VOICE/DATA SERVICES MANUAL
(Preliminary)

Specifications Subject to Change.

Preliminary Edition (March 1985) A-09-00911-00-A

CONTENTS

LIST OF TABLES

1 INTRODUCTION

OVERVIEW

INSTALLATION AND CONFIGURATION INFORMATION

The Release Notice for the Voice/Data Services contains instructions on installation, configuration, and other information required for use of the Voice/Data Services software.

WHAT ARE THE VOICE/DATA SERVICES?

The Voice/Data Services are a set of commands, software programs and data files that provides an interface between the hardware and application software programs.

Telephone Server

The Telephone Server system service is an extension of your current operating system. Application programs use the Telephone Server request and procedural interfaces to access the hardware for voice, data, and telephone operations.

Voice Management

The voice management facility of the Telephone Server system service allows an application program to use the analog to digital signal conversion (codec) feature of the hardware to encode or decode voice (such as a telephone conversation) to or from binary data stored on disk. Note that the hardware and software do not support voice recognition or speech synthesis.

Data Management

The data management facility of the Telephone Server system service allows an application program to use the internal modem of the hardware, eliminating the need for an external modem. Data transfers can be made from a local workstation or from a remote location with access to the Telephone Server. User interaction with the modem can be direct, via the Asynchronous Terminal Emulator (ATE), or transparent, via an application program such as the Operator or CT-Mail.

Telephony Management

The telephony management feature of the Telephone Server system service allows an application program to take control of all the functions of the hardware, including automatic dialing, placing a call on hold, switching between calls on different lines, and automatic answering when a line is ringing.

Status Monitor

The Status Monitor program is a systems administration or software programming tool. It provides a simplified, visual picture of the hardware circuits that are connected when a telephone management function is carried out. The Status Monitor has soft function keys that allow the user to select which lines to dial or answer, which calls to place on hold, which lines to hangup or disconnect, and which lines to link for conference calling. As each function is carried out, the screen visually connects the circuits that makes the operation take place.

The Status Monitor can be used to verify the proper operation of the Voice/Data Services, and as a debugging tool during program development.

HOW TO USE THIS MANUAL

The Voice/Data Services manual is a guide for users of the Voice/Data Services software. In addition, this manual serves as a reference for programmers creating application software that uses the Telephone Server services.

The manual is divided into five sections and an appendix.

Section 1 gives a short introduction to the manual, and a quick overview of the product.

Section 2 covers the Voice/Data Services commands Install Telephone Server, Deinstall Telephone Server, and Telephone Status.

Section 3 presents a discussion of programming concepts needed for working with the Voice/Data Services. These include naming conventions, as well as explanations of voice, data, and telephony management operations.

Section 4 contains an alphabetical list of all the Voice/Data Services operations.

Section 5 lists the data structures used with the Voice/Data Services operations.

Appendix A lists the status codes that may be returned by the Voice/Data Services.

2 COMMANDS

INTRODUCTION

For information about installing the Voice/Data Services software on your workstation hard disk, see the current Release Notice for the Voice/Data Services. In this section, installation refers to the loading of the Telephone Server system service from disk into memory (RAM).

The Voice/Data Services commands allow you install the Telephone Server system service into memory, deinstall the system service, and access the Status Monitor for a visual map of the internal hardware state.

INSTALL TELEPHONE SERVER

You install the Telephone Server into active memory by using the Install Telephone Server command.

Command Form

Install Telephone Server
[Configuration file] _____

Parameter Fields

[Configuration file]

The configuration file contains information used by the Telephone Server system service for the generation of dial characters, ringing, etc. as described in tables 5-6 and 5-7.

If this field is left blank, then the default configuration file, [sys]<sys>TmConfig.sys, is used (if it exists). If a file is specified in this field, it must be a valid file.

The configuration file specified here is also used by the Operator program for storing information such as the phone numbers of each line. Because of this, each workstation in a cluster should be given its own configuration file.

DEINSTALL TELEPHONE SERVER

The Deinstall Telephone Server command is used to remove the Telephone Server system Service from active memory.

Command Form

DeInstall Telephone Server

TELEPHONE STATUS

The Telephone Status command is used to enter the Voice/Data Services Status Monitor. The Status Monitor is a visual map of the analog crosspoint switch inside the hardware. This switch is used to connect incoming and outgoing calls with the rest of the hardware.

The Status Monitor displays several intersecting lines that originate and end in the upper right hand corner of the screen. These lines represent the telephone connection and external telephone lines attached to the back of the Voice Processor Module. By using the soft function keys shown at the bottom of the screen, you can perform several telephone functions and watch as they are carried out inside the Telephone Manager Module.

Command Form

Telephone Status

Status Monitor Screen

The Status Monitor screen shows four vertical lines, marked JØ through J3, and eight horizontal lines, LØ through L7, that are used to connect the telephone, phone line 1, and phone line 2 to the hardware components of the Voice Processor Module. These components are the modem, dialer, detector, encoder and decoder. For detailed information about these components, see the Voice Processor manual.

As circuits are completed and functions carried out, the Status Monitor lights up the lines on the screen to show a map of the internal circuitry of the module. All connections begin and end with the telephone, and take place through phone line 1, phone line 2, or both.

Phone, Lines 1 and 2

These three connections correspond to the plug-in jacks on the back of the Voice Processor Module. This is where the telephone, and two external telephone lines, plug into the module.

Lines 1 and 2 are for seperate telephone lines from the local phone company. The telephone is the telephone in your office or home. For further information about connections to the Voice Processor Module, see the Voice Processor manual.

Lines JØ through J3

Lines JØ and J1 are the connecting lines between the telephone, and phone lines 1 and 2. They are lit when incoming calls pass through the internal hardware to make a connection with the telephone. Line JØ corresponds to phone line 1, and Line J1 corresponds to phone line 2.

Lines J2 and J3 are auxilliary lines that act as jumpers to facilitate internal hardware switching. Telephone calls to the modem, dialer or encoder/decoder, for example, are connected via lines J2 and J3. These lines also light up to show you how the internal switching takes place.

Lines LØ through L7

Lines LØ, L4, and L7 are the internal connections to the telephone handset.

Line LØ is a straight through connection to the phone, without any modulation of the signal. Line L4 has an amplifier to boost voice signals for better clarity, and Line L7 has an attenuator to reduce the sound of the high volume dialing signals generated inside the module. Each of these lines completes an internal circuit depending upon the function being performed.

Line L1 connects to the modem, which is used to generate and receive data calls over the phone lines. L2 is the dialer, which generates the tones necessary to dial outside numbers from the workstation keyboard. Line L3 is the detector, which is used to detect activity on the incoming and outgoing lines, such as busy signals, dial tones, etc.

Lines L5 and L6 are the encoder and decoder (codec). They make the conversion from analog (voice) to digital (binary) signals and back again, so that your workstation can process voice messages as file data. The answering machine functions of some application software take advantage of the encoder and decoder.

Status Monitor Function Keys

The function keys on the workstation keyboard allow you to perform several telephone operations that appear on the Status Monitor screen. These function keys are:

F1	Line 1	Makes the connection between the telephone and phone line 1.
F2	Line 2	Makes the connection between the telephone and phone line 2.
F3	Hold	Places the currently active phone line on hold (pressing F1 or F2 takes the line off hold).
F4	Monitor	Allows you to screen incoming calls without callers knowing you are listening in. (Designed for use with an application software's answering machine function).
F5	Link	Allows conference calling by connecting the telephone, phone line 1 and phone line 2 together.
F6	Unused	
F7	Unused	
F8	Redial	Allow you to type in a number on the keyboard's numeric keypad and dial it by pressing F9. (This replaces, but does not obstruct, dialing from the telephone handset).
F9	Dial	
F10	Hangup	Disconnects the phone line from the telephone and hangs it up.

To leave the Status Monitor and return to the Executive, press FINISH.

DIALING TIPS

Dialing can take place on the telephone attached to your workstation, or on the numeric keypad of the keyboard. Numbers entered from the keypad, including long distance calls, can be entered without spaces and parentheses.

Some numbers need special characters entered for the Telephone Server to complete the call. For example, an outside call placed from a PBX system needs a pause inserted between the PBX number and the outside number dialed, like this number below.

```
9~9462233
```

For a list of special characters used with the Telephone Server, see Section 5, Table 5-3, "Dial Characters."

Mistakes made while entering numbers can be corrected by using eitherthe BACKSPACE key or the arrow direction and DELETE keys.

3 PROGRAMMING CONCEPTS

INTRODUCTION

The Voice/Data Services provide access to the hardware for three classes of applications: voice, data, and telephone management.

The voice management services allow applications (such as Word Processor and Mail) to do voice digitization.

