ADVENT INSTRUMENTS INC.

AITSPC DLL & DDP1100 DEMO PROGRAM

User Guide & Reference Manual

Release 1.2
October 1998

Table of Contents

Section 1 Installation & Setup

Before You Begin

Verifying the System Requirements

The AiTspc DLL and the DDP1100 Demo Program require the following minimum system setup for the target computer.

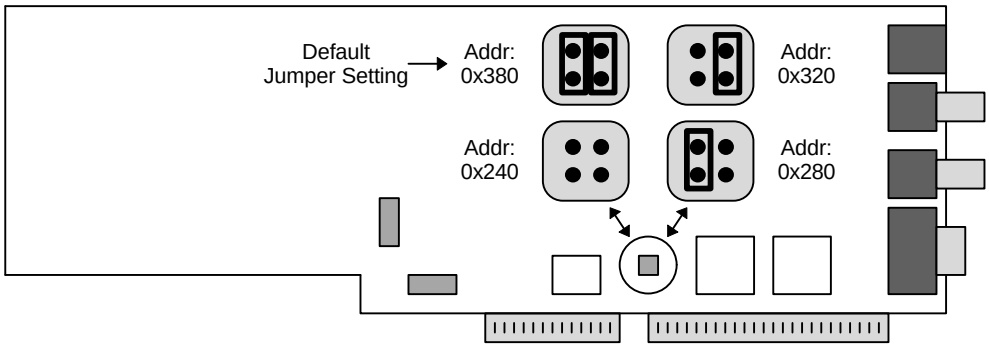
- Intel 386SX PC computer
- VGA type monitor
- Microsoft Windows 3.1, 3.11, 95 operating system
- Four Megabytes of RAM
- One full length 16 bit ISA expansion slot

Backup the Software

Make a backup copy of the software disk before beginning the software installation. Keep the original disk in a safe location.

Installing the Telephone Signal Processing Card

Before installing the Telephone Signal Processing Card (TSPC), verify that the correct base I/O address is set. The base I/O address used is determined by the settings of two jumpers shown in the figure below. The factory default settings includes both jumpers, as shown below, for a base I/O address of 0x0380. This address should be free of conflict from most other peripheral cards; however, in case of conflict, change the jumpers to select an alternate base I/O address. The DDP1100 program will automatically scan all four possible base I/O addresses for a TSPC on startup. Note, that the TSPC is designed to avoid interrupt and DMA channel conflicts, and as such, these parameters do not need to be configured. The only possible conflict between peripheral cards is with the base I/O address selected.



Base I/O Address

To install the TSPC:

1. Turn off the computer, including all external peripherals
2. Leave the power cable connected to a grounded outlet, so that the system is grounded
3. Remove the cover from the computer
4. Locate an unused 16 bit ISA expansion slot in your system
5. Insert the TSPC into the expansion slot
6. Replace the cover from the computer

Note: The TSPC can create high voltages. Ensure adequate space between the TSPC and other adjacent peripheral cards in the computer. Also, always re-attach the cover to the computer before turning it on in order to ensure that no contact can be made to the high voltage portions of the TSPC.

Installing the Windows Software

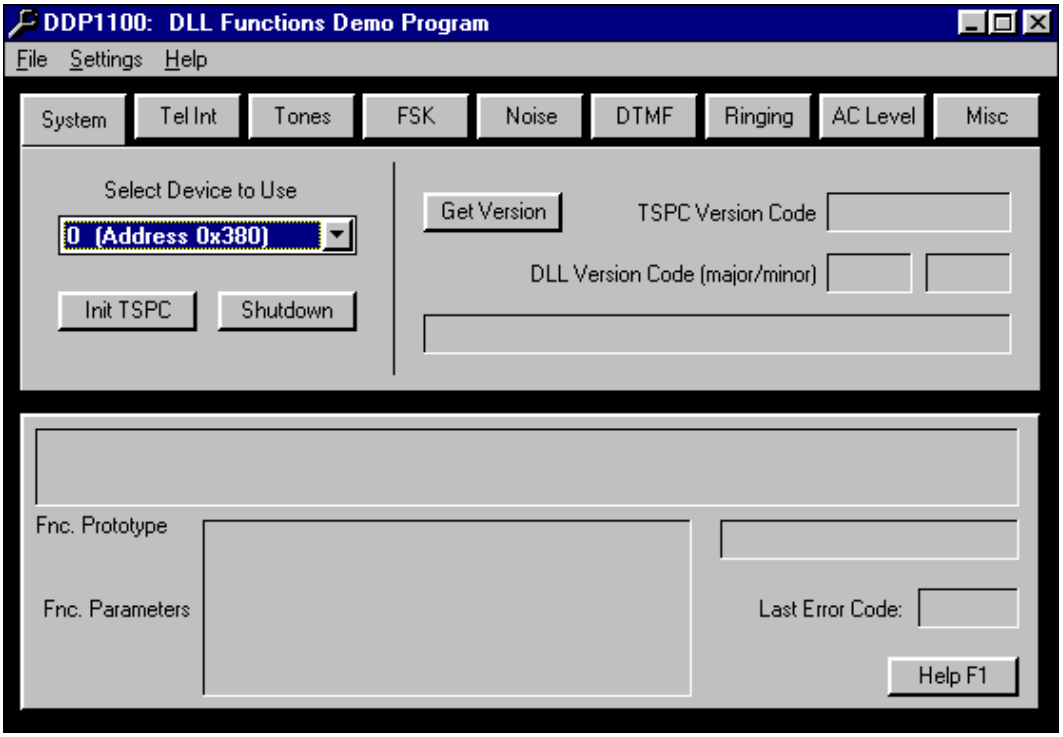
To install the software package, turn on the system and launch either the Windows 3.1, 3.11, or 95 operating system. Insert the floppy disk into the 3.5 Disc Drive (usually drive A:). For users of Windows 3.1 or 3.11, from the Windows Program Manager, select menu command [FILE] [RUN]. Type "A:\setup" and press ENTER. For users of Windows 95, press the START button, followed by the RUN selection. Then type "A:\setup" and press ENTER.

The setup program will extract all the necessary files and install them into the directory you select. The default directory is set to "C:\AI\DDP1100"; however, alternate directories may be chosen.

The Setup program will automatically create both a program group and program icon within the Windows Program Manager

Testing the Installation

Once you have finished installing the hardware and software, you can verify the correct setup by launching the "DDP1100" program from the Windows Program Manager. Once the program has finished loading itself in the system, the screen should display the following.



■ Section 2 Getting Started

What is the DDP1100 and AiTspc DLL?

The DDP1100 program serves as a demonstration of the usage and operation of the functions available in the AiTspc Dynamic Linked Library (DLL). The AiTspc DLL will operate with the AI-150 Caller ID Simulator's PC card (TSPC). This allows users of the AI-150 to construct their own custom test programs, or integrate the TSPC into their existing test systems by using the AiTspc DLL. Any Windows development environment that supports access to 16 bit DLL's will be able to use the AiTspc DLL. The DLL contains a wide array of functions that can be used to control various aspects of the TSPC. These include generating tones, FSK data, ringing, DTMF, setting the characteristics of the telephone interface, measuring AC signal levels, and measuring DTMF signals. The AiTspc DLL is also able to control up to four TSPC's per PC. This allows for very sophisticated testing environments that can perform multiple functions in parallel.

The DDP1100 program uses the AiTspc DLL and allows the user an easy way to understand the requirements and operation of each of the available functions calls. Each function in the DLL can be executed at any time, with the program showing the values of all the parameters passed to the function, along with any data returned from the function. The program is also linked to the on-line help that presents more information on each of the DLL functions along with the possible errors that may result.

32 Bit Windows Support:

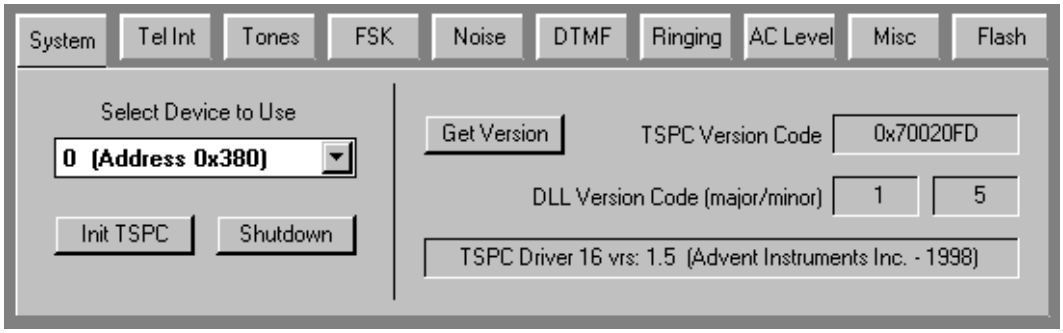
Though the DDP1100 program operates under 16 bit Windows environments, a 32 bit version of the DLL is included on the setup disk for use within 32 bit development environments operating under Windows 95. For more information, see the section: Using the 32 Bit DLL

DOS Support:

DOS based applications can utilize the TSPC hardware for custom test systems by integrating a DOS library file into their development environment. For more information on the DOS library, see the section: Using the DOS Library

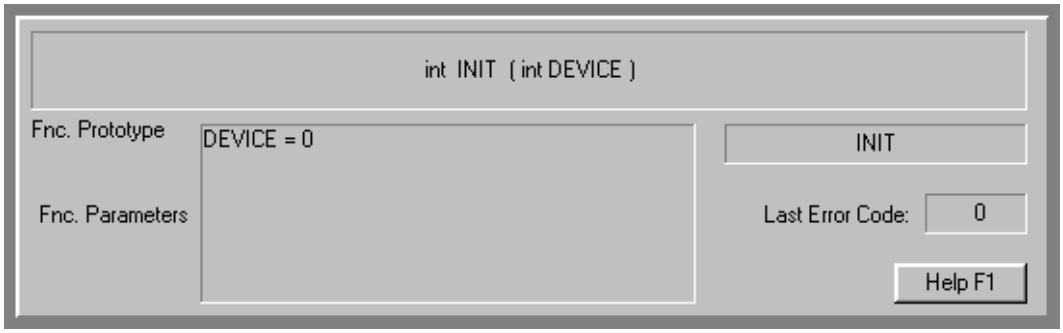
Using the DDP1100 Program

The DDP1100 is a simple program designed to help users understand how the AiTspc DLL functions operate and can be used. Users can call every function in the DLL by passing various values for the parameters. The program window is composed of two main sections. The top half of the window is used to display the various function parameters along with the command buttons to execute those functions. All of the DLL functions are broken out into their logical groupings and displayed in ten different panels. The panel shown below holds the command buttons for the system type function calls. To display the other panels, either click the mouse on the panel header, or use the appropriate menu command.



The above system panel is used to initialize and shutdown the TSPC hardware as well as retrieving the TSPC and DLL version information. To initialize a TSPC, simply select the device number from the drop-down list and press the INIT TSPC button.

The bottom half of the window is dedicated to showing the details of each function called. Pressing the INIT TSPC button on the system panel will cause a call to the AiTspc DLL function INIT. The details of this function are displayed below.

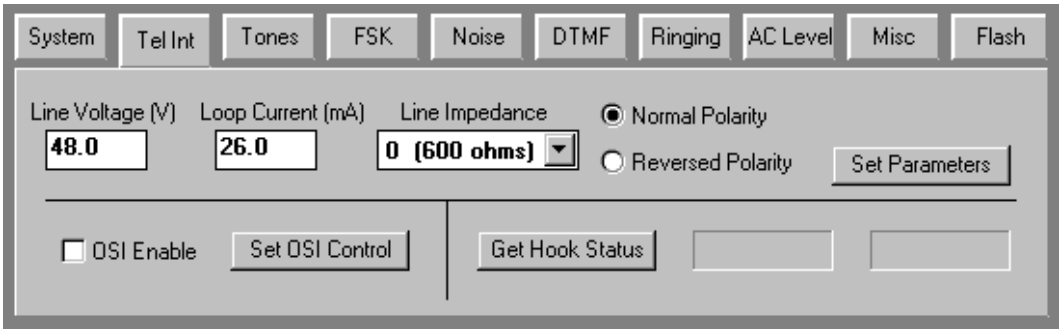


The top part shows the function prototype using the 'C' language convention. Below this, the values passed or returned in each parameter of the function are displayed. Over to the right side, the function name and returned error code are displayed. If no error has occurred, an error code of zero will be shown. A non-zero error code is highlighted in yellow. Pressing the HELP button will bring up a help window on the function displayed. In the case of the INIT command, if a

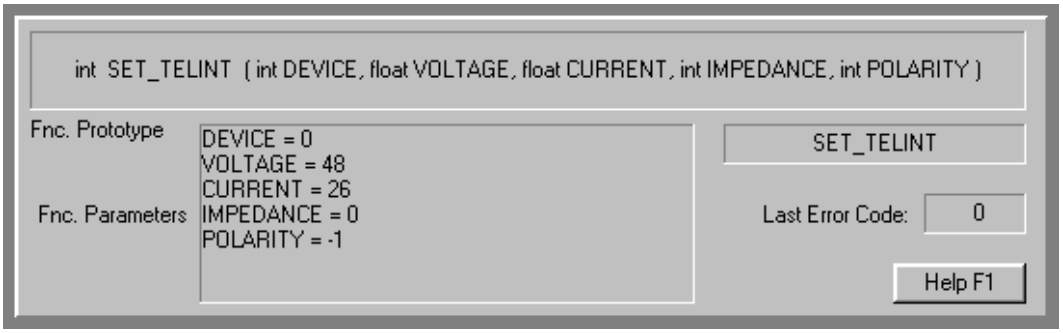
TSPC was present at the selected device number, the error code should be zero. If no TSPC was present, then a non-zero error code will be displayed.

