Volume 3, Number 5 October 1985
IBM Personal Computer Voice Communications Option

IBM Personal Computer Seminar Proceedings

The Publication for Independent Developers
of Products
for IBM Personal Computers

Published by International Business Machines Corporation
Entry Systems Division

Changes are made periodically to the information herein; any such changes will be reported in subsequent Proceedings.

It is possible that this material may contain reference to, or information about IBM products (machines and programs), programming or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such products, programming or services in your country.

IBM believes the statements contained herein are accurate as of the date of publication of this document. However, IBM makes no warranty of any kind with respect to the accuracy or adequacy of the contents hereof.

This publication could contain technical inaccuracies or typographical errors. Also, illustrations contained herein may show prototype equipment. Your system configuration may differ slightly.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever.

All specifications are subject to change without notice.

Printed in the
United States
of America

Contents

Introduction and Welcome

These are the Proceedings of the IBM Personal Computer Seminar, designed for independent developers of products for IBM Personal Computers. The purpose of these Proceedings is to aid you in your development efforts by providing relevant information about new product announcements and enhancements to existing products. This issue is prepared in conjunction with this seminar. The Proceedings of future seminars for IBM Personal Computers also will be published and will cover topics presented at those seminars.

The IBM Personal Computer Voice Communications Option, which is discussed in this issue of the Proceedings (3.5), can execute on the IBM Personal Computer, the IBM Personal Computer XT and the IBM Personal Computer AT (hereafter referred to as PC).

Purpose

What is our purpose in issuing a publication such as this? It is quite simple.

The IBM Personal Computer family is a resounding success. We've had a lot of help in achieving this success, and much of it came from the independent developers.

As you proceed with your development, do you at times wish for some bit of information or direction which would make the job easier? Information which IBM can provide? This is the type of information we want to make available to you.

Since we want to be assured of giving you the information you need, we ask you to complete the questionnaire which appears at the end of these Proceedings. Your response to this questionnaire will be taken into account in preparing the content of future issues, as well as the content of seminars we will present at microcomputer industry trade shows.

Topics

The following list gives a general indication of the topics we plan to cover in future seminars and include in the IBM Personal Computer Seminar Proceedings:

- Information exchange forum — letters to the editor format
- Development tools — languages, database offerings
- Compatibility issues
- New devices — capacities and speeds
- System capacities — disk and memory
- Enhancements in maintenance releases
- Tips and techniques
- New system software
- Hardware design parameters
- Tips on organizing and writing documents for clear and easy reading
- Changes to terms and conditions

The IBM Personal Computer Voice Communications Option

Introduction

The IBM Personal Computer Voice Communications Option includes the following components:

- An adapter card
- Cables to attach the card to the telephone system
- A diskette containing the Voice Communications Operating Subsystem (VCOS)
- An "Exploring Voice Communications" diskette (sample diskette)
- An *Installation and Setup* Manual

Not included with the package, but separately available are:

- The *IBM Voice Communication Application Program Interface (VCAPI) Reference* (part number 6280743). This product provides the tools that application developers need when writing programs that take advantage of the option's capabilities.
- An update to the *Technical Reference Option and Adapters* Manual

This Seminar Proceedings concentrates on the VCAPI and its implementation (VCOS) both on the IBM Personal Computers (hereafter referred to as PC) and the adapter card.

Highlights

The Interface (VCAPI)

The VCAPI provides a complete functional interface to the Voice Communications Operating Subsystem (VCOS) for telephone management, asynchronous communications, audio record and playback, speech synthesis and speech recognition.

The functions in the interface are grouped into sets. The Base function set, which is loaded when the subsystem is initialized, provides functions to manage the resources of the subsystem. Other dynamically loadable function sets are:

- Telephony

  The telephony function set provides the capability to dial phone numbers and to monitor the progress of an outgoing telephone call as it is being placed.

- Line Monitoring

  The line monitoring function set allows an application to be notified of incoming signals, generated by telephone equipment, after a call has been established. This includes Dual Tone Multiple Frequency (DTMF) tone detection, generated when the keys of a remote tone telephone are pressed.

- Asynchronous Communications

  The asynchronous communications function set provides the capability to send and receive data over a telephone line using the normal asynchronous protocols. The adapter card provides both the protocol support and an internal 103/212 compatible modem.

- Audio Input

  The audio input function sets provide the capability to receive audio input from a device, digitize and compress it, and give the resulting data to the application in memory buffers. There is a separate function set for each compression algorithm. These function sets differ in the form and amount of digitized data they produce, as well as in the quality of the resulting playback.

- Audio Output

  The audio output function sets do the reverse of audio input: data in memory buffers is reconstructed into audio output signals and sent to an output device. Separate audio output function sets are provided for each compression algorithm and must be used in correspondence with the function set used to record.

- Speech Synthesis (Text-to-Speech)

  The speech synthesis function set provides a means of generating intelligible speech from an ASCII text string. This speech can be directed to any output device attached to the adapter.

- Speech Recognition

  The speech recognition function set provides the capability of recognizing previously trained utterances and parsing sequences of recognized utterances, while converting them into integer IDs defined by a command language. The Recognizer is a discrete utterance recognizer, which is characterized by the need for a short pause between each utterance.

Functions in the VCAPI fall into two categories: synchronous and asynchronous. A synchronous function completes (successfully or otherwise) before returning control to the application, while an asynchronous function is initiated and immediately returns control to the application. In most cases, the asynchronous function signals its completion with an interrupt condition. Whenever an asynchronous function is requested, it is always possible to get a "Busy" return code, signifying that some asynchronous function in the same function set needs to complete before this function can be executed.

The Adapter

The adapter is a general-purpose, signal-processing co-processor connected to the host PC via shared memory.

The signal processor is a TMS32010. The adapter includes analog circuitry, digital control circuitry and two groups of fast static RAM (one for instructions; one for data).

The card contains connections for two telephone lines, one telephone instrument, a speaker and a microphone. These devices are multiplexed through two analog multiplexers which, in turn, are multiplexed to a single pair of A/D and D/A converters. The two analog multiplexers act as full-duplex I/O channels (corresponding to the two ports in the VCAPI), and thus provide a logical four-channel analog conversion subsystem, with two A/D and two D/A channels.

The data memory on the card is tri-ported; it can be accessed concurrently by the A/D and D/A converters, the signal processor and the PC.

All functions performed by the card are provided through signal processing code (SP code), and this code must first be down-loaded into the instruction RAM. Base functions of the VCAPI provide this capability.

The Signal Processing (SP) Control Program

The SP code executes on the signal processor under the control of a multi-tasking, interrupt-driven control program that resides in the instruction RAM.

The SP control program manages the instruction RAM in partitions and allows the SP code in the partitions to run concurrently, sharing the resources of the data RAM, the A/D and D/A converters and the signal processor.

The SP code in a partition corresponds to a function set in the API, and the API subsystem loads the SP code into the appropriate partition in response to API commands from the application.

In order to properly balance the resources on the card, no more than three partitions are ever used by the API subsystem. Restrictions on which function sets are allowed to operate concurrently are enforced. The three partitions include a base partition that executes commands in support of the base functions of the API, and it is loaded when the control program is loaded. The two other partitions are used to support the loadable function sets of the API.

