HP-41 Virtual Tape Drive



© 2026 Systemyde International Corporation

Every effort has been made to ensure the accuracy of the information contained herein. If you find errors or inconsistencies please bring them to our attention.

Copyright © 2026 Systemyde International Corporation. All rights reserved.

Notice:

“HP-41C”, “HP-41CV”, “HP-41CX” and “HP” are registered trademarks of Hewlett-Packard, Inc. All uses of these terms in this document are to be construed as adjectives, whether or not the noun “calculator”, “CPU” or “device” are actually present.

Acknowledgements:

Table of Contents

1. READ THIS FIRST 5

2. Introduction 6

3. Virtual Tape Drive Overview 7

Volume Header 8
Directory Structure 9
File Types 11
Function Parameters 14

4. Volume Management Functions 15

VDIR (List Directory Entries) 15
VFMT (Format Volume Header) 16
VLBL? (Report Volume Label) 16
VPACK (Pack Volume) 16
VREC? (Report Volume Records Remaining) 16
VREM? (Report Directory Entries Remaining) 17
VSIZ? (Report Directory Size) 17

5. File Management Functions 18

DS? (Report Datestamp) 18
FDEL (Delete File) 18
FEXP (Export File via HP-IL) 18
FIMP (Import File via HP-IL) 19
FLOC? (Report File Start Record) 19
FREC? (Report File Records Used) 19
FREN (Rename File) 19
FSEC (Secure File) 20
FSIZ? (Report File Size) 20
FTYPE? (Report File Type) 20
FUNSEC (Unsecure File) 20
FZERO (Clear File) 20
TOUCH (Update Datestamp & Timestamp) 21
TS? (Report Timestamp) 21

© 2026 Systemyde International Corporation

6. General File (AS, CA, HE, ML, RM, ST, WA, XM or YM) Operations 22

FRD (Read File) 22
FWR (Write File) 22

7. Program File (PR) Operations 23

PFRD (Read Program) 23
PFRDS (Read Subroutine) 23
PFWR (Write Program) 24
PFWRP (Write Program - Private) 24

8. Data File (DA) Operations 25

DFNEW (Create Data File) 25
DFRD (Read Registers) 25
DFRDX (Read Registers by X) 25
DFSEEK (Set Pointer to File and Register) 26
DFWR (Write Registers) 26
DFWRX (Write Registers by X) 26

9. Error Messages 27

10. Function XROM Numbers 28

11. Internal Details 29

12. Revision History 30

© 2026 Systemyde International Corporation

READ THIS FIRST

Two versions

This document describes two versions of the HP-41 Virtual Tape Drive software. The first version is contained in a plug-in module, which also contains enough memory to emulate a single HP Mini Data Cassette tape, as used in the 82161A Tape Drive. The 41CL version is a ROM image that can be virtually plugged into a port and uses the 41CL RAM to emulate a tape up to six times larger than a single tape.

Port Restriction

The 41CL version of the software can only be plugged into the upper half of Port 4, which is page 0xF. When using the 41CL Extreme Functions to plug in images this restriction is automatically enforced because it is encoded in the Image Database (IMDB). The module version can be inserted in any port, where it will occupy the lower page in the port.

All Virtual Tape memory is volatile!

All of the RAM in the HP-41 Virtual Tape Drive module is volatile and will lose its contents when power is removed. Similarly, all of the RAM in the 41CL is volatile and will lose its contents when power is removed.

When First Inserting the HP-41 Virtual Tape Drive Module

The HP-41 Virtual Tape Drive module uses a Field-Programmable Gate Array (FPGA) to implement all of the logic and memory in the module. An FPGA must be loaded with the logic and program memory contents at power-up, and until this process is complete the FPGA outputs are undefined. This undefined state can sometimes cause the bus drivers in the module to drive the HP-41 bus.

One particular bus signal (ISA) is used to wake up the HP-41 processor if driven while the HP-41 is off. (This is how the button on Wand works and how the Time Module displays the time continuously.) If this happens the HP-41 may wake up and then lock up when the HP-41 Virtual Tape Drive module is inserted into the calculator.

To clear this condition, first try pressing **ON** several times. If this doesn't work remove the batteries and then press **ON** several times before re-inserting the batteries. Insert the batteries, wait a few seconds, and then turn on the calculator. It may take a few tries for this to work, but once the calculator starts responding everything should be fine.

© 2026 Systemyde International Corporation

Introduction

The HP-41 Virtual Tape Drive (VTD) software contains updated versions of the Mass Storage functions in the 82160A HP-IL module. These functions are augmented by new functions that provide even more capability when working with a virtual (RAM-based) tape drive.

If you are not already familiar with the Mass Storage functions in the 82160A HP-IL module you will find that the HP-41 Virtual Tape Drive software contains a number of the file-related functions found in a simple operating system such as CP/M or MS-DOS.

