TEX.DOC
Bob Clements
06 July 89
TEX.EXE Version: 3.001

This document describes the program "TEX", the TCS Master program for the TC2000 computer.

General info about TEX

"TEX" stands for "Tcs EXecutive".

TEX is the TCS master program for the first release of the TC2000 computer, implementing a subset of the full TCS master's features.

TEX provides power on/off, reset, bootstrap, examine/deposit and TTY simulation functions, as well as direct access to the low-level TCS slave functions. It also contains routines to scan the system configuration and to report that information to the operating system software.

TEX provides a "console terminal" for the operating system running on the TC2000 computer. Characters are passed between the TCS's console and buffers in the TC2000 computer's memory.

See the section "How TEX runs the TC2000 system" for a discussion of TEX's operation and how it relates to the TC2000 operating system. Also, see the section "Power-on Servicing features" for information on TEX's support for servicing the TC2000.

TEX provides a number of services to the operating system at its request. These allow the operating system to control halting, reloading, reconfiguring and monitoring of the butterfly 2 hardware.

TEX also contains some initial diagnostic routines for testing a TC2000 processor node and for setting board revision and serial number information. It is expected that these functions will eventually be moved out of TEX and into other programs used by manufacturing.

TEX is written in C, compiled using Microsoft C and Sun's PC-NFS, with sources on our Sun servers. At present, the sources and this document reside in /usr/bfly2/src/tcs-m/tex/rel.

TEX is loaded from a hard disk or a floppy disk, running under MS-DOS or PC-DOS.

The user interface to TEX is via a menu system which is essentially the same as that used in other TC2000 system diagnostic software.

TEX takes commands from the console terminal of the TCS master. This can be either a PC-style keyboard and display or a serial terminal on the TCS master's COM port. A separate device driver (SDRIVER.SYS) implements the serial port and modem port interface.

TEX will execute a script of commands from a file if that file's name is included on the DOS command line which starts TEX. Additionally, TEX will execute a script file by using the DOS input redirection mechanism, i.e., "tex < file" or by specifying the command file to a "Utility Do" command within TEX.

TEX provides calendar time service for the TC2000 computer's operating system. Since the DOS operating system does not properly deal with daylight savings time, TEX requires that the TCS master's calendar clock and DOS times be in UTC (GMT).

Some additional features are yet to be implemented. As they get done, this document may therefore be incomplete.

Invoking TEX from the command line

TEX takes a few switches on the DOS command line. The switches can be typed as any unique abbreviation of their full names.

-Autoboot causes TEX to automatically read the bootstrap configuration file BOOTCFG.TCS and the slot configuration file SLOTCFG.TCS before executing any other commands. These files configure TEX for a particular installation's hardware and operational characteristics. They are described in detail in a later section. TEX then begins to operate the system as described for the "Application Run" command.

-Menu and -NoMenu control whether the menu system displays menus to the user. -Menu, the default, is equivalent to specifying the commands "Terminal Hardcopy" and "Terminal Menu-On". -NoMenu turns menu displays off, and is equivalent to "Terminal Hardcopy" and "Terminal Menu-Off".

-TCS-Addr followed by a hex number sets the default slave address. This is equivalent to the "Utility Slave" command at the TEX command level, but is provided for use in batch files.

-help prints out the list of these command line switches.

A filename may also be specified on the command line. This will cause TEX to execute the commands in that file as if a "Utility Do filename" had been entered, before executing any other commands from the keyboard (or from any redirected input stream).

Commands to TEX

The user interface to TEX is through a menu system. The status line of most menus is:

TC2000 TEX[x.y.z] version n.nnn"

"x.y.z" is the current default TCS slave address and "n.nnn" is the version number of this copy of TEX. When started, TEX displays the top-level menu and a command prompt of "TEX->". The prompt changes to include the name of the current menu when a sub-menu is selected. Typing question mark - carriage

return will produce a list of the available commands or sub-menus at any point.

Commands may be entered by typing either their name or their number as shown on the current menu. Names may be shortened to any unique abbreviation, often (but not always) a single letter.

Selecting a particular command is done by entering the names of any submenus between the current menu and the desired command. For example, from the top-level TEX menu, the "blink" command is reached via the intermediate "Hardware" submenu. The command would be entered as "Hardware Blink" (or just "h b") from the top menu. It would be entered as "Blink" (or just "b") from the hardware menu. If the desired command is not in the current menu or its submenus, the command can still be entered by preceding it with a "/" character. For example, "/h b" could be used while the current menu is the "Memory" submenu.

In this document, command names will be shown as if the top-level menu is the current menu.

When numbers are supplied as arguments to commands they are assumed to be in decimal. Hexadecimal numbers must be specified by preceding the number with "0x". That is, the number 16 decimal may be specified either as "16" or as "0x10".

Card addresses

Many of TEX's commands refer to a specific TC2000 card or a group of cards. In the descriptions below, the term "card address" means a description of this address argument to a TEX command. The simplest case of a "card address" is a group of three numbers separated by dots. For example, the command "Hardware Blink 0 7.0.1" turns off the yellow LED on the processor node card which is located in bay 7, midplane 0, slot 1 of the machine.

The third field of a card address, the "slot" number, can also refer to a switch or clock card. The "slots" SA, SB, RA, RB, CA and CB refer to the "A" and "B" Switch Server cards, Switch Requestor cards and Clock generator cards. For example, the command "Hardware Blink 0 7.1.RA" turns off the yellow LED on the Requestor card of the B switch in bay 7, midplane 1.

For the clock cards, the Bay and Midplane fields should be omitted. For example, the command "Hardware Blink 0 CA" turns off the yellow LED on Clock card A.

The third field can also be replaced by the letter "g" and a hex number from 0 to FF. This means to use a target address containing a "Group address" as defined in the TCS slave documentation. The command "Hardware Blink 0 G4" sends a command to all slaves which respond to group address 4 (normally all processor nodes). The special address "ALL" is recognized as the "broadcast" group address.

Card addresses can also contain "wild cards". These cause a command to be repeatedly executed for each of a number of addresses in succession. For example, "Hardware Blink 0 6.*.*" performs a "Hardware blink 0" command for each existing node in bay 6. And the command "Hardware Blink 0 *.*.RB" turns off the yellow LED on each existing Switch Requestor card connected to switch B of the machine. The "slots" "C*", "R*" and "S*" mean both A and B clocks or switch cards.

In the preceding paragraph, "existing" means that a card was mentioned in the slot configuration file or that it has responded to a message from TEX since TEX was started. This speeds operations up by skipping cards which are not actually in the machine. If the "*" is replaced by a "!", however, the wild card will be taken to include all cards in the described range, even if they have not been heard from and are not in the slot configuration file.

Finally, a card address can also be an arbitrary four-digit hex number in the range 0000-7fff. This number will be placed in the two address bytes of the TCS slave messages. This is useful for testing TCS slave implementations. The number "3907", for example, is the same as the card address "7.1.7".

Menu summary

This section lists the submenus available from the top level TEX menu. Next, the lower submenus and the individual commands are listed.

