1

TEX Service Calls

1.1 Introduction

This document lists the services (SVCs) that the TCS Master provides to the operating system or diagnostic programs running on the TC2000. It describes the commands to be given and describes the responses that will be generated.

1.2 SVC Request Protocol

There is a one-byte HEX number or "code" associated with each SVC command which the TCS Master reads from location ***to_tcs_cmd*** in page zero.

The argument and response areas reside in the node's physical memory. The arguments are read from physical memory pointed to by ***to_tcs_args*** located in page zero. The argument area may or may not be located in page zero. Similarly, the results are returned into a physical memory area pointed to by ***from_tcs_response***. The argument and response areas most both be word-aligned.

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

An SVC command summary along with the format and values for each SVC argument and response area are detailed in later sections.

1.3 BootStrap Variables

The TCS Master maintains a block of sixteen 32-bit variables for use by the operating system and it's bootstrap. These can be set from the "BOOTCFG.TCS" file, and can be set and retrieved via SVC requests.

These variables are designed to be stored in non-volatile memory during the process of booting or re-booting the operating system.

The TCS Master assigns meaning to the first two "bootstrap" variables (zero and one). Bootstrap variable[0] represents the current panic count, while variable[1] represents the panic limit. The remaining fourteen variables are uninterpreted by the TCS Master.

The TCS Master also maintains one variable string, the "boot string", in the same way. This string is typically "(sd1,0,0)/vmunix". The current boot string will have been read from the "BOOTCFG.TCS" file or from an **SVC_SET_BOOTSTRING** SVC command. The boot string may be up to 127 characterslong and is NULL terminated.

1.4 SVC Commands

This section provides a summary of each SVC command by "code" number, followed by it's category and a brief description. The SVCs are broken up into six categories:

1. Card State Support SVC Functions
2. Sdriver Support SVC Functions
3. BootStrap Support SVC Functions
4. Diagnostic Support SVC Functions
5. DOS File Support SVC Functions
6. TCS Slave Request SVC Functions

1.4.1 SVC Command Summary List

Code	Mneumonic	Category	Description
00	SVC_NONE		No Command
10	SVC_SET_COMP_STATE	Card State	Set desired state of card
11	SVC_GET_COMP_STATE	Card State	Read current state of card
12	SVC_SET_TTY_XOFF	Sdriver	Set Sdriver tty_off parameter
13	SVC_GET_TTY_XOFF	Sdriver	Read current state of tty_off
14	SVC_SET_DIALUP_ENA	Sdriver	Set Modem dialup enable parameter
15	SVC_GET_DIALUP_ENA	Sdriver	Read modem dialup enable state

16	SVC_SET_DIALUP_PWD	Sdriver	Set modem password
17	SVC_GET_DIALUP_PWD	Sdriver	Read Modem password
18	SVC_SET_DIALUP_INI	Sdriver	Set modem init string
19	SVC_GET_DIALUP_INI	Sdriver	Read modem init string
1A	SVC_GET_TEX_VER		Read TEX version number
1B	SVC_GET_SDRIVER_VER	Sdriver	Read Sdriver version number
1C	SVC_GET_DIALUP_LOGGED	Sdriver	Read modem login flag
20	SVC_SHUTDOWN	BootStrap	Shutdown system
21	SVC_REBOOT	BootStrap	Reboot system, counting panics
22	SVC_REREAD_CONFIG	BootStrap	Reread configuration files
23	SVC_GET_BOOTSTRING	Bootstrap	Read current boot string
24	SVC_SET_BOOTSTRING	Bootstrap	Set boot string
25	SVC_GET_BOOTVAR	Bootstrap	Read bootstrap variable
26	SVC_SET_BOOTVAR	Bootstrap	Set bootstrap variable
27	SVC_GET_CALENDAR	Bootstrap	Read TCS Master's calendar clock
28	SVC_SET_CALENDAR	Bootstrap	Set TCS Master's calendar clock
29	SVC_EXIT	BootStrap	Exit (Return to TEX prompt)
2A	SVC_WARM_BOOT_NODE	BootStrap	Warm boot a processor node
2B	SVC_COLD_BOOT_NODE	BootStrap	Cold boot a processor node
2C	SVC_START_NODE	BootsStrap	Reset start a processor node
2D	SVC_CONFIG_ARRAY	BootStrap	Load configuration table into a node
2E	SVC_TYPESCRIPT	BootStrap	Save console-log to "ttyhalt.tes" file
30	SVC_DO_UBOOT_NODE	Diagnostic	Download UBOOT into a node
31	SVC_DO_POST_NODE	Diagnostic	Download POST into a node
32	SVC_KILL_NODE	Diagnostic	Halt and power down a node
33	SVC_JUMP_APP_NODE	Diagnostic	Change slave node, do app tty
45	SVC_DOS_READ	DOS	Perform a read on a DOS file
46	SVC_DOS_WRITE	DOS	Perform a write on a DOS file
49	SVC_DOS_DELETE	DOS	Delete a DOS file
4A	SVC_DOS_MKDIR	DOS	Perform a mkdir on DOS file system
4B	SVC_DOS_RMDIR	DOS	Perform a rmdir on DOS filke system
4C	SVC_DOS_GETCWD	DOS	Read current directory
4D	SVC_DOS_OPEN	DOS	Open a DOS file
4E	SVC_DOS_CLOSE	DOS	Close a DOS file
4F	SVC_DOS_SEEK	DOS	Seek on a DOS file
50	SVC_DOS_READLN	DOS	Read a text line from a DOS file
80	SVC_TCS_RD_ACTION	TCS Slave	Read Action Register
81	SVC_TCS_WR_ACTION	TCS Slave	Write Action Register
82	SVC_TCS_RD_EEPROM	TCS Slave	Read EE PROM Register
83	SVC_TCS_WR_EEPROM	TCS Slave	Write EE PROM Register
84	SVC_TCS_RD_GATEARRAY	TCS Slave	Read Gatearry Register
85	SVC_TCS_WR_GATEARRAY	TCS Slave	Write Gatearray Register
86	SVC_TCS_RD_HWARE	TCS Slave	Read Hardware Register
87	SVC_TCS_WR_HWARE	TCS Slave	Write Hardware Register
88	SVC_TCS_TBUS_RD	TCS Slave	Perform Tbus read-cycle
89	SVC_TCS_TBUS_WR	TCS Slave	Perform Tbus write-cycle