The Mass Storage functions in the 82160A HP-IL module provide separate functions to read and write different types of files. This approach works well when the number of different file types is small, but quickly becomes unwieldy as the number of different file types increases. For this reason the HP-41 Virtual Tape Drive software takes the approach of common functions with the file type passed as a parameter. As in the familiar CP/M or MS-DOS cases, files are specified in a *filename.filetype* format. Despite using this format, all files must still have a unique *filename*, that is independent of *filetype*.

The HP-41 Virtual Tape Drive module contains both the software and a 131,072 (2^{17}) byte RAM that is organized to appear as a single tape in an 82161A Digital Cassette Drive. Also included is a copy of the 82182A Time Module, which will be activated only if such a device is not already present in the HP-41.

When the HP-41 Virtual Tape Drive software is plugged into a port in the 41CL much of the RAM memory on the 41CL board will be organized to appear as an 82161A Digital Tape Drive with a tape that is up to six times larger than the original

© 2026 Systemyde International Corporation

Virtual Tape Drive Overview

The tape in an 82161A Cassette Tape Drive holds 131,072 (2^{17}) bytes of information. This information is organized into two *tracks*, with each track containing 256 *records* of 256 bytes each. Each byte is individually addressable, although the tape drive itself only reads and writes one full record at a time. The fact that the tape drive only reads and writes full records is hidden from the user.

The RAM memory in a module-based virtual tape drive consists of 65,536 (2^{16}) 16-bit words. Each byte is individually addressable, but the memory only reads and writes words. The fact that the RAM only reads and writes words is also hidden from the user.

The HP-41 Virtual Tape Drive software presents the user with a view of the RAM memory that is identical to that used in a physical tape drive, with 512 or more records of 256 bytes each.

The 82161A Cassette Tape Drive numbers bytes starting with track 0, record 0 and byte 0. Bytes within a record are numbered consecutively from 0 through 255. Records are likewise numbered consecutively, from 0 through 255, and track 0 is followed by track 1. Bytes within a physical 41C memory are numbered with the most-significant byte of a word as byte 0. This byte numbering is only important if you choose to access bytes via the 41CL view of physical memory.

Physical tape	Byte Address
track	A[19:16]
record	A[15:8]
byte	A[7:0]

The contents of a tape, virtual or otherwise, are organized according to the Logical Interchange Format (LIF) standard used by many other HP devices and systems. LIF is a Sequential Flat File System, which is ideal for transferring information between dissimilar computers. A LIF *volume* consists of a volume *header*, a file *directory*, and some number of *files*.

The first two records (numbers 0 and 1) are used for a volume header and the next set of records are used for a file directory. Each record in the directory holds eight file entries. For the 82161A Cassette Tape Drive this file directory can range in size from one record to 56 records. The HP-41 Virtual Tape Drive software restricts the directory size to 255 records. All remaining records in the volume are used to hold files.

Both the directory and files can only occupy an integer number of sequential records, which greatly simplifies file access, at the cost of some storage efficiency.

Multi-byte values in the volume header and directory use big endian format.

© 2026 Systemyde International Corporation

Volume Header

The volume header occupies the first two records of the volume, although only a few of the bytes are actually used in the case of an HP-41C. The table below shows the organization of the first record of the volume header.

Bytes	Contents	Meaning
0-1	0x8000	LIF Volume Identifier
2-7	*vlabel*	Volume Label in ASCII
8-11	0x00000002	Directory Start Record
12-15	0x10000000	Fixed for HP-41
16-19	0x000000nn	Directory Length (records)
20-23	0x00010000	Fixed for HP-41
24-27	0x0000000n	Tracks
28-31	0x00000001	Heads
32-35	0x00000100	Sectors
36-38	mmddyy	Date Stamp
39-41	hhmmss	Time Stamp
42-239	all 0x00	Unused for HP-41
240-255	*internal use*	VTD software internal use

The main purpose of the volume header, as far as an HP-41C is concerned, is to indicate where the directory is located and how big it is. More sophisticated systems will use more of the information in the volume header.

The Mass Storage functions in the 82160A HP-IL module do not use several of the fields in the volume header. These include the Volume Label, Volume Size fields, Date Stamp and Time Stamp. These fields can be accessed by functions in the HP-41 Virtual Tape Drive software.

The hexadecimal Tracks value will be 0x2 for the HP-41 VTD module, 0x4 for a 41CL V2 machine, and 0xC for a 41CL V3/4/5 machine. This value is automatically set when the volume is formatted and is used to determine when the volume is full or when the volume remaining size is reported.

The Date Stamp is not Y2K compliant, because only one byte is stored for the year. The HP-41 Virtual Tape Drive software uses a windowing algorithm to determine the century. Any year value less than or equal to 80 is assumed to be 21st century (20xx) and any year value greater than 80 is assumed to be 20th century (19xx).

The second record of the volume header is not used and is initialized with all zeros when the volume is formatted.