The data management services allow programs (such as ATE and Mail) to access the module's modem. There is a request interface which works either locally or from a remote workstation.

The telephone management services allow application software (such as the Operator) to control the telephone functions of the hardware.

NAMING CONVENTIONS

The following conventions are those which are used to express numbers and name variables in the operations listed below.

NUMBERS

Numbers are decimal except when suffixed with "h" for hexadecimal. Thus, 11 = 11 and 11h = 17.

VARIABLE NAMES

The name given a variable indicates its characteristics, according to a formal naming convention. The parameters used in the procedure definitions, fields of request blocks and other data structures of the Voice/Data Services are named according to this convention. For further information on these naming conventions, see the CTOS Operating System manual, Volume 1.

A variable name is composed of up to three parts: a prefix, a root, and a suffix.

Prefixes

The prefix identifies the data type of the variable. Common prefixes are the following:

a absolute memory address (24 bits)

b byte (8 bit character or unsigned number)

c count (unsigned number)

f flag (TRUE = ØFFh or FALSE = Ø)

i index (unsigned number)

p logical memory address (pointer) (32 bits consisting of the offset and the segment base address)

q quad (32-bit unsigned integer)

rg array of....

s size in bytes (unsigned number)

Prefixes can be compound. Common compound prefixes are

cb count of bytes (the number of bytes in a string of bytes)

pb pointer to (logical memory address of) a string of bytes

Roots

The root of a variable name can be unique to that variable, can be selected from the list below, or can be a compound of the two. Common roots are the following:

erc status (error) code

exch exchange

fh file handle

lfa logical file address

lh line handle

ph partition handle

rq request block

th telephone handle

Suffixes

The suffix identifies the use of the variable. Suffixes are

Last — the largest allowable index of an array

Max — the maximum length of an array or buffer (thus one greater than the largest allowable index)

Min — the minimum length of an array or buffer

Ret — identifies a variable whose value is to be set by the called process or procedure rather than specified by the calling process

VOICE MANAGEMENT

The Voice Interface is a set of services that allow application programs to use the codec (voice encoder/decoder) features of the hardware for voice annotation, voice messaging, or as part of a software answering machine.

USING THE VOICE SERVICES

There are two general classes of programs which access the voice services; ones which use the services in a simple fashion, and ones which are more advanced in their use of the voice services.

Voice Control Structure (VCS)

The TsVoiceRecordToFile and TsVoicePlaybackFromFile operations both require the caller to pass a pointer to a stucture called the Voice Control Structure (VCS), which is described in table 5-3. This structure contains information such as the file handle of the voice file, starting and ending logical file addresses, and data byte counts.

Simple Use of the Voice Services

The simple type of application is one in which the details of recording and playback connections, line quality, and so on are not important. An example of this type of application would be one where voice annotation of documents is desired.

Recording Voice

To record a voice message, the simple type of application needs to allocate a voice control structure, and set the following fields:

fh set to the file handle returned by OpenfFile operation.

lfaStart
lfaMax
the logical file addresses of starting and ending locations in the file where data is to be placed (normally 512 for lfaStart and the file size for lfaMax).

qSampleStart
: the sample number to be assigned to the first data byte recorded (normally Ø).

qSampleMax
: the sample number that, when reached, will cause the recording to terminate (normally ØFFFFFFFFh).

fAutoStart
: set this to true (ØFFh). This causes the recording to be started as soon as the voice unit (aka phone or telephone set) becomes offhook (picked up).

All other fields should be set to Ø. The file should be created large enough for a large voice message. See the current Release Notice for memory and disk requirements.

After the operation completes, the first 512 bytes of the file should be used as a header to store information about the recording as described in Table 5-4.

To conserve disk space, truncate the file (using ChangeFileLength) to the size returned in the variable pointed to by "pLfaNext" in the TsVoiceRecordToFile operation.

Playing Back Voice

To play back voice, the VCS should be initialized as above, except that the "lfaMax" and "qSampleMax" fields in the VCS should be set to the values returned in the variables pointed to by "pLfaNext" and "pqSampleNext" in the TsVoiceRecordToFile operation and stored in the first 512 bytes of the file.

Advanced Use of the Voice Services

More advanced use of the voice services is possible for applications in which the details of controlling connections and so forth are important. An example of this type of application is an answering machine.

Such an application would set the "fAutoStart" field of the VCS to false (Ø), and set other fields as appropriate. The TsOffHook, TsVoiceConnect, and other operations would also be used.

For an example of advanced use of the voice services, see the "Telephony Management" section below.

Termination of Recording or Playback

An application program controls the termination of a recording or playback by setting the control information in the VCS. A recording is terminated when any of the following conditions occur:

- An error occurs.
- A TsVoiceStop command is issued.
- The phone is hung up (recording from phone).
- A TsOnHook or TsHold command is issued (recording from line).
- A TsDoFunction (function hangup) command is issued.
- The voice data fills the portion of the file specified ("lfaStart" and "lfaMax").
- The maximum number of data bytes is reached ("qSampleMax").
- A pause (silence) greater than the maximum occuers ("cPauseMax").
- Dial tone was detected ("fStopOnDialTone").

Data format

The voice data is encoded (after hardware compression) at a rate of either 8KHz: 32K bits per second (4096 data bytes per second), or 6KHz: 24K bits per second (3072 data bytes per second). The Telephone Server system service stores the data in 512-byte records as described in Section 6, Table 5-5.

Pause Compression

During recording, the hardware inserts escape bytes into the data stream. These escape bytes tell the system service when the voice levels fall below (7Fh) or rise above (0F7h) a certain threshhold. 7Fh replaces the data byte, and 0F7h is added to the data.

If the "fRawData" flag is true in the VCS, then all of the data, including escape sequences, is written to disk during recording, and escape sequences are ignored during playback. If the "fRawData" flag is false and the "fNoPause" flag is true in the VCS, then the escape sequences are not written to disk.

Setting "fNoPause" to true will result in slightly higher quality voice recordings, at the expense of disk space (aproximately 50% more).

If both flags are false, then during voice recording the Telephone Server system service replaces a portion of the data which is below the threshhold with an escape sequence. The escape sequence consists of the byte 7Fh followed by a word containing the number of data bytes being discarded.

The field "sPauseGap" in the VCS is used to determine how much of the data is not discarded after a 7Fh escape is encountered and before the 0F7h escape is encountered.

During playback the discarded data bytes (number specified in the stored escape sequence) are replaced with value 08h, which is the encoded value of silence.

Memory Usage

The voice services require a 13312-byte or larger (1Ø24 byte increments) work area. The work area is divided into an 8192-byte data queue used by the hardware, and the balance (512Ø minimum) is used for two file system I/O buffers.

Additional memory improves performance. The disk performance is largely determined by disk seek time, which occurs once per I/O regardless of the number of bytes transferred. The larger the buffer sizes, the more bytes transferred per I/O, and fewer disk seeks needed. Performance improves with more memory because the system has more time in which to process the data (although the average time to process the data remains the same, a longer time interval allows a more uneven distribution of CPU and disk usage to occur).

There is no routing of requests: the application must run on the workstation where the hardware is located.

The hardware for digitizing is non-shareable. It is not possible to playback on one line while recording on the other.

OPERATIONS

The procedures are:

```
TsVoiceConnect
TsVoicePlaybackFromFile
TsVoiceRecordToFile
TsVoiceStop
```

DATA MANAGEMENT

The data interface allows a program to use the modem contained in the hardware. The modem is 1200 or 300 baud, asynchronous, originate or answer, Western Electric 212A compatible, and may be used with either telephone line.

The modem allows the workstation and another computer or terminal with a compatible modem to transfer data over either telephone line.

STARTING A DATA CALL

A data line is opened with TsDataOpenLine. Data is transfered with either TsDataRead or TsDataWrite. A line is closed with TsDataClose. TsDataCheckpoint is used to checkpoint the data, allowing a program to guarantee transmission of all data. TsDataChangeParams and TsDataRetreiveParams are used to modify or examine control parameters such as parity or line control. TsDataUnacceptCall is used to retreive a TsDataOpenLine request which is waiting to answer an incoming call.

The TsDataOpenLine service returns a handle which is used by subsequent operations.

Use of the modem by a system service running under a single partition (sp) version of CTOS requires that "fLL" be true in the Data Control Structure (Section 6, Table 5-2). "fLL" should be false in all other cases.

Convert Voice Call To Data Call

An existing voice connection may be converted to a data call. This is done with the TsDataOpenLine service, setting "openMode" in the Data Control Structure to 0. In this case, the specified line must be off hook. The modem is connected to it and the voice unit and/or codec are disconnected if neccessary. "fOriginate" may be set to true or false depending on which modem (remote or local) acts as originator. One modem must be the originator and one must be the answerer.