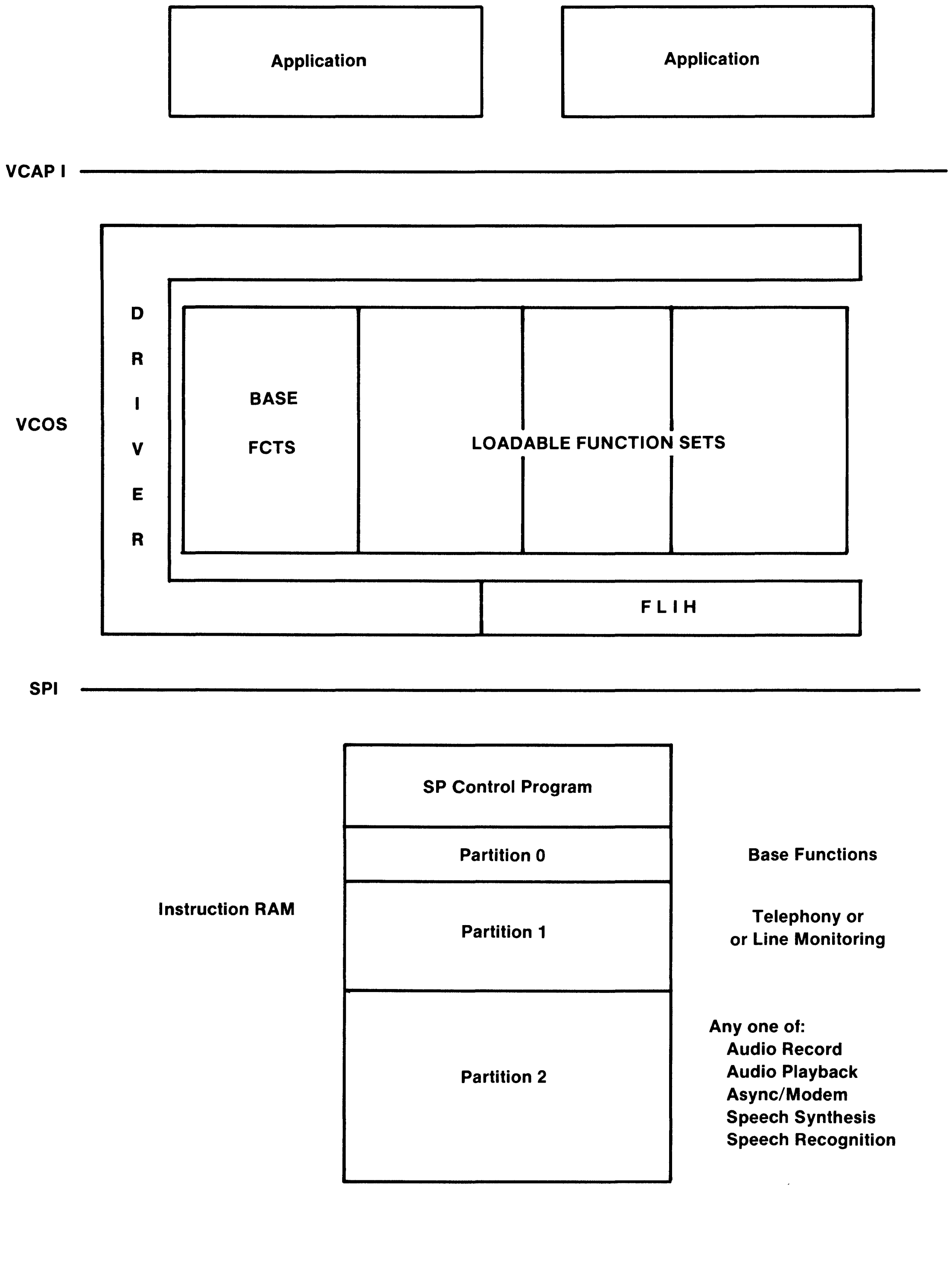
The VCAPI Subsystem - VCOS

The VCAPI is the interface to the Voice Communications Operating Subsystem (VCOS), which appears as a loadable system extension and executes on the PC processor. Functions in the VCAPI are invoked by issuing an INT 14H interrupt. The DX register identifies the partition associated with the function set, and the AX register is loaded with the appropriate function code within the function set. Any parameters required by the function are placed in a structure pointed to by the ES.BX registers.

The VCOS is composed of:

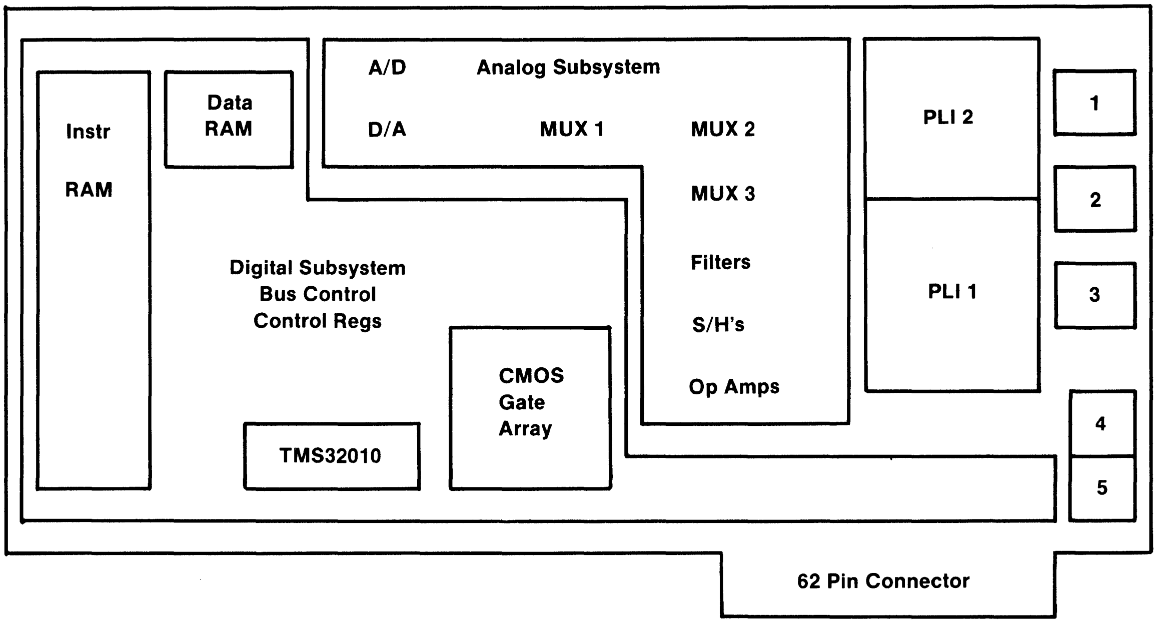
- A driver that routes the INT 14H commands to the appropriate function set support code and provides the functional interface to the SP control program.
- A First Level Interrupt Handler (FLIH) that routes all interrupts from the SP control program to the appropriate interrupt handler in the function set support code (the SLIHs).
- Support code for each function set. This code executes in cooperation with the SP code on the adapter to interpret the API commands invoked by the application.

  The base support code operates in conjunction with the base partition; the other function sets operate with the corresponding dynamically loaded partition.


Application
Application
VCAP I
VCOS
DRIVER
BASE
FCTS
LOADABLE FUNCTION SETS
FLIH
SPI
SP Control Program
Partition 0
Partition 1
Partition 2
Instruction RAM
Base Functions
Telephony or
or Line Monitoring
Any one of:
Audio Record
Audio Playback
Async/Modem
Speech Synthesis
Speech Recognition

