MARCH 1992

WRL
Technical Note TN-26

TurboChannel Versatec Adapter

David Boggs

Western Research Laboratory 250 University Avenue Palo Alto, California 94301 USA

The Western Research Laboratory (WRL) is a computer systems research group that was founded by Digital Equipment Corporation in 1982. Our focus is computer science research relevant to the design and application of high performance scientific computers. We test our ideas by designing, building, and using real systems. The systems we build are research prototypes; they are not intended to become products.

There is a second research laboratory located in Palo Alto, the Systems Research Center (SRC). Other Digital research groups are located in Paris (PRL) and in Cambridge, Massachusetts (CRL).

Our research is directed towards mainstream high-performance computer systems. Our prototypes are intended to foreshadow the future computing environments used by many Digital customers. The long-term goal of WRL is to aid and accelerate the development of high-performance uni- and multi-processors. The research projects within WRL will address various aspects of high-performance computing.

We believe that significant advances in computer systems do not come from any single technological advance. Technologies, both hardware and software, do not all advance at the same pace. System design is the art of composing systems which use each level of technology in an appropriate balance. A major advance in overall system performance will require reexamination of all aspects of the system.

We do work in the design, fabrication and packaging of hardware; language processing and scaling issues in system software design; and the exploration of new applications areas that are opening up with the advent of higher performance systems. Researchers at WRL cooperate closely and move freely among the various levels of system design. This allows us to explore a wide range of tradeoffs to meet system goals.

We publish the results of our work in a variety of journals, conferences, research reports, and technical notes. This document is a technical note. We use this form for rapid distribution of technical material. Usually this represents research in progress. Research reports are normally accounts of completed research and may include material from earlier technical notes.

Research reports and technical notes may be ordered from us. You may mail your order to:

Technical Report Distribution
DEC Western Research Laboratory, WRL-2
250 University Avenue
Palo Alto, California 94301 USA

Reports and notes may also be ordered by electronic mail. Use one of the following addresses:

Digital E-net:	`DECWRL::WRL-TECHREPORTS`
Internet:	`WRL-Techreports@decwrl.dec.com`
UUCP:	`decwrl!wrl-techreports`

To obtain more details on ordering by electronic mail, send a message to one of these addresses with the word ‘‘`help`’’ in the Subject line; you will receive detailed instructions.

TurboChannel Versatec Adapter

David Boggs

Abstract

This note is the technical reference for the VTEC adapter, a Versatec plotter option module for TurboChannel I/O systems. It implements the Versatec Parallel Interface (VPI) and can control all models of Versatec plotters.

This adapter is a research prototype; it is not a product.

digital **Western Research Laboratory** 250 University Avenue Palo Alto, California 94301 USA

1. Introduction

The VTEC adapter is a Versatec plotter option module for TurboChannel I/O systems. It implements the *Versatec Parallel Interface* (VPI) and can control all models of Versatec plotters. The adapter was designed to allow a workstation to act as a plotter server for a network of other workstations. This adapter hardware and its device driver software implement a Unix line printer device which can be controlled by the standard Unix line printer software (lpr, lpc, lpd, printcap, etc) [1]. Figure 1 shows a typical configuration.

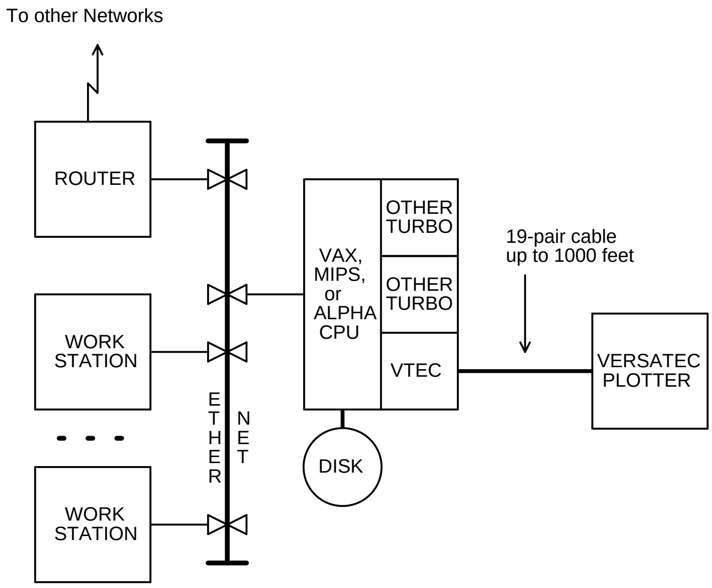

Figure 1: A Typical Configuration

2. Installation Information

Figure 2 is a full-scale drawing of the adapter, a single-slot TurboChannel option module. It uses programmed I/O, not DMA, and occupies 4 MBytes of address space. Power consumption is less than 1 Amp at 5 Volts = 5 watts; no 12 Volt power is used. There are no jumpers or switches to configure; just plug it in.

The maximum data rate at the VPI interface is typically 1 MByte/sec. The maximum data rate of the VTEC adapter is about 1.5 MBytes/sec. The maximum data rate of an Ethernet is 1.25 MBytes/sec. So a Versatec plotter controlled by a VTEC adapter installed in a computer connected to an Ethernet is a balanced system: data can arrive at the server via the Ether at about the same rate that it can depart for the plotter via the VPI.

