EXTENDED IL ROM

THE HP-IL ENHANCEMENT MODULE

Another Product From

SKWID INK

EduCALC

CORRECTIONS

After EXT IL ROM was sent for manufacture the following difficulty was found.

SNEWM The SNEWM function will not work correctly on a medium which has not been previously formatted. The MEDM ERR message will come up.

To get around this problem use the NEWM function in the HP-IL module and then use SNEWM.

EXT IL ROM

User's manual

THE HP-IL Enhancement module

For use with the HP-41 Handheld computer

Foreword

With the advent of 80 column printers, and mass storage devices of larger capacities than the cassette drive, HP-IL has become quite versatile. However, the HP-41 has been left out in the cold. Trying to use the extended capacity of these devices has been difficult. Now, with the **EXT IL ROM**, it is possible to take, full advantage of the extended capacities offered by the latest HP-IL devices.

All of the write functions in the HP-IL module have been modified to allow their use on mass storage devices of up to **1 megabyte**. Functions such as NEWM and DIR have been updated and expanded to allow greater flexibility. the **EXT IL ROM** enables you to create three new types of mass storage files: an XMEM file for saving extended memory directly to mass storage, a CALC file allowing you to store away all of the contents of the calculator in one file, and the BUFF file type which allows you to save individual buffers. Still other functions allow you to determine the number of directory entries left on a medium, the number of records left on a medium, and provide the medium a name.

The **MCPRP** and **MCLIST** functions in the printer portion of **EXT IL ROM** allow you to produce multi-column listings of your programs. The **SACA** function permits you to send out strings, which have been stored in a buffer, to a printer using just one alpha character. All of the **EXT IL ROM** functions that interact with a printer (including **SDIR**) allow you to specify if you wish the data to be printed to an HP-IL printer or to a specified device on the loop. This feature allows you to print a listing of a directory of a mass storage medium to any type of device on the loop, whether or not it is actually a printer.

We believe that the mix of functions in **EXT IL ROM** will greatly benefit anyone who uses the HP-41 with HP-IL. Have fun on your adventures through **EXT IL ROM**.

Table of contents page

Table of contents page

EduCALC

27953 CABOT ROAD
LAGUNA NIGUEL, CA 92677
(714) 582-2637

INSTALLING EXT IL ROM

Before installing or removing **EXT IL ROM**, ensure that your HP-41 is turned off. If this is not done, damage may result to the module, the computer, or its operation may be disrupted. If you are removing **EXT IL ROM**, place a port cap over the unused port to protect it from dust and dirt.

EXT IL ROM may be plugged into any of the four ports at the top of your HP-41, although if a single density memory module is plugged into the calculator it must be in a lower numbered port than **EXT IL ROM**.

EXT IL ROM has the same XROM identity as the HP WAND (part number HP82153A). In order to use **EXT IL ROM** with the WAND plugged into the calculator, **EXT IL ROM** must be in a lower numbered port than the WAND. With this configuration, you will still be able to read in barcode using the WAND, but you won't be able to use any of the WAND ROM functions (WANDSCN, WANDSUB, etc.).

Definitions

This section shall define some terms that will be used throughout this manual.

HP-IL - Abbreviation of Hewlett Packard Interface Loop.

Loop - Another term used to describe HP-IL.

Program pointer - This is a register that contains the address of where you are in program memory. If you are at step 2 of a program then this register points to that address.

Device - A term used to describe a piece of equipment on HP-IL.

Mass Storage - Refers to the device used to store programs and data. These devices typically can store much more data than the memory of the computer. The HP-IL cassette and disc drives are examples of mass storage devices.

Medium -The physical object upon which the mass storage device saves data and programs. For example, this could be a disc or cassette tape.

Format - The setting of a medium to a predetermined state. This usually includes setting aside space for the directory and checking the integrity of the medium.

Volume Label - A six character alpha string that resides in bytes 2 through 7 of record 0. This is used to put a distinctive name on the medium. This name is the first line printed during an SDIR listing. It is not used by the 41 HP-IL module, but is used by EXT IL ROM and most other HP-IL controllers.

Bit - <u>B</u>inary dig<u>IT</u>. This is a single number which may have a value of either zero or one. It is like a switch, either on (equal to 1) or off (equal to 0).

Byte - Eight bits. The value of each byte may be from 0 to 255. Computers frequently break up their memory into bytes because of hardware restrictions.

Kilobyte - Used to describe 1,024 bytes of memory. Commonly abbreviated K as in 4K to mean 4,096 bytes.

Megabyte - This term is used to signify 1,048,576 bytes. This is 2 raised to the 20th power or 1024K.

Address - Each device on the loop is given a distinct numerical address.

Counting starts from 1 (the HP-41 is 0) and begins with the device that is connected to the male plug from the HP-IL module. This is the device with address 1. The device with address 2 is the one that is connected to the male plug from the device at address 1, and so on until the loop has come full circle back to the HP-41.

Accessory ID - AbbreviatedAID. This is a number given to an HP-IL loop device when it is manufactured. These may be in the range of 0 to 255. This range has been further subdivided into classes of 16 sets of numbers, i.e. 0 to 15 form one class (loop controllers), 16 to 31 (mass storage) form the next and so on. You may find the AID of a device by using the FINDAID function in the EXTENDED I/O ROM, and it is usually specified in the owner's manual for each device.

Buffer - Buffers are regions of memory below the .END. set up by individual ROM's to allow them to store data away from the user so that it may not be easily altered. There are no functions built into the calculator that allow you to create a buffer; they must be created by a plug in ROM. If the module that created the buffer is removed from the calculator and the 41 is turned on, the corresponding buffer is deleted.

Selected Device - This is the device on the loop from which the HP-IL module starts all searches for printers and mass storage devices. For example, if you have two printers in the loop at addresses 1 and 3, a cassette drive at address 2, and if the cassette drive is selected (using the SELECT function from the HP-IL Module), then all printing will occur on the printer at address 3 since it is the first printer after address 2. The same thing occurs with mass storage devices, the first one starting with the selected device is the one initially addressed. However with mass storage devices, if the file is not found on the first, it will search for additional drives until the named file is found or the last drive is checked.

MASS STORAGE FUNCTIONS

Before introducing the mass storage functions, here is a review of how the mass storage devices are structured. The medium of an HP-IL mass storage device is partitioned into sections of 256 bytes. Each of these sections is called a record. The record (256 bytes) is the smallest allowable space to be used on the medium. This means that if you save a program which is only twenty bytes long to a mass storage device it will use one record, or 256 bytes of space on the medium.

The medium of a mass storage device is formatted by the HP-41 in the following manner:

- First a command is sent to set the medium to all FF bytes.
- then the first two records are filled with 00 bytes.
- and finally the first twenty bytes of record 0 contain information about how large the directory is and the volume label.

The directory starts at record number two (counting of records starts with zero). Each directory entry takes up 32 bytes, so there is room for 8 directory entries per record. The last directory entry of the last directory record is not useable since the system uses this to mark the end of the directory. When the medium was formatted the number of directory entries you wanted was asked. The actual number of available directory entries may not be the same as the number you input since the number of directory entries is the number of records used by the directory times eight and then minus one. To calculate the actual number of entries from the number you input, use the following formula:

where H = the number you input and
R = the actual number of entries

$$R = [\mathrm{INT}((H+7)/8)]*8-1$$

When you create a file, it takes up one directory entry and a specified number of records on the medium. The physical address of the space on the medium used by the file specified by this directory entry cannot be changed. Even if the entry is purged, its space on the medium is still taken by the purged entry. When you try to create another file, purged entries are first searched in an effort to find one large enough to accommodate the new file. This is why a new file does not always end up at the end of the directory.

All **EXT IL ROM** functions which write to a mass storage device are able to write to the entire storage area of these devices (up to 1 megabyte). The write functions in the HP-IL module allow you to write to only 128K, no matter how large the medium actually is.

All of the **EXT IL ROM** mass storage functions have the following possible error messages. If the HP-IL module is not plugged into the 41 and you try to execute one of the mass storage functions the message "NO HPIL" will appear. If there is no mass storage device in the loop the "NO DRIVE" error will be displayed. If there is no medium in the drive a "NO MEDM" message is generated. A "TRANSMIT ERR" can also be generated if the loop is not connected correctly, or one of the devices in the loop malfunctions, or if the device is off. Error messages specific to each function will be discussed in the description of each function.

