Spare Time Gizmos

VT4/VT5/VT6 USER'S MANUAL

```
Self test OK: 128K RAM, 1K EEPROM
VT6 Firmware Apr  5 2009 17:41:13 Version 48 Size 48K Checksum 01FB
Copyright (C) 2004-2009 by Spare Time Gizmos.  All rights reserved.
```

First Edition

CONTENTS

1 OVERVIEW

The Spare Time Gizmos' VT4, VT5 and VT6 are a family of general purpose video display terminals. They communicate with a host computer using standard RS-232 ASCII communications and accept ANSI standard escape sequences. The VT series of terminals use any standard IBM PC compatible VGA display as a monitor and an IBM PC PS/2 style keyboard for input.

The VT family terminals are perfect for use with any computer that requires a serial console, especially those of the classic variety. It works with the Spare Time Gizmos SBC6120 and Elf 2000 kits, as well as any VAX, PDP-11, S-100, or other vintage machine that you might have. Since the VT family has near perfect emulation of the DEC VT220 terminal, it works with any vintage software that you might have, including EDT or emacs.

As of this writing there are three members of the Spare Time Gizmos' VT family.

- The VT4 was the original prototype unit. It is no longer in production, but there are a few out there and they continue to be supported, albeit with reduced functionality, by the firmware.
- The VT5 is an enhanced version with two serial ports and support for multiple host sessions. The VT5 is normally a standalone product and is housed in a small metal enclosure approximately 4” by 5” by 1½”.
- The VT6 is a simplified version with only one serial port. It consists of a 2.5” by 4” PC board and would normally be mounted inside the enclosure for a larger computer. This would allow, for example, an older PDP-8 or S-100 computer to use a VGA monitor and PS/2 keyboard for a console terminal.

A more complete comparison of the three models, including specific differences, can be found in Appendix A.

1.1 TERMINAL FEATURES

Except where noted, all members of the VT family share the same feature set.

1.1.1 Display Features

- Standard VGA video output at 640x400@70Hz
- 80 columns and 25 rows of text using a 8 by 16 character glyph
- Multiple frame buffers with split screen and paged displays (VT5 only)
- Reverse video, underline, dim, bold and blinking character attributes
- Double high and/or double width characters
- A programmable 25th line status display
- Eight programmable foreground and background colors
- Automatic screen saver and power down for VESA DPMI compatible monitors
- VT100 style 4 LED display (VT4/5 only)

1.1.2 Keyboard Features

- Supports standard PS/2 104 key PC keyboards
- Standard mapping of PS/2 keys to DEC standard LKxxx keys
- Setup option to swap CAPS LOCK and CTRL keys on PS/2 keyboards
- Setup option to map BACKSPACE to DELETE on PS/2 keyboards

- ALT behaves as a COMPOSE key for PS/2 keyboards
- Supports LK250, LK411, LK412 and LK450 keyboards in DEC native mode

1.1.3 Communications Features

- Two independent serial ports for two hosts, or host and printer (VT5 only)
- 5/6/7/8 bit characters, Even/Odd/Mark/Space/None parity (VT5 only)
- All baud rates from 110 to 115,200 bps (VT5 only)
- Independent transmit and receive baud rates (VT5 only)
- Software XON/XOFF or hardware RTS/CTS flow control
- Full modem control (VT5 only)
- Software selectable DTE or DCE configuration (VT5 only)
- F5 key sends BREAK (long or short)

1.1.4 Font Features

- DEC Multinational font
- UK National font
- DEC Special Graphics font
- DEC Display Controls font
- VT52 graphics font
- Downloadable fonts

1.1.5 Cursor and Editing Features

- Cursor up, down, forward, backward, move, index, reverse index
- Insert or delete characters or lines
- Clear or erase to BOL, EOL, BOS, EOS, or entire line or screen
- Set scrolling region
- Save or restore cursor
- Programmable tabs

1.1.6 Supported Modes

- Auto wrap mode
- Line-feed or newline mode
- Reverse video mode
- Insert or replace mode
- Enable/disable cursor
- Line or local echo mode
- Keyboard lock mode
- Cursor key mode
- Application keypad mode
- Send-receive mode
- Origin mode
- Smooth scrolling
- VT100 or VT52 compatibility mode
- 7-bit or 8-bit controls modes

- ❖ Display controls mode
- ❖ Compose mode
- ❖ Hold screen mode
- ❖ Setup mode

1.1.7 Printer Features (VT5 only)

- ❖ Serial printer port, 300-9600 bps, 8N1, XON/XOF or CTS/RTS flow control
- ❖ Print form feed mode
- ❖ Print extent mode
- ❖ Auto print mode
- ❖ Print cursor line
- ❖ Print screen
- ❖ VT52 printer codes
- ❖ Print screen key

1.1.8 Other Features

- ❖ Bell or margin bell
- ❖ Setup selectable foreground and background colors
- ❖ Supports VT525 “assign color” escape sequence
- ❖ Save and load all settings from EEPROM
- ❖ Terminal hard and soft reset

1.2 Regulatory Warning

In the United States, the Federal Communications Commission requires that devices that use and radiate radio frequency energy be certified in accordance with CFR Title 47, Parts 2 and 15. Other countries will have different requirements.

The VT family terminals are not in finished product form and have NOT been approved by the FCC or any other regulatory agency worldwide. The user understands that approvals may be required prior to the operation of a VT terminal, and agrees to utilize them in keeping with all laws governing its operation in the country of use.

1.3 Warranty

Spare Time Gizmos offers no warranty, express or implied, as to the reliability or accuracy of the VT4, VT5 or VT6 (“VT”) design. Spare Time Gizmos offers no warranty, express or implied, as to the accuracy of the information presented in this document. Spare Time Gizmos offers no warranty, express or implied, as to the suitability or correctness of any software or firmware supplied in conjunction with the VT.

Spare Time Gizmos makes no representations as to the suitability of the VT for any application. It is solely and exclusively your responsibility to evaluate the accuracy, completeness, and usefulness of the VT and all related designs, software, and other information provided by Spare Time Gizmos. The entire risk as to the use and performance of the VT is assumed solely by you.

No representation or other affirmation of fact, including, but not limited to, statements regarding capacity, performance of products, or suitability for use, whether made by Spare Time Gizmos employees or otherwise, will be deemed to be a warranty for any purpose, or give rise to any liability on the part of Spare Time Gizmos.