Here is the top-level TEX menu. Each entry is a submenu except for the "Quit-to-DOS" command. The submenus are described in succeeding paragraphs.

```
Menu:   TEX
    1.  Application
    2.  Configuration
    3.  Hardware
    4.  Memory
    5.  Slave
    6.  Terminal-Type
    7.  Utility
    8.  Quit-to-DOS
```

The "Quit-to-DOS" command causes TEX to exit to the DOS operating system. This should not be done in normal operation, because TEX's knowledge of the current state of the TC2000 system is discarded when TEX quits.

```
Menu:   TEX Application
    1.  Configuration
    2.  Examine-Card
    3.  File-Select
    4.  King-Node
    5.  Mode
    6.  Run
    7.  System-Status
    8.  TTY
    9.  Quit-to-TEX-Menu
```

The TEX Application menu controls the normal operation of TEX when it is supporting the TC2000 computer's operating system or any other program which runs in a TC2000 function card (as opposed to running in the TCS Master itself). This menu contains the commands for setting up an application and its bootstraps. It also directs TEX to run that application and it contains commands for examining the state of the system and of individual cards.

```
Menu:   TEX Application File-Select
    1.  Application
    2.  POST
    3.  UBoot
    9.  Quit-to-Application-Menu
```

The TEX Application File-Select menu contains commands for selecting the three binary files used in running the application. The first two files are those which initialize the TC2000 function cards. The third is either the application or its bootstrap.

```
Menu:   TEX Configuration
    1.  Card-Use
    2.  Forget
    3.  Reconfigure
    4.  Scan
    5.  Write
    6.  Quit-to-TEX-Menu
```

The TEX Configuration menu contains commands which are used to control configuration information about the TC2000 system. This information generally resides in configuration files but it can also be modified in the TCS Master processor after the files have been read, and modified files can be written.

```
Menu:   TEX Configuration Write
    1.  Boot-Configuration
    2.  Slot-Configuration
    3.  Quit-to-Write-Menu
```

The TEX Configuration Write menu contains commands to update the two main configuration files.

```
Menu:   TEX Hardware
    1.  Blink
    2.  Diagnostic
    3.  Frequency
    4.  Ftest
    5.  Go
    6.  Halt
    7.  Initialize
    8.  Margin
    9.  PDU
    10. Power
    11. Temperature
    12. Voltage
    13. Quit-to-TEX-Menu
```

The TEX Hardware menu contains low-level commands which access hardware features of the system. These commands control and examine the various cards directly, and they should be used with caution. Some of these commands can crash the TC2000 application if used indiscriminately.

```
Menu:   TEX Memory
    1.  Deposit
    2.  Disassemble
    3.  Dump
    4.  Examine
    5.  Load-File
    6.  UBWait
    7.  Zero
    8.  Quit-to-TEX-Menu
```

The TEX Memory menu provides commands for reading and writing the memory of a TC2000 function card. The "Deposit", "Load-File" and "Zero" commands should be used with caution to avoid crashing the TC2000 application.

```
Menu:   TEX Slave
    1.  Read
    2.  Write
    3.  Quit-to-TEX-Menu
```

The TEX Slave menu, through its Read and Write submenus, provides access to all the low-level registers in the TCS slave processors of the system. It also provides access to the nonvolatile EEPROM storage within each TCS slave processor.

Menu: TEX Slave Read
1. Action-Register
2. EEPROM-Register
3. Gate-Array-Register
4. Hardware-Register
5. Card-Type
6. Serial-Number
7. Artwork-Rev
8. Electrical-Rev
9. TCS-Slave-Rev
10. Quit-to-Read-Menu

The TCS Slave Read menu provides commands to read all the registers in a TCS slave processor, including the nonvolatile EEPROM registers which describe the card's serial numbers and revision levels.

Menu: TEX Slave Write
1. Action-Register
2. EEPROM-Register
3. Gate-Array-Register
4. Hardware-Register
5. Card-Type
6. Serial-Number
7. Artwork-Rev
8. Electrical-Rev
9. TCS-Slave-Rev
10. Quit-to-Write-Menu

The TCS Slave Write menu provides commands to write all the registers in a TCS slave processor, including the nonvolatile EEPROM registers which describe the card's serial numbers and revision levels. These commands should be used with caution to avoid crashing the application using the card. Serial number and revision level information should never be changed except under strict revision control procedures.

Menu: TEX Terminal-Type
1. VT320
2. VT100
3. Sun
4. X
5. PC
6. Hardcopy
7. Menu-On
8. Menu-Off
9. Quit-to-TEX-Menu

The TEX Terminal-Type menu provides commands which turn the menu display on and off and which select the particular terminal type being used. This allows the menu interface to correctly manage the screen display.

Menu: TEX Utility
1. Do
2. Slave
3. Trace
4. UART-Init
5. Version
6. Quit-to-TEX-Menu

The TEX Utility menu is a catch-all for some overall control commands for TEX.

Menu: TEX Utility Trace
1. Driver
2. Loader
3. PhysIO
4. RunFSM
5. Service
6. Error
7. Quit-to-TEX-Menu

The TEX Utility Trace menu contains the trace-level setting commands. These are generally used for finding system problems by producing extra output as TEX operates.

Command descriptions

This section lists all of TEX's commands in groups according to the menu in which they are found, and alphabetically by their full menu path name.

TEX Application Configuration

This command writes the configuration information for the system into the memory of the current slave (which should be a processor, not a switch or clock card). The information must have been previously obtained, either by a "Configuration Scan" command (see below) or automatically as a result of an "Application Run" command. This information is written in a predefined format in page 0 of the processor's memory. The serial number of the current slave is also read from its TCS EEPROM and stored in the processor's memory.

In automatic operation, the configuration information is written to each node after the microboot successfully completes and again before the application bootstrap starts.

TEX Application Examine-Card

This command produces a display of information about the selected TC2000 card. The card's type, serial number and revision levels are displayed (from the EEPROM on the card). The current use, goal and state information are also shown.

TEX Application File-Select Application

This command specifies one of the files to be used by the "Application Run" command. This file is used as the application program (or its bootstrap) and is loaded into the King node after all nodes are initialized. The file must exist on the TCS master's DOS filesystem. This file is usually named "BOOT.88".

TEX Application File-Select POST

This command specifies one of the files to be used by the "Application Run" command. This file is used as the Power-On Self Test (POST). It is loaded and run in every node after the successful completion of the microboot program. The file must exist on the TCS master's DOS filesystem. This file is usually named "POST.88".

TEX Application File-Select UBoot

This command specifies one of the files to be used by the "Application Run" command. This file is used as the microboot. It is loaded and run in every node after the TCS slave initializes the node. The file must exist on the TCS master's DOS filesystem. This file is usually named "UBOOT.88".

TEX Application King-Node

This command specifies the card address of the king node, which is the master node of a multi-processor application. This node will be used by functions of the "Application Run" command. It specifies the node which will be loaded with the application bootstrap and polled for TTY characters and service requests. This command also sets the "default" TCS address, usually specified by the "Utility Slave" command, to the same node.

TEX Application Mode

The mode command sets the "system mode" which directs TEX's operation under the "Application Run" command. After selecting the desired mode, the "Application Run" command must be used to accomplish the selected operation.

"Application Mode Idle" directs TEX to discover all cards in the system and to turn on their power and to initialize them, but not to load any application program. TEX will run the microboot and POST programs on processor nodes.

"Application Mode application" directs TEX to first initialize the system as described under "Application Mode Idle". Then it loads and runs the application (or its bootstrap) as specified in the "Application File-Select a" command into the current "king" node as specified by the "Application King" command. The application bootstrap file and king node may also be specified in the BOOTCFG.TCS file.

"Application Mode boot" directs TEX to halt all processors and then restart the initialization and bootstrap procedures as described for "Application Mode application".

"Application Mode off" directs TEX to halt all processors and then turn off power to all processors, switches and clocks.

"Application Mode wait" sets the system and card goals to an "unknown" state. This can be used in testing to cause TEX to communicate with the current king node for TTY interactions but not to bring any other nodes up or down.

TEX Application Run

The "Application Run" command places TEX in automatic operation as selected by the "Application Mode" command. While running, TEX polls all cards in the system and brings them to the state defined by the current system mode. This includes sending initialization commands to the TCS slaves, running the microboot and POST programs and loading the application bootstrap into the "king" node. TEX also acts as the console terminal for the current king node, once that node is operational. TEX polls for "service requests" from the king node also. Service requests are described in a later section.

For a more complete discussion of operation under the "Application Run" command, see the section "How TEX runs the TC2000 system".

The operator can return to the TEX command prompt by typing the two characters "^." if TEX is acting as the console for a king node. If there is no running king node then typing a control-C character will return to the command prompt.

To send an arbitrary control character to the processor node, the escape sequence "<backprime><char>" sends a

Control-<char> to the processor. To send a back-prime to the processor, enter two of them in a row.

To use the console terminal function without running the state machines which bring the system up, reboot it, or shut it down, use the "tty" command.

TEX Application System-Status

The "Application System-Status" command prints out the state of the clock cards and any switch and processor cards.

