The Vinculum Utilities
For the H8 and H89 Computer

Version 4.0
February 2022
Glenn Roberts

This document describes a set of programs called the Vinculum Utilities, written for use with Heathkit computers equipped with the VDIP1 module [1] from Future Technology Devices International (FTDI). The VDIP1 has been incorporated in several recent board enhancements for the H8 and H89 series of Heathkit computers [3, 4, 5, 6, 7]. The Vinculum Utilities provide tools for copying files to and from USB flash drives and organizing and listing the contents of the flash drive.

The Vinculum Utilities currently consist of six programs, which have been compiled and tested on all major Heathkit software platforms (HDOS 2, HDOS 3, CP/M 2.2, CP/M 3 and MP/M). By convention the program names all start with the letter "V". Briefly they are:

Utility	Function
VCD	Change the current directory on the USB flash device.
VDIR	List the file contents of the currently selected directory on the USB flash device.
VGET	Retrieve a file from the USB flash device to the local file system.
VPUT	Send a file from the local file system to the USB flash device.
VTALK	Communicate directly to the Vinculum firmware via the command set.
VPIP	A Peripheral Interchange Program modeled after the PIP utilities supplied with CP/M and HDOS. VPIP recognizes wild card file names, allowing for selected multi-file transfers and directory listings based on file naming patterns.

All the utilities support one or more switches, which determine how the utility should behave. Switches are specified on the command line when invoking the utility. They are designated by the "-" character and can contain an optional parameter. Switches must be to the right of all other arguments expected by the utility. For example, all the utilities allow you to override the default base port number for the VDIP1 device via the "-p" switch. Examples are shown in the following sections.

The programs are all written in the C language and compiled with the Software Toolworks C/80 compiler [8]. One beneficial side effect of this is that program input and output can be redirected using Unix-style "<" and ">" redirection. For example, to create a file containing the contents of the flash drive you can perform:

```
VDIR >CONTENTS.TXT
```

You can also redirect to a print device using this technique. VTALK, which uses system calls and direct I/O with the UART, does not support redirection.

In general, any of the programs can be interrupted by using the Control-C character, however CP/M is a bit fussy about when it checks for these interrupts. Special system calls have been strategically placed in the code to try and watch for Control-C requests, however there may be occasions where you can't interrupt a program under CP/M.

The following are descriptions of each of the six Vinculum utilities. The examples shown are for a CP/M system, but HDOS commands are the same except for the use of HDOS style device descriptors (e.g. SY0: instead of A:).

VCD

The Vinculum firmware is designed to be used with flash drives formatted the Microsoft File Allocation Table (FAT) structure using a 12-, 16-, or 32-bit field for the cluster count. Most common flash drives today are formatted for FAT32. The firmware supports only 8.3 file naming conventions (an 8-character file name and 3-character extension).

Flash drives also can have subdirectories, of course. While HDOS and CP/M do not have subdirectory capability it is possible to make use of them for file copy and backup ability by changing the directory on the flash drive. That is the purpose of the VCD routine.

Syntax:

```
VCD path {-pxxx}
```

Where path is a list of one or more subdirectories on the USB flash drive, separated by forward slash characters ('/'). If the first character is '/' it is a rooted path (beginning at the root level), otherwise it is relative to the current directory. The -p option may be used to specify an alternate I/O base port in octal. The default is 331.

VCD only supports Unix-style forward slashes for directory specification.

When you insert a flash drive in the VDIP1 device, or when you reset or reboot the computer, the Vinculum firmware will be initialized to view files in the root subdirectory on the device. You can use VCD to change that directory.

