UNITED STATES DEPARTMENT OF INTERIOR

GEOLOGICAL SURVEY

OCEAN BOTTOM SEISMOMETER (OBS) SOFTWARE:
MODIFICATION III

by

G. K. Miller[1]

Open-file Report 86 - 269

This report is preliminary and has not been reviewed for conformity with U.S. Geological Survey editorial standards. Use of tradenames is for purposes of identification only and does not constitute any endorsement by the USGS.

Woods Hole, Mass.

1986

OCEAN BOTTOM SEISMOMETER (OBS) SOFTWARE:

MODIFICATION III

This open-file publication is a listing of the modified software program for the U.S. Geological Survey´s Ocean Bottom Seismometer (OBS). This listing supercedes the listing contained in Open-File Report 84 - 842: Ocean Bottom Instrument Package (OBIP) Software: Modification II. Portions of the original program were written by John Godley (WHOI) and Ogden Hammond under contract to the U.S. Geological Survey.

SOFTWARE LISTING: MOD III, OBS

PROGRAMMING INSTRUCTIONS FOR THE USGS 4-CHANNEL OBS

PROGRAM STARTUP

1. Connect the RS232 cable from a standard computer terminal to the RS232 connector on the OBS electronics rack (Figure 1). The terminal should be setup for space parity, 7 data bits, 1200 baud, 1 stop bit, and no handshake lines.

2. Install a blank data tape into the tape recorder of the OBS.

3. Apply Power by connecting the plug from the battery pack to the plug located on the power interface board (Figure 1).

4. Press the reset button on power interface board in the OBS (Figure 1) to start the OBS program.

5. The system will start by displaying the random access memory (RAM) test. This test will take approximately 30 seconds to perform, and any failures found will be displayed on the terminal. If you do encounter a RAM error, remove power and replace the CPU board. After the successful completion of the RAM test, the following message will be displayed at the lower left corner of your screen:

```
"ENTER CURRENT TIME PLUS 1 MINUTE"
"YR/Mth/Day/Hr/Min?"
```

6. The first information entered into the OBS is the current time , which is used by the OBS to set its calendar in sync with the satellite clock or other time reference. This time sync procedure is important as time is the only reference for data recorded by the OBS system. Connect the satellite clock to the CPU board of the OBS (Figure 1). Read the current time from the satellite clock and add at least one minute. It is critical that you enter the time correctly, as there is no way for the instrument to verify your accuracy. Time is entered into the OBS in the following format: Year (YY = last two digits of the 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(RETURN)
```

The entry must be followed by a carriage return (RETURN) to enter the time into the OBS.

NOTE: This instrument is programmed on the basis of a 24-hour clock notation (i.e., each day extends from 00:00 to 23:59). This clock cannot correct for leap year; in leap years, the clock will be off by one day if the transition from FEB 29 to MAR 1 occurs during deployment.

7. The OBS will check the entry for any mistakes in format, and will ask again for the time entry if a error is found. If the entry made is in the correct format but is not the correct time, push the reset button on the power interface board to restart the program. If the time entry

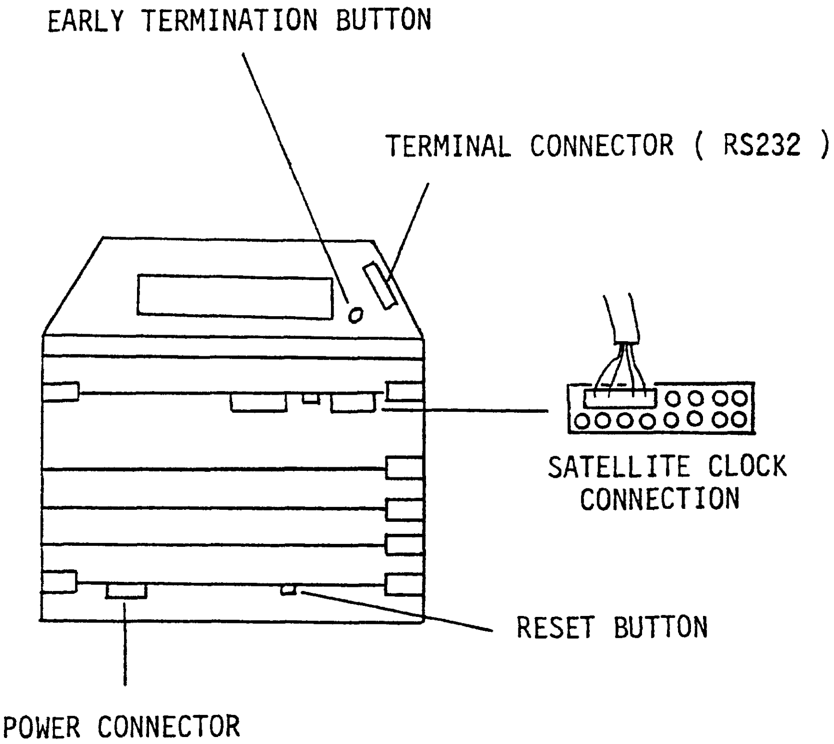

Figure 1. OBS Connections

is valid, the following message will appear on the terminal:

"HIT ANY KEY TO CONTINUE"

To continue the program, simply hit any key (using the RETURN key is suggested). The screen will then be cleared, and the following instruction will appear in the lower left corner of the screen:

"HIT RETURN WHEN LESS THAN ONE MINUTE TO GO"

Immediately below this will appear a message indicating whether or not the satellite clock is correctly connected. Just below this, the time you entered earlier will appear. Your response at this point depends on where or not the satellite clock is connected. The following descriptions provide the course of action for either condition:

A. The system displays the following message:

"SATELLITE CLOCK NOT CONNECTED"

This means that either the OBS has failed to detect that the satellite clock is connected, or that no satellite clock is available to sync the time.

If the satellite clock is connected, check to be sure the connector is correctly installed on the CPU board (Figure 1). To test whether or not the problem has been corrected, press RETURN. A successful repair will display nothing and the OBS will wait for the clock pulse to start the internal clock as described in clock-connected section (B). The following message will appear if the clock is still not properly connected:

"HIT ANY KEY TO CONTINUE"

At this point, recheck the connections and the satellite clock output to be sure of proper operation. Hitting any key, will cause the screen to clear and the satellite clock connection message to be redisplayed. If the message

"SATELLITE CLOCK NOT CONNECTED"

appears, you will almost certainly miss your synchronous start. To avoid that, push the reset button on the power board and start again. If the problem persists, change the CPU board and try again.

If you do not have access to a satellite clock or any other type of time sync, there is an alternate way to start the clock. If you type "Control B", the clock will start. Note that this method is accurate within one second and depends heavily on how accurate you are at the moment you strike the keys.

B. Assuming a normal state of affairs, the following message will be displayed:

"HIT RETURN WHEN LESS THAN ONE MINUTE TO GO"
"SATELLITE CLOCK CONNECTED"

This indicates that the OBS has successfully detected the satellite clock connection and is ready to start the internal clock. Read the time from the satellite clock. When the clock´s time is less than a minute behind the time you entered, press RETURN. The internal clock of the OBS will start when the minute pulse from the satellite clock is detected.

8. At this point, the OBS will automatically perform a tape test. The data tape, if already in the drive, will be rewound to the beginning of tape (BOT). If there is no tape in the OBS, The following message will appear on the screen and will be repeated until a tape is inserted into the drive:

"INSERT TAPE CARTRIDGE"

Once the auto-rewind sequence is completed, The OBS will check to see if the tape is write protected. If it is, the message

"CODE 1 WRITE PROTECTED"
"REMOVE TAPE CARTRIDGE"

will be repeated until the tape is removed. Once the tape has been removed, the insert tape message will be repeated until a new tape is installed. At this point, a test pattern is generated in memory , written to tape, and the message

"TESTING TAPE"

will appear. The OBS will then clear the RAM and read the test pattern from the tape for comparison to the original. If there is a failure, the messages

"BAD TAPE OR DRIVE"
or
"TIME OUT ERROR"

will appear. Install a new tape in the tape drive and restart the program by pushing the reset button. If this fails, check the power connections to be sure the proper voltages are getting to the drive. If this still doesn´t work, the system needs major repair.

If no errors occurred, the message

"TAPE DRIVE OK"
"HIT ANY KEY TO CONTINUE"

will appear in less than 15 seconds. Disconnect the satellite clock and hit any key to continue.

9. The screen will clear and the prompt

```
"DEPLOYMENT #    ?"
```

will appear in the lower left corner of the screen. This is the first of the "header" prompts. The OBS provides no checking of these entries, since they have no effect on the operation of the system. The "header" prompts are provided as a convenience to record variables which identifies a particular data tape. Any entry of up to 80 characters (including hitting RETURN at the end) can be entered for each prompt. If no entry is desired, then hit RETURN. If you notice your mistake in an entry before pressing RETURN, the normal CPM editing keys (Back Space, Delete, or Control X) can be used. The following prompts will appear after the previous one has been entered:

```
"INSTRUMENT #    ?"
"CHIEF SCIENTIST?"
"CRUISE #        ?"
"SPHERE #        ?"
"LATITUDE        ?"
"LONGITUDE       ?"
"FRONT END GAIN"
"CHANNEL 1  ?"
"CHANNEL 2  ?"
"CHANNEL 3  ?"
"CHANNEL 4  ?"
"FRONT END DAMPING"
"CHANNEL 1  ?"
"CHANNEL 2  ?"
"CHANNEL 3  ?"
"CHANNEL 4  ?"
```

After the last prompt has been entered , the message

```
"IS THIS CORRECT (Y/N)?"
```

will appear. If there is an error in your entries, answering no (N) will restart the "header" prompts at the beginning. All prompts will have to be reentered. If there are no errors, answering yes (Y) will store the entries, and the program will move on to the experiment parameters. No RETURN is necessary in answering this question, as the OBS will respond immediately to the first key entered.

Remember that your response, or lack of it,has no effect on the performance of the instrument.

10. At this point, the experimental parameters which affect the operation of the OBS are entered. There are two basic modes of operation for the OBS: the Timer (or window) mode and the Event mode. The OBS can operate in either mode but not simultaneously.

The screen will clear and the following message will be displayed in the lower left corner:

"SERIES #1"

"# OF CHANNELS (1-4)"

"SERIES #1" indicates that the parameters that are about to be entered are for the first series. There can be up to eight series per deployment, and each series represents a group of data recordings which are defined by the parameters entered for that series.

There are four parameters which are common to both modes of operation, so they are entered before determining which mode is desired. These four parameters are combined to determine the length of time for each data recording and include the number of channels, the base channel, the record size, and the sample rate. If the length of the data recording is important, some calculations should be done prior to parameter entry. The OBS calculates the length of recording after these parameters are entered, but at that point, all of the rest of the parameters must be entered before there is an chance of reentering the first four entries again. The length of recording can be determined using the number of channels(1 to 4), record size (1, 2, or 4 8-K byte blocks), and sample rate (1, 2, 4, or 8 milliseconds) in the following equation:

$$\frac{[(\text{record size})\text{x}(8192\ \text{bytes/block})-256\ \text{bytes}]\text{x}(\text{sample rate})}{(\text{no. channels})\text{x}(2\ \text{bytes/channel})\text{x}(1000\ \text{msec/sec.})}$$

The 8192 bytes/block converts the record size value to the total number of bytes of data recorded. The 256 bytes is subtracted from this value to account for the loss of memory space due to the 256-byte trailer written on the data tape at the end of each data recording. The 2 bytes/channel converts the number of channels to the total number of bytes recorded for all channels. The 1000 msec./sec. converts the sampling rate from milliseconds to seconds between data. The options for entering these parameters are as follows:

The first prompt is shown above and requests an entry for the number of channels that are recorded at each data recording. The OBS can record from as few as one or as many as four channels simultaneously. The number 1, 2, 3, or 4 is entered and followed by a carriage return (RETURN). The OBS will check to see if the entry was valid, and if not, reprompt for the number of channels. If the entry is valid, the following prompt will appear:

"BASE CHANNEL (1-4)"

The base channel is the first active channel used to record data, and the other active channels must lie above it. In other words, if four channels are active (i.e, the total number of channels is 4), then channel 1 must be the base channel. If three channels are active, then either channel 1 or channel 2 can be the base channel. A base channel of one would make channels 1, 2, and 3 active. A base channel of 2 would make channels 2, 3, and 4 active. The number 1, 2, 3, or 4 is entered (followed by a RETURN). The OBS will check to see if this entry is valid when compared with the total number of channels entered earlier. If the entry is not valid, the OBS will go back and reprompt for both the total number of channels and the base channel. If the entry is valid then the following prompt will appear:

```
"ENTER 1, 2, or 4 BLOCKS OF 8K"
"RECORD SIZE = "
```

The series record size can be one of three lengths -- either 8-K (an entry of 1), 16-K (an entry of 2), or 32-K (an entry of 4) bytes long. This entry represents the total amount of memory used to store the data. The number 1, 2, or 4 is entered (followed by a RETURN). If the entry is not valid, the OBS will go back and reprompt for the record size. If the entry is valid then the following prompt will appear:

```
"ENTER 1ms, 2ms, 4ms, or 8ms "
"SAMPLE RATE = "
```

The sample rate determines how fast the data is acquired. The sample rate can be either 8 ms (128 hertz), 4 ms (256 hertz), 2 ms (512 hertz), or 1 ms (1024 hertz). Data are gathered simultaneously from all active channels at the selected rate until the memory space, determined by the record size entry, is full. The number 1, 2, 4, or 8 is entered (followed by a RETURN). The OBS will check this entry, and reprompt for the sample rate if the entry is not valid. If the entry is valid, the OBS will display a warning message to make sure that the proper resistor headers have been installed in the analog board of the system. The data are filtered at different frequencies for each sampling rate, and four resistor headers located on the analog board set these frequencies. The warning message will indicate which resistor headers should be installed. If the wrong headers are installed, the power must be removed, the analog board removed, the correct headers installed, and the program restarted from the beginning. The warning message is followed by the record time as calculated by the OBS and by a prompt requesting the mode of operation. As an example, 4 is entered for the number of channels, 1 is entered for the base channel, 4 is entered for the record size, and 8 is entered for the sampling rate. This would result in the following message:

```
"164K RESISTOR HEADERS ON FILTER BOARD FOR THIS SAMPLE RATE"
"*** WARNING ***"

"RECORD TIME = 32 sec"

"TIMER or EVENT MODE (T/E)"
```

11. At this point, you must decide which mode of operation (i.e. timer or event) this series will use. If event mode is desired, skip this step and proceed to step 12.

Timer mode in the OBS records data at fixed intervals during a predetermined "window" of time. This mode of operation is commonly used for refraction lines where a acoustic source is triggered at well defined intervals of time. The OBS, which at this point is on the ocean floor, will record data at times in sync with the time that the sound source is triggered. The series start and stop times usually correspond to the beginning and end of the refraction line. When the start time for the series is reached, the OBS will record data in fixed, minute intervals of time until the stop time or a maximum number of data recordings is reached. The specific parameters controlling when, how often, and how many times the OBS records data are described in the following paragraphs.

To program the OBS for timer mode, simply enter the letter "T" followed by a RETURN. The following message will then appear:

"# OF RECORDS = "

The number of records is the total number of data recordings to be done during this series. The maximum number of records that the OBS software can handle is 9999, which is a larger value than the current data tape can record. The maximum number of records that can be recorded onto the data tape is determined by knowing the total number of series that will be programed for this deployment, the total number of records for each series, and the record size for each series. The data tape records up to 2000, 8-K byte blocks of data. The record size, entered earlier, equals the number of 8-K byte blocks written to tape for each data recording, so the maximum number of records for one series is determined by dividing the record size into 2000. For example, the data tape can record up to 500 records (data recordings) with a record size of 4. If you desire more than one series per deployment, some juggling of numbers will be necessary to determine how many records per series can fit onto the data tape. A number from 1 to 9999 is entered (followed by a RETURN). The OBS will check this entry, and reprompt for the number of records if the entry is not valid. If the entry is valid, the following messages (using series 1 and 2 as examples) will appear:

"TIME NOW: 3/28/86 12:34"

"START TIME SERIES 1 (YR/MTH/DAY/HR/MIN)"

or

"TIME NOW: 3/28/86 12:34"

"STOP TIME OF LAST SERIES 3/29/86 17:05"

"START TIME SERIES 2 (YR/MTH/DAY/HR/MIN)"

This prompt requests the start time for this series of records. The OBS uses this time to determine when to begin acquiring data at regular intervals, and the time usually corresponds the time of the first sound source trigger for a refraction line. The time entry is done in the same format as described in step 6, and the same checking is done by the OBS. The system will also check to be sure that you have not entered a start time that is earlier than the current time, for obvious reasons, and reprompt if an error is made. The current time is displayed in the first line to help prevent this error. The display of current time is also be used to check the operation of the clock in the OBS. Check this time against your time reference. If they do not agree, push the reset button and start over. If this does not work, change the CPU board and begin again. After series 1, the OBS will check the start time against the stop time of the last series to be sure that there is no overlap between the two series. The stop time for the last series is displayed to help prevent any errors. If the system finds that the start time is before the stop time of the last series the following message (using series 1 and 2 as examples) will appear:

"IMPROPER START TIME "

"TIME NOW: 3/28/86 12:34"

"STOP TIME OF LAST SERIES 3/29/86 17:05"

"START TIME SERIES 2 (YR/MTH/DAY/HR/MIN)"

This checking will repeat until a valid time is entered. If the start time is still in error, in spite of all the checking done by the OBS, you have another opportunity to correct the mistake at the end of the parameter entry for the series. This requires, however, that all of the parameters for the series be reentered. After successful entry of the start time, the following message (using series 1 as an example) will appear:

"STOP TIME SERIES 1 (YR/MTH/DAY/HR/MIN)"

When the stop time is reached, the OBS will stop collecting data and automatically setup for the next series (or terminate data collection completely if there is no other series). This stop time has precedence over any other parameter (i.e., number of records) and will unconditionally terminate the series. The time is entered in the same manner as described for the start time. The OBS will check to see that the stop time is later than the start time and reprompt, if an error is found, as follows:

"IMPROPER STOP TIME"
"START TIME SERIES 1 (YR/MTH/DAY/HR/MIN)"

Notice that the system is asking for the start time again rather than the stop time. You must reenter the start time and the stop time, as the OBS has no way of determining which time is in error. If the stop time is still in error, in spite of all the checking done by the OBS, you have another opportunity to correct the mistake at the end of the parameter entry for the series. This requires, however, that all of the parameters for the series be reentered. After successful entry of the stop time, the following message will appear:

"ENTER (0 - 59 sec.)"
"WINDOW OFFSET = "

There are situations where it is desirable to start acquiring data at a time "offset" from the time of sound source trigger used in a refraction lines. In other words, to start data-acquisition several seconds after the sound source has been triggered. This is particularly useful if something (such as explosives) are detonated on the even minute and the time that soundwaves must travel is known. The offset will save wasting valuable data space if no useful data can be acquired for the several seconds after the detonation. The offset entry can be any number of seconds from 0 to 59 (followed by hitting RETURN), and represent the number of seconds after the even minute that the OBS will begin acquiring data. Remember that this offset will be applied to every record in this series. The OBS will check this entry, and reprompt for the offset if the entry is not valid. If the entry is valid, the following message will appear:

```
"ENTER (1-99 min.)"
"PERIOD OF RECORDS = "
```

The period of records is the time from the beginning of one record to the beginning of the next record. This entry determines how often the data will be acquired between the start and stop times. The minimum period of records is calculated by the OBS. If the time required to fill the data buffer and write the data to tape exceeds one minute, the prompt will appear as follows:

```
"ENTER (2-99 min.)"
"PERIOD OF RECORDS = "
```

The period number is entered and followed with a RETURN. The OBS will check the entry to be sure you are within the minimum and maximum values and reprompt if in error. If the entry is valid, the following message will appear:

```
"HIT ANY KEY TO CONTINUE"
```

You have reached the end of the parameter entry for this series. Hitting any key (RETURN is suggested), will cause the screen to clear and all of the entered parameters to be displayed. At the end of this list, you will be asked if these entries are correct. As an example, you entered, for series 1, a value of 4 for the number of channels, 1 for the base channel, 4 for the record size, 8 for the sampling rate, "T" for the mode of operation, 100 for the number of records, 86/3/28/17/05 for the start time, 86/3/28/21/00 for the stop time, 4 for the offset, and 2 for the period. The following information will be displayed after the last parameter has been entered:

```
"SERIES 1"
"# OF RECORDS = 100"
"TIMER MODE"
"START    3/28/86 17:05"
"STOP     3/28/86 21:00"
"ACTIVE CHANNEL(S) = 1,2,3,4
"RECORD SIZE = 32K"
"SAMPLE RATE = 8ms"
"RECORD TIME = 32sec."

"WINDOW OFFSET = 4sec."
"PERIOD = 2min."

