**The Vinculum Utilities**

*For the H8 and H89 Computer*

*Version 4.0*

*February 2022*

*Glenn Roberts*

## Introduction

Modern personal computers include Universal Serial Bus (USB) interfaces that allow the user to easily back up and transfer files using USB “flash” drives (also known as “memory sticks”) and other USB storage devices. Hobbyists who collect, use, and restore legacy computers, such as the Heathkit H8 and H89, have historically not had such a capability since the USB protocols are complex and not easily implemented on smaller 8-bit platforms (and didn’t even exist back in the H8/H89 days). The removable storage, file interchange, and backup solution originally developed for these systems was the floppy disk, but 40+ year-old floppy disk media and disk drives can be quite unreliable in the modern day.

Beginning around 2013 the Society of Eight-Bit Heathkit Computerists (SEBHC) began to develop hardware and software solutions that enable these classic machines to use USB flash drives. This document primarily describes the software but provides links to key reference documents for the hardware designs as well.

## The FTDI VDIP-1

The main advancement that enabled this capability was a product called the VDIP1 from Future Technology Devices International (FTDI). Here is a brief, edited description of the device based on material taken from the FTDI datasheet:

*“The VDIP1 module is a microcontroller-to-embedded USB host controller development module. It is supplied on a printed circuit board designed to fit into a 24-pin dual in-line package (DIP) socket. The module is based on the Vinculum VNC1L integrated circuit (IC), which is the first of FTDI’s Vinculum family of embedded USB host controller ICs.”*

*“Not only is it able to handle the USB Host interface, and data transfer functions, but owing to the built-in microcontroller and embedded flash memory, Vinculum can encapsulate the USB device classes as well. When interfacing to mass storage devices such as USB flash drives, Vinculum also transparently handles the File Allocation Table (FAT) file structure, communicating via* *universal asynchronous receiver-transmitter (UART), Serial Peripheral Interface (SPI) or parallel first-in first-out (FIFO) interfaces via a simple command set. Vinculum provides a new cost-effective solution for providing USB Host capability into products that previously did not have the hardware resources available.”*

The key point is: this module and firmware handle all the complex details of the USB interface protocol. The SEBHC designs all provide for interfacing to the module via the parallel FIFO interface. The VDIP1 is mounted on a PC board and the interfacing circuitry provides for direct access via IN and OUT instructions. The “command set” referred to in the above description is a set of short ASCII commands that the Heathkit host uses to talk to the VDIP1. These are documented in the Vinculum Firmware Manual.

In order to perform useful operations with the VDIP1 (e.g. file transfer) it was necessary to develop a set of utilities for the Heathkit system to communicate to the VDIP1 via the command set. The software to do this was developed by Glenn Roberts and has evolved and improved with input from the SEBHC community. These utilities are described in the remainder of this document.

## The Vinculum Heathkit Utilities

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 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.

I/O redirection…

## 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.

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

A:VCD COM

VCD v4 [331] - G. Roberts

USB:/GAMES

A>vdir -b

A:VDIR COM

VDIR v4 [331] - G. Roberts

. <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

A:VCD COM

VCD v4 [331] - G. Roberts

USB:../TEST

A>vdir -b

A:VDIR COM

VDIR v4 [331] - G. Roberts

. <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 256 files in any flash directory. If you exceed that number you will receive a warning message and VDIR will list only the first 256 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

A:VDIR COM

VDIR v4 [331] - G. Roberts

. <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 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 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

A:VCD COM

VCD v4 [331] - G. Roberts

USB:TEST

A>vdir -b

A:VDIR COM

VDIR v4 [331] - G. Roberts

. <DIR> .. <DIR> HELLO .C HELLO .MAC

HELLO .REL HELLO .ABS HELLO .TXT

5 Files

A>vget hello.c

A:VGET COM

VGET v4 [331] - G. Roberts

USB:HELLO.C 256 bytes --> HELLO.C

A>vget hello.c l:hello.bak

A:VGET COM

VGET v4 [331] - G. Roberts

USB:HELLO.C 256 bytes --> L:HELLO.BAK

A>dir l:

L: HELLO BAK

A>

## VPUT

VPUT is used to transfer one or more files from the H8/H89 to the USB flash drive.

## VTALK

VTALK is a simple terminal communication program that connects the console directly to the VDIP1 device to allow for configuration and testing.

## VPIP

The purpose of the VPIP program is to provide a convenient interface to the USB device from the CP/M command prompt. This version, written expressly for CP/M 3, takes advantage of features of the OS such as file time/date stamping.

VPIP is patterned after the PIP utility provided with CP/M, however it only implements disk-to-disk copy and file listing (directory) commands. You’ll still need PIP to delete, rename or copy files to the screen or printer. Similarly to manipulate (copy, delete, rename, etc.) files on the flash drive you’ll need to use a personal computer or other device. VPIP currently can only be used to access *storage* devices via the USB port, not printers, keyboards or other accessories.

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, type CTRL-C, CTRL-Z or 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 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 one-letter CP/M drive designator. VPIP can only be used to copy files from a CP/M storage device to the USB device *or* from the USB device to a CP/M 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 an HDOS 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 SY0:, 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 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 currently VPIP currently does not check whether a file already exists*, so the above command would (without any warning) *overwrite* (replace) any existing CP/M 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”. If the “\*” character starts a name or extension field *the characters beyond it in that field will be ignored*. So

A:\*V.\*

Is the same as

A:\*.\*

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

or

VPIP B:\*.\* -l

**Specifying an Alternate Port**

VPIP assumes the USB card is set to use ports 331/332 (octal). If your card is set to a different port number you can specify that on the command line using the “-p” switch and specifying the port number of the *first* of the paired I/O ports *in octal* with no space between the switch and the port number, for example:

VPIP USB:=B:\*.C -p261

If you specify the incorrect port number the utility will time out after several attempts and print the message:

Error initializing VDIP-1 device!

***Notes on Version 3.2***

1. Files copied to the USB device are time stamped with the current time and date returned by the CP/M 3 BDOS.
2. If a CP/M file system becomes full during a copy operation, the last file written will be closed even though it is only partially copied.
3. There is currently no special handling of Control-C. If you Control-C during a file transfer you may leave the USB interface in an unknown state that requires a reboot.

**Future Plans**

1. Support for file concatenation
2. Support for text devices (e.g. TT:, LP:, etc.)
3. Support for subdirectories on the USB device.
4. Warning before overwriting files.
5. Better error handling (including capturing Control-C interrupts).
6. Ability to use CP/M file time/date stamp (either “access” or “update”) when transferring to the USB device (useful for tracking backups).

## 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” or if the flag -qHDOS=1 is specified at compile time then the HDOS code is compiled, otherwise the CP/M code is compiled. The code 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. 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.:

1. C vget
2. M80 =vget

The final executable is produced by L80:

1. 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.

Time and date functions

I/O redirection

Control-c

Directory file limit

Memory requirements

OS versions

Incompatibility with certain USB sticks?