Accept Data Call

A data call is accepted by using the TsDataOpenLine service, and setting "openMode" (Data Control Structure) to 1 and "fOriginate" to false. When the specified line rings, the line is placed offhook and connected to the modem. If the line is not ringing when the request is issued, then the service waits until it does ring.

Originate Data Call

A data call is originated by using the TsDataOpenLine service, and setting "openMode" to 2 and setting "fOriginate" to true. The line must be on hook. It is placed off hook and a number is dialed. The number to be dialed may contain special characters as described in Section 6, Table 5-1. Unless the "fNoWaitForDialTone" flag is set in the Data Control Structure, the server will wait for dial tone before dialing.

READING AND WRITING DATA

Most applications use the Voice Processor modem asynchronously to read and write data at the same time. Asynchronour processing is accomplished in one of two ways:

- by having two processes in the application program which each use the procedural interface to TsDataRead and TsDataWrite, or
- by having one process which issues TsDataRead and TsDataWrite requests with the Request kernel primitive, and then waits for responses to the TsDataRead and TsDataWrite (and other messages) with the Wait kernel primitive.

TERMINATING A DATA CALL

A data call is terminated when a TsDataClose occurs, when the program using the modem terminates, or when a non-recoverable error occurs. A non-recoverable error is any error returned by one of the data services, except

11205	Invalid handle (all)
11260	Duplicate request (TsDataCheckpoint)
11287	Data overrun (TsDataRead)
11288	Bad parity (TsDataRead)
11289	Data timeout (TsDataRead, TsDataWrite or TsDataCheckpoint)
11290	End of block character (TsDataRead)

If a call is terminated due to a non-recoverable error, subsequent operations (until a TsDataCloseLine occurs) using the same line handle will be returned with the terminating error code. A TsDataCloseLine must be issued (or the application program terminated) before the modem can be used again.

OPERATIONS

The procedures are:

TsDataChangeParams
TsDataCheckpoint
TsDataCloseLine
TsDataOpenLine
TsDataRead
TsDataRetreiveParams
TsDataUnAcceptCall
TsDataWrite

TELEPHONY MANAGEMENT

The Telephony interface allows a program to directly control the functions of the hardware.

The module supports two telephone lines and a voice unit which can be connected to each other and/or to dialing hardware, detectors, a voice encoder/decoder (codec), and a modem.

MAKING CONNECTIONS

The TsConnect, TsDoFunction, TsHold, TsOffHook, and TsOnHook services are used to make connections between the two telephone lines and the voice unit.

DIALING

The TsDial service is used to dial out flash, pulse or DTMF (touch tone) characters from a telephone line or the voice unit. It is also used for call progress tone receiving (CPTR) to detect dial tone, or detect when the called number has answered.

The TsReadTouchTone service is used to read DTMF (touch tone) characters from a telephone line or the voice unit.

Automatic Calling

Automatic dialing is accomplished by combining use of the Voice and Telephone services. The following program fragment shows an example of dialing a number and playing a message (created with the Operator, for example), invoked from the Executive with the following command form:

```
Autodial
  [Phone number]
  [Voice file]
```

variable and structure declarations are
ited. The AllocMemorySL, CheckErc, Exit,
nFile, Read, and RgParam operations are
cribed in the CTOS Manual.

```
/*
** Allocate memory for voice.
*/
sWorkArea := 8000h;
CheckErc(AllocMemorySl(sWorkArea, ADS pWorkArea));

/*
** Get parameters and open voice file.  "VCS" is the voice
** control structure (table 5-3) and "VDFH" is the voice data
** file header (table 5-4).
*/
CheckErc(RgParam(1,0,ADS sdNumber));
CheckErc(RgParam(2,0,ADS sdFile));
CheckErc(OpenFile(ADS VCS.fh,sdFile.pb,sdFile.cb,0,0,'mr'));
CheckErc(Read(VCS.fh,ADS VDFH,size(VDFH),0,ADS sDataRet));

/*
** Setup the fields of the voice control structure from the
** voice data file header, set autostart to false.
*/
VCS.lfaStart := VDFH.lfaStart;
VCS.lfaMax := VDFH.lfaMax;
VCS.qSampleStart := VDFH.qSampleStart;
VCS.qSampleMax := VDFH.qSampleMax;
VCS.f6KHz := VDFH.f6KHz;
VCS.fAutoStart := false;

/*
** Let server use its defaults for the rest of the fields.
*/
VCS.cPauseMax := 0;
VCS.cSampleOn := 0;
VCS.cSampleOff := 0;
VCS.fNoPause := false;
VCS.fStopOnDialTone := false;
VCS.fAltConnection := false;
VCS.nSectorStatusUpdate := 0;
VCS.sPauseGap := 0;
VCS.fRawData := false;
```

```
/*
** Put line offhook, wait for dial tone, dial, wait
** answer.
*/
CheckErc(TsOffHook(1,ADS th,0));
ch := '=';
CheckErc(TsDial(1,0,ADS ch,1,100,ADS cbDialRet));
CheckErc(TsDial(1,0,sdNumber.pb,sdNumber.cb,100,
                ADS cbDialRet));
ch := '?';
CheckErc(TsDial(1,0,ADS ch,1,100,ADS cbDialRet));

/*
** Lock the codec, specify connection, and play messa
*/
CheckErc(TsDoFunction(1,12));
CheckErc(TsVoiceConnect(1,false,true,false));
CheckErc(TsVoicePlaybackFromFile(1,pWorkArea,sWorkAre
         ADS VCS,size(VCS),ADS lfaNext,ADS qSampleNe
         ADS status));

/*
** Hangup and exit.
*/
CheckErc(TsOnHook(1,0));
Exit;
```

STATUS

The TsGetStatus service reports changes in the state of the module such as a line ringing.

CONFIGURATION

The TsQueryConfigParams, TsSetConfigParams, and TsRing services are used to query and update the configuration.

OPERATIONS

The procedures are:

TsConnect
TsDeinstall
TsDial
TsDoFunction
TsGetStatus
TsHold
TsOffHook
TsOnHook
TsQueryConfigParams
TsReadTouchTone
TsRing
TsSetConfigParams

4 OPERATIONS

INTRODUCTION

This section lists all the Voice/Data Services operations available to programmers working with the Voice Processor Module.

Each of the listed operations is presented in the following form:

```
RoutineName(param-1, ..., param-n): Type
```

where:

Routine Name is the name of the operation

param-n is the parameter used in the call

Type is the type of the returned value (usually ercType or word).

The type of parameter used in an operation is identified by the name of the variable. For more information about the naming conventions used, see Section 3, "Programming Concepts", or see the CTOS Operating System Manual, Volume 1 and 2.

TsConnect

Description

The TsConnect service is used to connect or disconnect the voice unit and telephone lines to each other.

Connecting a line which is on hold (see TsHold) terminates the hold condition.

Error code 11202 (invalid connection) is returned if the connection is not allowed (such as connecting voice unit to modem) or if it would require use of hardware already connected.

Error code 11205 (bad handle) is returned if the handle specified is not valid.

Procedural Interface

TsConnect (iTmModule, th, device, fAdd) : ErcType

where

iTmModule — is the Voice Processor Module, 1 being the first Voice Processor Module to the right of the CPU module.

th — is a telephone handle returned by TsOffHook or TsGetStatus.

device — specifies the line or phone mode, where

- 0 means telephone line 1.
- 1 means telephone line 2.
- 2 means voice unit
- 3 means voice unit connected in monitor mode.

fAdd — if TRUE, the device is added to the connection, otherwise it is removed.

Request Block

TsConnect

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	8
1	RtCode	1	Ø
2	nReqPbCb	1	Ø
3	nRespPbCb	1	Ø
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø2Eh
12	iTmModule	2	
14	th	2	
16	device	2	
18	fAdd	2	

TsDataChangeParams

Description

The TsDataChangeParams service allows a program to change the parity, line control, and other parameters for an open line.

The baud rate can not be changed after the line has been opened.

Procedural Interface

```
TsDataChangeParams (lh, pbLineControl,
                    cbLineControl) : ErcType
```

where

lh — is the line handle returned by an TsDataOpenLine call.

pbLineControl
cbLineControl — describe a new communications Data Control Structure (table 5-2).

Request Block

TsDataChangeParams

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	2
1	RtCode	1	ØEh
2	nReqPbCb	1	1
3	nRespPbCb	1	Ø
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø2Ah
12	lh	2	
14	pbLineControl	4	
18	cbLineControl	2	

TsDataCheckpoint

Description

The TsDataCheckpoint service causes a data line to be checkpointed. The request returns only after all TsDataWrite requests have been returned, and after the modem has transmitted the last character.