"IS THIS CORRECT (Y/N)?"
```

If you answer N (no) to this question, you must reenter all of the parameters for this series. If you answer Y (yes) to this question go to step 13. No RETURN is necessary in answering this question, as the OBS will respond immediately to the first key entered.

12. Event mode in the OBS records data triggered by an external seismic event. This mode is used when the acoustic source does not occur at defined intervals of time, as is the case in earthquake detection studies. The OBS has an event-detector as part of its circuitry, which monitors the data continuously and initiates data recording when a seismic event (i.e., earthquake) has been detected. The system continuously puts data into a circular buffer which has been defined by the parameters entered in step 10. When an event occurrs, the OBS will continue to record this data until the buffer is filled and then write the data to the data tape. The time of the event is recorded as the time the data buffer is full. The time is recorded at the end of data collection to prevent the loss of data that would occur if the system read the clock at the moment of the event (the OBS cannot do two things at once). The exact time of the event can only be calculated when the data is processed. Note that because of the filter design (and the very low frequencies that are of interest to geologists), there exists a significant phase lag in the data. Arrival times computed from data gathered by this mode or the Timer mode could easily be off by 50% of the period of the waveform of interest. The following parameters are entered to setup the event-detector, and establish when and how much data is recorded. To enter event mode for this series type an "E" followed by a RETURN, and the following prompt will appear:

```
"ENTER 87(.5), 75, 50, or 25%"
"POST-EVENT SAMPLE (%) "
```

The post-event sample represents the percentage of the data buffer that is recorded after an event is detected and determines when the OBS will stop collecting data for this event. The "87(.5)" option is entered by typing 87 even though the actual value is 87.5. This was done to simplify the software in the OBS required to check the entry for errors. The number 87, 75, 50, or 25 is entered (followed by a RETURN). The OBS will check the entry and reprompt if an error is found. If the entry is valid, the following message will appear:

```
"ENTER .05, .10, .25, or .50 SEC."
"STA TIME CONSTANT = "
```

The short-term averager (STA) time constant represents the time base, used in the event-detector, for averaging the short-term data signals. The smaller the time constant, the faster the event-detector will react. The STA entry must be exactly as shown in the prompt, or the OBS will not accept it. The entry must be .05, .10, .25, or .50 (followed by a RETURN). The OBS will check the entry and reprompt if an error is found. If the entry is valid, the following message will appear:

```
"ENTER 6, 12, 18, or 24 db"
"THRESHOLD = "
```

The threshold value for the hardware event-detector represents a fixed signal level above the long-term average level that is used to determine if an event has occurred. Although the entry is an exact number, the actual value is highly dependent on the hardware and the stability of the background noise. This entry does give some measure of control in determining how far above the background noise the event will occur. The number 6, 12, 18, or 24 is entered (followed by a RETURN). The OBS will check the entry and reprompt if an error is found. If the entry is valid, the following message will appear:

"# OF RECORDS = (Enter 0 for maximum number)"

In most event-detection deployments, the number of records (data recordings) will be unimportant. Event detection will occur until the data tape is filled, or the OBS is retrieved. If this is the case, simply enter the number 0 followed by a RETURN. If, however, you desire to end this series after a certain number of events, enter a number from 1 to 9999 followed by a RETURN. The number of records,in this case, is the total number of events to be done during this series. The maximum number of records that the OBS software can handle is 9999, which is a larger value than the current data tape can record. The maximum number of records that can be recorded onto the data tape is determined by knowing the total number of series that will be programed for this deployment, the total number of records for each series, and the record size for each series. The data tape records up to 2000, 8-K byte blocks of data. The record size, entered earlier, equals the number of 8-K byte blocks written to tape for each data recording, so the maximum number of records for one series is determined by dividing the record size into 2000. For example, the data tape can record up to 500 records (data recordings) with a record size of 4. If you desire more than one series per deployment, some juggling of numbers will be necessary to determine how many records per series can fit onto the data tape. The OBS will check this entry, and reprompt for the number of records if the entry is not valid. If the entry is valid, the following messages (using series 1 and 2 as examples) will appear:

"TIME NOW: 3/28/86 12:34"

"START TIME SERIES 1 (YR/MTH/DAY/HR/MIN)"

or

"TIME NOW: 3/28/86 12:34"

"STOP TIME OF LAST SERIES 3/29/86 17:05"

"START TIME SERIES 2 (YR/MTH/DAY/HR/MIN)"

This prompt requests the start time for this series of records. The OBS uses this time to determine when to begin acquiring data in the event mode. Allow 8 hours after the start time for the event-detector to stabilize before valid events will be recorded. This is due to the slow response of the long-term averager, in the event-detector, to background noise. The time entry is done in the same format as described in step 6, and the same checking is done by the OBS. The system will also check to be sure that you have not entered a start time that is earlier than the current time, for obvious reasons, and reprompt if an error is made. The current time is displayed in the first line to help prevent this error. The display of current time is also be used to check the operation of the clock in the OBS. Check this time against your time reference. If they do not agree, push the reset button and start over. If this does not work, change the CPU board and begin again. After series 1, the OBS will check the start time against the stop time of the last series to be sure that there is no overlap between the two series. The stop time for the last series is displayed to help prevent any errors. If the system finds that the start time is before the stop time of the last series the following message (using series 1 and 2 as examples) will appear:

```
"IMPROPER START TIME "

"TIME NOW: 3/28/86 12:34"

"STOP TIME OF LAST SERIES 3/29/86 17:05"

"START TIME SERIES 2 (YR/MTH/DAY/HR/MIN)"
```

This checking will repeat until a valid time is entered. If the start time is still in error, in spite of all the checking done by the OBS, you have another opportunity to correct the mistake at the end of the parameter entry for the series. This requires, however, that all of the parameters for the series be reentered. After successful entry of the start time, the following message (using series 1 as an example) will appear:

```
"STOP TIME SERIES 1 (YR/MTH/DAY/HR/MIN)"
```

When the stop time is reached, the OBS will stop collecting data and automatically setup for the next series (or terminate data collection completely if there is no other series). This stop time has precedence over any other parameter (i.e., number of records) and will unconditionally terminate the series. In event mode, however, the stop time is not checked until the end of an event, so one event must occur after the stop time before this series will end. The time is entered in the same manner as described for the start time. The OBS will check to see that the stop time is later than the start time and reprompt, if an error is found, as follows:

```
"IMPROPER STOP TIME"
"START TIME SERIES 1 (YR/MTH/DAY/HR/MIN)"
```

Notice that the system is asking for the start time again rather than the stop time. You must reenter the start time and the stop time, as the OBS has no way of determining which time is in error. If the stop time is still in error, in spite of all the checking done by the OBS, you have another opportunity to correct the mistake at the end of the parameter entry for the series. This requires, however, that all of the parameters for the series be reentered. After successful entry of the stop time, the following message will appear:

"HIT ANY KEY TO CONTINUE"

You have reached the end of the parameter entry for this series. Hitting any key (RETURN is suggested), will cause the screen to clear and all of the entered parameters to be displayed. At the end of this list, you will be asked if these entries are correct. As an example, you entered, for series 1, a value of 4 for the number of channels, 1 for the base channel, 4 for the record size, 8 for the sampling rate, "E" for the mode of operation, 75 for the post-event samples, .10 for the STA time constant, 12 for the threshold, 0 (maximum) for the number of records, 86/3/28/12/00 for the start time, and 86/4/10/8/00 for the stop time. The following information will be displayed after the last parameter has been entered:

```
"SERIES 1"
"# OF RECORDS = 9999"
"EVENT MODE"
"START    3/28/86 12:00"
"STOP     4/10/86 8:00"
"ACTIVE CHANNEL(S) = 1,2,3,4
"RECORD SIZE = 32K"
"SAMPLE RATE = 8ms"
"RECORD TIME = 32sec."

"POST-EVENT SAMPLE 75%"
"STA TIME CONSTANT = .10sec."
"THRESHOLD = 12db"

"IS THIS CORRECT (Y/N)?"
```

If you answer N (no) to this question, you must reenter all of the parameters for this series. If you answer Y (yes) to this question go to step 13. No RETURN is necessary in answering this question, as the OBS will respond immediately to the first key entered.

13. If you answer Y (yes) to this question, the following question will appear:

"DO YOU WISH ANOTHER SERIES?"

If you answer Y (yes) to this question, the parameter entry routines for the next series will begin. No RETURN is necessary in answering this question, as the OBS will respond immediately to the first key entered. If you answer N (no) to this question, you will be asked the following question:

"IS THIS A TEST? (N/Y)"

Answering Y (yes) to this question is strictly for testing purposes and should not be used in field work. In testing mode, the header is written to tape immediately, and series information is displayed on the terminal.

The terminal is usually left connected to monitor progress of each series. This mode has none of the checking procedures used to ensure that the system is working before deployment.

If you answer N (no) to this question, you will be instructed to disconnect the terminal. After the terminal is unplugged, you should be able to hear the tape drive turn on to write the header information to tape. If this happens, you have successfully programmed the OBS and are ready for deployment.

If this does not happen, something went wrong. It is strongly suggested that you start over.

PROGRAM TERMINATION

1. Push the Early Termination button (Figure 1) to end data gathering.

2. Connect the terminal as described in step 2 of the program startup section and hit RETURN. The OBS should write two end of file marks on the data tape and display the time once a second on the terminal. If this does not happen, try entering a Control U. This is a keyboard interrupt that goes directly to the clock read routines. If the time still does not appear on the terminal, press the reset button on the Power Interface Board (Figure 1). After the RAM test is completed and when the OBS asks you to enter the time, enter a Control U instead. If the clock is still working, you will see the current time displayed on the terminal. The year, in this case, will be zeros.

3. Connect the satellite clock (Figure 1) and check the time sync.

4. Remove the data tape and set the write protect.

5. Disconnect the power.

PROGRAM LISTING

FOR THE USGS 4 CHANNEL OBS

SYSTEM EQUATES

```
;
;________________________________________________________________
;Original code was written by jhg
;UPDATE -  8/8/84 code was revised for window mode by ohh
;UPDATE -  9/6/84 Code was revised to combine W.EQU,
;         WNSC810.EQU, and WPWRPRT.EQU with diagnostic
;         switch by gkm
;UPDATE -  9/13/85 code was revised to alter memory
;         locations of series parameters to accomodate
;         expanded series and exp. numbers by gkm
;UPDATE -  2/24/86 code was revised to accomodate the
;         new clock (58167) by gkm
;________________________________________________________________
```

```
; ASSEMBLY CONTROL
;________________________________________________________________

FALSE     equ     0
TRUE      equ     NOT FALSE
DIAG      equ     FALSE            ;Diagnostic switch
OUTP      equ     01h              ;Define as output
INP       equ     00h              ;Define as input
HI        equ     01h              ;Define as high
LO        equ     00h              ;Define as low

; WINDOW CONSTANTS
;________________________________________________________________

BIT0      equ     01h              ; \
BIT1      equ     02h              ;  \
BIT2      equ     04h              ;   \
BIT3      equ     08h              ;    \ Use as values for
BIT4      equ     10h              ;    /  bit masks
BIT5      equ     20h              ;   /
BIT6      equ     40h              ;  /
BIT7      equ     80h              ; /

; ASCII EQUATES
;________________________________________________________________

BS        equ     08h              ;Back space
CR        equ     0dh              ;Carriage return
CTLU      equ     15h              ;Control U (jump to clock read)
CTLX      equ     18h              ;Control X
CTLZ      equ     1ah              ;Control Z (string terminator)
DEL       equ     7fh              ;Delete
LF        equ     0ah              ;Line feed
SPC       equ     20h              ;Space

; POWER BOARD EQUATES
;________________________________________________________________

;--- PORT LOCATIONS ---

PWRPRT    equ     0ffh             ;Power interface board port

;--- COMMAND EQUATES ---

LCDCLL    equ     01h              ;Set LCD clock line low
LCDCLH    equ     81h              ;Set LCD clock line high
LCDRST    equ     02h              ;Pull reset low on LCD ctr
NLCDR     equ     82h              ;Reset hi on LCD (not reset)
OFF12V    equ     03h              ;Turn off 12v to cartridge
ON12V     equ     083h             ;Turn on 12v to cartridge
OFF5V     equ     00h              ;Turn off 5v to cartridge
ON5V      equ     080h             ;Turn on 5v to cartridge
```

```
; ANALOG BOARD EQUATES
;_______________________________________________________________

;--- PORT LOCATIONS ---

ANAPRT   equ     010h              ;Analog board STA & threshold
AVGPRT   equ     ANAPRT+01h        ;Analog board average enable

; A-D BOARD EQUATES
;_______________________________________________________________

;--- PORT LOCATIONS ---

ADPORT   equ     018h              ;A-D board base address

; CONTROL BOARD EQUATES
;_______________________________________________________________

; CPU
;---------------------------------------------------------------

;--- PORT LOCATIONS ---

INTPRT   equ     0bbh              ;NSC-800 interupt mask port

;--- COMMAND EQUATES ---

AD_INT_EN  equ   08h               ;RSTA from A-D enable
T1_INT_EN  equ   04h               ;Timer 1 interrupt enable
RSTC_EN    equ   02h               ;RSTC enable
EARLY_TERM
           equ   01h               ;Early terminator enable
```

```
; NSC810
;--------------------------------------------------------------

;--- PORT LOCATIONS ---

NSCIOT   equu   3080h           ;NSC810 I/O-timer base
PBDATA   equ    NSCIOT+01h      ;NSC810 port B- data reg.
PCDATA   equ    NSCIOT+02h      ;Port C data reg
PBDDIR   equ    NSCIOT+05h      ;Port B data direction reg
PCDDIR   equ    NSCIOT+06h      ;Port C data direction reg
NSCMDR   equ    NSCIOT+07h      ;Mode definition reg
PBCLRB   equ    NSCIOT+09h      ;NSC810 port B- bit clear reg.
PCCLRB   equ    NSCIOT+0ah      ;Port C bit clear reg
PBSETB   equ    NSCIOT+0dh      ;NSC810 prot B- bit set reg.
PCSETB   equ    NSCIOT+0eh      ;Port C bit set reg
TMR0     equ    NSCIOT+010h     ;Timer 0 reg
TMR0LS   equ    NSCIOT+010h     ;Timer 0 lsb
TMR0MS   equ    NSCIOT+011h     ;Timer 0 msb
TMR1     equ    NSCIOT+012h     ;Timer 1 reg
TMR1LS   equ    NSCIOT+012h     ;Timer 1 lsb
TMR1MS   equ    NSCIOT+013h     ;Timer 1 msb
STOPT0   equ    NSCIOT+014h     ;Timer 0 stop
STRT0    equ    NSCIOT+015h     ;Timer 0 start
STOPT1   equ    NSCIOT+016h     ;Timer 1 stop
STRT1    equ    NSCIOT+017h     ;Timer 1 start
CMDT0    equ    NSCIOT+018h     ;Timer 0 command reg
CMDT1    equ    NSCIOT+019h     ;Timer 1 command reg

;                       - ADDRESSES RELATIVE TO NSCIOT(IN IX REG.)

PADADD   equ    0h              ;Port A data reg
PBDADD   equ    01h             ;Port B data reg
PCDADD   equ    02h             ;Port C data reg
PADIR    equ    04h             ;Port A data direction reg
PBDIR    equ    05h             ;Port B data direction reg
PCDIR    equ    06h             ;Port C data direction reg
CMDRAD   equ    07h             ;Mode definition reg
PACADD   equ    08h             ;Port A bit clear reg
PBCADD   equ    09h             ;Port B bit clear reg
PCCADD   equ    0ah             ;Port C bit clear reg
PASADD   equ    0ch             ;Port A bit set reg
PBSADD   equ    0dh             ;Port B bit set reg
PCSADD   equ    0eh             ;Port C bit set reg
T0LSB    equ    10h             ;Timer 0 lsb
T0MSB    equ    11h             ;Timer 0 msb
T1LSB    equ    12h             ;Timer 1 lsb
T1MSB    equ    13h             ;Timer 1 msb
T0STOP   equ    14h             ;Timer 0 stop
T0STRT   equ    15h             ;Timer 0 start
T1STOP   equ    16h             ;Timer 1 stop
T1STRT   equ    17h             ;Timer 1 start
T0CMD    equ    18h             ;Timer 0 command reg
T1CMD    equ    19h             ;Timer 1 command reg
```

```
;--- COMMAND EQUATES ---
;                        - PORT A DIRECTION

PA0      equ     INP             ;
PA1      equ     OUTP SHL 1      ;Seconds pulse out
PA1D     equ     LO SHL 1        ;Normal low
PA2      equ     OUTP SHL 2      ;Minute pulse out
PA2D     equ     LO SHL 2        ;Normal low
PA3      equ     INP SHL 3       ;
PA4      equ     INP SHL 4
PA5      equ     INP SHL 5
PA6      equ     INP SHL 6       ;Satellite clock in
PA7      equ     INP SHL 7
ADIR     equ     PA0 OR PA1 OR PA2 OR PA3 OR PA4 OR PA5 OR PA6 OR PA7
                                 ;Port A direction
ADAT     equ     PA1D OR PA2D    ;Port A data

;                        - PORT B DIRECTION

PB0      equ     OUTP            ;Serial out, inverted
PB0D     equ     HI              ;Space
PB1      equ     OUTP SHL 1      ;Formerly used for reel to reel
PB1D     equ     HI SHL 1        ;Normal high
PB2      equ     OUTP SHL 2      ;Not cartridge reset
PB2D     equ     LO SHL 2        ;Reset cartridge
PB3      equ     INP SHL 3       ;Time sync in (must be connected)
PB4      equ     INP SHL 4       ;From not INTR
PB5      equ     INP SHL 5       ;From not RSTA
PB6      equ     INP SHL 6       ;From TO out
PB7      equ     INP SHL 7       ;Serial in ,inverted (also
                                 ;connected to NRSTC)

BDIR     equ     PB0 OR PB1 OR PB2 OR PB3 OR PB4 OR PB5 OR PB6 OR PB7
                                         ;Port B direction
BDAT     equ     PB0D OR PB1D OR PB2D    ;Port B data

;                        - PORT C DIRECTION

PC0      equ     OUTP            ;Not RSTA disable
PC0D     equ     HI              ;Disable RSTA initially (will change)
PC1      equ     OUTP SHL 1      ;Disable power save
PC1D     equ     LO SHL 1        ;Disable power save intially
PC2      equ     OUTP SHL 2      ;Remap memory addressing
PC2D     equ     HI SHL 2        ;Don´t remap
PC3      equ     INP SHL 3       ;Timer gate, connected to serial
                                 ;in and NRSTC
PC4      equ     INP SHL 4       ;Timer 1 input, connected to 4040
PC5      equ     OUTP SHL 5      ;Timer 1 output, connected to
                                 ;NRSTB and NAND
PC5D     equ     HI SHL 5        ;Normal high
CDIR     equ     PC0 OR  PC1 OR PC2 OR PC3 OR PC4 OR PC5
                                 ;Port C direction
CDAT     equ     PC0D OR PC1D OR PC2D OR PC5D
                                 ;Port C data
```

```
;                           - NSC 810 MODES

MODE0     equ     00h             ;Port A basic IO
MODE1     equ     01h             ;Strobed input mode (affects A and C)
MODE2     equ     03h             ;Strobed output mode
MODE3     equ     07h             ;Tristate strobed output

;                           - TIMER MODES

TMODE0    equ     00h             ;Kill timer
TMODE1    equ     01h             ;Event counter mode
TMODE2    equ     02h             ;Stopwatch event timer
TMODE3    equ     03h             ;Event timer with reset
TMODE4    equ     04h             ;One shot
TMODE5    equ     05h             ;Square wave
TMODE6    equ     06h             ;Pulse generator

;                           - TIMER PRESCALER

PRE1      equ     00h             ;No prescale
PRE2      equ     08h             ;Divide by 2
PRE64     equ     018h            ;Divide by 64,timer 0 only
                                  ;(SEE NSC810 SPECS)

;                           - TIMER READ/WRITE MODE

BIT8T     equ     020h            ;Single byte read/write mode (Hi or LO)
BIT16T    equ     00h             ;16 bit timer. Read or write low
                                  ;byte first

;                           - TIMER GATE CONTROL

GPOLH     equ     00H             ;Gate input active high(PC3).
                                  ;This is the common gate for both timers
GPOLL     equ     040h            ;Gate input active low

;                           - TIMER OUTPUT CONTROL

OUTPOLH   equ     080h            ;Timer output active high.
                                  ;T1=PC5,T0=PIN 6
OUTPOLL   equ     00h             ;Timer output active low
```

```
; MEMORY EQUATES
;-----------------------------------------------------------------

;---- REAL TIME CLOCK (2000 - 2017h) ----

TMEM     equu    2000h           ;RTC base address
```

```
TMEM     equ     2000h           ;RTC base address
MSEC     equ     TMEM            ;Thousands of seconds
SEC.1    equ     TMEM+01h        ;Hundreds & Tenths of seconds
SEC      equ     TMEM+02h        ;Seconds
MIN      equ     TMEM+03h        ;Minutes
HRS      equ     TMEM+04h        ;Hours
DAYWK    equ     TMEM+05h        ;Day of week
DAYS     equ     TMEM+06h        ;Day
MTH      equ     TMEM+07h        ;Month
LMSEC    equ     TMEM+08h        ;Msec. latch
LSEC.1   equ     TMEM+09h        ;Hundreds & tenths latch
LSEC     equ     TMEM+0ah        ;Seconds latch
LMIN     equ     TMEM+0bh        ;Minutes latch
LHRS     equ     TMEM+0ch        ;Hours latch
LDAYWK   equ     TMEM+0dh        ;Day of week latch
LDAYS    equ     TMEM+0eh        ;Days latch
LMTH     equ     TMEM+0fh        ;Months latch
INTSTAT  equ     TMEM+10h        ;Clock interrupt status reg.
INTCMD   equ     TMEM+11h        ;Interrupt command reg.
CTRRST   equ     TMEM+12h        ;Counter reset
LTHRST   equ     TMEM+13h        ;Latch reset
CLKSTAT  equ     TMEM+14h        ;Status
STPST    equ     TMEM+15h        ;Start reg
INTREG   equ     TMEM+16h        ;Standby interrupt reg.
CTEST    equ     TMEM+17h        ;Test only

;--- NSC810 RAM (3000H - 307FH) ---

;                         - SERIES/EXPERIMENT VARIABLES

NSCRAM   equ     3000h           ;NSC-810 ram area
IO       equ     NSCRAM          ;Base IO for AD
NCX2     equ     NSCRAM+01h      ;# samples _ 2
SERTYP   equ     NSCRAM+02h      ;Series type
EXPN     equ     NSCRAM+03h      ;# Experiments in series
STRTTB   equ     NSCRAM+05h      ;Start time for series
STOPTB   equ     NSCRAM+0ah      ;Stop time for series
BUFSIZ   equ     NSCRAM+0fh      ;Buffer size in 8K blocks
PESAMPS  equ     NSCRAM+010h     ;# post event samples
BSTART   equ     NSCRAM+012h     ;High order buffer start address
MSAMPS   equ     NSCRAM+013h     ;# samples (max)
OFFSET   equ     NSCRAM+015h     ;Window offset
PERIOD   equ     NSCRAM+016h     ;Window period
ADVAL    equ     NSCRAM+017h     ;Sample rate code
ANVAL    equ     NSCRAM+018h     ;STA & THRSH code
ENDPARA  equ     NSCRAM+019h     ;End of series parameters
NOPARA   equ     LOW(ENDPARA-NSCRAM)
                                 ;Amount of parameter storage
```

```
;                            - WORKING PARAMETERS

PESAMS    equ      NSCRAM+019h       ;Post event verify
THRSHSV   equ      NSCRAM+01ah       ;Threshold verify
STASAV    equ      NSCRAM+01bh       ;Short term average verify
SRATE     equ      NSCRAM+01ch       ;Sample rate verify
BSZSAV    equ      NSCRAM+01dh       ;Buffer size verify
PARBUF    equ      NSCRAM+01eh       ;Storage for DE during parameter entry
LSTCP     equ      NSCRAM+020h       ;Last valid comp. for compare routine
HDRBUF    equ      NSCRAM+022h       ;Pointer to ASCII header
GPCTRL    equ      NSCRAM+024h       ;GP counter low byte
GPCTRH    equ      NSCRAM+025h       ;GP counter high byte
NSAMPS    equ      NSCRAM+026h       ;# samples working
BUFPTR    equ      NSCRAM+028h       ;Data aquisition buffer pointer
WBUFSAV   equ      NSCRAM+02ah       ;Storage of HL during write to tape
WBSTART   equ      NSCRAM+02ch       ;High order buffer start address
                                     ;for write
RDBUF     equ      NSCRAM+02dh       ;Buffer storage for read pointer
TIMSAVE   equ      NSCRAM+02fh       ;10 bytes to save time in case
                                     ;of entry error
STACK     equ      NSCRAM+050h       ;Run time stack
RAMST     equ      NSCRAM+05ah       ;Start of ram to test
ENDRAM    equ      NSCRAM+05ch       ;End of ram to be tested
ADDRESS   equ      NSCRAM+05eh       ;Storage for bad ram address
PATRN     equ      NSCRAM+05fh       ;Storage of bad ram pattern
RECN      equ      NSCRAM+062h       ;Record number storage
RWN       equ      NSCRAM+065h       ;Rewrite storage
ABRTV     equ      NSCRAM+067H       ;Abort vector storage
LSN       equ      NSCRAM+069h       ;2 byte, last series +1
EQFY      equ      NSCRAM+06bh       ;Event qualify counter
BEND      equ      NSCRAM+06ch       ;Storage of end of data buffer

;--- SCRATCH PAD RAM (4000 - 5FFFh) ---

SCRATCH   equ      4000h             ;Scratch pad ram
NSTACK    equ      SCRATCH+200h      ;Operating stack after ramtest
TSTORE    equ      NSTACK+02h        ;Temporary storage
TSTFLG    equ      NSTACK+04h        ;Flag for ram test
RECTIME   equ      NSTACK+06h        ;Length of recording (mins,secs)
SHIFTER   equ      NSTACK+08h        ;Used to divideto get above value
ESECT     equ      NSTACK+0ah        ;Save sector count for write error
EBUFSAV   equ      NSTACK+0bh        ;Save buffer ptr for write error
T1INT     equ      NSTACK+0dh        ;T1 interupt vector
DS        equ      NSTACK+0fh        ;Stored drive status
IS        equ      NSTACK+010h       ;Stored interface status
IMA       equ      NSTACK+011h       ;Mode argument copy
IPA       equ      NSTACK+012h       ;Pos. arg copy
ICA       equ      NSTACK+013h       ;Command argument copy
GCODE     equ      NSTACK+014h       ;Analog board gain code
SRCODE    equ      NSTACK+015h       ;Sample rate value for A-D board
ELAPSED_MIN
          equ      SCRATCH+300h      ;Elapsed minute count
BASAD     equ      SCRATCH+350h      ;Base address
IBUFF     equ      SCRATCH+400h      ;ASCII input buffer
NCI       equ      IBUFF+MAXB+1      ;No. characters in buffer
```

