UNITED STATES DEPARTMENT OF INTERIOR

GEOLOGICAL SURVEY

A TELEMETRY-BASED DATA-ACQUISITION SYSTEM

PART 1:

THE REMOTE DATA-ACQUISITION TRANSMITTER

by

Gregory K. Miller

Open-File Report 89 -553A

Woods Hole, Mass.

1989

TABLE OF CONTENTS

ILLUSTRATIONS

INTRODUCTION

The design requirements for this data-acquisition system were established for a U.S. Geological Survey study of coastal erosion on a remote sand island in the Gulf of Mexico. The primary function of this system is to acquire hourly data concerning sand height, wave conditions, overwash processes, and weather conditions, and to provide immediate access to that data through a telephone link. The complete system consists of a battery-powered, computer-based, remotely sited data-acquisition package, that has the equipment to transmit the acquired data to a second computer system via a VHF (very high frequency) radio link. This second computer, sited on the mainland, records the data in permanent storage and is connected to a phone line to allow off-site access to the recorded data.

The basic design was developed in 1986 utilizing off-the-shelf hardware for as much of the system as possible to minimize development time. The data-acquisition receiver is described in Gregory K. Miller, 1989, A Telemetry-Based Data-Acquisition System Part 2: The Data-Acquisition Receiver, USGS Open File Report 89- . This report describes the remote data-acquisition transmitter.

The battery-powered data-acquisition system includes: a controller, a real-time clock, an analog-to-digital converter, and a solid-state memory for mass storage of data. Several additional circuit boards limit power to the sensors when data are not being recorded, and a final circuit acts as a flood switch to determine if the island is being overwashed. All are contained in a weather-resistant housing. This system records the analog voltage from up to sixteen channels as a time series of 34 minutes (4096 times at twice a second), six times a day. Three channels of weather data are also recorded for one minute (128 times at twice a second) just prior to the time series. Several channels are recorded for 17 minutes (2048 times at twice a second) for each hour in which the main data set is not being recorded. When the island is being overwashed, all channels are recorded for 34 minutes every hour. This information is then either transmitted via radio link to the land-based computer system that records the data in permanent storage, or it is recorded on-site. On-site storage consists of either a solid-state memory device, which is configured to emulate a 1- or 4-megabyte floppy-disk, or a digital tape recorder with a 20-megabyte capacity. A block diagram of this system is shown in Figure 1; the system specifications are shown in Appendix I.

The software controlling the data-acquisition resides in read-only-memory (ROM) and is structured to provide a flexible operating environment useful in a wide variety of studies. All data-acquisition is initiated by the time-of-day, which is set by the user at startup. To accommodate the sampling scheme required for the program, the software provides the option of two time series for each data acquisition and a further option of collecting additional data between the main data. The program can also be set to increase the frequency of data collection during some external event such as an overwash.

SECTION 1. HARDWARE DESCRIPTION

1.1 DATA-ACQUISITION COMPUTER

The data-acquisition computer is comprised of several circuit boards manufactured by Citadel Computer Corp., Amherst, New Hampshire. These boards include a power control board, a central processing unit (CPU) board, an extended memory board , a real-time-clock board, an analog-to-digital board, and the interface board for the solid-state memory. Figures 5,6,7, and 8 illustrate the jumper settings used to configure each board for this application. All boards are housed in a small enclosure that contains all of the connections necessary for data-acquisition (Figure 2).

1.2 POWER REGULATION AND CONTROL

Operating the data-acquisition system in the field requires a battery-supplied source of power. All power connections to the system are designed for a +12 volt DC (direct current) battery. The power for the computer is connected to the power connector (P-3) shown in Figure 2. The data-acquisition system will operate on any input voltage ranging from +10 to +14 volts DC. For use in the field, a voltage-sensing relay has been added to the power input, which will disengage all power to the computer if the voltage drops below +10.2 volts. The relay will reconnect power to the computer when the power rises above +10.75 volts. This prevents possible damage to the electronic circuits.

Power is also connected to the interface boards used for power control, remote event detection, and analog signal conditioning. This connection is illustrated at the top of Figure 4. Note that there is also a connection for a -12 volt DC power input. This is provided for the analog signal-conditioning boards that require both +8 and -8 volts DC for operation. Each board has a jumper J1 (Figure 15) which allows the board to be powered using either +12 and -12 volts DC or only +12 volts DC. Placing a jumper on pins 2 and 3 requires only +12 volt power. The jumper connects the output of a voltage inverter U13 (Figure 14), which inverts the +12 volts to -12 volts, to the -8 volt regulator U16. Placing a jumper pins 1 and 2 requires a -12 volt input to power the -8 volt regulator. The latter connection is preferred, as it eliminates the need for the voltage inverter. This reduces power consumption and eliminates the 5 millivolts of noise generated by the inverter.

Battery voltage is used by the power control boards to provide power to external devices. External power control is optional, and is included as a means of conserving battery power by allowing the user to disconnect any device when not in use. There are three independent control circuits located on each board. Each circuit (Figure 9) contains optic isolation to protect the control signals coming from the computer, a four-contact relay configured to provide +12 volts output and two contact closures when engaged, an LED (light emitting diode) to indicate when the relay is on, a standard 3AG-type fuse to

protect against short circuits, and a switch which will bypass the relay and provide +12 volt output for testing. The configuration of the relays is such that the power to external devices can be connected directly to the relay, or the relay can provide control (i.e., the +12 volt signal or the contact closures) to some external control circuitry. Each board has all three control inputs connected connected to jumper pad J2 (Figure 10). The pins on jumper pad J1 are connected to control lines originating from the computer. By connecting each of the jumper pins of J2 to one of the pins on J1, external power control is enabled. The jumper pad arrangement allows multiple boards to be used in the system.

The software gives the user the option of controlling power to the sensors when entering the program parameters (section 2.4). If no control of sensor power is desired, there are three different methods of providing power to the sensors: (1) the first method is simply to connect the power directly to the sensors. In this case the power control boards are not needed. (2) In the second method, power to the sensors is connected to the +12 volt output of the relays (Connectors P1, P2, and P3). Pin 1 is the +12 volt output, and pin 2 is the battery ground (Figure 10). The switch (SW1, SW2, and SW3) is engaged providing continuous power to the sensor. Each fuse (F1,F2,F3) provides protection and the LED (CR1, CR2, and CR3) provides an indication that power is going to the sensors. (3) The third method involves connecting the control inputs (pins 1, 2, and 3 of J2) to a continuous logic high (+5 volts) from the computer (pin G of J1). If the computer loses power, the relays will disengage, removing power to all of the sensors. When power is restored to the computer, sensor power is also restored. If control of sensor power is desired, the control inputs on J2 must be connected to one of the sensor power control lines on J1 (pins 8, 9, A, and C). In this mode, the computer will set these control lines to a logical high, enabling power to the sensors, and delay one minute before acquiring data. The delay allows the sensors time to equilibrate before data-acquisition begins. If sensor power control is not used, there is no one-minute delay before acquiring data.

Power to the digital tape drive and the radio transmitter can be controlled by connecting one of the pins on J2 to pin E of J1 for the tape drive and pin F of J1 for the radio transmitter. The computer sets these control lines automatically, and there is no difference in operation if power control is used. If power control to these devices is not desired, any one of the three methods described above for sensor power can be used.

The computer will also control power during remote events (section 2.3). Connecting one of the pins on J2 to pin B on J1 enables this option. At the start of any data-acquisition, the computer will check to see if any remote events have occurred. If an event has occurred, the computer will enable power on the relay before any data-acquisition occurs. If the computer sees that no events have occurred, the relay is turned off. This means that the relay will remain on, during all events, until the computer determines that no more events have occurred.

1.3 REMOTE EVENT DETECTION

The computer provides 16 lines of input for detecting remote events. A logic high (+5 volts) on any one of these inputs is an indication to the computer that an event has occurred. One output line from the computer is available to reset any remote event circuitry. This line is normally held at a logic high (+5 volts) and momentarily brought to ground (0 volts) after the computer has tested the 16 lines of input.

The remote event board, shown in Figure 11, was designed to detect sea water flooding of the island being studied. The board consists of two independent detection circuits. The flood sensor is, essentially, two bare wires suspended in air above the ground. Their height above the ground determines how high the flooding must be to trigger the event. The flood sensor is connected to the remote event board using connectors shown in Figure 4. When the sea water comes in contact with the the two wires, the +5 volts present on one of the wires is reduced, which triggers a voltage comparator (U1 and U2) on the board. The sensitivity, or trigger voltage, can be set by adjusting the potentiometer (R2 and R6) and measuring the voltage present on the test point (TP1 and TP2) as shown in Figure 12. The output of each voltage comparator is connected to a counter (U5 and U6), which increments each time the voltage comparator triggers. If the counter should reach maximum value, it sets a flip-flop on U4. When the computer reads the output of these two circuits, it reads the output of the voltage comparators to determine if a flood is currently present, the outputs of the flip-flops to see if the counters have overflowed, and the current count of each counter to determine the number of events that have occurred. In this way the computer can detect a current event, a past event, or a certain number of events. After the computer has read the output of each circuit, the counter and flip-flops are reset using a control line to the board.

1.4 ANALOG SIGNAL CONDITIONING

The analog signal conditioning boards are provided as an option to filter out any high-frequency noise that might be present on the sensor signals. This high-frequency noise can be generated by the use of long cables to connect the computer and the sensors, and (or) the use of switching regulators to provide power to the sensors. Each board contains four independent preamplifier and filter circuits (see Figures 13 and 14). The preamplifier (U1, U4, U7, AND U10), an instrumentation amplifier set at a gain of 1, uses a differential input that has its negative input connected to ground. The rest of each circuit comprises a fifth-order Butterworth low-pass filter that is set with a cutoff frequency of 61 hertz. The circuit introduces only 200 microvolts of noise (well below the resolution of the analog-to-digital converter) when used with both +12 and -12 volts power. If only +12 volts power is used, a voltage inverter (U13) must be used to generated the necessary negative voltage. The voltage inverter generates approximately 5 millivolts of noise on the output of each circuit, so its use can only reduce noise levels to 5 millivolts. The analog signals from each sensor are connected to the signal conditioning board as shown in Figure 4. There is also a 5-pin connector provided to monitor the signal output of each channel. These outputs are connected to a unity-gain, buffer amplifier which protects the signals going to the analog-to-digital converter form any interference that might be generated by using these outputs.

1.5 RADIO TRANSMITTER

The radio transmitter used in this system is part of a MICRO-TEL instrument data-link system manufactured by Measurement Devices LTD., Houston Texas. The data-link is a one-way communication system that takes RS232 data from one computer, transmits that information via a 406 megahertz transmitter, and converts the data back to RS232 at the receiver. The data transmission rate is 1200 baud. Connections to the transmitter are: +12 volts, ground, the RS232 data line, and an enable line, which must be set to +12 volts 500 milliseconds before transmitting data. The RF output is 0.5 watts at 12 volts. This limits the range of the system to about 10 kilometers, using the standard radial antennae. Greater range is achieved by using a 10-watt booster on the transmitter output and a dual, stacked yagi antennae array.

1.6 SOLID-STATE MEMORY

The solid-state memory drive used in this system is a Datavault, manufactured by Citadel Computer Corp., Amherst, N.H. This mass storage unit provides primary storage of data in situations where radio transmission of data is not possible and auxiliary data storage during remote events when radio transmission may not be reliable. The Datavault consists of a removable, 1-megabyte cartridge that is connected to the computer via a CMX900 interface board. A similar interface board is used in an IBM PC-compatible computer for reading the data produced by the data-acquisition system. On the PC, the datavault is configured to appear as a floppy disk drive, complete with formatting, directory, and the necessary house-keeping sectors found on any IBM-compatible floppy disk. To simply the requirements for reading the data on a PC computer, the data-acquisition system will format the data and write it to the Datavault as a file. The directory is updated with each data file to provide a file name (using the sample number for the label and either INT or PRI for the file extension), the sample time, and the file size. All of the house-keeping sectors are updated with each data file written. Each data cartridge must be formatted on a PC computer before it is installed in the data-acquisition system. The boot and housekeeping sectors created by the format process provide the data-acquisition system with the size of the cartridge and the locations needed to write IBM-compatible files. This process eliminates any need for special software to read or manipulate the data on a PC computer. Larger capacity datavault cartridges are available, but only the 1-megabyte cartridge has been used thus far.

1.7 DIGITAL TAPE RECORDER

The digital tape recorder used in this system is a FEEDBACK 344, 20-megabyte cartridge tape drive from ADPI, Troy, Ohio. This mass storage unit allows both primary storage of data in situations where radio transmission of data is not possible and auxiliary data storage during remote events when radio transmission may not be reliable. The tape drive is powered with +12 volts DC and has a standby power option in case of power loss. Data are sent to the drive through an RS232 interface, which is set at 9600 baud. Although the tape drive can record binary data, the data are converted to ASCII characters by the computer before they are sent to the drive. This allows the computer to control the operation of the tape drive through the same RS232 interface using ASCII control characters.

There are two major drawbacks to using this tape drive. The first is an operating temperature range of only $+10^{o}C$ to $+45^{o}C$, which limits its usefulness in the field to areas of mild climate. The second drawback is the power consumption of the drive: 36 watts while recording data and 12 watts while idle. This is a severe drain on most battery systems. Several attempts were made to power-down the drive when not in use, and use the power down option to save the tape position. The 6-amp current surge that occurs when the drive is powered up would occasionally reset the electronics in the drive or, worse, cause the drive to malfunction. The best use of this mass storage is in controlled environments where enough power is available to leave the tape drive powered at all times.

SECTION 2. PROGRAMMING THE DATA-ACQUISITION SYSTEM

This small, battery-powered computer system is designed to be as flexible as possible in acquiring analog data from as many as 16 channels and recording these data via a radio link to another computer, a cartridge tape recorder, or a solid-state memory device, or any combination of the three. A number of parameters must be entered to program the system to acquire and store data. The software embedded in the computer will automatically prompt the user by displaying all of the questions that must be answered before it can set up the data-acquisition procedure. The following sections provide an explanation of what each response to the computer's questions means in terms of the data acquired by the system. The last section will summarize the system test functions available in the software.

2.1. STARTING THE COMPUTER

Figure 2 illustrates the various connectors located on the top cover of the computer. Connect the power plug, the analog signal plug, and the terminal (used to enter the operational parameters) to the locations shown in the figure. If you are using the radio link, the tape recorder, or the solid-state memory be sure to connect these devices to the appropriate connectors. If remote event mode or power control to the sensors is desired, then connect the external control plug.

The terminal should be setup for 9600 baud, space parity (no parity), 1 stop bit, and an 8-bit data word. The tape recorder should be setup for 19,200 baud, 1 stop bit, no parity, and an 8-bit data word.

The system will automatically go into the data-acquisition mode using the last-entered parameters stored in the battery-backed RAM. This procedure allows the system to return to normal operation in the event power is temporarily lost while in operation. To enter new values, hit return (CR) on the terminal. The system will delay 5 seconds and then display a sign-on message followed by a request to enter the time. The computer is now started and ready for parameter entry.

2.2. SETTING THE TIME

Data-acquisition is done on regular intervals based on the time-of-day. To correlate these data with any other processes that may be going on, the computer must start its own clock in reference to a standard time. This is accomplished with the following prompts.

```
Enter TIME + 1 MIN
 YR/MTH/DY/HR//MIN
```

Time is entered in the following format: Year(YY = last two digits of the current year), any separator (non-numerical), month (MM = a number from 1 to 12), any separator, day (DD = a number from 1 to 31), any separator, hour (HH = a number from 0 to 23), any separator, and minute (mm = a number from 0 to 59).The

separator commonly used is a slash(/), although commas, periods, etc., can also be used. The time entry should appear as follows:

YY/MM/DD/HH/mm(CR)

Each entry must be followed by a carriage return (CR) to enter it into the system. Remember to add 1 minute to the current time since the clock remains off until you tell it to start. The system will check the entry for any mistakes in format and will ask again if an error is found. If the entry made is in the correct format but is not the correct time, you must remove the power to the system (or push the reset button) and start over. The computer will then load the entered time into its own clock and setup to start it. The next request provides the computer with the means of starting the clock with whatever time reference the user is employing.

Hit CR to set
the clock

At this point, you must hit return (CR) to start the clock. Wait until the exact time and then hit CR. The computer will start its clock and display the time set message as an indication that the time is set.

Setting the time

Time-of-day is now set, and the computer will request the next parameter.

2.3. REMOTE EVENT MODE

In some instances, the ability to sample at a higher data rate during an outside event is desirable. The computer can, at the user's option, select a remote event mode that will test a series of 16 inputs (on the external control connector) for a logic high (+5 volts). If an event has occurred, the computer will suspend the normal data rate and begin sampling at a data rate entered by the user for the remote mode. When the computer detects that no more remote events have occurred, the normal data rate will be resumed. The contents of the remote input lines are recorded into the header on each sample (section 3). The remote event can be enabled with the following questions. If the user answers no(N) to the first question, the computer will not test the port for events and will skip to the next step.

Remote event
mode(Y/N)

Answering yes(Y) to this question will result in the computer asking for the data rate during remote events.

Sample by the
hour or minute?
(H/M)

Answering with hour (H) will result in the following message:

ENTER # samples
per day

Enter the number of times each day that data are to be collected. To keep the system on a standard sampling scheme, the number must be one that is evenly divided into 24 hours. This allows the user to know exactly what times each day the system will collect data. For example, if you enter 4 (times a day),the system will collect data every six hours at 0000, 0600, 1200, and 1800. The computer will check your entry and if it is not valid, it will ask again. Valid entries are 1, 2, 3, 4, 6, 8, 12, and 24.

Answering with minute (M) will result in the following message:

ENTER # samples
per hour

Enter the number of times each hour data are to be collected. To keep the system on a standard sampling scheme, the number must be one that is evenly divided into 60 minutes. The user will then know exactly what times in each hour the system will collect data. For example, an entry of 4 (times per hour) will result in the system collecting data every 15 minutes at :00, :15, :30, and :45. The computer will check your entry and if it is not valid, it will ask again. Valid entries are 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30 and 60.

2.4. POWER CONTROL TO THE SENSORS

The computer has the capability to control the power to any sensors used in the data-acquisition system (saving power, if needed). There are four output lines on the external control connector that will go to a logic high (+5 volts) at the start of each data collection, and the computer will wait for one minute before taking any samples. In addition, a flag is set in each data header to let the user know that a 1-minute delay occurred before data collection began (section 3). If the above question is answered with no (N), the computer will not set the output lines and there will be no one minute delay before the system collects data.

Control power
to sensors?
(Y/N)

2.5. RECORDING THE SCAN COUNTS

In some data-acquisition modes, it is desirable to include the scan count among the data recorded by the system. An end-of-scan separator is always recorded, but in some situations where recovery of garbled data is desired, the scan count can determine where in the time series the data lie.

```
Scan count in
data(Y?N)
```

Answering yes (Y) to this question will cause the scan count to be included into the data. if you answer no (N) to this question,the computer will not record the scan count into the data, thus freeing up more memory for more data (the scan count uses two bytes for each scan of data).

2.6. SENDING DATA TO THE RADIO LINK

One option for storing the acquired data is to transmit the data through an radio link to another computer. The data transmission rate is slow (1200 baud) but provides a means of monitoring the data without visiting the data collection site. The computer will first ask if you want to send the collected data to the system's radio link.

```
Write data to
RADIO?(Y/N)
```

If no (N) is entered, no data will be sent and the computer will skip to the next step. If yes (Y) is entered, the computer will offer the user the following options for recording data:

```
1.Every time
2.Remote only
ENTER mode
```

If 1 is entered,the computer will send the data to the radio link every time data are collected. If 2 is entered, the computer will send data to the radio link only during remote events.

2.7. SENDING DATA TO THE CARTRIDGE TAPE RECORDER

The second storage option available for storing the acquired data is to record the data on a digital tape recorder. The computer will first ask if you want to send the collected data to the system's tape recorder.

```
Write data to
TAPE?(Y/N)
```

If no (N) is entered, no data will be sent and the computer will skip to the next step. If yes (Y) is entered, the computer will offer the user the following options for recording data:

```
1.Every time
2.Remote only
ENTER mode
```

If 1 is entered,the computer will send the data to the tape recorder every time data are collected. If 2 is entered, the computer will send data to the tape recorder only during remote events.

2.8. SENDING DATA TO THE SOLID-STATE MEMORY DEVICE

The third option available for storing the acquired data is to record the data to a solid-state memory device that acts like a high-capacity, floppy disk drive. The computer will first ask if you want to send the collected data to the system's solid-state memory.

```
Write data to
S-STATE?(Y/N)
```

If no (N) is entered, no data will be sent and the computer will skip to the next step. If yes (Y) is entered, the computer will offer the user the following options for recording data:

```
1.Every time
2.Remote only
ENTER mode
```

If 1 is entered,the computer will send the data to the solid-state memory every time data are collected. If 2 is entered, the computer will send data to the solid-state memory only during remote events.

CAUTION ### - the computer will allow the user to answer no to all three options for recording data. If no (N) is entered for all three options, the data will be lost.

2.9. COLLECTING DATA AT INTERVALS BETWEEN THE PRIMARY DATA SET

The computer has the capability of collecting additional data sets between the primary data sets. This allows the user to record some channels at different data rates or to record a shorter data record between the primary data sets. This option is enabled by answering yes (Y) to the next question.

```
Write additional
data between
main data sets?
(Y/N)
```

If no (N) is entered, this option will not be used.

2.10. COLLECTING AN ADDITIONAL SCAN SET TOGETHER WITH EXISTING DATA

The computer is also has the capable of collecting one additional set of data for each data mode. This is enabled by the following message:

```
Write only 1 set
of scans in each
data set?(Y/N)
```

If yes (Y) is entered, the system will collect only one data set and skip the next step. If no (N) is entered, the system will record an additional set of data prior to collecting the primary data set (or the intermediate data set if section 2.9 was answered yes).

2.11. SET PARAMETERS FOR 1ST SCAN SET

This step will occur only if section 2.10 was answered with a yes (Y). The first question will be for the lowest channel of analog data to be acquired.

```
ENTER - 1st scan
set for all data
Base chan.
```

There are 16 channels of data that can be recorded by this system. Each scan set has the option of recording any combination of channels as long as they are consecutive channels. The message above asks for the base channel (the lowest channel number) to be used for this first set of scans. It can be anything from 1 to 16. After this entry, the following message will appear:

```
# chans.?
```

The entry here depends on what was entered for the base channel. All 16 channels can be used if the base channel was 1. The total number of channels plus the base channel must not exceed 16. After this entry, the following message will appear:

scans?

Any number up to 9999 will be accepted. After all entries have been made, the computer will check to be sure that the number of scans does not exceed the available memory. The next message is:

Scan rate less
than 1Hz?(Y/N)

The system is capable of digitizing data at rates up to 255 scans per second. Sampling slower than 1 scan per second, however, requires using a different time base. Answering yes (Y) to this question selects the slower time base and the following message will appear:

ENTER scans/min
(1 - 30)

This entry will set the scan rate at values from a scan of data every 2 seconds (an entry of 30 scans/min) to a scan of data every 60 seconds (an entry of 1 scan/min). To achieve even slower rates, the system must be programmed to record each scan as a separate data set (i.e., enter 1 scan of data and use the samples/hour entry for anything from 1 scan/ min to once an hour or use the samples/day entry for anything from 1 scan/hour to once a day).

If a scan rate faster than every 2 seconds is desired, the scan rate question should have been answered with a no (N). This would have resulted in the following message appearing:

ENTER scans/sec
(1 - XXXX)

The XXXX part of the display is a four-digit number that the computer will automatically calculate for the maximum scan rate. Since it takes longer to digitize 16 channels than one, the maximum sample rate is dependent on the number of channels entered previously. The computer takes this number and calculates the maximum scan rate based on that information. This entry will set the scan rate from a maximum of 255 scans per second down to 1 scan per second.

2.12. SET PARAMETERS FOR INTERMEDIATE DATA SET

This step will occur only if section 2.9 was answered with a yes (Y). The first question will be for the lowest channel of analog data to be acquired. There are 16 channels of data that can be recorded by this system. Each scan set has the option of recording any combination of channels as long as they are consecutive.

ENTER - scan set
for in-between data
Base chan.

The message asks the user to identify the base channel (the lowest channel number) to be used for this first set of scans. It can be anything from 1 to 16. After this entry, the following question will appear:

chans.?

The entry here depends on what was entered for the base channel. All 16 channels can be used if the base channel was 1. The total number of channels plus the base channel must not exceed 16. After this entry, the following message will appear:

scans?

Any number up to 9999 will be accepted. After all entries have been made, the computer will check to be sure that the number of scans does not exceed the available memory. The next message is:

Scan rate less
than 1Hz?(Y/N)

The system is capable of digitizing data at rates up to 255 scans per second. Sampling slower than 1 scan per second, however, requires using a different time base. Answering yes (Y) to this question selects the slower time base and the following message will appear:

ENTER scans/min
(1 - 30)

This entry will set the scan rate at values from a scan of data every 2 seconds (an entry of 30 scans/min) to a scan of data every 60 seconds (an entry of 1 scan/min). To achieve even slower rates, the system must be programmed to record each scan as a separate data set (i.e., enter 1 scan of data and use the samples/hour entry for anything from 1 scan/ min to once an hour or use the samples/day entry for anything from 1 scan/hour to once a day).

If a scan rate faster than every 2 seconds is desired, the scan rate question should have been answered with a no (N). This would have resulted in the following message appearing:

ENTER scans/sec
(1 - XXXX)

The XXXX part of the display is a four-digit number that the computer will automatically calculate for the maximum scan rate. Since it takes longer to digitize 16 channels than one, the maximum sample rate is dependent on the number of channels entered previously. The computer takes this number and

calculates the maximum scan rate based on that information. This entry will set the scan rate from a maximum of 255 scans per second down to 1 scan per second.

Next, the data rate is set by responding to the prompt:

```
Sample by the
hour or minute?
(H/M)
```

Answering with hour (H) will result in the following message:

```
ENTER # samples
per day
```

Enter the number of times each day data are to be collected. To keep the system on a standard sampling scheme, the number must be one that is evenly divided into 24 hours. This allows the user to know exactly what times each day the system will collect data. For example, an entry of 4 (times a day) will result in the system collecting data every six hours at 0000, 0600, 1200, and 1800. The computer will check your entry and if it is not valid, it will ask again. Valid entries are 1, 2, 3, 4, 6, 8, 12, and 24.

Answering with minute (M) will result in the following message:

```
ENTER # samples
per hour
```

Enter the number of times each hour data are to be collected. To keep the system on a standard sampling scheme, the number must be one that is evenly divided into 60 minutes. The user will then know exactly what times in each hour the system will collect data. For example, an entry of 4 (times per hour) will result in the system collecting data every 15 minutes at :00, :15, :30, and :45. The computer will check your entry and if it is not valid, it will ask again. Valid entries are 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30 and 60.

2.13. SET PARAMETERS FOR PRIMARY DATA SET

This step will occur regardless of any other data modes. These parameters are also (except the data rate) the parameters that will be used by the remote event mode during remote events. The first question will be for the lowest channel of analog data to be acquired.

```
ENTER - scan set
for main data
Base chan.
```

Sixteen channels of data can be recorded by this system. Each scan set has the option of recording any combination of channels as long as they are consecutive channels. The message above asks for the base channel (the lowest channel number) to be used for this first set of scans. It can be anything from 1 to 16. After this entry, the following message will appear:

```
# chans.?
```

The entry here depends on what was entered for the base channel. All 16 channels can be used if the base channel was 1. The total number of channels plus the base channel must not exceed 16. After this entry, the following message will appear:

```
# scans?
```

Any number up to 9999 will be accepted. After all entries have been made, the computer will check to be sure that the number of scans does not exceed the available memory. The next message is:

```
Scan rate less
than 1Hz?(Y/N)
```

The system is capable of digitizing data at rates up to 255 scans per second. Sampling slower than 1 scan per second, however, requires using a different time base. Answering yes (Y) to this question selects the slower time base and the following message will appear:

```
ENTER scans/min
(1 - 30)
```

This entry will set the scan rate at values from a scan of data every 2 seconds (an entry of 30 scans/min) to a scan of data every 60 seconds (an entry of 1 scan/min). To achieve even slower rates, the system must be programmed to record each scan as a separate data set (i.e., enter 1 scan of data and use the samples/hour entry for anything from 1 scan/ min to once an hour or use the samples/day entry for anything from 1 scan/hour to once a day).

If a scan rate faster than every 2 seconds is desired, the scan rate question should have been answered with a no (N). This would have resulted in the following message appearing:

```
ENTER scans/sec
(1 - XXXX)
```

The XXXX part of the display is a four-digit number that the computer will automatically calculate for the maximum scan rate. Since it takes longer to digitize 16 channels than one, the maximum sample rate is dependent on the number of channels entered previously. The computer takes this number and calculates the maximum scan rate based on that information. This entry will set the scan rate from a maximum of 255 scans per second down to 1 scan per second.

Next, the data rate is set by responding to the prompt:

```
Sample by the
hour or minute?
(H/M)
```

Answering with hour (H) will result in the following message:

```
ENTER # samples
per day
```

Enter the number of times each day data are to be collected. To keep the system on a standard sampling scheme, the number must be one that is evenly divided into 24 hours. This allows the user to know exactly what times each day the system will collect data. For example, an entry of 4 (times a day) will result in the system collecting data every six hours at 0000, 0600, 1200, and 1800. The computer will check your entry and if it is not valid, it will ask again. Valid entries are 1, 2, 3, 4, 6, 8, 12, and 24.

Answering with minute (M) will result in the following message:

```
ENTER # samples
per hour
```

Enter the number of times each hour data are to be collected. To keep the system on a standard sampling scheme, the number must be one that is evenly divided into 60 minutes. The user will then know exactly what times in each hour the system will collect data. For example, an entry of 4 (times per hour) will result in the system collecting data every 15 minutes at :00, :15, :30, and :45. The computer will check your entry and if it is not valid, it will ask again. Valid entries are 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30 and 60.

2.14. ERROR CHECKING

At this point the computer will check the entries in all data sets to make sure that the data will fit into the available memory and that there is enough time to collect and record the data between sample intervals. If the the computer finds that the data will not fit into the available memory, the following message will appear :

```
ERROR:not enough
memory!

CR to continue
```

After hitting return (CR), the entire entry procedure must be repeated.

If the computer finds that there is not enough time between samples, the following message will appear:

```
ERROR:not enough
time!

CR to continue
```

After hitting return (CR), the entire entry procedure must be repeated.

If no error occurs, the computer will proceed to the next step without displaying any message.

2.15. PLAYBACK OF PARAMETERS ENTERED

If there are no errors in time or memory, the computer will display the entered parameters your for review. Since the terminal most commonly used to enter the parameters has only 4 lines of display, the playback procedure will display the information a little bit at a time, followed by:

CR to continue

To view all of parameters, it is necessary to hit the return key (CR) to complete this procedure.

The first display will show either

A.

Remote event is
not selected

CR to continue

OR

B.

Remote event is
selected
data every XXmin
CR to continue

OR

C.

Remote event is
selected
data every XXhr
CR to continue

Message A will be displayed if section 2.3 was answered with a no (N). Message B will be displayed if section 2.3 was answered with a yes (Y), the data rate was question answered with an M, and XX represents the number of minutes between data acquisition in the remote mode. Message C will be displayed if section 2. 3 was answered with a yes (Y), the data rate question answered with an H, and XX is the entry for the number of hours between data acquisition in the remote mode.

After hitting return (CR), one of the following messages will be displayed:

A.

```
Scan count

in data
CR to continue
```

OR

B.

```
Scan count

not in data
CR to continue
```

If the answer to section 2.4 was yes (Y), then message A will be displayed. If the answer was no, message B will be displayed.

After hitting return(CR), one of the following messages will be displayed:

A.

```
Data to RADIO
at no time

CR to continue
```

OR

B.

```
Data to RADIO
every time

CR to continue
```

OR

C.

```
Data to RADIO
only during
remote events
CR to continue
```

Message A will be displayed if section 2.6 was answer with a no (N). Message B will be displayed if section 2.6 was answered with a yes(Y) and 1 was entered for the option. Message C will be displayed if section 2.6 was answered with a yes (Y) and 2 was entered for the option.

After hitting return (CR), one of the following messages will be displayed:

A.
```
Data to TAPE
at no time

CR to continue
```
OR

B.
```
Data to TAPE
every time

CR to continue
```
OR

C.
```
Data to TAPE
only during
remote events
CR to continue
```

Message A will be displayed if section 2.7 was answered with a no (N). Message B will be displayed if section 2.7 was answered with a yes (Y) and 1 was entered for the option. Message C will be displayed if section 2.7 was answered with a yes (Y) and 2 was entered for the option.

After hitting return (CR), one of the following messages will be displayed:

A.
```
Data to S-STATE
at no time

CR to continue
```
OR

B.
```
Data to S-STATE
every time

CR to continue
```
OR

C.
```
Data to S-STATE
only during
remote events
CR to continue
```

Message A will be displayed if section 2.8 was answered with a no (N). Message B will be displayed if section 2.8 was answered with a yes (Y) and 1 was entered for the option. Message C will be displayed if section 2.8 was answered with a yes (Y) and 2 was entered for the option.

If the answer to the question in section 2.10 was no (N), the following display will appear:

A.
```
1st scan set
XXsec XXXXscan
ch XX - XX
CR to continue
```

OR

B.
```
1st scan set
XXmin XXXXscan
ch XX - XX
CR to continue
```

XXXXscan is the number of scans of data that will be acquired. ch XX - XX represents the first and last channels of data that will be recorded. XXsec and XXmin represent the scan rate in scans per second or scans per minute, depending on what was entered.

If the answer to the question in section 2.9 was yes (Y), the following display will appear:

A.
```
In-between data
XXsec XXXXscan
ch XX-XX XXmin
CR to continue
```

OR

B.
```
In-between data
XXmin XXXXscan
ch XX-XX XXhr
CR to continue
```

XXXXscan is the number of scans of data that will be acquired. ch XX - XX represents the first and last channels of data that will be recorded. XXsec and XXmin represent the scan rate in scans per second or scans per minute, depending on what was entered. XXmin and XXhr represent the data rate in data every XX minutes or data every XX hours, depending on what was entered.

After hitting return (CR), the following display will appear:

A.

```
Main data
XXsec XXXXscan
ch XX-XX XXmin
CR to continue
```

OR

B.

```
Main data
XXmin XXXXscan
ch XX-XX XXhr
CR to continue
```

XXXXscan is the number of scans of data that will be acquired. ch XX - XX represents the first and last channels of data that will be recorded. XXsec and XXmin represent the scan rate in scans per second or scans per minute, depending on what was entered. XXmin and XXhr represent the data rate in data every XX minutes or data every XX hours, depending on what was entered.

After hitting return, the computer will ask you if the entries are correct. Answer the question no (N) to re-enter the parameters, and answer yes (Y) to continue.

2.16. ENTER THE START TIME

ENTER start time
YR/MTH/DY/HR/MIN

This entry is done in exactly the same format as the time entered at section 2.2. The time entry refers to when you want the system to collect the first data set. If you do not start the system at an even hour, it will acquire data at the entered time and then start collecting data at the next even hour.

###CAUTION### The computer does not check the start time against the current time, so be sure the entry is correct before hitting return (CR).

After the start time has been entered, the program entry is completed. No other prompts will appear. Disconnect the terminal and seal up the system. You are now ready.

EXAMPLE

YOUR ENTRIES ARE SHOWN IN **BOLD** PRINT

Enter TIME + 1 MIN
YR/MTH/DY/HR//MIN

You enter the current time.

88/1/27/17/12 CR

Hit CR to set
the clock
17:12 1/27/88

When the time you have entered reaches the exact minute,

CR

Setting the time

Remote event
mode(Y/N)

Answer yes(Y) to enable remote event mode.

Y CR

Sample by the
hour or minute?
(H/M)

Enter hour(H)

H CR

ENTER # samples
per day

Enter 24 times per day (every hour)

24 CR

Control power
to sensors?
(Y/N)

Answer yes(Y) to enable control of power to the sensors.

Y CR

```
 Scan count in
 data(Y?N)
```

Answer yes(Y) to put the scan count in the data

```
Y CR

 Write data to
 RADIO?(Y/N)
```

Answer yes(Y) to send data to the radio link.

```
Y CR

 1.Every time
 2.Remote only
 ENTER mode
```

Enter 1 for every time.

```
1 CR

 Write data to
 TAPE?(Y/N)
```

Answer yes(Y) to send data to the tape recorder.

```
Y CR

 1.Every time
 2.Remote only
 ENTER mode
```

Enter 2 for remote events only.

```
2 CR

 Write data to
 S-STATE?(Y/N)
```

Answer no(N) to send no data to the solid-state memory.

```
N CR

 Write additional
 data between
 main data sets?
 (Y/N)
```

Answer yes(Y) to collect intermediate data sets.

```
Y CR
```

```
 Write only 1 set
 of scans in each
 data set?(Y/N)
```

Answer no(N) to collect an additional scan set.

```
N CR
```

```
 ENTER - 1st scan
 set for all data
 Base chan.
```

Enter 1 for the first channel of data

```
1 CR
```

```
 # chans.?
```

Enter 3 for a total of 3 channels of data per scan.

```
3 CR
```

```
 # scans?
```

Enter 128 for 128 scans of data.

```
128 CR
```

```
 Scan rate less
 than 1Hz?(Y/N)
```

Answer no(N).

```
N CR
```

```
 ENTER scans/sec
 (1 - 0255)
```

Enter a scan rate of 10 scans per second.

```
10 CR
```

```
 ENTER - scan set
 for in-between data
 Base chan.
```

Enter channel 4 for the first channel.

```
4 CR
```

```
 # chans.?
```

Enter a total of 2 channels.

```
2 CR
```

```
 # scans?

     Enter 2048 scans.
2048 CR

 Scan rate less
 than 1Hz?(Y/N)

     Answer no(N).

N CR

 ENTER scans/sec
 (1 - 0255)

     Enter a scan rate of 30 scans per second.

30 CR

 Sample by the
 hour or minute?
 (H/M)

     Answering with hour(H).

H CR

 ENTER # samples
 per day

     Enter 24 times per day (every hour).

24 CR

 ENTER - scan set
 for main data
 Base chan.

     Enter channel 4 for the first channel.

4 CR

 # chans.?

     Enter a total of 12 channels.

12 CR

 # scans?

     Enter 4096 scans.

4096 CR
```

```
 Scan rate less
 than 1Hz?(Y/N)
```

Answer no(N).

N CR

```
 ENTER scans/sec
 (1 - 0083)
```

Enter a scan rate of 2 scans per second.

2 CR

```
 Sample by the
 hour or minute?
 (H/M)
```

Answering with hour(H).

H CR

```
 ENTER # samples
 per day
```

Enter 6 times per day (every 4 hours).

6 CR

```
 Remote event is
 selected
 data every 01hr
 CR to continue
```

CR

```
 Scan count

 in data
 CR to continue
```

CR

```
 Data to RADIO
 every time

 CR to continue
```

CR

```
 Data to TAPE
 only during
 remote events
 CR to continue
```

CR

```
 Data to S-STATE
 at no time

 CR to continue
```

CR

```
 1st scan set
 10sec 000128scan
 ch 01 - 03
 CR to continue
```

CR

```
 In-between data
 30sec 002048scan
 ch 04-05 01hr
 CR to continue
```

CR

```
 Main data
 02sec 004096scan
 ch 04-15 04hr
 CR to continue
```

CR

```
 Are entries
 correct?(Y/N)
     Answer yes(Y).
```

Y CR

```
 ENTER start time
 YR/MTH/DY/HR/MIN
```

You enter the time the data will begin to be acquired for the first time.

88/1/27/18/00

done.

The above entries will result in the system acquiring its first data set on January 27, 1988 at 18:00. The first data set is always the primary data set. Remote event is enabled and, when active will acquire a primary data set every hour. At all other times the primary data set will be acquired at 04:00, 08:00, 12:00, 16:00, 20:00, and 24:00. Intermediate data will be acquired at each hour in which a primary data set is not acquired. The system will control the power to the sensors and will not acquire data until 1 minute after the hour. The scan count will be recorded in the data. Data will be sent to the radio link each time a data set is acquired, and sent to the tape recorder only during remote events. The main data will consist of 128 scans of data from channels 1, 2, and 3, sampled at 10 scans per second. The data will also contain 4096 scans of data from channels 4 to 15, sampled at 2 scans per second. The intermediate data set will consist of 128 scan of data from channels 1, 2, and 3 sampled at 10 scans per second and 2048 scans of data from channels 4 and 5, sampled at 30 scans per second.