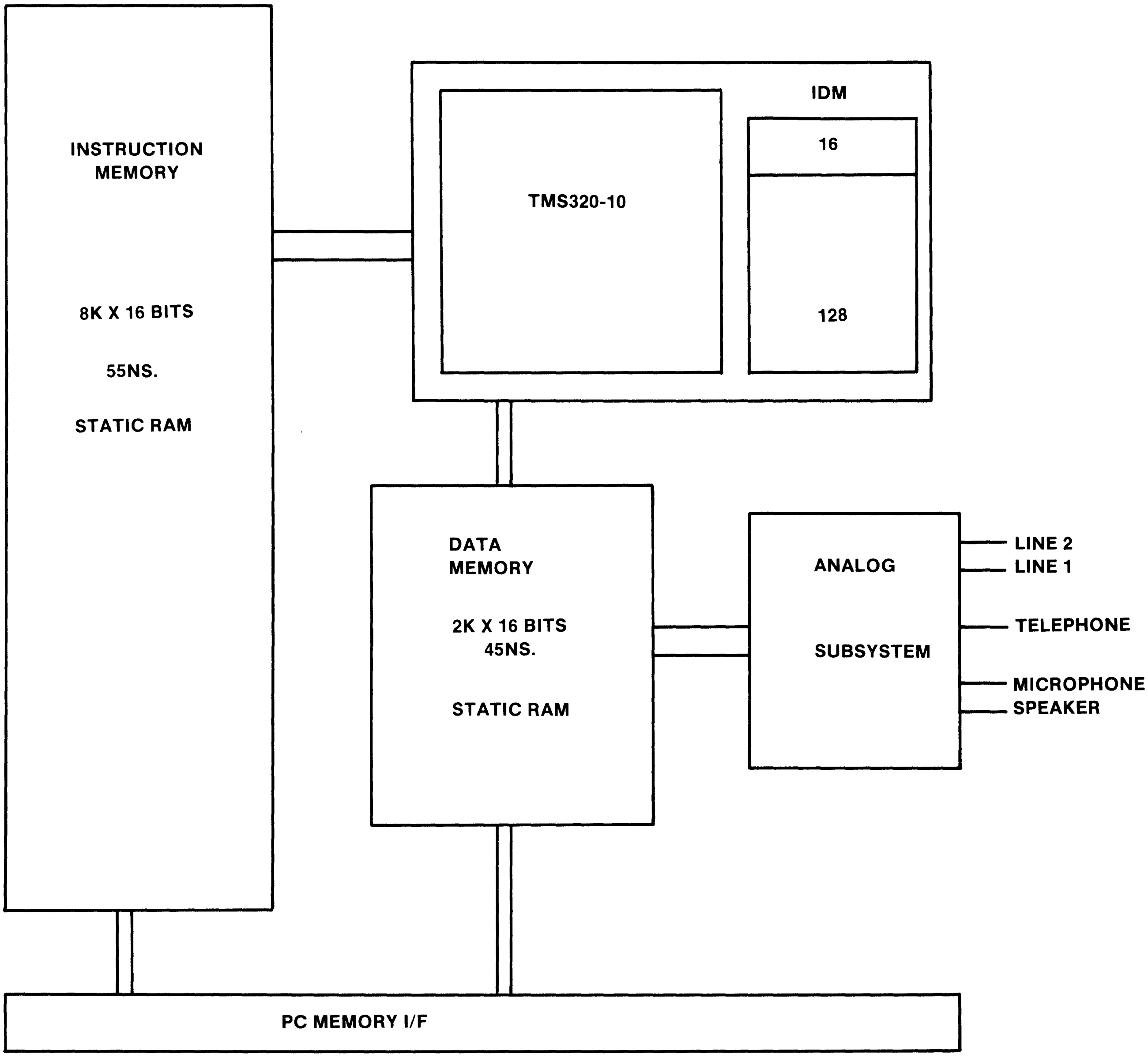
Hardware

Physical Layout



1. RJ11C Connector for Telephone Line 2.
2. RJ11C Connector for Telephone Line 1.
3. RJ11C Connector for Telephone Instrument.
4. Audio Subminiature Jack for Microphone.
5. Audio Subminiature Jack for Speaker.

Functional Block Diagram



Internal Data Memory (IDM) on the TMS320 chip consists of 144 words; 16 of which are reserved for the control program and 128 for use by the partitions.

Hardware Interface

The hardware is essentially nonfunctional until its instruction memory is loaded and the signal processor started. The interface between the PC and the hardware consists of:

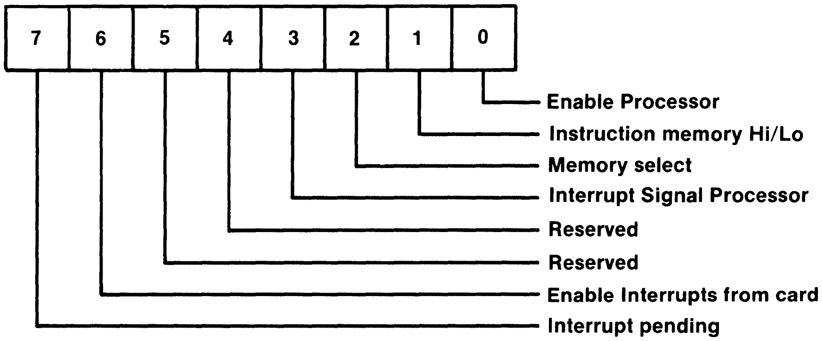
1. An I/O interface
2. A memory interface
3. An interrupt interface

I/O Interface

The I/O interface is provided by INs and OUTs to a one-byte I/O control register mapped to the PC I/O address space. Jumpers on the card select the I/O address of the control register from any one of four addresses:

```
021F - no jumpers (required for Rel. 1.0)
221F
421F
621F
```

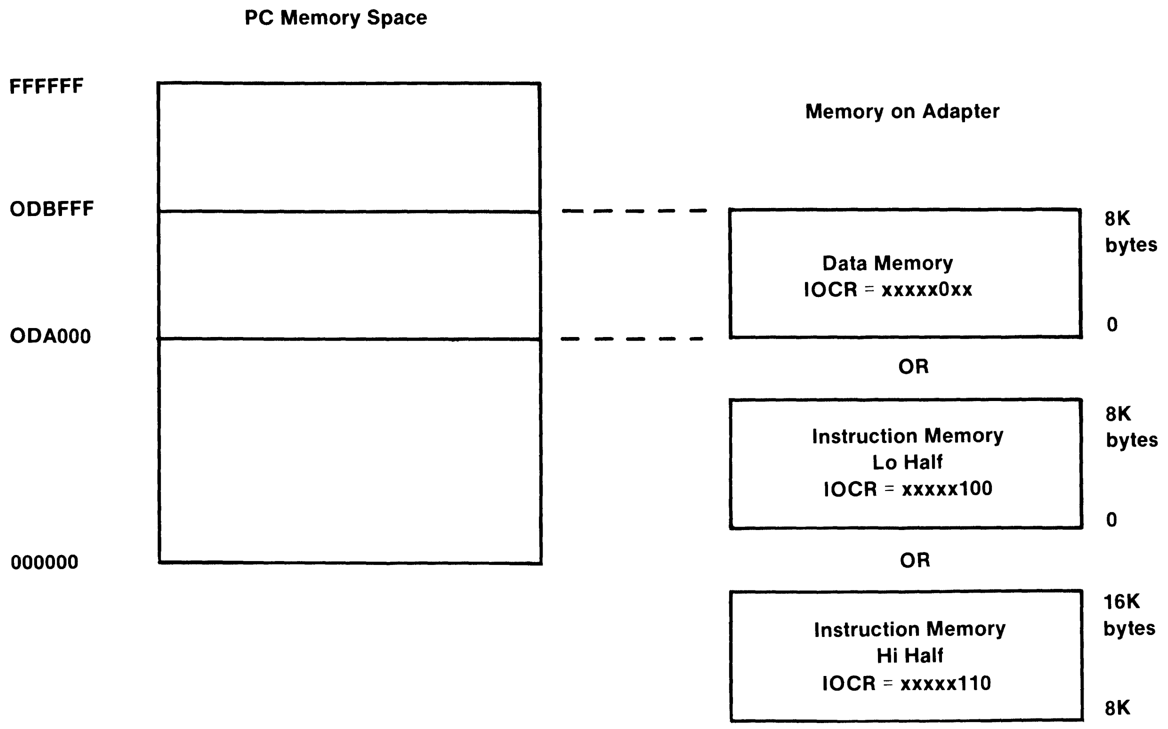
I/O Control Register (IOCR)



```
Bit 0 = 0  Hold Signal Processor in reset state
      = 1  Start Signal processor at instruction memory location 0

Bit 1 = 0  Map instruction memory 0 - 8K to PC memory bus
      = 1  Map instruction memory 8K - 16K to PC memory bus
           (Note: Bit 1 in effect only if Bit 2 = 1)

Bit 2 = 0  Map Data memory to PC memory bus
      = 1  Map Instruction memory to PC bus as controlled by Bit 1
           (Note: Signal processor must be in the reset state in order to
           write directly to instruction memory)

Bit 3 = 0  Interrupt from PC acknowledged
      = 1  Interrupt from PC to signal processor
           (Note: the VCOS does not use this facility)

Bit 6 = 0  Disable interrupts from the Adapter
      = 1  Enable interrupts from the adapter

Bit 7 = 0  Interrupt from adapter acknowledged
      = 1  Interrupt from adapter pending
```

There is no other function interface to the hardware except for IN and OUT to the IOCR. The VCOS only uses INs and OUTs to IPL the SP control program, excercise the diagnostics, and during normal operation, to acknowledge interrupts from the adapter. The command interface between the VCOS and the SP control program is simply a mailbox convention in data memory. This will be described later.

