is provided to the part on these two pins, V_{CC} being the most positive voltage. The minimum and maximum voltages are listed in the Electrical Characteristics tables.

• OSC_1, OSC_2 – Time Base (Inputs)

The time base for the time functions may be an external signal or the crystal oscillator. External square waves at 4.194304 MHz, 1.048576 MHz, or 32.768 kHz may be connected to OSC_1 as shown in Figure 17. The time-base frequency to be used is chosen in Register A.

The on-chip oscillator is designed for a parallel resonant crystal at 4.194304 MHz or 1.048576 MHz or 32.768 kHz frequencies. The crystal connections are shown in Figure 11.

• CKOUT – Clock Out (Output)

The CKOUT pin is an output at the time-base frequency divided by 1 or 4. A major use for CKOUT is as the input clock to the microprocessor; thereby saving the cost of a second crystal. The frequency of CKOUT depends upon the time-base frequency and the state of the CKFS pin as shown in Table 5.

• CKFS – Clock Out Frequency Select (Input)

The CKOUT pin is an output at the time-base frequency divided by 1 or 4. CKFS tied to V_{CC} causes CKOUT to be the same frequency as the time base at the OSC_1 pin. When CKFS is at V_{SS}, CKOUT is the OSC_1 time-base frequency divided by four. Table 5 summarizes the effect of CKFS.

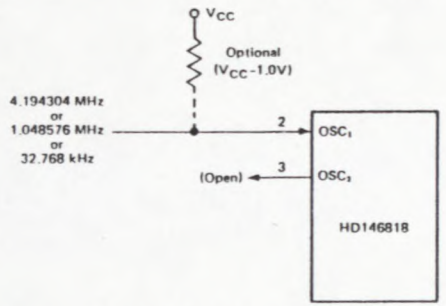


Figure 17 External Time-Base Connection

Table 5 Clock Output Frequencies

Time Base (OSC_1) Frequency	Clock Frequency Select Pin (CKFS)	Clock Frequency Output Pin (CKOUT)
4.194304 MHz	"High"	4.194304 MHz
4.194304 MHz	"Low"	1.048576 MHz
1.048576 MHz	"High"	1.048576 MHz
1.048576 MHz	"Low"	262.144 kHz
32.768 kHz	"High"	32.768 kHz
32.768 kHz	"Low"	8.192 kHz

• SQW – Square Wave (Output)

The SQW pin can output a signal one of 15 of the 22 internal-divider stages. The frequency and output enable of the SQW may be altered by programming Register A, as shown in Table 4. The SQW signal may be turned on and off using a bit in Register B.

• $AD_0 \sim AD_7$ – Multiplexed Bidirectional Address/Data Bus

Multiplexed bus processors save pins by presenting the address during the first portion of the bus cycle and using the same pins during the second portion for data. Address-then-data multiplexing does not slow the access time of the HD146818 since the bus reversal from address to data is occurring during the internal RAM access time.

The address must be valid just prior to the fall of AS/ALE at which time the HD146818 latches the address from AD_0 to AD_5. Valid write data must be presented and held stable during the latter portion of the DS or $\overline{WR}$ pulses. In a read cycle, the HD146818 outputs 8 bits of data during the latter portion of the DS or $\overline{RD}$ pulses, then ceases driving the bus (returns the output drivers to three-state) when DS falls in the HD6801, HD6301 case or $\overline{RD}$ rises in the other case.

• AS – Multiplexed Address Strobe (Input)

A positive going multiplexed address strobe pulse serves to demultiplex the bus. The falling edge of AS or ALE causes the address to be latched within the HD146818. The bus control circuit in the HD146818 also latches the state of the DS pin with the falling edge of AS or ALE.

• DS – Data Strobe or Read (Input)

The DS pin has two interpretations via the bus control circuit. When emanating from 6801 family type processor, DS is a positive pulse during the latter portion of the bus cycle, and is variously called DS (data strobe), E (enable), and ϕ_2 (ϕ_2 clock). During read cycles, DS signifies the time that the RTC is to drive the bidirectional bus. In write cycles, the trailing edge of DS causes the Real-Time Clock plus RAM to latch the written data.

The second interpretation of DS is that of $\overline{RD}$, $\overline{MEMR}$, or $\overline{I/OR}$ emanating from the 8085 type processor. In this case, DS identifies the time period when the real-time clock plus RAM drives the bus with read data. This interpretation of DS is also the same as an output-enable signal on a typical memory.

The bus control circuit, within the HD146818, latches the state of the DS pin on the falling edge of AS/ALE. When 6801 mode, DS must be "Low" during AS/ALE, which is the case with 6801 family multiplexed bus processors. To insure the 8085 mode of this circuit the DS pin must remain "High" during the time AS/ALE is "High".

• R/$\overline{W}$ – Read/Write (Input)

The bus control circuit treats the R/$\overline{W}$ pin in one of two ways. When 6801 family type processor is connected, R/$\overline{W}$ is a level which indicates whether the current cycle is a read or write. A read cycle is indicated with a "High" level on R/$\overline{W}$ while DS is "High", whereas a write cycle is a "Low" on R/$\overline{W}$ during DS.

The second interpretation of R/$\overline{W}$ is as a negative write pulse, $\overline{WR}$, $\overline{MEMW}$, and $\overline{I/OW}$ from 8085 type processors. This circuit in this mode gives R/$\overline{W}$ pin the same meaning as the write ($\overline{W}$) pulse on many generic RAMs.

• $\overline{CE}$ – Chip Enable (Input)

The chip-enable ($\overline{CE}$) signal must be asserted ("Low") for a bus cycle in which the HD146818 is to be accessed. $\overline{CE}$ is not latched and must be stable during DS and AS (in the 6801 case) and during $\overline{RD}$ and $\overline{WR}$ (in the 8085 case). Bus cycles which take place without asserting $\overline{CE}$ cause no actions to take place within the HD146818. When $\overline{CE}$ is "High", the multiplexed bus output is in a high-impedance state.

When $\overline{CE}$ is "High", all address, data, DS, and R/$\overline{W}$ inputs from the processor are disconnected within the HD146818. This permits the HD146818 to be isolated from a powered-down processor. When $\overline{CE}$ is held "High", an unpowered device cannot receive power through the input pins from the real-time clock power source. Battery power consumption can thus be reduced by using a pullup resistor or active clamp on $\overline{CE}$ when the main power is off.

• $\overline{IRQ}$ – Interrupt Request (Output)

The $\overline{IRQ}$ pin is an active "Low" output of the HD146818 that may be used as an interrupt input to a processor. The $\overline{IRQ}$ output remains "Low" as long as the status bit causing the interrupt is present and the corresponding interrupt-enable bit is set. To clear the $\overline{IRQ}$ pin, the processor program normally reads Register C. The $\overline{RES}$ pin also clears pending interrupts.

When no interrupt conditions are present, the $\overline{IRQ}$ level is in the high-impedance state. Multiple interrupting devices may thus be connected to an $\overline{IRQ}$ bus with one pullup at the processor.

• $\overline{RES}$ – Reset (Input)

The $\overline{RES}$ pin does not affect the clock, calendar, or RAM functions. On powerup, the $\overline{RES}$ pin must be held "Low" for the specified time, t_{RLH}, in order to allow the power supply to stabilize, Figure 18 shows a typical representation of the $\overline{RES}$ pin circuit.

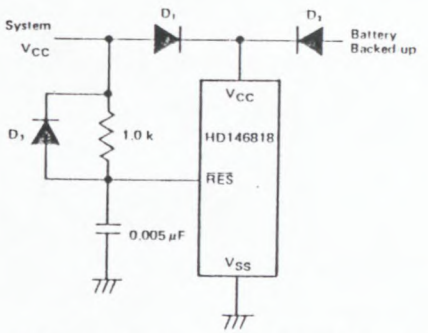
When $\overline{RES}$ is "Low" the following occurs:

a) Periodic Interrupt Enable (PIE) bit is cleared to "0".
b) Alarm Interrupt Enable (AIE) bit is cleared to "0".
c) Update ended interrupt Enable (UIE) bit is cleared to "0".
d) Update ended Interrupt Flag (UF) bit is cleared to "0".
e) Interrupt Request status Flag (IRQF) bit is cleared to "0".
f) Periodic Interrupt Flag (PF) bit is cleared to "0".
g) Alarm Interrupt Flag (AF) bit is cleared to "0".
h) $\overline{IRQ}$ pin is in high-impedance state, and
i) Square Wave output Enable (SQWE) bit is cleared to "0".

• PS – Power Sense (Input)

The power-sense pin is used in the control of the valid RAM and time (VRT) bit in Register C. When the PS pin is "Low" the VRT bit is cleared to "0".

During powerup, the PS pin must be externally held "Low" for the specified time, t_{PLH}. As power is applied the VRT bit remain "Low" indicating that the contents of the RAM, time registers, and calendar are not guaranteed. When normal operation commences PS should be permitted to go "High". Output signal from external power sence circuit will be connected to this input.



(NOTE) If the RTC is isolated from the MPU or MCU power by a diode drop, care must be taken to meet V_{in} requirements.

Figure 18 Typical Powerup Delay Circuit for $\overline{RES}$

■ REGISTERS

The HD146818 has four registers which are accessible to the processor program. The four registers are also fully accessible during the update cycle.

• Register A ($0A)

MSB							LSB	
b7	b6	b5	b4	b3	b2	b1	b0	Read/Write Register except UIP
UIP	DV2	DV1	DV0	RS3	RS2	RS1	RS0	

UIP – The update in progress (UIP) bit is a status flag that may be monitored by the program. When UIP is a "1" the update

Table 6 Update Cycle Times

UIP Bit	Time Base (OSC_1)	Update Cycle Time (t_{UC})	Minimum Time Before Update Cycle (t_{BUC})
1	4.194304 MHz	248 μs	–
1	1.048576 MHz	248 μs	–
1	32.768 kHz	1984 μs	–
0	4.194304 MHz	–	244 μs
0	1.048576 MHz	–	244 μs
0	32.768 kHz	–	244 μs

cycle is in progress or will soon begin. When UIP is a "0" the update cycle is not in progress and will not be for at least 244 μs (for all time bases). This is detailed in Table 6. The time, calendar, and alarm information in RAM is fully available to the program when the UIP bit is zero – it is not in transition. The UIP bit is a read-only bit, and is not affected by Reset. Writing the SET bit in Register B to a "1" inhibit any update cycle and then clear the UIP status bit.

DV2, DV1, DV0 – Three bits are used to permit the program to select various conditions of the 22-stage divider chain. The divider selection bits identify which of the three time-base frequencies is in use. Table 3 shows that time bases of 4.194304 MHz, 1.048576 MHz, and 32.768 kHz may be used. The divider selection bits are also used to reset the divider chain. When the time/calendar is first initialized, the program may start the divider at the precise time stored in the RAM. When the divider reset is removed the first update cycle begins half a second later. These three read/write bits are never modified by the RTC and are not affected by $\overline{RES}$.

RS3, RS2, RS1, RS0 – The four rate selection bits select one of 15 taps on the 22-stage divider, or disable the divider output. The tap selected may be used to generate an output square wave (SQW pin) and/or a periodic interrupt. The program may do one of the following: 1) enable the interrupt with the PIE bit, 2) enable the SQW output pin with the SQWE bit, 3) enable both at the same time at the same rate, or 4) enable neither. Table 4 lists the periodic interrupt rates and the square-wave frequencies that may be chosen with the RS bits. These four bits are read/write bits which are not affected by $\overline{RES}$ and are never changed by the RTC.

- **Register B ($0B)**

MSB … LSB — Read/Write Register

b7	b6	b5	b4	b3	b2	b1	b0
SET	PIE	AIE	UIE	SQWE	DM	24/12	DSE

SET – When the SET bit is a "0", the update cycle functions normally by advancing the counts once-per-second. When the SET bit is written to a "1", any update cycle in progress is aborted and the program may initialize the time and calendar bytes without an update occurring in the midst of initializing. SET is a read/write bit which is not modified by $\overline{RES}$ or internal functions of the HD146818.

PIE – The periodic interrupt enable (PIE) bit is a read/write bit which allows the periodic-interrupt flag (PF) bit to cause the $\overline{IRQ}$ pin to be driven "Low". A program writes a "1" to the PIE bit in order to receive periodic interrupts at the rate specified by the RS3, RS2, RS1, and RS0 bits in Control Register A. A "0" in PIE blocks $\overline{IRQ}$ from being initiated by a periodic interrupt, but the periodic flag (PF) bit is still at the periodic rate. PIE is not modified by any internal HD146818 functions, but is cleared to "0" by a $\overline{RES}$.

AIE – The alarm interrupt enable (AIE) bit is a read/write bit which when set to a "1" permits the alarm flag (AF) to assert $\overline{IRQ}$. An alarm interrupt occurs for each second that the three time bytes equal the three alarm bytes (including a "don't care" alarm code of binary 11XXXXXX). When the AIE bit is a "0", the AF bit does not initiate an $\overline{IRQ}$ signal. The $\overline{RES}$ pin clears AIE to "0". The internal functions do not affect the AIE bit.

UIE – The UIE (update-ended interrupt enable) bit is a read/write bit which enables the update-end flage (UF) bit to assert $\overline{IRQ}$. The $\overline{RES}$ pin going "Low" or the SET bit going "1" clears the UIE bit.

SQWE – When the square-wave enable (SQWE) bit is set to a "1" by the program, a square-wave signal at the frequency specified in the rate selection bits (RS3 to RS0) appears on the SQW pin. When the SQWE bit is set to a "0" the SQW pin is held "Low". The state of SQWE is cleared by the $\overline{RES}$ pin. SQWE is a read/write bit.

DM – The data mode (DM) bit indicates whether time and calendar updates are to use binary or BCD formats. The DM bit is written by the processor program and may be read by the program, but is not modified by any internal functions or $\overline{RES}$. A "1" in DM signifies binary data, while a "0" in DM specified binary-coded-decimal (BCD) data.

24/12 – The 24/12 control bit establishes the format of the hours bytes as either the 24-hour mode (a "1") or the 12-hour mode (a "0"). This is a read/write bit, which is affected only by the software.

DSE – The daylight savings enable (DSE) bit is a read/write bit which allows the program to enable two special updates (when DSE is a "1"). On the last Sunday in April the time increments from 1:59:59 AM to 3:00:00 AM. On the last Sunday in October when the time first reaches 1:59:59 AM it changes to 1:00:00 AM. These special updates do not occur when the DSE bit is a "0". DSE is not changed by any internal operations or reset.

- **Register C ($0C)**

MSB … LSB — Read-Only Register

b7	b6	b5	b4	b3	b2	b1	b0
IRQF	PF	AF	UF	0	0	0	0

IRQF – The interrupt request flag (IRQF) is set to a "1" when one or more of the following are true:

PF = PIE = "1"
AF = AIE = "1"
UF = UIE = "1"

i.e., IRQF = PF · PIE + AF · AIE + UF · UIE

Any time the IRQF bit is a "1", the $\overline{IRQ}$ pin is driven "Low". All flag bits are cleared after Register C is read by the program or when the $\overline{RES}$ pin is low. A program write to Register C does not modify any of the flag bits.

PF – The periodic interrupt flag (PF) is a read-only bit which is set to a "1" when a particular edge is detected on the selected tap of the divider chain. The RS3 to RS0 bits establish the periodic rate. PF is set to a "1" independent of the state of the PIE bit. PF being a "1" initiates an $\overline{IRQ}$ signal and the IRQF bit when PIE is also a "1". The PF bit is cleared by a $\overline{RES}$ or a software read of Register C.

AF – A "1" in the AF (alarm interrupt flag) bit indicates that the current time has matched the alarm time. A "1" in the AF causes the $\overline{IRQ}$ pin to go "Low", and a "1" to appear in the IRQF bit, when the AIE bit also is a "1". A $\overline{RES}$ or a read of Register C clears AF.

UF – The update-ended interrupt flag (UF) bit is set after each update cycle. When the UIE bit is a "1", the "1" in UF causes the IRQF bit to be a "1", asserting $\overline{IRQ}$. UF is cleared by a Register C read or a $\overline{RES}$.

b3 to b0 – The unused bits of Status Register C are read as "0's". They can not be written.

- **Register D ($0D)**

MSB … LSB — Read Only Register

b7	b6	b5	b4	b3	b2	b1	b0
VRT	0	0	0	0	0	0	0

VRT – The valid RAM and time (VRT) bit indicates the condition of the contents of the RAM, provided the power sense (PS) pin is satisfactorily connected. A "0" appears in the VRT bit when the power-sense pin is "Low". The processor program can set the VRT bit when the time and calendar are initialized to indicate that the RAM and time are valid. The VRT is a read/only bit which is not modified by the $\overline{RES}$ pin. The VRT bit can only be set by reading the Register D. For setting this bit, PS signal needs to be "High" level.

b6 to b0 – The remaining bits of Register D are unused. They cannot be written, but are always read as "0's".

■ **NOTE FOR USE**

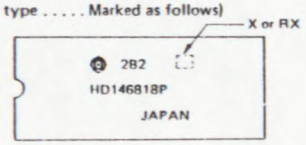
Input Signal, which is not necessary for user's application, should be used fixed to "High" or "Low" level. This is applicable to the following signal pins.

CKFS, PS

RESTRICTION ON HD146818 USAGE (1)

The daylight saving function can not be performed on the HD146818P (X type). So do not use this function for the system design.

< Type number > HD146818P (X type . . . . . Marked as follows)



< Restriction on usage >

Please set "0" to DSE bit (Daylight Saving Enable bit) on initializing the control register B.
DSE = "1" is prohibited.

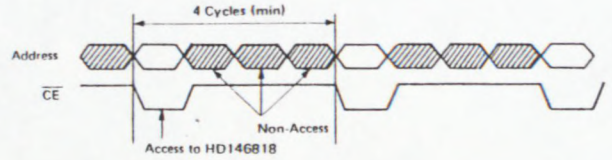
RESTRICTION ON HD146818 USAGE (2)

Access to HD146818 needs to be performed under following conditions.

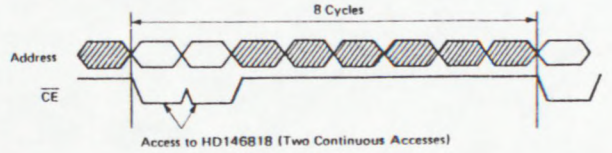
(i) Chip enable ($\overline{CE}$) must be asserted to active "Low" level only when MPU performs read/write operation from/into internal RAM (Time and Calendar RAM, Control register, User RAM).

(ii) User RAM and control register must be accessed in less than 1/4 frequency shown below.
(Example: After one access, non-access cycles more than three cycles are necessary to be inserted.)

[Example 1]



[Example 2]



As shown in the above [example 2], when HD146818 is accessed continuously, continuous access must not be executed over fifty times.

(iii) The application that User RAM is used for program area should be avoided. (Inhibit continuous access.)

(iv) Minimize the noise by inserting noise bypass condenser between power supply and ground pin (V_{CC}-V_{SS}). (Insert noise bypass condenser as near HD146818 as possible.)

HD6318, HD63A18

RTC (Real Time Clock Plus RAM)

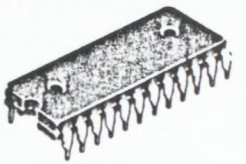
–ADVANCE INFORMATION–

The HD6318 is a HMCS6800 peripheral CMOS device which combines three unique features: a complete time-of-day clock with alarm and one hundred calendar, a programmable periodic interrupt and square-wave generator, and 50 bytes of Low-power static RAM.

This device includes HD6801, HD6301 multiplexed bus interface circuit and 8085's multiplexed bus interface as well, so it can be directly connected to HD6801, HD6301 and 8085.

The Real-Time Clock plus RAM has two distinct uses. First, it is designed as battery powered CMOS part including all the common battery backed-up functions such as RAM, time, and calendar. Secondly, the HD6318 may be used with a CMOS microprocessor to relieve the software of timekeeping work-load and to extend the available RAM of an MPU such as the HD6301.

HD6318P, HD63A18P

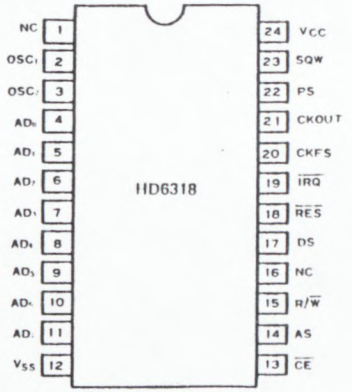


(DP-24)

■ FEATURES

- Revised product of HD146818 in both Function and Characteristics
- Compatible with HD146818 and Motorola MC146818
- Time-of-Day Clock and Calendar
  - Counts Seconds, Minutes, and Hours of the Day
  - Counts Days of Week, Date, Month, and Year
- Binary or BCD Representation of Time, Calendar, and Alarm
- 12- or 24 Hour Clock with AM and PM in 12-Hour Mode
- Automatic End of Month Recognition
- Automatic Leap Year Compensation
- Interfaced with Software as 64 RAM Locations
  - 14 Bytes of Clock and Control Register
  - 50 Bytes of General Purpose RAM
- Three Interrupt are Separately Software Maskable and Testable
  - Time-of-Day Alarm, Once-per-Second to Once-per-Day
  - Periodic Rates from 30.5 μs to 500 ms
  - End-of-Clock Update Cycle
- Programmable Square-Wave Output Signal
- Three Time Base Input Options
  - 4.194304 MHz
  - 1.048576 MHz
  - 32.768 kHz
- Clock Output May be used as Microprocessor Clock Input
  - At Time Base Frequency ÷ 4 or ÷ 1
- Multiplexed Bus Interface Circuit of HD6801, HD6301 and 8085
- Low-Power, High-Speed, High-Density CMOS
- Battery Backed-up Operation

■ PIN ARRANGEMENT



(Top View)

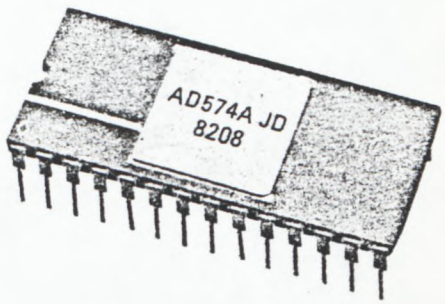
■ TYPE OF PRODUCTS

Type No.	Bus Timing
HD6318	1.0 MHz
HD63A18	1.5 MHz

Fast, Complete 12-Bit A/D Converter with Microprocessor Interface

FEATURES

- Complete 12-Bit A/D Converter with Reference and Clock
- Full 8- or 16-Bit Microprocessor Bus Interface
- 250ns Bus Access Time
- Guaranteed Linearity Over Temperature
  - 0 to +70°C – AD574AJ, AK, AL
  - −55°C to +125°C – AD574AS, AT, AU
- No Missing Codes Over Temperature
- Fast Successive Approximation Conversion – 25μs
- Buried Zener Reference for Long-Term Stability and Low Gain T.C. 10ppm/°C max AD574AL
  12.5ppm/°C max AD574AU
- Low Profile 28-Pin Ceramic DIP
- Low Power: 390mW
- Low Cost



PRODUCT DESCRIPTION

The AD574A is a complete 12-bit successive-approximation analog-to-digital converter with 3-state output buffer circuitry for direct interface to an 8-, 12-or 16-bit microprocessor bus. The AD574A design is implemented with two LSI chips each containing both analog and digital circuitry, resulting in the maximum performance and flexibility at the lowest cost.

One chip is the high performance AD565A 12-bit DAC and voltage reference. It contains the high speed current output switching circuitry, laser-trimmed thin film resistor network, low T.C. buried zener reference and the precision input scaling and bipolar offset resistors. This chip is laser-trimmed at the wafer stage (LWT) to adjust ladder network linearity, voltage reference tolerance and temperature coefficient, and the calibration accuracy of input scaling and bipolar offset resistors.

The second chip uses the proven LCI (linear-compatible integrated injection logic) process to provide the low-power I^2L successive-approximation register, converter control circuitry, clock, bus interface, and the high performance latching comparator. The precision, low-drift comparator is adjusted for initial input offset error at the wafer stage by the "zener-zap" technique which trims the comparator input stage to 1/10 LSB typical error. This form of trimming, while cumbersome for complex ladder networks, is an attractive alternative to thin film resistor trimming for a simple offset adjustment and eliminates the need for thin film processing for this portion of the circuitry.

The AD574A is available in six different grades. The AD574AJ, AK, and AL grades are specified for operation over the 0 to +70°C temperature range. The AD574AS, AT, and AU are specified for the −55°C to +125°C range. All grades are packaged in a low-profile, 0.600 inch wide, 28-pin hermetically-sealed ceramic DIP.

PRODUCT HIGHLIGHTS

1. The AD574A interfaces to most popular microprocessors with an 8-, 12-, or 16-bit bus without external buffers or peripheral interface controllers. Multiple-mode three-state output buffers connect directly to the data bus while the read and convert commands are taken from the control bus. The 12-bits of output data can be read either as one 12-bit word or as two 8-bit bytes (one with 8 data bits, the other with 4 data bits and 4 trailing zeros).
2. The precision, laser-trimmed scaling and bipolar offset resistors provide four calibrated ranges, 0 to +10 and 0 to +20 volts unipolar, or −5 to +5 and −10 to +10 volts bipolar. Typical bipolar offset and full scale calibration of ±0.1% can be trimmed to zero with one external component each.
3. The internal buried zener reference is trimmed to 10.00 volts with 1% maximum error and 15ppm/°C typical T.C. The reference is available externally and can drive up to 1.5mA beyond that required for the reference and bipolar offset resistors.
4. The two-chip construction renders the AD574A inherently more reliable than hybrid multi-chip designs. All three military grades have guaranteed linearity error over the full −55°C to +125°C and are especially recommended for high performance needs in harsh environments. These units are available processed to MIL-STD-883B, Level B.

P.O. Box 280; Norwood, Massachusetts 02062 U.S.A.
Tel:617/329-4700 Twx: 710/394-6577
Telex: 924491 Cables: ANALOG NORWOODMASS

SPECIFICATIONS (typical @ +25°C with V_{CC} = +15V or +12V, V_{LOGIC} = +5V, V_{EE} = -15V or -12V, unless otherwise specified)

DC AND TRANSFER ACCURACY SPECIFICATIONS

MODEL	AD574AJ	AD574AK	AD574AL	UNITS
RESOLUTION (max)	12	12	12	Bits
LINEARITY ERROR				
25°C (max)	±1	±1/2	±1/2	LSB
T_{min} to T_{max} (max)	±1	±1/2	±1/2	LSB
DIFFERENTIAL LINEARITY ERROR (Minimum resolution for which no missing codes are guaranteed)				
25°C	11	12	12	Bits
T_{min} to T_{max}	11	12	12	Bits
UNIPOLAR OFFSET (max) (Adjustable to zero)	±2	±2	±2	LSB
BIPOLAR OFFSET (max) (Adjustable to zero)	±10	±4	±4	LSB
FULL SCALE CALIBRATION ERROR (with fixed 50Ω resistor from REF OUT to REF IN)				
(Adjustable to zero) 25°C (max)	0.25	0.25	0.25	% of F.S.
T_{min} to T_{max} (Without Initial Adjustment)	0.47	0.37	0.30	% of F.S.
(With Initial Adjustment)	0.22	0.12	0.05	% of F.S.
TEMPERATURE RANGE		0 to +70		°C
TEMPERATURE COEFFICIENTS (Using internal reference)				
Guaranteed max change				
T_{min} to T_{max}				
Unipolar Offset	±2	±1	±1	LSB
	(10)	(5)	(5)	(ppm/°C)
Bipolar Offset	±2	±1	±1	LSB
	(10)	(5)	(5)	(ppm/°C)
Full Scale Calibration	±9	±5	±2	LSB
	(50)	(27)	(10)	(ppm/°C)
POWER SUPPLY REJECTION				
Max change in Full Scale Calibration				
$+13.5V \leq V_{CC} \leq +16.5V$ or $+11.4V \leq V_{CC} \leq +12.6V$	±2	±1	±1	LSB
$+4.5V \leq V_{LOGIC} \leq +5.5V$	±1/2	±1/2	±1/2	LSB
$-16.5V \leq V_{EE} \leq -13.5V$ or $-12.6V \leq V_{EE} \leq -11.4V$	±2	±1	±1	LSB
ANALOG INPUTS				
Input Ranges				
Bipolar		-5 to +5		Volts
		-10 to +10		Volts
Unipolar		0 to +10		Volts
		0 to +20		Volts
Input Impedance				
10 Volt Span		5k (3k min, 7k max)		Ω
20 Volt Span		10k (6k min, 14k max)		Ω
POWER SUPPLIES				
Operating Range				
V_{LOGIC}		+4.5 to +5.5		Volts
V_{CC}		+11.4 to +16.5		Volts
V_{EE}		-11.4 to -16.5		Volts
Operating Current				
I_{LOGIC}		30 typ, 40 max		mA
I_{CC}		2 typ, 5 max		mA
V_{EE}		18 typ, 30 max		mA
POWER DISSIPATION		390 typ, 725 max		mW
INTERNAL REFERENCE VOLTAGE		10.00 ±0.1 (max)		Volts
Output Current (available for external loads) (External load should not change during conversion)		1.5 max[1]		mA

NOTES

[1] The reference should be buffered for operation on ±12V supplies.

Specifications subject to change without notice.

DC AND TRANSFER ACCURACY SPECIFICATIONS

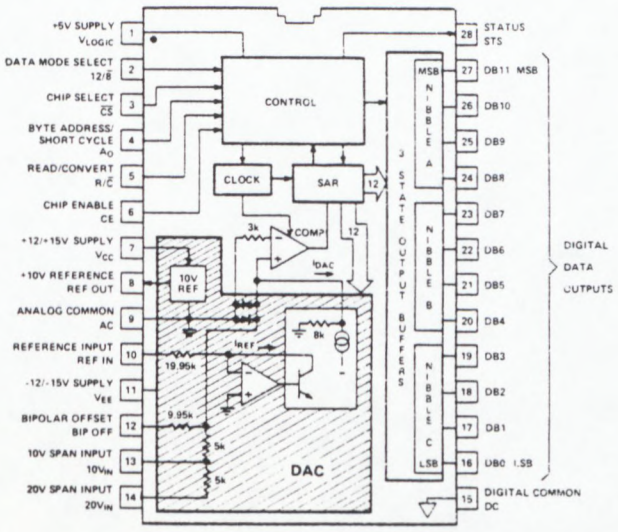
MODEL	AD574AS	AD574AT	AD574AU	UNITS
RESOLUTION (max)	12	12	12	Bits
LINEARITY ERROR				
25°C (max)	±1	±1/2	±1/2	LSB
-25°C to +85°C (max)	±1	±1/2	±1/2	LSB
-55°C to +125°C (max)	±1	±1	±1	LSB
DIFFERENTIAL LINEARITY ERROR (Minimum resolution for which no missing codes are guaranteed)				
25°C	11	12	12	Bits
T_{min} to T_{max}	11	12	12	Bits
UNIPOLAR OFFSET (max) (Adjustable to zero)	±2	±2	±2	LSB
BIPOLAR OFFSET (max) (Adjustable to zero)	±10	±4	±4	LSB
FULL SCALE CALIBRATION ERROR (with fixed 50Ω resistor from REF IN to REF OUT)				
(Adjustable to zero) 25°C (max)	0.25	0.25	0.25	% of F.S.
T_{min} to T_{max} (Without Initial Adjustment)	0.75	0.5	0.37	% of F.S.
(With Initial Adjustment)	0.5	0.25	0.12	% of F.S.
TEMPERATURE RANGE	-55 to +125			°C
TEMPERATURE COEFFICIENTS (using internal reference) Guaranteed max change				
T_{min} to T_{max}				
Unipolar Offset	±2	±1	±1	LSB
	(5)	(2.5)	(2.5)	(ppm/°C)
Bipolar Offset	±4	±2	±1	LSB
	(10)	(5)	(2.5)	(ppm/°C)
Full Scale Calibration	±20	±10	±5	LSB
	(50)	(25)	(12.5)	(ppm/°C)
POWER SUPPLY REJECTION Max change in Full Scale Calibration				
$+13.5V \leq V_{CC} \leq +16.5V$ or $+11.4V \leq V_{CC} \leq +12.6V$	±2	±1	±1	LSB
$+4.5V \leq V_{LOGIC} \leq +5.5V$	±1/2	±1/2	±1/2	LSB
$-16.5V \leq V_{EE} \leq -13.5V$ or $-12.6V \leq V_{EE} \leq -11.4V$	±2	±1	±1	LSB
ANALOG INPUTS				
Input Ranges				
Bipolar	-5 to +5			Volts
	-10 to +10			Volts
Unipolar	0 to +10			Volts
	0 to +20			Volts
Input Impedance				
10 Volt Span	5k (3k min, 7k max)			Ω
20 Volt Span	10k (6k min, 14k max)			Ω
POWER SUPPLIES				
Operating Range				
V_{LOGIC}	+4.5 to +5.5			Volts
V_{CC}	+11.4 to +16.5			Volts
V_{EE}	-11.4 to -16.5			Volts
Operating Current				
I_{LOGIC}	30 typ, 40 max			mA
I_{CC}	2 typ, 5 max			mA
I_{EE}	18 typ, 30 max			mA
POWER DISSIPATION	390 typ, 725 max			mW
INTERNAL REFERENCE VOLTAGE	10.00 ±0.1 (max)			Volts
Output Current (available for external loads) (External load should not change during conversion)	1.5max[1]			mA

NOTES
[1] The reference should be buffered for operation on ±12V supplies.
Specifications subject to change without notice.

DIGITAL CHARACTERISTICS[1] (All grades, T_{min} – T_{max})

	Min	Typ	Max	
Logic Inputs[2] (CE, $\overline{CS}$, R $\overline{C}$, A_0)				
Voltages				
Logic "1"	+2.0V		+5.5V	
Logic "0"	−0.5V		+0.8V	
Current	−50μA		+50μA	
Capacitance		5pF		
Logic Outputs (DB11–DB0, STS)				
Logic "0"			+0.4V	$I_{SINK} \leq 1.6mA$
Logic "1"	2.4V			$I_{SOURCE} \leq 500\mu A$
Leakage (When in high-Z state)	−40μA		+40μA	DB11 – DB0 Only
Capacitance		5pF		

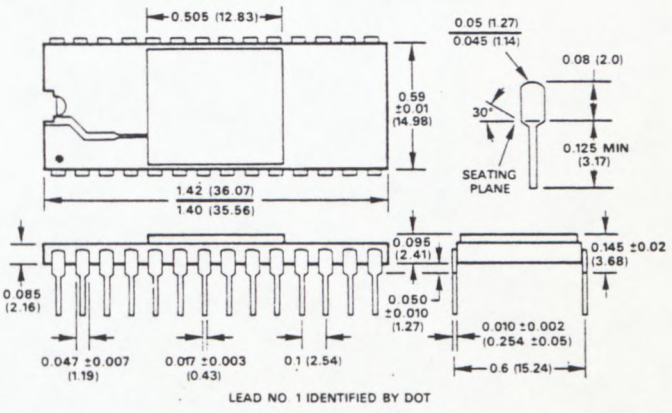
[1]Detailed Timing Specifications appear in the Digital Interface Section.
[2]12/$\overline{8}$ Input is not TTL-compatible and must be hard-wired to V_{LOGIC} or DIGITAL COMMON.



AD574A Block Diagram and Pin Configuration

AD574A PACKAGE OUTLINE

Dimensions shown in inches and (mm).



ABSOLUTE MAXIMUM RATINGS

(Specifications apply to all grades, except where noted)

V_{CC} to Digital Common	0 to +16.5V
V_{EE} to Digital Common	0 to −16.5V
V_{LOGIC} to Digital Common	0 to +7V
Analog Common to Digital Common	±1V
Control Inputs (CE, $\overline{CS}$, A_0, 12/$\overline{8}$, R/$\overline{C}$) to Digital Common	−0.5V to V_{LOGIC} +0.5V
Analog Inputs (REF IN, BIP OFF, $10V_{IN}$) to Analog Common	±16.5V
$20V_{IN}$ to Analog Common	±24V
REF OUT	Indefinite short to common Momentary short to V_{CC}
Chip Temperature (J, K, L grades)	100°C
(S, T, U grades)	150°C
Power Dissipation	1000mW
Lead Temperature, Soldering	300°C, 10 sec.
Storage Temperature	−65°C to +150°C
Thermal Resistance, θ_{JA}	60°C/W

AD574A ORDERING GUIDE

Model	Temp. Range	Linearity Error Max (T_{min} to T_{max})	Resolution No Missing Codes (T_{min} to T_{max})	Max Full Scale T.C. (ppm/°C)
AD574AJD	0 to +70°C	±1LSB	11 Bits	50.0
AD574AKD	0 to +70°C	±1/2LSB	12 Bits	27.0
AD574ALD	0 to +70°C	±1/2LSB	12 Bits	10.0
AD574ASD	−55°C to +125°C	±1LSB	11 Bits	50.0
AD574ASD/883B	−55°C to +125°C	±1LSB	11 Bits	50.0
AD574ATD	−55°C to +125°C	±1LSB	12 Bits	25.0
AD574ATD/883B	−55°C to +125°C	±1LSB	12 Bits	25.0
AD574AUD	−55°C to +125°C	±1LSB	12 Bits	12.5
AD574AUD/883B	−55°C to +125°C	±1LSB	12 Bits	12.5

THE AD574A OFFERS GUARANTEED MAXIMUM LINEARITY ERROR OVER THE FULL OPERATING TEMPERATURE RANGE

DEFINITIONS OF SPECIFICATIONS

LINEARITY ERROR

Linearity error refers to the deviation of each individual code from a line drawn from "zero" through "full scale". The point used as "zero" occurs ½LSB (1.22mV for 10 volt span) before the first code transition (all zeros to only the LSB "on"). "Full scale" is defined as a level 1½LSB beyond the last code transition (to all ones). The deviation of a code from the true straight line is measured from the middle of each particular code.

The AD574AK, AL, AT, and AU grades are guaranteed for maximum nonlinearity of ±½LSB. For these grades, this means that an analog value which falls exactly in the center of a given code width will result in the correct digital output code. Values nearer the upper or lower transition of the code width may produce the next upper or lower digital output code. The AD574AJ and AS grades are guaranteed to ±1LSB max error. For these grades, an analog value which falls within a given code width will result in either the correct code for that region or either adjacent one.

Note that the linearity error is not user-adjustable.

DIFFERENTIAL LINEARITY ERROR (NO MISSING CODES)

A specification which guarantees no missing codes requires that every code combination appear in a monotonic increasing sequence as the analog input level is increased. Thus every code must have a finite width. For the AD574AK, AL, AT, and AU grades, which guarantee no missing codes to 12-bit resolution, all 4096 codes must be present over the entire operating temperature ranges. The AD574AJ and AS grades guarantee no missing codes to 11-bit resolution over temperature; this means that all code combinations of the upper 11 bits must be present; in practice very few of the 12-bit codes are missing.

UNIPOLAR OFFSET

The first transition should occur at a level ½LSB above analog common. Unipolar offset is defined as the deviation of the actual transition from that point. This offset can be adjusted as discussed on the following two pages. The unipolar offset temperature coefficient specifies the maximum change of the transition point over temperature, with or without external adjustment.

BIPOLAR OFFSET

Similarly, in the bipolar mode, the major carry transition (0111 1111 1111 to 1000 0000 0000) should occur for an analog value ½LSB below analog common. The bipolar offset error and temperature coefficient specify the initial deviation and maximum change in the error over temperature.

QUANTIZATION UNCERTAINTY

Analog-to-digital converters exhibit an inherent quantization uncertainty of ±½LSB. This uncertainty is a fundamental characteristic of the quantization process and cannot be reduced for a converter of given resolution.

LEFT-JUSTIFIED DATA

The data format used in the AD574A is left-justified. This means that the data represents the analog input as a fraction of full-scale, ranging from 0 to $\frac{4095}{4096}$. This implies a binary point to the left of the MSB.

FULL SCALE CALIBRATION ERROR

The last transition (from 1111 1111 1110 to 1111 1111 1111) should occur for an analog value 1 1/2LSB below the nominal full scale (9.9963 volts for 10.000 volts full scale). The full scale calibration error is the deviation of the actual level at the last transition from the ideal level. This error, which is typically 0.05 to 0.1% of full scale, can be trimmed out as shown in Figures 3 and 4. The full scale calibration error over temperature is given with and without the initial error trimmed out. The temperature coefficients for each grade indicate the maximum change in the full scale gain from the initial value using the internal 10 volt reference.

TEMPERATURE COEFFICIENTS

The temperature coefficients for full-scale calibration, unipolar offset, and bipolar offset specify the maximum change from the initial (25°C) value to the value at T_{min} or T_{max}.

POWER SUPPLY REJECTION

The standard specifications for the AD574A assume use of +5.00 and ±15.00 or ±12.00 volt supplies. The only effect of power supply error on the performance of the device will be a small change in the full scale calibration. This will result in a linear change in all lower order codes. The specifications show the maximum change in calibration from the initial value with the supplies at the various limits.

CODE WIDTH

A fundamental quantity for A/D converter specifications is the code width. This is defined as the range of analog input values for which a given digital output code will occur. The nominal value of a code width is equivalent to 1 least significant bit (LSB) of the full scale range or 2.44mV out of 10 volts for a 12-bit ADC.

CIRCUIT OPERATION

The AD574A is a complete 12-bit A/D converter which requires no external components to provide the complete successive-approximation analog-to-digital conversion function. A block diagram of the AD574A is shown in Figure 1. The device consists of two chips, one containing the precision 12-bit DAC with voltage reference, the other containing the comparator, successive-approximation register, clock, output buffers and control circuitry.

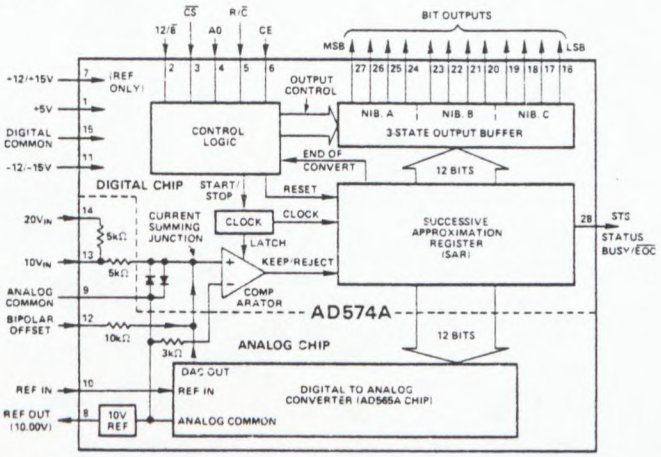


Figure 1. Block Diagram of AD574A 12-Bit A-to-D Converter

When the control section is commanded to initiate a conversion (as described later), it then enables the clock and resets the successive-approximation register (SAR) to all zeros. Once a conversion cycle has begun, it cannot be stopped or re-started and data is not available from the output buffers. The SAR, timed by the clock, will then sequence through the conversion cycle and return an end-of-convert flag to the control section. The control section will then disable the clock, bring the output status flag low, and enable control functions to allow data read functions by external command.

During the conversion cycle, the internal 12-bit current output DAC is sequenced by the SAR from the most-significant-bit (MSB) to least-significant-bit (LSB) to provide an output current which accurately balances the input signal current through the 5kΩ (or 10kΩ) input resistor. The comparator determines whether the addition of each successively-weighted bit current causes the DAC current sum to be greater or less than the input current; if the sum is less, the bit is left on; if more, the bit is turned off. After testing all the bits, the SAR contains a 12-bit binary code which accurately represents the input signal to within ±½LSB.

The temperature-compensated buried Zener reference provides the primary voltage reference to the DAC and guarantees excellent stability with both time and temperature. The reference is trimmed to 10.00 volts ±1%; it can supply up to 1.5mA to an external load in addition to that required to drive the reference input resistor (0.5mA) and bipolar offset resistor (1mA) when the AD574A is powered from ±15V supplies. If the AD574A is used with ±12V supplies, or if external current must be supplied over the full temperature range, an external buffer amplifier is recommended. Any external load on the AD574A reference must remain constant during conversion. The thin film application resistors are trimmed to match the full scale output current of the DAC. There are two 5kΩ input scaling resistors to allow either a 10 volt or 20 volt span. The 10kΩ bipolar offset resistor is grounded for unipolar operation or connected to the 10 volt reference for bipolar operation.

DRIVING THE AD574A ANALOG INPUT

The AD574A is a successive-approximation type analog-to-digital converter. During the conversion cycle, the ADC input current is modulated by the DAC test current at approximately a 500kHz rate. Thus it is important to recognize that the signal source driving the AD574A must be capable of holding a constant output voltage under dynamically-changing load conditions.

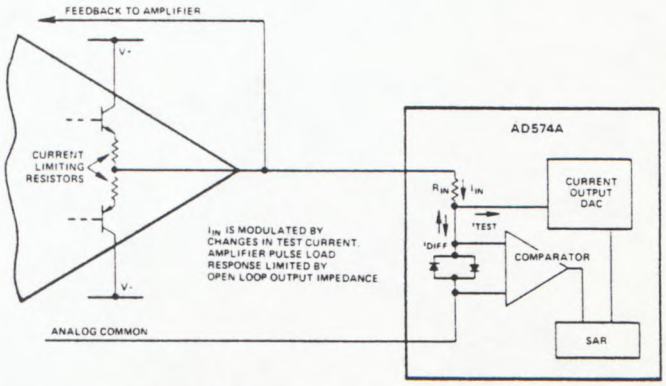


Figure 2. Op Amp – AD574A Interface

The closed loop output impedance of an op amp is equal to the open loop output impedance (usually a few hundred ohms) divided by the loop gain at the frequency of interest. It is often assumed that the loop gain of a follower-connected op amp is sufficiently high to reduce the closed loop output impedance to a negligibly small value, particularly if the signal is low frequency. However, the amplifier driving an AD574A must either have sufficient loop gain at 500kHz to reduce the closed loop output impedance to a low value or have low open loop output impedance.