© 2026 Systemyde International Corporation

Directory Structure

The directory is stored in the records immediately following the volume header. In the 82161A Cassette Tape Drive the directory is limited to 56 records, numbered 2 through 57. The VTD software allows for up to 255 records to be reserved as the directory. But a smaller directory results in a better directory search time. This is important in the HP-41 Virtual Tape Drive module but less so when the software image is used in a 41CL because of the 41CL Turbo mode.

Each directory entry requires 32 bytes, so there are eight directory entries per record. The last record used for the directory must have at least the final entry filled with 0xFF in every byte to mark the end of the directory. If this last record contains less than seven valid directory entries all of the unused entries must be filled with all 0xFF. The table below shows how a directory entry is organized.

Byte(s)	Contents	Meaning
0-6	*filename*	File Name in ASCII
7-9	0x202020	ASCII space characters
10-11	0xnnnn	File Type
12-15	0x0000nnnn	File Start Record
16-19	0x0000nnnn	File Length (records)
20-22	mmddyy	Date Stamp
23-25	hhmmss	Time Stamp
26-27	0x8001	Fixed Value
28-29	0xnnnn	File size in bytes/registers
30	0x0n	File attributes
31	0x20	ASCII space character

The LIF standard allows for up to ten characters for a file name, but the HP-41C is restricted to seven characters and the remaining bytes are filled with the ASCII space character (0x20). File names less than seven characters long are likewise filled with ASCII space characters. File names must be unique, independent of file type, within the directory.

Two bytes are used to specify the file type, and the LIF standard documents a large number of different types. The Mass Storage functions in the 82160A HP-IL module only support half a dozen file types, but the HP-41 Virtual Tape Drive software adds several new types that were used by third-party modules, plus two that are specific to the 41CL.

© 2026 Systemyde International Corporation

The contents of the different file types will be described later, but the table below shows the different types of files allowed by the HP-41 Virtual Tape Drive software.

Value	*filetype*	File Type	Comments
0x0000	**-**	"Deleted Entry"	deleted file directory entry
0x0001	**AS**	ASCII	from X-Memory file
0xE020	**CA**	Calculator	registers plus X-memory
0xE030	**XM**	Extended Memory	entire X-memory
0xE040	**WA**	Write All	entire register memory
0xE050	**KE**	Key Assignments	OS keyboard assignments
0xE060	**ST**	Status	user-visible registers
0xE070	**ML**	Machine Language	compressed ROM page
0xE080	**PR**	Program	user program
0xE090	**YM**	Expanded Memory	entire page (1024 registers)
0xE0C0	**RM**	ROM	uncompressed ROM page
0xE0D0	**DA**	Data	registers
0xE0D0	**HE**	HEPAX	compressed ROM page
0xFFFF	**-**		final directory entry

The Mass Storage functions in the 82160A HP-IL module do not use the Date Stamp or Time Stamp fields, but these fields can be accessed by the HP-41 Virtual Tape Drive software and are automatically updated when a file is created or modified, as long as a Time Module is present in the system.

Bytes 28-29 are used for information specific to the HP-41, such as the actual byte length of a stored program or the length, in registers, of a data file. These bytes are managed automatically by the HP-41 Virtual Tape Drive software.

Byte 30 is used to hold the file attribute information, with the *Secure* status in bit 3, the *Auto-execute* status in bit 1 and the *Private* status in bit 0. This information is managed automatically by the HP-41 Virtual Tape Drive software.

The HP-41 Virtual Tape Drive software uses bit 2 of byte 30 to hold some extra file attribute information. For the DA file type this bit indicates that the file is a HEPAX-encoded page image. Otherwise the only way to know that this is the case is if the user remembers it from the file name.

Byte 31 is not used by the HP-41.

© 2026 Systemyde International Corporation

File Types

The HP-41 Virtual Tape Drive software supports twelve different types of files, which is double what the 82160A HP-IL module supports. Two of the new types come from the third-party EXT-IL module, one comes from the Eramco MLDL-OS module, one comes from the HEPAX module, and two are for the 41CL.

The 82160A HP-IL module uses dedicated functions to read and write each type of file. This approach becomes cumbersome with a large number of file types, so for most file types the HP-41 Virtual Tape Drive software requires that the file type (*filetype*) be specified along with the name when reading or writing the file. This allows the software to determine where in the HP-41C memory the data is going to or from, and any encoding or decoding that is required.

Some file types are variable in size, but several have a fixed size. Only two file types allow random (register) access and there are dedicated functions to support this operation.

An ASCII file with a *filetype* of **AS** contains text characters and is sourced from a file in Extended Memory. The 82160A HP-IL module does not contain functions to read and write this type of file, but the Extended Functions/Memory module does. The HP-41 Virtual Tape Drive software provides equivalent functionality. An **AS** file is variable in size.

A Status file with a *filetype* of **ST** contains calculator status information, which consists of the size, the position of the statistics registers, the state of the user flags, and the contents of the stack registers and the ALPHA register. An **ST** file always contains ten registers, which is seventy bytes, or one record. One caution when reading an **ST** file is if the number of allocated data registers is smaller than the value in the saved file. In this case the contents of the excess storage registers are lost. If the size contained in the file is not possible in the calculator the size is not changed, but all other status will be changed.