Clicking the mouse on the GET VERSION command button will retrieve the version information of the TSPC and the DLL. Like the INIT command, the bottom half of the window will display the function prototype along with the contents of all the parameters passed to the function or returned from it.

To change the telephone interface settings, click the mouse on the TEL INT panel. You should see the following display:



In this panel the telephone interface characteristics can be programmed. Simply enter the new voltage or current in the appropriate text box, or select a new line impedance from the drop-down list, or click the mouse on a new line polarity setting. Once the desired settings have been entered, press the SET PARAMETERS button to execute the SET_TELINT function. The details of this function call is displayed similar to the INIT and GET VERSION function calls. The function prototype, parameter values, and error code will be displayed on the bottom half of the window as follows.



All of the ten control panels work in a similar manner. Pressing a button will invoke the selected function with the displayed settings. The details of any of the function calls is displayed in the lower half of the window.

At any time, clicking the Help button, or pressing the F1 key will display the on-line help for the last function executed.

Section 3 AiTspc DLL Function Reference

The following list shows all of the available functions for the AiTspc DLL. The functions are ordered in logical groups of system calls, telephone interface calls, tone generator calls, FSK modulator calls, noise generator calls, DTMF calls, ring generator calls, level meter calls, and finally signal flow and digital I/O calls.

INIT	Initialize the selected TSPC
SHUTDOWN	Shutdown the selected TSPC
GET_VERSION	Retrieve the DLL and selected TSPC version codes
SET_TELINT	Set the telephone interface characteristics
GET_HOOKSTATUS	Return the status of the hook switch
RESET_FLASHTIMING	Reset the flash timing analyzer
GET_FLASHTIMING	Return the results of the flash timing analyzer
RESET_TIMER	Reset the flash timing analyzer timer
SET_TONE	Define the levels and frequency for the tone generator
SET_TONEPATTERN	Define the duration and interval for the tone generator
START_TONE	Start the tone generator
STOP_TONE	Stop the tone generator
GET_TONESTATUS	Return whether the tone generator is active
SET_FSK	Define the FSK modulation parameters
START_FSK	Start the FSK generator
STOP_FSK	Stop the FSK generator
GET_FSKSTATUS	Return whether the FSK generator is active
SET_NOISE	Set the noise generator level
START_NOISE	Start the noise generator
STOP_NOISE	Stop the noise generator
SEND_DTMF	Generate a DTMF digit
SET_DTMF	Define the received DTMF requirements
GET_DTMF	Wait for a received DTMF digit
GET_DTMF2	Wait for a received DTMF digit
SET_RING	Define the ringing frequency and level
SET_RINGPATTERN	Define the duration and interval for ringing
START_RING	Start the ringing generator
STOP_RING	Stop the ringing generator
GET_RINGSTATUS	Return whether the ringing generator is active
SET_ACFILTER	Define any AC level meter filters
SET_ACSPEED	Define the AC level meter integration time
GET_ACLEVEL	Return the reading of the AC level meter

SET_AUDIO	Set the audio switching paths
SET_DOUT	Set the digital output port bits
GET_DIN	Return the state of the digital input port
XFER_SERIALDATA	Send and receive data from serial bus
WAIT	Wait until a specified time interval expires

Note	For all of the following function descriptions, their parameter lists reflect that of the 16 bit Windows DLL version. For the 32 bit Windows DLL, the type of parameters passed to and from the functions are modified in some cases. See the section: Using the 32 Bit DLL.

Function: INIT

Description:

The INIT function attempts to locate a TSPC at the specified base I/O address. If it is successful, it will reset and initialize the TSPC settings as well as initializing the internal data structures of the AiTspc DLL. This should be the first function call made, before attempting any other function calls. If successful, the function will return zero as its error code. If a non-zero error code was returned, no further function calls should be attempted to the selected device number.

Function Prototype:

int INIT (int device)

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Up to 4 TSPC's can be controlled from the PC ISA bus, each at a different base I/O address. Each device number refers to a specific TSPC at a specific base I/O address as follows:

DEVICE = 0	Base I/O address = 0x380
DEVICE = 1	Base I/O address = 0x320
DEVICE = 2	Base I/O address = 0x280
DEVICE = 3	Base I/O address = 0x240

See Section 1: Installation & Setup for how to change the base I/O address used by a TSPC.

Function Error Codes:

0	No Error
100 to 101	No TSPC Found with Specified Device Number
102	TSPC SRAM Check Failure
105	TSPC Revision Incompatible with AiTspc DLL
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: SHUTDOWN

Description:

The SHUTDOWN function puts the selected TSPC into its reset state. This stops the generation of all tones and sets all outputs to a low state. After SHUTDOWN has been called, any further attempts to communicate with the hardware will result in error. Use the INIT command to re-initialize the TSPC.

Function Prototype:

int SHUTDOWN (int device)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: GET_VERSION

Description:

The GET_VERSION function returns various information about the TSPC hardware and the AiTspc DLL. This includes an ASCII text string, TSPC hardware revision code, and revision codes for the DLL itself.

Function Prototype:

```
int  GET_VERSION  (  int device,  char version[60],  long *TSPCrev,
                          int *DDLmajor,  int *DLLminor  )
```

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.
VERSION	The version parameter is a pointer to an array of at least 60 characters. The function will return an ASCII version string in this array.
TSPCREV	The TSPCrev parameter is a pointer to a 32 bit long word. The function returns a 32 bit TSPC revision code using this parameter. This revision code should be given in any correspondence with technical support, such that the specific revision and history of the TSPC can be traced.
DLLMAJOR	The DLLmajor parameter is a pointer to a 16 bit word. The function returns the major or primary revision code of the AiTspc DLL using this parameter. The major revision code may be changed in the future when backward compatibility for some functions is no longer supported.
DLLMINOR	The DLLminor parameter is a pointer to a 16 bit word. The function returns the minor revision code of the AiTspc DLL using this parameter. Any future changes to this DLL will be reflected in a new minor revision code. However, as long as the major revision code does not change, a different minor revision code will only signify new functions, and full backward compatibility will be maintained with earlier DLL's of the same major revision.

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: SET_TELINT

Description:

The SET_TELINT function is used to specify the line voltage, loop current, source impedance, and line polarity of the telephone line interface. Any changes in the settings will take place immediately after the function call. If successful the function returns an error code of zero.

Function Prototype:

```
int  SET_TELINT  (  int device,  float voltage,  float current,
                    int impedance,  int polarity  )
```

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

VOLTAGE — The voltage parameter is a 32 bit floating point value that represents the desired line voltage setting of the telephone line interface. The value is specified in units of Volts between the minimum of 20.0 and the maximum of 52.0.

CURRENT — The current parameter is a 32 bit floating point value that represents the desired loop current setting of the telephone interface. The value is specified in units of mA between the minimum of 20.0 and the maximum of 40.0.

IMPEDANCE — The impedance parameter is a 16 bit integer value that is used to represent different possible AC line impedance's. The valid range for this parameter is from 0 to 2 with the following representations:
0 = 600 ohms source impedance
1 = 900 ohms source impedance
2 = complex source impedance

POLARITY — The polarity parameter is a 16 bit integer value that is used to represent the polarity of the telephone interface. Passing a non-zero value will set the polarity to its reversed state. A zero value will set the line polarity to its normal state.

Function Error Codes:

0	No Error
110	Line Voltage Outside Valid Range
111	Loop Current Outside Valid Range
112	Line Impedance Outside Valid Range
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: SET_OSICONTROL

Description:

The SET_OSICONTROL function is used to generate an OSI (open switching interval). During an OSI, any CPE connected to the telephone interface is disconnected from the DC feeding voltage, as well as the AC transmit and receive circuits. In effect, the tip and ring leads of the RJ-11 jack are open-circuited from each other and any other circuit on the TSPC card. Depending on the parameter value passed, the tip and ring leads can be either connected to the internal telephone interface circuitry, or completely disconnected.

Function Prototype:

int SET_OSICONTROL (int device, int OSIenable)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.
OSIenable	The OSIenable parameter is a 16 bit integer value that is used to control whether or not the tip and ring leads of the telephone interface are open circuited. Passing a value of zero will disconnect the DC feeding voltage and the AC transmit and receive circuits from the tip and ring leads. A non-zero value connects the DC feeding voltage and AC circuits to the tip and ring leads for normal operation.

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: GET_HOOKSTATUS

Description:

The GET_HOOKSTATUS function can be used to ascertain the telephone interface status. The two parameters returned will indicate if the telephone line is on or off hook, and whether the telephone line is in an unbalanced condition. An unbalanced condition indicates a mismatch in the current flow of the tip and ring leads. This usually occurs when either lead is connected to earth ground via a low impedance.

Function Prototype:

int GET_HOOKSTATUS (int device, int *offhook, int *unbalanced)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.
OFFHOOK	The offhook parameter is a pointer to a 16 bit integer word. The function will return a value of 0 if the TSPC detects an on-hook condition, or a value of -1 for an off-hook condition.
UNBALANCED	The unbalanced parameter is a pointer to a 16 bit integer word. The function will return a value of 0 if the TSPC's telephone interface is NOT unbalanced, or a value of -1 if it is unbalanced.

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: RESET_FLASHTIMING

Description:

The RESET_FLASHTIMING function sets the polarity control of the line flash timer and resets its internal state machine. This routine in conjunction with GET_FLASHTIMING can be used to accurately determine the timing of hook switch transitions of the CPE under test. The line flash timer will return time stamps for either off-hook to on-hook to off-hook transitions (for flashes or pulse dialing), or on-hook to off-hook to on-hook transitions (for timing line seizure and release). The polarity parameter defines what order of hook switch transitions the GET_FLASHTIMING function will look for. The line flash timer operates as a state machine with three internal states. These are: 1) waiting for 1st transition, 2) waiting for 2nd transition, and 3) measurement done. Calling this routine will reset the state machine to wait for the 1st transition. Retrieving the current state value and the time stamps for the transitions is done by calling the GET_FLASHTIMING function.

Function Prototype:

int RESET_FLASHTIMING (int device, int polarity)

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

POLARITY — The polarity parameter controls whether the flash timing analyzer measures the timing of an off-hook to on-hook to off-hook (normal flash), or an on-hook to off-hook to on-hook series of transitions (line seizure and release.)

POLARITY= 0 Set to measure off-hook to on-hook to off-hook
POLARITY= -1 Set to measure on-hook to off-hook to on-hook

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: GET_FLASHTIMING

Description:

The GET_FLASHTIMING function, in conjunction with the RESET_FLASHTIMING function, can be used for accurately measuring CPE hook switch flashes, pulse dialing, or other transitions of the hook switch. The line flash timer returns time stamps for either off-hook to on-hook to off-hook transitions (for flashes or pulse dialing), or on-hook to off-hook to on-hook transitions (for timing line seizure and release). This setting can be specified by the RESET_FLASHTIMING function. The line flash timer operates as a state machine with three internal states. These are: 1) waiting for 1st transition, 2) waiting for 2nd transition, and 3) measurement done. Calling the GET_FLASHTIMING function will return the current state value along with the current timer value, time stamp for the first transition and time stamp for the second transition.