In the case of a timeout, any TsDataWrite requests are returned with error 11206 (Telephone Server timeout).

Procedural Interface

```
TsDataCheckpoint (lh, cTimeOut) : ErcType
```

where

lh	is the line handle returned by a TsDataOpenLine call.
cTimeOut	is the maximum amount of time in units of 100ms that the user wishes to wait for any outstanding TsDataRead or TsDataWrite operations to complete (0FFFFh implies wait forever).

Request Block

TsDataCheckpoint

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	4
1	RtCode	1	ØEh
2	nReqPbCb	1	Ø
3	nRespPbCb	1	Ø
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø28h
12	lh	2	
14	cTimeOut	2	

TsDataCloseLine

Description

The TsDataCloseLine service closes a data line and terminates the call. It may be used either to terminate a call and hangup the line, or to convert a data call to a voice call. In the later case it would be followed by a TsConnect request to attach the voice unit.

Procedural Interface

```
TsDataCloseLine (lh, cTimeOut) : ErcType
```

where

lh	is the line handle returned by an TsDataOpenLine call.
cTimeOut	is the maximum amount of time in units of 100ms that the user wishes to wait for any outstanding TsDataRead or TsDataWrite operations to complete (0FFFFh implies wait forever).

Request Block

TsDataCloseLine

Offset	Field	Size (bytes)	Contents
0	sCntInfo	1	4
1	RtCode	1	0Eh
2	nReqPbCb	1	0
3	nRespPbCb	1	0
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	8029h
12	lh	2	
14	cTimeOut	2	

TsDataOpenLine

Description

The TsDataOpenLine service is used to start a data session using the modem.

Procedural Interface

```
TsDataOpenLine (pLhRet, pbLineSpec, cbLineSpec,
                pbLineControl, cbLineControl,
                pbDialString, cbDialString,
                pcbStringRet) : ErcType
```

where

pLhRet points to the line handle which will be returned when a call is accepted.

pbLineSpec
cbLineSpec describe a phone line specification. The specification is composed of the following parts:

device	(optional)	{node}[device]
module	(optional)	1 - 9
line	(required)	1 or A = line 1 2 or B = line 2

sample specifications:

"[PHONE]1A"	(module 1, line 1 at local node)
"{Nyc}[Phone]2"	(module 1, line 2 at node "Nyc")
"[phone]51"	(module 5, line 1 at local node)

pbLineControl
cbLineControl describe a Data Control Structure (table 5-2).

pbDialString
cbDialString — describe the number to be dialed. The string contains characters as described in table 5-1.

pcbStringRet — points to a return status structure which contains the number of dial characters processed.

Request Block

TsDataOpenLine

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	Ø
1	RtCode	1	41h
2	nReqPbCb	1	3
3	nRespPbCb	1	2
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø25h
12	pbLineSpec	4	
16	cbLineSpec	2	
18	pbLineControl	4	
22	cbLineControl	2	
24	pbDialString	4	
28	cbDialString	2	
3Ø	pLhRet	4	
34	sLhRet	2	2
36	pcbStringRet	4	
4Ø	scbStringRet	2	2

TsDataRead

Description

The TsDataRead service moves a block of received data from the modem to the specified user memory. If the time interval expires before the memory buffer is full then the service returns with error 11281

If a parity error occurs, then the last byte returned will be the byte with the parity error.

Procedural Interface

```
TsDataRead (lh, cTimeOut, pbData, cbDataMin,
            cbDataMax, pCbDataRet) : ErcType
```

where

lh	is the line handle returned by a TsDataReadOpenLine call.
cTimeOut	is the maximum amount of time in units of 100ms that the user wishes to wait before returning from this call.
pbData	describes the memory area to which the data is returned. The size of the memory area must be large enough for at least cbDataMax bytes.
cbDataMin	is the minimum number (must be at least 1) of bytes which is returned (unless an error occurs).
cbDataMax	is the maximum number of bytes that can be returned into the memory area. cbDataMax must not be smaller than cbDataMin.
pCbDataRet	points to a word where the actual count of bytes read will be returned.

Request Block

TsDataRead

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	4
1	RtCode	1	8h
2	nReqPbCb	1	Ø
3	nRespPbCb	1	2
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø26h
12	lh	2	
14	cTimeOut	2	
16	cbDataMin	2	
18	pbData	4	
22	cbDataMax	2	
24	pcbDataRet	4	
28	scbDataRet	2	2

TsDataRetreiveParams

Description

The TsDataRetreiveParams service returns the parameters of the specified line.

Procedural Interface

```
TsDataRetreiveParams (lh, pStatusRet,
                      sStatusRetMax) : ErcType
```

where

lh — is the line handle returned by TsDataRetreiveParamsOpenLine or TsDataAcceptCall.

pStatusRet
sStatusRetMax — describe the memory area where the status structure is returned. The status information is described in table 5-2.

Request Block

TsDataRetreiveParams

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	2
1	RtCode	1	8h
2	nReqPbCb	1	Ø
3	nRespPbCb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø2Bh
12	lh	2	
14	pStatusRet	4	
18	sStatusRetMax	2	

TsDataUnAcceptCall

Description

The TsDataUnAcceptCall service allows a program to receive a TsDataOpenLine request before the timeout expires.

This is useful for a program that wishes to change control parameters as the result of user input, or for a program which only wishes to accept calls when some other condition is satisfied, such as time of day.

The TsDataAcceptCall is returned with error code 11285 (TsDataUnAcceptCall issued).

If no TsDataAcceptCall is at the server, then the operation returns error 11286 (No TsDataAcceptCall request).

Procedural Interface

```
TsDataUnAcceptCall (pbLineSpec, cbLineSpec) :
                        ErcType
```

where

pbLineSpec
cbLineSpec — are the same specification used in the TsDataAcceptCall request.

Request Block

TsDataUnAcceptCall

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	Ø
1	RtCode	1	1h
2	nReqPbCb	1	1
3	nRespPbCb	1	Ø
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø2Ch
12	pbLineSpec	4	
16	cbLineSpec	2	

TsDataWrite

Description

The TsDataWrite service writes a data block to the modem.

Procedural Interface

```
TsDataWrite (lh, cTimeout, pbData, cbData,
             pCbDataRet) : ErcType
```

where

lh	is the line handle returned by an TsDataReadOpenLine call.
cTimeout	is
pbData cbData	describe the memory area to containing the data.
pCbDataRet	points to a word where the actual count of bytes written will be returned.

Request Block

TsDataWrite

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	4
1	RtCode	1	8h
2	nReqPbCb	1	1
3	nRespPbCb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø27h
12	lh	2	
14	cTimeout	2	
16	pbData	4	
2Ø	cbData	2	
22	pcbDataRet	4	
26	scbDataRet	2	2

TsDeinstall

Description

The TsDeinstall request deinstalls the telephone server. The server will un-serve all of its request codes, respond to any outstanding requests with the specified error code, cleanup its internal state, place the Voice Processor Module in power-off state, and then respond to the TsDeinstall request with its partition handle.

Procedural Interface

```
TsDeinstall(iTmModule, pPhRet, erc) : ErcType
```

where

iTmModule	is the Voice Processor Module, 1 being the first Voice Processor Module to the right of the CPU module.
pPhRet	points to a word where the telephone server partition handle will be returned.
erc	is the error code to be returned to any client programs of the telephone server.

Request Block

TsDeinstall

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	4
1	RtCode	1	Ø
2	nReqPbCb	1	Ø
3	nRespPbCb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø38h
12	iTmModule	2	
14	erc	2	
16	pPhRet	4	
2Ø	sPhRet	2	2

TsDial

Description

The TsDial service causes the DTMF to generate the specified characters. The DTMF generator is automatically connected to the specified line.

Procedural Interface

```
TsDial (iTmModule, line, pbString, cbString,
        cErrorTimeout, pcbStringRet) : ErcType
```

where

iTmModule	is the Voice Processor Module, 1 being the first Voice Processor Module to the right of the CPU module.
line	is the phone line to be connected.
pbString cbString	describe the number to be dialed. The string contains characters as defined in table 5-1.
cErrorTimeout	Is the maximum time that is allowed before returning an error when waiting for dial tone.
pcbStringRet	points to a return status structure which contains the number of characters processed.

Request Block

TsDial

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	6
1	RtCode	1	Ø
2	nReqPbCb	1	1
3	nRespPbCb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø2Fh
12	iTmModule	2	
14	line	2	
16	cErrorTimeOut	2	
18	pbString	4	
22	cbString	2	
24	pcbStringRet	4	
28	scbStringRet	2	2

TsDoFunction

Description

The TsDoFunction service causes the telephone server to perform functions similar to those available by pressing buttons on a typical two line telephone.