A Key Assignments file with a *filetype* of **KE** contains key assignment information for standard functions. Reading this type of file cancels existing key assignments and substitutes those from the file. A **KE** file is variable in size.

A Data file with a *filetype* of **DA** contains HP-41C register data, and is numbered starting with R_0. The data read from such a file will be written starting with calculator register R_0. Writing continues until all file registers have been copied or until all HP-41C registers are filled. But the contents of a **DA** file can also be accessed directly, by register number or range of registers. A **DA** file must be created before use and is variable in size.

Historically, the **DA** file type was also used for a HEPAX-encoded page of instruction memory. Normally the only way to know that a **DA** file contains this encoded information is for the user to remember the file name. The HP-41 Virtual Tape Drive software adds a special status bit to the directory entry for such a file and provides an alias *filetype* of **HE**

© 2026 Systemyde International Corporation

to access this type of file. As far as legacy software is concerned the file type is **DA**, but the HP-41 Virtual Tape Drive software recognizes this alias to enable the encoding and source or destination. This type of file always contains 5120 bytes, or 20 records.

A file with a *filetype* of **CA** contains the contents of all of main memory (as with a **WA** file), plus all of the available Extended Memory. This is effectively a backup of the entire calculator, so when reading this type of file the calculator hardware must be identical to when the file was created. The **CA** *filetype* was introduced by the third-party EXT-IL module, although that software did not handle the Extended Memory exactly the way the HP-41 VTD software does. The size of a **CA** file depends on the size of main memory, plus Extended Memory in the calculator.

A file with a *filetype* of **RM** contains an image of a full page of instruction memory. Unlike the **HE** and **ML** file types, the 10-bit instructions are not encoded, but instead are stored just as in the common .ROM file format, with sixteen bits per instruction. This format preserves the extra speed-control bits that are used in the 41CL. An **RM** file is 8192 bytes or 32 records in length.

A file with a *filetype* of **ML** contains an ERAMCO-encoded page of 10-bit instruction memory. Unlike the HEPAX-encoded page of instruction memory, ERAMCO employed this dedicated *filetype*, which the HP-41 Virtual Tape Drive software also supports. This type of file is always 5120 bytes, or 20 records, in length.

A file with a *filetype* of **WA** contains the entire contents of the calculator main memory. This is effectively a backup of the calculator main memory, so when reading this type of file the main memory hardware of the calculator must be the same size as when the file was created. The size of a **WA** file depends on the size of main memory in the calculator.

The **XM** *filetype* is used for copies of the contents of Extended Memory. The entire available Extended Memory is included in this type of file, so it has variable size. This *filetype* was introduced by the third-party EXT-IL module, although that software did not include the entire Extended Memory in the file. The size of an **XM** files depends on the amount of Extended Memory present in the machine when the file is created.

The **YM** *filetype* accommodates the Expanded Memory introduced in the 41CL. The entire 1024 register block of Expanded Memory is included in the file, so it is 8192 bytes, or 32 records, in length. The contents of a **YM** file can also be accessed directly, by register number or range of registers.

The Program (**PR**) *filetype* is probably the most useful for everyday use because it is used to hold individual HP-41C programs. Files with the **PR** *filetype* can be given different names than the program they contain, so dedicated functions are required to handle the different naming options. Program files can be made *Private* when they are created to prevent the program from being listed, edited, or recorded elsewhere. Different options are available for where a retrieved program is placed in program memory, and whether it is used as a subroutine in a running program.

© 2026 Systemyde International Corporation

The table below shows the file size for the various file types:

filetype	Registers	Bytes	Records
AS	-	variable	variable
CA	208-942	-	7-30
DA	1-4095		1-128
HE		5120	20
KE	1-35	-	1-2
ML	-	5120	20
PR	-	variable	variable
RM	-	8192	32
ST	10	-	1
XM	128-606	-	5-19
WA	80-336	-	3-11
YM	1024	-	32

HP-41C registers are seven bytes in length, but are always stored using eight bytes, with the same arrangement of bytes that is used by the 82161A Cassette Tape Drive for full compatibility.

The 82160A HP-IL module writes files (**CA**, **DA**, **KE**, **PR** and **ST** file types) with an accompanying checksum and verifies that this checksum is correct on file reads. The VTD software also appends this checksum to these file types, but does not verify the checksum on file reads because of the inherent reliability of RAM access. Accessing data from a tape over HP-IL can be subject to corruption, which is why the checksum is appropriate in that environment.

When using the HP-41 Virtual Tape Drive software in a 41CL it is important to remember that files are a logical construct on top of a physical memory. The 41CL Extra Functions and 41CL Extreme Functions allow direct access of physical memory, which can corrupt volume, directory, and file information if used inappropriately.

© 2026 Systemyde International Corporation

Function Parameters

Functions in the HP-41 Virtual Tape Drive software use a uniform method to accept parameters. Alphanumeric parameters use the ALPHA register and any numeric parameters use the X-register. Return values use the same convention. What follows is an explanation of input parameter restrictions. The comma (,) is used to separate parameters when more then one is required.