See next page

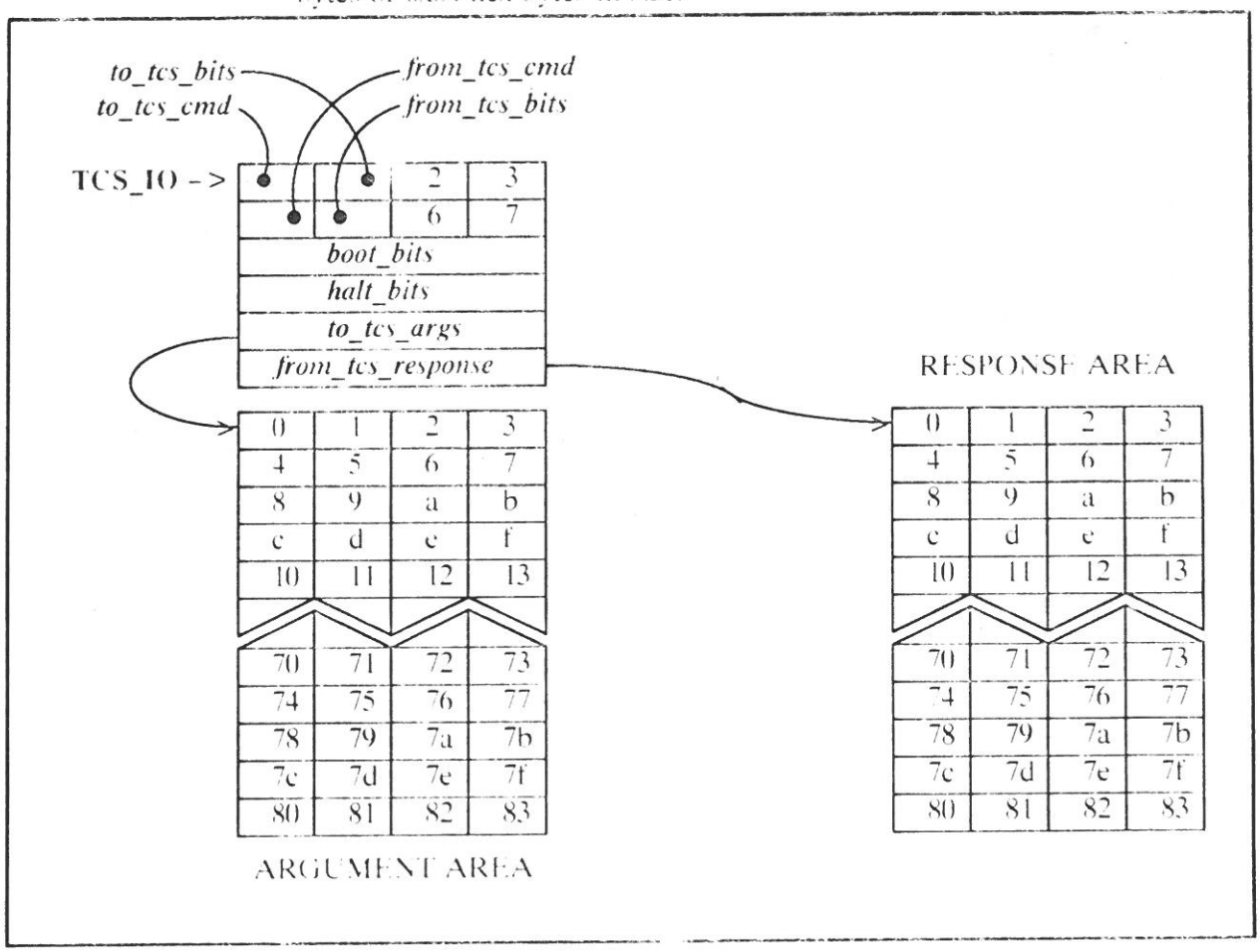
16	SVC_SET_DIALUP_PWD	Sdriver	Set modem password
17	SVC_GET_DIALUP_PWD	Sdriver	Read Modem password
18	SVC_SET_DIALUP_INI	Sdriver	Set modem init string
19	SVC_GET_DIALUP_INI	Sdriver	Read modem init string
1A	SVC_GET_TEX_VER		Read TEX version number
1B	SVC_GET_SDRIVER_VER	Sdriver	Read Sdriver version number
1C	SVC_GET_DIALUP_LOGGED	Sdriver	Read modem login flag
20	SVC_SHUTDOWN	BootStrap	Shutdown system
21	SVC_REBOOT	BootStrap	Reboot system, counting panics
22	SVC_REREAD_CONFIG	BootStrap	Reread configuration files
23	SVC_GET_BOOTSTRING	Bootstrap	Read current boot string
24	SVC_SET_BOOTSTRING	Bootstrap	Set boot string
25	SVC_GET_BOOTVAR	Bootstrap	Read bootstrap variable
26	SVC_SET_BOOTVAR	Bootstrap	Set bootstrap variable
27	SVC_GET_CALENDAR	Bootstrap	Read TCS Master's calendar clock
28	SVC_SET_CALENDAR	Bootstrap	Set TCS Master's calendar clock
29	SVC_EXIT	BootStrap	Exit (Return to TEX prompt)
2A	SVC_WARM_BOOT_NODE	BootStrap	Warm boot a processor node
2B	SVC_COLD_BOOT_NODE	BootStrap	Cold boot a processor node
2C	SVC_START_NODE	BootsStrap	Reset start a processor node
2D	SVC_CONFIG_ARRAY	BootStrap	Load configuration table into a node
2F	SVC_TYPESCRIPT	BootStrap	Save console-log to "ttyhalt.tes" file
2F	SVC_REN_SCRIPT	BootStrap	Save and rename console-log file
30	SVC_DO_UBOOT_NODE	Diagnostic	Download UBOOT into a node
31	SVC_DO_POST_NODE	Diagnostic	Download POST into a node
32	SVC_KILL_NODE	Diagnostic	Halt and power down a node
33	SVC_JUMP_APP_NODE	Diagnostic	Change slave node, do app tty
45	SVC_DOS_READ	DOS	Perform a read on a DOS file
46	SVC_DOS_WRITE	DOS	Perform a write on a DOS file
49	SVC_DOS_DELETE	DOS	Delete a DOS file
4A	SVC_DOS_MKDIR	DOS	Perform a mkdir on DOS file system
4B	SVC_DOS_RMDIR	DOS	Perform a rmdir on DOS filke system
4C	SVC_DOS_GETCWD	DOS	Read current directory
4D	SVC_DOS_OPEN	DOS	Open a DOS file
4E	SVC_DOS_CLOSE	DOS	Close a DOS file
4F	SVC_DOS_SEEK	DOS	Seek on a DOS file
50	SVC_DOS_READLN	DOS	Read a text line from a DOS file
80	SVC_TCS_RD_ACTION	TCS Slave	Read Action Register
81	SVC_TCS_WR_ACTION	TCS Slave	Write Action Register
82	SVC_TCS_RD_EEPROM	TCS Slave	Read EEPROM Register
83	SVC_TCS_WR_EEPROM	TCS Slave	Write EEPROM Register
84	SVC_TCS_RD_GATEARRAY	TCS Slave	Read Gatearry Register
85	SVC_TCS_WR_GATEARRAY	TCS Slave	Write Gatearray Register
86	SVC_TCS_RD_HWARE	TCS Slave	Read Hardware Register
87	SVC_TCS_WR_HWARE	TCS Slave	Write Hardware Register
88	SVC_TCS_TBUS_RD	TCS Slave	Perform Tbus read-cycle
89	SVC_TCS_TBUS_WR	TCS Slave	Perform Tbus write-cycle