Procedural Interface

TsDoFunction (iTmModule, function) : ErcType

where

iTmModule — is the Voice Processor Module, 1 being the first Voice Processor Module to the right of the CPU module.

function — is the function to be done, where

- Ø — Selects line 1 and connects to it, placing other line on hold if it is connected
- 1 — Selects line 2 and connects to it, placing other line on hold if it is connected
- 2 — Selects line 1 and connects to it, hanging up other line if it is connected
- 3 — Selects line 2 and connects to it, hanging up other line if it is connected
- 4 — Places selected line on hold
- 5 — Hangs up selected line
- 6 — Turns on speaker phone and connects to selected line

7	Turns off speaker phone and hangs up lines attached to it
8	Links lines 1 and 2 together
9	Unlinks lines and places unselected line on hold
1Ø	Switches the voice unit into monitor mode.
11	Switches the voice unit into normal mode.
12	Locks the voice encoder/decoder for the exclusive use of caller.
13	Unlocks the voice encoder/decoder.

Request Block

TsDoFunction

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	4
1	RtCode	1	Ø
2	nReqPbCb	1	Ø
3	nRespPbCb	1	Ø
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø31h
12	iTmModule	2	
12	function	2	

TsGetStatus

Description

The TsGetStatus service returns the state of the hardware and all users.

Procedural Interface

```
TsGetStatus (iTmModule, pStatusRet, sStatusRetMax,
             fNoWait) : ErcType
```

where

iTmModule	is the Voice Processor Module, 1 being the first Voice Processor Module to the right of the CPU module.
pStatusRet sStatusRetMax	describe the memory area where the status is to be returned.
fNoWait	if TRUE, then the status is returned immediately. if FALSE then the status is returned the next time something changes, or immediately if the iEvent field in the status block pointed to by pStatusRet is not the same as the current value of iEvent.

Request Block

TsGetStatus

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	4
1	RtCode	1	Ø
2	nReqPbCb	1	Ø
3	nRespPbCb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø2Dh
12	iTmModule	2	
14	fNoWait	2	
16	pStatusRet	4	
2Ø	sStatusRetMax	2	

TsHold

Description

The TsHold service causes the specified line to be disconnected from all devices and placed on hold.

Procedural Interface

TsHold (iTmModule, line) : ErcType

where

iTmModule	is the Voice Processor Module, 1 being the first Voice Processor Module to the right of the CPU module.
line	is either Ø or 1 to specify the telephone line.

Request Block

TsHold

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	4
1	RtCode	1	Ø
2	nReqPbCb	1	Ø
3	nRespPbCb	1	Ø
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø34h
12	iTmModule	2	
14	line	2	

TsOffHook

Description

The TsOffHook service makes the specified line go off hook. It does not change any connections of the cross point switch.

The service returns a handle which may be used for adding devices with the TsConnect operation.

Procedural Interface

```
TsOffHook (iTmModule, pThRet, line) : ErcType
```

where

iTmModule	is the Voice Processor Module, 1 being the first Voice Processor Module to the right of the CPU module.
pThRet	points to the word where a telephone connection handle will be returned.
line	is either Ø or 1 to specify telephone line 1 or 2.

Request Block

TsOffHook

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	4
1	RtCode	1	Ø
2	nReqPbCb	1	Ø
3	nRespPbCb	1	Ø
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø32h
12	iTmModule	2	
14	line	2	
16	pThRet	4	
2Ø	sThRet	2	2

TsOnHook

Description

The TsOnHook service causes the specified line to be disconnected from all devices and go on hook or "hungup".

Procedural Interface

TsOnHook (iTmModule, line) : ErcType

where

iTmModule	is the Voice Processor Module, 1 being the first Voice Processor Module to the right of the CPU module.
line	is either Ø or 1 to specify telephone line 1 or 2.

Request Block

TsOnHook

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	4
1	RtCode	1	Ø
2	nReqPbCb	1	Ø
3	nRespPbCb	1	Ø
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø33h
12	iTmModule	2	
14	line	2	

TsQueryConfigParams

Description

The TsQueryConfigParams service is used to get the Telephone Server configuration file name and the current configuration information.

Procedural Interface

```
TsQueryConfigParams (iTmModule,
                     pConfigRet,sConfigRetMax,
                     pbFileSpecRet,
                     cbFileSpecRetMax,
                     pcbFileSpecRet) : ErcType
```

where

iTmModule — is the Voice Processor Module, 1 being the first Voice Processor Module to the right of the CPU module.

pConfigRet
sConfigRetMax — describe the memory where the Telephone Server Configuration Block (as described in table 5-7) is returned.

pbFileSpecRet
cbFileSpecRetMax — describe the memory where the file specification of the Telephone Server configuration file is returned.

pcbFileSpecRet — points to a word where the actual size of the file specification is returned.

Request Block

TsQueryConfigParams

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	2
1	RtCode	1	Ø
2	nReqPbCb	1	Ø
3	nRespPbCb	1	3
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø35h
12	iTmModule	2	
14	pConfigRet	4	
18	sConfigRetMax	2	
2Ø	pbFileSpecRet	4	
24	cbFileSpecRetMax	2	
26	pcbFileSpecRet	4	
3Ø	scbFileSpecRet	2	2

TsReadTouchTone

Description

The TsReadTouchTone service is used to read the DTMF signal from a telephone line or the voice unit.

Procedural Interface

```
TsReadTouchTone (iTmModule, device, pbDigits,
                 cbDigitsMax, pcbDigitsRet,
                 bStopDigit, cTimeout) : ErcType
```

where

iTmModule — is the Voice Processor Module, 1 being the first Voice Processor Module to the right of the CPU module.

device — specifies the line or phone, where

- Ø means telephone line 1.
- 1 means telephone line 2.
- 2 means voice unit

pbDigits
cbDigitsMax — describe the memory area where the digits are returned.

pcbDigitsRet — is a pointer to a word where the actual number of digits read is returned.

bStopDigit — is a character which will terminate the input. The character may be Ø to 9, *, #, or A to D. An illegal character is interpreted as meaning there is no stop digit.

cTimeOut — is the time (in units of 1ØØms) after which the read will be terminated if no characters are received.

Request Block

TsReadTouchTone

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	8
1	RtCode	1	Ø
2	nReqPbCb	1	Ø
3	nRespPbCb	1	2
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø3Øh
12	iTmModule	2	
14	device	2	
16	bStopDigit	2	
18	cTimeOut	2	
2Ø	pbDigits	4	
24	cbDigitsMax	2	
26	pcbDigitsRet	4	
3Ø	scbDigitsRet	2	2

TsRing

Description

The TsRing service is used to turn on (or off) the monitor ringing, allowing an application and user to select a ring frequency interactively. If either line rings or if the the voice unit goes off hook, the monitor ringing will return to normal.

Procedural Interface

TsRing (iTmModule, hz) : ErcType

where

iTmModule	is the Voice Processor Module, 1 being the first Voice Processor Module to the right of the CPU module.
hz	is the frequency to be used. A value of Ø means no ringing. The range is Ø to 255.

Request Block

TsRing

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	3
1	RtCode	1	Ø
2	nReqPbCb	1	Ø
3	nRespPbCb	1	Ø
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø37h
12	iTmModule	2	
14	hz	1	

TsSetConfigParams

Description

The TsSetConfigParams service is used to set the Telephone Server configuration information.

Procedural Interface

```
TsSetConfigParams (iTmModule, pConfig,sConfig) :
                    ErcType
```

where

iTmModule — is the Voice Processor Module, 1 being the first Voice Processor Module to the right of the CPU module.

pConfig
sConfig — describe the Telephone Server configuration block as described in table 5-7.

Request Block

TsSetConfigParams

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	2
1	RtCode	1	Ø
2	nReqPbCb	1	1
3	nRespPbCb	1	Ø
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø36h
12	iTmModule	2	
14	pConfig	4	
18	sConfig	2	

TsVoiceConnect

Description

The TsVoiceConnect service is used to specify the connection for a TsVoicePlaybackFromFile or TsVoiceRecordToFile operation when the "fAutoStart" flag in the Voice Control Structure is false. If the TsVoiceConnect operation is done before the record or playback operation is done, then the voice connection must be locked first with the TsDoFunction operation.

Error code 11202 (invalid connection) will be returned if a connection is invalid or if the voice connection is not locked.

Procedural Interface

```
TsVoiceConnect (iTmModule, fVoiceUnit, fLine0,
                fLine1) : ErcType
```

where

iTmModule	is the Voice Processor Module, 1 being the first Voice Processor Module to the right of the CPU module.
fVoiceUnit	if true then the codec is connected to the voice unit.
fLine0	if true then the codec is connected to line 0.
fLine1	if true then the codec is connected to line 1.