THE WARRANTIES AND CORRESPONDING REMEDIES AS STATED IN THIS SECTION ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, WRITTEN OR ORAL. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THE LIMITED WARRANTIES AND CONDITION REFERENCED ABOVE GIVE YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE OTHERS, WHICH VARY FROM JURISDICTION TO JURISDICTION.

IN NO EVENT SHALL SPARE TIME GIZMOS OR ITS EMPLOYEES BE LIABLE FOR ANY COSTS OR DIRECT, INDIRECT, PUNITIVE, INCIDENTAL, SPECIAL, CONSEQUENTIAL DAMAGES OR ANY OTHER DAMAGES WHATSOEVER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF DATA, INTERRUPTION OF BUSINESS, OR LOSS OF USE, ARISING OUT OF OR IN ANY WAY CONNECTED WITH THE USE OR PERFORMANCE OF THE VT OR YOUR RELIANCE ON THE VT OR RESULTS FROM MISTAKES, OMISSIONS, INTERRUPTIONS, DELETION OF FILES, ERRORS, DEFECTS, DELAYS IN OPERATION OR TRANSMISSION, OR ANY FAILURE OF PERFORMANCE WHETHER BASED ON CONTRACT, TORT, STRICT LIABILITY OR OTHERWISE, EVEN IF SPARE TIME GIZMOS HAS BEEN ADVISED OF THE POSSIBILITY OF DAMAGES. BECAUSE SOME STATES/JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

IN NO EVENT SHALL SPARE TIME GIZMOS' LIABILITY, IN THE AGGREGATE, EXCEED THE SUMS ACTUALLY PAID BY YOU TO SPARE TIME GIZMOS AND ACCEPTED BY SPARE TIME GIZMOS FOR THE USE OF THE VT.

2 ASSEMBLY

This chapter gives assembly hints, tips and techniques for the VT5 and VT6 models. The VT4 is no longer sold and is not discussed here. A special thanks to the builders who have contributed their experiences, suggestions and frustrations to this chapter.

2.1 ERRATA

The following sections list the known errors in various versions of the VT boards. To determine the revision of your PC board, look for the text “VTn-x” printed in copper along the edge of the PC board. In this text, the “x” is the revision of your PC board – for example, “VT6-1B” for revision1B.

2.1.1 VT5

TBA

2.1.2 VT6

The earliest revision of the VT6 board that was shipped to customers is 1B, and there are no known errors in this version.

2.2 PART SELECTION

2.2.1 VT5

- ✓ The 16C550 ACE chip as a long history and there are many similar but not quite identical parts such as the 16C450, 16450, 8250, 16552, etc. The VT5 is designed to work *only* with the 16C550.
- ✓ You must use a 74AC part for the 74AC373 address latch. Don’t substitute a part from the 74HC or other family.
- ✓ More stuff to be added about the VT6 beeper, et al.

2.2.2 VT6

- ✓ The buzzer specified for BZ1 is self oscillating. If you substitute another device, be sure that it’s not just a speaker and can in fact generate a tone on its own.
- ✓ Be careful of the difference between RP1 and RP2. Not only are they different values, but RP1 has five resistors in a six pin package with one pin common to all. RP2 has three resistors in a six pin package with no common pin.

2.2.3 All Models

- ✓ The firmware and hardware can accommodate a wide range of serial EEPROM sizes, anywhere from the 24C01 (128 bytes) up to the 24C512 (64K bytes).
- ✓ Notice that the bypass capacitors used are 0.01μF, not 0.1μF. A 0.1μF ceramic capacitor has a self resonant frequency of something like 5 to 8MHz, and would not be very effective in this circuit.

2.3 Optional Subsystems

2.3.1 VT5

- ✓ The 16C550 ACE chip is optional in the VT5, and without it the firmware will operate as a single session terminal with only one port, the printer port. If you omit the ACE, you can also omit the MAX214 as well as the associated capacitors and the DB25 connector.
- ✓ The VT5 will operate with either a 128K byte (628128 type) or a 512K byte (628512 type) SRAM.

2.3.2 VT6

- ✓ The ten pin serial port header, `J6`, is optional and can be omitted. Conversely, if you intend to use J6 then you can omit the DB9 connector, `J3`.
- ✓ The VT6 has provisions for a direct +5VDC input which you can use if a regulated power supply is available. In this case you would install `J5` and omit `VR1`, `C12`, `C13`, `D1` and `J2`.

2.4 Assembly Hints

- ✓ The VT6 uses six 0.01µF 50VDC monolithic bypass capacitors, and the VT5 uses ??. On the PC board these are identified by a small oval shaped outline on the silk screen. The reference designator (e.g. `C5`, `C6`, etc) is not shown on the PC board for these parts.
- ✓ The tantalum and aluminum electrolytic capacitors used are *polarized* devices and must be installed correctly. The polarization is shown on the silk screen of all PC boards.
- ✓ The speaker used in the VT6 is a *polarized* part and must be installed in the correct orientation.
- ✓ Space is a little tight around the PLCC sockets and you may find it easier to install the nearby capacitors and resistors *before* installing these sockets.

2.5 Initial Checkout

After you finish assembly, apply power before installing any ICs. Place a DC milliammeter inline with the power supply; with no ICs installed the current consumption should be essentially zero. Use a DC voltmeter to verify that the voltage between pins 10 (negative) and 20 (positive) on the APU (AT89LP4052) socket; you should read 4.9 to 5.1V.

Next remove power and install all ICs including the 25.175MHz oscillator. With the milliammeter still inline, turn on the power and check the power consumption – it should be between 200 and 300mA for all versions. *If you don't get these results, and especially if the current drain is significantly more than predicted, then stop and figure out what's wrong before proceeding.*