2.17 TEST FUNCTIONS

There are several tests available in the software that allow the user to check out the main system functions. All test functions are initiated by entering a control character on the terminal. Starting a test requires getting the computer into the parameter entry mode (see section 2.1). Any time the computer asks for a parameter entry (such as the current time), the user may instead enter the appropriate control character to start a test. All tests are ended by resetting the computer, which is done by either pushing the reset button on the computer housing (Figure 2) or removing power to the system.

The real-time clock may be tested by entering a 'Control T' (On most terminals, this is done by entering the 'T' while holding down the control key). This results in the computer displaying the current each second until the even minute. At the even minute, the computer halts and will have to be reset before doing anything else.

The radio link can be tested by entering a 'Control S'. This results in the computer sending an ASCII character test pattern to the radio every 20 seconds, until the computer is reset. This allows the user to check the transmitter and the entire data link by setting up a terminal on the receiver.

The analog-to-digital circuits can be tested by entering a 'Control A'. The computer will then ask which channel to test. After the channel number is entered, the computer will continually display the digital word value (in hexadecimal) for that channel until the computer is reset.

SECTION 3. FORMAT OF THE DATA FILES

A 34-byte header is recorded at the beginning of every data set to define the parameters that set the format of the data. The header contains the time-of-day the sample was acquired, the sample number of the data, the voltage value of the channel 16 analog input (provided as a means of monitoring a system variable, such as battery voltage or temperature, that does not require a time-series measurement), a flag to indicate whether the scan count is placed in the data, a flag to indicate whether the following data comprise a primary data set or an intermediate data set, It also displays the value of both remote event counters, a flag to indicate whether the computer is controlling power to the sensors (which means that data-acquisition is occurring one minute beyond the recorded start time), the base channels for all data sets, the number of channels for all data sets, the scan rate for all data sets, and the number of scans recorded for all data types. All data are recorded as hexadecimal (h) values. All 2-byte words are recorded low-byte first, then high-byte.

3.1 HEADER FORMAT

BYTE 0	Year of time of sample (00 - 63h)
BYTE 1	Day of time of sample (01 - 1Fh)
BYTE 2	Month of time of sample (01 - 0Ch)
BYTE 3	Seconds of time of sample (00 - 3Bh)
BYTE 4	Minutes of time of sample (00 - 3Bh)
BYTE 5	Hours of time of sample (00 - 17h)
BYTE 6	Tenths of seconds of time of sample (00 - 63h)
BYTE 7	Low byte of sample number (00 - FFh)
BYTE 8	High byte of sample number (00 - FFh)
BYTE 9	Low byte of channel 16 data(00 - FFh)
BYTE 10	High byte of channel 16 data(00 - 0Fh)
BYTE 11	Flag to indicate whether the scan count is in the data (81h = no, F2h = yes)
BYTE 12	Flag to indicate whether data is the primary data set (81h = no, F2h = yes)
BYTE 13	Remote event counter A(00h - FFh)
BYTE 14	Remote event counter B(00h - FFh)
BYTE 15	Flag to indicate whether the computer controls power to the sensors (81h = no, F2h = yes) NOTE: Yes indicates that the start of data is actually 1 minute later than time recorded.
BYTE 16	Base channel for the first scan data (00h-10h)
BYTE 17	Base channel for the intermediate data (00h-10h)
BYTE 18	Base channel for the primary data (00h-10h)
BYTE 19	# channels for the first scan data (00h-10h)
BYTE 20	# channels for the intermediate data (00h-10h)
BYTE 21	# channels for the primary data (00h-10h)
BYTE 22	Scan rate for the first scan data (00h-FFh)
BYTE 23	Scan rate for the intermediate data (00h-FFh)
BYTE 24	Scan rate for the primary data (00h-FFh)
BYTE 25	Scan time base flag for the first scan data (F2h = scans/min, 81h = scan/sec)
BYTE 26	Scan time base flag for the intermediate data (F2h = scans/min, 81h = scan/sec)
BYTE 27	Scan time base flag for the primary data (F2h = scans/min, 81h = scan/sec)
BYTE 28	Low byte of scan count for first scan data
BYTE 29	High byte of scan count for first scan data
BYTE 30	Low byte of scan count for intermediate data
BYTE 31	High byte of scan count for intermediate data
BYTE 32	Low byte of scan count for primary data
BYTE 33	High byte of scan count for primary data

\ \ \ \

\ \ \ \

BYTE 34 Start of data

The format of the data depends on the values set in the header information. If byte 19 (# channels of 1st scan data) is not zero, then the first byte of data will be the low byte of data for the last channel of the first scan set (Base channel plus # channels). This is followed by the high byte of the last channel and then by the lower channels (low byte then high byte) until the base channel has been recorded.

The next data value depends on byte 11 (scan count flag). If the value of this flag is F2h (yes), the low byte of the scan count will be recorded followed by the high byte. The next byte will be the end of scan separator (FFh). If the flag is 81h (no), only the end of scan separator is recorded. This format is repeated until the number of scans indicated in bytes 28 and 29 of the header have been recorded.

The format of the rest of the data depends on the value of byte 12 of the header. If the value is F2h (yes), the remaining data form the primary data set unless the number of channels for this data set is zero, in which case there will be no further data. If byte 21 (# channels of primary data) is not zero, the first byte of data will be the low byte of data for the last channel of the primary data(Base channel plus # channels). This is followed by the high byte of the last channel and then by the lower channels (low byte then high byte) until the base channel has been recorded.

The next data value depends on byte 11 (scan count flag). If the value of this flag is F2h (yes), the low byte of the scan count will be recorded followed by the high byte. The next byte will be the end of scan separator (FFh). If the flag is 81h (no), only the end of scan separator is recorded. This format is repeated until the number of scans indicated in bytes 32 and 33 of the header have been recorded. If the value of byte 12 is 81h (no), the remaining data form the intermediate data unless the number of channels for this data set is zero in which case there will be no further data. If byte 20 (# channels of intermediate data) is not zero then the first byte of data will be the low byte of data for the last channel of the intermediate data(Base channel plus # channels). This is followed by the high byte of the last channel and then by the lower channels (low byte then high byte) until the base channel has been recorded.

The next data value depends on byte 11 (scan count flag). If the value of this flag is F2h (yes), the low byte of the scan count will be recorded followed by the high byte. The next byte will be the end of scan separator (FFh). If the flag is 81h (no), only the end of scan separator is recorded. This format is repeated until the number of scans indicated in bytes 30 and 31 of the header have been recorded.

If byte 19 (# channels of 1st scan data) is zero then the first byte of data will be the low byte of data for the last channel of for the intermediate data or the primary data.

3.2 EXAMPLE OF DATA FORMAT

HEADER

```
BYTE  0     = 58h     Year = 88
BYTE  1     = 1Bh     Day  = 27
BYTE  2     = 01h     Month = 1 (January)
BYTE  3     = 00h     Seconds = 0
BYTE  4     = 00h     Minutes = 0
BYTE  5     = 12h     Hours   = 18
BYTE  6     = 00h     Tenths of seconds = 0
BYTE  7     = 3Dh
BYTE  8     = 12h     sample number (123Dh) = 4669
BYTE  9     = 37h
BYTE 10     = 0Dh     channel 16 data(0D37h) = 3383 = 3.12 volts
BYTE 11     = F2h     scan count is in the data
BYTE 12     = F2h     primary data set
BYTE 13     = 00h     Remote event counter A = no events
BYTE 14     = 00h     Remote event counter B = no events
BYTE 15     = F2h     computer controls power to the sensors
BYTE 16     = 01h     Base channel for the first scan data = 1
BYTE 17     = 04h     Base channel for the intermediate data = 4
BYTE 18     = 04h     Base channel for the primary data = 4
BYTE 19     = 03h     3 channels for the first scan data
BYTE 20     = 02h     2 channels for the intermediate data
BYTE 21     = 0Ch     12 channels for the primary data
BYTE 22     = 0Ah     first scan data = 10 scans/min (with byte 25)
BYTE 23     = 1Eh     intermediate data = 30 scans/sec (with byte 26)
BYTE 24     = 02h     primary data = 2 scans/sec (with byte 27)
BYTE 25     = F2h     first scan data = scans/min
BYTE 26     = 81h     intermediate data = scan/sec
BYTE 27     = 81h     primary data = scan/sec
BYTE 28     = 80h
BYTE 29     = 00h     128 scans of first scan data (0080h)
BYTE 30     = 00h
BYTE 31     = 08h     2048 scans of intermediate data (0800h)
BYTE 32     = 00h
BYTE 33     = 10h     4096 scans of primary data (1000h)
```

INTERPRETATION: This data set is an example of a primary data set (byte 12). The start time of data-acquisition occurs on January 27,1988 at 18:01:00.00. Byte 15 indicates that the computer has switched on the power to the sensors. The system will wait one minute before acquiring data to allow time for the sensors to equilibrate. The sample (123Dh) is the 4669th sample collected since the program was started. The remote event counters indicate that none have occurred, so the sample is not in the remote event mode. Channel 16 indicates 3.12 volts. This value gets written whether the channel is used or not. It provides a means of monitoring a signal level (such as the battery voltage) without being repeated throughout the data. The data file

consists of one set of 128 scans (bytes 28 and 29) of channels 1 through 3 (bytes 16 and 19) sampled at a scan rate of 10 scans/minute (bytes 22 and 25). The second set is 4096 scans (bytes 32 and 33) of channels 4 through 15 (bytes 18 and 21) at 2 scans/second (bytes 24 and 27). Byte 11 indicates that the scan count is written into the data. The total time required for the sample is 12.8 minutes for the first scan and 34.1 minutes for the rest of the data. The following are the byte counts (starting with byte 0) for the data.

DATA

BYTE 34	Low byte of channel 3 data (00 - FFh)
BYTE 35	High byte of channel 3 data (00 - 0Fh)
BYTE 36	Low byte of channel 2 data (00 - FFh)
BYTE 37	High byte of channel 2 data (00 - 0Fh)
BYTE 38	Low byte of channel 1 data (00 - FFh)
BYTE 39	High byte of channel 1 data (00 - 0Fh)
BYTE 40	Low byte of scan count (00 - FFh)
BYTE 41	High byte of scan count (00 - FFh)
BYTE 42	Scan separator (FFh)
\ \ \ \	
	this pattern continues for 128 scans of data for channels 1 through 3
\ \ \ \	

BYTE 1184	Low byte of channel 15 data (00 - FFh)
BYTE 1185	High byte of channel 15 data (00 - 0Fh)
BYTE 1186	Low byte of channel 14 data (00 - FFh)
BYTE 1187	High byte of channel 14 data (00 - 0Fh)
BYTE 1188	Low byte of channel 13 data (00 - FFh)
BYTE 1189	High byte of channel 13 data (00 - 0Fh)
BYTE 1190	Low byte of channel 12 data (00 - FFh)
BYTE 1191	High byte of channel 12 data (00 - 0Fh)
BYTE 1192	Low byte of channel 11 data (00 - FFh)
BYTE 1193	High byte of channel 11 data (00 - 0Fh)
BYTE 1194	Low byte of channel 10 data (00 - FFh)
BYTE 1195	High byte of channel 10 data (00 - 0Fh)
BYTE 1196	Low byte of channel 9 data (00 - FFh)
BYTE 1197	High byte of channel 9 data (00 - 0Fh)
BYTE 1198	Low byte of channel 8 data (00 - FFh)
BYTE 1199	High byte of channel 8 data (00 - 0Fh)
BYTE 1200	Low byte of channel 7 data (00 - FFh)
BYTE 1201	High byte of channel 7 data (00 - 0Fh)
BYTE 1202	Low byte of channel 6 data (00 - FFh)
BYTE 1203	High byte of channel 6 data (00 - 0Fh)
BYTE 1204	Low byte of channel 5 data (00 - FFh)
BYTE 1205	High byte of channel 5 data (00 - 0Fh)
BYTE 1206	Low byte of channel 4 data (00 - FFh)
BYTE 1207	High byte of channel 4 data (00 - 0Fh)
BYTE 1208	Low byte of scan count (00 - FFh)
BYTE 1209	High byte of scan count (00-FFh)
BYTE 1210	Scan separator (FFh)

\ \ \ \

this pattern continues for 4096 scans of data
for channels 4 through 15

\ \ \ \

BYTE 111751	Low byte of channel 15 data (00 - FFh)
BYTE 111752	High byte of channel 15 data (00 - 0Fh)
BYTE 111753	Low byte of channel 14 data (00 - FFh)
BYTE 111754	High byte of channel 14 data (00 - 0Fh)
BYTE 111755	Low byte of channel 13 data (00 - FFh)
BYTE 111756	High byte of channel 13 data (00 - 0Fh)
BYTE 111757	Low byte of channel 12 data (00 - FFh)
BYTE 111758	High byte of channel 12 data (00 - 0Fh)
BYTE 111759	Low byte of channel 11 data (00 - FFh)
BYTE 111760	High byte of channel 11 data (00 - 0Fh)
BYTE 111761	Low byte of channel 10 data (00 - FFh)
BYTE 111762	High byte of channel 10 data (00 - 0Fh)
BYTE 111763	Low byte of channel 9 data (00 - FFh)
BYTE 111764	High byte of channel 9 data (00 - 0Fh)
BYTE 111765	Low byte of channel 8 data (00 - FFh)
BYTE 111766	High byte of channel 8 data (00 - 0Fh)
BYTE 111767	Low byte of channel 7 data (00 - FFh)
BYTE 111768	High byte of channel 7 data (00 - 0Fh)
BYTE 111769	Low byte of channel 6 data (00 - FFh)
BYTE 111770	High byte of channel 6 data (00 - 0Fh)
BYTE 111771	Low byte of channel 5 data (00 - FFh)
BYTE 111772	High byte of channel 5 data (00 - 0Fh)
BYTE 111773	Low byte of channel 4 data (00 - FFh)
BYTE 111774	High byte of channel 4 data (00 - 0Fh)
BYTE 111775	Low byte of scan count (00 - FFh)
BYTE 111776	High byte of scan count (00-FFh)
BYTE 111777	Scan separator (FFh)