The normal procedure for measuring CPE flash times, pulse dialing, or other hook switch transitions, is to first call RESET_FLASHTIMING with the desired polarity setting. Following this, poll GET_FLASHTIMING to read the timer value and the time stamps of any transitions detected. Once the desired data has been read, the procedure can be repeated for collecting more hook switch transitions.

Note that for the first transition only (going either on-hook or off-hook), a 10 msec debouncing period is activated. As such, two fast hook switch transitions within 10 msec from each other will be ignored.

Function Prototype:

```
int  GET_FLASHTIMING  (  int device,  int *status,  float *timenow,  float *time1,
                                                    float *time2  )
```

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

STATUS — The status parameter is a pointer to a 16 bit integer word. The function will return the value representing the current state of the flash timer. This value can be either 1, 2, or 3.

STATUS= 1	Waiting for first transition
STATUS= 2	Waiting for second transition
STATUS= 3	Measurement done

TIMENOW — The timenow parameter is a pointer to a 32 bit floating point value. The function returns the current time (in seconds) of the flash timer. The flash timer has a maximum time value of

approximately 44739 seconds (12.4 hours) . Beyond this, the timer will start again at 0 seconds. The timer value can be forced to zero anytime by calling the RESET_TIMER function.

TIME1 — The time1 parameter is a pointer to a 32 bit floating point value. The function returns the time stamp for the first transition in units of seconds. This first CPE hook switch transition can be either going on-hook or going off-hook depending on the polarity setting (see RESET_FLASHTIMING function). If the first transition has not been detected yet (status is 1), then time1 will return the value 0.0.

TIME2 — The time2 parameter is a pointer to a 32 bit floating point value. The function returns the time stamp for the second (and final) transition in units of seconds. This second CPE hook switch transition can be either going off-hook or going on-hook depending on the polarity setting (see RESET_FLASHTIMING function). If the second transition has not been detected yet (status is 1 or 2), then time2 will return the value 0.0.

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: RESET_TIMER

Description:

The RESET_TIMER function is used to zero the line flash timer. The line flash timer is used with the RESET_FLASHTIMING and GET_FLASHTIMING functions to accurately measure CPE hook switch transitions. Transition time stamp values can range from 0 to approximately 44739 seconds (12.4 hours). Calling the RESET_TIMER function will zero the time counter to 0 seconds.

Note that the GET_DTMF2 function also makes use of the same timer for returning the start and stop time of a DTMF tone. As such, calling RESET_TIMER will also effect the results from the GET_DTMF2 function.

Function Prototype:

int RESET_TIMER (int device)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: SET_TONE

Description:

The SET_TONE command is used to specify the frequency and level of the dual tone generator. This tone generated is initiated with the START_TONE command. If a tone is being generated, then using the SET_TONE will not change the frequency or level of the current tone. To update the tone to the new settings, call the START_TONE function again. The signal levels are specified in Vrms for an unterminated telephone line. As the telephone line is loaded by a CPE under test, the actual voltage level at tip and ring will be reduced. If the tone generator output is programmed to be sent to the BNC output connector, via the SET_AUDIO function, then the level at the BNC output will be 12 dB lower than the level specified for an unterminated line.

Function Prototype:

int SET_TONE (int device, float freqA, float levelA, float freqB, float levelB)

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

FREQA — The freqA parameter is a 32 bit floating point value that represents the frequency of tone generator A. The value is specified in units of Hz, and must be between the limits of 10 to 20,000 Hz.

LEVELA — The levelA parameter is a 32 bit floating point value that represents the level of tone generator A. The value is specified in units of Vrms for an unterminated telephone line, and must be between the limits of 0.0 and 4.0 Vrms.

FREQB — The freqB parameter is a 32 bit floating point value that represents the frequency of tone generator B. The value is specified in units of Hz, and must be between the limits of 10 to 20,000 Hz.

LEVELB — The levelB parameter is a 32 bit floating point value that represents the level of tone generator B. The value is specified in units of Vrms for an unterminated telephone line, and must be between the limits of 0.0 and 4.0 Vrms.

Function Error Codes:

0	No Error
120	Tone Frequency Outside Valid Range
121	Tone Level Outside Valid Range
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: SET_TONEPATTERN

Description:

The SET_TONEPATTERN function is used to define a tone on/off pattern for the dual tone generator. The tone on time is specified by the duration parameter, while the interval parameter specifies the time between the tone burst starts. The interval time must be greater than the duration time, otherwise a non-zero error code will be returned by this function.

Function Prototype:

int SET_TONEPATTERN (int device, long duration, long interval)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.
DURATION	The duration parameter is a 32 bit long integer value that specifies the tone duration time. The tone duration is given in units of milliseconds and can range between 1 and 1,000,000 msec.
INTERVAL	The interval parameter is a 32 bit long integer value that specifies the time interval between the start of the tone bursts. The interval time is given in units of milliseconds and can range between 1 and 1,000,000 msec.

Function Error Codes:

0	No Error
122	Duration Outside Valid Range
123	Interval Outside Valid Range
124	Duration Greater or Equal to Interval
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: START_TONE

Description:

The START_TONE function is used to start or update the settings of the dual tone generator. The tone generator can be put into a continuous or pulsed mode of operation as determined by the pattern parameter. Once started, the tone generator can be pre-programmed to turn off at a specified time or run continuously depending on the duration parameter. The time at which the function returns control to the calling routine can be immediately after starting the tone, or only when the tone has completed. This is set by the wait parameter. The tone generator can not be started if the ringing generator is active. Also, if the FSK modulator is currently active, starting the tone generator will stop the FSK modulator.

Function Prototype:

int START_TONE (int device, int pattern, long duration, int wait)

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

PATTERN — The pattern parameter is a 16 bit integer value that controls if the tone generator will be producing a tone pattern or continuous tones. If the value is non-zero, then the tone generator will produce a tone pattern, following the duration and interval set by the SET_TONEPATTERN function. Otherwise, the tone produced will be continuous.

DURATION — The duration parameter is a 32 bit long integer value that specifies how long the tone will be generated for. The duration is given in units of milliseconds and must be between the values of 1 and 1,000,000 msec. For this duration, the tone can be either continuous or pulsed as determined by the pattern parameter. If the pattern parameter is non-zero, then the tone generator will produce tone bursts for the total duration specified. There is a maximum of up to 32767 tone bursts during the entire period. A special case exists when the duration parameter contains the value of -1. In this case, the tone generator will stay on indefinitely (continuous, or pulsed).

WAIT — The wait parameter is a 16 bit integer value that determines when the AiTspc DLL will return control to the calling program. A wait value of zero, will cause the AiTspc DLL to start the tone generator and then return control back to the calling routine. However, if the wait value is non-zero, then the AiTspc DLL will not return control back to the calling routine until the tone has

been generated for the time specified by the duration parameter. If the duration parameter is set to -1, then the AiTspc DLL will always return control immediately after starting the tone, regardless of the wait value.

Function Error Codes:

0	No Error
125	Ringing Generator is Active
126	Invalid Tone Duration Value
127	Too Many Tone Pattern Cycles
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: STOP_TONE

Description:

The STOP_TONE command can be used to stop the dual tone generator. If the tone generator is producing a tone, then this function will stop it immediately, otherwise this function has no effect.

Function Prototype:

int STOP_TONE (int device)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: GET_TONESTATUS

Description:

The GET_TONESTATUS function will return the state of the dual tone generator. There are only two possible states for the tone generator, being either active or inactive.

Function Prototype:

int GET_TONESTATUS (int device, int *active)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.
ACTIVE	The active parameter is a pointer to a 16 bit integer value. The function will set this value to -1 if the tone generator is currently active and generating a tone. Otherwise, a value of 0 will be returned.

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: SET_FSK

Description:

The SET_FSK command is used to specify the baud rate, frequencies, and levels of the FSK modulator. This modulator is initiated with the START_FSK command. If the FSK modulator is currently active, then using the SET_FSK will have no effect until the START_FSK function is called again. The signal levels are specified in Vrms for an unterminated telephone line. As the telephone line is loaded by a CPE under test, the actual voltage level at tip and ring will be reduced. If the tone generator output is programmed to be sent to the BNC output connector, via the SET_AUDIO function, then the level at the BNC output will be 12 dB lower than the level specified for an unterminated line.

Function Prototype:

```
int  SET_FSK  (  int device,  float baud,  float mfreq,  float sfreq,
                                 float mlevel,  float slevel  )
```

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

BAUD — The baud parameter is a 32 bit floating point value that defines the baud rate of the FSK modulator. The value is specified in units of bits per seconds and must be between the limits of 100 and 5000 bps.

MFREQ — The mfreq parameter is a 32 bit floating point value that defines the frequency of the mark tone. The value is specified in units of Hz, and must be between the limits of 10 Hz and 20,000 Hz.

SFREQ — The sfreq parameter is a 32 bit floating point value that defines the frequency of the space tone. The value is specified in units of Hz, and must be between the limits of 10 Hz and 20,000 Hz.

MLEVEL — The mlevel parameter is a 32 bit floating point value that defines the level of the mark tone for an unterminated telephone line. The value is specified in units of Vrms, and must be between the limits of 0.0 and 4.0 Vrms.

SLEVEL — The slevel parameter is a 32 bit floating point value that defines the level of the space tone for an unterminated telephone line. The value is specified in units of Vrms, and must be between the limits of 0.0 and 4.0 Vrms.

Function Error Codes:

0	No Error
130	FSK Frequency Outside Valid Range
131	FSK Level Outside Valid Range
132	FSK Baud Rate Outside Valid Range
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: START_FSK

Description:

The START_FSK function is used to start the FSK modulator. The FSK modulator can be put into one of several different modes of operation for single-shot, continuous, or external data input. Once the FSK modulator has been started, it can be turned off by the STOP_FSK function. The time at which the function returns control to the calling routine can be immediately after starting the modulator, or only when all the FSK data has been sent. This is set by the wait parameter. The FSK modulator can not be started if the ringing generator is active. Also, if the tone generator is active when the FSK modulator is started, then tone A of will be turned off, while tone B will continue to run during the FSK data. In this manner, the FSK data can be sent in conjunction with an interfering tone. Calling the FSK_STOP function will terminate both the FSK modulator and tone B. Also, if the FSK modulator has ended by sending all of its data, calling the GET_FSKSTATUS will terminate tone B. Tone B can not be terminated with the STOP_TONE command once the FSK modulator has been started, since it is then considered part of the FSK modulator.

Function Prototype:

int START_FSK (int device, int bits, long data[], int mode, int wait)

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

BITS — The bits parameter is a 16 bit integer value that specified the number of bits of data to send. This value must be in the range of 1 to 4096 bits.

DATA — The data parameter is a pointer to an array of 32 bit long integers. This array should contain the data to send to the FSK modulator. The bits are generated from LSB to MSB, with 32 bits stored in each long integer.

MODE — The mode parameter is a 16 bit integer, which sets the operating mode for the FSK modulator. The possible modes are as follows:

MODE = 0 Single Shot
MODE = 1 Continuous
MODE = 2 External Data Input

For the single shot mode of operation, the FSK modulator will cease to produce any tones once the last data bit has been sent. In the continuous mode, once the last bit of data has been sent,

the FSK modulator will then repeat the cycle by sending the first bit again. Thus the FSK modulator will run indefinitely until the FSK modulator is turned off. The external data input mode, causes the FSK modulator to use the external input B as the data source. A high logic level at this pin will send a mark tone, while a low logic level sends a space tone.

WAIT — The wait parameter is a 16 bit integer value that controls when this function returns control back to the calling routine. If wait is set to a non-zero value and the mode is set to single-shot, then the START_FSK function will not return program control back to the calling routine until the last bit of data has been sent. A zero value for the wait parameter, or a mode that is not single-shot, will cause the function to return program control once the FSK modulator has started.

Function Error Codes:

0	No Error
135	Ringing Generator is Active
136	Invalid FSK Mode
137	Invalid Number of Bits
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: STOP_FSK

Description:

The STOP_FSK function can be used to stop the FSK modulator. If the modulator is active, then this function will stop it immediately, otherwise this function has no effect. In the case tone B is used in conjunction with the FSK modulator, then this function will also stop it.

Function Prototype:

int STOP_FSK (int device)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: GET_FSKSTATUS

Description:

The GET_FSKSTATUS function will return the state of the dual tone generator. There are only two possible states for the tone generator, being either active or inactive. If the FSK modulator was started with tone B active, then tone B will remain active during the FSK data transmission. The GET_FSKSTATUS command will stop tone B if the FSK modulator has finished sending its data.