File Name: *filename* = up to seven characters from {A…Z, a…e, 0…9}. Punctuation marks are not allowed, because they are used as field or parameter delimiters. *newname* is used to specify a new file name.

File Type: *filetype* = {**AS**, **CA**, **DA**, **HE**, **KE**, **ML**, **PR**, **RM**, **ST**, **WA**, **XM**, **YM**}. The *filetype* is always preceded by a period delimiter to separate it from the *filename*. The *filetype* is usually required unless the function is specific to a particular type of file.

Program Name: *prgmname* = up to seven characters from {A…Z, a…e, 0…9}. Punctuation marks are not allowed, because they are used as field or parameter delimiters. This is only used when storing a program, and has the regular HP-41C restrictions.

Volume Label: *vlabel* = up to six characters from {A…Z, a…e, 0…9}. Punctuation marks are not allowed.

Note that the above character restrictions are not enforced by the VTD software, but violating these restrictions will make transferring files into or out of the virtual tape drive impossible.

© 2026 Systemyde International Corporation

Volume Management Functions

The Volume Management functions operate on the volume header or the entire volume.

VDIR

The **VDIR** (*Volume Directory*) function displays the name of each file currently stored in the volume, in the order that they appear in the directory, plus the file type and any special attributes associated with the file.

The function uses the ALPHA register to display the information and also appends the file length (in registers) to the ALPHA register, although this length is not sent to the display. If a printer is found in the system all of the information is also sent to the printer.

The listing can be paused and restarted using the **R/S** key and is cancelled using the ← key. When cancelled the most recent entry remains in the ALPHA register.

The file type is indicated by the usual two-letter *filetype* as shown in the table below.

Type	Contents
AS	ASCII
CA	Calculator
XM	Extended Memory
WA	Write All
KE	Key Assignments
ST	Status
ML	Machine Language
PR	Program
YM	Expanded Memory
RM	ROM
DA	Data
HE	HEPAX
??	unknown file type

The special file attributes are indicated as shown in the table below.

Code	Attribute	Comments
A	Automatic Execution	Only for Program files
P	Private	Only for Program files
S	Secured	

© 2026 Systemyde International Corporation

VFMT (*size* in X-register, *vlabel* in ALPHA)

The **VFMT** (*Format Volume Header*) function initializes the volume header and creates the directory structure necessary to index the desired number of files. Up to 2039 directory entries (one for each file) can be created in the directory, but the larger the number of entries, the more time it will take to search the directory. Once a volume is formatted the number of directory entries cannot be changed. This function is not programmable.

If no *vlabel* is present in the ALPHA register this function automatically uses "VOLUME" for the volume label.

The volume header requires two records and each eight directory entries require one record. A reasonable rule of thumb is to reserve about 10% of the total available records for the file directory. This would be 400 entries for the VTD module, 800 entries for a 41CL V2 and the function maximum of 2039 for a 41CL V3/4/5.

This function automatically sets the volume size to the maximum supported by the hardware. If a smaller volume size is desired, it is possible to manually edit the size parameter by POKEing a different value to the appropriate location in the volume header.

Formatting a volume does not initialize the file area of the volume. The volume header is completely initialized and all of the records required for the directory are initialized with 0xFF in every byte.

Once formatted, a volume cannot be reformatted using this function, because the function checks for the presence of the LIF Volume Identifier in the first word of the volume. Destroying an existing volume requires manually POKEing a value other than 0x8000 to the first word of the volume and then executing **VFMT** to create a new volume.

VLBL?

The **VLBL?** (*Report Volume Label*) function reads the volume header and reports the volume label in the ALPHA register.

VPACK

The **VPACK** (*Pack Volume*) function scans the directory for any entries marked as "Deleted" and then reclaims the corresponding file area while at the same time reclaiming the area used for the directory entry.

© 2026 Systemyde International Corporation

Files deleted by the **FDEL** function have their directory entry marked as "Deleted" but no other action is taken by that function. This leaves the directory slot, and file space, available for future use; but only for a file that will fit in the available file space. This is a characteristic of a sequential file system, and results in a couple of potential problems. First, it means that a small file might potentially be assigned to a block of file memory that is much larger than is actually required. Second, if only "Deleted" directory entries are available and they are all smaller than the size required for a new file, the volume may appear to be full.

This function steps through the directory and for every "Deleted" entry moves all subsequent directory entries back one space. At the same time the corresponding file area is also reclaimed by moving subsequent file area data back to the available file area.

The packing algorithm is intelligent, so that each record in the file area of the volume is only moved once, but this function can still take a significant amount of time if the number of records in use is large. The function displays a ***PACKING*** message while running and blinks the display after every deleted entry is processed.

VREC?

The **VREC?** (*Report Records Remaining*) function scans the directory and reports the number of records in the volume that are currently unused in the X-register.

VREM?