Memory Interface



The same jumpers used to select the I/O address also select the base memory address that is used to map the adapter's memory to the PC memory bus. These values are:

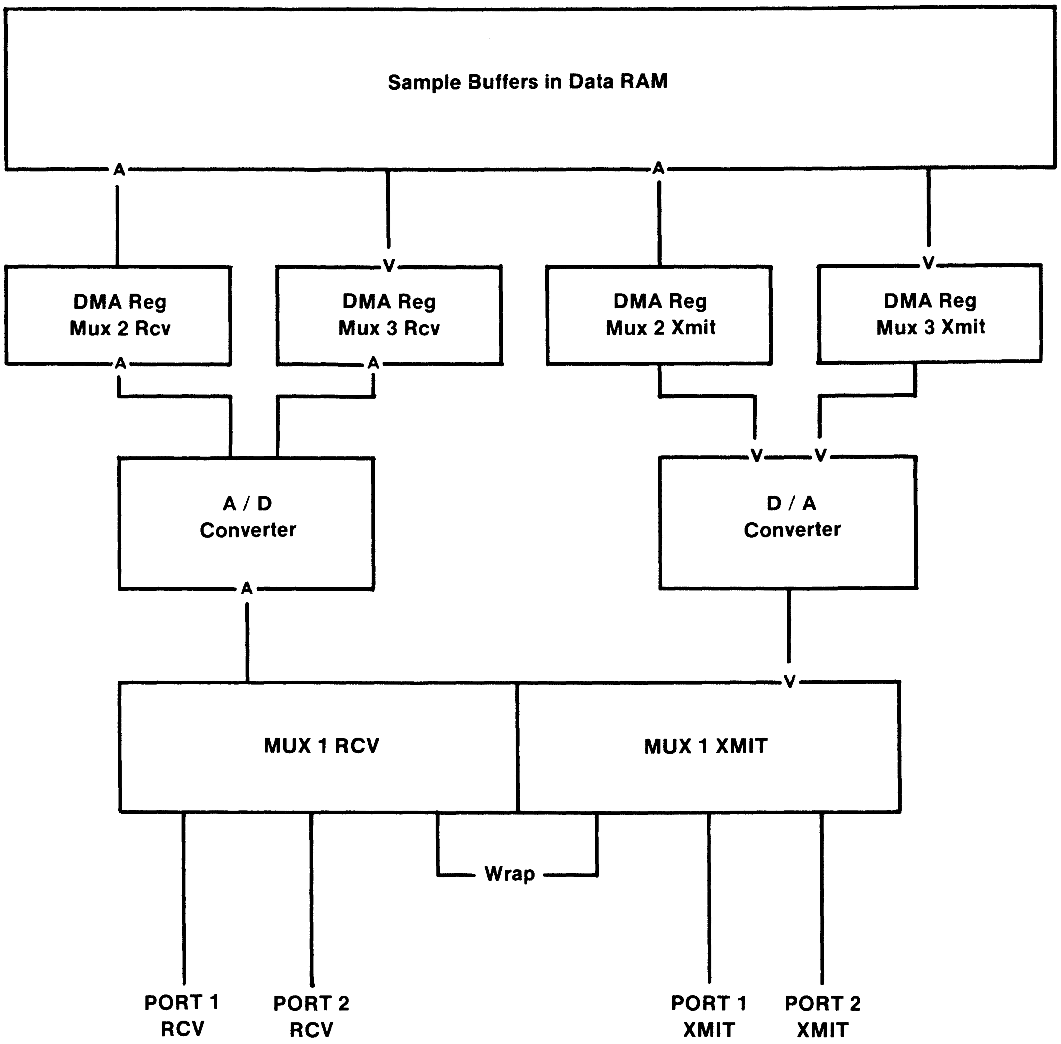
```
ODA000 - no jumpers (required for Rel. 1.0)
OD8000
ODC000
ODE000
```

Interrupt Interface

The SP code on the card may cause the adapter to interrupt the PC on any one of four interrupt levels selected by jumpers on the card: 2, 3, 4 or 7. These interrupts may be disabled by setting IOCR bit 6 to 0. If the interrupts are enabled, the signal processor sets IOCR bit 7 to 1, indicating that an interrupt from the card is pending and interrupting the PC on the jumpered interrupt level. The interrupt handling mechanism in VCOS uses the pending bit in the IOCR to support the hardware interrupt level sharing scheme with other adapters and software that follow this convention. This convention is described in the latest edition of the *IBM Personal Computer AT Technical Reference* Manual.

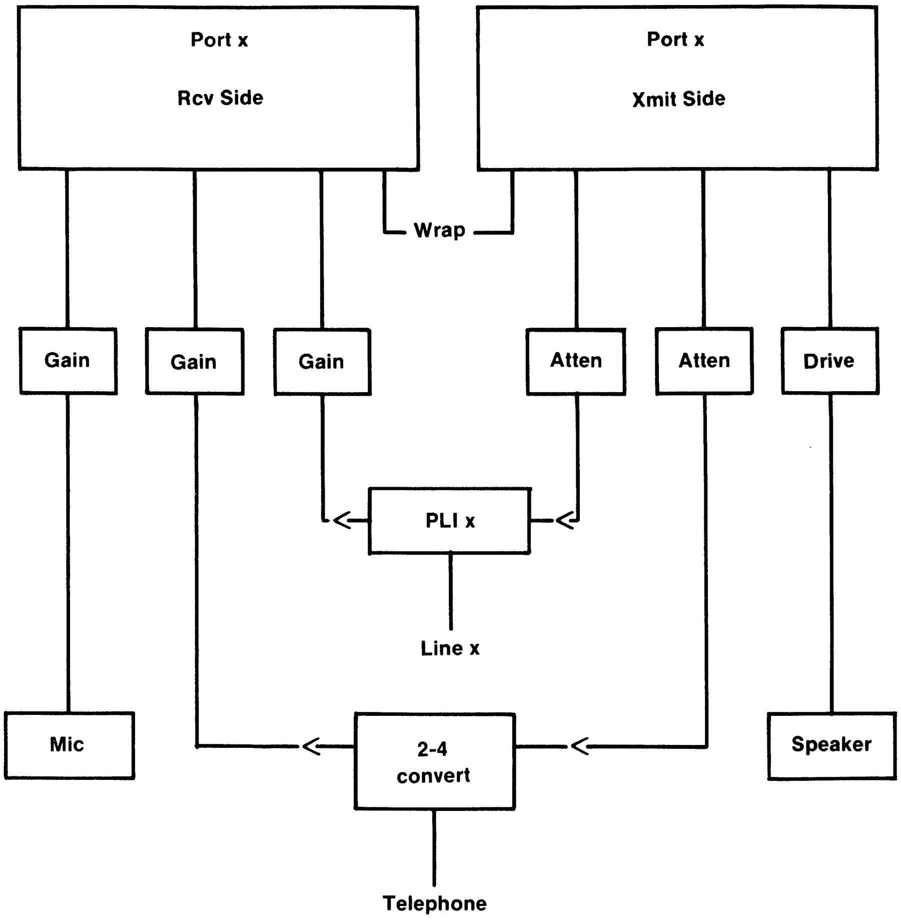
Any further information transfer from the adapter to the PC on an interrupt would be a convention between the SP code on the adapter and the PC software. The particular convention between the SP control program and the VCOS will be discussed later.

Analog Functional Block Diagram



The basic sampling rate of the A/D and D/A converter can be set by SP code to either 8000Hz or 9600Hz (the SP control program always sets this to 9600Hz). Within each cycle, four digital samples are processed: two through the D/A converter and two through the A/D converter. The four samples being multiplexed by Multiplexer (Mux) 1 are from the two ports represented by Mux 2 and Mux 3. Where the samples are located in data memory is controlled by four Dynamic Memory Addressing (DMA) registers, each of which cycles around a 16-sample buffer. The input/output devices being used are those currently connected to the ports. Each sample is 12 bits, and at the end of each 16-sample cycle the analog subsystem interrupts the TMS320. This 16-sample cycle (1.67ms) provides the SP control program with its basic time-slicing mechanism.

The I/O Ports



Port 1 is represented by Mux 2; Port 2 by Mux 3.

Control registers exist to:

- Enable/Disable each half of Mux 2 and Mux 3.
- Specify which device is attached to Mux 2, or wrap.
- Specify which device is attached to Mux 3, or wrap.
- Wrap the A/D and D/A converters.

The SP Control Program

The SP control program performs a number of related functions:

1. Interrupt Handler

   The interrupt handler is triggered by the interrupts from the A/D and D/A converter, which occur every 16 samples (1.67ms at 9600Hz). The interrupt handler passes control, in sequence, to the interrupt subroutine in each partition, which process the samples.

2. Time Slice Supervisor

   The time slice supervisor uses the remaining time in each 1.67ms cycle to route commands from the PC to the appropriate command processors in the partitions, and it then gives the remaining time to background sample processing.

3. Partition Services

   A number of services are provided to the SP code in the partitions. These services (SVCs) are invoked by calling the appropriate entry in a branch table located in the control program.

   - Post Status - Send status to the PC with an Interrupt
   - Activate Interrupt Subroutine
   - Deactivate Interrupt Subroutine
   - Activate Background Task
   - Deactivate Background Task
   - Activate A/D D/A
   - Deactivate A/D D/A
   - Give up Background (for Resume)
   - Give up Background (for Restart)
   - Set 8000Hz
   - Set 9600Hz
   - Set Gain
   - Reset Background entry point; resume address

4. Telephone Line Protection

   The control program performs two monitoring functions to comply with FCC regulations:

   a. three-second averaging

      The control program constantly monitors the signals being sent to the telephone lines, and if they exceed the amplitude limits set by the FCC for a three-second period, they will be attenuated.