This can be accomplished either by using a wideband op amp or by placing a discrete-transistor or integrated buffer inside the amplifier's feedback loop.

SUPPLY DECOUPLING AND LAYOUT CONSIDERATIONS

It is critically important that the AD574A power supplies be filtered, well-regulated, and free from high frequency noise. Use of noisy supplies will cause unstable output codes to be generated. Switching power supplies are not recommended for circuits attempting to achieve 12-bit accuracy unless great care is used in filtering any switching spikes present in the output. Remember that a few millivolts of noise represents several counts of error in a 12-bit ADC.

Decoupling capacitors should be used on all power supply pins; the +5V supply decoupling capacitor should be connected directly from pin 1 to pin 15 (digital common) and the $+V_{CC}$ and $-V_{EE}$ pins should be decoupled directly to analog common (pin 9). A suitable decoupling capacitor is a 47μF tantalum type in parallel with a 0.1μF disc ceramic type.

Circuit layout should attempt to locate the AD574A, associated analog input circuitry, and interconnections as far as possible from logic circuitry. For this reason, the use of wire-wrap circuit construction is not recommended. Careful printed-circuit construction is preferred.

UNIPOLAR RANGE CONNECTIONS FOR THE AD574A

The AD574A contains all the active components required to perform a complete 12-bit A/D conversion. Thus, for most situations, all that is necessary is connection of the power supplies (+5, +12/+15 and −12/−15 volts), the analog input, and the conversion initiation command, as discussed on the next page. Analog input connections and calibration are easily accomplished; the unipolar operating mode is shown in Figure 4.

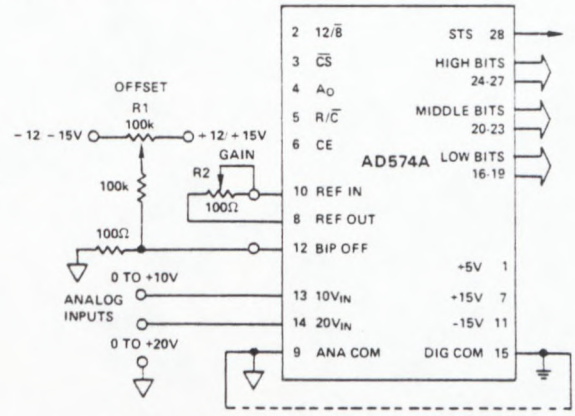


Figure 3. Unipolar Input Connections

All of the thin film application resistors of the AD574A are trimmed for absolute calibration. Therefore, in many applications, no calibration trimming will be required. The absolute accuracy for each grade is given in the specification tables. For example, if no trims are used, the AD574AK guarantees ±2LSB max zero offset error and ±0.25% (10LSB) max full scale error. (Typical full scale error is ±2LSB.) If the offset trim is not required, pin 12 can be connected directly to pin 9; the two resistors and trimmer for pin 12 are then not needed. If the full scale trim is not needed, a 50Ω ±1% metal film resistor should be connected between pin 8 and pin 10.

The analog input is connected between pin 13 and pin 9 for a 0 to +10V input range, between 14 and pin 9 for a 0 to +20V input range. The AD574A easily accommodates an input signal beyond the supplies. For the 10 volt span input, the LSB has a nominal value of 2.44mV, for the 20 volt span, 4.88mV. If a 10.24V range is desired (nominal 2.5mV/bit), the gain trimmer (R2) should be replaced by a 50Ω resistor, and a 200Ω trimmer inserted in series with the analog input to pin 13 (for a full scale range of 20.48V (5mV/bit), use a 500Ω trimmer into pin 14). The gain trim described below is now done with these trimmers. The nominal input impedance into pin 13 is 5kΩ, and 10kΩ into pin 14.

UNIPOLAR CALIBRATION

The AD574A is intended to have a nominal ½LSB offset so that the exact analog input for a given code will be in the middle of that code (halfway between the transitions to the codes above and below it). Thus, when properly calibrated, the first transition (from 0000 0000 0000 to 0000 0000 0001) will occur for an input level of +1/2LSB (1.22mV for 10V range).

If pin 12 is connected to pin 9, the unit typically will behave in this manner, within specifications. If the offset trim (R1) is used, it should be trimmed as above, although a different offset can be set for a particular system requirement. This circuit will give approximately ±15mV of offset trim range.

The full scale trim is done by applying a signal 1 1/2LSB below the nominal full scale (9.9963 for a 10V range). Trim R2 to give the last transition (1111 1111 1110 to 1111 1111 1111).

BIPOLAR OPERATION

The connections for bipolar ranges are shown in Figure 4. Again, as for the unipolar ranges, if the offset and gain specifications are sufficient, one or both of the trimmers shown can be replaced by a 50Ω ±1% fixed resistor. The analog input is applied as for the unipolar ranges. Bipolar calibration is similar to unipolar calibration. First, a signal ½LSB above negative full scale (−4.9988V for the ±5V range) is applied and R1 is trimmed to give the first transition (0000 0000 0000 to 0000 0000 0001). Then a signal 1½LSB below positive full scale (+4.9963V for the ±5V range) is applied and R2 trimmed to give the last transition (1111 1111 1110 to 1111 1111 1111).

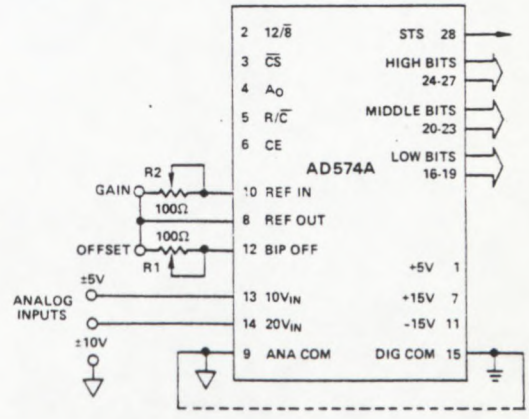


Figure 4. Bipolar Input Connections

GROUNDING CONSIDERATIONS

The analog common at pin 9 is the ground reference point for the internal reference and is thus the "high quality" ground for the AD574A; it should be connected directly to the analog reference point of the system. In order to achieve all of the high accuracy performance available from the AD574A in an environment of high digital noise content, it is required that the analog and digital commons be connected together at the package. In some situations, the digital common at pin 15 can be connected to the most convenient ground reference point; analog power return is preferred.

CONVERSION START/DATA READ CONTROL LOGIC

The AD574A contains on-chip logic to provide conversion initiation and data read operations from signals commonly available in microprocessor systems. Figure 5 shows the internal logic circuitry of the AD574A.

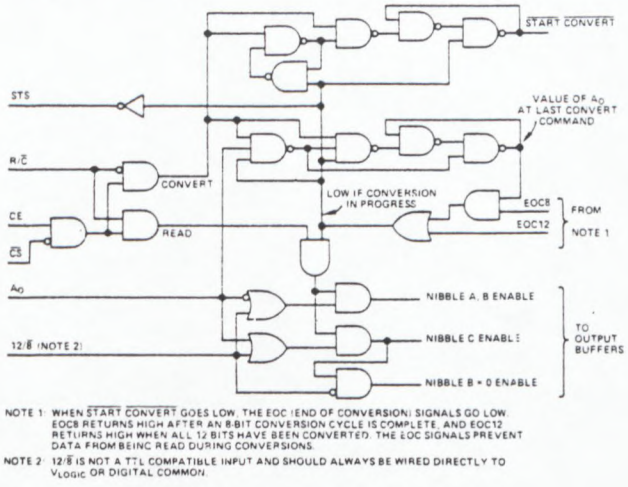


Figure 5. AD574A Control Logic

The control signals CE, $\overline{CS}$, and R/$\overline{C}$ control the operation of the converter. The state of R/$\overline{C}$ when CE and $\overline{CS}$ are both asserted determines whether a data read (R/$\overline{C}$ = 1) or a convert (R/$\overline{C}$ = 0) is in progress. The register control inputs A_0 and 12/$\overline{8}$ control conversion length and data format. The A_0 line is usually tied to the least significant bit of the address bus. If a conversion is started with A_0 low, a full 12-bit conversion cycle is initiated. If A_0 is high during a convert start, a shorter 8-bit conversion cycle results. During data read operations, A_0 determines whether the three-state buffers containing the 8 MSBs of the conversion result (A_0 = 0) or the 4 LSBs (A_0 = 1) are enabled. The 12/$\overline{8}$ pin determines whether the output data is to be organized as two 8-bit words (12/$\overline{8}$ tied to DIGITAL COMMON) or a single 12-bit word (12/$\overline{8}$ tied to VLOGIC). The 12/$\overline{8}$ pin is not TTL-compatible and must be hard-wired to either VLOGIC or DIGITAL COMMON. In the 8-bit mode, the byte addressed when A_0 is high contains the 4 LSBs from the conversion followed by four trailing zeroes. This organization allows the data lines to be overlapped for direct interface to 8-bit buses without the need for external three-state buffers.

It is not recommended that A_0 change state during a data read operation. Asymmetrical enable and disable times of the three-state buffers could cause internal bus contention resulting in potential damage to the AD574A.

An output signal, STS, indicates the status of the converter. STS goes high at the beginning of a conversion and returns low when the conversion cycle is complete.

CE	$\overline{CS}$	R/$\overline{C}$	12/$\overline{8}$	A_0	Operation
0	X	X	X	X	None
X	1	X	X	X	None
1	0	0	X	0	Initiate 12-Bit Conversion
1	0	0	X	1	Initiate 8-Bit Conversion
1	0	1	Pin 1	X	Enable 12-Bit Parallel Output
1	0	1	Pin 15	0	Enable 8 Most Significant Bits
1	0	1	Pin 15	1	Enable 4LSBs + 4 Trailing Zeroes

Table 1. AD574A Truth Table

TIMING

The AD574A is easily interfaced to a wide variety of microprocessors and other digital systems. Discussion of the timing requirements of the AD574A control signals will provide the system designer with useful insight into the operation of the device.

Figure 6 shows a complete timing diagram for the AD574A convert start operation. R/$\overline{C}$ should be low before both CE and $\overline{CS}$ are asserted; if R/$\overline{C}$ is high, a read operation will momentarily occur, possibly resulting in system bus contention. Either CE or $\overline{CS}$ may be used to initiate a conversion. As shown in Figure 6,

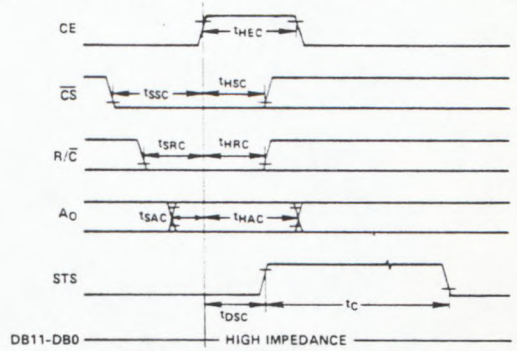


Figure 6. Convert Start Timing

CE is used. If $\overline{CS}$ is used to trigger conversion or if the specified set-up times are not met, appropriately longer pulses are necessary (to provide at least 200ns when R/$\overline{C}$, CE, and $\overline{CS}$ are all valid). Note that CE includes one less propagation delay than $\overline{CS}$ and is therefore the faster input.

Once a conversion is started and the STS line goes high, convert start commands will be ignored until the conversion cycle is complete. The output data buffers cannot be enabled during conversion.

CONVERT START TIMING – FULL CONTROL MODE

Symbol	Parameter	Min	Typ	Max	Units
t_{DSC}	STS Delay from CE			300	ns
t_{HEC}	CE Pulse Width	300			ns
t_{SSC}	$\overline{CS}$ to CE Setup	300			ns
t_{HSC}	$\overline{CS}$ Low During CE High	200			ns
t_{SRC}	R/$\overline{C}$ to CE Setup	250			ns
t_{HRC}	R/$\overline{C}$ Low During CE High	200			ns
t_{SAC}	A_0 to CE Setup	0			ns
t_{HAC}	A_0 Valid During CE High	300			ns
t_C	Conversion Time				
	8-Bit Cycle	10		24	µs
	12-Bit Cycle	15		35	µs

Figure 7 shows the timing for data read operations. The AD574A differs from the original AD574 design in that the three-state output buffers feature faster access time and shorter data latency

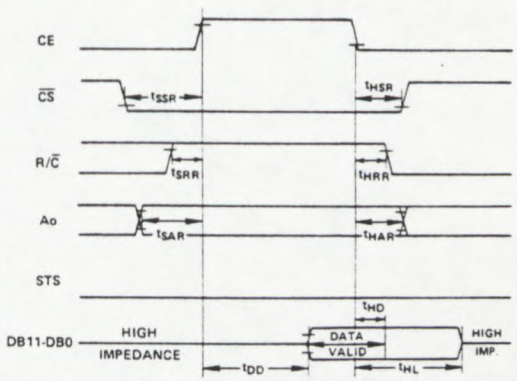


Figure 7. Read Cycle Timing

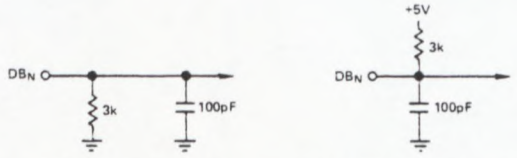
times. This speed improvement simplifies the interface to faster microprocessors. During data read operations, access time is measured from the point where CE and R/$\overline{C}$ both are high (assuming $\overline{CS}$ is already low). If $\overline{CS}$ is used to enable the device, access time is extended by 100ns.

In the 8-bit bus interface mode (12/$\overline{8}$ input wired to DIGITAL COMMON), the address bit, A_0, must be stable at least 150ns prior to $\overline{CE}$ going high and must remain stable during the entire read cycle. If A_0 is allowed to change, damage to the AD574A output buffers may result.

READ TIMING – FULL CONTROL MODE

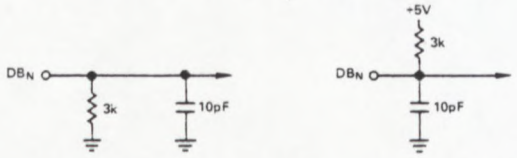
Symbol	Parameter	Min	Typ	Max	Units
t_{DD}[1]	Access Time (from CE)		210	250	ns
t_{HD}	Data Valid after CE Low	25			ns
t_{HL}[2]	Output Float Delay		110	150	ns
t_{SSR}	$\overline{CS}$ to CE Setup	150			ns
t_{SRR}	R/$\overline{C}$ to CE Setup	0			ns
t_{SAR}	A_0 to CE Setup	150			ns
t_{HSR}	$\overline{CS}$ Valid After CE Low	50			ns
t_{HRR}	R/$\overline{C}$ High After CE Low	0			ns
t_{HAR}	A_0 Valid After CE low	50			ns

[1] t_{DD} is measured with the load circuit of Figure 8 and defined as the time required for an output to cross 0.4V or 2.4V.
[2] t_{HL} is defined as the time required for the data lines to change 0.5V when loaded with the circuit of Figure 9.



a. High-Z to Logic 1 *b. High-Z to Logic 0*

Figure 8. Load Circuit for Access Time Test



a. Logic 1 to High-Z *b. Logic 0 to High-Z*

Figure 9. Load Circuit for Output Float Delay Test

"STAND-ALONE" OPERATION

The AD574A can be used in a "stand-alone" mode, which is useful in systems with dedicated input ports available and thus not requiring full bus interface capability.

In this mode, CE and 12/$\overline{8}$ are wired high, $\overline{CS}$ and A_0 are wired low, and conversion is controlled by R/$\overline{C}$. The three-state buffers are enabled when R/$\overline{C}$ is high and a conversion starts when R/$\overline{C}$ goes low. This gives rise to two possible control signals–a high pulse or a low pulse. Operation with a low pulse is shown in Figure 10. In this case, the outputs are forced into the high-impedance state in response to the falling edge of R/$\overline{C}$ and return to valid logic levels after the conversion cycle is completed. The STS line goes high 500ns after R/$\overline{C}$ goes low and returns low 300ns after data is valid.

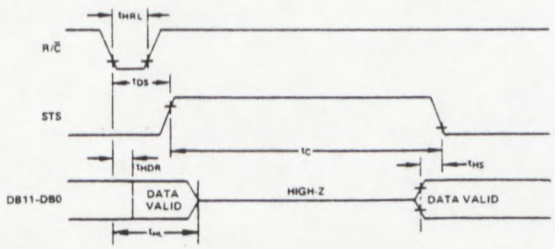


Figure 10. Low Pulse for R/$\overline{C}$ – Outputs Enabled After Conversion

If conversion is initiated by a high pulse as shown in Figure 11, the data lines are enabled during the time when R/$\overline{C}$ is high. The falling edge of R/$\overline{C}$ starts the next conversion and the data lines return to three-state (and remain three-state) until the next high pulse of R/$\overline{C}$.

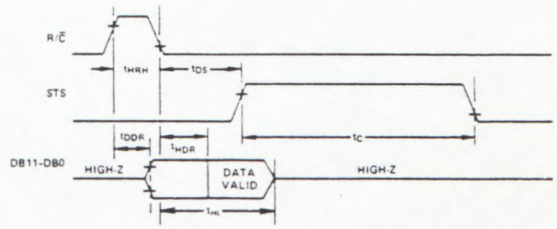


Figure 11. Low Pulse for R/$\overline{C}$ – Outputs Enabled While R/$\overline{C}$ High, Otherwise High-Z

STAND-ALONE MODE TIMING

Symbol	Parameter	Min	Typ	Max	Units
t_{HRL}	Low R/$\overline{C}$ Pulse Width	350			ns
t_{DS}	STS Delay from R/$\overline{C}$			500	ns
t_{HDR}	Data Valid After R/$\overline{C}$ Low	25			ns
t_{HL}	Output Float Delay		110	150	ns
t_{HS}	STS Delay After Data Valid	300		1000	ns
t_{HRH}	High R/$\overline{C}$ Pulse Width	250			ns
t_{DDR}	Data Access Time			250	ns

INTERFACING THE AD574A TO MICROPROCESSORS

The control logic of the AD574A makes direct connection to most microprocessor system buses possible. While it is impossible to describe the details of the interface connections for every microprocessor type, several representative examples will be described here.

GENERAL A/D CONVERTER INTERFACE CONSIDERATIONS

Analog-to-digital converters, like any I/O device, may be interfaced to microprocessors by several methods. These methods include (but are not limited to) direct memory access, isolated or accumulator I/O, and memory-mapped I/O. Direct memory access (DMA) is the fastest, since conversions occur automatically and data updates into memory are transparent to the processor. DMA logic is very processor-dependent and makes use of dedicated specialized hardware.

Memory-mapped and accumulator I/O are more often used and somewhat easier to understand. Memory-mapped I/O assigns the I/O device to one or more locations in the memory space of the microprocessor. This technique has the advantage that the full range of memory reference instructions may be used to operate on the data. The potential disadvantages include limiting the memory space available for program and data memory, somewhat more complex address decoding and more difficult isolation of device select pulses for system debugging. Many processors offer only memory-mapped I/O.

Accumulator I/O uses a set of control signals which are distinct and different from the memory control signals. These control signals, combined with the address bus, serve to define a totally

separate I/O address space. This architecture is simpler from a hardware standpoint, since address decoding requirements are less severe and distinct I/O read and write pulses are more easily located for system debugging purposes. However, processors using accumulator I/O generally can only send data to an output device from the accumulator. This can make the software more cumbersome, since processor-controlled transfers of I/O device data to a memory location cannot be accomplished in a single instruction.

A typical A/D converter interface routine involves several operations. First, a write to the ADC address initiates a conversion. The processor must then wait for the conversion cycle to complete, since most integrated circuit ADCs take longer than one instruction cycle to complete a conversion. Valid data can, of course, only be read after the conversion is complete. The AD574A provides an output signal (STS) which indicates when a conversion is in progress. This signal can be polled by the processor by reading it through an external three-state buffer (or other input port). The STS signal can also be used to generate an interrupt upon completion of conversion, if the system timing requirements are critical (bear in mind that the maximum conversion time of the AD574A is only 35 microseconds) and the processor has other tasks to perform during the ADC conversion cycle. Another possible time-out method is to assume that the ADC will take 35 microseconds to convert, and insert a sufficient number of "do-nothing" instructions to ensure that 35 microseconds of processor time is consumed.

Once it is established that the converter is done with its cycle, the data can be read. In the case of an ADC of 8-bit resolution (or less), a single data read operation is sufficient. In the case of converters with more data bits than are available on the bus, a choice of data formats is required, and multiple read operations are needed. The AD574A includes internal logic to permit direct interface to 8-bit or 16-bit data buses, selected by connection of the 12/$\overline{8}$ input. In 16-bit bus applications (12/$\overline{8}$ high) the data lines (DB11 through DB0) may be connected to either the 12 most significant or 12 least significant bits of the data bus. The remaining four bits should be masked in software. The interface to an 8-bit data bus (12/$\overline{8}$ low) is done in a left-justified format. The even address (A0 low) contains the 8MSBs (DB11 through DB4). The odd address (A0 high) contains the 4LSBs (DB3 through DB0) in the upper half of the byte, followed by four trailing zeroes, thus eliminating bit masking instructions.

	D7							D0
XXX0 (EVEN ADDR):	DB11 (MSB)	DB10	DB9	DB8	DB7	DB6	DB5	DB4
XXX1 (ODD ADDR):	DB3	DB2	DB1	DB0 (LSB)	0	0	0	0

Figure 12. AD574A Data Format for 8-Bit Bus

It is not possible to rearrange the AD574A data lines for right-justified 8-bit bus interface.

The AD574A three-state buffers feature access times and data latency times comparable to presently-available memory devices. Therefore, the AD574A can interface directly to many processor buses without the need for wait states or external data buffers.

SPECIFIC PROCESSOR INTERFACE EXAMPLES

6800/6502–Type Systems

The control signals and bus architecture of the 6800 series and 6502 series microprocessors are very similar. In each, the state of the R/$\overline{W}$ signal at the rising edge of the Ø2 (or equivalent) clock establishes whether a memory read or write is in progress. The memory address being exercised is signaled by decoding the address bits to (usually) an active low signal.

This control structure is directly compatible with the AD574A. The R/$\overline{W}$ line can be used for R/$\overline{C}$, the active-low decoded base address (the AD574A occupies two memory locations) is applied to $\overline{CS}$, and Ø2 is used for CE. The least-significant address line ties to the AD574A A0 input.

In this interface, the processor can write to one address (A0 low) to start a full 12-bit conversion or another address (A0 high) to start a short 8-bit conversion. The contents of the data bus are meaningless during these writes. After sufficient time has passed for the conversion to complete, the processor can read the data in the two memory locations occupied by the AD574A. The even location (A0 low) contains the eight MSBs and the odd location contains the four LSBs and four trailing zeroes.

The AD574A may be used directly with 6800 series processors running at clock speeds up to 1.5MHz.

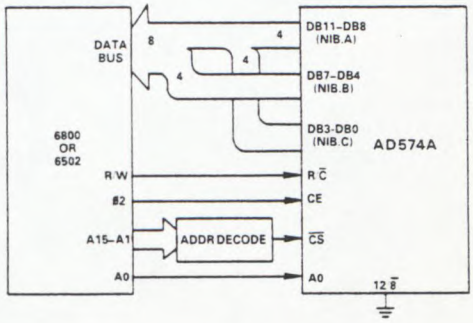


Figure 13. AD574A–6800/6502 Interface Connections

8085A Interface

The 8085A microprocessor uses a multiplexed address/data bus. At the beginning of a machine cycle, this bus contains the low byte of the address being exercised. The ALE output signal is available to strobe a latch to hold the low address byte. For the rest of the machine cycle, this bus carries data to or from the CPU.

The 8085A can use either accumulator I/O or memory-mapping for I/O devices. The system $\overline{RD}$ and $\overline{WR}$ are gated with IO/$\overline{M}$ to provide distinct I/O read and write signals and memory read and write signals. The control signals required for the AD574A are easily derived from the 8085A control bus. $\overline{CS}$ is taken from an address decoder on the high-order address bits. R/$\overline{C}$ can be taken from $\overline{WR}$ (either I/O write or memory write), A0 is tied to the LSB of the address bus, and CE is taken from the output of a NAND gate driven from $\overline{RD}$ and $\overline{WR}$. All bus access and float delay requirements are met for direct bus interface for 8085A clock rates up to 3MHz.

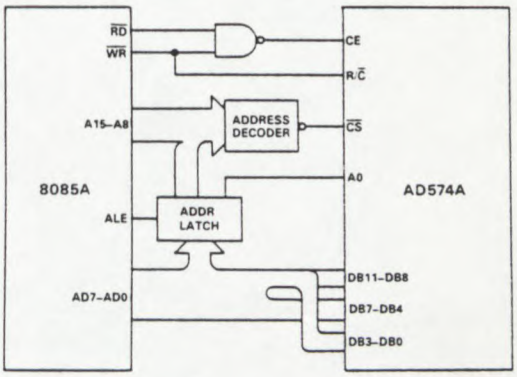


Figure 14. AD574A–8085A Direct Bus Interface

In 8085A systems running at high clock frequencies some external circuitry is required. First, the AD574A delay from CE going low to the data lines going into three-state will cause a bus conflict when the 8085A sends out the low byte of the next instruction address. This conflict will occur if the AD574A data outputs are tied directly to the 8085A bus. In systems where bus transceivers (e.g., 74LS245, 8286, etc.) are used to separate the address and data lines, the conflict is eliminated. The transceivers are disabled at the end of the read cycle and thus isolate the AD574A from the 8085A bus. Since most systems incorporate such buffers, this does not add to system complexity.

A second consideration when interfacing to higher speed 8085A systems is the width of the convert start pulse. The $\overline{WR}$ pulse from a 5MHz 8085A is only guaranteed to be 230 nanoseconds wide and is thus not long enough to initiate a conversion. There are two solutions to this problem. One possibility is to use a dual D-type flip-flop connected as shown in Figure 15 to insert a single wait state in read and write operations directed towards the AD574A. Another solution is to substitute the earlier-occurring S1 and S0 outputs from 8085A for $\overline{RD}$ and $\overline{WR}$ in the circuit of Figure 14 to generate the required control signals. It is important that bus transceivers be employed if S1 and S0 are used for control signals since these signals remain active longer than $\overline{RD}$ and $\overline{WR}$, enabling the AD574A output buffers in read operations for too long, causing potential bus conflicts.

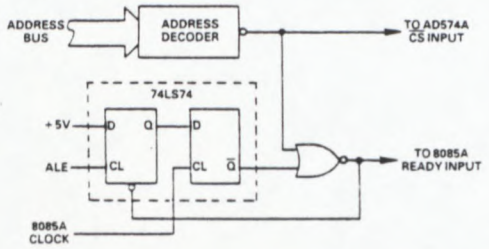


Figure 15. Wait State Generator for 5MHz 8085A Interface

Z-80 System Interface

The Z-80 series of 8-bit microprocessors, like the 8085A, offers both memory-mapped and accumulator I/O capability. While the 8085A only includes two instructions for accumulator I/O (IN and OUT), the Z-80 I/O instruction set is considerably more extensive.

The control signals available on the Z-80 include $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$. The $\overline{RD}$ and $\overline{WR}$ signals indicate direction of data flow while $\overline{MREQ}$ and $\overline{IORQ}$ determine whether the read or write cycle in progress is a memory or I/O cycle. During I/O reads and writes, only 8 address lines are active (as in the 8085A). An interesting feature of the Z-80 is that I/O read and write cycles are automatically extended by one clock cycle (one wait state is inserted) and are thus slower. The Z-80 control signal connections to the AD574A are identical to the 8085A connections.

The AD574A can be interfaced to Z-80 series processors with clock speeds up to 2.5MHz in the memory address space using the $\overline{MWR}$ and $\overline{MRD}$ signals. At higher clock rates (4 and 6MHz), the memory write pulse is not wide enough to properly start a conversion. The extra wait state added during I/O write operations will extend this pulse to a suitable width at clock rates up to 6MHz so that accumulator I/O is possible.

INTERFACING THE AD574A TO THE APPLE II COMPUTER

The AD574A can be used to provide a low-cost precision analog input port for the Apple II microcomputer without the need for additional power supplies or extensive digital interface logic. The AD574A can be mounted on a hobby card designed to plug into an Apple II I/O slot.

Hardware

All required supply voltages and control signals are available on the Apple's peripheral connectors. Each connector contains, on pin 41, a $\overline{DEVICE\ SELECT}$ output which is active when the address bus holds a hexadecimal address between C0n0 and C0nF, where n is equal to the slot number plus 8. This signal can be connected to pin 3 ($\overline{CS}$) of the AD574A. The Φ0 clock on pin 40 of the peripheral connector can be used for the AD574A CE input (pin 6). The AD574A R/$\overline{C}$ input (pin 5) can be driven directly by the R/$\overline{W}$ output available on peripheral connector pin 18. Pin 2 of the peripheral connector, A0, connects directly to the AD574A pin 4. The connections between the peripheral connector and the AD574A are shown in Figure 16.

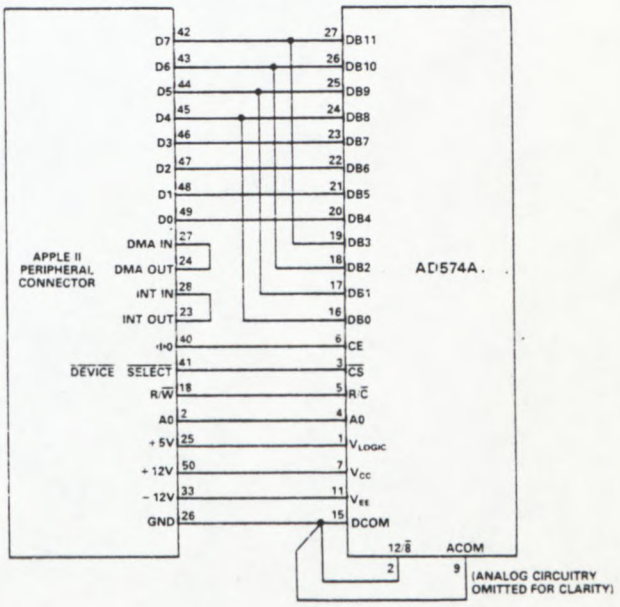


Figure 16. AD574A Connections to Apple II Peripheral Connector

The Apple II represents a relatively hostile electrical environment to the AD574A. The high frequency clocks radiate a large amount of noise which can be inadvertently coupled into analog signal lines. Furthermore, the switching power supply in the Apple is noisy, and this noise will often pollute the analog signals. It is possible, however, by judicious bypassing, decoupling, and ground management, to achieve a data acquisition system with only occasional flicker. A suggested grounding and decoupling scheme is shown in Figure 17.

It is recommended that any signal preamplification used in such a system be physically located outside the Apple cabinet. A full-scale signal range is less susceptible to electromagnetically coupled interference than a smaller signal range would be. Thus, the preferred method is to deliver a buffered, high-level signal to the AD574A through a shielded cable. The ±5V or ±10V

range is suggested. Full-scale and offset trims, if desired, are performed as shown on page 7.

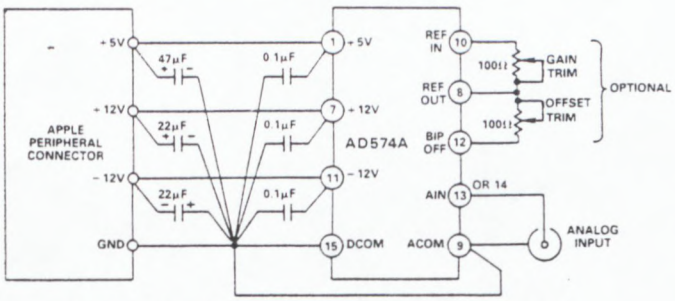


Figure 17. Recommended Grounding Procedure

Software

In this discussion, the AD574A is assumed to be located in I/O slot 2 of the Apple II and be the only device in that slot. The AD574A thus occupies the sixteen locations from $C0A0 through $C0AF, even though only two locations are actually required.

It is possible to operate the AD574A from either machine language or a high-level language. In machine language, the converter is started by writing data to either $C0A0 or $C0A1, using a STA instruction. Writing to $C0A0 will start a full 12-bit conversion cycle; writing to $C0A1 starts an 8-bit cycle. Accumulator contents are unimportant during convert start operations. It is then necessary to wait for the AD574A to finish converting before attempting to read the data. This can be accomplished by loading the accumulator with the value 02 and calling the WAIT subroutine located at $FCA8 in the Apple Monitor.

When data is read, it can be read only 8 bits at a time, as explained on page 10. The sample subroutine below starting at location $4000 performs the control for the AD574A and returns the result in RAM locations $0300 and $0301.

```
4000   A9 02       LDA   #$02
4002   8D A0 C0    STA   $C0A0
4005   20 A8 FC    JSR   $FCA8
4008   AD A0 C0    LDA   $C0A0
400B   8D 00 03    STA   $0300
400E   AD A1 C0    LDA   $C0A1
4012   8D 01 03    STA   $0301
4015   60          RTS
```

Figure 18. Assembly-Language AD574A Control Subroutine

Programs written in Applesoft Basic can also operate the AD574A. Conversion is started by POKEing into location 49312 decimal for a 12-bit conversion (or location 49313 for an 8-bit conversion). Basic executes slowly enough that no delay routines are needed. The output of the AD574A is read by PEEKing into those locations. In order to compute the actual analog voltage, it is necessary to establish the proper weighting for the two bytes read.

The Basic subroutine shown in Figure 19 will accomplish this arithmetic. This routine assumes a ±5V analog signal range and returns the value of actual analog signal voltage in the variable V.

```
100   POKE (49312)
110   A = PEEK (49312): B = PEEK (49313): C = 256
120   A = (A + B/C)/C
130   V = A*10 - 5
140   RETURN
```

Figure 19. Applesoft II Basic Subroutine for AD574A Control

T/H REQUIREMENTS FOR HIGH RESOLUTION APPLICATIONS

The characteristics required for high resolution track-and-hold amplifiers are low feedthrough, low pedestal shifts with changes of input signal or temperature, high linearity, low temperature coefficients, and minimal droop rate.

For sampling a 20kHz signal to 14 bit and 16 bits for example, the following specs are required:

Spec	14 Bit	16 Bit	AD389KD	Units
Aperture Jitter (max)	2.4	0.6	0.4	ns
Slew Rate (max w/20V pk-pk signal)	1.26	1.26	30	V/µs
Feedthrough (1LSB max)	−84.3	−96.3	−86	dB
Droop Rate (1LSB max in 15µs)	40.7	10.2	0.1	µV/µs
Droop Rate (1LSB max in 50µs)	12.2	3.0	0.1	µV/µs
Acquisition Time (to ± 1LSB max) for 20kHz Signal w/15µs ADC	10	10	3-5	µs
Pedestal Shift (max) with Input Signal	−84.3	−96.3	−86	dB
Gain Temperature Coefficient (max) for ± 10°C Ambient Operation	6.1	1.5	2.0	ppm/°C
Thermal Tail (max) within 50µs after Hold	1.2	0.3	0.1	mV
Linearity Error (max)	± 0.0061	0.0015	0.003	%FSR

Table II. T/H Amplifier Requirements vs. AD389 Specs

Aperture Jitter will affect exactly when the switch closes, even though the T/H control line is driven by a very precise clock. All high speed sampled data systems are very dependent on low aperture jitter for digitizing high frequency signals for spectrum analysis and accurate signal reconstruction.

The T/H amplifier slew rate determines the maximum frequency tracking rate and part of the settling time when sampling pulses and square waves. The feedthrough from input to output while in the hold mode should be less than 1LSB. The amplitude of 1LSB of the companion A/D converter for a given input range will vary from 610µV for a 14-bit A/D using a 0 to 10V input range to 4.88mV for a 12-bit A/D using a ±10V input range. The hold mode droop rate should produce less than 1LSB of droop in the output during the conversion time of the A/D converter. For 610µV/LSB, as noted in the example above, for a 50µs 14-bit A/D converter, the maximum droop rate will be 610µV/50µs or 12µV/µs during the 50µs conversion period.

The linearity error should be less than 1LSB over the transfer function, as set by the resolution of the A/D converter. The T/H acquisition time, T/H settling time along, with the conversion time of the A/D converter determines the highest sampling rate. This in turn will determine the highest input signal frequency that can be sampled at twice a cycle per the Nyquist criteria. The pedestal shift due to input signal changes should either be linear, to be seen as a gain error, or negligible as with the feedthrough spec. The temperature coefficients for drift should be low enough such that full accuracy is maintained over some minimum temperature range. The droop rate and pedestal will shift more over temperature above + 70°C (+ 158°F). For commercial and industrial users, these shifts will only appear above the highest temperatures their equipment will ever expect to experience. Most precision instrumentation is installed only in human inhabitable work spaces or in controlled enclosures if the area has a hostile environment.

Minimal thermal tail effects are another requirement of high resolution applications. The self-heating errors induced by the changing current levels in the output stages of T/H amps may cause more than 1LSB of error due to thermal tail effects. The performance of a typical AD389 in contrast to a typical 12-bit T/H circuit is shown in Figures 11a. and 11b. The test circuit is shown in Figure 8.

OFFSET ADJUST TRIM

In most data acquisition systems only one offset adjustment is made. In many cases it is the offset adjust of the ADC that is used to cancel all other accumulated system offsets. The offset or pedestal of the AD389 can be nulled by means of 5kΩ potentiometer between pins 7, 9, and 11. If the offset of the AD389 is not adjusted, then connect pins 7 and 9 to pin 14, the negative supply. Otherwise the high impedance of the null pin together with parasitic capacitances can cause tail effects.

ANALOG DEVICES

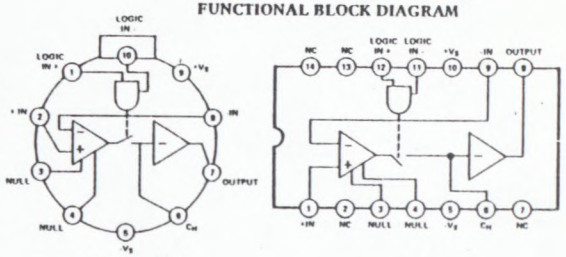
Low Cost Sample/Hold Amplifier

AD582

FEATURES

- Suitable for 12-Bit Applications
- High Sample/Hold Current Ratio: 10^7
- Low Acquisition Time: 6µs to 0.1%
- Low Charge Transfer: <2pC
- High Input Impedance in Sample and Hold Modes
- Connect in Any Op Amp Configuration
- Differential Logic Inputs

AD582 PIN CONFIGURATION AND FUNCTIONAL BLOCK DIAGRAM



10-PIN TO-100
TOP VIEW

14-PIN TO-116
TOP VIEW

PRODUCT DESCRIPTION

The AD582 is a low cost integrated circuit sample and hold amplifier consisting of a high performance operational amplifier, a low leakage analog switch and a JFET integrating amplifier – all fabricated on a single monolithic chip. An external holding capacitor, connected to the device, completes the sample and hold function.

With the analog switch closed, the AD582 functions like a standard op amp; any feedback network may be connected around the device to control gain and frequency response. With the switch open, the capacitor holds the output at its last level, regardless of input voltage.

Typical applications for the AD582 include sampled data systems, D/A deglitchers, analog de-multiplexers, auto null systems, strobed measurement systems and A/D speed enhancement.

The device is available in two versions: the "K" specified for operation over the 0 to +70°C commercial temperature range and the "S" specified over the extended temperature range, -55°C to +125°C. All versions may be obtained in either the hermetic sealed, TO-100 can or the TO-116 DIP.

PRODUCT HIGHLIGHTS

1. The specially designed input stage presents a high impedance to the signal source in both sample and hold modes (up to ±12V). Even with signal levels up to $\pm V_S$, no undesirable signal inversion, peaking or loss of hold voltage occurs.
2. The AD582 may be connected in any standard op amp configuration to control gain or frequency response and provide signal inversion, etc.
3. The AD582 offers a high, sample-to-hold current ratio: 10^7. The ratio of the available charging current to the holding leakage current is often used as a figure of merit for a sample and hold circuit.
4. The AD582 has a typical charge transfer less than 2pC. A low charge transfer produces less offset error and permits the use of smaller hold capacitors for faster signal acquisition.
5. The AD582 provides separate analog and digital grounds, thus improving the device's immunity to ground and switching transients.

14

SPECIFICATIONS (typical @ +25°C, V_S = ±15V and C_H = 1000pF, A = +1 unless otherwise specified)

MODEL	AD582K	AD582S
SAMPLE/HOLD CHARACTERISTICS		
Acquisition Time, 10V Step to 0.1%, C_H = 100pF	6µs	*
Acquisition Time, 10V Step to 0.01%, C_H = 1000pF	25µs	*
Aperture Time, 20V p-p Input, Hold 0V	200ns	*
Aperture Jitter, 20V p-p Input, Hold 0V	15ns	*
Settling Time, 20V p-p Input, Hold 0V, to 0.01%	0.5µs	*
Droop Current, Steady State, ±10V_{OUT}	100pA max	*
Droop Current, T_{min} to T_{max}	1nA	150nA max
Charge Transfer	5pC max (1.5pC typ)	*
Sample to Hold Offset	0.5mV	*
Feedthrough Capacitance 20V p-p, 10kHz Input	0.05pF	*
TRANSFER CHARACTERISTICS		
Open Loop Gain V_{OUT} = 20V p-p, R_L = 2k	25k min (50k typ)	*
Common Mode Rejection V_{CM} = 20V p-p	60dB min (70dB typ)	*
Small Signal Gain Bandwidth V_{OUT} = 100mV p-p, C_H = 100pF	1.5MHz	*
Full Power Bandwidth V_{OUT} = 20V p-p, C_H = 100pF	70kHz	*
Slew Rate V_{OUT} = 20V p-p, C_H = 100pF	3V/µs	*
Output Resistance Hold Mode, I_{OUT} = ±5mA	12Ω	*
Linearity V_{OUT} = 20V p-p, R_L = 2k	±0.01%	*
Output Short Circuit Current	±25mA	*
ANALOG INPUT CHARACTERISTICS		
Offset Voltage	6mV max (2mV typ)	*
Offset Voltage, T_{min} to T_{max}	4mV	8mV max (5mV typ)
Bias Current	3µA max (1.5µA typ)	*
Offset Current	300nA max (75nA typ)	*
Offset Current, T_{min} to T_{max}	100nA	400nA max (100nA typ)
Input Capacitance, f = 1MHz	2pF	*
Input Resistance, Sample or Hold 20V p-p Input, A = +1	30MΩ	*
Absolute Max Diff Input Voltage	30V	*
Absolute Max Input Voltage, Either Input	±V_S	*
DIGITAL INPUT CHARACTERISTICS		
+Logic Input Voltage Hold Mode, T_{min} to T_{max}, -Logic @ 0V	+2V min	*
Sample Mode, T_{min} to T_{max}, -Logic @ 0V	+0.8V max	*
+Logic Input Current Hold Mode, +Logic @ +5V, -Logic @ 0V	1.5µA	*
Sample Mode, +Logic @ 0V, -Logic @ 0V	1nA	*
-Logic Input Current Hold Mode, +Logic @ +5V, -Logic @ 0V	24µA	*
Sample Mode, +Logic @ 0V, -Logic @ 0V	4µA	*
Absolute Max Diff Input Voltage, +L to -L	+15V/-6V	*
Absolute Max Input Voltage, Either Input	±V_S	*
POWER SUPPLY CHARACTERISTICS		
Operating Voltage Range	±9V to ±18V	±9V to ±22V
Supply Current, R_L = ∞	4.5mA max (3mA typ)	*
Power Supply Rejection, ΔV_S = 5V, Sample Mode (see next page)	60dB min (75dB typ)	*
TEMPERATURE RANGE		
Specified Performance	0 to +70°C	-55°C to +125°C
Operating	-25°C to +85°C	-55°C to +125°C
Storage	-65°C to +150°C	*
Lead Temperature (Soldering, 15 sec)	+300°C	*
PACKAGE OPTION[1]		
"H" Package: TO-100	AD582KH	AD582SH
"D" Package: TO-116 Style (D14A)	AD582KD	AD582SD

NOTES
*Specifications same as AD582K.
[1] See Section 19 for package outline information.
Specifications subject to change without notice.

APPLYING THE AD582

Both the inverting and non-inverting inputs are brought out to allow op amp type versatility in connecting and using the AD582. Figure 1 shows the basic non-inverting unity gain connection requiring only an external hold capacitor and the usual power supply bypass capacitors. An offset null pot can be added for more critical applications.

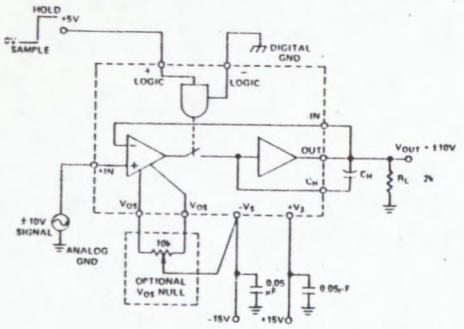


Figure 1. Sample and Hold with A = +1

Figure 2 shows a non-inverting configuration where voltage gain, A_V, is set by a pair of external resistors. Frequency shaping or non-linear networks can also be used for special applications.