1.5 SVC Command Descriptions

This section provides a detailed description of each SVC command. The argument and response data areas for each command are described for use by the diagnostic and operating system programmers. The SVC command descriptions are described in "code" number order, as found in the SVC summary list.

1.5.1 Argument and Response Data Areas

SVCs that require arguments retrieve those arguments from the argument data area, which is pointed by ***to_tcs_args*** page zero location. Not all the SVCs have the same argument format. The same holds true for SVC results. The SVC results are stored in the response data area pointed to by ***from_tcs_response*** located in page zero. The result data format may or may not be the same for each SVC. Both the argument and response data areas can be at most 132 bytes or 0x84 hex bytes in size.



1.5.2 SVC Function Descriptions

00 SVC_NONE

This is the state of *to_tcs_cmd* (page zero) when no SVC request is pending.

10 SVC_SET_COMP_STATE

Set the desired state for the selected card.

This command causes the TCS Master to begin changing the state of the selected card. Since some state changes take time, such as running microboot or POST, the final desired state may not be achieved immediately.

Arguments:

byte 0	Card Class/Type
byte 1	Unused, for alignment
bytes 2–3	Card Index
byte 4	Card State

Response:

NONE

11 SVC_GET_COMP_STATE

Read the current state for the selected card.

Arguments:

byte 0	Card Class/Type
byte 1	Unused, for alignment
byte 2–3	Card Index

Response:

byte 0	Non-zero if the card has responded to a TCS message
byte 1	Non-zero if the card is "disabled" by the BOOTCFG.TCS file or other TCS command
byte 2	Non-zero if the card is powered on
byte 3	Non-zero if the card is "sick"; e.g., failed to respond correctly to microboot or POST
byte 4	The desired state of the card
byte 5	Non-zero if the card has reached the desired state

1.5.2 SVC Function Descriptions

00 SVC_NONE

This is the state of *to_tcs_cmd* (page zero) when no SVC request is pending.

01 SVC_SET_COMP_STATE

Set the desired state for the selected card.

This command causes the TCS Master to begin changing the state of the selected card. Since some state changes take time, such as running microboot or POST, the final desired state may not be achieved immediately.