Function Prototype:

int GET_FSK_STATUS (int device, int *active)

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

ACTIVE — The active parameter is a pointer to a 16 bit integer value. The function will set this value to -1 if the FSK modulator is currently active and generating a tone. Otherwise, a value of 0 will be returned.

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: SET_NOISE

Description:

The SET_NOISE function, defines the level of the noise generator in units of Vrms. The level is specified for an unterminated telephone line. As the telephone line is loaded by a CPE under test, the actual voltage level at tip and ring will be reduced. If the tone generator output is programmed to be sent to the BNC output connector, via the SET_AUDIO function, then the level at the BNC output will be 12 dB lower than the level specified for an unterminated line. If the noise generator is active, then the SET_NOISE function will not change the level of the noise generator until START_NOISE is called again.

Function Prototype:

int SET_NOISE (int device, float noise)

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

NOISE — The noise parameter is a 32 bit floating point value that represents the desired level of the noise generator. The level is specified in units of Vrms and must be between the limits of 0.0 to 1.0 Vrms. The noise produced is spectrally flat till about 20 kHz. The noise level refers to the total noise power produced in the band 0 to 20 kHz.

Function Error Codes:

0	No Error
180	Noise Level Outside Valid Range
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: START_NOISE

Description:

The START_NOISE function is used to start the noise generator and produces the level set by the last SET_NOISE command. The noise generator will continue to operate until the STOP_NOISE function is called, or unless the ringing generator is started with the START_RING function. The noise generator is independent of the dual tone generator and the FSK modulator, as the noise can be started or stopped during other tones.

Function Prototype:

int START_NOISE (int device)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: STOP_NOISE

Description:

The STOP_NOISE function will stop the noise generator from producing any noise.

Function Prototype:

int STOP_NOISE (int device)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: SEND_DTMF

Description:

The SEND_DTMF function generates one of the 16 possible DTMF digits at the specified level, twist, and duration. The functionality of this function can be reproduced by using the tone generator functions; however, this routine provides an easier way to generate DTMF tones. Sending a DTMF tone will stop any other tones that are being generated along with stopping the FSK modulator, if it is active. If the ringing generator is active, then this function will return an error code. Also, this function will not return control to the calling routine until the DTMF tone is finished.

Function Prototype:

int SEND_DTMF (int device, int digit, float level, float twist, int duration)

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

DIGIT — The digit parameter is a 16 bit integer that represents which DTMF digit to generate. The following list relates the digit value to the DTMF digit produced and the corresponding low and high group frequencies.

DIGIT = 0	DTMF 0	Flow = 941.0	Fhigh = 1336.0
DIGIT = 1	DTMF 1	Flow = 697.0	Fhigh = 1209.0
DIGIT = 2	DTMF 2	Flow = 697.0	Fhigh = 1336.0
DIGIT = 3	DTMF 3	Flow = 697.0	Fhigh = 1477.0
DIGIT = 4	DTMF 4	Flow = 770.0	Fhigh = 1209.0
DIGIT = 5	DTMF 5	Flow = 770.0	Fhigh = 1336.0
DIGIT = 6	DTMF 6	Flow = 770.0	Fhigh = 1477.0
DIGIT = 7	DTMF 7	Flow = 852.0	Fhigh = 1209.0
DIGIT = 8	DTMF 8	Flow = 852.0	Fhigh = 1336.0
DIGIT = 9	DTMF 9	Flow = 852.0	Fhigh = 1477.0
DIGIT = 10	DTMF *	Flow = 941.0	Fhigh = 1209.0
DIGIT = 11	DTMF #	Flow = 941.0	Fhigh = 1477.0
DIGIT = 12	DTMF A	Flow = 697.0	Fhigh = 1633.0
DIGIT = 13	DTMF B	Flow = 770.0	Fhigh = 1633.0
DIGIT = 14	DTMF C	Flow = 852.0	Fhigh = 1633.0
DIGIT = 15	DTMF D	Flow = 941.0	Fhigh = 1633.0

LEVEL — The level parameter is a 32 bit floating point value that represents the signal level of the DTMF digit. This level is specified in units of Vrms and must be between the limits of 0.0 and 4.0 Vrms. Like that of the other signal levels, the DTMF level is referred to an unterminated telephone line.

TWIST	The twist parameter is a 32 bit floating point value that represents the level difference between the high group tone and the low group tone. The twist is specified in dB's, with a positive value meaning that the high group tone will be higher in level than the low group tone.
DURATION	The duration parameter is a 16 bit integer value that represents the duration of the DTMF tone. Its value is specified in units of milliseconds and must be between 1 and 32767.

Function Error Codes:

0	No Error
150	Invalid DTMF Digit
151	DTMF level Outside Valid Range
153	DTMF Duration Outside Valid Range
155	Ringing Generator is Active
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: SET_DTMF

Description:

The SET_DTMF function is used to specify the criteria by which the DTMF receiver will accept or reject DTMF tones. Both the maximum frequency error and minimum signal level can be specified by this function.

Function Prototype:

int SET_DTMF (int device, float freqtol, float minlevel)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.
FREQTOL	The freqtol parameter is a 32 bit floating point value that sets the maximum frequency error accepted by the DTMF digit receiver. The value is specified in percent between the limits of 0.0 and 3.0 %. If either the low group or high group frequency of a DTMF digit is outside this tolerance, then the DTMF receiver will ignore the digit.
MINLEVEL	The minlevel parameter is 32 bit floating point value that sets the minimum accepted level by the DTMF digit receiver. The value is specified in units of Vrms, and must be between the limits of 0.0 and 4.0 Vrms. If either the low group or high group tone level is below this value, then the DTMF receiver will ignore the digit.

Function Error Codes:

0	No Error
160	Frequency Tolerance Outside Value Range
161	Minimum Level Outside Valid Range
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: GET_DTMF

Description:

The GET_DTMF routine is used to detect DTMF tones being sent by the CPE. The function will wait until a valid DTMF digit has been received or the time-out duration has elapsed. Upon either event, the function will return the DTMF digit code received (if any), along with the frequency and level of the low and high group tones. This function will not return program control back to the calling routine until a DTMF digit has been received or the time-out duration has elapsed. The SET_DTMF tone is used to define the characteristics of a valid DTMF tone. This function will return an error code if either the tone generator, FSK modulator, or ring generator is currently active.

Function Prototype:

```
int  GET_DTMF  (  int device,  int duration,  float *levelL,  float *freqL,
                  float *levelH,  float *freqH,  int *digit  )
```

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

DURATION — The duration parameter is a 16 bit integer value that represents the amount of time to wait for a DTMF digit. The units for this time duration are in milliseconds and the value must be between 1 and 32767 msec.

LEVELL — The levelL parameter is a pointer to a 32 bit floating point value. This function will set that value to the level of the DTMF low group tone. The returned value will be in units of Vrms. In the case that no DTMF digit was detected, then the level value is set to zero.

FREQL — The freqL parameter is a pointer to a 32 bit floating point value. This function will set that value to the frequency of the DTMF low group tone. The returned value will be in units of Hz. In the case that no DTMF digit was detected, then the frequency value is set to zero.

LEVELH — The levelH parameter is a pointer to a 32 bit floating point value. This function will set that value to the level of the DTMF high group tone. The returned value will be in units of Vrms. In the case that no DTMF digit was detected, then the level value is set to zero.

FREQH — The freqH parameter is a pointer to a 32 bit floating point value. This function will set that value to the frequency of the DTMF

high group tone. The returned value will be in units of Hz. In the case that no DTMF digit was detected, then the frequency value is set to zero.

DIGIT — The digit parameter is a pointer to a 16 bit integer value. This function will set that value to an integer value representing the decoded DTMF tone. If no DTMF tone was detected, then the value returned will be -1. All of the possible values for the DTMF digit are as follows:

DTMF 0	DIGIT value = 0
DTMF 1	DIGIT value = 1
DTMF 2	DIGIT value = 2
DTMF 3	DIGIT value = 3
DTMF 4	DIGIT value = 4
DTMF 5	DIGIT value = 5
DTMF 6	DIGIT value = 6
DTMF 7	DIGIT value = 7
DTMF 8	DIGIT value = 8
DTMF 9	DIGIT value = 9
DTMF *	DIGIT value = 10
DTMF #	DIGIT value = 11
DTMF A	DIGIT value = 12
DTMF B	DIGIT value = 13
DTMF C	DIGIT value = 14
DTMF D	DIGIT value = 15
(none)	DIGIT value = -1

Function Error Codes:

0	No Error
165	Duration is Outside Valid Range
166	Tone or FSK or Ringing is Active
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: GET_DTMF2

Description:

The GET_DTMF2 function is used to detect DTMF tones generated by a CPE under test, similar in nature to the GET_DTMF function. However some significant differences exist between the two routines. The primary purpose of the GET_DTMF routine is to accurately measure the high and low group tone levels and frequencies. As such, the routine engages a pair of low ripple high order DTMF band splitting filters in order to perform the level and frequency analysis. These filters have no dial tone rejection capabilities. Also the GET_DTMF function will return to the calling routine as soon as a valid DTMF tone has been measured. The GET_DTMF2 function is different in that its primary purpose is the detection of DTMF tones for dialing purposes along with measuring the timing aspects of the DTMF tone burst. The band splitting filters used with this function are not suitable for accurate level measurements, but include significant dial tone rejection below 450 Hz. As such, the tone generator can used to produce dial tone or other network tones in conjunction with GET_DTMF2. However any tones generated should be lower than 450 Hz to ensure reliable DTMF detection. Also unlike the GET_DTMF function, GET_DTMF2 will not return to the calling routine until the DTMF key has been released, or the time out period expires.

In addition to detecting the DTMF tones, the GET_DTMF2 function will return the start and stop time for the DTMF tone. The timing routines will monitor the signal power level for both the low and high group tones. Once both levels exceeds the minimum DTMF detector level as specified in the SET_DTMF function by 10%, the starting time will be recorded. Likewise, if the signal level for either high and low group tone falls below 90% of the specified level, the ending time will be recorded. The time stamps returned are synchronized to the same timer used for measuring flash or pulse dialing times. The time stamps are returned in units of seconds and can range from 0 to approximately 44739 seconds. The timer can be reset to zero by calling the RESET_TIMER function. The internal routines that monitor the low and high group levels do so only once every millisecond. As such, this forms the minimum resolution of the DTMF timing. For best accuracy, the minimum DTMF level should be set to approximately 10 dB lower than the DTMF level. This minimizes timing errors due to unequal signal integration rise and fall times for the level meters. It's important to note that the timing aspect of this function is independent of the actual DTMF detection routine. As long as the tones detected exceed the minimum DTMF level by 10%, a starting time stamp will be recorded. This is regardless of the frequency accuracy of the DTMF tone.

Using GET_DTMF2:

Generally, the GET_DTMF2 routine will be used to receive a string of DTMF digits from a CPE under test. Before the function can detect any DTMF digits, it must be called with the ACTION parameter set to 1. This initializes the dial tone reject band splitting filters. The DURATION parameter is ignored in this case and the function will return when the filters have been setup. Once this is done, the DTMF digits can be received by calling the function with the ACTION parameter set to 2. The DURATION parameters specifies the maximum time to wait for the release of the digit. To received more than 1 digit, repeat calls to GET_DTMF2 with ACTION set to 2 until all the digits have been detected.

When no more digits are to be received, call the GET_DTMF2 routine with the ACTION parameter set to 0. This resets the internal filters back to their pervious settings. The reason behind having to setup the band splitting filters manually is that if they are enabled and disabled automatically on each call to GET_DTMF2, the transients created can falsely trigger the timing level meters. The low group filter with the dial tone rejection has a fairly length settling time associated with it. It is good practice to insert a 100 msec delay between calling GET_DTMF2 with ACTION set to 1 and subsequent calls with ACTION set to 2 (receive the digit). Note that the SET_ACFILTER function should not be called while using the GET_DTMF2 function with ACTION set to 2.

Function Prototype:

```
int  GET_DTMF2  (  int device,  int action,  int duration,  float *timestart,
                                float *timestop,  int *digit  )
```

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

ACTION — The action parameter determines what operations the GET_DTMF2 function will perform. The function should be called initially with the action parameter value of 1 to setup to the internal band splitting filters. To receive DTMF digits, the value of 2 is used for the action value. Once all the digits have been detected, call GET_DTMF2 with action set to zero in order to return various internal parameters back to their previous settings.