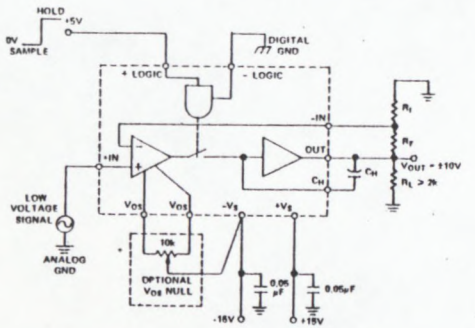


Figure 2. Sample and Hold with A = (1 + R_F/R_I)

The hold capacitor, C_H, should be a high quality polystyrene (for temperatures below +85°C) or Teflon type with low dielectric absorption. For high speed, limited accuracy applications, capacitors as small as 100pF may be used. Larger values are required for accuracies of 12 bits and above in order to minimize feedthrough, sample to hold offset and droop errors (see Figure 6). Care should be taken in the circuit layout to minimize coupling between the hold capacitor and the digital or signal inputs.

In the hold mode, the output voltage will follow any change in the -V_S supply. Consequently, this supply should be well regulated and filtered.

Biasing the +Logic Input anywhere between -6V to +0.8V with respect to the -Logic will set the sample mode. The hold mode will result from any bias between +2.0V and (+V_S - 3V). The sample and hold modes will be controlled differentially with the absolute voltage at either logic input ranging from -V_S to within 3V of +V_S (+V_S - 3V). Figure 3 illustrates some examples of the flexibility of this feature.

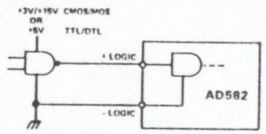


Figure 3A. Standard Logic Connection

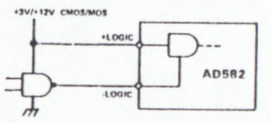


Figure 3B. Inverted Logic Sense Connection

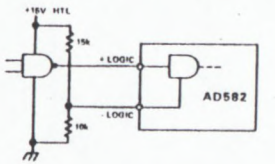


Figure 3C. High Threshold Logic Connection

14

DEFINITION OF TERMS

Figure 4 illustrates various dynamic characteristics of the AD582.

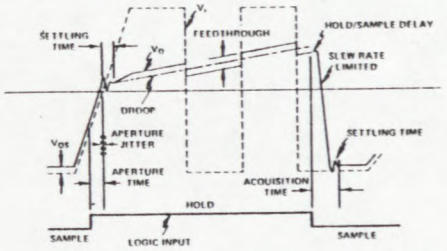


Figure 4. Pictorial Showing Various S/H Characteristics

Aperture Time is the time required after the "hold" command until the switch is fully open and produces a delay in the effective sample timing. Figure 5 is a plot giving the maximum frequency at which the AD582 can sample an input with a given accuracy (lower curve).

Aperture Jitter is the uncertainty in Aperture Time. If the Aperture Time is "tuned out" by advancing the sample-to-hold command 200ns with respect to the input signal, the Aperture Jitter now determines the maximum sampling frequency (upper curve of Figure 5).

Acquisition Time is the time required by the device to reach its final value within a given error band after the sample command has been given. This includes switch delay time, slewing time and settling time for a given output voltage change.

Droop is the change in the output voltage from the "held" value as a result of device leakage. In the AD582, droop can be in either the positive or negative direction. Droop rate may be calculated from droop current using the following formula:

$$\frac{\Delta V}{\Delta T}\ (\text{Volts/sec}) = \frac{I(\text{pA})}{C_H(\text{pF})}$$

(See also Figure 6.)

Feedthrough is that component of the output which follows the input signal *after* the switch is open. As a percentage of the input, feedthrough is determined as the ratio of the feedthrough capacitance to the hold capacitance (C_F/C_H).

Charge Transfer is the charge transferred to the holding capacitor from the interelectrode capacitance of the switch when the unit is switched to the hold mode. The charge transfer generates a sample-to-hold offset where:

$$\text{S/H Offset (V)} = \frac{\text{Charge (pC)}}{C_H(\text{pF})}$$

(See also Figure 6.)

Sample-to-Hold Offset is that component of D.C. offset independent of C_H (see Figure 6). This offset may be nulled using a null pot, however, the offset will then appear during the sampling mode.

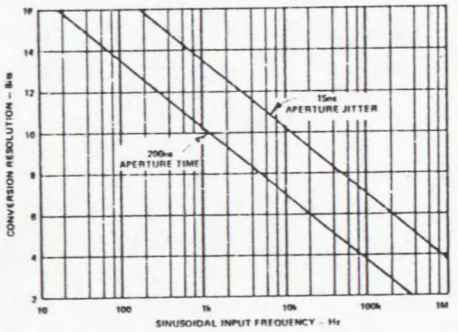


Figure 5. Maximum Frequency of Input Signal for ½LSB Sampling Accuracy

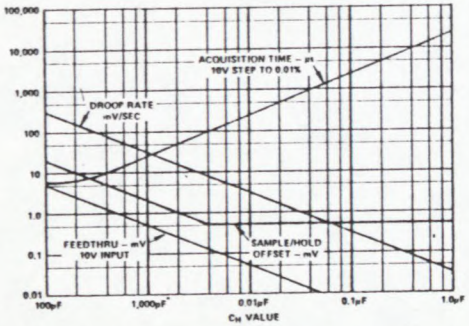


Figure 6. Sample-and-Hold Performance as a Function of Hold Capacitance

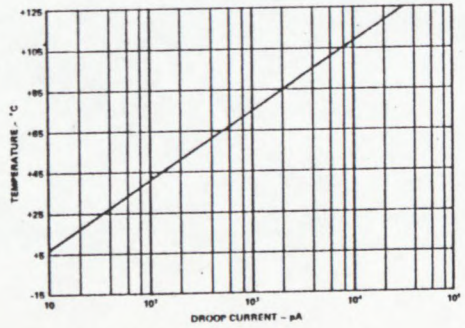


Figure 7. Droop Current vs. Temperature

ANALOG DEVICES

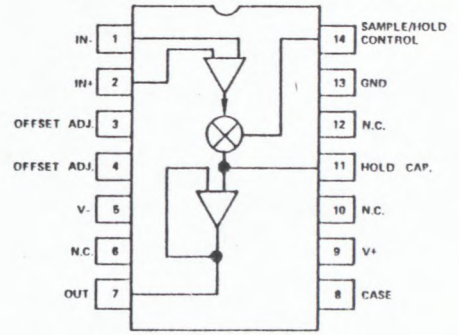
IC Sample-and-Hold Gated Op Amp

AD583

FEATURES

- High Sample-to-Hold Current Ratio: 10^6
- High Slew Rate: 5V/μs
- High Bandwidth: 2MHz
- Low Aperture Time: 50ns
- Low Charge Transfer: 10pC
- DTL/TTL Compatible
- May Be Used as Gated Op Amp

AD583 FUNCTIONAL BLOCK DIAGRAM



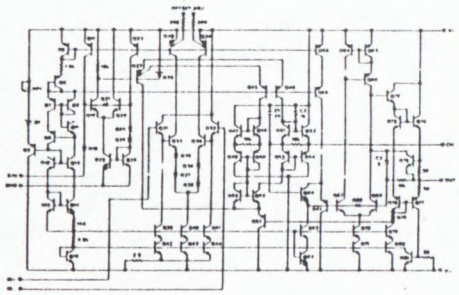
TO-116

PRODUCT DESCRIPTION

The AD583 is a monolithic sample-and-hold circuit consisting of a high performance operational amplifier in series with an ultra-low leakage analog switch and unity gain amplifier. An external holding capacitor, connected to the switch output, completes the sample-and-hold or track-and-hold function.

With the analog switch closed, the AD583 functions like a standard op amp; any feedback network may be connected around the device to control gain and frequency response. With the switch open the capacitor holds the output at its previous level.

The AD583 may also be used as a versatile operational amplifier with a gated output for applications such as analog switches, peak holding circuits, etc.



Schematic Diagram

PRODUCT HIGHLIGHTS

1. Sample-and-hold operation is obtained with the addition of one external capacitor.
2. Low charge transfer (10pC) and high sample-to-hold current ratio insure accurate tracking.
3. Any gain or frequency response is available using standard op amp feedback networks.
4. High slew rate and low aperture time permit sampling of rapidly changing signals.
5. Output, gated through a low leakage analog switch, also makes the AD583 useful for applications such as analog switches, peak holding circuits, etc.

14

IH6108 APPLICATION INFORMATION (Continued)

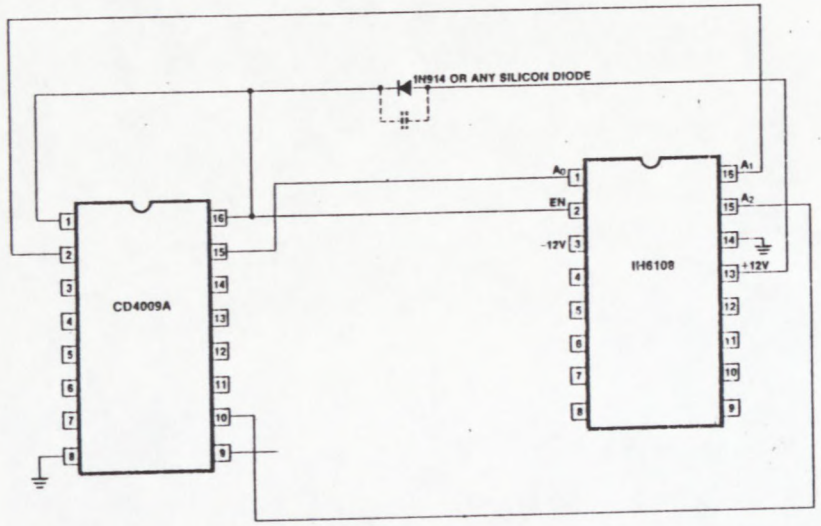


Figure 7. IH6108 Connection Diagram with ENable Input Strobing for less than ±15V Supply Operation.

III. Peak-to-Peak Signal Handling Capability

The IH6108 can handle input signals up to ±14V (actually −15V to +14.3V because of the input protection diode) when using ±15V supplies.

The electrical specifications of the IH6108 are guaranteed for ±10V signals, but the specifications have very minor changes for ±14V signals. The notable changes are slightly lower $r_{DS(on)}$ and slightly higher leakages.

IH6116

16-Channel CMOS Analog Multiplexer

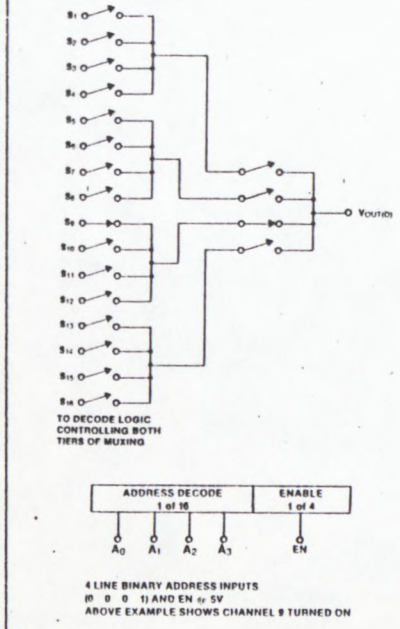
FEATURES

- Pin compatible with DG506, HI-506 & AD7506
- Ultra Low Leakage — $I_{D(off)} \leq 100pA$
- ±11 analog signal range
- $r_{DS(on)}$ <700 ohms over full signal and temperature range
- Break-before-make switching
- TTL and CMOS compatible Address control
- Binary Address control (4 Address inputs control 16 channels)
- Two tier submultiplexing to facilitate expandability
- Power supply quiescent current less than 100μA
- No SCR latchup

GENERAL DESCRIPTION

The IH6116 is a CMOS monolithic, one of 16 multiplexer. The part is a plug-in replacement for the DG506. Four line binary decoding is used so that the 16 channels can be controlled by 4 Address inputs; additionally a fifth input is provided to use as system enable. When the ENable input is high (5V) the channels are sequenced by the 4 line Address inputs, and when low (0V), all channels are off. The 4 Address inputs are controlled by TTL logic or CMOS logic elements with a "0" corresponding to any voltage less than 0.8V and a "1" corresponding to any voltage greater than 3.0V. Note that the ENable input must be taken to 5V to enable the system and less than 0.8V to disable the system.

FUNCTIONAL DIAGRAM

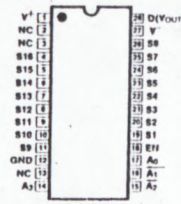


DECODE TRUTH TABLE

A_3	A_2	A_1	A_0	EN	ON SWITCH
x	x	x	x	0	NONE
0	0	0	0	1	1
0	0	0	1	1	2
0	0	1	0	1	3
0	0	1	1	1	4
0	1	0	0	1	5
0	1	0	1	1	6
0	1	1	0	1	7
0	1	1	1	1	8
1	0	0	0	1	9
1	0	0	1	1	10
1	0	1	0	1	11
1	0	1	1	1	12
1	1	0	0	1	13
1	1	0	1	1	14
1	1	1	0	1	15
1	1	1	1	1	16

Logic "1" = $V_{AH} \geq 3.0V$ $V_{ENH} \geq 4.5V$
Logic "0" = $V_{AL} \leq 0.8V$

PIN CONFIGURATION



ORDERING INFORMATION

Ceramic package available as special order only (IH6116MDI/CDI)

PART NUMBER	TEMPERATURE RANGE	PACKAGE
IH6116MJI	−55°C to +125°C	28 pin CERDIP
IH6116CJI	0°C to 70°C	28 pin CERDIP
IH6116CPI	0°C to 70°C	28 pin Plastic DIP

ABSOLUTE MAXIMUM RATINGS

V_{IN} (A, EN) to Ground ..... −15V to 15V
V_S or V_D to V^+ ..... 0, −32V
V_S or V_D to V^- 0, 32V
V^+ to Ground 16V
V^- to Ground ..... −16V
Current (Any Terminal) ..... 30 mA
Current (Analog Source or Drain) ..... 20 mA
Operating Temperature ..... −55 to 125°C
Storage Temperature ..... −65 to 150°C
Lead Temperature (Soldering, 10 secs) ..... 300°C
Power Dissipation (Package)* ..... 1200 mW

*All leads soldered or welded to PC board. Derate 10 mW/°C above 70°C.

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions above those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS $V^+ = 15V$, $V^- = -15V$, $V_{EN} = +5V$ (Note 1), Ground = 0V, unless otherwise specified.

	CHARACTERISTIC		MEASURED TERMINAL	NO TESTS PER TEMP	TYP 25°C	MAX LIMITS M SUFFIX −55°C	M SUFFIX 25°C	M SUFFIX 125°C	C SUFFIX 0°C	C SUFFIX 25°C	C SUFFIX 70°C	UNIT	TEST CONDITIONS	
SWITCH	$r_{DS(ON)}$		S to D	16	480	600	600	700	650	650	750	Ω	$V_D = 10V$, $I_S = -10mA$	Sequence each switch on
				16	300	600	600	700	650	650	750		$V_D = -10V$, $I_S = 10mA$	$V_{AL} = 0.8V$, $V_{AH} = 3V$
	$\Delta r_{DS(ON)}$				20							%	$\Delta r_{DS(on)} = \frac{r_{DS(on)max} - r_{DS(on)min}}{r_{DS(on)avg}}$	$V_S = \pm 10V$
	$I_{S(OFF)}$		S	16	0.01		0.1	50		0.2	50	nA	$V_S = 10V$, $V_D = -10V$	$V_{EN} = 0$
				16	0.01		0.1	50		0.2	50		$V_S = -10V$, $V_D = 10V$	
	$I_{D(OFF)}$		D	1	0.1		0.2	100		0.4	100		$V_D = 10V$, $V_S = -10V$	
				1	0.1		0.2	100		0.4	100		$V_D = -10V$, $V_S = 10V$	
	$I_{D(ON)}$		D	16	0.1		0.2	100		0.4	100		$V_{S(All)} = V_D = 10V$	Sequence each switch on
				16	0.1		0.2	100		0.4	100		$V_{S(All)} = V_D = -10V$	$V_{AL} = 0.8V$, $V_{AH} = 3V$
INPUT	$I_{A(on)}$ or $I_{A(off)}$			4	.01		−10	−30		−10	−30	μA	$V_A = 3.0V$	
				4	.01		10	30		10	30		$V_A = 15V$	
	I_A		A_0 A_1 A_2 A_3	4			−10	−30		−10	−30		$V_{EN} = 5V$	All $V_A = 0$
			EN	1			−10	−30		−10	−30		$V_{EN} = 0$	
DYNAMIC	t_{trans}		D		0.6		1					μs	See Fig. 1	
	t_{open}		D		0.2								See Fig. 2	
	$t_{EN(on)}$		D		0.8		1.5						See Fig. 3	
	$t_{EN(off)}$		D		0.3		1							
	"OFF" Isolation		D		60							dB	$V_{EN} = 0$, $R_L = 200\Omega$, $C_L = 3pF$, $V_S = 3$ VRMS, f = 500 kHz	
	C_s(OFF)		S		5							pF	$V_S = 0$	$V_{EN} = 0$, f = 140 kHz to 1 MHz
	C_d(OFF)		D		40								$V_D = 0$	
	C_{ds}(OFF)		D to S		1								$V_S = 0$, $V_D = 0$	All $V_A = 0$ or 3V
SUPPLY	Supply	+	V^+	1	55		200			1000		μA	$V_{EN} = 5V$	
	Current	−	V^-	1	2		100			1000				
	Standby	+	V^+	1	1		100			1000			$V_{EN} = 0$	
	Current	−	V^-	1	1		100			1000				

NOTE 1: See Section V. Enable Input Strobing Levels.

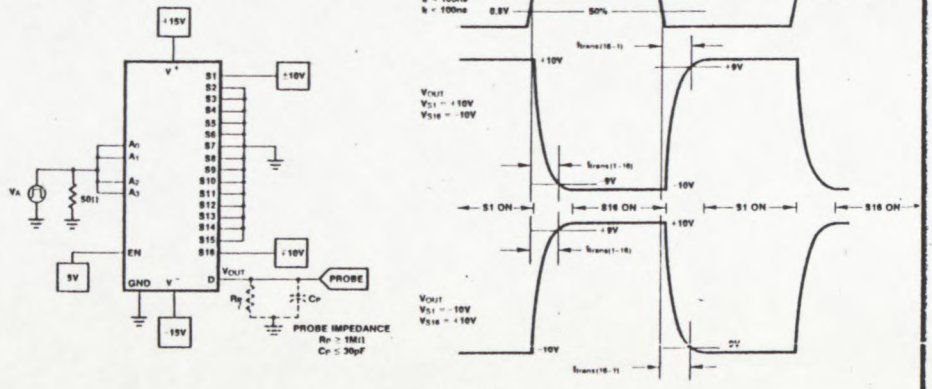
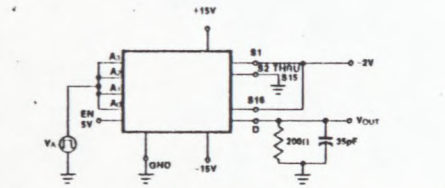


Figure 1



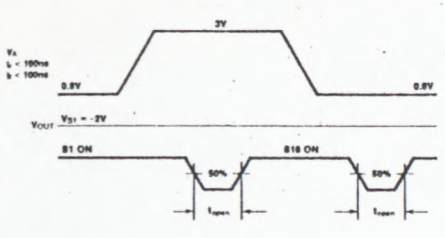
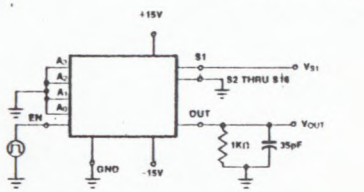


Figure 2



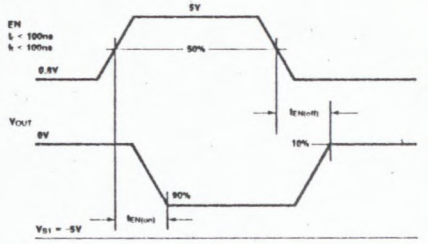
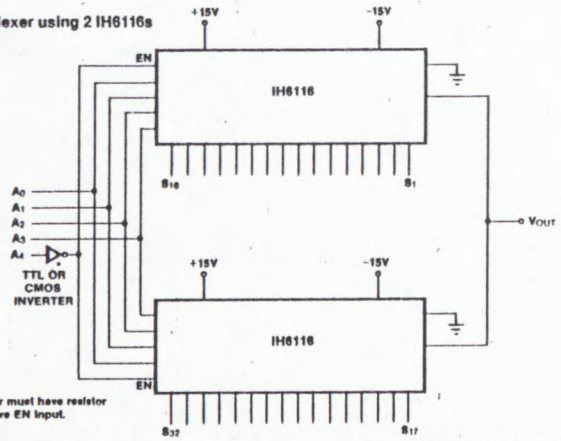


Figure 3

IH6116 APPLICATIONS

I. 1 out of 32 channel multiplexer using 2 IH6116s



DECODE TRUTH TABLE

A_4	A_3	A_2	A_1	A_0	ON SWITCH
0	0	0	0	0	S1
0	0	0	0	1	S2
0	0	0	1	0	S3
0	0	0	1	1	S4
0	0	1	0	0	S5
0	0	1	0	1	S6
0	0	1	1	0	S7
0	0	1	1	1	S8
0	1	0	0	0	S9
0	1	0	0	1	S10
0	1	0	1	0	S11
0	1	0	1	1	S12
0	1	1	0	0	S13
0	1	1	0	1	S14
0	1	1	1	0	S15
0	1	1	1	1	S16

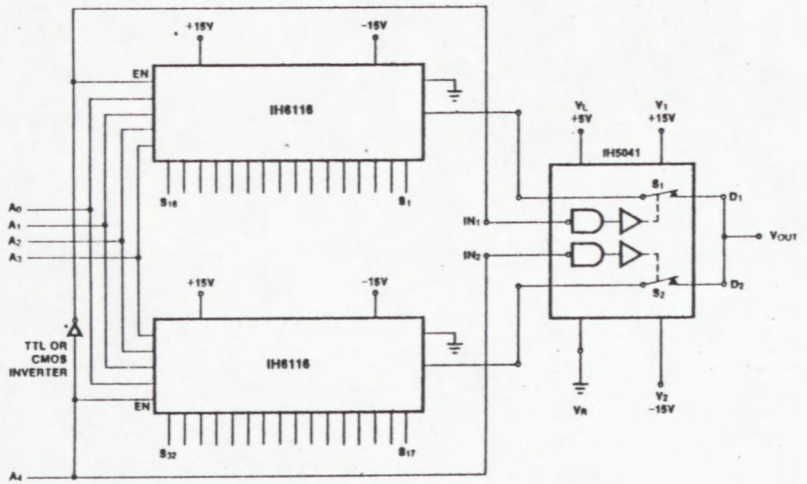
DECODE TRUTH TABLE

A_4	A_3	A_2	A_1	A_0	ON SWITCH
1	0	0	0	0	S17
1	0	0	0	1	S18
1	0	0	1	0	S19
1	0	0	1	1	S20
1	0	1	0	0	S21
1	0	1	0	1	S22
1	0	1	1	0	S23
1	0	1	1	1	S24
1	1	0	0	0	S25
1	1	0	0	1	S26
1	1	0	1	0	S27
1	1	0	1	1	S28
1	1	1	0	0	S29
1	1	1	0	1	S30
1	1	1	1	0	S31
1	1	1	1	1	S32

Figure 4

IH6116 APPLICATIONS (Continued)

II. 1 out of 32 channel multiplexer using 2 IH6116s; using an IH5041 for submultiplexing



*TTL gate must have pullup resistor to +5V to drive EN inputs

DECODE TRUTH TABLE

A_4	A_3	A_2	A_1	A_0	ON SWITCH
0	0	0	0	0	S1
0	0	0	0	1	S2
0	0	0	1	0	S3
0	0	0	1	1	S4
0	0	1	0	0	S5
0	0	1	0	1	S6
0	0	1	1	0	S7
0	0	1	1	1	S8
0	1	0	0	0	S9
0	1	0	0	1	S10
0	1	0	1	0	S11
0	1	0	1	1	S12
0	1	1	0	0	S13
0	1	1	0	1	S14
0	1	1	1	0	S15
0	1	1	1	1	S16

DECODE TRUTH TABLE

A_4	A_3	A_2	A_1	A_0	ON SWITCH
1	0	0	0	0	S17
1	0	0	0	1	S18
1	0	0	1	0	S19
1	0	0	1	1	S20
1	0	1	0	0	S21
1	0	1	0	1	S22
1	0	1	1	0	S23
1	0	1	1	1	S24
1	1	0	0	0	S25
1	1	0	0	1	S26
1	1	0	1	0	S27
1	1	0	1	1	S28
1	1	1	0	0	S29
1	1	1	0	1	S30
1	1	1	1	0	S31
1	1	1	1	1	S32

Figure 5

IH6116 APPLICATIONS (Continued)

III. 1 out of 64 multiplexer using 4 1/16s and IH5053 as submultiplexer

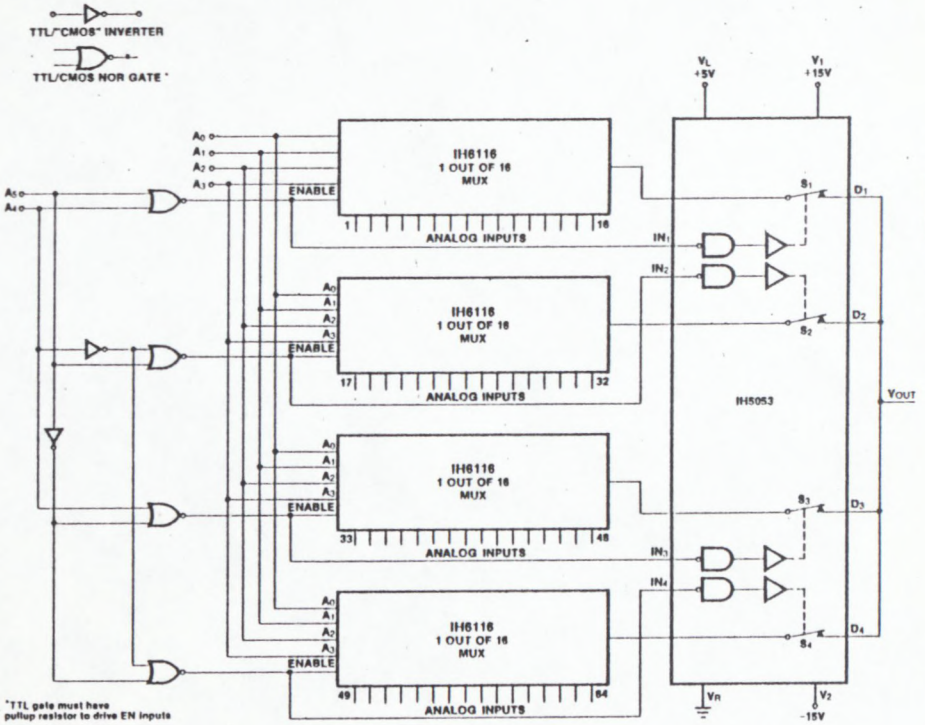


Figure 6

IV. General note on expandability of IH6116

The IH6116 is a two tier multiplexer, where sixteen input channels are routed to a common output in blocks of 4. Each block of 4 input channels is routed to one common output channel, and thus the submultiplexed system looks like 4 blocks of 4 inputs routed to 4 different outputs with the 4 outputs tied together. Thus 20 switches are needed to handle the 16 channels of information. The advantage of this is lower output capacity and leakage that would be possible using a system with all 16 channels tied to one common output. Also the expandability into 32, 64, 128, etc. is facilitated. Figures 4, 5, and 6 show how the IH6116 is expanded.

Figure 4 shows a 1 of 32 multiplexer, using 2 IH6116s. Since the 6116 is itself a 2 tier MUX, the system as shown is basically a 2 tier system. The four output channels of each 6116 are tied together so that 8 channels are tied to the V_{OUT} common point. Since only one channel of information is on at a time, the common output will consist of 7 OFF channels and 1 ON channel. Thus the output leakage will correspond to 7 $I_{D(offs)}$ and 1 $I_{D(on)}$, or about 1.0 nA of typical leakage at room temperature. Thruput speed will be typically 0.8μs for t_{on} and 0.3μs for t_{off}. Thruput channel resistance will be in the 500Ω area.

Figure 5 shows the 1 of 32 MUX of Figure 4, with a third tier of submultiplexing added to further reduce leakage and output capacity. The IH5041 has typical ON resistances of 50Ω (max. is 75Ω) so it only increases thruput channel resistance from the 500 ohms of Figure 4 to about 550 ohms for Figure 5. Thruput channel speed is a little slower by about 0.5μs for both ON and OFF time, and output leakage is about 0.2 nA.

Figure 6 shows a 1 of 64 MUX using 3 tier MUXing (similar to Figure 5). The Intersil IH5053 is used to get the third tier of MUXing. The V_{OUT} point will see 3 OFF channels and 1 ON channel at any one time, so that the typical leakages will be about 0.4 nA. Thruput channel resistance will be in the 550 ohm area with thruput switching speeds about 1.3μs for ON time and 0.8μs for OFF time.

The IH5053 was chosen as the third tier of the MUX because it will switch the same AC signals as the IH6116 (typically plus and minus 15V) and uses break before make switching. Also power supply quiescent currents are on the order of 1-2μA so that no excessive system power is generated. Note that the logic of the 5053 is such that it can be tied directly to the ENable input (as shown in the figures) with no extra logic being required.

V. Enable input strobing levels

The enable input (EN) acts as an enabling or disabling pin for the IH6116 when used as a 16 channel MUX. However, when expanding the MUX to more than 16 channels, the EN pin acts as another address input. Figures 4 and 5 show the EN pin used as the A4 input.

For the system to function properly the EN input (pin 18) must go to 5V ±5% for the high state and less than 0.8V for the low state. When using TTL logic, a pull-up resistor of 1kΩ or less should be used to pull the output voltage up to 5V. When using CMOS logic, the high state goes to the power supply so no pull-up is required.

If used on high voltage logic supplies, EN should be at least 0.7V below V + at all times. See IH6108 data sheet for details.

APPLICATION NOTES

Further information may be found in:

A003 "Understanding and Applying the Analog Switch," by Dave Fullagar

A006 "A New CMOS Analog Gate Technology," by Dave Fullagar

A020 "A Cookbook Approach to High Speed Data Acquisition and Microprocessor Interfacing," by Ed Stieger

R009 "Reduce CMOS Multiplexer Troubles Through Proper Device Selection," by Dick Wilenken

NOTE: This multiplexer does not require external resistors and/or diodes to eliminate what is commonly known as a latch up or SCR action. Because of this fact, the $r_{DS(ON)}$ of the switch is maintained at specified values.

IH6201
Dual CMOS Driver/ Voltage Translator

FEATURES

- **Driven direct from TTL or CMOS logic**
- **Translates logic levels up to 30V levels**
- **Switches 20V_{ACPP} signals when used in conjunction with Intersil IH401A Varafet (as an analog gate)**
- **$t_{ON} \leq 300nS$ & $t_{OFF} \leq 200nS$ for 30V level shifts**
- **Quiescent supply current ≤ 100μa for any state (d.c.)**
- **Provides both normal & inverted outputs**

GENERAL DESCRIPTION

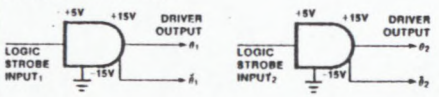
The IH6201 is a CMOS, Monolithic, Dual Voltage Translator; it takes the low level TTL or CMOS logic level and converts them to higher levels (i.e. to ±15V swings). This translator is typically used in making solid state switches, or analog gates.

When used in conjunction with the Intersil IH401 family Varafets, the combination makes a complete solid state switch capable of switching signals up to 22Vpp and up to 20MHz in frequency. This switch is a "break-before-make" type (i.e. t_{off} time < t_{on} time). The combination has typical t_{off} ≈ 80nS and typ. t_{on} ≈ 200nS for signals up to 20Vpp in amplitude.

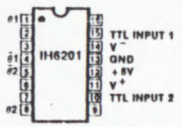
A TTL "1" input strobe will force the θ driver output up to V^+ level; the $\bar{\theta}$ output will be driven down to the V^- level. When the TTL input goes to "0", the θ output goes to V^- and $\bar{\theta}$ goes to V^+; thus θ and $\bar{\theta}$ are 180° out of phase with each other. These complementary outputs can be used to create a wide variety of functions such as SPDT and DPDT switches, etc.; alternatively the complementary outputs can be used to drive an N and P channel Mosfet, to make a complete Mosfet analog gate.

The driver typically uses +5V and ±15V power supplies; however a wide range of V^+ and V^- is possible, however $V^+ > 5V$ is necessary for the driver to work properly.

BLOCK DIAGRAM



PIN CONFIGURATION



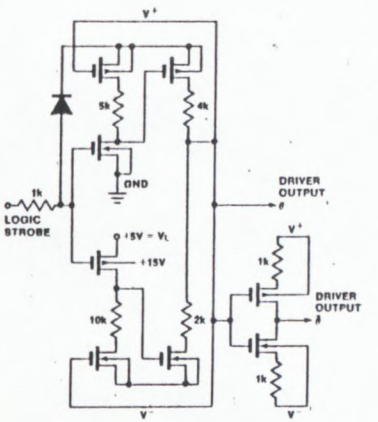
OUTLINE DWGS
DE, JE, PE

ORDERING INFORMATION

PART NUMBER	TEMPERATURE RANGE
*IH6201CDE	0°C to 70°C
*IH6201MDE	-55°C to +125°C
IH6201CJE	0°C to 70°C
IH6201MJE	-55°C to 125°C
IH6201CPE	0°C to 70°C

*Special Order Only

SCHEMATIC DIAGRAM (ONE CHANNEL)



FAIRCHILD

A Schlumberger Company

96LS488
General Purpose Interface Bus (GPIB) Circuit

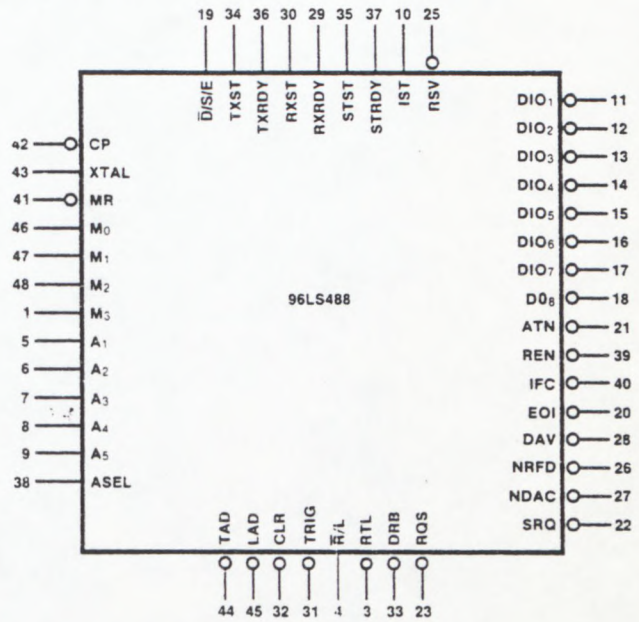
Digital Integrated Circuits

The 96LS488 is a TTL LSI circuit containing all of the logic necessary to interface Talk, Listen and Talk/Listen type instruments and system components in accordance with the IEEE-488 standard for programmable instrumentation. All outputs that drive the IEEE-488 bus are guaranteed to sink 48 mA, and all bus inputs have Schmitt triggers and bus terminating networks. All pins that interface to the instrument logic are LSTTL compatible.

The 96LS488 has programming inputs that determine whether it is to be a talker, listener or both, single or dual address, high or low speed, etc., according to the instrument and system requirements. It operates with a minimum of external support logic and readily interfaces with most microprocessors. It operates from a single 5.0 V supply and a 10 MHz single phase clock and is capable of operating the bus handshake at the full 1 MHz data rate. It offers a variety of handshaking and status connections to the instrument logic for versatility and ease of design.

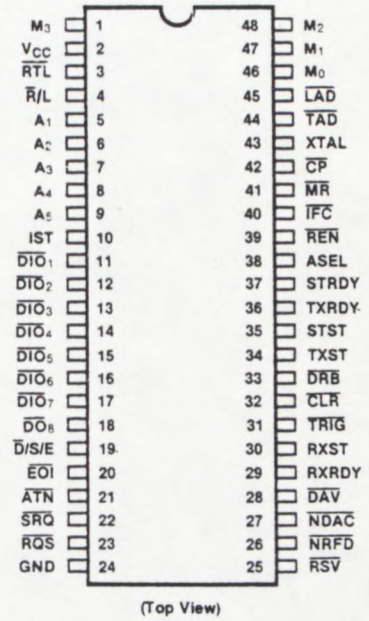
- SINGLE 5.0 V SUPPLY
- COMPLETE SOURCE AND ACCEPTOR HANDSHAKE LOGIC
- SAME OR SEPARATE TALK AND LISTEN ADDRESS
- SECONDARY ADDRESS CAPABILITY
- TALK ONLY OR LISTEN ONLY CAPABILITY
- SOURCE HANDSHAKE DELAY PROGRAMMABLE FOR LOW OR HIGH SPEED
- SERIAL POLL CAPABILITY
- PARALLEL POLL CAPABILITY
- SYNC TRIGGER AND DEVICE CLEAR OUTPUTS
- IMPLEMENTS REMOTE/LOCAL FUNCTION
- ON-CHIP CLOCK OSCILLATOR
- SERVICE REQUEST INTERRUPT FACILITY
- ALL BUS I/O SIGNALS COMPLY WITH THE IEEE-488 (1980) AND IEC-625-1 INPUT THRESHOLD, TERMINATION AND OUTPUT SPECIFICATIONS
- ALL INSTRUMENT INTERFACE SIGNALS ARE LSTTL COMPATIBLE
- GPIB PINS PRESENT NO ELECTRICAL LOAD WHEN DEVICE IS POWERED OFF

Logic Symbol



V_{CC} = Pin 2
Gnd = Pin 24

Connection Diagram
48-Pin DIP



(Top View)

Fairchild Camera and Instrument Corporation
464 Ellis Street, Mountain View, California 94042
415/962-5011 TWX 910 379 6435

Input Loading/Fan-Out

Pin Names	Description	96LS (U.L.)** HIGH/LOW
A_1–A_5	Address Inputs	0.5/0.25
$\overline{ATN}$	Attention Input (Active LOW)	T*/1.4
$\overline{CP}$	Clock Input (Active Falling Edge)	0.5/0.25
$\overline{IFC}$	Interface Clear Input (Active LOW)	T*/1.4
IST	Instrument Status Input	0.5/0.25
$M_0 - M_3$	Mode Control Inputs	0.5/0.25
$\overline{MR}$	Master Reset Input (Active LOW)	0.5/0.25
$\overline{REN}$	Remote Enable Input (Active LOW)	T*/1.4
$\overline{RSV}$	Request Service Input (Active LOW)	0.5/0.25
$\overline{RTL}$	Return to Local Input (Active LOW)	0.5/0.25
RXRDY	Receiver Ready Input (Active HIGH)	0.5/0.25
STRDY	Status Ready Input (Active HIGH)	0.5/0.25
TXRDY	Transmitter Ready Input (Active HIGH)	0.5/0.25
$\overline{DAV}$	Data Valid Input (Active LOW)	T*/1.4
	as Output (Active LOW)	130/30
$\overline{DIO_1}$–$\overline{DIO_7}$	Data Inputs	T*/1.4
	as Outputs (Active LOW)	T*/30
$\overline{DO_8}$	Data Output (Active LOW)	T*/30
$\overline{EOI}$	End or Identify Input (Active LOW)	T*/1.4
$\overline{NDAC}$	Not Data Accepted Input (Active LOW)	T*/1.4
	as Output (Active LOW)	T*/30
$\overline{NRFD}$	Not Ready for Data Input	T*/1.4
	as Output (Active LOW)	T*/30
ASEL	Address Select Output	10/5.0 (2.5)
$\overline{CLR}$	Device Clear Output (Active LOW)	10/5.0 (2.5)
$\overline{D}$/S/E	Data/Status Output (Active LOW)	10/5.0 (2.5)
	End-of-String Output (Active HIGH)	10/5.0 (2.5)
$\overline{DRB}$	Bus Drive Enable Output (Active LOW)	10/5.0 (2.5)
$\overline{LAD}$	Listen Address Status Output (Active LOW)	10/5.0 (2.5)
$\overline{RQS}$	Requested Service Status Output (Active LOW)	130/30
RXST	Receiver Strobe Output (Active HIGH)	10/5.0 (2.5)
$\overline{R}$/L	Remote/Local Output	10/5.0 (2.5)
$\overline{SRQ}$	Service Request Output (Active LOW)	T*/30
STST	Status Strobe Output (Active HIGH	10/5.0 (2.5)
$\overline{TAD}$	Talk Address Status Output (Active LOW)	10/5.0 (2.5)
TXST	Transmitter Strobe Output (Active HIGH)	10/5.0 (2.5)
$\overline{TRIG}$	Device Trigger Output (Active LOW)	10/5.0 (2.5)
XTAL	Crystal Output	10/5.0 (2.5)

*T = Resistive Termination per IEEE-488 Standard.

**Unit Load (U.L.) definitions
LOW State: 1.6 mA = 1 U.L.
HIGH State: 40 μA = 1 U.L.

Where two sets of output LOW loading factors are shown, the one in parentheses applies over the Military V_{CC} and temperature ranges.

GPIB Protocol

A full description and specification of the GPIB system is published in the IEEE document 'IEEE Standard Interface for Programmable Instrumentation" IEEE Std 488–1978 and is used as the reference in this description.

The standard is a 16-wire interface that can transmit byte serial data at rates of up to 1 megabyte/second. Using this standard up to 15 individual devices (instruments or system components) may be interconnected in a star or linear network, with a maximum cable length of 20 m and automatically controlled or programmed. Data may be exchanged between instruments or between a controller and instruments. The use of a bus extender or a controller which can handle a number of separate instrument buses allows more than 15 instruments to be used in a system:

Talkers—These instruments can only transmit data (when addressed), e.g., a timer or counter.

Listeners—These instruments can only receive information, e.g., a programmable power supply or a printer.

Talker/Listeners—These instruments can receive data or functional instructions and later transmit data, e.g., a programmable DVM or multichannel a/d converter.

Controller—A device that is able to generate control instructions for instruments on the bus, e.g., a mini-computer or programmable calculator.

The 96LS488 is designed for use in any of the first three types of devices.

The 16-wire bus is organized as 3 functional groups *(Figure 1).* The 8-line Data bus, $\overline{DIO}_1$-$\overline{DIO}_7$, $\overline{DO}_8$, is used to transfer commands in bit parallel/byte serial form from Talkers to Listeners. The 3-line Data Byte Transfer Control bus, $\overline{NRFD}$, $\overline{NDAC}$ and $\overline{DAV}$, implements a data handshake which ensures that information transfer proceeds as fast as the device will allow but no faster than the slowest device currently addressed as active *(Figure 2).* The 5-line General Interface Management bus $\overline{ATN}$, $\overline{REN}$, $\overline{EOI}$, $\overline{IFC}$ and $\overline{SRQ}$ is principally used by the Controller.

In its simplest configuration the GPIB can consist of only two instruments, a Talker and a Listener; in its most complex configuration up to 961 instruments could be controlled by one or more mini-computers. A bus controller dictates the role of each device by making the $\overline{ATN}$ line LOW and sending Talk and/or Listen Addresses on the bus data lines. Those devices with matching addresses are activated accordingly. Device addresses are set by switches or PC board jumpers. In single address mode each device has a 5-bit address allowing up to 31 different addresses (one code is used as an unaddress command). In extended address mode each device has a 10-bit address, allowing up to 961 different addresses and the Controller must send two bytes in order to activate a device.

In the configuration shown in *Figure 1* a sequence to initiate a data transfer from device A to device D would proceed as follows:

Controller sends $\overline{ATN}$ (attention) command; whenever the $\overline{ATN}$ line goes LOW, devices using the Data bus immediately stop all operations. The Controller keeps the $\overline{ATN}$ line LOW during the remainder of this sequence.

Controller sends UNL (unlisten) command; this instruction disables any devices that are in Listen mode.

Controller sends Talk Address command to device A; this instruction puts device A into Talk mode and disables any other devices that had been in that mode.

Controller sends Listen Address command to device D, putting it in Listen mode.

When the Controller stops sending $\overline{ATN}$, the bus will be released for data transfer functions and device A will begin transmitting to device D.

The $\overline{SRQ}$ line allows any device to interrupt the Controller and request service. The Controller can identify the interrupting devices by conducting a Serial Poll. To do this it issues an Unlisten (UNL) command followed by a Serial Poll Enable (SPE) command and then the Talk Address of each device in turn. The interrupting device will optionally drive $\overline{DIO}_7$ LOW. Alternatively the Controller can conduct a Parallel Poll by making both $\overline{EOI}$ and $\overline{ATN}$ LOW. Devices will then return one bit of status on a $\overline{DIO}$ line previously assigned via a Parallel Poll Enable (PPE) command.

The $\overline{REN}$ line allows the Controller to put all Listen addressed instruments into remote control mode, while the $\overline{IFC}$ line allows the Controller to initialize the system.