IMPORTANT

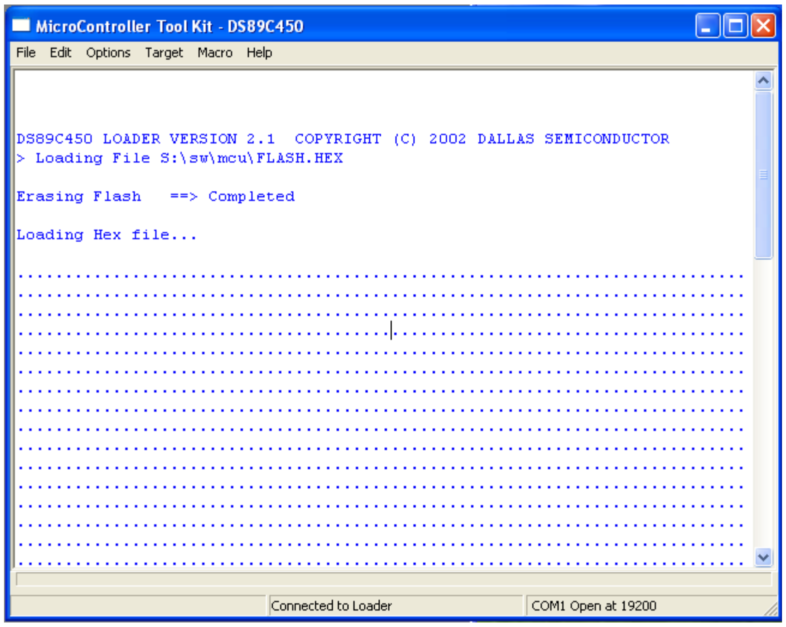
At this point you may be tempted to hook up a monitor and keyboard to try it out. Don't bother – the MCU does not contain any firmware and won't work until you download it, so you'll only be disappointed.

2.6 Downloading Firmware

Now you're ready to program the DS89C450 MCU with the latest VT firmware. This is done "in system," with the MCU installed in the VT board, using the VTs serial port, a PC and the MTK programming software supplied by Dallas Semiconductor. First you'll need to visit the VT project firmware page at

http://vt4.sourceforge.net

and download the latest released firmware for your version of the VT board. Next download the latest MTK release from the Dallas web site and follow the instructions supplied to install it on your PC.


MicroController Tool Kit - DS89C450
File Edit Options Target Macro Help
Connected to Loader
COM1 Open at 19200

```
DS89C450 LOADER VERSION 2.1  COPYRIGHT (C) 2002 DALLAS SEMICONDUCTOR
> Loading File S:\sw\mcu\FLASH.HEX

Erasing Flash   ==> Completed

Loading Hex file...

...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
```

Photo 1 - The MTK Window

http://files.dalsemi.com/microcontroller/dev_tool_software/mtk/

Connect the VT's serial port (*use the Printer port on the VT5*) to a serial port on your PC using a nine pin *null modem* serial cable. *Be sure that the VT board programming jumper is installed as described in section 3.2.1. If you have a VT5, be sure that the serial port jumpers are set for DSR/DTR handshaking as described in section 3.2.2.* Apply power to the VT board, start the MTK software on your PC, and follow these steps –

1) MTK will ask you to select a device. Pick "***DS89C450***" from the list.

2) From the MTK menu bar, select ***Options*** >> ***Configure Serial Port***. Select your serial port and set the speed to 19,200bps.

3) Now select ***Target*** >> ***Open COM … at …*** from the MTK menu bar.

4) You should see a line something like this appear in the MTK window

   `DS89C450 LOADER VERSION 2.1 COPYRIGHT (C) 2002 DALLAS SEMICONDUCTOR`

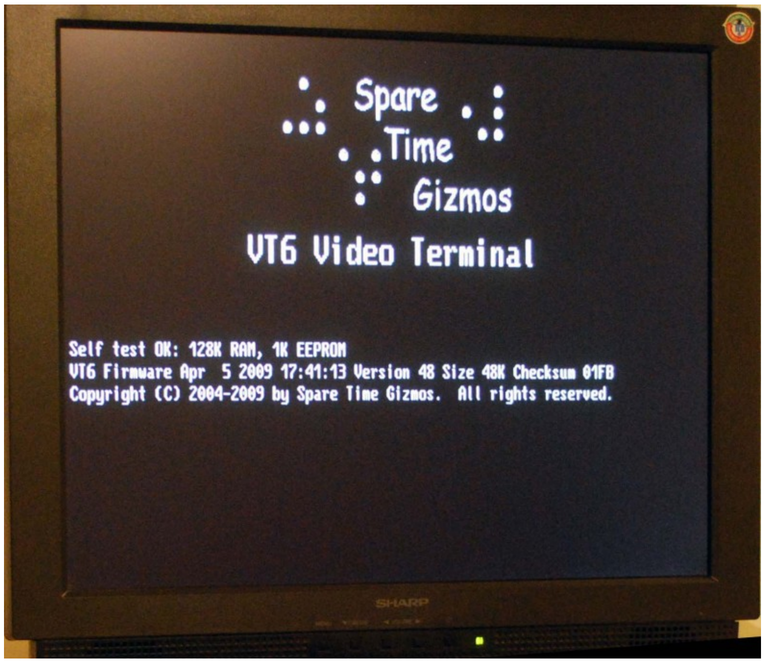
   If you don't get this result, then double check your jumpers, serial connections and that the VT board has power. If that still doesn't work, then you have a problem with the CPLD, MCU, oscillator or MAX232 chips and/or connections.

5) Select ***File*** >> ***Load Flash*** from the MTK menu and then browse for the firmware .HEX file that you downloaded from SourceForge. Click ***Open*** and the download should start.

6) If all went well, you will see

   `Load complete.`

   at the end of the process. Congratulations! Your firmware is now loaded.



```
Spare
Time
Gizmos
VT6 Video Terminal

Self test OK: 128K RAM, 1K EEPROM
VT6 Firmware Apr  5 2009 17:41:13 Version 48 Size 48K Checksum 01FB
Copyright (C) 2004-2009 by Spare Time Gizmos.  All rights reserved.
```

Photo 2 - VT Startup Screen

2.7 Final Checkout

After you've successfully programmed the MCU, remove power, disconnect the cables and *remove the programming jumper*. Attach a VGA monitor and a PS/2 keyboard to their respective connectors. Plug in the power again, and you should see the VT startup screen something like the one shown in Photo 2.

3 INSTALLATION AND INTERFACES

3.1 CONNECTORS

3.1.1 Power

`J2` is the main power connector in all versions of the terminal. On the VT4 and VT6 it is a standard 2.1mm ID, 5.5mm OD, coaxial power connector and the center conductor is positive. The VT5 is similar, however it uses a miniature 1.35mm ID, 3.5mm OD coaxial connector instead, still center positive. The miniature connector is used on the VT5 because of space limitations on the rear panel; in all other respects the VT5 power supply is identical to the other models.

In all models diode D1 protects against accidental reverse polarity. Using a standard 7805 regulator for VR1, the applied power may be anywhere from 9 to 12VDC, *with 9V being recommended to minimize the power dissipation of the regulator*. Use care when applying voltages higher than 12V not to exceed the power dissipation limits of VR1.