End of data

FIGURES

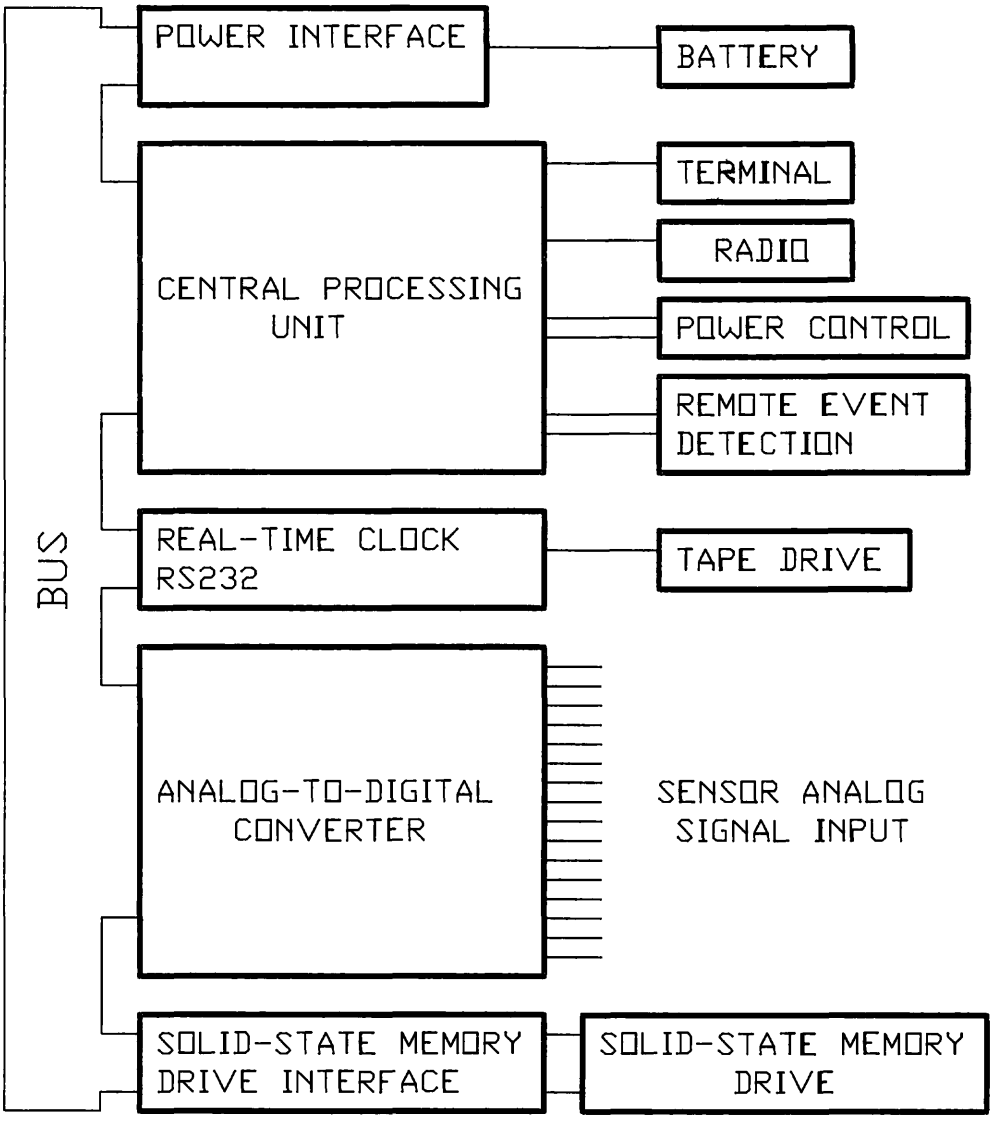

Figure 1. Block diagram of the data-acquisition system

TELEMETRY COMPUTER HOUSING

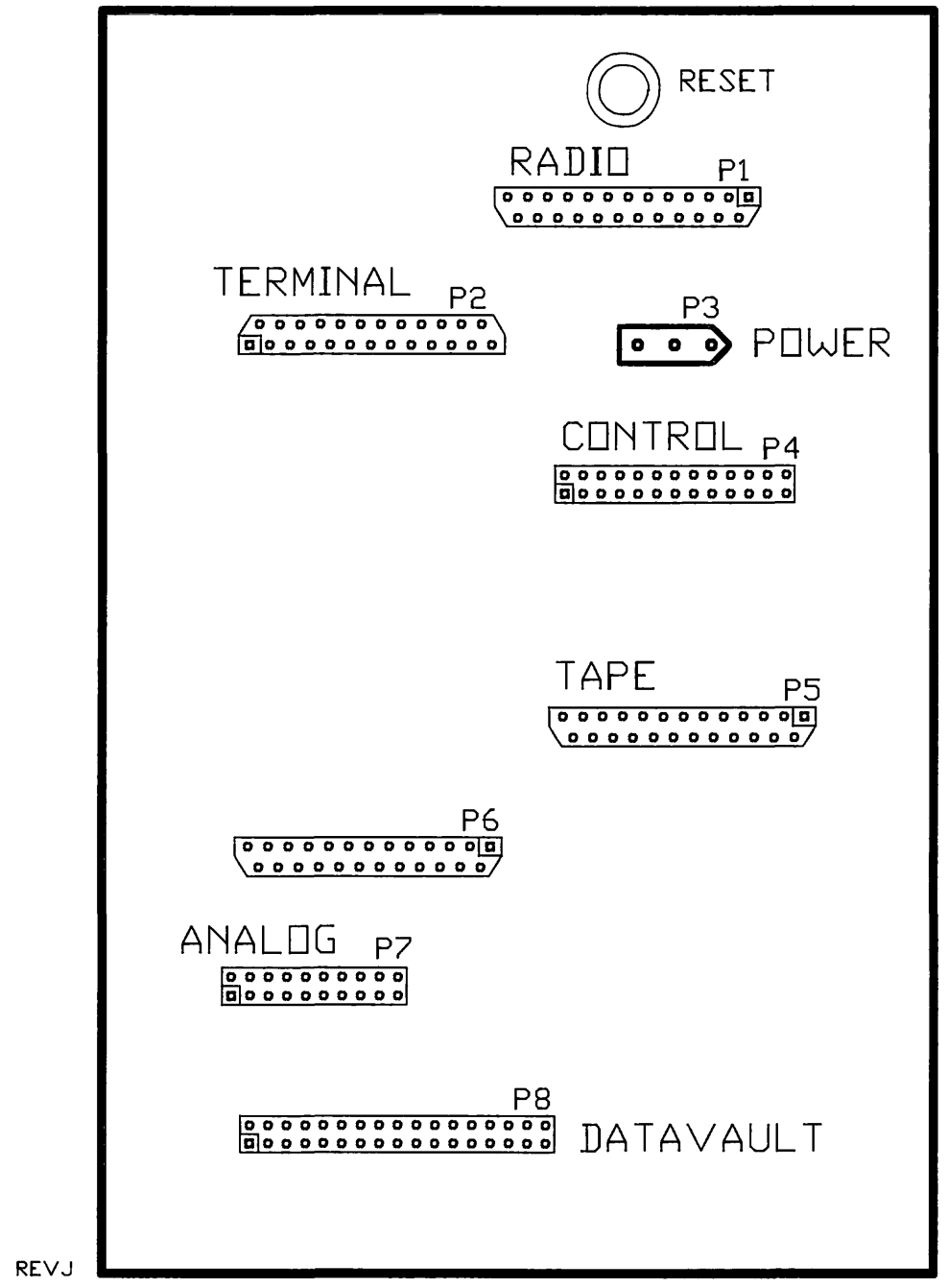

Figure 2. Computer connector locations

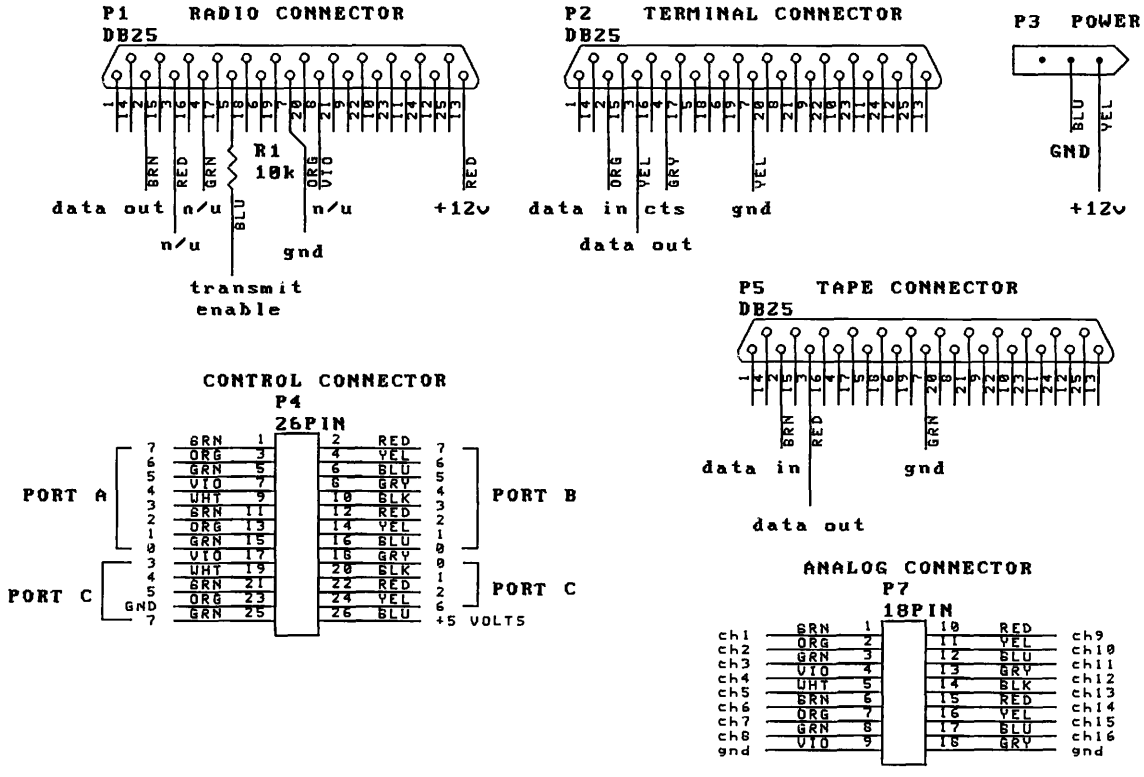

Figure 3. Computer Connector Wiring

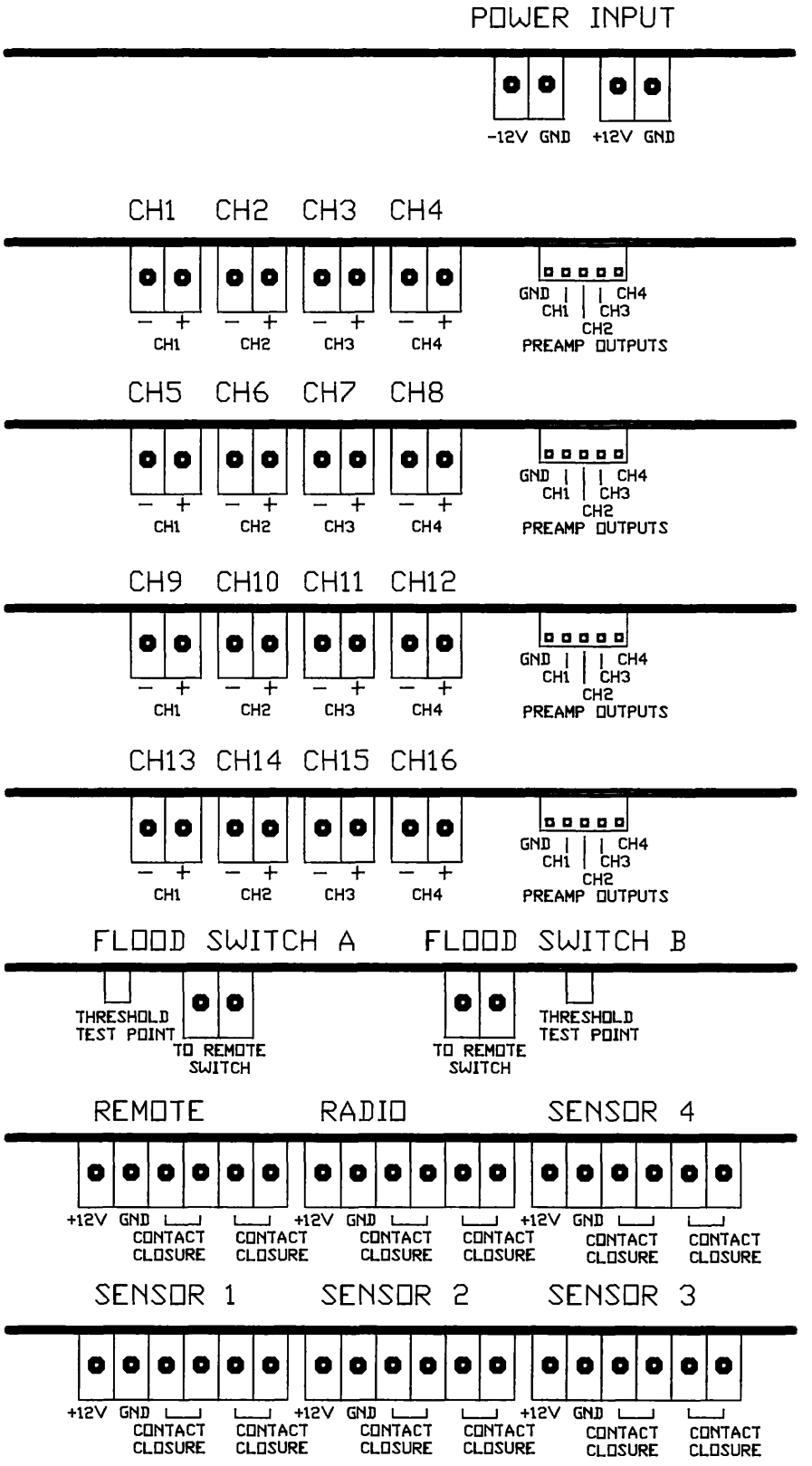

Figure 4. Interface Board Connector Locations

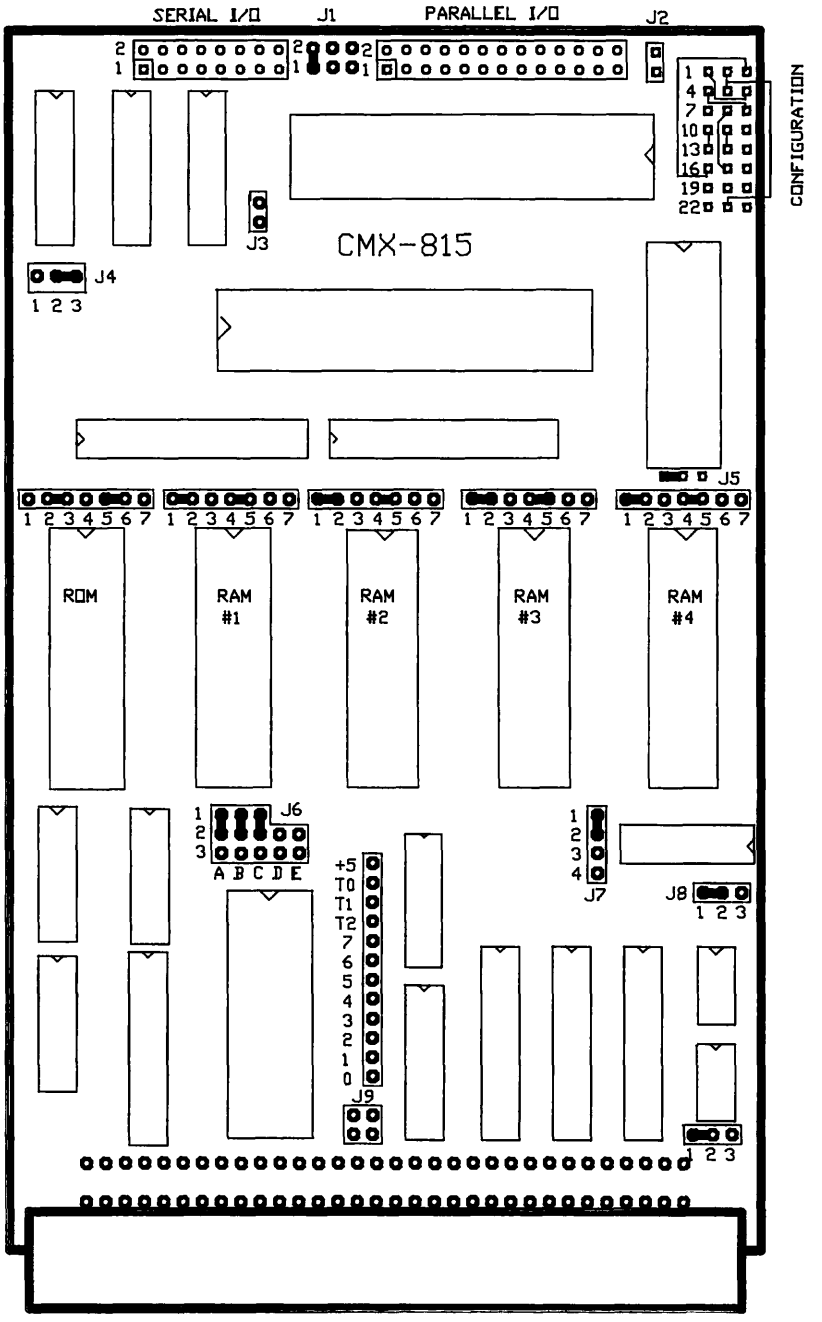

Figure 5. Controller Board Jumper Locations

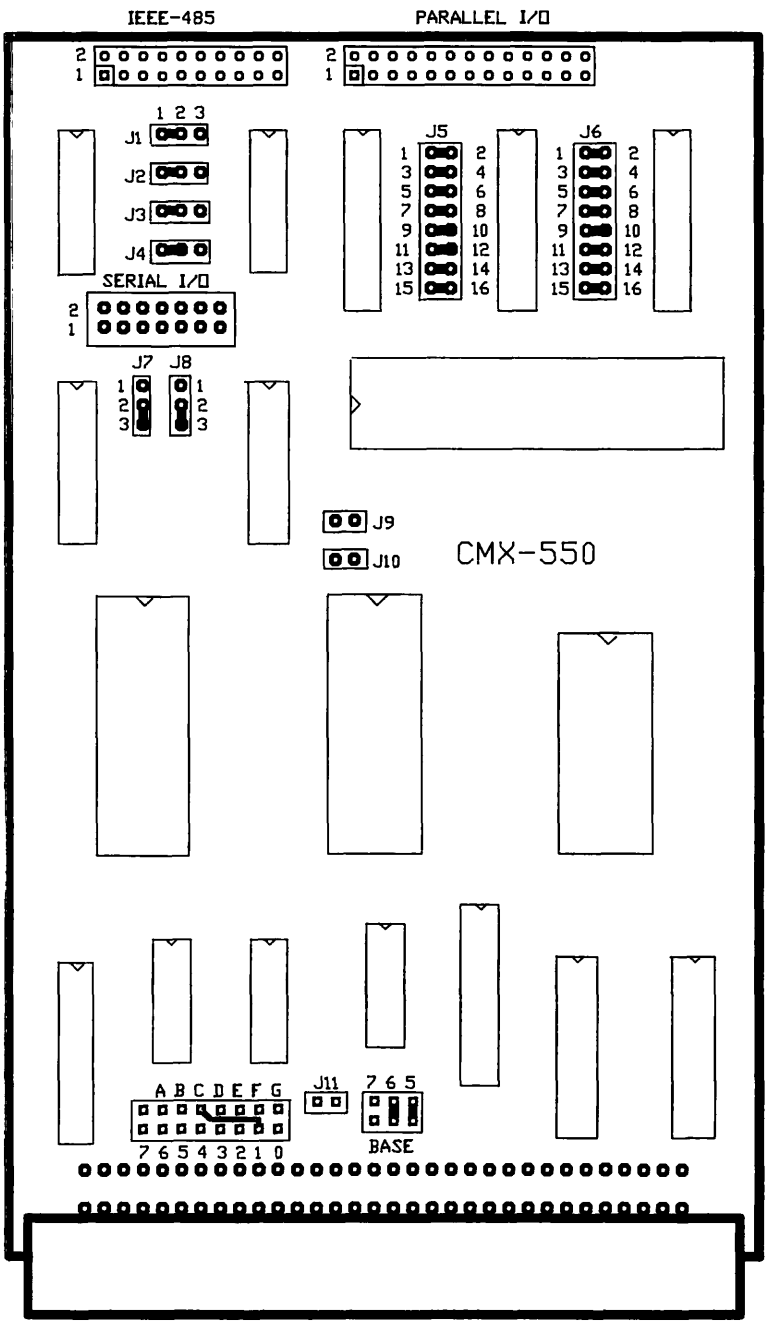

Figure 6. Multi-Function Board Jumper Locations

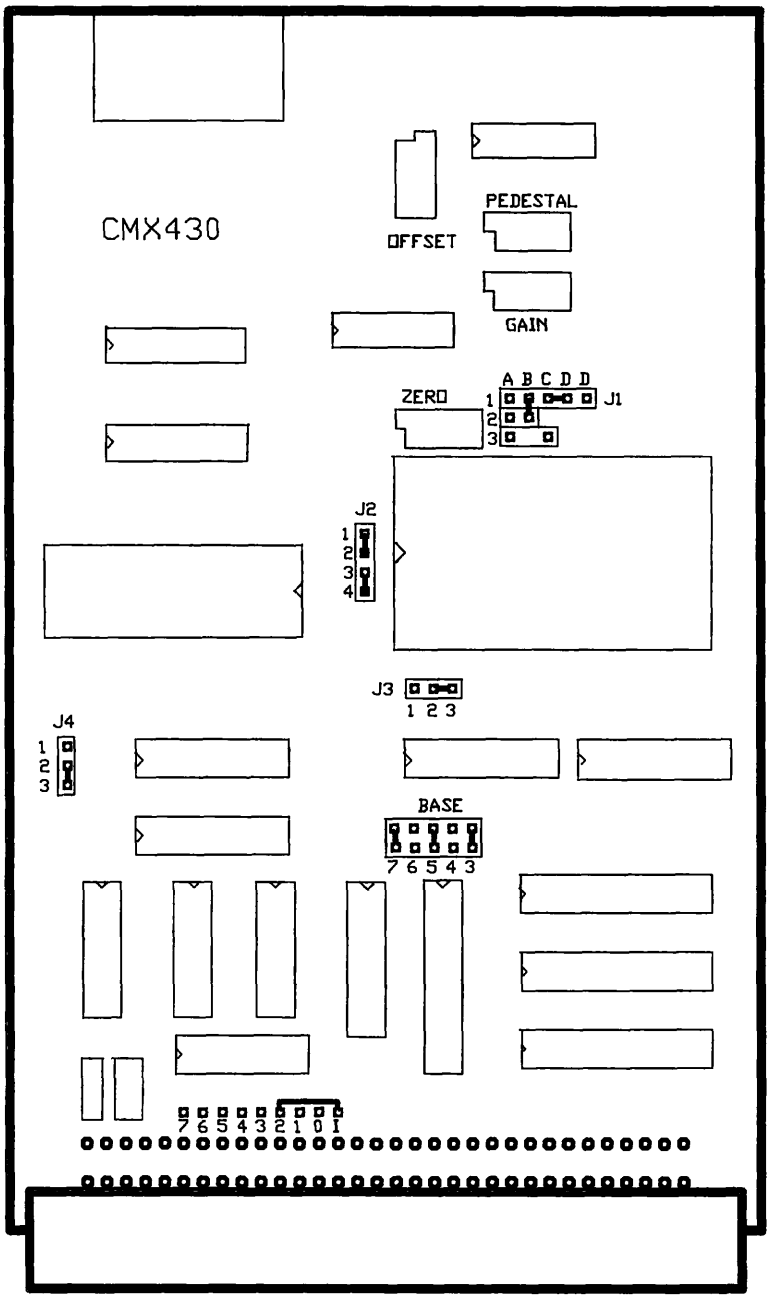

Figure 7. Analog-to-Digital Converter Board Jumper Locations

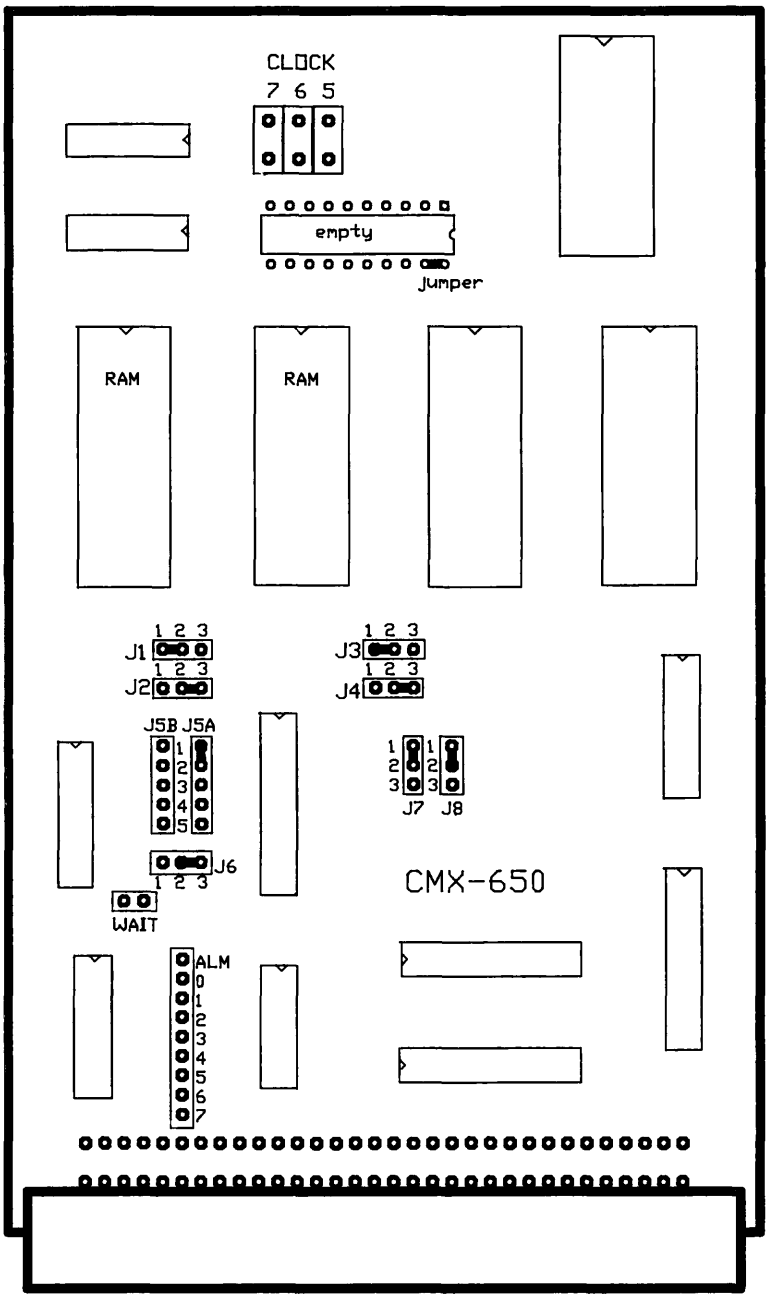

Figure 8. Memory Board Jumper Locations

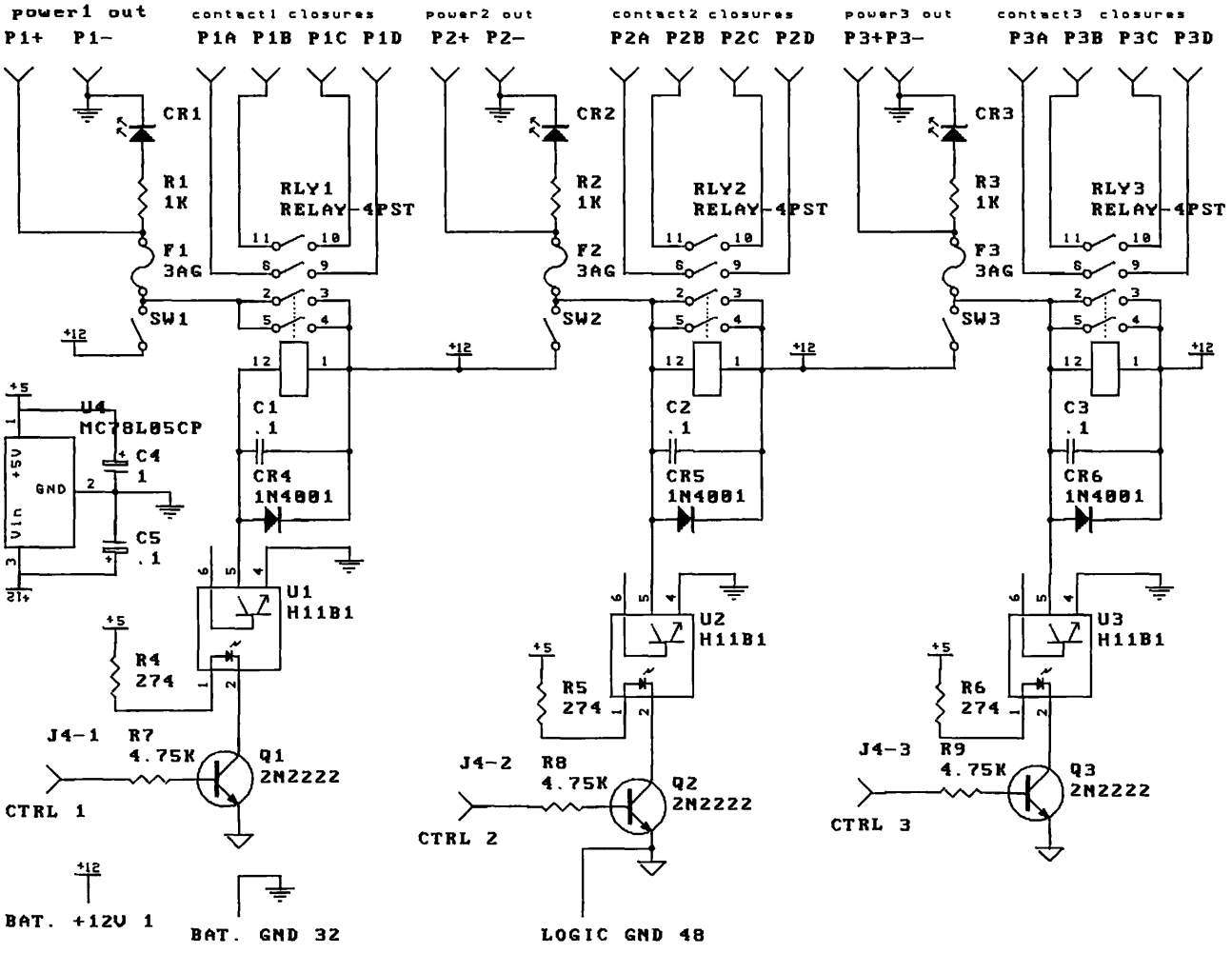

Figure 9. Power Control Board

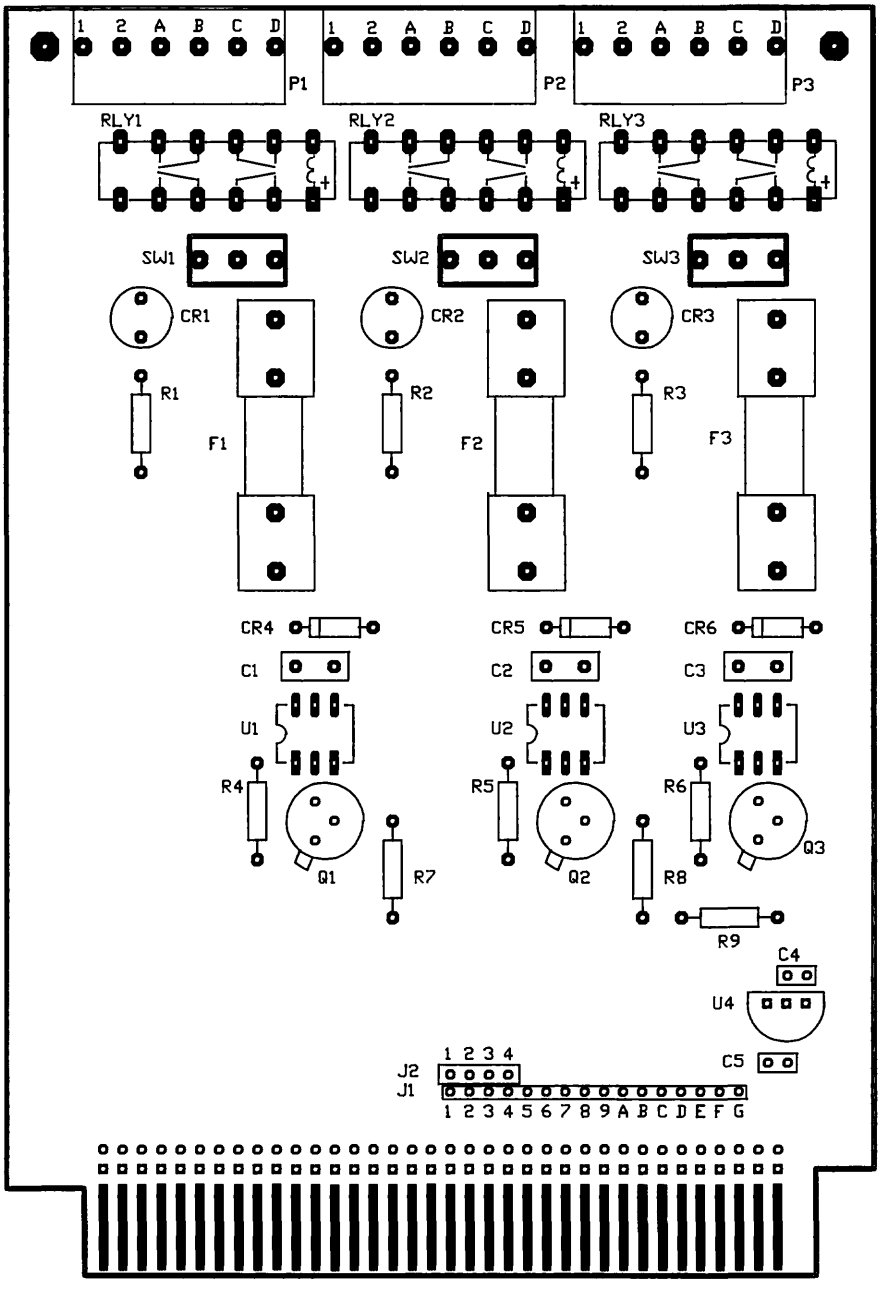

Figure 10. Power Control Board Layout

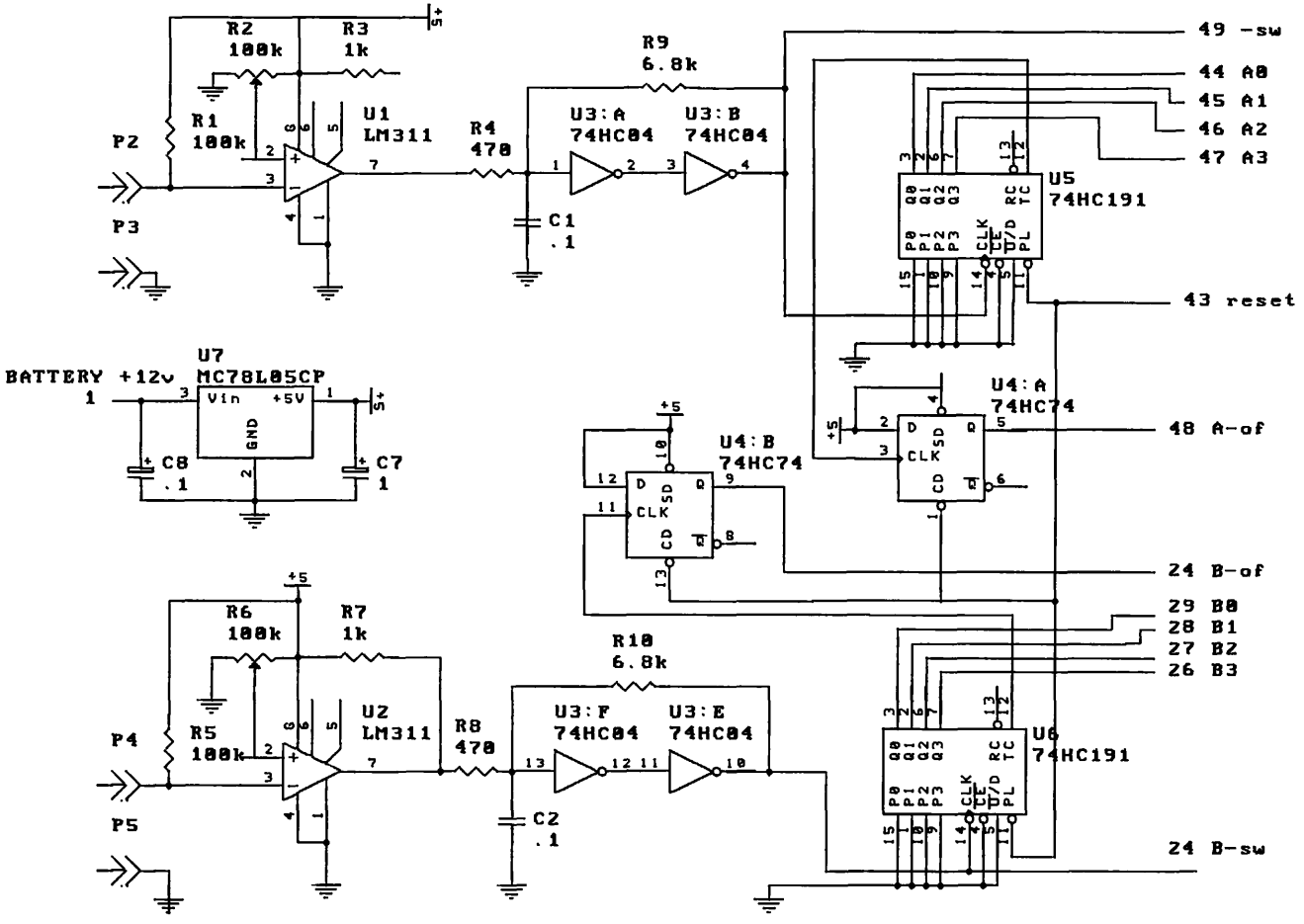

Figure 11. Event Detection Board

skm

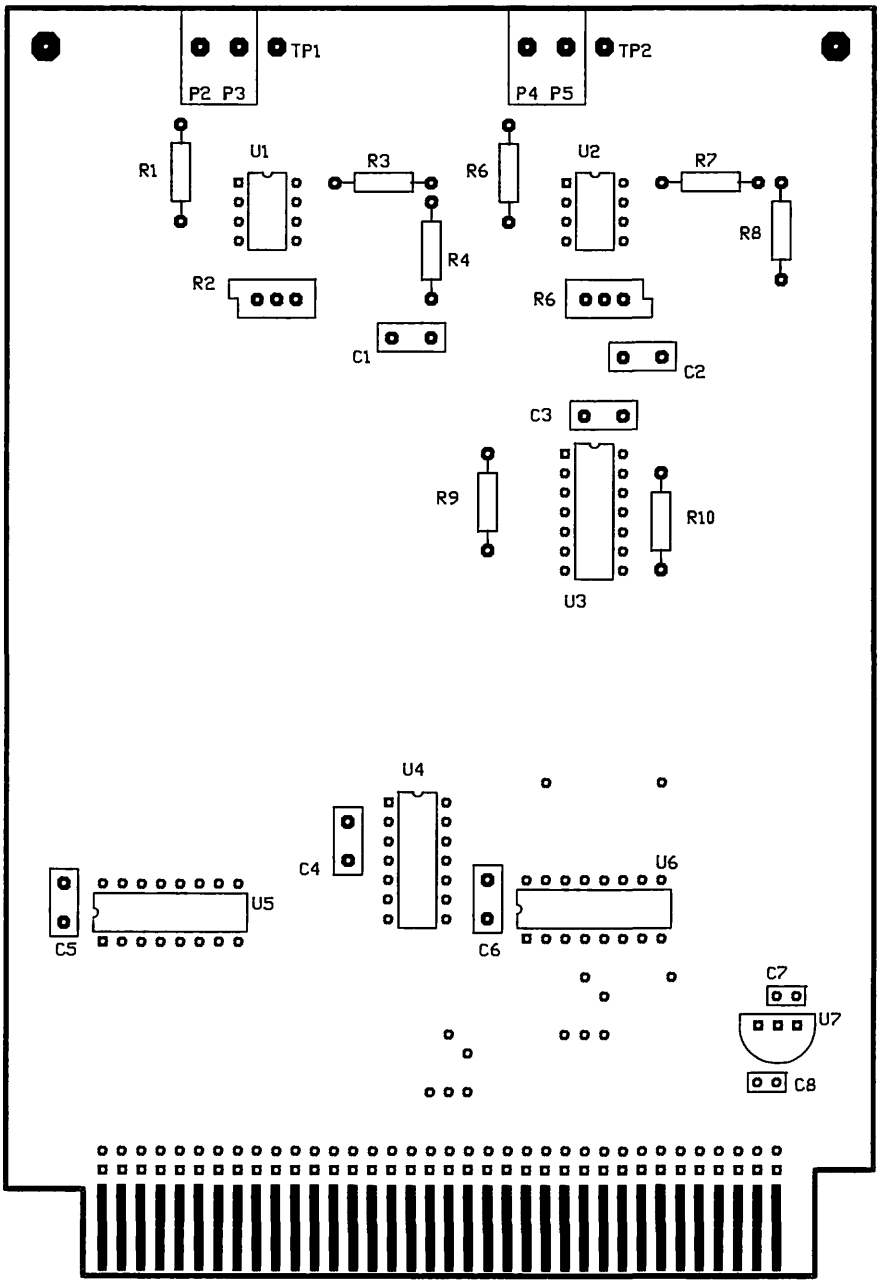

Figure 12. Event Detection Board Layout

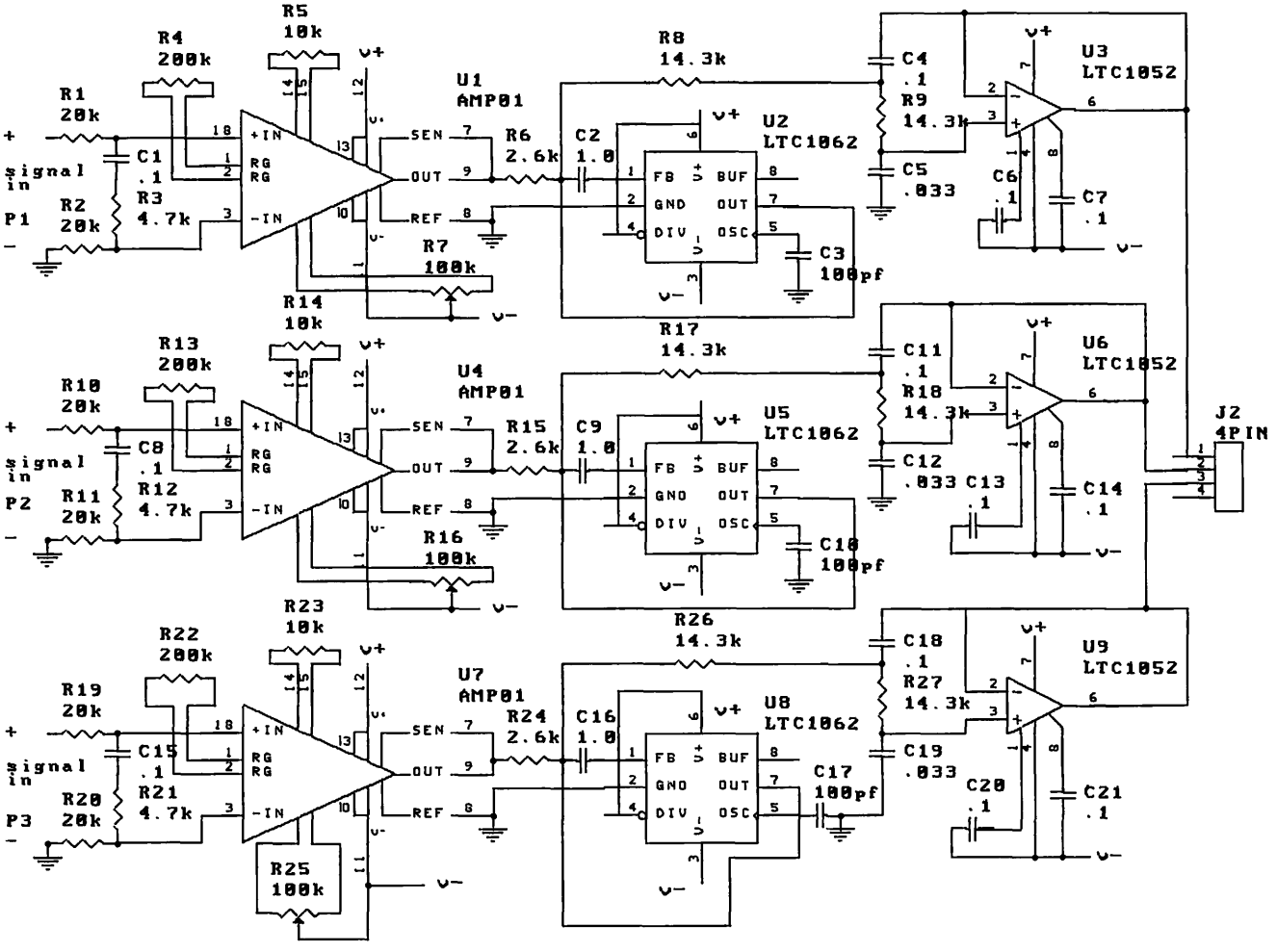

Figure 13. Analog Signal Conditioning —Part A

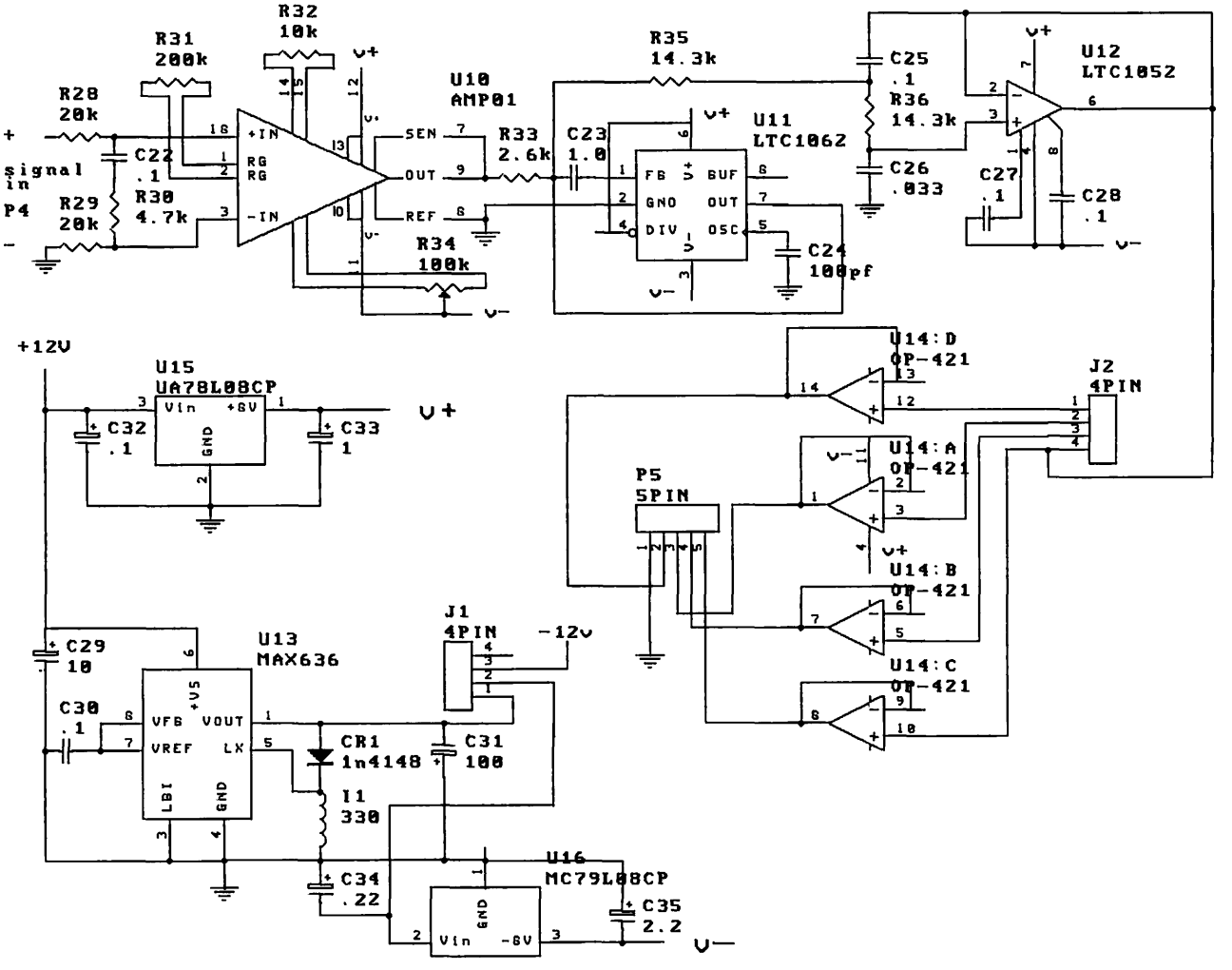

Figure 14. Analog Signal Conditioning —Part B

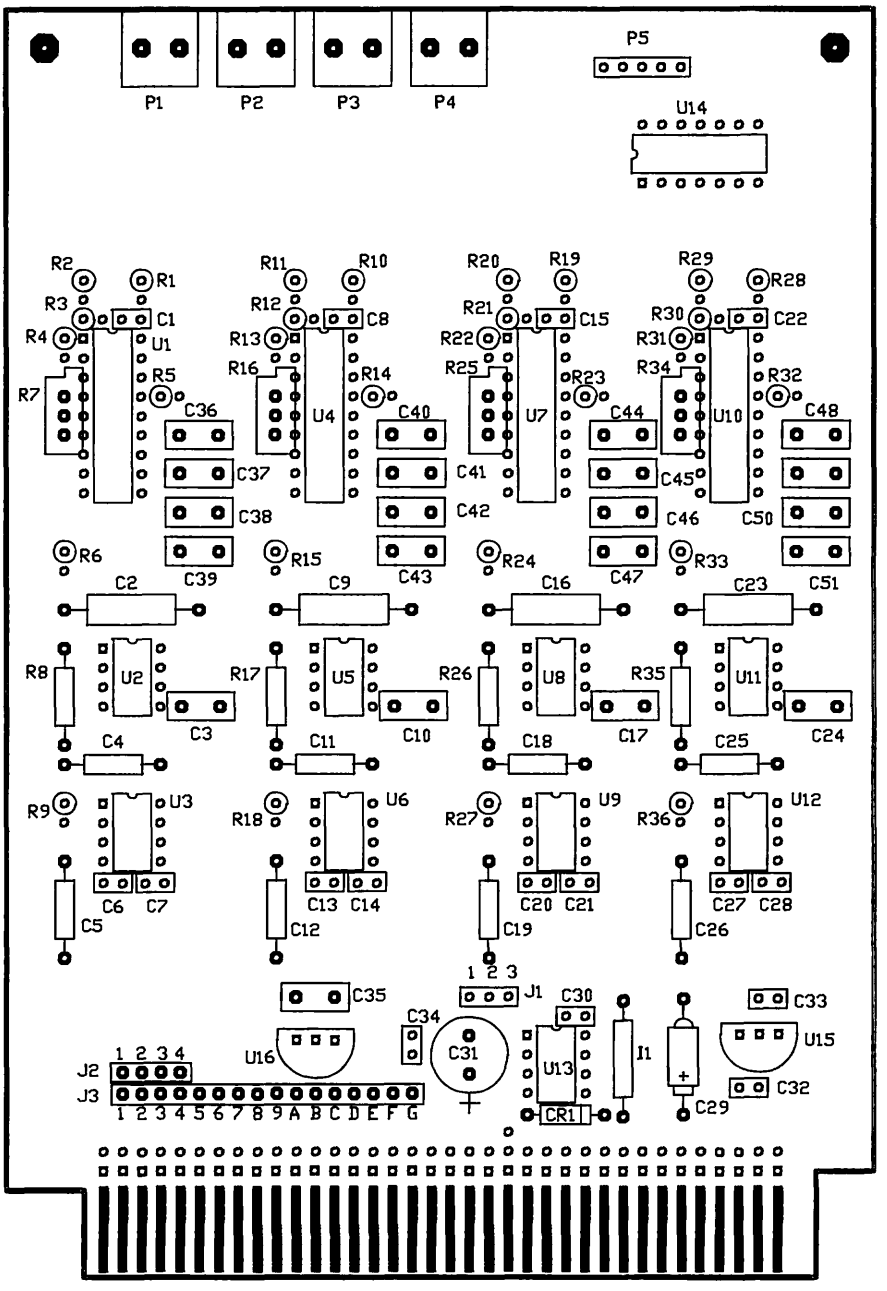

Figure 15. Analog Signal Conditioning Board Layout

(p. 56 follows)

APPENDIX I

SYSTEM SPECIFICATIONS

Analog-to-Digital Converter

number of channels	1	to 16 (single-ended)
data word size	12	bits
resolution	2.44	millivolts
conversion time (minimum)	500	microseconds (1 channel)
	16	milliseconds (16 channels)
analog input range	+5	to -5 volts DC
filter frequency	61	hertz
filter rolloff	30	dB/octave

External Control

remote event	16	input lines
	1	output (for reset)
power control	7	output lines

Memory

total	196,608	bytes
used by program	32,768	bytes
available for data	143,355	bytes

Real-Time Clock

resolution	.1	seconds
alarm modes		programmed time
	1	hour
	1	minute
	1	second

Digital Tape Drive

capacity	20	megabytes
interface		RS232
baud-rate	9,600	
data type		ASCII

Solid-State Memory

capacity	1	megabyte
interface		propriety
data type		binary in a MSDOS compatible file

Radio Transmitter

frequency	406.475	megahertz
		FCC authorized
interface		RS232
baud-rate	1,200	
RF output power	.5	watts (10 watts with booster)

Power consumption

+12v input	operating	idle	
computer system	1.8	1.0	watt
preamplifier/filter	2.5	2.5	watt
power control	0.7	0.1	watt
radio transmitter	3.0	0.1	watt
solid-state memory	1.2	1.2	watt
total	9.2	4.9	watt

(does not include tape drive)