One data byte is transferred to the adapter on each I/O write. 2K bytes are transferred to the adapter on each interrupt. So at the peak data rate of 1 MByte/sec, the adapter will generate 500 interrupts per second. If the CPU can't keep up (that is, if it can't take an interrupt, transfer 2K bytes, and dismiss the interrupt in less than 2 milliseconds), then the plotter will simply slow down, degrading gracefully. 8K FIFO chips are just coming on the market (they cost $75 each) but they permit 4K bytes to be transferred per interrupt, halving the maximum interrupt rate from 500/sec to 250/sec.

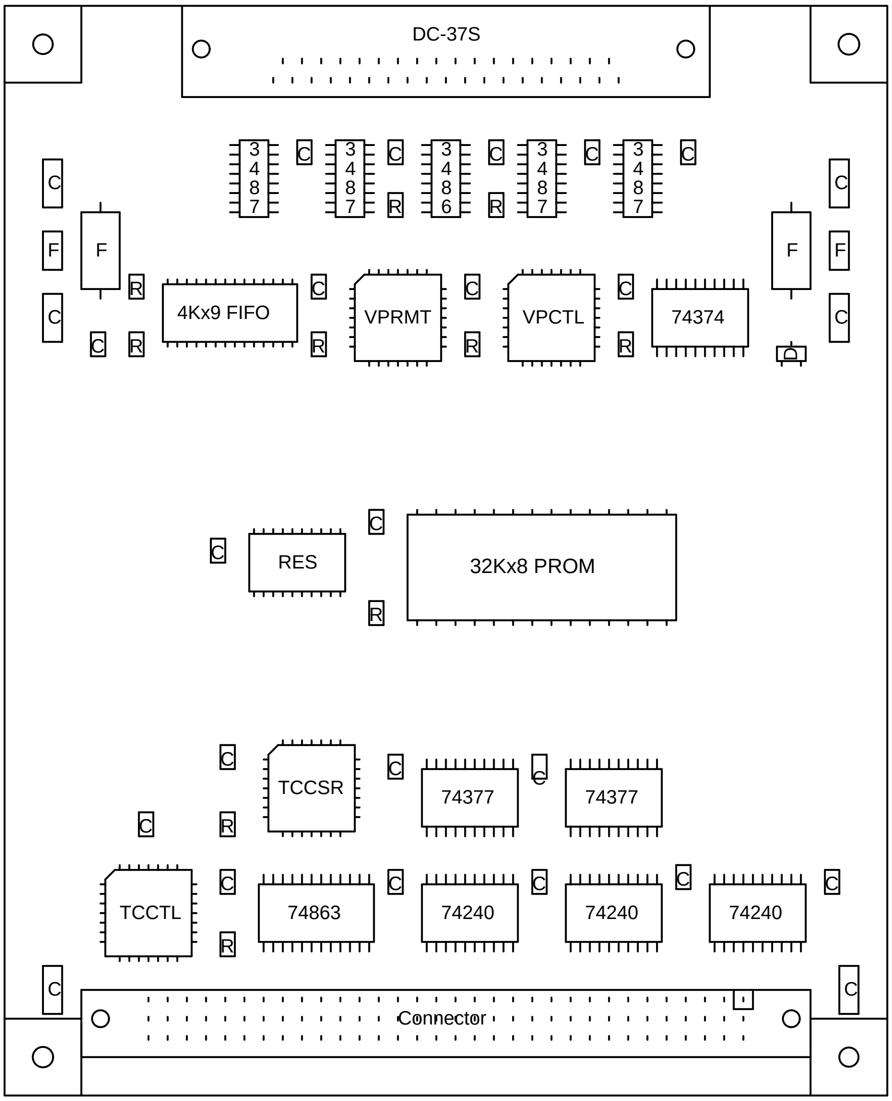

Figure 2: Option Module Layout

A plotter server system should spool arriving plots on a *local* disk, not a disk accessed over the Ethernet. If the spool disk is out on the Ethernet, then plot data will move through the Ether *three* times before being plotted. The spool disk doesn't have to be very big: just big enough to hold the plot in progress, plus any other plot jobs that arrive while the plotter is busy. One or two hundred megabytes of local disk should suffice for most configurations.

3. What is a Versatec Parallel Interface?

The Versatec Parallel Interface (VPI) [5] is an 8-bit parallel data transport method that moves graphic, control, and character data from an adapter to a plotter or printer. The signalling is EIA-422 [4] (differential TTL levels) and handshaking between adapter and plotter is done on a byte-by-byte basis.

The cable between adapter and plotter is 19 twisted pairs with a characteristic impedance of about 120 ohms; maximum length is about 1000 feet. 37-pin D-subminiature cable connectors are used (DC-37), the adapter has a female connector with screw-locks and plotters have male connectors with bail-mount locks.

Figure 3 shows the byte-by-byte handshaking protocol at the VPI interface. If the plotter is `READY` then the adapter can set up a data byte and assert `PICLK` (*parallel interface clock*). `PICLK` must be asserted for a minimum of 300 ns, and must not be deasserted until `READY` drops.

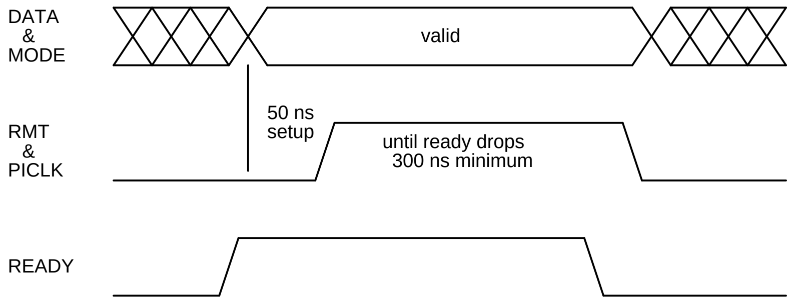

Figure 3: VPI timing diagram

Four signals in the VPI are used by an adapter to remotely control a plotter. These signals must only be asserted singly, and they use a protocol similar to the data handshake, but instead of `PICLK` the remote control signal itself is asserted until `READY` drops and then it is dropped. The exact behavior of a remote command depends on the model; consult the plotter manual.