3.1.2 Video

`J1` is the VGA video output in all models of the VT terminal and should be a standard HD15 *female* connector. Fortunately, and unlike RS-232, the VGA connector wiring and gender is very standardized and you should be able to directly connect a VGA monitor without any adapters.

Pin	Signal	Description
1	**Red**	Analog video, approximately 1V P-P
2	**Green**	
3	**Blue**	
13	**HSync**	Negative polarity TTL sync
14	**VSync**	
5, 6, 7, 8, 10	**Ground**	

Table 1 – VGA Connector

3.1.3 Serial Port (9 pin)

All models of the video terminal use a *male* DB9 connector for the serial port and is designated `J3` on all versions of the PC board. In the VT4 and VT6 this is the primary and only serial port, and on the VT5 this is the secondary or printer port. The pin out used is the standard for an IBM PC compatible serial port.

The VT4 and VT6 versions support only the DSR/DTR signals and on these PC boards the RTS and CTS signals (pins 7 and 8) are permanently wired together. The VT5 supports *either* RTS/CTS *or* DSR/DTR handshaking, but not both. On the VT5 this selection, and what happens with the unused signal pair, is determined by the way you install the jumpers. See section 3.2.2 for more information.

This serial connector is also used to download firmware to the DS89C450 MCU in all versions of the PC board. To do this, the appropriate programming jumper (see section 3.2.1) must be installed and the port connected to a PC via a *null modem* cable.

WARNING

The Dallas Semiconductor Microprocessor Toolkit software, which is used to download, requires DSR/DTR handshaking and, on the VT5 board, the jumpers *must* be set to this mode.

Finally, pin 1, carrier detect, is hardwired to pin 6, data set ready, on all versions of the terminal.

Pin	Signal	Description
1	**CD**	Carrier Detect *Connected internally to pin 6*
2	**RXD**	Receive Data
3	**TXD**	Transmit Data
4	**DTR**	Data Terminal Ready
5	**GND**	Signal Ground
6	**DSR**	Data Set Ready
7	**RTS**	Ready to Send
8	**CTS**	Clear to Send
9	**RI**	Ring Indicator *Unused on all VT models*

Table 2 – DB9 Connector

3.1.4 Serial Port (25 pin)

The VT5 has two independent serial ports – the secondary, or printer, port uses a DB9 connector as described above, and the primary communications port uses a *male* DB25. This port is capable of being switched, from the setup menu, between DTE (*Data Terminal Equipment*, the wiring normally used by a terminal) and DCE (*Data Communications Equipment*, the wiring normally used by a modem) modes. The ability to switch modes eliminates the need for a null modem cable, although a gender changer may still be required.

The secondary channel, pins 13, 14, 16 and 19, are wired in parallel with the corresponding pins in the DB9 printer port connector. This allows you to access both serial ports of the VT5 using only the on DB25 connector on the rear panel, albeit at the price of making up a special “Y” cable[1]. These pins are unaffected by the primary port DTE/DCE selection. Also, since only the secondary RTS/CTS signals are available in the DB25, if the printer port jumpers (see section 3.2.2) are set for DTR/DSR handshaking then the secondary RTS/CTS pins will be disconnected.

Pin	Signal (DTE)	Signal (DCE)
1	**Ground**	
2	**TXD**	**RXD**
3	**RXD**	**TXD**
4	**RTS**	**CTS**
5	**CTS**	**RTS**
6	**DSR**	**DTR**
7	**Ground**	
8	**DCD**	
20	**DTR**	**DSR**
13	**Secondary CTS**	

[1] It turns out that such “Y” cables are not unheard of. The DEC Multia UDB uses a single DB25 connector, wired identically to the VT5, for both its primary console and secondary serial ports. Some models of Sun workstations also used a similar arrangement.

Pin	Signal (DTE)	Signal (DCE)
14	Secondary TXD	
16	Secondary RXD	
19	Secondary RTS	
22	RI	

Table 3 – DB25 Connector

3.1.5 Keyboard

The PS/2 keyboard connector is `J4` on all models, and is the familiar 6 pin female mini-DIN. The pin out for this connector is summarized in the table below. This connector supplies +5V directly to the keyboard, and there is no fuse or other over current protection (although the on board 7805 regulator, VR1, will shut down in the event of a short circuit). All modern PC keyboards use a CMOS microprocessor and require something like 10mA, so this normally shouldn't be an issue.

Pin	Signal (DTE)
1	Keyboard Data
3	Ground
4	+5V
5	Keyboard Clock

Table 4 – Keyboard Connector

3.1.6 Auxiliary Serial Connector (VT6 only)

The VT6 board contains a ten pin header, designated `J6`, which is a secondary serial connector and is wired in parallel with the corresponding pins on the DB9 connector. When the VT6 is used in conjunction with the SBC6120, this connector can be connected directly to the SBC6120's serial port using a 10 pin IDC header cable. Other than that, it offers no advantage over the DB9.

3.1.7 Auxiliary Power Connector (VT6 only)

The VT6 board contains a secondary power connector, `J5`. This connector is used to supply +5 VDC, regulated, directly to the PC board and if you're using it you should omit parts `VR1`, `C12`, `C13`, `D1` and `J2`.

3.2 CONFIGURATION

3.2.1 Programming Jumper

All versions contain a jumper, `JP1` on the schematic and marked `PROGRAM` on the silkscreen, that enables the DSR signal in the DB9 connector to place the DS89C450 in program mode. This jumper must be installed in order to use the Dallas Semiconductor MTK software to download firmware to the MCU. In normal use this jumper should be removed.

NOTE

In the VT5, programming is always done thru the DB9 printer port connector. Additionally, the jumpers for this port (see 3.2.2) must be set to enable DSR/DTR for compatibility with MTK.

3.2.2 Serial Port Configuration Jumpers

The printer port on the VT5 version can support either DSR/DTR or CTS/RTS hand shaking, but not both. Jumpers JP2 and JP3 select between these two options and the selections are clearly marked in the PC board silk screen. Notice that it is also possible to install a jumper “sideways” over the two unused pins to connect RTS and CTS together, or DSR and DTR together.

3.3 POWER ON SELF TEST

The VT firmware contains power on self test code which attempts to verify the hardware operation. This code is executed only once, after a power up, and if it detects a failure it will signal the error status and halt. This prevents the terminal from ever starting up.