The states will depend on the current system mode and on whether the cards have successfully been initialized and whether their bootstraps and POST have run.

If the system is in "application" mode and the processors have finished all their booting operations the status will be "HAPPY". The king node, if defined, will be in state "KING". Nodes which failed their initialization or their startup tests will be reported as "SICK".

A "goal" will also be reported for each card. This is determined by the system's current mode and by the current "use" of the card (see the "Configuration Card-Use" command).

TEX Application TTY

This command enters the console TTY protocol routine, using the current default slave as specified by the "Utility Slave" command. NOTE: This is NOT necessarily the same node as the "King" (or "Master") node.

Characters are passed to and from the TC2000 processor node at the default address over the TCS bus, using ring buffers in the processor's memory. The escape sequence to return to the TEX command scanner, is backprime-dot (“`.”). (Backprime is un-shifted tilde on most keyboards, so this is similar to "tip"'s escape, but different, so that TEX can be run from a tip line.)

To send an arbitrary control character to the processor node, the escape sequence "<backprime><char>" sends a Control-<char> to the processor. To send a back-prime to the processor, enter two of them in a row.

TEX Configuration Card-Use

This command sets the configuration "use" field for a card or group of cards. The "use" field then directs TEX's operation under the "Application Run" command. The "use" for a card is defaulted to "system" if it has not been set before the "Application Run" operation starts. "Use" can also be set by the BOOTCFG.TCS file when TEX starts up.

The following values are defined for the "use" field.

OFF: Tells TEX to leave the particular card turned off and not to use it. This can be used for a known defective card.

ISOLATE: Tells TEX to turn the card on, but to leave its switch port turned off so that it cannot interact with other cards in the system. This is intended to allow power-on servicing of TC2000 cards. A suspected card might be marked "OFF" in the BOOTCFG.TCS file and then switched to "ISOLATE" to be serviced.

SYSTEM: Tells TEX to use the card normally as part of the normal system operation.

AUXILIARY: Tells TEX to use the card normally, except that it will not be reloaded on a system reboot. This is intended for nodes running under pSOS or other special applications. Typically, the node will be treated as "SYSTEM" usage until the operating system tells TEX, via a service request, to consider the node as "AUXILIARY".

FORGET: Tells TEX to clear its lknowledge of the card's usage. This is used for testing.

TEX Configuration Forget

This command clears the configuration determined by "Configuration Scan", "Configuration Use" and "Application Run" commands, so that it will be re-discovered by the next "Application Run" or "Configuration Reconfigure" command. This command is useful when changing the BOOTCFG.TCS file.

TEX Configuration Reconfigure

This command causes TEX to re-read the two configuration files, BOOTCFG.TCS and SLOTCFG.TCS. This is a quick way to restore the configuration variables after having made some changes.

TEX Configuration Scan

This command causes TEX to attempt to contact all of the TCS slaves in the machine and to record and display what it finds.

The argument selects the portion of the machine to be scanned: If "tile" is specified, all cards mentioned in the SLOTCFG.TCS file are scanned. If "large" is specified, all eight midplanes in all eight bays are scanned. If "medium" is specified, all eight midplanes in bay 7 are scanned. If "small" is specified, only midplane 7 of bay 7 is scanned.

As the scan is done, the board type, serial number, and revision levels are typed out. The board types are also saved for the "Application Configuration" command, see above.

TEX Configuration Write Boot-Configuration

This command creates a bootstrap-time configuration file. The file used by TEX is BOOTCFG.TCS, but a different file name can be used in this command to create other copies or versions of the configuration. The contents of the BOOTCFG.TCS file are described in detail in a later section.

This command is typically used as follows: The existing BOOTCFG.TCS file is read in, perhaps by "Configuration Forget" followed by "Configuration Reconfigure". Then a

configuration item is changed, such as "Configuration Card-Use OFF". Finally a new BOOTCFG.TCS file is written by this command.

TEX Configuration Write Slot-Configuration

This command creates a slot configuration file. The file used by TEX is SLOTCFG.TCS, but a different file name can be used in this command to create other copies or versions of the configuration. When TEX is first installed, a "Configuration Write-Slot slotcfg.tcs" must be done before automatic operation is used.

The slot configuration file contains the card address, card type, serial number and revision information about each card in the system. The "Configuration Write-Slot" command writes such a file, including all cards which are presently known to TEX, due to a "Configuration Scan" or Application Run" command, and which are not disabled (by a "Configuration Card-Use OFF" command or a "DisabledCards" or "CardUse" line in the BOOTCFG.TCS file).

TEX Hardware Blink

This command sets the "blinking" state of the yellow "FLAG" indicator on a card or group of cards. The states are: 0: off 1: slow blink 2: fast blink 3: on

The conventional usage of the yellow LED is:

On:	Not yet initialized or cannot be initialized
Blinking:	Being initialized or in need of service
Off:	Operational

TEX Hardware Diagnostic

This command runs one of a set of builtin diagnostic routines. Test number zero means to run all of the tests in order. The following tests exist:

1 Write/Read floating 0 and floating 1 patterns to the TCS slave's action register 7. This is a RAM register within the TCS slave provided for this purpose, to test the Master-to-Slave communication path.

2 Write/Read floating 0 and floating 1 patterns to SIGA-A on the current default slave's board. This test uses the data and address registers in the SIGA but does NOT cause any T-bus cycles. This tests the communication path from the TCS Master to the SIGA.

3 Write/Read floating 0 and floating 1 patterns to location zero (32-bits) of the processor's RAM. This tests the path from the SIGA through the T-bus to the RAM.

4 Write/Read the values 00 through FF to locations 00 through FF of the processor's RAM. Then write/read the complement of those values. This tests the addressing of processor's RAM, but only the low eight address lines. The TCS serial bus is too slow to support a full memory test of this sort.

5 Test that the 88K processor can be started and that it can access memory. This test loads a short program into locations 600-614 of memory, sets a data word at location 700, and sets the start vector at locations zero and four. It reads these words back to verify them. Then it starts the processor by releasing the reset signal, waits briefly and stops the processor. It then checks to see that the data at location 700 has been complemented by the 88K's program as it should be.

The actual program and data words loaded are:

```
000/ c0000180  br 600
004/ f4005800  or r0,r0,r0 (nop)
600/ 15400700  ld r10,r0,700
604/ f560540a  xor.c r11,r0,r10
608/ 25600700  st r11,r0,700
60c/ c0000000  br 60c
610/ f4005800  or r0,r0,r0 (nop)   (for prefetching)
614/ f4005800  or r0,r0,r0 (nop)   (for prefetching)
700/ 12345555
```

TEX Hardware Frequency

This command tells TEX the frequency of the master clock for the system.

The default frequency is 38 for a 38 megahertz clock. The argument must be an even number within the range of the clock card. This number is later used to initialize the clock (by "Hardware Initialize" or "Application Run").

The frequency is given to the TC2000 processor nodes (by "Application Configure" or "Application Run") to allow them to initialize properly. These numbers may also be set by the BOOTCFG.TCS file.

TEX Hardware Ftest

This diagnostic command exercises the TC/CLK card to determine the lock range of its phase-locked loop. The printout reports the results. The current default slave is tested; it must be a clock card.

TEX Hardware Go

This command starts the currently selected TC2000 processor node at the specified address. If the address is zero, the last start address from a loaded file is used. This function operates by writing two words in memory at locations zero and four. The first is a br.n instruction to the start address. The second is a load instruction which sets register r2 to the board's node number, from 0 through 511 decimal (1ff hex). TEX then asserts a processor reset, waits briefly, and removes the reset. The processor will begin by executing these two instructions.

TEX Hardware Halt

Halt a processor node (or group of nodes). TEX does this by asserting the processor reset line, but not the board reset

line. While the processor is halted other commands may be performed, such as examine and deposit and load. When the processor reset is removed, such as by the "Hardware Go" command, the processor will start at the reset vector at location zero.

TEX Hardware Initialize

The "Hardware Initialize" command performs a board-specific initialization sequence for a processor node, switch or clock card. The card address may have wildcards.