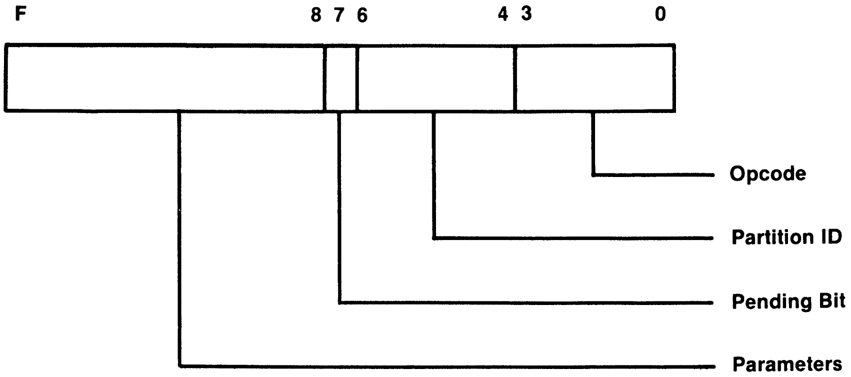
   b. two-second billing delay

      The control program will force any outbound data to silence for two seconds after the phone line has gone off-hook on all incoming calls. This situation is detected when an OFFHOOK occurs within eight seconds of a ring-in signal. This does not stop the user speaking into the telephone for this two-second period.

Interface to Control Program

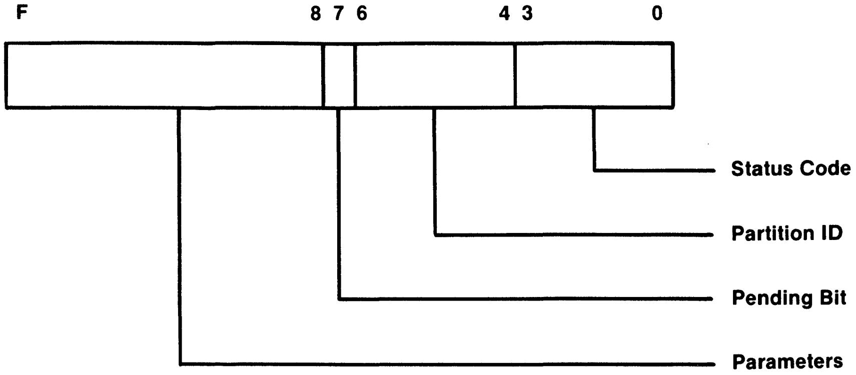
The interface between the VCOS and the SP control program is a pair of mail boxes located in the shared data RAM. One mail box is the command word to allow the subsystem to send commands to the SP code; the other is the status word to allow the SP control program to respond to the subsystem. Each mail box contains a pending bit to indicate that there is something in the box. The bit will be set when the information is placed in the box and reset when the contents are acknowledged. No more command/status words will be placed in a box if it is full. The subsystem polls the command pending bit when it wishes to send another command; the SP control program stacks up to four status responses if the status pending bit is on. Each time the SP control program sets the status pending bit, it also interrupts the PC on whatever level is set by the jumpers on the adapter.

Command Word



The command word forms a pseudo machine language for the machine comprising the SP code executing on the adapter card.

Status Word



As can be seen from the structure of the command and status words, the control program can, in fact, support a structure of eight partitions. Partition 0 is always the base partition, and it processes commands that support the base functions of the API. Only two further partitions, partition 1 and partition 2, are used by VCOS. Partition 1 can contain the SP code to support either the telephony or line monitoring function sets, while partition 2 can contain the SP code to support any one of the other function sets. These partitions execute concurrently.

Partitions

Each partition has a control block in the data RAM that contains certain configuration information for the SP code currently running in that partition, as well as local-state information for the execution of the partition. The control block is initialized by VCOS when a partition is loaded with SP code during the connection of a function set. It may be modified by VCOS when the application makes configuration changes to the function set while it is connected.

The partition code consists of command processors, an interrupt subroutine and a background task.

Command Processors

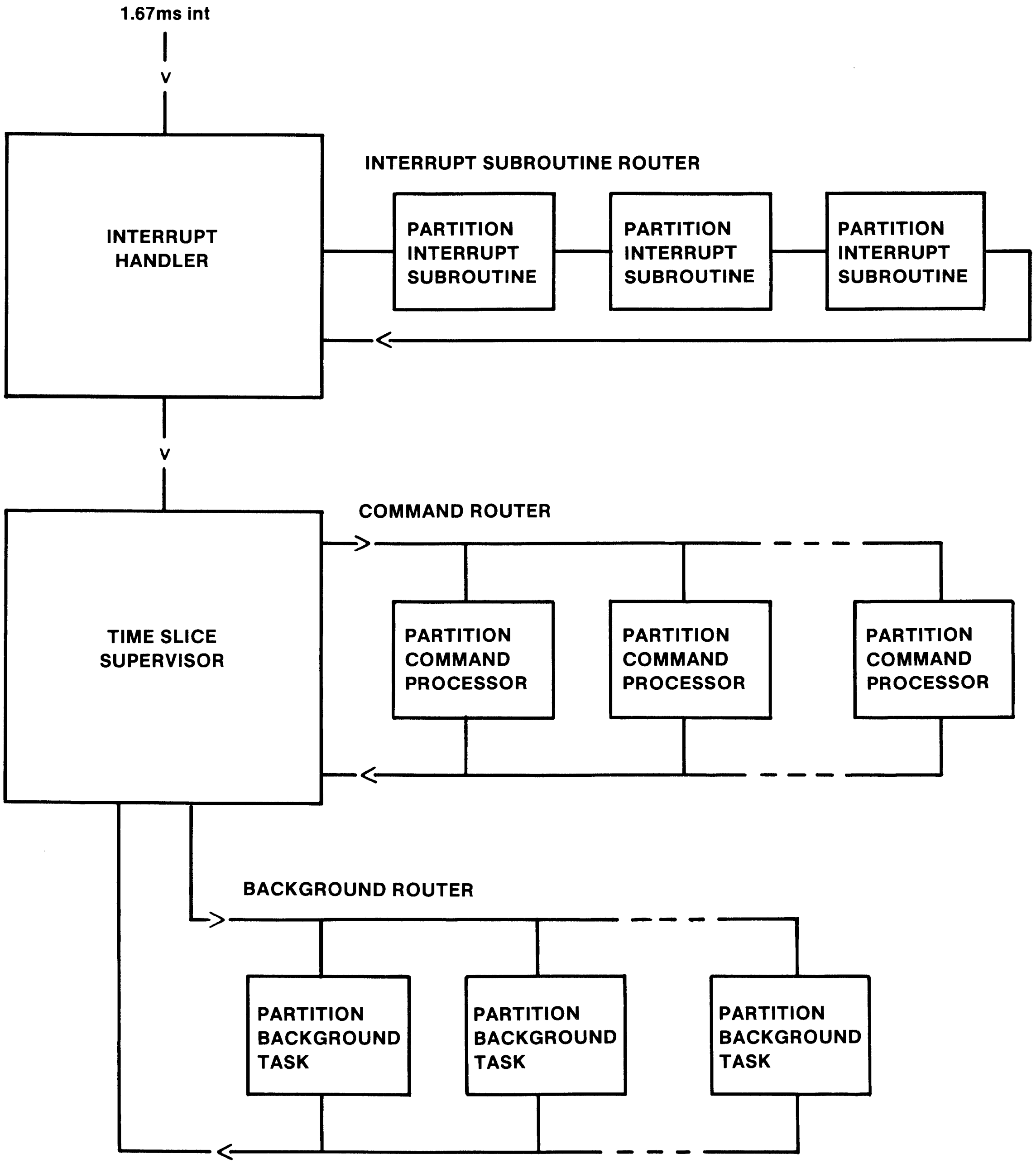
The command processors interpret commands from the command word directed to their partition. The command router, which is part of the SP control program, reads the command word and passes control to the command processor which is to handle the command. The control program gives control to the command router whenever a command is pending. The control program continues to give control to the command processor until it has completed the command or initiated a background task to do so. At this point, the command processor returns to the command router, which acknowledges the command by resetting the command pending bit in the command word.