Arguments:

byte 0	Card Class/Type
byte 1	Unused, for alignment
bytes 2–3	Card Index
byte 4	Card State

Response:

NONE

02 SVC_GET_COMP_STATE

Read the current state for the selected card.

Arguments:

byte 0	Card Class/Type
byte 1	Unused, for alignment
byte 2–3	Card Index

Response:

byte 0	Non-zero if the card has responded to a TCS message
byte 1	Non-zero if the card is "disabled" by the BOOTCFG.TCS file or other TCS command
byte 2	Non-zero if the card is powered on
byte 3	Non-zero if the card is "sick"; e.g., failed to respond correctly to microboot or POST
byte 4	The desired state of the card
byte 5	Non-zero if the card has reached the desired state

12 SVC_SET_TTY_XOFF

Enable or disabled flow control on TCS console device.

Arguments:
bytes 0–3 Non-zero value for enabling flow control

Response:
NONE

13 SVC_GET_TTY_XOFF

Read the current state of flow control on TCS console device.

Arguments:
NONE

Response:
bytes 0–3 Non-zero value if flow control is enabled

14 SVC_SET_DIALUP_ENA

Enable or disable dialup access to the TCS modem.

Arguments:
bytes 0–3 Non-zero value for enabling dialup access

Response:
NONE

15 SVC_GET_DIALUP_ENA

Read the current state of dialup access for the TCS modem.

Arguments:
NONE

Response:
bytes 0–3 Non-zero value if dialup access is enabled

16 SVC_SET_DIALUP_PWD

Set the TCS modem access password.

The TCS Master will read the dialup password string from the argument area. The string may be up to eight characters long and null-terminated.

Arguments:

bytes 0-8 Up to eight character password string terminated with a NULL character

Response:

NONE

17 SVC_GET_DIALUP_PWD

Read the current TCS modem access password.

The TCS Master will write the current dialup password string into the response area. The string may be up to eight characters long and null-terminated.

Arguments:

NONE

Response:

bytes 0-8 Up to an eight character password string terminated with a NULL character

18 SVC_SET_DIALUP_INI

Set the TCS modem initialization string.

The TCS Master will read the dialup initialization string from the argument area. The string may be up to 64 characters long and null-terminated.

Arguments:

bytes 0-64 Up to a 64 character initialization string terminated with a NULL character

Response:

NONE

18 SVC_GET_DIALUP_INI

Read the current TCS modem initialization string.

The TCS Master will write the current modem initialization string into the response area. The string may be up to 64 characters long and null-terminated.

Arguments:

NONE

Response:

bytes 0–64 Up to a 64 character initialization string terminated with a NULL character

1A SVC_GET_TEX_VER

Read the TEX version string.

The TCS Master will write the current TEX version string into the response area. The string may be up to 127 characters long and null-terminated.

Arguments:

NONE

Response:

bytes 0–127 Up to a 127 character TEX version string terminated twith a NULL character

1B SVC_GET_SDRIVER_VER

Read the Sdriver version string.

The TCS Master will write the current Sdriver version string into the response area. The string may be up to 79 characters long and null-terminated.

Arguments:

NONE

Response:

bytes 0–79 Up to a 79 character Sdriver version string terminated with a NULL character

1C SVC_GET_DIALUP_LOGGED

Read the state of TCS modem login flag.

Arguments:

NONE

Response:

bytes 0–3 Non-zero value if TCS modem is currently being used

20 SVC_SHUTDOWN

Shutdown the TC2000 system

The TCS Master will begin a shutdown procedure. All processors will be halted. If requested, all cards 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 complete. The TCS Master 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:

byte 0 If zero, halt and power down all processors. If non-zero, leave all processors on after halting.

Response:

NONE

21 SVC_REBOOT

Reboot the TC2000 system, counting panics

The TCS Master will reset all cards and then re-initialize them. Running microboot and POST on all processor nodes. It will then run the bootstrap and reload the system. The TCS Master 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 Master will increment the bootstrap variable[0], the panic count, and will compare it to the bootstrap variable[1], the panic limit. If the panic count reaches the panic limit, the system will not be reloaded and the service request will be treated as a **SVC_SHUTDOWN**, with the 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 Master. The operating system should not make the request until it has finished its own shutdown procedures and all I/O operations have been completed.

Arguments:

byte 0 — Meaningful only if the panic count has reached the panic limit. Then, If zero, halt and power down all processors. If non-zero leave all processors on after halting.

Response:

NONE

22 SVC_REREAD_CONFIG

Reread the TCS system configuration files.

The TCS Master will reread the configuration files, BOOTCFG.TCS and SLOTCFG.TCS. This will override any parameter changes that may 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:

NONE

Response:

NONE

23 SVC_GET_BOOTSTRING

Read the current boot string.

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

Arguments:

NONE

Response:

bytes 0-127 Up to a 127 character boot string terminated with a NULL character

24 SVC_SET_BOOTSTRING

Set the boot string.

The TCS Master will read a string of characters from the argument area and store it as the current noot string. The string may be upt to 127 characters long and null-terminated. This string will generally have been read from the BOOTCFG.TCS file, but may be overwridden by this function. This does NOT change the boot string contained in the BOOTCFG.TCS file.

Arguments:

bytes 0-127 Up to a 127 character initialization string terminated with a NULL character

Response:

NONE

25 SVC_GET_BOOTVAR

Read the selected bootstrap variable.