Fig. 1 Interface Capabilities and Bus Structure

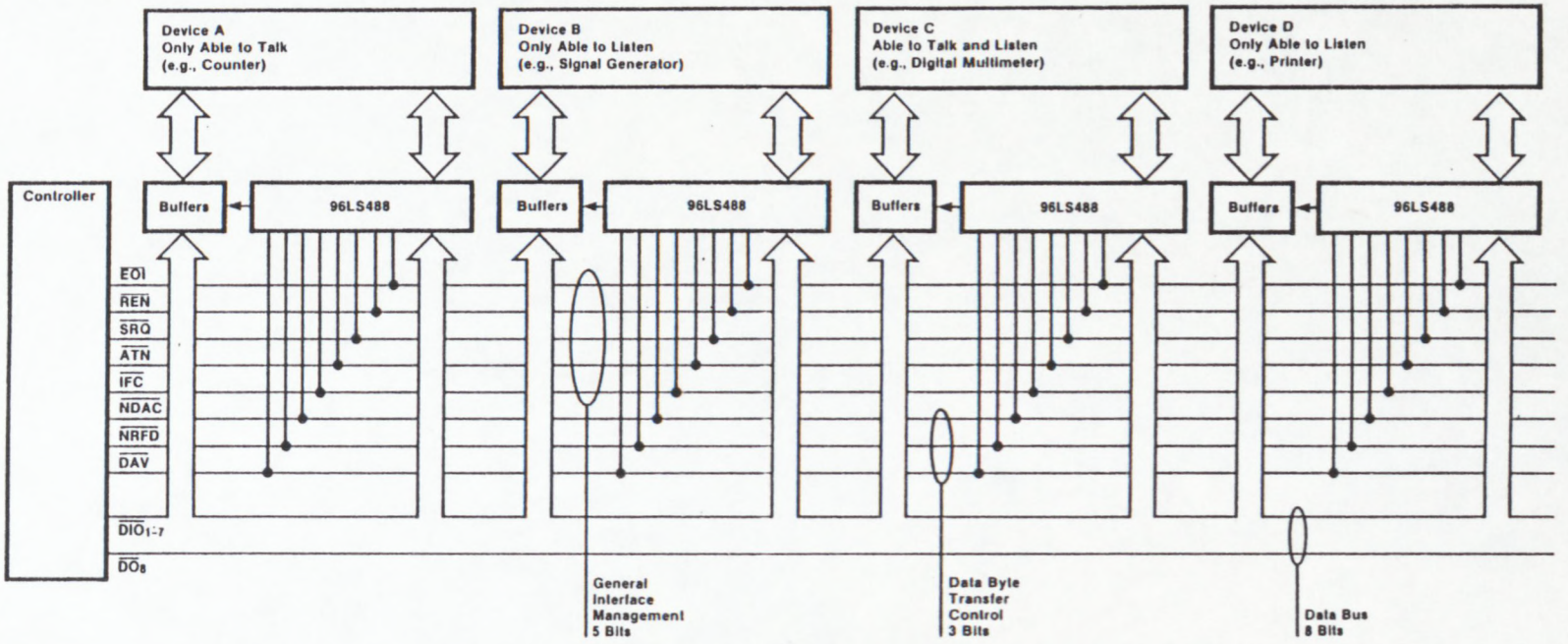
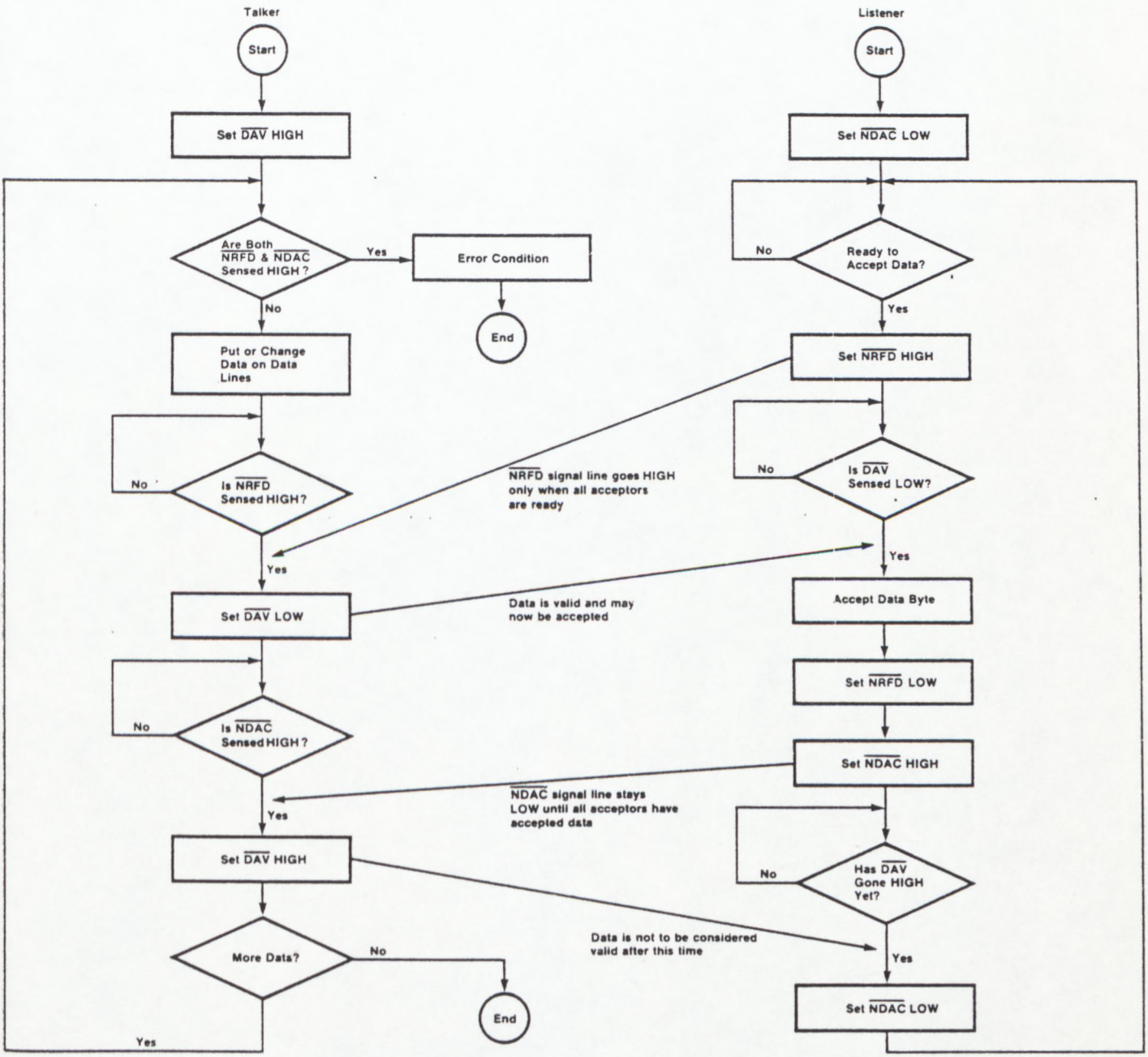
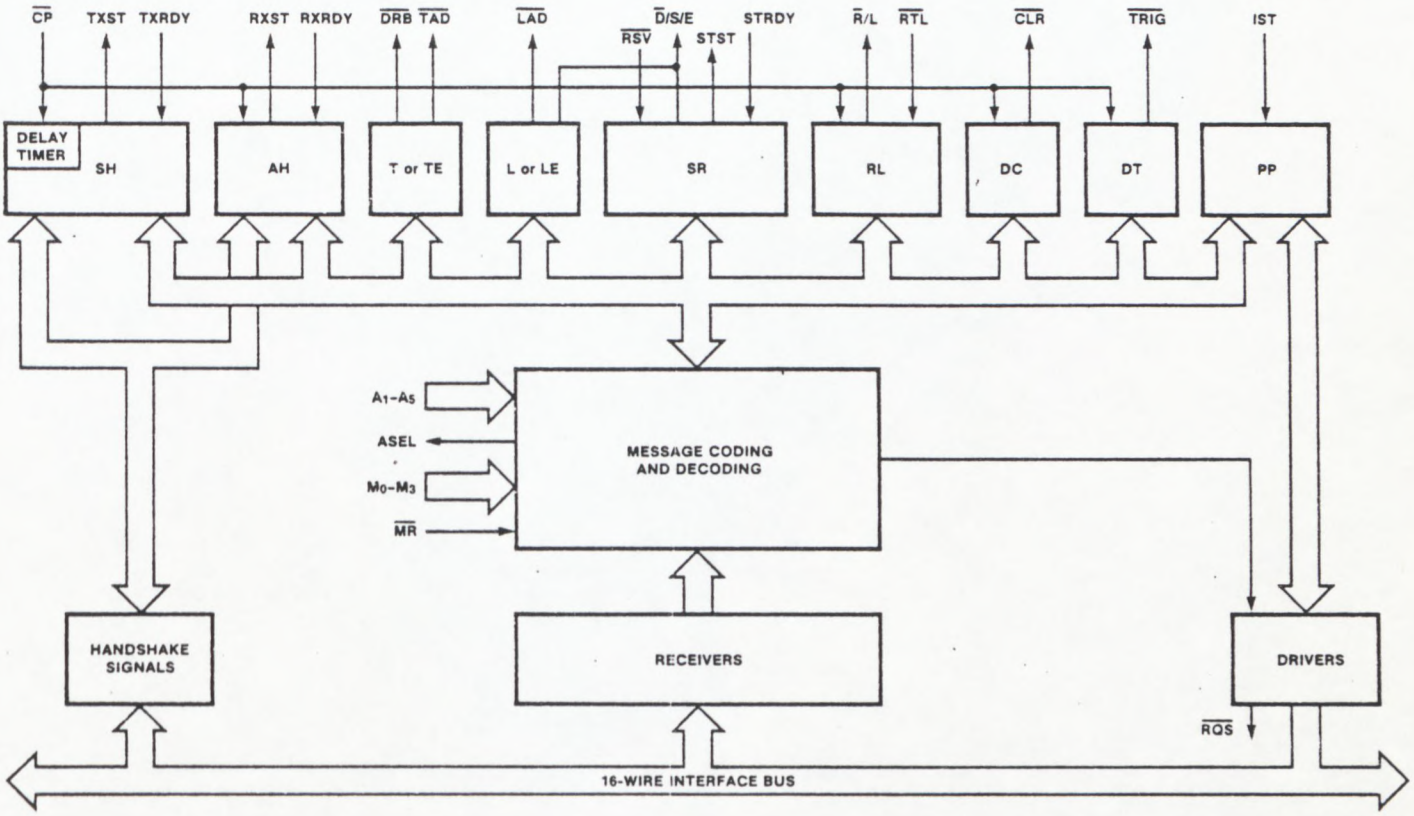


Fig. 2 Source and Acceptor Handshake Logic



96LS488 Functional Block Diagram



Functions Implemented by the 96LS488

Acceptor Handshake (AH)
Controls the acquisition of addresses, interface commands and data bytes from the bus. The AH passes data bytes to the instrument logic via a 2-wire handshake (RXST, RXRDY) and a status output ($\overline{LAD}$).

Source Handshake (SH)
Controls the passing of data bytes and status information from the instrument to the bus. The SH communicates with the instrument via a 2-wire handshake (TXST, TXRDY) and a status output ($\overline{TAD}$). It has two operating speeds, selectable via the M_0-M_3 inputs. Low speed is used with open collector data drivers and gives a settling delay of 2.0 μs. High speed is used with 3-state bus drivers and gives a settling delay of 1.1 μs for the first byte sent and 0.5 μs for the subsequent bytes (clock frequency 10 MHz).

Listener, Extended Listener (L or LE)
Either the single or extended address L function can be selected by the M_0-M_3 inputs. The 96LS488 flags the listener addressed status via the $\overline{LAD}$ output. The listen address is defined by the A_1-A_5 inputs and, where the extended address feature is used, the ASEL output controls an external multiplexer to select the primary or secondary address as required.

Talker, Extended Talker (T OR TE)
Either the single or extended address T function can be selected. The T function employes the M_0-M_3 and A_1-A_5 inputs in the same manner as the L function. The 96LS488 flags the talker addressed status via the $\overline{TAD}$ output. The T function also incorporates the Serial Poll Function.

Talker/Listener, Extended Talker/Listener
For instruments with both talk and listen capabilities, the 96LS488 implements both the T and L functions. In this mode, the "Untalk if My Listen Address" and "Unlisten if My Talk Address" feature is standard. Where the single address mode is selected, different Talk and Listen addresses can be used. In the extended address mode, the Talk and Listen addresses must be identical.

Device Trigger, Device Clear (DT, DC)
These functions generate a pulse on the Trigger or Clear output upon receipt of the relevant bus command.

Remote Local (RL)
The complete RL function is implemented, including the Local Lock-Out feature.

Talk Only, Listen Only
The 96LS488 can operate in either of these two modes via the M_0-M_3 input code selection.

Service Request (SR)
The 96LS488 initiates an SR on receipt of the $\overline{RSV}$ input and returns its status on the $\overline{RQS}$ line when serial polled. The $\overline{RQS}$ output can be used to drive the $\overline{DIO_7}$ bus line directly.

Parallel Poll (PP)
The Controller assigns, via the PPE command, one of the eight data lines for use as the Parallel Poll Response output. When the Controller issues the IDY command, the 96LS488 compares the state of the IST input with the state defined by the last PPE command. If the comparison is True, the previously assigned $\overline{DIO}$ line is driven LOW by the 96LS488.

Pin Functions

488 Bus Signals
All bus inputs have Schmitt trigger buffers, and all bus outputs can sink 48 mA. Each bus signal is terminated with a resistive load and meets the dc load characteristics specified in section 3.5.3 of the IEEE Std 488 – 1978 specification.

$\overline{DIO_1}$-$\overline{DIO_7}$ (Data Input/Output)—These are used as inputs to receive addresses and interface commands. They are used as outputs, along with $\overline{DO_8}$ to provide Parallel Poll response.

$\overline{ATN}$ (Attention)—This is an input from the GPIB Controller. When $\overline{ATN}$ is LOW the 96LS488 interprets the data on $\overline{DIO_1}$-$\overline{DIO_7}$ lines as commands or addresses. If the 96LS488 is interrupted by $\overline{ATN}$ while sending data, it will relinquish control of the Data and Management lines within 200 ns.

$\overline{DAV}$ (Data Valid)—A bidirectional signal with a 3-state output driver, $\overline{DAV}$ is part of the handshake system and is driven LOW by current talker when a valid data byte, command or address is on the GPIB. $\overline{DAV}$ is treated as an input when the 96LS488 is addressed to Listen, or is receiving $\overline{ATN}$. It is an output when the 96LS488 is addressed to Talk and $\overline{ATN}$ is HIGH.

$\overline{NRFD}$ (Not Ready for Data)—A bidirectional signal with an open-collector output driver, $\overline{NRFD}$ is part of the handshake system and is driven LOW to indicate that an instrument is not ready to receive data. The 96LS488 drives $\overline{NRFD}$ HIGH when addressed as a Listener and the instrument is ready to accept a data byte or when $\overline{ATN}$ is LOW and the 96LS488 is ready to accept an address or interface command. $\overline{NRFD}$ is treated as an input when the 96LS488 is addressed to Talk.

$\overline{NDAC}$ (Not Data Accepted)—A bidirectional signal with an open-collector output driver, $\overline{NDAC}$ is part of the handshake system and is pulled LOW to indicate that a device has not yet accepted a data byte. $\overline{NDAC}$ is treated as an input when the 96LS488 is addressed to Talk. It is an output and is driven LOW when the 96LS488 is addressed to Listen and the instrument has not accepted a data byte or when receiving $\overline{ATN}$ and the 96LS488 has not accepted an address or interface command.

$\overline{SRQ}$ (Service Request)—This is an open-collector output driven LOW when the instrument requests service (via $\overline{RSV}$) from the Controller.

$\overline{RQS}$ (Requested Service)—A 48 mA 3-state output, enabled during a Serial Poll response and driven LOW if the 96LS488 initiated an $\overline{SRQ}$, $\overline{RQS}$ can be directly connected to $\overline{DIO_7}$ in applications where it is the only status information to be sent.

$\overline{EOI}$ (End or Identify)—This is an input from the GPIB Controller used to elicit a Parallel Poll response, or from the current active talker to indicate the End-of-string (END) message.

$\overline{REN}$ (Remote Enable)—This is an input driven by the Controller. The Controller drives it LOW when it needs to remotely program an instrument.

$\overline{IFC}$ (Interface Clear)—This is an input from the Controller, driven LOW to clear the interface logic. (See *Figure 18.*)

Instrument Interface and Auxiliary Pins
Instrument Logic Signals—All instrument logic signals are standard low-power Schottky compatible.

$\overline{CP}$ (10 MHz Clock)—Used to clock internal-state flip-flops and is divided down internally to generate the SH data settling delay. All output changes are synchronous with the negative clock edge. The $\overline{CP}$ input can be driven by an external oscillator or used in conjunction with XTAL output, as an RC or a crystal oscillator. (See *Figure 6.*)

XTAL (Crystal)—Used to connect a crystal or external RC timing components for the on-chip oscillator. (See *Figure 6.*)

$\overline{MR}$ (Active-LOW Master Reset)—Initializes all internal latches and is completely asynchronous. After a reset, all outputs to the GPIB are passive HIGH and $\overline{RQS}$ is in the high-Z state; R/L, $\overline{TRIG}$, $\overline{CLR}$, $\overline{DRB}$, ASEL, $\overline{TAD}$ and $\overline{LAD}$ are HIGH; D/S/E, RXST, STST and TXST are LOW.

M_0-M_3 (Mode Control Inputs)—Define 1 of 14 possible operating modes for the 96LS488. M_0-M_3 are HIGH-true inputs.

A_1-A_5 (Device Address Inputs)—Define the instrument address and originate from switches, PC jumpers or software-loaded register. Where different Talk and Listen addresses are required or when the secondary address feature is used, these inputs must be externally multiplexed, using the ASEL output to control the multiplexer. A_1-A_5 are HIGH-true inputs (H = 1, L = 0) and thus have the opposite polarity of the $\overline{DIO_1}$-$\overline{DIO_5}$ addresses.

ASEL (Address Select Output)—Selects, via an external multiplexer, the Talk/Listen or primary/secondary address input, depending on the operating mode selected; LOW for Talk or primary address; HIGH for Listen or secondary address. (See *Figure 11.*)

$\overline{LAD}$, $\overline{TAD}$ (Address Status Outputs)—Indicate the Listen-Address or Talk-Address status respectively. They are also activated in the Talk-Only and Listen-Only modes. The outputs are active-LOW to facilitate driving LED indicator lamps. (See *Figures 7, 10.*)

RXST (Receiver Strobe Output)—Forms part of the handshake logic to pass data bytes to the instrument. When addressed to Listen, the 96LS488 takes RXST HIGH when a valid data byte is on the bus and holds it HIGH until the instrument signals (via the RXRDY input) that it has processed the byte. RXST may be inverted and connected to RXRDY, in which case the 96LS488 will receive data bytes from the bus at a data rate determined solely by the bus handshake. (See *Figure 8.*)

TXST (Transmit Strobe Output)—Forms part of the handshake logic to pass data bytes from the instrument to the bus. When addressed to Talk, the 96LS488 takes TXST HIGH to signal the instrument that the bus has accepted the data. TXST does not go LOW again until the instrument has acknowledged that the byte has been accepted (via the TXRDY INPUT). TXST may be inverted and connected to TXRDY, in which case the 96LS488 will transmit data bytes to the bus at a data rate determined solely by the bus handshake. (See *Figure 9.*)

STST (Status Strobe Output)—Forms part of the handshake logic to pass a status byte from the instrument to the bus during a Serial Poll sequence. It operates in conjunction with the STRDY input in the same way as the TXST and TXRDY signals. STST may be inverted and connected to STRDY, in which case the status byte will be repeated as long as the 96LS488 is addressed to Talk. (See *Figure 16.*)

RXRDY (Receiver Ready Input)—Forms part of the handshake logic controlling the passing of data from the bus to the instrument. RXRDY is driven HIGH when the instrument is ready to receive a data byte and LOW to acknowledge receipt of a data byte. (See *Figure 8.*)

TXRDY (Transmitter Ready Input)—Forms part of the handshake logic controlling the passing of data from the instrument to the bus. When the 96LS488 is addressed to Talk, TXRDY is driven HIGH when the instrument has a data byte to send and LOW to acknowledge that the byte has been accepted by the bus. (See *Figure 9.*)

STRDY (Status Ready Input)—Forms part of the handshake logic controlling the passing of a status byte to the bus during a Serial Poll sequence. It operates in a similar fashion to TXRDY. (See *Figure 16.*)

$\overline{RSV}$ (Request Service Input)—Is pulled LOW by the instrument to request service and initiate an $\overline{SRQ}$ interrupt to the controller. This interrupt will be cleared if $\overline{RSV}$ goes HIGH before it is serviced, but once the $\overline{SRQ}$ is serviced, $\overline{RSV}$ must, after exiting SPAS, go HIGH then LOW to initiate another service request. (See *Figure 16.*)

$\overline{CLR}$ (Clear Output)—Issues a negative pulse when the 96LS488 receives a Device Clear (DC) command, or when it is addressed to Listen and receives a Selected Device Clear. The $\overline{CLR}$ Output will stay LOW during Accept Data State (ACDS) or until $\overline{ATN}$ goes HIGH. (See *Figure 13.*)

$\overline{TRIG}$ (Trigger Output)—Issues a negative pulse when the 96LS488 is addressed to Listen and receives a DT command. The $\overline{TRIG}$ output will stay LOW during ACDS or until $\overline{ATN}$ goes HIGH. (See *Figure 13.*)

$\overline{DRB}$ (Drive Bus Output)—Taken LOW to enable an external data bus driver when the 96LS488 is addressed to Talk and is in the Talker Active State. $\overline{DRB}$ will go LOW one clock period after $\overline{ATN}$ goes HIGH, and will go HIGH asynchronously within 200 ns (typically 70 ns) after $\overline{ATN}$ goes LOW. (See *Figures 9, 10, 11.*) $\overline{DRB}$ can also be used to tell the instrument logic to fetch the first byte.

$\overline{RQS}$ (Requested Service Output)—A 48 mA 3-state output, enabled during a Serial Poll response and driven LOW if the 96LS488 initiated an $\overline{SRQ}$. $\overline{RQS}$ can be directly connected to $\overline{DIO_7}$ in applications where it is the only status information to be sent. (See *Figures 4, 5, 16.*)

$\overline{D}$/S/E (Data/Status or END Output)—Valid during the Talk Addressed state and indicates to the instrument logic whether the information to be sent via the bus is to be data or status (LOW for data, HIGH for status). A status byte is sent only in response to a Serial Poll, and, in this case, $\overline{D}$/S/E may be used to control a multiplexer to select data or status as the source to the bus data drivers. (See *Figures 4, 5, 16.*) Valid during the Listener Active State (LACS) to indicate that the current talker is sending the END message; a HIGH output indicates that the END message is true.

$\overline{R}$/L (Remote/Local Output)—Goes LOW when the Controller puts the instrument into Remote mode via the $\overline{REN}$ command. (See *Figures 14, 15.*)

$\overline{RTL}$ (Return to Local Input)—Taken LOW to request return of the instrument to local control. $\overline{RTL}$ will set $\overline{R}$/L HIGH unless the Controller has put the 96LS488 into Local Lock-out state. (See *Figure 14.*)

IST (Instrument Status Input)—Used by the Parallel Poll logic, IST is compared with the logic state defined by $\overline{DIO_4}$ during the last PPE command. If IST is in the defined state, the 96LS488 will make an affirmative response to the next IDY message by making the assigned $\overline{DIO}$ line LOW. Note that IST is a HIGH-true input while $\overline{DIO_4}$ is LOW-true. (See *Figure 17.*)

Operating Modes

The 96LS488 has 14 operating modes, defined by a 4-bit input code M_0-M_3 which would normally be selected by switches or PC board jumpers.

Table 1 defines the input codes and operating modes.

Table 1

Mode Inputs				Operating Mode	Function
M_0	M_1	M_2	M_3		
L	L	L	L	Off Line	The device cannot take part in any GPIB operations
L	L	L	H	TON (LOW Speed)[1]	The device goes directly to the talk addressed state and can source data to the bus
L	L	H	L	LON	The device goes directly to the listen addressed state and can receive data from the bus
L	L	H	H	TON (HIGH Speed)[1]	As for TON (LOW Speed)
L	H	L	L	T (LOW Speed)[1]	Talker Only, single address mode
L	H	L	H	TE (LOW Speed)[1]	Talker Only, extended address mode
L	H	H	L	T (HIGH Speed)[1]	Talker Only, single address mode
L	H	H	H	TE (HIGH Speed)[1]	Talker Only, extended address mode
H	L	L	L	L	Listener Only, single address mode
H	L	L	H	LE	Listener Only, extended address mode
H	H	L	L	T/L (LOW Speed)[1, 2]	Talker/Listener, dual address mode
H	H	L	H	TE/LE (LOW Speed)[1]	Talker/Listener, extended address mode
H	H	H	L	T/L (HIGH Speed)[1, 2]	Talker/Listener, dual address mode
H	H	H	H	TE/LE (HIGH Speed)[1]	Talker/Listener, extended address mode

Notes

1. The LOW speed talker option is selected where open-collector data drivers are used. The delay from putting valid data on the GPIB to $\overline{DAV}$ going true is 2.0 μs. The HIGH speed option is selected where 3-state drivers are used. The settling delay (data to $\overline{DAV}$) is 1.1 μs for the first byte sent after a LOW to HIGH transition of $\overline{ATN}$ and 500 ns for subsequent bytes.
2. For dual address Talker/Listener modes the Talk and Listen addresses can be different.

Table 2

$\overline{DO_8}$	$\overline{DIO_7}$	$\overline{DIO_6}$	$\overline{DIO_5}$	$\overline{DIO_4}$	$\overline{DIO_3}$	$\overline{DIO_2}$	$\overline{DIO_1}$	
X	H	L	$\overline{A}_5$	$\overline{A}_4$	$\overline{A}_3$	$\overline{A}_2$	$\overline{A}_1$	Primary Listen Address
X	H	L	L	L	L	L	L	Unlisten
X	L	H	$\overline{A}_5$	$\overline{A}_4$	$\overline{A}_3$	$\overline{A}_2$	$\overline{A}_1$	Primary Talk Address
X	L	H	L	L	L	L	L	Untalk
X	L	L	$\overline{S}_5$	$\overline{S}_4$	$\overline{S}_3$	$\overline{S}_2$	$\overline{S}_1$	Secondary Address

Addresssing Modes

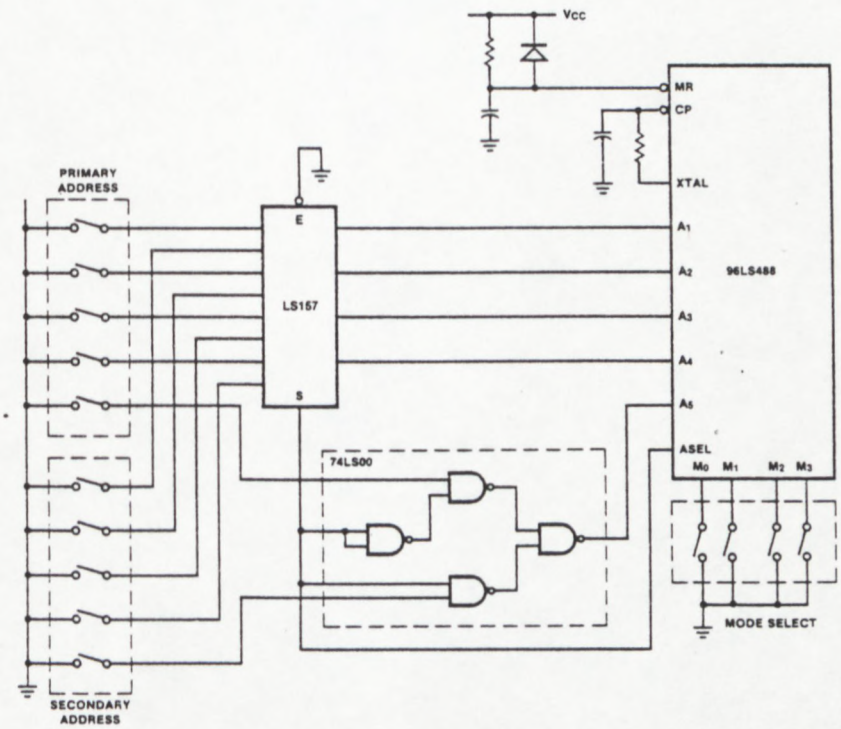
Where extended addressing or different Talk and Listen addresses are required, the address codes must be externally multiplexed, using ASEL. (See *Figure 3.*) In the extended address modes, ASEL is LOW for the primary address and HIGH for the secondary address. (See *Figure 11.*) In the dual address modes, ASEL is HIGH for the Listen address and LOW for the Talk address.

In single address mode the 96LS488 will go into the addressed state on receipt of the primary address. In extended address mode it will go to the addressed state on receipt of its secondary address if, and only if, it has received its primary address. If a device is addressed to Talk and receives its Listen address it will un-address as a Talker and go to Listener addressed state, and vice versa. A Talker Addressed device will un-address if it receives a non-matching talk address. The 96LS488 indicates its address status on the $\overline{TAD}$, $\overline{LAD}$ and $\overline{D}$/S/E outputs. (See *Table 3.*)

Table 3
Status Codes

$\overline{TAD}$	$\overline{LAD}$	$\overline{D}$/S/E	State
H	H	L	Off Line
H	L	L	Addressed to Listen (LADS)
L	H	L	Addressed to Talk (TADS)
L	H	H	Serial Poll Mode (SPM)
H	L	H	Receiving END Message (LACS)

Fig. 3 Address Multiplexer



Status Response

In Serial Poll Active State (SPAS) the instrument is requested to return a status byte, via the usual handshake, to the Controller. Seven bits are defined by the instrument. Bit 7 denotes the Request Service Status ($\overline{RQS}$) and is provided by the 96LS488 on the $\overline{RQS}$ output. If the instrument provides no status, other than $\overline{RQS}$, then the $\overline{RQS}$ output can drive the bus directly *(Figure 4)*. If the instrument provides status information this can be multiplexed to the bus using $\overline{D}$/S/E *(Figure 5)*. After the bus handshake, the instrument can send a second status byte by making STRDY LOW then HIGH again, which starts the handshake. This sequence can be repeated to send additional bytes, as long as $\overline{ATN}$ remains HIGH. *(Figure 16.)*

Fig. 4 Status Bit Driven Direct by RQS

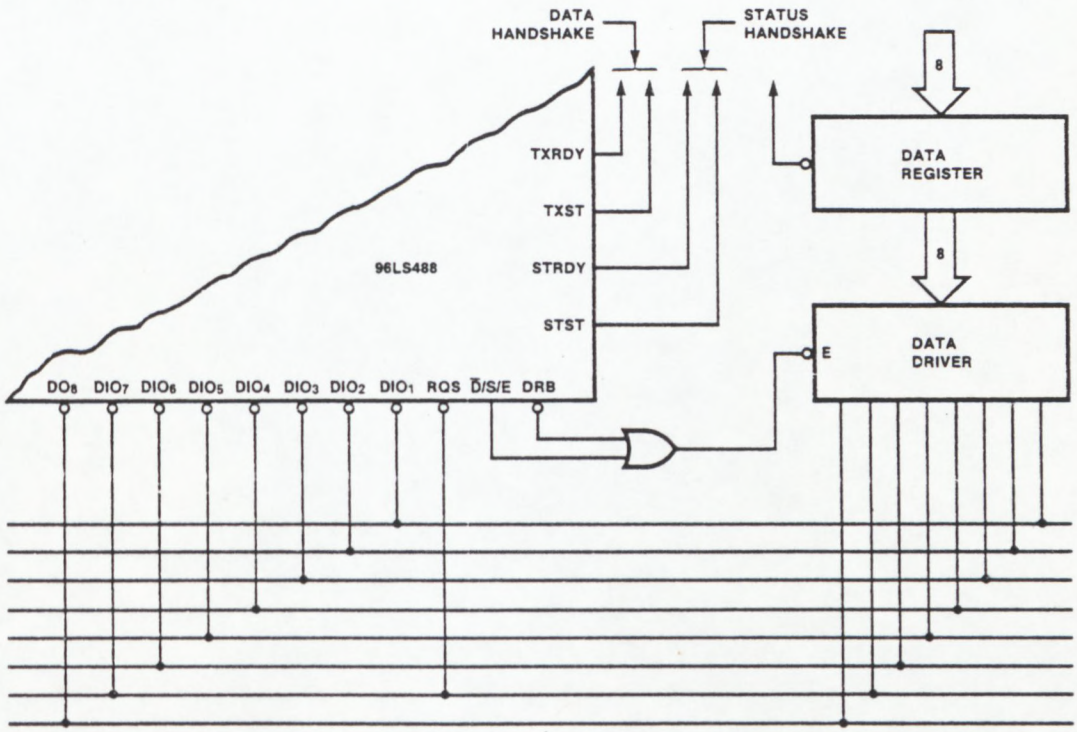
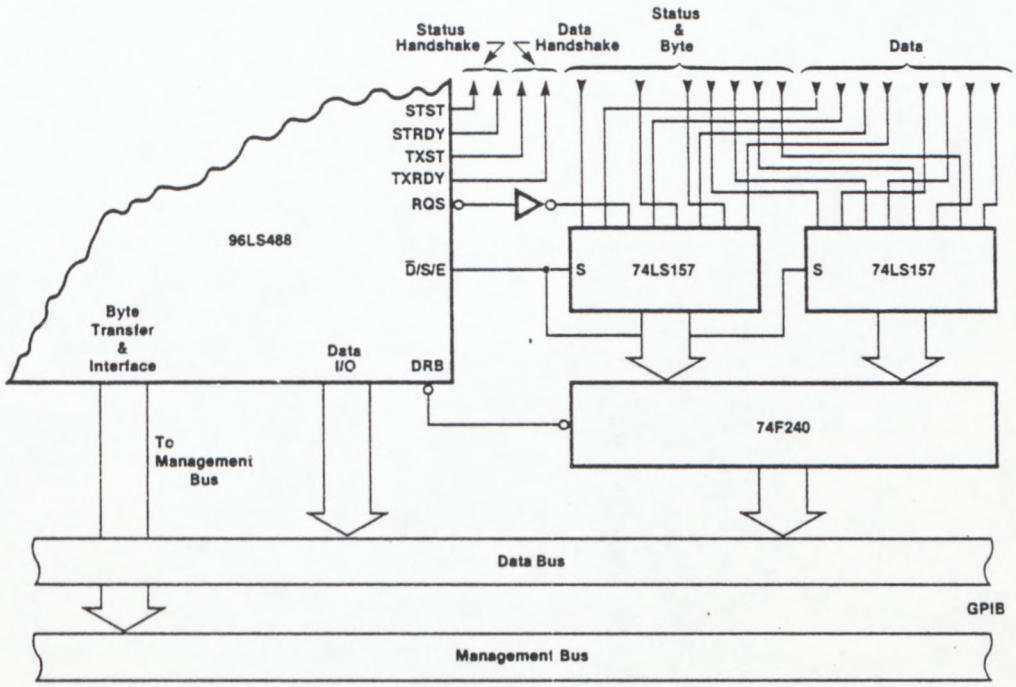


Fig. 5 Status Byte Multiplexing



Clock Input

The $\overline{CP}$ and XTAL inputs allow the 96LS488 either to accept external clock pulses or to generate its own clock. (See *Figure 6.*)

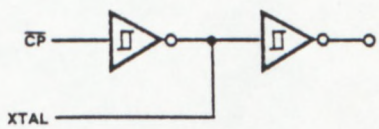
(a) An external clock can drive $\overline{CP}$. The XTAL pin may be used as an inverted buffered version of the external clock.

(b) The Schmitt-trigger buffer-inverter between the $\overline{CP}$ and XTAL pins can be used as a relaxation oscillator by connecting an RC network to provide feedback. The frequency of oscillation is variable up to 10 MHz. XTAL may be used to clock external circuits provided it is buffered.

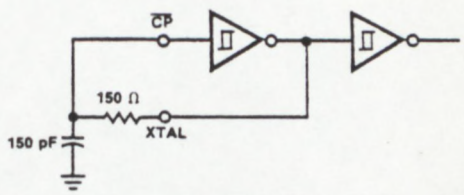
(c) The $\overline{CP}$ and XTAL pins will form a stable crystal oscillator by connecting a 10 MHz crystal and an RC network to provide positive feedback. XTAL may be used to clock external circuits provided it is buffered.

Fig. 6 Clock Timing Component Connections

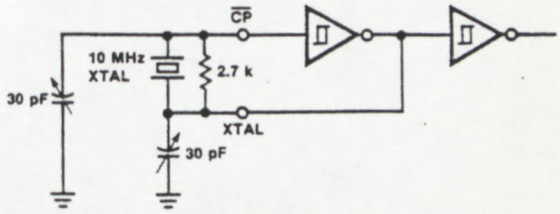
a



b



c



Timing Sequences

A 10 MHz clock frequency is recommended to give the correct Source Handshake delays. The 96LS488 can be clocked at a slower rate if the GPIB is not running at its maximum data rate of 1M byte. The lowest clock frequency allowable is dependent on the GPIB speed.

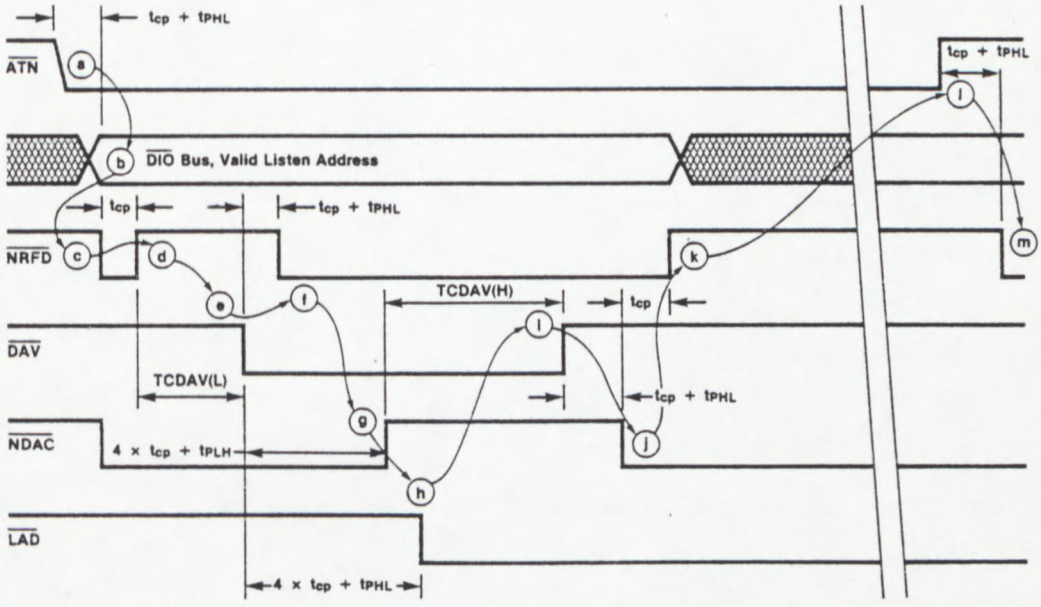
Regardless of clock frequency, the 96LS488 will respond to $\overline{ATN}$ within 200 ns by disabling $\overline{NRFD}$, $\overline{NDAC}$ and $\overline{DAV}$ while forcing $\overline{DRB}$ HIGH to disable the data drivers. In Parallel Poll sequences the relevant $\overline{DIO}$ line will be enabled or disabled within 200 ns of an IDY transition.

Since the internal logic of the 96LS488 is synchronous with the $\overline{CP}$ input, while in general all inputs are asynchronous, the precise timing of all responses to external signals is subject to a maximum uncertainty equal to one clock period. This is illustrated in the following timing diagrams, where some delays are defined as $t_{cp} + t_x$ (i.e., clock period + propagation delay).

Listen Address Sequence (See *Figure 7*)

a. Controller takes $\overline{ATN}$ line LOW followed by
b. putting the Listen address on the GPIB
c. Within ($t_{cp} + t_{PHL}$) the 96LS488 takes $\overline{NRFD}$ and $\overline{NDAC}$ LOW
d. and a t_{cp} later takes $\overline{NRFD}$ HIGH, indicating that the 96LS488 is ready for data
e. After a delay TCDAV(L) (determined by the Controller logic) the Controller takes $\overline{DAV}$ LOW
f. Within ($t_{cp} + t_{PHL}$) the 96LS488 takes $\overline{NRFD}$ LOW and
g. 3 clock periods later $\overline{NDAC}$ goes HIGH, indicating that the 96LS488 has accepted the data
h. followed by $\overline{LAD}$ indicating listen addressed status
i. After a Controller dependent delay, TCDAV(H), the $\overline{DAV}$ line goes HIGH and
j. within ($t_{cp} + t_{PHL}$) the 96LS488 takes $\overline{NDAC}$ LOW
k. A t_{cp} later the 96LS488 allows $\overline{NRFD}$ to go HIGH. $\overline{NRFD}$ stays HIGH until the Controller takes $\overline{ATN}$ HIGH, unless a further command is sent over the GPIB when a similar handshake sequence takes place
l. $\overline{ATN}$ goes HIGH followed by
m. $\overline{NRFD}$ going LOW within ($t_{cp} + t_{PHL}$). This occurs if the instrument is not ready to receive data (RXRDY LOW). If RXRDY is HIGH, $\overline{NRFD}$ will stay HIGH allowing a data transfer to take place

Fig. 7 Timing Diagram for Listen Address Sequence



Note: $\overline{ATN}$, $\overline{DIO}$ Bus & $\overline{DAV}$ driven by controller
$\overline{NRFD}$, $\overline{NDAC}$ Driven by 96LS488

Data Transfer from Bus to Listener (See *Figure 8*)
Assuming the instrument logic responds to RXST within one clock period. (See *Figure 8a.*)

a. The instrument signals it is ready to receive a byte by taking RXRDY HIGH (keeping RXRDY LOW constitutes an "$\overline{\text{NRFD}}$ hold").
b. Provided the 96LS488 is in the Listen Addressed State (LADS), $\overline{\text{NRFD}}$ is taken HIGH within (t_{cp} + t_{PHL})
c. When the current Talker sees the $\overline{\text{NRFD}}$ line HIGH it takes $\overline{\text{DAV}}$ LOW after a settling delay TTDAV(L).
d. The 96LS488 takes RXST HIGH within (t_{cp} + t_{PLH}) to inform the instrument that the GPIB data is valid and takes $\overline{\text{NRFD}}$ LOW
e. Assuming the instrument responds by pulsing RXRDY LOW within one clock period (t_{PRX} < t_{cp}) then RXST will remain HIGH only for one clock
f. At the same time as RXST returns LOW, $\overline{\text{NDAC}}$ is taken HIGH to inform the Talker that data has been accepted
g. After a delay TTDAV(H) determined by the Talker the $\overline{\text{DAV}}$ line goes HIGH
h. Within (t_{cp} + t_{PHL}) $\overline{\text{NDAC}}$ goes LOW
i. Followed by $\overline{\text{NRFD}}$ going HIGH one clock later

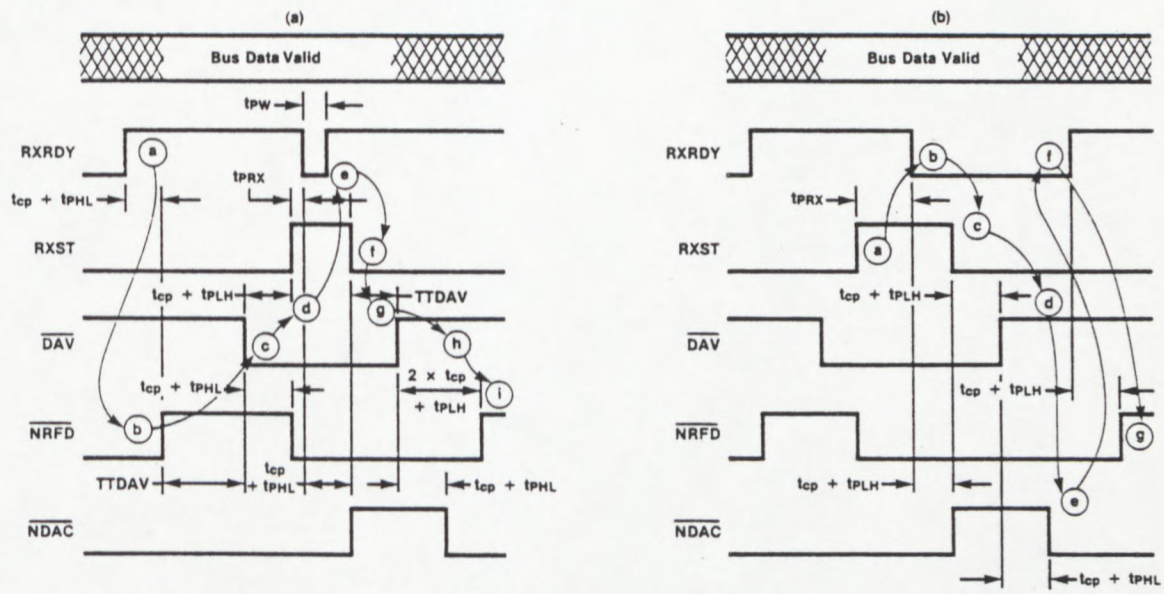
Note
In applications where the bus data can be latched by the RXST rising edge and handshaking is not necessary, RXRDY can be driven by RXST via an Inverter.