```
SERBUF    equ     5000h            ;Beginning of series parameter storage
ESERBUF   equ     50D8h            ;End of series parameter storage
SERPTR    equ     ESERBUF+01h      ;Pointer to next series parameters
CSN       equ     ESERBUF+03h      ;Current series number in BCD
CEXPN     equ     ESERBUF+05h      ;Current exp. no. in BCD
SAMPTIM   equ     ESERBUF+07h      ;Time of beginning of record
                                   ;(5 BYTES ALLOWED)
CSEC1     equ     SAMPTIM          ;Tenths of seconds
CSEC      equ     SAMPTIM+01h      ;Seconds
C10SEC    equ     SAMPTIM+02h      ;Tens of seconds
CMIN      equ     SAMPTIM+03h      ;Unit minutes
C10MIN    equ     SAMPTIM+04h      ;Tens of minutes
CHRS      equ     SAMPTIM+05h      ;Unit hours
C10HRS    equ     SAMPTIM+06h      ;Tens of hrs.
CDAYS     equ     SAMPTIM+07h      ;Unit days
C10DAY    equ     SAMPTIM+08h      ;Tens of Days
CDAYWK    equ     SAMPTIM+09h      ;Day of week
CMTH      equ     SAMPTIM+0ah      ;Unit months
CTENMTH   equ     SAMPTIM+0bh      ;Tens of months
CYR       equ     SAMPTIM+0ch      ;Year(PACKED BCD)
CTIMER    equ     SAMPTIM+0dh      ;Millisec. (LO BYTE FIRST,2 BYTES)
CTIMLO    equ     CTIMER           ;Low byte (millisec)
CTIMHI    equ     SAMPTIM+0eh      ;High byte (tenths and hundredths)
DATABLOCK equ     ESERBUF+016h     ;No. of sectors to write (128bytes)
EEXPBUF   equ     ESERBUF+014h     ;End of tape data if 0ffh

;                         - MEMORY IMAGE OF CLOCK IO

MTMEM     equ     5200h            ;RTC base address
MMSEC     equ     MTMEM            ;Milliseconds
MSEC.1    equ     MTMEM+01h        ;Tenths of seconds
MSEC      equ     MTMEM+02h        ;Seconds
MMIN      equ     MTMEM+03h        ;Minutes
MHRS      equ     MTMEM+04h        ;Hours
MDAYWK    equ     MTMEM+05h        ;Day of week
MDAYS     equ     MTMEM+06h        ;Day
MMTH      equ     MTMEM+07h        ;Month
MYRS      equ     MTMEM+08h        ;Yr storage
OTENMTH   equ     MTMEM+09h        ;Old month storage
TIMEN     equ     MTMEM+0bh        ;5 bytes for clock in PBCD

;                         - TIP TAPE BUFFER

BUF16     equ     5300h            ;Tape header buffer for TIP
B16NAME   equ     BUF16+01h        ;Header name
BSERNO    equ     BUF16+02h        ;Series number
BEXPNH    equ     BUF16+07h        ;High byte of exp. no.
BEXPNL    equ     BUF16+09h        ;Low byte of exp. no.
B16LB     equ     BUF16+0dh        ;Last block flag
B16RC     equ     BUF16+0fh        ;Record size
TIPBUF    equ     BUF16+10h        ;TIP buffer read location
```

```
;--- AQUISITION RAM (8000 - FFFFh)---

BUFMEM   equ     08000h          ;Aquistion ram

;                        - HEADER RAM

HDRRAM   equ     0e000h          ;Start of header ram
```

```
; CARTRIDGE CONTROLLER BOARD EQUATES
;________________________________________________________________

;--- PORT LOCATIONS ---

CPORT    equ     0f0h            ;Cartridge controller base
MA       equ     CPORT           ;Mode argument
PA       equ     CPORT+1         ;Position argument
CA       equ     CPORT+2         ;Command argument
DA       equ     CPORT+3         ;Data argument(write data only)
PS       equ     CPORT+1         ;Port status

;--- COMMAND EQUATES ---

READC    equ     01h             ;Read a record
WRITEC   equ     02h             ;Write a record
WRITEFMC
         equ     03h             ;Write a file mark
FWDSPRECC
         equ     04h             ;Foward space a record
FWDSPFILC
         equ     05h             ;Forward space a file
REVSPCRECC
         equ     06h             ;Reverse space a record
REVSPCFILC
         equ     07h             ;Reverse space a file
CURSTATC
         equ     08h             ;send current status command to
                                ;tape
SETRECLENC
         equ     09h             ;Set record length
WRITEWCHC
         equ     0ah             ;Write a record with check
RECSERMASC
         equ     0bh             ;Mask search
REWINDC  equ     40h             ;Rewind
BREADC   equ     41h             ;Buffer read
BWRITEC  equ     42h             ;Buffer write
BWRITEFMC
         equ     43h             ;Buffer file mark
RWREADC  equ     81h             ;Ram read
RSTCC    equ     04h             ;Cartridge controller reset line
                                 ;on port B
                                 ;clear to reset, set to allow operation

;--- MASKS ---

BOTM     equ     08h             ;BOT mask for DS
COMSTATM
         equ     030h            ;Command status mask for IS check
EOTM     equ     04h             ;End of track mask
MAN      equ     060h            ;Status mask
TRACKMA  equ     03h             ;Track mask
MSMA     equ     0e0h            ;Mask search MA excl. track bits
```

SYSTEM MACROS

```
;______________________________________________________________
;Original code written by jhg
;UPDATE -  5/ 6/84 PRINT macro was added by ohh
;UPDATE - 10/18/84 code compiled into a seperate module by gkm
;UPDATE -  7/30/85 QPRINT macro was added by gkm
;______________________________________________________________
```

```
;--- SAVE ALL REGISTERS ---

PUSHALL  MACRO
 push    af
 push    bc
 push    de
 push    hl
 endm

;--- RESTORE ALL REGISTERS ---

POPALL   MACRO
 pop     hl
 pop     de
 pop     bc
 pop     af
 endm

;--- IMMEDIATE STORE ---
;                        - ALLOWS THE DIRECT STORAGE OF ANY
;                          REGISTER OR & IMMEDIATE VALUE IN A
;                          SPECIFIED MEMORY LOCATION.
;                        - NO REGISTERS AFFECTED.
;                        - 6 BYTES FOR REGISTER,38 TSTATES
;                        - 7 BYTES FOR IMMEDIATE,41 TSTATES

STA      MACRO   ADDR,ARG
 push    af
 ld      a,ARG
 ld      ADDR,a
 pop     af
 endm
```

```
;--- DELAY n MSEC ---
;                          - THIS MACRO DELAYS N HUNDRED T-STATES
;                          - MACRO USES 45 TSTATES.
;                          - DELAYR WASTES 55 TSTATES AND CALLS A
;                            100 T-STATE
;                          - DELAY N-1 TIMES FOR A TOTAL DELAY OF N
;                            HUNDRED TSTATES.
;                          - MAX. DELAY FOR THIS MACRO IS 25.6 MS.
;                          - DELAY TIMES ARE BASED ON T-STATE=1
;                            MICRO-SEC WHICH IS ONLY APPROXIMATE.
;                          - APPROXIMATION IS NOT USED FOR CRITICAL
;                            DELAYS

DELAY    MACRO   NHUNDRED
 push    bc                ;11T--SAVE B REGISTER
 ld      b,NHUNDRED-1      ; 7T--SET UP B FOR DJNZ IN DELAYR
 call    DELAYR            ;17T--CALL DELAY SUBROUTINE
 pop     bc                ;10T--RESTORE B AND CONTINUE
 endm

;--- NORMAL PRINT ROUTINE ---

;                          - STRING MUST BE TERMINATED WITH A 0H

PRINT    MACRO   STRING
 push    hl                ;Save regs.
 ld      hl,STRING         ;Usage:    PRINT DIAGMS
 call    SNDMES            ;DIAGMS:   DB 'MESSAGE',0
 pop     hl                ;Restore regs.
 endm
```

```
;--- QUESTION AND ANSWER MACRO #1 ---

;                          - PRINTS STRING WITH A ?
;                          - ANSWER IS MOVED TO THE HEADER BUFFER

WPRINT  MACRO   STRING
 push   af                 ;Save acc.
 exx                       ;Save regs.
 ld     hl,STRING          ;Point to Prompt
 call   WPRA               ;Save prompt
 ld     hl,STRING          ;Get answer
 call   WPRB               ;And save it
 exx                       ;Restore regs.
 pop    af                 ;Restore acc.
 ENDM

;--- QUESTION AND ANSWER MACRO #2 ---

;                          - PRINTS STRING WITH A ?
;                          - ANSWER IS IN REG. A

QPRINT  MACRO   STRING
 push   hl                 ;Save HL
 ld     hl,STRING          ;Point to Prompt
 call   WPRA               ;Print it
 call   GBYTNE             ;Get answer
 pop    hl                 ;Restore HL
 ENDM
;
```

PAGE ZERO, INTERRUPT, AND RST VECTORS

```
;_______________________________________________________________
; Original code written by jhg
; UPDATE -  5/ 6/84 code commented by ohh
; UPDATE - 10/18/84 code converted to module format
;                   and commented by gkm
; UPDATE -  9/12/85 code was revised to use Tl as a
;                   timeout trap for tape routines by gkm
;_______________________________________________________________
```

```
;--- STARTING POINT ---

        ASEG
        org  0000h

;                         - PROCESSOR STARTS HERE ON RESET.
;                         - INTERRUPT AND REFRESH REGS ARE CLEARED.
;                         - ALL INTERRUPTS ARE DISABLED(HARDWARE DI
;                           INSTRUCTION).
;                         - INTERRUPT CONTROL REGISTER IS SET TO 01,
;                           WHICH ENABLES "NOT INTR" AND MASKS OFF
;                           "NOT" RSTA,RSTB,RSTC.
;                         - NO INTERRUPT CAN WORK UNLESS BOTH AN EI
;                           INSTRUCTION HAS BEEN ISSUED AND THERE IS
;                           A ONE IN THE APPROPRIATE SPOT IN THE
;                           INTERRUPT MASK REGISTER,WHICH IS A WRITE
;                           ONLY REGISTER ADDRESSED AS AN OUTPUT PORT
;                           (LOWER 4 BITS ONLY) BY AN OUT BBH OR
;                           EQUIV. INSTRUCTION.
;                         - THE FOLLOWING VALUES ENABLE THE
;                           CORRESPONDING INT. LINES:
;                                 08H            ENABLES RSTA
;                                 04H            ENABLES RSTB
;                                 02H            ENABLES RSTC
;                                 01H            ENABLES INTR
;                         - NOTE THAT 0FH ENABLES ALL INT. LINES
;                         - 8080 INTERRUPT MODE IS AUTOMATICALLY
;                           SELECTED ON RESET

;---RESTART 0 (11000111)---

BEGIN:  di                      ; Disable interrupts
        jp      MAIN            ; Start program

;                         - END OF RESET CODE

;                         - THESE ARE THE 8 NORMAL Z80 RESTART
;                           LOCATIONS.
;                         - IN 8080 MODE ,THESE CONSTITUTE A MEANS
;                           FOR EXTERNAL DEVICES TO FORCE A CALL TO
;                           ONE OF 8 LOCATIONS WITH ONE INSTRUCTION
;                           PUT ON THE BUS DURING AN INTACT CYCLE.
;                         - ALSO CAN BE USED AS A VERY SHORT CALL
;                           VIA RST INSTRUCTION.
;                         - NOTE THAT A DI INSTRUCTION IS
;                           AUTOMATICALLY EXECUTED.
;                         - THE INTERRUPTS NOT CURRENTLY USED BY
;                           THE OBS INCLUDE RETURNS FOR SAFETY.
```

```
;--- RESTART 1 (11001111) ---

        aseg
        org     0008h
        ret

;--- RESTART 2 (11010111) ---

        aseg
        org     0010h
        ret

;--- RESTART 3 (11011111) ---

        aseg
        org     0018h
        ret

   ;--- RESTART 4 (11100111) ---

        aseg
        org     0020h
        ret

;--- RESTART 5 (11101111) ---

        aseg
        org     0028h
        ret

;--- RESTART C ---

        aseg
        org     002Ch
        ret

;--- RESTART 6 (11110111) ---

        aseg
        org     0030h
        ret
```

```
;---RESTART B ---

;                         - THIS INTERRUPT IS FROM TIMER 1.  IT IS
;                           USED AS A TIME OUT ERROR TRAP.

        aseg
        org     0034h
        jp      TFAIL           ;Tape fail routine

;--- RESTART 7 (11111111) ---

;                         - IN MODE 1,WILL COME HERE ON NOT INTR
;                           GOING LOW
;                         - PUSHING THE EARLY TERMINATION BUTTON
;                           BRINGS YOU HERE.

        aseg
        org     0038h
        jp      ETERM              ;Early termination routine

;                         - NOTE THAT THIS REQUIRES HARDWARE OR
;                           MODE1 INTERRUPTS!
; NSC800 SPECIAL INTERRUPT LOCATIONS
;_______________________________________________________________

;                                 - IF INT. ENABLED AND MASK OK WILL COME
;                                   HERE AS SHOWN.
;                                 - NO INSTRUCTION NEEDED ON BUS-- JUST
;                                   PULL THE APPROPRIATE PIN LOW.

;--- RESTART A ---

;                                 - INTERRUPT IS GENERATED ON THE A/D BOARD.
;                                 - IT IS CONTROLLED BY A NUMBER OF CPU
;                                   BOARD OUTPUTS.
;                                 - THE NSC 810 PORT C PIN 0 CONTROLS AN
;                                   OR GATE WHICH DISABLES THE INTERRUPT
;                                   COMING FROM THE A/D BOARD ON S-100 PIN 4
;                                   IF C-0 IS HIGH.
;                                 - THE A/D BOARD WILL NOT GENERATE AN
;                                   INTERRUPT IF ITS FIRST INTERRUPT WAS NOT
;                                   ACKNOWLEDGED.
;                                 - THE A/D INTERRUPT OCCURS IMMEDIATELY
;                                   AFTER CONVERSION COMPLETE FOLLOWING A
;                                   TIME-OUT OF THE SOFTWARE SETTABLE COUNTER
;                                   WHICH GIVES THE APPROXIMATE SAMPLE(WILL
;                                   HAVE ABOUT 200 MICROSECONDS OF SLOP) TIME

        aseg
        org     003Ch
AQUINT: ex      af,af'          ;Save all reg. in alternate set
        exx
        ld      hl,(BUFPTR)     ;Get pointer for acquisition data
        ld      bc,(NSCRAM)     ;Set up B&C reg. for block input

;                                 - C REGISTER IS THE IO PORT BASE
;                                   ADDRESS-1: B REGISTER IS THE COUNTER
;                                 - FETCH DATA FROM IO 18 THROUGH 18 +2
;                                   TIMES Number Channels--NCX2 BYTES TOTAL

        ld      a,(BSTART)      ;Get high order buffer start address
        ld      d,a             ;Save the start address to keep
                                ;the circular buffer wrap in the
                                ;correct area of ram
        or      h               ;Set HI bit to allow wrap corr.
        ld      h,a             ;back to H
AQLP:   inc     c               ;IO port = port + 1(FOR LOOP)
        ini                     ;Move data from AD to buffer
```

```
;                         - FETCH FROM IO PORT (C),STORE AT HL,
;                           DECR. B, INCR. HL .

         jr      z,ALLIN         ;Do until B=0 (NCX2 TIMES)
         ld      a,h             ;If FFFF increments to 0000
         or      d               ;Bring address back to data
         ld      h,a             ;Back to H
         jr      AQLP

;                         - TO MAINTAIN CIRCULAR 32K BUFFER,MAKE
;                           SURE HL ALWAYS HAS HIGH BIT SET.
;                         - THUS FFFF INCREMENTS TO 0000,BUT THIS
;                           SETS IT BACK TO 8000H.
;                         - SAME PRINCIPLE FOR 8K OR 16K BUFFER.
;                           (NOTE 24K IS NOT POSSIBLE).

ALLIN:   ld      (BUFPTR),hl     ;Save buffer pointer

;                         - NOW KEEP TRACK OF THE NUMBER OF
;                           SAMPLES:NOTE THAT EACH SAMPLE USES NCX2
;                           RAM LOCATIONS.

         ld      hl,(NSAMPS)     ;Since you just did a sample,
         dec     hl              ;Decrement the sample count
         ld      (NSAMPS),hl     ;and store it

;                         - NOW DECREMENT EVENT QUALIFY COUNTER

         ld      a,(EQFY)        ;Get event qualifier
         dec     a               ;decrement it
         ld      (EQFY),a        ;Save new count

         exx                     ;Restore original regs.
         ex      af,af'
         ei                      ;Enable interrupts
         ret
```

MAIN PROGRAM

```
;_______________________________________________________________
;Original code written by jhg
;UPDATE -  5/ 6/84 code commented by ohh
;UPDATE - 10/18/84 code converted to module format and
;         commented by gkm
;UPDATE -  7/30/85 code was revised for the following
;         by gkm:
;               1. T0 and T1 timer routines were
;                 eliminated as these caused
;                 some time errors.
;               2. The ram test was replaced
;                 with a simpler, faster test
;                 which is now done at power up.
;               3. End of program routine was
;                 altered to obtain second
;                  pulses from the RTC rather
;                 than T0.
;UPDATE -  9/13/85 code was revised to add an error
;         trap for tape routine at program termination to
;         allow time recovery if tape fails by gkm
;UPDATE -  2/24/86 code for clock sync was rewritten to
;         accommodate the new clock (58167) by gkm
;_______________________________________________________________
```

```
;---- INITIALIZE OPERATING PARAMETERS ----

MAIN:    im      1               ;Set mode interrupt 1
         xor     a               ;Clear accum.
         out     (INTPRT),a      ;Disable interrupts
         ld      ix,NSCIOT       ;Sets up the IX register to point
                                 ;to the NSC-810

;--- SET-UP STACK ---

         ld      sp,STACK        ;Set stack top -DATA PUSHED ON
                                 ;STACK FROM 304FH DOWN TO 3000H
                                 ;IN NSC 810 RAM

;--- INITIALIZE NSC810 ---

;                       - THE FOLLOWING CODE INITIALIZES THE
;                         NSC 810.
;                       - AFTER RESET THE 810 IS IN THE FOLLOWING
;                         CONDITION:
;                       1.ALL INTERNAL REGISTERS ARE ZEROED
;                       2.ALL COUNTER/TIMERS ARE STOPPED AND RESET.
;                       3.ALL IO PORTS GO TO HIGH Z INPUT MODE
;                       4.THE RAM IS LEFT UNCHANGED

INI810:  ld      (IX+PADADD),ADAT        ;Intialize port A data
         ld      (IX+PBDADD),BDAT        ;Init. port B data
         ld      (IX+PCDADD),CDAT        ;Init. port C data
         ld      (IX+PADIR),ADIR         ;Init. port A direction
         ld      (IX+PBDIR),BDIR         ;Init. port B direction
         ld      (IX+PCDIR),CDIR         ;Init. port C direction
         ld      (IX+CMDRAD),MODE0       ;Init. NSC-810 mode

T1DAT    equ     TMODE1 or PRE1 or BIT16T or GPOLH or OUTPOLL
         ld      (IX+T1CMD),T1DAT        ;Init. timer 1

;--- LOAD TIMER ---

         ld      (STOPT1),a              ;Stop timer 1
         ld      (IX+T1LSB),0ffh         ;Load low byte
         ld      (IX+T1MSB),06h          ;Load high byte = 6 sec.

;--- INITIALIZE EVENT DETECTOR ---

         ld      a,1                     ;Set average high and
         out     (AVGPRT),a              ;Clear pending interrupts
```

```
;--- INITIALIZE A/D BOARD ---

;                           -THE A/D BOARD CONSISTS OF 4 CHANNELS OF
;                            QUASI 16 BIT A/D CONVERSION
;                           -THIS IS ACCOMPLISHED BY A 12 BIT A/D
;                            USING 4 BITS OF D/A AS A GAIN RANGING
;                            DEVICE.
;                           -THESE 4 BITS OF D/A MAY BE SET BY
;                            SOFTWARE, BUT,IF SO,THEY ARE SENT TO ALL
;                            FOUR CHANNELS, WHILE THE HARDWARE GAIN
;                            SETTING OPTION WORKS ON EACH CHANNEL
;                            INDIVIDUALLY.

;---- FORCE A-D GAIN TO ZERO ---

         ld      a,0dh           ;Select 128 Hz and force
         out     (ADPORT),a      ;Manual gain of zero
         ld      a,0             ;Select 128 Hz and
         out     (ADPORT),a      ;Gain ranging mode

;--- INITIALIZE POWER BOARD ---

PIBINIT:call     OFFCART         ;Kill cartridge power

;--- WAIT FOR TERMINAL TO TURN ON ---

;                           -BAUD RATE MUST BE 1200

TERMWAI:ld       a,(PBDATA)      ;Look for terminal connect
        rla
        jp       nc,TERMWAI
```

```
; RAMTEST
;______________________________________________________________

;--- TEST NSCRAM ---

;                       - THIS CHECKS THE RAM LOCATIONS FROM
;                         3000H TO 3080H.
;                       - THE TEST FILLS THE AREA OF RAM AND THEN
;                         RECREATES THE TEST PATTERN TO CHECK THE
;                         VALUE LOADED IN EACH LOCATION.

R810:    PRINT   CLRSCR
         ld      hl,NSCRAM       ;Point to the start of the NSCRAM
         xor     a               ;0 is the first pattern loaded
         ex      af,af´          ;Save acc. in preparation

;                        - FILL THE RAM WITH THE PATTERN

RSFILL:  ex      af,af´          ;Restore pattern
         ld      (hl),a          ;Store the pattern
         inc     hl              ;Point to the next ram location
         inc     a               ;Increment the pattern loaded
         ex      af,af´          ;Save the pattern
         ld      a,l             ;Get low byte of ram
         cp      80h             ;Check if end of ram
         jr      nz,RSFILL       ;If not, store the pattern and
                                 ;continue

;                        - TEST THE RAM

         ld      hl,NSCRAM       ;Restore ram start
         xor     a               ;Create start pattern
         ex      af,af´          ;Save the pattern inpreperation
RSCHEK:  ex      af,af´          ;Restore pattern
         cp      (hl)            ;Compare the pattern
         jp      nz,BADRAM       ;If no compare then bad
         inc     hl              ;Point to the next ram location
         inc     a               ;Recreate the pattern
         ex      af,af´          ;Save pattern
         ld      a,l             ;Get low byte of address
         cp      80h             ;Check if end of ram
         jr      nz,RSCHEK       ;If not, do next
         ex      af,af´
         PRINT   OKNSC           ;Print OK message
```

```
;--- TEST SCRATCHPAD RAM ---

;                        - THIS CHECKS THE RAM LOCATIONS FROM
;                          4000H TO 6000H.
;                        - THIS RAM IS CLEARED AT COMPLETION.

         ld      hl,4000h        ;Point to scratch pad ram
         ld      (RAMST),hl      ;Save it
         ld      bc,60h          ;BC is end of ram to test
         ld      (ENDRAM),hl     ;Save it
         PRINT   TSCR
         call    RTEST
         PRINT   OKRAM           ;Print OK message
         ld      hl,4000h        ;Point to scratch pad ram
         call    CLR2K           ;Clear it

;--- TEST DATA BUFFER ---
;                      - THIS TESTS THE RAM FROM LOCATIONS
;                        8000H TO FFFFH.
;                      - THE AREA DESIGNATED FOR THE HEADER
;                        INFO IS CLEARED.