For the clock card, the card is reset and powered on and the phase-locked loop is set to the frequency specified by a previous "frequency" command (default 38 MHz).

For the switch cards, the sequence is similar, except that the switch gate arrays are initialized and all eight function card ports are disabled. Switch cards should be initialized before processor nodes. This sequence will leave unused switch ports disabled. Active ports are enabled when the processor nodes are initialized.

For a processor node, TEX reads the board status, turns on the power, asserts processor and board resets, and turns on the LED blinker. Then it waits a while and removes the board reset. The processor remains halted. TEX then enables the switch path in each LCON and sets the addresses of each SIGA to their default values. Finally it turns on the switch path for this node in the switch cards that serve it.

TEX Hardware Margin

This command sets the voltage margin logic on the specified card. The first argument defines the margin level as follows:

```
H - HIGH  +10%
h - high  +5%
n - normal
l - low   -5%
L - LOW   -10%
```

Note that this command inherently turns ON the power for the card if it is not already on.

TEX Hardware PDU

The "Hardware PDU" (Power Distribution Unit) command is used to turn on or off the bulk power supplies for the system by controlling the main power relays.

The first argument is either "on" or "off".

In a large system the PDUs are split into two groups, controlled by relays "1" and "2". The second argument is either "1", "2" or "both".

If TEX is started with a command file or with the "-Automatic" switch, it will turn on both PDUs when starting.

TEX Hardware Power

Turn power on or off at a card or group of cards. If "all" is used, a broadcast message is used to turn the slaves off. "Hardware Power On" is equivalent to "Hardware Initialize".

TEX Hardware Temperature

The "Hardware Temperature" command produces a report of the temperature found at the sensor or sensors on the specified card or cards.

TEX Hardware Voltage

The "Hardware Voltage" command produces a report of the various voltages within the specified card or cards. The present reporting algorithm does NOT correctly use the calibration points stored in the EEPROM on the cards. This will be improved in a future release.

TEX Memory Deposit

This command performs one write into the memory space of a processor node. The first argument is "b" or "h" or "w" for a byte, halfword or word cycle. This command may be executed while the processor is halted or running. The address need not be in the processor's local RAM. Anything currently available to the T-bus may be written, though only 32 bits of address are supported.

TEX Memory Disassemble

This command dumps an area of processor memory formatted as MC88100 instructions. The area starts at "address" and is "word_count" words long. "Word_count" is truncated to 512 words.

TEX Memory Dump

This command produces a hex dump of an area of main memory on the current slave, which should be a processor node (not a switch or clock). The area starts at "address" and is "byte-count" bytes long. "Byte_count" is truncated to 2048 bytes.

TEX Memory Examine

This command performs one read from the memory space of a processor node. The first argument is b or h or w for a byte, halfword or word cycle. This command may be executed while the processor is halted or running. The address need not be in the local RAM. Anything currently available to the T-bus may be read, though only 32 bits of address are supported.

TEX Memory Load-File

This command loads a file from the TCS Master's DOS file system into the current default node, which should be a processor (not a switch or clock).

The loader can read both COFF and a.out format files. The header information is summarized on the console and dots are printed as the loading proceeds (one dot

every 256 bytes). The default starting address for the "Hardware Go" command is set from the file's entry vector.

The processor is NOT started by the "Memory Load-File" command -- you have to type "Hardware Go". This allows loading of multiple files, and multiple nodes, before starting the processor.

TEX Memory UBWait

Wait for microboot to finish. This command should be used in a script file to allow the microboot to finish. It knows where the microboot's state register lives in the processor's memory and it monitors that cell for completion. (It will also time out after a while in case the boot fails.)

"Memory UBWait" should be used to wait for the microboot to finish before loading the next application or bootstrap program.

TEX Memory Zero

This command clears an area of memory in the current default node. It writes words of zero into the specified range (and bytes at the edges if the range is not word aligned). At TCS-bus speeds, this will be acceptable for a few K words, but will be slow for larger areas. (Remember that the microboot clears all of RAM.)

TEX Slave Read Action-Register

Read a TCS slave's "action register". The "action registers" are described in the TCS slave code document. This command allows direct access to these registers. The data is one byte wide. See the TCS slave documentation for specifics.

TEX Slave Read EEPROM-Register

Read a TCS slave's EEPROM. The EEPROM registers are described in the TCS slave code document. This command allows direct access to these registers. The data is one byte wide. See the TCS slave documentation for specifics.

Note that the EEPROM of a TC/SR card is accessed by a different method, since the TC/SR does not have a TCS slave processor on it. [In particular, you could specify "Utility Slave 7.0.RA" and try to do an eeprom access, but it would not do the right thing.]

TEX Slave Read Gate-Array-Register

Read a gate array on the node or switch. The various boards address the gate arrays differently and the various gate arrays have differently sized registers. See the TCS slave documentation and the gate array documentation for specifics.

TEX Slave Read Hardware-Register

Read a TCS slave's hardware port. The slave processors have access to some direct I/O ports on the various boards. Details vary among the boards. This command performs read operations to the "hardware" registers. See the TCS slave code description for specifics.

TEX Slave Read Card-Type

This command reads a portion of the EEPROM on the currently selected TCS slave. This command reads the EEPROM byte which specifies the card type.

TEX Slave Read Serial-Number

This command reads a portion of the EEPROM on the currently selected TCS slave. This command reads the EEPROM bytes which specify the card's serial number.

TEX Slave Read Artwork-Rev

This command reads a portion of the EEPROM on the currently selected TCS slave. This command reads the EEPROM bytes which specify the card's artwork revision level.

TEX Slave Read Electrical-Rev

This command reads a portion of the EEPROM on the currently selected TCS slave. This command reads the EEPROM bytes which specify the card's electrical revision level.

TEX Slave Read TCS-Slave-Rev

This command reads a portion of the EEPROM on the currently selected TCS slave. This command reads the EEPROM bytes which specify the card's TCS slave processor code revision level.

TEX Slave Write Action-Register

Write a TCS slave's "action register". The "action registers" are described in the TCS slave code document. This command allows direct access to these registers. The data is one byte wide. See the TCS slave documentation for specifics.

TEX Slave Write EEPROM-Register

Write a TCS slave's EEPROM. The EEPROM registers are described in the TCS slave code document. This command allows direct access to these registers. The data is one byte wide. See the TCS slave documentation for specifics.

Note that the EEPROM of a TC/SR card is accessed by a different method, since the TC/SR does not have a TCS slave processor on it. [In particular, you could specify "slave 7.0.RA" and try to do an eeprom access, but it would not do the right thing.]

Changing EEPROM registers which specify serial numbers and revision levels should only be done under strict revision control procedures.

TEX Slave Write Gate-Array-Register

Write a gate array on the node or switch. The various boards address the gate arrays differently and the various gate arrays have differently sized registers. See the TCS slave documentation and the gate array documentation for specifics.

TEX Slave Write Hardware-Register

Write a TCS slave's hardware port. The slave processors have access to some direct I/O ports on the various boards. Details vary among the boards. This command performs write operations to the "hardware" registers. See the TCS slave code description for specifics.

TEX Slave Write Card-Type

This command writes a portion of the EEPROM on the currently selected TCS slave. This command writes the EEPROM byte which specifies the card's type.

TEX Slave Write Serial-Number

This command modifies the serial number field of the EEPROM in the selected TCS slave.

THIS SHOULD NEVER BE DONE WITHOUT PROPER MANUFACTURING TRACKING OF THE NUMBERS!

The serial number is a 16-character text string and the other fields are 2-character text strings. At this writing, the formats for some of these strings are not yet defined. The software expects strings of letters, digits and hyphens, with no blanks. The serial number string should be in the form: X123-9999 where the "9999" may be from 1 to 4 digits. Strings are padded with trailing spaces to fill the two or sixteen character fields in the EEPROM.

Note that this command does not yet provide a method of writing these values to the TC/SR The TC/SR has no TCS slave processor. The logic to use the TC/SS processor to write the TC/SR EEPROM is not yet completed.

TEX Slave Write Artwork-Rev

This command writes a portion of the EEPROM on the currently selected TCS slave. This command writes the EEPROM bytes which specify the card's artwork revision level.