SNEWM (non-programmable)

This is a modified version of the NEW Medium function in the HP-IL module. It is nonprogrammable and when executed it displays four prompts. The A through J keys do not have any effect on the display; only the numeric keys and the backarrow key will alter the display. SNEWM also allows you to fill in the volume label by using the six leftmost characters in the alpha register. To use SNEWM, just place the characters you want to have as your volume label in alpha and execute SNEWM. Then just fill in the four prompts with the number of directory entries you want. The number of directory entries is right justified with any prompts that are not used being zeroed. For example, a directory size of 127 would be keyed in as 0127. If you make a mistake while keying in a digit, the backarrow key can be used to remove the digit just keyed in. If the number was the fourth and final digit then you must hold the key down until the "NULL" message comes up to abort the program. Pressing the backarrow key with 4 prompts in the display terminates the function.

Let's do an example. We shall format a cassette tape and specify a directory size of 25 and have a volume label of "HOSER1". To accomplish this do the following steps.

- Place "HOSER1" in alpha.
- Execute SNEWM
- Fill in the first two prompts with zeros and then a 2 and then a 5.

Now just sit back and wait for the tape to finish formatting. The additional possible error message for SNEWM is "DATA ERROR". This occurs when the number of directory entries specified exceeds the number allowed on the medium. For the cassette the maximum number of entries is 447, for a HP9114 disc drive this number is 2183, and for the Steinmetz and Brown SB10160 series disc drives it is 1271. These are the only HP-IL mass storage devices available at this time. However, EXT IL ROM will calculate the maximum number of directory entries for any mass storage devices that may be produced in the future.

Inputs for **SNEWM**

X: none
Alpha: Characters for the Volume Label.
Other: Fill in the prompts with the number of directory entries desired.

NAMEMED

For those of you who already have your mass storage medium formatted, here is a function that allows you to place a volume label on the medium without formatting it. To use the function NAME MEDium, just place the characters you wish to have as your volume label in the alpha register and execute **NAMEMED**. Only the six leftmost characters are used.

Let's rename the tape we just formatted to "HOSED". To do this just place "HOSED" in alpha and then execute **NAMEMED**. Now the volume label of the tape is "HOSED". Nothing else on the tape was changed.

Inputs for **NAMEMED**

X: none
Alpha: Characters to be placed in the volume label.

SCREATE

This function is the same as the CREATE function in the HP-IL module except that it is possible to create files of up to 65,535 registers in length if the medium you are using is large enough. The usage of **SCREATE** is the same as for CREATE: put the file name in alpha and the file size (number of data registers in the data file) in X. If you place a file size greater than 65,535 in X, a DATA ERROR will be generated. File sizes larger than this are not allowed because the file size bytes in the directory entry can only handle a number up 65,535 (this is a file size of half a megabyte). Another possible error is if you try to create a data file with the same name as one that already exists. If you try to do this the DUP FL NAME message will be shown. To remedy this situation either use a different name for the file you are trying to create or purge the file that is now on the tape using the PURGE function in the HP-IL module.

Let's create a data file of 256 registers on our newly formatted tape. The name of the file shall be "FRIEDMN". The maximum number of characters allowed for the name is seven. If you put more than seven characters in alpha only the leftmost seven will be used. To create the file do the following.

- Place "FRIEDMN" in the alpha register.
- Enter 256 in the X register.
- execute **SCREATE**

At the conclusion of this function, each register of the file has been zeroed, and the pointer has been set to register zero of the data file. To save data to this file use the WRTR and WRTRX functions in the HP-IL module. For instructions on their usage, consult the HP-IL module owners manual.

Inputs for **SCREATE**

X: Number of registers in data file.
Alpha: Name of the data file.

Each of the following write functions allows you to write over an existing file if it is of the same type and has the same name. So if you have a program saved under the file name "FINK" and you write another program file with the same name, the second will overwrite the first and the first will be lost forever. If the same name is used but the type is different (say you tried to save a write all file under "FINK") then the DUP FL NAME error message is generated.

SWRTA

This function works exactly the same as the WRTA function in the HP-IL module. SWRTA writes the main memory of the calculator to the mass storage device. The memory written does not include extended memory. The length of the file is 336 registers (this is for a 41CV or 41CX, it will be less if you don't have a full complement of memory in your 41C). To use SWRTA, just place the name of the file in alpha and execute **SWRTA**.

For example, we will save the main memory of the calculator under the file name "HORNE". To do this follow these steps.

- place "HORNE" in the alpha register
- execute **SWRTA**

You may come up with either of the following error messages, ROM, indicating that the program pointer in a ROM program, or PRIVATE, telling you there is a private program in memory. To place the program pointer into RAM, execute a CATalog 1. You are not allowed to save private programs to mass storage so you must delete the private program to save the rest of memory.

If you want the calculator to automatically start executing a specified program when this file is read back using the READA instruction of the HP-IL module, just set flag 11 and go to the program step where you want program execution to start. If you choose to do this, then an "A" will be appended after "WALL" during a listing of the directory when using SDIR to tell you that the file is an autostart file.

To accomplish this next example place the following program in memory.

```
01*LBL "CARTER"
02 "THIS IS AN"
03 AVIEW
04 PSE
05 "AUTOSTARTING"
06 AVIEW
07 PSE
08 "WRITE ALL FILE"
09 AVIEW
10 END
```

Now, from the keyboard while in run mode, do a GTO "CARTER" and set flag 11. We shall save the file under ERBAS. So follow these steps.

- Place "ERBAS" in alpha.
- Execute SWRTA

Now preform a MEMORY LOST by holding down the backarrow key as you turn the calculator on. To retrieve the file just saved.

- Place "ERBAS" in alpha.
- execute READA

The file will be read in, you will hear a beep and then the "CARTER" program will start executing.

Inputs for SWRTA

X:	none
Alpha:	Name of write all file.
Other:	You may set flag 11 to have the file automatically start executing a program when the file is read back into the calculator.

SWRTK

The SWRTK function will write the key assignments of catalog 2 and 3 functions out to a mass storage device. The assignments that are saved do not include assignments of programs in user RAM. To use this function, place the name of the file in alpha and execute SWRTK.

For this example clear all of your key assignments. Now make the following assignments, **SWRTP** to the 1/X key, **CLRBUF** to the SQRT key, and **WRTCAL** to the LOG key. For information on how to make user key assignments consult the calculator owners manual under the ASN function. We shall save these assignments under the name "WHITE".

- place "WHITE" in alpha.
- execute **SWRTK**

Now the key assignments have been saved to mass storage. When they are read back, using READK, the assignments of catalog 2 and 3 functions that were in the calculator are deleted and replaced by the new assignments; they are not merged with the new assignments. If there is not enough room in the calculator to hold all of the key assignments from the file you are reading in the "NO ROOM" error message will be generated and nothing will be changed.

Inputs for **SWRTK**

X: none
Alpha: name of key file

SWRTP
SWRTPV

These functions replace the WRTP and WRTPV functions in the HP-IL module. Their use is the same as their HP-IL module counterparts. To use them, place the name of the program you wish to save in alpha and execute SWRTP. If you wish to save the program under a name other than the name of the program, then follow the program name with a comma and then the file name you wish the program to be saved under.

As an example, let's save the following program under the name "WOPOUT".

```
01*LBL "HELP"
02 "EXT IL ROM"
03 AVIEW
04 PSE
05 "IS THE GREATEST"
06 AVIEW
07 END
```

Since the program name is different from the desired file name we will do the following.

- Place "HELP,WOPOUT" in the alpha register. This is the program name followed by a comma and then the file name. If the program pointer was positioned to the HELP program, then we could just place ",WOPOUT" in

alpha to get the same result.
- Now execute **SWRTP**, and the program will be saved.

Possible error messages are, ROM, you are trying to save a program that is in a plug-in module, NONEXISTENT, indicating you tried to save a function and not a user code program, and PRIVATE, to remind you that private programs may not be saved to a mass storage device.

If you wish to save the program and make it private so it can no longer be singlestepped or altered, just use the **SWRTPV** function. The steps for using this function are exactly the same as for **SWRTP**. Any program saved using SWRTPV will have a "P" appended after "PRGM" during an **SDIR** listing so you can tell that it is a private program.

Inputs for **SWRTP** and **SWRTPV**

X: none
Alpha: Name of the program file.

SWRTS

This function writes the status of the calculator out to the mass storage medium. The status includes the current size, the position of the sigma registers, the state of flags 0 to 43, the contents of the X, Y, Z, T, LASTX, and alpha registers. The length of the file created by this function is always 10 registers.

For this example we shall need to do the following.