Assuming the instrument logic is slow and requires more than one clock period to process a data byte. (See *Figure 8b.*)

The timing sequence is identical to *Figure 8a* until RXST goes HIGH

a. After RXST goes HIGH the instrument delays t_{PRX} (>1 clock cycle) before taking RXRDY LOW
b. RXST will remain HIGH and $\overline{\text{NDAC}}$ will remain LOW during this period, causing the bus data to be maintained valid
c. RXST goes LOW, and $\overline{\text{NDAC}}$ goes HIGH within (t_{cp} + t_{PHL}) of RXRDY
d. After a delay TTDAV(H) the Talker takes $\overline{\text{DAV}}$ HIGH
e. Within (t_{cp} + t_{PHL}) $\overline{\text{NDAC}}$ goes LOW
f. Assuming the instrument is not ready (i.e., is holding RXRDY LOW) the 96LS488 holds $\overline{\text{NRFD}}$ LOW preventing another data transfer from starting
g. Within (t_{cp} + t_{PLH}) of RXRDY going HIGH, $\overline{\text{NRFD}}$ goes HIGH and the next data transfer cycle can start

Fig. 8 Timing Diagram, Data Transfer from Bus to Listener



Data Transfer from Talker to GPIB (See *Figure 9*)

a. $\overline{ATN}$ goes HIGH after completion of a Talk address sequence; $\overline{TAD}$ (not shown) is already LOW.

b. Within (t_{cp} + t_{PHL}) $\overline{DRB}$ goes LOW to enable the bus drivers

c. At a time determined by the instrument logic, TXRDY goes HIGH

d. If $\overline{NRFD}$ is already HIGH, the 96LS488 drives $\overline{DAV}$ LOW after delay T1 (11 × t_{cp} + t_{PHL} in high or 20 × t_{cp} + t_{PHL} in low speed). If $\overline{NRFD}$ is LOW, $\overline{DAV}$ will stay HIGH until $\overline{NRFD}$ goes HIGH (assuming T1 has expired). The $\overline{DAV}$ LOW period corresponds to the Source Transfer State (STRS).

e. The Listener(s) respond by eventually taking $\overline{NDAC}$ HIGH

f. Within (t_{cp} + t_{PLH}) 96LS488 takes $\overline{DAV}$ HIGH and TXST HIGH to inform the instrument that the data has been accepted and a new byte can be presented.

g. The instrument takes TXRDY LOW and holds it there until it has provided a new byte, h or h'. The minimum time TXRDY must be LOW is t_{PWL}

h. If TXRDY pulses LOW within t_{cp} of TXST, TXST(H) will be a minimum of t_{cp} wide. Otherwise TXST goes LOW within (t_{cp} + t_{PHL}) of TXRDY,j. After the first byte is sent, T1 drops from 11 × t_{cp} to 5 × t_{cp} in high speed mode.

i. If TXRDY is HIGH before $\overline{NRFD}$, $\overline{DAV}$ goes LOW within T2 of $\overline{NRFD}$ going HIGH (5 × t_{cp} in high speed or 20 × t_{cp} in low speed)

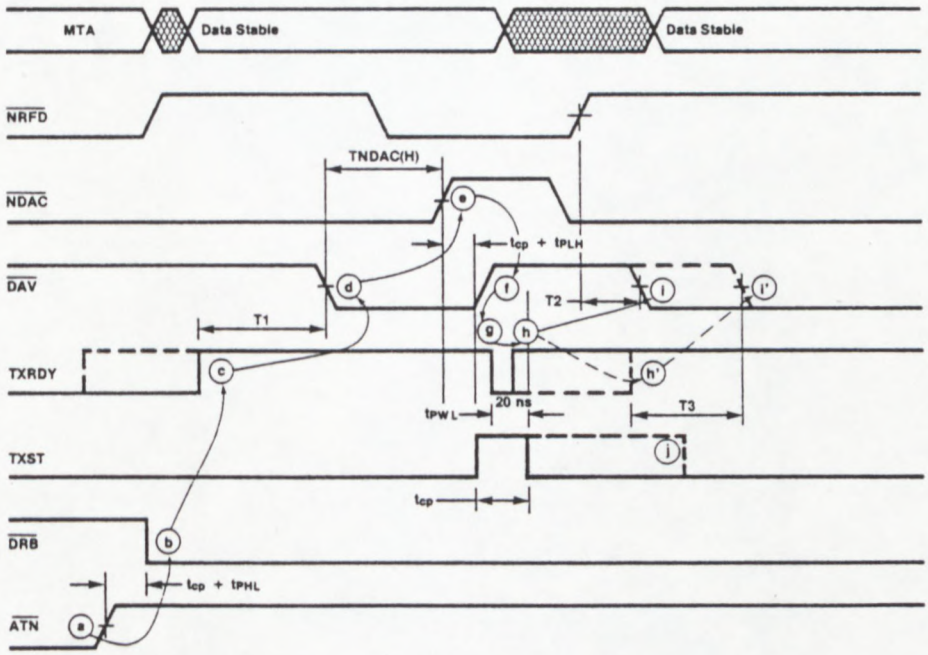
If TXRDY does not go HIGH until h' (after $\overline{NRFD}$ goes HIGH) $\overline{DAV}$ goes LOW T3 later,i'. (T3 = 6 × t_{cp} in HIGH and 21 × t_{cp} in low speed).

If the Talk sequence is interrupted by $\overline{ATN}$, while the instrument is generating a new byte, (between f and h), $\overline{DRB}$ will go HIGH within 200 ns and the $\overline{DAV}$ line will be relinquished. $\overline{DRB}$ will return LOW and the sequence will continue within (t_{cp} + t_{PLH}) of $\overline{ATN}$ going HIGH. If the 96LS488 is in high speed mode the first data byte sent will have a delay T1 = 11 × t_{cp}.

Note

In order to get the source handshake delays specified in IEEE Stad 488 - 1978, t_{cp} must be 100 ns (f_{clock} = 10 MHz).

Fig. 9 Timing Diagram Data Transfer from Talker to Bus



Notes

$\overline{NRFD}$, $\overline{NDAC}$ are driven by the current listener(s) $\overline{DAV}$ is driven by 96LS488.

T1 = 11 × t_{cp} + t_{PHL} in high speed, 20 × t_{cp} + t_{PHL} in low speed

T2 = 5 × t_{cp} + t_{PHL} in high speed, 20 × t_{cp} + t_{PHL} in low speed

T3 = 6 × t_{cp} + t_{PHL} in high speed, 21 × t_{cp} + t_{PHL} in low speed

Talk Address Sequence (See *Figure 10*)
This is similar to the Listen address sequence.

Fig. 10 Talk Address Sequence

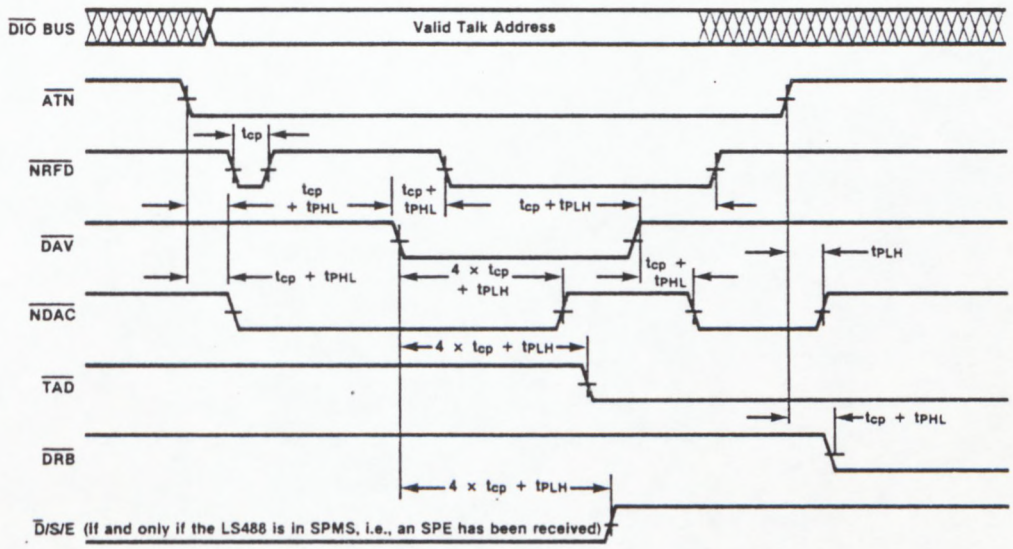


Fig. 11 Timing Diagram for Secondary Address Sequence

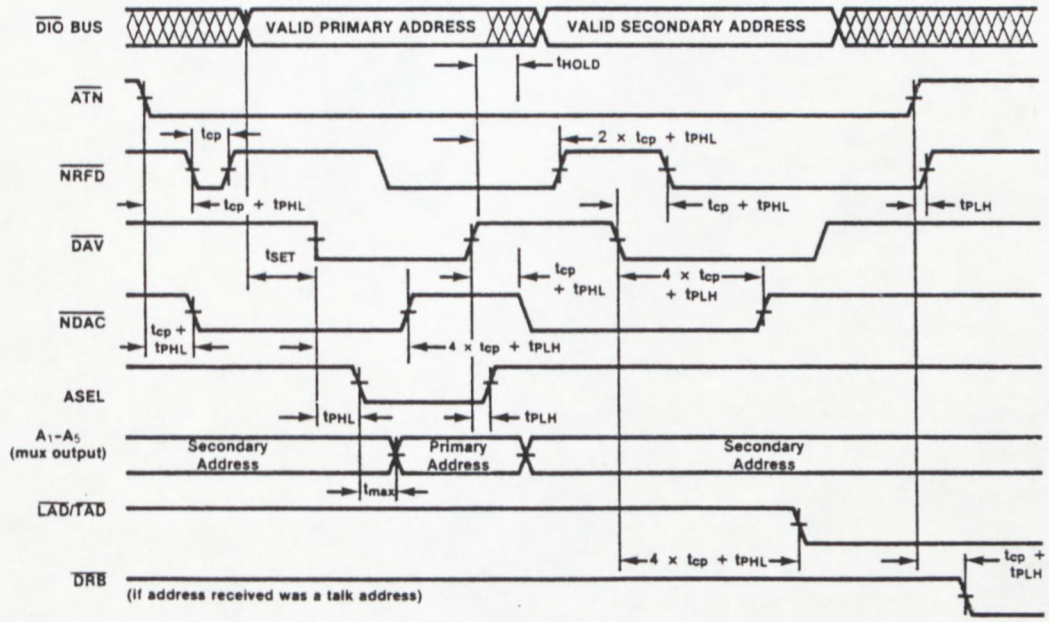
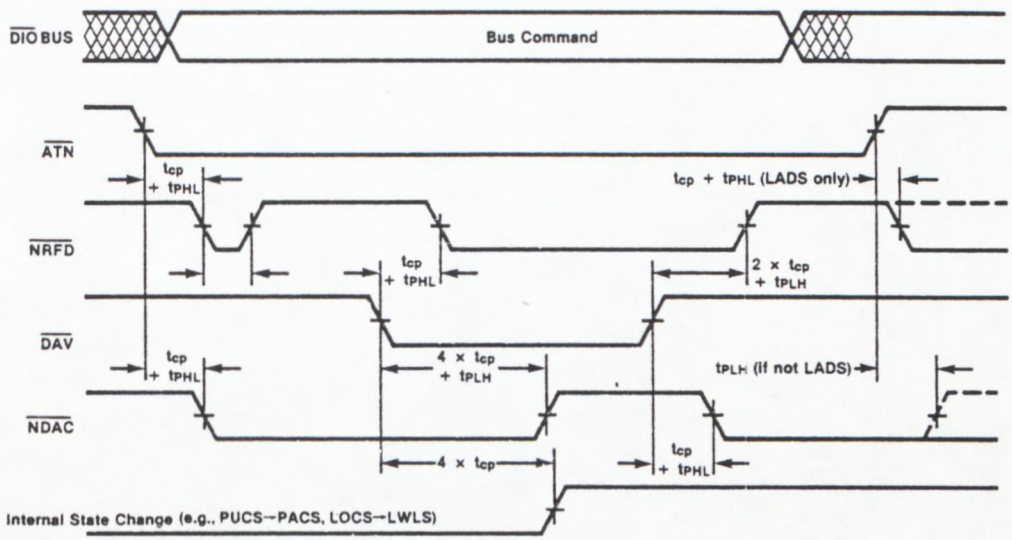


Fig. 12 Timing Diagram for 96LS488 Receiving Bus Commands



Note
Internal state changes do not necessarily change any 96LS488 outputs.

Fig. 13 Timing Diagram for Device Clear and Device Trigger Commands

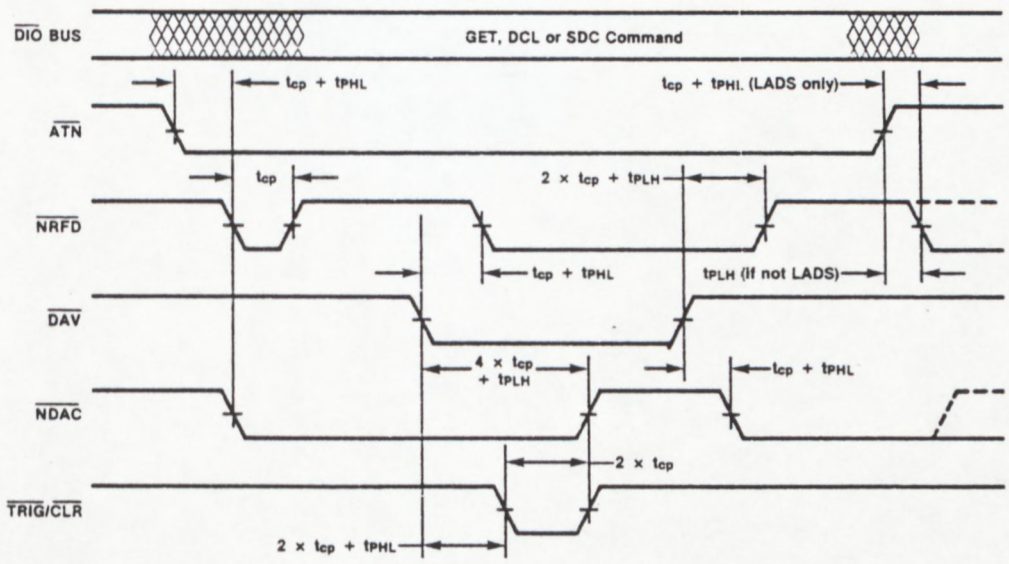
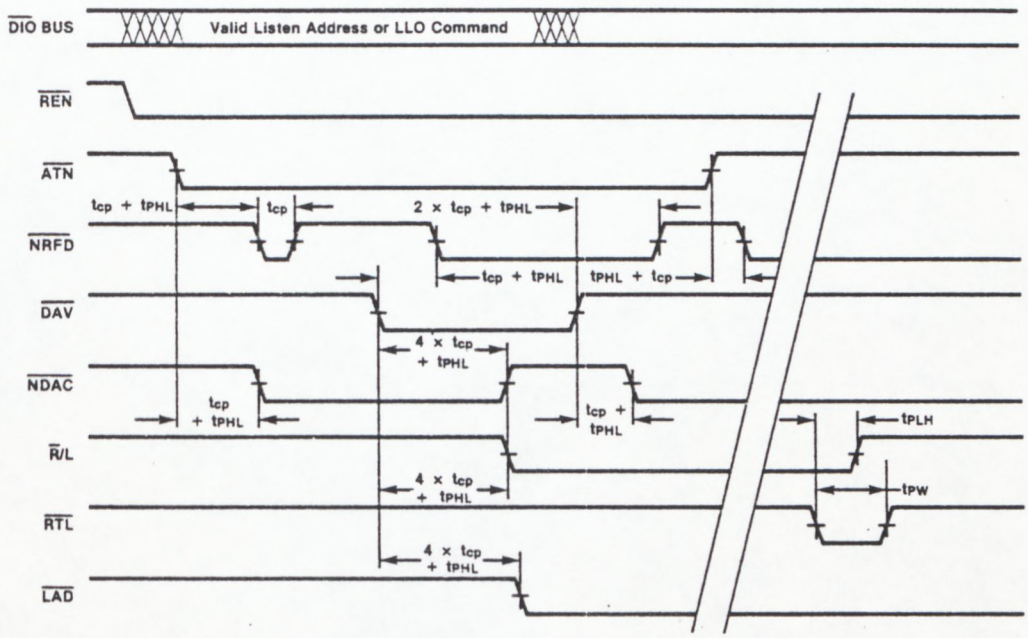
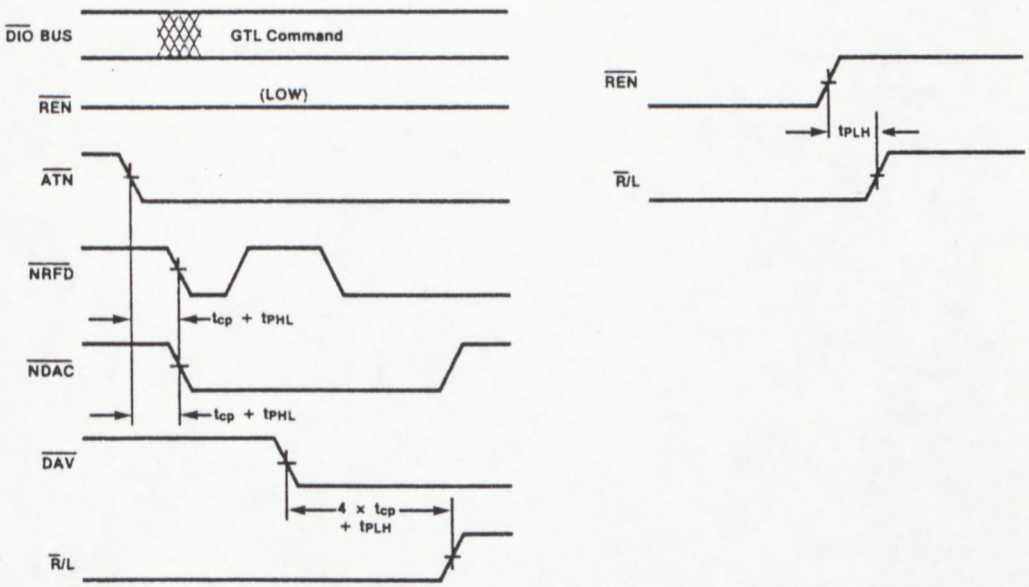


Fig. 14 Timing Diagram for Remote/Local Logic
(Starting in LOCAL State)



Note
If LLO has been sent, $\overline{RTL}$ will not cause $\overline{R/L}$ to change.

Fig. 15 Timing Diagram for Remote/Local Logic
(Starting in REMOTE State)



Serial Poll Sequence (See *Figure 16*)

a. The Controller has sent the Unlisten (UNL) and Serial Poll Enable (SPE) commands, generally in response to a LOW signal on $\overline{SRQ}$, and places a Talk address on the GPIB. The 96LS488 is sending the Service Requst ($\overline{SRQ}$) message, which was caused by the instrument logic making $\overline{RSV}$ LOW.

b. The 96LS488 allows $\overline{NRFD}$ to float passive HIGH to initiate normal handshake routine.

c. The Controller forces $\overline{DAV}$ LOW. Since $\overline{ATN}$ is also LOW, the 96LS488 receives the GPIB information as the message My Talk Address (MTA)

d. $\overline{NRFD}$ is active, acknowledging that the device is receiving a data byte

e. The 96LS488 enters the Talker Addressed State at time ($4 \times t_{cp} + t_{PHL}$) after $\overline{DAV}$ was forced LOW

f. $\overline{D}$/S/E goes HIGH at time ($4 \times t_{cp} + t_{PLH}$) after $\overline{DAV}$ was forced LOW. This indicates that the 96LS488 is in the Serial Poll Mode.

g. $\overline{NDAC}$ is allowed to float passive HIGH, indicating that the command data byte has been received

h. The Controller takes $\overline{DAV}$ HIGH

i. $\overline{NDAC}$ is pulled LOW, showing that the 96LS488 is ready for a new handshake cycle

j. The Controller allows the bus to float

k. The Controller releases $\overline{ATN}$. Because the 96LS488 is in the Serial Poll Mode, it now enters the Serial Poll Active State (SPAS), which prevails until the Controller makes $\overline{ATN}$ LOW again.

l. $\overline{DRB}$ goes active LOW, allowing the instrument to place its status byte on the bus. $\overline{DRB}$ goes LOW at time ($t_{cp} + t_{PHL}$) after $\overline{ATN}$ goes HIGH

m. When the 96LS488 enters SPAS, $\overline{SRQ}$ goes HIGH at time t_{PLH} after $\overline{ATN}$ goes HIGH

n. The instrument indicates that a status byte is ready by taking STRDY HIGH. This may occur earlier than shown, without affecting any of the foregoing

o. $\overline{DRB}$ enables the 3-state output $\overline{RQS}$ when in SPAS. $\overline{RQS}$ goes LOW at time ($t_{cp} + t_{PZL}$) after $\overline{ATN}$ is released

p. The Controller has taken $\overline{NRFD}$ HIGH, acknowledging that it is ready for the status byte

q. The 96LS488 takes $\overline{DAV}$ LOW after the time interval T_1, which starts either at the rising edge of STRDY or the falling edge of $\overline{DRB}$, whichever occurs later. The $\overline{DAV}$ LOW period corresponds to the Source Transfer State (STRS).

r. The Controller takes $\overline{NRFD}$ LOW

s. The instrument may release $\overline{RSV}$ at any time after the 96LS488 enters SPAS

t. The Controller takes $\overline{NDAC}$ HIGH, acknowledging that it has received the status byte

u. The 96LS488 releases $\overline{DAV}$ at time ($t_{cp} + t_{PLH}$) after $\overline{NDAC}$ goes HIGH

v. STST goes HIGH at time ($t_{cp} + t_{PLH}$) after $\overline{NDAC}$ goes HIGH. This tells the instrument that the status byte has been accepted. The instrument takes STRDY LOW to indicate that the data is no longer valid and to allow STST to go LOW again one t_{cp} after STRDY goes LOW

w. The Controller takes $\overline{NDAC}$ LOW once more after $\overline{DAV}$ goes HIGH

x. The data on the bus is no longer valid. If the $\overline{ATN}$ line remains HIGH, the instrument can provide another status byte by making STRDY HIGH to indicate valid data and to start the handshake

y. At the completion of the Serial Poll of this instrument, the Controller assumes control of the bus by forcing $\overline{ATN}$ LOW

z. $\overline{DRB}$ goes HIGH at time t_{PLH} after $\overline{ATN}$ goes LOW. This places the bus drivers of this instrument in the high-impedance (3-state output) or off (open-collector outputs) state

aa. Since $\overline{DRB}$ is no longer valid, the $\overline{RQS}$ output reverts to its high-impedance state

bb. $\overline{NRFD}$ goes HIGH, indicating that devices are ready to receive a command from the bus

cc. The Controller forces $\overline{DAV}$ LOW to show that it has placed a control byte on the bus

dd. $\overline{NRFD}$ goes LOW to acknowledge $\overline{DAV}$

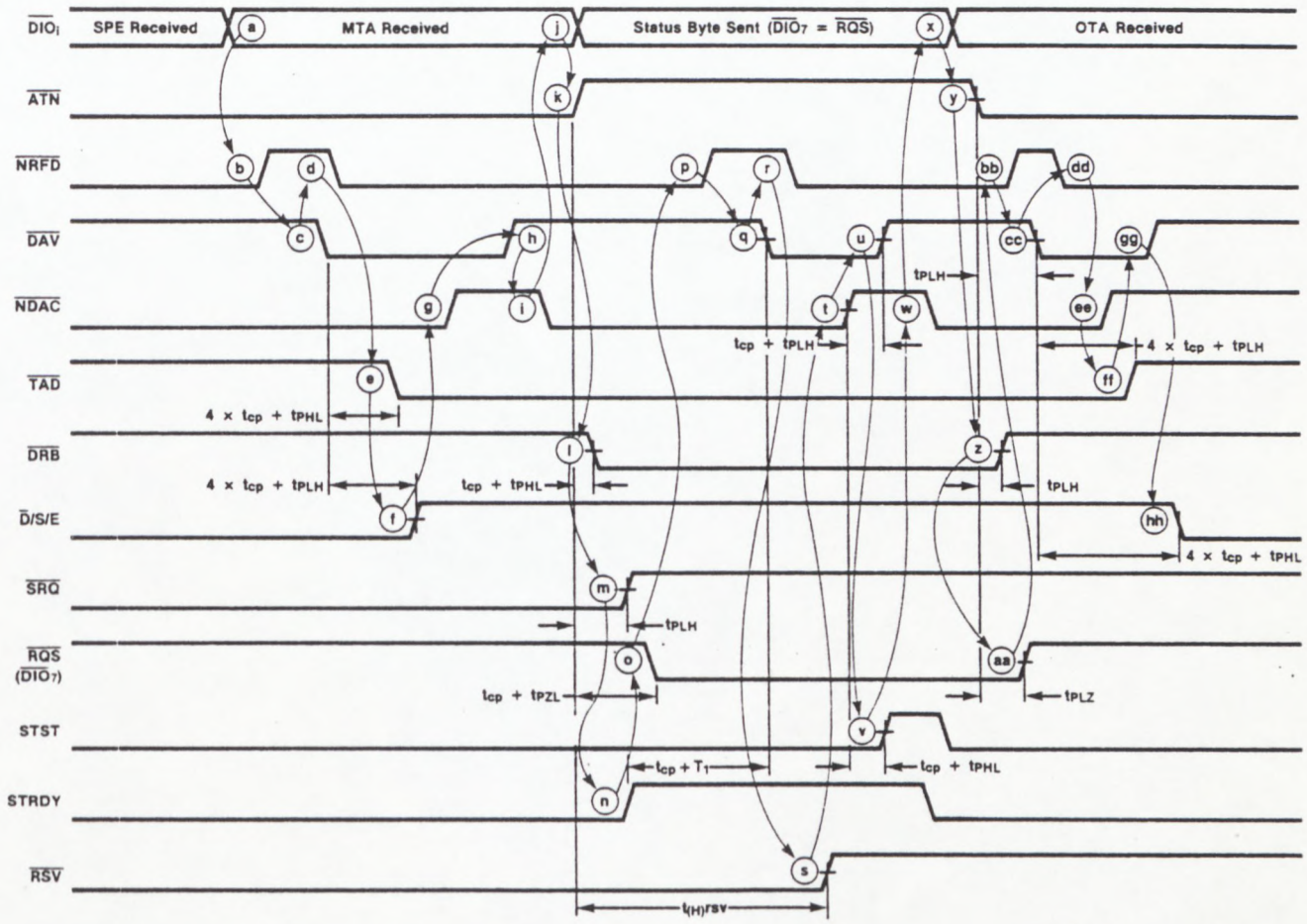
ee. $\overline{NDAC}$ goes HIGH when devices all acknowledge acceptance of the command byte

ff. The 96LS488 has received the Other Talk Address (OTA) command and reverts to its unaddressed state. $\overline{TAD}$ goes HIGH at time ($4 \times t_{cp} + t_{PLH}$) after $\overline{DAV}$ went LOW

gg. $\overline{DAV}$ is set HIGH by the Controller

hh. Because the 96LS488 has been unaddressed it is no longer in the Serial Poll Active State (SPAS). The $\overline{D}$/S/E output goes LOW at time ($4 \times t_{cp} + t_{PHL}$) after $\overline{DAV}$ went HIGH. The 96LS488 is still in the Serial Poll Mode State (SPMS), however, and will return to SPAS ($\overline{D}$/S/E HIGH, STST and STRDY valid) if subsequently addressed to talk. The 96LS488 enters the Serial Poll Idle State (SPIS) when the Controller either issues the Serial Poll Disable (SPD) command or makes $\overline{IFC}$ LOW.

Fig. 16 Timing Diagram for Serial Poll Sequence



Parallel Poll Sequence (See *Figure 17*)

a. The Controller sends the instrument's listen address

b. Then Controller issues the Parallel Poll Configure command: this enables the 96LS488 to receive a subsequent PPE command.

c. The Controller issues the Parallel Poll Enable command. Bits $\overline{D}_1$–$\overline{D}_3$ of the command byte determine which data output ($\overline{DIO}_j$) will be valid when the IDY command is sent. Bit $\overline{D}_4$ of the command is compared with IST (Instrument Status) during the IDY command.

d. The Controller issues the Unlisten command. The 96LS488 will now not respond to further PPE commands, allowing the Controller to configure other instruments.

e. The Instrument Status bit (IST) is set by the instrument before an IDENTIFY command is received. IST must be in a stable state when IDY is received so the setup and hold times must be observed (t_s IST and t_h IST).

f. During the IDY command the Controller releases the data lines $\overline{DIO}_1$–$\overline{DIO}_7$, $\overline{DO}_8$. The assigned data line $\overline{DIO}_j$ will be taken active LOW by the 96LS488 if IST compares with bit $\overline{D}_4$ of the PPE command.

g. The IDY message is received (IDY = $\overline{EOI} \cdot \overline{ATN}$). At this time the Instrument Status bit IST is latched in the 96LS488, and the output data $\overline{DIO}_j$ is true if IST compares with bit $\overline{D}_4$ of the PPE command. $\overline{DIO}_j$ is LOW if $\overline{D}_4$ was LOW and IST was HIGH, or if $\overline{D}_4$ was HIGH and IST was LOW.

h. The 96LS488 enables data bit $\overline{DIO}_j$ after a delay t_{PHL} from the time $\overline{EOI}$ goes active LOW. $\overline{DIO}_j$ remains valid while IDY is active. Note that this is an asynchronous message sent and is not governed by the handshake protocol.

i. IST may be altered not less than time t_h IST after IDY is active. Because IST is latched by IDY it will not affect the status byte sent during this IDY routine.

j. IDY is false and the 96LS488 stops sending a status byte. Outputs $\overline{DIO}_1$-$\overline{DIO}_7$, $\overline{DO}_8$ again float passive HIGH.

k. The status bit $\overline{DIO}_j$ floats passive HIGH at a time not greater than t_{PLH} after IDY goes FALSE.

l. The Controller can now place information on the data bus.

m. Once the 96LS488 has been configured via steps a-c, the Controller can examine IST at any time by issuing the IDY command. To change the $\overline{D}_1$-$\overline{D}_4$ assignment of a particular 96LS488, the Controller must address it to listen, issue the PPC command, then the Parallel Poll Disable (PPD) command (which clears the $\overline{D}_1$-$\overline{D}_4$ latches), then the PPE command with the revised $\overline{D}_1$-$\overline{D}_4$ assignment. The $\overline{D}_1$-$\overline{D}_4$ assignment will also be cleared by the universal Parallel Poll Unconfigure (PPU) command or by $\overline{MR}$, but not by $\overline{IFC}$. PPU, $\overline{MR}$ or the MLA/PPC/PPD sequence puts the 96LS488 in the Parallel Poll Idle State (PPIS) and it will not respond to IDY until it is subsequently reconfigured.

Fig. 17 Timing Diagram for Parallel Poll Sequence

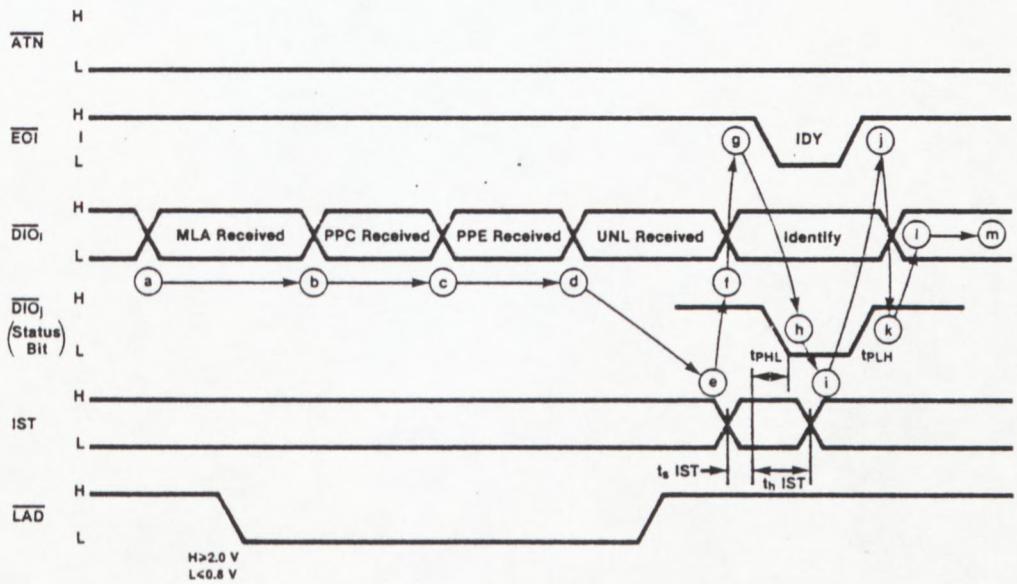
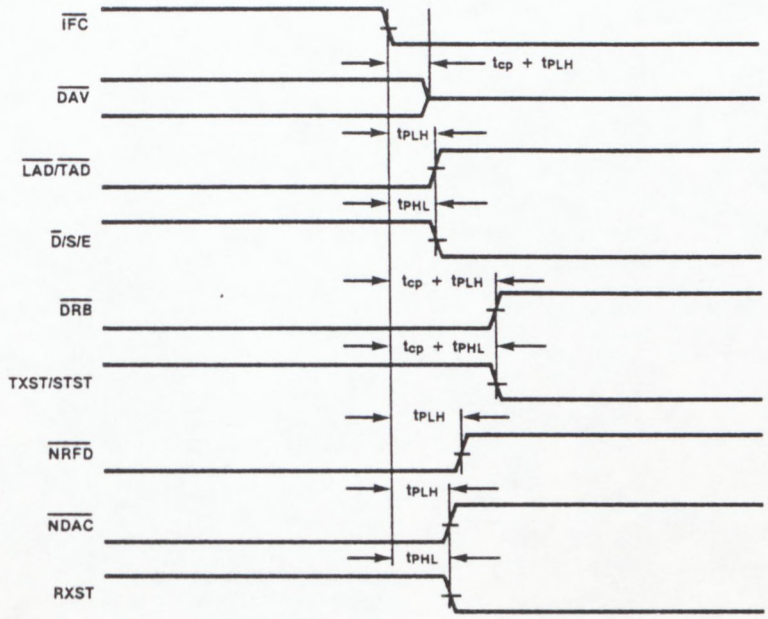


Fig. 18 IFC Timing Diagram



Absolute Maximum Ratings
(above which the useful life may be impaired)

Storage Temperature	−65 °C to +150 °C
Temperature (Ambient) Under Bias	−55 °C to +125 °C
V_{CC} Pin Potential to Ground Pin	−0.5 V to +7.0 V
*Input Voltage (dc)	−0.5 V to +15 V
*Input Current (dc)	−30 mA to +5.0 mA
Voltage Applied to Outputs (Output HIGH)	−0.5 V to +5.5 V
Output Current (dc) (Output LOW)	+50 mA

*Either Input Voltage limit or Input Current limit is sufficient to protect the inputs.

DC Characteristics

Symbol	Parameter	Min	Max	Unit	V_{CC}	Condition
V_{IH}	Input HIGH Voltage All Inputs	2.0		V		Recognized as a HIGH Signal Over Recommended V_{CC} and T_A Range
V_{IL}	Input LOW Voltage DM All Inputs DC		0.7 0.8	V		Recognized as a LOW Signal Over Recommended V_{CC} and T_A Range
$V_{T+}-V_{T-}$	Hysteresis Voltage	0.4		V		All Bus Inputs
V_{CD}	Input Clamp Diode Voltage		1.2	V	Min	I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage $\overline{RQS}$, $\overline{DAV}$	2.5		V	Min	I_{OH} = −5.2 mA
	$\overline{DIO}_1 - \overline{DIO}_7$, $\overline{DO}_8$, $\overline{SRQ}$, $\overline{NRFD}$, $\overline{NDAC}$	2.5	3.7	V	Min	Open-Collector Bus Pins I_{OH} = 0 mA
	$\overline{R}/L$, $\overline{D}/S/E$, $\overline{RXST}$, $\overline{TXST}$, STST, $\overline{CLR}$, $\overline{TRIG}$, $\overline{DRB}$, ASEL, XTAL, $\overline{LAD}$, $\overline{TAD}$	2.5		V	Min	I_{OH} = −0.4 mA
V_{OL}	Output LOW Voltage $\overline{DIO}_1 - \overline{DIO}_7$, $\overline{DO}_8$, $\overline{SRQ}$, $\overline{RQS}$, $\overline{NRFD}$, $\overline{NDAC}$, $\overline{DAV}$		0.5	V	Min	I_{OL} = 48 mA
	All Other Outputs DM, DC		0.4	V	Min	I_{OL} = 4.0 mA
	All Other Outputs DC		0.5	V	Min	I_{OL} = 8.0 mA
I_{IH}	0.5 U.L. Input HIGH Current 1.0 U.L. n U.L.		20 40 n(40)	μA	Max	I_{IH} = 40 μA Multiplied by Input HIGH U.L. Shown on Data Sheet; V_{IN} = 2.7 V
I_{IL}	0.25 U.L. Input LOW Current 0.5 U.L. n U.L.		−0.4 −0.8 n(−1.6)	mA	Max	I_{IL} = −1.6 mA Multiplied by Input LOW U.L. Shown on Data Sheet; V_{IN} = 0.4 V
I_{POFS}	Leakage into GPIB Pins in Powered-off State		100	μA	0	V_{IN} = 2.5 V
I_{OZH}	3-State Output OFF Current HIGH		40	μA	Max	V_{OUT} = 2.4 V
I_{OZL}	3-State Output OFF Current LOW		−40	μA	Max	V_{OUT} = 0.4 V
I_{CC}	Power Supply Current	*	250	mA	Max	Any 3 Bus Outputs in LOW State

*Typical Value for I_{CC} is 180 mA.

AC Characteristics V_{CC} = 5.0 V, T_A = +25 °C

Symbol	Parameter	Min	Typ	Max	Unit	Fig. No.	Condition
f_{max}	Maximum Clock Frequency	12	15		MHz		For correct timing for SH or AH, $\overline{CP}$ must be 10 MHz.
t_{PLH} t_{PHL}	Propagation Delay $\overline{CP}$ to $\overline{NRFD}$		150 90		ns ns	7, 8, 10, 11 12, 13, 14	C_L = 300 pF, R_L = 200 Ω
t_{PLH}	Propagation Delay $\overline{ATN}$ to $\overline{NRFD}$		150		ns	11	C_L = 300 pF, R_L = 200 Ω
t_{PLH} t_{PHL}	Propagation Delay $\overline{CP}$ to $\overline{NDAC}$		150 90		ns ns	7, 8, 10, 11 12, 13, 14	C_L = 300 pF, R_L = 200 Ω
t_{PLH}	Propagation Delay $\overline{ATN}$ to $\overline{NDAC}$		150		ns	10, 12, 13	C_L = 300 pF, R_L = 200 Ω
t_{PLH} t_{PHL}	Propagation Delay $\overline{CP}$ to $\overline{DAV}$		100 90		ns ns	9	C_L = 300 pF, R_L = 200 Ω
t_{PLZ} t_{PHZ}	Output Disable Time $\overline{ATN}$ to $\overline{DAV}$		100 100		ns ns	9	C_L = 300 pF, R_L = 200 Ω
t_{PZL} t_{PZH}	Output Enable Time $\overline{CP}$ to $\overline{DAV}$		100 100		ns ns	9	C_L = 300 pF, R_L = 200 Ω
t_{PLZ} t_{PHZ}	Output Disable Time $\overline{CP}$ to $\overline{DAV}$		100 100		ns ns		C_L = 300 pF, R_L = 200 Ω
t_{PLH}	Propagation Delay $\overline{IFC}$ to $\overline{NRFD}$ or $\overline{NDAC}$		180		ns	18	C_L = 300 pF, R_L = 200 Ω
t_{PHL}	Propagation Delay $\overline{IFC}$ to RXST		160		ns	18	C_L = 15 pF
t_{PHL}	Propagation Delay, $\overline{IFC}$ to $\overline{D}$/S/E		100		ns	18	C_L = 15 pF
t_{PLH} t_{PHL}	Propagation Delay $\overline{CP}$ to $\overline{LAD}$ or $\overline{TAD}$		150 150		ns ns	7, 8, 10	C_L = 15 pF
t_{PLH} t_{PHL}	Propagation Delay $M_0 - M_3$ to $\overline{LAD}$ or $\overline{TAD}$		60 60		ns ns		C_L = 15 pF
t_{PLH} t_{PHL}	Propagation Delay $\overline{CP}$ to RXST, TXST or STST		75 75		ns ns	8, 9	C_L = 15 pF
t_{PLH} t_{PHL}	Propagation Delay $\overline{CP}$ to $\overline{DRB}$		85 85		ns ns	9, 11	C_L = 15 pF
t_{LH}	Propagation Delay, $\overline{ATN}$ to $\overline{DRB}$		70		ns	9	C_L = 15 pF
t_{PLH} t_{PHL}	Propagation Delay $\overline{CP}$ to $\overline{TRIG}$ or $\overline{CLR}$		130 130		ns ns	13	C_L = 15 pF
t_{PLH} t_{PHL}	Propagation Delay $\overline{DAV}$ to ASEL		25 25		ns ns	11	C_L = 15 pF

AC Characteristics V_{CC} = 5.0 V, T_A = +25°C (Cont'd)

Symbol	Parameter	Min	Typ	Max	Unit	Fig. No.	Condition
t_{PHL}	Propagation Delay, $\overline{MR}$ to RXST, TXST, STST, $\overline{D}$/S/E		100		ns		C_L = 15 pF
t_{PLH}	Propagation Delay $\overline{MR}$ to $\overline{LAD}$, $\overline{TAD}$ or $\overline{R}$/L		50		ns		C_L = 15 pF
t_{PLH}	Propagation Delay, $\overline{MR}$ to $\overline{NRFD}$, $\overline{NDAC}$, $\overline{DAV}$, $\overline{DIO_1}$-$\overline{DIO_7}$, $\overline{DO_8}$ or $\overline{SRQ}$		100		ns		C_L = 300 pF, R_L = 200 Ω
t_{PLH} t_{PHL}	Propagation Delay $\overline{CP}$ to XTAL		15 15		ns ns	6	C_L = 15 pF, $\overline{CP}$ driven from ext. oscillator
t_{PLH} t_{PHL}	Propagation Delay, $\overline{CP}$ to $\overline{R}$/L		150 150		ns ns	14, 15	C_L = 15 pF
t_{PLH}	Propagation Delay, $\overline{RTL}$ to $\overline{R}$/L		150		ns	14	C_L = 15 pF
t_{PLH}	Propagation Delay, $\overline{REN}$ to $\overline{R}$/L		45		ns	14	C_L = 15 pF
t_{PLH} t_{PHL}	Propagation Delay, $\overline{CP}$ to $\overline{D}$/S/E		140 140		ns ns	16	C_L = 15 pF
t_{PLH} t_{PHL}	Propagation Delay, $\overline{RSV}$ to $\overline{SRQ}$		90 30		ns ns		C_L = 300 pF, R_L = 200 Ω
t_{PLH}	Propagation Delay, $\overline{ATN}$ to $\overline{SRQ}$		150		ns	16	C_L = 300 pF, R_L = 200 Ω
t_{PZL} t_{PZH}	Output Enable Time, $\overline{CP}$ to $\overline{RQS}$		100 100		ns ns	16	C_L = 300 pF, R_L = 200 Ω
t_{PLZ} t_{PHZ}	Output Disable Time $\overline{ATN}$ to $\overline{RQS}$		85 85		ns ns	16	C_L = 300 pF, R_L = 200 Ω
t_{PLH} t_{PHL}	Propagation Delay $\overline{EOI}$ to $\overline{DIO_1}$-$\overline{DIO_7}$, $\overline{DO_8}$		110 50		ns ns	17	C_L = 300 pF, R_L = 200 Ω
t_{PLH} t_{PHL}	Propagation Delay $\overline{EOI}$ to $\overline{D}$/S/E		30 30		ns ns		C_L = 15 pF

AC Operating Requirements V_{CC} = 5.0 V, T_A = +25 °C

Symbol	Parameter	Min	Typ	Max	Unit	Fig. No.	Condition
t_s (L) t_s (H)	Setup Time, HIGH or LOW IST to $\overline{EOI}$	20 20			ns ns	17	
t_h (L) t_h (H)	Hold Time, HIGH or LOW IST to $\overline{EOI}$	5.0 5.0			ns ns	17	
t_s (L)	Setup Time, $\overline{RSV}$ to $\overline{ATN}$		−50		ns		Device about to enter SPAS on next L to H $\overline{ATN}$ transition.
t_h (L)	Hold Time, $\overline{RSV}$ to $\overline{ATN}$		−50		ns		Device about to enter SPAS on next L to H $\overline{ATN}$ transition.
t_s (H) t_s (L)	Setup Time, HIGH or LOW $\overline{DIO_1}$-$\overline{DIO_7}$ to $\overline{DAV}$	100 100			ns ns	11	
t_h (H) t_h (L)	Hold Time, HIGH or LOW $\overline{DIO_1}$-$\overline{DIO_7}$ to $\overline{DAV}$	0 0			ns ns	11	
t_m (H) t_m (L)	Delay in external address mux from ASEL to valid address			100 100	ns ns	11	
t_w (H) t_w (L)	$\overline{CP}$ Pulse Width	30 30			ns ns		
t_w (L)	Pulse Width RXRDY, TXRDY or STRDY	20			ns	8, 9, 16	
t_w (L)	$\overline{MR}$ Pulse Width	50			ns		
t_w (L)	$\overline{RTL}$ Pulse Width	60			ns	14	

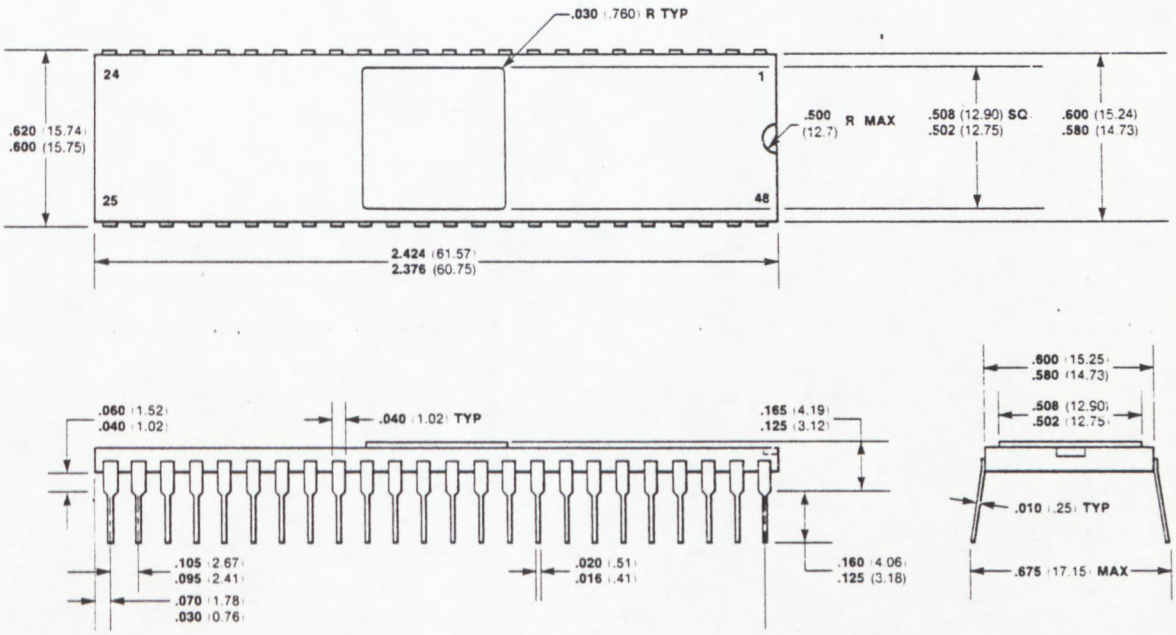
Ordering Information

Order Code	Package	Temperature*
96LS488DC	Ceramic DIP	Commercial
96LS488DM	Ceramic DIP	Military

*C = 0°C to +70°C
M = −55°C to +125°C

Package Outlines

48-Pin Side-brazed Package



Notes
All dimensions in inches **bold** and millimeters in parentheses
Pin material is nickel, gold-plated kovar or alloy 42
Cap is kovar or alloy 42
Base is ceramic
Cavity size is **.310** × **.310** (7.874 × 7.874)
Package weight is 7.7 grams

APPENDIX E

DATA PROCESSING BY HP85 COMPUTER

The IMT General Purpose Data Logger was designed to be able to output data, via the IEEE-488 port, to an external computer for data processing or display purposes.