         PRINT   TSTB?
         ld      hl,8000h        ;Point to acquisition ram
         ld      (RAMST),hl      ;Save it
         ld      bc,0h           ;Load end point
         ld      (ENDRAM),hl     ;Save it
         call    RTEST           ;Test it
         PRINT   OKRAM           ;Print OK message
         call    INIT_HDR_RAM    ;Initialize header ram

;--- JUMP TO USER PORTION OF PROGRAM ---

         jp      START
```

```
; RAM TEST SUBROUTINES
;_______________________________________________________________

;--- TEST THE AREA OF RAM ---

RTEST:  ld      hl,(RAMST)      ;Get ram start
        ld      bc,(ENDRAM)     ;Get ram end
        xor     a               ;0 is the first pattern loaded
        ex      af,af´          ;Save acc. in preparation

;                       - FILL THE RAM WITH THE PATTERN

RFILL:  ex      af,af´          ;Restore pattern
        ld      (hl),a          ;Store the pattern
        inc     hl              ;Point to the next ram location
        inc     a               ;Increment the pattern loaded
        ex      af,af´          ;Save the pattern
        ld      a,h             ;Get high byte of ram
        cp      c               ;Check if end of ram
        jr      nz,RFILL        ;If not, store the pattern and
                                ;continue

;                       - TEST THE RAM

        ld      hl,(RAMST)      ;Restore ram start
        ld      bc,(ENDRAM)     ;Restore ram end
        xor     a               ;Create start pattern
        ex      af,af´          ;Save the pattern in preparation
RCHEK:  ex      af,af´          ;Restore pattern
        cp      (hl)            ;Compare the pattern
        jp      nz,BADRAM       ;If no compare then bad
        inc     hl              ;Point to the next ram location
        inc     a               ;Recreate the pattern
        ex      af,af´          ;Save pattern
        ld      a,h             ;Get high byte of address
        cp      c               ;Check if end of ram
        jr      nz,RCHEK        ;If not, do next
        ex      af,af´
        ret

BADRAM: ld      (ADDRESS),hl    ;Save address of bad ram
        ld      (PATRN),a       ;Save pattern at bad ram
        call    RAMER           ;Go to display routine

;--- CLEAR HEADER RAM ---

INIT_HDR_RAM:
        ld      hl,HDRRAM       ;Point to start of header ram
        ld      (HDRBUF),hl     ;Save it
        call    CLR2K           ;Clear it
        ret
```

```
;--- CLEAR A 2K HEX BLOCK OF RAM ---

;                          - CALL WITH HL SET TO BEGINNING OF RAM
;                            TO CLEAR
;                          - NO REGISTERS ALTERED

CLR2K:   push    bc              ;Save regs.
         push    hl              ;Save regs.
         ld      b,64            ;Load B with no. 128 byte blocks
CLR2KL:  call    CLR128          ;Actual clear routine
         djnz    CLR2KL          ;Until done
         pop     hl              ;restore regs.
         pop     bc
         ret

;--- CLEAR 128 BYTES OF RAM ---

CLR128:  push    af              ;Save regs.
         push    bc              ;Save regs.
         xor     a               ;Zero A reg.
         ld      b,128           ;No. of bytes to clear
CLELOP:  ld      m,a             ;Load contents of A into Memory
         inc     hl              ;Increment to next memory
         djnz    CLELOP          ;Clear until done
         pop     bc              ;Restore regs.
         pop     af              ;Restore regs.
         ret

;--- RAM TEST ERROR ROUTINE ---

RAMER:   PRINT   RAMER1          ;Print message
         ld      hl,(ADDRESS)    ;Get bad address
         call    PHREG           ;Print address
         call    CRLF
         PRINT   RAMER2          ;Print next message
         ld      a,(PATRN)       ;Get bad pattern
         call    HEXOUT          ;Print it
         call    CRLF
         PRINT   RAMER3          ;Print next message
         call    GBYT            ;Wait for character
         cp      CR              ;Is it a carriage return?
         ret     z               ;If so normal exit
         HALT                    ;Else, stop
```

```
;--- PRINT RAM ADDRESS ---

PHREG:   ld      a,h             ;Get high two digits
         call    HEXOUT          ;Print them
         ld      a,l             ;Get low two digits
HEXOUT:  push    af              ;Save it
         rrca                    ;Put high nibble into bits 0-3
         rrca
         rrca
         rrca
         call    PCD             ;Print digit
         pop     af              ;Get low digit
PCD:     and     0fh
         add     a,90h
         daa
         adc     a,40h
         daa
         jp      PBYT            ;Print it
```

```
; PROGRAM CONTROL
;______________________________________________________________

;--- RECORD HEADER AFTER TERMINAL UNPLUGGED ---

;                            - IN EXECUTION YOU HAVE THE OPTION OF A
;                              DATA ACQUISITION MODE OR A TEST MODE.
;                            - THE DATA ACQUISITION MODE WILL ASK YOU
;                              TO DISCONNECT THE TERMINAL.  AFTER THE
;                              TERMINAL IS DISCONNECTED, THE HEADER WILL
;                              BE WRITTEN AND THE SYSTEM WILL GO TO
;                              WORK.  DOING THIS IS THE ONLY WAY TO
;                              DETERMINE IF THE SYSTEM IS WORKING AFTER
;                              EVERYTHING IS DISCONNECTED AND READY FOR
;                              DEPLOYMENT.
;                            - THE TEST MODE WILL WRITE THE HEADER
;                              IMMEDIATELY AND DISPLAY EXPERIMENT
                               PARAMETERS AND CURRENT EXPERIMENT NUMBER.

EXECU:  QPRINT  TSTM            ;Operator must indicate if this
                                ;is a test
        cp      'Y'             ;Is it yes
        jp      z,YTEST         ;If so go to test
        cp      'y'
        jp      z,YTEST
        PRINT   UNPLUGM         ;Print instruction
DISCON: ld      a,(PBDATA)      ;Look for terminal disconnect
        rla
        jr      c,DISCON
        ld      b,20            ;Load count
```

```
;                          - IF SLOW IN DISCONNECT, THE TERMINAL
;                            WILL BEEP AT YOU UNTIL YOU DO.
SLOWPOKE:
         PRINT   BEEP            ;Beep
         DELAY   256             ;every 25.6 ms
         ld      a,(PBDATA)      ;Check terminal for disconnect
         rla
         jr      c,SLOWPOKE      ;Continue until disconnect
         djnz    SLOWPOKE        ;Do 20 times to be sure
 YTEST:  xor     a               ;Start of write buffer is 00
         ld      (WBSTART),a     ;Save it (noncircular)
         STA     (DATABLOCK),40h-2
                                 ;Store number of write sectors
         ld      hl,HDRRAM       ;Point to header ram
         ld      (WBUFSAV),hl    ;Save it in case of trouble
         call    BLNKB16         ;Clear the TIP header
         ld      hl,HEADER       ;Point to the header ram
         call    MNAME           ;Loader the TIP header name
         call    ONCART          ;Turn on the cartridge
         call    WRITED          ;Write the GPheader
         call    OFFCART         ;Turn off the cartridge
         call    BLNKB16         ;Clear the TIP header
         ld      hl,BDTAHDR      ;Point to series header TIP name
         call    MNAME           ;Create it
         ld      hl,HDRRAM       ;Point to header ram
         call    CLR2K           ;Clear it
LTMR1:   jp      CONTROL         ;User now unnecessary
```

```
;--- PROGRAM TERMINATION ---

ETERM:  call    OFFCART
        PRINT   TERMIN          ;Print message
        call    CONT            ;Wait
        jp      ENDPRO          ;Go to clock sync routine


;--- CHECK CLOCK SYNC ---

;                       - THIS ROUTINE DISPLAYS THE TIME UPDATING
;                         EVERY SECOND AND SENDS OUT THE SECOND
;                         PULSE TO ALLOW COMPARISON WITH THE
;                         SATELLITE CLOCK.

ENDPRO: call    CRLF            ;Print CRLF
        PRINT   OVR             ;Print message
        ld      hl,CLKRD        ;Save abort vector
        call    ONCART
        call    FMK             ;Write first file mark
        call    FMK             ;Write second file mark
        call    OFFCART
CLKRD:  di                      ;Disable interrupts
CLKRD1: ld      a,(TMEM+2)      ;Look for second transition
        inc     a               ;Increment second
        daa                     ;Decimal adjust
        cp      60h             ;Check if minute
        jr      nz,CLKRD2       ;If not, continue
        xor     a               ;Set sec. to 0
CLKRD2: ld      e,a             ;Move it to E
EDPR1:  ld      a,(TMEM+2)      ;Loop until valid second
        cp      e               ;transition
        jr      nz,EDPR1

;                       - OUTPUT SECONDS TO PORT A

        ld      (IX+PASADD),2           ;Set port A of NSC810
        call    GCLOCK                  ;Read the clock
        ld      hl,TIMEN                ;Point to clock
        call    TIMOUT                  ;Print the time
        call    SECOUT                  ;Print seconds
        call    CRLF                    ;Print CRLF
        ld      (IX+PACADD),0FFH        ;Clear port A of NSC810
        jp      CLKRD1
```

```
; DEFINE STATEMENTS
;_______________________________________________________________

OKNSC:          db      'NSC RAM IS OK',CR,LF,0
OKRAM:          db      'THIS RAM OK',CR,LF,0
OVR:            db      'PROG OVR',CR,LF,0
RAMER1:         db      'BAD RAM @ HEX ADDRESS ',0
RAMER2:         db      'HEX PATTERN LOADED = ',0
RAMER3:         db      'HIT RETURN TO TEST NEXT RAM',CR,LF,0
TERMIN:         db      'TERMINATED',CR,LF,0
TSCR:           db      'TESTING SCRATCH PAD RAM',CR,LF,0
TSTB?:          db      'TESTING DATA BUFFER',CR,LF,0
TSTM:           db      CR,LF,'IS THIS A TEST? (Y/N)',0
UNPLUGM:        db      CR,LF,'UNPLUG TERMINAL FROM OBS',CR,LF
                db      'HEADER WILL BE WRITTEN ABOUT 5 SEC. LATER'
                db      0
BEEP:           db      7,0
```

CLOCK SUBROUTINES

```
;______________________________________________________________
;Original code written by jhg
;UPDATE -  8/ 8/84 code was revised to include use of
;                  NSC810 timers by ohh
;UPDATE - 10/18/84 code was modularized and commented
;                  by gkm
;UPDATE -  7/30/85 code was revised to eliminate use of
;                  the NSC810 timers T0 and T1 since
;                  they caused errors - by gkm.
;UPDATE -  2/24/86 code was rewritten to accommodate the
;                  new clock (58167) by gkm
;______________________________________________________________
```

```
;--- ENTER THE TIME ---

SETCLK:  PUSHALL                   ;Save reg.
         PRINT   SETM              ;Print set clock message
         ld      hl,TIMEN          ;Point to memory storage of time
         call    TIMINP            ;Get time
         ld      a,(TIMEN)         ;Get year
         ld      (MYRS),a          ;Save it
         ld      a,(TIMEN+1)       ;Get Month
         ld      (OTENMTH),a       ;Save it for year check
         call    LCM               ;Load the clock

;--- SYNCH WITH SATELLITE CLOCK ---

         ld      ix,NSCIOT         ;Set up for IO
SETCLK2: call    CONT              ;Pause
         PRINT   CLRSCR            ;Clear screen
         PRINT   CLKPRM            ;Print message
         ld      a,(IX+0)          ;Read port "A" BIT 6
         bit     6,a               ;Test bit 6
         jr      z,SCCON           ;Is sat. clock connected?
         PRINT   SATNC             ;NO! Not connected
         jr      CLK2              ;Skip connected message
SCCON:   PRINT   SATC              ;Print sat clk. conn.
CLK2:    call    GCLOCK            ;Print time of projected start
         ld      hl,TIMEN
         call    TIMOUT
         call    GBYT              ;Test for control B from
         cp      02                ;console
         jr      z,SKIP            ;Start clock if cntl "b"
         cp      CR                ;Look for carriage return
         jr      nz,SETCLK2        ;Repeat procedure is screw-up
         call    LCM               ;Load clock
         ld      a,(IX+0)          ;Read port "a"
         bit     6,a               ;Test bit 6
         jr      nz,SETCLK2        ;Must have sat clk. connected
CLKLOO:  ld      a,(IX+0)          ;Read port "a"
         bit     6,a               ;stay in loop until bit 6
         jr      z,CLKLOO          ;goes high
 SKIP:   call    LCM               ;Load clock
         DELAY   10                ;Delay 1ms
         ld      a,0               ;Load acc. with 0 to start
         ld      (STPST),a         ;real time clock

;                        - Satellite start sequence complete.

         PRINT   SYNTM             ;Print message
         ld      a,(TMEM+2h)       ;Get second
         inc     a                 ;increment it
         ld      e,a               ;Move it to E
SECLP:   ld      a,(TMEM+2h)       ;Get second again
         cp      e                 ;Has second incremented?
         jr      nz,SECLP          ;Repeat until they do
         POPALL                    ;Restore reg.
         ret
```

```
;--- LOAD CLOCK ---

LCM:     xor     a               ;Clear a
         ld      (INTCMD),a      ;Clear interrupts
         ld      a,0ffh          ;Value to clear counters
         ld      (CTRRST),a      ;Clear counter
         ld      (LTHRST),a      ;Clear latches
         ld      b,5             ;Load count for transfer
         ld      hl,TIMEN+4      ;Point to entered time
         ld      de,TMEM+3       ;Point to RTC buffer
MCLOCK:  ld      a,(hl)          ;Get value entered
         ld      (de),a          ;Move it to the clock
         DEC     hl              ;Increment to next value
         INC     de              ;Decrement clock pointer
         ld      a,b             ;Load value for compare
         cp      4               ;Is count at 3
         jr      nz,NCLOCK       ;If not, continue
         INC     de              ;Skip Day of week
NCLOCK:  xor     a
         djnz    MCLOCK          ;Continue til done
         ret

;--- GET TIME ---

;                       - INPUTS YEAR,MONTH,DAY,HOUR,MINUTE FROM
;                         TERMINAL.
;                       - AT LEAST ONE DIGIT (AND NO MORE THAN
;                         TWO) MUST BE ENTERED FOR EACH.(2 FOR YEAR)
;                       - EACH UNIT OF TIME MUST HAVE A SEPARATOR
;                         (ANY NON-DIGIT).
;                       - ENTERED TIME IS STORED IN FIVE BYTES,
;                         STARTING WITH BYTE POINTED TO BY HL.
;                       - WILL NOT RETURN UNTIL VALID ENTRY IS
;                         MADE.

TIMINP:  PUSHALL                 ;Save reg.
         ld      (TSTORE),hl     ;Save pointer
REINT:   ld      hl,(TSTORE)     ;Get it back to be sure
         PRINT   YMDHM           ;Here the operator must enter
         PRINT   QUES            ;the time
         call    GETLN
         ld      de,IBUFF        ;Save in input buffer
         ld      a,(NCI)         ;Check no. characters for valid
         cp      12              ;entry
         jr      c,REINT         ;If not right get again
         ld      b,5             ; Load the count for buffer
```

```
;                        - CHECKING IS DONE TO INSURE VALID ENTRY
;                          FOR VALUE

TINLP:   call    BCDIN           ;Get entry as packed BCD
         jr      c,REINT
         inc     hl              ;Increase pointer
         inc     de              ;Pass separator
         djnz    TINLP           ;Get next
         ld      hl,(TSTORE)     ;Restore original pointer
         inc     hl              ;Point to month
         ld      a,12H           ;Load test value
         cp      m               ;Test it
         jr      c,REINT         ;If not less than 12, try again
         xor     a               ;Clear A
         cp      m               ;Test for zero
         jr      z,REINT         ;If 0 then try again
         inc     hl              ;Point to days
         ld      a,31H           ;Load test
         cp      m               ;Check if greater than 31
         jr      c,REINT         ;If so, try again
         xor     a               ;Clear A
         cp      m               ;Test for zero
         jr      z,REINT         ;If so, try again
         inc     hl              ;Point to hours
         ld      a,23H           ;Load test
         cp      m               ;Test if greater than 23
         jr      c,REINT         ;If so, try again
         inc     hl              ;Point to minutes
         ld      a,59H           ;Load test
         cp      m               ;Test if greater than 59
         jr      c,REINT         ;If so, try again
         POPALL                  ;Restore reg.
         ret
```

```
;--- READ THE CLOCK ---

;                        - CLOCK IS READ AND STORED IN THE MEMORY
;                          IMAGE.
;                        - VALUES ARE ALSO SAVED IN TIMEN FOR
;                          PRINTING.

GCLOCK:  PUSHALL                         ;Save reg.
GCLOCK1: ld       hl,TMEM                ;Point to read reg. of the clock
         ld       de,MTMEM               ;Point to memory image
         ld       bc,8                   ;Set up count
         LDIR                            ;Load time into memory
         ld       a,(CLKSTAT)            ;Check clock status
         cp       1                      ;Look for transition flag and
         jr       z,GCLOCK1              ;reread if clock updating
         call     CLKCK                  ;Check month transition
         ld       hl,MTMEM+3             ;Point to memory storage
         ld       de,TIMEN+4             ;Point to print buffer
         ld       b,6                    ;Load count
GCLOCK2: ld       a,(hl)                 ;Get value
         ld       (de),a                 ;Save it
         inc      hl                     ;Increment memory
         dec      de                     ;Decrement buffer
         ld       a,b                    ;Check count
         cp       5
         jr       nz,GCLOCK3             ;If not continue
         inc      hl                     ;Bypass day of week
GCLOCK3: xor      a
         djnz     GCLOCK2                ;Loop until done
         POPALL
         ret

;                        - CHECK FOR MONTH TRANSITION.
;                        - USED FOR YEAR UPDATE.

CLKCK:   PUSHALL
         ld       a,(MTMEM+7)            ;Get month
         ld       hl,OTENMTH             ;Point to old month value
         cp       m                      ;Compare
         jr       z,OLDYR                ;If same, exit
         ld       (hl),a                 ;Save new month
         cp       1                      ;Check for year change
         jr       nz,OLDYR               ;Exit if not
         ld       hl,MYRS                ;Point to year
         inc      m                      ;Increment year
         daa                             ;decimal adjust
         ld       a,(hl)                 ;Look for max
         cp       9Ah                    ;Compare it
         jr       nz,OLDYR               ;Exit if not
         xor      a                      ;Load a 0
         ld       (hl),a                 ;Store new year
OLDYR:   POPALL
         ret
```

```
;--- CHECK TIME ---
;                        - THIS SUBROUTINE IS RESPONSIBLE FOR
;                          CHECKING THE TIME,WRITING THE TIME,ETC.
;                        - THIS ROUTINE IS FOR DATA ACQUISITION.
;                          THE TIME IS COMPARED WITH THE START TIME
;                          (STRTTB) AND CORRECTIONS FOR ANY OFFSET
;                          (OFFSET).
;                        - ALL EXP. AFTER 1 ARE CALCULATION ON
;                          MINUTE PERIODS (PERIOD).

TCHECK:  PUSHALL                 ;Save regs.
         ld      a,(CEXPN)       ;Get current low byte of exp. no.
         cp      01h             ;Check if first one
         jr      nz,TCK1         ;If not, use minute counter
         ld      a,(CEXPN+1)     ;Get high byte of exp. no.
         cp      00h             ;Check if 0001
         jr      nz,TCK1         ;If not, use minute counter
NOT_YET: ld      a,(TMEM+2)      ;Look for 0 seconds
         cp      0
         jr      nz,NOT_YET      ;Loop until 0
         call    GCLOCK          ;Read time
         ld      hl,STRTTB       ;Point to start time
         ld      de,TIMEN        ;Point to timer
         call    CTIM            ;Compare them
         jr      c,NOT_YET       ;Repeat until compare
         jp      OFST

;                        - AFTER FIRST ONE USE MINUTE TRANSITIONS
;                          TO DETERMINE PERIOD.

TCK1:    ld      b,0h            ;Reset the minute counter
TCK2:    ld      a,(MTMEM+3)     ;Get last valid minute
         ld      (GPCTRH),a      ;Save it
TCK3:    ld      a,(TMEM+3)      ;Get current minute
         ld      hl,GPCTRH       ;Point to the last valid minute
         cp      m               ;Compare them
         jr      z,TCK3          ;If no change, try again
         inc     b               ;Increment the minute counter
         ld      a,(PERIOD)
         cp      b               ;Check if period is complete
         jr      nz,TCK2         ;If not, try again

;                        - HANDLE OFFSET.

OFST:    ld      a,(OFFSET)      ;Load offset
         cp      0               ;Is it 0?
         jr      z,AGIN          ;Is so, do it
         ld      e,a             ;Move it to E
OFFSET_WAIT:
         ld      a,(TMEM+2)      ;Get seconds
         cp      e               ;Compare with offset
         jr      nz,OFFSET_WAIT  ;Loop til compare
AGIN:    call    TIMREC          ;Record the time
         POPALL                  ;Restore the registers
         ret
```

```
;--- CHECK TIME FOR STOP ---

;                        - THIS SUBROUTINE IS RESPONSIBLE FOR
;                          DETERMINING THAT WE ARE NOT AT STOP-TIME.
;                        - CARRY SET IF PRESENT TIME > STOP TIME.

NOTSTOP: di                      ;Disable interrupts
         call    GCLOCK          ;Read the clock
         ei                      ;Enable interrupts
         ld      hl,TIMEN        ;Point to time
         ld      de,STOPTB       ;Load stop time
         call    CTIM            ;Compare
         ret

;--- COMPARE TIMES ---

;                        - THIS SUBROUTINE COMPARES TWO TIMES
;                          POINTED TO BY HL,DE.
;                        - CARRY SET IF HL IS LATER.
;                        - AF IS ALTERED.

CTIM:    push    bc              ;Save reg.
         push    de
         push    hl
         ld      b,5             ;Set up count
CTIMLP:  ld      a,(de)          ;Load timer value
         cp      m               ;Compare with stored time
         inc     hl              ;Increment pointer
         inc     de              ;Increment pointer
         jr      c,BIGGER        ;If carry set, exit
         jr      nz,BIGGER       ;If nonzero, exit
         djnz    CTIMLP          ;If count is nonzero, get next value
BIGGER:  pop     hl              ;Restore reg.
         pop     de
         pop     bc
         ret
```

```
;--- RECORD THE TIME ---

;                         - THIS SUBROUTINE RECORDS THE TIME FOR
;                           THE TAPE RECORD IN THE EXPERIMENT SECTION
;                           OF THE TRAILING PARAMETERS.

TIMREC: PUSHALL                   ;Save regs.
RCLOCK: call    GCLOCK            ;Get time
        ld      a,(MYRS)          ;Get year
        ld      (CYR),a           ;Save that
        ld      a,(MTMEM)         ;Get millisecond
        ld      (CTIMER),a        ;Save it
        ld      a,(MTMEM+1)       ;Get .1 sec value
        ld      (CTIMER+1),a      ;Save it
        xor     a                 ;Clear a
        ld      de,SAMPTIM+0bh    ;Point to buffer for sample time
        ld      hl,MTMEM+7        ;Point to time in memory
        call    MOVT              ;Load time into the buffer
        ld      a,0               ;Load dummy value for day of week
        ld      (de),a            ;Save it
        dec     de                ;Point to next storage
        call    MOVT              ;Save day
        dec     hl                ;Pass day of week
        ld      b,3               ;Load count
RCLK1:  call    MOVT              ;Move hours,min,sec
        djnz    RCLK1             ;Loop til done
        rld                       ;Get .1 sec value
        ld      (de),a            ;Save it
        POPALL                    ;Restore reg.
        ret

;                         - THIS ROUTINE SPLITS THE PACKED BCD INTO
;                           ONE DIGIT PER ADDRESS.

MOVT:   rld                       ;Get high digit
        ld      (de),a            ;Save it
        dec     de                ;Go to next address
        xor     a                 ;Clear a for next value
        rld                       ;Get low byte
        ld      (de),a            ;Save it
        dec     de                ;Go to next address
        dec     hl                ;Point to next time value
        xor     a                 ;Clear A in preparation
        ret
```

```
;--- PRINT TIME ---

;                       -THIS SUBROUTINE PRINTS OUT DATE AND TIME.

TIMOUT: PUSHALL                 ;Save reg.
        push    hl              ;Save HL in DE
        pop     de
        inc     hl              ;Increment pointer
        call    PBCD            ;Print value
        ld      a,"/"           ;Load the separator
        call    PBYT            ;Print it
        inc     hl              ;Point to next value
        call    PBCD            ;Print it
        ld      a,"/"           ;Load the separator
        call    PBYT            ;Print it
        ex      de,hl           ;Restore HL to year
        call    PBCD            ;Print year
        ex      de,hl           ;Restore HL
        ld      a,' '           ;Load a space
        call    PBYT            ;Print it
        inc     hl              ;Point to hour
        call    PBCD            ;Print it
        ld      a,':'           ;Load a colon
        call    PBYT            ;Print it
        inc     hl              ;Point to minutes
        call    PBCD            ;Print it
        call    CRLF            ;Print a carriage return
        POPALL                  ;Restore reg.
        ret

;--- PRINT SECONDS ---

SECOUT: PUSHALL
        ld      a,':'           ;Load a colon
        call    PBYT            ;Print it
        ld      hl,MTMEM+2      ;Point to seconds
        call    PBCD            ;Print it
        ld      a,'.'           ;Load a decimal point
        call    PBYT            ;Print it
        dec     hl              ;Point to fractional sec.
        call    PBCD            ;Print it
        dec     hl              ;Point to milliseconds
        call    PBCD            ;Print it
        POPALL
        ret
```

```
; DEFINE STATEMENTS
;______________________________________________________________

CLKPRM:          db       'HIT RETURN WHEN LESS THAN 1 MINUTE TO GO'
                 db       CR,LF,0
SATNC:           db       'SATELLITE CLOCK NOT CONNECTED',CR,LF,0
SATC:            db       'SATELLITE CLOCK CONNECTED',CR,LF,0
SETM:            db       'ENTER CURRENT TIME + 1 MINUTE',CR,LF,0
SYNTM:           db       'SYNCHRONIZING TIMERS',CR,LF,0
YMDHM:           db       'YR/MTH/DAY/HR/MIN',0
```

MACRO SUBROUTINES

```
;______________________________________________________________
;original code written by jhg
;UPDATE -  9/13/85 BCD handling routines were added by gkm
;______________________________________________________________
```

```
;--- SEND A MESSAGE TO THE TERMINAL ---

;                           - THIS SUBROUTINE MUST BE HERE TO AVOID
;                             PROBLEMS WITH MACROS.