- Using the SIZE function set the number of data registers to 069.
- using ΣREG set the sigma registers to 36.
- set flags 0, 2, and 4
- set the display to FIX 7.

We shall save the new status under the file name "HOOPED".

- place "HOOPED" in alpha
- execute **SWRTS**

Now clear flags 0, 2, 4, and set flag 1. Set the size to 000. Set the display mode to FIX 0. Now read in the "HOOPED" file using the READS function of the HP-IL module. To do this place "HOOPED" in alpha and execute the function. Now the calculator will be back to the same state as when the file was saved.

If there is not enough room for the calculator to accommodate the size specified by one of these files when it is read back in the "SIZE ERR" message is generated and the number of data registers in the calculator is not changed. However, everything else is set to the specifications in the status file.

Inputs for **SWRTS**

X: none
Alpha: Name of the status file.

Digression

Now we have come to the real juicy part of **EXT IL ROM**, the new write functions. These functions allow you to save portions of calculator memory that cannot be saved using the HP-IL module. This includes extended memory by itself, the entire calculator (including main memory and extended memory), and individual buffers. Functions have also been included to allow you to read each of these file types back in. And now on with the manual!

WRTBUFX
READBUF

The READ BUFfer and WRiTe BUFfer by X functions allow you to read and write a buffer to and from a mass storage device. These buffers are stored in the calculator memory just above the key assignments and below the .END.. In order to save one of these buffers, the name under which the buffer is to be saved must be in alpha and the buffer ID number must be in X. A list of known buffers and their ID's is given in the explanation of the **CLRBUF** function on page 17 . If the buffer you are trying to save is not present in the calculator, then the error message "NO BUFFER" comes up. If the buffer exists, and there is enough room on the medium, it will be saved. However, the number of registers displayed by **SDIR** function is one more than the buffer actually uses. Therefore, if the number under the REGS column of the directory listing is 29, the buffer actually only uses 28 registers of memory in the calculator. The extra register is used to store the type of buffer that is saved (it is used by the **READBUF** function). Since there is no way for you to tell the buffer ID from a directory listing, it is recommended that you only use six characters for the name and the last one to remind you of the buffer ID. Let's do an example.

You have some time alarms you wish to save since the program you are attempting to use can cause MEMORY LOST. We shall save them under the name "JARETTA". The first six letters are the name and the last is the hexadecimal equivalent of 10 (in hex 10 = A, 11 = B, and so on, up to 15 which is F). To save any time alarms do the following:

- Place "JARETTA" in alpha

- Put 10 in X
- Execute **WRTBUFX**

If there were no alarms to be saved, the message "NO BUFFER" would be generated.

To use the **READBUF** function, just place the name of the buffer file into alpha and execute **READBUF**. You do not have to place the buffer ID into X. If there is not enough room in the unused portion of memory, the "NO ROOM" error message comes up. If a buffer of the same ID as the buffer you are trying to read in already exists in the calculator, the error message "BUF EXISTS" is generated. In this case, the buffer ID that you are trying to retrieve is placed into X and the contents of the stack are rolled up with T being lost. To finish off our example, place "JARETTA" in alpha and execute **READBUF**. If the old time module buffer did not get deleted, the "BUF EXISTS" error message will be generated and a 10 will be placed in X. If this happens, just execute **CLRBUF** followed by **READBUF** again, and the timer buffer will reappear.

WARNING If you save away the time module buffer it is possible that when it is read back some of the alarms have become past due. They will be serviced (started beeping etc.) when the calculator is turned off. To service them immediately cycle the calculator off then on or execute the ALMNOW function in the time module and the pending alarms will be updated. For more information on past due alarms consult the time module owners manual.

Input for **READBUF** and **WRTBUFX**

X: ID number of the buffer to be saved (**WRTBUFX** only)
Alpha: Name of the buffer file.

WRTXM
READXM

These functions allow you to specifically save extended memory out to mass storage. Only the portion of extended memory that is being used is saved. These functions do not operate on individual files in extended memory, only on the entire contents.

To use **WRTXM**, just place the file name (up to seven letters) in alpha and execute **WRTXM**. If there is no extended functions module in the calculator, the "NO XFM" error message in displayed. If there are no files in extended memory, then the "DIR EMPTY" message comes up.

For this example clear your extended memory so we can load in two files. The first will be a 43 register data file with a name of NEMO and the second will be an ASCII file with a length of 69 registers whose name will be RASCAL. To accomplish this do the following.

- Place "NEMO" in alpha
- put 43 in the X register
- execute the instruction CRFLD from the extended functions module
- Place "RASCAL" in alpha
- put 69 in X
- execute CRFLAS in the extended functions module

Now when you execute the EMDIR function you should see the following:

```
NEMO  D043
RASCALA069
```

Upon completion of this function the number of registers available for use in extended memory will be left in X. To calculate the total number of registers used in extended memory use the following formula.

$$\text{total regs.} = \left(\sum (\text{file length} + 2)\right) + 1$$

This number will be the length of the file saved on the mass storage medium. Let's save extended memory under the name "HOOPED".

- Place "HOOPED" in alpha
- execute **WRTXM**

Now wait a minute, the error message "DUP FL NAME" came up, how come? It happened because we had a different file type (a status file) saved under this name. To save this file let's use the name PLG.

- place "PLG" in alpha
- execute **WRTXM**

Now extended memory has been saved away. To retrieve the file just place the name in alpha and execute **READXM**. If there is enough room to accommodate the whole file it will be read in and the old contents of extended memory will be lost. When **READXM** is finished successfully executing the working file is the same file as when extended memory was saved away using **WRTXM**. If there is not enough room to accommodate the whole file the "NO ROOM" error is generated. If there is no extended functions module in the machine then a "NO XFM" error message is displayed.

Inputs for **WRTXM** and **READXM**

X: none
Alpha: Name of the XMEM file.

WRTCAL
READCAL

The READ CALculator and WRite CALculator functions permit you to save all memory, both main and extended, out to a mass storage device. This allows you to save several different calculator setups on a medium and then retrieve them in an instant (instead of having to reload everything file by file). To use **WRTCAL** just place the name you want for the CALC file and execute **WRTCAL**. No other inputs are necessary. The amount of space used by one of these files is 337 registers plus the number of registers being used in extended memory. Thus if you have 600 registers of extended memory but are only using 50 of them, then only 50 will be saved (not all 600). During **WRTCAL**, if there are no files in extended memory, then the "DIR EMPTY" error message will come up. If there isn't an extended functions module present in the calculator the "NO XFM" error message is displayed. If this happens the main memory of the calculator is not saved, you should use **SWRTA** to do that. The "ROM" message is displayed if the program pointer is pointing to a ROM program and the "PRIVATE" error message comes up if there is a private program in memory. Suggestions on how to get arround these errors see **SWRTA**.

For our example we shall save the extended memory files that were saved in the example under **WRTXM** along with the rest of the calculator. They will be save under the name ABE.

- Place ABE in alpha
- execute **WRTCAL**

Now perform a MEMORY LOST on the calculator. Place ABE in alpha and execute **READCAL**. The entire contents of the calculator will be retrieved, including extended memory. The "NO ROOM" error message will come up if there is not enough room to store the main memory portion of the file, (this could only happen if you have a 41C that doesn't have 320 registers of main memory), or if there is not enough extended memory to contain the extended memory part of the file.

If you want to have a program automatically start executing when the CALC file is read back in, just set flag 11 and position the program pointer to the location where you want to start executing the program and then write the calculator to mass storage using **WRTCAL**. Instructions for doing this are the same as for **SWRTA**.

Inputs for **WRTCAL** and **READCAL**

X:	none
Alpha:	Name of the file
Other:	Set flag 11 if you wish to have the file automatically start executing a program when the file is read back in.

SDIR

The **SDIR** function is a highly enhanced version of the HP-IL module DIRectory function. It recognizes three new file types and the names of each type have been expanded to four characters. The following file types are recognized:

HP File Type	**EXT IL ROM** File Type	
AS	ASCI	ASCII file type
KE	KEYS	Key assignment file type
WA	WALL	Write all file type
ST	STAT	Status file type
DA	DATA	Data file
PR	PRGM	Program file type
??	????	File type not recognized by 41

and the three new file types

??	XMEM	**EXT IL ROM** extended memory file type
??	CALC	**EXT IL ROM** calculator file type
??	BUFF	**EXT IL ROM** buffer file type

When you execute **SDIR**, the first thing to appear in the display is the volume label. If this was not specified, then the display will be blank and a blank line will be printed if a printer is in the loop.