APPENDIX II

SOFTWARE LISTING

```
;                          TEL.MAC - REV J
;
;-------------------------------------------------------------
;Original code written by gkm - 7/25/86
;REV B code by gkm -  7/31/86 - to correct bugs.
;REV C code by gkm - 12/23/86 - upgrade to CMX-815 CPU board.
;REV D code by gkm -  2/18/87 - upgrade to include 16 channels
;       of data into 4 banks of memory.
;REV E code by gkm -  3/15/87 - change data-acquisition from a
;       polled mode to an interrupt driven system.  The startup
;       code was rewritten to support automatic startup using
;       battery-backed RAM.
;REV F code by gkm -  4/13/87 - upgrade to include software
;       tests for troubleshooting.
;REV G code by gkm -  7/ 8/87 - upgrades for the following
;     1. Changed end-of-scan marker to OFFH
;     2. Included scan number in data
;     3. Added a flag for primary data to header
;     4. Added channel 16 voltage to header
;     5. Moved header and 128 scan data to accommodate extra
;        memory needed for scan counts in data
;     6. Install intermediate data mode
;     7. Send signals for external power control
;REV H code by gkm -  7/15/87 - modification of REVG code for
;     1. Turn on power for A-D test
;     2. Write data to a tape drive
;REV I code by gkm - complete rewrite
;     1. remote event mode now has it own sample rate
;     2. There is now an option to acquire ch16 in the data
;     3. Remote mode now reads two counters into the header
;     4. Options have been included to send data via radio
;          a) all the time
;          b) only during remote events
;          c) at no time
;     5. Options have been included to send data via tape
;          a) all the time
;          b) only during remote events
;          c) at no time
;     6. Mode options have been changed to simply select if
;        you want intermediate data sets between primary data sets
;     7. It is now an option to add an additional scan set of
;        data at the beginning of each data set
;     8. With the 1st scan the following options can be set
;          a) number of channels (starts at 1)
;          b) number of scans
;          c) scan rate
```

```
;    9. With the intermediate data the following options can be set
;         a) number of channels (starts at end of 1st scan)
;         b) number of scans
;         c) scan rate
;         d) number of samples per day or hour
;    10.With the primary data the following options can be set
;         a) number of channels (starts at end of 1st scan)
;         b) number of scans
;         c) scan rate
;         d) number of samples per day or hour
;    11.Checking is now done to ensure that
;         a) the number of channels does not exceed 16
;         b) the data size does not exceed available memory
;         c) the time required for sampling does not exceed
;            the sample rate
;    12.The program now displays the entered parameters and
;       asks if they are correct
;    13.Power control routines have been rewritten
;    14.The header has been expanded to 32 bytes to allow room
;       for scan #'s, channel #'s,etc.
;    15.All of the data has been switched so that it is recorded
;       Low-byte first and then High-byte
; REV J code by gkm 10/04/1988 - added following functions:
;    1. Added solid state memory drivers.
;    2. Added extended memory functions
;    3. Altered remote event functions to prevent the system
;       from staying in remote mode after a power failure.
;
;-----------------------------------------------------------------
```

```
;           EQUATES USED FOR THE CMX815 COMPUTER BOARD
;
; code written by gkm 7/31/86
;
;--- CONTROLLER BOARD PORT LOCATIONS ---
;
;                        - I/O PORT LOCATIONS
;
CPUBASE         equ      00h            ;Controller base address
CNTLA0          equ     CPUBASE+00h     ;Serial #0 control A reg.
CNTLA1          equ     CPUBASE+01h     ;Serial #1 control A reg.
CNTLB0          equ     CPUBASE+02h     ;Serial #0 control B reg.
CNTLB1          equ     CPUBASE+03h     ;Serial #1 control B reg.
STAT0           equ     CPUBASE+04h     ;Serial #0 status reg.
STAT1           equ     CPUBASE+05h     ;Serial #1 status reg.
TDR0            equ     CPUBASE+06h     ;Serial #0 Tx reg.
TDR1            equ     CPUBASE+07h     ;Serial #1 Tx reg.
TSR0            equ     CPUBASE+08h     ;Serial #0 Rx reg.
TSR1            equ     CPUBASE+09h     ;Serial #1 Rx reg.
CNTR            equ     CPUBASE+0Ah     ;CSI/O control reg.
TRDR            equ     CPUBASE+0Bh     ;CSI/O Tx/Rx reg.
TMDR0L          equ     CPUBASE+0Ch     ;Timer 0 low-byte data
TMDR0H          equ     CPUBASE+0Dh     ;Timer 0 high-byte data
RLD0L           equ     CPUBASE+0Eh     ;Timer 0 low-byte reload
RLD0H           equ     CPUBASE+0Fh     ;Timer 0 high-byte reload
TCR             equ     CPUBASE+10h     ;Timer control reg.
TMDR1L          equ     CPUBASE+14h     ;Timer 1 low-byte data
TMDR1H          equ     CPUBASE+15h     ;Timer 1 high-byte data
RLD1L           equ     CPUBASE+16h     ;Timer 1 low-byte reload
RLD1H           equ     CPUBASE+17h     ;Timer 1 high-byte reload
SAR0L           equ     CPUBASE+20h     ;DMA #0 low-byte source
SAR0H           equ     CPUBASE+21h     ;DMA #0 high-byte source
SAR0B           equ     CPUBASE+22h     ;DMA #0 bank source
DAR0L           equ     CPUBASE+23h     ;DMA #0 low-byte destination
DAR0H           equ     CPUBASE+24h     ;DMA #0 high-byte destination
DAR0B           equ     CPUBASE+25h     ;DMA #0 bank destination
BCR0L           equ     CPUBASE+26h     ;DMA #0 low-byte count
BCR0H           equ     CPUBASE+27h     ;DMA #0 high byte count
MAR1L           equ     CPUBASE+28h     ;DMA #1 low-byte memory addr.
MAR1H           equ     CPUBASE+29h     ;DMA #1 high-byte memory addr.
MAR1B           equ     CPUBASE+2Ah     ;DMA #1 bank memory address
IAR1L           equ     CPUBASE+2Bh     ;DMA #1 low-byte I/O address
IAR1H           equ     CPUBASE+2Ch     ;DMA #1 high-byte I/O addr.
BCR1L           equ     CPUBASE+2Eh     ;DMA #1 low-byte count
BCR1H           equ     CPUBASE+2Fh     ;DMA #1 high-byte count
DSTAT           equ     CPUBASE+30h     ;DMA status reg.
DMODE           equ     CPUBASE+31h     ;DMA mode reg.
DCNTL           equ     CPUBASE+32h     ;DMA control reg.
IL              equ     CPUBASE+33h     ;Interrupt vector reg.
ITC             equ     CPUBASE+34h     ;INT/TRAP control reg.
RCR             equ     CPUBASE+36h     ;Refresh control reg.
CBR             equ     CPUBASE+38h     ;MMU common base reg.
BBR             equ     CPUBASE+39h     ;MMU bank base reg.
CBAR            equ     CPUBASE+3Ah     ;MMU common/bank area reg.
ICR             equ     CPUBASE+3Fh     ;I/O control reg.
```

```
;
;                       - PARALLEL PORT LOCATIONS
;
PIBASE          equ     0FCh            ;Base address
PADATA          equ     PIBASE+00h      ;Port A data reg.
PBDATA          equ     PIBASE+01h      ;Port B data reg.
PCDATA          equ     PIBASE+02h      ;Port C data reg.
PICNTL          equ     PIBASE+03h      ;Control reg.
;
;                       - TIMER PORT LOCATIONS
;
CTRGC           equ     0EFh    ;Counter gate control (DB1)
CTR0            equ     0F8h    ;Counter 0 reg.
CTR1            equ     0F9h    ;Counter 1 reg.
CTR2            equ     0FAh    ;Counter 2 reg.
CTRC            equ     0FBh    ;Counter control reg.
;
;                       - INTERRUPT PORT LOCATIONS
;
INTPRT          equ     0F4h    ;Interrupt mask reg.
INTPRTA         equ     0F5h    ;Interrupt command reg.
;
```

```
;      EQUATES FOR THE CMX550 MULTI-FUNCTION BOARD
;
; code by gkm 7/17/87
;
;--- CMX-550 BOARD PORT LOCATIONS ---
;
CMX550          equ     080h            ;Board base address
RTCBASE         equ     CMX550+00h      ;Real-time-clock address
PPCBASE         equ     CMX550+01Ch     ;Parallel port base
DUART1          equ     CMX550+20h      ;Dual UART #1 base
DUART2          equ     CMX550+30h      ;Dual UART #2 base
;
;                               - REAL-TIME-CLOCK PORT LOCATIONS
;
SEC.1           equ     RTCBASE+00h     ;Tenths of seconds
HR              equ     RTCBASE+01h     ;Hour
MIN             equ     RTCBASE+02h     ;Minutes
SEC             equ     RTCBASE+03h     ;Seconds
MTH             equ     RTCBASE+04h     ;Month
DAY             equ     RTCBASE+05h     ;Day
YEAR            equ     RTCBASE+06h     ;Year
DAYWK           equ     RTCBASE+07h     ;Day of week
ASEC.1          equ     RTCBASE+08h     ;Alarm-tenths of sec.
AHR             equ     RTCBASE+09h     ;Alarm-hour
AMIN            equ     RTCBASE+0Ah     ;Alarm-minutes
ASEC            equ     RTCBASE+0Bh     ;Alarm-seconds
AMTH            equ     RTCBASE+0Ch     ;Alarm-month
ADAY            equ     RTCBASE+0Dh     ;Alarm-day
AYEAR           equ     RTCBASE+0Eh     ;Alarm-year
ADAYWK          equ     RTCBASE+0Fh     ;Alarm-day of week
RTCSTAT         equ     RTCBASE+10h     ;Interrupt status reg.
RTCCMD          equ     RTCBASE+11h     ;Command register
;
;                               - PARALLEL PORT LOCATIONS
;
PORTA           equ     PPCBASE+00h     ;Port A
PORTB           equ     PPCBASE+01h     ;Port B
PORTC           equ     PPCBASE+02h     ;Port C
CTPRT           equ     PPCBASE+03h     ;Control port
;
;                               - SERIAL PORT LOCATIONS
;
SP3             equ     DUART1+00h      ;Serial port 3
SP4             equ     DUART1+08h      ;Serial port 4
SP5             equ     DUART2+00h      ;Serial port 5
SP6             equ     DUART2+08h      ;Serial port 6
```

```
;
;                                   - REGISTER OFFSETS FOR EACH
;
MR1             equ     00h             ;Mode register 1
MR2             equ     00h             ;Mode register 2
SR              equ     01h             ;Status register
CSR             equ     01h             ;Clock selct register
DCR             equ     02h             ;Duart command register
RHR             equ     03h             ;Rx holding register
THR             equ     03h             ;Tx holding register
;
;                                   - OFFSETS FOR EACH SERIAL PORT
;
ISRM            equ     02h             ;Int. status reg. masked
IPCR            equ     04h             ;Input port change reg.
ACR             equ     04h             ;Aux. control register
ISRU            equ     05h             ;Int. status reg. umasked
IMR             equ     05h             ;Int. mask register
CTUR            equ     06h             ;Counter upper reg.
CTLR            equ     07h             ;Counter lower reg.
IVR             equ     0Ch             ;Interrupt vector reg.
STRT            equ     0Eh             ;Start counter
STOP            equ     0Fh             ;Stop counter
SOPB            equ     0Eh             ;Set output port bits
ROPB            equ     0Fh             ;Reset output port bits
```

```
;          EQUATES FOR THE SOLID-STATE MEMORY STORAGE UNIT
;
;code by gkm 8/17/88
;
; DATAVAULT CONTROLLER I/O ADDRESS MAP
;_____________________________________________________________________
;
DV_BASE         equ     060h            ;Controller base address
;
;                               - DMA PORT LOCATIONS
;
DMA_A0          equ     DV_BASE+00h     ;DMA CH0 address reg.
DMA_C0          equ     DV_BASE+01h     ;DMA CH0 count reg.
DMA_A1          equ     DV_BASE+02h     ;DMA CH1 address reg.
DMA_C1          equ     DV_BASE+03h     ;DMA CH1 count reg.
DMA_A2          equ     DV_BASE+04h     ;DMA CH2 address reg.
DMA_C2          equ     DV_BASE+05h     ;DMA CH2 count reg.
DMA_A3          equ     DV_BASE+06h     ;DMA CH3 address reg.
DMA_C3          equ     DV_BASE+07h     ;DMA CH 3 count reg.
DMA_CMD         equ     DV_BASE+08h     ;DMA command reg.
DMA_RQ          equ     DV_BASE+09h     ;DMA request reg.
DMA_MOD         equ     DV_BASE+0Bh     ;DMA mode reg.
DMA_CFF         equ     DV_BASE+0Ch     ;DMA clear byte flip flop
DMA_CLR         equ     DV_BASE+0Dh     ;DMA master clear
;
PORT_A          equ     DV_BASE+010h    ;82C55 port A
PORT_B          equ     DV_BASE+011h    ;82C55 port B
PORT_C          equ     DV_BASE+012h    ;82C55 port C
CTRL_PORT       equ     DV_BASE+013h    ;82C55 control port
;
CRC_RESET       equ     DV_BASE+014h
CRC_LO          equ     DV_BASE+015h    ;CRC lo_byte data port
CRC_HI          equ     DV_BASE+016h    ;CRC hi-byte data port
CFG_PORT        equ     DV_BASE+017h    ;Configuration port
DATA_PORT       equ     DV_BASE+018h    ;Data port
HADDR           equ     DV_BASE+019h    ;DMA hi address byte
UNIT_SEL        equ     DV_BASE+01Ah    ;Unit select port
;
;                               - PORT C EQUATES
;
DV_PWR          equ     01h     ;Cartridge power control
DV_CRCMEM       equ     04h     ;CRC memory enable
DV_LED          equ     08h     ;LED control
DV_WRENB        equ     10h     ;Write protect
DV_OPEN         equ     20h     ;Door open
DV_LOPEN        equ     40h     ;Latched door open
DV_POWOFF       equ     80h     ;Power off
DV_NOCART       equ     80h     ;No cartridge present
;
;                               - DATAVAULT COMMANDS
;
DV_READ         equ     00h     ;Read a sector
DV_WRITE        equ     01h     ;Write a sector
DV_SIZE         equ     02h     ;Cartridge size
DV_WFIL         equ     03h     ;Write a DOS file
```

```
;
;                                   - RETURNED STATUSES
;
BAD_CMD         equ     01h     ;Bad command
WRT_PROT        equ     03h     ;Write protested cartridge
REC_NO_FND      equ     04h     ;Could not find requested sector
MEDIA_CHNG      equ     06h     ;Media changed
BAD_CRC         equ     10h     ;Bad CRC
TIME_OUT        equ     80h     ;Failed to respond
;
```

```
;
; EQUATES FOR ALL BOARDS
;-----------------------------------------------------------------
;
;--- GENERAL EQUATES ---
;
TRUE            equ  0F2h
FALSE           equ  081h
;
;---ASCII EQUATES ---
;
BS              equ  08h            ;Back space
CR              equ  0Dh            ;Carriage return
DEL             equ  7Fh            ;Delete
LF              equ  0Ah            ;Line feed
SPACE           equ  20h            ;Space
;
;--- EXTERNAL CONTROL COMMANDS ---
;
;                           - REMOTE EVENT CONTROL WORDS
;
REMCLR          equ  06h            ;Clear the remote counters
REMEN           equ  07h            ;Enable remote counters
REMPROFF        equ  0Ah            ;Disable remote power
REMPRON         equ  0Bh            ;Enable remote power
;
;                           - TAPE AND RADIO(RF) CONTROL WORDS
;
TAPOFF          equ  0Ch            ;Disable tape power
TAPON           equ  0Dh            ;Enable tape power
RFOFF           equ  0Eh            ;Disable RF transmitter
RFON            equ  0Fh            ;Enable RF transmitter
;
;                           - TAPE COMMANDS
;
TRESET          equ  018h           ;Reset
TWRITE          equ  012h           ;Write data
TREAD           equ  011h           ;Read data
TSTP            equ  004h           ;Stop read write
TSTAT           equ  005h           ;Get tape status
TBOF            equ  008h           ;Skip to beginning of file
TEOF            equ  009h           ;Skip to end of file
TTRACK          equ  00Bh           ;Increment track
TMARK           equ  01Ch           ;Write tape mark
TWORD           equ  010h           ;2 command control word
TUNLOAD         equ  031h           ;Unload tape
TRDBLK          equ  032h           ;Read block
TRDFIL          equ  033h           ;Read file
TREWIND         equ  034h           ;Rewind
TEOD            equ  035h           ;Skip to end of data
TSKPFBLK        equ  036h           ;Skip forward 1 block
TSKPBBLK        equ  037h           ;Skip backward 1 block
```

```
;
;--- ANALOG-TO-DIGITAL (A-D) BOARD PORT LOCATIONS ---
;
ADPORT          equ  050h           ;Base address
ADLO            equ  ADPORT+00h     ;Low-byte data
ADHI            equ  ADPORT+01h     ;High-byte data
ADSTART         equ  ADPORT+02h     ;Start A-D
ADCG            equ  ADPORT+03h     ;Channel/gain
MUXEN           equ  ADPORT+04h     ;Enable multiplexer
MUXDIS          equ  ADPORT+05h     ;Disable multiplexer
ADINT           equ  ADPORT+06h     ;Clear interrupt
;
;--- MEMORY LOCATIONS ---
;
SCRATCH         equ  0F000h         ;Base for common area
;
;                        - HEADER DATA
;
DATAST          equ  0EFFFh         ;Start of data
HEADER          equ  SCRATCH        ;Base of header data
SCAN3           equ  HEADER+00h     ;Scan # for primary data
SCAN2           equ  HEADER+02h     ;Scan # for int. data
SCAN1           equ  HEADER+04h     ;Scan # for 1st scan data
SRFLGF          equ  HEADER+06h     ;Sam. time base flag for primary data
SRFLGI          equ  HEADER+07h     ;Sam. time base flag for int. data
SRFLGA          equ  HEADER+08h     ;Sam. time base flag for 1st scan
SRATEF          equ  HEADER+09h     ;Scan rate for primary data
SRATEI          equ  HEADER+0Ah     ;Scan rate for int. data
SRATEA          equ  HEADER+0Bh     ;Scan rate for 1st scan data
CHNLF           equ  HEADER+0Ch     ;# channels for primary data
CHNLI           equ  HEADER+0Dh     ;# channels for int. data
CHNLA           equ  HEADER+0Eh     ;# channels for 1st scan data
FCHB            equ  HEADER+0Fh     ;Base channel for primary data
ICHB            equ  HEADER+010h    ;Base channel for int. data
ACHB            equ  HEADER+011h    ;Base channel for 1st scan data
PWRFLG          equ  HEADER+012h    ;Sensor power flag
RCTB            equ  HEADER+013h    ;Remote event count B
RCTA            equ  HEADER+014h    ;Remote event count A
RCDFLG          equ  HEADER+015h    ;Primary data record flag
SCNFLG          equ  HEADER+016h    ;Scan count flag
CH16L           equ  HEADER+017h    ;Low byte of CH16
CH16H           equ  HEADER+018h    ;High byte of CH16
SAMNO           equ  HEADER+019h    ;Sample number
MTMEM           equ  HEADER+01Bh    ;Clock memory storage
MSEC.1          equ  HEADER+01Bh    ;Tenths of seconds
MHR             equ  HEADER+01Ch    ;Hour
MMIN            equ  HEADER+01Dh    ;Minutes
MSEC            equ  HEADER+01Eh    ;Seconds
MMTH            equ  HEADER+01Fh    ;Month
MDAY            equ  HEADER+020h    ;Day
MYEAR           equ  HEADER+021h    ;Year
HEADST          equ  HEADER+021h    ;End of header
```

```
;
;                          - ENTRY VARIABLES
;
SCANA           equ  HEADST+010h      ;No. scans 1st scan data
SCANI           equ  HEADST+012h      ;No. scans int. data
SCANF           equ  HEADST+014h      ;No. scans primary data
ACHE            equ  HEADST+016h      ;End channel of 1st scan
ICHE            equ  HEADST+017h      ;End channel of int. data
FCHE            equ  HEADST+018h      ;End channel of primary data
CHNLN           equ  HEADST+019h      ;Channel number storage
CHNLT           equ  HEADST+01Ah      ;Total channel # storage
SCANN           equ  HEADST+01Bh      ;Scan number storage
SRATEN          equ  HEADST+01Dh      ;Scan rate storage
BCNT            equ  HEADST+01Eh      ;Storage bank count
CHBAS           equ  HEADST+01Fh      ;Base channel storage
CHEND           equ  HEADST+020h      ;End channel storage
TIMTL           equ  HEADST+021h      ;Time total
;
;                          - PROGRAM CONTROL VARIABLES
;
REFLAG          equ  HEADST+029h      ;Remote event flag
RESMDY          equ  HEADST+02Ah      ;Remote samples/day
REMIN           equ  HEADST+02Bh      ;Remote minute flag
RFFLAG          equ  HEADST+02Ch      ;Radio mode
TPFLAG          equ  HEADST+02Dh      ;Tape mode
SSFLAG          equ  HEADST+02Eh      ;Solid state mode
HDRFLG          equ  HEADST+02Fh      ;CH16 header/data flag
SMODE           equ  HEADST+030h      ;Intermediate mode flag
SMDYI           equ  HEADST+031h      ;Intermediate samples/day
SMDYF           equ  HEADST+032h      ;Primary samples/day
IMODE           equ  HEADST+033h      ;Int. data minute flag
FMODE           equ  HEADST+034h      ;Primary data minute flag
STRTTIM         equ  HEADST+035h      ;Start time storage
```

```
;
;                          - ACQUISITION VARIABLES
;
BUFFER          equ  HEADST+05Bh      ;Data buffer pointer
ALARM           equ  HEADST+05Dh      ;Hour alarm storage
SAMNUM          equ  HEADST+05Eh      ;Sample number storage
SCNCH           equ  HEADST+060h      ;Byte count per scan
SCNSZ           equ  HEADST+062h      ;Number of scans
LSTBNK          equ  HEADST+064h      ;Last bank of data
BANK1           equ  HEADST+065h      ;End address #1 for data
BANK2           equ  HEADST+067h      ;End address #2 for data
BANKL           equ  HEADST+069h      ;Last end address
FRSTCH          equ  HEADST+06Bh      ;Base channel storage
LSTCH           equ  HEADST+06Ch      ;Last channel storage
SCANS           equ  HEADST+06Dh      ;Scan size storage
RSMODE          equ  HEADST+06Fh      ;Remote mode storage
REFLG           equ  HEADST+070h      ;Remote event indicator flag
RMODE           equ  HEADST+071h      ;Minute mode storage
RSDAY           equ  HEADST+072h      ;Sample rate storage
MMODE           equ  HEADST+075h      ;Minute mode flag
SAMDAY          equ  HEADST+076h      ;Samples/day storage
SFLAG           equ  HEADST+077h      ;Scan rate time base storage
SMPLRT          equ  HEADST+078h      ;Scan rate entry storage
TRACK           equ  HEADST+079h      ;Tape track storage
;
;                          - DATAVAULT EQUATES
;
BOOT_SECT       equ  SCRATCH+0100h  ;Start of boot sector
BYT_SCT         equ  BOOT_SECT+0Bh  ;Bytes/sector
SCT_CL          equ  BOOT_SECT+0Dh  ;# sectors/cluster
F1_STRT         equ  BOOT_SECT+0Eh  ;Sector # for 1st fat table
NO_FAT          equ  BOOT_SECT+010h ;# of fat tables
NO_DIR          equ  BOOT_SECT+011h ;# of directory entries
NO_SECT         equ  BOOT_SECT+013h ;Total # of sectors
SCT_FAT         equ  BOOT_SECT+016h ;# sectors/fat table
;
SECT_BUF        equ  BOOT_SECT+020h ;Start of sector buffer
;
DOS_EQ          equ  SECT_BUF+0200h ;Start of working parameters
F2_STRT         equ  DOS_EQ+00h     ;Start of 2nd Fat table
RD_STRT         equ  DOS_EQ+02h     ;Start of root directory
D_STRT          equ  DOS_EQ+04h     ;Start of data
NO_CL           equ  DOS_EQ+06h     ;Total # of clusters
CL_OFF          equ  DOS_EQ+08h     ;Starting cluster for data
FAT_OFF         equ  DOS_EQ+0Ah     ;Starting cluster label
FAT_BUFF        equ  DOS_EQ+0Ch     ;FAT pointer
```

```
;
UNIT_NO         equ   DOS_EQ+0Eh      ;Unit number
CART_SBYTE      equ   DOS_EQ+0Fh      ;Data byte storage
DV_CMD          equ   DOS_EQ+011h     ;Data vault command storage
SECT_NO         equ   DOS_EQ+013h     ;Sector number
SECT_CNT        equ   DOS_EQ+015h     ;Sector count
SECT_TRNS       equ   DOS_EQ+017h     ;Sectors transferred
DV_BUFF         equ   DOS_EQ+019h     ;Buffer pointer
GEN_CRC         equ   DOS_EQ+01Bh     ;Generated CRC
STOR_CRC        equ   DOS_EQ+01Dh     ;Stored CRC
CL_NXT          equ   DOS_EQ+01Fh     ;Fat cluster update
DATA_CNT        equ   DOS_EQ+021h     ;Number of sectors of data
TTL_HI          equ   DOS_EQ+023h     ;MSB of byte count
TTL_LO          equ   DOS_EQ+025h     ;LSB of byte count
;
;                       - ENTRY STORAGE BUFFER EQUATES
;
BCI             equ   SCRATCH+0400h   ;BCD number count
BCDBUF          equ   SCRATCH+0402h   ;BCD entry storage
NCI             equ   SCRATCH+0500h   ;No. characters in IBUFF
IBUFF           equ   SCRATCH+0502h   ;ASCII input buffer
;
;                       - CLOCK AND I/O BUFFERS
;
BANKSTO         equ   SCRATCH+0600h   ;Bank select storage
BANKNO          equ   SCRATCH+0601h   ;Bank select storage
TSTORE          equ   SCRATCH+0602h   ;Temp. storage for pointers
TIMEM           equ   SCRATCH+0650h   ;Clock print storage
STACK           equ   0FF00h          ;Stack storage
```

```
;
;INTERRUPT ROUTINES
;-----------------------------------------------------------------------
;
;--- RESTART 0 ---
;
;                       - PROGRAM COMES HERE ON RESET
;
        DEFSEG  RESTART,ABSOLUTE
        SEG     RESTART
        org     00h
;
        di                              ;Disable interrupts
        jp      BEGIN                   ;Jump to start of program
;
;--- INTERRUPT JUMP TABLE ---
;
        DEFSEG  INTRPTS,START=020h,ABSOLUTE
        SEG     INTRPTS
        org     020h
;
INTTBL:
        jp      INTERR                  ;No support for INT 0
        db      00h
        jp      CLKINT                  ;Clock interrupt on INT 1
        db      00h
        jp      ATDINT                  ;A-D interrupt on INT 2
        db      00h
        jp      INTERR                  ;No support for INT 3
        db      00h
        jp      INTERR                  ;No support for INT 4
        db      00h
        jp      INTERR                  ;No support for INT 5
        db      00h
        jp      INTERR                  ;No support for INT 6
        db      00h
        jp      INTERR                  ;No support for INT 7
        db      00h

;
;                       - INTERNAL INTERRUPTS
;
        jp      INTERR                  ;INT 1
        jp      INTERR                  ;INT 2
        jp      INTERR                  ;Timer 0
        jp      INTERR                  ;Timer 1
        jp      INTERR                  ;DMA 0
        jp      INTERR                  ;DMA 1
        jp      INTERR                  ;CSI/O
        jp      INTERR                  ;SER 0
        jp      TERMST                  ;SER 1
```

```
;
;SYSTEM INITIALIZATION
;-----------------------------------------------------------------
;
;--- SET ALL I/O PORTS AND INITIALIZE ALL PARAMETERS ---
;
          DEFSEG  PROGRAM,START=060h,ABSOLUTE
          SEG     PROGRAM         ;Locate at 60 HEX
          org     060h
;
BEGIN:    in0     a,(ITC)         ;Get status of interrupts
          ld      b,a             ;Save it in B
          and     038h            ;Strip mask
          out0    (ITC),a         ;Resave interrupts
          bit     7,b             ;Test for TRAP bit
          jr      z,MODSET        ;If 0, then proceed
          ld      hl,OPCERRM      ;Else print opcode error
          call    PSTRG
          pop     hl              ;Load stack value
          ld      (TSTORE),hl
          ld      hl,TSTORE+1
          call    HEXP
MODSET:   im      0               ;Set mode 0 interrupts
;
INIT:     ld      de,ILTBL        ;Point to int. vector table
          ld      a,d             ;Get high-byte of address
          ld      i,a             ;Save it in I
          out0    (IL),e          ;Save low-byte in IL reg.
;
;                    - SET CONTROLLER SERIAL PORTS
;
          ld      hl,SERTBL       ;Point to port parameters
          ld      c,00h           ;Initial port is at 00 HEX
          ld      b,05h           ;Load count
          otimr                   ;Load ports with values
;
;                    - SET DELAY TIMERS
;
          ld      hl,TIMTBL       ;Point to timer parameters
          ld      c,0Ah           ;Initial port is at 0A HEX
          ld      b,0Eh           ;Load count
          otimr                   ;Load timers with values
;
;                    - SET DMA
;
          ld      hl,DMATBL       ;Point to DMA parameters
          ld      c,020h          ;Initial port is at 20 HEX
          ld      b,03h           ;Load count
          otimr                   ;Load DMA with values
```

```
;
;                   - CLEAR INTERRUPTS
;
            xor     a               ;Clear A
            out0    (RCR),A         ;Clear refresh control
            ld      a,01h           ;Load A with 01 HEX
            out0    (ITC),a         ;Clear internal interrupts
;
;                   - SET SERIAL PORT #0
;
            in0     a,(CNTLA0)      ;Get control word
            or      010h            ;Mask it to clear CTS
            out0    (CNTLA0),a      ;Send it back out
;
;                   - SET PARALLEL PORT
;
            ld      a,092h          ;Set port A & B for input
            out     (PICNTL),a      ; and port C for output
            ld      a,TAPOFF        ;Tape power off
            out     (PICNTL),a
            ld      a,RFOFF         ;RF power off
            out     (PICNTL),a
            call    RE_RST          ;Reset event counters
            call    SENOFF          ;Power down sensors
;
;                   - SET MULTI-FUNCTION BOARD
;
            ld      a,013h          ;No RTS,no parity,RXRDY,8 bit
            out     (SP3+MR1),a     ;Load the RS232 port
            out     (SP4+MR1),a     ;Load the RS232 port
            out     (SP5+MR1),a     ;Load the RS232 port
            out     (SP6+MR1),a     ;Load the RS232 port
            ld      a,07h           ;Normal, 1 stop bit
            out     (SP3+MR2),a     ;Load the RS232 port
            out     (SP4+MR2),a     ;Load the RS232 port
            out     (SP5+MR2),a     ;Load the RS232 port
            out     (SP6+MR2),a     ;Load the RS232 port
            ld      a,0E0h          ;BRG = 1,timer mode
            out     (DUART1+ACR),a    ;Load the RS232 port
            out     (DUART2+ACR),a    ;Load the RS232 port
            ld      a,0CCh          ;9600 baud
            out     (SP3+CSR),a     ;Load the RS232 port
            out     (SP4+CSR),a     ;Load the RS232 port
            out     (SP5+CSR),a     ;Load the RS232 port
            out     (SP6+CSR),a     ;Load the RS232 port
            ld      a,020h          ;Reset Rx
            out     (SP3+DCR),a     ;Load the RS232 port
            out     (SP4+DCR),a     ;Load the RS232 port
            out     (SP5+DCR),a     ;Load the RS232 port
            out     (SP6+DCR),a     ;Load the RS232 port
            ld      a,030h          ;Reset Tx
            out     (SP3+DCR),a     ;Load the RS232 port
            out     (SP4+DCR),a     ;Load the RS232 port
            out     (SP5+DCR),a     ;Load the RS232 port
            out     (SP6+DCR),a     ;Load the RS232 port
```

```
            ld      a,05h           ;Enable port 3
            out     (SP3+DCR),a     ;Load the RS232 port
            ld      a,0C0h          ;Put on standby
            out     (SP3+DCR),a     ;Load the RS232 port
            out     (SP4+DCR),a     ;Load the RS232 port
            out     (SP5+DCR),a     ;Load the RS232 port
            out     (SP6+DCR),a     ;Load the RS232 port
;
;--- DELAY BEFORE DOING ANYTHING ---
;
            ld      c,03Ch          ;Load 1st count for delay
SWAIT:      ld      b,05h           ;Load 2nd count for delay
            ld      ix,SWAIT1       ;Load return point
            jp      TO_DLY          ;Delay for 1 sec.
;
SWAIT1:     dec     c               ;Decrement 1st count
            jr      nz,SWAIT        ;Loop for 5 seconds
;
;--- TEST TO BE SURE THERE IS A BATTERY-BACKED RAM --
;
ROMCK:      ld      a,0F8h          ;Common area = F000h
                                    ; bank = 8000h
            out0    (CBAR),a        ;Load memory areas
ROMCK1:     ld      a,018h          ;Load value for bank #4
            out0    (BBR),a         ;Select bank #4
            out0    (CBR),a         ;Select common area
            ld      (BANKNO),a      ;Save bank #
            ld      (BANKSTO),a     ;Again
            ld      c,a             ,Move it to C
            xor     a               ;Clear A
            ld      a,(BANKNO)      ;Get Bank # back
            cp      c               ;Is it the same?
            jr      z,MAIN          ;Yes, then RAM is OK
            ld      hl,RAMERRM      ;Else load error message
            call    PSTRG           ;Display it
            halt
```

```
;
;MAIN PROGRAM
;-------------------------------------------------------------------------
;
MAIN:       ld      sp,STACK        ;Load the stack
;
;                   - THE PROGRAM START WILL SEARCH IN
;                     BATTERY-BACKED RAM FOR ANY
;                     PREVIOUSLY ENTERED VARIABLES AND
;                     BEGIN SAMPLING BASED ON THOSE.
;                   - IF NO VALUES ARE FOUND, THE
;                     PROGRAM WILL START SAMPLING 4 TIMES A DAY
;                     FOR PRIMARY DATA WITH INT. DATA EVERY
;                     HOUR
;
            xor     a               ;Clear remote count storage
            ld      (RCTA),a
            ld      (RCTB),a
            ld      a,FALSE         ;Set remote event flag False
            ld      (REFLG),a
WAKE:       PROC
            ld      a,(REFLAG)      ;Load remote enable flag
            cp      FALSE           ;Is it false?
            jr      z, ..A          ;Yes, then skip next
            ld      a,TRUE          ;Set it TRUE
            ld      (REFLAG),a
            ld      a,(RESMDY)      ;Load sample rate
            cp      019h            ;Is it valid
            jr      c, ..A          ;Yes, then skip next
            ld      a,01h           ;Sample each hour
            ld      (RESMDY),a
  ..A:      ld      a,(PWRFLG)      ;Load sensor power control flag
            cp      FALSE           ;Is it false?
            jr      z, ..B          ;Yes, then skip next
            ld      a,TRUE          ;Set it true
            ld      (PWRFLG),a
  ..B:      ld      a,(SCNFLG)      ;Load scan count flag
            cp      FALSE           ;Is it false?
            jr      z, ..C          ;Yes, then skip next
            ld      a,TRUE          ;Set it true
            ld      (SCNFLG),a
  ..C:      ld      a,(RFFLAG)      ;Load radio flag
            cp      FALSE           ;Is it false?
            jr      z, ..D          ;Yes, then skip next
            cp      02h             ;Is remote only?
            jr      z, ..D          ;Yes, then skip next
            ld      a,01h           ;Set it every time
            ld      (RFFLAG),a
  ..D:      ld      a,(TPFLAG)      ;Load tape flag
            cp      01h             ;Is it every time?
            jr      z, ..E          ;Yes, then skip next
            cp      02h             ;Is remote only?
            jr      z, ..E          ;Yes, then skip next
            ld      a,FALSE         ;Set it FALSE
            ld      (TPFLAG),a
```

```
..E:    ld      a,(SSFLAG)      ;Load solid-state flag
        cp      FALSE           ;Is it false?
        jr      z, ..F          ;Yes, then skip next
        cp      01h             ;Is every time?
        jr      z, ..F          ;Yes, then skip next
        ld      a,02h           ;Set it remote only
        ld      (SSFLAG),a
..F:    ld      a,(SMODE)       ;Get the sample mode
        cp      FALSE           ;Is it primary mode?
        jr      z, ..6          ;Yes, then start that mode
        cp      TRUE            ;Is it int. mode?
        jr      z, ..1          ;Yes, then start that mode
        ld      a,TRUE          ;Default startup is int. mode
        ld      (SMODE),a       ;Save it
..1:    ld      a,(IMODE)       ;Load minute flag for int. data
        cp      TRUE            ;Is it set?
        jr      z, ..4          ;Yes then test value
        cp      FALSE           ;Is it not set?
        jr      z, ..2          ;Yes then check hour mode
        ld      a,FALSE         ;Load default value (hour mode)
        ld      (IMODE),a
..2:    ld      a,(SMDYI)       ;Load number samples/day
        cp      00h             ;Make sure it is not 0
        jr      z, ..3          ;Yes then set default
        cp      019h            ;Is it greater than 24?
        jr      c, ..6          ;No, then skip next
..3:    ld      a,01h           ;Default value is every hour
        ld      (SMDYI),a       ;Save it
        ld      hl,0800h        ;Save default scans 2048
        ld      (SCANI),hl
        ld      a,05h           ;Save default # channels 5
        ld      (CHNLI),a
        ld      (ICHE),a        ;Save last channel
        xor     a               ;Clear A
        ld      (ICHB),a        ;Base channel is 0
        jr      ..6             ;Skip next
..4:    ld      a,(SMDYI)       ;Load number samples/hour
        cp      00h             ;Make sure it is not 0
        jr      z, ..5
        cp      03Dh            ;Is it greater than 60?
        jr      c, ..6          ;No, then skip next
..5:    ld      a,0Fh           ;Default value is every 15 mins.
        ld      (SMDYI),a       ;Save it
        ld      hl,080h         ;Save default scans 128
        ld      (SCANI),hl
        ld      a,05h           ;Save default # channels 5
        ld      (CHNLI),a
        ld      (ICHE),a        ;Save last channel
        xor     a               ;Clear A
        ld      (ICHB),a        ;Base channel is 0
..6:    ld      a,(FMODE)       ;Load minute flag for int. data
        cp      TRUE            ;Is it set?
        jr      z, ..9          ;Yes then test value
        cp      FALSE           ;Is it not set?
        jr      z, ..7          ;Yes then check hour mode
```

```
        ld      a,FALSE         ;Load default value
        ld      (FMODE),a
..7:    ld      a,(SMDYF)       ;Load number samples/day
        cp      00h             ;Make sure it is not 0
        jr      z, ..8          ;Yes then set default
        cp      019h            ;Is it greater than 24?
        jr      c, ..11         ;No, then skip next
..8:    ld      a,04h           ;Default value is every 6 hours
        ld      (SMDYF),a       ;Save it
        ld      hl,01000h       ;Save default scans 4096
        ld      (SCANF),hl
        ld      a,0Ch           ;Save default # channels 12
        ld      (CHNLF),a
        ld      (FCHE),a        ;Save last channel
        ld      a,5             ;
        ld      (FCHB),a        ;Base channel is 5
        jr      ..11            ;Skip next
..9:    ld      a,(SMDYF)       ;Load number samples/hour
        cp      00h             ;Make sure it is not 0
        jr      z, ..10
        cp      03Dh            ;Is it greater than 60?
        jr      c, ..11         ;No, then skip next
.10:    ld      a,03Ch          ;Default value is every hour
        ld      (SMDYF),a       ;Save it
        ld      hl,040h         ;Save default scans 64
        ld      (SCANF),hl
        ld      a,010h          ;Save default # channels 16
        ld      (CHNLF),a
        ld      (FCHE),a        ;Save last channel
        xor     a               ;Clear A
        ld      (FCHB),a        ;Base channel is 0
..11:   ld      a,080h          ;Load disable mask for alarm
        ld      bc,0800h        ;08h = count
        ld      c,RTCBASE+08h     ;C = alarm port
.12:    out     (c),a           ;Send mask value
        inc     c               ;Increment port
        djnz    ..12            ;Loop for all alarm regs.
        call    GCLOCK          ;Get current time
        ld      a,(MHR)         ;Load the current hour
        inc     a               ;Increment hour
        cp      018h            ;Is it greater than 23?
        jr      c, ..13         ;No, then skip next
        sub     18h             ;Subtract 24
..13:   out     (AHR),a         ;Load hour alarm reg.
        ld      a,01h           ;Set alarm mode
        out     (RTCSTAT),a
        ld      a,01Ch          ;Load alarm command to clock
        out     (RTCCMD),a
        call    CLRMEM          ;Clear the data buffer
        call    INTST           ;Enable the interrupt controller
        call    RE_RST          ;Clear remote counters
        halt                    ;Wait for alarm to sample
        ENDPROC
```

```
;
;                    - PROGRAM COMES HERE WHEN A RETURN IS
;                      HIT ON THE TERMINAL.  THIS GENERATES
;                      AN INTERRUPT TO SIGNAL THAT DATA
;                      ENTRY IS DESIRED.
;
TERMST: PROC
        di                      ;Disable interrupts
        call    S1IN            ;Clear entry
        cp      CR              ;Is it a carriage return?
        jr      z, ..1          ;No, then go back to sleep
        ei                      ;Enable interrupts
        halt
;
 ..1:   ld      a,01h           ;Disable internal interrupts
        out0    (ITC),a
        ld      a,04h           ;Disable Serial#1 interrupt
        out0    (STAT1),a
        xor     a               ;Clear tape track number
        ld      (TRACK),a
        call    SETCLK          ;Set the clock with current time
;
;                 - START HERE FOR RE-ENTRY OF PARAMETERS
;
TRMST2: ld      hl,SCAN3        ;Point to start of parameters
        ld      b,065h          ;Load count
        xor     a               ;Clear A
 ..2:   ld      (hl),a          ;Clear parameters
        inc     hl              ;Increment pointer
        djnz    ..2
        ld      hl,00h          ;Clear sample number
        ld      (SAMNUM),hl
        ENDPROC
;
;                       - ENTER REMOTE EVENT MODE
;
REMOT:  ld      ix,REMOTM       ;Ask for remote mode
        ld      de,REFLAG       ;Point to flag
        call    ANSR            ;Enter flag value
        ld      a,(REFLAG)      ;Into A
        cp      FALSE           ;Is it No?
        jr      z,SPWR          ;Yes, then done
        call    SAMPL           ;Get # samples/day
        ld      a,(MMODE)       ;Save minute flag
        ld      (REMIN),a
        ld      a,(SAMDAY)      ;Save samples/day
        ld      (RESMDY),a
;
;                       - ENTER SENSOR POWER FLAG
;
SPWR:   ld      ix,SPWRM        ;Enter sensor power ind.
        ld      de,PWRFLG       ;Point to flag
        call    ANSR            ;Enter flag value
```

```
;
;                       - ENTER SCAN COUNT FLAG
;
        ld      ix,SCANCTM      ;Ask for scan in data
        ld      de,SCNFLG       ;Point to flag
        call    ANSR
;
;                       - ENTER RADIO TRANSMIT MODE
;
RADIO:  ld      ix,RADIOM       ;Ask for radio mode
        ld      de,RFFLAG       ;Point to flag
        call    GMODE           ;Enter mode
;
;                       - ENTER TAPE MODE
;
TAPE:   ld      ix,TAPEM        ;Ask for tape mode
        ld      de,TPFLAG       ;Point to flag
        call    GMODE           ;Enter mode
;
;                       - ENTER SOLID-STATE MODE
;
STATE:  ld      ix,SOLIDM       ;Ask for s-state mode
        ld      de,SSFLAG
        call    GMODE           ;Enter mode
;
;                       - DETERMINE DATA MODE
;
ENTRY:  ld      ix,MODE1M       ;Enter int. mode
        ld      de,SMODE        ;Point to flag
        call    ANSR            ;Get answer
;
ENTRY2: ld      hl,MODE2M       ;Determine all data mode
        call    PSTRG           ;Display it
        call    GETLN           ;Get answer
        ld      a,(IBUFF)       ;Load reply into A
        cp      059h            ;Is it Yes?
        jr      z,ENTRY4        ;Yes, then skip next
        cp      04Eh            ;Is it No?
        jr      z,ENTRY3        ;Yes, then set it
        jr      ENTRY2          ;Else, error
```

```
;
;                    - ENTER PARAMETERS FOR 1ST SCAN
;
ENTRY3: ld      hl,SCNFRSTM     ;Display data type
        call    PSTRG
        call    PARGET          ;Enter parameters
        ld      a,(CHNLN)       ;Save # channels
        ld      (CHNLA),a
        cp      00h             ;Is it 0?
        jr      z,ENTRY4        ;Yes then skip next
        ld      de,(SCANN)      ;Save # scans
        ld      (SCANA),de
        ld      a,(SRATEN)      ;Save scan rate
        ld      (SRATEA),a
        ld      a,(SFLAG)       ;Save scan rate flag
        ld      (SRFLGA),a
        ld      a,(CHBAS)       ;Save base channel
        ld      (ACHB),a
        ld      a,(CHEND)       ;Save end channel
        ld      (ACHE),a
;
;                    - ENTER PARAMETERS FOR INTERMEDIATE DATA