In this instance the HP85 desk top computer was chosen because it is small, portable and has an integral CRT Display, Printer and Data Cassette as well as an optional IEEE-488 (HPIB) interface.

Two programs have been written (in Basic) which are controlled by menu prompted function keys:

Test (Autost):

This program is automatically loaded and run if the program tape is inserted before the HP85 is switched on. The program displays the following "header" for 3 seconds:

```
**********************************
*       I.M.T  DATA LOGGER       *
**********************************
*                                *
*PROGRAM NAME:TEST(Autost)       *
*                                *
*BY:MIKE GARDENER                *
*                                *
*DISCRIPTION:THIS PROGRAM TESTS*
*THE IMT LOGGER/HP85 IEEE 488   *
*INTERFACE OR ALLOWS THE HP85   *
*TO PLOT REAL TIME DATA [SCOPE]*
*                                *
*DATE:02/01/85                   *
**********************************
```

and then the select option menu and description

```
          SELECT OPTION!
          **************
TEST: HP85/DATA LOGGER IEEE-488
INTERCONECTION TEST

SCOPE: DISPLAYS PLOT OF REAL
       TIME DATA

READ: LOADS 'IMTLOG' PROGRAM
      (DATA READ & PLOT)
********************************

---------------------------------

TEST     SCOPE              READ
```

At this point the operator should ensure that the Data Logger is connected to the HP85 HPIB interface cable and is powered up.

TEST: [K1]

When selected the CRT will display:

IEEE TEST MODE
WAITING FOR DATA!

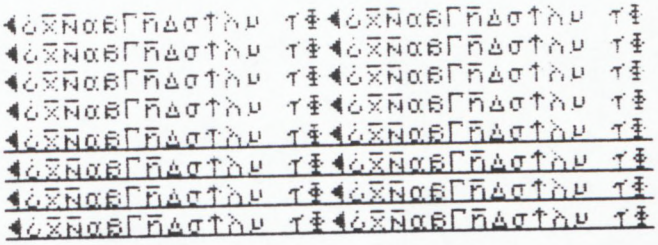
The Data Logger IEEE-488 port test procedure (7.5) should now be followed which will result in either

```
 !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz
 !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz

IEEE TEST PASS
```

being displayed for 2 seconds before reverting to the "Select Option" display.

OR



IEEE TEST FAILURE!!!
IEEE TEST FAILURE!!!
IEEE TEST FAILURE!!!

This display is accompanied by a beep sound and both the message and beep are repeated until the operator aborts the program.

If the test fails then the Data Logger / HP85 interconnection is suspect. Locate and rectify the cause and repeat TEST to make sure that it is now correct.

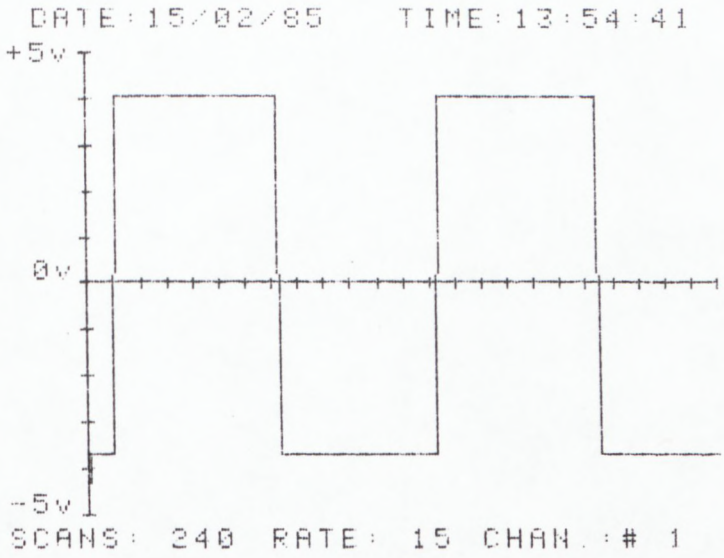
SCOPE: [K2]

This routine allows a pre-selected Data Logger channel to be displayed on the HP85 CRT in real-time. The axes and labels are initially set up and then

WAITING FOR DATA!

is displayed.

If a log is now initiated or the Data Acquisition Test procedure (7.4) is followed then the real-time data will be plotted onto the CRT.



Note: 1. This program is limited to a maximum scan rate of 125 msec and will only display the first 240 points.

2. The Data Logger IEEE MODE must be ON.

3. In the Data Acquisition Test mode the scan rate is governed by the speed at which the HP85 can receive and process the data i.e. scan rate is maximum for this configuration.

READ: [K4]

This option will load and run the second program, "IMTLOG", and display the following header for 3 seconds.

```
**********************************
*      I.M.T  DATA LOGGER        *
**********************************
*                                *
*PROGRAM NAME:IMTLOG             *
*                                *
*BY:MIKE GARDENER                *
*                                *
*DISCRIPTION:THIS PROGRAM READS*
*AND PLOTS DATA TRANSFERED FROM*
*THE I.M.T. DATA LOGGER RECORD *
*BY RECORD. (240 SCANS MAX.)    *
*                                *
*DATE:03/01/85                   *
**********************************
```

and then

```
        SELECT OPTION!
       ****************
HD-CPY: HARD COPY OF PLOT

PLOT: PLOTS DATA vs TIME FOR A
    SINGLE (SELECTABLE) CHANNEL

TXFR-D: TRANSFERS DATA FROM IMT
        LOGGER TO HP85

T-DUMP: DUMPS DATA TO HP TAPE

T-READ: READS DATA FROM HP TAPE
--------------------------------
                   T-READ
HD-CPY  PLOT       T-DUMP  TXFR-D
```

This is an option select description and menu for five new functions:

TXFR-D: [K4]

This function allows data to be transfered from the Data Logger to the HP85.

WAITING FOR DATA FROM LOGGER

is displayed until data is received.

The Data Logger DATA PLAYBACK procedure (7.8) with IEEE Mode ON, should now be followed.

After selecting the record to be read the [READ] key will initiate the Data Logger cassette read and IEEE data transfer. The HP85 will indicate the transfer status by displaying

TRANSFER OF DATA IN PROGRESS
Press [CONT] TO CONTINUE

Otherwise

TRANSFER OF DATA COMPLETE

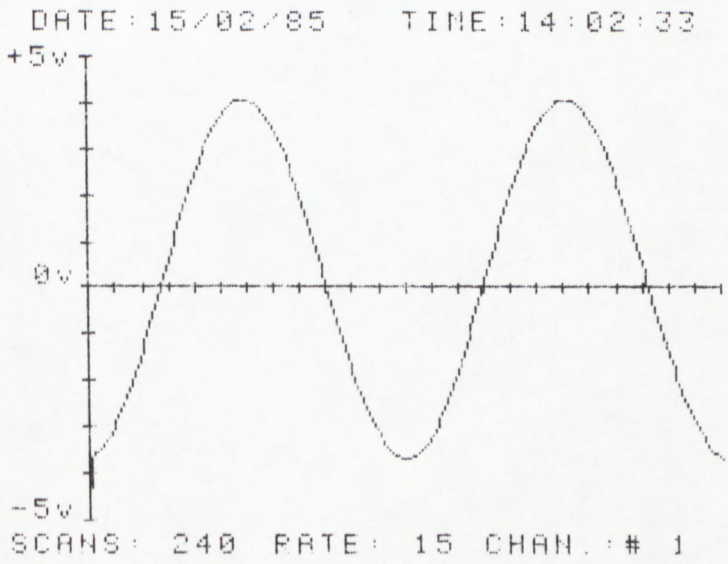
is displayed for 2 seconds before the returning to the "Select Option" display.

PLOT: [K2]

This routine allows a channel to be selected (limits 1 - 27) when

SELECT CHAN TO BE PLOTTED:
1 TO 27

is displayed. The desired number should be keyed in and then [END LINE]. The program now plots this channels data onto the CRT.



After the plot is complete the program returns to the "Select Option" display.

HD-CPY [K1]

Allows a CRT dump to the integral HP85 printer of the last PLOT.

T-DUMP [K3]

Allows the data transfered by TXFR-D to be dumped to the HP85 data tape. It first prompts:

INSERT HP DATA TAPE, PRESS [CONT]

and then ENTER FILE NAME

at which point a 6 character (max) file name should be entered. After the data is recorded the program returns to the "Select Option" display.

T-READ [K7]

Reads back data from a HP85 data tape which has previously been recorded by T-DUMP.

INSERT HP DATA TAPE, PRESS [CONT]

is displayed and then ENTER FILE NAME.

The file name used in T-DUMP must be entered which allows the data to be read back into the HP85 memory.

The program then returns to the "Select Option" display.

HP85 COMPUTER
PROGRAM LISTINGS

```
10 DISP "**********************************"
20 DISP "*      I.M.T  DATA LOGGER       *"
30 DISP "**********************************"
40 DISP "*                                *"
50 DISP "*PROGRAM NAME:TEST(Autost)       *"
60 DISP "*                                *"
70 DISP "*BY:MIKE GARDENER                *"
80 DISP "*                                *"
90 DISP "*DISCRIPTION:THIS PROGRAM TESTS*"
100 DISP "*THE IMT LOGGER/HP85 IEEE 488  *"
110 DISP "*INTERFACE OR ALLOWS THE HP85  *"
120 DISP "*TO PLOT REAL TIME DATA [SCOPE]*"
130 DISP "*                               *"
140 DISP "*DATE:02/01/85                  *"
150 DISP "**********************************"
160 WAIT 3000
170 OPTION BASE 1
180 ON KEY# 1,"TEST" GOTO 530
190 ON KEY# 2,"SCOPE" GOTO 330
200 ON KEY# 4,"READ" GOTO 760
210 CLEAR
220 KEY LABEL
230 DISP "        SELECT OPTION!"
240 DISP "        ***************"
250 DISP "TEST: HP85/DATA LOGGER IEEE-488       INTERCONECTION TEST"
260 DISP @ DISP "SCOPE: DISPLAYS PLOT OF REAL          TIME DATA"
270 DISP @ DISP "READ: LOADS 'IMTLOG' PROGRAM         (DATA READ & PLOT)"
280 DISP "**********************************"
290 GOTO 290
300 ! ***************************
310 ! *        SCOPE            *
320 ! ***************************
330 GCLEAR
340 SCALE -30,250,-500,10500
350 XAXIS 5000,10,0,240
360 YAXIS 0,1000,0,10000
370 MOVE -30,10000 @ LABEL "+5v"
380 MOVE -20,5000 @ LABEL "0v"
390 MOVE -30,0 @ LABEL "-5v"
400 CLEAR
410 DISP "WAITING FOR DATA!"
420 ENTER 710 USING "#,W" ; A
430 MOVE 0,A
440 FOR I=1 TO 239
450 ENTER 710 USING "#,W" ; A
460 DRAW I,A
470 NEXT I
480 WAIT 2000
490 GOTO 210
500 ! ***************************
510 ! *      IEEE TEST          *
520 ! ***************************
530 DIM A$[264]
540 IOBUFFER A$
550 A,B=0
560 CLEAR
570 DISP "IEEE TEST MODE"
580 DISP @ DISP
590 DISP "  WAITING FOR DATA!"
```

```
600 BEEP
610 TRANSFER 710 TO A$ FHS
620 CLEAR
630 DISP A$
640 FOR I=1 TO 256
650 A=NUM(A$[I])+A
660 B=B+I-1
670 NEXT I
680 BEEP
690 DISP @ DISP
700 IF A=B THEN DISP "IEEE TEST PASS" @ WAIT 2000 @ GOTO 210
710 IF A<>B THEN DISP "IEEE TEST FAILURE!!!" @ BEEP
720 GOTO 710
730 ! ************************
740 ! *        READ          *
750 ! ************************
760 CHAIN "IMTLOG"
770 END
```

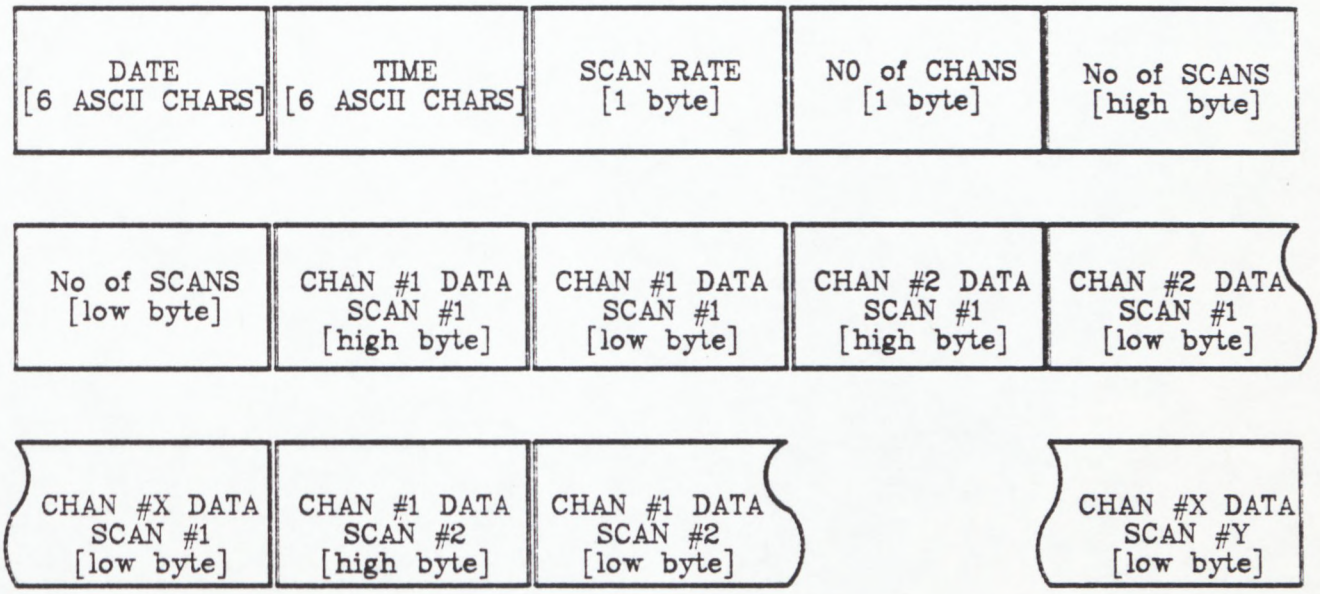
```
10 DISP "*********************************"
20 DISP "*      I.M.T. DATA LOGGER       *"
30 DISP "*********************************"
40 DISP "*                               *"
50 DISP "*PROGRAM NAME:IMTLOG            *"
60 DISP "*                               *"
70 DISP "*BY:MIKE GARDENER               *"
80 DISP "*                               *"
90 DISP "*DISCRIPTION:THIS PROGRAM READS*"
100 DISP "*AND PLOTS DATA TRANSFERED FROM*"
110 DISP "*THE I.M.T. DATA LOGGER RECORD *"
120 DISP "*BY RECORD. (240 SCANS MAX.)    *"
130 DISP "*                               *"
140 DISP "*DATE:03/01/85                  *"
150 DISP "*********************************"
160 WAIT 3000
170 DIM A$[24],B$[13000],D$[6],T$[6],X$[6]
180 ON KEY# 1,"HD-CPY" GOTO 870
190 ON KEY# 2,"PLOT" GOTO 540
200 ON KEY# 3,"T-DUMP" GOTO 910
210 ON KEY# 4,"TXFR-D" GOTO 350
220 ON KEY# 7,"T-READ" GOTO 1030
230 CLEAR
240 KEY LABEL
250 DISP "        SELECT OPTION!"
260 DISP "        ***************"
270 DISP "HD-CPY: HARD COPY OF PLOT"
280 DISP @ DISP "PLOT: PLOTS DATA vs TIME FOR A      SINGLE (SELECTABLE) CHANNEL
"
290 DISP @ DISP "TXFR-D: TRANSFERS DATA FROM IMT         LOGGER TO HP85"
291 DISP @ DISP "T-DUMP: DUMPS DATA TO HP TAPE"
292 DISP @ DISP "T-READ: READS DATA FROM HP TAPE"
310 GOTO 310
320 ! *************************
330 ! *       TRANSFER        *
340 ! *************************
350 IOBUFFER A$
360 IOBUFFER B$
370 CLEAR @ DISP "WAITING FOR DATA FROM LOGGER"
380 TRANSFER 710 TO A$ FHS
390 BEEP ! FIRST TXFR COMPLETE
400 CLEAR @ DISP "TRANSFER OF DATA IN PROGRESS"
410 D$=A$[1,6]
420 T$=A$[7,12]
430 R=NUM(A$[13])
440 C=NUM(A$[14])
450 S=NUM(A$[15])*256+NUM(A$[16])
460 IF C<>27 THEN GOTO 800
470 B1=C*S*2
480 TRANSFER 710 TO B$ FHS ; COUNT B1
490 CLEAR @ DISP "TRANSFER OF DATA COMPLETE" @ BEEP @ WAIT 2000
500 GOTO 230
510 ! *************************
520 ! *         PLOT          *
530 ! *************************
540 CLEAR
550 DISP "SELECT CHAN TO BE PLOTTED:"
560 DISP "1 TO";C
570 INPUT D1
580 GCLEAR
590 SCALE -30,250,-1000,11000
```

```
600 XAXIS 5000,10,0,240
610 YAXIS 0,1000,0,10000
620 MOVE -30,9750 @ LABEL "+5v"
630 MOVE -20,5000 @ LABEL "0v"
640 MOVE -30,0 @ LABEL "-5v"
650 MOVE -20,10500 @ LABEL "DATE:",D$[1,2],"/",D$[3,4],"/",D$[5,6]
660 MOVE 120,10500 @ LABEL "TIME:",T$[1,2],":",T$[3,4],":",T$[5,6]
670 MOVE -30,-750 @ LABEL "SCANS:",S
680 MOVE 70,-750 @ LABEL "RATE:",R
690 MOVE 150,-750 @ LABEL "CHAN.:#",D1
700 MOVE 0,0
710 B2=D1*2-1
720 FOR I=1 TO S
730 A=NUM(B$[B2])*256+NUM(B$[B2+1])
740 A=INT(A/16*10000/4095)
750 DRAW I,A
770 B2=B2+C*2
780 NEXT I
790 WAIT 2000 @ GOTO 230
800 CLEAR @ DISP "TAPE READ ERROR!!"
810 PAUSE
820 DISP @ DISP "PRESS [CONT] TO CONTINUE"
830 GOTO 230
840 ! ************************
850 ! *      HARD COPY       *
860 ! ************************
870 GRAPH @ COPY @ GOTO 230
880 ! ************************
890 ! *     HP TAPE DUMP     *
900 ! ************************
910 CLEAR @ DISP "INSERT HP DATA TAPE,PRESS [CONT]"
920 PAUSE
930 DISP "ENTER FILE NAME"
940 INPUT X$[1,6]
950 CREATE X$,1,13056
960 ASSIGN# 1 TO X$
970 PRINT# 1 ; A$,B$
980 ASSIGN# 1 TO *
990 GOTO 230
1000 ! ************************
1010 ! *     HP TAPE READ     *
1020 ! ************************
1030 CLEAR @ DISP "INSERT HP DATA TAPE,PRESS [CONT]"
1040 PAUSE
1050 DISP "ENTER FILE NAME"
1060 INPUT X$[1,6]
1070 ASSIGN# 1 TO X$
1080 READ# 1 ; A$,B$
1090 ASSIGN# 1 TO *
1100 D$=A$[1,6]
1110 T$=A$[7,12]
1120 R=NUM(A$[13])
1130 C=NUM(A$[14])
1140 S=NUM(A$[15])*256+NUM(A$[16])
1150 IF C<>27 THEN GOTO 800
1160 GOTO 230
1170 END
```

APPENDIX F

TAPE HEADER and DATA FORMAT

TAPE HEADER & DATA FORMAT.



NOTE: X is determined by No of CHANS selected.
Y is determined by No of SCANS selected.

DATE = 6 ASCII characters in DDMMYY format.

TIME = 6 ASCII characters in HHMMSS format.

SCAN RATE = 1 binary byte [limits 1–15]

No of CHANS = 1 binary byte [limits 1–32]

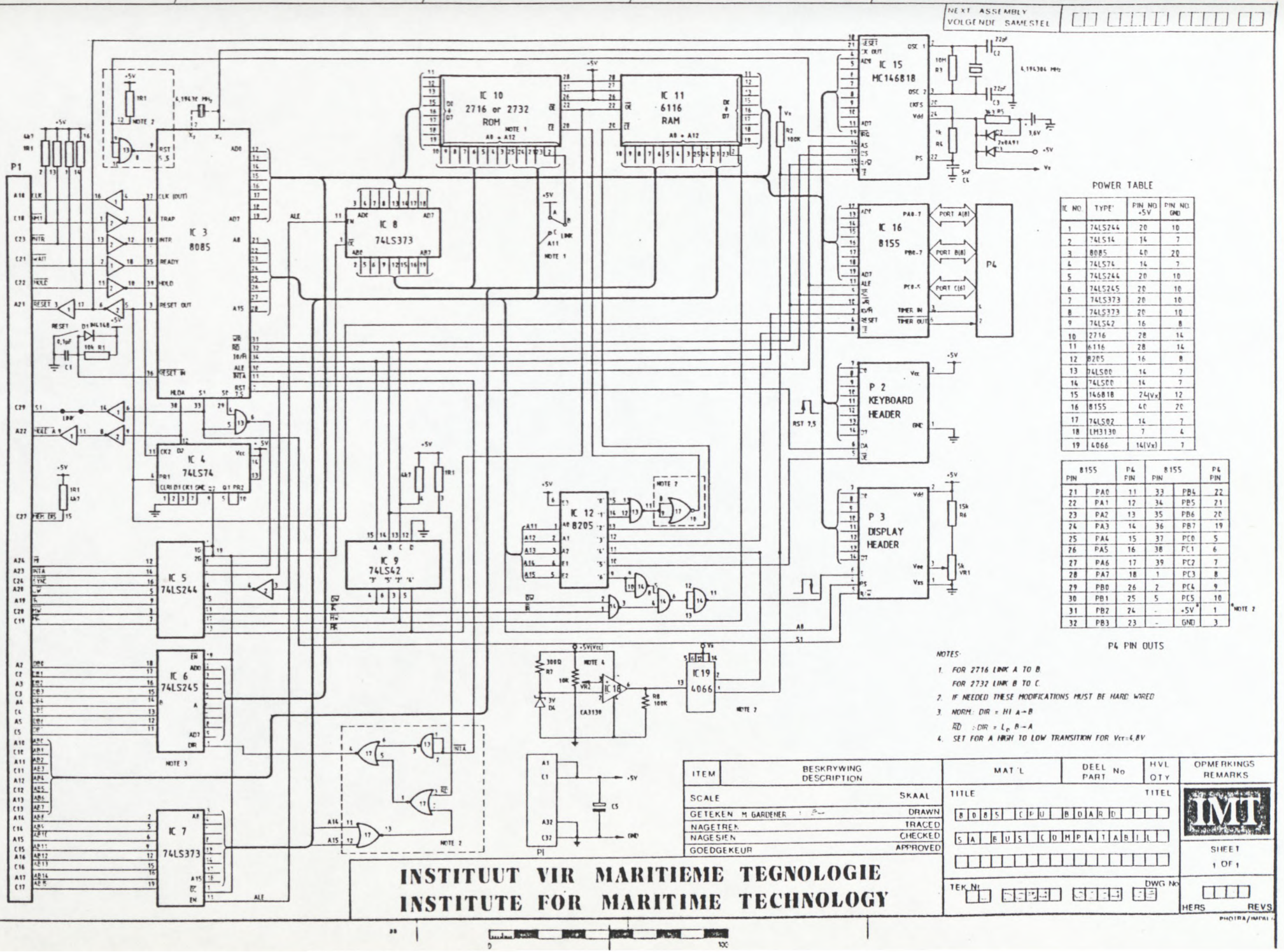
No of SCANS = 2 binary bytes [limits 1–9999]

DATA [high byte] = 8 msb's

DATA [low byte] = 4 lsb's & 4 trailing zeros

APPENDIX G

Component Lists, Component Overlays
and Schematic Diagrams



POWER TABLE

IC NO.	TYPE	PIN NO. +5V	PIN NO. GND
1	74LS244	20	10
2	74LS14	14	7
3	8085	40	20
4	74LS74	14	7
5	74LS244	20	10
6	74LS245	20	10
7	74LS373	20	10
8	74LS373	20	10
9	74LS42	16	8
10	2716	28	14
11	6116	28	14
12	8205	16	8
13	74LS00	14	7
14	74LS00	14	7
15	146818	24(Vx)	12
16	8155	40	20
17	74LS02	14	7
18	LM3130	7	4
19	4066	14(Vx)	7

8155 PIN	P4 PIN	8155 PIN	P4 PIN
21	PA0	11	33	PB4	22
22	PA1	12	34	PB5	21
23	PA2	13	35	PB6	20
24	PA3	14	36	PB7	19
25	PA4	15	37	PC0	5
26	PA5	16	38	PC1	6
27	PA6	17	39	PC2	7
28	PA7	18	1	PC3	8
29	PB0	26	2	PC4	9
30	PB1	25	5	PC5	10
31	PB2	24	-	+5V	1
32	PB3	23	-	GND	3

NOTE 2

P4 PIN OUTS

NOTES:

1. FOR 2716 LINK A TO B
   FOR 2732 LINK B TO C
2. IF NEEDED THESE MODIFICATIONS MUST BE HARD WIRED
3. NORM: DIR = HI A→B
   $\overline{RD}$: DIR = L_o B→A
4. SET FOR A HIGH TO LOW TRANSITION FOR Vcc=4,8V

ITEM	BESKRYWING DESCRIPTION	MAT'L	DEEL No PART	HVL QTY	OPMERKINGS REMARKS
SCALE	SKAAL	TITLE		TITEL	
GETEKEN M GARDENER	DRAWN	8085 CPU BOARD			IMT
NAGETREK	TRACED	SAASBUS COMPATIBLE			
NAGESIEN	CHECKED				SHEET 1 OF 1
GOEDGEKEUR	APPROVED	TEK Nr	DWG No		HERS REVS

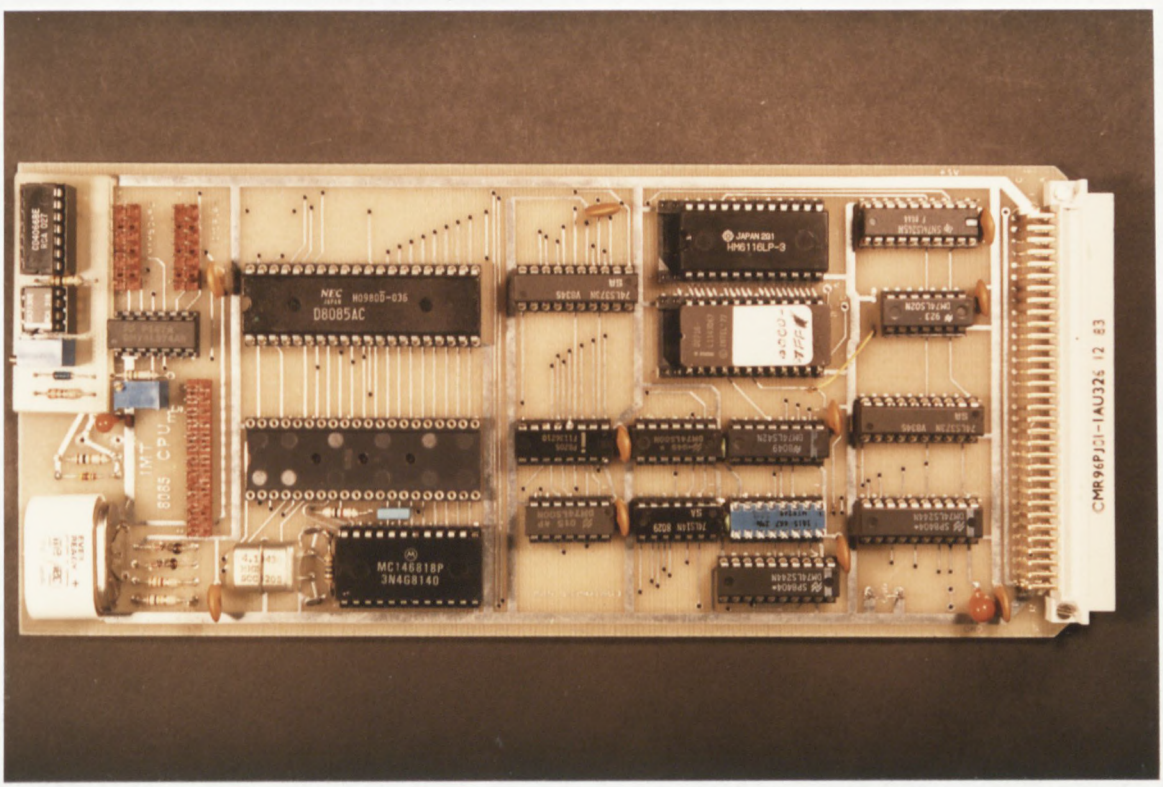
INSTITUUT VIR MARITIEME TEGNOLOGIE
INSTITUTE FOR MARITIME TECHNOLOGY

8085 C.P.U. Board

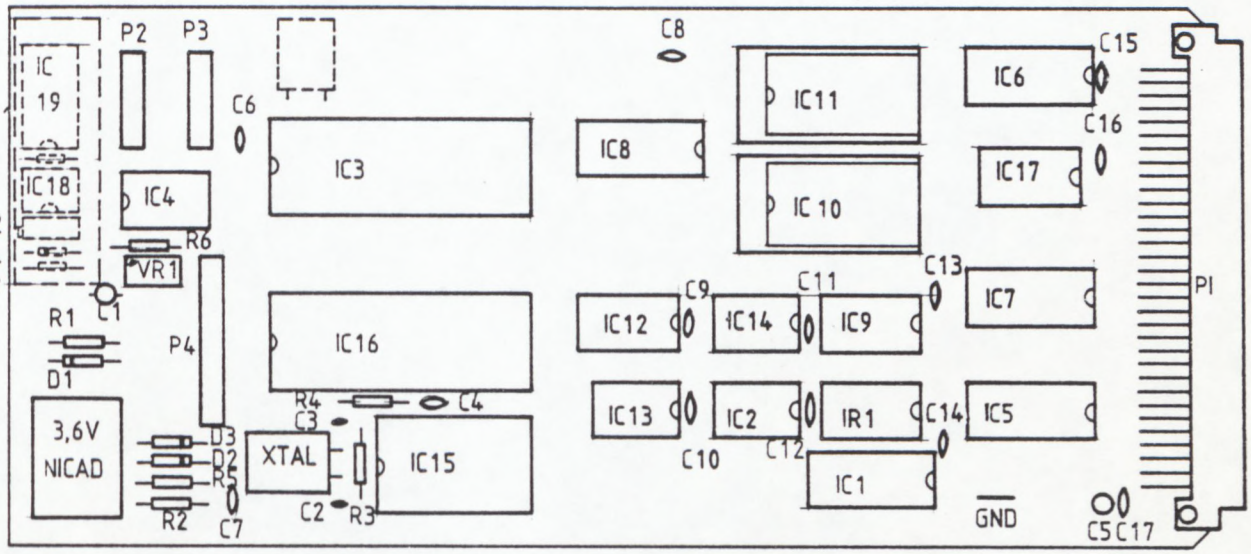
Parts List

R1	10k	$\frac{1}{4}$ Watt 5%
R2	5k6	$\frac{1}{4}$ Watt 5%
R3	10M	$\frac{1}{4}$ Watt 5%
R4	1k	$\frac{1}{4}$ Watt 5%
R5	3k3	$\frac{1}{4}$ Watt 5%
R6	15k	$\frac{1}{4}$ Watt 5%
VR1	5k	20 Turn
1R1	15x4k7	Type 1615
C1	0,1µF	Tantalum
C2	22pF	Ceramic
C3	22pF	"
C4	5nF	"
C5-17	0,01µF	"
D1	IN4148	
D2,3	OA91	
D4	3V Zenner	400mW
XTAL	4.194304 MHz	
Ni Cad	3,6V 100mA hr	
P2,3	4x7 Molex 0,1" Wafer	
P4	2x13 Molex 0,1" Wafer	
P1	DIN 41612 64 Pole Connector (A+C, 90°)	

1C1,5	74LS244
1C2	74LS14
1C3	8085
1C4	74LS74
1C6	76LS245
1C7,8	74LS373
1C9	74LS42
1C10	2716 or 2732 EPROM
1C11	6116 CMOS RAM
1C12	8205
1C13,14	74LS00
1C15	146818
1C16	8155
1C17	74LS02
1C18	LM3130
1C19	MC4066

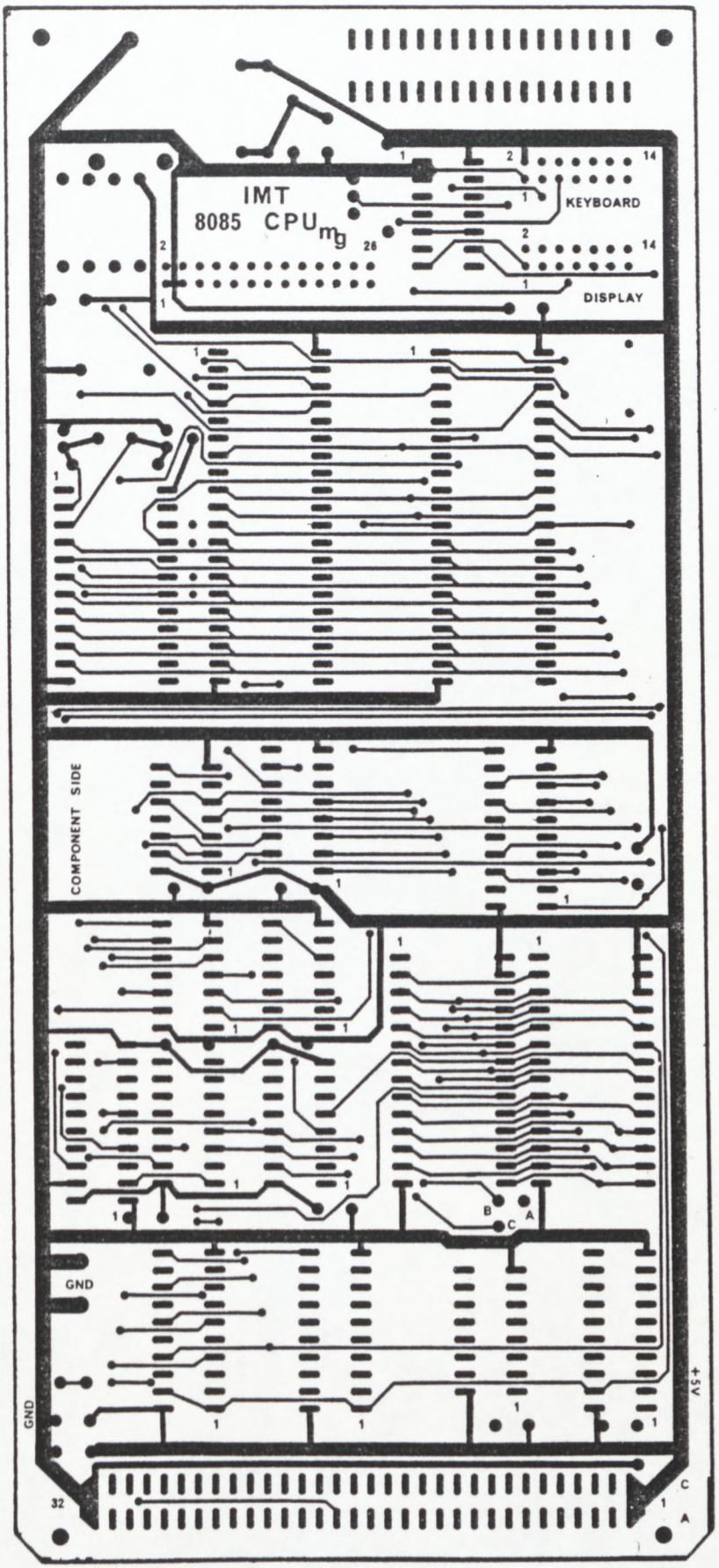


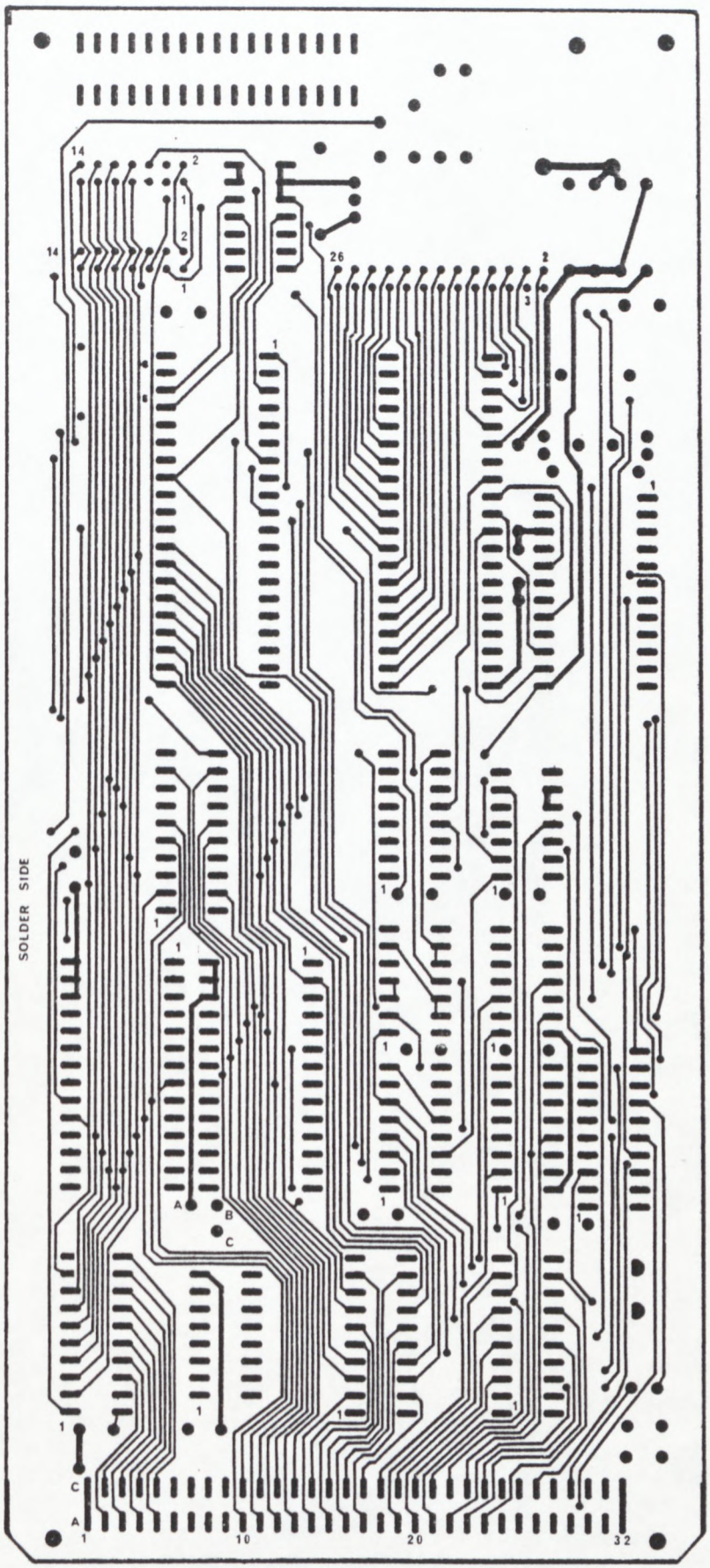
NEXT ASSEMBLY
VOLGENDE SAMESTEL

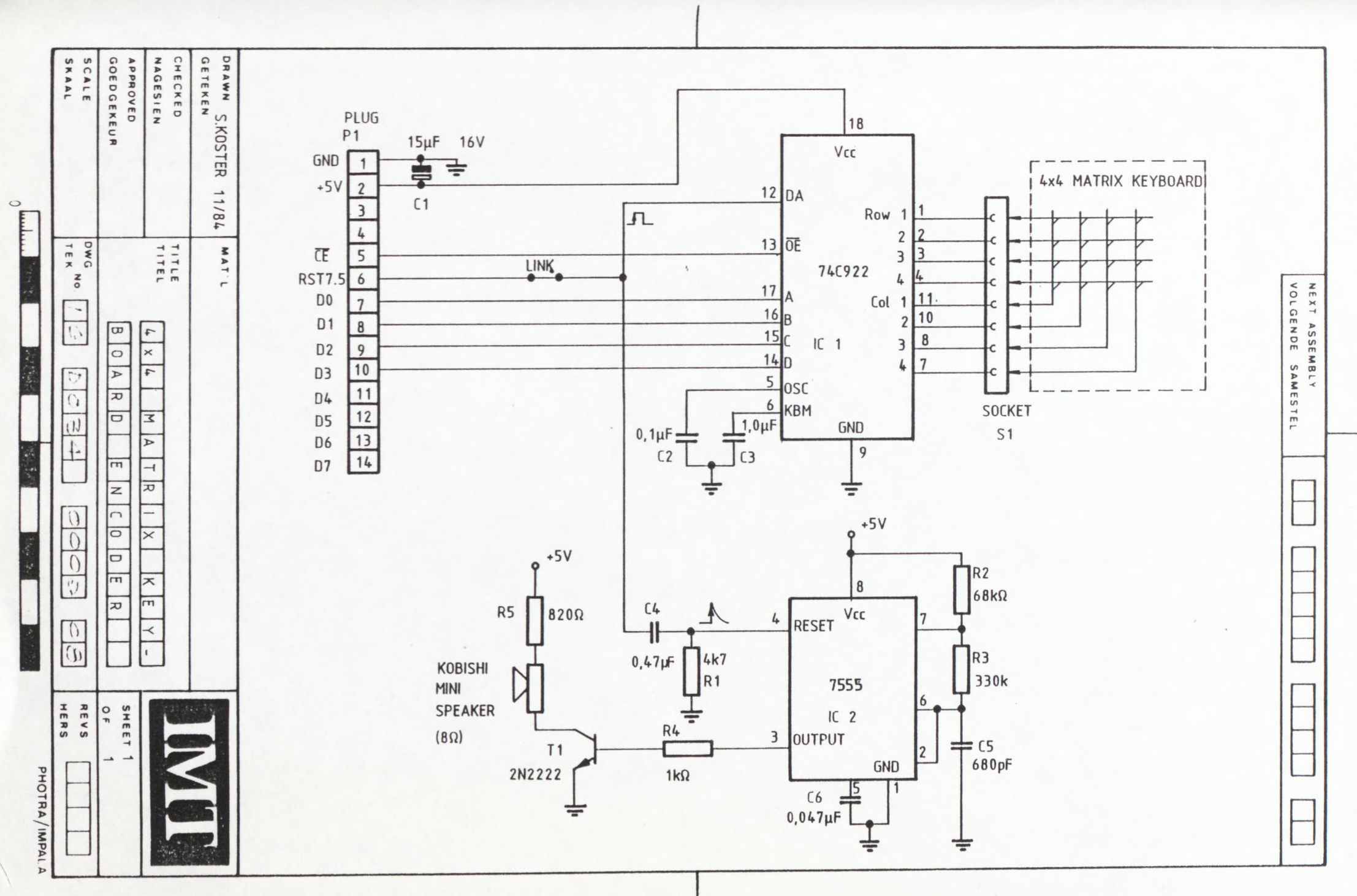


DRAWN GETEKEN L. N. 3/86	MAT'L	IMT
CHECKED NAGESIEN	TITLE TITEL 8085 C.P.U. BOARD OVERLAY	
APPROVED GOEDGEKEUR		SHEET OF
SCALE SKAAL	DWG TEK NO IMD-D034-0018-08	REVS HERS

PHOTRA/IMPALA


IMT
8085 CPUmg 26
KEYBOARD
DISPLAY
COMPONENT SIDE
GND
GND
+5V
32
1
A
B
C


SOLDER SIDE
14
14
26
A
B
C
C
A
1
10
20
32


PLUG
P1
GND
+5V
15µF 16V
C1
CE
RST7.5
D0
D1
D2
D3
D4
D5
D6
D7
LINK
18
Vcc
12 DA
13 OE
17 A
16 B
15 C
14 D
5 OSC
6 KBM
74C922
IC 1
GND
9
0,1µF
C2
1,0µF
C3
Row 1
Col 1
4x4 MATRIX KEYBOARD
SOCKET
S1
+5V
R5 820Ω
KOBISHI
MINI
SPEAKER
(8Ω)
T1
2N2222
C4
0,47µF
4k7
R1
R4
1kΩ
8
Vcc
4
RESET
7555
IC 2
3
OUTPUT
GND
1
5
C6
0,047µF
7
6
2
R2
68kΩ
R3
330k
C5
680pF
DRAWN S.KOSTER 11/84
GETEKEN
CHECKED
NAGESIEN
APPROVED
GOEDGEKEUR
SCALE
SKAAL
MAT'L
TITLE
TITEL
4x4 MATRIX KEYBOARD ENCODER
DWG
TEK No
REVS
MERS
SHEET 1
OF 1
NEXT ASSEMBLY
VOLGENDE SAMESTEL
PHOTRA/IMPALA

4x4 Matrix Keyboard Encoder

Parts List

R1	4k7	$\frac{1}{4}$ Watt 5%
R2	68k	$\frac{1}{4}$ Watt 5%
R3	330k	$\frac{1}{4}$ Watt 5%
R4	1k	$\frac{1}{4}$ Watt 5%
R5	820k	$\frac{1}{4}$ Watt 5%

C1	15µF 16V Tantalum
C2	0,1µF polycarbonate
C3	1,0µF 16V Tantalum
C4	0,47µF polycarbonate
C5	680pF ceramic
C6	0,047µF polycarbonate

T1	2N2222

S1	1x8 Molex socket

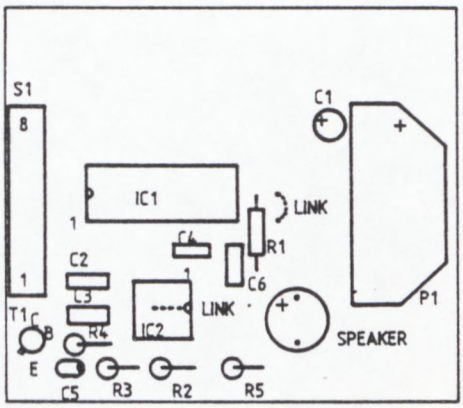
P1	2x7 Molex 0,1" Wafer

8Ω	Mini Speaker (Kobishi)

IC1	74C922
IC2	7555 (CMOS)

4x4	Matrix Keyboard

NEXT ASSEMBLY VOLGENDE SAMESTEL				



DRAWN GETEKEN S. KOSTER	MAT'L	IMT
CHECKED NAGESIEN	TITLE TITEL KEYBOARD ENCODER OVERLAY	
APPROVED GOEDGEKEUR		SHEET OF
SCALE SKAAL	DWG TEK No MD D034 0023 10	REVS HERS

PHOTRA/IMPALA

SABUS 32 k Static Memory Module-140/3

Originator: ISCOR

- 32 k memory addressable to 1 megabyte
- Interlaced RAM and EPROM in 2 k intervals
- Switch selectable page address
- Switch selectable 16 or 20-bit address option
- Switch selectable card address 0—32 k or 32—64 k
- Write protect switches for each of four 8 k blocks
- Low or high data byte option for 8- and 16-bit CPUs
- Single 5 V supply

Circuit description*

All the bus signals used on the card are buffered and place a maximum of one low-power Schottky TTL load on the buses.