3.3.1 VT4 and VT5 POST Codes

The VT4 and VT5 contain five LEDs, four of which are used to display the code for any POST errors, and the fifth LED is simply a power on indicator which is always illuminated. In the event of a POST failure the firmware will blink a four bit POST error code on the LEDs according to Table 5.

POST	Description
1	Basic CPU checks (i.e. the CPU is not a DS89C450)
2	Program flash checksum wrong
3	External RAM failure
4	Unsupported CPLD revision
5	ACE chip failure
6 .. E	unused
F	APU failure

Table 5 – VT4/5 POST Codes

In the VT4 and 5, the four LEDs can also be turned on and off under program control, just as the keyboard LEDs on a real VT100 can. Don’t confuse this with a POST failure – POST codes are displayed only immediately after startup, and POST errors cause the LEDs to blink. Under normal conditions the LEDs will never blink.

3.3.2 VT6 POST LED

The VT6 contains a single LED which is a combination of a power indicator and POST display.

LED	Description
Off	No power or APU POST failure
Blinking	MCU POST failure
On	Normal Operation

Table 6 – VT6 POST Codes

4 OPERATOR INFORMATION

4.1 SENDING TEXT

4.1.1 IBM PC/104 Keyboard

PC/104 to LK201 key translations

4.1.2 DEC Keyboards

Using the LK250, LK450, LK411 and LK412 keyboards.

4.1.3 Special Keys

Setup, break, compose, etc.

4.1.4 Configurable Keys

Backspace, delete, tilde, dot/comma, escape, etc.

4.2 RECEIVING TEXT

4.2.1 Control Characters

4.2.2 VT100 and VT220 Escape Sequences

4.2.3 VT52 Escape Sequences

4.2.4 VT4/5/6 Extended Escape Sequences

4.3 SETUP SCREEN

5 HARDWARE DESCRIPTION

5.1 OVERVIEW

This chapter will discuss the hardware design of the VT series terminals. It's not intended to be all inclusive, but it will help you understand the design while you're looking at the schematics or studying the firmware.

The VT products contain not just one but two independent microprocessors, a primary "MCU" which does most of the work including generating the video, and a secondary "APU" which interfaces to the PS/2 keyboard and handles the LEDs and beeper. Some people may raise their eyebrows that the idea of a second CPU just for the keyboard, LEDs and a beeper, but the part costs approximately $2 and offloads the main CPU firmware (and therefore the firmware programmer!) of any critical timing issues in bit banging the PS/2 serial keyboard protocol. It's well worth the investment.

The video generation is done by a combination of hardware and firmware. The hardware generates the basic horizontal timing, and for each active scan line it signals the software when it is time to begin generating pixels. The firmware is then responsible for transferring bytes from the frame buffer to the hardware video shift register, which is then shifted out serially to drive the VGA display. The video shift register hardware contains no synchronization logic, and the firmware must output one byte exactly every eight clock cycles to avoid glitching the video. After 80 bytes (640 pixels) have been transferred, the firmware is free to do whatever it wants (i.e. process keyboard and serial port input and generally act like a terminal) until it's time to generate the video for the next scan line.

A 640x400 bitmap requires approximately 30K bytes of RAM, which is far more than the internal memory of the D89C450, so all VT models provide for external SRAM. The VT4 supports either 32 or 64K bytes of SRAM, the VT5 supports either 128K or 512K bytes, and the VT6 always has exactly 128K bytes of SRAM. Since the maximum address space of the MCU is 64K bytes, the VT5 and VT6 models implement a simple bank switching scheme to extend the addressing abilities of the MCU.

5.2 MCU

The primary CPU on the VT boards is the Dallas Semiconductor DS89C450, which is often called the "MCU" (*micro controller unit*) in the firmware and documentation. The DS89C450 is extraordinarily fast by the standards of many embedded controllers – over 30 MIPs – and is generally able to execute one instruction per clock cycle. It's this speed that makes it possible for the video terminals to handle much of the video generation in firmware.

The DS89C450 also features 64K of flash based program memory and has a simple bootstrap loader program in an on chip masked ROM. This bootstrap loader, with a little hardware support from the VT board, allows the MCU firmware to be downloaded from a PC with a standard RS232 serial connection. No special programmer hardware is required to program the MCU, and you don't even have to remove it from the circuit.

The DS89C450 also contains two independent serial ports, three timers, a priority interrupt system, 1K bytes of internal RAM, and a host of other features, not all of which are used by the VT firmware. The other members of the family, the DS89C440, 430, and 420, are identical to the 450 *except* for the amount of program flash memory. As of this writing, though, the VT firmware is too large to fit in any of the others, and the DS89C450 is the only usable option.

5.3 CPLD

The Atmel ATF2500C CPLD used in all VT versions has 14 input pins, 24 output pins, 48 internal flip flops, and a correspondingly large number of product terms. It's also flexible enough to allow the internal flip flops to be disconnected from the associated pins and used independently in many cases. It performs several major functions in the VT design, as well as implementing all the random logic necessary

5.3.1 CPU and APU Clock Generation

The MCU (DS89C450) clock is normally exactly the same as the pixel clock, but that's a problem in programming mode. The DS89C450's built in serial loader isn't very smart about generating baud rates, and it's not possible to download the MCU at any standard baud rate when using a 25.175MHz clock. However, James Markevitch cleverly observed that

25.175MHz * 4/9 = 11.1889MHz

and this value is close enough to the standard 11.0592MHz baud clock (it's about a 1% error) to allow serial downloads. The MCU clock logic in the CPLD generates this 11.1889MHz clock by multiplying the pixel clock by 4/9ths, and then switches the CPU clock output when we are in programming mode.

The APU clock is exactly ½ of the pixel clock or 12.5875MHz. It's derived directly from the LSB of the horizontal timing chain, and doesn't cost us any extra logic in the CPLD to generate. Unlike the MCU programming clock, there's no magic to the APU clock frequency. This value was picked simply because the APU chip isn't fast enough to run directly from the 25.175MHz pixel clock, and by running it at half this frequency we save the expense of a separate crystal or oscillator.

5.3.2 Video Timing

The CPLD generates all of the horizontal timing, including a "character" clock, the horizontal sync and blanking signals, and a "start" signal to tell the software when to begin pseudo-DMA operations. The "character" clock simply counts groups of eight pixels per byte and has no direct connection to the character glyphs on the screen. The horizontal position register counts horizontal character clocks ("HCC") from 0 to 99, corresponding to 800 horizontal pixels per line. The horizontal sync output is active for the first 12 character clocks (that's 96 pixel clocks) of every scan line.