Examples showing changing directories and listing the contents (VDIR will be discussed in the next section):

```
A>vcd /games
VCD v4 [331]
USB:/GAMES

A>vdir -b
VDIR v4 [331]
.        <DIR>  ..       <DIR>  COMPOSE .ABS    PLAY    .ABS
SPACEWAR.ABS    LIFE    .ABS   MASTMIND.BAS     BATLSHIP.BAS
README  .DOC    REVERSI .DOC   REVERSI .ASM     REVERSI .ABS
PINBALL .DOC    PINBALL .ABS   LINES   .BAS     DIET    .BAS
VEGAS   .BAS    OTHELLO .DOC   OTHELLO .BAS     BIORYTHM.BAS
GRWUMPUS.BAS    MUSICK  .ASM   MUSICK  .ABS     MUSICBOX.PLA
GLOBE   .ASM    GLOBE   .ABS   GLOBE   .DAT     GLOBE   .BAS
BLKJCK  .ASM    BLKJCK  .ABS   DOODLE  .ASM     DOODLE  .ABS
AFLAG   .ASM    RGT     .SYS   GRT     .SYS     DIRECT  .SYS
AFLAG   .ABS    HDOS    .ACM   MTR     .ACM     VDPDEF  .ACM
WVDPX   .ACM    IVDP    .ACM   UDDN    .ACM     CDB     .ACM
MCU     .ACM    TYPCC   .ACM   WARLORDS.COM     WARLORDS.SPR


46 Files

A>vcd ../test
VCD v4 [331]
USB:../TEST

A>vdir -b
VDIR v4 [331]
.       <DIR>  ..       <DIR>  HELLO   .C       HELLO   .MAC
HELLO   .REL   HELLO   .ABS   HELLO   .TXT

5 Files
```

VDIR

The VDIR utility lists a directory of all the files on the currently selected subdirectory of the USB flash drive. The program must make a two-pass reading of the flash drive – the first pass detects the file names and builds an internal table to store them, and the second pass queries each file name for more detailed information. This is a constraint of the Vinculum command set and firmware. The time to respond will increase, of course, with the number of files on the flash drive.

Syntax:

```
VDIR {-b} {-pxxx}
```

The -b switch requests a "brief" directory listing. In "brief" mode just the file names are listed, four to a column. This requires only a single pass and is much faster if all you want is the names of the files. The -p option may be used to specify an alternate I/O base port in octal. The default is 331.

One way to make file and directory operations speedier is to make use of subdirectories so that no one directory has a large number of files. VDIR currently can handle at most 400 files in any flash directory. If you exceed that number, you will receive a warning message and VDIR will list only the first 400 files found.

VDIR does not take any path or wild card arguments. It lists all the files in the currently selected subdirectory on the USB flash drive. If you want a directory listing of only certain files on the flash drive (e.g., all the .ASM files, for example) you need to use VPIP which has a -l switch for this purpose (see the directions for VPIP later in this document.)

VDIR Example:

```
A>vdir
VDIR v4 [331]
.        <DIR>
..       <DIR>
HELLO    .C                   256    2/06/22    7:44 AM
HELLO    .MAC                 256    2/06/22    7:44 AM
HELLO    .REL                 256    2/06/22    7:44 AM
HELLO    .ABS               4,608    2/06/22    7:44 AM
HELLO    .TXT                 256    2/06/22    7:44 AM

5 Files
```

VGET

VGET and its companion utility VPUT (below) are designed for quick file transfers of a small number of files. They are fast but not as flexible or powerful as VPIP (discussed later).

VGET is used to retrieve a file from the current subdirectory on the flash drive and copy it to the local file system on the H8 or H89. It can only retrieve one file at a time, and you must know the name (in 8.3 format) of the file (use VDIR first to see the names on the flash drive).

Syntax:

```
VGET source {dest} {-pxxx}
```

source is the name of the file on the flash drive, and dest is an optional destination file specification. If only source is provided the file will be created on the current local default drive (or SY0: for HDOS). VGET will overwrite any file with that same name without warning. The destination can either be a drive specifier (e.g. B: or SY1:) or a file name, or a drive and file name. As with all the Vinculum utilities the -p option may be used to specify an alternate I/O base port in octal. The default is 331.

VGET is only for single file transfer, but it is very fast. For transferring multiple files with a single command use VPIP.