Page address selection
1 megabyte in 64 k pages
The most significant four bits of address (AB16 to AB19) go to one of the inputs of four exclusive OR-gates (IC22). The switches G, H, I and K select one of 16 64 k pages of memory. When AB16 to AB19 contain the address selected on the switches, the open collector outputs of the XOR-gates will all be high. The wired AND of all these outputs goes to switch E. This switch allows the selection of 16 or 20 bits of addressable memory. The pull-up resistor R6 will maintain a logic high when the switch is open. This line is the page-enable line and goes to pins 10 of IC16, 9, 12, 13 of IC17 and 6 of IC19. Switches G, H, I and K correspond to AB16, AB17, AB18 and AB19 respectively. When switch E is open the page switches are irrelevant.

Card address selection
32 k at 0—32 k or 32—64 k
The card address is selected by switch F on pin 13 of IC21. When the switch is closed the output of IC21 pin 11 is low when AB15 is low; when switch F is open pin 11 will be low when AB15 is high. This line partly enables ICs 19 and 20.

Chip select lines
One of 16 2 k chips
The AB14 line, when low, enables pin 4 of IC19 and, when high, pin 6 of IC20. The 4 to 16 line decoder, ICs 19 and 20, decode address buses 11 to 14 to one of 16 active low chip select lines. IC0 is the lowest and IC15 is the highest addressable memory device.

Memory address
0 to 2 k
Address buses 0 to 10 go through buffers directly to the address pins of each of the 16 memories in the array.

Memory devices
The 32 k memory array may consist of a random mixture of RAM and EPROM. The EPROMs are of Intel type 2716 or TI 2516. The RAM may be Hitachi type 6116, Toshiba 5516 or OKI 2128.

Memory read
The output enable pins of the memory devices will go active low when the card is correctly addressed during a memory read cycle. The active low memory read line is buffered and inverted by IC16 pin 5; pins 3 and 4 are held high by the output of IC17 pin 6. The read line is qualified by the card address selection. This line (pin 8 IC16) controls the direction of the bidirectional data buffer.

Memory write
The active low memory write line (MW) is buffered and inverted by IC17 pin 1; pin 2 is held high by pin 6 of IC17. The memory write signal is again inverted by IC18. Each output of the four NAND gates is connected to an 8 k block of memory. The outputs are controlled by four dual in line switches, acccessible at the card panel. These switches A to D are the memory write protect switches. When open, the switches allow the memory write line on V_{pp}/WR (pin 21 of the memory devices) to become active low during a memory write cycle. In all other cases V_{pp} will remain at a logical high level.

The specifications are that V_{pp} must be low prior to V_{cc} going low. Somehow the data sheets led to the popular belief that V_{pp} may *never* be low. For the unbelievers, the write protect switch may be used — writing to an EPROM with V_{pp} being low is not harmful to the device.

Data buffers
The data may be placed on the low or high byte of the data buses by installing a 74LS245 in either socket IC25 or IC26. When IC25 is installed the data will appear on DB 0 to 7, and when IC26 is installed the data will go to DB 8 to 15. If both ICs are installed, identical 8-bit data will appear on the low and high bytes. The direction of the data path is always from the bus to the memory card, only a memory read operation will direct the data towards the data bus. The bidirectional data buffer will go into 'tri-state' when either MEMDIS is low or the page address is incorrect.

Using two 32 k memory modules with the same address, but one with a low data byte buffer and the other with a high data byte buffer, the combination will form a 16 bit memory of 32 k.

**The circuit diagram is overleaf*

Options summary

A, B, C and D switches
Write protect for RAM in 8 k blocks
closed = write protected.

E switch
AB16—AB19 disable
open = 16-bit address
closed = 20-bit address.

F switch
Card address selection
closed = 0 to 32 k
open = 32 to 64 k.

G, H, I and K switches
Page address selection

G = AB16
H = AB17
I = AB18
K = AB19

High data byte
Install only IC26 in buffer sockets.

Low data byte
Install only IC25 in buffer sockets.

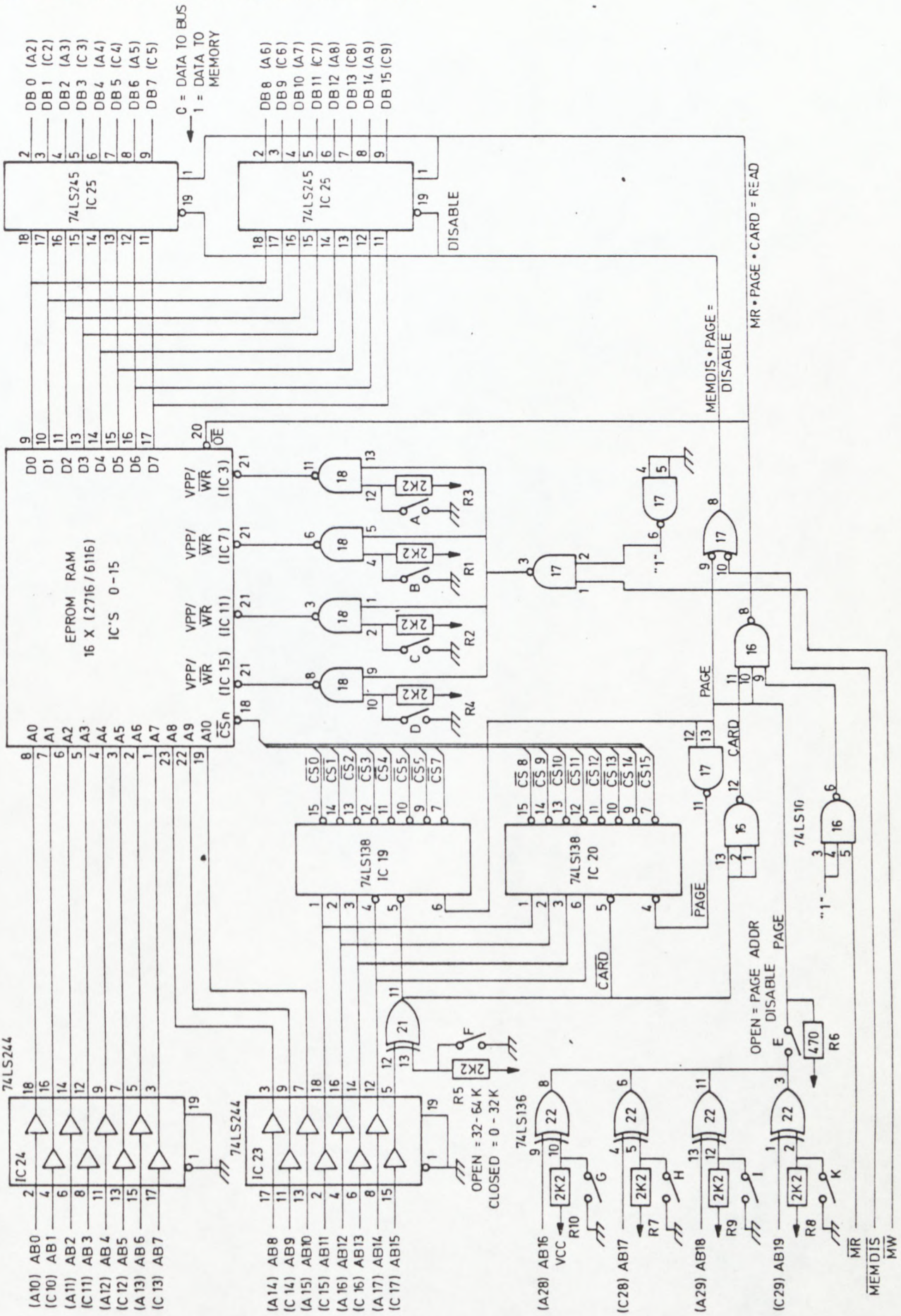
Hexadecimal chip address table

IC0	—	0 to 7FF
IC1	—	800 to FFF
IC2	—	1000 to 17FF
IC3	—	1800 to 1FFF
IC4	—	2000 to 27FF
IC5	—	2800 to 2FFF
IC6	—	3000 to 37FF
IC7	—	3800 to 38FF
IC8	—	4000 to 47FF
IC9	—	4800 to 4FFF
IC10	—	5000 to 57FF
IC11	—	5800 to 5FFF
IC12	—	6000 to 67FF
IC13	—	6800 to 6FFF
IC14	—	7000 to 77FF
IC15	—	7700 to 7FFF

Component list

IC 0 to 15	Memory devices
EPROM	Intel 2716, TI 2516
RAM	Hitachi 6116, Toshiba 5516, OKI 2128
IC 16	74LS10
IC 17, 18	74LS00
IC 19, 20	74LS138
IC 21	74LS86
IC 22	74LS136
IC 23, 24	74LS244
IC 25, 26	74LS245
R1 to R5	2,2 kΩ All resistors are 0,25 W 470 Ω
R7 to R10	2,2 kΩ
All decoupling capacitors 22 µF 16 V	
Switches	
A to D	4 DIL make/break
E,F	2 DIL make/break
G to K	4 DIL make/break

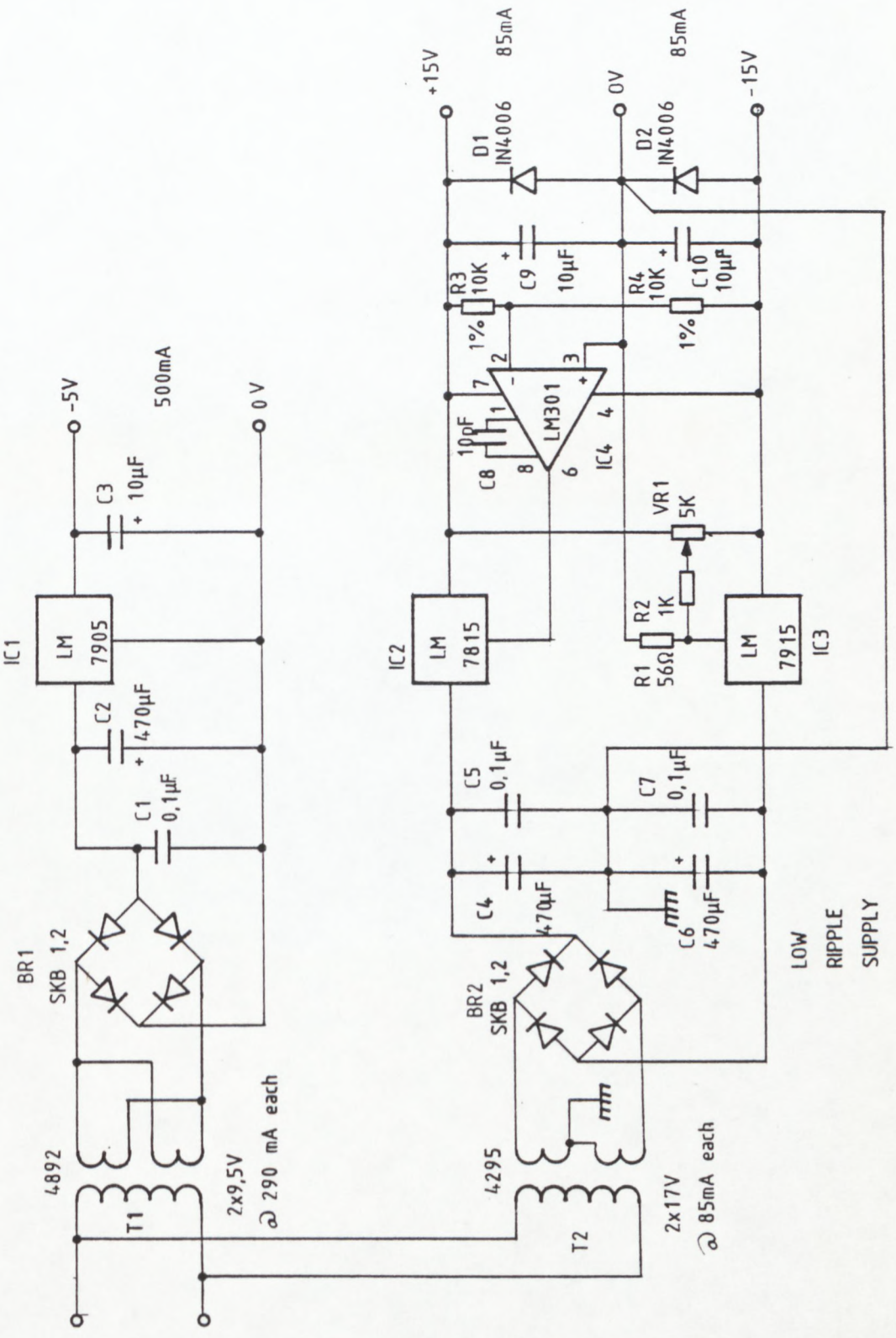
SABUS 32 k Static Memory Module-140/3



MICROCOM S-64

KJS 4
32K RAM-ROM 140-3 COMP SIDE

NEXT ASSEMBLY
VOLGENDE SAMESTEL



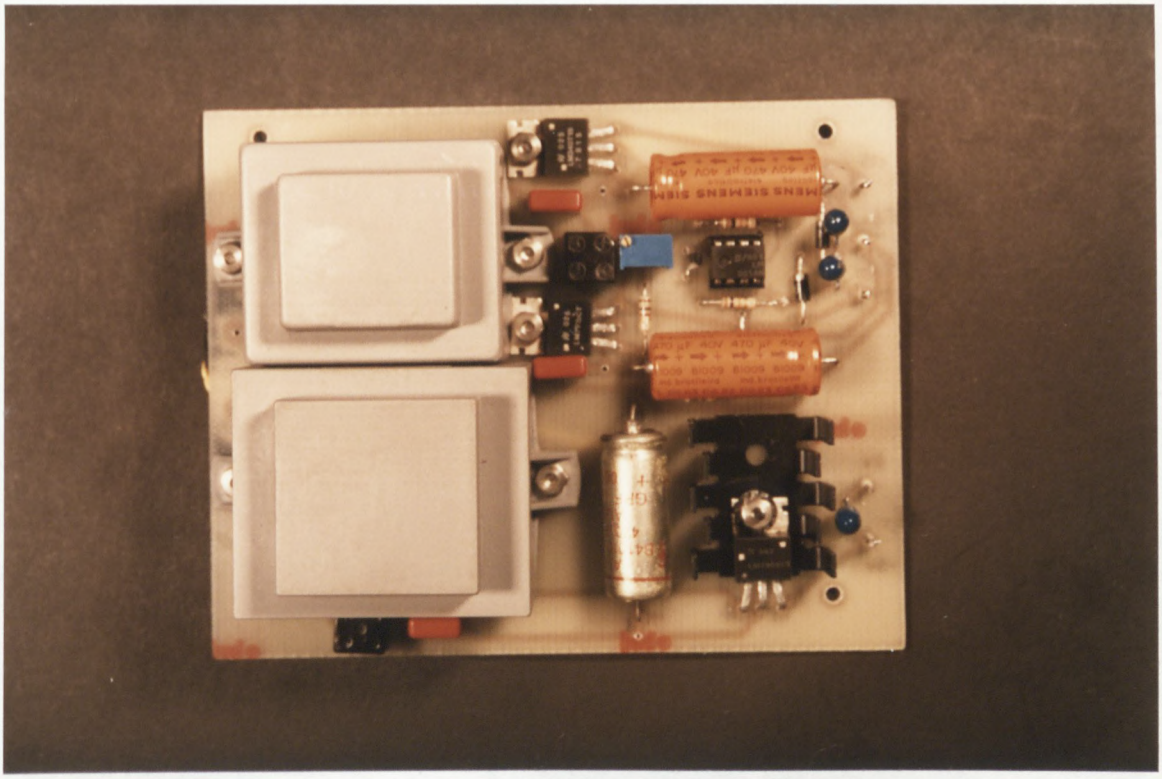
DRAWN GETEKEN L. N. 3/86	MAT'L	IMT
CHECKED NAGESIEN	TITLE TITEL DATA LOGGER P.S.U. BOARD	
APPROVED GOEDGEKEUR		SHEET OF
SCALE SKAAL	DWG TEK No IMD-D034-0020-08	REVS HERS

PHOTRA/IMPALA

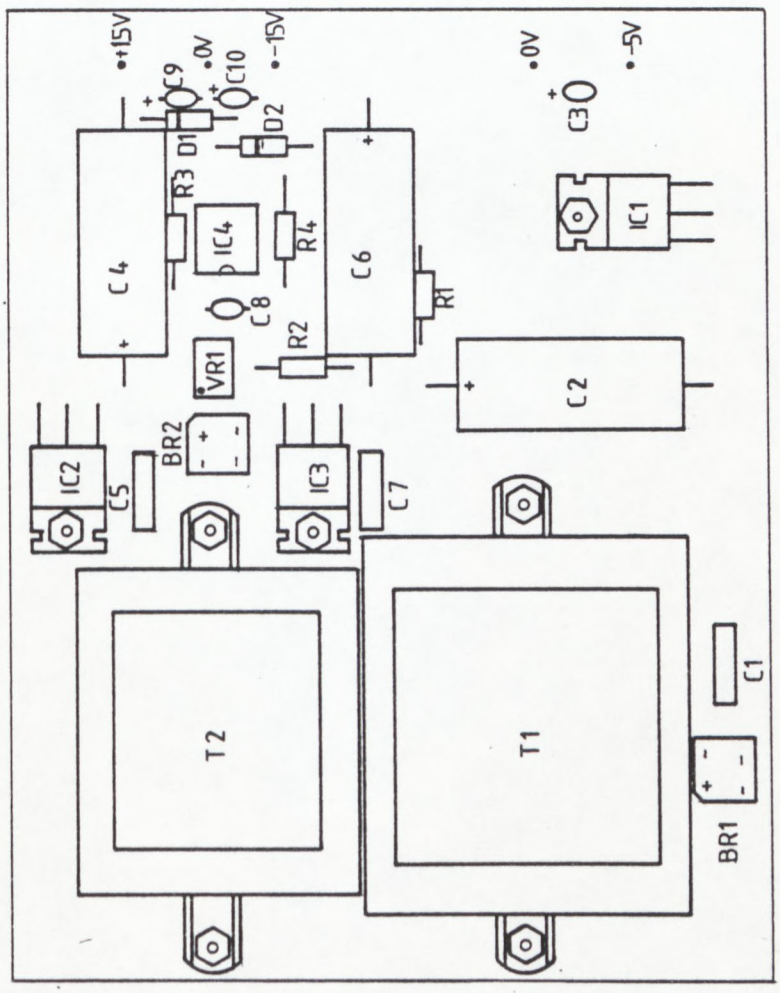
Data Logger P.S.U.

Parts List

R1	56Ω	$\frac{1}{4}$ Watt 5%
R2	1k	$\frac{1}{4}$ Watt 5%
R3,4	10k	$\frac{1}{4}$ Watt 1%
VR1	5k	20 Turn
C1,5,7	0,1µF	Polycarbonate
C8	10µF	Ceramic
C2,4,6	470µF	25V Electrolytic
C3,8,10	10µF	35V Tantalum
D1,2	IN4006	
BR1,2	1A	
T1	2x9,5V @ 290 mA	secondary, 220V primary. Type 4892.
T2	2x17V @ 85 mA	secondary, 220V primary. Type 4295.
IC1	LM 301	
IC2	LM 7915 CT	
IC3	LM 340T15 or LM 7815 CT	
IC4	LM 7905 CT.	



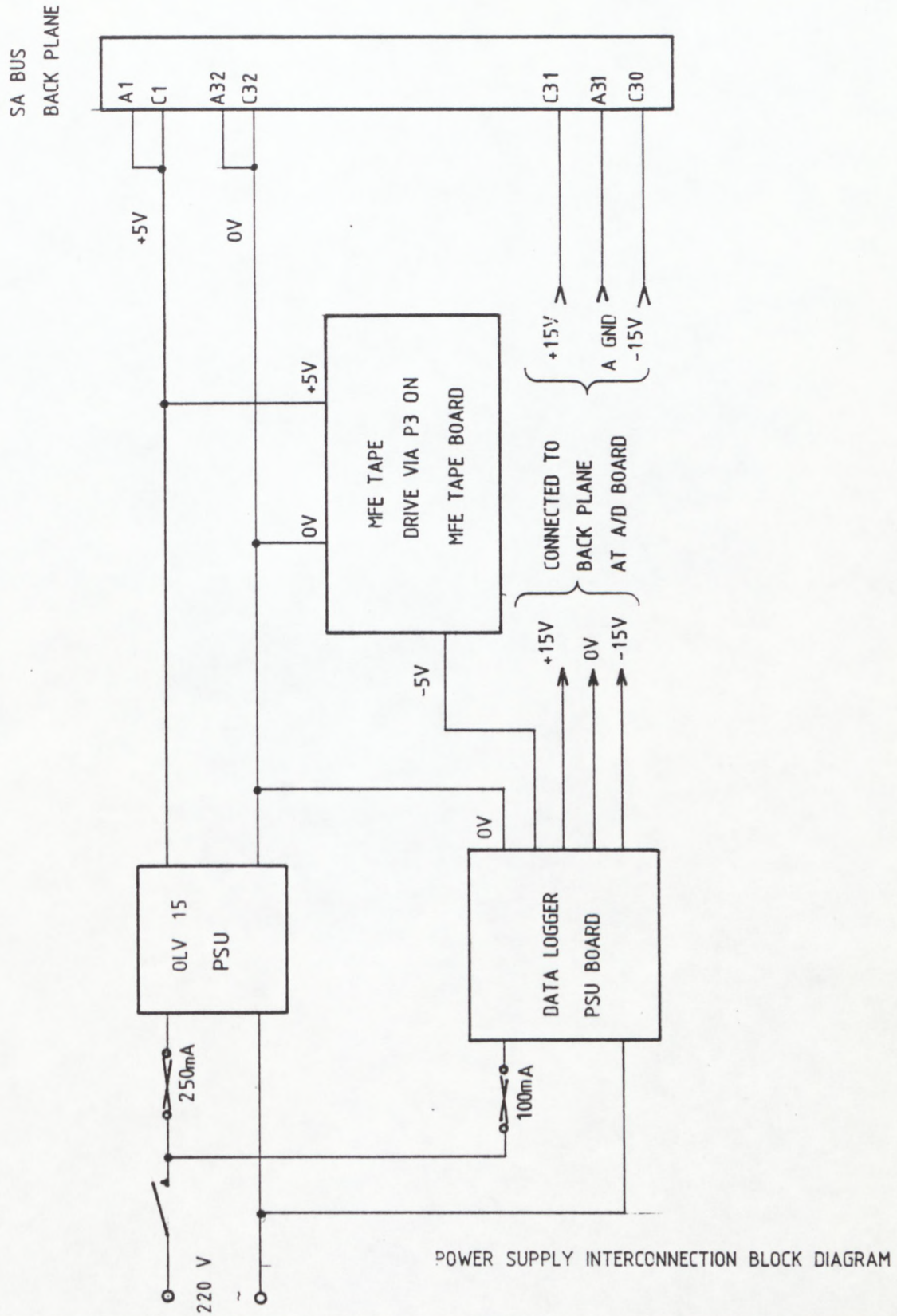
NEXT ASSEMBLY
VOLGENDE SAMESTEL



DRAWN GETEKEN L. N. 3/86	MAT'L	IMT
CHECKED NAGESIEN	TITLE TITEL DATA LOGGER P.S.U. OVERLAY	
APPROVED GOEDGEKEUR		SHEET OF
SCALE SKAAL	DWG TEK No IMD-D034-0016-08	REVS HERS

PHOTRA/IMPALA

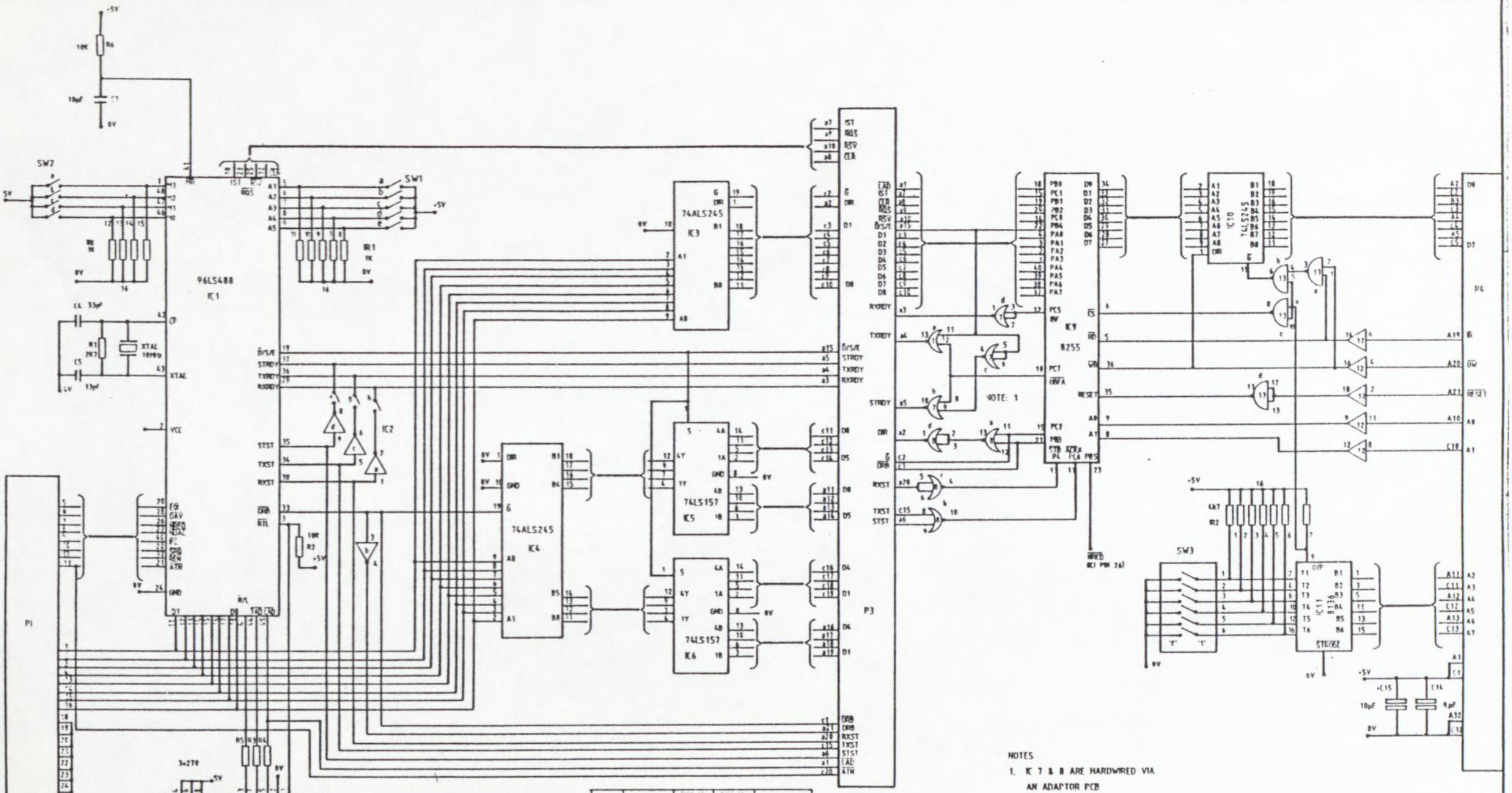
NEXT ASSEMBLY
VOLGENDE SAMESTEL



POWER SUPPLY INTERCONNECTION BLOCK DIAGRAM

DRAWN GETEKEN HR 2.85	MAT'L	IMT
CHECKED NAGESIEN	TITLE TITEL POWER SUPPLY INTERCONNECTION BLOCK DIAGRAM	
APPROVED GOEDGEKEUR		SHEET OF
SCALE SKAAL	DWG TEK No IMD-D034-0014-08	REVS HERS

PHOTRA/IMPALA



POWER & DECOUPLING TABLE

IC No	TYPE	PIN No. +5V	PIN No. 0V	CAP 0,01µF
1	96LS488	2	24	C1
2	74LS04	14	7	C3
3	74ALS245	20	10	C6
4	74ALS245	20	10	C2
5	74LS157	16	8	-
6	74LS157	16	8	C8
7	74LS02	14	7	C9a
8	74LS02	14	7	C9b
9	8255	26	7	C10
10	74LS245	20	10	C11
11	DM8136	16	8	C12
12	74LS241	20	10	C13
13	74LS00	14	7	C14&C15

ITEM	BESKRYWING DESCRIPTION	MAT'L	DEEL No PART	HVL QTY	OPMERKINGS REMARKS
SCALE	SKAAL	TITLE	TITEL		
GETEKEN	M. G. DRAWN	IEEE 488 INTERFACE BOARD			
NAGETREK	L. N TRACED				
NAGESIEN	M. G. CHECKED				
GOEDGEKEUR	APPROVED				
TEK Nr	DWG Nr	IMD-0034-0022-08			SHEET OF
HERS	REVS				

INSTITUUT VIR MARITIEME TEGNOLOGIE
INSTITUTE FOR MARITIME TECHNOLOGY

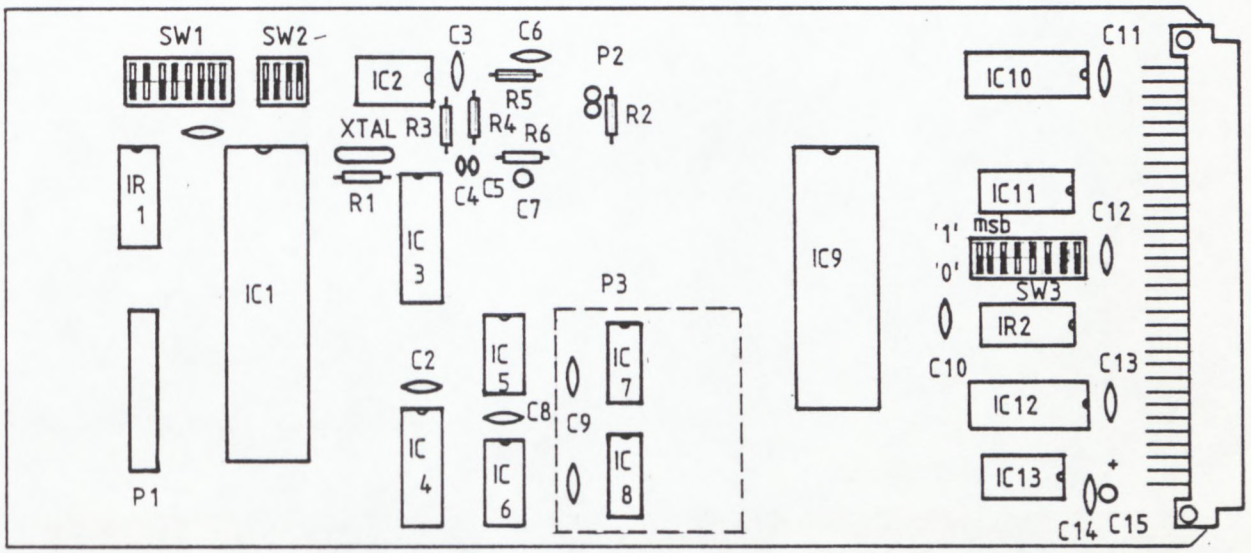
PHOTRA/IMPALA

IEEE 488 Interface Board

Parts List

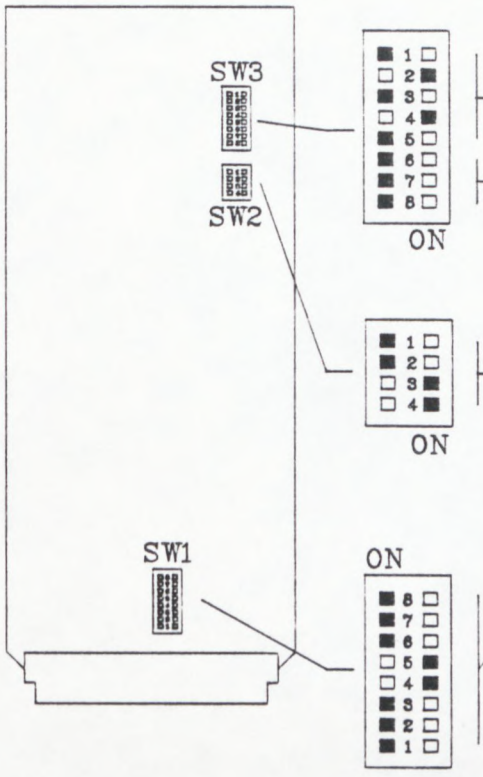
R1	2k7	$\frac{1}{4}$ Watt 5%
R2,6	10k	$\frac{1}{4}$ Watt 5%
R3-5	270k	$\frac{1}{4}$ Watt 5%
C1-3	0,01µF polycarbonate	
C4,5	33pF ceramic	
C6	0,01µF polycarbonate	
C7,15	10µF 35V Tantolum	
C8-14	0,01µF polycarbonate	
P1	2x13 Molex 0,1" Wafer	
P2	-	
P3	-	
P4	DIN 41612 64 Pole connector (A+C, 90°)	
1R1	15x1k Type 1615	
1R2	15x4k7 Type 1615	
XTAL	10,000 MHz	
SW1,3	8 Pole D.I.L. switch	
SW2	4 Pole D.I.L. switch	
1C1	96LS488	
1C2	74LS04	
1C3,4,10	74LS245	
1C5,6	74LS157	
1C7,8	74LS02	
1C9	8255	
1C11	DM8136	
1C12	74LS241	
1C13	74LS00	

NEXT ASSEMBLY
VOLGENDE SAMESTEL



DRAWN GETEKEN L. N. 3/86	MAT'L	IMT
CHECKED NAGESIEN	TITLE TITEL IEEE 488 INTERFACE OVERLAY	
APPROVED GOEDGEKEUR		SHEET OF
SCALE SKAAL	DWG TEK No IMD-D034-0017-08	REVS HERS

PHOTRA/IMPALA



IEEE ADDRESS

5	4	3	2	1
A5	A4	A3	A2	A1
0	1	0	1	0

SET FOR ADDR. 10
ON = '1'

AUTO HANDSHAKE

6	7	8
STRDY	TXRDY	RXRDY

IEEE MODE SELECT

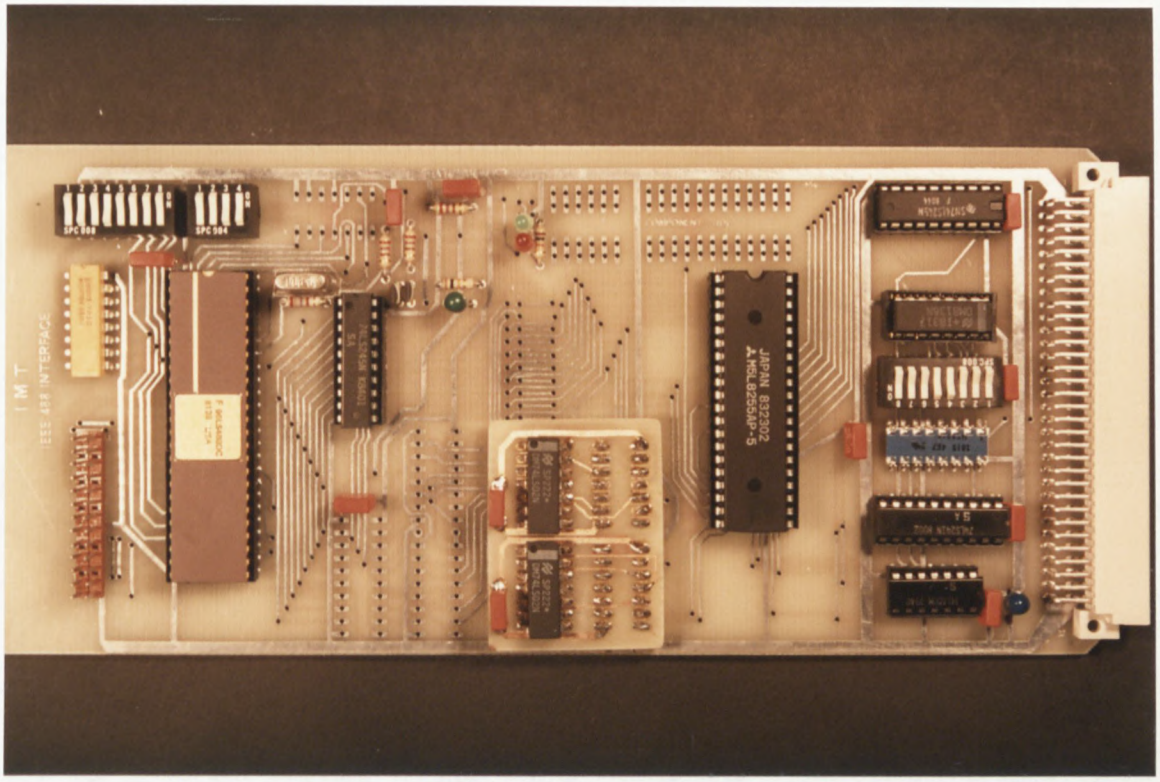
1	2	3	4
M3	M2	M1	M0
0	0	1	1

SET FOR
TALKER/LISTNER MODE
ON = '1'

SA BUS ADDRESS

8	7	6	5	4	3	2	1
NC	NC	A7	A6	A5	A4	A3	A2
X	X	0	1	1	0	0	0

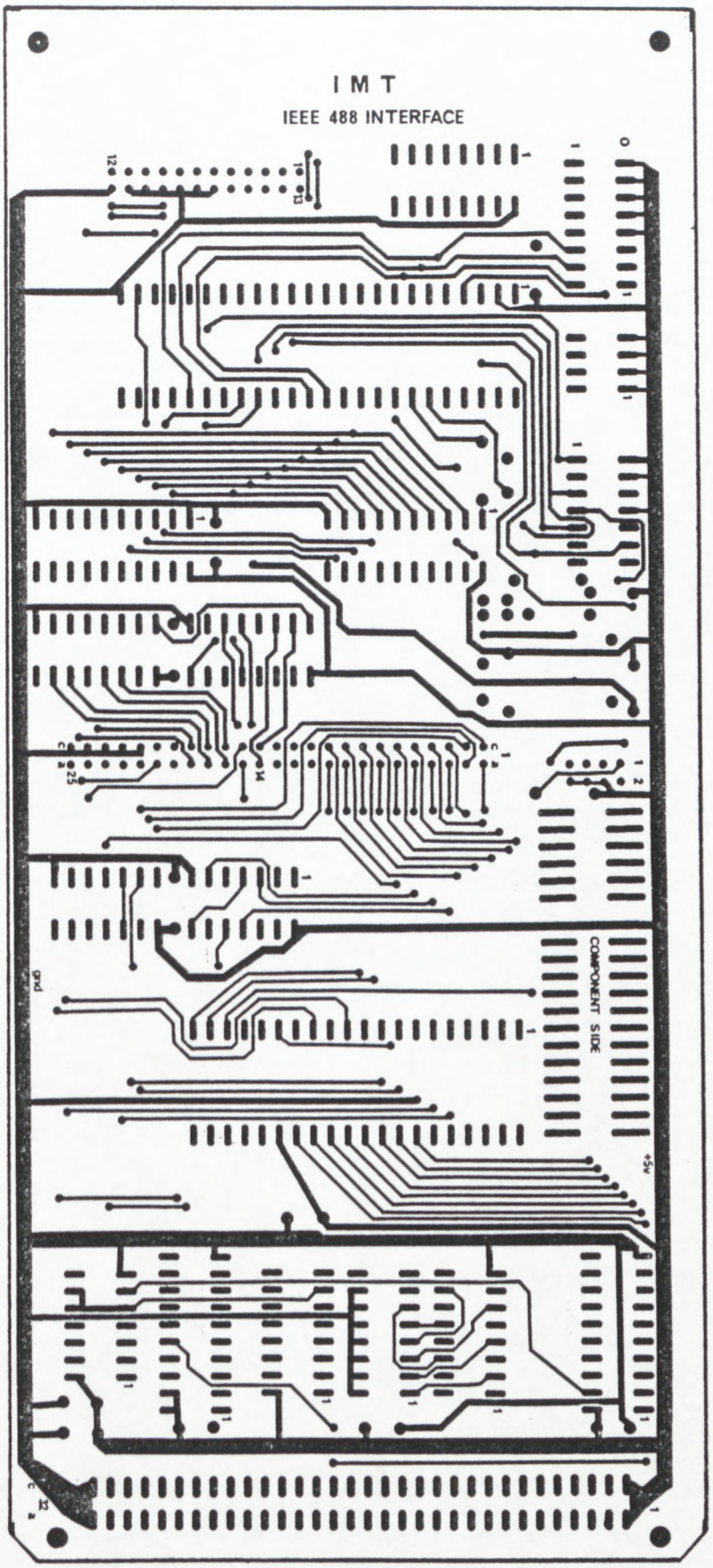
SET FOR 60H
ON = '0'