ACTION = 1	Prepare to receive DTMF digits
ACTION = 2	Receive a single DTMF digit
ACTION = 0	Finished receiving DTMF digits

DURATION — The duration parameter is a 16 bit integer value that represents the amount of time to wait for a DTMF digit. The units for this time duration are in milliseconds and the value must be between 1 and 32767 msec.

TIMESTART — The timestart parameter is a pointer to a 32 bit floating point value. This function will set that value to the starting time of the detected DTMF digit. The time stamp is returned in units of seconds in the range of 0 to 44739 seconds. If the level for both the high and low tone group did not exceed the minimum DTMF level (set by SET_DTMF) by 10%, then this value will be set to zero.

TIMESTOP — The timestop parameter is a pointer to a 32 bit floating point value. This function will set that value to the ending time of the detected DTMF digit. The time stamp is returned in units of seconds in the range of 0 to 44739 seconds. The stop time is determined by when either the low or high group tone level has fallen below 90% of the minimum DTMF level. If the tones have

not fallen below this level before the time out period specified in the DURATION parameter, then a value of 0 will be returned for timestop.

DIGIT — The digit parameter is a pointer to a 16 bit integer value. This function will set that value to an integer value representing the decoded DTMF tone. If no DTMF tone was detected, then the value returned will be -1. All of the possible values for the DTMF digit are as follows:

DTMF 0	DIGIT value = 0
DTMF 1	DIGIT value = 1
DTMF 2	DIGIT value = 2
DTMF 3	DIGIT value = 3
DTMF 4	DIGIT value = 4
DTMF 5	DIGIT value = 5
DTMF 6	DIGIT value = 6
DTMF 7	DIGIT value = 7
DTMF 8	DIGIT value = 8
DTMF 9	DIGIT value = 9
DTMF *	DIGIT value = 10
DTMF #	DIGIT value = 11
DTMF A	DIGIT value = 12
DTMF B	DIGIT value = 13
DTMF C	DIGIT value = 14
DTMF D	DIGIT value = 15
(none)	DIGIT value = -1

Function Error Codes:

0	No Error
167	FSK or Ringing is Active
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: SET_RING

Description:

The SET_RING command is used to specify the frequency and level of the ringing generator. The ringing generator is initiated with the START_RING command. If ringing is already being generated, then using the SET_RING function will not change the frequency or level of the current ringing. To update the ringing to the new settings, call the START_RING function again.

Function Prototype:

int SET_RING (int device, float freqR, float levelR)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.
FREQR	The freqR parameter is a 32 bit floating point value that represents the frequency of the ringing generator. The frequency is specified in units of Hz, and must be between the limits of 10.0 to 600.0 Hz.
LEVELR	The levelR parameter is a 32 bit floating point value that represents the level of the ringing generator. The level is specified in units of Vrms and must be between the limits of 0.0 to 80.0 Vrms.

Function Error Codes:

0	No Error
140	Ringing Frequency Outside Valid Range
141	Ringing Level Outside Valid Range
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: SET_RINGPATTERN

Description:

The SET_RINGPATTERN function is used to define a ringing on/off pattern for the ringing generator. The ring on time is specified by the duration parameter, while the interval parameter specifies the time between the ringing burst starts. The interval time must be greater than the duration time, otherwise a non-zero error code will be returned by this function.

Function Prototype:

int SET_RINGPATTERN (int device, long duration, long interval)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.
DURATION	The duration parameter is a 32 bit long integer value that specifies the ringing duration time. The ringing duration is given in units of milliseconds and can range between 1 and 1,000,000 msec.
INTERVAL	The interval parameter is a 32 bit long integer value that specifies the time interval between the start of the ringing bursts. The interval time is given in units of milliseconds and can range between 1 and 1,000,000 msec.

Function Error Codes:

0	No Error
142	Ringing Duration Outside Valid Range
143	Ringing Interval Outside Valid Range
144	Ringing Duration Greater or Equal to Interval
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: START_RING

Description:

The START_RING function is used to start or update the settings of the ringing generator. The ringing generator can be put into a continuous or pulsed mode of operation as determined by the pattern parameter. Once started, the ringing generator can be pre-programmed to turn off at a specified time or run continuously depending on the duration parameter. The time at which the function returns control to the calling routine can be immediately after starting the ringing, or only when the ringing has been completed. This is set by the wait parameter. Calling this function will stop any tones, FSK data, or noise currently being generated. Also, this routine does not check to see if the CPE is on-hook. If the CPE is off-hook, this routine will still start the ringing generator. However, the TSPC hardware will not produce any ringing signals when the CPE is off-hook.

Function Prototype:

int START_RING (int device, int pattern, long duration, int wait)

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

PATTERN — The pattern parameter is a 16 bit integer value that controls if the ringing generator will be producing a ringing pattern or a continuous ring. If the value is non-zero, then the ring generator will produce an on/off pattern, following the duration and interval set by the SET_RINGPATTERN function. Otherwise, the ringing will be continuous.

DURATION — The duration parameter is a 32 bit long integer value that specifies how long the ringing will be generated for. The duration is given in units of milliseconds and must be between the values of 1 and 1,000,000 msec. For this duration, the ringing can be either continuous or pulsed as determined by the pattern parameter. If the pattern parameter is non-zero, then the ringing generator will produce ring bursts for the total duration specified. There is a maximum of up to 32767 ringing bursts during the entire period. A special case exists when the duration parameter contains the value of -1. In this case, the ringing generator will stay on indefinitely (continuous, or pulsed).

WAIT — The wait parameter is a 16 bit integer value that determines when the AiTspc DLL will return control to the calling program. A wait value of zero, will cause the AiTspc DLL to start the ringing generator and then return control back to the calling

routine. However, if the wait value is non-zero, then the AiTspc DLL will not return control back to the calling routine until the ringing has been generated for the time specified by the duration parameter. If the duration parameter is set to -1, then the AiTspc DLL will always return control immediately after starting the ringing, regardless of the wait value.

Function Error Codes:

0	No Error
146	Invalid Ringing Duration Value
147	Too Many Ring Pattern Cycles
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: STOP_RING

Description:

The STOP_RING function can be used to stop the ringing generator. If the ringing generator is active, then this function will stop it immediately, otherwise this function has no effect.

Function Prototype:

int STOP_RING (int device)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: GET_RINGSTATUS

Description:

The GET_RINGSTATUS function will return the state of the ringing generator. There are only two possible states for the ringing generator, being either active or inactive.

Function Prototype:

int GET_RINGSTATUS (int device, int *active)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.
ACTIVE	The active parameter is a pointer to a 16 bit integer value. The function will set this value to -1 if the ringing generator is currently active. Otherwise, a value of 0 will be returned.

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: SET_ACFILTER

Description:

The SET_ACFILTER function is used to control whether any filters are used in front of the AC level meter. There are two banks of filters that can be engaged before the meter. The first bank of filters can be set to implement a 2nd order Butterworth low pass filter or high pass filter. The second bank is used to implement DTMF band splitting filters or a C-message weighted filter. The settings of the two banks is controlled by the value of the filter1 and filter2 parameters. Note that if filter1 is set to the value of 3, thus selected both low pass and high pass filters, then the value of filter2 will be forced to 0 (no filter), regardless of what parameter value is passed to it.

Function Prototype:

int SET_ACFILTER (int device, int filter1, int filter2)

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

FILTER1 — The filter1 parameter is a 16 bit integer value that represents which, if any, of the second order filters are engaged. The range of this parameter is from 0 to 3, with each value corresponding to the following filter selections.

- FILTER1 = 0 No second order LPF or HPF
- FILTER1 = 1 Second order Butterworth LPF (fc = 7000 Hz)
- FILTER1 = 2 Second order Butterworth HPF (fc = 150 Hz)
- FILTER1 = 3 Both Second order LPF and HPF

FILTER2 — The filter2 parameter is a 16 bit integer value that represents which, if any, of the following filter types are engaged. The range of this parameter is from 0 to 3, with each value corresponding to the following filter selections.

- FILTER2 = 0 No filters selected
- FILTER2 = 1 DTMF low group filter (LPF)
- FILTER2 = 2 DTMF high group filter (HPF)
- FILTER2 = 3 C-Message weighted filter

Function Error Codes:

0	No Error
170	Filter1 Outside Valid Range
171	Filter2 Outside Valid Range
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: SET_ACSPEED

Description:

The SET_ACSPEED function is used to set the integration time of the AC level meter. Three possible settings are available that can be used to match the optimum integration time to the signal being measured. The slowest setting will give accurate level readings down to 10 Hz; however the settling time will be correspondingly longer than the other settings. For signals not lower than 100 Hz, the medium setting provides accuracy to within 0.1 dB, but with a much faster response time. Likewise, the fast setting is targeted to measuring signals above 500 Hz with an even shorter settling time.

Function Prototype:

int SET_ACSPEED (int device, int speed)

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

SPEED — The speed parameter is a 16 bit integer value that represents the integration speed of the AC level meter. The valid range for this parameter is between 0 and 2, corresponding to the following useful frequency ranges:

SPEED = 0	Slow (use for frequencies down to 10 Hz)
SPEED = 1	Medium (use for frequencies down to 100 Hz)
SPEED = 2	Fast (use for frequencies down to 500 Hz)

The higher the speed setting the faster the AC level meter will settle to within 0.1 dB of the RMS signal level. However, if using a fast settling speed for measuring low frequency signals, the level meter will not settle to within 0.1 dB and reading can fluctuate in level.

Function Error Codes:

0	No Error
175	Speed Value Outside Valid Range
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: GET_ACLEVEL

Description:

The GET_ACLEVEL function is used to read the current level at the AC level meter. The value return is in units of Vrms. Use the SET_ACFILTER function to select any high or low pass filters before the level meter, and the SET_ACSPEED function will set the integration speed of the level meter.

Function Prototype:

int GET_ACLEVEL (int device, float *level)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.
LEVEL	The level parameter is pointer to a 32 bit floating point value. The current reading of the AC level meter is returned to this value in units of Vrms.

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: GET_ACTHDN

Description:

The GET_ACTHDN function will return the measured total harmonic distortion plus noise (THD+N) of the signals at either the telephone interface or the BNC input. The signal source is set by the SET_AUDIO function. The THD+N value is computed as a percentage of the ratio of the noise plus distortion over the signal plus noise plus distortion. The range of the measurement is between 0 to 100%. In order to perform the measurement an internal notch filter is used to block the desired signal, thus leaving only the harmonic and noise components, which can then be measured. The center frequency of the notch filter must be specified as part of the function call. In addition to the notch filter, optional low pass and high pass filters can be added to the signal path in order to reduce the level of any undesired signals.

The notch filter has an attenuation of approximately 80 dB at the specified frequency. At a 1 percent frequency offset, the attenuation is reduced to approximately 50 dB. As such, it is important to accurately specify the notch frequency for low distortion measurements. The minimum distortion readings is dependent on the notch filter depth as well as the integration time used by the level meters. In order to prevent excessively long measurement times, the minimum distortion measurements are approximately 0.1% between 1 kHz to 10 kHz, and 0.3% between 50 Hz to 100 Hz. The measurement times are approximately 1 second for frequencies between 1 kHz to 10 kHz, 2 seconds at 100 Hz, and 3 seconds at 50 Hz.

Function Prototype:

int GET_ACTHDN (int device, float NotchFreq, int filter, float *THDNlevel)

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

NOTCHFREQ — The NotchFreq parameter is a 32 bit floating point value that represents the frequency of measurement notch filter. The value is specified in units of Hz, and must be between the limits of 50 to 10,000 Hz.

FILTER — The filter parameter is a 16 bit integer value that represents which, if any, of the second order filters are engaged. The range of this parameter is from 0 to 3, with each value corresponding to the following filter selections.

FILTER = 0	No second order LPF or HPF
FILTER = 1	Second order Butterworth LPF (fc = 7000 Hz)
FILTER = 2	Second order Butterworth HPF (fc = 150 Hz)

FILTER = 3 Both Second order LPF and HPF

THDNLEVEL The level parameter is pointer to a 32 bit floating point value. The THD+N measurement returned by the pointer is in units of percent between the range of 0 to 100.

Function Error Codes:

0	No Error
190	NotchFreq Parameter Outside Valid Range
191	Filter Parameter Outside Valid Range
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: SET_AUDIO

Description:

The SET_AUDIO function is used to control the various audio signal routing switches available. This allows for routing the AC signals between the telephone interface, BNC input and output connectors, and the tone generators and level meters in various configurations. The BNC connectors can be configured to monitor the tip and ring levels, act as a loop-thru for the telephone interface in the transmit or receive paths, or as an independent audio input and output.