THIS SHOULD NEVER BE DONE WITHOUT PROPER MANUFACTURING TRACKING OF THE NUMBERS!

TEX Slave Write Electrical-Rev

This command writes a portion of the EEPROM on the currently selected TCS slave. This command writes the EEPROM bytes which specify the card's electrical revision level.

THIS SHOULD NEVER BE DONE WITHOUT PROPER MANUFACTURING TRACKING OF THE NUMBERS!

TEX Slave Write TCS-Slave-Rev

This command writes a portion of the EEPROM on the currently selected TCS slave. This command writes the EEPROM bytes which specify the card's TCS slave code revision level.

THIS SHOULD NEVER BE DONE WITHOUT PROPER MANUFACTURING TRACKING OF THE NUMBERS!

TEX Terminal-Type

This command informs TEX's menu interface what type of terminal is in use so that it can properly maintain the menus on the screen. It also allows the user to turn the menu display on or off by means of the "Menu-On" and "Menu-Off" selections.

TEX Utility Do

This command specifies a file of commands for TEX to execute. Any sequence of TEX commands may be put in this file, one per line.

If a "do" file contains another "Utility Do" command, the second file will be executed, but the first file will not be resumed. That is, "do" files cannot be nested. However, a "do" file can be executed within a REDIRECTED command file. For example, if file "script1" contains:

```
command1
utility do script2
command2
```

then command2 will NOT be executed by a "Utility Do script1" command or by a "tex script1" DOS command, but it WILL be executed by a "tex < script1" command.

TEX Utility Slave

This command sets the default target TCS address for those commands which allow a default or which take no address argument. See also the "king" command.

TEX Utility Trace [Driver | Loader | PhysIO | RunFSM | Service | Error]

This group of commands sets various trace printing control variables within TEX. Some of these traces produce masses of output; use them with caution.

The "driver" variable, if non-zero, causes traces at the TCS command driver level.

The "physio" variable, if non-zero, causes traces at the TCS UART I/O level.

The "error" variable, if non-zero, causes tracing of some error conditions dealing with TTY emulation. (It is used for debugging. Its use may change in later versions.)

The "loader" variable is bit encoded: The "1" bit prints a summary of the loaded file; the "2" bit prints info on each "section" of the file; the "4" bit prints the value and address of each word of the file.

The "runfsm" variable controls printing of state changes in the finite state machine logic of the "Application Run" command.

TEX Utility UART-Init

This command causes the UART for the TCS serial bus to be reinitialized. The argument specifies the baud rate of the serial bus. The default is "fast". The "slow" argument is now obsolete.

TEX Utility Version

The version command causes TEX to print its version number and compilation date.

How TEX runs the TC2000 system

This section is an overview of the mechanisms used by TEX to operate the TC2000 system. It is included as an aid to understanding and using the application and configuration control commands of TEX.

TEX deals with each processor (TC/FPV, etc), each switch pair (TC/SS and TC/SR) and each master clock (TC/CLK) as a separate task. Each card is represented internally as a finite state machine (FSM) with states such as "Unresponsive", "Initializing", "Happy" etc. TEX runs these state machines (and thus the system) when it has been commanded to do so by an "Application Run" command or by being started by "tex -Auto" from DOS.

The overall system mode, the usage of each card and the responses to messages sent by TEX to each card are the inputs to the card state machines.

As an example, consider a system being run normally but with one processor node disabled due to a hardware failure. When starting, TEX reads the two configuration files, BOOTCFG.TCS and SLOTCFG.TCS. These tell TEX what cards should be in the system, whether any of them are to be disabled, which node is to be the master node and what the overall system operation mode should be. For this example, the system mode will be "Application". The card uses will be "System" except for one card which has the use "Off" because it is defective. The "Off" usage is declared by a "CardUse" line in BOOTCFG.TCS. All components start in the "Unknown" state when TEX is started. The desired state is "King" for the master node, "Happy" for all other working nodes and "Disabled/Off" for the defective node.

This status information can be seen by using the "Application System-Status" command, which lists information about all parts of the system, or by the "Application Examine-Card" command which give more information about individual cards. These commands show the current state, the usage and the desired state (goal) of each card.

As the system is brought up from a cold start TEX initializes the clock card(s), then the switch card(s) and finally the processor cards. Initialization involves a sequence of states, to achieve parallelism. TEX can be loading the microboot into one node while others are executing the microboot, for example. Finally, after all nodes have either completed or failed their microboot and POST operations, the application file (typically BOOT.88) is loaded into the master (King) node and started.

Any cards which fail to initialize properly will be put into a "Sick/On" or "Sick/Off" state, though their usage will still be "System" (or whatever was specified by the BOOTCFG.TCS file).

The known defective card whose usage is "Off" will be put in the "Disabled/Off" state. All "Sick" or "Disabled" cards will be so marked in the configuration array that TEX passes to the application.

The operator and the operating system can each change the overall

system mode and the usage of individual cards to control TEX's actions. The operator does this by using the "Application Mode" and "Configuration Card-Use" commands. Of course, the operator can use any number of lower-level commands that can affect the operation of the system, disrupting TEX's operation. For example, a "Hardware Power Off" command will turn off a card immediately, regardless of the system mode and card usage. Also, changing a card's usage should not be done unless the operating system is capable of withstanding the change. A node being used by the operating system should not be switched to usage "Off" or "Isolate" without shutting down the operating system.

Power-on Servicing features

As described in the preceding section, TEX has two "use" values for cards to support maintenance and servicing. These are "Off" and "Isolate".

The system administrator, upon determining that a card appears to be defective, should mark that card's use as "Off". If the system configuration without that card is useful, it can then continue to be used until maintenance is scheduled.

The system administrator should mark the card as "Off" in the BOOTCFG.TCS file. This is most easily done by the following sequence (using abbreviations, of course, instead of the following fully-spelled examples):

1) Shut down the operating system (normally unavoidable when a bad card causes a system crash).

2) At the TEX prompt, enter "Configuration Forget".

3) Enter "Configuration Reconfigure". This reads a fresh copy of the BOOTCFG.TCS file.

4) Enter "Configuration Card-Use Off n.n.n" to disable the particular card, substituting its address for "n.n.n".

5) Enter "Configuration Write Boot-Configuration BOOTCFG.TCS" to overwrite the configuration file.

(Alternatively, an editor can be used to manually edit the file.)

Now the system can be restarted by entering "Application Mode Boot" and "Application Run". The system will come up with the defective node disabled.

When service personnel arrive, the defective node can be replaced and the new node tested by entering a "Configuration Card-Use Isolate" command. This will cause the card to be turned on but to have its switch path left off so that it cannot interrupt the operating system. The card can be tested by commands such as:

"Utility Slave n.n.n" (to select the card),
"Hardware Diagnostic 0" to run the TEX-based diagnostics,
"Memory Load \path\file.88" to load a diagnostic file,
"Hardware Go" and "Application TTY" to talk to that diagnostic, and so on.

When testing is done, the card should be halted and TEX should be switched back to normal operation by "Hardware Halt" (while the current slave is still the new card), then "Utility Slave 7.7.7" (or whichever node is the master) and "Application Run".

Finally, the new card can be brought into service at the next scheduled system shutdown by using the same procedure which marked the card as "Off" to mark it as "System". If the operating system can dynamically accept new processors, the card can be made available immediately by the "Configuration Card-Use System n.n.n" command. This is not yet possible

under the current TC2000 operating systems.]

It should be noted that while TEX is NOT running the system (i.e., from the typing of "`." to begin the maintenance work until the "Application Run" command at the end) TEX will not be performing console TTY activity, will not respond to service requests from the operating system and will not run the state machines for all the cards in the system. If this is done while the system is running and stable, the operating system will survive this but some console output is likely to be lost.

Configuration files

There are two configuration files which TEX will automatically read at startup if the -Autoboot switch is included on the DOS command line. These are SLOTCFG.TCS and BOOTCFG.TCS in the current directory. These files are described in the next two sections.