The **VREM?** (*Report Directory Entries Remaining*) function scans the directory and reports the number of directory entries that are currently unused (either erased or never programmed) in the X-register.

VSIZ?

The **VSIZ?** (*Report Directory Size*) function reads the volume header and reports the directory size, in allowed number of entries, in the X-register.

© 2026 Systemyde International Corporation

File Management Functions

The File Management functions operate on entire files. All of these functions (except for **FIMP**) check whether the specified *filename* already exists and will exit with an error message if this is not the case.

DS? (*filename* in ALPHA)

The **DS?** (*Report Datestamp*) function reads the directory entry for the specified file and extracts the *datestamp*. If the ALPHA register is empty the volume header is read. The *datestamp* is returned in the X-register and respects Flag 31. Flag 31 clear selects *mm.ddyyyy* format, while Flag 31 set selects *dd.mmyyyy* format.

The *datestamp* stored in the volume header or a directory entry is not Y2K compliant, so this software uses a windowing algorithm to determine the century. Any year value less than or equal to 80 is assumed to be 21st century (20xx) and any year value greater than 80 is assumed to be 20th century (19xx).

FDEL (*filename* in ALPHA)

The **FDEL** (*Delete File*) function removes the specified file by marking it as "Deleted" in the directory. The data in the file is not overwritten by this function. Use the **FZERO** function to erase the data in a file prior to deletion if necessary. The directory entry and space formerly occupied by the file becomes available for a new file, but only if the new file is the same size or smaller.

If the file being deleted is the last file in the directory this function will replace the entire directory entry with all ones, which marks the directory entry as unused. This operation continues recursively if the new last file in the directory is also marked as "Deleted."

A file with the *Secured* bit set cannot be deleted.

FEXP (*filename* in ALPHA)

The **FEXP** (*Export File via HP-IL*) function copies the specified file to an external mass storage device via HP-IL. If no 82160A HP-IL module (or equivalent) is detected this function will exit with an error.

© 2026 Systemyde International Corporation

The file is created in the external volume, and if the *filename* already exists there the function will also exit with an error. All of the information in the directory entry for the file is replicated in the external volume.

Remember that not all file types supported by the Virtual Tape Drive software may not be recognized correctly by an 82160A HP-IL module.

FIMP (*filename* in ALPHA)

The **FIMP** (*Import File via HP-IL*) function copies the specified file from an external mass storage device via HP-IL. If no 82160A HP-IL module (or equivalent) is detected this function will exit with an error.

If the *filename* already exists the function will exit with an error. All of the information in the directory entry for the file in the external volume is replicated in the local directory.

FLOC? (*filename* in ALPHA)

The **FLOC?** (*Report File Location*) function accesses the directory and reports the record number for the start of the file in the X-register.

The record number for the start of the file is only useful if you need to access file data directly, using 41CL functions. The table below shows the *starting* physical memory address for several record numbers. Note that the VTD module uses the same physical addressing to simplify the software.

Record	41CL Word Address	Comments
0	0x820000	Volume Header
2	0x820010	Start of Directory
511	0x82FF80	Final Record (VTD module)
1023	0x83FF80	Final record (V2)
3071	0x87FF80	Final record (V3/V4/V5)

FREN (*filename,newname* in ALPHA)

The **FREN** (*Rename File*) function changes the name of a file from *filename* to *newname*. A file that has the *Secured* bit set cannot be renamed.

© 2026 Systemyde International Corporation

FSEC (*filename* in ALPHA)

The **FSEC** (*Secure File*) function sets the *Secure* bit in the directory entry for the specified file, which protects the file from being erased, renamed or altered.

FSIZ? (*filename* in ALPHA)

The **FSIZ?** (*Report File Size*) function accesses the directory and reports the actual size of the file, in either registers or bytes, depending on the type of file, in the X-register.

For file types **AS**, **HE**, **ML**, **PR** and **RM** the size is reported in bytes, while all other file types report the number of registers. Note that each register requires eight bytes.

The Virtual Tape Drive software always sets the size field in the directory entry correctly, but the same cannot always be said for files from an external source. Keep this in mind when accessing files originally from an external source.

FTYPE? (*filename* in ALPHA)

The **FTYPE?** (*Report File Type*) function accesses the directory and returns the *filename.filetype* in both the display and the ALPHA register.

FUNSEC (*filename* in ALPHA)

The **FUNSEC** (*Unsecure File*) function clears the *Secure* bit in the directory entry for the specified file, removing the protection given to a secured file.

FZERO (*filename* in ALPHA)

The **FZERO** (*Clear File*) function writes all of the data in the specified file with zeros. This may take some time, depending on the size of the file.

Deleting a files with the **FDEL** function does not erase the file contents, so this function should be executed on the file prior to file deletion if the file contains sensitive data.

© 2026 Systemyde International Corporation

TOUCH (*filename* in ALPHA)

The **TOUCH** (*Update File Datestamp and Timestamp*) function updates the directory entry for the specified file with the current date and time. If the ALPHA register is empty the volume header is updated.