Function Prototype:

int SET_AUDIO (int device, int TRsrc, int BNCsrc, int metersrc)

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

TRSRC — The TRsrc parameter is a 16 bit integer value that controls the AC signal source for the telephone interface on the TSPC. If the TRscr value is zero, then the signal source for the telephone interface will originate with the tone, FSK, noise, and DTMF generators. A non-zero value will set the telephone interface to use the signal present at the BNC input connector for its signal source.

BNCSRC — The BNCsrc parameter is a 16 bit integer value that controls the source of the AC signal present at the BNC output connector. If the BNCsrc value is set to zero, then the signals at the BNC output will come from the tone, FSK, noise, and DTMF generators. Otherwise, the signals at the BNC output will represent the signal present on the tip and ring leads of the telephone interface.

METERSRC — The meterscr parameter is a 16 bit integer value that controls the signal source for the AC level meter and the DTMF receiver. If set to zero, the level meter and DTMF receiver will measure signals present on the tip and ring leads of the telephone interface. Otherwise, the level meters will measure the signal present at the BNC input connector.

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: SET_DOUT

Description:

The SET_DOUT function is used to control the logic level of the two general purpose output pins located at the rear DB9 connector on the TSPC. The output A and B pins will remain at a low state regardless of this function unless the output port is enabled by grounding the port enable pin. For a complete description of the auxiliary digital input and output port, see section 5: Auxiliary Digital Inputs and Outputs.

Function Prototype:

int SET_DOUT (int device, int outA, int outB)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.
OUTA	The outA parameter is a 16 bit integer value that controls the output logic level of digital output A. A non-zero value will set output A to a high logic level, while a zero value will clear output A to a low logic level.
OUTB	The outB parameter is a 16 bit integer value that controls the output logic level of digital output B. A non-zero value will set output B to a high logic level, while a zero value will clear output B to a low logic level.

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: GET_DIN

Description:

The GET_DIN function is used to read the three digital inputs present at the DB9 connector at the rear of the TSPC. The function will return the logic level at the output port enable pin, input A pin, and the input B pin. For a complete description of the auxiliary digital input and output port, see section 5 Auxiliary Digital Inputs and Outputs.

Function Prototype:

int GET_DIN (int device, int *portEN, int *inputA, int *inputB)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.
PORTEN	The portEN parameter is a pointer to a 16 bit integer value. This value will be set to 0 if the port enable pin is at a low level (outputs enabled), or a value of -1 if the pin is at a high logic level.
INPUTA	The inputA parameter is a pointer to a 16 bit integer value. This value will be set to 0 if the input A pin is at a low logic, or a value of -1 if the pin is at a high logic level.
INPUTB	The inputB parameter is a pointer to a 16 bit integer value. This value will be set to 0 if the input B pin is at a low logic, or a value of -1 if the pin is at a high logic level.

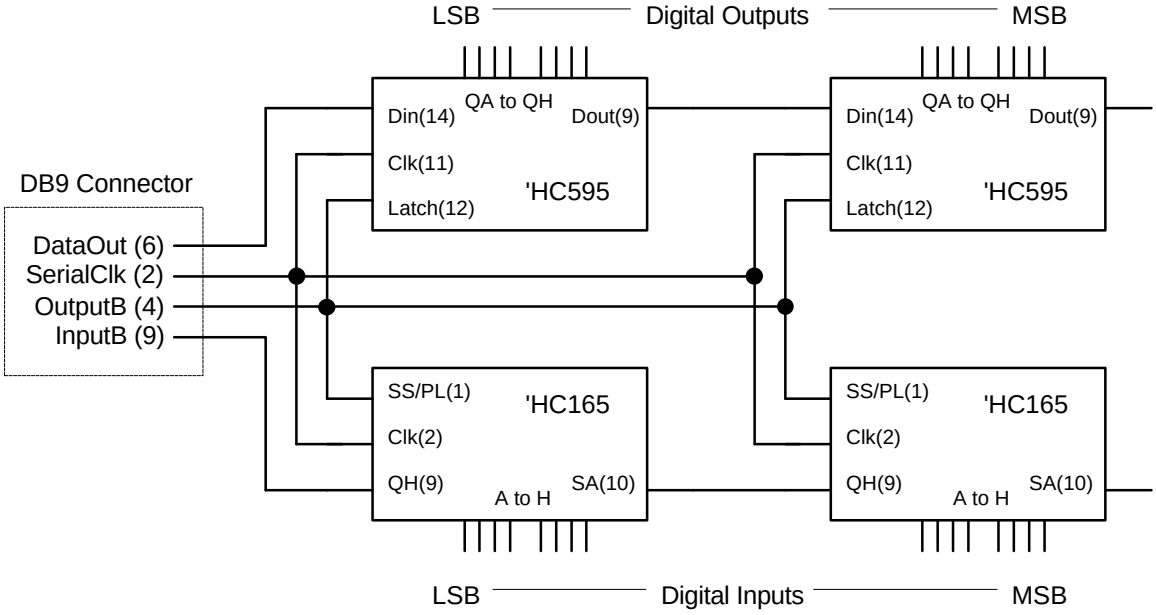
Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: XFER_SERIALDATA

Description:

The XFER_SERIALDATA provides for expanded digital I/O capability on the TSPC. The function will serially send and receive 16 bits of data over a synchronous serial bus using the DB9 connector at the rear of the hardware card. External serial to parallel converters can then be used to expand the serial data into 16 digital outputs and 16 digital inputs. Common logic IC's can be used to perform the serial to parallel conversion. The following figure shows an implementation using 74HC595 and 74HC165 devices.



Four pins are required from the rear DB9 connector. These are termed; Data Out (pin 6), Serial Clock (pin 2), Output B (pin 4), and Input B (9). Calling the XFER_SERIALDATA function will cause 32 clock pulses to be generated on the serial clock pin. The first 16 clock cycles are used to shift the new output data into the 'HC595 serial shift registers. Shortly after the 16th clock cycle, Output B goes high to latch the data to the internal output registers. At this time the new data will appear at the outputs. At that same instant, the input data present at the 'HC165's will be sampled. The next 16 clock cycles are used to read the data stored in the 'HC165's one bit at a time. After the 32nd clock cycle, Output B will go low, and the entire transfer will be completed.

If only 8 digital outputs are needed, then the second 'HC595 can be removed. Likewise, for only 8 digital inputs, the second 'HC165 can be removed. It is also possible to have only digital outputs or only digital inputs by removing the unused 'HC595's or 'HC165's.

Note: Care should be taken that excessively long cables are not connected to the DB9 connector, as this can degrade the signal integrity of the serial data

signals. The entire transfer takes approximately 125 microseconds. The clock cycle duration is no faster than 1.5 microseconds, with the high logic level being no shorter than 300 nsec. The data setup time is also approximately 300 nsec.

Function Prototype:

int XFER_SERIALDATA (int device, int dataout, int *datain)

Function Parameters:

DEVICE — The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.

DATAOUT — The dataout parameter is a 16 bit integer value. The contents of this integer will be transferred serially (msb first) over the synchronous serial port.

DATAIN — The datain parameter is a pointer to a 16 bit integer value. This value will contain the results of reading 16 bits of data from the synchronous serial port (lsb first).

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Function: WAIT

Description:

Calling the WAIT function will not return program control to the calling routine until the specified time interval has elapsed. The time to wait is specified in units of milliseconds from 0 to a maximum of 32767. The processor speed of the PC will have an influence on the accuracy of the time interval to a slight degree. A low speed PC will extend the time interval slightly due to communication delays with the TSPC. With a reasonable fast 486 class processor or a low end Pentium class processor, the timing accuracy will be within a couple of milliseconds. The faster the PC, the more accurate the timing of the WAIT command.

Function Prototype:

int WAIT (int device, int delay)

Function Parameters:

DEVICE	The device parameter is used by all the functions to specify which TSPC is being referred to. Its value can range from 0 to 3. See the function INIT for more information on the DEVICE parameter value selection.
DELAY	The delay parameter specifies the time interval to wait in units of milliseconds. The valid range of this value is from 0 to 32767. Negative values will be interpreted as a delay time of 0.

Function Error Codes:

0	No Error
256	Invalid Device Number
257 to 263	TSPC Communication Error

Section 4 Error Code Reference

Every one of the functions in the AiTspc DLL returns an integer value that represents an error code for the function. A returned value of zero indicates that no problems or unusual circumstances were encountered. A non-zero value will indicate some sore of error, defined by the specific value returned. The following list shows all of the possible error codes and the meaning behind them.

0 **No Error**
The function call proceeded without error

100, 101 **No TSPC Found with Specified Device Number**
The INIT function was unable to locate a TSPC with the given device number. Either no TSPC is present in the PC, or the base I/O address is set to a different value, or the TSPC is faulty.

102 **TSPC SRAM Check Failure**
The INIT function has found a TSPC with the given device number, but the TSPC has failed a check of its SRAM. The initialization was not completed.

105 **TSPC Revision Incompatible with AiTspc DLL**
The INIT function has found a TSPC with the given device number, but the TSPC is incompatible with the AiTspc DLL. The AiTspc DLL can only work with TSPC's that are part of either the AI-150 Caller ID simulator package or the AI-410.

110 **Line Voltage Outside Valid Range**
The line voltage parameter passed to the SET_TELINT function is either higher or lower than the limits of 20.0 Volts to 52.0 Volts.

111 **Loop Current Outside Valid Range**
The loop current parameter passed to the SET_TELINT function is either higher or lower than the limits of 20.0 mA to 40.0 mA.

112 **Line Impedance Outside Valid Range**
The line impedance parameter passed to the SET_TELINT function is outside the range of 0 to 2.

120 **Tone Frequency Outside Valid Range**
Either the tone A frequency parameter or the tone B frequency parameter is set to something outside the range of 10 Hz to 20,000 Hz.

121 **Tone Level Outside Valid Range**
Either the tone A level parameter or the tone B level parameter is set to something outside the valid range of 0 Vrms to 4.0 Vrms

122 **Duration Outside Valid Range**
The duration parameter for the SET_TONEPATTERN function is outside the valid range of 1 msec to 1,000,000 msec.

123 **Interval Outside Valid Range**
The interval parameter for the SET_TONEPATTERN function is outside the valid range of 1 msec to 1,000,000 msec.

124 **Duration Greater or Equal to Interval**
The duration parameter for the SET_TONEPATTERN function is equal or greater than the interval pattern. In order to define a valid tone burst pattern, the duration of the tone must always be less than the interval. The interval defines the time between tone starts.

125 **Ringing Generator is Active**
The tone generator can not be started because the ring generator is currently active. Stop the ring generator first, before starting the tone generator.

126 **Invalid Tone Duration Value**
The tone duration parameter for the START_TONE function must be a value in the range of 1 msec to 1,000,000 msec, or the value of -1 (which indicates an indefinite duration time).

127 **Too Many Tone Pattern Cycles**
The START_TONE function was attempting to generate a tone pattern; however, the number of tone bursts exceeded 32767, which is the maximum number of tone bursts possible. The number of tone bursts can be calculated by taking the total duration and dividing it by the tone burst interval.

130 **FSK Frequency Outside Valid Range**
Either the mark tone frequency parameter or the space tone frequency parameter is set to something outside the range of 10 Hz to 20,000 Hz.

131 **FSK Level Outside Valid Range**
Either the mark tone level parameter or the space tone level parameter is set to something outside the valid range of 0.0 Vrms to 4.0 Vrms.

132 **FSK Baud Rate Outside Valid Range**
The baud rate parameter value for the SET_FSK function is outside the valid range of 100.0 bps to 5000.0 bps.

135 **Ringing Generator is Active**
The FSK modulator can not be started because the ring generator is currently active. Stop the ring generator first, before starting the FSK modulator.

136 **Invalid FSK Mode**
The mode parameter value for the START_FSK function is outside the valid range of 0 to 2.

137 **Invalid Number of Bits**
The number of data bits specified in the bits parameter for the START_FSK function is outside the valid range of 1 to 4096.

140 **Ringing Frequency Outside Valid Range**
The ringing frequency parameter for the SET_RING function is outside the valid range of 10.0 Hz to 600.0 Hz.

141 **Ringing Level Outside Valid Range**
The ringing level parameter for the SET_RING function is outside the valid range of 0.0 Vrms to 80.0 Vrms.