VGET usage example (go to subdirectory /TEST and fetch HELLO.C to the current drive, and copy a backup to L:HELLO.BAK):

```
A>vcd test
VCD v4 [331]
USB:TEST

A>vdir -b
VDIR v4 [331]
.        <DIR>  ..        <DIR>  HELLO   .C       HELLO    .MAC
HELLO   .REL     HELLO   .ABS     HELLO   .TXT

5 Files

A>vget hello.c
VGET v4 [331]
USB:HELLO.C                     256 bytes --> HELLO.C

A>vget hello.c l:hello.bak
VGET v4 [331]
USB:HELLO.C                     256 bytes --> L:HELLO.BAK

A>dir l:
L: HELLO    BAK
A>
```

VPUT

VPUT is the counterpart to VGET and is used to transfer one or more files from the H8/H89 to the USB flash drive. One difference is that VPUT allows multiple files to be specified on the command line, separated by spaces. Unlike VGET you cannot specify the destination file name or location, so if you want to put files into a certain subdirectory you must use VCD first to point there. The destination file name will be the same as the source file name, and any existing file on the flash drive with that name will be overwritten.

Syntax:

```
VPUT file_1 {file_2} … {file_n} {-pxxx}
```

Where file_1 through file_n are the names of files to be transferred. If just the file names are given VPUT will look on the current drive (or SY0: for HDOS), but the names can be fully qualified with drive specifications. As with all the Vinculum utilities the -p option may be used to specify an alternate I/O base port in octal. The default is 331.

The files are processed sequentially, left to right, so if you have a duplicate file name the rightmost one will be what remains on the flash drive once the command is complete, e.g.

```
VPUT A:MYPROG.FOR B:MYPROG.FOR
```

Would cause the version on the B: drive to overwrite the first transfer (the A: version).

Note: the CP/M version of VPUT allows wild cards in the file description, for example:

```
VPUT *.BAS
```

This is because CP/M have an easy-to-use "find first match" … "find next match" set of BDOS calls (functions 17 and 18) that search the directory, and the C/80 product encapsulates these in a routine called command(). Adding a single line to the VPUT source code, therefore, allows for handling wild card matching. For HDOS this would be much harder as it would require reading and parsing DIRECT.SYS and adding code to search for matches. This capability exists in VPIP, which is generally the preferred way to handle wild card file name transfers.

To ensure that a meaningful date and time are recorded on the flash drive VPUT will attempt to determine the information by one of two methods. In the case of CP/M 3 and MP/M the operating system has formal support for time and date functions and VPUT calls the appropriate BDOS call to look it up. HDOS has a more informal approach. In HDOS there are defined memory locations for date and time, but those values are not maintained in real time unless an appropriate device driver (e.g. the HUG-developed CK:) is loaded. CP/M 2.2 does not have built in date and time functions, though add-ons such as DSLIB have been developed for that purpose.

VPUT does not rely on these add-ons but rather looks directly for a real time clock. Currently the only real time clock chip it looks for is the Epson 72421, which has been used in all recent real time clock solutions. The base port number is assumed to be octal 240.

If there is no OS support for time/date and a real time clock chip cannot be found, the Vinculum firmware will use 2004-12-04 00:00:00 for the file creation date and time.

VTALK

VTALK is a simple terminal communication program that connects the H8/H89 console directly to the VDIP1 device. This utility was written initially for development and testing purposes but offers sufficient value to be included here for general use.

The Vinculum firmware offers an ASCII command set for communicating with the VDIP1 device. The commands are defined in the FTDI Vinculum Firmware User Manual [2].

Syntax:

```
VTALK {-pxxx}
```

As with all the Vinculum utilities the -p option may be used to specify an alternate I/O base port in octal. The default is 331. To exit VTALK you type control-C.

There should be little need for VTALK, as most common functions have been encapsulated in the Vinculum CP/M and HDOS utilities, however occasionally there can be a need to verify that things are functioning normally. The standard prompt when you are talking to the Vinculum firmware is "D:\". If you use the firmware command CD to change the current directory on the flash drive it will remain that way after you exit VTALK. One possible use of VTALK would be to create a new subdirectory on the USB flash drive (via the MKD command), as the current set of CP/M and HDOS Vinculum utilities does not support this.

The HDOS version of VTALK bypasses the operating system and talks directly to the UART. It currently only supports the 8250 UART, which limits its use to the H89 or the H8 with H8-4 serial I/O capability. The earlier H8-5 serial board uses the 8251 USART, which the software won't recognize (it will print an error message and exit).