;
ENTRY4: ld      a,(SMODE)       ;Load mode flag
        cp      FALSE           ;Is it false?
        jr      z,ENTRY5        ;Yes, then skip next
        ld      hl,SCNMODE1M    ;Display data type
        call    PSTRG
        call    PARGET          ;Get parameters
        ld      a,(CHNLN)       ;Get # channels entry
        ld      (CHNLI),a       ;Save it
        cp      00h             ;Is it zero?
        jr      z,ENTRY5        ;Yes then skip next
        ld      de,(SCANN)      ;Get # scans entry
        ld      (SCANI),de      ;Save it
        ld      a,(SRATEN)      ;Get scan rate entry
        ld      (SRATEI),a      ;Save it
        ld      a,(SFLAG)       ;Save scan rate flag
        ld      (SRFLGI),a
        ld      a,(CHBAS)       ;Save base channel
        ld      (ICHB),a
        ld      a,(CHEND)       ,Save end channel
        ld      (ICHE),a
        call    SAMPL           ;Get sample/day
        ld      a,(SAMDAY)      ;Save value
        ld      (SMDYI),a
        ld      a,(MMODE)       ;Load minute mode
        ld      (IMODE),a       ;Save it
```

```
;
;                       - ENTER PARAMETERS FOR PRIMARY DATA
;
ENTRY5: ld      hl,SCNMAINM     ;Display data type
        call    PSTRG
        call    PARGET          ;Get parameters
        ld      a,(CHNLN)       ;Get # channels entry
        ld      (CHNLF),a       ;Save it
        cp      00h             ;is it zero?
        jr      z,PCK           ;Yes then skip next
        ld      de,(SCANN)      ;Get # scans entry
        ld      (SCANF),de      ;Save it
        ld      a,(SRATEN)      ;Get scan rate entry
        ld      (SRATEF),a      ;Save it
        ld      a,(SFLAG)       ;Save scan rate flag
        ld      (SRFLGF),a
        ld      a,(CHBAS)       ;Save base channel
        ld      (FCHB),a
        ld      a,(CHEND)       ;Save end channel
        ld      (FCHE),a
        call    SAMPL           ;Get # samples/day
        ld      a,(SAMDAY)      ,Save value
        ld      (SMDYF),a
        ld      a,(MMODE)       ;Load minute mode
        ld      (FMODE),a       ;Save it
```

```
;
;                       - CHECK ENTRIES
;
PCK:    xor     a               ;Load bank count
        ld      (BCNT),a
        ld      hl,DATAST       ;Load start of buffer
        ld      (TSTORE),hl
        ld      (BUFFER),hl     ;Save pointer
        ld      a,(CHNLA)       ;Load # channels
        ld      (LSTCH),a
        ld      hl,(SCANA)      ;Load # scans
        ld      (SCANS),hl
        call    PARCK           ;Check entries
        ld      hl,(BUFFER)
        ld      (TSTORE),hl     ;Save new count
        ld      a,(BCNT)        ;Save bank count
        ld      (BANK1),a
        ld      a,(CHNLI)       ;Load # channels
        ld      (LSTCH),a
        ld      hl,(SCANI)      ;Load # scans
        ld      (SCANS),hl
        call    PARCK           ;Check entries
        ld      a,(SRATEI)      ;Load scan rate
        ld      (SRATEN),a
        ld      a,(SMDYI)       ;Load samples/day
        ld      (SAMDAY),a
        ld      a,(IMODE)       ;Load minute mode
        ld      (MMODE),a
        ld      a,(SRFLGI)      ;Load scan rate flag
        ld      (SFLAG),a
        call    SAMCK           ;Check time
        ld      a,(BANK1)       ;Reload bank count
        ld      (BCNT),a
        ld      a,(CHNLF)       ;Load # channels
        ld      (LSTCH),a
        ld      hl,(SCANF)      ;Load # scans
        ld      (SCANS),hl
        call    PARCK
        ld      a,(SRATEF)      ;Load scan rate
        ld      (SRATEN),a
        ld      a,(SMDYF)       ;Load samples/day
        ld      (SAMDAY),a
        ld      a,(FMODE)       ;Load minute mode
        ld      (MMODE),a
        ld      a,(SRFLGF)      ;Load scan rate flag
        ld      (SFLAG),a
        call    SAMCK           ;Check time
```

```
;
;                    - DISPLAY ENTRY PARAMETERS
;
PLAY:   ld      hl,PREMOTM      ;Display remote mode
        call    PSTRG
        ld      a,(REFLAG)      ;Load flag
        cp      TRUE            ;Is it set?
        jr      z,PLAY1         ;Yes, then skip next
        ld      hl,PNSELM       ;Display a 'not selected'
        call    PSTRG1
PLAY1:  ld      hl,PSELM        ;Display selected
        call    PSTRG1
        ld      a,(RESMDY)      ;Load # sample/day
        cp      00h             ;Is it 0?
        jr      z,PLAYB         ;Yes, then skip next
        ld      hl,PSAMPLM      ;Display 'samples each'
        call    PSTRG1
        ld      a,(RESMDY)      ;Load sampling
        call    PENTRY
        ld      hl,PHOURM       ;Point to samples/day
        ld      a,(REMIN)       ;Check sample mode
        cp      FALSE           ;Is it 0?
        jr      z,PLAYA         ;Yes then skip next
        ld      hl,PMINM        ;Display samples/hour
PLAYA:  call    PSTRG1
PLAYB:  call    PCONT           ;Wait to continue
        ld      hl,PSCNCTM      ;Display scan count mode
        call    PSTRG
        ld      hl,PNOTINM      ;Point to scan not in data
        ld      a,(SCNFLG)      ;Load mode entry
        cp      FALSE           ;Is it false?
        jr      z,PLAYC         ;Yes, skip next
        ld      hl,PINM         ;Scan count in data
PLAYC:  call    PSTRG
        call    PCONT           ;Wait to continue
        ld      hl,PRADIOM      ;Display Radio mode
        call    PSTRG
        ld      a,(RFFLAG)      ;Load mode entry
        call    PMODE
        call    PCONT           ;Wait to continue
        ld      hl,PTAPEM       ;Display tape mode
        call    PSTRG
        ld      a,(TPFLAG)      ;Load mode entry
        call    PMODE
        call    PCONT           ;Wait to continue
        ld      hl,PSOLIDM      ;Display solid state mode
        call    PSTRG
        ld      a,(SSFLAG)      ;Load mode entry
        call    PMODE
        call    PCONT           ;Wait to continue
        ld      a,(CHNLA)       ;Load # channels for 1st scan
        cp      00h             ;Is it 0?
        jr      z,PLAY2         ;Yes, then skip next
        ld      hl,PSCANAM      ;Display all scan message
```

```
        call    PSTRG
        ld      a,(SRATEA)          ;Load scan rate
        call    PENTRY              ;Display it
        ld      hl,PMINM            ;Load min. label
        ld      a,(SRFLGA)          ;Load scan rate flag
        cp      TRUE                ;Is it true?
        jr      z,PLAYC1            ;Yes, then print minute label
        ld      hl,PSECM            ;Else,print sec. label
PLAYC1: call    PSTRG1
        ld      a,03h               ;Load count
        ld      (NCI),a
        xor     a                   ;Clear A
        ex      af,af'
        xor     a
        ld      hl,(SCANA)          ;Display # scans
        call    DEC16
        call    DECP
        ld      hl,PSCANSM          ;Display label
        call    PSTRG1
        ld      a,(ACHB)            ;Display base channel
        inc     a                   ;Increment for print
        call    PENTRY
        ld      hl,TOM              ;Display label
        call    PSTRG1
        ld      a,(ACHE)            ;Display end channel
        call    PENTRY
        call    PCONT               ;Wait to continue
PLAY2:  ld      a,(SMODE)           ;Load # channels for int. data
        cp      FALSE               ;Is it false?
        jr      z,PLAY3             ;Yes, then skip next
        ld      hl,PSCANIM          ;Display all scan message
        call    PSTRG
        ld      a,(SRATEI)          ;Load scan rate
        call    PENTRY              ;Display it
        ld      hl,PMINM            ;Load min. label
        ld      a,(SRFLGI)          ;Load scan rate flag
        cp      TRUE                ;Is it true?
        jr      z,PLAY2A            ;Yes, then print minute label
        ld      hl,PSECM            ;Else,print sec. label
PLAY2A: call    PSTRG1
        ld      a,03h               ;Load count
        ld      (NCI),a
        xor     a                   ;Clear A
        ex      af,af'
        xor     a
        ld      hl,(SCANI)          ;Display # scans
        call    DEC16
        call    DECP
        ld      hl,PSCANSM          ;Display label
        call    PSTRG1
        ld      a,(ICHB)            ;Display base channel
        inc     a                   ;Increment for print
        call    PENTRY
        ld      hl,TOM              ;Display label
        call    PSTRG1
```

```
        ld      a,(ICHE)        ;Display end channel
        call    PENTRY
        call    PSPACE
        ld      a,(SMDYI)       ;Load samples/day
        call    PENTRY          ;Display it
        ld      hl,PHOURM       ;Point to hour mess.
        ld      a,(IMODE)       ;Check sampling
        cp      FALSE           ;Is it false?
        jr      z,PLAYD         ;Yes, then skip next
        ld      hl,PMINM        ;Display label
PLAYD:  call    PSTRG1
        call    PCONT           ;Wait to continue
PLAY3:  ld      a,(CHNLF)       ;Load # channels for main data
        cp      00h             ;Is it 0?
        jr      z,PLAY4         ;Yes, then skip next
        ld      hl,PSCANFM      ;Display all scan message
        call    PSTRG
        ld      a,(SRATEF)      ;Load scan rate
        call    PENTRY          ;Display it
        ld      hl,PMINM        ;Load min. label
        ld      a,(SRFLGF)      ;Load scan rate flag
        cp      TRUE            ;Is it true?
        jr      z,PLAY3A        ;Yes, then print minute label
        ld      hl,PSECM        ;Else,print sec. label
PLAY3A: call    PSTRG1
        ld      a,03h           ;Load count
        ld      (NCI),a
        xor     a               ;Clear A
        ex      af,af'
        xor     a
        ld      hl,(SCANF)      ;Display # scans
        call    DEC16
        call    DECP
        ld      hl,PSCANSM      ;Display label
        call    PSTRG1
        ld      a,(FCHB)        ;Display base channel
        inc     a               ;Increment for print
        call    PENTRY
        ld      hl,TOM          ;Display label
        call    PSTRG1
        ld      a,(FCHE)        ;Display end channel
        call    PENTRY
        call    PSPACE
        ld      a,(SMDYF)       ;Display sampling
        call    PENTRY
        ld      hl,PHOURM       ;Point to hour mess.
        ld      a,(FMODE)       ;Check sampling
        cp      FALSE           ;Is it false?
        jr      z,PLAYE         ;Yes, then skip next
        ld      hl,PMINM        ;Display label
PLAYE:  call    PSTRG1
        call    PCONT           ;Wait to continue
PLAY4:  ld      hl,OKM          ;Ask if ok
        call    PSTRG           ;
        call    GETLN           ;Get answer
```

```
        ld      a,(IBUFF)       ;Into A
        cp      059h            ;Is it Yes?
        jr      z,SAM1          ;Yes, then done
        cp      04Eh            ;Is it No?
        jp      z,TRMST2        ;Yes, then do again
        jr      PLAY4           ;Else, error


;
;                   -ENTER START TIME
;
SAM1:   call    GCLOCK          ;Display current time
        ld      hl,MTMEM
        call    TIMOUT
        ld      hl,CLKSTRTM     ;Point to message
        call    PSTRG1          ;Display it
        ld      hl,STRTTIM      ;Point to storage
        call    TIMINP          ;Get the start time
        jr      c,SAM1          ;If carry set, then error
        ld      bc,0800h        ;B = Count = 08
        ld      c,RTCBASE+08h   ;C = port
        ld      hl,MTMEM        ;Point to memory storage
        otimr                   ;Load clock for alarm
        ld      a,01h           ;Set alarm on clock
        out     (RTCSTAT),a
        ld      a,01Ch
        out     (RTCCMD),a
;
;                   - CLEAR DATA BUFFER AND START
;                     INTERRUPTS.
;
        call    CLRMEM          ;Clear data buffer
        ld      a,(FMODE)       ;Save start mode
        call    INTST           ;Start interrupts
        halt                    ;Wait for interrupt
```

```
;
;--- GET DATA MODE ROUTINE ---
;                       - DE MUST POINT TO FLAG LOCATION
;                       - IX MUST POINT TO QUESTION
;
GMODE:  call    ANSR            ;Get answer
        ld      a,(de)          ;Into A
        cp      FALSE           ;Is it No?
        ret     z               ;Yes, then done
GMODE1: ld      hl,STORM        ;Load 2nd message
        call    PSTRG           ;Display it
        call    GETLN           ;Get answer
        ld      a,(NCI)         ;Load number of entries
        cp      02h             ;Is it greater than 1?
        jr      nc,GMODE        ;If more than 1, then error
        ld      a,(IBUFF)       ;Load reply into A
        call    CONVAS          ;Convert to hex
        jr      c,GMODE1        ;If error, ask again
        ld      (de),a          ;Save mode
        cp      03h             ;Is > 2?
        jr      nc,GMODE1       ;Yes, then error
        cp      00h             ;Is it zero?
        jr      z,GMODE1        ;Yes then error
        ret
;
;--- SET FLAG ROUTINE ---
;                       - DE MUST POINT TO FLAG LOCATION
;                       - HL MUST POINT TO QUESTION
;
ANSR:   ld      a,FALSE         ;Set flag to false
        ld      (de),a
        push    ix              ;Move pointer to HL
        pop     hl
        call    PSTRG           ;Display question
        call    GETLN           ;Get answer
        ld      a,(IBUFF)       ;Load reply into A
        cp      059h            ;Is it Yes?
        jr      z,ANSR1         ;Yes, then set it
        cp      04Eh            ;Is it No?
        ret     z               ;Yes, done
        jr      ANSR            ;Else, error
ANSR1:  ld      a,TRUE          ;Load flag for sensor power
        ld      (de),a
        ret
```

```
;
;--- ENTER PARAMETERS ---
;                       - # CHANNELS IN CHNLN
;                       - # SCANS IN SCANN
;                       - SCAN RATE IN SRATEN
;
PARGET: ld      hl,BCHNLM       ;Ask for # channels
        call    PSTRG
        call    GETLN
        ld      a,(NCI)         ;Load number of entries
        cp      03h             ;Is it greater than 2?
        jr      nc,PARGET       ;If more than 2, then error
        ld      hl,CHBAS        ;Point to storage
        ld      de,IBUFF        ;Point to input buffer
        call    ASCHEX          ;Convert to hex
        jr      c,PARGET
        ld      a,(CHBAS)       ;Load base channel
        cp      00h             ;Is it zero?
        jr      z,PARGET        ;Yes, then error
        dec     a               ;Decrement base for A-D
        ld      (CHBAS),a
        ld      hl,CHANM        ;Ask for # channels
        call    PSTRG1
        call    GETLN
        ld      a,(NCI)         ;Load number of entries
        cp      03h             ;Is it greater than 2?
        jr      nc,PARGET       ;If more than 2, then error
        ld      hl,CHNLN        ;Point to storage
        ld      de,IBUFF        ;Point to input buffer
        call    ASCHEX          ;Convert to hex
        jr      c,PARGET
        ld      a,(CHNLN)       ;Load # channels
        cp      00h             ;Is it zero?
        ret     z               ;Yes, then done
PARGTB: ld      a,(CHBAS)       ;Load base channel
        ld      b,a             ;Move it to B
        ld      a,(CHNLN)       ;Load number of channels
        add     a,b             ;Add them
        ld      (CHEND),a       ;Save result
        cp      011h            ;Is it less than 16?
        jr      c,PARGT1        ;Yes, then continue
PARGTA: ld      hl,CHERRM       ;Display error message
        call    PSTRG
        jp      PARGET          ;Try again
PARGT1: ld      hl,SCANM        ;Ask for # of scans
        call    PSTRG1
        call    GETLN
        ld      a,(NCI)         ;Load number of entries
        cp      05h             ;Is it greater than 4?
        jr      nc,PARGT1       ;If more than 4, then error
        call    ASCBCD          ;Convert entry to BCD
        jr      c,PARGT1
        call    BCDHEX          ;Convert entry to hex
        ld      (SCANN),hl      ;Save it
```

```
PARGT2: ld      ix,SRATEM       ;Display scan rate entry
        ld      de,SFLAG        ;Point to flag
        call    ANSR
        ld      a,(SFLAG)       ;Load the flag
        cp      TRUE            ;Is it < 1Hz?
        jr      z,PARGT4        ;Yes, then skip next
        ld      hl,SCNSECM      ;Display samples/sec
        call    PSTRG1
        ld      de,03E8h        ;Load 1000
        ld      a,(CHNLN)       ;Load number of channels
        call    DIVIDE
        ld      a,d             ;Load MSB
        cp      00h             ;Is it 1000?
        jr      z,PARGT3
        ld      de,0FFh         ;Yes, then load 500
PARGT3: ld      a,e             ;Move result to A
        ld      (TSTORE),a      ;Save result
        ld      a,02h           ;Load count
        ld      (NCI),a
        xor     a               ;Clear A
        ex      af,af'
        xor     a
        ex      de,hl           ;Display max rate
        call    DEC16
        call    DECP
        ld      hl,SCNSIM       ;Display label
        call    PSTRG1
        call    GETLN           ;Get response
        ld      a,(NCI)         ;Load number of entries
        cp      04h             ;is it greater than 3?
        jr      nc,PARGT2       ;If more than 3, then error
        call    ASCBCD          ;Convert to BCD
        jr      c,PARGT2
        call    BCDHEX          ;Convert to hex
        ld      a,h             ;Load MSB
        cp      00h             ;Is it zero?
        jr      nz,PARGT2       ;No, then error
        ld      a,(TSTORE)      ;Load max
        cp      l               ;Is it less than entry?
        jr      c,PARGT2        ;Yes, then error
        ld      a,l             ;Else save result
        ld      (SRATEN),a
        ret
```

```
;
PARGT4: ld      hl,SCNMINM      ;Display samples/sec
        call    PSTRG
        call    GETLN
        ld      a,(NCI)         ;Load number of entries
        cp      03h             ;Is it greater than 2?
        jr      nc,PARGT4       ;If more than 2, then error
        ld      hl,SRATEN       ;Point to storage
        ld      de,IBUFF        ;Point to input buffer
        call    ASCHEX          ;Convert to hex
        jr      c,PARGT4
        ld      a,(SRATEN)      ;Load scan rate
        cp      02h             ;Is it less than minimum?
        jr      c,PARGT4        ;Yes, then error
        cp      3Dh             ;Is it less than max?
        jr      nc,PARGT4       ;No, then error
        ret

;
;--- GET NUMBER OF SAMPLES/DAY ---
;                       - SAMDAY WILL CONTAIN SAMPLES/DAY
;                       - MMODE WILL CONTAIN MINUTE FLAG
;
SAMPL:  ld      hl,SAMPL1M      ;Point to message
        call    PSTRG           ;Display it
        call    GETLN           ;Get answer
        ld      a,(IBUFF)       ;Into A
        cp      048h            ;Is it H?
        jr      z,SAMPL1        ;Yes, then hour
        cp      04Dh            ;Is it M?
        jr      z,SAMPL2        ;Yes, then minute
        jr      SAMPL           ;Else, error
SAMPL1: ld      a,FALSE         ;Clear minute flag
        ld      (MMODE),a
        ld      hl,SAMPL2M      ;Point to message
        call    PSTRG           ;Display it
        call    GETLN           ;Get answer
        ld      a,(NCI)         ;Load number of entries
        cp      03h             ;Is it greater than 2?
        jr      nc,SAMPL1       ;Yes, then error
        ld      hl,SAMDAY       ;Point to storage
        ld      de,IBUFF        ;Point to entry
        call    ASCHEX          ;Convert entry to hex
        jr      c,SAMPL1        ;If carry set, then error
        ld      a,(SAMDAY)      ;Load entry
        ld      de,018h         ;Load dividend
        call    DIVIDE          ;Divide entry by 24
        xor     a               ;Clear A
        cp      b               ;Is remainder 0?
        jr      nz,SAMPL1       ;No, then error
        ld      a,e             ;Save result
        ld      (SAMDAY),a
        ret
```

```
SAMPL2: ld      a,TRUE          ;Set minute flag
        ld      (MMODE),a
        ld      hl,SAMPL3M      ;Point to message
        call    PSTRG           ;Display it
        call    GETLN           ;Get answer
        ld      a,(NCI)         ;Load number of entries
        cp      03h             ;Is it greater than 2?
        jr      nc,SAMPL2       ;Yes, then error
        ld      hl,SAMDAY       ;Point to storage
        ld      de,IBUFF        ;Point to entry
        call    ASCHEX          ;Convert entry to hex
        jr      c,SAMPL2        ;If carry set, then error
        ld      a,(SAMDAY)      ;Load entry
        ld      de,03Ch         ;Load dividend
        call    DIVIDE          ;Divide entry by 24
        xor     a               ;Clear A
        cp      b               ;Is remainder 0?
        jr      nz,SAMPL2       ;No, then error
        ld      a,e             ;Save result
        ld      (SAMDAY),a
        ret
;
;                       - CHECK DATA PARAMETERS
;
PARCK:  ld      hl,(TSTORE)     ;Load buffer address
        ld      a,(LSTCH)       ;Load end channel of scan
        cp      00h             ;Is it zero?
        ret     z               ;Yes then no check needed
        ld      (BUFFER),hl
        xor     a               ;Clear bank count
        ld      (BCNT),a
        ld      de,00h          ;Load scan start
PARCK1: ld      hl,(BUFFER)
        ld      a,(LSTCH)       ;Load # channels 1st scan
        ld      b,a             ;Save it in B
PARCK2: dec     hl              ;Decrement buffer
        dec     hl
        djnz    PARCK2          ;Loop for all channels
        ld      a,(SCNFLG)      ;Load scan flag
        cp      FALSE           ;Is it set?
        jr      z,PARCK5        ;No, then skip next
        dec     hl              ;Decrement buffer
        dec     hl              ;for scan # & terminator
PARCK5: dec     hl              ;
        inc     de              ;Increment scan number
        ld      (BUFFER),hl     ;Save pointer
        ld      a,h             ,Is high-byte pointer at 080h?
        cp      080h
        jr      nz,PARCK3       ;No, then continue
        ld      a,l
        cp      022h            ;Is Low-byte pointer at end?
        jr      c,PARCK4        ;Yes, then do next bank
PARCK3: ld      hl,(SCANS)      ;Load scan count
        sbc     hl,de           ;Subtract the two
        ret     z               ;If zero then done
```

```
          jr      PARCK1              ;Get next scan of data
          ;
PARCK4:   ld      a,(BCNT)            ,Load current bank number
          inc     a                   ;Increment count
          ld      (BCNT),a            ;Save new count
          cp      06h                 ;Was it the last bank?
          jr      z,PARERR            ;No, then skip next
          ld      hl,0EFFFh           ;Load new start for buffer
          ld      (BUFFER),hl
          jr      PARCK1
;
PARERR:   ld      hl,MEMERRM          ;Display ERROR
          call    PSTRG
          call    PCONT               ;Wait for a return
          jp      TRMST2              ;Start over
;
;--- CHECK IF ENOUGH TIME FOR SAMPLES ---
;
SAMCK:    ld      a,(LSTCH)           ;Is there a sample?
          cp      00h
          ret     z                   ;Return, if not
          ld      de,00h              ;Clear DE
          ld      a,(CHNLA)           ;Is there a 1st scan?
          cp      00h
          jr      z,SAMCK1            ;No then skip next
          ld      de,(SCANA)          ;Load # scans
          ld      a,(SRATEA)          ;Load the sample rate
          call    DIVIDE              ;Determine time of scans
          ld      a,(SRFLGA)          ;Load time base flag
          cp      TRUE                ;Is it minutes?
          jr      z,SAMCK1            ;Yes then skip next
          ld      a,03Ch              ;Divide by 60
          call    DIVIDE              ;To get minutes
SAMCK1:   ld      (TIMTL),de          ;Save the total time
          ld      de,(SCANS)          ;Load # scans
          ld      a,(SRATEN)          ;Load the sample rate
          call    DIVIDE              ;Determine time of scans
          ld      a,(SFLAG)           ;Load time base flag
          cp      TRUE                ;Is it minutes?
          jr      z,SAMCK1            ;Yes then skip next
          ld      a,03Ch              ;Divide by 60
          call    DIVIDE              ;To get minutes
          ld      hl,(TIMTL)          ;Load previous total
          add     hl,de               ;Total minutes for sample in HL
          ld      a,(PWRFLG)          ;Is there sensor power?
          cp      FALSE
          jr      z,SAMCKA            ;No, then skip next
          inc     hl                  ;Increment time by 1 min.
```

```
SAMCKA: ld      a,(RFFLAG)      ;Load radio flag
        cp      FALSE           ;Is it set
        jr      z,SAMCK2        ;No, skip next
        ld      a,(BCNT)        ;Load # banks
        inc     a
        ld      bc,05h          ;Load # minutes for RF
        ld      b,a             ;Move # banks to B
        mlt     bc              ;Determine total time for RF
        add     hl,bc           ;Add to total
SAMCK2: ld      a,(TPFLAG)      ;Load tape flag
        cp      FALSE           ;Is it set
        jr      z,SAMCK3        ;No, skip next
        ld      a,(BCNT)        ;Load # banks
        ld      bc,01h          ;Load # minutes for RF
        ld      b,a             ;Move # banks to B
        mlt     bc              ;Determine total time for RF
        add     hl,bc           ;Add to total
SAMCK3: ld      a,(SSFLAG)      ;Load tape flag
        cp      FALSE           ;Is it set
        jr      z,SAMCK4        ;No, skip next
        inc     hl              ;Increment total for 1 min
SAMCK4: ex      de,hl           ;Move total to DE
        ld      a,(MMODE)       ;Is sample in minutes?
        cp      TRUE
        jr      z,SAMCK5        ;Yes, then skip next
        ld      a,03Ch          ;Divide by 60
        call    DIVIDE          ;To get total hours
SAMCK5: ld      a,d             ;Load high byte of count
        cp      00h             ;Is it 0?
        jr      nz,SAMERR       ;No, then error
        ld      a,(SAMDAY)      ;Load # samples/hour
        ld      b,a             ;Save in B
        ld      a,e             ;Load total time
        cp      b               ;Is total less
        jr      nc,SAMERR       ;No, then error
        ret
;
SAMERR: ld      hl,TIMERRM      ;Display ERROR
        call    PSTRG
        call    PCONT           ;Wait for a return
        jp      TRMST2          ;Start over
;
;--- DISPLAY DATA MODE ---
;
PMODE:  cp      FALSE           ;Is it false?
        jr      nz,PMODE1       ;No, skip next
        ld      hl,PSTOR1M      ;Display not selected
        jr      PMODE3
PMODE1: cp      01h             ;Is it 1?
        jr      nz,PMODE2       ;No, then skip next
        ld      hl,PSTOR2M      ;Display every time
        jr      PMODE3
PMODE2: ld      hl,PSTOR3M      ;Display event mode
PMODE3: call    PSTRG
        ret
```

```
;
;INTERRUPT ROUTINES
;-----------------------------------------------------------------
;
;--- ENABLE INTERRUPT CONTROLLER ----
;
INTST:  ld      a,037h          ;Load 1st command word to 82C59
                                ;Low-byte jump address = 20 HEX
                                ;Calls in 4-byte intervals
                                ;Edge triggered
        out     (INTPRT),a
        ld      a,00h           ;Load 2nd command word
                                ;High-byte jump address = 00 HEX
        out     (INTPRTA),a
        ld      a,02h           ;Load 4th command word
                                ;Normal end-of-interrupt
        out     (INTPRTA),a
        ld      a,040h          ;Load 2nd operational word
                                ;Specific end-of-int. command
                                ;IR level acted on = 0
        out     (INTPRT),a
        ld      a,068h          ;Load 3rd operational word
                                ;Set special mask mode
                                ;No poll command
                                ;No read register command
        out     (INTPRT),a
        ld      a,0F9h          ;Load 1st operational word
                                ;Enable INT1 & INT2 interrupts
        out     (INTPRTA),a
        in      a,(RTCSTAT)     ;Clear any clock interrupts
        out     (ADINT),a       ;Clear any A-D interrupts
        ei                      ;Enable interrupts
        ret
;
;--- DISPLAY BAD INTERRUPT MESSAGE IF INVALID INTERRUPT ---
;
INTERR: di                      ;Disable interrupts
        ld      hl,INTERRM      ;Point to message
        call    PSTRG           ;Display it
        ei                      ;Enable interrupts
        reti                    ;Wait for next interrupt
```

```
;
;--- CLOCK INTERRUPT ROUTINES ---
;
CLKINT: PROC
        di                          ;Disable interrupts
        call    GCLOCK              ;Get the current time
;
;                                   - TEST FOR REMOTE EVENT
;
        ld      a,(REFLAG)          ;Check the remote mode flag
        cp      TRUE
        call    z,RE_TST            ;Yes, then test for event
;
;                           -LOAD THE DATA HEADER
;
        ld      hl,SCANF+1          ;Point to scan counts
        ld      de,SCAN3            ;Point to header storage
        ld      bc,06h              ;Load count
..1.    ldi                         ;Move it to header
        dec     hl                  ;Decrement pointer
        dec     hl
        xor     a                   ;Clear a
        cp      c                   ;Is it zero?
        jr      nz, ..1             ;No, loop til done
        ld      de,(SAMNUM)         ;Put sample count in header
        ld      hl,SAMNO            ;Point to storage
        ld      (hl),d              ;Save low byte
        inc     hl                  ;Increment pointer
        ld      (hl),e              ;Save high byte
;
;                           - DETERMINE INTERRUPT MODE
;
        in      a,(RTCSTAT)         ;Get alarm status
        bit     0,a                 ;Is it from an entered time?
        jp      nz,ALINT            ;Yes, then primary data
        bit     5,a                 ;Test hour alarm
        jp      nz,HRINT            ;Yes, then hour interrupt
        bit     4,a                 ;Test minute alarm
        jp      nz,MININT           ;Yes, then minute interrupt
        ENDPROC
;
;                           - INTERRUPT FOR CLOCK TEST
;
SECINT: ld      hl,MTMEM            ;Point to memory storage
        call    TIMOUT              ;Display the time
        ei                          ;Enable the interrupts
        halt                        ;Wait for next interrupt
```

```
;
;                       - SAMPLE MODE IN MINUTES
;
MININT: PROC
        ld      a,(REFLG)       ;Is it remote event?
        cp      FALSE           ;No, then skip next
        jr      z, ..3
        ld      a,(REMIN)       ;Is remote data by minutes
        cp      FALSE
        jr      z,  .2          ;No, then done
        ld      de,00h          ;Clear DE
        ld      a,(MMIN)        ;Load current min
        cp      00h             ;Is it 0?
        jp      z,FCOLL         ;Yes, then collect data
        ld      e,a             ;Move current minute to E
        ld      a,(RESMDY)      ;Load #samples/hour
        call    DIVIDE          ;Divide to see if time
        xor     a               ;Clear A
        cp      b               ;Is remainder 0?
        jr      nz, ..1         ;NO, then check int. data
        jp      FCOLL           ;Collect a primary data set
;
 ..3:   ld      a,(FMODE)       ;Is primary data by minutes
        cp      FALSE
        jr      z, ..1          ;No, then skip next
        ld      de,00h          ;Clear DE
        ld      a,(MMIN)        ;Load current min
        cp      00h             ;Is it 0?
        jp      z,FCOLL         ;Yes, then collect data
        ld      e,a             ;Move current minute to E
        ld      a,(SMDYF)       ;Load #samples/hour
        call    DIVIDE          ;Divide to see if time
        xor     a               ;Clear A
        cp      b               ;Is remainder 0?
        jr      nz, ..1         ;NO, then check int. data
        jp      FCOLL           ;Collect a primary data set
;
 ..1:   ld      a,(SMODE)       ;Is int. mode set?
        cp      FALSE
        jp      z,  .2          ;No, then done
        ld      a,(IMODE)       ;Is int. data by minutes
        cp      FALSE
        jp      z, ..2          ;No, then done
        ld      de,00h          ;Clear DE
        ld      a,(MMIN)        ;Load current min
        cp      00h             ;Is it 0?
        jp      z,ICOLL         ;Yes, then collect data
        ld      e,a             ;Move current minutes to E
        ld      a,(SMDYI)       ;Load #samples/hour
        call    DIVIDE          ;Divide to see if time
        xor     a               ;Clear A
        cp      b               ;Is remainder 0?
        jp      nz, ..2         ;No, then done
        jp      ICOLL           ;Collect intermediate data
;
```

```
  ..2:  ld      a,(ALARM)       ;Load alarm value
        out     (RTCSTAT),a
        ld      a,01Ch          ;Enable clock interrupt
        out     (RTCCMD),a
        ld      a,0F9h          ;Load 1st operational word
                                ;Enable INT1 & INT2 interrupts
        out     (INTPRTA),a
        in      a,(RTCSTAT)     ;Clear any clock interrupts
        out     (ADINT),a       ;Clear any A-D interrupts
        ld      sp,STACK        ;Reinitialize stack pointer
        ei                      ;Enable interrupts
        halt                    ;Wait for next interrupt
        ENDPROC
;
;                       - SAMPLE MODE IN HOURS
;
HRINT:  PROC
        ld      a,(REFLG)       ;Is it remote event?
        cp      FALSE           ;No, then skip next
        jr      z, ..2
        ld      a,(REMIN)       ;Is remote data by minutes
        cp      TRUE
        jp      z,MININT        ;No, then try minutes
        ld      de,00h          ;Clear DE
        ld      a,(MHR)         ;Load current hour
        cp      00h             ;Is it 0?
        jp      z,FCOLL         ;Yes, then collect data
        ld      e,a             ;Move current hour to E
        ld      a,(RESMDY)      ;Load samples/day
        call    DIVIDE          ;Divide to see if time
        xor     a               ;Clear A
        cp      b               ;Is remainder 0?
        jr      nz, ..1         ;NO, then check int. data
        jp      FCOLL           ;Collect a primary data set
;
  .2:   ld      a,(FMODE)       ;Is primary data by hours
        cp      TRUE
        jr      z, ..1          ;No, then skip next
        ld      de,00h          ;Clear DE
        ld      a,(MHR)         ;Load current hour
        cp      00h             ;Is it 0?
        jp      z,FCOLL         ;Yes, then collect data
        ld      e,a             ;Move current hour to E
        ld      a,(SMDYF)       ;Load samples/day
        call    DIVIDE          ;Divide to see if time
        xor     a               ;Clear A
        cp      b               ;Is remainder 0?
        jr      nz, ..1         ;NO, then check int. data
        jp      FCOLL           ;Collect a primary data set
```

```
;
..1:    ld      a,(SMODE)       ;Is int. mode set?
        cp      FALSE
        jp      z,MININT        ;No, then try minutes
        ld      a,(IMODE)       ;Is int. data by minutes
        cp      TRUE
        jp      z,MININT        ;Yes, then try that
        ld      de,00h          ;Clear DE
        ld      a,(MHR)         ;Load current hour
        cp      00h             ;Is it 0?
        jp      z,ICOLL         ;Yes, then collect data
        ld      e,a             ;Move current hour to E
        ld      a,(SMDY)        ;Load samples/day
        call    DIVIDE          ;Divide to see if time
        xor     a               ;Clear A
        cp      b               ;Is remainder 0?
        jp      nz,MININT       ;No, then try minutes
        jp      ICOLL           ;Collect intermediate data
        ENDPROC
;
;                               - SET UP ALARM MODE
;
ALINT:  PROC
        ld      hl,ALARM        ;Point to alarm storage
        xor     a               ;Clear interrupt
        ld      (hl),a
        ld      a,(REFLAG)      ;Load remote mode flag
        cp      FALSE           ;Is it set?
        jr      z, ..1          ;No then skip next
        ld      a,(REMIN)       ;Load remote minute flag
        cp      TRUE            ;Is it set?
        jr      z,MINSET        ;Set minute mode
..1:    ld      a,(FMODE)       ;Load primary data min. flag
        cp      TRUE            ;Is it set?
        jr      z,MINSET        ;Set minute mode
        ld      a,(IMODE)       ;Load int. data min. flag
        cp      TRUE            ;Is it set?
        jr      z,MINSET        ;Set minute mode
..2:    ld      a,(REFLAG)      ;Load remote mode flag
        cp      FALSE           ;Is it set?
        jr      z, ..3          ;No then skip next
        ld      a,(REMIN)       ;Load remote minute flag
        cp      FALSE           ;Is it set?
        jr      z,HRSET         ;Set hour mode
..3:    ld      a,(FMODE)       ;Load primary data min. flag
        cp      FALSE           ;Is it set?
        jr      z,HRSET         ;Set hour mode
        ld      a,(IMODE)       ;Load int. data min. flag
        cp      FALSE           ;Is it set?
        jr      z,HRSET         ;Set hour mode
        jr      FCOLL           ;Collect data
```

```
MINSET: ld      a,(hl)          ;Load alarm mode
        or      10h             ;Load minute interrupt
        ld      (hl),a          ;Save it
        jr      ..2             ;Check for hour mode

HRSET:  ld      a,(hl)          ;Load interrupt
        or      020h            ;Add in hour interrupt
        ld      (hl),a          ;Save it
        ENDPROC
;
;                       - COLLECT AND SEND PRIMARY DATA SET
;
FCOLL:  PROC
        call    STAPE           ;Make sure tape is off
        ld      a,0F8h          ;Disable clock interrupts
        out     (INTPRTA),a
        ld      a,(PWRFLG)      ;Load power flag
        cp      FALSE           ;Is it set?
        jr      z, ..1          ;No, then skip next
        call    SENON           ;Enable power to sensors
        call    M1_DLY          ;Delay for sensor power up
;
;                       - LOAD CHANNEL 16 INTO HEADER
;
 ..1:   out     (MUXEN),a       ;Enable A-D multiplexer
        ld      hl,CH16H        ;Point to storage for CH16
        ld      (BUFFER),hl     ;Save it
        ld      b,0Fh           ;Load channel 16
        call    AQUIR           ;Get data
        ld      hl,DATAST       ;Load buffer start
        ld      (BUFFER),hl
        ld      a,(CHNLA)       ;Load 1st scan channels
        cp      00h             ;Is it zero?
        jr      z, ..2          ;Yes then skip 1st scan
        ld      a,(SRATEA)      ;Save scan rate entry
        ld      (SMPLRT),a
        ld      a,(SRFLGA)      ;Save scan rate flag
        ld      (SFLAG),a
        call    SR_LD           ;Start the scan timer
        call    ACOLL           ;Collect 1st scan data
 ..2:   ld      a,(SRATEF)      ;Save scan rate entry
        ld      (SMPLRT),a
        ld      a,(SRFLGF)      ,Save scan rate flag
        ld      (SFLAG),a
        call    SR_LD           ;Load the scan rate timer
        ld      a,TRUE          ;Set long record flag
        ld      (RCDFLG),a
        ld      a,(FCH8)        :Load base channel
        ld      (FRSTCH),a
        ld      a,(FCHE)        ;Load last channel
        ld      (LSTCH) a
        ld      hl,(SCANF)      ;Load number of scans
        inc     hl              ;Increment for count
        ld      (SCANS),hl
        call    ATOD            ;Get data
```

```
        ld      a,(BANKNO)      ;Load current bank number
        ld      (LSTBNK),a      ;Save it
        out     (MUXDIS),a      ;Disable A-D multiplexer
        call    SENOFF          ;Disable power to sensors
        ld      hl,(SCANF)      ;Load number scans
        ld      (SCNSZ),hl      ;Save it
        ld      a,(CHNLF)       ;Load # channels
        ld      (SCNCH),a       ;Save it for data dump
        ld      a,(RFFLAG)      ;Load radio flag
        cp      FALSE           ;Does data go to radio?
        jp      z,SS_WRT        ;No, then try solid state
        cp      01h             ;All the time?
        jr      z, ..3          ;Yes, then skip next
        ld      a,(REFLG)       ;Is remote event set?
        cp      TRUE
        jp      nz,SS_WRT       ;No, then no write
..3:    call    DTDMP           ;Send the data
        jp      SS_WRT          ;Exit
        ENDPROC
;
;                       - COLLECT AND SEND INTERMEDIATE DATA SET
;
ICOLL:  PROC
        call    STAPE           ;Make sure tape is off
        ld      a,0F8h          ;Disable clock interrupts
        out     (INTPRTA),a
        ld      a,(PWRFLG)      ;Load power flag
        cp      FALSE           ;Is it set?
        jr      z, ..1          ;No, then skip next
        call    SENON           ;Enable power to sensors
        call    M1_DLY          ;Delay for sensor power up
;
;                       - LOAD CHANNEL 16 INTO HEADER
;
..1:    out     (MUXEN),a       ;Enable A-D multiplexer
        ld      hl,CH16H        ;Point to storage for CH16
        ld      (BUFFER),hl     ;Save it
        ld      b,0Fh           ;Load channel 16
        call    AQUIR           ;Get data

        ld      hl,DATAST       ;Load buffer start
        ld      (BUFFER),hl
        ld      a,(CHNLA)       ;Load 1st scan channels
        cp      00h             ;Is it zero?
        jr      z, ..2          ;Yes then skip 1st scan
        ld      a,(SRATEA)      ;Save scan rate entry
        ld      (SMPLRT),a
        ld      a,(SRFLGA)      ;Save scan rate flag
        ld      (SFLAG),a
        call    SR_LD           ;Load the scan rate timer
        call    ACOLL           ;Collect 1st scan data
..2:    ld      a,(SRATEI)      ;Save scan rate entry
        ld      (SMPLRT),a
        ld      a,(SRFLGI)      ;Save scan rate flag
        ld      (SFLAG),a
```

```
        call    SR_LD           ;Load the scan rate timer
        ld      a,FALSE         ;Clear long record flag
        ld      (RCDFLG),a
        ld      a,(ICHB)        ;Load base channel
        ld      (FRSTCH),a
        ld      a,(ICHE)        ;Load last channel
        ld      (LSTCH),a
        ld      hl,(SCANI)      ;Load number of scans
        inc     hl              ;Increment for count
        ld      (SCANS),hl
        call    ATOD            ;Get data
        ld      a,(BANKNO)      ;Load current bank number
        ld      (LSTBNK),a      ;Save it
        out     (MUXDIS),a      ;Disable A-D multiplexer
        call    SENOFF          ;Disable power to sensors
        ld      hl,(SCANI)      ;Load number scans
        ld      (SCNSZ),hl      ;Save it
        ld      a,(CHNLI)       ;Load # channels
        ld      (SCNCH),a       ;Save it for data dump
        ld      a,(RFFLAG)      ;Load radio flag
        cp      FALSE           ;Does data go to radio?