142 **Ringing Duration Outside Valid Range**
The duration parameter for the SET_RINGPATTERN function is outside the valid range of 1 msec to 1,000,000 msec.

143 **Ringing Interval Outside Valid Range**
The interval parameter for the SET_RINGPATTERN function is outside the valid range of 1 msec to 1,000,000 msec.

144 **Ringing Duration Greater or Equal to Interval**
The duration parameter for the SET_RINGPATTERN function is equal or greater than the interval pattern. In order to define a valid ringing burst pattern, the duration of the tone must always be less than the interval. The interval defines the time between ringing starts.

146 **Invalid Ringing Duration Value**
The ringing duration parameter for the START_RING function must be a value in the range of 1 msec to 1,000,000 msec, or the value of -1 (which indicates an indefinite duration time).

147 **Too Many Ring Pattern Cycles**
The START_RING function was attempting to generate a ringing pattern; however, the number of ringing bursts exceeded 32767, which is the maximum number of ringing bursts possible. The number of ringing bursts can be calculated by taking the total duration and dividing it by the ringing burst interval.

150 **Invalid DTMF Digit**
The digit parameter for the SEND_DTMF function is outside the valid range of 0 to 15.

151 **DTMF level Outside Valid Range**
The level parameter for the SEND_DTMF function is outside the valid range of 0.0 Vrms to 4.0 Vrms.

153 **DTMF Duration Outside Valid Range**
The duration parameter for the SEND_DTMF function is outside the valid range of 1 to 32767 msec.

155 **Ringing Generator is Active**
The DTMF generator can not be started because the ring generator is currently active. Stop the ring generator first, before starting the DTMF generator.

160 **Frequency Tolerance Outside Value Range**
The frequency tolerance parameter for the SET_DTMF function is outside the valid range of 0.0 to 3.0 %.

161 **Minimum Level Outside Valid Range**
The minimum level parameter for the SET_DTMF function is outside the valid range of 0.0 Vrms to 4.0 Vrms.

165 **Duration is Outside Valid Range**
The duration parameter for the GET_DTMF parameter is outside the valid range of 1 to 32767 msec.

166 **Tone or FSK or Ringing is Active**
The GET_DTMF function can not operate since either the tone generator, FSK modulator, or ringing generator is currently active. Stop all the signal generators before calling the GET_DTMF function.

167 **FSK or Ringing is Active**
The GET_DTMF2 function can not operate since either the FSK modulator, or ringing generator is currently active. Stop the FSK modulator or ringing generator before calling the GET_DTMF2 function.

170 **Filter1 Outside Valid Range**
The filter1 parameter value for the SET_ACFILTER function is outside the valid range of 0 to 3.

171 **Filter2 Outside Valid Range**
The filter2 parameter value for the SET_ACFILTER function is outside the valid range of 0 to 3.

175 **Speed Value Outside Valid Range**
The speed parameter value for the SET_ACSPEED function is outside the valid range of 0 to 2.

180 **Noise Level Outside Valid Range**
The noise level parameter for the SET_NOISE function is outside the valid range of 0.0 Vrms to 1.0 Vrms.

190 **Notch Frequency Parameter Outside Valid Range**
The notch frequency parameter for the GET_ACTHDN function is outside the valid range of 50 Hz to 10,000 kHz.

191 **Filter Parameter Outside Valid Range**
The filter parameter for the GET_ACTHDN function is outside the valid range of 0 to 3.

256 **Invalid Device Number**
The device number parameter value for all of the functions is outside the valid range of 0 to 3.

257-263 **TSPC Communication Error**
An error occurred during the communication process with the TSPC. This can be due to either no TSPC present at the

selected device number, or a faulty TSPC, or a bus contention problem on the PC.

Section 5 Using the DLL's and DOS Library

Using the 16 Bit Windows DLL

Relevant Files
The various files located in the DDP1100 application directory are used by the sample application, as well as all the files needed for 32 bit DLL or DOS application development.

If using the 16 bit DLL for custom applications, the following files are relevant.

AiTspc.dll
AiTspc.lib
AiTspc.h
AiTspc.bas

DDL Files
For 16 bit DLL applications, two DLL files must be located in the \Windows\System directory. These are:

AiTspc.dll
Bc450rtl.dll

Normally, during the installation of the DDP1100 demo program, both of these files are automatically added to the \Windows\System directory.

Header or Declaration Files
Function prototype files are included for both C or Visual Basic development. For developing custom applications in 16 Bit Window C, include the AiTspc.lib file in the project file list, while the AiTspc.h file can be included in any module accessing the DLL functions. For Visual Basic applications, simply include the AiTspc.bas file into the project file list. This file provides all of the declarations needed by Visual Basic.

Using the 32 Bit Windows DLL

Relevant Files
The various files located in the DDP1100 application directory are used by the sample application, as well as all the files needed for 16 bit DLL or DOS application development.

If using the 32 bit DLL for custom applications, the following files are relevant.

AiTspc32.dll

AiTspc32.lib
AiTspc32.h
AiTspc32.bas

DDL Files
For 32 bit DLL applications, three DLL files must be located in the \Windows\System directory. These are:

AiTspc32.dll
cw3220.dll
win95io.dll

Normally, during the installation of the DDP1100 demo program, these files are automatically added to the \Windows\System directory.

Header or Declaration Files
Function prototype files are included for both C or Visual Basic development. For developing custom applications in 32 Bit Window C, include the AiTspc32.lib file in the project file list, while the AiTspc32.h file can be included in any module accessing the DLL functions. For Visual Basic applications, simply include the AiTspc32.bas file into the project file list. This file provides all of the declarations needed by Visual Basic.

Calling the 32 Bit Functions
The function reference information provided in this document applies to the 16 bit Windows DLL only. For accessing the 32 bit DLL, it is important to note:

1) All function names remain the same.
2) The number of parameters in each function remain the same
3) The purpose of the parameters in each function remain the same
4) The value limits of the parameters in each function remain the same
5) All 16 bit integers parameters must be replaced with 32 bit integers
6) All pointers to 16 bit integers are must be replaced with pointers to 32 bit integers
7) All functions return 32 bit integers instead of 16 bit integers

Using the DOS Function Library

Relevant Files
The various files located in the DDP1100 application directory are used by the sample application, as well as all the files needed for 16 bit DLL or 32 bit DLL application development.

If using the DOS function library for custom applications, the following files are relevant.

AiTsDOS.lib
AiTsDOS.h

Creating Applications
The AiTsDOS.lib file was created with Borland Turbo C version 3.0 (DOS). To use the library file in an application, include the AiTsDOS.lib file in the program

file list. The AiTsDOS.h file can then be included in any module that requires access to the functions. The memory model under which the library was created is the “large” memory model. This allows for multiple code segments and multiple data segments.

Calling the Functions

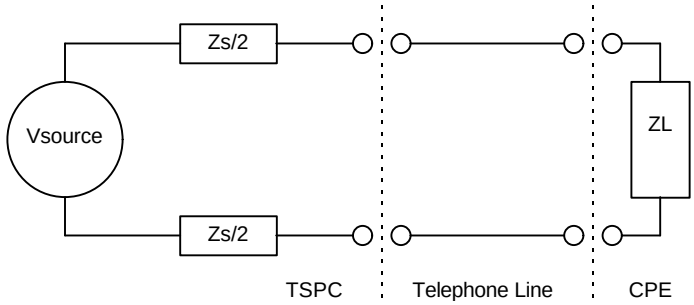
The functions contained within the DOS library are identical in all respects to the functions documented here.

■ Section 6 Additional Information

Telephone Line Signal Levels

Signal levels are a common source of confusion with most signal generation equipment. This section aims to explain how signal levels are calculated within the program.

It is important to understand how the Telephone Signal Processing Card (TSPC) delivers its AC signals to the CPE under test. The figure below shows a simplified version of the AC signal path from the TSPC to the CPE. The TSPC consists of basically a programmable AC voltage source followed by two impedance's. The two impedance's combined represent the total source impedance (Zs) of the TSPC which can be set to 600 ohms, 900 ohms, or a complex impedance. The CPE under test is represented by the impedance ZL.



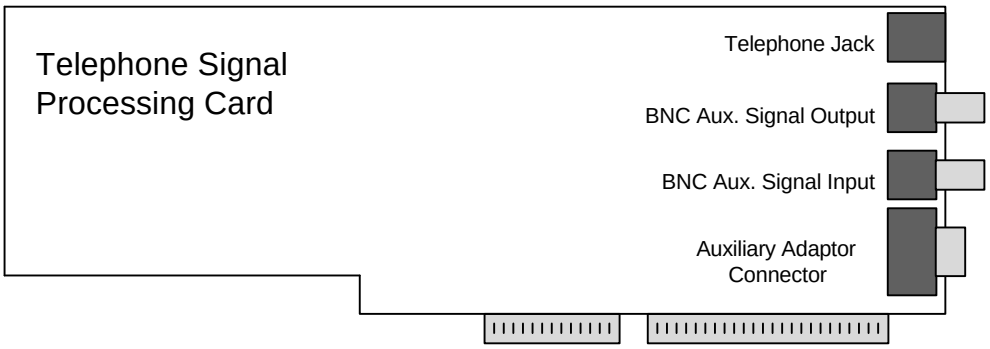
All signal level parameters passed to the AiTspc DLL refer to the level of Vsource in the units of Volts RMS. As such, the signal level measured at the telephone line is only equal to the specified level with the telephone unterminated. In the case that the telephone line is terminated by a CPE of impedance ZL, then the signal level present at the tip and ring interface is given by:

$$Vcpe = Vsource * \frac{ZL}{Zs + ZL}$$

where: Vcpe is the voltage level at the CPE in Vrms
Vsource is given in Vrms
ZL is the load impedance of the CPE
Zs is the source impedance setting of the TSPC

External BNC Signal Levels

The two BNC connectors on the rear of the Telephone Signal Processing Card (TSPC) can be used to monitor various signals to and from the CPE, and to inject signals directly to the CPE. The figure below shows the BNC signal output connector being located below the telephone jack, and just above the BNC signal input connector.



BNC Output Connector:
The auxiliary signal output has an output impedance of 600 ohms, and can either be set to monitor the signal tones, FSK modulator, and noise signals generated by the TSPC, or the signals being received from the CPE connected via the telephone jack. When monitoring the signals generated by the TSPC, the output level present at the BNC connector (with a high impedance load) is equal to one quarter (12 dB less) of the AC voltage source driving the telephone interface. For example, setting the Tone Generator for an output level of 1.0 Vrms will cause the level at the BNC output connector will be 0.25 Vrms into a high impedance.

If monitoring signals generated from the CPE, the signal level present at the BNC connector (with a high impedance load) is 6 dB less than the voltage level across the CPE. So if a level of 1 Vrms is present at the CPE terminals, routing this signal to the BNC output connector will give it a level of 0.50 Vrms.

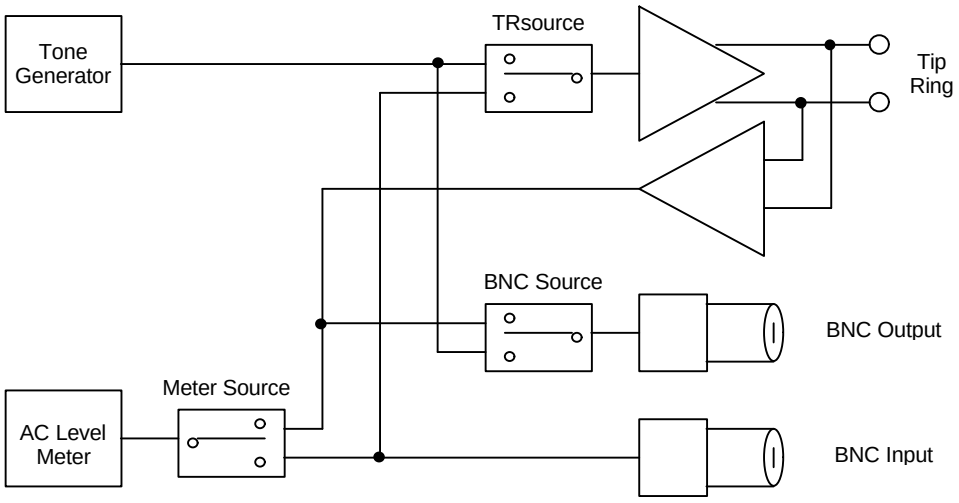
BNC Input Connector:
The auxiliary signal input has an input impedance of approximately 100 kohms. Signals injected in this input can be routed to the CPE, or to the level analyzer within the TSPC. If set to route the input signal to the CPE, the voltage generated by the AC voltage source driving the telephone interface will be four times higher than the voltage level injected at the BNC input connector.