Then comes the heading "NAME   TYPE   REGS". After this, the files are listed in much the same way as the DIR function, except the file type is 4 letters instead of 2. When the last directory entry has been printed, the number of unused directory entries and the number of unused records are listed.

SDIR may be interrupted by pressing either the R/S or ON keys. All other keys have no effect on the function.

As our example let's list out what is on our sample tape so far. The number of records left may be different if you are not using a cassette drive, but everything else will be the same (if you followed all of the examples). The listing is at the top of the next page.

Another feature of **SDIR** is the ability to send the directory to any device on the loop. This is done by specifying the accessory ID of the device to which **SDIR** will print the directory. You can do this by using the **PRTAID** function (explained on page 19). If the specified device does not exist on the loop, then no printing is attempted.

```
HOSED                            Volume label
NAME     TYPE      REGS          Column labels
FRIEDMN DATA        256
HORNE    WALL       336
ERBAS    WALL,A     336
WHITE    KEYS         2
WOPOUT   PRGM         6
HOOPED   STAT        10
JARETTA BUFF          7
PLG      XMEM       117
ABE      CALC       454
DIR ENTRIES LEFT     16          Number of unused directory entries
RECORDS  LEFT       453          Number of unused records
```

Inputs for **SDIR**

X: none
Alpha: none
Other: May specify which device is the printer by using the **PRTAID** function.

DIRSIZE

DIRectory SIZE places the largest possible number of directory entries into the X register. To use **DIRSIZE** just execute the function and the total possible number of directory entries will be placed in X. The old X register is pushed down to Y, Y goes to Z, and the old Z ends up in T. T is lost.

Inputs for **DIRSIZE**

X: none
Alpha: none

DIRLEFT

The DIRectory entries LEFT function will place into X the number of possible entries that may be added to the medium in the mass storage device. This number includes all entries that have not been used as well as all entries that have been purged. To use this function no input is necessary; just execute **DIRLEFT** and the number of remaining directory entries will be placed in X. Stack usage is the same as for **DIRSIZE**.

Here is a program that may be used to determine the number of directory entries that are currently being used.

```
01*LBL"DIRUSED
02 DIRSIZE
03 DIRLEFT
04 -
05 "DIR USED "
06 FIX 0
07 ARCL X
08 AVIEW
09 END
```

Step 02 gets the total number of possible entries. Step 03 pushes the answer from the step 02 into Y and gets the total number of entries not used into X. These two numbers are then subtracted from one another to obtain the number of directory entries being used. The remaining steps output the answer to alpha.

Inputs for **DIRLEFT**

X: none
Alpha: none

RECLEFT

RECords LEFT calculates the number of unused records on the medium and places this number in X. The number calculated by this function is the number of records not yet used plus the number of records from entries that have been purged. Stack usage and possible errors are the same as for **DIRSIZE** and **DIRLEFT**. To use this function just execute **RECLEFT** and the number of records avaìlable for storage will be placed in X.

WARNING The number given by this function may be non zero and you could still end up with the "MEDM FULL" error message. This is because the unused records are in purged directory entries and none of the spaces allotted for these entries is large enough for the file you are trying to create.

Inputs for **RECLEFT**

X: none
Alpha: none

SCOPYFL

This function allows you to copy files between two different mass storage devices. The file to be copied must be a type that the 41 recognizes. Therefore, you cannot copy HP-71B basic program files using **SCOPYFL**. If you try to do this the message "FL TYPE ERR" will appear. If you try to copy a private program file, the message "PRIVATE" is displayed. In order to copy files, the mass storage device with the source file must be the selected device. The address of the mass storage device to which the file will be copied must be in X, and the name of the file to be copied is placed in alpha. If the number in X is greater than 30, or equal to zero, the "ADR ERR" error message is generated.

To do this example you will need two mass storage devices. Place them both in the loop. Select the one which will have the source tape in it using the SELECT function in the HP-IL module. Now place the destination tape in the other drive. To ensure that you have selected the correct device execute the **SDIR** function. If the directory listing is from the source drive then the correct device is selected. If it is not, then select the other drive. Place the name of the file you wish to copy in alpha and place the address of the destination drive in X. Now execute **SCOPYFL**. If there is not enough room on the destination medium for the file the "MEDM FULL" message will come up. If the directory is full the "DIR FULL" error message is generated.

Inputs for **SCOPYFL**

X: Address of the destination drive
Alpha: Name of the file to copy
Other: The source drive must be the selected device.

X>AR

This function is similar to the XTOA function of the extended functions ROM. It takes the number in X and converts it into an alpha character which is appended as the last character in alpha. The number in X may be from 0 to 255. If it is greater than 255 the "DATA ERROR" message will be brought forth. **X>AR** will generate an "ALPHA DATA" error if there is alpha data in X.

Inputs for **X>AR**

X: Number of the character to be placed in alpha.
Alpha: none

CLRBUF

The CLeaR BUFfer function allows you to clear a buffer from the calculator and thereby provide room for more programs, data, or key assignments. These buffers reside in calculator memory just above the key assignments. They are used by various ROM's to hold data that is safe from manipulation by the user. The most famous of these buffers is the one used by the TIME module to hold information relating to any alarms. Below is a list of all ROM's we know of that create buffers and their buffer ID's.

ROM	Buffer ID	Used for
David Assem	1 and 2	To save pointers, status, and labels.
CCD	5	Matrix and random number functions.
HP Advantage	5	same as CCD.
EXT IL ROM	6 and 7	Store printer AID and text strings (buffer #6). Used during **MCLIST** and **MCPRP** (buffer #7).
Time	10	Save alarms.
Plotter	11	Store plotter information.
HP-IL Dev.	12	Input and output of frames from HP-IL.
CMT	13	CMT measurement system.
DATAFILE (ES-41)	14	To save data file pointers.

To use this function just place the ID number of the buffer you wish to clear in the X register and execute **CLRBUF**. No error message is generated if the buffer did not exist before **CLRBUF** was executed. If the buffer ID specified in X is zero or greater than 14 then "DATA ERROR" will be displayed. If X is greater than 999 then the "NONEXISTENT" message comes up.

Let's try an example.

You need some memory so you can read a program in from your HP-IL cassette drive. You know there are alarms being saved by the time module. However the reading in of the program take precedence over these alarms and you wish to eliminate them. Since the time module buffer is number 10, you would place 10 in X and execute **CLRBUF**. More free registers would then appear, allowing you to read in the program.

Here is a short program for clearing all buffers in the calculator.

```
01*LBL "CLRB"
02 1.014
03*LBL 01
04 CLRBUF
05 ISG X
06 GTO 01
07 END
```

The program puts a counter in X and then deletes the buffer with an ID of 1 if it existed. Remember, **CLRBUF** does not give an error if the buffer did not exist before you tried to clear it. Also, **CLRBUF** ignores the fractional part of X. Step 03 increments X and will skip the GTO when we reach 15, otherwise the program loops back to clear the next buffer.

Inputs for **CLRBUF**

X: ID number of the buffer to be cleared.
Alpha: none

The last five functions are from the printer portion of **EXT IL ROM**. The possible error messages given by these functions will be discussed with each function.

PRTAID

The function PRinTer Accessory ID allows you to specify the device you wish to receive the data output by **EXT IL ROM** functions which interact with a printer. In effect it allows you to have a manual I/O mode without actually being in manual I/O mode (for more information on manual I/O consult the HP-IL owners manual under the MANIO function). Using **PRTAID** in combination with **SDIR**, you may print a directory listing to any device on the loop. The routine for finding the specified device obeys manual I/O mode; i.e., it will send the data to the selected device if you are in MANIO mode, otherwise it will look for the specified device in the loop.

As an example, let's say you have an HP82166A converter which you hook up to your Centronics printer. You have wanted to list the directory of your cassettes on this printer but have been unable to do so using functions from the HP-IL module. However, **EXT IL ROM** comes to the rescue. Just specify the AID of the converter (64) in X and execute **PRTAID**. Now when you execute **SDIR**, the directory will be printed on the printer connected to the converter even if there is an HP-IL printer in the loop. If the converter is removed from the loop and you execute **SDIR** once again, no printing will occur since **SDIR** believes you only want printing sent to the converter. To permit printing to an HP-IL printer, just execute **PRTAID** with zero in X. If the specified printer AID is zero, then we assume you want all **EXT IL ROM** printer output to go to an HP-IL printer.