- The ***FF*** remote control signal moves paper a predetermined distance after a plot is complete (*formfeed*).
- The ***CLR*** remote control signal *clears* the scan line buffer in the plotter.
- The ***EOL*** remote control signal *ends* a scan *line*. Subsequent data starts filling the next scan line.
- The ***EOP*** remote control signal *ends* a *plot* job. It moves the paper so the plot is outside of the imaging mechanism, and then shuts down the plotter.

Many plotter models have a built-in character generator and can print character data as well as plot raster data. The `MODE` signal in the VPI modifies the interpretation of the eight data signals. If `MODE` is true, then the data signals are an ASCII character code, otherwise they are raster data bits. The first byte of a scan line is on the left and the most significant bit of a data byte is the leftmost bit when plotted. Newer model plotters use ASCII escape sequences to control special functions (e.g. color selection) that were not anticipated when the original four remote control signals were defined.

`INOP` is the only other signal from the plotter to the adapter besides `READY`. It is true if the plotter power is off, or the cable from adapter to plotter is unplugged.

4. What is a TurboChannel?

A TurboChannel is a synchronous asymmetric input/output bus designed by Digital Equipment Corporation for use in workstations [3]. It is *synchronous* because there is a global clock whose rising edge is the reference for all bus signals. It is *asymmetric* because there is one *system module* and some number of *option modules* connected to it. The system module can read and write the option modules, and option modules can read and write the system module, but option modules can't talk to each other.

When a program running in the system module executes a load or store instruction whose effective address falls within the address space of an option module, an *I/O read* or *I/O write* bus transaction is performed. The system module asserts a signal selecting an option module, and places an address on the 32-bit address/data bus, which the option module saves in a register. In the next cycle the system module reads or writes a word; if the option module is not ready, it can assert a signal to stall the transaction. A DS5000/200 can do 5 I/O writes or 2 I/O reads per microsecond. The VTEC adapter requires one I/O write per microsecond when going full speed.

An option module can read and write main memory via the system module by performing a *Direct Memory Access* or DMA bus transaction. An option module asserts a read or write signal to start a DMA operation, and then waits for an acknowledgement from the system module. Upon receipt of an ack, the option module drives an address onto the bus and then writes, or after a pause, reads 32-bit words on each succeeding cycle, for a peak bandwidth of 100 MBytes/sec (one 32-bit word every 40 ns = 800 Mbits/sec = 100 MBytes/sec). At 200 KBytes/sec typical, 1 MByte/sec max, this adapter does not use DMA.

There is nothing more to the TurboChannel; it is refreshingly simple. Many of DEC's I/O buses are asynchronous and symmetric, making them slow and complex, and raising the cost to implement simple devices.

5. Technical Description

Figure 4 is a block diagram of the adapter. Data flows from the TurboChannel, through bus transceivers into a 4 KB FIFO, and out through the VPI line drivers.

- `IOADDR` is a 16-bit register that holds the address during a TurboChannel transaction.
- `PROM` is a 32KB EPROM containing TurboChannel configuration information and diagnostic programs.
- `TCCTL` is a PAL containing a finite state machine which handles TurboChannel bus transactions.
- `TCCSR` is a PAL containing the TurboChannel control and status register.
- `FIFO` is a 4KB FIFO buffer between the TurboChannel interface and the Versatec Parallel Interface.
- `VPCTL` is a PAL containing a finite state machine which handles Versatec Parallel Interface transactions.

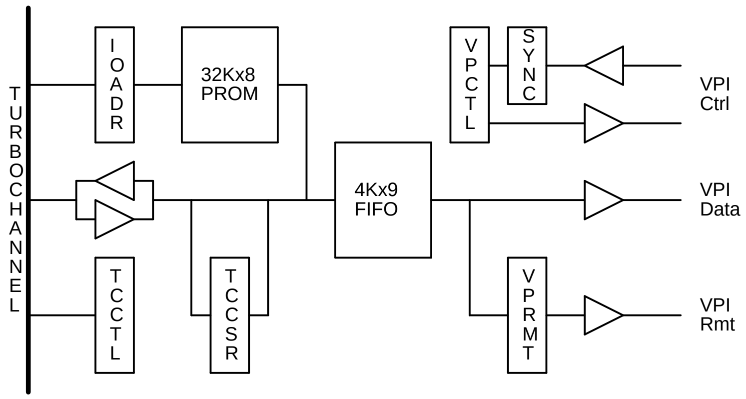

Figure 4: Block Diagram

- `VPRMT` is a PAL containing the VPI remote control signals.
- `READY` and `INOP` from the plotter are synchronized to the adapter clock before going to the VPCTL FSM.

Remote control commands are issued, and the print/plot mode bit is manipulated by writing a byte into the FIFO with the ninth bit set. When such a byte emerges from the FIFO on the Versatec side, it is acted upon by the Versatec FSM. By passing remote commands and mode changes through the FIFO, they get executed in the proper order relative to surrounding data bytes. The mode bit appears as a read-only bit in the CSR.