SLOTCFG.TCS

The slot configuration file, SLOTCFG.TCS, contains a description of the cards which should be found in the machine. Cards listed here will be included in the expansion of "*"-type wildcards as arguments to TEX commands. For example, the command "Hardware Temperature *.*.*" will report the temperature of all nodes which are listed in this file. This allows TEX to avoid polling cards which are not going to respond. Similarly, the "Application Run" command, which initializes all cards in the machine, expects to find the cards in this file.

The file contains one line for each card. The card's card address, type, serial number and revision levels are included. Discrepancies of serial numbers and the like are reported if TEX notices them. This file is written by TEX, by the "Configuration Write Slot-Configuration" command. It should not be edited manually.

When TEX is started with the -Autoboot switch, it will report missing or extra cards, and it will report cards whose types, serial numbers or revision levels are different from those in this file.

BOOTCFG.TCS

The bootstrap configuration file, BOOTCFG.TCS, gives specific parameters which guide the bootstrap process. Different versions of this file may be kept in different directories, such as for production operation or testing.

BOOTCFG.TCS tells whether to automatically bring up the operating system. It specifies cards which should be disabled, that is, which should not be initialized even if they are present. It specifies the preferred switch and clock in a redundant system. It specifies the microboot, POST and application bootstrap files. And it specifies the primary and secondary nodes for booting the operating system.

The specific commands allowed in BOOTCFG.TCS are described below. Each command consists of the command's name followed immediately by a colon, optional spaces, and the argument. Lower case and upper case are equivalent. For example,

"AlternatePaths: A: Yes"

AlternatePaths

The AlternatePaths command takes two arguments. The first is "A" or "B" and the second is "Yes" or "No". This informs TEX, and the initialization programs, whether alternate paths in the Butterfly switches should be used. The default is to use these paths. Some prototype systems, or defective systems, may run with these paths disabled.

Autoboot

The "autoboot" command takes one argument, a Y or an N. If "Y" is specified, TEX will automatically begin running the system after reading the configuration files. If "N" is specified, TEX will wait for commands from the console after reading the configuration files.

Bootfile

The "bootfile" command takes one argument, a filename for a file on the TCS master's disk. This file will be run on the king node after the microboot and POST have been run on all nodes.

BootParam

The "bootparam" command takes two arguments, both decimal numbers. The first number selects one of sixteen bootstrap parameter variables. The variable is set to the second number. The use of these bootstrap variables is described in the "services" section of this document. They are provided to support the operating system bootstrap.

BootString

The "bootstring" command takes the rest of the line (after the "bootstring:") as a string argument. TEX makes no use of this string; it simply holds it for use by the the application bootstrap. A typical string is

```
"(xy0,0,0)/vmunix"
```

CardUse

The "carduse" command takes two arguments, a card group address and a "use" definition. The "use" may be one of "system", "auxilliary", "isolated", or "off". See the "Configuration Card-Use" command for futther details.

DisabledCards

The "disabledcards" command takes one argument, a card address, possibly containing "!"-type wildcards. There may be multiple "disabledcards" commands in the BOOTCFG.TCS file. Each card named in a "disabledcards" command will be marked with "Use OFF" (See the "Configuration Card-Use" command), meaning that it will not be brought up automatically even though it may respond to TCS probes and may be present in the SLOTCFG.TCS file.

POSTFile

The "postfile" command takes one argument, a filename for a file on the TCS master's disk. This file will be run on each node which successfully completes the microboot.

PrimaryClock

The "primaryclock" command takes one argument, an A or a B. This specifies whether TC/CLK card A or B is to be used if both are present and working. [Not yet implemented.]

PrimaryNode

The "primarynode" command takes one argument, a specific card address. This specifies the node which will be used for the application bootstrap (bootfile command) if it is working. This node is also known as the "King" node or "Master" node.

PrimarySwitch

The "primaryswitch" command takes one argument, an A or a B. This specifies which butterfly switch is to be used if both are present and working. [Not yet implemented.]

SecondaryNode

The "secondarynode" command takes one argument, a specific card address. This specifies the node which will be used for the application bootstrap (bootfile command) if the primary node is not working. [Not yet implemented.]

SwitchColumns

The "switchcolumns" command takes one argument, a decimal number. This is the number of columns in the butterfly switch. It must be either two or three. The default, if this command is not present, is two columns.

SwitchFreq

The "switchfreq" command takes one argument, a decimal number. This is the frequency, in megaHertz, of the butterfly switch. It must be an even number in the range 16-50. The default, if this command is not present, is 38 MHz. (This range is for testing and does not imply that operation at all such speeds is supported or will work correctly.)

UBootFile

The "ubootfile" command takes one argument, a filename for a file on the TCS master's disk. This file will be run on each node which is mentioned in the SLOTCFG.TCS file and which is not disabled by a "disabledcards" command.

SERVICES

This section lists the services that the TCS Master provides to the operating system or diagnostic programs running on the BF2. It describes the commands to be given to the TCS master and describes the responses that will be generated.

[At this writing, services numbered 40 through 4D (those dealing with DOS files) have not been implemented.]

Service summary:

Code	Meaning
00	No command
10	Set desired state of component
11	Read current state of component
..	Shutdown system
21	Reboot system, counting panics
22	Reread configuration files
23	Get current boot string
24	Set boot string
25	Get bootstrap variable
26	Set bootstrap variable
27	Read TCS Master's calendar clock
28	Set TCS Master's calendar clock
29	Exit (return to TEX prompt)
2A	Warm Boot a node
2B	Cold Boot a node
40	Open DOS file for reading
41	Open DOS file for writing
42	Open DOS file for appending
43	Seek read file
44	Seek write file
45	Read
46	Write
47	Close read file
48	Close write file
49	Delete DOS file
4A	Mkdir
4B	Rmdir
4C	Connect to directory
4D	Get current directory
80/81	Read/write action register
82/83	Read/write EEPROM register
84/85	Read/write Gate-Array register
86/87	Read/write hardware register
88/89	Read/write T-Bus location

The following sections describe the services in more detail.

REQUEST PROTOCOL, ARGUMENTS AND RESULTS

The "code" for each command is a one-byte HEX number which the TCS reads from location to_tcs_cmd in page zero. A suggested mnemonic is shown.

The argument area and response area are described for each command. The arguments are read from physical memory pointed to by to_tcs_args in page zero. The arguments themselves may or may not be in page zero. Similarly, the results are returned into a physical memory area pointed to by from_tcs_response. The argument and response areas must each be word-aligned.

The results are valid when the TCS clears to_tcs_cmd. Immediately after clearing the command byte, the TCS will assert FROM_TCS_SRV_DONE in from_tcs_bits. If an error occurs in processing the request, FROM_TCS_SRV_ERR will also be set (simultaneously). The TCS will then cause an interprocessor interrupt if TO_TCS_SRVREQ_INT_EN is set in to_tcs_bits.

Values for the argument and response items are detailed at the end of this document.

BOOTSTRAP VARIABLES

The TCS maintains a block of sixteen 32-bit variables for use by the operating system and its bootstrap. These can be set from the BOOTCFG.TCS file and can be set and gotten by service requests.

These variables are meant as non-volatile memory over the process of booting or re-booting the operating system.

The TCS assigns meaning to variables number zero and one, the current panic count and panic limit. (See discussion under SVC_REBOOT.) The remaining variables are uninterpreted by the TCS.

The TCS also maintains one string, the "boot string", in the same way. This string is typically

"(xy0,0,0)/vmunix"

The current string will have been read from the BOOTCFG.TCS file or from a SVC_WRITE_BOOTSTRING command. The string may be up to 127 characters long and is null-terminated.

0E) FILESYSTEM SERVICES

[These services are not yet implemented.]

The TCS will provide the operating system with the ability to read one DOS file and simultaneously to write one DOS file. The TCS will not allow writes or appends to already-existing files with the DOS attributes "read-only", "system" or "hidden". Other than this restriction, the TCS does NOT protect itself from these file operations. The operating system can, if it wants, destroy anything on the TCS's disk. These functions should be used with due caution.