Interrupt Subroutine

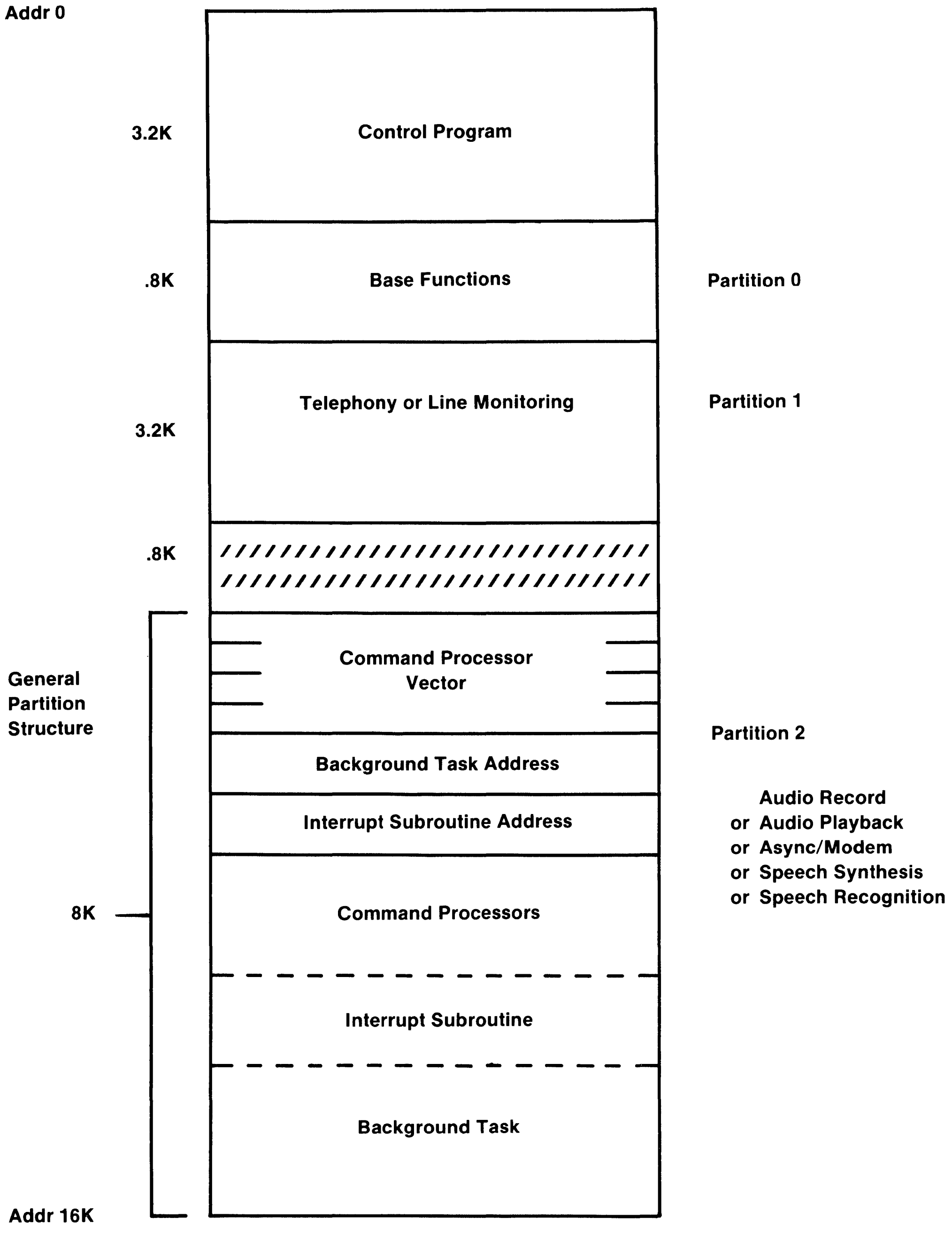
The interrupt subroutine is the code that must execute within one interrupt interval (1.67ms at 9600Hz) and process the next 16 samples. The interrupt subroutine is activated with a system service (SVC) call to the control program. Once active, it will be called every 1.67ms cycle until an SVC call is made to deactivate it.

Background Task

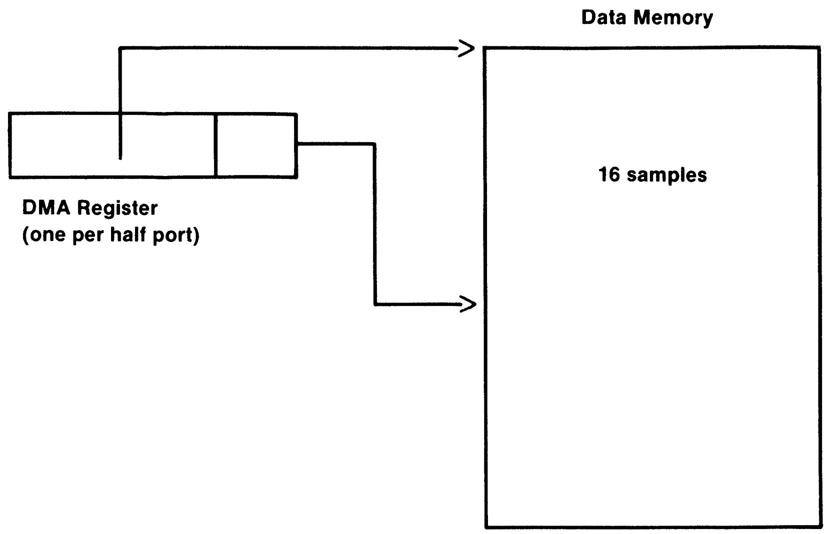
The background task is all the other code in the partition. A background task gets control only after the interrupt subroutines and command processors have completed. The amount of time remaining in the 1.67ms cycle is considered to be one time slice available to dispatch the next eligible background task in the partitions. Control is passed to the background tasks by the background router. A background task can set the number of consecutive time slices it needs in order to run successfully. When this number expires, the background task's state is saved, to be restored when it is next dispatched. A background task may give up its time slices voluntarily if it cannot do useful work until more interrupts occur.


1.67ms int
INTERRUPT HANDLER
INTERRUPT SUBROUTINE ROUTER
PARTITION INTERRUPT SUBROUTINE
PARTITION INTERRUPT SUBROUTINE
PARTITION INTERRUPT SUBROUTINE
TIME SLICE SUPERVISOR
COMMAND ROUTER
PARTITION COMMAND PROCESSOR
PARTITION COMMAND PROCESSOR
PARTITION COMMAND PROCESSOR
BACKGROUND ROUTER
PARTITION BACKGROUND TASK
PARTITION BACKGROUND TASK
PARTITION BACKGROUND TASK

Instruction RAM Map



Sample Buffer Management



The A/D and D/A converter processes 16 samples for each half of the two ports and interrupts the processor for each of the four sets. For each half-port there is a DMA control register that points to a ring buffer of 16 words. The control program sets the initial values of these registers, and the converter rotates around the lower four bits. The control program, by suitably setting the registers, actually directs the samples into a ring buffer whose size is specified by the SP code in the partition handling the particular sample stream. These ring buffers can vary in length from 32 samples up to 192 samples. On each interrupt from the converter, the control program switches the corresponding DMA register to the next set of 16 samples in the buffer. By synchronizing the setting of the DMA registers, the control program synchronizes the four interrupts within one 80-microsecond interval, using the last of these interrupts to trigger its Interrupt Handler.

The SP code in a partition thus has three basic strategies to choose from:

1. Completely processing 16 samples in its interrupt subroutine; switching between two sets of samples with a ring buffer of 32 samples.

2. The same as in 1, except that the interrupt subroutine copies the 16 samples into temporary storage to be processed in the background.

3. Processing a larger ring buffer in the background, using its interrupt subroutine to monitor the progress through the buffer.

VCOS Structure

The VCOS is loaded as a system extension using VCAPIDRV.COM. This would typically be done from an AUTOEXEC.BAT file, but must be done before TopView is loaded.