TS? (*filename* in ALPHA)

The **TS?** (*Report Timestamp*) function reads the directory entry for the specified file and extracts the *timestamp*. If the ALPHA register is empty the volume header is read. The *timestamp* is returned in the X-register in *hh.mmss* format.

© 2026 Systemyde International Corporation

File Operations

The Store/Retrieve File functions transfer file contents between the volume and HP-41C memory. Which HP-41C memory is used depends on the file type. These functions can be used with any file types, although there are dedicated functions for file types **DA** and **PR**.

Specifying a *filetype* of **DA** automatically invokes the dedicated functions for data files. So in this case **FWR** branches directly into the **DFWR** function and **FRD** branches directly to the **DFRD** function.

Likewise, specifying a *filetype* of **PR** automatically invokes the dedicated functions for program files. So in this case **FWR** branches directly into the **PFWR** function and **FRD** branches directly to the **PFRD** function.

A *filetype* of **RM**, **HE** or **ML** requires a *parameter* in the X-register to select a page of instruction memory. Only Bank 1 of the selected page can be accessed by these functions. A page number in the range of 1 to 16 is valid.

A *filetype* of **YM** requires a *parameter* in the X-register to select a bank of Expanded Memory. A bank number in the range of 1 to 3 is valid.

FRD (*parameter* in X-register, *filename.filetype* in ALPHA)

The **FRD** (*Read File*) function reads a file and writes it to HP-41C memory as appropriate for the type of file.

Reading a file that was saved on a calculator with different memory sizes, whether main memory or Extended memory, will result in an error message.

For file types that write to instruction memory this function does not check whether the selected page exists in the calculator or if it is enabled for writes. Only a calculator with QROM in the selected page can be written from these types of files.

FWR (*parameter* in X-register, *filename.filetype* in ALPHA)

The **FWR** (*Write File*) function reads HP-41C memory as appropriate for the type of file and writes it to the volume. A file with *filename* must not already exist, or an error message will result. The file with name *filename.filetype* will be created by this function.

For file types that read instruction memory this function does not check that the selected page exists in the calculator. A nonexistent page will be written as a file of all zeros.

© 2026 Systemyde International Corporation

Program File Operations

The Program File Operation functions transfer program (**PR**) files between the volume and HP-41C memory. Dedicated functions allow different actions to be taken when a file is read and different parameter types used for file writes. With these functions the *filetype* never needed.

PFRD (*filename* in ALPHA)

The **PFRD** (*Read Program*) function operates identically to the **READP** function in the 82160A HP-IL module, copying a program stored in a file to the HP-41C program memory. The copied program replaces the last program in program memory. Since packing creates a blank program at the end of program memory, doing a pack before executing this function means the copied program will replace the blank program and leave all other existing programs unaffected.

Executing **PFRD** leaves the calculator at its position in program memory unless it was positioned in the last program, in which case it is positioned at the first line of the new program.

PFRDS (*filename* in ALPHA)

The **PFRDS** (*Read Program as Subroutine*) function operates identically to the **READSUB** function in the 82160A HP-IL module, copying a program stored in a file to the HP-41C program memory. The copied program is loaded after the last program in program memory. This function can be used from within a program to load and execute a subroutine before continuing the main program. This function does not change to position of the calculator in program memory.

© 2026 Systemyde International Corporation

PFWR	**(*prgmname,filename* in ALPHA)** OR **(,*filename* in ALPHA)** OR **(*prgmname* in ALPHA)**

The **PFWR** (*Write Program File*) function operates identically to the **WRTP** function in the 82160A HP-IL module, copying a program from HP-41C program memory to a file. If *filename* already exists the contents of the file will be replaced by the program. Otherwise a new file will be created. If the calculator is positioned within the program then *prgmname* does not need to be specified, but the comma must still be present. If *filename* is not specified it will be identical to *prgmname*. Any key assignments for labels within the program are included in the file.

PFWRP	**(*prgmname,filename* in ALPHA)** OR **(,*filename* in ALPHA)** OR **(*prgmname* in ALPHA)**

The **PFWRP** (*Write Program File - Private*) function is identical to the **PFWR** function except the file is marked as *Private* in the directory. A Private program can only be retrieved and executed. It cannot be listed, edited or stored elsewhere.

© 2026 Systemyde International Corporation

Data File Operations

The Data File Operation functions create, read or write Data (**DA**) files. Data files are treated differently from other types of files because access to only part of the file is allowed.

The functions that allow partial file access (**DFRDX**, **DFSEEK** and **DFWRX**) can also be used with an Expanded Memory (**YM**) file.

DFNEW (*filesize* in X-register, *filename* in ALPHA)

The **DFNEW** (*Create Data File*) function creates a new file with the specified name and size. The file is filled with all zeros, which may take some time. The *filesize* in the X-register is the number of registers to be allocated in the file. Each register requires eight bytes.

The HP-41 Virtual Tape Drive software allows for a *filesize* of up to 4095 registers, but does check that the volume will support the requested size.