6. Programming Interface

The VTEC adapter is a programed I/O device; it does not use DMA. A device driver communicates with the adapter by executing load and store instructions whose effective addresses are within the adapter's 4 MByte address space.

There are three regions in the adapter address space: control/status, data, and option rom.

- The ***Control/Status Register*** (CSR) is a single address. Some of the bits in this register are read-only, such as FIFO status, and some are read/write, such as interrupt enable.
- The ***Data Register*** is a single address to which bytes are written. Bytes are either data to be plotted or remote control functions to be performed.
- The ***Option ROM*** occupies 128K addresses in the adapter's space. It contains information about the adapter (type, manufacturer, revision level, etc) and diagnostic programs that can be loaded into memory and executed under the ROM Executive.

6.1. Control/Status Register

The control/status register, figure 5, is addressed when the adapter is selected and address bits 17 and 2 are one. Other address bits are ignored, so many other I/O addresses also reference the CSR.

```
 31              9    8    7    6    5    4    3    2    1    0
+------------------+----+----+----+----+----+----+----+----+----+
|                  |    |    |    |    |    |    |    |    |    |
|     Ignored      |  1 |  1 |Mode| In |Fifo|Fifo|Fifo|Int | Re |
|                  |    |    |    | Op |Empt|Half|Full|Enbl| Set|
+------------------+----+----+----+----+----+----+----+----+----+

Bits 31..9: no connection.
Bit 8 - always reads one.
Bit 7 - always reads one.
Bit 6 - Mode; read-only; cleared by hardware reset.
  One if in PRINT mode, zero if in PLOT mode.
  Changed by software writing a control byte to the FIFO.
Bit 5 - Inoperative; read-only; unaffected by hardware reset.
  One if device power is off or cable is unplugged.
Bit 4 - Fifo Empty; read-only; set by hardware reset.
  One if the FIFO is empty.
Bit 3 - Fifo Half; read-only; cleared by hardware reset.
  One if the FIFO is more than half full (2049-4096 bytes).
Bit 2 - Fifo Full; read-only; cleared by hardware reset.
  One if the FIFO is full.
Bit 1 - Interrupt Enable; read-write; cleared by hardware reset.
  Set by software to permit the adapter to cause interrupts.
Bit 0 - Reset; read-write; cleared by hardware reset.
  Set and then cleared by software to reset adapter and device.
```

Figure 5: Control/Status Register (CSR)

`csr.mode` is the Versatec print/plot mode bit. It is set by writing the remote control register (see below).

`csr.inop` is true if the plotter is turned off or the cable between adapter and plotter is unplugged. While `csr.inop` is true, an interrupt is generated if enabled.

The three FIFO bits indicate the status of the FIFO. After a reset, `csr.fifoEmpt` will be true and the other two will be false. While `csr.fifoHalf` is false, an interrupt is generated if enabled. Normally, software fills the FIFO and then enables interrupts. When half of the bytes in the FIFO have been sent to the plotter, `csr.fifoHalf` goes false, an interrupt occurs, and software fills up the FIFO again. As long as the FIFO never runs dry, the plotter runs at full speed.

The adapter can be operated without interrupts by clearing `csr.IntEnbl` and polling `csr.fifoHalf` and `csr.inop`. In addition to the Interrupt Enable bit in the adapter CSR, a bit in a CPU register must be set for interrupts to occur. `csr.fifoHalf` must be one and `csr.inop` must be zero before enabling interrupts or else an interrupt will immediately happen.

`csr.reset` empties the FIFO, resets the finite state machines, and asserts the `RESET` signal in the VPI. Software sets and then clears this bit to reset everything; this is a drastic action.

6.2. Data Register

The data register, figure 6, is addressed when address bit 17 is one, bit 2 is zero, and data bit 8 is zero. Other address bits are ignored, so many other I/O addresses also reference the data register.

```
 31              9    8    7                                  0
+-------------------+----+----------------------------------------+
|                   |    |                                        |
|      Ignored      | 0  |                 DATA                   |
|                   |    |                                        |
+-------------------+----+----------------------------------------+

Bits 31..9: no connection.
Bit 8:      must be zero
Bits 7..0:  Data - byte to be plotted or printed
```

Figure 6: Data Register

A byte written to the data register goes into the FIFO. When it emerges on the Versatec side, because bit 8 is zero, it is driven onto the VPI data lines with a handshake. The plotter will interpret the byte as character data to be printed or raster data to be plotted depending on `MODE`.

6.3. Remote Control Register

The remote control register, figure 7, is addressed when address bit 17 is one, bit 2 is zero, and data bit 8 is one. Other address bits are ignored, so many other I/O addresses also reference the remote control register.

```
 31              9    8    7    6    5    4    3    2    1    0
+-------------------+----+----+----+----+----+----+----+----+----+
|                   |    |         |         |    |    |    |    |
|      Ignored      | 1  | Ignored |   Mode  | CLR| FF | EOL| EOP|
|                   |    |         |         |    |    |    |    |
+-------------------+----+----+----+----+----+----+----+----+----+

Bits 31..9: no connection.
Bit 8:      must be one
Bits 7..6:  no connection.
Bit 5..4:   Mode - 11 = PRINT mode; 10 = PLOT mode; 0X = no change.
Bit 3: CLR  Clear Scan Line Buffer
Bit 2: FF   Form Feed
Bit 1: EOL  End of Scan Line
Bit 0: EOP  End of Plot
```

Figure 7: Remote Control Register

A byte written to the remote control register goes into the FIFO. When it emerges on the Versatec side, because bit 8 is one, `MODE` is changed or a remote command is issued. To issue a remote control command, set the mode field to zero and set the desired remote command bit. *Only one remote command bit* (`CLR`, `FF`, `EOL`, *or* `EOP`) *should be set at a time.* `MODE` *must not be changed when a command bit is set.* `MODE` is only changed if bit 5 is one; this permits changing a remote control bit without having to refresh the mode bit value.