PRTAID creates a buffer, and stores the AID number in it. If the **EXT IL ROM** buffer already exists then the new AID is placed in the buffer and the rest of the buffer is left unchanged. If **PRTAID** must create a buffer it will take two registers. If there are not two registers available then the "PACKING" followed by "TRY AGAIN" error messages will be displayed providing the function was executed from the keyboard. In a program the "NO ROOM" error message would be displayed. If the AID specified in X is greater than 255 "DATA ERROR" will come up. If this number is greater than 999 the "NONEXISTENT" message is put into the display. Executing **PRTAID** with zero in X does not delete the **EXT IL ROM** buffer from memory. To do this you must execute **CLRBUF** with 6 in X. The only way to allow **EXT IL ROM** functions to print on the first printer in the loop is to clear the **EXT IL ROM** buffer or to execute **PRTAID** with zero in X.

Inputs for **PRTAID**

X: Accessory ID of the device you wish to have all printing sent to when using **EXT IL ROM** functions. 0 clears the AID and allows you to use **EXT IL ROM** functions with the first HP-IL printer in the loop.
Alpha: none

ATOBUFX
SACA

Alpha TO BUFfer by X allows you to place string of up to 24 alpha characters into the **EXT IL ROM** buffer for use later by the function SACA. Each of these character strings is identified by a number from 1 to 31. This number is placed in X when **ATOBUFX** is executed and is used to identify each character string. In order to use this function just place the characters you want in the string into alpha, place the string ID number in X and execute **ATOBUFX**. The string will be placed into the **EXT IL ROM** buffer and be ready for use by **SACA**. If a string with the ID in the X register already exists in the **EXT IL ROM** buffer it is deleted and the new alpha string is put in its place. When alpha is empty the string is just deleted. If the number in X is 0, or greater than 31, "DATA ERROR" will be displayed. The buffer may need to be expanded if there is not enough room in the current buffer for the whole character string. In this case the "NO ROOM, "PACKING" "TRY AGAIN" error messages could show up if there are not enough free registers. Now let's do an example.

You are printing a chart on your ThinkJet printer and continually have to change back and forth between expanded and normal print pitches. Additionly everything printed in expanded print needs to be underlined. To do this using the **ATOBUFX**, **SACA** combination you would need two character strings, one for turning on expanded print and underlining and one for turning everything off. We shall place the string to turn on everything under ID 5 and the other string under ID 6. After careful examination of the owners manual for the ThinkJet (the codes are on the back cover) you find out that the following character strings will do the trick.

Expanded print: 27, 38, 107, 49, 83
Normal print: 27, 38, 107, 48, 83
Underline on: 27, 38, 100, 68
Underline off: 27, 38, 100, 64

The numbers given are the ASCII decimal equivalents of the characters needed in the alpha register. First clear the alpha register, now place the characters in alpha using the **X>AR** function. To get the first string (the one with ID 5) into alpha do the following:

- Place 27 in X and execute **X>AR**, then put 38 in X and execute **X>AR** again. Now put 107 in X and **X>AR** another time, then do the same with 49, 83, 27, 38, 100, and 68. You now have the escape codes for turning on expanded print and underlining in alpha.
- Place 5 in X.
- Execute **ATOBUFX.**

The character string to turn on expanded print and underlining is now in the **EXT IL ROM** buffer under an ID of 5. Now clear alpha and place the following characters into alpha using the **X>AR** function: 27, 38, 107, 48, 83, 27, 38, 100, 64. These characters must be placed into alpha in the order given or they won't function correctly. Place 6 in X and execute **ATOBUFX.** Now the character string to turn everything off is in the **EXT IL ROM** buffer under an ID of 6. Whenever **SACA** comes across the 5 character or the 6 character it will instead send out the character strings associated with those numbers.

The **SACA** function is similar to the ACA function in the HP-IL printer ROM. Both accumulate characters from the alpha register into the buffer of a printer. However, this is where the similarities end. **SACA** does not change character 13 (the angle character) to character number 124. Since character 13 is the carriage return byte for printers, this made it impossible to send this character using ACA. Also the 124 byte is not the character for an angle sign on most printers (the HP82162A printer being the sole exception). **SACA** does not change the sigma character (126) to character 28 as ACA does. Again, this is because most printers don't have the Σ character in their character set. ACA does not let you send out any bytes greater than 127. If you tried to do this the character would first have 128 subtracted from it and then be sent out. **SACA** allows you to send out characters greater than 127. When **SACA** comes across a byte whose value is between 1 and 31 it searches for the **EXT IL ROM** buffer. If the buffer is not found the original byte is sent out. Otherwise the search is started for an ID that matches this byte. If a match is not found the original byte is sent out. If a matching ID is found the character is replaced by the characters in the string under its ID.

Let's try an example that uses the two character strings we input using **ATOBUFX.**

We shall place the message, "EXT IL ROM, THE HP-IL ENHANCEMENT ROM", into the printer buffer of your ThinkJet. **EXT IL ROM** shall be placed in expanded print and underlined. To do this follow these steps.

- Clear alpha.
- Place the 5 character into alpha using the **X>AR** function. This character will be intercepted by **SACA** and the escape codes to set expanded print and turn on underlining will be sent instead.
- APPEND (enter alpha mode and hit shift K) "EXT IL ROM" to the character in alpha.

- Exit alpha mode. Put 6 in X and execute X>AR, this character is intercepted by SACA and the string under the ID of 6 is sent to the printer. This will get us back to normal print mode and turn off underlining.
- Execute SACA.
- Place ", THE HP-IL ENHANCEMENT ", into alpha.
- Execute SACA again. Notice that nothing has been printed, SACA only accumulates characters in the buffer (it does not print them).
- Place "ROM" in alpha and execute SACA again.

Our message is now in the printer buffer and we can print it by executing the PRBUF function in the HP-IL printer ROM.

```
EXT IL ROM, THE HP-IL ENHANCEMENT ROM
```

SACA also allows you to specify which device you want the data sent to by using the **PRTAID** function. To use this option just place the AID of the device you wish the data to be sent to in X and execute **PRTAID**. SACA will then attempt to send the data to this device. If the device is not in the loop at the time SACA is executed the "AID ERROR" is generated. If the specified printer AID is zero then SACA searches for an HP-IL printer. The routine to search for a printer locks out the HP82162A printer (the thermal strip printer) and will skip over it. All other printers and video interface devices are searched for by the routine that searches for printers. If you wish to use SACA with the HP-IL strip printer (the dedicated HP-41 strip printer will not work with SACA) just place 32 in X and execute **PRTAID**. All SACA output will then be forced to this printer. The printer enable switch on the back of the HP-IL module must be set to enable or the "NO IL PR. ROM" error message comes up.

Inputs for **ATOBUFX**

X: ID number of the string in alpha.
Alpha: Characters you wish to save in the **EXT IL ROM** buffer.

Inputs for **SACA**

X: none
Alpha: Alpha characters to be sent to the printer.

MCPRP (non-programmable)
MCLIST (non-programmable)

These are the Multi-Column PRint Program and Multi-Column LIST functions. They allow you to print user code programs in up to 15 columns. Like the rest of the **EXT IL ROM** print functions, their output may be forced to any device using the **PRTAID** function. Since these are nonprogrammable functions they may not be inserted as a program line. To use them clear flag 0 and place the number of columns you wish to print into X. If the number of columns specified is greater than 15 a "DATA ERROR" is generated. The default value for the width of a column is 22 characters, so only 3 columns will fit if the printer prints 80 characters per line. Now execute **MCPRP** and a single prompt will appear in the display after **MCPRP**. The function is asking for the name of the program you wish to print out. Press the alpha key and fill in the name of the program and press the alpha key again. The program will do nothing for a short time and then it will start to print. If the program takes more than one page, the first page will have the number of columns you specified with each being 60 lines long. At the end of the first page a formfeed will be sent and then the next page will be printed and so on until the program is all printed.

Both of these functions may be terminated by pressing either the R/S or the ON keys. You may have to hold the key down for a little while since **MCPRP** and **MCLIST** only check if a key is down after all columns of each line are printed. All other keys have no effect on the output.

Let's do an example using **MCPRP**. In our example we shall print out the PRPLOT program in the HP-IL printer ROM. We will have 3 columns. Here is how to do this.

- Place 3 into X and clear flag 00.
- execute **MCPRP**
- press the alpha key and enter PRPLOT and press the alpha key again.