        jr      z,SS_WRT        ;No, then try solid state
        cp      01h             ;All the time?
        jr      z, ..3          ;Yes, then send data
        ld      a,(REFLG)       ;Is remote event set?
        cp      TRUE
        jr      nz,SS_WRT       ;No, then send no data
..3:    call    DTDMP           ;Send the data
        ENDPROC
;
;                       - WRITE DATA TO SOLID STATE
;
SS_WRT: ld      a,(SSFLAG)      ;Load solid-state flag
        cp      FALSE           ;Does data go to S-state?
        jr      z,CLK_DN        ;No, then exit
        cp      01h             ;All the time?
        jr      z,CLKDN1        ;Yes, then skip next
        ld      a,(REFLG)       ;Is remote event set?
        cp      TRUE
        jr      nz,CLK_DN       ;No, then no write
CLKDN1: call    SS_DMP          ;Write data to solid-state
;
;                       - SET CLOCK INTERRUPT
;
CLK_DN: PROC
        ld      hl,(SAMNUM)     ;Load sample number
        inc     hl              ;Increment it
        ld      (SAMNUM),hl
        ld      a,(ALARM)       ;Load alarm value
        out     (RTCSTAT),a
        ld      a,01Ch          ;Enable clock interrupt
        out     (RTCCMD),a
        ld      a,0F9h          ;Load 1st operational word
                                ;Enable INT1 & INT2 interrupts
        out     (INTPRTA),a
```

```
;
;                          - WRITE DATA TO TAPE
;
TP_CHCK:
        ld      a,(TPFLAG)          ;Load tape flag
        cp      FALSE               ;Does data go to tape?
        jr      z,SAM_DN            ;No, then exit
        cp      01h                 ;All the time?
        jr      z, ..1              ;Yes, then skip next
        ld      a,(REFLG)           ;Is remote event set?
        cp      TRUE
        jr      nz,SAM_DN           ;No, then no write
  ..1:  call    TP_WRT              ;Write data to tape
        ENDPROC
;
;                            - CLEAR INTERRUPTS AND WAIT FOR
;                              NEXT SAMPLE TIME
;
SAM_DN: ld      sp,STACK            ;Reinitialize stack pointer
        in      a,(RTCSTAT)         ;Clear any clock interrupts
        out     (ADINT),a           ;Clear any A-D interrupts
        ei                          ;Enable interrupts
        halt                        ;Wait for next interrupt



;
;--- COLLECT 1ST SCAN DATA SET ---
;
ACOLL:  out     (MUXEN),a           ;Enable A-D multiplexer
        ld      a,(ACHB)            ;Load base channel
        ld      (FRSTCH),a          ;Save it
        ld      a,(ACHE)            ;Load last channel
        ld      (LSTCH),a
        ld      hl,(SCANA)          ;Load number of scans
        inc     hl                  ;Increment for count
        ld      (SCANS),hl          ;Save it
        call    ATOD                ;Get data
        ret
```

```
;
; DATA-ACQUISITION ROUTINES
;-----------------------------------------------------------------
;
;--- COLLECT DATA INTO MEMORY ---
;                          - LSTCH = END CHANNEL OF EACH SCAN
;                          - FRSTCH = BASE CHANNEL OF EACH SCAN
;                          - SCANS = # OF SCANS TO ACQUIRE
;                          - BUFFER = START OF DATA BUFFER
;
ATOD:   PROC
        ld      a,06h           ;Load upper memory bank
        call    BNK_SEL         ;Select it
        ld      a,(LSTCH)       ;Load end channel
        cp      00h             ;Is it zero?
        ret     z               ;Yes then no data
        ld      de,01h          ;Load scan start
..1:    call    SR_STRT         ;Start the scan rate
        ld      a,(LSTCH)       ;Load end channel
        ld      b,a             ;Save it in B
..2:    dec     b               ;Decrement channel number
        call    AQUIR           ;Get the data
        ld      a,(FRSTCH)      ;Load base channel number
        cp      b               ;Is it the base channel?
        jr      nz, ..2         ;No, then get next channel
        ld      hl,(BUFFER)     ;Point to data buffer
        ld      a,(SCNFLG)      ;Load scan flag
        cp      FALSE           ;Is it set?
        jr      z, ..3          ;No then don't load scan #
        ld      (hl),e          ;Save low byte of scan count
        dec     hl              ;Decrement buffer
        ld      (hl),d          ;Save high byte of scan count
        dec     hl              ;Decrement buffer
..3:    ld      a,0FFh          ;Load end-of-scan value (FF)
        ld      (hl),a          ;Save end-of-scan value
        dec     hl              ;Decrement buffer pointer
        ld      (BUFFER),hl     ;Save new pointer
        dec     hl              ;Decrement pointer for compare
        inc     de              ;Increment scan number
        ld      a,h             ;Is high-byte pointer at 080h?
        cp      080h
        jr      nz, ..4         ;No, then continue
        ld      a,l
        cp      022h            ;Is Low-byte pointer at end?
        jr      c, ..6          ;Yes, then do next bank
..4:    ld      (BANKL),hl      ;Save address
        ld      hl,(SCANS)      ;Load total # of scans
        sbc     hl,de           ;Subtract the two
        jr      z, ..8          ;If zero then done
..5:    ld      a,0E8h          ;Read status of scan rate clock
        out     (CTRC),a
        in      a,(CTR2)        ;Get status
        and     080h            ;Strip all but done bit
        jr      z, ..5          ;Loop til set
        jp        ..1           ;Else, do next scan
```

```
;
 ..6:   ld      (BANKL),hl      ;Save memory pointer
        ld      a,(BANKNO)      ;Load current bank number
        cp      00h             ;Is it 0?
        ret     z               ;Yes, then done
        dec     a               ;Decrement bank #
        cp      04h             ;Is it 4?
        jr      nz, ..9         ;No, then skip next
        dec     a               ;Bank 4 is not used
 ..9:   call    BNK_SEL         ;Select the new bank
        cp      05h             ;Was it the 1st bank?
        jr      nz, ..7         ;No, then skip next
        ld      (BANK1),hl      ;Save memory pointer
 ..7:   ld      (BANK2),hl
        ld      hl,0EFFFh       ;Load new start for buffer
        ld      (BUFFER),hl     ;Save it
        jr       ..1
;
 ..8:   xor     a               ;Disable the scan rate clock
        out     (CTRGC),a
        ld      a,0B0h          ;Clear scan rate count
        out     (CTRC),a
        ENDPROC
        ret
;
;--- ACQUIRE A DATA WORD ---
;                       - SELECTS THE CHANNEL AND STARTS THE
;                         A-D CONVERSION
;
AQUIR:  ld      a,b             ;Load channel number
        out     (ADCG),a
AQUIR1: call    DELAY           ;Allow time for settling
        ei                      ;Enable interrupts
        out     (ADSTART),a     ;Start the A-D
        halt                    ;Wait for interrupt
        ret
;
;--- A-D INTERRUPT COMES HERE ---
;                       - ENTERS FIRST LOW BYTE THEN
;                         HIGH BYTE OF DATA INTO MEMORY
ATDINT: di
        ld      hl,(BUFFER)     ;Load buffer address
        in      a,(ADLO)        ;Get low-byte data
        ld      (hl),a          ;Store it
        dec     hl              ;Decrement buffer pointer
        in      a,(ADHI)        ;Get high-byte data
        ld      (hl),a          ;Store it
        dec     hl              ;Decrement buffer pointer
        ld      (BUFFER),hl     ;Save the pointer
        out     (ADINT),a       ;Clear A-D interrupt
        ex      (sp),hl         ;Correct stack
        dec     hl
        dec     hl
        ex      (sp),hl
        ret
```

```
;
;TEST ROUTINES
;-----------------------------------------------------------------
;
;--- SEND ASCII DATA TO RADIO TRANSMITTER ---
;
;                       - COMES HERE ON ^S ENTRY
;                       - WILL SEND INDEFINITELY OR
;                         UNTIL RESET.
;
ASCDP:  call    S0SET           ;Enable serial #0
        ld      a,21h           ;Load first ASCII value
ASCDP1: call    S0OUT           ;Send it
        inc     a               ;Increment value
        cp      07Eh            ;Is it last value?
        jr      nz,ASCDP1       ;No, then send next value
        call    S0OFF           ;Disable serial #0
        call    DELAY           ;Delay before repeating
        jr      ASCDP           ;Do again
;
;--- CLOCK TEST ---
;                       - DISPLAYS TIME AT SECONDS INTERRUPT
;                       - MUST RESET TO HALT
;
CLKTST: ld      a,08h           ;Set alarm mode for seconds
        out     (RTCSTAT),a
        ld      a,01Ch          ;Set clock interrupt
        out     (RTCCMD),a
        call    INTST           ;Enable interrupt controller
        halt                    ;Wait for interrupt
;
;--- A-D TEST ---
;                       - DISPLAYS SELECTED CHANNEL TO TERMINAL
;                       - ALL REGISTERS ARE ALTERED
;                       - COMPUTER MUST BE RESET TO EXIT
;
AD_TST: PROC
        ld      hl,ADMESS       ;Point to message
        call    PSTRG           ;Display it
        call    GETLN           ;Get answer
        ld      hl,TSTORE       ;Point to temporary storage
        ld      de,IBUFF        ;Point to first entry
        ld      a,(NCI)         ;Load number of entries
        cp      03h             ;Is it greater than 2?
        jr      nc,AD_TST       ;Yes then error
        ccf                     ;Clear the carry flag
        call    ASCHEX          ;Convert entry to hex
        jr      c,AD_TST        ;If carry, then error
        ld      a,(TSTORE)      ;Load hex value of channel
        cp      011h            ;Is a valid number?
        jr      nc,AD_TST       ;No, then error
        dec     a               ;Decrement channel number for AD
        out     (MUXEN),a       ;Enable A-D multiplexer
        out     (ADCG),a        ;Load channel number
        ld      a,018h          ;Load memory bank #4
```

```
        ld      (BANKNO),A      ;Save it
        out0    (BBR),a         ;Select it
        ld      hl,ADCHM        ;Point to message
        call    PSTRG           ;Display it
        ld      hl,TSTORE       ;Point to channel number
        call    PDEC            ;Display it and line feeds
        call    PCRLF
        call    PCRLF
        call    SENON           ;Enable power to sensors
        out     (ADINT),a       ;Clear any A-D interrupts
        call    INTST           ;Enable interrupt controller
..1:    ld      hl,0E04Ah       ;Point to buffer
        ld      (BUFFER),hl     ;Save it
        call    AQUIR1          ;Get data
        ld      hl,0E049h       ;Point to buffer
        ld      a,(hl)          ;Load high-byte of data
        call    HEXOUT          ;Display it
        inc     hl              ;Increment pointer
        ld      a,(hl)          ;Load low byte of data
        call    HEXOUT          ;Display it
        ld      a,CR            ;Display a carriage return
        call    S1OUT
        jr      ..1             ;Continue forever
        ENDPROC
;
;--- TAPE TEST ---
;                       - WRITES ONE BANK OF MEMORY TO
;                         TAPE EVERY 20 SECONDS
;                       - COMPUTER MUST BE RESET TO EXIT
;
TP_TST: halt
;       call    S20_DLY         ;Delay 20 seconds
;       ld      hl,08000h       ;Load end address for tape
;       ld      (BANKL),hl      ;Save end address
;       ld      (BANK1),hl      ;Save again
;       ld      a,07h           ;Load last bank
;       ld      (LSTBNK),a      ;Save it
;       call    TP_WRT          ;Write data to tape
;       jr      TP_TST          ;Do again
;
;--- TEST FOR REMOTE EVENTS ---
;                       - CHECKS PARALLEL PORTS A AND B
;                         FOR ANY EVENTS (A +5 VOLTS ON
;                         ANY BIT)
;                       - REGISTER A IS ALTERED
;
RE_TST: in      a,(PADATA)      ;Load event counter A
        ld      (RCTA),a        ;Save value
        in      a,(PBDATA)      ;Load event counter B
        ld      (RCTB),a        ;Save value
        cp      00h             ;Any events on B?
        jr      nz,RE_YES       ;Yes, then set remote mode
        ld      a,(RCTA)        ;Load counter A value
        cp      00h             ;Any events on A?
        jr      z,RE_NO         ;No, then test for prior events
```

```
;
;--- REMOTE EVENT DETECTED ---
;                           - REMOTE EVENT FLAG IS SET TO TRUE
;                           - THE REMOTE EVENT POWER IS TURNED
;                             ON (BIT 5 OF PARALLEL PORT C)
;                           - REGISTER A IS ALTERED
;
RE_YES: ld      a,REMPRON       ;Enable remote event power
        out     (PICNTL),a
        ld      a,TRUE          ;Set remote event flag
        ld      (REFLG),a
        call    RE_RST          ;Reset remote counters
        ret
;
;--- NO REMOTE EVENTS DETECTED ---
;                           - CHECKS TO SEE IF AN EVENT WAS THERE
;                             ON PREVIOUS CHECK
;                           - IF SO, THE REMOTE EVENT FLAG
;                             IS SET TO FALSE, AND THE REMOTE EVENT
;                             POWER IS TURNED OFF (BIT 5 OF
;                             PARALLEL PORT C)
;                           - REGISTER A IS ALTERED
;
RE_NO:  ld      a,(REFLG)       ;Was event there before?
        cp      TRUE
        ret     nz              ;No, then exit
        ld      a,REMPROFF      ;Disable remote power
        out     (PICNTL),a
        ld      a,FALSE         ;Clear remote event flag
        ld      (REFLG),a
        call    RE_RST          ;Reset remote counters
        ret
;
;--- RESET REMOTE EVENT COUNTERS ---
;                           - BIT 3 OF PARALLEL PORT C IS USED
;                             TO RESET THE COUNTERS USED FOR
;                             REMOTE EVENTS
;                           - SETTING BIT 3 LOW (0 VOLTS) RESETS
;                             THE COUNTERS
;                           - SETTING BIT 3 HIGH (+5 VOLTS)
;                             ENABLES THE COUNTERS
;                           - REGISTER A IS ALTERED
;
RE_RST: ld      a,REMCLR        ;Clear counters
        out     (PICNTL),a
        ld      a,REMEN         ;Enable counters
        out     (PICNTL),a
        ret
```

```
;
; RADIO TRANSMISSION SUBROUTINES
;-----------------------------------------------------------------------
;
;--- SEND A DATA RECORD USING SERIAL PORT 0 ---
;                          - RTS LINE IS SET TO ACT AS A
;                            TRANSMITTER ENABLE LINE
;                          - A .5 SECOND DELAY WILL OCCUR
;                            BEFORE SENDING DATA TO ALLOW THE
;                            TRANSMITTER TIME TO WARM UP
;                          - BANKL = ADDRESS OF LAST BYTE OF
;                            DATA IN REMAINING BANKS OF MEMORY
;                          - BANK1 = ADDRESS OF LAST BYTE OF
;                            DATA IN THE FIRST BANK
;                          - LSTBNK = VALUE OF THE LAST MEMORY
;                            BANK OF DATA
;                          - ALL REGISTERS ARE ALTERED
;
DTCMP:  PROC
        ld      a,RFON          ;Enable power to transmitter
        out     (PICNTL),a
        call    S0SET           ;Enable Serial # 0 port
        ld      hl,HEADST       ;Point to first of header
        ld      b,022h          ;Load header count
..1:    ld      a,(hl)          ;Load header byte
        call    S0OUT           ;Sent it
        dec     hl              ;Decrement data pointer
        djnz    ..1             ;Loop til done
        ld      a,06h           ;Select first bank
        call    BNK_SEL
        ld      hl,DATAST
        ld      de,(BANK1)      ;Load first bank end address
        ld      a,(LSTBNK)      ;Load last bank
        cp      06h             ;Is it only 1 bank?
        jr      nz, ..2         ;No, then skip next
        ld      de,(BANKL)      ;Else load last address
..2:    ld      a,(hl)          ;Load data byte
        call    S0OUT           ;Send it out
        dec     hl              ;Decrement buffer pointer
        ld      a,h             ;Get high byte
        cp      d               ;Is it at the end?
        jr      nz, ..2         ;No, then do next
        ld      a,l             ;Load low byte of address
        cp      e               ;Is it at the end?
        jr      nz, ..2         ;No, then do next
        call    S20_DLY         ;Wait before sending next bank
        ld      de,(BANK2)      ;Load new end address
        ld      a,(LSTBNK)      ;Load last bank
        ld      b,a             ;Move to reg. B
        ld      a,(BANKNO)      ;Load current bank #
        cp      b               ;Is it the last?
        jr      z, ..5          ;Yes, then exit
        dec     a               ;Decrement bank #
        cp      04h             ;Is it 4?
        jr      nz, ..3         ;No, then skip next
```

```
        dec     a               ;Bank 4 is not used
..3:    call    BNK_SEL         ;Select new bank
        ld      b,a             ;Move to reg. B
        ld      a,(LSTBNK)      ;Load current bank #
        cp      b               ;Is it the last?
        jr      nz, ..4         ;Yes, then skip next
        ld      de,(BANKL)      ;Load last address
..4:    ld      hl,0EFFFh       ;Load new starting address
        jr      ..2             ;Loop til done
;
..5:    call    S0OFF           ;Disable Serial #0 port
        ld      a,RFOFF         ;Disable power to transmitter
        out     (PICNTL),a
        ret
        ENDPROC
```

```
;
; DATAVAULT ROUTINES
;-----------------------------------------------------------------------
;
;--- TEST UNIT ---
;
;                          - CHECK THAT UNIT 0 IS PRESENT
;
SS_CMP: xor     a                   ,Load a zero
        ld      (UNIT_NO),a         ;Save unit #
        call    DV_SEL              ;Select unit 0
        call    DV_TST              ,Test if present
        ret     c                   ;If carry, then not present
;
;                          - READ BOOT SECTOR
;                          - DETERMINE STORAGE PARAMETERS
;
DV_STRT:
        ld      hl,00h              ;Load sector 0 for read
        ld      (SECT_NO),hl
        ld      hl,BOOT_SECT        ;Load buffer address
        ld      (DV_BUFF),hl        ;Save entry
        call    DOS_R1              ;Read the sector
        ld      de,(SCT_FAT)        ;Load Fat table size
        ld      a,(NO_FAT)          ;Load # of Fats
        call    MULTPLY             ;Multiply for # Fat sectors
        ld      hl,(F1_STRT)        ;Load offset to 1st fat
        add     hl,de               ;Add them
        ld      (RD_STRT),hl        ;Save offset to root directory
        ld      de,(F1_STRT)        ;Load offset to 1st Fat
        ld      hl,(SCT_FAT)        ;Load # sectors/Fat
        add     hl,de               ;Add them
        ld      (F2_STRT),hl        ;Save offset to 2nd Fat
        ld      de,(NO_DIR)         ;Load # of directory entries
        ld      a,020h              ;Load # bytes/directory entry
        call    MULTPLY             ;Multiply for # bytes for directory
```

```
;
;                       - NOTE: THE DATA VAULT ROUTINES REQUIRE
;                         WRITING 512 BYTES PER SECTOR. ANY OTHER
;                         FORMAT WILL NOT WORK.
;
        ousn    de                  ;Save # bytes for root directory
        ld      hl,DOSERR1          ;Load error message
        ld      bc,(BYT_SCT)        ;Load # bytes per sector
        ld      de,0200h            ;Load value for 512 bytes
        ld      a,d                 ;Load high byte into reg. A
        cp      b                   ;Are they the same?
        jp      nz,DOS_ERR          ;No, then error
        ld      a,e                 ;Load low byte into reg. A
        cp      c                   ;Are they the same?
        jp      nz,DOS_ERR          ;No, then error
        pop     de                  ;Else, restore # bytes for dir.
        ld      a,080h              ;Load 128
        call    DIVIDE              ;Divide
        ld      a,04h               ;Load 4
        call    DIVIDE              ;Divide again for 512 bytes/sector
        ld      hl,(RD_STRT)        ;Load offset to directory
        add     hl,de               ;Add in amount of directory space
        ld      (D_STRT),hl         ;Save start of data
        ex      de,hl               ;Move data start to DE
        ld      hl,(NO_SECT)        ;Load total # of sectors
        sbc     hl,de               ;Subtract for # available sectors
        ex      de,hl               ;Move result to DE
        ld      a,(SCT_CL)          ;Load number of sectors/cluster
        call    DIVIDE              ;Divide to get total # clusters
        ld      (NO_CL),de          ;Save result
;
;                       - READ ROOT DIRECTORY TO FIND THE
;                         START OF THE LAST FILE WRITTEN
;
        call    DIR_RD              ;Find last offset
        ld      a,d                 ;Load Hi-byte
        or      e                   ;Is offset zero?
        jr      nz,DV_ST2           ;No, then skip next
        ld      de,02h              ;Load first cluster
```

```
;
;                       - READ THE FAT TABLE TO FIND NEXT
;                         AVAILABLE CLUSTER
;
DV_ST2: ld      (CL_OFF),de     ;Save cluster offset
        ld      hl,00h          ;Clear fat table offset
        ld      (FAT_OFF),hl
        ld      de,0FF7h        ;Load compare value
        ld      hl,(NO_CL)      ;Load # clusters
        call    BCDCP           ;Compare them
        jr      c,DV_ST8        ;If smaller, use 12-bit Fat
        ld      de,(CL_OFF)     ;Load cluster number
        ex      de,hl           ;Move cluster # to HL
        add     hl,hl           ;Byte offset = cluster# X 2
        ld      (FAT_BUFF),hl   ;Save start of buffer
        jr      DV_STC
DV_ST8: ld      de,(CL_OFF)     ;Load cluster number
        call    CL_CVT          ;Convert cluster to byte offset
DV_STC: ld      hl,(F1_STRT)    ;Load 1st fat sector
        ld      (SECT_NO),hl    ;Save it for read
        call    FAT_TST         ;Check for sector #
        call    FAT_RD          ;Find next available cluster
;
;                       - CHECK CLUSTER NUMBER FOR MAXIMUM
;                         VALUE
;
DV_ST4: ld      hl,(CL_OFF)     ;Load current cluster
        ld      de,(NO_CL)      ;Load total # clusters
        call    BCDCP           ;Is it max?
        jr      c,DV_ST5        ;No, then continue
        ld      hl,DOSERR3      ;Display error message
        jp      DOS_ERR         ;Jump to error routine
;
;                       - UPDATE FAT TABLE FOR NEW FILE
;
DV_ST5: ld      hl,00h          ;Clear fat table offset
        ld      (FAT_OFF),hl
        ld      de,(CL_OFF)     ;Load cluster offset
        ld      de,0FF7h        ;Load compare value
        ld      hl,(NO_CL)      ;Load # clusters
        call    BCDCP           ;Compare them
        jr      c,DV_STD        ;If smaller, use 12-bit Fat
        ld      de,(CL_OFF)     ;Load cluster number
        ex      de,hl           ;Move cluster # to HL
        add     hl,hl           ;Byte offset = cluster# X 2
        ld      (FAT_BUFF),hl   ;Save start of buffer
        jr      DV_STE
DV_STD: ld      de,(CL_OFF)     ;Load cluster number
        call    CL_CVT          ;Convert cluster to byte offset
DV_STE: ld      hl,(F1_STRT)    ;Load 1st fat sector
        ld      (SECT_NO),hl    ;Save it for read
        call    FAT_TST         ;Check for sector #
        call    DOS_RD          ;Read the sector
```

```
;
;                       - DETERMINE NUMBER OF SECTORS TO WRITE
;
SS_STR: PROC
        ld      hl,00h          ;Clear the # sectors
        ld      (DATA_CNT),hl
        ld      (TTL_LO),hl     ;Clear low word of byte count
        ld      (TTL_HI),hl     ;Clear high word of byte count
        ld      bc,0600h        ;Load bank count (6 in B)
        ld      hl,0F021h       ;Load first starting address
        ld      de,(BANK1)      ;Load 1st bank
        inc     de              ;Increment count
        ld      a,(LSTBNK)      ;Load last bank
        cp      b               ;Is it only 1 bank?
        jr      nz, ..1         ;No, then skip next
        ld      de,(BANKL)      ;Else load last address
..1:    xor     a               ;Clear any carry
        sbc     hl,de           ;Subtract for total bytes
        ex      de,hl           ;Move total to reg. DE
        ld      hl,(TTL_LO)     ;Load total for record
        add     hl,de           ;Add in total for this bank
        ld      (TTL_LO),hl     ;Save new total
        jr      nc, ..2         ;If no carry, skip next
        ld      hl,(TTL_HI)     ;Load high word of total
        inc     hl              ;Increment it
        ld      (TTL_HI),hl     ;Save new value
..2:    ld      a,(LSTBNK)      ;Load last bank
        cp      b               ;Was it the last bank?
        jr      z, ..4          ;Yes, then done
        dec     b               ;Decrement bank count
        ld      a,04h
        cp      b               ;Is it bank 4?
        jr      nz, ..8         ;No, skip next
        dec     b               ;Skip bank 4
..8:    ld      hl,0EFFFFh      ;Load start address
        ld      de,(BANK2)      ;Load intermediate end address
        inc     de
        ld      a,(LSTBNK)
        cp      b               ;Is this the last bank?
        jr      nz, ..1         ;No, then proceed
        ld      de,(BANKL)      ;Else load last address
        inc     de
        jr      ..1
..4:    ld      hl,(TTL_HI)     ;Load high word of byte count
        ld      a,080h          ;Load # 512 bytes/high bit
        ld      h,a             ;Move it to reg. H
        mlt     hl              ;Multiply for total sectors
        ld      (DATA_CNT),hl   ;Save total
        ld      hl,(TTL_LO)     ;Load low word of byte count
        ld      de,0200h        ;Load number of bytes/sector
        ld      bc,01h          ;Load starting count
        xor     a               ;Clear any carry
..5:    sbc     hl,de           ;Subtract total by 512 bytes
        call    BCOCP           ;Is total < 512 bytes
        jr      c, ..6          ;Yes, then done
```

```
        inc     bc              ;Increment sector count
        jr      ..5             ;Loop til done
..6:    ld      a,h              Load high byte of remainder
        or      l               ;Is it zero?
        jr      z, ..7          ;Yes, then skip next
        inc     bc              ;Else, extra sector needed
..7:    ld      hl,(DATA_CNT)   ;Load total sectors
        add     hl,bc           ;Add in new total
        ld      (DATA_CNT),hl   ;Save new total
        ENDPROC
;
;                       - ADD NEW CLUSTER NUMBERS
;
        call    F_UPDT          ;Load the fat table
        ld      hl,DOSERR4      ;Point to error message
        jp      c,DOS_ERR       ;If carry, then error
        ld      hl,(SECT_NO)    ;Load sector number
        dec     hl              ;Decrement to rewrite fat
        ld      (SECT_NO),hl
        ld      hl,SECT_BUF     ;Load buffer address
        ld      (DV_BUFF),hl    ;Save it
        ld      hl,01h          ;Write 1 sector
        ld      (SECT_CNT),hl
        ld      a,DV_WRITE      ;Load a write command
        ld      (DV_CMD),a
        xor     a
        ld      (UNIT_NO),a     ;Load unit zero
        call    DV_INT          ;Read Fat into memory
        jp      c,DVT_E0
```

```
;
;                          - WRITE DATA
;
SS_DAT: PROC
        ld      de,(CL_OFF)       ;Load starting cluster
        dec     de                ;Decrement by 2
        dec     de
        ld      a,(SCT_CL)        ;Multiply result by # sectors
        call    MULTPLY
        ld      hl,(D_STRT)       ;Add to data start
        add     hl,de
        ld      (SECT_NO),hl      ;Save starting sector
        ld      a,06h             ;Select first bank
        call    BNK_SEL
        ld      hl,HEADST
        ld      de,SECT_BUF       ;Point to Sector buffer
        ld      bc,(BANK1)        ;Load first bank end address
        ld      a,(LSTBNK)        ;Load last bank
        cp      06h               ;Is it only 1 bank?
        jr      nz, ..1           ;No, then skip next
        ld      bc,(BANKL)        ;Else load last address
..1:    inc     bc
        ld      (TSTORE),bc       ;Save compare address
        ld      bc,0200h          ;Load count
..2:    ld      a,(hl)            ;Load data in reg. A
        ld      (de),a            ;save in sector buffer
        dec     hl                ;Decrement data pointer
        call    SS_CK             ;Check for end of bank
        ld      a,h               ;Check for end of data
        or      l                 ;Is it zero?
        jr      z, ..3            ;Yes then done
        inc     de                ;Increment sector pointer
        dec     bc                ;Decrement count
        ld      a,b               ;Load MSB in reg. A
        or      c                 ;Is it zero?
        jr      nz, ..2           ;No, the continue
        push    hl                ;Save data pointer
        ld      bc,01h            ;Load count
        ld      (SECT_CNT),bc
        ld      hl,SECT_BUF       ;Load buffer address
        ld      (DV_BUFF),hl      ;Save it
        ld      a,DV_WRITE        ;Load a write command
        ld      (DV_CMD),a
        xor     a
        ld      (UNIT_NO),a       ;Load unit zero
        call    DV_INT            ;Write sector
        jp      c,DVT_E0
        pop     hl                ;Restore pointers
        ld      de,SECT_BUF
        ld      bc,0200h          ;Load new count
        jp      ..2
..3:    ld      bc,01h            ;Load count
        ld      (SECT_CNT),bc
        ld      hl,SECT_BUF       ;Load buffer address
        ld      (DV_BUFF),hl      ;Save it
```

```
        ld      a,DV_WRITE      ;Load a write command
        ld      (DV_CMD),a
        xor     a
        ld      (UNIT_NO),a     ;Load unit zero
        call    DV_INT          ;Write sector
        jp      c,DVT_EO
;
;                       - UPDATE DIRECTORY
;
        ld      hl,(CL_OFF)     ;Save starting cluster
        ld      (CL_NXT),hl     ,
        call    DIR_RD          ;Find last entry
        push    hl              ;Save pointer
        ld      hl,(CL_OFF)     ;Save starting cluster
        ld      de,03h          ;Load compare value
        call    8CDCP           ;Is it the first cluster?
        pop     hl              ;Restore pointer
        jr      c, ..4          ;Yes, then skip next
        ld      de,05h          ;Increment back to start
        add     hl,de           ; of empty directory
..4:    ex      de,hl           ;Move pointer to reg. DE
        call    MK_LBL          ;Load file label
        ex      de,hl           ;Move pointer back to HL
        ld      de,010h         ;Load directory offset
        add     hl,de           ;Increment pointer
        push    hl              ;Save pointer
        ld      de,0800h        ;Load value for time convert
        ld      a,(MHR)         ;Load sample hour
        call    MULTPLY
        ex      de,hl           ;Save result in HL
        ld      de,020h         ;Load minute convert
        ld      a,(MMIN)        ;Load sample minute
        ld      d,a             ;Move it to reg. D
        mlt     de
        add     hl,de           ;Add them together
        ex      de,hl           ;Move result back to DE
        pop     hl              ;Restore pointer
        ld      (hl),e          ;Load low byte
        inc     hl              ;Increment pointer
        ld      (hl),d          ;Load high byte
        inc     hl              ;Increment pointer
        ld      a,(MMTH)        ;Load sample month
        add     a,a             ;X2
        push    af              ;Save result
        and     0Fh             ;Strip upper bits
        rl      a               ;Shift to upper bits
        rl      a
        rl      a
        rl      a
        ld      b,a             ;Save it in B
        pop     af              ;Restore result
        and     0F0h            ;Strip lower bits
        rr      a               ;Shift to lower bits
        rr      a
        rr      a
```

```
        rr      a
        ld      c,a             ;Save in C
        ld      a,(MDAY)        ;Load sample day
        add     a,b             ;Add to reg. B
        ld      b,a             ;Save result in reg. A
        ld      a,(MYEAR)       ;Load sample year
        sub     050h            ;Subtract 80
        add     a,a             ;X2
        add     a,c             ;Add to register C
        ld      (hl),b          ;Load high byte of date
        inc     hl              ;Increment pointer
        ld      (hl),a          ;Load low byte of date
        inc     hl              ;Increment pointer
        ld      de,(CL_NXT)     ;Load cluster start
        ld      (hl),e
        inc     hl
        ld      (hl),d
        inc     hl
        ld      de,(TTL_LO)     ;Save file size
        ld      (hl),e
        inc     hl
        ld      (hl),d
        inc     hl
        ld      de,(TTL_HI)
        ld      (hl),e
        inc     hl
        ld      (hl),d
        ld      hl,(SECT_NO)    ;Load sector number
        dec     hl              ;Decrement to rewrite root
        ld      (SECT_NO),hl
        ld      hl,SECT_BUF     ;Load buffer address
        ld      (DV_BUFF),hl    ;Save it
        ld      hl,01h          ;Write 1 sector
        ld      (SECT_CNT),hl
        ld      a,DV_WRITE      ;Load a write command
        ld      (DV_CMD),a
        xor     a
        ld      (UNIT_NO),a     ;Load unit zero
        call    DV_INT          ;Read Fat into memory
        jp      c,DVT_E0
        ret
        ENDPROC
;
;--- DOS ERROR EXIT ---
;                       - REGISTER HL MUST POINT TO ERROR
;                         MESSAGE
DOS_ERR:
        call    PSTRG           ;Display message
        ret
```

```
;
;--- MAKE A DOS FILE LABEL ---
;
MK_LBL: PROC
        push    de              ;Save directory pointer
        ld      hl,LABLP        ;Point to primary label
        ld      a,(RCDFLG)      ;Load primary data flag
        cp      TRUE            ;Is it a primary data set?
        jr      z, ..1          ;Yes, then skip next
        ld      hl,LABLI        ;Else, point to int. label
..1:    ld      bc,019h         ;Load count
        ldir                    ;Load label into memory
        ld      de,(SAMNO)      ;Load sample number
        ld      h,e             ;Invert it
        ld      l,d
        xor     a               ;Clear reg. AF
        ex      af,af'
        xor     a
        call    DEC16           ;Convert to ASCII
        ld      a,03h           ;Load number of characters
        ld      b,a             ;Save in B
        dec     a               ;Decrement count
        ld      hl,IBUFF        ;Point to buffer
        ld      de,00h          ;Clear DE
        ld      e,a             ;Move count to E
        add     hl,de           ;Add them
        pop     de              ;Restore pointer
..3:    ld      a,30h           ;Load ASCII mask
        rld                     ;Rotate in upper bits
        ld      (de),a          ;Load into label
        inc     de              ;Increment pointer
        rld                     ;Rotate in lower bits
        ld      (de),a          ;Load into label
        rld                     ;Restore original number
        inc     de
        dec     hl              ;Loop til done
        djnz    ..3
        ENDPROC
        ret
```

```
;
; --- TEST FOR END OF MEMORY BANK ---
;
SS_CK:  PROC
        push    de              ;Save pointer
        push    bc
        ld      de,(TSTORE)     ;Load end address
        ld      a,h             ;Get high byte
        cp      d               ;Is it at the end?
        jr      nz, ..6         ;No, then do next
        ld      a,l             ;Load low byte of address
        cp      e               ;Is it at the end?
        jr      nz, ..5         ;No, then do next
        ld      de,(BANK2)      ;Load new end address
        ld      a,(LSTBNK)      ;Load last bank
        ld      b,a             ;Move to reg. B
        ld      a,(BANKNO)      ;Load current bank #
        cp      b               ;Is it the last?
        jr      z, ..5          ;Yes, then exit
        dec     a               ;Decrement bank #
        cp      04h             ;Is it 4?
        jr      nz, ..3         ;No, then skip next
        dec     a               ;Bank 4 is not used
..3:    call    BNK_SEL         ;Select new bank
        ld      b,a             ;Move to reg. B
        ld      a,(LSTBNK)      ;Load current bank #
        cp      b               ;Is it the last?
        jr      nz, ..4         ;Yes, then skip next
        ld      de,(BANKL)      ;Load last address
..4:    inc     de
        ld      (TSTORE),de     ;Save new end address
        pop     bc
        pop     de
        ld      hl,0EFFFh       ;Load new starting address
        ret
;
..5:    ld      hl,00h          ;Clear pointer to set end
..6:    pop     bc
        pop     de
        ret
        ENDPROC

;
;--- UPDATE THE FAT ---
;
F_UPDT: PROC
        ld      hl,(CL_OFF)     ;Load starting cluster
        ld      de,(DATA_CNT)   ;Load # sectors to write
        add     hl,de           ;Add to current
        ex      de,hl           ;Move total to reg. DE
        ld      hl,(NO_CL)      ;Compare with total available
        call    BCDCP           ;Is it MAX?
        ret     c               ;Yes then error
```

```
;
;                       - CHECK FOR 12 OR 16-BIT ALLOCATION
;
        ld      de,0FF7h        ;Load compare value
        ld      hl,(NO_CL)      ;Load # clusters
        call    BCDCP           ;Compare them
        jr      c, ..3          ;If smaller, use 12-bit Fat
        ld      de,(CL_OFF)     ;Reload starting cluster
        ld      bc,(DATA_CNT)   ;Load count
        dec     bc              ;Do all but last cluster
        ld      a,b             ;Load MSB of count
        or      c               ;Is zero?
        jr      z, ..2          ;Yes,then write end of file
..1:    inc     de              ;Increment for next value
        ld      (CL_NXT),de     ;SAve value to be written
        dec     de              ;Restore original cluster
        push    bc              ;Save sector count
        call    F16_WT          ;Update fat
        pop     bc              ;Restore count
        ld      de,(CL_NXT)     ;Load cluster #
        dec     bc              ;Decrement count
        ld      a,b             ;Load MSB into reg. A
        or      c               ;Is it zero?
        jr      nz, ..1         ;Loop for all clusters
..2:    ld      hl,0FFFFh       ;Load end of file mark
        ld      (CL_NXT),hl     ;SAve it
        call    F16_WT          ;Write end of file
        xor     a               ;Clear any carry
        ret
;
..3:    ld      de,(CL_OFF)     ;Reload starting cluster
        ld      bc,(DATA_CNT)   ;Load count
        dec     bc              ;Do all but last cluster
        ld      a,b             ;Load MSB of count
        or      c               ;Is zero?
        jr      z, ..5          ;Yes,then write end of file
..4:    inc     de              ;Increment for next value
        ld      (CL_NXT),de     ;SAve value to be written
        dec     de              ;Restore original cluster
        push    bc              ;Save sector count
        call    F12_WT          ;Update fat
        pop     bc              ;Restore count
        ld      de,(CL_NXT)     ;Load cluster #
        dec     bc              ;Decrement count
        ld      a,b             ;Load MSB into reg. A
        or      c               ;Is it zero?
        jr      nz, ..4         ;Loop for all clusters
..5:    ld      hl,0FFFh        ;Load end of file mark
        ld      (CL_NXT),hl     ;SAve it
        call    F12_WT          ;Write end of file
        xor     a               ;Clear any carry
        ret
        ENDPROC
```

```
;
;--- UPDATE AVAILABLE CLUSTER (16-BIT)
;
F16_WT: ld      de,(CL_OFF)     ;Load cluster number
        ex      de,hl           ;Move cluster # to HL
        add     hl,hl           ;Byte offset = cluster# X 2
        ld      de,SECT_BUF     ;Point to start of buffer
        add     hl,de           ;Add byte offset
        call    FAT_CK          ;Check for end of sector
        ld      de,(CL_NXT)     ;Load cluster value
        ld      (hl),e          ;Load low byte of offset
        inc     hl              ;Increment pointer
        call    FAT_CK          ;Check if end of sector
        ld      (hl),d          ;Load high byte of offset
        ret
;
;--- UPDATE AVAILABLE CLUSTER (12-BIT)---
;
F12_WT: PROC
        call    CL_CVT          ;Convert cluster to byte offset
        ld      hl,SECT_BUF     ;Point to start of buffer
        xor     a               ;Clear A
        add     hl,de           ;Add byte offset
        cp      b               ;Is remainder zero?
        jr      z, ..1          ;Yes, then start at even
        call    FAT_CK          ;Check for end of sector
        ld      de,(CL_NXT)     ;Load cluster # to write
        ld      a,e             ;Load low byte
        and     0Fh             ;Strip upper bits
        rrd                     ;Shift into buffer
        or      (hl)            ;Add existing value back in
        ld      (hl),a          ;Save it
        inc     hl              ;Increment buffer
        call    FAT_CK          ;Check for end of sector
        ld      (hl),e          ;Put lo-byte into buffer
        ld      a,d             ;Load hi-byte into reg. A
        rrd                     ;Shift upper bit into buffer
        ret
  .1:   call    FAT_CK          ;Check if end of sector
        ld      de,(CL_NXT)     ;load next value
        ld      (hl),e          ;Save low byte of offset
        inc     hl              ;Increment pointer
        call    FAT_CK          ;Check if end of sector
        ld      (hl),a          ;Load high byte of offset
        ENDPROC
        ret
```

```
;
FAT_CK: PROC
        push    af
        push    bc
        push    de                  ;Save regs. DE
        push    hl
        ld      de,DOS_EQ           ;Load end of sector
        call    BCDCP               ;Compare with current address
        jr      c, ..1              ;If smaller then exit
        pop     hl
        ld      hl,(SECT_NO)        ;Load sector number
        dec     hl                  ;Decrement count
        ld      (SECT_NO),hl        ;Resave it
        ld      hl,SECT_BUF         ;Point to buffer
        ld      (DV_BUFF),HL        ;Save it
        ld      hl,01h              ;Load 1 sector
        ld      (SECT_CNT),hl
        ld      a,DV_WRITE          ;Load a write command
        ld      (DV_CMD),a
        xor     a
        ld      (UNIT_NO),a         ;Load unit zero
        call    DV_INT              ;Read next Fat sector
        jp      c,DVT_ER
        ld      hl,(SECT_NO)        ;Load current sector #
        ld      de,07h              ;Load compare value
        call    BCDCP               ;Is it last sector?
        jr      c, ..3              ;No then continue
        ld      hl,DOSERR3          ;Load error message
        jp      DOS_ERR             ;Exit
..3:    call    DOS_RD              ;Read next sector