6.4. Option ROM

The 32 KByte Option ROM occupies the lowest 128 KBytes of adapter address space. The Option ROM is addressed when the adapter is selected and address bit 17 is zero. Address bits 21..18 are ignored so there are many other I/O address ranges which also reference the ROM. Address bits 16..2 select a location in the ROM. Address bits 1..0 are ignored because I/O references are to 32-bit words. Consecutive ROM bytes are the least significant bytes of consecutive 32-bit words.

The Option ROM has two parts: fixed-format configuration information and variable-format objects such as scripts, and executable code. System module firmware called the *ROM Executive*, or REX, can interpret the script objects and execute the code objects. Rather than reprogramming a ROM for each new version during development, REX can be directed to read an Option ROM image from an Ethernet boot server rather than from the ROM on the Option module. A Unix program written in Modula-2 generates the PROM programmer file and the net-bootable file.

REX loads a code object at a fixed address in memory and transfers control to it, passing a *callback vector*, an array of utility procedures which the code object can use to print messages on the console, etc. There is only one code object in this adapter's Option ROM, a keyboard-driven diagnostic program written in Modula-2. The program is loaded in the cached kernel memory region because it runs about four times faster than uncached.

To use an Option ROM image booted from the net rather than the ROM on the adapter, register the file with a boot server, and use the undocumented REX command `boot -D n 6/tftp`, where `n` is the slot number of the adapter whose ROM is to be replaced (`6/mop` works too). REX commands invoking code objects will read this image rather than the adapter ROM until REX is reinitialized.

7. Diagnostic Program

The diagnostic program is started by typing `t n` to the REX prompt of `>>`, where `n` is the TurboChannel slot number of the adapter to be tested; i.e. `t 0` starts the diagnostic program of an adapter in slot zero.

The diagnostic is a keyboard-driven program which is used to debug adapters and test Versatec plotters without the complication of going through an operating system. Typing "?" lists all of the commands; typing "help" prints a list of the commands along with one-line descriptions. To invoke a command, type an unambiguous initial substring terminated by a carriage return or space. To stop a running command, type any character.

- The ***Quit*** and ***Exit*** commands leave the adapter diagnostic program and return to the ROM Executive.
- The ***TestPattern*** command prints graph paper. The plot is 4096 by 4096 bits with vertical and horizontal lines every 128 bits. This command should ask the user for the dimensions of the plotter, and it should generate more thorough test patterns.
- The ***Reset*** command clears the adapter and the Versatec plotter. It resets the FIFO, making it empty. It resets the VPI finite state machine if it is hung waiting for

READY. It is passed on to the plotter, where its effects depend on the plotter model; consult the manual.

- The ***Status*** command prints the contents of the Control/Status Register (CSR) as a hex number and as named fields.
- The ***Mode*** command toggles the print/plot mode bit in the Versatec Parallel Interface.
- The ***FormFeed*** command issues a form feed command to the plotter by asserting the form feed signal in the Versatec Parallel Interface.
- The ***Interrupt*** command tests the interrupt generation logic. It resets the adapter, making the FIFO empty, which is a reason to interrupt. It enables interrupts by setting the bit in the CSR, and then it asks REX if it sees an interrupt from this slot.
- The ***ScopeLoop*** command continuously reads the control and status register. ScopeLoop is terminated by typing any character. This command generates steady oscilloscope pictures for debugging the TurboChannel control logic.

8. Unix Installation

The device driver was written and debugged for the MIPS architecture version of Ultrix version 4.2. It is very simple and only uses a few basic Unix kernel routines, so it should be easy to port. This explanation assumes that an Ultrix kernel-building directory tree exists with a top-level directory named `sys`. This example will define three adapters installed in any slots. Volume six of the Ultrix Software Development documentation is a comprehensive manual on how to build device drivers [2].

Install the two source files:

```
File name        Directory
----------------------------
vtec.c           sys/io/tc
vtecioctl.h      sys/h, /usr/include
```

Add the following line to `sys/conf/mips/files.mips`:

```
io/tc/vtec.c       optional vtec device-driver Notbinary
```

Add the following lines to `sys/conf/mips/VERSATEC` (or whatever you call your kernel configuration):

```
device  vtec0   at ibus?        vector vtec_intr
device  vtec1   at ibus?        vector vtec_intr
device  vtec2   at ibus?        vector vtec_intr
```

Add the following line to `sys/data/tc_option_data.c` after the comment "add any new devices or controllers here":

```
{"VTEC-AA ",    "vtec", 0,      1,      'D',    0},
```

Add the following lines to `sys/machine/common/conf.c`:

```
#include "vtec.h"
```

```
#if NVTEC > 0
int vtec_open(), vtec_close(), vtec_write(), vtec_ioctl();
#else
#define vtec_open       nodev
#define vtec_close      nodev
#define vtec_write      nodev
#define vtec_ioctl      nodev
#endif
```

Also in `sys/machine/common/conf.c`, add the following lines to the `cdevsw` table: The index of the table entry (83 in this example) defines the *major device number* of the adapter.

```
        {vtec_open,     vtec_close,     nodev,  vtec_write,
        vtec_ioctl,     nodev,          nodev,  0,
        seltrue,        nodev,          0,      0}  /* 83 */
```