A listing of what the printout should look like starts on the next page. It is two pages long. On the last page of a printout **MCPRP** tries to make all of the columns as even in length as possible.

```
01*LBL "PRPLOT"
02 AON
03 "NAME ?"
04 PROMPT
05 AOFF
06 ASTO 11
07*LBL 11
08 "Y MIN ?"
09 PROMPT
10 STO 00
11 "Y MAX ?"
12 PROMPT
13 STO 01
14 X<=Y?
15 GTO 11
16*LBL 12
17 "AXIS ?"
18 CF 23
19 PROMPT
20 STO 04
21 FS? 23
22 ASTO 04
23 RCL 01
24 X<Y?
25 GTO 12
26 CLX
27 RCL 00
28 X>Y?
29 GTO 12
30*LBL 13
31 "X MIN ?"
32 PROMPT
33 STO 08
34 "X MAX ?"
35 PROMPT
36 STO 09
37 X<=Y?
38 GTO 13
39 "X INC ?"
40 PROMPT
41 STO 10
42*LBL "PRPLOTP"
43 CF 12
44 ADV
45 6
46 SKPCHR
47 "PLOT OF "
48 ARCL 11
49 ACA
50 PRBUF
51 RCL 08
52 RCL 09
53 "X"
54 XEQ 09
55 STO 07
56 7
57 ACCHR
58 PRBUF
59 130
60 STO 02
61 XROM "PRAXIS"
62 RCL 10
63 X>0?
64 GTO 00
65 RCL 09
66 RCL 08
67 -
68 RCL 10
69 ABS
70 /
71 STO 10
72*LBL 00
73 RCL 09
74 RCL 08
75 ABS
76 X<Y?
77 X<>Y
78 RCL 07
79 /
80 LOG
81 INT
82 2
83 -
84 STO 05
85 RCL 08
86 STO 06
87*LBL 14
88 FIX IND 05
89 RCL 07
90 /
91 RND
92 ACX
93 3
94 SKPCOL
95 RCL 06
96 XEQ IND 11
97 REGPLOT
98 RCL 10
99 ST+ 06
100 RCL 09
101 RCL 06
102 X<=Y?
103 GTO 14
104 FIX 4
105 RTN
106*LBL "PRAXIS"
107 CF 12
108 RCL 00
109 RCL 01
110 "Y"
111 XEQ 09
112 STO 06
113 125
114 ACCHR
115 PRBUF
116 RCL 02
117 INT
118 ABS
119 STO 02
120 168
121 X<Y?
122 GTO 10
123 RCL 00
124 RCL 06
125 /
126 RND
127 ACX
128 XEQ 05
129 R^
130 RCL 01
131 XEQ 04
132 R^
133 +
134 -
135 7
136 X<=Y?
137 RDN
138 SKPCOL
139 RCL 01
140 RCL 06
141 /
142 RND
143 ACX
144 ADV
145 RCL 04
146 SIGN
147 X=0?
148 GTO 03
149 LASTX
150 RCL 00
151 X>Y?
152 GTO 10
153 -
154 RCL 01
155 RCL 00
156 -
157 X<Y?
158 GTO 10
159 /
160 RCL 02
161 1
162 -
163 *
164 .5
165 +
166 INT
167 STO Y
168 RCL 04
169 RCL 06
170 /
171 RND
172 ACX
173 XEQ 05
174 2
175 /
176 X>Y?
177 GTO 00
178 +
179 RCL 02
180 1
```

```
181 -
182 X<Y?
183 ENTER^
184 -
185 GTO 01
186*LBL 00
187 ENTER^
188 +
189 RCL 02
190 -
191*LBL 01
192 SKPCOL
193 ADV
194 XEQ 08
195 STO 05
196 X=0?
197 GTO 00
198 RCL 02
199 1
200 -
201 X=Y?
202 GTO 00
203 X<>Y
204 1
205 -
206 XEQ 06
207 RCL 05
208 1
209 +
210 GTO 01
211*LBL 03
212 XEQ 08
213*LBL 00
214 RCL 02
215 2
216*LBL 01
217 -
218 XEQ 06
219 ADV
220 RCL 02
221 RCL 05
222 1
223 +
224 1 E3
225 /
226 +
227 ENTER^
228 CHS
229 X<>Y
230 RCL 04
231 SIGN
232 X=0?
233 RDN
234 RDN
235 STO 02
236 FIX 4
237 RTN
238*LBL 04
239 RCL 06
240 /
241 RND
242*LBL 05
243 ABS
244 INT
245 X≠0?
246 GTO 00
247 RDN
248 5
249*LBL 00
250 LOG
251 INT
252 RCL 05
253 +
254 3
255 +
256 7
257 *
258 RTN
259*LBL 06
260 ENTER^
261 ENTER^
262 7
263 MOD
264 2
265 /
266 INT
267 SKPCOL
268 -
269 "-"
270*LBL 07
271 7
272 X>Y?
273 GTO 00
274 -
275 ACA
276 GTO 07
277*LBL 00
278 RDN
279 SKPCOL
280*LBL 08
281 127
282 ACCOL
283 R^
284 RTN
285*LBL 09
286 "⊦ <UNITS="
287 X<=Y?
288 GTO 10
289 X<>Y
290 ABS
291 X<Y?
292 X<>Y
293 LOG
294 X<0?
295 GTO 00
296 INT
297 2
298 X<>Y
299 X>Y?
300 GTO 01
301 -
302 STO 05
303 0
304 GTO 02
305*LBL 00
306 FRC
307 X≠0?
308 1
309 LASTX
310 INT
311 X<>Y
312 -
313*LBL 01
314 "⊦ E"
315*LBL 02
316 4
317 SKPCHR
318 ACA
319 FIX 0
320 RDN
321 X=0?
322 GTO 00
323 ACX
324 10^X
325 2
326 STO 05
327 FIX 2
328 RDN
329 GTO 01
330*LBL 00
331 1
332 ACX
333 FIX IND 05
334*LBL 01
335 "> "
336 ACA
337 RTN
338*LBL 10
339 0
340 /
341 END
```

It is also possible to specify the width of a column when using MCPRP. This is done by setting flag 00 and placing the number of characters per line your printer prints. Using this feature allows you to squeeze more columns on one page than when using the default values. You may also use this to spread the columns out so as to leave room for documentation (yeech). An example of how this is done is given below. If the column width is calculated to be less than 11 then a "DATA ERROR" is generated.

With the ThinkJet printer it is possible to print 142 characters in compressed print mode. Using this mode you are able to very easily fit 7 columns on one page. Here's how to do it.

- Place the ThinkJet into compressed print mode by sending the following bytes out to it, 27, 38, 107, 50, 83.
- set flag 00 142
- place 140 in X hit ENTER and put 7 in X.
- now execute MCPRP
- fill in the prompt with PRPLOT as was done before.