For example, if the CPE is disconnected (open circuit on tip and ring), the voltage at the telephone line would be 12 dB more than the voltage present at the BNC input connector. If the CPE presents a 600 ohm load and the telephone interface impedance (Zs) is 600 ohms, the voltage level across the CPE would be 6 dB more than the voltage level at the BNC input connector. Changing the telephone interface's source impedance (Zs) to 900, reduces the voltage level across the CPE by 1.96 dB.

As above, see the section dealing with Telephone Line Signal Levels for a more in-depth look at the signal levels within the telephone interface. If the BNC input is routed to the level analyzer within the TSPC, the voltage read by the level meter will be the signal level at the BNC connector.

Signal Routing

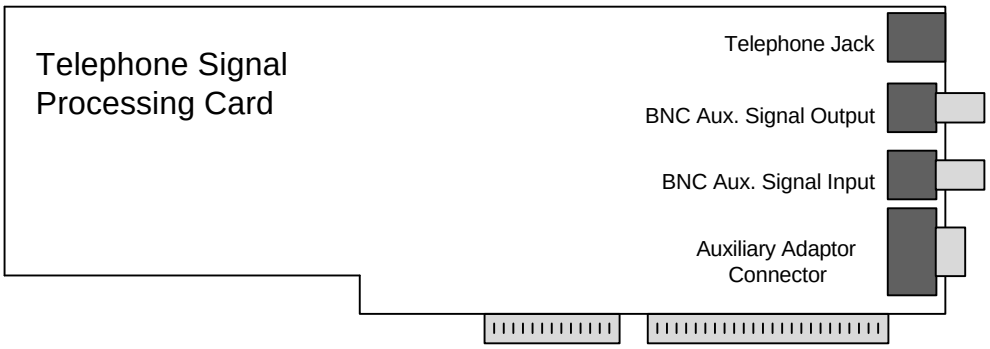
A large degree of flexibility exists in how various audio signals are connected within the TSPC. The BNC output and input can be used to monitor or inject signals at different points along with acting as loop through for the transmit or receive signals to the CPE. The following diagram shows the possible signals paths within the TSPC as well as the switching points.



The signal source for the telephone interface is controlled by the TRsource switch. This switch either selects the on board tone/FSK/noise generator, or the BNC input connector. Likewise, the signal source for the BNC output is controlled by the BNCsource switch. This allows the BNC output to monitor either the signals present at the tip and ring leads, or the signals created by the tone generator. Finally, the third switch selects the signal input to the AC Level meter and the DTMF detect. This is either the telephone interface or the BNC input. The SET_AUDIO function is used to control the position of each of the three signal switches.

Auxiliary Digital Inputs and Outputs

The DB9 pin connector at the rear of the Telephone Signal Processing Card (TSPC) provides access to general purpose digital outputs and inputs that can be used for various functions. Along with the general purpose inputs and outputs, the DB9 connector provides access to a synchronous serial bus that can be used to expand the digital input and output capabilities.



The DB9 female connector allows for the use of up to 2 digital outputs, 2 digital inputs, and a +5V supply from the PC. The digital outputs are only active when Pin 7 is connected to ground. This serves as a output port enable. The outputs are driven from 5V 'HC CMOS logic. Pin 3 is defined as Output A, while pin 4 is Output B. The two available inputs are at pins 8 (Input A) and 9 (Input B). The input voltages should be limited to ground and +5V in order to prevent damage to the internal 'HC CMOS buffers. Both Input A and Input B have 10 kohm resistors to ground in order to prevent floating inputs. The pin assignment for the DB9 connector is as follows:

Pin 1: +5V (can draw up to 0.5A from this pin) (internally fused)
Pin 2: Serial Bus Clock Output ('HC CMOS 5V output)
Pin 3: External Output A ('HC CMOS 5V output)
Pin 4: External Output B ('HC CMOS 5V output)
Pin 5: Ground
Pin 6: Serial Bus Date Output ('HC CMOS 5V output)
Pin 7: Digital Output Enabled (connect to ground to enable outputs)
Pin 8: External Input A ('HC CMOS 5V input)
Pin 9: External Input B ('HC CMOS 5V input)

The SET_DOUT function can be used to set the logic state of Output A and Output B to either a high or low state. Likewise, the GET_DIN function will return the current logic level at Input A and Input B pins, along with returning the logic level at the output port enable (pin 7). If the FSK modulator is started with the MODE parameter equaling 2, then the logic level at Input B determines whether a mark or space frequency will be generated.

The XFER_SERIALDATA function is used to increase the number of digital outputs and inputs to 16. This is done by adding an external circuit to perform serial to parallel conversion as shown in the section explaining the XFER_SERIALDATA function. The four wire serial bus used is composed of a serial clock line (pin 2), data output (pin 6) and input (pin 9) line, and a single control line (pin 4).

Telephone Line Unbalance and Grounding

As with any system of instruments and devices, proper grounding is essential in order to minimize circuit hum and susceptibility to interference. It is important to understand the circuit grounds of all the devices in the test setup to ensure their proper connection.

Like most PC based instruments the TSPC's ground point is referenced to the PC's ground, which should be connected to earth ground. The ground conductor on the BNC input and output and the DB9 connector are at earth ground potential, assuming the computer is properly grounded. The tip and ring leads of the telephone interface have a negative potential with respect to ground depending on the programmed line voltage. At the default setting of 48 volts, the voltages present on tip and ring are approximately -52 volts and -4 volts with respect to the computer's ground.

Care must be taken with CPE's that do not maintain isolation from the telephone interface. Most Caller ID adjunct units fall into this category. This can cause grounding problems when connecting the ground leads of oscilloscopes or other equipment that is normally earth grounded. The most common cause is that the CPE connects either tip or ring to its circuit ground via a bridge rectifier. If the CPE's circuit ground is then connected to earth ground by use of an oscilloscope ground lead, an emulator, or logic analyzer, then this effectively shorts either the tip or ring line to earth ground. The imbalance in loop current is detected by the TSPC and using the GET_HOOKSTATUS function will return a parameter signifying if the line is unbalanced.

If the CPE has an isolated telephone interface, then this condition should not occur, since the CPE's circuit ground is not connected to its telephone interface circuitry. If this is not the case, and external equipment must be connected to the CPE under test that forces its ground to be connected with earth ground, there are a few possible solutions available. In some cases, the telephone interface is not a requirement for the testing situation, and the BNC inputs or outputs can be used, since the BNC's are ground referenced unbalanced signals. If the telephone interface is required, then the grounding "loop" must be broken. This can mean isolating the PC that contains the TSPC, or the test equipment connected to the CPE, or the telephone line itself. Depending on what testing is required, the telephone line can be isolated via an AC coupling transformer and a DC feeding bridge. Of course, the DC supply would have to be isolated from earth ground, but this is generally easier to do than isolating the PC.

Note: Disconnecting the earth ground wire from the computer is both an improper and potentially dangerous method of isolating the computer from the remainder of the test setup. The AC isolation achieved is generally poor due to the large parasitic capacitance's to earth ground. An alternative approach may be the use of a UPS. UPS's are an excellent method of achieving near perfect isolation between circuit grounds.

Modifying the Complex Impedance

The telephone line impedance that is presented to the CPE can be set to ,under program control, to one of three possible values. These are:

i) 600 ohms (real)
ii) 900 ohms (real)
iii) Complex impedance ZR

Where the complex impedance ZR is modeled by the following schematic diagram:

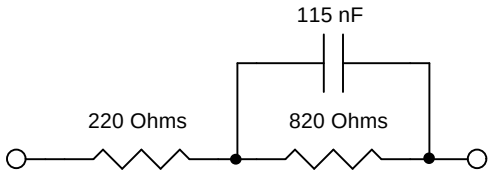


Figure 2. ZR Impedance

However, it is possible to change various components on the TSPC in order to present other custom or user defined impedance's. Figure 3 represents a network of components present on the TSPC that determine the value of the complex line impedance. The line impedance presented to the CPE will be 0.1 times the impedance of the network shown in figure 3. As such, the default complex impedance of ZR can be changed to other values simply by changing values of components R1, R2, C1, C2, and C3.

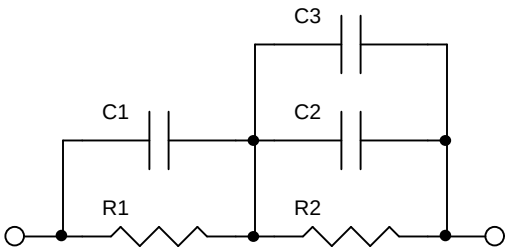


Figure 3. Complex Impedance Network

The default values for the components, in order to present a ZR impedance, are:

R1 = 2200 ohms
R2 = 8200 ohms
C1 = open (not used)
C2 = 10.0 nF
C3 = 1.50 nF

The position of these components on the TSPC is shown in figure 4.

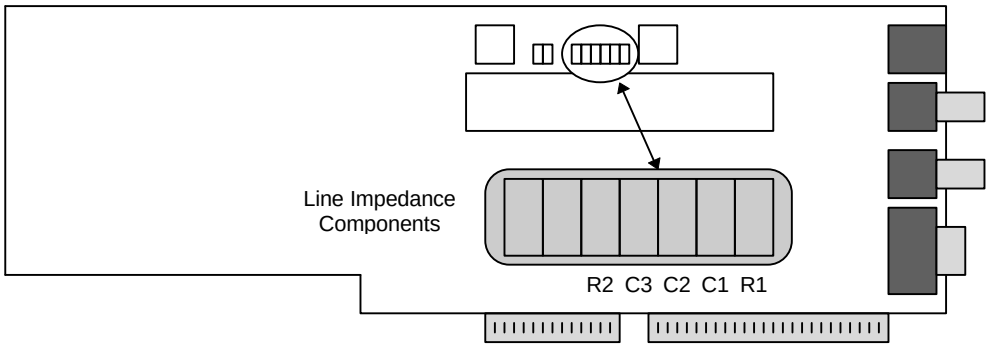


Figure 4. Complex Line Impedance Components

If the component values are to be changed, make sure that standard ESD precautions are taken during the removal of the existing components and soldering of the new components. Of course, the PC should be turned off and the TSPC removed from the PC before any changes are made.

Appendix A General Specifications

Tone Generator

Output Level	-70 dBm to +6 dBm +/- 0.3 dB
Frequency Range	50 Hz to 10 kHz
Flatness	+/- 0.3 dB
THD+N	0.09% C-message
Harmonic Distortion	> 65 dBc
Frequency Accuracy	0.015%

FSK Generator

Output Level	-60 dBm to 0 dBm +/- 0.3 dB
Frequency Range	100 Hz to 5 kHz
Flatness	+/- 0.1dB

Noise Generator

Output Level	-60 dBm to -8 dBm +/- 0.3 dB

Ring Generator

Output Level	0 Vrms to 80 Vrms.
Frequency Range	10 Hz to 600 Hz
Flatness	+/- 0.2 dB
THD+N	0.1%
Frequency Accuracy	0.015%
Ringer Load	5 REN

Telephone Line

Output Impedance	600, 900 +/- 2%
Output Impedance	complex +/- 3%
Loop Voltage	20 - 52 Volts +/- 1V
Loop Current	20 - 40 mA +/- 10%

Level Meter

Level Accuracy	+/- 0.2 dB
Frequency Range	10 Hz to 10 kHz
Flatness	100 Hz to 5 kHz +/- 0.2 dB
Maximum Input	+14 dBm
Residual Noise	<-70 dBmC

Appendix B Technical Support

If you encounter problems while using the AiTspc DLL or the DDP1100 Demo Program, please contact us so that we can provide assistance. You may reach us in any one of the following manners:

Email: techsupport@adventinst.com

In North America:

Tel: (604) 944-4298
Fax: (604) 944-7488

Mail: Advent Instruments Inc.
111 - 1515 Broadway Street
Port Coquitlam, BC V3C6M2
Canada

In Asia:

Tel: (852) 8108-1338
Fax: (852) 2900-9338

Mail: Advent Instruments (Asia) Limited
Room 13, 23rd Floor, Peninsula Tower
538 Castle Peak Road
Kowloon
Hong Kong

Note: Please include the program version, TSPC product code, revision code, option code, and TSPC base I/O address as given in the [HELP] [ABOUT] window in your correspondence.