VCAPIDRV.COM loads the driver and the hardware interrupt service routine, if they have not already been loaded, and then:

- Allocates space for the control block structure

- Loads the VCAPI directory APIDIR.AIC

- Loads and runs PC code to drive the basic acceptance tests for the adapter

- Loads the base function set along with the SP control program and base partition

- Chains the hardware interrupt router into the DOS vector

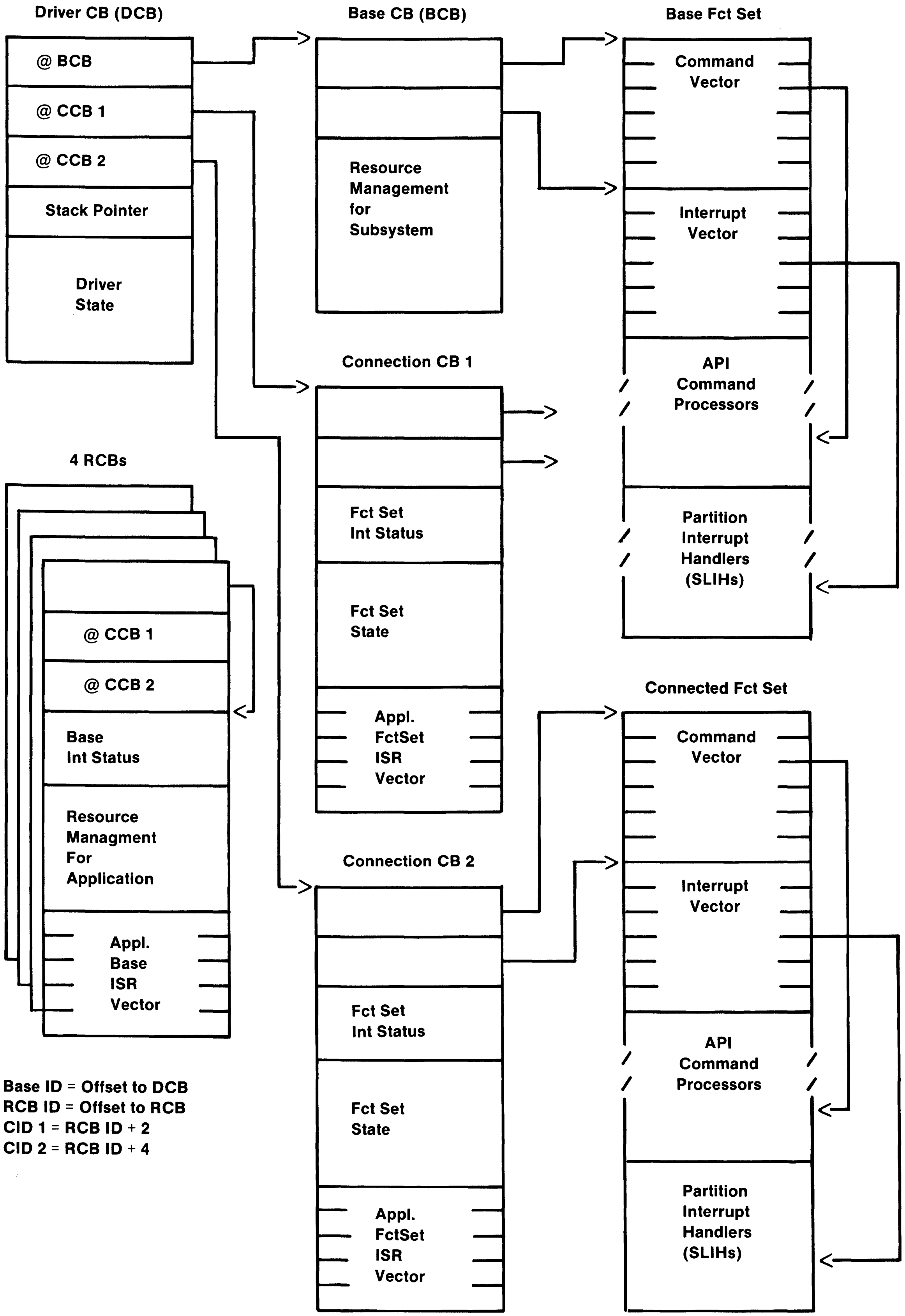
- Chains the interrupt 14H router into the DOS vector

- Preloads any requested function sets into memory (discussed later)

- Calls the base code to initialize the control block structure

- Terminates and stays resident

Control Block Structure



Base ID = Offset to DCB
RCB ID = Offset to RCB
CID 1 = RCB ID + 2
CID 2 = RCB ID + 4

Resource Control Blocks (RCBs)

An RCB handle is returned by OPEN and is used by an application to request services of the VCOS.

The number of RCBs represents the number of concurrent services that the subsystem can provide. The limit of four (which is a restriction of the implementation, not the VCAPI) is governed by the extent to which concurrent operations can be supported. Four allows:

- Two applications using partitions 1 and 2 independently; for example, partition 1 being used to place a telephone call at the same time that partition 2 is being used for speech recognition.
- One or two applications behaving as auto-answer applications (for data and/or voice calls). Each has an open RCB which owns the ring-in interrupt for a particular line. Such an application owns no other resources until a call comes in.

An RCB records:

1. The resources claimed on its behalf.
2. The state of the hardware interrupts directed to it, according to the hardware resources that it owns.
3. The addresses of the application interrupt service routines for these interrupts.

Connection Control Blocks (CCBs)

Since the subsystem allows up to two loadable function sets to be connected at any point in time, there are two control blocks that represent those connections. CCB 1 corresponds to partition 1; CCB 2 to partition 2. A handle, CID 1, is returned by OPEN for the application to refer to CCB 1; the handle for CCB 2 is CID 2. A CCB contains:

- The addresses of the command and interrupt vectors in the support code for the function set connected to the corresponding partition.
- The interrupt status for that function set.
- The state of the function set.
- The addresses of the application interrupt service routines for the interrupts generated by the function set.

Each RCB points to the CCBs corresponding to the partitions that it owns (if any).

Base Control Block (BCB)

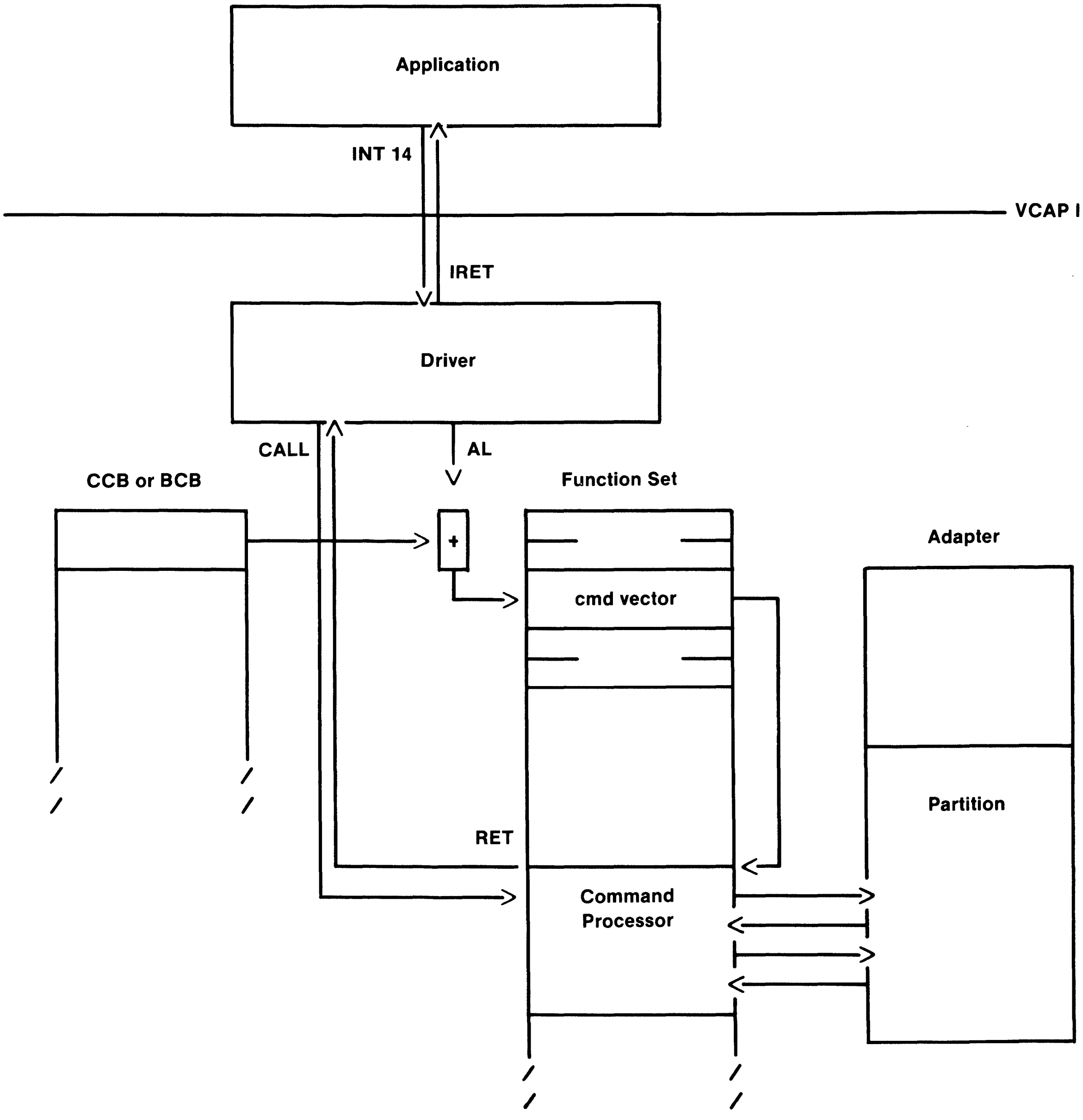
A base control block whose handle is returned on OPEN, serves as a connection control block for partition 0. Partition 0 is always active and available through any RCB. As well as identifying the base function set support code, the BCB is used to maintain the overall state of the subsystem's resources, independent of which RCB owns them.