The next page is the full listing of the PRPLOT program as done on the thinkjet printer. There are seven columns of 20 characters with 2 characters at the end of each line being unused.

```
01*LBL "PRPLOT"
02 AON
03 "NAME ?"
04 PROMPT
05 AOFF
06 ASTO 11
07*LBL 11
08 "Y MIN ?"
09 PROMPT
10 STO 00
11 "Y MAX ?"
12 PROMPT
13 STO 01
14 X<=Y?
15 GTO 11
16*LBL 12
17 "AXIS ?"
18 CF 23
19 PROMPT
20 STO 04
21 FS? 23
22 ASTO 04
23 RCL 01
24 X<Y?
25 GTO 12
26 CLX
27 RCL 00
28 X>Y?
29 GTO 12
30*LBL 13
31 "X MIN ?"
32 PROMPT
33 STO 08
34 "X MAX ?"
35 PROMPT
36 STO 09
37 X<=Y?
38 GTO 13
39 "X INC ?"
40 PROMPT
41 STO 10
42*LBL "PRPLOTP"
43 CF 12
44 ADV
45 6
46 SKPCHR
47 "PLOT OF "
48 ARCL 11
49 ACA
50 PRBUF
51 RCL 08
52 RCL 09
53 "X"
54 XEQ 09
55 STO 07
56 7
57 ACCHR
58 PRBUF
59 130
60 STO 02
61 XROM "PRAXIS"
62 RCL 10
63 X>0?
64 GTO 00
65 RCL 09
66 RCL 08
67 -
68 RCL 10
69 ABS
70 /
71 STO 10
72*LBL 00
73 RCL 09
74 RCL 08
75 ABS
76 X<Y?
77 X<>Y
78 RCL 07
79 /
80 LOG
81 INT
82 2
83 -
84 STO 05
85 RCL 08
86 STO 06
87*LBL 14
88 FIX IND 05
89 RCL 07
90 /
91 RND
92 ACX
93 3
94 SKPCOL
95 RCL 06
96 XEQ IND 11
97 REGPLOT
98 RCL 10
99 ST+ 06
100 RCL 09
101 RCL 06
102 X<=Y?
103 GTO 14
104 FIX 4
105 RTN
106*LBL "PRAXIS"
107 CF 12
108 RCL 00
109 RCL 01
110 "Y"
111 XEQ 09
112 STO 06
113 125
114 ACCHR
115 PRBUF
116 RCL 02
117 INT
118 ABS
119 STO 02
120 168
121 X<Y?
122 GTO 10
123 RCL 00
124 RCL 06
125 /
126 RND
127 ACX
128 XEQ 05
129 R^
130 RCL 01
131 XEQ 04
132 R^
133 +
134 -
135 7
136 X<=Y?
137 RDN
138 SKPCOL
139 RCL 01
140 RCL 06
141 /
142 RND
143 ACX
144 ADV
145 RCL 04
146 SIGN
147 X=0?
148 GTO 03
149 LASTX
150 RCL 00
151 X>Y?
152 GTO 10
153 -
154 RCL 01
155 RCL 00
156 -
157 X<Y?
158 GTO 10
159 /
160 RCL 02
161 1
162 -
163 *
164 .5
165 +
166 INT
167 STO Y
168 RCL 04
169 RCL 06
170 /
171 RND
172 ACX
173 XEQ 05
174 2
175 /
176 X>Y?
177 GTO 00
178 +
179 RCL 02
180 1
181 -
182 X<Y?
183 ENTER^
184 -
185 GTO 01
186*LBL 00
187 ENTER^
188 +
189 RCL 02
190 -
191*LBL 01
192 SKPCOL
193 ADV
194 XEQ 08
195 STO 05
196 X=0?
197 GTO 00
198 RCL 02
199 1
200 -
201 X=Y?
202 GTO 00
203 X<>Y
204 1
205 -
206 XEQ 06
207 RCL 05
208 1
209 +
210 GTO 01
211*LBL 03
212 XEQ 08
213*LBL 00
214 RCL 02
215 2
216*LBL 01
217 -
218 XEQ 06
219 ADV
220 RCL 02
221 RCL 05
222 1
223 +
224 1 E3
225 /
226 +
227 ENTER^
228 CHS
229 X<>Y
230 RCL 04
231 SIGN
232 X=0?
233 RDN
234 RDN
235 STO 02
236 FIX 4
237 RTN
238*LBL 04
239 RCL 06
240 /
241 RND
242*LBL 05
243 ABS
244 INT
245 X≠0?
246 GTO 00
247 RDN
248 5
249*LBL 00
250 LOG
251 INT
252 RCL 05
253 +
254 3
255 +
256 7
257 *
258 RTN
259*LBL 06
260 ENTER^
261 ENTER^
262 7
263 MOD
264 2
265 /
266 INT
267 SKPCOL
268 -
269 "-"
270*LBL 07
271 7
272 X>Y?
273 GTO 00
274 -
275 ACA
276 GTO 07
277*LBL 00
278 RDN
279 SKPCOL
280*LBL 08
281 127
282 ACCOL
283 R^
284 RTN
285*LBL 09
286 "⊢ <UNITS="
287 X<=Y?
288 GTO 10
289 X<>Y
290 ABS
291 X<Y?
292 X<>Y
293 LOG
294 X<0?
295 GTO 00
296 INT
297 2
298 X<>Y
299 X>Y?
300 GTO 01
301 -
302 STO 05
303 0
304 GTO 02
305*LBL 00
306 FRC
307 X≠0?
308 1
309 LASTX
310 INT
311 X<>Y
312 -
313*LBL 01
314 "⊢ E"
315*LBL 02
316 4
317 SKPCHR
318 ACA
319 FIX 0
320 RDN
321 X=0?
322 GTO 00
323 ACX
324 10^X
325 2
326 STO 05
327 FIX 2
328 RDN
329 GTO 01
330*LBL 00
331 1
332 ACX
333 FIX IND 05
334*LBL 01
335 "> "
336 ACA
337 RTN
338*LBL 10
339 0
340 /
341 END
```

MCLIST works essentially the same as **MCPRP** except that it puts up three prompts instead of one. It is asking for the number of lines you wish to list, starting with the one the program pointer is pointing to. If you ask to print more lines than are left in the program the rest of the program is printed up to and including the END.

For this example we will list 99 lines of the PRPLOT program beginning with line 100.

- First GTO "PRPLOT" and then GTO . 100.
- place 3 in X. Clear flag 00.
- execute MCLIST and fill in the prompts with 099.

Here is what the listing should look like.

```
100 RCL 09            133 +              166 INT
101 RCL 06            134 -              167 STO Y
102 X<=Y?             135 7              168 RCL 04
103 GTO 14            136 X<=Y?          169 RCL 06
104 FIX 4             137 RDN            170 /
105 RTN               138 SKPCOL         171 RND
106*LBL "PRAXIS"      139 RCL 01         172 ACX
107 CF 12             140 RCL 06         173 XEQ 05
108 RCL 00            141 /              174 2
109 RCL 01            142 RND            175 /
110 "Y"               143 ACX            176 X>Y?
111 XEQ 09            144 ADV            177 GTO 00
112 STO 06            145 RCL 04         178 +
113 125               146 SIGN           179 RCL 02
114 ACCHR             147 X=0?           180 1
115 PRBUF             148 GTO 03         181 -
116 RCL 02            149 LASTX          182 X<Y?
117 INT               150 RCL 00         183 ENTER^
118 ABS               151 X>Y?           184 -
119 STO 02            152 GTO 10         185 GTO 01
120 168               153 -              186*LBL 00
121 X<Y?              154 RCL 01         187 ENTER^
122 GTO 10            155 RCL 00         188 +
123 RCL 00            156 -              189 RCL 02
124 RCL 06            157 X<Y?           190 -
125 /                 158 GTO 10         191*LBL 01
126 RND               159 /              192 SKPCOL
127 ACX               160 RCL 02         193 ADV
128 XEQ 05            161 1              194 XEQ 08
129 R^                162 -              195 STO 05
130 RCL 01            163 *              196 X=0?
131 XEQ 04            164 .5             197 GTO 00
132 R^                165 +              198 RCL 02
```

The **MCLIST** function may print all of the specified columns with the same number of lines even if this means printing a few more lines than you specified. This will only happen if the number of lines specified is not evenly divisible by the number of column wanted. In this case the number of lines printed will be the next highest number that is evenly divisible by the number of columns unless an END is hit, then the printing stops.

Here is an example.

- put 3 in X, and clear flag 00.
- GTO . 200 (we assume you are still in the PRPLOT program)
- execute **MCLIST** and fill in the prompts with 010.

```
200 -               204 1               208 1
201 X=Y?            205 -               209 +
202 GTO 00          206 XEQ 06          210 GTO 01
203 X<>Y            207 RCL 05          211*LBL 03
```

Notice that 12 lines were listed since this is the next number after ten that is evenly divisible by 3.

MCPRP and **MCLIST** use the same routine as **SACA** does for finding a printer on the loop. The 82162A thermal strip printer is locked out, all other printer and video class devices are searched for. If you use **PRTAID** to specify the device you wish to have **MCPRP** or **MCLIST** send their printouts to, they will send the output to the specified device. If the device is not in the loop the "AID ERROR" will be generated. The printer enable switch on the back of the HP-IL module must be in the enabled position for **MCPRP** or **MCLIST** to work, if this is not the case or the module is not plugged in then the "NO IL PR. ROM" error message is displayed.

MCPRP and **MCLIST** use buffer number 7 to store pointers for each column so you must have some free registers or the "PACKING" "TRY AGAIN" error message will be displayed. The number of registers needed depends on the number of columns specified. To calculate this number use the following formula:

$$\# \text{ registers} = \text{INT}(\# \text{ columns} / 2 + .5) + 1$$

Therefore the number of registers needed for a 7 column output would be INT(7 / 2 + .5) + 1 which is 5 registers. This buffer is automatically destroyed whenever **MCPRP** or **MCLIST** are terminated.

There are some characters that the 41 uses that most 80 column printers do not have in their characters set so we have made accommodations for these characters to be represented.

41 character	MCPRP, MCLIST character	Characters sent out to represent this
≠ not equal	≠	61, 8, 47
∡ angle	∤	60, 8, 124
⊦ append	∤	62, 8, 124
Σ sigma	~	126
▩ boxed star	⊕	79, 8, 43

All characters with a value greater than 127 will be replaced with the boxed star character (see last entry in above table) as will all characters whose value is less than 32 (except for the ≠ and angle characters).