To make the new vmunix, type the following lines:

```
        cd sys/conf/mips
        config VERSATEC
        cd sys/MIPS/VERSATEC
        make depend all
```

As `superuser`, type the following commands: This creates three *character special* inodes with the major device number from `cdevsw` (above), and minor device numbers 0, 1, and 2.

```
        cd /dev
        mknod vtec0 c 83 0
        mknod vtec1 c 83 1
        mknod vtec2 c 83 2
```

After creating the inodes, check them by typing `ls -l /dev/vtec*`:

```
crw-rw-r--  1 root      83,   0 May 10 15:23 /dev/vtec0
crw-rw-r--  1 root      83,   1 May 10 15:23 /dev/vtec1
crw-rw-r--  1 root      83,   2 May 10 15:23 /dev/vtec2
```

Add the following lines to `/etc/printcap`: The *versatec filter*, `/usr/lib/vdmpc` is a local program which reads site-specific Versatec-format files and writes them to the adapter. It is invoked by the line printer daemon if the `-v` switch is passed to `lpr`. It might accept rasterized bitmap files, or it might accept higher-level formatted files and rasterize them itself. `px#8000` declares that there are 8000 bits per scan line; `py#2400` declares that for accounting purposes a "page" is 2400 scan lines. Consult your plotter manual for the correct value of `px`.

```
versatec0:\
        :lp=/dev/vtec0:mx#0:\
        :px#8000:py#2400:\
        :vf=/usr/lib/vdmpc:\
        :of=/usr/lib/vpf:\
        :lf=/usr/adm/versatec0.log:\
        :af=/usr/adm/versatec0.acct:\
        :sd=/usr/spool/printq/versatec0:
versatec1:\
        :lp=/dev/vtec1:mx#0:\
        :px#8000:py#2400:\
```

```
        :vf=/usr/lib/vdmpc:\
        :of=/usr/lib/vpf:\
        :lf=/usr/adm/versatec1.log:\
        :af=/usr/adm/versatec1.acct:\
        :sd=/usr/spool/printq/versatec1:
versatec2:\
        :lp=/dev/vtec2:mx#0:\
        :px#8000:py#2400:\
        :vf=/usr/lib/vdmpc:\
        :of=/usr/lib/vpf:\
        :lf=/usr/adm/versatec2.log:\
        :af=/usr/adm/versatec2.acct:\
        :sd=/usr/spool/printq/versatec2:
```

This is what you would see after typing `ls -l /usr/spool/printq/versatec0` when a plot was waiting for versatec0 to be available:

```
-rw-rw----  1 root          130 May 10 15:22 cfA031scrooge
-rw-rw----  1 root     32001024 May 10 15:22 dfA031scrooge
-rw-rw----  1 root           10 May  9 15:31 init
-rw-r--r--  1 root           30 May 10 15:23 lock
-rw-rw-r--  1 root           46 May 10 18:14 status
```

It is easy to confuse physical TurboChannel slot numbers and logical Unix minor device numbers. The diagnostic program deals in physical TurboChannel slot numbers, which are stenciled on the rear bulkhead of the computer. Unix deals in logical device numbers which it calls *minor device numbers*. When Unix boots, it discovers I/O devices by interrogating the ROM in each TurboChannel slot, starting at physical slot 0. The first Versatec adapter that it encounters becomes minor device 0, the second one becomes minor device 1, etc. If the first Versatec adapter is plugged into TurboChannel slot 1, then Unix will call it `vtec0`, even though the adapter is in slot 1.

9. Acknowledgements

Robert Coulson of the Work Station Engineering group (DECWSE) designed the first version of this adapter in late 1990. He made five printed circuit boards, assembled one, and partially debugged it as an unofficial project. New product deadlines and lack of any volunteers to write the Unix device driver forced him to set it aside.

The Western Research Lab has a very high-mileage, early model Versatec color plotter which has been faithfully plotting our chips for almost 10 years. It has always been controlled by a VAX with a Unibus adapter designed by Versatec; we scrounged the Unix device driver from somewhere and carefully ported it as we moved from 4.1 bsd to the present. Alpha computers are coming, and we are retiring all of our VAX timesharing computers, so we needed a new way to control our plotter.

In mid January 1992 I finished debugging Coulson’s adapter. I used the LP11 device driver as a model to write a device driver. About one month later, I disconnected the old Unibus adapter and cut over to the new TurboChannel adapter. There were the normal number of bugs in any new design, and while revising the printed circuit artwork, I simplified the design somewhat. Then I built 25 copies, which I am giving to friendly users.

Ed Gould answered my questions about Unix device drivers. Henry Petras and Jeff Mogul showed me how to build Unix kernels. Bob Mayo helped with expertise on user-level Unix issues.

10. References

[1] Ralph Campbell.
4.2 BSD Line Printer Spooler Manual.
4.2BSD Release Notes, UC Berkeley CSRG, July, 1983.

[2] Digital Equipment Corporation.
Guide to Writing and Porting VMEbus and TURBOchannel Device Drivers.
Document AA-PGL5A-TE, DEC, May, 1991.

[3] Digital Equipment Corporation.
TurboChannel Specification Package.
Document EK-TCDEV-DK, DEC, 1992.

[4] Electronic Industries Association.
Electrical Characteristics of Balanced Voltage Digital Interface Circuits.
Document EIA-422-A, EIA, December, 1978.

[5] Xerox/Versatec.
Versatec Parallel Interface Specification.
Model 8900 User Guide, Appendix C, Versatec, 1992.