The VSTART output tells the firmware when it should actually start generating pixels. VSTART goes high somewhere in the middle of the horizontal sync and remains high until the end of the back porch (i.e. just before the time the active video should start). "Somewhere in the middle ..." is really a very precise term meaning an empirical value selected so that the firmware has enough time to setup and get ready for the active video. Since the sync width is 12 HCCs and the back porch is 6, the video should start at HCC 18. The current firmware needs about 80 clock cycles or 10 HCCs to set up, so the CPLD will assert VSTART around HCC 8. This is not critical, though, because the firmware can adjust the precise position by inserting extra NOP cycles before starting the DMA.

The CPLD has nothing to do with generating the vertical sync signal; this is done entirely by the firmware and the vertical sync appears on a port pin of the MCU. This is passed thru the CPLD before going to the VGA connector, however this done simply to buffer the signal and ensure that VSYNC has the same drive ability as HSYNC.

5.3.3 Video Generation

To load a byte into the video shift register ("VSR"), the firmware first asserts a particular MCU I/O pin and then reads a byte from external RAM. This MCU I/O pin is simply another general purpose port bit and, since its function is to enable a "pseudo-DMA" operation for loading the

VSR, it's known as the "DMABIT". It's important to realize that any read from external memory when the DMABIT is set, no matter when it happens or what address is read, will load the VSR and that byte will then eventually end up on the screen. It's strictly up to the firmware to ensure that this happens at exactly the right instant with the right data.

The video shift register should be loaded on the seventh pixel clock of every character cell, and the CPLD generates a video blanking signal which will be asserted any time the firmware misses its cue for loading the VSR. The video blanking signal is applied to the VGA output and forces all three colors to black, and this is used to blank the video during the inactive portions of the horizontal and vertical retrace interval. Remember that blanked video is always *black*, not the color map background color. This is critical for many VGA monitors, which will assume that the video is black during the blanking interval and use this to automatically adjust the black level.

5.3.4 Color Map

The CPLD implements an internal color map register which can be written by the firmware. The color map contents are loaded when the firmware writes to *any* location in RAM while the pseudo DMA bit is set. Since we normally only read from RAM with the DMA bit set, this is an unambiguous condition, if a bit crude. The color map register has six bits as shown in Table 7.

Color Map Register							
Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*
Blue	**Green**	**Red**	**0**	**0**	**Blue**	**Green**	**Red**
Background Color					**Foreground Color**		

Table 7 – Color Map Register

Notice that bits 3 and 4 of any byte written to the color map register must be zero – this is to distinguish color map updates from the bank select registers (see section ???).

The foreground color determines the RGB value output when there is a one bit in the corresponding pixel of the video shift register. Likewise, the background color is the RGB value used when there is a zero bit in the VSR. Blanked video, that is all video outside the active scan area of the screen, is always black regardless of the color map value (see section 5.3.3).

5.3.5 Bank Switching

Since the DS89C4x\50 can only address 64K of external data memory, the VT5 and VT6 need to implement some simple bank switching logic to extend that range to either 128 or 512K. The lower 15 address bits for the RAM chip always come directly from the microprocessor, and the bank switching logic in the CPLD supplies either two (for 128K) or four (for 512K) extra upper address bits to the RAM. This effectively divides the external RAM into 4 32K banks (for a 128K SRAM) or 16 32K banks (a 512K SRAM) .

When the MCU accesses an address in the range $0000..7FFF_{16}$, RAM bank zero is always selected regardless of the contents of any bank switching register. When the MCU accesses an address in the range $8000..FFFF_{16}$, the RAM bank selected is determined by the current contents of the Upper Memory Bank (UMB) register. The idea is that the firmware will put shared data - buffers, global variables, etc - into the range $0000..7FFF_{16}$. This memory is always mapped and ISRs and other code can access it without any need to worry about the memory mapping in use. The video frame buffer and other special data lives in the range $8000..FFFF_{16}$, giving three or fifteen additional banks for available for storing this information.

The VT5 hardware implements a separate four bit register, DMAUMB, which supplies the memory bank while video DMA occurs (i.e. when the DMABIT is set). This allows the frame buffer to be stored in any bank of memory. In the VT6 this logic is simplified to force bank 1 to always be selected during DMA, and there is no explicit DMAUMB register.

The VT5 also has an external ACE chip with a memory mapped register file. This chip is selected when the UMB register is set to zero. Remember that bank 0 is always accessible as addresses 0000..$7FFF_{16}$, and rather than allowing it to be mapped twice we can use this case to access the ACE chip instead. The VT6 has no ACE, but for compatibility with the VT5 the SRAM chip is disabled when the UMB register is zero - this tricks the firmware into thinking that the VT6 is a VT5 with no UART chip installed.

The UMB and DMAUMB registers are written in same way as the color map (refer to section 5.3.4), by writing to any address in external RAM while the DMA bit is asserted. The CPLD distinguishes writes to the color map, UMB and DMAUMB registers by the contents of data bits 4 and 5. Table 8 summarizes the format of the UMB register.

Upper Memory Bank Register							
Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*
X	**X**	**0**	**1**	**BS3**	**BS2**	**BS1**	**BS0**

Table 8 – Upper Memory Bank Register

And Table 9 summarizes the format DMAUMB register.

DMA Memory Bank Register							
Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*
X	**X**	**1**	**1**	**BS3**	**BS2**	**BS1**	**BS0**

Table 9 – DMA Memory Bank Register

Remember that the VT6 has no DMAUMB register (any writes to this register will be ignored) and that the VT6 only implements two bits, BS1 and BS0, in the UMB register. Any bits written to BS2 and BS3 on the VT6 are ignored. Finally, the VT4 supports a maximum of 64K of memory and does not implement any form of bank switching logic at all.

5.4 APU

The APU (the *auxiliary processor unit*) is an ATMEL AT89LP4052 microprocessor. Like the DS89C450 this is another 8051 derivative device and that allows us to use the same tool chain for development as we're using for the MCU. The AT89LP4052 is also a flash memory based device, but unfortunately it is not in system programmable and you'll need a programmer that supports this device in order to update its firmware. The AT89LP4052 has 4K bytes of program memory, 256 bytes of internal RAM, two timers, a full duplex hardware UART, and a few other features.