SNDMES:  push    af
MESSND:  ld      a,(hl)          ; Get byte at HL.
         cp      0               ; Test for terminator.
         jp      z,SNDRET        ; Done if 0
         call    PBYT            ; Else print character.
         inc     hl              ; Point at next.
         jr      MESSND          ; and continue.
SNDRET:  pop     af
         ret

;--- THESE SUBROUTINES ARE PART OF WPRINT MACRO ---

;                           - WARNING: THESE SUBROUTINES ARE PART OF
;                             MACROS-REGISTERS ARE PRESERVED IN MACROS,
;                             NOT IN SUBROUTINES!

WPRA:    call    SNDMES          ;Print message
         ld      hl,QUES         ;Now question mark
         call    SNDMES          ;Print it
         ret

WPRB:    ld      de,(HDRBUF)     ;Point to buffer
         call    MOVLP           ;Store message
         push    de              ;Save reg.
         call    GETLN           ;Get answer
         pop     de              ;Restore reg.
         call    RMOV            ;Store the answer
         ret

MOVLP:   ldi                     ;Transfer data
         xor     a               ;Clear A
         cp      m               ;Test for end
         jr      nz,MOVLP        ;If not, transfer next
         ld      (HDRBUF),de     ;Save pointer
         ret

RMOV:    ld      hl,IBUFF        ;Point to ASCII buffer
         ld      b,0
         ld      a,(NCI)         ;Get no. characters
         ld      c,a             ;Move it to C
         LDIR                    ;Move it
         ld      a,0             ;Clear A
INS:     ld      (de),a          ;Insert terminator
         inc     de              ;Account for insertion
         ld      (HDRBUF),de     ;Save pointer
         ret
```

```
; DELAY SUBROUTINES
;________________________________________________________________

;--- DELAY ENTRY ---

DELAYR:  call    D87T             ;17T--This call included in 87T time
         djnz    DELAYR           ;13T--This+87=100 per loop
         pop     bc               ;10T--Waste time.Note that we have 8
                                  ;when we fall through
         push    bc               ;11T--Restore reg.
         NOP                      ;4T
         NOP                      ;A total of 8+21+16+10=55 TSTATES
         NOP
         NOP
         ret                      ;10T

;--- DELAY 50 TSTATES ---

;                        - CALL TO COME HERE=17 TSTATES

D50T:    ex      af,af´           ; 4T--Switch reg. to save flags
         and     0FFh             ; 7T--This only affects F´
         ex      af,af´           ; 4T--Restore flags
         NOP                      ; 4T--Waste time
         NOP                      ; 4T--4+4+4+4+7+10+call to come
                                  ;(17)=50
         ret                      ;10T

;--- 87 TSTATE DELAY ---

;                        - CALL TO COME HERE=17 T

D87T:    jp      D77T             ;10T--waste time
D77T:    jp      D67T             ;10T--here down=77 including the
D67T:    jp      D57T             ;call in 10T--and so on.
D57T:    jp      D47T
D47T:    jp      D37T
D37T:    jp      D27T
D27T:    ret                      ;10T--NOTE THAT THIS FITS WELL
                                  ;WITH DJNZ WHICH IS 13T

;--- 100 TSTATE DELAY ---

;                        - CALL TO COME HERE=17 T

D100T:   call    D50T             ;Call is included in 50T´S timing
         ex      af,af´           ;Same tricks as in 50T
         and     0FFh
         ex      af,af´
         NOP
         NOP
         ret
```

```
; MISC ROUTINES
;______________________________________________________________

;--- DIVIDE ROUTINE ---

;                        - DIVISOR=ACC
;                        - DE=DIVIDEND
;                        - ANSWER IN DE,CARRY SET IF /0
;                        - REMAINDER IN B

DIVIDE:  or      a               ;Check for 0
         jr      nz,OKTODV       ;If not divide
         scf                     ;Set carry flag
         ret
OKTODV:  ld      c,a             ;Move divisor
         ld      b,16            ;Count = number of bits in divisor
         xor     a               ;Clear A
DLOOP:   sla     e               ;Shift contents left
         rl      d               ;Same
         rl      a               ;Same
         cp      c               ;Divide compare
         jr      c,NOINC         ;If not greater, no increments
         inc     e               ;Increment number
         sub     c               ;Subtract the divisor from A
NOINC:   djnz    DLOOP           ;Continue until B = 0
         ld      b,a             ;Save remainder in A
NOINC2:  ret

;--- CONVERT TO PACKED BCD ---

;                       - CONVERTS THE BYTE POINTED TO BY HL TO
;                         PACKED BCD.
;                       - IF THE BYTE IS BIGGER THAN 99 DECIMAL,
;                         THEN CARRY IS SET.

HTBCD:   push    bc              ;Save reg.
         push    af              ;Save acc.
         ld      b,m             ;Get value
         xor     a               ;Clear A
         cp      b               ;Compare
         jr      z,HTRET         ;If 0, then exit
         xor     a               ;Make sure all flags are clear
BCDL:    inc     a               ;Increment A
         daa                     ;Decimal adjust
         jr      c,HTRET         ;If carry then exit
LT99:    djnz    BCDL            ;Continue til done
         ld      m,a             ;Save it
HTRET:   pop     af              ;Restore acc.
         pop     bc              ;Restore reg.
         ret
```

```
;--- BCD COUNTER INITIALIZATION ---

;                         - LOCATION OF COUNTER MUST BE IN HL.
;                         - 1ST IS LOW BYTE AND 2ND IS HIGH BYTE.
;                         - BOTH BYTES WILL CONTAIN 00 AT EXIT.

BCDCLR: xor     a               ;Clear A
        ld      m,a             ;Store 0 in first byte
        inc     hl              ;Point to high byte
        ld      m,a             ;Clear it
        ret

;--- BCD COUNTER ROUTINE ---

;                         - LOCATION OF COUNTER MUST BE IN HL.
;                         - 1ST IS LOW BYTE AND 2ND IS HIGH BYTE.

BCDCT:  push    af              ;save acc.
        ld      a,(hl)          ;Get stored low byte
        inc     a               ;Increment the count
        daa
        ld      (hl),a          ;Save it
        jr      nc,BCDCT1       ;If no carry then exit
        inc     hl              ;Get high byte
        adc     a,(hl)          ;Adjust high byte
        daa
        ld      (hl),a          ;Save it
BCDCT1: pop     af              ;Restore acc.
        ret

;--- 2 BYTE BCD PRINT ROUTINE ---

;                         - LOCATION OF COUNTER MUST BE IN HL.
;                         - 1ST IS LOW BYTE AND 2ND IS HIGH BYTE.

BCDPT:  inc     hl              ;Get high byte first
        call    PBCD            ;Print it
        dec     hl              ;Get low byte
        call    PBCD            ;Print it
        ret

;--- PRINT THE PACKED BCD DIGIT POINTED TO BY HL ---

PBCD:   push    af              ;Save reg.
        ld      a,30h           ;Convert to ASCII
        rld
        call    PBYT            ;Print it
        rld
        call    PBYT            ;print it
        rld
        pop     af              ;Restore reg.
        ret
```

```
;--- GET BCD ENTRY ---

;                       - PRINTS ´?´
;                       - GETS BCD DIGITS, AND STORES THEM AT HL.
;                       - CARRY FLAG IS SET IF IMPROPER ENTRY.

INDAT:   PRINT   QUES            ;Print ´?´
         call    GETLN           ;Get response
         ld      de,IBUFF        ;Point to ASCII buffer
         ld      a,(NCI)
         cp      6
         jr      z,INDAT2
         cp      5
         jr      z,INDAT3
         call    BCDIN           ;Enter as BCD
         ret
INDAT2:  inc     hl
         call    BCDIN
         dec     hl
         call    BCDIN
         ret
INDAT3:  inc     HL
         call    BCDIN
         xor     a
         rrd
         dec     de
         dec     hl
         call    BCDIN
         ret
```

```
;--- ENTER A PACKED BCD DIGIT ---

;                          - CLEARS LOCATION POINTED TO BY HL,
;                            THEN ENTERS PACKED BCD DIGIT.
;                          - ONE OR TWO DIGITS ARE ACCEPTABLE,
;                            FOLLOWED BY TERMINATOR (ANY NON DIGIT).
;                          - CARRY IS SET IF INPUT IS INVALID.
;                          - DE MUST POINT TO ASCII CHARACTER STRING.

BCDIN:   push    bc              ;Save regs.
         ld      b,2             ;Load B with count
         ld      (hl),0          ;Clear location
BCDLOP:  ld      a,(de)          ;Get character from string
         call    CONVAS          ;Convert to BCD
         jr      c,NVALIT        ;If not a valid entry then exit
         rld                     ;Shift it
         inc     de              ;Increment pointer of string
         djnz    BCDLOP          ;Continue for the count
         ld      a,(de)          ;Get character
         call    CONVAS          ;Convert to BCD
IOK:                             ;Increment pointer of string
         ccf                     ;Complement carry flag
IBAD:    pop     bc              ;Restore reg.
         ret

NVALIT:  ld      a,b             ;Get the count
         cp      2               ;Check it
         scf                     ;Set carry flag
         jr      z,IBAD          ;If count is 2 then exit
         jr      IOK             ;Here if OK

;--- CONVERT ASCII TO BCD DIGIT ---

;                          - RETURNS WITH CARRY SET IF INVALID INPUT.

CONVAS:  sub     30h             ;Strip ASCII value
         ret     c               ;Carry is set if less than 30HEX
         cp      0ah             ;Must be <= 9
         jr      nc,NOTVAL       ;If no carry then not valid
         ccf                     ;Compliment carry (clear it)
         ret

NOTVAL:  scf                     ;Set carry flag
         ret
```

```
;--- COMPARE TWO BCD NUMBERS ---

;                             - ENTRY REQUIRES HL POINT TO HIGH BYTE.
;                             - CARRY IS SET IF CONTENTS OF HL EXCEED
;                               DE.

BCDCP:  ld      a,(de)          ;Get high byte
        cpd                     ;Compare byte
        ret     nz              ;If no compare then exit
        dec     de              ;Get low byte
        ld      a,(de)
        cp      (hl)            ;If larger carry is set
        ret
```

TERMINAL ROUTINES

```
;______________________________________________________________
;Original code written by jhg
;UPDATE - 2/15/85 code was revised to create a separate
;                 module of all terminal routines by gkm
;UPDATE - 9/13/85 code was revised to include a jump to
;                 the time check by entering a ^U by gkm
;______________________________________________________________
```

```
;--- GET BYTE AND ECHO IF ACC=0 ---

GBYTNE:  ld      a,1              ;No echo
         jr      NE               ;Jump over echo
GBYT:    ld      a,0              ;Echo entry
NE:      push    bc               ;No echo entry
         push    af               ;Save reg.
FRAMERR: ld      b,8              ;Set index for no. of bits input.

;                           - WE MUST DETECT THE BEGINNING OF THE
;                             START BIT-SO THE FIRST STEP IS TO MAKE
;                             SURE THAT IT HAS NOT BEGUN!

MARK:    ld      a,(PBDATA)       ;Get input and make sure it is
                                  ;a mark
         rla                      ;Input to carry(IS INVERTED)
         jp      nc,MARK          ;Wait till mark

;                           - NOW GET TRANSITION

GTSTRT:  ld      a,(PBDATA)       ;13T-- Look for start bit.
         rla                      ; 4T-- Move to carry flag.
         jp      c,GTSTRT         ;10T-- IF   start bit not present
                                  ; THEN go back and look again.

;                           - MAXIMUM ERROR IN START BIT TIME IS 27
;                             T-STATES.
;                           - FOR THIS SYSTEM THE CLOCK IS 1.0486*10^6
;                           - AND THE NO. OF T-STATES IN ONE BIT TIME
;                             FOR COMMON BAUD RATES ARE:

;                                     9600          109.23 T-STATES
;                                     4800          218.46
;                                     2400          436.92
;                                     1200          873.83
;                                     300          3495.30

;                           - FOR NOW, RUNNING AT 1200 BAUD, FIXED.
;                           - 1/2 BIT TIME = 437 TSTATES FOR ALL
;                             INTENTS AND PURPOSES.
;                           - WE NOW HAVE A HYPOTHETICAL START BIT.
;                           - SO NOW WAIT 1/2 BIT TIME AND SEE IF IT
;                             IS STILL THERE.
;                           - TIME NOW=TRANSITION+14 T.

         DELAY   4                ;Delay 400 TSTATES
         jp      CHSTRT           ;10T

;                           - TIME NOW=TRANS.+424T(13T TO READ LINE
;                         MAKES 437).
```

```
CHSTRT: ld      a,(PBDATA)      ;13T-- Look for start bit again.
        rla                     ; 4T-- Move it to carry flag.
        jp      c,GTSTRT        ;10T-- IF start bit is gone,
                                ;TRY AGAIN

;                      - PREPARE TO ECHO START BIT.
;                      - TIME IS 451 TSTATES AFTER START BIT
;                        EDGE DETECTED.
;                      - WE WILL NOW NEED TO DELAY ABOUT 1 BIT
;                        TIME SO THAT WE CONTINUE TO HIT THE
;                        MIDDLE OF EACH BIT AND SO THAT THE BITS
;                        WE ECHO ARE THE RIGHT LENGTH.

        ld      a,1             ; 7T-- Prepare A as start bit
        call    D50T            ;50T-- Delay 50T states
        ld      (PBSETB),a      ;13T-- and send it.(NO LONGER)
        call    D27T            ;27T-- Wait 27 T

;                      - TIME IS NOW 111 T-STATES AFTER MIDDLE
;                        OF BIT 763 TO GO BEFORE WE READ NEXT,847
;                        (IDEALLY) BEFORE NEXT ECHO.

XMIT:   DELAY   7               ;Delay 700 T-STATES
        call    D50T            ;Delay 50 more

;                      - 763-750=13 WHICH IS HOW LONG IT TAKES
;                        TO READ.

        ld      a,(PBDATA)      ;13T Get receive bit 7 in A.
                                ;84T to echo
        rlca                    ; 4T Rotate into carry flag+BIT 0
        rr      c               ; 8T then into C.
        and     1               ; 7T Mask other bits and set flags
                                ;for test
        ld      a,1             ; 7T Prepare A to set/reset line.
                                ;Flags unaltered from AND above
        NOP                     ; 8 more to add
        NOP

;                      - 50 T-STATES TO GO .LESS 13T TO DO IT
;                        AND 10T FOR THE JUMP LEAVES 27.

        call    D27T            ; 27T delay
        jp      z,GTCLRB        ; 10T test flag and goto
                                ;appropriate echo
```

```
;                            - THESE ROUTINES NO LONGER DO AN ECHO--
;                              LEFT HERE FOR TIMING (IF ECHO DESIRED
;                              PLEASE RESTORE PBCLRB WHERE APPROPRIATE).

;                            - ECHO A ONE.NOTE THAT BOTH INPUT AND
;                              OUTPUT ARE INVERTED SO THAT THIS IS
;                              REALLY A ZERO.


GTSETB:  ld      (PBSETB),a       ; 13T Set output high
         jp      GTREST           ; 10T and continue,keep timing
                                  ;the same

;                            - 23 T-STATES FOR THIS BRANCH.
;                            - THESE ROUTINES NO LONGER DO AN ECHO--
;                              LEFT HERE FOR TIMING (IF ECHO DESIRED
;                              PLEASE RESTORE PBCLRB WHERE APPROPRIATE).

GTCLRB:  ld      (PBSETB),a       ; 13T Set output low
         jp      GTREST           ; 10T and keep timing constant.

GTREST:  NOP                      ; 4T--DELAY
         djnz    XMIT             ;13T Go till all bits in.

;                            - TIMING SINCE LAST READ=4+8+7+7+8+27+10
;                              +13+10+4+13=111.
;                            - TIMING SINCE LAST WRITE=10+4+13=27.
;                            - NOTE TIMING PRESERVED THROUGH XMIT LOOP
;                              (SEE VALUES AT START OF LOOP).
;                            - 8T WHEN WE FALL THROUGH.
;                            - TIMING SINCE LAST READ=111-13+8=106.
;                            - TIMING SINCE LAST WRITE=27-13+8=22.

         DELAY   7
         call    D47T
         NOP
         NOP

;                            - TIMING=755+106=861 SINCE LAST READ(+13
;                              FOR READ=874).
;                            - TIMING=755+22=777 SINCE LAST WRITE.

         ld      a,(PBDATA)       ;13T Get receive bit 7 in A.
         rla                      ; 4T to carry
         jp      nc,FRAMERR       ;10T Framing error if not a stop
                                  ;bit
```

```
;                         - TIMING=777+27=804 SINCE LAST WRITE.

        call    D50T            ;50T
        ld      a,1             ; 7T Prepare A as stop bit.
        ld      (PBSETB),a      ;13T Echo stop bit AT 874T

        DELAY   9               ;Timing no longer critical,must
                                ;be long enough
        pop     af              ;Restore reg.
        push    af              ;Save reg.
        cp      0               ;Echo required if zero
        jp      z,ECHO          ;

ERET:   pop     af              ;No echo needed
        ld      a,c             ; Transfer assembled byte to A
        pop     bc              ;Restore reg.
        ret

ECHO:   ld      a,c             ;Get character
        cp      BS              ;Backspace?
        jp      z,BACK          ;Do a backspace if yes
        cp      DEL             ;DEL=BS FOR US
        jp      z,BACK
        cp      CTLU            ;Control U?
        jp      z,CLKRD         ;Read clock
        cp      CR              ;Carriage return?
        jp      z,CRLFR         ;DO CR,LF ECHO BUT KEEP CR
        cp      SPC             ;Space?
        jp      c,ERET          ;DO NOT ECHO OTHER CONTROL CODES
        call    PBYT            ;Normal echo
        jp      ERET

CRLFR:  call    CRLF            ;Do a CRLF
        jp      ERET

BACK:   ld      c,BS            ;Make sure DEL and BS are same
        jp      ERET            ;Let caller decide what to do(ECHO)
```

```
;--- SEND CHARACTER TO TERMINAL ---

;                        - GET READY

PBYT:    push    af              ;Save reg.
         push    bc              ;Save reg.
         ld      c,a             ;4T Move output byte to C register.
         ld      b,8             ;Word length
         ld      a,1             ;Prepare A to send bits,

 ;                       - SEND START BIT

         ld      (PBCLRB),a      ;And send start bit

;                        - TIMING=874 T-STATES TO END OF NEXT SEND

         NOP                     ; 4T
         NOP                     ; 4T
         NOP                     ; 4T
         NOP                     ; 4T
         call    D27T            ;27T
         DELAY   8               ;800T

;                        - TIMING=874-843=31 TO SEND

PUTLP:   rrc     c               ;8T--Put bit into carry
         jp      nc,PTCLRB       ;10T--If a zero there is no carry
                                 ;Rest is like GBYT above

;                        - FROM PUTLP TO SEND=18+13=31.
;                        - TIMING = 23T STATES.

PTSETB:  ld      (PBSETB),a      ;13T
         jp      PUTRST          ;10T

;                        - FROM PUTLP TO SEND=18+13=31.
;                        - TIMING = 23T FOR THIS BRANCH.

PTCLRB:  ld      (PBCLRB),a      ;13T
         jp      PUTRST          ;10T

;                        - TIMING=874-10=864 T-STATES TO SEND.
;                        - TIMING=864-833=31.
;                        - TIMING PRESERVED IN LOOP, 8T FALLING
;                          THROUGH.
```

```
PUTRST: DELAY   8               ;800T
        jp      DLY1            ;10T
DLY1:   jp      DLY2            ;10T
DLY2:   djnz    PUTLP           ;13T LOOP TILL DONE

;                       - TIMING=31+13-8=36

        and     OFFh            ; 7T    CLEAR CARRY,DELAY
        NOP                     ; 4T    DELAY
        NOP                     ; 4T    DELAY
        NOP                     ; 4T    DELAY
        NOP                     ; 4T    DELAY

;                       - TIMING=36-23=13--JUST TIME TO DO IT.

        ld      (PBSETB),a      ;13T    STOPBIT
        DELAY   9               ;TIMING NOT CRITICAL NOW
        pop     bc              ;Restore reg.
        pop     af              ;REstore reg.
        ret

;--- BACK SPACE AND ERASE ROUTINE---

BSPCR:  ld      a,BS            ;BS/SPC/BS
        call    PBYT            ;Print it
        ld      a,SPC           ;Get space
        call    PBYT            ;Print it
        ld      a,BS            ;Get backspace
        call    PBYT            ;Print it
        ret

;--- SEND A CARRIAGE RETURN LINE FEED TO TERMINAL ---

CRLF:   push    hl              ;Save reg.
        ld      hl,CRLFM        ;Point to message
        call    SNDMES          ;Print it
        pop     hl              ;Restore reg.
        ret

;--- PAUSE BEFORE CLEARING SCREEN ---

CONT:   PRINT   CONTIN          ;Print continue message
        call    GBYTNE          ;Look for key
        ret
```

```
;--- GET A RESPONSE TO A QUERY ---

GETLN:  PUSHALL
GLN1:   ld      hl,IBUFF        ;ASCII input buffer
        ld      c,0             ;No. of characters
        ld      b,MAXB          ;Get max buffer length
GLN2:   call    GBYT            ;Get character
        cp      BS              ;Backspace?
        jp      nz,NOBS         ;NO
        ld      a,b             ;Compare size
        cp      MAXB
        jp      z,GLN1          ;Ignore at beginning
        call    BSPCR           ;Backspace
        inc     b               ;Restore character count
        dec     c               ;No. of characters
        dec     hl
        jr      GLN2            ;Try again
NOBS:   cp      CTLX            ;CONTROL X?
        jp      z,CTLXR         ;Do it if so
        cp      CTLU            ;CONTROL U?
        jp      z,CLKRD         ;Read clock
        cp      CR              ;Carriage return?
        jr      z,NOTST
        cp      SPC             ;Space?
        jr      c,GLN2          ;NO OTHER CONTROL CHARACTERS
                                ;RECOGNIZED
NOTST:  ld      (hl),a          ;Store character
        inc     hl              ;Increase pointer
        inc     c               ;Increase character count
        cp      CR              ;Look for carriage return
        jp      z,GLEND         ;DONE SO EXIT PROPERLY
        djnz    GLN2            ;Get next
        jp      CTLXR           ;INPUT TOO LONG,MUST BE ERROR

GLEND:  ld      (hl),LF         ;Add line feed
        inc     c               ;Inc. character count
        ld      a,c             ;Save it
        ld      (NCI),a
        POPALL                  ;Restore reg.
        ret

CTLXR:  ld      a,c             ;Get character count
        cp      0               ;Check if zero
        jr      z,GLN1          ;If so, no action needed
        ld      b,c
BLOOP:  call    BSPCR           ;Clean up screen
        djnz    BLOOP
        jr      GLN1
```

```
; DEFINE STATEMENTS
;_______________________________________________________________

CONTIN:  db      'HIT ANY KEY TO CONTINUE',0
CRLFM:   db      CR,LF,0
QUES:    db      ' ?     ',0
CLRSCR:  REPT    22
         db      LF
         ENDM
         db      CR
         db      '                  '
         db      U.S. GEOLOGICAL SURVEY 4-CHANNEL DATA RECORDER'
         db      CR,LF
         db      8
         db      CR
         REPT    13
         db      LF
         ENDM
         db      CR
         db      0
OKQ:     db      'IS THIS CORRECT (Y/N)',0
```

TAPE ROUTINES

```
;_______________________________________________________________
;Original code written by jhg
;UPDATE -  8/ 8/84 code was revised to include a tape
;                  test and improved error checking by
;                  ohh
;UPDATE - 10/18/84 code was modularized and commented
;                  by gkm
;UPDATE -  7/16/85 code was revised by gkm for the
;                  following:
;                        1.Addition of print routines for
;                         display of tape commands and
;                         status on diagnostic switch.
;                        2.Addition of routines to keep
;                         track of number records written
;                         and write errors.
;                        3.Altered end of tape routine
;                         to back up 2 records to allow
;                         room for 2 file marks and make
;                         possible to recover time.
;                        4.Altered tape insertion routine
;                         so that it is no longer necessary
;                         to reinsert cartridge to find BOT.
;UPDATE -  7/30/85 code was added to add more protection
;                  with a time out routine on read status
;                  to restart the drive on failure by gkm.
;_______________________________________________________________
```

```
;POWER CONTROL FOR TAPE DRIVE
;_______________________________________________________________

;--- TURN OFF POWER TO TAPE DRIVE ---

;                        - DRIVE IS TURNED OFF AND RESET
;                        - POWER TO THE DRIVE MUST BE TURNED OFF
;                          BEFORE THE POWER TO THE CONTROLLER TO
;                          PREVENT STRAY MOTOR MOVEMENT.