Peculiarities of **MCPRP** and **MCLIST**

Due to the fact that these functions ask for data there is always the chance of getting strange output if the data is not consistent. Always remember the rule of computing, "garbage in = garbage out". So if you want 15 columns printed on your printer but it only prints 80 characters per line, then some strange output will result (15 * 22 = 330 which is greater than the number of characters/line on the printer, if flag 00 is clear).

If you have flag 00 set and the width of the columns you specified is less than the number of characters to print the longest line in the program then the lines to the right will be moved over. For example, you specify 3 columns and there are 48 characters per line. This gives a column width of 16 characters. Since the line number takes four characters (five for line numbers over 999) there would be room for only 12 characters to print the function. If there is a 15 character text line in the program it will use more than 12 characters and the columns will not print out straight. An example of how this would look is given below.

```
100 STO 99      110 RCL 15      120*LBL 01
101 "YOU SCREWED UP!"111 *         121 SWRTP
102 AVIEW       112 +           122 FS? 25
```

As you can see the middle line is pushed over. This is what happens when there is not enough room to print a program line in the space you specified. The default width of 22 characters per column is the minimum number of characters needed to take care of the longest possible program line that could be printed.

Inputs for MCPRP and MCLIST

flag 00 clear

X: Number of columns
Alpha: none
Other: Fill in the name of the program you wish to print at the single prompt (MCPRP).
Fill in the number of lines to list at the triple prompt (MCLIST)

flag 00 set

X: Number of columns.
Y: Number of chacters per line.
Alpha: none
Other: Same as above.

XROM numbers

Function	XROM number
EXT IL ROM	27,00
CLRBUF	27,01
DIRLEFT	27,02
DIRSIZE	27,03
NAMEMED	27,04
READBUF	27,05
READCAL	27,06
READXM	27,07
RECLEFT	27,08
SCOPYFL	27,09
SCREATE	27,10
SDIR	27,11
***SNEWM**	27,12
SWRTA	27,13
SWRTK	27,14
SWRTP	27,15
SWRTPV	27,16
SWRTS	27,17
WRTBUFX	27,18
WRTCAL	27,19
WRTXM	27,20
PRINT FCNS	27,21
ATOBUFX	27,22
***MCLIST**	27,23
***MCPRP**	27,24
PRTAID	27,25
SACA	27,26
X>AR	27,27

Functions with an * next to them are non-programmable.

List of error messages

Error	Functions	Reason
ALPHA DATA	SCREATE	Alpha characters in the X register.
	WRTBUFX	
	SCOPYFL	
	X>AR	
	CLRBUF	
	PRTAID	
	ATOBUFX	
	MCLIST	
	MCLIST	
DATA ERROR	SNEWM	Number input for total directory entries is too large.
	SCREATE	X > 65535
	WRTBUFX CLRBUF	X = 0, or X > 14
	X>AR PRTAID	X > 255
	ATOBUFX	X = 0 or X > 31
	MCLIST	X = 0 or X > 15, if flag 00 is set Y / X
	MCPRP	less than 11.
NO PRINTER	SACA	No printer device in the loop (occurs only
	MCLIST	in auto I/O mode)
	MCPRP	
PRIVATE	MCLIST	An attempt was made to list or print a
	MCPRP	private program.
	SWRTA	Tried to save the calculator when a
	WRTCAL	private program was present.
	SWRTP	Tried to save a private program.
	SWRTPV	
	SCOPYFL	Tried to copy a private program.
NONEXISTENT	MCPRP	Specified program does not exist.
	WRTBUFX	
	SCOPYFL	
	X>AR	X > 999
	PRTAID	
	ATOBUFX	
	MCLIST	Y > 999 (flag 00 set)
	MCPRP	
DIR FULL	All **EXT IL ROM** mass storage write functions.	

DRIVE ERR	ALL MASS STORAGE	Medium stalled, try new medium. Drive or medium may be bad.
DUP FL NAME	SCREATE	A data file with the same name already exists.
	SWRTA SWRTK SWRTP SWRTPV SWRTS WRTBUFX WRTCAL WRTXM	A file with the same name but of a different type already exists on the medium.
FL NOT FOUND	READBUF READCAL READXM	A file with the specified name does not exist on the medium.
FL SECURED	All **EXT IL ROM** write functions. File is secured, use UNSEC to cancel security.	
FL TYPE ERR	READBUF READCAL READXM	Specified file not type required by function.
	SCOPYFL	Tried to copy a non 41 file.
MEDM ERR	ALL	Medium improperly installed or possible damage.
MEDM FULL	SCREATE SCOPYFL SWRTA SWRTK SWRTP SWRTPV SWRTS WRTBUFX WRTCAL WRTXM	No room left on the medium to store the file.
MEMORY LOST	READCAL	File not read in correctly, memory cleared.
NAME ERR	All write functions. Alpha is empty.	
NO DRIVE	All mass storage functions. No mass storage device is in the loop or selected device is not mass storage in MANIO mode.	

NO KEYS	SWRTK	No key assignments to save.
NO ROOM or PACKING, TRY AGAIN	READBUF	Not enough free registers to read in buffer.
	READXM	Not enough extended memory to contain file.
	READCAL	Main memory too small to contain main memory portion of file, or not enough extended memory available.
	PRTAID	Not enough room to create the **EXT IL ROM** buffer and store the AID.
	ATOBUFX	No room to expand buffer and place the alpha string in the buffer.
	MCLIST MCPRP	No room to create buffer 7.
NO MEDM	ALL mass storage functions	There is no medium in the drive. Also comes up when the HP9114 disc drive is low on battery power.
READ ERR	READBUF READCAL READXM	Can not read specified file.
ROM	SWRTP SWRTPV	Specified program is in a plug in module.
	SWRTA WRTCAL	Program pointer is in ROM. It must be in RAM.
NO HPIL	ALL mass storage functions.	The HP-IL module is not plugged in.
NO IL PR. ROM	SACA MCPRP MCLIST	The HP-IL module is not plugged in or the printer enable switch on the back is set to the disable position.
NO XFM	WRTXM WRTCAL READXM READCAL	No extended functions module is plugged in the calculator.
AID ERROR	SACA MCLIST MCPRP	The AID specified by **PRTAID** does not exist on the loop.
ADR ERR	SCOPYFL	X = 0 OR X > 30.

NO BUFFER	WRTBUFX	Buffer specified in X is not present in the machine.
BUF EXISTS	READBUF	A buffer of the type you are reading in already exists in the machine.
DIR EMPTY	WRTCAL WRTXM	No files in extended memory.

Warranty Information

The **EXT IL ROM** module does not require maintenance, and contains no serviceable parts.

LIMITED 90 DAY WARRANTY

EXT IL ROM is warranted, with the exception of software content, against defects in materials and workmanship affecting electronic and mechanical performance for a period of 90 days from the original date of purchase. During the warranty period, SKWID INK will replace or, at our option, repair a product that proves to be defective, provided it is returned, shipping prepaid to SKWID INK.

SKWID INK makes no expressed or implied warranty with regard to the program material offered, nor to merchantability or fitness of the program material for any particular purpose. Program material is made available to the user on an "as is" basis, with the entire risk as to quality and performance resting with the user. While every effort has been made to eliminate deficiencies in the program material, the user (and not SKWID INK, nor any other party) shall bear the entire cost of all necessary correction and all incidental or consequential damages.

EXT IL ROM is sold on the basis of specifications at manufacture. SKWID INK shall be under no obligation to modify or update **EXT IL ROM** once manufactured.

LIMITATIONS OF WARRANTY

The **EXT IL ROM** warranty does not, and shall not apply if **EXT IL ROM** has been damaged by accident, misuse, or if attempts have been made to modify the module. No other expressed or implied warranty is given.

SHIPPING FOR SERVICE

In the unlikely event that **EXT IL ROM** is found defective, return the module, postage prepaid to:

SKWID INK
P. O. Box 3103
Tustin, CA 92681 USA

When returning **EXT IL ROM**, be sure to include the following items.

- A sales receipt or other proof of purchase indicating the 90 day warranty has not expired.
- A detailed description of the problem, indicating when and how the problem occurs.

Whether **EXT IL ROM** is still under warranty or not, it is your responsibility to ensure that the unit is securely packaged to prevent damage in transit (this is not covered by the warranty) and that postage costs to SKWID INK are paid.