SERVICE FUNCTION DESCRIPTIONS

Code Name Description

00 SVC_NONE No command

This is the state of to_tcs_cmd when no request is pending.

10 SVC_SET_COMP_STATE
Set desired state of component.

This causes the TCS to begin changing the state of the component. Since some state changes take time, such as running the microboot, the final state will not be seen immediately.

Arguments are:
byte 0 COMPONENT CLASS
byte 1 Unused, for alignment
bytes 2-3 COMPONENT INDEX
byte 4 COMPONENT STATE

Results are:
None.

11 SVC_GET_COMP_STATE
Read current state of component.

Arguments are:
byte 0 COMPONENT CLASS
byte 1 Unused, for alignment
bytes 2-3 COMPONENT INDEX

Results are:
byte 0 Nonzero if the component has responded to a TCS message
byte 1 Nonzero if the component is "disabled" (Use: OFF) by the BOOTCFG.TCS file or other command
byte 2 Nonzero if the component is powered on
byte 3 Nonzero if the component is "sick"; e.g., failed to respond correctly to microboot or POST
byte 4 The current desired state of the component.
byte 5 Nonzero if the component has reached the desired state.

BOOTSTRAP SUPPORT FUNCTIONS:

20 SVC_SHUTDOWN
Shutdown system

The TCS will begin a shutdown procedure. All processors will be halted. If requested, all components will be powered down. The operating system should not make this request until its own shutdown process has been done and all I/O operations are completed. The TCS will acknowledge this request, giving a completion interrupt if requested. The actual halt will take place after a delay on the order of a few seconds.

Arguments are:
Byte 0 If 0, turn off power. If non-zero, leave power on after halting.

Results are:
None.

21 SVC_REBOOT
Reboot system, counting panics

The TCS will reset all components and reinitialize them, running the microboot and POST. It will then run the bootstrap and reload the system. The TCS will acknowledge this request, giving a completion interrupt if requested. The actual halt will take place after a delay on the order of a few seconds.

Before restarting the system, the TCS will increment bootstrap variable zero, the "panic count", and will compare it to bootstrap variable one, the "panic limit". If the count reaches the limit, the system will not be reloaded and the service request will be treated as a SVC_SHUTDOWN, with optional power shutdown.

Before making this request, the operating system should assure that the current boot string and other bootstrap variables have been passed to the TCS. The operating system should not make this request until it has finished its own shutdown process and all I/O operations have been completed.

Arguments are:
Byte 0 Meaningful only if the "panic limit" is reached. Then, if 0, turn off power. If non-zero, leave power on after halting.

Results are:
None.

22 SVC_REREAD_CONFIG
Reread configuration files

The TCS will reread the configuration files, BOOTCFG.TCS and SLOTCFG.TCS. This will override any parameter changes that may have been made since they were last read. The operating system may then choose to change a subset of the parameters and request a reboot.

Arguments are:
None.

Results are:
None.

23 SVC_GET_BOOTSTRING
Get current boot string

The TCS will write the current boot string into the result area. The string may be up to 127 characters long and is null-terminated.

Arguments are:
None.

Results are:
The string, starting at byte zero.

24 SVC_SET_BOOTSTRING
Set boot string

The TCS will read a string from the argument area and hold it as the current boot string. The string may be up to 127 characters long and is null-terminated. This string will generally have been read from BOOTCFG.TCS, but may be overridden by this function. This does NOT change the variable in the BOOTCFG.TCS file.

Arguments are:
The string, starting at byte zero.

Results are:
None.

25 SVC_GET_BOOTVAR
Get bootstrap variable

The TCS will copy the requested 32-bit bootstrap variable into the result area.

Arguments are:
Byte 0 The number of the desired variable, zero through fifteen.

Results are:
Bytes 0-3 The variable

26 SVC_SET_BOOTVAR
Set bootstrap variable

The TCS will read the variable from the argument area. This does NOT change the variable in the BOOTCFG.TCS file.

Arguments are:
Byte 0 The number of the desired variable, zero through fifteen.

Bytes 1-3 Ignored (for word alignment)
Bytes 4-7 The variable.

Results are:

None.

27 SVC_GET_CALENDAR
Get calendar/clock info from TCS master

The TCS reads the current date and time from DOS. This number will have been set by DOS at TCS startup time from its onboard battery-backed clock.

Arguments are:
None:

Results are:
Bytes 0-3 a 32-bit number, the number of seconds since 00:00:00 on January 1 1970, GMT.

28 SVC_SET_CALENDAR
Set TCS master's calendar/clock info

The TCS accepts a date and time from the argument area and places them into DOS and into the battery-backed clock.

Arguments are:
Bytes 0-3 a 32-bit number, the number of seconds since 00:00:00 on January 1 1970, GMT.

Results are:
None:

29 SVC_EXIT
Exit from application.

Arguments are:
Bytes 0-3 a 32-bit number

This service causes TEX to type "TCS: Exit." and then to return to its command prompt (as if the operator had typed "^."). It is provided as a less drastic halt than SVC_REBOOT or SVC_SHUTDOWN.

If the argument is non-zero, its value is also printed.

2A SVC_WARM_BOOT_NODE
Reboot the specified node.

This causes the TCS to halt the specified node, reload and start the microboot in its wait loop.

Arguments are:
byte 0 COMPONENT CLASS (Must be a procesor)
byte 1 Unused, for alignment
bytes 2-3 COMPONENT INDEX

Results are:
None.

This function will fail if the node cannot be accessed or if the microboot cannot be loaded. It does NOT wait for the microboot to enter its idle loop.

2B SVC_COLD_BOOT_NODE
Reset and reboot the specified node.

This causes the TCS to reset the specified node and restart the process of bringing the node to the its currently selected component state.

Arguments are:
byte 0 COMPONENT CLASS (Must be a procesor)
byte 1 Unused, for alignment
bytes 2-3 COMPONENT INDEX

Results are:
None.

This function will fail if the node cannot be accessed.

DOS FILE FUNCTIONS:

In the following functions, the pathname and filename strings must be less than 128 characters, and are terminated by a NULL.

40 SVC_DOS_OPENR
Open DOS file for reading

The TCS will attempt to open a file for reading. Only one file may be open for reading at one time.

Arguments are:
Byte 0-n The filename string

Results are:
Byte 0 Zero for success, non-zero for various errors conditions (values to be specified)

41 SVC_DOS_OPENW
42 SVC_DOS_OPENA
Open DOS file for writing (or appending)

The TCS will attempt to open a file for writing. Only one file may be open for writing at one time. If the ...OPENA call is used, an append will be done if the file already exists.

Arguments are:
Byte 0-n The filename string

Results are:
Byte 0 Zero for success, non-zero for various errors conditions (values to be specified)

43 SVC_DOS_SEEKR
44 SVC_DOS_SEEKW
Seek (read or write) file

The TCS will perform a seek operation on the specified file.

Arguments are:
Bytes 0-3 The file position (byte number). -1 means end of file.

45 SVC_DOS_READ
Read from file

The TCS will read bytes from the currently open "read" file. The file is read in binary; the string is neither null-terminated nor broken at an EOL. No conversion is done between Unix newlines and DOS CR/LF pairs.

Arguments are:
Bytes 0-3 Maximum number of bytes to read. This number must be no more than 128.

Results are:
Bytes 0-3 Number of bytes read. May be smaller than requested if the file reaches EOF. A negative number means an error (file not open, disk error). A zero means that the file is at EOF.

Bytes 4-n The data from the file.

46 SVC_DOS_WRITE
Write to file

The TCS will write bytes into the currently open "write" file. The file is written in binary. NULLs are not treated specially. No conversion is done between Unix newlines and DOS CR/LF pairs.

Arguments are:
Bytes 0-3 Maximum number of bytes to write. This number must be no more than 128.
Bytes 4-n The data to be written.

Results are:
Bytes 0-3 Number of bytes written. May be smaller than requested if the output device fills up. A negative number means an error (file not open, disk error).

47 SVC_DOS_CLOSER
48 SVC_DOS_CLOSEW
Close (read or write) file

The TCS will close the specified file.

Arguments are:
None.