        ld      hl,(FAT_OFF)        ;Load last offset
        ld      de,0200h            ;Add 512 bytes
        adc     hl,de
        ld      (FAT_OFF),hl        ;Save for correction factor
        ld      hl,SECT_BUF         ;Load new pointer
        jr      ..2
..1:    pop     hl
..2:    pop     de                  ;Restore regs.
        pop     bc
        pop     af
        ENDPROC
        ret
```

```
;
;--- READ THE FAT TABLE ---
;                           - DETERMINES IF 12 OR 16-BIT ALLOCATION
;                           - STARTS WITH STARTING CLUSTER OF LAST
;                             FILE (CL_OFF) AND TRACES THROUGH THE
;                             CLUSTER #'S UNTIL IT FINDS AN EMPTY ONE
;
FAT_RD: PROC
        call    DOS_RD
        ld      de,0FF7h        ;Load compare value
        ld      hl,(NO_CL)      ;Load # clusters
        call    BCDCP           ;Compare them
        jr      c, ..4          ;If smaller, use 12-bit Fat
;
;                           - FIND THE NEXT AVAILABLE CLUSTER (16-BIT)
;
        ld      de,(CL_OFF)     ;Load cluster number
..1:    ex      de,hl           ;Move cluster # to HL
        add     hl,hl           ;Byte offset = cluster# X 2
        ld      de,SECT_BUF     ;Point to start of buffer
        add     hl,de           ;Add byte offset
..2:    call    SECT_CK         ;Check for end of sector
        ld      e,(hl)          ;Load low byte of offset
        inc     hl              ;Increment pointer
        call    SECT_CK         ;Check if end of sector
        ld      a,(hl)          ;Load high byte of offset
        ld      d,a             ;Save in reg. D
        or      e               ;Is it zero?
        jr      z, ..3          ;Yes then done
        cp      0FFH            ;Is it a valid cluster #?
        jr      c, ..1          ;Yes, then do next cluster
        ld      a,e             ;Load low byte into reg. A
        cp      0F0h            ;Is it a valid cluster #?
        jr      c, ..1          ;Yes, then do next cluster
        inc     hl              ;ELSE, increment pointer
        jr       ..2            ;Else, skip to next byte
..3:    xor     a               ;Clear carry
        dec     hl              ;Adjust back to first byte
        ld      de,SECT_BUF     ;Load starting address
        sbc     hl,de           ;Subtract for the offset
        ld      (FAT_BUFF),hl   ;Save starting cluster pointer
        ld      de,(FAT_OFF)    ;Load sector offset
        add     hl,de           ;Add it in
        ex      de,hl           ;Move result to reg. DE
        ld      a,02h           ;LOad divisor = 2
        call    DIVIDE          ;Divide to get cluster number
        ld      (CL_OFF),de     ;Save result
        ret
```

```
;
;                       - FIND THE NEXT AVAILABLE CLUSTER (12-BIT)
;
  ..4:    ld      de,(CL_OFF)     ;Load byte offset
  ..5.    call    CL_CVT          ;Convert cluster to byte offset
          ld      hl,SECT_BUF     ;Point to start of buffer
          xor     a               ;Clear A
          adc     hl,de           ;Add byte offset
          cp      b               ;Is remainder zero?
          jr      z, ..7          ;Yes, then start at even
  ..6:    call    SECT_CK         ;Check for end of sector
          xor     a               ;Clear A
          rld                     ;Shift lower 4 bits into A
          inc     hl              ;Increment pointer
          call    SECT_CK         ;Check if end of sector
          rld                     ;A = MSB, (hl) = LSB
          ld      d,a             ;Load MSB into reg. D
          ld      e,(hl)          ;Load LSB into reg. E
          ld      a,d             ;Load hi-byte
          or      e               ;Is it zero?
          jr      z, ..8          ;Yes, then done
          ld      a,d             ;Reload MSB
          cp      0Fh             ;Is it a valid cluster
          jr      nz, ..5         ;Yes, then read next
          ld      a,e             ;Load lo-byte
          cp      0F0h            ;Is it a valid cluster?
          jr      c, ..5          ;Yes, then read next
          inc     hl              ;Increment pointer
  ..7:    call    SECT_CK         ;Check if end of sector
          ld      e,(hl)          ;Load low byte of offset
          inc     hl              ;Increment pointer
          call    SECT_CK         ;Check if end of sector
          ld      a,(hl)          ;Load high byte of offset
          and     0Fh             ;Strip upper bits
          ld      d,a             ;Save in reg. D
          or      e               ;Is it zero?
          jr      z, ..8          ;Yes then done
          ld      a,d             ;Reload MSB
          cp      0Fh             ;Is it a valid cluster
          jr      nz, ..5         ;Yes, then read next
          ld      a,e             ;Load lo-byte
          cp      0F0h            ;Is it a valid cluster?
          jr      c, ..5          ;Yes, then read next
          jr        ..6           ;Else, skip to next byte
  ..8:    xor     a               ;Clear carry
          dec     hl              ;Adjust back to first byte
          ld      de,SECT_BUF     ;Load starting address
          sbc     hl,de           ;Subtract for the offset
          ld      (FAT_BUFF),hl   ;Save starting cluster pointer
          ld      de,(FAT_OFF)    ;Load sector offset
          add     hl,de           ;Add it in
          ex      de,hl           ;Move result to reg. DE
          call    BT_CVT          ;Convert to cluster number
          ret
          ENDPROC
```

```
;--- CHECK FAT COUNTER ---
;
SECT_CK:PROC
        push    af
        push    bc
        push    de              ;Save regs. DE
        push    hl
        ld      de,DOS_EO       ;Load end of sector
        call    BCDCP           ;Compare with current address
        jr      c, ..1          ;If smaller then exit
        pop     hl
        call    DOS_RD
        ld      hl,(FAT_OFF)    ;Load last offset
        ld      de,0200h        ;Add 512 bytes
        adc     hl,de
        ld      (FAT_OFF),hl    ;Save for correction factor
        ld      hl,SECT_BUF     ;Load new pointer
        jr        .2
  .1:   pop     hl
  .2:   pop     de              ;Restore regs.
        pop     bc
        pop     af
        ENDPROC
        ret
; --- CONVERT CLUSTER OFFSET TO SECTOR OFFSET ---
;
FAT_TST:PROC
        ld      hl,(FAT_BUFF)   ;Load byte offset
        ld      de,0200h        ;Load compare value
        call    BCDCP           ;Is offset smaller?
        ret     c               ;Yes then sector 1
        sbc     hl,de           ;Subtract first sector
        ld      bc,02h          ;Load count for # sectors
  ..1:  ld      (SECT_NO),bc    ;Save current sector number
        ld      (FAT_BUFF),hl   ;Save new pointer
        push    hl              ;Save new pointer
        ld      hl,(FAT_OFF)    ;Load offset value
        add     hl,de           ,Add new sector
        ld      (FAT_OFF),hl    ;Save new value
        pop     hl              ;Restore offset
        call    BCDCP           ;Compare new offset
        ret     c               ;If smaller then done
        sbc     hl,de           ;Subtract sector
        inc     bc              ;Increment sector count
        ld      a,05h           ;Load compare value
        cp      c               ;Is it max?
        jr      nc, ..1         ;No, then do next sector
        push    hl              ;SAve pointer
        ld      de,01D7h        ;Load offset
        ld      hl,(FAT_BUFF)   ,Load current offset
        call    BCDCP           ;Is it max?
        pop     hl              ;Restore pointer
        jr      c, ..1          ;If less then valid
        ld      hl,DOSERR3      ;Point to error message
        jp      DOS_ERR         ;Jump to error routine
```

```
        ENDPROC
;
;--- CONVERT CLUSTER NUMBER TO BYTE OFFSET ---
;
CL_CVT: ld      a,03h           ;Multiply by 3
        call    MULTPLY
        ld      a,02h           ;Divide by 2
        call    DIVIDE          ; = multiply by 1.5
        xor     a               ;Clear any carry
        ex      de,hl           ;Move DE to HL
        ld      de,(FAT_OFF)    ;Load Fat table offset
        sbc     hl,de           ;subtract for sector offset
        ld      (FAT_BUFF),hl   ;Save Fat pointer
        ex      de,hl           ;Move result back to reg. DE
        ret
;
;--- CONVERT BYTE OFFSET TO CLUSTER NUMBER ---
;
BT_CVT: PROC
        ld      a,02h           ;Multiply by 2
        call    MULTPLY
        ld      a,03h           ;Divide by 3
        call    DIVIDE          ; = divide by 1.5
        xor     a               ;Clear reg. A
        cp      b               ;Was there a remainder
        jr      z, ..1          ;No then skip next
        inc     de              ;Else, increment offset
..1:    ld      (CL_OFF),de     ;Save result
        ENDPROC
        ret
```

```
;
;--- READ THE ROOT DIRECTORY---
;                          - FINDS THE STARTING CLUSTER OF THE
;                            LAST FILE ENTERED
;                          - REGISTER DE CONTAINS THE STARTING
;                            CLUSTER ON EXIT
DIR_RD: PROC
        ld      hl,(RD_STRT)     ;Load Start sector
        ld      (SECT_NO),hl
        call    DOS_RD           ;Read the sector
        ld      de,00h           ;Clear regs. DE
        ld      hl,SECT_BUF      ;Point to start of sector
        ld      a,(hl)           ;Load first byte
        cp      00h              ;is it zero?
        ret     z
 ..1:   ld      a,(hl)           ;Load first byte of directory
        cp      00h              ;Is it zero?
        jr      z, ..2           ;Yes, then done
        xor     a                ;Clear any carry
        ld      de,020h          ;Load address offset
        add     hl,de            ;Add to pointer
        call    DIR_TST          ;Check for end
        jr       ..1
 ..2:   xor     a                ;Clear any carry
        ld      de,C5h
        sbc     hl,de            ;Subtract back to last entry
        ld      e,(hl)           ;Load lo-byte
        inc     hl               ;Increment pointer
        ld      d,(hl)           ;Load hi-byte
        ret
        ENDPROC
;
; --- CHECK DIRECTORY SEARCH FOR END OF SECTOR ---
;                          - HL MUST CONTAIN THE CURRENT ADDRESS
;                            OF THE SECTOR BUFFER
DIR_TST:PROC
        ld      de,DOS_EO        ;Load compare value
        call    BCDCP            ;Is offset smaller?
        ret     c                ;Yes then sector 1
        ld      hl,(SECT_NO)     ;Load current sector #
        ld      de,(D_STRT)      ;Load end of directory
        call    BCDCP            ;Is it at the end?
        jr      nc, ..1          ;Yes, then error
        call    DOS_RD
        ld      hl,SECT_BUF      ;Load start of buffer
        ret
 ..1:   ld      hl,DOSERR2       ;Load error message
        jp      DOS_ERR          ;Jump to error routine
        ENDPROC
```

```
;
;--- SEND A COMMAND TO READ ONE SECTOR ---
;                          - STARTING SECTOR MUST BE IN SECT_NO
;                            ON ENTRY
;
DOS_RD: ld      hl,SECT_BUF     ;Load buffer address
        ld      (DV_BUFF),hl    ;Save it
DOS_R1: ld      hl,01h          ;Read only one sector
        ld      (SECT_CNT),hl
        ld      a,DV_READ       ;Load a read command
        ld      (DV_CMD),a
        xor     a
        ld      (UNIT_NO),a     ;Load unit zero
        call    DV_INT          ;Read directory into memory
        jp      c,DVT_E0        ;If carry, then error
        ret
;
;--- PROCESS A COMMAND ---
;
DV_INT: call    DV_SEL          ;Select unit
        ld      bc,00h          ;Clear # sectors transferred
        ld      (SECT_TRNS),bc
        ld      bc,(SECT_CNT)   ;Load sector count
DV_LP:  ld      a,(DV_CMD)      ;Load command entry
        cp      DV_READ         ;Is it a read command?
        jr      z,DVI_RD        ;Yes, then read
        cp      DV_WRITE        ;Is it a write command?
        jr      z,DVI_WRT       ;Yes then write
        ld      a,BAD_CMD       ;Else load error
        scf                     ;Set carry for error
        ret
;
;                          - PROCESS A READ COMMAND
;
DVI_RD: ld      a,DV_OPEN
        or      DV_LOPEN
        or      DV_POWOFF
        call    DV_STAT         ;Is drive ready?
        call    nc,DV_RD        ;Yes, then read a sector
        jr      DVI_CC
;
;                          - PROCESS A WRITE COMMAND
;
DVI_WRT:
        ld      a,DV_OPEN
        or      DV_LOPEN
        or      DV_POWOFF
        call    DV_STAT         ;Is drive ready?
        call    nc,DV_WRT       ;Yes, then read a sector
```

```
;
;                       - COMMAND COMPLETE
;
DVI_CC: ret     c               ;Carry sets error
        push    hl              ;Save HL
        ld      hl,(SECT_NO)    ;Load sector number
        inc     hl              ;Increment number
        ld      (SECT_NO),hl    ;Save updated number
        dec     bc              ;Increment sectors transferred
        ld      hl,(SECT_CNT)   ;Load total # sectors
        xor     a               ;Clear any carry
        sbc     hl,bc           ;Subtract current #
        ld      (SECT_TRNS),hl  ;Save new value
        pop     hl              ;Restore HL
        ld      a,b             ;Load MSB of count
        or      c               ;Is it zero?
        jr      nz,DV_LP        ;Loop til done
        xor     a               ;Clear A for no errors
        ret
;
;                       - DISPLAY ERROR
;
DVT_ER: push    af              ;Save status
        ld      hl,DVER_MSG     ;Display error message
        call    PSTRG
        pop     af              ;Restore status
        call    HEXOUT          ;Display it
        call    PCRLF           ;Display a return
        ret
```

```
;
;---- READ A SECTOR ---
;
DV_RD:  push    bc              ;Save registers
        push    de
        push    hl
;
;                       - SEEK BEGINNING OF SECTOR
;
        ld      hl,(SECT_NO)    ;Load sector no.
        add     hl,hl           ;Convert # into 512 byte block
        ex      de,hl           ;Move to DE
        ld      a,e             ;Load low byte of sector
        out     (PORT_A),a      ;Send it
        ld      a,d             ;Load high byte of sector
        out     (PORT_B),a      ;Send it
        ld      a,DV_LED        ;Turn on LED
        or      DV_PWR          ;Turn off CRC memory
        out     (PORT_C),a
        ld      a,CRC_RESET     ;?
        out     (CRC_RESET),a   ;Reset CRC
        ld      c,DATA_PORT     ;Load port address
        ld      hl,(DV_BUFF)    ;Load buffer address
        ld      b,0             ;Count = 256
        inir                    ;read data
        inc     e               ;Increment lo-byte
        ld      a,e             ;Move it to A
        out     (PORT_A),a      ;Send it
        ld      c,DATA_PORT     ;Point to data port
        ld      b,0             ;Load count
        inir                    ;Send it
;
;                       - READ CRC
;
        in      a,(CRC_LO)      ;Read generated lo-byte CRC
        ld      e,a             ;Save in E
        in      a,(CRC_HI)      ;Read generated hi-byte CRC
        ld      d,a             ;Save in D
        ld      (GEN_CRC),de    ;Save CRC word

        ld      a,DV_LED
        or      DV_CRCMEM
        or      DV_PWR
        out     (PORT_C),a      ;Turn on CRC memory
        in      a,(DATA_PORT)
        ld      d,a             ;Load high byte of word
        in      a,(PORT_A)      ;Get sector count
        dec     a               ;Decrement count
        out     (PORT_A),a      ;Reload it
        in      a,(DATA_PORT)   ;Get low byte
        ld      e,a             ;Load low byte of CRC word
        ld      (STOR_CRC),de   ;Save it
        ld      a,DV_PWR        ;Turn off CRC memory
        out     (PORT_C),a      ;
;
```

```
;                       - COMPARE CRC VALUES
;
        ld      bc,(GEN_CRC)    ;Generated in BC
        ld      a,b             ;Compare hi-bytes
        cp      d
        jr      nz,RD_ERR       ;NO, then error
        ld      a,c             ;Compare lo-bytes
        cp      e
        jr      nz,RD_ERR       ;NO, then error
;
;                       - UPDATE BUFFER POINTER
;
        ld      (DV_BUFF),hl    ;Save HL
        xor     a               ;Clear for no errors
RD_RET: pop     hl              ;Restore regs.
        pop     de
        pop     bc
        ret
RD_ERR: ld      a,BAD_CRC       ;Load error
        scf                     ;Set carry for error
        jr      RD_RET

;
;--- WRITE A SECTOR OF DATA ---
;
DV_WRT: push    bc              ;Save registers
        push    de
        push    hl
;
;                       - SEEK BEGINNING OF SECTOR
;
        ld      hl,(SECT_NO)    ;Load sector no.
        add     hl,hl           ;Convert # into 512 byte block
        ex      de,hl           ;Save in DE
        ld      a,e             ;Load low byte of sector
        out     (PORT_A),a      ;Send it
        ld      a,d             ;Load high byte of sector
        out     (PORT_B),a      ;Send it
        ld      a,DV_LED        ;Turn on LED
        or      DV_PWR          ;Turn off CRC memory
        out     (PORT_C),a
        ld      a,CRC_RESET     ;?
        out     (CRC_RESET),a   ;Reset CRC
        ld      c,DATA_PORT     ;Load port address
        ld      hl,(DV_BUFF)    ;Load buffer address
        ld      b,0             ;Count = 256
        otir                    ;Write data
        inc     e               ;Increment lo-byte
        ld      a,e             ;Move it to A
        out     (PORT_A),a      ;Send it
        ld      c,DATA_PORT     ;Point to data port
        ld      b,0             ;Load count
        otir                    ;Send it
```

```
;
;                       - READ CRC
;
        in      a,(CRC_LO)      ;Read generated lo-byte CRC
        ld      e,a             ;Save in E
        in      a,(CRC_HI)      ;Read generated hi-byte CRC
        ld      d,a             ;Save in D
        ld      (GEN_CRC),de    ;Save CRC word
        ld      a,DV_LED
        or      DV_CRCMEM
        or      DV_PWR
        out     (PORT_C),a      ;Turn on CRC memory
        ld      a,d             ;Load high byte of word
        out     (DATA_PORT),a
        in      a,(PORT_A)      ;Load current lo-byte sector
        dec     a               ;Decrement sector
        out     (PORT_A),a      ;Send it
        ld      a,e             ;Load low byte of CRC word
        out     (DATA_PORT),a   ;Send it
        ld      a,DV_PWR        ;Turn off CRC memory
        out     (PORT_C),a      ;
;
;                       - UPDATE BUFFER POINTER
;
        ld      (DV_BUFF),hl    ;Save HL
        xor     a               ;Clear for no errors
        pop     hl              ;Restore regs.
        pop     de
        pop     bc
        ret
;
;--- SELECT UNIT ---
;                       - UNIT_NO MUST HAVE #
;
DV_SEL: PROC
        ld      a,(UNIT_NO)     ;Load unit number
        cp      00h             ;Is it unit 0?
        jr      nz, ..1         ;No then skip next
        in      a,(UNIT_SEL)    ;Read to select 0
        ret
..1:    out     (UNIT_SEL),a    ;Send to select unit 1
        ret
        ENDPROC
```

```
;
;--- TEST UNIT STATUS ---
;                          - A MUST CONTAIN STATUS MASK
;                          - CARRY IS SET IF ERROR
;
DV_STAT:
        push    bc              ;Save reg.
        ld      b,a             ;Save status mask in B
        in      a,(PORT_C)      ;Read port C
        xor     DV_WRENB        ;Invert write enable bit
        and     b               ;Strip mask
        tst     DV_POWOFF       ;Is power off?
        jr      z,DS_1          ;No, then skip next
        ld      b,a             ;Save status in B
        ld      a,DV_PWR        ;Set power bit
        out     (PORT_C),a
        in      a,(PORT_C)      ;Get status
        tst     DV_POWOFF       ;Is power still off?
        jr      nz,DS_2         ;Yes, the no cartridge

        push    bc              ;Save BC
        ld      b,0FFh          ;Load count for delay
DS_OLP: push    bc
        ld      b,0FFh
DS_ILP: djnz    DS_ILP          ;Loop for delay
        pop     bc
        djnz    DS_OLP          ;Outer loop
        pop     bc              ;Restore BC
        ld      a,b             ;Get status
;
DS_1:   tst     DV_OPEN         ;Is door open?
        jr      nz,DS_2         ;Yes, then exit
        tst     DV_LOPEN        ;Latched door open?
        jr      nz,DS_3         ;Yes then exit
        tst     DV_WRENB        ;Write enabled?
        jr      nz,DS_5         ;No, then exit
        xor     a               ;Clear any flags
        pop     bc              ;Restore reg.
        ret
DS_2:   ld      a,TIME_OUT      ;Load error
        jr      DS_6            ;exit
DS_3:   ld      a,MEDIA_CHNG    ;Load error
DS_4:   jr      DS_6            ;Exit
DS_5:   ld      a,WRT_PROT      ;Load error
DS_6:   scf                     ;Set carry
        pop     bc              ;Restore reg.
        ret
```

```
;
;--- TEST IF UNIT PRESENT ---
;
;                          - CARRY IS SET IF NOT
;
UP_PAT: db      0FFh,055h,0CCh,0F0h
;
DV_TST: push    bc              ;Save reg.
        push    de
        push    hl
        ld      a,088h          ;Configure 82C55
        out     (CTRL_PORT),a   ;Lo-C input, rest output
        ld      hl,UP_PAT       ;Point to patterns
UP_LP:  ld      a,(hl)          ;Load pattern byte
        ld      b,a             ;Save in B
        out     (PORT_A),a
        cpl                     ;Invert pattern
        ld      c,a             ;Save in C
        out     (PORT_B),a      ;Load in port B
        in      a,(PORT_A)      ;Read port A
        ld      d,a             ;Save in d
        in      a,(PORT_B)      ;Read port B
        cp      c               ;Is it the same?
        jr      nz,NO_DV        ;No, then not there
        ld      a,d             ;Load first value
        cp      b               ;Is it the same?
        jr      nz,NO_DV        ;No, then not there
        inc     hl              ;Increment pointer
        cp      0F0h            ;Was pattern the last?
        jr      nz,UP_LP        ;No, then loop
        xor     a               ;Clear port A & B
        out     (PORT_A),a
        out     (PORT_B),a
UP_RET: pop     hl              ;Restore regs
        pop     de
        pop     bc
        ret
;
NO_DV:  scf                     ;Set carry flag
        jr      UP_RET
```

```
;
; TAPE SUBROUTINES
;-----------------------------------------------------------------
;--- ENABLE RS232 PORT AND TAPE DRIVE POWER POWER
;                          - SERIAL PORT 3 COMMUNICATES WITH
;                            THE TAPE DRIVE
;                          - BIT 6 OF PARALLEL PORT C IS USED
;                            TO CONTROL POWER TO THE TAPE DRIVE
;                            (SET TO +5 VOLTS TO ENABLE POWER)
;                          - THE 5 SECOND DELAY ALLOWS TIME FOR
;                            THE TAPE DRIVE TO STABILIZE BEFORE
;                            TRYING TO WRITE DATA
;
TP_WRT: PROC
        ld      a,0D0h          ;Enable Port #3
        out     (SP3+DCR),a
        ld      a,05h           ;Set parameters for Port #3
        out     (SP3+DCR),a
        ld      a,TAPON         ;Enable tape power(bit 6 of
        out     (PICNTL),a      ; parallel port C)
        call    S5_DLY          ;Delay 5 seconds
;
;                          - TEST DRIVE STATUS BY READING A BLOCK
;
        call    TP_CLR          ;Clear any tape response
        ld      b,05h           ;Load count for tape read tries
        ld      a,TWORD         ;Read a block
        call    S3OUT
        ld      a,TRDBLK
 ..1:   call    TP_COM
        in      a,(SP3+SR)      ;Read status bit
        and     01h             ;Strip for ready bit
        jr      nz, ..5         ;If ready, then skip next
        ld      a,TSTAT         ;Else, try again for a response
        djnz    ..1
;
;                          - IF NO RESPONSE, RESET THE DRIVE
;
 ..2:   ld      a,TRESET        ;Else reset the drive
        call    S3OUT
        call    S3IN            ;Wait for response
;
;                          - SEARCH FOR END OF DATA AFTER RESET
;
 ..3:   call    TP_ST           ;Get status
        ld      a,b             ;Load 2nd word
        cp      00h             ;Is it valid?
        jr      z, ..2          ;No, reset again
        and     03h             ;Strip for track bits
        ld      b,a             ;Save in B
        ld      a,(TRACK)       ;Load track storage
        cp      b               ;Compare tracks
        jr      z, ..4          ;Yes then search for end
        ld      a,TTRACK        ;Else, increment track
        call    TP_COM
```

```
        call    S5_DLY          ;Delay for rewind
        jr      ..3             ;Loop til correct track
..4:    ld      a,TWORD         ;Search for end of data
        call    S3OUT
        ld      a,TEOD
        call    S3OUT
        call    S3IN            ;Wait for response
        jr      ..8             ;Write data
;
;                       - IF FILE MARK, THEN OK TO WRITE
;
..5:    call    TP_ST           ;Get status
        cp      00h             ;Is it valid?
        jp      z,TP_WRT        ;No, try again
        bit     3,a             ;Is there a file mark?
        jr      nz, ..8         ;Yes then write data
        ld      de,(SAMNO)      ;Load sample number
        ld      a,d             ;Is it zero?
        or      e
        jr      z, ..7          ;Yes then skip next
        ld      a,b             ;Load 2nd word
        bit     3,a             ;Is tape ready?
        jr      z, ..2          ;No, then reset
        and     03h             ;Strip to track bits
        ld      c,a             ;Save in C
        ld      a,(TRACK)       ;Load track storage
        ld      b,a             ;Save in B
        ld      a,c             ;Get first back
        cp      b               ;Compare with tape
        jr      z, ..7          ;Yes then backup to write
        jr      c, ..6          ;If smaller, increment track
        inc     a               ;Increment track
        and     03h             ;Strip to track bits
        cp      00h             ;Is it zero?
        jp      z,TP_UNL        ;Unload if out of tape
        ld      (TRACK),a       ;Save new value
..6:    ld      a,TTRACK        ;Increment track
        call    TP_COM
        jp      TP_WRT          ;Try again
;
..7:    ld      a,TWORD         ;Backup to beginning of track
        call    S3OUT
        ld      a,TSKPBBLK
        call    TP_COM
```

```
;
;                       - WRITE THE DATA
;
..8:    call    TP_CLR          ;Flush input buffer
        ld      a,TWRITE        ;Start writing to tape
        call    TP_COM
        ld      hl,HEADST       ;Point to first of header
        ld      de,(BANK1)      ;Load first bank end address
        inc     de              ;Increment count
        ld      a,018h          ;Select memory bank #4
        ld      (BANKNO),a      ;Save value
        out0    (BBR),a         ;Load new bank
        ld      a,(LSTBNK)      ;Load last bank
        cp      018h            ;Is it only 1 bank?
        jr      nz,..10                 ;No, then skip next
        ld      de,(BANKL)      ;Else load last address
        inc     de              ;Increment count
..9:    ld      a,(hl)          ;Load data byte
        push    af              ;Save data
        and     0Fh             ;Strip upper bits
        or      30h             ;Convert to ASCII
        call    S3OUT           ;Send it out
        pop     af              ;Restore value
        and     0F0h            ;Strip lower bits
        rra                     ;Shift upper bits
        rra
        rra
        rra
        or      30h             ;Convert to ASCII
        call    S3OUT           ;Send it
        dec     hl              ;Decrement buffer pointer
        ld      a,h             ;Get high byte
        cp      d               ;Is it at the end?
        jr      nz, ..9         ;No, then do next
        ld      a,l             ;Load low byte of address
        cp      e               ;Is it at the end?
        jr      nz, ..9         ;No, then do next
        ld      de,(BANK2)      ;Load new end address
        ld      a,(LSTBNK)      ;Load last bank
        ld      b,a             ;Move to reg. B
        ld      a,(BANKNO)      ;Load current bank #
        cp      b               ;Is it the last?
        jr      z, ..11         ;Yes, then exit
        sub     08h             ;Decrement bank #
        ld      (BANKNO),a      ;Save value
        out0    (BBR),a         ;Load new bank
        ld      b,a             ;Move to reg. B
        ld      a,(LSTBNK)      ;Load current bank #
        cp      b               ;Is it the last?
        jr      nz, ..10                ;Yes, then skip next
        ld      de,(BANKL)      ;Load last address
..10:   inc     de              ;Increment count
        ld      hl,0E000h       ;Load new starting address
        jr      ..9             ;Loop til done
;
```

```
;                       - STOP WRITING, WRITE FILE MARKS, AND
;                         BACKUP TO FIRST FILE MARK
;
..11:   ld      a,TSTP          ;Stop writing
        call    TP_COM
        call    TP_ST           ;Check status for rewind
        cp      00h             ;Was there no status?
        jp      z,TP_TRK        ;Yes, then increment track
        ld      a,TMARK         ;Write a tape mark
        call    TP_COM
        ld      a,TMARK         ;Write a 2nd tape mark
        call    TP_COM
        ld      a,TWORD         ;Search for end of data
        call    S3OUT
        ld      a,TSKPBBLK      ;Backup to 1st file mark
        call    TP_COM
        ld      a,TWORD         ;Search for end of data
        call    S3OUT
        ld      a,TSKPBBLK
        call    TP_COM

STAPE.  ld      a,TAPOFF        ;Disable tape power
        out     (PICNTL),a
        ld      a,0C0h          ;Put port on standby
        out     (SP3+DCR),a
        ENDPROC
        ret
;
;--- CHECK TAPE WRITE STATUS ---
;
;
TP_CK.  in      a,(SP3+SR)      ;Read status bit
        and     01h             ;Strip for ready bit
        ret     z               ;Exit if none
        in      a,(SP3+RHR)     ;Load data
        ld      a,014h          ;Suspend writing
        call    TP_COM
        call    TP_ST           ;Get status
        ld      a,TWRITE        ;Enable write
        call    TP_COM
        call    TP_CLR          ;Clear input buffer
        ret
```

```
;
;--- INCREMENT TRACK ---
;                           - INCRSMENTS TRACK NUMBER STORED
;                             IN MEMORY
;                           - IF THE NEXT RACK IS ZERO, THEN THE
;                             TAPE IS UNLOADED
;                           - REGISTER A IS ALTERED
;
TP_TRK: PROC
        ld      a,(TRACK)       ;Load track value
        inc     a               ;Increment it
        ld      (TRACK),a       ;Save it
        ld      a,TSTP          ;Stop writing
        call    TP_COM
        ld      a,(TRACK)       ;Reload track value
        and     03h             ;Strip upper bits
        cp      00h             ;Is it zero?
        jr      nz, ..1         ;No, then exit
TP_UNL: ld      a,TWORD         ;Else Unload the tape
        call    S3OUT
        ld      a,TUNLOAD
        call    S3OUT
  ..1:  ei                      ;Enable interrupts
        ENDPROC
        halt                    ;Leave power on for rewind
```

```
;
;--- CHECK TAPE STATUS ---
;                           - SENDS A STATUS COMMAND
;                           - TRIES 5 TIMES TO READ STATUS
;                           - IF NO STATUS IS READ, RETURNS WITH
;                             ZERO IN REGISTERS A AND B
;                           - STATUS IS CHECK FOR REWIND AND JUMPS
;                             TO THE TRACK INCREMENT ROUTINE IF IT IS
;                           - RETURNS WITH THE FIRST STATUS WORD IN
;                             REGISTER A AND THE SECOND IN REGISTER B
;                           - NO OTHER REGISTERS ARE ALTERED
;
TP_ST:  PROC
        ld      b,05h           ;Load count for tries
 ..1:   call    TP_CLR          ;Flush input buffer
        ld      a,TSTAT         ;Send tape command for status
        call    TP_COM
        in      a,(SP3+RHR)     ;Load status
        cp      40h             ;Is it valid?
        jr      nc, ..2         ;Yes, then save values
        dec     b               ;Decrement count
        xor     a               ;Clear A
        cp      b               ;Is count zero?
        jr      nz, ..1         ;Else try 5 times
        xor     a               ;Clear status values
        ld      b,a             ;For error
        ret                     ;Exit
 ..2:   push    af              ;Save first word
        bit     0,a             ;Test busy bit
        jp      nz,TP_TRK       ;Busy means end of track
        in      a,(SP3+RHR)     ;Load 2nd word
        ld      b,a             ;Save in B
        pop     af              ;Restore first word
        ENDPROC
        ret

;
;--- CLEAR ANY PREVIOUS CHARACTERS FROM THE TAPE DRIVE ---
;                           - REGISTER A IS ALTERED
;
TP_CLR: in      a,(SP3+RHR)     ;Clear any tape response
        in      a,(SP3+SR)      ;Read status bit
        and     01h             ;Strip for ready bit
        jr      nz,TP_CLR       ;Loop till empty
        ret
;
;--- SEND A COMMAND TO THE TAPE DRIVE ---
;                           - COMMAND MUST BE IN REGISTER A
;                           - DELAYS 5 SECONDS FOR COMPLETION
;                           - NO REGISTERS ARE ALTERED
;
TP_COM: call    S3OUT           ;Send command
        call    S5_DLY          ;Delay for command
        ret
```

```
;
;--- SEND A CHARACTER OUT SERIAL PORT 3 ---
;                          - SENDS DATA TO TAPE DRIVE
;                          - CHARACTER TO SEND MUST BE IN
;                            REGISTER A
;                          - NO REGISTERS ARE ALTERED
;
S3OUT:  PROC
        push    af                  ;Save the data
 ..1:   in      a,(SP3+SR)          ;Read status bit
        and     04h                 ;Strip for ready bit
        jr      z, ..1              ;loop til ready
        pop     af                  ;Restore data
        out     (SP3+THR),a         ;Send it
        ENDPROC
        ret
;
;--- RECEIVE A CHARACTER FROM SERIAL PORT 3
;                          - RECEIVES DATA FROM TAPE DRIVE
;                          - LOOPS UNTIL A CHARACTER IS RECEIVED
;                          - INTERRUPTS ARE ENABLED DURING READ
;                          - CHARACTER IS RETURNED IN REGISTER A
;                          - NO OTHER REGISTERS ARE ALTERED
;
S3IN:   PROC
        ei                          ;Enable interrupts
 ..1:   in      a,(SP3+SR)          ;Read port status
        and     01h                 ;Strip for ready bit
        jr      z, ..1              ;Loop til ready
        in      a,(SP3+RHR)         ;Load data
        di                          ;Disable interrupts
        ENDPROC
        ret
```

```
;
; CLOCK SUBROUTINES
;-----------------------------------------------------------------------
;--- SET THE CURRENT TIME ---
;                           - ENTERED TIME IS LOADED INTO THE CLOCK
;                           - PROGRAM WILL WAIT UNTIL THE USER
;                             ENTERS A CARRIAGE RETURN TO START THE
;                             CLOCK
;                           - ALL REGISTERS ARE ALTERED
;
SETCLK: PROC
        ld      hl,CLKSETM      ;Point to message
        call    PSTRG           ;Display it
        ld      hl,TIMEM        ;Point to memory storage
        call    TIMINP          ;Get the time
;
;                           - LOAD THE CLOCK
;
        ld      a,04h           ;Set command for 24 hr. mode
        out     (RTCCMD),a
        ld      bc,0800h        ;B = Count=08h
        ld      c,RTCBASE       ;C = port
        ld      hl,MTMEM        ;Point to memory storage
        otimr                   ;Load the clock
;
;                           - SYNCH THE CLOCK
;
        ld      hl,CLKPRM       ;Point to message
        call    PSTRG           ;Display it
        call    GCLOCK          ;Display time of projected start
        ld      hl,MTMEM
        call    TIMOUT
..1:    call    SIIN            ;Test for a RETURN from terminal
        cp      CR
        jr      nz, ..1         ;Loop until a RETURN
        xor     a               ;Clear all clock interrupts
        out     (RTCSTAT),a
        ld      a,0Ch           ;Start the clock
        out     (RTCCMD),a
;
;                           - CLOCK IS STARTED
;
        ld      hl,CLKSYNCM     ;Point to message
        call    PSTRG           ;Display it
        ENDPROC
        ret
```

```
;
;--- READ THE CLOCK INTO MEMORY ---
;                           - THE TIME IS STORED STARTING AT MTMEM
;                           - NO REGISTERS ARE ALTERED
;
GCLOCK: push    af              ;Save the registers
        push    bc
        push    de
        push    hl
GCLCK1: in      a,(SEC.1)       ;Read .1 secs to freeze clock
        ld      hl,MTMEM        ;Point to memory storage
        ld      c,RTCBASE-1     ;Point to clock port
        ld      b,08h           ;Load count
GCLCK2: inc     c               ;Increment port address
        ini                     ;Get clock data byte
        jr      nz,GCLCK2       ;Loop for count (8 bytes)
        pop     hl              ;Restore registers
        pop     de
        pop     bc
        pop     af
        ret
;
;--- ENTER THE TIME ---
;
;                           - INPUTS YEAR,MONTH,DAY,HOUR,MINUTE
;                             FROM TERMINAL.
;                           - AT LEAST ONE DIGIT (AND NO MORE
;                             THAN TWO) MUST BE ENTERED FOR EACH.
;                           - EACH UNIT OF TIME MUST HAVE A
;                             SEPARATOR (ANY NON-DIGIT).
;                           - ENTERED TIME IS STORED IN 5 BYTES,
;                             STARTING WITH BYTE POINTED TO BY HL.
;                           - WILL NOT RETURN UNTIL A VALID ENTRY
;                                   IS MADE.
;
TIMINP: PROC
        ld      (TSTORE),hl     ;Save pointer
  ..1:  ld      hl,YMDHM        ;Point to message
        call    PSTRG           ;Display it
        ld      a,03Fh          ;Load a "?"
        call    S1OUT           ;Display it
        call    GETLN           ;Get answer
        ld      hl,(TSTORE)     ;Restore pointer
        ld      de,I8UFF        ;Point to ASC buffer
        ld      a,(NCI)         ;Load number of entries
        cp      11              ;Is there at least 11?
        jr      c, ..1          ;No, then error
        ld      b,05h           ;Load count
```

```
;                       - CHECKING IS DONE TO ENSURE VALID
;                         ENTRY FOR VALUE.
;
..2:    call    ASCHEX          ;Convert entry to HEX
        jr      c, ..1          ;If carry is set, then error
        inc     hl              ;Increment pointer
        inc     de              ;Pass the separator
        djnz    ..2             ;Loop for the 5 bytes
        ld      hl,(TSTORE)     ;Restore pointer
        inc     hl
        ld      a,0Ch           ;Load max value for month
        cp      (hl)            ;Is it < 13?
        jr      c, ..1          ;No, then error
        xor     a               ;Load min value for month
        cp      (hl)            ;Is it 0?
        jr      z, ..1          ;Yes, then error
        inc     hl              ;Increment pointer
        ld      a,01Fh          ;Load max value for day
        cp      (hl)            ;Is it < 32?
        jr      c, ..1          ;No, then error
        xor     a               ;Load min value for month
        cp      (hl)            ;Is it 0?
        jr      z, ..1          ;Yes, then error
        inc     hl              ;Increment pointer
        ld      a,017h          ;Load max value for hour
        cp      (hl)            ;Is it < 24?
        jr      c, ..1          ;No, then error
        inc     hl              ;Increment pointer
        ld      a,03Bh          ;Load max value for minutes
        cp      (hl)            ;Is it < 60?
        jr      c, ..1          ;No, then error
;
;                       - LOAD TIME ENTRY INTO MEMORY IMAGE
;                         OF CLOCK
;
        ld      hl,(TSTORE)     ;Point to memory
        xor     a               ;Load 0 for seconds
        ld      (MSEC.1),a
        ld      (MSEC),a
        ld      a,(hl)          ;Load year
        ld      (MYEAR),a
        inc     hl              ;Increment pointer
        ld      a,(hl)          ;Load month
        ld      (MMTH),a
        inc     hl              ;Increment pointer
        ld      a,(hl)          ;Load day
        ld      (MDAY),a
        inc     hl              ;Increment pointer
        ld      a,(hl)          ;Load hour
        ld      (MHR),a
        inc     hl              ;Increment pointer
        ld      a,(hl)          ;Load minutes
        ld      (MMIN),a
        ENDPROC
        ret
```

```
;
;--- DISPLAY THE TIME ---
;                           - DISPLAYS THE TIME TO THE TERMINAL
;                           - FORMAT = HH:mm MM/DD/YY
;                           - REGISTER HL MUST POINT TO THE
;                             START OF TIME STORAGE
;                           - REGISTERS A AND HL ARE ALTERED
;
TIMOUT: inc     hl              ;Bypass .1 secs
        call    PDEC            ;Display hour
        ld      a,':'           ;Display a colon
        call    S1OUT
        inc     hl              ;Increment pointer
        call    PDEC            ;Display minute
        ld      a,SPACE         ;Display a space
        call    S1OUT
        inc     hl              ;Bypass seconds
        inc     hl
        call    PDEC            ;Display month
        ld      a,'/'           ;Display separator
        call    S1OUT
        inc     hl              ;Increment pointer
        call    PDEC            ;Display day
        ld      a,'/'           ;Display separator
        call    S1OUT
        inc     hl              ;Increment pointer
        call    PDEC            ;Display year
        call    PCRLF           ;Display line feed
        ret
```

```
;
; GENERAL SUBROUTINES
;-------------------------------------------------------------------
;--- SWITCH MEMORY BANKS ---
;
BNK_SEL:
        push    af              ;Save value
        ld      (BANKNO),a      ;Save value
        ld      bc,08h          ;Load bank value
        ld      b,a             ;Move bank # to reg. B
        mlt     bc              ;Multiply them
        ld      a,c             ;Move result to reg. A
        out0    (BBR),a         ;Load new bank
        pop     af              ;Restore value
        ret
;
;--- TURN ON POWER TO THE SENSORS ---
;                       - SETS 4 LINES OF THE PARALLEL PORT
;                         TO A HIGH STATE(+5 VOLTS)
;                       - PROVIDES A MEANS FOR THE COMPUTER
;                         TO ENABLE POWER TO THE SENSORS AT
;                         THE BEGINNING OF EACH DATA SAMPLE
;                       - REGISTER A IS ALTERED
;
SENON:  ld      a,01h           ;Switch on bit 0 of C port
        out     (PICNTL),a