In the VT Gizmos, the APU's primary job is to interface to the PS/2 keyboard. It "bit bangs" the proprietary PS/2 serial protocol, and converts the PS/2 key codes to ASCII which it then sends to the MCU via the internal UART. The APU also manages the self test and status LEDs (there are four on the VT4 and VT5, and one LED on the VT6) and it controls the beeper. The VT4 and VT6 have a simple, self oscillating beeper which can only be switched on and off by the APU to generate either short or long beeps.

The VT5 has more elaborate beeper logic including a programmable volume control. In the VT5, three I/O port bits control the beeper volume in eight levels, and a fourth port bit serves as a "master enable" to turn the beeper on or off. The VT5 also uses a non oscillating beeper – not a beeper at all, really, but just a piezo speaker – and the APU firmware can generate a limited range of different pitches.

5.5 Serial Ports

The MCU contains two serial ports. Port 0 of the MCU internal ports is converted to RS232 signal levels and then brought out to a DB9 connector. In addition, two general purpose I/O bits, one input and one output, are used to drive the DSR/DTR signals in the DB9 for the VT4 and VT6. The VT5 is similar, however in this version jumpers allow the two GPIO bits to be applied to either DSR/DTR or the RTS/CTS pins. This is the only external serial port in the VT4 and VT6 designs, and it is the printer port of the VT5.

The secondary MCU serial port, port 1, is connected to the APU and the APU and the MCU exchange data and commands over this bidirectional serial link.

The VT5 hardware contains an additional 16C550 ACE (*Asynchronous Communications Element*, National Semiconductor's marketing name for a UART) which serves as the primary communications port in the VT5. The ACE register file is memory mapped into the MCU's address space by the bank switching hardware, and the ACE's interrupt output drives one of the MCU's external interrupt inputs. The ACE port provides hardware handshaking, split baud rates, a full set of modem control signals, and a hardware FIFO for received characters. A Maxim MAX214 RS232 level shifter provides the necessary internal logic to shift between DTE and DCE wiring for the primary communications port.

5.6 Power Supply

The power supply in all VT versions is essentially the same, a simple 7805 regulator, and the boards can accept any unregulated DC input from approximately 9 to 12V. Lower input voltages are possible by substituting a low drop out (LDO) regulator such as the LT1117 or LM2940T-5 for the standard 7805. All three boards require approximately 200mA.

6 FIRMWARE DESCRIPTION

6.1 DEVELOPMENT ENVIRONMENT

6.1.1 Tools

The MCU and the APU are both 8051 derivatives and the firmware for both can be compiled with the free SDCC cross compiler. You can download SDCC from the Source Forge archive at

http://sdcc.sourceforge.net

If you want to use the Makefile that we supply, then you'll need some version of the Make utility. GNU Make works and you can obtain that from

http://www.gnu.org/software/make/make.html

Although you don't need it, we recommend the Programmer's Notepad IDE, which can easily be set up to work with SDCC. You can find Programmer's Notepad at

http://www.pnotepad.org/

And no matter what else you use you'll need the Dallas Semiconductor MicroController ToolKit (MTK) utility to program the DS89C450 microprocessor:

http://files.dalsemi.com/microcontroller/dev_tool_software/mtk/

All these tools are free.

6.1.2 Building the Code

Makefiles are supplied with this distribution for both the MCU and APU, and these automate most of the work involved in building the firmware. Assuming you've installed all the necessary tools, you can simply "cd" to the mcu or apu subdirectory and type

```
make
```

and it will compile, assemble, link and checksum a new firmware image. The result of all this is in the file `FLASH.HEX` which, in the case of the DS89C450 MCU, you can then download directly to your hardware using MTK. For the APU, `FLASH.HEX` is still the file to program, but this time you'll need an EPROM programmer to re-flash the part.

Other useful targets included in the Makefile are:

- `make clean` - delete all generated files *except* `FLASH.HEX`
- `make depend` - regenerate source file dependencies

There are several build options defined near the top of the Makefile which you may change to generate firmware for different configurations.

6.2 MAJOR SOURCE MODULES

There are roughly thirty source files in the VT MCU firmware, and a smaller number in the APU firmware. The following paragraphs will give you an overview of some of the more important ones.

6.2.1 MCU Source Modules

gp.c (*graphics painter*) ─ updates the bitmap memory with characters and any other types of drawing. This code can apply attributes, such as highlighting and color; however, this code does not have any knowledge of underline, blinking, or even cursors. This module is highly aware of the layout of the bitmap memory. This module has no knowledge of the terminal behavior; it just draws into the bitmap.

pb.c (*page buffer*) ─ buffers information on each character that is stored in the terminal, and/or visible on the screen. Also includes a mapping function to map from the buffer onto the display.

cursor.c ─ implements code to cause the cursor to be displayed on the bitmap. The main work performed by the code is to request the gp module to draw the cursor and to implement blink mode. This module does not have any knowledge of conflicts between drawing the cursor and normal drawing operations. Other modules resolve the conflict and request that this module turn the cursor off when a conflict might arise.

blinker.c ─ implements the character blinking feature. Periodically examines each character position and paints or erases the character, to implement blinking.

interp.c ─ interprets character sequences received from the UART and converts them into actions to be performed. The interpreter operates from a table that is terminal-type specific.

sendbuf.c ─ implements buffers that are sent to the UART under a variety of conditions.

setup.c ─ implements the setup configuration screen.

timer.c ─ implements timers that can be used by other modules for blinking or other features.

eeprom.c ─ implements code to read and write non-volatile memory for storing configuration information, fonts, sendbuf contents, interpreter tables, and setup tables.

serial.c ─ handles interrupts from the UART. Copies received characters into a circular buffer. Copies transmit characters out of a circular buffer.

video.asm ─ generates the VGA video in real time, including the vertical sync. This module also includes routines to program the CPLD's color map, UMB and DMAUMB registers.

post.c ─ contains routines for the *power on self test*, plus functions to automatically determine memory size, the VT model, and test for the presence of the ACE chip.

vt.c ─ initialization and startup code. Also includes the machine readable copyright notice, time stamp and version number.

6.2.2 APU Source Modules

ps2.asm ─ PS/2 keyboard serial prototcol bit banging routines.

host.c ─ MCU communications routines.

scancode.c ─ PS/2 scan codes to ASCII translation table.

keyboard.c ─ keyboard interpretation functions (e.g. handles shift, control, and special function keys).

post.c ─ *power on self test* for the APU.

apu.c ─ initlalization and startup code.