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

Arguments:

byte 0 The index of the desired bootstrap variable[15..0].

Response:

bytes 0-3 The 32-bit value of that bootstrap variable.

26 SVC_SET_BOOTVAR

Set the selected bootstrap variable

The TCS Master will read the 32-bit variable value from the argument area, and store it in the selected bootstrap variable. This does not change the bootstrap variable information contained in the BOOTCFG.TCS file.

Arguments:

byte 0	The index of the desired bootstrap variable[15..0].
bytes 1–3	Unused, for alignment
bytes 4–7	The 32-bit bootstrap variable value

Response:

NONE

27 SVC_GET_CALENDAR

Read the calendar/clock information from the TCS Master

The TCS Master reads the current date and time from DOS. This number is a 32-bit value that represents the number of seconds since 00:00:00 on January 1 1970, GMT. This number should have previously been set by DOS at TCS power-up time from its onboard battery-backed clock.

Arguments:

NONE

Response:

bytes 0–3	The 32-bit time representation

28 SVC_SET_CALENDAR

Set the TCS Master's calendar/clock information

The TCS Master reads the current date and time from the argument area. This number is a 32-bit value that represents the number of seconds since 00:00:00 on January 1 1970, GMT. This number is then stored in DOS and the battery-backed clock.

Arguments:

bytes 0–3	The 32-bit time representation

Response:

NONE

29 SVC_EXIT

Exit from application

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

Arguments:

byte 0–3	If this 32-bit value is non-zero, then its value will also be printed

Response:

NONE

2A SVC_WARM_BOOT_NODE

Reboot the specified processor node

This function causes the TCS Master to halt the specified node, reload and start microboot, leaving it in its wait loop. This function will fail if the node cannot be accessed or if the microboot image cannot be reloaded. It does NOT wait for the microboot program to enter its idle/wait loop.

Arguments:

byte 0	Card Class/Type (must be a processor)
byte 1	Unused, for alignment
bytes 2–3	Card Index

Response:

NONE

2B SVC_COLD_BOOT_NODE

Reset and reboot the specified processor node

This function will cause the TCS Master to reset the specified node and restart the process of bringing that node to its current desired state. This function will fail if the processor node cannot be accessed.

Arguments:

byte 0	Card Class/Type (must be a processor)
byte 1	Unused, for alignment
bytes 2–3	Card Index

Response:

NONE

2C SVC_START_NODE

Perform a reset start on the selected processor node

This function will cause the TCS Master to halt the specified processor node, and if requested, re-initialize the node. Then the TCS Master will load the proper branch instruction at address zero of the node. This branch instruction is based on the desired start address read from the argument area. The TCS Master will then issue the reset to the node.

Arguments:

byte 0	Card Class/Type (must be a processor)
byte 1	Unused, for alignment
bytes 2-3	Card Index
bytes 4-7	Start Address
byte 8	If non-zero perform node initialization after halt

Response:

NONE

2D SVC_CONFIG_ARRAY

Write the system configuration array into the specified node

This function causes the TCS Master to write out the node type and disable information into a predetermined area of the selected nodes main memory. The system configuration array contains node type and disable information, that both the diagnostics and the operating system use to quickly determine what nodes are present and usable within the system.

Arguments:

byte 0	Card Class/Type (must be a processor)
byte 1	Unused, for alignment
bytes 2-3	Card Index

Response:

NONE

2E SVC_TYPESCRIPT

Append the console-log to the "ttyhalt.tes" file located in the current working directory.

Arguments:

NONE

Response:

NONE

2F SVC_REN_SCRIPT

Append the console-log to the "ttyhalt.tes" file located in the current working directory, and then rename that file to the file name specified in the argument area.

Arguments: None
~~bytes 0-127 String of up to 128 characters containing the new file name~~

Response:
byte 0 Zero for success, otherwise a DOS **errno** value

30 SVC_DO_UBOOT_NODE

Load and start microboot on specified node

This function causes the TCS Master to re-initialize the specified node, then loads and starts microboot on it. The current state and desired state for this node is not changed. This function is to be used solely in a diagnostic environment. This function does NOT verify that microboot successfully completed.

Arguments:
byte 0 Card Class/Type (must be a processor)
byte 1 Unused, for alignment
bytes 2-3 Card Index

Response:
NONE

31 SVC_DO_POST_NODE

Load and start POST on specified node

This function causes the TCS Master to halts the specified node, then loads and starts POST on it. The current state and desired state for this node is not changed. This function is to be used solely in a diagnostic environment. This function does NOT verify that POST successfully completed.

Arguments:
byte 0 Card Class Type (must be a processor)
byte 1 Unused, for alignment
bytes 2-3 Card Index

Response:
NONE

32 SVC_KILL_NODE

Halt and power-down a specified node

This function causes the TCS Master to halt and power-down a specified node. The current state and desired state for this node is changed to UNKNOWN. This function is to be used solely in a diagnostic environment.

Arguments:

byte 0	Card Class/Type (must be a processor)
byte 1	Unused, for alignment
bytes 2-3	Card Index

Response:

NONE

33 SVC_JUMP_APP_NODE

Get console output from specified node

This function causes the TCS Master to perform a "utility slave" command for the specified node, and then an "application tty" command. Thus getting console from that node. This function is to be used solely in a diagnostic environment.

This function must be used with caution. If the specified node is NOT running an application, it will appear as if the TEX is hung. Simply type "^." to return to the TEX command prompt, if this is the case.