OFFCART:push    af              ;Save acc.
         IF     DIAG
         PRINT  SLPMES          ;Print sleep message if diagnostics
         ENDIF
         ld     a,OFF12V        ;Turn off drive first
         out    (PWRPRT),a
         DELAY  256             ;25.6ms to allow power to settle
         ld     a,OFF5V         ;Turn off controller
         out    (PWRPRT),a
         ld     a,RSTCC         ;Hold reset low
         ld     (PBCLRB),a
         pop    af              ;Restore acc.
         ret
```

```
;--- TURN ON POWER TO THE TAPE DRIVE ---

;                         - POWER IS TURNED ON AND INTERFACE RESET.
;                         - THE POWER TO THE CONTROLLER MUST BE
;                           TURNED ON BEFORE THE POWER TO THE DRIVE
;                           TO PREVENT POSSIBLE UNWANTED TAPE MOVEMENT.

ONCART: push    af              ;Save acc.
        IF      DIAG
        PRINT   WAKMES          ;Print wake message if diagnostics
        ENDIF
        ld      a,ON5V          ;Turn on cartridge controller
        out     (PWRPRT),a
        DELAY   30              ;Delay 3ms for power to settle
        ld      a,RSTCC         ;Set controller reset
        ld      (PBSETB),a
        DELAY   100             ;10ms pulse width
        ld      (PBCLRB),a      ;Set controller for operation
        ld      a,ON12V         ;Turn on drive
        out     (PWRPRT),a
        DELAY   256             ;Allow drive to stabilize
        ld      a,RSTCC         ;Reset again in case it came up
        ld      (PBCLRB),a      ;srewy
        DELAY   10              ;1ms pulse width
        ld      (PBSETB),a      ;Set controller for operation
        DELAY   256             ;Let the whole thing stabilize
        ld      b,0ffh          ;Load count for ready test

NRDY:   ld      a,CURSTATC
        call    EXECN           ;Get interface status
        ld      a,(IS)
        and     COMSTATM        ;Strip mask
        jr      z,RDY           ;If ready exit
        DELAY   20              ;Allow 2ms before next try
        djnz    NRDY            ;Try again
RDY:    pop     af              ;Restore acc.
        ret
```

```
;COMMANDS SPECIFIC TO OBS/TIP FORMAT
;______________________________________________________________

;--- WRITE DATA RECORD FOR WINDOW MODE ---

WDR:     call    ONCART          ;Turn on the drive
         call    NOBLOCK         ;Determine no. blocks to write
         ld      a,(BSTART)      ;Get start address
         ld      (WBSTART),a     ;Save it
         call    FIXNOS          ;Create TIP header
         call    WRITED          ;Write the data
         call    OFFCART         ;Turn off the drive
         ret

;--- WRITE DATA RECORD FOR EVENT MODE ---

EWDR:    call    ONCART          ;Turn on the drive
         call    NOBLOCK         ;Determine how many blocks
         ld      hl,(BSTART)
         ld      (WBSTART),hl    ;Save beginning of buffer
         ld      hl,(BEND)
         ld      (WBUFSAV),hl    ;Save beginning of data
         call    FIXNOS          ;Create TIP header
         call    WRITED          ;Write the data
         call    OFFCART         ;Turn off the drive
         ret
```

```
;--- WRITE DATA ---
;                        - INPUTS: NAME OF FILE MUST BE IN BUF16,
;                          WBSTART MUST BE SET FOR CIRCULAR OR
;                          NON-CIRCULAR BUFFER, NO. OF BLOCKS TO
;                          WRITE(EXCLUDING PARAMS) MUST BE IN THE
;                          DATABLOCK, BEGINNING ADDRESS OF BUFFER
;                          MUST BE IN WBUFSAV.
;                        - PARAMETERS ARE WRITTEN IF THE LAST
;                          BLOCK=40H-2 SECTORS OF 128 BYTES.
;                        - ERROR IF A BLOCK IS NOT EITHER 40H-2
;                          OR 40H.

WRITED:  call    SNDMA            ;Send MA
         ld      a,(DATABLOCK)    ;Load sector count
         ld      (ESECT),a        ;Save sector count if error
         ld      hl,(WBUFSAV)     ;Point to start of data
         ld      (EBUFSAV),hl     ;Save it in case of error

;NOTE:
;        ERROR HANDLER MUST RESTORE OLD SECTOR COUNT & OLD WBUFSAV,
;       THEN FIX STACK AND JUMP TO WRITED.  WRITED CANNOT SAVE
;       REGS BECAUSE OF THIS.

         cp      41h              ;See if less than or = to 40H
         jr      c,LASTBLK        ;If so end data
         sub     40h              ;Decrement sector counter
         ld      (DATABLOCK),a    ;Save it
         ld      b,40h
         call    RECSIZ           ;Set RECSIZ = 40h, LDF=00
         call    WRITE16          ;Write 16byte header
BWR:     call    WRITE128         ;Write data
         djnz    BWR              ;Until done
         call    WRITEIT          ;This sub sends CA and
                                  ;handles errors
         jr      WRITED           ;Repeat as needed

LASTBLK: ld      b,a
         call    RECSIZ           ;RECSIZ = 40h
         call    SETLDF           ;Set last block flag = 1
         call    WRITE16          ;Write header
LDB:     call    WRITE128         ;Write data
         djnz    LDB              ;Until done
         cp      40h              ;Check if done
         jr      z,NPB            ;If so, write it to tape
         call    WRITEPAR         ;Write parameters
NPB:     call    WRITEIT          ;Write to tape
         ret
```

```
;--- WRITE THE PARAMETERS TO TAPE BUFFER ---

;                         - PARAMETERS ARE LAST TWO BLOCKS OF ALL
;                           DATA RECORDS(256 BYTES).
WRITEPAR:
         PUSHALL                  ;Save reg.
         ld      hl,SERBUF        ;Point to series parameters
         ld      (WBUFSAV),hl     ;WBUFSAV is preserved(in case of
                                  ; error) in EBUFSAV.We are done
                                  ;with data buffer if there is no
                                  ; error.
         ld      a,(WBSTART)      ;Preserve circular status in
                                  ;case of error.
         STA     (WBSTART),0      ;Set to non circ. buffer
         ld      b,2              ;Load for 2 128byte records?
WRPARA:  call    WRITE128         ;Write to tape buffer
         djnz    WRPARA           ;Write until done
         ld      (WBSTART),a      ;Restore circ. status since no
                                  ;regs. affected
         POPALL                   ;Restore reg.
         ret

;--- SET THE RECORD SIZE IN BUF16 = B REG. & LDF=0 ---

;                         - SET RECORD SIZE TO 40H for PARAMS.

RECSIZ:  push    af               ;Save reg.
         xor     a                ;Clear A
         ld      (B16LB),a        ;Save last block flag
         ld      a,40H            ;Load rec. size
         ld      (B16RC),a        ;Save it
         pop     af               ;Restore reg.
         ret

;--- SET LAST BLOCK FLAG ---

SETLDF:  STA     (B16LB),1        ;Load last block flag
         ret

;--- COMPUTE # OF SECTORS OF 128B FROM BUFSIZ --

;                         - CALL BEFORE WRITING ANY DATA RECORD.

NOBLOCK: push    af               ;Save reg.
         push    bc               ;Save reg.
         ld      a,(BUFSIZ)       ;Load buffer size
         ld      b,a              ;Move it to B
         xor     a                ;Clear A
ADDREC:  add     a,40H            ;Add 64 byte groups
         djnz    ADDREC           ;Until filled
         sub     2                ;Allow for parameters
         ld      (DATABLOCK),a    ;Save it
         pop     bc               ;Restore reg.
         pop     af               ;Restore reg.
         ret
```

```
;--- WRITE A 128 BYTE RECORD TO TAPE BUFFER ---

;                        - INPUT PARAMETERS: WBSTART,WBUFSAV.

WRITE128:
         PUSHALL                 ;Save reg.
         ld      hl,(WBUFSAV)    ;Get pointer
         ld      b,128           ;Load no. bytes
         call    WBLOCK          ;Write it
         ld      (WBUFSAV),hl    ;Save pointer
         POPALL                  ;Restore reg.
         ret

;--- MOVE NAME TO 16 BYTE HDR BUFFER ---

;                        - CALL WITH HL SET TO APPROPRIATE NAME.

MNAME:   PUSHALL                 ;Save reg.
         ld      de,B16NAME      ;Point to name
         ld      bc,8            ;Load no. bytes
         ldir                    ;Fill buffer
         POPALL                  ;Restore reg.
         ret

;--- FIX EXP # AND SERIES # IN 16 BYTE BUFFER ---

;                        - SERIES NO. IS LIMITED TO 4 BYTES.
;                        - EXP. NUMBER IS LIMITED TO 4 BYTES.

FIXNOS:  PUSHALL
         ld      b,02h           ;Load count for BCDFIX
         ld      hl,CSN+1        ;Point to MSB of series number
         ld      de,BSERNO       ;Point to TIP buffer
         call    BCDFIX          ;Insert the series number into
                                 ;TIP buffer
         ld      b,02h           ;Load count for BCDFIX
         ld      hl,CEXPN+1      ;Point to MSB of exp. no.
         ld      de,BEXPNH       ;Point to TIP buffer
         call    BCDFIX          ;Insert exp. no. into TIP buffer
         POPALL
         ret
```

```
BCDFIX:  ld      a,(hl)          ;Get High byte
         and     11110000B       ;Use only upper four bits
         srl     a               ;Shift upper 4 bytes to lower 4
         srl     a
         srl     a
         srl     a
         xor     30h             ;Convert to ASCII
         ld      (de),a          ;Save it
         inc     de              ;Set up for next part of number
         ld      a,(hl)          ;Get the number back
         and     00001111B       ;Use the lower four bits
         xor     30h             ;Convert to ASCII
         ld      (de),a          ;Save it
         inc     de              ;Set up for next byte
         dec     hl
         djnz    BCDFIX          ;Repeat til done
         ret

;--- WRITE THE 16 BYTE TIP HEADER TO TAPE BUFFER ---

WRITE16: PUSHALL                 ;Save reg.
         ld      hl,BUF16        ;Point to header buffer
         ld      b,16            ;Load no. bytes
         ld      a,(WBSTART)     ;Get circ. status
         STA     (WBSTART),0     ;No circular buffer for 16B
         call    WBLOCK          ;Write it
         ld      (WBSTART),a     ;Preserve WBSTART
         POPALL                  ;Restore reg.
         ret

;--- CLEAR THE 16 BYTE HEADER BUFFER ---

BLNKB16: PUSHALL                 ;Save regs.
         ld      bc,16           ;Load number of bytes to clear
         ld      hl,BLANK16      ;Point at clear data
         ld      de,BUF16        ;Point to header buffer
         ldir                    ;Fill buffer with spaces
         POPALL                  ;Restore reg.
         ret
```

```
;--- WRITE BLOCK TO PORT BUFFER ---

;                        - B MUST CONTAIN BYTE COUNT.
;                        - HL MUST POINT TO FIRST BYTE TO WRITE.
;                        - WBSTART MUST BE LOADED WITH ZERO IF NOT
;                          CIRCULAR OR WITH BSTART IF CIRC.

WBLOCK:  push    af              ;Save reg.
         push    bc              ;Save reg.
         ld      c,DA            ;Point to Data argument
WLOOP:   call    WSTAT           ;Check status
         outi                    ;Write block of data to buffer
         jr      z,WEND          ;If through end it
         ld      a,(WBSTART)     ;Save new count
         or      h
         ld      h,a
         jr      WLOOP           ;Continue sending data
WEND:    pop     bc              ;Restore reg.
         pop     af              ;Restore reg.
         ret

;--- CHECK FOR END OF TRACK ---

EOTCH:   ld      a,(DS)          ;Get drive status
         bit     2,a             ;Check for EOT flag
         ret     z               ;If no EOT, return
         call    EOT             ;End the track
         or      0ffh            ;Set flag
         ret

;--- END THE TRACK ---

EOT:     push    af              ;Save the reg.
         ld      a,(IMA)         ;Get mode argument
         and     TRACKMA         ;Strip mask
         inc     a               ;Increase the track
         cp      4               ;Check for end ot tape
         jr      z,ENDTAPE       ;If so, exit
         or      MAN             ;Restore mask
         ld      (IMA),a         ;Save new mode argument
         call    FMK             ;Write the file mark
         pop     af
         ret

ENDTAPE: ld      a,REVSPCRECC    ;Back up a record to allow room
         call    EXECN           ;for 1st file mark
         ld      a,REVSPCRECC    ;Back up another rec. to allow
         call    EXECN           ;room for 2nd file mark
         PRINT   OT1             ;Print end of tape message
         jp      ETERM
```

```
;GENERAL TAPE COMMAND EXECUTION
;______________________________________________________________

;--- INSTALL A TAPE IN THE DRIVE ---

;                        - IF TAPE IS ALREADY IN THE DRIVE AT
;                          POWER UP, THE TAPE WILL REWIND TO BOT.
;                        - THIS ROUTINE ALSO CHECKS FOR WRITE
;                          PROTECT.
;                        - ALL OTHER ERRORS WILL SIMPLY RESTART THE
;                          ROUTINE.

STAPE:   ld      hl,0000h        ;Save abort vector
         ld      (ABRTV),hl
         call    ONCART          ;Turn on drive
         STA     (IMA),MAN       ;Update normal mode argument
         call    BLNKB16         ;Clear TIP header buffer

;                        - NO TAPE IN DRIVE?

STAP1:   ld      a,CURSTATC
         call    EXECN           ;Get interface status
         ld      a,(IS)
         cp      92h             ;Look if tape in drive
         jr      nz,STAP4        ;If there, check tape
NOTAPE:  PRINT   INTAPE          ;Loop til tape is inserted
         ld      a,CURSTATC
         call    EXECN           ;Get interface status
         ld      a,(IS)
         cp      92h             ;Look if tape in drive
         jr      z,NOTAPE

;                        -MAKE SURE TAPE IS AT BOT AND NOT SAFE.

STAP4:   ld      a,REWINDC       ;Rewind to BOT
         call    EXECN
         ld      a,(DS)          ;Get drive status
         bit     0,a             ;Check for write protect
         jr      z,STAP3         ;If so, print error
         bit     3,a             ;Check for BOT
         jr      z,STAP1         ;If not,try again
STAP2:   IF      DIAG
         PRINT   BOTMES          ;Print BOT if diagnostics

;                        - INITIALIZE WRITE ERROR COUNT.

         ld      hl,RWN          ;Point to 1st byte of count
         call    BCDCLR          ;Zero rewrite counter
         ENDIF
         call    OFFCART         ;Tape ready, so power down
         ret
STAP3:   call    PROTECTED       ;Print write protected and restart
         jr      NOTAPE
```

```
;--- TEST THE TAPE ---
;                         - THIS ROUTINE WRITES 1 8K RECORD TO TAPE
;                           USING A TEST PATTERN.
;                         - THE TEST PATTERN IS RECREATED IN MEMORY,
;                           THE TAPE IS THEN BACKED UP AND READ.
;                         - THE PATTERN READ FROM TAPE IS THEN
;                           COMPARED WITH THE PATTERN IN MEMORY.
TSTTAP: ld      hl,TSTTAP         ;Save abort vector
        ld      (ABRTV),hl
                                  ;Load 8000H with test pattern
        ld      bc,0a0h           ;ONE TIME THROUGH,END AT A000H
        ld      (ENDRAM),bc       ;Save it
        ld      hl,8000h          ;point to location of ram to test
        ld      (RAMST),hl        ;Save it
        call    RTEST             ;This loads the pattern and tests

;                         - PATTERN LOADED,SO WRITE IT TO TAPE

        STA     (WBSTART),0H      ;Load the start address for a write
        STA     (DATABLOCK),40H ;40H sectors (no param.)
        ld      (WBUFSAV),hl      ;Save the pointer during write
        call    ONCART
        call    WRITED            ;Write the test pattern
        ld      hl,8000h          ;Point to test pattern in memory
        call    CLR2K             ;Clear the memory

;                                 - NOW READ IT BACK

        xor     a                 ;Clear A (track 0)
        out     (PA),a            ;Send position arg. to controller
        ld      (RDBUF),hl        ;Point to the read buffer
        ld      a,REVSPCRECC      ;Back the tape one record
        call    EXECN
        call    READIT            ;Read the record
        ld      bc,(ENDRAM)
        ld      hl,(RAMST)        ;Point to start of ram
        xor     a                 ;Start pattern
        ex      af,af´            ;Save it in preparation
TST:    ex      af,af´            ;Restore pattern
        cp      (hl)              ;Test pattern
        jr      nz,TST3           ;No compare, then error
        inc     hl                ;Point to next byte
        inc     a                 ;Increment pattern
        ex      af,af´            ;Save pattern
        ld      a,h               ;Get high byte
        cp      c                 ;Test for end
        jr      nz,TST            ;If not, do again
        xor     a                 ;Clear A
        jr      z,COK             ;If OK, then proceed
TST3:   PRINT   NOCOMPM           ;Print error message
        jp      STAP1             ;Start over
COK:    PRINT   COMPSM            ;Print OK message
        call    OFFCART           ;Turn off cartridge
        call    CONT              ;Pause before clearing screen
        ret
```

```
;--- EXECUTE A COMMAND TO DRIVE ---
;                         - COMMAND MUST BE IN A BEFORE ENTRY.
EXEC:    ld      (ICA),a          ;Save command
         call    SNDMA            ;Send mode argument
         ld      a,(ICA)          ;Restore command
EXECN:   call    TIMERR           ;Set up time out routine
         out     (CA),a           ;Execute command
         IF      DIAG
         PRINT   EXEMES
         add     a,30h
         call    PBYT             ;Print command if diagnostics
         ld      a,' '
         call    PBYT             ;Follow command with a space
         ENDIF
         call    ISTAT            ;Get status
         call    TIMOK            ;Stop time out error routine
         ret

;--- TIME OUT ERROR ROUTINE ---

TIMERR:  cp      40h              ;No time out if rewind
         ret     z
         push    af
         ld      hl,(ABRTV)       ;Get abort vector
         xor     a                ;Clear A
         cp      h                ;Compare with memory
         jr      nz,TIMER1        ;Continue if not start tape
         pop     af               ;Restore command
         ret
TIMER1:  di                       ;Disable interrupts
         ld      (STOPT1),a       ;Stop timer just in case
         ld      (IX+T1LSB),0ffh  ;Reload values
         ld      (IX+T1MSB),06h
         ld      (STRT1),a        ;Start timer 1
         ld      a,T1_INT_EN or EARLY_TERM
                                  ;Enable interrupts
         out     (INTPRT),a
         ei                       ;Enable interrupts
         pop     af
         ret

TIMOK:   push    af
         di                       ;Disable interrupts
         ld      (STOPT1),a       ;Stop timer 1
         ei                       ;Enable interrupts
         pop     af
         ret

TFAIL:   di                       ;Disable interrupts
         ld      (STOPT1),a       ;Stop timer 1
         ei                       ;Enable interrupts
         PRINT   TERR
         call    OFFCART          ;Reset drive
         ld      hl,(ABRTV)
         jp      (hl)             ;Try again
```

```
;--- READ STATUS PORT ---

;                        - THIS ROUTINE WILL READ THE STATUS PORTS
;                          AND SAVE THE VALUES IN (IS) AND (DS).

ISTAT:   push    af              ;Save accumulator
         call    RSTAT           ;Wait til data is there
         in      a,(CA)          ;Read drive status and store it
         ld      (DS),a
         in      a,(CA)          ;Get interface and store it
         ld      (IS),a
         IF      DIAG
         PRINT   SD              ;Print drive status
         ld      a,(DS)
         call    TOHEX
         ld      a,' '           ;Separate with a space
         call    PBYT
         PRINT   SI              ;Print interface status
         ld      a,(IS)
         call    TOHEX
         call    CRLF
         ENDIF
         pop     af              ;Restore accumulator
         ret

;                        - PRINT ROUTINE FOR IS AND DS

         IF DIAG
TOHEX:   push    bc              ;Save reg.
         ld      c,a             ;Move A to C
         rra                     ;Shift A
         rra
         rra
         rra
         call    SHWHEX
         ld      a,c
         call    SHWHEX
         pop     bc
         ret

SHWHEX:  and     0fh
         add     a,30h
         cp      3ah
         jp      c,PBYT
         add     a,7
         jp      PBYT
         ret
         ENDIF
```

```
;--- CHECK READ STATUS ---

;                          - THIS ROUTINE WILL LOOP UNTIL DATA IS
;                            AVAILABLE.

RSTAT:   in      a,(PS)          ;Read port status
         rrca
         rrca
         jr      nc,RSTAT        ;If not, loop til ready
         ret

;--- CHECK WRITE STATUS ---

;                          - THIS ROUTINE WILL LOOP UNTIL DRIVE IS
;                            READY FOR DATA.

WSTAT:   in      a,(PS)          ;Read port status
         rrca                    ;Ready for data?
         jr      nc,WSTAT        ;If not, loop til ready
         ret

;--- SEND MODE ARGUMENT ---

SNDMA:   push    af              ;Save reg.
         call    WSTAT           ;Ready to receive?
         ld      a,(IMA)         ;Get mode argument
         out     (MA),a          ;Send it to drive
         pop     af              ;Restore reg.
         ret
```

```
;--- READ A RECORD FROM TAPE ---

;                        - READ AN 8K BLOCK FROM TAPE
;                        - HEADER IS READ TO TIPBUF
;                        - BODY TO ADDRESS IN RDBUFF WHICH MUST BE
;                          SET BEFORE CALLING.