6.3 CONFIGURATION DATABASE

The VT4 firmware supports the concept of a "database" that stores all of the configurable information. Specifically, it includes:

- **Fonts** — the bitmap representations of fonts.
- **Setup menus** — the setup code is table-driven and the tables are stored in the database.
- **Status lines** — status lines are table-driven and the tables are stored in the database.
- **Variables** — values to be loaded into the global variables can be stored in the database.
- **Keymaps** — mappings of keyboard keys to sequences are stored in the database.
- **Configurations** -- overall configurations of a terminal emulation. These specify the fonts, setup screens, etc. to be accessed during emulation.
- **Character buffers** -- strings to be sent in response to inquires (such as the ENQ character or ANSI "report" sequences) can be stored in the database.

A database consists of a contiguous set of bytes that are position-independent. The database manager can read/search (and update) each database when requested by other modules.

There are three databases that are accessed by the database manager. Accesses are made in the following order, from first-searched to last-searched. The first matching entry encountered is used, so that first-searched databases can effectively "override" later-searched ones.

- **RAM database** — A writable RAM database is maintained by the database manager. This database is used to store database records that have been downloaded, or that have been read from the EEPROM.
- **EEPROM database** — A writable database is maintained in EEPROM by the database manager. This database is often only read, but can be written via a "save" request. This database will normally be used to store persistent variables that are changed via one or more setup screens, but this can also contain other downloaded fonts, setup screens, or any other type of records.
- **Firmware database** — A basic database is compiled into the firmware. This database contains the fonts, "factory default" settings, setup screens, and other information that has been compiled in. This is a read-only database.

6.4 APU COMMUNICATIONS

The MCU and the APU communicate over a dull duplex asynchronous serial link. This link is implemented using the secondary UART in the DS89C450 MCU and the primary UART in the AT89LP4052 APU and uses the 8N1 character format at 4800 bps. Except for the voltage levels, which are TTL, it’s the same as any standard RS232 connection and with a suitable level converter it’s easy to “eavesdrop” on it with any PC. All messages sent in either direction are always exactly one byte in length, which means that there are no messy synchronization issues to worry about. The link is assumed to be error free and there are no provisions for error correction or even error detection.

6.4.1 APU to MCU Messages

All bytes in the range 0x00 to 0x7F that the APU sends to the MCU represent actual ASCII character codes. The APU handles all the basic translation of the typewriter array into ASCII, including shift, control and caps lock, and just sends the resulting text upstream. APU to MCU bytes in the range 0x80 to 0xCF represent special function keys that don't have any direct ASCII translation (e.g. **`F1`**..**`F12`**, **`Ins`**, **`Del`**, **`Home`**, **`End`**, **`Pg Up`**, **`Pg Down`**, arrow keys, numeric keypad keys, etc). The APU just converts these keys into these special bytes and then sends them up to the MCU. The MCU then figures out the appropriate escape sequence to send to the host.

The APU uses bytes in the range 0xF0 to 0xFF to send explicit up/down notifications to the MCU for all the usual modifier keys - **`shift`**, **`control`**, **`alt`**, and **`Windows`**. This allows the MCU to detect key combinations like **`Shift-F1`**, **`Alt-Pg-Up`**, etc and do whatever it wants. In each message, the LSB reflects the current key state, so for example 0xE8 would be shift up and 0xE9 shift down.

The APU sends messages in the range 0xE0 to 0xEF to tell the MCU the type of the attached keyboard. Codes in the range 0xB0 to 0xDF are current reserved for future expansion.

6.4.2 MCU to APU Messages

The MCU sends single byte messages to the APU to tell it to

- set or clear various modes (caps lock, key repeat, key click, swap caps lock and control, etc)
- set the translation for certain keys (backspace, delete, escape, comma, tilde, etc)
- turn on or off any of the VT4 or VT5 LEDs
- display an MCU POST code on the LEDs
- turn on or off keyboard LEDs
- set the keyboard type (PC/104, DEC LK450, etc)

A. VT Model Differences

	VT4	VT5	VT6
Serial Ports	1	1	2
RAM Size	32 or 64K bytes	128 or 512K bytes	128K bytes
Bank Switching Logic	None	4 bit UMB	2 bit UMB
Programmable DMA Bank		Yes	No
Multiple Host Sessions	No	No	Yes
LEDs	4 (POST) + 1 (Power)	4 (POST) + 1 (Power)	1 (POST & Power)

B. Parts List

VT6 Parts List

Reference Designator	Manufacturer	Part Number	Supplier	Stock Number	Description
Resistors					
`R1, R2`			Anchor		47Ω 1/8W 5% carbon resistor
`RP1`			Anchor		5x4.7kΩ SIP6 resistor (*one* common pin)
`RP2`			Anchor		3x330Ω SIP6 resistor (*no* common pin)
Capacitors					
`C1, C2, C3, C4`			Anchor		1μF 35V radial lead tantalum (0.1")
`C5`			Anchor		10μF 6V radial lead tantalum (0.1")
`C6-C11`			Anchor		0.01μF 50V ceramic mono (0.1”)
`C12`			Anchor		0.1μF 50V ceramic mono (0.1")
`C13`			Anchor		100μF 16V radial lead aluminum (0.2")
Semiconductors					
`D1`		1N4001	Anchor		50PIV 1A power diode
`D2`		1N914	Anchor		Small signal switching diode
`VR1`		7805	Anchor		5.0V fixed output TO-220 regulator
`U1`	Dallas/Maxim	DS89C450			8 bit ultra high speed microprocessor PLCC44
`U2`	Atmel	AT89LP4052			8 bit microprocessor DIP20
`U3`	Maxim	MAX232			RS232 Driver 5V DIP16
`U4`	Atmel	24C01			128x8 I2C EEPROM DIP8
`U5`	Atmel	ATF2500C			EE CPLD 20ns PLCC44
`U6`		628128			128Kx8 SRAM 55ns 0.6" DIP32
`LED1`					2mm LED 5V (Red)

REFERENCE DESIGNATOR	MANUFACTURER	PART NUMBER	SUPPLIER	STOCK NUMBER	DESCRIPTION
OSC1	CTS	MX045HS			25.175MHz oscillator DIP8
MISCELLANEOUS					
BZ1	Cui	CEM-1205			5VDC Magnetic Buzzer
J1					HD-15 Female PC mount right angle
J2	Cui	PJ-102AH			Coaxial power jack (2.1 x 5.5mm)
J3					DB-9 Male PC mount right angle
J4	Cui	MD-60SM			6 pin Mini-DIN Female PC mount right angle
JP1					2 pin 0.1" header (jumper)

Table 10 – VT6 Parts List