Request Block

TsVoiceConnect

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	5
1	RtCode	1	Ø
2	nReqPbCb	1	Ø
3	nRespPbCb	1	Ø
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø23h
12	iTmModule	2	
14	fVoiceUnit	1	
15	fLineØ	1	
16	fLinel	1	

TsVoicePlaybackFromFile

Description

The TsVoicePlaybackFromFile service plays back voice data from the specified file.

Procedural Interface

```
TsVoicePlaybackFromFile (iTmModule, pWorkArea,
                         sWorkArea, pVoiceControl,
                         sVoiceControl, pLfaLast,
                         pqSampleLast, pStatusRet)
                         : ErcType
```

where

iTmModule — is the Voice Processor Module, 1 being the first Voice Processor Module to the right of the CPU module.

pWorkArea
sWorkArea — describes a memory location to be used by the Telephone Service for communicating to the Voice Processor module.

pVoiceControl
sVoiceControl — describe the Voice Control Structure (table 5-3).

pLfaLast — points to the returned lfa at the end of playback.

pqSampleLast — points to the returned sample number at end of playback.

pStatusRet — points to the return status of the operation, where

- Ø means termination because of an error.
- 1 means termination because the TsVoiceStop command was issued.

2 means termination because line or voice unit was placed onhook.

3 means termination because the maximum lfa was reached

4 means termination because the maximum sample was reached

Request Block

TsVoicePlaybackFromFile

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	2
1	RtCode	1	Ø
2	nReqPbCb	1	2
3	nRespPbCb	1	3
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø21h
12	iTmModule	2	
14	pWorkArea	4	
18	sWorkArea	2	
2Ø	pVoiceControl	4	
24	sVoiceControl	2	
26	pLfaLast	4	
3Ø	sLfaLast	2	4
32	pqSampleLast	4	
36	sqSampleLast	2	4
38	pStatusRet	4	
42	sStatusRet	2	2

TsVoiceRecordToFile

Description

The TsVoiceRecordToFile service copies voice data to the specified file.

Procedural Interface

```
TsVoiceRecordToFile (iTmModule, pWorkArea,
                     sWorkArea, pVoiceControl,
                     sVoiceControl, pLfaNext,
                     pqSampleNext, pStatusRet) :
                     ErcType
```

where

iTmModule — is the Voice Processor Module, 1 being the first Voice Processor Module to the right of the CPU module.

pWorkArea
sWorkArea — describes a memory location to be used by the Telephone Service for communicating to the Voice Processor Module.

pVoiceControl
sVoiceControl — describe the Voice Control Structure (table 5-3).

pLfaNext — points to the returned lfa of the next sector in the recording file at time of termination.

pqSampleNext — points to the returned sample number which would have been given the next sample in the recording at time of termination.

pStatusRet — points to the return status of the operation, where

- Ø means termination because of an error.
- 1 means termination because the TsVoiceStop command was issued.

2 means termination because line or voiceunit was placed onhook.

3 means termination because the maximum lfa was reached

4 means termination because the maximum sample was reached

5 means termination because the maximum pause was detected.

Request Block

TsVoiceRecordToFile

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	2
1	RtCode	1	Ø
2	nReqPbCb	1	2
3	nRespPbCb	1	3
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø22h
12	iTmModule	2	
14	pWorkArea	4	
18	sWorkArea	2	
2Ø	pVoiceControl	4	
24	sVoiceControl	2	
26	pLfaNext	4	
3Ø	sLfaNext	2	4
32	pqSampleNext	4	
36	sqSampleNext	2	4
38	pStatusRet	4	
42	sStatusRet	2	2

TsVoiceStop

Description

The TsVoiceStop service causes the digitization to be suspended.

Procedural Interface

TsVoiceStop (iTmModule) : ErcType

where

iTmModule	is the Voice Processor Module, 1 being the first Voice Processor Module to the right of the CPU module.

Request Block

TsVoiceStop

Offset	Field	Size (bytes)	Contents
Ø	sCntInfo	1	2
1	RtCode	1	Ø
2	nReqPbCb	1	Ø
3	nRespPbCb	1	Ø
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
1Ø	rqCode	2	8Ø24h
12	iTmModule	2	

5 VOICE/DATA SERVICES STRUCTURES

INTRODUCTION

The Voice/Data Structures included here are an integral part of the operations in Section 4, "Operations". They are also referred to frequently in Section 3, "Programming Concepts". The tables are as follows.

5-1 Dial Characters

5-2 Data Control Structure
This structure is passed to the Telephone Server system service with the TsDataOpenLine operations to control the data call characteristics such as baud rate, parity, or line control.

5-3 Voice Control Structure

5-4 Voice Data File Header

5-5 Voice Data File Record

5-6 Configuration File Format

5-7 Telephone Server Configuration Block

5-8 Telephone Status Structure

5-9 Telephone Module Control Block (TMCB)

Table 5-1: Dial Characters

Character(s)	Function
Ø-9,A-Y,a-y,*,#	are the dial or DTMF characters normally found on the touch pad or rotary switch. "Q","q","Z","z" are not mapped.
!A,!B,!C,!D	are the DTMF characters A-D which do not normally appear on the touch pad.
%	switch to pulse dialing
&	switch to tone dialing
@	flash for the default flash time.
!@x	flash for x units of 1ØØms (x is a byte)
=	wait for dial tone
!=	wait for any tone
?	wait for the phone being called to answer.
~	Pause for the default pause time.
!~x	pause for x units of 1ØØms
$tl	generate part of a dual-tone multi-frequency (DTMF) tone "t" for time "l" (units of 1Øms). The primary use of this is to generate a single tone (for use in answering machines, for example).

Table 5-1: Dial Characters (continued)

Character(s)	Function
	the tone generated is derived from a combination of the two frequencies specified by the upper (column) and lower (row) 4-bit nibbles of "t". The valid values of nibble are ØEh, ØDh, ØBh, and Ø7h. If both upper and lower nibbles are valid, then the tone generated will be one of the 16 DTMF tones as specified in the following table:

	ØE0h	ØD0h	ØB0h	Ø70h
ØEh	1	2	3	A
ØDh	4	5	6	B
ØBh	7	8	9	C
Ø7h	*	Ø	#	D

For example, a value of ØE7h will generate the tone for "*".

If only one of the nibbles is valid (such as Ø7h or ØE0h), then a single tone is generated. If neither nibble is valid then no tone is generated.

SP,(,),-,/ — these characters are ignored (SP is a space, or ASCII 20h).

Table 5-2: Data Control Structure

Offset	Field	Size (bytes)	Description
Ø	openMode	1	the data open mode, where Ø convert. convert an existing voice call to a data call 1 accept. wait for an incoming call to occur 2 dial. make an outgoing call using the dial string described below
1	fOriginate	1	if true then the modem will be programmed in originate mode instead of answer mode
2	cTimeOut	2	is the maximum time allowed to complete the open after the initial connection before returning error 11206 (time out)
4	baudrate	2	either 300 or 1200
6	parity	1	the parity control, where the values are defined as: Ø none 1 even (bit 7 is set or cleared so that there are always an even number of bits set in each byte).

Table 5-2: Data Control Structure (continued)

Offset	Field	Size (bytes)	Description
			2 odd (bit 7 is set or cleared so that there are always an odd number of bits set in each byte).
			3 1 (bit 7 is always set).
			4 Ø (bit 7 is always cleared).
7	lineControl	1	either XON/XOFF or none:
			Ø none
			1 XON/XOFF
8	fLL	1	If true, then the service is not closed when a termination request is received for that user number (for system services running on SP versions of CTOS)
9	fEndOfBlock	1	If true, then the byte value in bEndOfBlock is used as the last character in any TsDataRead operation
1Ø	bEndOfBlock	1	The character which terminates a TsDataRead operation
11	fHangup	1	if true, then the line is placed on hook, otherwise it is placed on hold when the data call is closed or terminated

Table 5-2: Data Control Structure (continued)

Offset	Field	Size (bytes)	Description
12	fCharMode	1	if true, then data is processed on a character basis, rather than on a block basis
13	fNoWaitForDialTone	1	if true, then the Telephone Server does not wait for dial tone before dialing
14	fReadTimoutReset	1	if true, then the Telephone Server resets its timeout counter for the TsDataRead operation whenever a character is received
15	fWriteTimoutReset	1	if true, then the Telephone Server resets its timeout counter for the TsDataWrite operation when ever a character is transmitted

Table 5-3: Voice Control Structure (VCS)