WRL Research Reports

‘‘Titan System Manual.’’
Michael J. K. Nielsen.
WRL Research Report 86/1, September 1986.

‘‘Global Register Allocation at Link Time.’’
David W. Wall.
WRL Research Report 86/3, October 1986.

‘‘Optimal Finned Heat Sinks.’’
William R. Hamburgen.
WRL Research Report 86/4, October 1986.

‘‘The Mahler Experience: Using an Intermediate Language as the Machine Description.’’
David W. Wall and Michael L. Powell.
WRL Research Report 87/1, August 1987.

‘‘The Packet Filter: An Efficient Mechanism for User-level Network Code.’’
Jeffrey C. Mogul, Richard F. Rashid, Michael J. Accetta.
WRL Research Report 87/2, November 1987.

‘‘Fragmentation Considered Harmful.’’
Christopher A. Kent, Jeffrey C. Mogul.
WRL Research Report 87/3, December 1987.

‘‘Cache Coherence in Distributed Systems.’’
Christopher A. Kent.
WRL Research Report 87/4, December 1987.

‘‘Register Windows vs. Register Allocation.’’
David W. Wall.
WRL Research Report 87/5, December 1987.

‘‘Editing Graphical Objects Using Procedural Representations.’’
Paul J. Asente.
WRL Research Report 87/6, November 1987.

‘‘The USENET Cookbook: an Experiment in Electronic Publication.’’
Brian K. Reid.
WRL Research Report 87/7, December 1987.

‘‘MultiTitan: Four Architecture Papers.’’
Norman P. Jouppi, Jeremy Dion, David Boggs, Michael J. K. Nielsen.
WRL Research Report 87/8, April 1988.

‘‘Fast Printed Circuit Board Routing.’’
Jeremy Dion.
WRL Research Report 88/1, March 1988.

‘‘Compacting Garbage Collection with Ambiguous Roots.’’
Joel F. Bartlett.
WRL Research Report 88/2, February 1988.

‘‘The Experimental Literature of The Internet: An Annotated Bibliography.’’
Jeffrey C. Mogul.
WRL Research Report 88/3, August 1988.

‘‘Measured Capacity of an Ethernet: Myths and Reality.’’
David R. Boggs, Jeffrey C. Mogul, Christopher A. Kent.
WRL Research Report 88/4, September 1988.

‘‘Visa Protocols for Controlling Inter-Organizational Datagram Flow: Extended Description.’’
Deborah Estrin, Jeffrey C. Mogul, Gene Tsudik, Kamaljit Anand.
WRL Research Report 88/5, December 1988.

‘‘SCHEME->C A Portable Scheme-to-C Compiler.’’
Joel F. Bartlett.
WRL Research Report 89/1, January 1989.

‘‘Optimal Group Distribution in Carry-Skip Adders.’’
Silvio Turrini.
WRL Research Report 89/2, February 1989.

‘‘Precise Robotic Paste Dot Dispensing.’’
William R. Hamburgen.
WRL Research Report 89/3, February 1989.

‘‘Simple and Flexible Datagram Access Controls for Unix-based Gateways.’’
Jeffrey C. Mogul.
WRL Research Report 89/4, March 1989.

‘‘Spritely NFS: Implementation and Performance of Cache-Consistency Protocols.’’
V. Srinivasan and Jeffrey C. Mogul.
WRL Research Report 89/5, May 1989.

‘‘Available Instruction-Level Parallelism for Superscalar and Superpipelined Machines.’’
Norman P. Jouppi and David W. Wall.
WRL Research Report 89/7, July 1989.

‘‘A Unified Vector/Scalar Floating-Point Architecture.’’
Norman P. Jouppi, Jonathan Bertoni, and David W. Wall.
WRL Research Report 89/8, July 1989.

‘‘Architectural and Organizational Tradeoffs in the Design of the MultiTitan CPU.’’
Norman P. Jouppi.
WRL Research Report 89/9, July 1989.

‘‘Integration and Packaging Plateaus of Processor Performance.’’
Norman P. Jouppi.
WRL Research Report 89/10, July 1989.

‘‘A 20-MIPS Sustained 32-bit CMOS Microprocessor with High Ratio of Sustained to Peak Performance.’’
Norman P. Jouppi and Jeffrey Y. F. Tang.
WRL Research Report 89/11, July 1989.

‘‘The Distribution of Instruction-Level and Machine Parallelism and Its Effect on Performance.’’
Norman P. Jouppi.
WRL Research Report 89/13, July 1989.

‘‘Long Address Traces from RISC Machines: Generation and Analysis.’’
Anita Borg, R.E.Kessler, Georgia Lazana, and David W. Wall.
WRL Research Report 89/14, September 1989.

‘‘Link-Time Code Modification.’’
David W. Wall.
WRL Research Report 89/17, September 1989.

‘‘Noise Issues in the ECL Circuit Family.’’
Jeffrey Y.F. Tang and J. Leon Yang.
WRL Research Report 90/1, January 1990.

‘‘Efficient Generation of Test Patterns Using Boolean Satisfiablilty.’’
Tracy Larrabee.
WRL Research Report 90/2, February 1990.

‘‘Two Papers on Test Pattern Generation.’’
Tracy Larrabee.
WRL Research Report 90/3, March 1990.

‘‘Virtual Memory vs. The File System.’’
Michael N. Nelson.
WRL Research Report 90/4, March 1990.

‘‘Efficient Use of Workstations for Passive Monitoring of Local Area Networks.’’
Jeffrey C. Mogul.
WRL Research Report 90/5, July 1990.