        ld      a,03h           ;Switch on bit 1 of C port
        out     (PICNTL),a
        ld      a,05h           ;Switch on bit 2 of C port
        out     (PICNTL),a
        ld      a,09h           ;Switch on bit 4 of C port
        out     (PICNTL),a
        ret
;
;--- TURN OFF POWER TO THE SENSORS ---
;                       - SETS 4 LINES OF THE PARALLEL PORT
;                         TO A LOW STATE(0 VOLTS)
;                       - PROVIDES A MEANS FOR THE COMPUTER
;                         TO DISABLE POWER TO THE SENSORS AT
;                         THE END OF EACH DATA SAMPLE
;                       - REGISTER A IS ALTERED
;
SENOFF: ld      a,00h           ;Switch off bit 0 of C port
        out     (PICNTL),a
        ld      a,02h           ;Switch off bit 1 of C port
        out     (PICNTL),a
        ld      a,04h           ;Switch off bit 2 of C port
        out     (PICNTL),a
        ld      a,08h           ;Switch off bit 4 of C port
        out     (PICNTL),a
        ret
```

```
;
;--- LOAD THE SAMPLE RATE CLOCK ---
;                           - SCAN RATE IS SET USING 3 TIMERS
;                           - TIMER 0 IS SET TO PROVIDE A 1KHz
;                             CLOCK TO TIMER 1
;                           - TIMER 1 IS SET TO PROVIDE THE
;                             TIME BASE FOR EITHER SCANS/SECOND
;                             (SFLAG = FALSE) OR SCANS/MINUTE
;                             (SFLAG = TRUE)
;                           - REGISTER A IS ALTERED
;
SR_LD:  PROC
        ld      a,034h          ;Enable counter 0
        out     (CTRC),a
        ld      a,066h          ;Load divisor for 1K Hz
        out     (CTR0),a
        ld      a,02h
        out     (CTR0),a
        ld      a,074h          ;Enable counter 1
        out     (CTRC),a
        ld      a,(SFLAG)       ;Load scan rate flag
        cp      FALSE           ;Is it set for scans/sec?
        jr      z, ...          ;No, then load for samples/min
        ld      a,0F4h          ;Load divisor for scan base
        out     (CTR1),a
        ld      a,01h
        out     (CTR1),a
        ret
;
...:    ld      a,0Ah           ;Load divisor for scans/sec
        out     (CTR1),a
        ld      a,00h
        out     (CTR1),a
        ENDPROC
        ret
```

```
;
;--- START SCAN RATE CLOCK ---
;                          - LOADS THE SAMPLE RATE DIVISOR INTO
;                            TIMER 2
;                          - STARTS ALL COUNTERS
;                          - END OF SCAN TIME IS DETERMINED BY
;                            TESTING THE DONE BIT ON TIMER 2
;                          - NO REGISTERS ARE ALTERED
;
SR_STRT:PROC
        push    af                ;Save registers
        push    bc
        push    de
        ld      a,0B0h            ;Enable counter 2
        out     (CTRC),a
        ld      a,(SFLAG)         ;Load scan rate flag
        cp      FALSE             ;Is it set for scans/sec?
        jr      z, ..1            ;No, then do scans/min
        ld      de,04B0h          ;Load dividend for scans/min
        ld      a,(SMPLRT)        ;Load scan rate entry
        call    DIVIDE            ;Divide to get counter setting
        jr       ..2              ;Load counter
 ..1:   ld      de,03E8h          ;Load dividend for scans/sec
        ld      a,(SMPLRT)        ;Load scan rate entry
        call    DIVIDE            ;Divide to get counter setting
 ..2:   ld      a,e               ;Load low-byte
        out     (CTR2),a
        ld      a,d               ;Load high-byte
        out     (CTR2),a
        ld      a,0FFH            ;Enable counting
        out     (CTRGC),a
        pop     de                ;Restore registers
        pop     bc
        pop     af
        ENDPROC
        ret
;
;--- DELAY FOR 100 MILLISECONDS ---
;                          - PROVIDES A DELAY TO ALLOW THE A-D
;                            CIRCUITS TIME TO SETTLE BEFORE TAKING
;                            DATA
;                          - NO REGISTERS ARE ALTERED
;
DELAY:  push    bc                ;Save contents of register
        ld      b,07Fh            ;Load delay count
DLYA:   NOP                       ;Do nothing
        djnz    DLYA              ;For 128 times
        pop     bc                ;Restore register
        ret
```

```
;
;--- DELAY FOR 5 SECONDS ---
;                           - PROVIDES A 5 SECOND DELAY TO ALLOW
;                             THE TIME TO EXECUTE A COMMAND TO
;                             THE TAPE DRIVE
;                           - REGISTERS A IS ALTERED
;
S5_DLY: PROC
        push    bc
        ld      c,03Ch          ;Load 1st count for delay
 ..1:   ld      b,05h           ;Load 2nd count for delay
        ld      ix, ..2         ;Load return point
        jp      T0_DLY          ;Delay for 1 sec.
;
 ..2:   dec     c               ;Decrement 1st count
        jr      nz, ..1         ;Loop for 5 seconds
        pop     bc
        ENDPROC
        ret

;--- DELAY 20 SECONDS ---
;                           - PROVIDES A 20 SECOND DELAY TO ALLOW
;                             THE COMPUTER RECEIVING THE RF DATA,
;                             TIME TO WRITE THE DATA TO HARD DISK
;                           - REGISTERS A AND BC ARE ALTERED
;
S20_DLY:PROC
        ld      c,03Ch          ;Load 1st count
 ..1:   ld      b,22h           ;Load 2nd count
        ld      ix, ..2         ;Load pointer to next routine
        jp      T0_DLY          ;Delay
;
 ..2:   dec     c               ;Decrement count
        jr      nz,  .1         ;Loop til done
        ENDPROC
        ret
;
;--- DELAY 1 MINUTE ---
;                           - PROVIDES 1 MINUTE DELAY FOR SENSOR
;                             POWER
;                           - REGISTERS A AND BC ARE ALTERED
;
M1_DLY: PROC
        ld      c,078h          ;Load 1st count
 ..1:   ld      b,030h          ;Load 2nd count
        ld      ix, ..2         ;Load jump location
        jp      T0_DLY          ;Delay 1 second
;
 ..2:   dec     c               ;Decrement count
        jr      nz, ..1         ;Loop til done
        ENDPROC
        ret
```

```
;
;--- TIMING ROUTINE USING TIMER 0 ---
;                          - DELAYS 1 SECOND PER PASS
;                          - REGISTER B MUST CONTAIN THE COUNT
;                            FOR NUMBER OF PASSES
;                          - REGISTERS A AND B ARE ALTERED
;
T0_DLY: PROC
        in0     a,(TCR)          ;Load timer status
        and     0FEh             ;Mask to enable timer 0
        out0    (TCR),a          ;Enable timer 0
        ld      a,0Ch            ;Load low-byte of count
        out0    (TMDR0L),a
        out0    (TMDR0H),a       ;Load high-byte of count
        in0     a,(TCR)          ;Load timer status
        or      01h              ;Start timer 0
        out0    (TCR),a
..1:    in0     a,(TCR)          ;Load timer status
        bit     6,a              ;Test if done
        jr      z, ..1           ;No, then loop til done
        in0     a,(TMDR0L)       ;Read counter
        in0     a,(TMDR0H)
        djnz    T0_DLY           ;Loop through count
        ENDPROC
        jp      (ix)             ;Return to location in IX
;
;--- TIMING ROUTINE USING TIMER 1 ---
;                          - DELAYS .1 SECONDS EACH PASS
;                          - REGISTER B MUST CONTAIN THE COUNT
;                            FOR NUMBER OF PASSES
;                          - REGISTERS A AND B ARE ALTERED
;
T1_DLY: PROC
        in0     a,(TCR)          ;Get timer status
        and     0FEh             ;Mask to enable timer 1
        out0    (TCR),a          ;Enable timer 1
        ld      a,07Ch           ;Load low-byte of count
        out0    (TMDR1L),a
        ld      a,092h           ;Load high-byte of count
        out0    (TMDR1H),a
        in0     a,(TCR)          ;Get timer status
        or      02h              ;Start timer 1
        out0    (TCR),a
..1:    in0     a,(TCR)          ;Get timer status
        bit     7,a              ;Test if done
        jr      z, ..1           ;No, then loop til done
        in0     a,(TMDR1L)       ;Read counter
        in0     a,(TMDR1H)
        djnz    T1_DLY           ;Loop through count
        ENDPROC
        ret
```

```
;
;--- CLEAR THE DATA BUFFER ---
;                          - CLEARS THE USER AREA OF EACH
;                            BANK OF MEMORY
;                          - REGISTERS A AND HL ARE ALTERED
;
CLRMEM: PROC
        ld      a,020h          ;Load memory bank #5
..1:    sub     08h             ;Decrement bank #
        ld      (BANKNO),a      ;Save it
        out0    (BBR),a         ;Select it
        ld      hl,08000h       ;Point to beginning of bank
..2:    xor     a               ;Load 00h into memory
        ld      (hl),a
        inc     hl              ;Increment memory pointer
        ld      a,l             ;Get low-byte of address
        cp      0FFH            ;Test for end
        jr      nz, ..2         ;No, then continue filling
        ld      a,h             ;Get high-byte of address
        cp      0FFh            ;Test for end
        jr      nz, ..2         ;No, then continue filling
        ld      a,(BANKNO)      ;Load current bank #
        cp      00h             ;Is it 0?
        jr      nz, ..1         ;No, then do next bank
        ENDPROC
        ret
```

```
;
; MATH AND CONVERSION ROUTINES
;-----------------------------------------------------------------------
;
;--- CONVERT AN ASCII ENTRY TO 16-BIT BINARY CODED DECIMAL ---
;                          - WORKS FOR NUMBERS UP TO 9999
;                          - CONVERTS ENTRY IN (IBUFF)
;                            TO BCD
;                          - BCD NUMBER IS PUT IN (BCDBUF)
;                          - (BCI) CONTAINS THE NUMBER OF BYTES
;                          - CARRY IS SET IF ERROR
;                          - ALL REGISTERS ARE ALTERED
;
ASCBCD: PROC
        xor     a                       ;Clear BCD number count
        ld      (BCI),a
        ld      de,BCDBUF               ;Point to BCD buffer
        ld      hl,IBUFF                ;Point to entry buffer
        ld      bc,00h                  ;Clear BC
        ld      a,(NCI)                 ;Get number of characters
        ld      b,a                     ;Save count
        cp      06h                     ;Is it less than 5?
        jr      nc, ..3                 ;No, then error
..1:    cp      00h                     ;Is count 0?
        ret     z                       ;Yes then return
        bit     0,a                     ;Is it odd, or even?
        jr      z, ..2                  ;Even, then do 10's unit
        ld      a,(hl)                  ;Get character
        and     0Fh                     ;Strip ASCII
        cp      0Ah                     ;Is it a number?
        jr      nc, ..3                 ;Return on error
        add     a,c                     ;Add the two
        daa                             ;Adjust for BCD
        ld      (de),a                  ;Save it
        ld      a,(BCI)                 ;Increment BCD number count
        inc     a
        ld      (BCI),a                 ;Resave it
        inc     de                      ;Increment BCD buffer
        inc     hl                      ;Increment entry buffer
        dec     b                       ;Decrement count
        ld      a,b                     ;Move it to A
        jr        ..1                   ;Get next
;
..2:    ld      a,(hl)                  ;Get first entry
        and     0Fh                     ;Strip ASCII
        cp      0Ah                     ;Is it a number?
        jr      nc, ..3                 ;Return on error
        ccf                             ;Clear carry flag
        rla                             ;Shift to upper bits
        rla
        rla
        rla
        ld      c,a                     ;Save in C
        inc     hl                      ;Increment entry buffer
        dec     b                       ;Decrement count
```

```
        ld      a,b             ;Move it to A
        jp      ..1             ;Do next
..3:    scf                     ;Set carry flag
        ENDPROC
        ret
;
;--- CONVERT A 16-BIT BINARY CODED DECIMAL NUMBER TO HEXADECIMAL ---
;                       - BCI MUST CONTAIN THE NUMBER OF
;                         BINARY CODED DECIMAL (BCD) DIGITS
;                       - THE BCD NUMBER MUST BE IN BCDBUFF
;                       - RESULTING HEXADECIMAL VALUE IS IN HL
;                       - ONLY REGISTER BC IS UNALTERED
BCDHEX: PROC
        ld      a,(BCI)         ;Get BCD count
        ld      de,00h          ;Clear DE
        ld      hl,BCDBUF       ;Point to BCD buffer
        cp      01h             ;Is there only one byte?
        jr      z,..1           ;Yes, then skip next
        ld      de,03E8h        ;Load multiplier(1000)
        ld      a,(hl)          ;Load MSB (upper byte)
        and     0F0h            ;Strip low bits
        rra                     ;Shift to low bits
        rra
        rra
        rra
        call    MULTPLY         ;X1000
        push    de              ;Save answer
        ld      a,(hl)          ;Get byte again
        and     0Fh             ;Strip upper bits
        ld      de,064h         ;X100
        ld      d,a             ;Move to D
        mlt     de              ;
        pop     hl              ;Get answer back
        add     hl,de           ;Add them
        ex      de,hl           ;Move answer to DE
        ld      hl,BCDBUF+1     ;Point to next byte
..1:    ld      a,(hl)          ;Load LSB (lower byte)
        push    af              ;Save BCD value
        and     0F0h            ;Strip low bits
        rra                     ;Shift to low bits
        rra
        rra
        rra
        ex      de,hl           ;Save result in HL
        ld      de,0Ah          ;X10
        ld      d,a
        mlt     de
        add     hl,de           ;Add result to HL
        pop     af              ;Restore BCD number
        and     0Fh             ;Strip upper bits
        ld      de,00h          ;Clear DE
        ld      e,a             ;Move lower bits to E
        add     hl,de           ;Add to total
        ENDPROC
        ret
```

```
;
;--- CONVERT AN ASCII ENTRY TO HEXADECIMAL ---
;
;                       - CLEARS LOCATION POINTED TO BY HL
;                         THEN ENTERS THE HEX DIGIT
;                       - ONE OR TWO DIGITS ARE ACCEPTABLE,
;                         FOLLOWED BY A TERMINATOR (ANY NON-DIGIT)
;                       - CARRY IS SET IF INPUT IS INVALID
;                       - DE MUST POINT TO ASCII CHARACTER STRING
;                       - RESULT IS IN THE MEMORY LOCATION
;                         POINTED TO BY REGISTER HL
;                       - THE BUFFER POINTER (REGISTER DE) IS
;                         INCREMENTED TO THE NEXT CHARACTER OF THE
;                         OF THE STRING
;                       - REGISTER A IS ALTERED
;
ASCHEX: push    bc              ;Save register
        ld      b,02h           ;Load count
        ld      (hl),00h        ;Clear HL
        ld      a,(de)          ;Get first entry
        call    CONVAS          ;Convert it to BCD
        jr      c,NOK           ;If error, return with carry
        rld                     ;Shift number into memory
        inc     de              ;Point to next character
        ld      a,(de)          ;Get next entry
        call    CONVAS          ;Convert to BCD
        jr      c,IOK           ;OK if separator
        push    af              ;Save 2nd number
        ld      a,(hl)          ;Get 1st number back
        add     a,a             ;X2
        ld      b,a             ;Save it in B
        add     a,a             ;X4
        add     a,a             ;X8
        add     a,b             ;X10
        ld      b,a             ;Save result in B
        pop     af              ;Restore 2nd number
        add     a,b             ;Add to form HEX number
        ld      (hl),a          ;Save number in memory
        inc     de              ;Increment ASCII buffer
IOK:    xor     a               ;Clear all flags
NOK:    pop     bc              ;Restore register
        ret
;
;--- CONVERT ASCII TO BINARY-CODED-DECIMAL (BCD) DIGIT ---
;                       - ASCII VALUE MUST BE IN REGISTER A
;                       - RETURNS WITH CARRY SET IF ERROR
;                       -  NO OTHER REGISTERS ALTERED
;
CONVAS: sub     30h             ;Strip ASCII value
        ret     c               ;Carry is set if < 30 HEX
        cp      0Ah             ;Must be <= 9
        jr      nc,NOTVAL       ;Exit with carry set if not
        ccf                     ;Clear carry flag
        ret
;
```

```
NOTVAL: scf                     ;Set carry flag
        ret
;
;--- DIVIDE ROUTINE ---
;                           - DIVIDES A 16-BIT NUMBER BY AN
;                             8-BIT NUMBER
;                           - DIVISOR MUST BE IN REGISTER A
;                           - DIVIDEND MUST BE IN REGISTER DE
;                           - ANSWER IN DE, CARRY SET IF /0
;                           - REMAINDER IN B
;                           - ONLY REGISTER HL IS NOT ALTERED
;
DIVIDE: PROC
        or      a               ;Check for zero
        jr      nz, ..1         ;If not, divide
        scf                     ;Else, set carry flag
        ret
 ..1:   ld      c,a             ;Move divisor
        ld      b,16            ;Count = no. bits in divisor
        xor     a               ;Clear A
 ..2:   sla     e               ;Subtract divisor for 16 bits
        rl      d
        rl      a
        cp      c
        jr      c, ..3          ;If carry, then skip next
        inc     e
        sub     c
 ..3:   djnz    ..2             ;Loop for 16 bits
        ld      b,a             ;Save remainder in B
        ENDPROC
        ret
;
;--- MULTIPLY ROUTINE ---
;                           - MULTIPLIES A 16-BIT NUMBER BY
;                             AN 8-BIT NUMBER
;                           - MULTIPLICAND IN DE
;                           - MULTIPLIER IN A
;                           - PRODUCT IN DE
;                           - REGISTERS A AND B ARE ALTERED
;
MULTPLY:PROC
        push    hl              ;Save HL
        ld      b,08h           ;Load count
        ld      hl,00h          ;Clear HL
 ..1:   add     hl,hl           ;X2
        rla                     ;Shift multiplier left
        jr      nc, ..2         ;No carry, then skip next
        add     hl,de           ;Add product
 ..2:   djnz    ..1             ;Loop for 8 bits
        ex      de,hl           ;Move to de
        pop     hl
        ENDPROC
        ret
```

```
;
;--- COMPARE TWO 16-BIT NUMBERS ---
;                          - HL MUST CONTAIN ONE NUMBER
;                          - DE MUST CONTAIN THE OTHER
;                          - CARRY IS SET IF HL IS SMALLER
;                            THAN DE
;                          - REGISTER A IS ALTERED
;
BCDCP:   PROC
         ld        a,h             ;Load high byte of HL
         cp        d               ;Compare with D
         jr        z,BCDCP1        ;Return if smaller
         ret
BCDCP1:  ld        a,l             ;Load low byte of HL
         cp        e               ;Carry is set if larger
         ENDPROC
         ret
```

```
;
;
; INPUT ROUTINES
;-----------------------------------------------------------------
;
;--- GET A LINE OF INPUT FROM SERIAL PORT #1 ---
;                          - RECEIVES A STRING OF CHARACTERS
;                            UNTIL A CARRIAGE RETURN IS DETECTED
;                          - INSERTS AN END TERMINATOR (00 HEX)
;                            AFTER A RETURN HAS BEEN DETECTED
;                          - EACH CHARACTER IS CHECKED FOR VALIDITY
;                            AND IF A LETTER, FORCED TO UPPER CASE
;                          - CHECKS ARE MADE FOR THE FOLLOWING TEST
;                            FUNCTIONS:
;                                  CONTROL A = A-D TEST
;                                  CONTROL S = RF TRANSMIT TEST
;                                  CONTROL T = CLOCK TEST
;                                  CONTROL R = TAPE TEST
;
GETLN:  PROC
        ld      b,00h           ;Clear B for character count
        ld      hl,IBUFF        ;Point to storage
..1:    call    S1IN            ;Get character
        cp      0Dh             ;Is it RETURN?
        jr      z, ..2          ;Yes, then end
        cp      08h             ;Is it backspace?
        call    z,PDEL          ;Yes, then delete
        cp      07Fh            ;Is it delete?
        call    z,PDEL          ,Yes, then delete
        cp      01h             ;Is it ^A?
        jp      z,AD_TST        ;Yes, then do A-D test
        cp      013h            ;Is it ^S?
        jp      z,ASCDP         ;Yes, then Serial port #0 test
        cp      014h            ;Is it ^T?
        jp      z,CLKTST        ;Yes, then clock test
        cp      012h            ;Is it ^R?
        jp      z,TP_TST        ;Yes, then tape test
        cp      020h            ;Is it a space?
        jr      c, ..1          ;Yes, then skip
        cp      07Fh            ;Is it > delete?
        jr      nc, ..1         ;Yes, then skip
        call    S1OUT           ;Display it
        call    INPTST          ;Force to upper case
        ld      (hl),a          ;Save it
        inc     hl              ;Increment buffer pointer
        inc     b               ;Increment count
        ld      a,b             ;Load count in A
        cp      028h            ;Is it > 40?
        jr      c, ..1          ;No, then get next
..2:    call    PCRLF           ;Display a return
        ld      (hl),00h        ;Insert end terminator
        ld      a,b             ;Load count in A
        ld      (NCI),a         ;Save the count
        ENDPROC
        ret
```

```
;
;--- FORCE LOWER CASE TO UPPER CASE ---
;                           - TO SIMPLIFY DECODING LETTER CHARACTERS,
;                             ALL LETTERS ARE CONVERTED TO UPPER CASE
;                           - CHARACTER MUST BE IN REGISTER A ON ENTRY
;                           - NO OTHER REGISTERS ARE ALTERED
;
INPT3T: cp      61h             ;Is it greater than 'a'?
        ret     c               ;No, then exit
        cp      07Bh            ;Is it less than 'z'?
        ret     nc              ;No, then exit
        and     05Fh            ;Convert to upper case
        ret
;
;--- GET A CHARACTER FROM SERIAL PORT #1 ---
;                           - GETS A CHARACTER FROM TERMINAL PORT
;                           - LOOPS UNTIL DATA IS PRESENT
;                           - RETURNS WITH CHARACTER IN REGISTER A
;                           - REGISTER A IS ALTERED
;
S1IN:   in0     a,(STAT1)       ;Read status
        and     080h            ;Strip for data ready bit
        jr      z,S1IN          ;Loop til data is there
        in0     a,(TSR1)        ;Get data byte
        and     07Fh            ;Strip for ASCII
        ret


;
;--- GET A CHARACTER FROM SERIAL PORT #0 ---
;                           - GETS A CHARACTER FROM RF PORT
;                           - LOOPS UNTIL DATA IS PRESENT
;                           - RETURNS WITH CHARACTER IN REGISTER A
;                           - REGISTER A IS ALTERED
;
S0IN:   in0     a,(STAT0)       ;Get status
        and     080h            ;Strip for data ready bit
        jr      z,S0IN          ;Loop til data is there
        in0     a,(TSR0)        ;Get data byte
S0IN1:  and     07Fh            ;Strip for ASCII
        ret
```

```
;
;DISPLAY ROUTINES
;-----------------------------------------------------------------
;
;--- DISPLAY A 16-BIT HEXADECIMAL NUMBER AS A DECIMAL ---
;                        - DISPLAYS DECIMAL VALUE ON TERMINAL PORT
;                        - HL MUST CONTAIN THE HEX NUMBER
;                        - AF AND AF' MUST BE CLEAR ON ENTRY
;                        - (NCT) MUST CONTAIN 3 FOR 16-BIT
;                                             2 FOR 12-BIT
;                                             2 FOR 8 -BIT
;                                             1 FOR 4 -BIT
;                        - CALL DEC16 FOR A 16-BIT NUMBER
;                        - CALL DEC12 FOR A 12-BIT NUMBER
;                        - CALL DEC8 FOR AN 8-BIT NUMBER
;                        - CALL DEC4 FOR A 4-BIT NUMBER
;                        - ALL REGISTERS ARE ALTERED
;
DEC16:  PROC
        ld      de,02710h       ;Load hex value for 10000
        call    HEXSUB          ;Determine # of 10000's
        jr      c,DEC12         ;If less, go to 12-bit test
        ex      af,af'          ;Get number
        add     a,01h           ;Add 10000
        ex      af,af'          ;Save it
        jr      DEC16           ;Loop til negative
DEC12:  ex      af,af'          ;Get number
        ld      ((BUFF+2),a     ;Save 10000's value
        xor     a               ;Clear A
        ex      af,af'          ;Save it
  ..1:  ld      de,03E8h        ;Load hex value for 1000
        call    HEXSUB          ;subtract 1000
        jr      c,DEC8          ;If negative do next
        ex      af,af'          ;Get number
        add     a,010h          ;Add 1000
        daa
        ex      af,af'          ;Save it
        jr        ..1           ;Loop til negative
DEC8.   ld      de,064h         ;Load hex value for 100
        call    HEXSUB          ;subtract 100
        jr      c,DEC4
        ex      af,af'          ;Get number
        add     a,01h           ;Add 100
        daa                     ;Adjust for BCD
        ex      af,af'          ;Save it
        jr      DEC8            ;Loop til negative
DEC4:   ex      af,af'
        ld      (IBUFF+1),a     ;Save 1000 and 100 value
        xor     a               ;Clear A
        ex      af,af'          ;Save it
  ..2:  ld      de,0Ah          ;Load hex value for 10
        call    HEXSUB          ;Subtract 10
        jr      c,DEC1          ;If negative do next
        ex      af,af'          ;Get number
        add     a,010h          ;Add 10
```

```
        daa                         ;Adjust for BCD
        ex      af,af'              ;Save it
        jr       ..2                ;Loop til negative
DEC1:   ex      af,af'              ;Get number
        add     a,l                 ;Add remainder
        daa
        ld      (IBUFF),a           ;Save it
        ENDPROC
        ret
;
;                       - DISPLAY THE HEX VALUE IN (IBUFF)
;
DECP:   PROC
        ld      a,(NCI)             ;Load number of characters
        ld      b,a                 ;Save in B
        dec     a                   ;Decrement count
        ld      hl,IBUFF            ;Point to buffer
        ld      de,00h              ;Clear DE
        ld      e,a                 ;Move count to E
        add     hl,de               ;Add them
 ..3:   call    PBCD                ;Display it
        dec     hl                  ;Loop til done
        djnz     ..3
        ENDPROC
        ret
;
;                       - SUBTRACT THE UNITS
;
HEXSUB: call    BCDCP               ;Compare HL with DE
        ret     c                   ;If negative do next
        sbc     hl,de               ;subtract
        ret
```

```
;
;--- DISPLAY HEXADECIMAL VALUE ---
;                           - DISPLAYS HEXADECIMAL VALUE TO THE
;                             TERMINAL PORT
;                           - DISPLAYS 16-BIT HEX VALUE IN THE
;                             MEMORY LOCATION POINTED TO BY HL
;                           - REGISTERS A AND HL ARE ALTERED
;
HEXP:   ld      a,(hl)          ;Get upper byte
        call    HEXOUT          ;Display it
        dec     hl              ;Decrement pointer
        ld      a,(hl)          ;Get lower byte
;
HEXOUT: push    af              ;Save value
        rrca                    ;Put upper 4-bits into
        rrca                    ; lower 4-bits
        rrca
        rrca
        call    PCD             ;Display digit
        pop     af              ;Restore value
;
PCD:    and     0Fh             ;Strip upper bits
        add     a,090h          ;Add 90 BCD
        daa
        adc     a,040h          ;Add 40 = letter
        daa                     ; 30 = number
        call    SIOUT           ;Display it
        ret

;
;--- DISPLAY AN ENTRY AS ASCII ---
;                           - CONVERTS A HEXADECIMAL VALUE TO
;                             DECIMAL AND DISPLAYS IT ON THE
;                             TERMINAL PORT
;                           - VALUE MUST BE IN REGISTER A
;                           - REGISTERS A AND B ARE ALTERED
;
PENTRY: ld      (TSTORE),a      ;Save it for print
        ld      hl,TSTORE       ;Point to it
```

```
;
;--- DISPLAY A HEXADECIMAL VALUE AS A DECIMAL ---
;
;                       - WORKS WITH NUMBERS UP TO 63 HEX
;                         (99 DECIMAL).
;                       - HL MUST POINT TO STORAGE OF NUMBER.
;                       - RESULT WILL BE IN A.
;                       - CARRY IS SET IF TOO BIG.
;                       - VALUE POINTED TO BY HL IS ALTERED
;                       - REGISTERS A AND B ARE ALTERED
;
PDEC:   PROC
        xor     a               ;Clear A
        rld                     ;Get high bits
        add     a,a             ;X2
        daa
        add     a,a             ;X4
        daa
        add     a,a             ;X8
        daa
        add     a,a             ;X16
        daa
        ld      b,a             ;Save in B
        xor     a               ;Clear A
        rld                     ;Get low bits
        cp      0Ah             ;Is it a number?
        jr      c, ..1          ;Yes, then skip next
        add     a,06h           ;Convert letter to number
..1:    add     a,b             ;Add upper bits to lower
        daa
        ld      (hl),a          ;Save it in memory
        call    PBCD            ;Display it
        ENDPROC
        ret
;
;--- DISPLAY THE PACKED BINARY CODED DECIMAL VALUE ---
;
;                       - HL MUST POINT TO LOCATION OF VALUE
;                       - NO REGISTERS ARE ALTERED
;
PBCD:   push    af              ;Save register
        ld      a,30h           ;Load ASCII mask
        rld                     ;Rotate in upper bits
        call    S1OUT           ;Display it
        rld                     ;Rotate in lower bits
        call    S1OUT           ;Display it
        rld                     ;Restore original number
        pop     af              ;Restore register
        ret
```

```
;
;--- DISPLAY 'CR TO CONTINUE' ---
;                           - DISPLAYS A MESSAGE TO THE TERMINAL PORT
;                           - MESSAGE INSTRUCTS USER TO PRESS THE
;                             RETURN KEY TO CONTINUE PROGRAMMING
;                           - REGISTERS A AND HL ARE ALTERED
;
PCONT:  PROC
        ld      hl,CONTM        ;Display the message
        call    PSTRG
..1:    call    S1IN            ;Test for a RETURN from terminal
        cp      CR
        jr      nz, ..1         ;Loop until a RETURN
        ENDPROC
        ret
;
;--- DISPLAY A STRING ---
;                           - DISPLAYS A STRING ON TERMINAL PORT
;                           - HL MUST POINT TO STRING
;                           - STRING MUST BE NULL TERMINATED
;                           - CALLING PSTRG DISPLAYS A RETURN BEFORE
;                             THE STRING
;                           - CALLING PSTRG1 DISPLAYS THE STRING ONLY
;                           - REGISTERS A AND HL ARE ALTERED
;
PSTRG:  call    PCRLF           ;Display RETURN
PSTRG1: ld      a,(hl)          ;Get byte
        or      a               ;Is it 0?
        ret     z               ;Yes, then done
        call    S1OUT           ;Display it
        inc     hl              ;Increment pointer
        jr      PSTRG1          ;Get next byte
;
;--- DISPLAY A CARRIAGE RETURN AND LINE FEED ---
;                           - DISPLAYS A RETURN ON THE TERMINAL PORT
;                           - REGISTER A IS ALTERED
;
PCRLF:  ld      a,CR            ;Load carriage return
        call    S1OUT           ;Display it
        ld      a,LF            ;Load a line feed
        call    S1OUT           ;Display it
        ret
```

```
;
;--- BACKSPACE AND DELETE A CHARACTER ---
;                           - DELETES A CHARACTER WHEN THE
;                             BACKSPACE KEY IS ENTERED ON
;                             THE TERMINAL PORT
;                           - REGISTERS A AND B ARE ALTERED
;
PDEL:   ld      a,b             ;Load character count
        or      a               ;Is it 0?
        ret     z               ;Yes, then no backspace
        ld      a,08h           ;Display a backspace
        call    S1OUT
        call    PSPACE          ;Display a space
        ld      a,08h           ;Display another backspace
        call    S1OUT
        dec     hl              ;Decrement buffer pointer
        dec     b               ;Decrement character count
        ret
;
;--- DISPLAY A SPACE ---
;                           - DISPLAY A SPACE TO THE TERMINAL PORT
;                           - REGISTER A ALTERED
;
PSPACE: ld      a,SPACE         ;Display a space
        call    S1OUT
        ret

;
;--- SEND A CHARACTER OUT SERIAL PORT #1 ---
;                           - SENDS A CHARACTER OUT THE PORT
;                             USED FOR THE TERMINAL
;                           - CHARACTER TO SEND MUST BE IN
;                             REGISTER A
;                           - NO REGISTERS ALTERED
;
S1OUT:  push    af              ;Save character
S1OUT1: in0     a,(STAT1)       ;Get status
        and     02h             ;Strip for ready bit
        jr      z,S1OUT1                ;Loop til ready
        pop     af              ;Restore character
        out0    (TDR1),a        ;Send it out
        ret
```

```
;
;
;--- SEND A CHARACTER OUT SERIAL PORT #0 ---
;                          - SENDS A CHARACTER OUT THE PORT
;                            USED FOR RF TRANSMISSION
;                          - CHARACTER TO SEND MUST BE IN
;                            REGISTER A
;                          - NO REGISTERS ALTERED
;
S0OUT:  PROC
        push    af              ;Save character
 ..1:   in0     a,(STAT0)       ;Get status
        bit     1,a             ;Test ready bit
        jr      z, ..1          ;Loop til ready
        pop     af              ;Restore character
        out0    (TDR0),a        ;Send it out
        ENDPROC
        ret
;
;--- ENABLE SERIAL PORT #0 ---
;                          - SETS RTS LINE WHICH ENABLES
;                            THE RF TRANSMITTER
;                          - DELAY ALLOWS TIME FOR TRANSMITTER
;                            WARM UP
;                          - REGISTER A IS ALTERED
;
S0SET:  in0     a,(CNTLA0)      ;Get setup
        and     0EFh            ;Mask to set RTS line
        out0    (CNTLA0),a      ;Send it
        ld      b,08h           ;Load count for delay
        call    T1_DLY          ;Delay for .5 sec
        ret
;
;--- DISABLE SERIAL PORT #0 ---
;                          - CLEARS RTS LINE WHICH DISABLES
;                            THE RF TRANSMITTER
;                          - NO REGISTERS ALTERED
;
S0OFF:  push    af              ;Save register
        in0     a,(CNTLA0)      ;Get setup
        or      010h            ;Mask to clear RTS line
        out0    (CNTLA0),a      ;Send it
        pop     af              ;Restore register
        ret
;
```

```
;
;DEFINE STATEMENTS
;-------------------------------------------------------------------
;                           - CLOCK MESSAGES
;
CLKPRM:         db      CR,LF,LF,'Hit CR to set',CR,LF
                db      'the clock',CR,LF,0
;
CLKSETM:        db      'Enter time+1 min',0
;
CLKSTRTM:       db      'ENTER start time',0
;
CLKSYNCM:       db      CR,LF,LF,LF,'Setting the time',CR,LF,0
;
YMDHM:          db      'YR/MTH/DY/HR/MIN',CR,LF,0
;
;                           - ERROR MESSAGES
;
CHERRM:         db      'ERROR: too many',CR,LF
                db      'channels!',CR,LF,0
;
INTERRM:        db      'Bad interrupt !',0
;
MEMERRM:        db      'ERROR:not enough',CR,LF
                db      'memory!',CR,LF,0
;
OPCERRM:        db      'Bad Opcode trap !',0
;
RAMERRM:        db      'ERROR:bad RAM!',CR,LF,0
;
TIMERRM:        db      'ERROR:not enough',CR,LF
                db      'time!',CR,LF,0
```

```
;
;                       - ENTRY MESSAGES
;
BCHNLM:         db      'Base chan. ',0
;
CHANM:          db      '# chans.? ',0
;
CONTM:          db      'CR to continue',0
;
CRLFM:          db      CR,LF,0
;
MODE1M:         db      'Write additional',CR,LF
                db      'data between',CR,LF
                db      'main data sets?',CR,LF
                db      '(Y/N) ',0
;
MODE2M:         db      'Write only 1 set',CR,LF
                db      'of scans in each',CR,LF
                db      'data set?(Y/N) ',0
;
OKM:            db      LF,'Are entries',CR,LF
                db      'correct?(Y/N)',0
;
RADIOM:         db      LF,'Write data to',CR,LF
                db      'RADIO?(Y/N)',0
;
REMOTM:         db      'Remote event',CR,LF
                db      'mode?(Y/N)',0
;
SAMPL1M:        db      'Sample by the',CR,LF
                db      'hour or minute?',CR,LF
                db      '(H/M) ',0
;
SAMPL2M:        db      LF,'ENTER # samples',CR,LF
                db      'per day ',0
;
SAMPL3M:        db      LF,'ENTER # samples',CR,LF
                db      'per hour ',0
;
SCANM:          db      '# scans? ',0
;
SCANCTM:        db      LF,'Scan count in',CR,LF
                db      'data?(Y/N) ',0
;
SCNFRSTM:       db      'ENTER - 1st scan',CR,LF
                db      'set for all data',0
;
SCNMODE1M:      db      'ENTER - scan set',CR,LF
                db      'in-between data',0
;
SCNMAINM:       db      'ENTER - scan set',CR,LF
                db      'for main data',0
;
SCNMINM:        db      'ENTER scans/min',CR,LF
                db      '(1 - 30) ',0
```

```
;
SCNSECM:        db      'ENTER scans/sec',CR,LF
                db      '(1 - ',0
;
SCNSIM:         db      ') ',0
;
SOLIDM:         db      LF,'Write data to',CR,LF
                db      'S-STATE?(Y/N) ',0
;
SPWRM:          db      'Control power',CR,LF
                db      'to sensors?',CR,LF
                db      '(Y/N) ',0
;
SRATEM:         db      'Scan rate less',CR,LF
                db      'than 1Hz(Y/N)? ',0
;
STORM:          db      '1.Every time',CR,LF
                db      '2.Remote only',CR,LF
                db      'ENTER mode ',0
;
TAPEM:          db      LF,'Write data to',CR,LF
                db      'TAPE?(Y/N)',0
```

```
;
;                       - PLAYBACK MESSAGES
;
PHOURM:         db      'hr',0
;
PINM:           db      'in data',0
;
PMINM:          db      'min ',0
;
PNOTINM:        db      'not in data',0
;
PREMOTM:        db      'Remote Event is',CR,LF,0
;
PSAMPLM:        db      'data each ',0
;
PSCNCTM:        db      LF,'Scan count ',CR,LF,0
;
PSECM:          db      'sec ',0
;
PRADIOM:        db      'Data to RADIO',0
;
PSOLIDM:        db      'Data to S-STATE',0
;
PSTOR1M:        db      'At no time',CR,LF,0
;
PSTOR2M:        db      'Every time',CR,LF,0
;
PSTOR3M:        db      'Only during',CR,LF
                db      'remote events',0
;
PTAPEM:         db      'Data to TAPE',0
;
PSELM:          db      'selected',CR,LF,0
;
PNSELM:         db      'not ',0
;
PSCANAM:        db      'ist scan set',CR,LF,0
;
PSCANIM:        db      'In-between data',CR,LF,0
;
PSCANFM:        db      'Main data',CR,LF,0
;
PSCANSM:        db      'scan',CR,LF
                db      'ch ',0
;
TOM:            db      '-',0
```

```
;
;                       - DATA VAULT MESSAGES
;
DVU0_MSG:       db      'Unit 0 ',0
;
DVU1_MSG:       db      'Unit 1 ',0
;
DVNP_MSG:       db      'Not '
;
DVUP_MSG:       db      'Present',0
;
SCTM:           db      'ENTER number of sectors to write ',0
;
LABLP:          db      '         PRI',20h,0h,0h,0h,0h
                db      0,0,0,0,0,0,87h,9Ah,9Bh,0Ah
;
LABLI:          db      '         INT',20h,0h,0h,0h,0h
                db      0,0,0,0,0,0,87h,9Ah,9Bh,0Ah
;
DOSERR1:        db      'ERROR: incorrect format',0
;
DOSERR2:        db      'ERROR: out of directory space',0
;
DOSERR3:        db      'ERROR:Out of disk space',0
;
DOSERR4:        db      'ERROR:Not enough room to write',0
;
DVE0_MSG:       db      'Error, status: ',0

;
;                       - TEST MESSAGES
;
ADMESS:         db      CR,LF,LF,'ENTER channel #',CR,LF
                db      'to test ',0
;
ADCHM:          db      CR,LF,'Value for CH ',0
;
```

```
;
;--- TABLE FOR LOADING SERIAL PORT PARAMETERS ---
;
SERTBL: db      024h            ;Control reg. A value SER0
                                ;MPE disabled
                                ;Tx enable
                                ;RX disabled
                                ;RTS low
                                ;1 Start + 8 bit data + 1 stop
        db      074h            ;Control reg. A value SER1
                                ;MPE disabled
                                ;Tx enable
                                ;RX enabled
                                ;1 Start + 8 bit data + 1 stop
        db      08h             ;Control reg. B value SER0
                                ;1200 baud, no parity
        db      02h             ;Control reg. B value SER1
                                ;9600 baud, no parity
        db      00h             ;Status reg. value SER0
                                ;No interrupts
        db      0Ch             ;Status reg. value SER1
                                ;Rx interrupts, set CTS
;
;--- TABLE FOR LOADING TIMER PARAMETERS ---
;
TIMTBL: db      017h            ;CSI/O control reg.
                                ;Must be set to enable SER0
        db      0h              ;Dummy parameter
        db      0Ch             ;Low-byte count for PRT0
        db      0Ch             ;High-byte count for PRT0
        db      0FFH            ;Low byte reload for PRT0
        db      0FFh            ;High-byte reload for PRT0
        db      00h             ;Disable timers
        db      00h             ;Dummy parameter
        db      00h             ;Dummy parameter
        db      0Fh             ;Low-byte count for PRT1
        db      0C8h            ;High-byte count for PRT1
        db      0FFH            ;Low byte reload for PRT1
        db      0FFh            ;High-byte reload for PRT1
;
;--- TABLE FOR LOADING DMA PARAMETERS ---
;
DMATBL: db      0Ch             ;Set DMA status
        db      00h             ;Set DMA mode
        db      01Ch            ;DMA control value
        db      00h             ;Refresh control value
```

```
;
;--- JUMP TABLE FOR INTERNAL INTERRUPTS ---
;
;                          -  SETS JUMP POINTS FOR
;                             INTERNAL INTERRUPTS
;
        DEFSEG  VECTORS,START=7F00h,ABSOLUTE
        SEG     VECTORS
        org     07F00h
;
ILTBL:  db      040h,00h            ;INT 1
        db      043h,00h            ;INT 2
        db      046h,00h            ;Timer 0
        db      049h,00h            ;Timer 1
        db      04Ch,00h            ;DMA 0
        db      04Fh,00h            ;DMA 1
        db      052h,00h            ;CSI/O
        db      055h,00h            ;SER 0
        db      058h,00h            ;SER 1
;
        END
```