Sample session (uses the Vinculum firmware command MKD to create a subdirectory BACKUP in /TEST and then, from the CP/M prompt, changes into that directory via VCD and copies a file there via VPUT):

```
A>vtalk
VTALK v4 [331]
Enter Vinculum commands, Ctrl-C to exit

D:\>cd test
D:\>dir

. DIR
.. DIR
HELLO.C
HELLO.MAC
HELLO.REL
HELLO.ABS
HELLO.TXT
D:\>mkd backup
D:\>              ←  (Control-C hit to exit VTALK)
A>vcd backup
VCD v4 [331]
USB:BACKUP

A>vput hello.c
VPUT v4 [331]
02-07-22 16:17:09 HELLO.C           --> USB:HELLO.C
256 bytes
```

VPIP

VPIP is patterned after the Peripheral Interchange Programs (PIP) provided with CP/M and HDOS. It allows files to be copied to/from the USB flash device using powerful wild card name matching techniques. It also lets you do a directory listing using wild cards.

Like VDIR, VPIP first internally builds a directory structure of the source drive (either the USB device or the local drive). The time required to do this increases with the number of files on the device but can be lengthy, especially on systems running at lower CPU clock speeds (e.g. 2Mhz). As a reminder to the user that it is "thinking", VPIP prints the message "Standby - cataloging USB file details..." while it is assembling the source drive's directory structure. Like VDIR, VPIP has a limit on how many files can be in the source drive. Currently this number is 400. The user will be warned if this number is exceeded, and some files will be omitted from operation.

As with all the Vinculum utilities the -p option may be used to specify an alternate I/O base port in octal. The default is 331.

When copying files from the H8/H89 system to the USB flash drive VPIP uses the same time/date stamping rules as VPUT (see above).

You can run VPIP two ways: with a single command provided on the command line, or interactively. To run VPIP interactively type VPIP at the command prompt:

```
>VPIP
:V:
```

The `:V:` prompt will be displayed at the left margin of the system console whenever the VPIP program is awaiting input. To exit VPIP simply enter a blank line.

VPIP refers to the USB device via a "pseudo device" designated USB:. In VPIP commands this looks and acts like a CP/M or HDOS device would.

Copying Files

The general form of the command for copying files specifies a "destination" followed by an "=" and then one or more "source" specifications:

```
:V:x:DESTINAT.EXT=USB:SOURCE.EXT
```

or

```
:V:USB:DESTINAT.EXT=x:SOURCE.EXT
```

Where 'x' is a drive designator (e.g. A: in CP/M or SY0: in HDOS). VPIP can only be used to copy files from a H8/H89 storage device to the USB device *or* from the USB device to a Heath storage device. As an example:

```
:V:USB:MYPROG.BAK=A:MYPROG.FOR

1 Files Copied
```

In this case, the destination is a file named MYPROG.BAK on the USB device and the source file is a file called MYPROG.FOR, located on the A: drive.

You can omit storage device specifications and VPIP will attempt to do the right thing. For example:

```
:V:A:*.*=MYPROG.C
```

Will cause VPIP to assume that the USB device is the source device (since the destination is a CP/M disk), and will look on the USB device for the program MYPROG.C and copy it to A:. If you specify only the USB device and not the system device, VPIP will assume the default drive on the H8/H80 (e.g. SY0: for HDOS or the current default for CP/M) for example:

```
:V:*.*=USB:MYPROG.C
```

Will look on the USB device for a file MYPROG.C and copy it to the current default drive.

If you omit both source and destination devices VPIP acts like VGET and assumes the source device is the USB drive and the destination device is the current default drive. For example:

```
:V:=*.c
```

Would copy all files on the USB device matching the file specification "*.c" to the current drive. *It is important to note that VPIP currently does not check whether a file already exists*, so the above command would (without any warning) *overwrite* (replace) any existing files on the default drive with files of the same name on the USB device.

The following are some examples of *illegal* VPIP commands:

Command	*Reason for being illegal*
`USB:*.*=USB:TEST.*`	USB to USB transfer not supported
`A:TEST.DAT=B:MYTEST.DAT`	Either source or destination needs to be USB:
`TT:=USB:MYPROG.C`	VPIP can only copy to/from storage class devices

Wildcards and Multiple File Designation

Wildcards

The "*.*" wildcard is another way of accessing multiple files. A "*" can be substituted for the file name or extension portion of a file specification, for example:

```
B:*.EXT
```

or

```
USB:FNAME.*
```

or

```
A:*.*
```

are all valid uses of the "*" wild card. You can also use "*" to complete a field. For example

```
USB:V*.*
```

Will match any file on the USB flash drive that starts with the letter "V.