‘‘A One-Dimensional Thermal Model for the VAX 9000 Multi Chip Units.’’
John S. Fitch.
WRL Research Report 90/6, July 1990.

‘‘1990 DECWRL/Livermore Magic Release.’’
Robert N. Mayo, Michael H. Arnold, Walter S. Scott, Don Stark, Gordon T. Hamachi.
WRL Research Report 90/7, September 1990.

‘‘Pool Boiling Enhancement Techniques for Water at Low Pressure.’’
Wade R. McGillis, John S. Fitch, William R. Hamburgen, Van P. Carey.
WRL Research Report 90/9, December 1990.

‘‘Writing Fast X Servers for Dumb Color Frame Buffers.’’
Joel McCormack.
WRL Research Report 91/1, February 1991.

‘‘A Simulation Based Study of TLB Performance.’’
J. Bradley Chen, Anita Borg, Norman P. Jouppi.
WRL Research Report 91/2, November 1991.

‘‘Analysis of Power Supply Networks in VLSI Circuits.’’
Don Stark.
WRL Research Report 91/3, April 1991.

‘‘TurboChannel T1 Adapter.’’
David Boggs.
WRL Research Report 91/4, April 1991.

‘‘Procedure Merging with Instruction Caches.’’
Scott McFarling.
WRL Research Report 91/5, March 1991.

‘‘Don’t Fidget with Widgets, Draw!.’’
Joel Bartlett.
WRL Research Report 91/6, May 1991.

‘‘Pool Boiling on Small Heat Dissipating Elements in Water at Subatmospheric Pressure.’’
Wade R. McGillis, John S. Fitch, William R. Hamburgen, Van P. Carey.
WRL Research Report 91/7, June 1991.

‘‘Incremental, Generational Mostly-Copying Garbage Collection in Uncooperative Environments.’’
G. May Yip.
WRL Research Report 91/8, June 1991.

‘‘Interleaved Fin Thermal Connectors for Multichip Modules.’’
William R. Hamburgen.
WRL Research Report 91/9, August 1991.

‘‘Experience with a Software-defined Machine Architecture.’’
David W. Wall.
WRL Research Report 91/10, August 1991.

‘‘Network Locality at the Scale of Processes.’’
Jeffrey C. Mogul.
WRL Research Report 91/11, November 1991.

‘‘Cache Write Policies and Performance.’’
Norman P. Jouppi.
WRL Research Report 91/12, December 1991.

‘‘Packaging a 150 W Bipolar ECL Microprocessor.’’
William R. Hamburgen, John S. Fitch.
WRL Research Report 92/1, March 1992.

‘‘Observing TCP Dynamics in Real Networks.’’
Jeffrey C. Mogul.
WRL Research Report 92/2, April 1992.

‘‘Systems for Late Code Modification.’’
David W. Wall.
WRL Research Report 92/3, May 1992.

WRL Technical Notes

‘‘TCP/IP PrintServer: Print Server Protocol.’’
Brian K. Reid and Christopher A. Kent.
WRL Technical Note TN-4, September 1988.

‘‘TCP/IP PrintServer: Server Architecture and Implementation.’’
Christopher A. Kent.
WRL Technical Note TN-7, November 1988.

‘‘Smart Code, Stupid Memory: A Fast X Server for a Dumb Color Frame Buffer.’’
Joel McCormack.
WRL Technical Note TN-9, September 1989.

‘‘Why Aren’t Operating Systems Getting Faster As Fast As Hardware?’’
John Ousterhout.
WRL Technical Note TN-11, October 1989.

‘‘Mostly-Copying Garbage Collection Picks Up Generations and C++.’’
Joel F. Bartlett.
WRL Technical Note TN-12, October 1989.

‘‘Limits of Instruction-Level Parallelism.’’
David W. Wall.
WRL Technical Note TN-15, December 1990.

‘‘The Effect of Context Switches on Cache Performance.’’
Jeffrey C. Mogul and Anita Borg.
WRL Technical Note TN-16, December 1990.

‘‘MTOOL: A Method For Detecting Memory Bottlenecks.’’
Aaron Goldberg and John Hennessy.
WRL Technical Note TN-17, December 1990.

‘‘Predicting Program Behavior Using Real or Estimated Profiles.’’
David W. Wall.
WRL Technical Note TN-18, December 1990.

‘‘Systems for Late Code Modification.’’
David W. Wall.
WRL Technical Note TN-19, June 1991.

‘‘Unreachable Procedures in Object-oriented Programming.’’
Amitabh Srivastava.
WRL Technical Note TN-21, November 1991.

‘‘Cache Replacement with Dynamic Exclusion’’
Scott McFarling.
WRL Technical Note TN-22, November 1991.

‘‘Boiling Binary Mixtures at Subatmospheric Pressures’’
Wade R. McGillis, John S. Fitch, William R. Hamburgen, Van P. Carey.
WRL Technical Note TN-23, January 1992.

‘‘A Comparison of Acoustic and Infrared Inspection Techniques for Die Attach’’
John S. Fitch.
WRL Technical Note TN-24, January 1992.

‘‘TurboChannel Versatec Adapter’’
David Boggs.
WRL Technical Note TN-26, January 1992.

‘‘A Recovery Protocol For Spritely NFS’’
Jeffrey C. Mogul.
WRL Technical Note TN-27, April 1992.

‘‘Electrical Evaluation Of The BIPS-0 Package’’
Patrick D. Boyle.
WRL Technical Note TN-29, July 1992.

Table of Contents

List of Figures