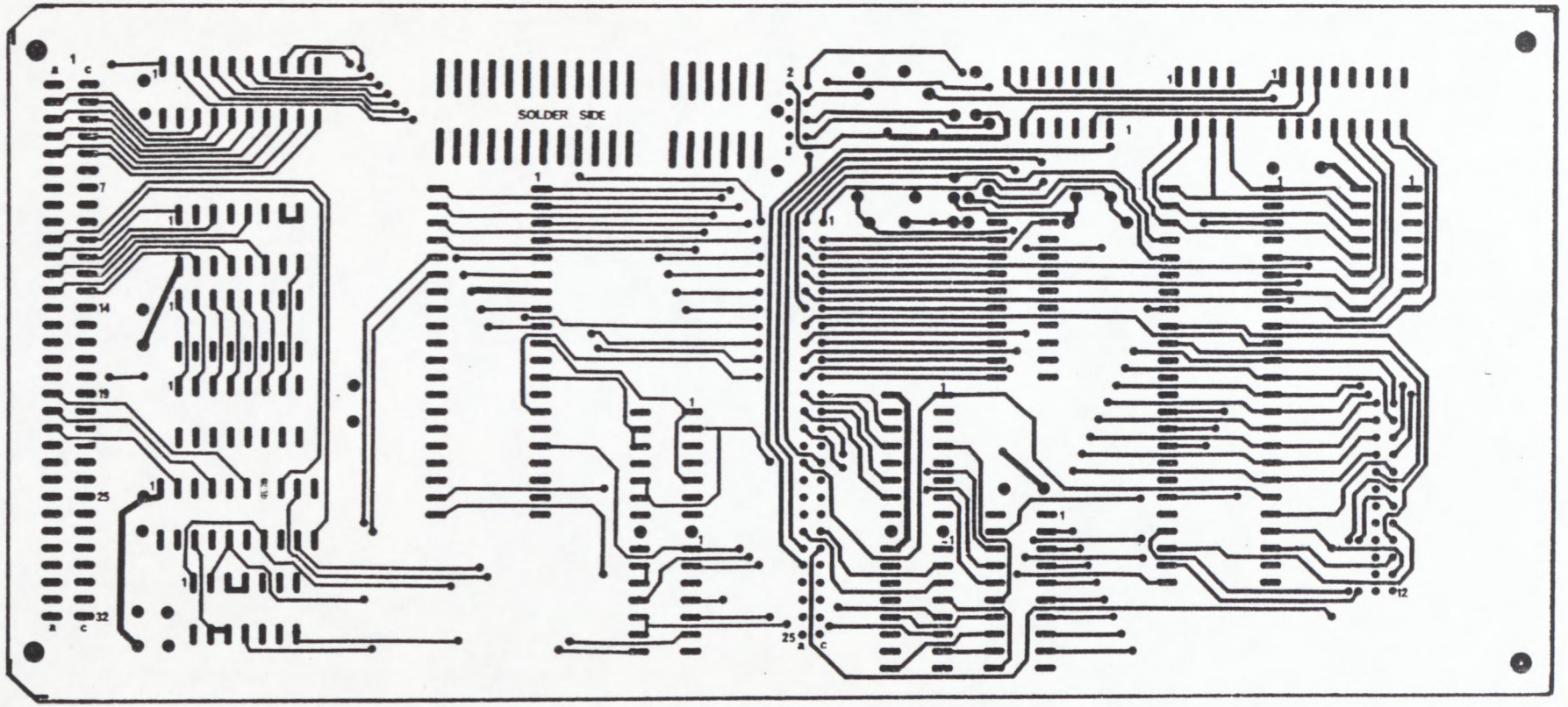


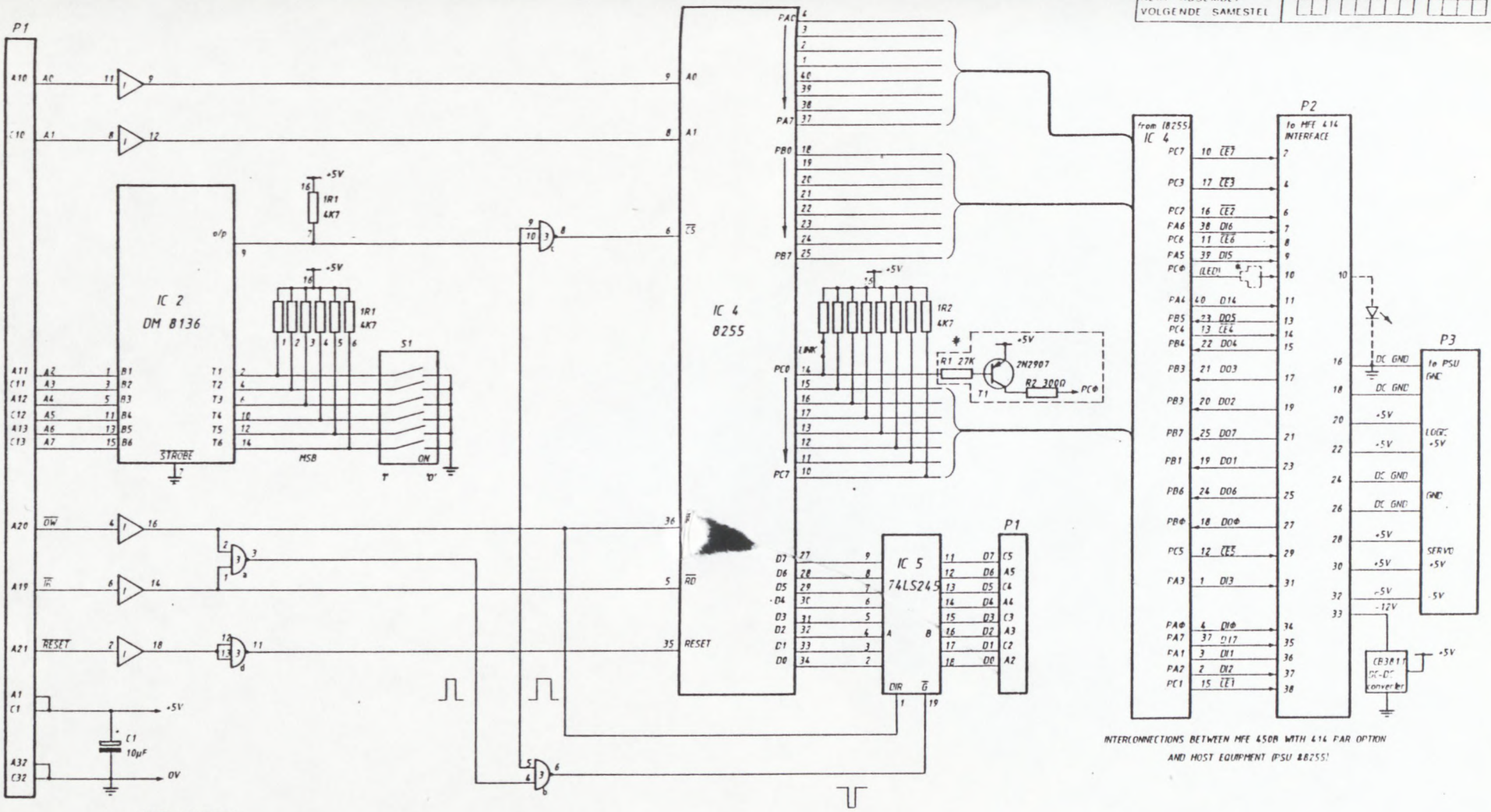
TITLE:
IEEE 488
SWITCH SETTINGS

IMT

DRG. NO.:	
FILE NAME: D 034	SCALE:
DRN. FOR:	SHEET 1 OF 1
DRN. BY: M GARDENER	DATE 08-10-86


I M T
IEEE 488 INTERFACE
COMPONENT SIDE
+5v
gnd


SOLDER SIDE



POWER & DECOUPLING TABLE

IC No	TYPE	PIN No +5V	PIN No 0V	CAP C 0,1µF
1	74LS241	19,20	1,10	C2
2	DM8136	16	8	C3
3	74LS00	14	7	C4
4	8255	26	7	C5
5	74LS245	20	10	C6

ITEM	BESKRYWING DESCRIPTION	MAT'L	DEEL No PART	HVL QTY	OPMERKINGS REMARKS
SCALE	SKAAL	TITLE		TITEL	IMT
GETEKEN M GARDENER JUNE 84	DRAWN	MFE TAPE INTERFACE BOARD			
NAGETREK HR OCT 84	TRACED				SHEET 1 OF 1
NAGESIEN	CHECKED				
GOEDGEKEUR	APPROVED	TEK Nr 16 0034 0003		DWG Nr	HERS REVS

INSTITUUT VIR MARITIEME TEGNOLOGIE
INSTITUTE FOR MARITIME TECHNOLOGY

MFE Tape Interface Board

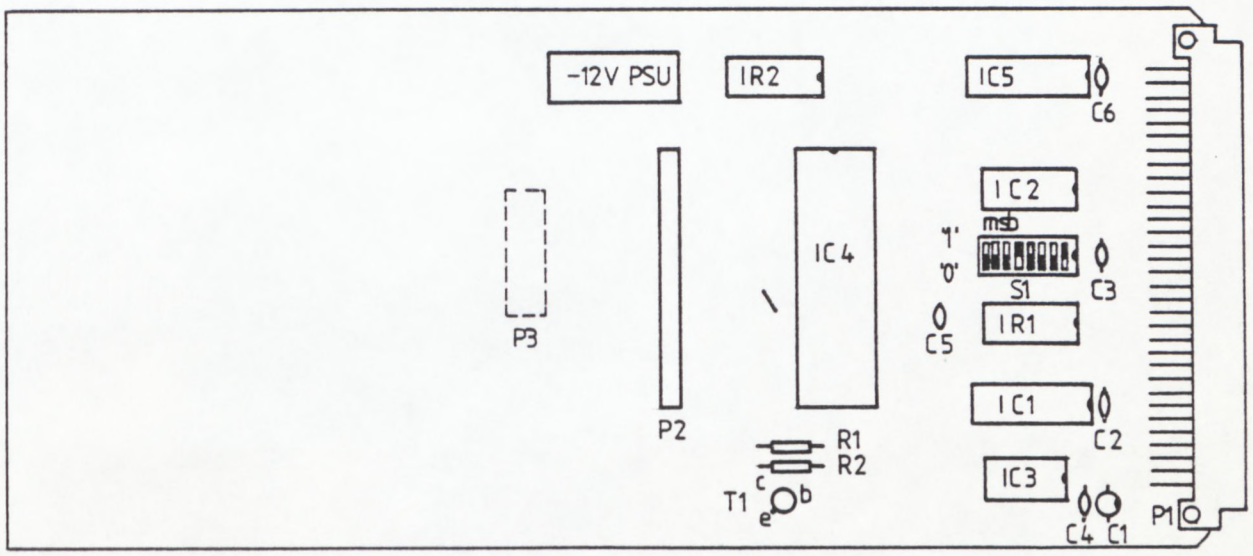
Parts List

R1	27k	$\frac{1}{4}$ Watt 5%
R2	300Ω	$\frac{1}{4}$ Watt 5%
1R1, 2	15 x 4k7 Type 1615	
C1	10µF 16V Tantalum	
C2-6	0,01µF Ceramic	
T1	2N2907	
P1	DIN 41612 64 Pole connector (A+C, 80°)	
P2	2x20 Molex 0,1" Wafer	
P3	10 Pole Stocko Pin Housing. Horizontal mounting.	
S1	8 Pole D.I.L. switch	

DC to DC convertor (5V to -12V) CB 3811

1C1	74LS241
1C2	DM 8136
1C3	74LS00
1C4	8255
1C5	74LS245

NEXT ASSEMBLY
VOLGENDE SAMESTEL



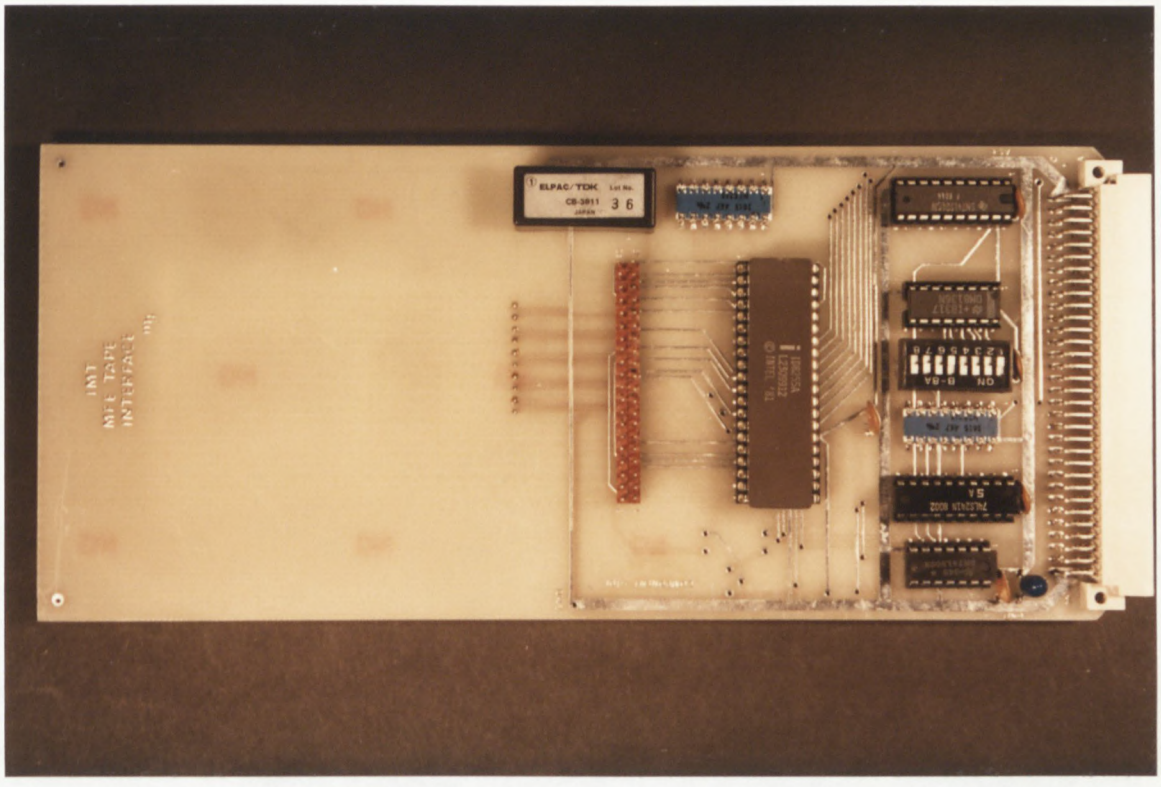
DRAWN GETEKEN L. N. 3/86	MAT'L	IMT
CHECKED NAGESIEN	TITLE TITEL MFE TAPE INTERFACE OVERLAY	
APPROVED GOEDGEKEUR		SHEET OF
SCALE SKAAL	DWG TEK No IMD-D034-0015-08	REVS HERS

PHOTRA/IMPALA

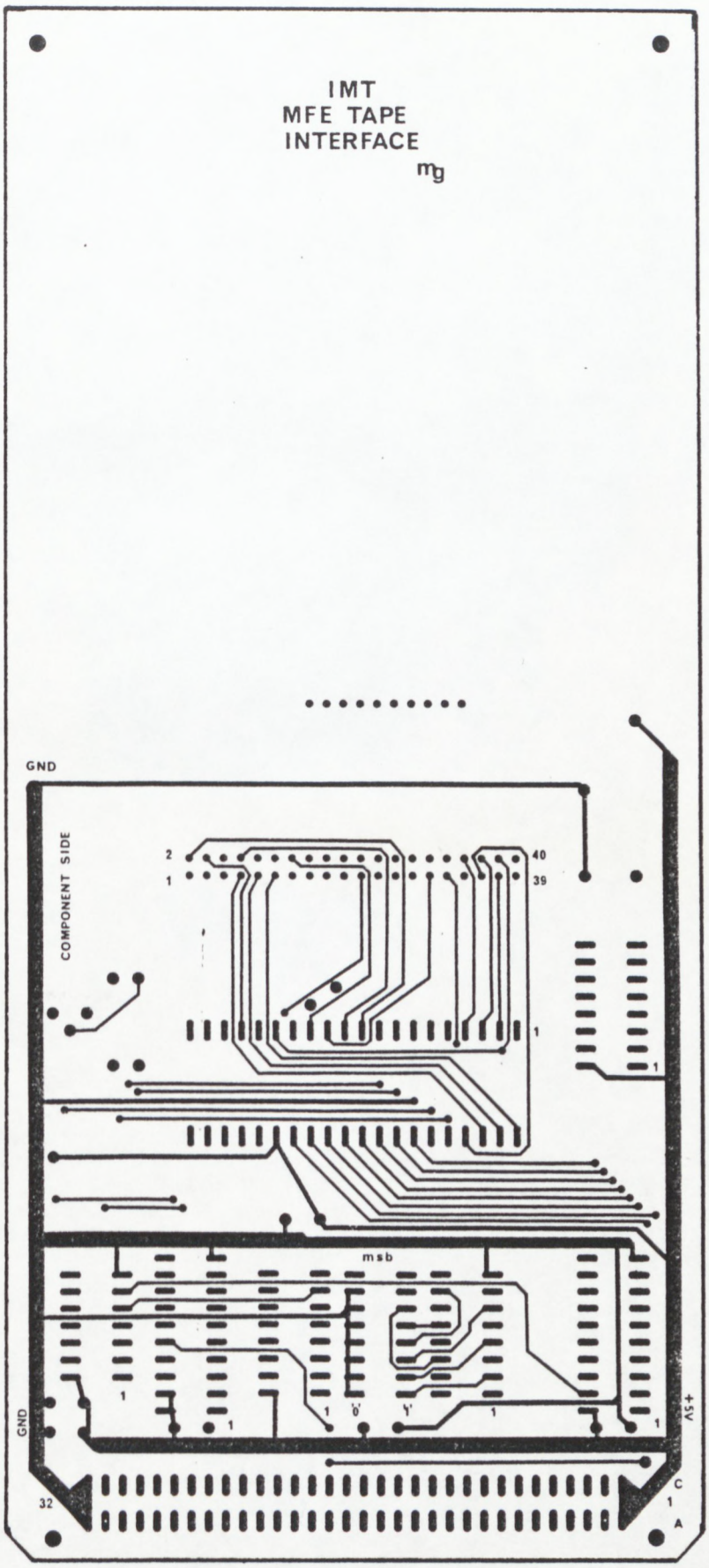
SA BUS ADDRESS

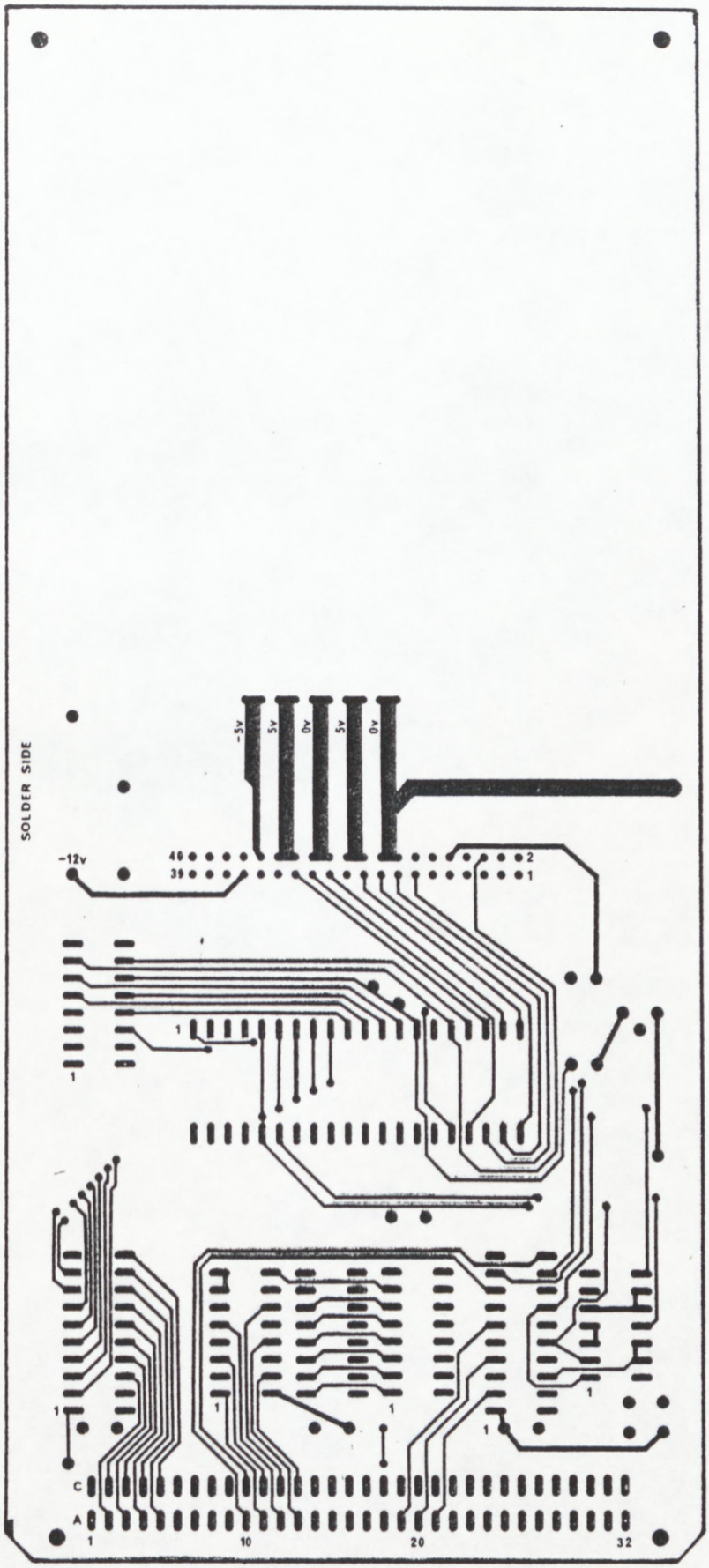
8	7	6	5	4	3	2	1
NC	NC	A7	A6	A5	A4	A3	A2
X	X	0	1	0	0	0	1

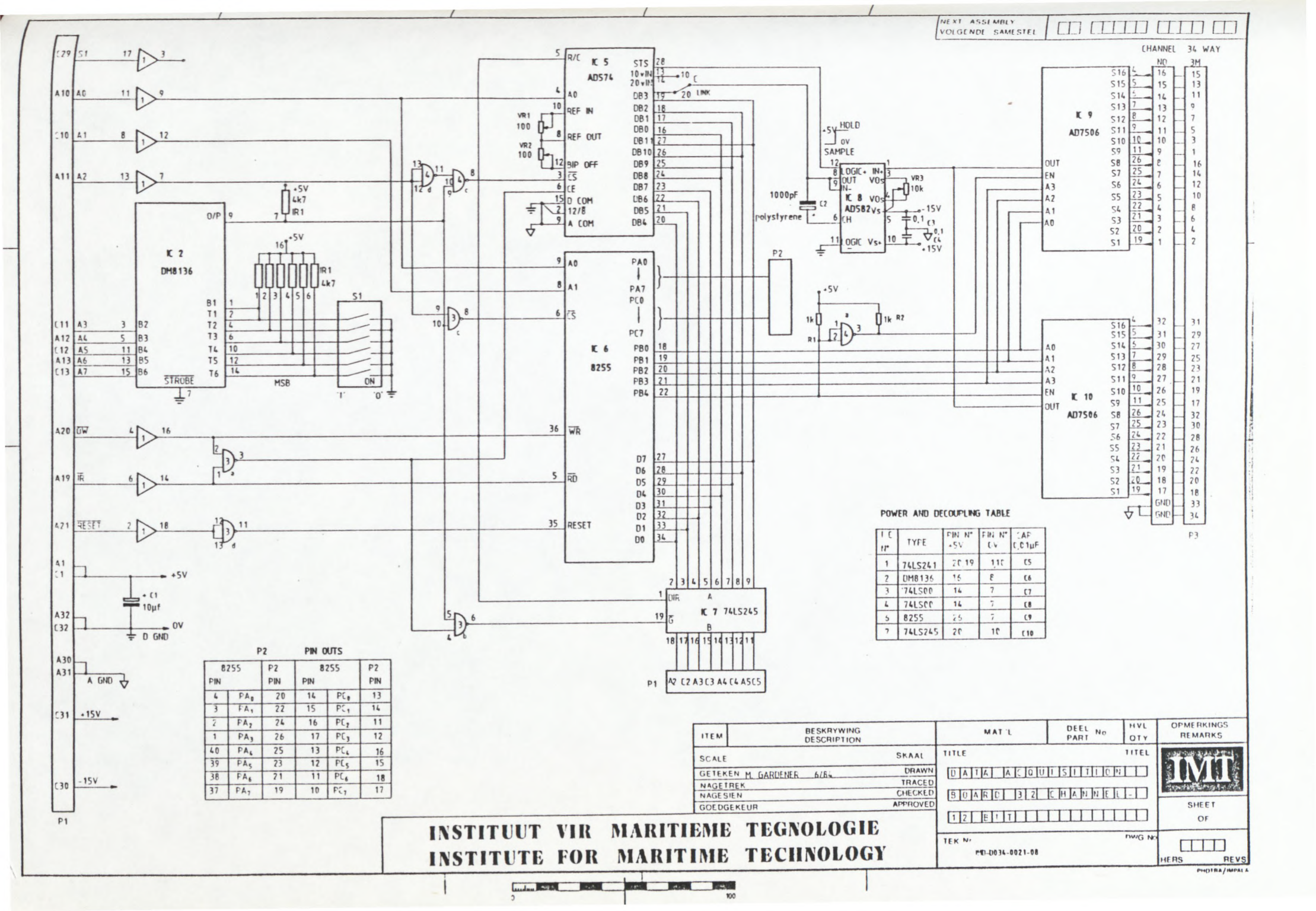
SET FOR 44H
ON = '0'



TITLE: MFE TAPE INTERFACE SWITCH SETTINGS	IMT	DRG. NO.:	
		FILE NAME: D 034	SCALE:
		DRN. FOR:	SHEET 1 OF 1
		DRN. BY: M GARDENER	DATE 08-10-86


IMT
MFE TAPE
INTERFACE
mg
GND
COMPONENT SIDE
2
1
40
39
1
1
msb
GND
1
1
1
'0'
'1'
1
1
+5V
32
C
1
A


SOLDER SIDE
-5v
5v
0v
5v
0v
-12v
40
39
2
1
1
1
1
1
1
1
1
C
A
1
10
20
32



P2 PIN OUTS

8255 PIN		P2 PIN	8255 PIN		P2 PIN
4	PA_0	20	14	PC_0	13
3	PA_1	22	15	PC_1	14
2	PA_2	24	16	PC_2	11
1	PA_3	26	17	PC_3	12
40	PA_4	25	13	PC_4	16
39	PA_5	23	12	PC_5	15
38	PA_6	21	11	PC_6	18
37	PA_7	19	10	PC_7	17

POWER AND DECOUPLING TABLE

IC No	TYPE	PIN N° +5V	PIN N° 0V	CAP 0,01µF
1	74LS241	20	10	C5
2	DM8136	16	8	C6
3	74LS00	14	7	C7
4	74LS00	14	7	C8
5	8255	26	7	C9
7	74LS245	20	10	C10

ITEM	BESKRYWING DESCRIPTION	MAT'L	DEEL No PART	HVL QTY	OPMERKINGS REMARKS
SCALE	SKAAL	TITLE		TITEL	IMT
GETEKEN M. GARDENER 6/86	DRAWN	DATA ACQUISITION BOARD 32 CHANNEL			
NAGETREK	TRACED				SHEET
NAGESIEN	CHECKED				OF
GOEDGEKEUR	APPROVED	TEK N° PMI-0034-0021-08	DWG N°		HERS REVS

INSTITUUT VIR MARITIEME TEGNOLOGIE
INSTITUTE FOR MARITIME TECHNOLOGY

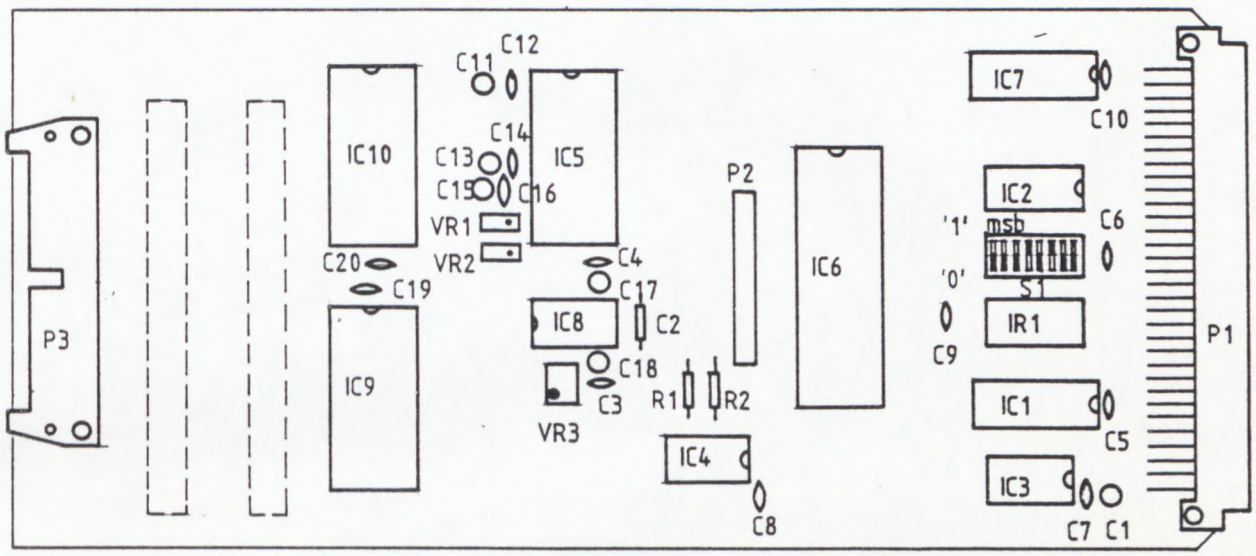
PHOTRA/IMPALA

32 Channel Data Acquisition Board

Parts List

R1,2	1k	$\frac{1}{4}$ Watt 5%
VR1,2	100Ω	20 turn
VR3	10k	20 turn
1R1	15x4k7	Type 1615
C1	10µF	Tantalum
C2	1000pF	Polystyrene
C3,4	0,1µF	Polycarbonate
C5-10	0,01µF	Ceramic
P1	DIN 41612 (A+C, 90°)	64 Pole connector
P2	Molex 0,1" Wafer	
P3	34 way, 3m, Ribbon cable connector	
S1	8 Pole D.I.L. switch	
1C1	74LS241	
1C2	DM1836	
1C3,4	74LS00	
1C5	AD574	
1C6	8255	
1C7	74LS245	
1C8	AD582	
1C9,10	AD5760 or 1H6116	

NEXT ASSEMBLY
VOLGENDE SAMESTEL



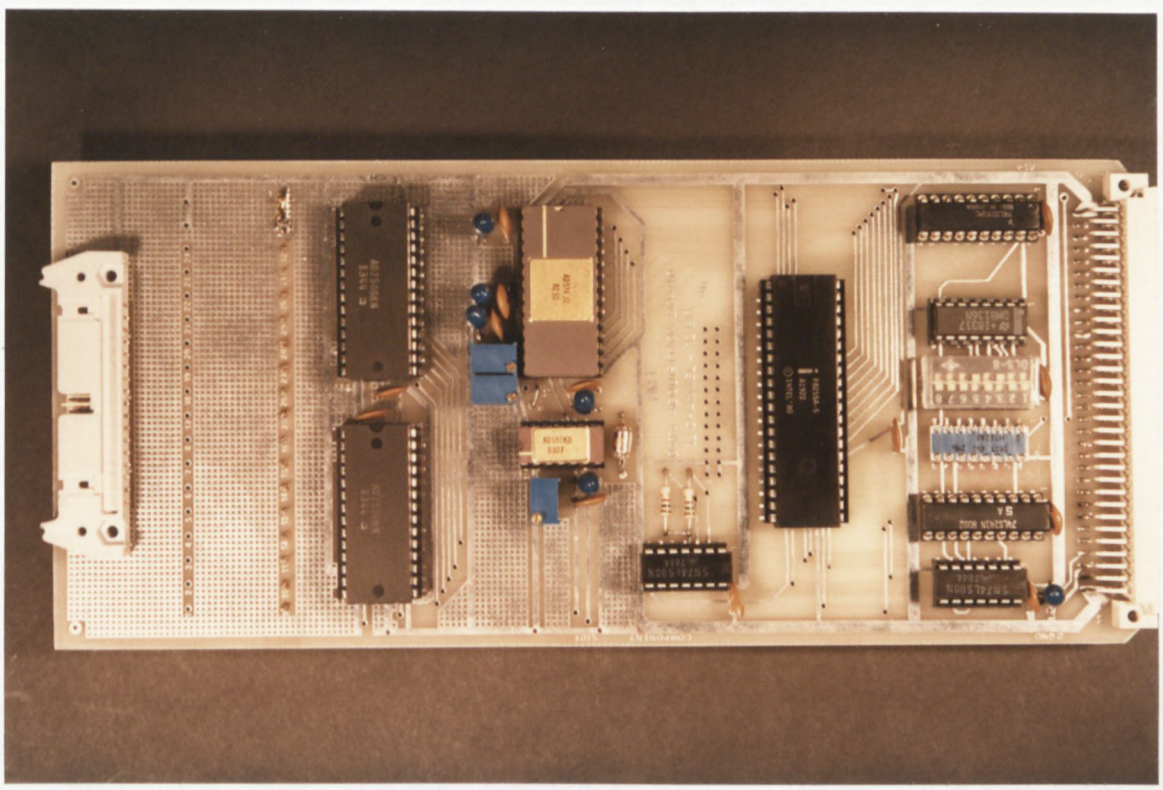
DRAWN GETEKEN L. N. 3/86	MAT'L	IMT
CHECKED NAGESIEN	TITLE TITEL DATA ACQUISITION BOARD	
APPROVED GOEDGEKEUR		SHEET OF
SCALE SKAAL	DWG TEK No IMD-D034-0019-08	REVS HERS

PHOTRA/IMPALA

SA BUS ADDRESS

8	7	6	5	4	3	2	1
NC	NC	A7	A6	A5	A4	A3	A2
X	X	0	1	0	1	0	0

SET FOR 50H
ON = '0'

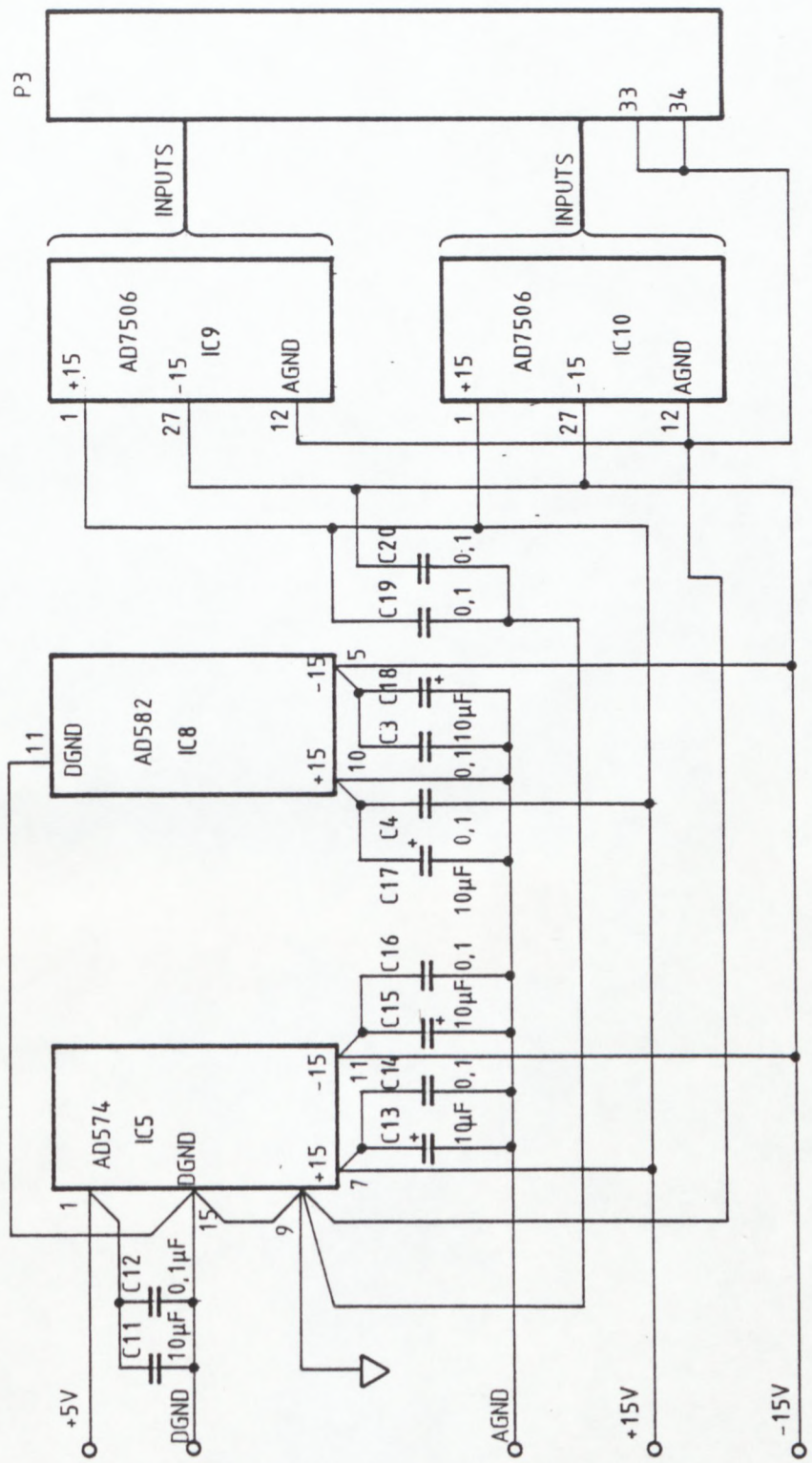


TITLE:
DATA ACQUISITION BD.
SWITCH SETTINGS

IMT

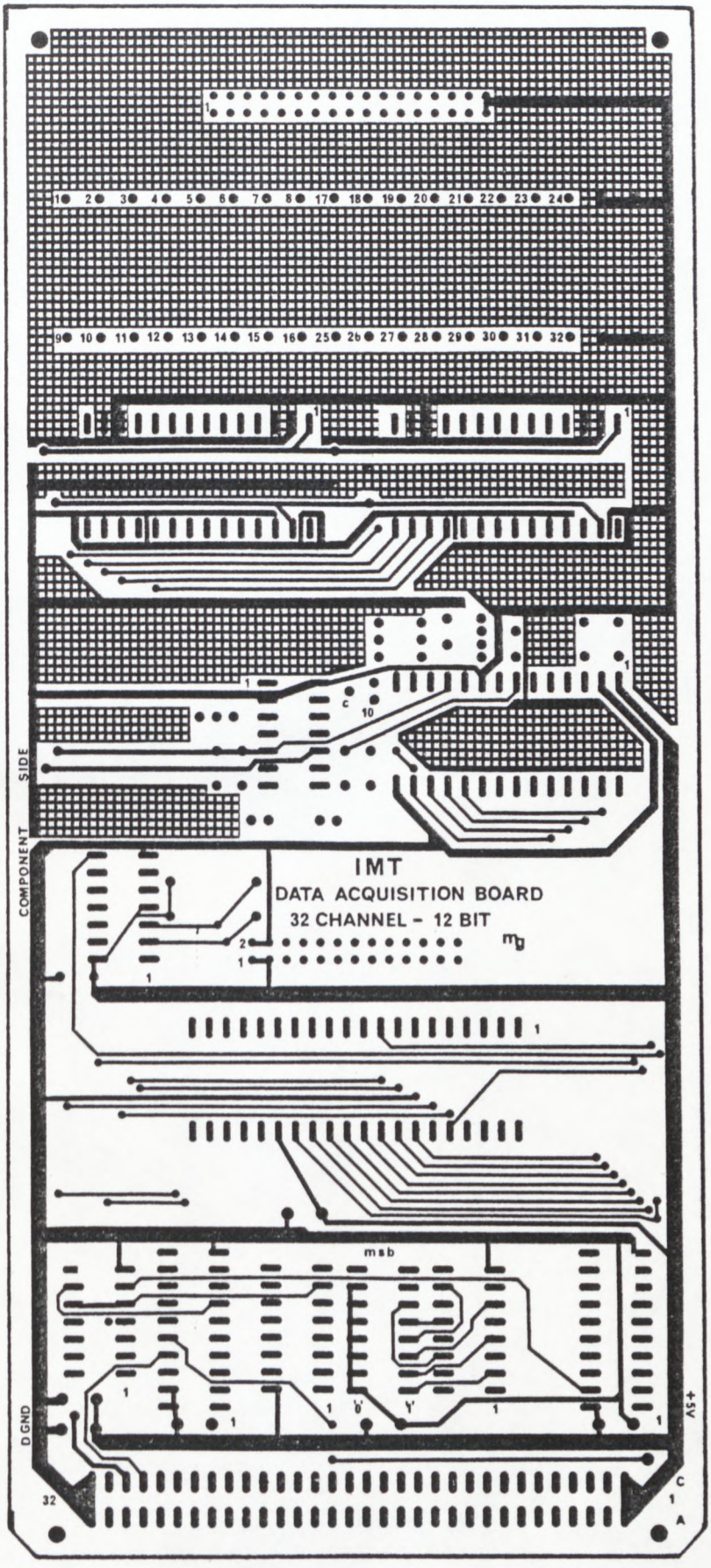
DRG. NO.:	
FILE NAME: D 034	SCALE:
DRN. FOR:	SHEET 1 OF 1
DRN. BY: M GARDENER	DATE 08-10-86

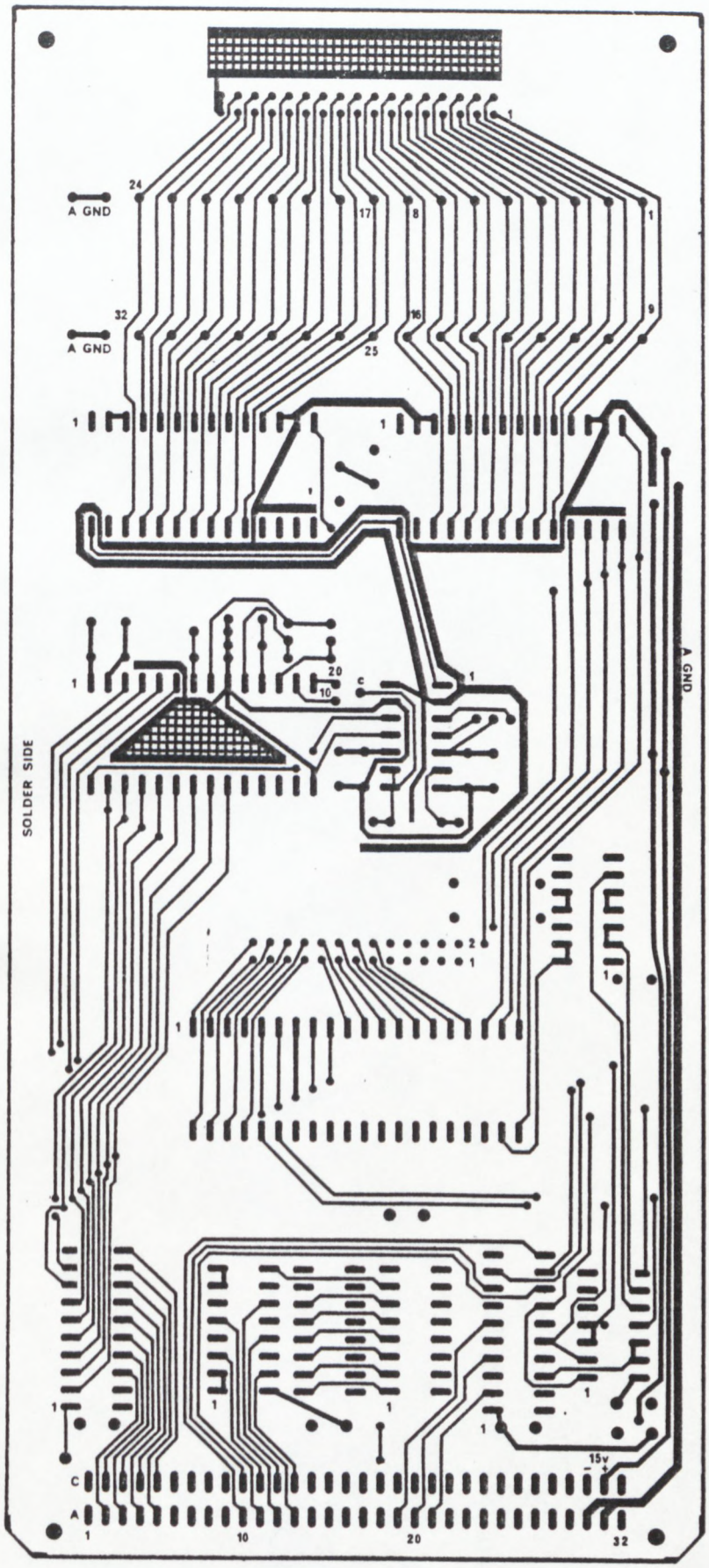
NEXT ASSEMBLY
VOLGENDE SAMESTEL



DRAWN M. GARDENER GETEKEN 11/84	MAT L	IMT
CHECKED NAGESIEN	TITLE TITEL DATA ACQUISITION BOARD ANALOG SUPPLY & DECOUPLING CIRCUIT	
APPROVED GOEDGEKEUR		SHEET OF
SCALE SKAAL	DWG TEK No IMD-D034-0005-08	REVS HERS

PHOTRA/IMPALA


IMT
DATA ACQUISITION BOARD
32 CHANNEL - 12 BIT
COMPONENT SIDE
DGND
+5V
msb


SOLDER SIDE
A GND
A GND
A GND
15v
+
-