Results are:
Byte 0 Zero for success, non-zero if DOS reports an error closing the file.

49 SVC_DOS_DELETE
Delete file

The TCS will delete the specified file from the DOS filesystem.

Arguments are:
Bytes 0-n The filename string, null terminated

Results are:
Byte 0 Zero for success, non-zero if DOS reports an error.

4A SVC_DOS_MKDIR
Make a directory

The TCS will create the specified directory in the DOS filesystem.

Arguments are:

Bytes 0-n The pathname string, null terminated

Results are:
Byte 0 Zero for success, non-zero if DOS reports an error.

4B SVC_DOS_RMDIR
Remove a directory

The TCS will delete the specified directory from the DOS filesystem.

Arguments are:
Bytes 0-n Th pathname string, null terminated

Results are:
Byte 0 Zero for success, non-zero if DOS reports an error.

4C SVC_DOS_GETCWD
Get current directory

The TCS will report the current directory from the DOS filesystem.

Note that a service to CHANGE the current directory is NOT provided, to avoid changing to different default configuration files.

Arguments are:
Byte 0 Number of available bytes for the string

Results are:
Bytes 0-n The pathname string, terminated by a NULL. DOS specifies the length to be a maximum of 67 characters including the NULL. The first three characters are the drive letter, a colon and a backslash.

TCS SLAVE REQUESTS

The following functions pass requests directly to the TCS slave processors on the various system cards. They should be used with caution, after understanding the relevant TCS slave programming specifications.

80 SVC_TCS_RD_ACTION

The TCS sends an ACTION REGISTER read request to the specified TCS slave.

Arguments are:
byte 0 COMPONENT CLASS
byte 1 Unused, for alignment
bytes 2-3 COMPONENT INDEX
Byte 4 Action register number

Results are:
Byte 0 Zero for success, non-zero for error (no response or NAK)
Byte 1 Returned data, if successful

81 SVC_TCS_WR_ACTION

The TCS sends an ACTION REGISTER write request to the specified TCS slave.

Arguments are:
byte 0 COMPONENT CLASS
byte 1 Unused, for alignment
bytes 2-3 COMPONENT INDEX
Byte 4 Action register number
Byte 5 Value to be written to the register

Results are:
Byte 0 Zero for success, non-zero for error (no response or NAK)

82 SVC_TCS_RD_EEPROM

The TCS sends an EEPROM read request to the specified TCS slave.

Arguments are:
byte 0 COMPONENT CLASS
byte 1 Unused, for alignment
bytes 2-3 COMPONENT INDEX
Byte 4 EEPROM byte address

Results are:
Byte 0 Zero for success, non-zero for error (no response or NAK)
Byte 1 Returned data, if successful

83 SVC_TCS_WR_EEPROM

The TCS sends an EEPROM write request to the specified TCS slave.

Arguments are:
byte 0 COMPONENT CLASS
byte 1 Unused, for alignment

```
                bytes 2-3   COMPONENT INDEX
                Byte 4      EEPROM byte address
                Byte 5      Value to be written to the EEPROM

            Results are:
                Byte 0      Zero for success, non-zero for
                            error (no response or NAK)
```

84 SVC_TCS_RD_GATEARRAY

The TCS sends a Gate Array register read request to the specified TCS slave.

```
            Arguments are:
                byte 0      COMPONENT CLASS
                byte 1      Unused, for alignment
                bytes 2-3   COMPONENT INDEX
                Byte 4      Gate Array number (Command modifier)
                Byte 5      Gate Array register number

            Results are:
                Byte 0      Zero for success, non-zero for
                            error (no response or NAK)
                Byte 1      Returned data, if successful
```

85 SVC_TCS_WR_GATEARRAY

The TCS sends a Gate Array write request to the specified TCS slave.

```
            Arguments are:
                byte 0      COMPONENT CLASS
                byte 1      Unused, for alignment
                bytes 2-3   COMPONENT INDEX
                Byte 4      Gate Array number (Command modifier)
                Byte 5      Gate Array register number
                Byte 6      Value to be written to the register

            Results are:
                Byte 0      Zero for success, non-zero for
                            error (no response or NAK)
```

86 SVC_TCS_RD_HWARE

The TCS sends a HARDWARE register read request to the specified TCS slave.

```
            Arguments are:
                byte 0      COMPONENT CLASS
                byte 1      Unused, for alignment
                bytes 2-3   COMPONENT INDEX
                Byte 4      Hardware register number

            Results are:
                Byte 0      Zero for success, non-zero for
                            error (no response or NAK)
                Byte 1      Returned data, if successful
```

87 SVC_TCS_WR_HWARE

The TCS sends a HARDWARE register write request to the specified TCS slave.

```
            Arguments are:
                byte 0      COMPONENT CLASS
                byte 1      Unused, for alignment
                bytes 2-3   COMPONENT INDEX
                Byte 4      Hardware register number
                Byte 5      Value to be written to the register

            Results are:
                Byte 0      Zero for success, non-zero for
                            error (no response or NAK)
```

88 SVC_TBUS_RD
T-Bus read cycle

The TCS sends a TCS T-Bus memory setup request and a T-Bus memory read request to the specified TCS slave. The requestor MUST correctly format all bytes of this message or the remote T-Bus may be hung. This command allows complete freedom in setting up the request, and does no validity checking.

```
            Arguments are:
                byte 0      COMPONENT CLASS
                byte 1      Unused, for alignment
                bytes 2-3   COMPONENT INDEX
                Byte 4      Memory setup CMD MOD byte
                            (SIGA A/B selector)
                Byte 5      Memory setup TBus Command
                byte 6      Memory Setup TBus CMOD1
                byte 7      Memory Setup TBus CMOD0
                byte 8      TBus Read CMD MOD (increment)
                byte 9-11   unused, for alignment
                bytes 12-15 TBus address bits 31-0

            Results are:
                Byte 0      Nack or Ack code from slave
                Byte 1      TBus response byte
                Bytes 2-3   unused, for alignment
                Bytes 4-7   Data from TBus cycle
```

89 SVC_TBUS_WR
T-Bus write cycle

The TCS sends a TCS T-Bus memory setup request and a T-Bus memory write request to the specified TCS slave. The requestor MUST correctly format all bytes of this message or the remote T-Bus may be hung. This command allows complete freedom in setting up the request, and does no validity checking.

```
            Arguments are:
                byte 0      COMPONENT CLASS
                byte 1      Unused, for alignment
                bytes 2-3   COMPONENT INDEX
                Byte 4      Memory setup CMD MOD byte
                            (SIGA A/B selector)
                Byte 5      Memory setup TBus Command
                byte 6      Memory Setup TBus CMOD1
                byte 7      Memory Setup TBus CMOD0
                byte 8      TBus Write CMD MOD (increment)
                byte 9-11   unused, for alignment
                bytes 12-15 TBus address bits 31-0
                bytes 16-19 TBus write data

            Results are:
```

Byte 0 Nack or Ack code from slave
Byte 1 TBus response byte

Values of argument and response bytes:

COMPONENT CLASS

Note that these numbers are not simply hardware card types but are "functional" classes. These correspond more closely to locations within the system than to types of hardware.

00 TC/CLK clock card
01 TC/SS in Switch A
02 TC/SR in Switch A
03 TC/SS in Switch B
04 TC/SR in Switch B
05 TC/FP, TC/FPV or other processor node

COMPONENT INDEX

This is the index of a particular component class.
For clocks, the index ranges from 0 through 1.
For switch cards, the index ranges from 0 through 63.
For processor nodes, the index ranges from 0 through 511.

COMPONENT STATE

This argument is used in the SVC SET COMP STATE request. This is an overall state of a component. It hides many other more detailed states which may change with versions of the software. The states are analogous to those specified by the "NodeUse" line in the BOOTCFG.TCS file and the "Configuration Card-Use" command.

The operating system may request that the component be placed in one of the following four states:

01 SVC_CS_SYS System usage

02 SVC_CS_AUX Auxiliary usage

03 SVC_CS_ISOLATE Isolated usage

04 SVC_CS_OFF Off, or disabled, usage

----------end of tex.doc----------