Offset	Field	Size (bytes)	Description
Ø	fh	2	is an open file handle
4	lfaStart	4	is the position in the file to start the recording into or playback from
6	lfaMax	4	recording or playback is terminated if this position in the file is reached
1Ø	qSampleStart	4	the sample number assigned to the first sample recorded, or to be skipped to on playback
14	qSampleMax	4	recording or playback is terminated if this sample number is reached
18	cPauseMax	2	on record, this is the maximum silence in units of 1ØØms before the recording is terminated (the terminating pause is not included in the recording) on playback, all pauses are truncated to this amount. A value of ØFFFFh means no pause termination or truncation
2Ø	cSampleOn	2	
22	cSampleOff	2	determine the playback "fast forward" rate. A zero value for "cSampleOff" means playback at normal speed. The "cSampleOn" value determines the number of pairs of samples which will be

			played back, and "cSampleOff" is the number of sample pairs which will be discarded. "cSampleOn" should be greater than 50. To playback at double speed, values of 100 and 100 could be used. To playback at triple speed, values of 100 and 200 could be used
24	f6KHz	1	if true, then the codec sampling rate will be 6KHz, otherwise 8KHz (8192 4-bit samples per second)
25	fAutoStart	1	if true, then the recording or playback starts as soon as the voice unit is placed off hook, or immediately if it is already off hook
26	fNoPause	1	if true, then the pause detection hardware is disabled and no pause detection or compression is done
27	fStopOnDialTone	1	if true, then the recording will be terminated if dial tone is detected
28	fAltConnection	1	if true, then an alternate connection will be used (if possible) to increase gain levels
29	nSectorStatusUpdate	2	if greater than zero, then any TsGetStatus requests are responded to each time the specified number of sectors have been processed

31	sPauseGap	2	number of data bytes after start of pause and before end of pause where the actual data is used. If sPauseGap is Ø then the default value (25Ø) is used
22	fRawData	1	if true, then the voice data is recorded or played back with out examination of the data for escape sequences

Table 5-4: Voice Data File Header

Offset	Field	Size (bytes)	Description
Ø	signature	2	must be "VC" (4356h)
2	version	2	version of file, currently 1
4	reserved	2	
6	f6KHz	1	if true, then the data was recorded at 6KHz
7	lfaStart	4	starting lfa in file of voice data
11	lfaMax	4	ending lfa in file of voice data
15	qSampleStart	4	starting data byte count of voice data
19	qSampleMax	4	ending data byte count of voice data
23	reserved	1	
24	dateTime	4	system date time when the recording was made
25	line	1	line over which recording was made, where Ø is the voice unit, and 1 or 2 is telephone line 1 or 2
26	reserved	233	

Table 5-5: Voice Data File Record

Offset	Field	Size (bytes)	Description
Ø	qSampleStart	4	the accumulated sample number of the first sample in this record
4	signature	1	always a 'V' (56h)
5	version	1	version number of data file. Currently always 1
6	rgbSample	5Ø6	voice data.

Table 5-6: Configuration File Format

The Telephone Server configuration file is shared with the Operator program. The first 256 bytes are used by the Telephone Server and the remainder of the file is used by the Operator.

If no configuration file is present, then the Telephone Server system service is initially configured so that ACTION-1 and ACTION-2 correspond to TsFunction values 2 and 3, respectively. Changing the values of rgAction and/or rgFunction will override this.

Offset	Field	Size (bytes)	Description
Ø	signature	2	always a 'tC'
2	version	2	version number of configuration file (currently not used)
4	TsConfig	18	configuration information (see Table 5-6)
22	reserved	1Ø6	
128	rgAction	64	array of action key values (unencoded keyboard values) to be mapped to server functions. A value of Ø terminates the list.
192	rgFunction	64	array of functions (see TsFunction) to be executed when the action key listed in rgAction is typed
256	reserved	128	
512	OperatorData	512	the Operator program uses this area for storing data such as the phone number of each line, diaing prefixes, etc.

Table 5-7: Telephone Server Configuration Block

In the following structure, all fields except rgRingHz are two-byte arrays, one byte for each telephone line. The default values given are used when no configuration file is present.

Offset	Field	Size (bytes)	Description
Ø	rgDtmfGenOff	2	
2	rgDtmfGenOn	2	specifies the time between and during DTMF tones when dialing, in units of 1Øms. Most PBX's and CBX's can handle values of 1Ø (1ØØms), and some as little as 6 (6Øms). The default is 6 for the off time and 8 for the on time.
4	rgFlashTime	2	specifies the length of a default "flash" ("@" character in a dial string) in units of 1ØØms. Most PBX's use a value of 1Ø (1 second). The default is 1Ø.
6	rgPauseTime	2	specifies the length of a default pause ("~" character in a dial string) in units of 1ØØms. The default is 2Ø.
8	rgfPulseDial	2	if true, then pulse dialing is used instead of DTMF generation. The default is false.
1Ø	rgRingHz	4	specifies the frequency of the workstation monitor's beeper when ring signal is detected on: Ø neither line (default Ø). 1 line 1 (default 4Ø).

2 line 2 (default 90).

3 both lines 1 and 2 (default 150).

A value of 0 means no tone is generated. The range of frequencies is 0 to 255.

14 rgiRingThrough

2 the ring number in which the ring current will be sent directly to the voice unit (phone). A value of 0FFFFh means the current is never passed through. The default is 4.

16 rgRingMode

2 the ring mode for each line. When a line rings, some combination of ringing the monitor or ringing the phone is done, where

0 do not ring either

1 ring only the phone

2 ring only the monitor

3 always ring both

4 ring the monitor for the number of rings specified in rgiRingThrough above, then ring the phone

5 ring the monitor for the number of rings specified in rgiRingThrough above, then ring both (default case)

Table 5-8: Telephone Status Structure

Offset	Field	Size (bytes)	Description
Ø	iEvent	2	a counter which is incremented every time the contents of the status structure changes
2	defaultLine	1	the default line as selected by TsFunction
3	fNeedCodecConnection	1	if true, then a voice operation is waiting for a connection before it can begin
4	fCodecInUse	1	if true, then the codec is recording or playing back
5	aTmcb	3	the 24-bit real address of the TMCB
8	reserved	2	
1Ø	baudRate	2	
12	originateMode	1	
13	parityMode	1	
14	lineControlMode	1	the modem control parameters as described in the Data Control Structure.
15	reserved	1	
16	vUnitState	16	the voice unit state, as described below in vUnitState.
32	rgLineState(2)	32	the line 1 and line 2 states, as described below in lineState.
64	codecState	16	the codec state as described below in codecState.

vUnitState

Offset	Field	Size (bytes)	Description
Ø	fOffHook	1	if true, then the voice unit is offhoo
1	rgfTLine(2)	2	an array of flags which, if true mean line 1 and/or line 2 is connected to the voice unit
3	rgfRingThrough(2)	2	an array of flags which, if true mean line 1 or line 2 is connected directly to the voice unit to allow ring voltage through
5	fCodec	1	if true, the voice unit is connected to the codec
6	fDtmfRec	1	if true, the voice unit is connected to the dtmf decoder
7	fMonitor	1	if true then the voice unit is connected in monitor mode
8	handle	2	the handle of the connection which the voice unit is attached to
1Ø	hookThroughMode	1	the mode for passing through on/off hook from the voice unit to the lines, where Ø means neither line, 1 means line 1, 2 means line 2, and 3 means both lines
11	reserved	5	

LineState

Offset	Field	Size (bytes)	Description
Ø	status	1	the line status, where Ø onhook 1 ring current detected 2 modem not ready 3 modem ready (FSK, 3ØØ baud) 4 modem ready (PSK, 12ØØ baud) 5 offhook
1	fDialing	1	if true, then a dial string is being processed on this line
2	dialState	1	the current state of dialing, where Ø idle 1 generating DTMF 2 Analysing CPTR (waiting for dial tone) 3 generating pulse 4 generating flash 5 generating pause 6 analysing CPTR (waiting for any tone) 7 analysing CPTR (waiting for answer) 255 doing error recovery
3	fRinging	1	if true, then the line is ringing

4	iRing	1	number of rings
5	fRingThrough	1	if true, then the ringing is being passed through to the voice unit
6	fOffHook	1	if true, the the line is offhook
7	fHold	1	if true, then the line is on hold
8	fCodec	1	if true, then voice is being used
9	fDtmfRec	1	if true, then the DTMF detector is being used
10	handle	2	the connection handle
12	reserved	1	
13	fModem	1	if true, then the modem is being used
14	reserved	2	

CodecState

Offset	Field	Size (bytes)	Description
0	lfaCurrent	4	lfa of data record last processed
4	qSampleCurrent	4	data byte number last processed
8	reserved	8	

Table 5-9: Telephone Module Control Block (TMCB)