READIT:  ld      a,READC         ;Issue read command
         call    EXEC
         ld      a,(IS)          ;Get interface status
         cp      0C0h            ;Check for error
         jp      nz,ABORT        ;If error, abort
         ld      c,CA            ;Test for data ready
         call    RSTAT
         in      e,(C)           ;Get first byte for count
         call    RSTAT           ;Wait for next
         in      d,(C)           ;Get second byte
         ld      hl,TIPBUF       ;Point to the TIP buffer

;                        - LOAD TIP BUFFER

READTIP: dec     de              ;Decrement the count
         call    RSTAT           ;Wait for data
         ini                     ;Move data to buffer and inc. HL
         bit     4,l             ;Look for end of data
         jr      nz,READTIP      ;Continue until done
         ld      hl,(RDBUF)      ;Point to the buffer for the body

;                        - LOAD DATA BUFFER

READLP:  dec     de              ;Decrement the count
         bit     7,d             ;Check for end of data
         jr      nz,DONEOK       ;Exit if done
         call    RSTAT           ;Wait for data
         ini                     ;Move data to buffer and inc. HL
         jr      READLP          ;Continue til done
DONEOK:  ret
```

```
;--- WRITE A FILE MARK ---

FMK:     ld      b,03h          ;Load number of tries
FMK1:    ld      a,WRITEFMC     ;Issue command to drive
         call    EXEC           ;Get status
         ld      a,(IS)
         and     COMSTATM       ;Strip mask
         ret     z              ;If OK exit
         call    OFFCART        ;If error, reset and try again
         DELAY   256
         call    ONCART
         djnz    FMK1
         ret

;--- WRITE DATA TO TAPE ---

WRITEIT: ld      a,WRITEWCHC    ;Execute write command
         call    EXECN
BOTFIX:  ld      a,(IS)         ;Get interface status
         and     COMSTATM       ;Strip mask
         jr      nz,WABORT      ;If error, abort
         ld      a,(DS)         ;Get disk status
         and     BOTM           ;Check for BOT
         jr      z,NOTBOT
         ld      a,BWRITEC      ;Do a dummy write if BOT
         call    EXECN
         jr      BOTFIX
NOTBOT:  call    EOTCH          ;Check for end of track during write
         ld      a,CURSTATC
         call    EXECN          ;Check for end of track now
         call    EOTCH
         ret
```

```
; ABORT AND ERROR ROUTINES
;______________________________________________________________

;--- WRITE ABORT HANDLER ---

WABORT:  ld      hl,WRECOVER     ;Point to routine
         push    hl              ;Save reg.
         IF      DIAG
         ld      hl,RWN          ;Point to rewrite number
         call    BCDCT           ;Increment counter
         ENDIF

;--- GENERAL ABORT HANDLER ---

ABORT:   PRINT   ABORTM          ;Syntax reject not handled
         ld      a,(IS)          ;Load interface status
         and     0Fh             ;Strip mask
         sla     a               ;X2
         ld      c,a             ;Move it to C
         ld      b,0             ;Fill B with 0
         ld      hl,JTABLEB      ;Point to error table
         add     hl,bc           ;Point to error
         ld      e,m             ;Load first part in E
         inc     hl              ;Incr. to next part of message
         ld      d,m             ;Load next part
         ex      de,hl           ;Move it to HL
         jp      (HL)            ;Go to error routine

;--- RECOVER A WRITE ERROR ---
;                       - SECTOR COUNT AND BUFFER PTR RESTORED.

WRECOVER:
         call    OFFCART         ;Turn off cartridge
         DELAY   256
         call    ONCART          ;Turn on cartridge
         ld      a,(ESECT)       ;Load sector count
         ld      (DATABLOCK),a   ;Save it
         ld      hl,(EBUFSAV)    ;Point to saved pointer
         ld      (WBUFSAV),hl    ;Save it
         pop     hl
         jp      WRITED          ;Continue writing
```

```
;--- PRINT ERROR ---

FLAG:    PRINT   CD0
         ret
PROTECTED:
         PRINT   CD1
         ret
NODRIVE: jp      NOTAPE
NORESP:  PRINT   CD3
         ret
FMVER:   PRINT   CD6
         ret
TABORT:  call    EOTCH            ;Check for end of track
         jr      z,NOEOT          ;If not print abort mess.
         ld      a,(IMA)          ;Load mode argument
         and     TRACKMA          ;Strip mask
         or      30h              ;Convert to decimal
         PRINT   TRACKNM          ;Print track number
         call    PBYT
         call    CRLF
         ret
NOEOT:   PRINT   CD7
         ret
RFHER:   PRINT   CD8
         ret
RFCRCC:  PRINT   CD9
         ret
RFSHORT: PRINT   CD10
         ret
RFBVP:   PRINT   CD11
         ret
WFRAW:   PRINT   CD12
         ret
WF:      PRINT   CD13
         ret
RFFMD:   PRINT   CD14
         ret
UNKNOWN:PRINT    CD15
         ret
```

```
; DEFINE STATEMENTS
;_____________________________________________________________________

;--- PRINT MESSAGES ---

ABORTM:         db      ´ABORT ´,0
BDTAHDR:        db      ´S0000E0000´
BLANK16:        db      0,20H,20H,20H,20H,20H,20H,20H,20H
                db      20H,20H,20H,0,0,0,0
CD0:            db      ´CODE 0 FLAG COND.´,CR,LF,0
CD1:            db      ´CODE 1 WRITE PROTECTED´,CR,LF,0
CD3:            db      ´CODE 3 DRIVE DID NOT DO IT´,CR,LF,0
CD6:            db      ´CODE 6 FILE MARK VER. ERR.´,CR,LF,0
CD7:            db      ´CODE 7 ABORT BEFORE DONE´,CR,LF,0
CD8:            db      ´CODE 8 H-E-R´,CR,LF,0
CD9:            db      ´CODE 9 BAD CRCC´,CR,LF,0
CD10:           db      ´CODE 10 SHORT REC.´,CR,LF,0
CD11:           db      ´CODE 11 BAD V. PAR.´,CR,LF,0
CD12:           db      ´CODE 12 R-A-W ERROR´,CR,LF,0
CD13:           db      ´CODE 13 WRITE FAIL´,CR,LF,0
CD14:           db      ´CODE 14 FILE MARK DET.´,CR,LF,0
CD15:           db      ´CODES 4,5,15 IMPROPER ABORT CODES´
                db      CR,LF,0
COMPSM:         db      ´TAPE DRIVE OK´,CR,LF,0
HEADER:         db      ´GPHEADER´
INTAPE:         db      ´INSERT TAPE CARTRIDGE´,CR,LF,0
NOCOMPM:        db      ´BAD TAPE OR DRIVE´,CR,LF,0
OT1:            db      ´OUT OF TAPE´,CR,LF,0
;REMTAPE:       db      ´REMOVE TAPE CARTRIDGE´,CR,LF,0
TERR:           db      ´TIME OUT ERROR´,CR,LF,0
TRACKNM:        db      CR,LF
                db      ´TRACK #´,0
TSTTAPM:        db      ´TESTING TAPE ´,CR,LF,0

;--- ERROR TABLE ---

JTABLEB:        dw      FLAG
                dw      PROTECTED
                dw      NODRIVE
                dw      NORESP
                dw      UNKNOWN
                dw      UNKNOWN
                dw      FMVER
                dw      TABORT
                dw      RFHER
                dw      RFCRCC
                dw      RFSHORT
                dw      RFBVP
                dw      WFRAW
                dw      WF
                dw      RFFMD
                dw      UNKNOWN
```

```
;--- DIAGNOSTIC MESSAGES ---

         IF      DIAG              ;Check DIAG flag
WAKMES:  db      'WAKE    ',0
SLPMES:  db      'ZZZZ    ',0
BOTMES:  db      'BOT ',0
EXEMES:  db      'EXEC CMD=',0
SD:      db      'DS=',0
SI:      db      'IS=',0
         ENDIF
```

PARAMETER ENTRY

```
;_______________________________________________________________
;Original code written for window operation by ohh
;UPDATE - 10/19/84 code modularized and commented by gkm
;UPDATE -  9/13/85 window routines were rewritten to :
;                1.make more compact
;                2.expand series and exp. nos.
;                3.add error trapping
;                                                  by gkm
;UPDATE - 11/15/85 event mode routines were added by gkm
;_______________________________________________________________
```

```
;--- INITIALIZE TIME AND TAPE ---

START:  PRINT   CLRSCR
        call    SETCLK          ;Set the clock
        call    STAPE           ;Install a tape in the drive
        call    TSTTAP          ;Test the tape

;--- GET HEADER INFO ---

;                       - THIS ROUTINE PRINTS THE STRING,
;                         MOVES IT TO HEADER MEMORY,
;                         GETS RESPONSE,AND MOVES IT TOO.
;                       - ALL REGS ARE SAVED.

WHEAD:  PRINT   CLRSCR          ;Clear the screen
        call    INIT_HDR_RAM
        ld      hl,(HDRBUF)
        WPRINT  DEPL            ;Get deployment
        WPRINT  INSTR           ;Get instrument #
        WPRINT  CHSCI           ;Get chief scientist
        WPRINT  CRUISE          ;Get cruise ID
        WPRINT  SPHERE          ;Get sphere #
        WPRINT  LAT             ;Get latitude
        WPRINT  LONG            ;Get longitude
        PRINT   FEG             ;Print gain message
        ld      hl,FEG          ;Point to message
        ld      de,(HDRBUF)     ;Point to header buffer
        call    MOVLP           ;Move it
        call    CHANNEL         ;Do it for all four channels
        PRINT   FED             ;Print damping message
        ld      hl,FED          ;Point to heading
        ld      de,(HDRBUF)     ;Point to buffer
        call    MOVLP           ;Move it
        call    CHANNEL         ;For all 4 channels
```

```
;                       - ADD TERMINATOR AND INSERT INTO GPHEADER.

        ld      a,CTLZ          ;ASCII buffer terminator
        ld      de,(HDRBUF)     ;Point to buffer
        call    INS             ;Put it in and fix DE

;                       - PLAY DATA BACK FOR VERIFICATION.

        call    CONT            ;Wait
        PRINT   CLRSCR          ;Clear the screen
        ld      hl,HDRRAM       ;Point to beginning of header
PBACK:  ld      a,(hl)          ;Play back until CTLZ
        cp      CTLZ            ;Look for terminator
        jr      z,ISOK          ;If so, then next
        call    PBYT            ;Print it
        inc     hl              ;Increment pointer
        jr      PBACK           ;Get next
ISOK:   QPRINT  OKQ             ;IS IT OK?
        or      20h             ;Force to lower case
        cp      ´n´
        jp      z,WHEAD         ;Do again
        cp      ´y´
        jp      z,SKGI          ;Start parameter entry
        jp      ISOK            ;If nothing compares, ask again
```

```
; EXPERIMENT PARAMETER ENTRY
;________________________________________________________________

;--- INITIALIZE THE SERIES NUMBER ---
;                        - CLEAR THE SERIES COUNT.

SKGI:    ld      hl,CSN          ;Point to current series number
         call    BCDCLR          ;Clear series count
         ld      de,SERBUF       ;Point to parameter storage
         ld      (PARBUF),de     ;Save it

;--- INCREMENT THE SERIES NUMBER ---
;                        - ONLY 9 SERIES SUPPORTED.
;                        - THIS LIMITATION IS DUE TO THE AMOUNT OF
;                          STORAGE ALLOCATED IN THE 256 BYTE TRAILER.
NSERIES:
         ld      hl,CSN          ;Get current series number
         call    BCDCT           ;Increment count
         ld      a,(CSN)         ;Get the series number
         cp      09h             ;Support only 9 series
         jp      z,EXECU         ;If max, then start

;                        - ENTER HERE ON RETRY.

REPENT:  PRINT   CLRSCR          ;Clear the screen
         call    SNPT            ;Print the series number

;--- ENTER NUMBER CHANNELS ---
;                        - SUPPORTS UP TO 4 CHANNELS.

NCHAN:   PRINT   NCAP            ;Print channel message
         ld      hl,NCX2         ;Point to channel storage
         call    INDAT           ;Get answer
         jr      c,NCHAN         ;Invalid entry if carry
         ld      a,4             ;4 to ACC
         cp      m               ;Compare to input
         jr      c,NCHAN         ;If input >4,ERROR
         ld      a,0             ;Check if entry is 0
         cp      m
         jr      z,NCHAN         ;If so, try again
```

```
;--- ENTER BASE CHANNEL ---
;                       - CHECKS TO ENSURE VALID ENTRY WITH
;                         RESPECT TO THE NUMBER OF CHANNELS.
;                       - THE BASE ADDRESS OF THE A-D IS
;                         CALCULATED AND STORED.
;
BCHAN:  PRINT   BCHAP           ;Print BASE CHANNEL?
        ld      hl,IO           ;Point to storage
        call    INDAT           ;Get it
        jr      c,BCHAN         ;If invalid, try again
        ld      a,(IO)          ;0 is invalid
        cp      0
        jr      z,BCHAN
        ld      a,(NCX2)        ;Get no. of channels
        dec     m               ;BASE CHANNEL-1
        add     a,m             ;Check for high channel <5
        cp      5
        jr      nc,NCHAN        ;If not, try again
        sla     (hl)            ;Multiply base adjustment by 2
        ld      a,ADPORT-1      ;Load normal starting channel
        add     a,m             ;Create correct base address
        ld      m,a             ;BASEPORT-1 computed and stored
                                ;in IO
        inc     hl              ;Now fix no. channels
        sla     (hl)            ;#CHANNELS*2
```

```
;--- ENTER BUFFER SIZE ---
;                       - ENTRY IS TESTED FOR VALIDITY.
;                       - RAM ADDRESSES FOR EACH ENTRY ARE STORED.

BSIZ:   call    CRLF            ;Print CRLF
        PRINT   BSIZP           ;Print HOW MANY 8K BLOCKS?
        ld      hl,BUFSIZ       ;Point to storage
        call    INDAT           ;Get it
        jr      c,BSIZ          ;If not valid, try again
        ld      a,m             ;Get answer
        cp      5               ;Check if entry exceeds table
        jr      nc,BSIZ
        ld      hl,BBTBL        ;Point to entry table
        ld      d,0h
        ld      e,a             ;Move entry to de
        add     hl,de           ;Add entry offset to table pointer
        ld      a,(hl)
        cp      0h              ;Check if valid entry
        jr      z,BSIZ
        ld      (BSTART),a      ;Save buffer start
        ld      hl,BSTBL        ;Point to buffer size table
        add     hl,de           ;Add entry offset
        ld      d,(hl)          ;Move it to de
        ld      e,0h

;                       - # OF SAMPLES IS CALCULATED AND STORED.

        ld      a,(BUFSIZ)      ;Get buffer size (in 8K blocks)
        ld      b,a             ;Move it to B
        xor     a               ;Clear A
BZLP:   add     a,8             ;load 8 into A
        daa                     ;Multiply AxB
        djnz    BZLP
        ld      (BSZSAV),a      ;Save it
        ld      a,(NCX2)        ;Get no. channelsX2
        call    DIVIDE          ;Divide to get #bytes/sample in DE
        ld      (MSAMPS),de     ;Save it
```

```
;--- SAMPLE RATE ---
;                       - ENTRY IS CHECKED.
;                       - AD CODE IS SAVED.
;                       - SHIFTER VALUE FOR RECORD TIME IS SAVED.

SAMRAT: call    CRLF            ;Print CRLF
        PRINT   SINTV           ;Print sample rate message
        ld      hl,SRATE
        call    INDAT           ;Get answer
        jr      c,SAMRAT        ;If not valid, try again
        ld      a,m
        cp      9
        jr      nc,SAMRAT       ;Check entry is within the table
        ld      hl,ADTBL        ;Point to AD entry table
        call    TBLAD
        cp      0               ;Check for valid entry
        jr      z,SAMRAT
        ld      (ADVAL),a       ;Save buffer start
        ld      hl,SHTBL        ;Point to shifter table
        add     hl,de           ;Add entry offset
        ld      a,(hl)          ;Save it
        ld      (SHIFTER),a

;--- PRINT WARNING MESSAGE FOR SAMPLE RATE ---
;                       - PRINTS WARNING TO ENSURE THAT THE PROPER
;                       RESISTOR HEADER IS INSTALLED ON THE ANALOG
;                       BOARD.

WARNM:  call    CRLF
        ld      a,(SRATE)       ;Get sample rate
        cp      8               ;Look for 8ms rate
        jr      nz,NAD8         ;8ms warning is a little different
        PRINT   WARN8           ;Print warning message for 8ms
        jr      NEXP
NAD8:   ld      hl,WNTBL        ;Point to warning table
        call    TBLAD           ;Find the right value
        ld      (GPCTRL),a      ;Cannot print unless in memory
        ld      hl,GPCTRL
        call    PBCD            ;Print value from table
        PRINT   WARN1           ;Print rest of message

;                       - PRINT RECORD TIME.

NEXP:   call    CRLF            ;Print CRLF
        call    RTIME           ;Print record time
        call    CRLF            ;Print CRLF
```

```
;--- ENTER WINDOW OR EVENT ---

EVENT:   PRINT   WE              ;Is it WINDOW or EVENT?
         call    GETLN           ;Get response
         ld      hl,IBUFF
         ld      a,(hl)
         or      20h             ;Force to lower case
         ld      (SERTYP),a
         cp      't'
         jr      z,WNEXP         ;If timer, go do that
         cp      'e'
         jp      z,PES           ;If event, go do that
         jp      EVENT           ;If not, entry not valid try again
```

```
; PARAMETERS SPECIFIC TO WINDOW OPERATION
;_______________________________________________________________

;--- ENTER NO. EXPERIMENTS ---
;                          - MAXIMUM NUMBER IS 9999.

WNEXP:   ld      hl,EXPN         ;Clear exp. number
         call    BCDCLR
         call    CRLF
         PRINT   NEXPP           ;Print NO. OF EXPERIMENTS IN SERIES
         ld      hl,EXPN         ;Point to parameter storage
         call    INDAT           ;Get response
WEXP4:   jr      c,WNEXP         ;If carry, then invalid entry

;--- ENTER START AND STOP TIME ---

         call    SAVTIME         ;Save the previous stop time for
                                 ;check
         call    RETIM           ;Enter start and stop time

;--- ENTER OFFSET ---
;                          - OFFSET CAN BE ANY WHOLE SEC LESS THAN 1 MIN.

WOFFSET: call    CRLF
         PRINT   WOFF1           ;Print message
         ld      hl,OFFSET       ;Point to storage
         call    INDAT           ;Get answer
         jr      c,WOFFSET       ;If error, try again
         ld      a,m             ;Load answer
         cp      60H             ;Check for valid response (<1min)
         jr      nc,WOFFSET      ;If not, try again

;--- ENTER PERIOD ---

WPERIOD: call    CRLF
         PRINT   WPER1           ;Print message
         call    MINALP          ;Calculate minimum window
         or      30H             ;Convert to ASCII
         call    PBYT            ;Print it
         PRINT   WPER2           ;Print message
         push    bc              ;Save min. window
         ld      hl,PERIOD       ;Point to storage
         call    INDAT           ;Get answer
         pop     bc              ;Restore min. window
         jr      c,WPERIOD       ;If error, try again
         ld      a,m             ;Load answer
         cp      b               ;Compare with B
         jr      c,WPERIOD       ;If too small,try again

         jp      SPITBACK
```

```
; PARAMETERS SPECIFIC TO EVENT OPERATION
;________________________________________________________________

;--- ENTER POST EVENT SAMPLES ---

PES:     ld      (SERTYP),a
         PRINT   PESP            ;Print POST- EVENT SAMPLES
         ld      hl,PESAMS       ;Point to storage
         call    INDAT           ;Get it
         jr      c,PES           ;If invalid, try again
         ld      a,m             ;Answer in ACC,look at prompt in
                                 ;WMESS.
         ld      de,(MSAMPS)     ;Get no. samples
         push    de              ;Save in HL
         pop     hl
         or      a               ;Clear carry
         rr      d               ;Divide DE by 2
         rr      e               ;DE is now 50% total no. samples
         cp      50H             ;For choice 3,DE=50%
         jr      z,PESDON        ;So done for choice 3
         or      a               ;Clear carry
         rr      d               ;Divide DE by 2
         rr      e               ;DE is  now 25% total no. samples
         cp      25H             ;(25%= choice 4)
         jr      z,PESDON        ;Done for choice 4
         cp      75H             ;NOTE:Carry cleared if equal
         jr      nz,PES1         ;If no jump, then choice 2
         sbc     hl,de           ;100%-25%=75%=CHOICE 2
         ex      de,hl           ;Load into DE
         jr      PESDON          ;Done for choice 2
PES1:    cp      87H             ;Compare and clear carry if =
         jr      c,PES           ;Zero not valid-
                                 ;make sure carry is clear
         jr      nz,PES          ;Not a valid answer
         rr      d               ;Divide by 2
         rr      e               ;DE=12.5%
         or      a               ;Clear carry
         sbc     hl,de           ;Subtract
         ex      de,hl           ;DE now has 87.5% for choice 1
PESDON:  ld      (PESAMPS),de    ;Store post event samples
```

```
;--- ENTER SHORT TERM AVERAGE ---

SHORTERM:
          PRINT    STA               ;Print message
          ld       hl,STASAV         ;Point to storage
          PRINT    QUES              ;Ask for reply
          call     GETLN             ;Get answer
          ld       de,IBUFF+1        ;Ignore decimal point
          call     BCDIN             ;Convert to BCD
          jr       c,SHORTERM        ;If error, try again
          ld       a,m               ;Load answer
          cp       5H                ;.05 SEC. ?
          jr       nz,P10            ;If not, go to next
          ld       a,10H             ;Load code for .05
          jr       SHOREND           ;Exit
P10:      cp       10H               ;.10 SEC. ?
          jr       nz,P25            ;If not goto next
          ld       a,20H             ;Load code for .10
          jr       SHOREND           ;Exit
P25:      cp       25H               ;.25 SEC. ?
          jr       nz,P50            ;If not, go to next
          ld       a,40H             ;Load code for .25
          jr       SHOREND           ;Exit
P50:      cp       50H               ;.50 SEC. ?
          jr       nz,SHORTERM       ;If not, error. Try again
          ld       a,80H             ;Load code for .50 SEC.
SHOREND:  ld       (ANVAL),a         ;Save result
```

```
;--- ENTER THRESHOLD ---

THRESHOLD:
          PRINT   THRSH            ;Print message
          ld      hl,THRSHSV       ;Point to storage
          call    INDAT            ;Get answer
          jr      c,THRESHOLD      ;If error, try again
          ld      a,m              ;Load answer
          cp      6H               ;6DB ?
          jr      nz,DB12          ;If not, goto next
          ld      a,1              ;Load code for 6 db
          jr      THREND           ;Exit
DB12:     cp      12H              ;12DB ?
          jr      nz,DB18          ;If not, goto next
          ld      a,2              ;Load code for 12 db
          jr      THREND           ;Exit
DB18:     cp      18H              ;18DB ?
          jr      nz,DB24          ;If not, goto next
          ld      a,4              ;Load code for 18 db
          jr      THREND           ;Exit
DB24:     cp      24H              ;24DB?
          jr      nz,THRESHOLD     ;If not, error. Try again
          ld      a,8              ;Load code for 24 db
THREND:   ld      b,a              ;Move it to B
          ld      a,(ANVAL)        ;Retrieve STA code
          or      b                ;Combine them
          ld      (ANVAL),a        ;Save result
```

```
;--- ENTER NO. EXPERIMENTS ---
;                         - MAXIMUM NUMBER IS 9999.

ENEXP:   ld      hl,EXPN          ;Clear exp. number
         call    BCDCLR
         call    CRLF
         PRINT   NEXPP            ;Print NO. OF EXPS IN SERIES
         PRINT   EEXPM
         ld      hl,EXPN          ;Point to parameter storage
         call    INDAT            ;Get response
         jr      c,ENEXP          ;If carry, then invalid entry
         ld      hl,EXPN          ;Get first byte
         xor     a                ;Clear A
         cp      m                ;See if first byte is 0
         jr      nz,ESTIME        ;Exit if not
         inc     hl               ;Get next byte
         cp      m                ;See if it is 0
         jr      nz,ESTIME        ;Exit if not
         ld      (hl),99h         ;Load byte for max
         dec     hl               ;Get back first byte
         ld      (hl),99h         ;Load this byte for max

;--- ENTER START AND STOP TIME ---

ESTIME:  call    SAVTIME          ;Save previous stop time
         call    RETIM            ;Enter start and stop times

         jp      SPITBACK         ;End of event parameters
```

```
;--- CLOCK ENTRY ROUTINES ---

;                         - START TIME.

RETIM:   PRINT    TIMNOW          ;Print TIME NOW
         call     GCLOCK          ;Read clock
         ld       hl,TIMEN        ;Point to buffer
         call     TIMOUT          ;Print it
         ld       a,(CSN)         ;Check if first series
         cp       01h
         jr       z,RETIM1        ;Don´t print last stop time if
                                  ;Series 1
         PRINT    TIM1M
         ld       hl,STOPTB       ;Point to last stop time
         call     TIMOUT          ;Print it
RETIM1:  PRINT    STFS            ;Print START TIME FOR SERIES
         call     SNPT            ;#
         ld       hl,STRTTB       ;Start time in 5 bytes,packed BCD
         call     TIMINP          ;Store and check it

;                         - CHECK AGAINST CURRENT TIME or
;                           PRECEDING STOP TIME.

         ld       hl,TIMEN        ;Point to buffer
         call     GCLOCK          ;Read the clock
         ld       hl,STRTTB       ;Point to start time
         ld       de,TIMEN        ;Point again to time buffer
         call     CTIM            ;Compare
         jr       c,NFST          ;If carry, then exit
         jr       BT              ;Error if here

NFST:    ld       de,STOPTB       ;Point to stop time
FST:     call     CTIM            ;Compare
         jr       c,TSOK          ;If OK, then exit

BT:      PRINT    IST             ;Print error message
         jr       RETIM           ;Try again

 ;                        - STOP TIME.