The "?" wild card can be used to match single letters in a portion of a file name. For example

```
CHAPTER?.DOC
```

Will match `CHAPTER1.DOC`, `CHAPTER2.DOC`, etc…

If you use "?" in a portion of a file designation you must use at least as many "?"s as there are characters in the name of the file you want to match. Thus

```
????.*
```

will match all files whose name contains *four or fewer* characters in the name portion of the filename. The file specification "????????.???" is identical to "*.*".

As a convenience, when copying files from one device to another you may omit the "*.*" altogether for the destination device – it will be implied. For example

```
:V:USB:=*.C
```

Will copy all "C" files on the current drive to the USB drive.

Listing Files (Directory)

VPIP also provides a way to simply list files on either the CP/M drive or the USB drive. This feature only works from the command line (not the interactive prompt). To specify a listing request include the switch "-l" (separated by at least one space.) For example:

```
VPIP USB:*.* -l
```

You can list local drive contents as well, for example:

```
VPIP B:*.* -l
```

General Notes on the Software Design

The programs are all written in the C language and compiled with the Software Toolworks C/80 compiler, Rev. 3.1 (with support for Floats and Longs). There is a single source file for each utility, however each source file can be compiled to produce either an HDOS or a CP/M version. If the source contains a "#define HDOS 1" then the HDOS code is compiled, otherwise the CP/M code is compiled. The program determines at run time what OS version it is running on and executes the appropriate code.

The C/80 compiler converts C source code to assembly language format (.MAC) which can be assembled using the Microsoft MACRO80 assembler (M80). MACRO80 produces relocatable (.REL) files for each component. The Microsoft linker, L80, is used to resolve all linkages and produce an executable image (.ABS for HDOS; .COM for CP/M).

General purpose routines are contained in three library files: VINC, VUTIL, and PIO. VINC includes the core routines to communicate with the VDIP1 via the command set. VUTIL contains support and utility code covering four broad types of functions: operating system functions, format conversion, string functions and time/date functions. PIO is an assembly language implementation of port input and output routines, e.g.:

```
outp(port,c);    /* output byte c to port  */
c = inp(port);   /* input byte c from port */
```

Modules are compiled and assembled individually, e.g.:

```
A> C vget
A> M80 =vget
```

The final executable is produced by L80:

```
A> L80 VGET,VUTIL,VINC,PIO,FPRINTF,FLIBRARY/S,STDLIB/S,
   CLIBRARY/S,VGET/N/E
```

Optionally, to simplify the link process, these .REL files can be combined into a single library file using the Microsoft librarian tool.

References

[1] VDIP1- Vinculum USB Host/Device Controller Development/Prototype Module, Future Technology Devices International, Ltd., 2008, https://ftdichip.com/products/vdip1/.

[2] Vinculum Firmware User Manual, Future Technology Devices International, Ltd., 2008, https://www.ftdichip.com/Firmware/Precompiled/UM_VinculumFirmware_V205.pdf.

[3] Norberto Collado, H8 USB Board, http://koyado.com/Heathkit/H-8_USB.html

[4] Norberto Collado, H-89 Seria/USB-FT245R/VDIP1 Controller, http://koyado.com/Heathkit/H-89_USB_Serial.html

[5] Norberto Collado, H8 Storage Controller, http://koyado.com/Heathkit/H8-H17-H37-H67-USB.html

[6] Norberto Collado, Heathkit H8 Z80 CPU Board V3.1, http://koyado.com/Heathkit/H8-Z80-64K-RTC-ORG0-V3.html

[7] Norberto Collado, Heathkit H8 Z80 CPU Board V4, http://koyado.com/Heathkit/H8-Z80-64K-RTC-ORG0-V4.html

[8] Mark Garlanger, The Software Toolworks, https://heathkit.garlanger.com/software/library/TheSoftwareToolworks/

[9] Wikipedia, File Allocation Table, https://en.wikipedia.org/wiki/File_Allocation_Table