Offset	Field	Size (bytes)	Description
Ø	signature	2	must be Ø8459h. It is checked by the Voice Processor Module firmware before generating each interrupt
2	version	1	This is the firmware version number
3	command	1	This is the command that the server gives to the Voice Processor Module
4	vUnitStatus	1	The current state of the voice unit, where Ø means onhook 1 means offhook
5	xpLine	1	The device to be connected in the XptSwitch, or for which data is to be started or stopped, where Ø voice unit (with attenuation added) 1 modem 2 DTMF generator 3 DTMF receiver 4 Call Progress Tone Receiver (CPTR) used for detecting dial tone and busy signals 5 CODEC voice digitization encoder 6 CODEC voice digitization decoder

			7 voice unit
6	xpJunctorMap	1	The bit map of telephone lines and terminals in the XptSwitch for a connection, or the telephone line for hold and hook commands, where bits are assigned as follows: Ø (1h) line 1 1 (2h) line 2 2 (4h) terminal 3 (8h) terminal with attenuation
7	paramØ	1	
8	param1	1	The param bytes are used in conjunction with various commands as described below.
9	tØstatus	1	
1Ø	t1status	1	The current state of the PBX or CBX lines 1 and 2, where Ø onhook 1 ring current detected 2 modem not ready 3 modem ready (Fsk) 4 modem ready (Psk) 5 offhook
11	dialerCQCB	8	control structure used for pulse dialing
19	sDtmfGenBuf	1	number of encoded characters to be dialed using DTMF generator
2Ø	aDtmfGenBuf	3	address (24-bit real address) of first encode character

23	bDtmfChar	1	character decoded using DTMF receiver
24	reserved	7	
31	modemRxCQCB	8	control structure for characters received over modem
39	modemTxCQCB	8	control structure for characters to be transmitted over modem
47	codecBPCB	8	control structure for CODEC encoding or decoding
55	ercCodec	2	error status from CODEC
57	ercModem	2	Error status from modem
59	cCptrLow	1	
60	cCptrHigh	1	high and low signal detected by CPTR. A low/high value of 0/255 means dial tone, 45-55/45-55 means busy, and 15-35/15-35 means fast busy. Other values are not defined and may mean no connection, voice/data, or noise on the line.
61	intStatus	1	the Voice Processor Module firmware places a status byte here before generating the XINT3 interrupt signal to wake up the server. The server puts a 0FFh here after processing the interrupt.
62	xptMap	8	the state of the cross point switch is updated by the Voice Processor Module

A VOICE/DATA SERVICES STATUS CODES

The following are the error codes generated by the Voice/Data Services Telephone Server:

11200 Invalid user.
The user number of the program making the request is not an allowed user at this time for the specified operation.

11201 reserved.

11202 Invalid connection.
The connection resulting from the specified operation is not allowed.

11203 Invalid parameters.
The parameters to the specified operation are not valid.

11204 Not offhook.
The voice unit or telephone line must be offhook for the specified operation.

11205 Bad handle.
The handle specified was invalid.

11206 Timeout.
The operation was terminated because a specified time limit for completion expired.

11207 Invalid module.
The module specfied is invalid.

11208 Not onhook.
The line must be onhook for the operation specified.

11209 Voice unit onhook.
The voice unit must be offhook for the operation specified.

11210 Invalid function.
The function specified to the TsDoFunction operation is illegal.

11211 On hold.
The operation was terminated due to the line being placed on hold as the result of command being issued.

11212 On hook.
The operation was terminated due to the

phone or line being placed onhook or to a TsOnHook or TsDoFunction (hangup) command being issued.

11213-11219 reserved.

11220 Voice work area too small.
The work area must be at least 13312 bytes and word alligned.

11221 Voice in use.
The voice hardware is either in use or has been locked with TsDoFunction.

11222 Voice sample not found.
The specified starting sample could not be found.

11223 Voice data lost (disk).
The voice operation was terminated due to the disk and/or cluster not being able to keep up with the voice data rate.

11224 Voice data lost (cpu).
The voice operation was terminated due to the Telephone Server process not being able to keep up with the voice data rate.

11225 Bad voice parameters.
One or more field in the voice control structure are invalid.

11226 Invalid voice data file.
The signature byte is invalid in a record of the file specified.

11227-11249 reserved.

11251 Initialization time out.
The TM-001 module did not respond due to a failure of the TM-001 module, XBus DMA or XBus interrupt hardware or software.

11252 No request file.
The Telephone Server request file did not get loaded by the Operating System during system initialization (boot).

11253 Bad request file.
The Telephone Server request file is not of the same revision level as the Telephone Server.

11254 Telephone Server already installed.

11255 TM-ØØ1 module failure.
The TM-ØØ1 module did not respond due to a failure of the TM-ØØ1 module, XBus DMA or XBus interrupt hardware or software.

11256 Bad request.
A request not recognized by the Telephone Server was received at the Telephone Server's service exchange.

11258 User swapped.
The operation was terminated due to the swapping to disk of the program using the service.

11259 Invalid configuration file.
The signature word in the configuration files specified is invalid.

11260 Duplicate request.
The request is a duplicate of one currently being processed and is not allowed.

11261 Firmware invalid.
The TM-ØØ1 module firmware is invalid.

11262-
11269 reserved.

11270 Bad dial character.
A character specified in the dial string is not valid.

11271 Dialing in progress.
A TsDial or TsDataOpenLine service is in progress using the dialing hardware.

11272 No dial tone: Busy.
A busy signal was detected instead of dial tone when a wait for dial tone command ("=") was processed.

11273 No dial tone: Fast busy.
A fast busy signal was detected instead of dial tone when a wait for dial tone command ("=") was processed.

11274 No dial tone.
No dial tone was detected when a wait for dial tone command ("=" or "!=") was processed.

11275 Bad dial pulse character.
A character was processed which can not be dialed with pulse dialing (such as "#" or "*").

11276 Too many characters to dial.
No more than 64 characters can be processed at a time.

11277 Detector in use.
The hardware required for detecting DTMF or dial tone is in use by another operation.

11278 No answer.
The party being called did not answer.

11278-11279 reserved.

11280 Line in use.
The line specified in a TsDataOpenLine operation is in use.

11281 Invalid line specification.
The line specifed in a TsDataOpenLine or TsDataUnAcceptCall operation is invalid.

11282 Invalid line parmeters.
The line control structure has invalid parameters or size.

11283 Unaccept issued.
The TsDataOpenLine operation is being terminated due to a TsDataUnAcceptCall being issued.

11284 No accept call issued.
The TsDataUnAcceptCall failed because no TsDataOpenLine operation for the line was waiting to accept a call.

11285 reserved.

11286 Lost carrier.
The modem lost carrier signal.

11287	Data overrun. The TsDataRead operation was returned due to a data overrun. Data was lost due to the application failing to issue TsDataRead operations fast enough to keep up with the modem, or due to multiple parity errors.
11288	Bad parity. The TsDataRead operation was returned due to bad parity being detected.
11289	TsDataRead or TsDataWrite timeout. The TsDataRead or TsDataWrite operation was returned due to the time limit expiring specified in the operation.
11290	End of block character received. The TsDataRead operation was returned due the specified end of block character being received.
11291- 11299	reserved.

GLOSSARY

analog crosspoint switch. The switching unit inside the Telephone Manager Module which makes connections between incoming and outgoing analog signals, and the Telephone Manager's digital hardware.

CODEC. The encoder/decoder hardware of the Telephone Manager Module which actually makes the analog to digital signal conversions.

CPTR. Stands for Call Progress Tone Receiver. The receiver is used for detecting dial tone, busy, fast busy, or whether the called party has answered.

DTMF. Stands for Dual Tone Multifrequency. The double toned signal produced by the telephone, which when dialed, alerts the Telephone Compnay exchange to the placement, or reception, of a call. The hardware has both a DTMF generator for producing DTMF tones, and a DTMF receiver for receiving them.

flash. A flash is used with some PBX's to signal that a PBX feature is desired. A flash is generated by placing the line onhook for a short period of time, usually about 1 second.

hertz (hz). Cycles or periods per second.

offhook. The electical state of a telephone line when a phone or other device is connected to the PBX or Telephone Company.

onhook. The electical state of telephone line when the line is not connected.

PBX. Stands for Private Branch Exchange. The internal telephone system of a business or office which processes telephone calls through its own switching system. This system is generally tied into, but independent of, the Telephone Company system.

pulse. An older form of dialing than DTMF in which the line is placed on and offhook rapidly.

switchook. Same as flash.

system service. A program, or sub-program, which provides a service for other programs. It may include one or more processes, and may be part of the operating system, an installable program , or an application.

telephone line. The connection to a PBX or Telephone Company.

voice unit. Any attachment to the workstation which can be used to send or receive voice messages, such as a phone, telephone, telephone set, or speaker phone.