Arguments:

byte 0	Card Class/Type (must be a processor)
byte 1	Unused, for alignment
bytes 2-3	Card Index

Response:

NONE

45 SVC_DOS_READ

Read the specified number of bytes from a file with the specified file handle.

This function causes the TCS Master to perform a DOS read on a file. The maximum number of bytes allowed to be read is 128. The file handle and number of bytes to be read are read from the argument area. The number of bytes actually read, and the bytes themselves, are written into the response area. If the bytes actually read is zero, an End-of-File was reached. If the bytes actually read is equal to minus one (-1) an error has occurred, and the DOS **errno** value is written into response area.

Arguments:

bytes 0-3	File handle (previously returned from **SVC_DOS_OPEN**)
bytes 4-7	Number of bytes to be read

Response:

bytes 0-3	Number of bytes actually read. If zero, indicates end-of-file reached, if minus one (-1) indicates an error.
byte 4	If an error occurred, this is the DOS **errno** value.
bytes 5-7	Unused, for alignment
bytes 8-131	Up to 128 bytes of data (if no errors occurred)

46 SVC_DOS_WRITE

Write the specified number of bytes into a file with the specified file handle.

This function causes the TCS Master to perform a DOS write on a file. The maximum number of bytes allowed to be written is 128. The file handle, number of bytes to be written, and the bytes themselves are read from the argument area. The number of bytes actually written are written into the response area. If the bytes actually written is equal to minus one (-1) an error has occurred, and the DOS **errno** value is written into response area.

Arguments:

bytes 0-3	File handle (previously returned from **SVC_DOS_OPEN**)
bytes 4-7	Number of bytes to be written
bytes 8-131	Up to 128 bytes of data

Response:

bytes 0-3	Number of bytes actually written. If minus one (-1) indicates an error.
byte 4	If an error occurred, this is the DOS **errno** value.

49 SVC_DOS_DELETE

Delete a DOS file

The TCS Master will delete a specified file from the DOS file system. A string of characters (up to 128) containing the name of the file to be deleted is read from the argument area.

Arguments:
bytes 0–127 String of up to 128 characters containing name of file

Response:
byte 0 Zero for success, otherwise a DOS **errno** value

4A SVC_DOS_MKDIR

Make a DOS directory

The TCS Master will create a specified directory within the DOS file system. A string of characters (up to 128) containing the name of the directory to be created is read from the argument area.

Arguments:
bytes 0–127 String of up to 128 characters containing name of directory

Response:
byte 0 Zero for success, otherwise a DOS **errno** value

4B SVC_DOS_RMDIR

Remove a DOS directory

The TCS Master will delete a specified directory within the DOS file system. A string of characters (up to 128) containing the name of the directory to be deleted is read from the argument area.

Arguments:
bytes 0–127 String of up to 128 characters containing name of directory

Response:
byte 0 Zero for success, otherwise a DOS **errno** value

4C SVC_DOS_GETCWD

Get the Current Working Directory

The TCS Master will report the current directory from the DOS file system. A string of characters (up to 67) containing the name of the current directory will be written into the response area.

Arguments:

byte 0	The number of available bytes for the string

Response:

byte 0–66	Up to 67 characters containing current directory name. 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.

4D SVC_DOS_OPEN

Open a specified file for specified operations.

This function causes the TCS Master to perform a DOS open on a file. A maximum of 10 files can be maintained by the TCS Master. The operations word and the file name are read from the arguments area. The file handle number is written into the response area upon completion of the open operation. If the file handle is equal to minus one (–1) an error has occurred, and the DOS **errno** value is also written into response area. The operations word is formed by bitwise-ORing the constants (or values) from the table below.

Constant	Value	Operations Word Description
TCS_RDONLY	0x0000	Opens file for reading only; if this flag is given neither TCS_RDWR nor TCS_WRONLY can given.
TCS_WRONLY	0x0001	Opens file for writing only; if this flag is given neither TCS_RDWR nor TCS_RDONLY can given.
TCS_RDWR	0x0002	Opens file for both reading and writing only; if this flag is given neither TCS_WRONLY nor TCS_RDONLY can given
TCS_APPEND	0x0008	Reposition file pointer to the end of the file before every write operation.
TCS_CREAT	0x0100	Creates and opens a new file for writing; this has no effect if the file already exists.
TCS_TRUNC	0x0200	Opens and truncates the file to zero length.
TCS_EXCL	0x0400	Returns an error value if the file already exists. Only applies when used with TCS_CREAT.
TCS_TEXT	0x4000	Opens the file in text (translated) mode.
TCS_BINARY	0x8000	Opens the file in binary (untranslated) mode.

Arguments:

bytes 0-3	Operations word
bytes 4-7	Unused, for alignment
bytes 8-131	Up to 128 characters containing file name (including path)

Response:

bytes 0	Zero for success, or DOS **errno** value
bytes 1-3	Unused, for alignment
bytes 4-7	File handle number

4E SVC_DOS_CLOSE

Close the DOS file with the specified file handle.

This function causes the TCS Master to perform a DOS close on a file. The file handle is read from the argument area. The number of bytes actually read, and the bytes themselves, are written into the response area.

Arguments:

bytes 0-3	File handle (previously returned from **SVC_DOS_OPEN**)

Response:

byte 0	Zero for success, or DOS **errno** value

4F SVC_DOS_SEEK

Seek the specified number of bytes within a DOS file with the specified file handle.