If the specified *filename* already exists no action is taken and the function signals an error. The reason for this lies in the format of the directory, which does not allow the size of a file to be increased. In theory a file could be made smaller, but it is simpler to just disallow file size changes.

DFRD (*filename* in ALPHA)

The **DFRD** (*Read Registers*) function copies all of the data in a **DA** file to HP-41C register memory, starting with R_0. Writing continues until all file registers have been copied or until all destination registers are filled.

DFRDX (*bbb.eee* in X-register)

The **DFRDX** (*Read Registers according to X*) function copies data from the file selected by the File Pointer to the specified registers in the calculator. Registers are read from the file starting with the register selected by the Register Pointer and are written starting with R_{bbb} and ending with R_{eee}. If *eee* is less than *bbb* only the first register will be written.

The register pointer increments after each transfer, so it will point at the next sequential register in the file after the function completes. Transfers continue until *eee* is reached or until memory is full.

© 2026 Systemyde International Corporation

DFSEEK (*register* in X-register, *filename* in ALPHA)

The **DFSEEK** (*Seek Register*) function initializes both a File Pointer and a Register Pointer to save time and effort when accessing registers within a file.

The Register Pointer sets the starting register within the file for read or write. Registers in a file are numbered starting with R_0 as the first register.

This function can be used with either **DA** or **YM** file types. Because a *filename* is unique, the *filetype* does not need to be specified, but this function does check for a *filetype* of **DA** or **YM**.

DFWR (*filename* in ALPHA)

The **DFWR** (*Write Registers*) function either copies all of HP-41C register memory, starting with R_0, to an existing **DA** file. The file is filled starting at the beginning of the file. If the file size is not sufficient copying stops when the file is full.

DFWRX (*bbb.eee* in X-register)

The **DFWRX** (*Write Registers according to X*) function copies data from the specified registers in the calculator to the file selected by the File Pointer. Registers are read starting with R_{bbb} and ending with R_{eee} to the file starting with the register selected by the Register Pointer and are written starting. If *eee* is less than *bbb* only the first register will be written.

The register pointer increments after each transfer, so it will point at the next sequential register in the file after the function completes. Transfers continue until *eee* or the end of memory is reached.

© 2026 Systemyde International Corporation

Error Messages

Error Message	Meaning
ALPHA DATA	X-register contains ALPHA data
CKSUM ERR	**PG** file checksum error
DATA ERROR	Data is out of range
DIR EXISTS	Volume is already initialized
DIR FULL	Directory is full
DUP FL NAME	File name already exists
END OF FILE	Reached end of file
FL NOT FOUND	No matching file name found
FL SECURED	File is secured
FL TYPE ERR	File is the wrong type
MM SIZ ERR	Main memory size mismatch
NAME ERR	Illegal file or volume name
NO DIR	Volume is not initialized
NO KEYS	No key assignments found
NO HP-IL	No HP-IL Module found
NO ROOM	Program too large for available memory
NO X-MEM	No X-Memory found
NONEXISTENT	Specified page does not exist
PRIVATE	Cannot save Private program
ROM	Specified program is in ROM
SIZE ERR	Size mismatch in **ST** file
VOL FULL	Volume is full
XM SIZ ERR	X-memory size mismatch

© 2026 Systemyde International Corporation

Function XROM Numbers

References to functions are stored in HP-41 programs as XROM numbers. This technique is not visible to the user as long as the module containing the function is present in the calculator, because the HP-41 software automatically looks up the function name and displays this name in a program line. If the module is removed the XROM numbers become visible in program lines. The XROM numbers for the HP-41 Virtual Tape Drive software are listed below.

Function	XROM Number	Function	XROM Number
DFNEW	XROM 18,01	FUNSEC	XROM 18,18
DFRD	XROM 18,02	FWR	XROM 18,19
DFRDX	XROM 18,034	FZERO	XROM 18,20
DFSEEK	XROM 18,04	PFRD	XROM 18,21
DFWR	XROM 18,05	PFRDS	XROM 18,22
DFWRX	XROM 18,06	PFWR	XROM 18,23
DS?	XROM 18,07	PFWRP	XROM 18,24
FDEL	XROM 18,08	TOUCH	XROM 18,25
FEXP	XROM 18,09	TS?	XROM 18,26
FIMP	XROM 18,10	VDIR	XROM 18,27
FLOC?	XROM 18,11	VFMT	not programmable
FRD	XROM 18,12	VLBL?	XROM 18,29
FREC?	XROM 18,13	VPACK	XROM 18,30
FREN	XROM 18,14	VREC?	XROM 18,31
FSEC	XROM 18,15	VREM?	XROM 18,32
FSIZ?	XROM 18,16	VSIZ?	XROM 18,33
FTYPE?	XROM 18,17		

© 2026 Systemyde International Corporation

Internal Details

TBD

© 2026 Systemyde International Corporation

Revision History

10/30/2026 First public release.

© 2026 Systemyde International Corporation