TSOK:    PRINT    STOP            ;Print STOP TIME
         PRINT    TFS             ;Print message
         call     SNPT
         ld       hl,STOPTB       ;Stored at STOPTB
         call     TIMINP          ;Get it
         ld       de,STRTTB       ;Point to start time
         call     CTIM            ;Compare
         ret      c               ;Stop time is later if  MINUS
         PRINT    IST1            ;Print error
         jp       RETIM1          ;Try again
```

```
;PARAMETER VERIFICATION ROUTINES
;_______________________________________________________________

;--- EVENT PARAMETER CHECK ---

SPITBACK:
         call    CRLF            ;Print CRLF
         call    CONT            ;Wait
         call    SPITP           ;Print common series info
         ld      a,(SERTYP)      ;Get Series type
         cp      'e'             ;Check if event mode
         jp      nz,WINDSPIT     ;If not, goto window routine

;                       - POST EVENT SAMPLES.

PESTO:   PRINT   PESPP           ;Print message
         PRINT   EQUALS          ;Print '='
         ld      hl,PESAMS       ;Point to value
         call    PBCD            ;Print value
         ld      a,87H           ;If post event is 87 add .5
         cp      m
         jr      nz,PESTY
         ld      a,'.'           ;Load a '.'
         call    PBYT            ;Print it
         ld      a,'5'           ;Load a '5'
         call    PBYT            ;Print it
PESTY:   ld      a,'%'           ;Load a '%'
         call    PBYT            ;Print it
         call    CRLF            ;Print CRLF

;                       - SHORT TERM AVERAGE.

SHORTSPIT:
         PRINT STA1              ;Print message
         ld      a,'.'           ;Load a '.'
         call    PBYT            ;Print it
         ld      hl,STASAV       ;Point to value
         call    PBCD            ;Print it
         PRINT   SECOND          ;Print 'seconds'
         call    CRLF            ;Print CRLF

;                       - THRESHOLD.

THRESHSPIT:
         PRINT   THRSH1          ;Print message
         ld      hl,THRSHSV      ;Point to value
         call    PBCD            ;Print it
         PRINT   DBP             ;Print message
         call    CRLF            ;Print CRLF
         jr      OKPARAM         ;Goto verify routine
```

```
;--- WINDOW PARAMETER CHECK ---

;                        - OFFSET.

WINDSPIT:
         PRINT   WOFF             ;Print message
         ld      hl,OFFSET        ;Point to value
         call    PBCD             ;Print it
         PRINT   SECOND           ;Print ´seconds´

;                        - PERIOD.

         PRINT   WPER             ;Print message
         ld      hl,PERIOD        ;Point to value
         call    PBCD             ;Print it
         PRINT   MINUTE           ;Print ´minutes´
         call    CRLF             ;Print CRLF
```

```
;--- VERIFY PARAMETERS ---

OKPARAM: QPRINT  OKQ
         or      20h              ;Force to lower case
         cp      ´y´              ;All OK?
         jr      z,PARSTOR        ;If so, store them
         call    RESTIME
         jp      REPENT           ;Start over if not
PARSTOR: ld      de,(PARBUF)      ;Point to buffer
         ld      hl,NSCRAM        ;Point to NSCRAM
PARMOV:  LDI                      ;Move data
         ld      a,l              ;Get low byte of parameter pointer
         cp      LOW(ENDPARA)     ;Compare
         jr      nz,PARMOV        ;If not end, do some more
         ld      (PARBUF),de      ;Save pointer
         call    CRLF
         QPRINT  NEXTSER
         or      20h              ;Force to lower case
         cp      ´n´
         jp      z,EXECU
         jp      NSERIES          ;Goto to next
         ret

;--- COMMON PARAMETERS ---

SPITP:   PUSHALL                  ;Save regs.
         PRINT   CLRSCR           ;Clear the screen
         call    SNPT             ;Print series no.
         call    CRLF
         PRINT   NEXPP            ;Next no. exp.
         ld      hl,EXPN          ;Point to storage
         call    BCDPT            ;Print it
         call    CRLF             ;Print CRLF
         ld      a,(SERTYP)       ;Get series type
         cp      ´e´              ;Check if event
         jr      nz,SWIND         ;If not, goto window routine
         PRINT   EVENTP           ;Print message
         jr      SPIT2            ;Goto next
SWIND:   PRINT   WPRMT            ;Print window message
```

```
SPIT2:    PRINT   STFS              ;Print 'start'
          ld      hl,STRTTB         ;Point to storage
          call    TIMOUT            ;Get time
          PRINT   STOP              ;Print 'stop'
          PRINT   TFS               ;Print message
          ld      hl,STOPTB         ;Point to storage
          call    TIMOUT            ;Get time
          call    CRLF              ;Print CRLF
          PRINT   ACHAN             ;Print no. active channels
          ld      a,(NCX2)          ;Get value
          rrca                      ;Shift it
          ld      b,a               ;Move it to B
          ld      a,(IO)            ;Get base channel
          sub     ADPORT-1          ;Remove port address
          or      a                 ;Clear carry
          rrca                      ;Divide by 2 (2 PORTS/CHAN)
          inc     a                 ;First channel =1 for user
          or      30H               ;Convert to ASCII
PCHLO:    call    PBYT              ;Print it
          push    af                ;Save acc.
          ld      a,','             ;Load a ','
          call    PBYT              ;Print it
          pop     af                ;Restore regs.
          inc     a                 ;Increment A
          djnz    PCHLO             ;For all channels
          call    BSPCR             ;Backspace
          call    CRLF              ;Pint CRLF
SPITBSIZ:
          PRINT   BSIZP1            ;Print buffer size
          ld      hl,BSZSAV         ;Point to storage
          call    PBCD              ;Print it
          ld      a,'K'             ;Load a 'K'
          call    PBYT              ;Print it
          call    CRLF              ;Print CRLF
SAMRATS:  PRINT   SINTV1            ;Print sample interval
          ld      hl,SRATE          ;Point to storage
          call    PBCD              ;Print it
          PRINT   MILSEC            ;Print 'ms'
          call    CRLF              ;Print CRLF
          call    RTIME             ;Print time
          call    CRLF              ;Print CRLF
          POPALL                    ;Restore regs.
          ret
```

```
; ENTRY SUBROUTINES
;____________________________________________________________________

;--- SAVE THE STOP TIME ---

SAVTIME:
        exx                     ;Save regs.
        ld      bc,10           ;Load count
        ld      de,TIMSAVE      ;Point to time temporary storage
        ld      hl,STOPTB       ;Point to stop time storage
        LDIR                    ;Move it
        exx                     ;Restore regs.
        ret

;--- RESTORE THE STOP TIME ---

RESTIME:
        exx                     ;Save regs.
        ld      bc,10           ;Load count
        ld      de,STOPTB       ;Point to stop time
        ld      hl,TIMSAVE      ;Point to temp storage
        LDIR                    ;Move it
        exx                     ;Restore regs.
        ret

;--- CALCULATE THE MINIMUM WINDOW PERIOD ---
;                          - THE MINIMUM PERIOD IS BASED ON THE
;                            AMOUNT TIME REQUIRED TO FILL THE BUFFER
;                            PLUS TIME TO WRITE DATA TO TAPE.
;                          - TIME TO WRITE TO TAPE IS BASED ON BUFSIZ.

MINALP: ld      a,(BUFSIZ)      ;Load buffer size
        sla     a               ;Shift it
        or      10H             ;Allow at least 10 sec for write
        ld      b,a             ;Move it to B
        ld      a,(RECTIME+1)   ;Load record time
        add     a,b             ;Add them
        daa                     ;Decimal adjust
        cp      59H             ;Check if greater than 1 min
        ld      b,0             ;Start minute count at 0
        jr      c,NOMIN         ;If less than 1 min. increment only once
        inc     b               ;Increment minute count
NOMIN:  inc     b               ;Again
        ld      a,(RECTIME)     ;Get record time
        add     a,b             ;add them together
        ld      b,a             ;Save it in B
        ret
```

```
;--- GET GAIN OR DAMPING FOR EACH CHANNEL ---
;                       - GETS 1 OR 2 BCD DIGITS, STORES THEM AT HL,
;                         AND CARRY FLAG IS SET IF IMPROPER ENTRY.

CHANNEL:
         WPRINT  CH1     ;Print channel1?
         WPRINT  CH2     ;Print channel2?
         WPRINT  CH3     ;Print channel3?
         WPRINT  CH4     ;Print channel4?
         ret

;--- COMPUTE AND DISPLAY RECORD TIME ---

RTIME:   push    af
         exx                     ;Save regs.
         PRINT   RTIMEP          ;Print message
         ld      de,(MSAMPS)     ;Get no. samples (max)
         ld      a,(SHIFTER)     ;Get AD value
         ld      b,a             ;Move it to B
SHLP:    sra     d               ;Divide by 4
         rr      e
         djnz    SHLP            ;Until shifter value goes to zero
         ld      a,60            ;Divide by 60 to get minutes
         call    DIVIDE
         ld      hl,RECTIME      ;Point to storage
         ld      m,e             ;Save minutes
         inc     hl
         ld      m,b             ;Save seconds
         dec     hl
         call    HTBCD           ;Convert it to packed BCD
         call    PBCD            ;Print the result minutes
         PRINT   MINUTE
         inc     hl
         call    HTBCD
         call    PBCD            ;Print the resulting seconds
         PRINT   SECOND
         call    CRLF
         pop     af
         exx
         ret

;--- PRINT SERIES # ---

SNPT:    PRINT   SN              ;Print "SERIES #"
         ld      hl,CSN          ;Point to the series number
         call    BCDPT           ;Print it
         call    CRLF
         ret
```

```
;--- TABLE LOOKUP ROUTINE ---
;                         - ALL REGISTERS ARE ALTERED.
;                         - ON ENTRY HL MUST POINT TO TABLE AND A
;                           MUST HAVE THE ENTRY.
;                         - ROUTINE EXITS WITH TABLE VALUE IN A,
;                           TABLE POINTER IN HL, AND THE TABLE
;                           OFFSET IN DE.

TBLAD:  ld      d,0h            ;Clear D for 16 bit addition
        ld      e,a             ;Move the entry to E
        add     hl,de           ;Add the entry to the table
        ld      a,(hl)          ;pointer
        ret
```

```
;CONTROL AND ACQUISITION ROUTINES
;________________________________________________________________

;--- SET UP SERIES NO. AND BUFFER ---
;                       - SAVE THE TOTAL NUMBER OF SERIES IN LSN.
;                       - START CURRENT SERIES WITH 0001.

CONTROL:
         ld      hl,LSN          ;Point to the last series number
         ld      de,CSN          ;Point to the current series no.
         ld      a,(de)          ;Get LSB of number
         ld      m,a             ;Save it in LSN
         inc     hl              ;Bump pointers
         inc     de
         ld      a,(de)          ;Get MSB of no.
         ld      m,a             ;Save it in LSN
         ld      hl,CSN          ;Point to current series no.
         call    BCDCLR          ;Clear it
         ld      a,1
         ld      (CSN),a         ;First series is #1
         ld      hl,SERBUF       ;Load start of series buffer
         ld      (SERPTR),hl     ;Save it

;--- INITIALIZE THE SERIES ---

NEWSER:  di                      ;Disable interrupts
         ld      a,1             ;Load 1
         out     (INTPRT),a      ;Send it to interrupt mask
         ld      (STOPT1),a      ;Stop timer 1
         ei                      ;Enable interrupts
         ld      hl,CEXPN        ;Clear the old exp no.
         call    BCDCLR
         STA     (CEXPN),1       ;First experiment is #1
         call    LDSER           ;Load series variables
         call    SPITP           ;Print them
         ld      a,(SERTYP)      ;Load series type
         cp      ´e´             ;Is it event mode?
         jr      z,EVENTX        ;If it is, setup event
         cp      ´t´             ;Is it timer mode?
         jr      z,WINDOW        ;If so, setup window
         jp      ENDPRO          ;If neither, then end it
```

```
;--- EVENT SETUP ---

EVENTX:  call    TCHECK          ;Look for start time
EVNT2:   ld      hl,EVNT1        ;Save abort vector
         ld      (ABRTV),hl
         call    INITW           ;Initialize ports and buffers
         xor     a               ;Clear A
         out     (AVGPRT),a      ;Enable long term average
         call    PEM             ;Print exp message
         call    EDOIT           ;Get data
EVNT1:   call    NEXTW           ;Increment exp no.
         jr      nc,EVNT2        ;Continue if not done
         ccf                     ;Clear carry flag
         call    INCSER          ;Increment series number
         jp      NEWSER          ;Get next series

;--- WINDOW SETUP ---

WINDOW:  ld      hl,WIN1         ;Save abort vector
         ld      (ABRTV),hl
         call    INITW           ;Initialize ports and buffer
         call    PEM             ;Print exp message
         call    TCHECK          ;Check time
         call    DOIT            ;Get data
WIN1:    call    NEXTW           ;Increment exp. no.
         jr      nc,WINDOW       ;Continue until done
         ccf                     ;Clear the carry flag
         call    INCSER          ;Increment series no.
         jp      NEWSER          ;Start new series

;--- LOAD SERIES PARAMETERS FROM THE SERIES BUFFER ---

LDSER:   exx                     ;Save regs.
         ld      hl,(SERPTR)     ;Get series pointer
         ld      bc,NOPARA       ;Load no. of parameters
         ld      de,NSCRAM       ;Point to NSCRAM
         ldir                    ;Move it
         ld      (SERPTR),hl     ;Save pointer
         exx                     ;Restore regs.
         ret

;--- PRINT EXPERIMENT MESSAGE ---

PEM:     PRINT   EXPM            ;Print message
         ld      hl,CEXPN        ;Point to experiment no.
         call    BCDPT           ;Print it
         PRINT   TM              ;Print track message
         ld      a,(IMA)         ;Load the track number
         AND     TRACKMA         ;Strip the mask
         or      30H             ;Convert to ASCII
         call    PBYT            ;Print it
         ld      a,CR            ;Load CR
         call    PBYT            ;Print it
         ret
```

```
;--- INITIALIZE BUFFER AND PORTS ---

INITW:  xor     a               ;Clear A
        ld      l,a             ;Move it to L
        ld      a,(BSTART)      ;Load starting address
        ld      h,a             ;Move that to H
        ld      (BUFPTR),hl     ;Save it as buffer pointer
        ld      (WBUFSAV),hl    ;Save it again
        ld      hl,(MSAMPS)     ;Point to no. samples (max)
        ld      (NSAMPS),hl     ;Save that
        ld      a,T1-INT-EN or EARLY_TERM
                                ;Load interrupt value
        di                      ;Disable interrupts
        out     (INTPRT),a      ;Send it out
        ei                      ;Enable interrupts
        ld      a,1             ;Load A
        ld      (PCCLRB),a      ;Clear port B of NSC810
        ret

;--- ACQUIRE WINDOW DATA ---
;                         - LOADS PORTS AND SETS INTERRUPTS.

DOIT:   ld      a,(ADVAL)       ;Load sample rate
        out     (ADPORT),a      ;Send it out
        ld      a,(ANVAL)       ;Load STA & THRES code
        out     (ANAPRT),a      ;Send it out
        di                      ;Disable interrupts
        ld      a,T1_INT_EN or AD_INT_EN or EARLY_TERM
        out     (INTPRT),a      ;Enable board interrupts
        ei                      ;Enable interrupts

 ;                        - LOOP UNTIL NSAMPS GOES TO 0.

        ld      hl,NSAMPS       ;Get no. samples
ALLAC? : xor    a               ;Clear A
        or      m
        inc     hl
        or      m
        jr      z,ACQUIRED
        bit     7,(hl)          ;JUST IN CASE OF STRANGE EVENTS
        jr      nz,ACQUIRED
        dec     hl
        jr      ALLAC?

;                         - RESETS INTERRUPTS AND WRITES DATA.

ACQUIRED:
        di                      ;Disable interrupts
        ld      a,T1_INT_EN or EARLY_TERM
        out     (INTPRT),a      ;Enable board interrupts
        ld      a,1             ;Load A
        ld      (PCSETB),a      ;Set Port B of NSC810
        ei                      ;Enable interrupts
        call    WDR             ;Write data
        ret
```

```
;--- ACQUIRE EVENT DATA ---
;                         - LOADS PORTS AND SETS INTERRUPTS.

EDOIT:   ld      a,(ADVAL)        ;Load sample rate
         out     (ADPORT),a       ;Send it out
         ld      a,(ANVAL)        ;Load STA & THRES code
         out     (ANAPRT),a       ;Send it out
         di                       ;Disable interrupts
         ld      a,T1_INT_EN or AD_INT_EN or EARLY_TERM
         out     (INTPRT),a       ;Enable board interrupts
         ei                       ;Enable interrupts

;                         - LOOP UNTIL NSAMPS GOES TO 0.

         ld      hl,NSAMPS        ;Get no. samples
ALLACE:  xor     a                ;Clear A
         or      m                ;Look bor both bytes NSAMPS
         inc     hl               ;to go to 0
         or      m
         jr      z,EGOT           ;If 0 then done
         bit     7,(hl)           ;JUST IN CASE OF STRANGE EVENTS
         jr      nz,EGOT
         dec     hl               ;Repoint to first byte
         jr      ALLACE           ;Try again

 ;                        - LOOK FOR AN EVENT.

EGOT:    in      a,(ANAPRT)       ;Look for an event
         rla                      ;Rotate bit 7 to carry
         jp      nc,EGOT          ;Continue until event

;                         - NOW QUALIFY THE EVENT.

         ld      a,14h            ;Event line must be high 20 events
         ld      (EQFY),a         ;Set counter
EGOT1:   in      a,(ANAPRT)       ;Look for event
         rla                      ;Rotate bit 7 to carry
         jp      nc,EGOT          ;No event
         ld      a,(EQFY)         ;Get count
         cp      0                ;Is it zero?
         jr      nz,EGOT1         ;If not,try again
         ld      a,1              ;Disable average
         out     (AVGPRT),a
```

```
;                      - GET POST EVENT SAMPLES.

         ld      hl,(PESAMPS)    ;Load no. post event samples
         ld      (NSAMPS),hl     ;Load counter
         ld      hl,NSAMPS       ;Point to counter
PELP1:   xor     a               ;Clear A
         or      m               ;Is it zero?
         inc     hl              ;Get next byte
         or      m               ;Is it zero?
         jr      z,EAQUR         ;If so, record data
         bit     7,(hl)          ;In case of strange events
         jr      nz,EAQUR        ;Then done
         dec     hl              ;Get back to PECTR
         jr      PELP1           ;Get some more

;                      - GET TIME AND RECORD DATA.

EAQUR:   ld      hl,(BUFPTR)     ;Get current pointer
         ld      (BEND),hl       ;Save it
         di                      ;Disable interrupts
         ld      a,T1_INT_EN or EARLY_TERM
         out     (INTPRT),a      ;Enable interrupts
         ld      a,1
         ld      (PCSETB),a      ;Set port B of NSC810
         ei                      ;Enable interrupts
         call    TIMREC          ;Get time
         call    EWDR            ;Write data
         ret
```

```
;--- INCREMENT THE SERIES # ---
;                         - CHECKS THAT RESULT IS LESS THAN LSN.

INCSER: push    af              ;Save acc.
        ld      hl,CSN          ;Increment series number
        call    BCDCT
        ld      de,LSN+1        ;Point to last series number
        ld      hl,CSN+1        ;Point to current series number
        call    BCDCP           ;Compare them
        jp      c,ENDPRO        ;If there, end it
        pop     af              ;Restore acc.
        ret

;--- INCREMENT THE BCD EXPERIMENT # ---
;                         - CHECKS THAT IT IS LESS THAN THE MAX.
;                         - ALSO CHECKS THAT NOT AT STOPTIME.
;                         - AF ALTERED.
;                         - CARRY IS SET IF AT MAX.

NEXTW:  call    NOTSTOP         ;Check if at stop time
        ret     c               ;End if carry
        ld      hl,CEXPN        ;Get exp. no.
        call    BCDCT           ;Increment it
        ld      hl,CEXPN+1      ;Point to current exp. no.
        ld      de,EXPN+1       ;Point to last exp. no.
        call    BCDCP           ;Compare them
        ret
```

```
; DEFINE STATEMENTS
;_______________________________________________________________

ACHAN:          db      'ACTIVE CHANNEL(S) = ',0

BCHAP:          db      'BASE CHANNEL  (1-4)',0

BSIZP:          db      'ENTER 1,2, or 4 blocks of 8K'
BSIZP1:         db      CR,LF,'RECORD SIZE  = ',0

CH1:            db      'CHANNEL 1 ',0

CH2:            db      'CHANNEL 2 ',0

CH3:            db      'CHANNEL 3 ',0

CH4:            db      'CHANNEL 4 ',0

CHSCI:          db      'CHIEF SCIENTIST ',0

CRUISE:         db      'CRUISE #        ',0

DEPL:           db      'DEPLOYMENT #    ',0

DBP:            db      ' db',0

EEXPM:          db      '(Enter 0 for maximum number) ',0

EQUALS:         db      '    =  ',0

EVENTP:         db      'EVENT MODE',CR,LF,0

EXPM:           db      'EXPERIMENT #',0

FED:            db      'FRONT END DAMPING',CR,LF,0

FEG:            db      'FRONT END GAIN',CR,LF,0

INSTR:          db      'INSTRUMENT #    ',0

IST:            db      'IMPROPER START TIME',CR,LF,0

IST1:           db      'IMPROPER STOP TIME',CR,LF,0

LAT:            db      'LATITUDE        ',0

LONG:           db      'LONGITUDE       ',0

MAX:            db      'MAXIMUM ',0

MINUTE:         db      ' min. ',0

MILSEC:         db      ' ms.',0
```

```
NCAP:            db      CR,LF,'# OF CHANNELS (1-4)',0

NEXPP:           db      '# OF RECORDS  = ',0
SN:              db      'SERIES # ',0

NEXTSER:         db      CR,LF,'DO YOU WISH ANOTHER SERIES',0

PESP:            db      CR,LF,'ENTER 87(.5), 75, 50, or 25%'
                 db      CR,LF
PESPP:           db      'POST-EVENT SAMPLE (%)',0

RTIMEP:          db      'RECORD TIME  = ',0

SECOND:          db      ' sec.',0

SINTV:           db      'ENTER  1ms, 2ms, 4ms, or 8msec '
SINTV1:          db      CR,LF,'SAMPLE RATE = ',0

SPHERE:          db      'SPHERE #        ',0

STA:             db      CR,LF,'ENTER .05, .10, .25, or .50 SEC.'
                 db      CR,LF
STA1:            db      'STA TIME CONST = ',0

STFS:            db      CR,LF,'START '
TFS:             db      'TIME  ',0

STOP:            db      'STOP  ',0

TIM1M:           db      'STOP TIME OF LAST SERIES ',0

TIMNOW:          db      'TIME NOW:   ',0

THRSH:           db      CR,LF,'ENTER 6, 12, 18, or 24 db.',CR,LF
THRSH1:          db      'THRESHOLD      = ',0

TM:              db      '    TRACK #',0

WARN8:           db      '164'
WARN1:           db      'K RESISTOR HEADERS ON FILTER BOARD FOR'
                 db      'THIS SAMPLE RATE'
WARNO:           db      CR,LF,'*** WARNING *** ',CR,LF,0

WE:              db      'TIMER or EVENT MODE(T/E) ',0

WOFF1:           db      'ENTER (0-59 sec.)'
WOFF:            db      CR,LF,'WINDOW OFFSET =  ',0

WPER1:           db      'ENTER (',0

WPER2:           db      '-99 min.)'
WPER:            db      CR,LF,'PERIOD of RECORDS = ',0

WPRMT:           db      'TIMER MODE',CR,LF,0
```

```
; TABLES
;_____________________________________________________________________

;                               - BUFFER SIZE TABLES

BBTBL:   db     00h,0e0h,0c0h,00h,80h
BSTBL:   db     00h,20h,40h,00h,80h

;                               - SAMPLE RATE TABLES

ADTBL:   db     0,2,6,0,1,0,0,0,5
SHTBL:   db     00h,0ah,09h,0,08h,0,0,0,07h


;                               - WARNING TABLE

WNTBL:          db      021h,041h,00h,082h

         END
```