This function causes the TCS Master to perform a DOS lseek on a file. The file handle and number of bytes to be seek from the beginning of the file are read from the argument area. The file position (number of bytes from the beginning) is written into the response area. If the file position is equal to minus one (-1) an error has occurred, and the DOS **errno** value is written into response area. It is possible to seek past the end of a DOS file!!!

Arguments:

bytes 0-3	File handle (previously returned from **SVC_DOS_OPEN**)
bytes 4-7	Number of bytes to seek from beginning

Response:

bytes 0	Zero for success, or DOS **errno** value
bytes 1-3	Unused, for alignment
bytes 4-7	File position (byte offset from beginning of file)

50 SVC_DOS_READLN

Read the specified number of bytes from a file with the specified file handle.

This function causes the TCS Master to perform a DOS modified read on a file. The maximum number of bytes allowed to be read is 128. The file handle and number of bytes to be read are read from the argument area. Characters are read from the current file position up to and including the first new-line character ('\n'), up to the end of the file, or until the number of characters read is equal to the bytes to be read, whichever comes first. The number of bytes actually read, and the bytes themselves, are written into the response area. If the bytes actually read is zero, an End-of-File was reached. If the bytes actually read is equal to minus one (-1) an error has occurred, and the DOS **errno** value is written into response area.

This routine is very similar to an fgets() function, and should only be performed on ASCII files opened in text mode. This routine can be executed on an ASCII file opened in binary mode, you will, however, get the < CR > character that precedes the < LF > or new-line character.

Arguments:

bytes 0–3	File handle (previously returned from **SVC_DOS_OPEN**)
bytes 4–7	Number of bytes to be read

Response:

bytes 0–3	Number of bytes actually read. If zero, indicates end-of-file reached, if minus one (-1) indicates an error.
byte 4	If an error occurred, this is the DOS **errno** value.
bytes 5–7	Unused, for alignment
bytes 8–131	Up to 128 bytes of data (if no errors occurred)

51 SVC_DOS_RDDIR

Read the contents of a specified directory.

This function causes the TCS Master to perform a series of DOS function calls, based on the arguments read from the argument area. Since the response to this SVC depends on the contents of a directory, the type of file information being requested, and the limited size of the response area, only five file information structures can be passed back per SVC call. This requires that the programmer must provide file index along with a file attributes flag and search filename (wildcards '*' and '?' are permissible).

Constant	Value	File attributes flag description
TCS_A_NORMAL	0x00	Normal. File can be written or read without restriction.
TCS_A_RDONLY	0x01	Read only. File cannot be opened for a "write", and a file with the same name cannot be created. Normal file information also returned.
TCS_A_HIDDEN	0x02	Hidden file. Cannot be found by a directory search. Normal file information also returned.
TCS_A_SYSTEM	0x04	System file. Cannot be found by a directory search. Normal file information also returned.
TCS_A_VOLID	0x08	Volume ID. Only one file can have this attribute, and it must be in the root directory.
TCS_A_SUBDIR	0x10	Subdirectory. Normal file information also returned.
TCS_A_ARCH	0x20	Archive. Set whenever the file is changed, and cleared by the MS-DOS BACKUP command.

The response area contains the number of file records read, whether there are potentially more matching files, the file record index to be used if there are potentially more matching files, and the file records themselves (up to five). Each file record is consists of 24-bytes, constructed as follows:

File record:

bytes 0–12	File name (13-bytes including NULL terminator)
byte 13	File attributes
bytes 14–15	File index
bytes 16–17	Time, bit values (hour[15..11].minutes[10..5].seconds[4..0])
bytes 18–19	Date, bit values (year[15..9].month[8..5].day[4..0])
bytes 20–23	Size, length of file in bytes

Arguments:

bytes 0–3	File attributes flag
bytes 4–7	File record index (first file record to be saved)
bytes 8–135	Search filename (wildcards are permissible)

Response:

byte 0	Number of file records read
byte 1	Done-flag (non-zero indicates there are NO more files matching the arguments descriptions)
bytes 2–3	Unused, for alignment
bytes 4–7	Index to the next file record (only used if the Done-flag had not been set)
bytes 8–127	Up to five 24-byte file records

80 SVC_TCS_RD_ACTION

This command causes the TCS Master to send an ACTION register read request to the TCS slave on the specified card.

Arguments:

byte 0	Card Class/Type
byte 1	Unused, for alignment
bytes 2-3	Card Index
byte 4	Action register number

Response:

byte 0	Zero for success, non-zero for error
byte 1	Data read from action register

81 SVC_TCS_WR_ACTION

This command causes the TCS Master to send an ACTION register write request to the TCS slave on the specified card.

Arguments:

byte 0	Card Class/Type
byte 1	Unused, for alignment
bytes 2-3	Card Index
byte 4	Action register number
byte 5	Data to be written

Response:

byte 0	Zero for success, non-zero for error

82 SVC_TCS_RD_EEPROM

This command causes the TCS Master to send an EEPROM register read request to the TCS slave on the specified card.

Arguments:

byte 0	Card Class/Type
byte 1	Unused, for alignment
bytes 2-3	Card Index
byte 4	EEPROM register number

Response:

byte 0	Zero for success, non-zero for error
byte 1	Data read from EEPROM register

83 SVC_TCS_WR_EEPROM

This command causes the TCS Master to send an EEPROM register write request to the TCS slave on the specified card.

Arguments:

byte 0	Card Class/Type
byte 1	Unused, for alignment
bytes 2-3	Card Index
byte 4	EEPROM register number
byte 5	Data to be written

Response:

byte 0	Zero for success, non-zero for error

84 SVC_TCS_RD_GATEARRAY

This command causes the TCS Master to send an Gate Array register read request to the TCS slave on the specified card.

Arguments:

byte 0	Card Class/Type
byte 1	Unused, for alignment
bytes 2-3	Card Index
byte 4	Gate Array number
byte 5	Gate Array register number

Response:

byte 0	Zero for success, non-zero for error
byte 1	Data read from Gate Array register

85 SVC_TCS_WR_GATEARRAY

This command causes the TCS Master to send an Gate Array register write request to the TCS slave on the specified card.

Arguments:

byte 0	Card Class/Type
byte 1	Unused, for alignment
bytes 2-3	Card Index
byte 4	Gate Array number
byte 5	Gate Array register number
byte 6	Data to be written

Response:

byte 0	Zero for success, non-zero for error

86 SVC_TCS_RD_HWARE

This command causes the TCS Master to send an HARDWARE register read request to the TCS slave on the specified card.

Arguments:

byte 0	Card Class/Type
byte 1	Unused, for alignment
bytes 2–3	Card Index
byte 4	Hardware register number

Response:

byte 0	Zero for success, non-zero for error
byte 1	Data read from Hardware register

87 SVC_TCS_WR_HWARE

This command causes the TCS Master to send an HARDWARE register write request to the TCS slave on the specified card.

Arguments:

byte 0	Card Class/Type
byte 1	Unused, for alignment
bytes 2–3	Card Index
byte 4	Hardware register number
byte 5	Data to be written

Response:

byte 0	Zero for success, non-zero for error

88 SVC_TBUS_RD

Perform a T-Bus read cycle on desired processor node.

This function causes the TCS Master to send a TCS T-Bus memory setup request and a T-Bus memory read request to the TCS Slave on the specified processor node. The requestor MUST correctly format all the bytes of this message or the remote T-Bus may be inadvertently hung. This command allows complete freedom in setting up the request, and does NO validity checking.

Arguments:

byte 0	Card Class/Type
byte 1	Unused, for alignment
bytes 2-3	Card 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)
bytes 9-11	Unused, for alignment
bytes 12-15	TBus address bits 31..0

TBUS
4 0x02
5 0x01 word
6 0x08
7 0x03
8 0x02

Response:

byte 0	Nack or Ack code from TCS Slave
byte 1	TBus response byte
bytes 2-3	Unused, for alignment
bytes 4-7	Data read from TBus cycle

89 SVC_TBUS_WR

Perform a T-Bus write cycle on desired processor node.

This function causes the TCS Master to send a TCS T-Bus memory setup request and a T-Bus memory write request to the TCS Slave on the specified processor node. The requestor MUST correctly format all the bytes of this message or the remote T-Bus may be inadvertently hung. This command allows complete freedom in setting up the request, and does NO validity checking.

Arguments:

byte 0	Card Class/Type
byte 1	Unused, for alignment
bytes 2-3	Card 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)
bytes 9-11	Unused, for alignment
bytes 12-15	TBus address bits 31..0
bytes 16-19	TBus write data

Response:

byte 0	Nack or Ack code from TCS Slave
byte 1	TBus response byte

1.5.3 SVC Argument Descriptions

This section describes the standard argument values that can be passed to the SVC commands that deal specifically with the various card types within a TC2000 system.

Card Class/Type

00	SVC_CC_CLK	TC CLK or TC CTI (level 1) clock card
01	SVC_CC_SA	TC SS or TC U S switch server A card
02	SVC_CC_SB	TC SS or TC U S switch server B card
03	SVC_CC_RA	TC U S switch requestor A card
04	SVC_CC_RB	TC U S switch requestor B card
05	SVC_CC_NODE	TC PPN or other processor node
06	SVC_CC_CLK2A	TC CTI (level 2) clock A card
07	SVC_CC_CLK2B	TC CTI (level 2) clock B card
08	SVC_CC_MA	TC U S switch middle A card
09	SVC_CC_MB	TC U S switch middle B card

Card Index

This argument is the index of a particular Card Class/Type. There can at most two first level clock cards, seven second level clock cards, sixty-four of each switch card type and 512 processors.

Card Class	Index Range
SVC_CC_CLK	0x000...0x001
SVC_CC_SA	0x000...0x03F
SVC_CC_SB	0x000...0x03F
SVC_CC_RA	0x000...0x03F
SVC_CC_NODE	0x000...0x1FF
SVC_CC_CLK2A	0x000...0x007
SVC_CC_CLK2B	0x000...0x007
SVC_CC_MA	0x000...0x03F
SVC_CC_MB	0x000...0x03F

Card State

This is the argument that is used in conjunction with the **SVC_SET_COMP_STATE** command. This is an overall state or goal for the specified card. It hides many other more detailed states which may change with versions od software. The states are analogous to those specified by the "CardUse" line in the BOOTCFG.TCS file and the "Configuration Card-Use" TEX command. The operating system or diagnostic may request that a specific card be placed in one of the states listed below.

01	SVC_CS_SYS	System usage
02	SVC_CS_AUX	Auxiliary usage
03	SVC_CS_ISOLATE	Isolated usage
04	SVC_CS_ON	Disabled on usage (nodes only)
05	SVC_CS_OFF	Disabled off usage