Driver Control Block (DCB)

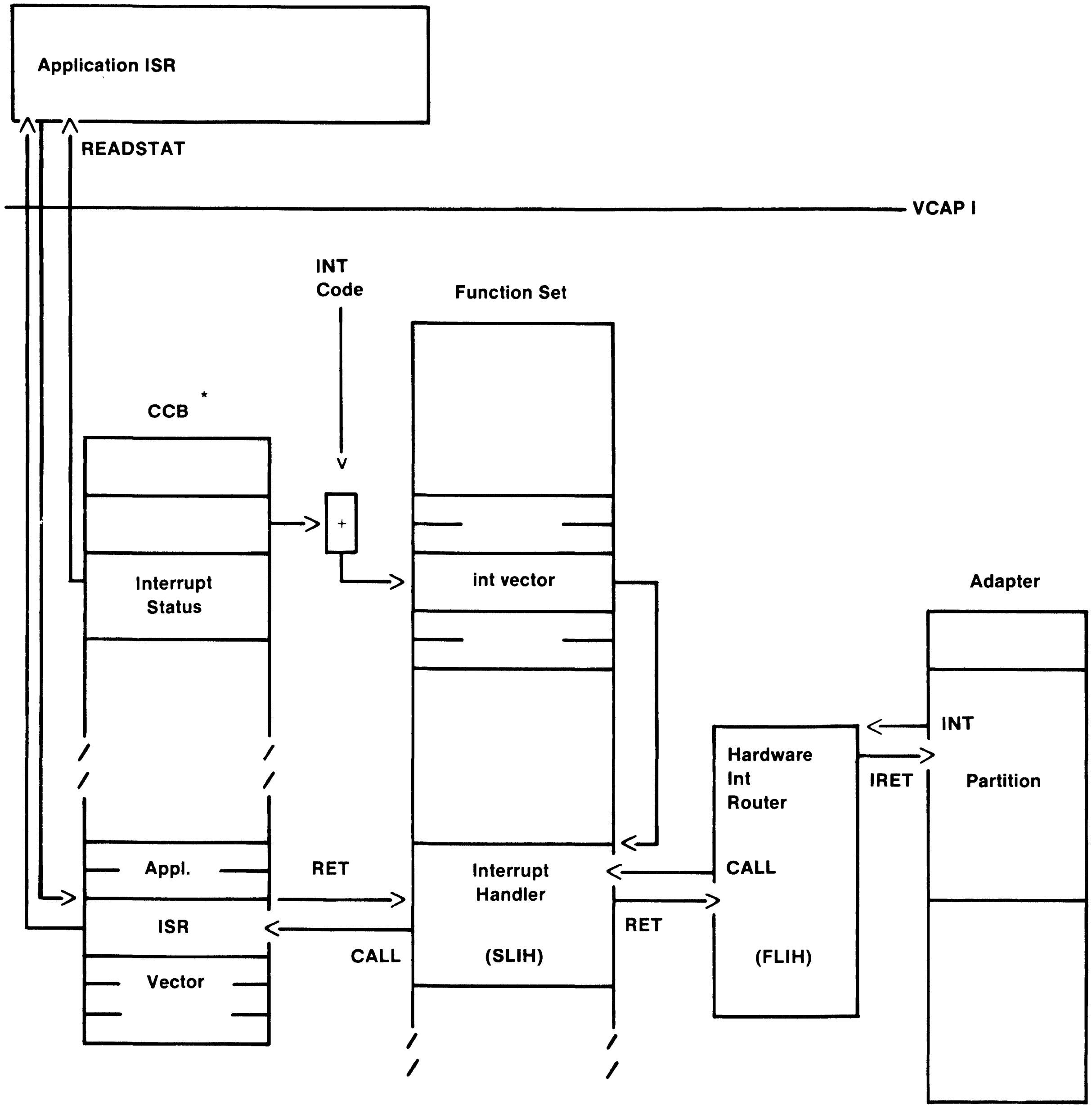
The Driver Control Block, which is the root of the control block structure, is used to:

- Locate the other control blocks.
- Maintain the state of the adapter interface.
- Locate the stack which is used to invoke API commands and function set SLIHs.

Command Mechanism



Interrupt Mechanism



* The application ISR vector is maintained in the application's RCB for base interrupts.

Note that the function set interrupt handler and the application ISR execute at interrupt level.

Function Set Management

A function set consists of two DOS files: one file for the PC code and one for the SP code. The VCAPI command to activate a function set is CONNFTOP (connect function set to port). This command causes both files to be loaded into PC memory and the SP code to be down-loaded to the instruction RAM on the card. This down-loading is a memory-to-memory move performed by the base partition. It is, however, complicated by the fact that it may be performed while the signal processor is executing in another partition. A typical partition takes no more than 50ms to down-load. If an application wants to avoid the disk I/O involved in connecting a function set, it may do so by pre-loading the function set into PC memory at some convenient initialization time. The VCAPI command to do this is OPTSET (optimize function set). It is possible to pre-load selected function sets when the subsystem is loaded by specifying their IDs on the VCAPIDRV command. For example:

```
VCAPIDRV /o 1, 3, 5, 8, 9, 10, 11
```

Once pre-loaded, the functions sets remain in PC memory until the VCOS is reset, either by IPL or by VCAPIDRV /r. To run in a TopView environment, the VCOS and all required function sets must be loaded before TopView.

No more than one copy of a function set can be resident in PC memory at the same time, even though two or more applications have optimized it. Similarly, a function set will never be freed while it is still active, even though all outstanding optimize requests have been cancelled (by DEOPTSET). This is achieved by maintaining independent counts for OPTSET and CONNFTOP for each function set; only when both counts are reduced to zero will a function set be freed from PC memory. Controls are also placed in the RCBs to prevent one application interfering with another; i.e., one application can neither disconnect nor deoptimize another's function set.

The focal point for this function set management is the API directory. The directory is delivered as a DOS file (APIDIR.AIC), and is loaded when the API subsystem is initialized. Each entry in the directory is 84 bytes long:

VCAPI Directory Entry Format

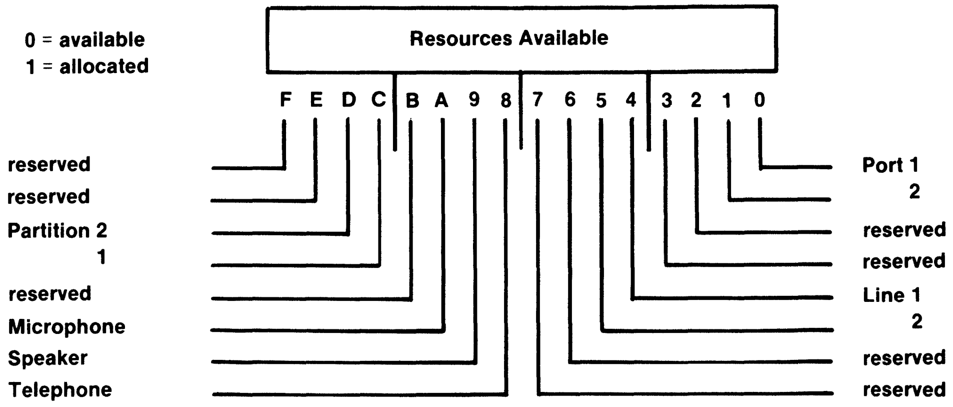
Byte#	Value	
1-2	80Fx	Basic Acceptance Tests
	8000	Base Function Set
	PPFF	PP = Partition number
		FF = Function Set number
		01 = Line Monitoring
		03 = Telephony
		05 = Async/Modem
		08 = CVSD/28.8 Record
		09 = CVSD/28.8 Playback
		0A = Speech Synthesis
		0B = Speech Recognition
		0C = CVSD/19.2 Record
		0D = CVSD/19.2 Playback
		0E = CVSD/14.4 Record
		0F = CVSD/14.4 Playback
3-4	EC level	
5-6	Version number	
7	Reserved	
8	4xH	SP code can be loaded dynamically
	8xH	SP code can only be loaded in reset mode
9-21	DOS filename of PC code	
22-23	Size in paragraphs of PC code	
24-36	DOS filename of SP code	
37-38	Size in paragraphs of SP code	
39-70	Default configuration values for function set	
71-72	PC segment address of where PC code is loaded	
73-78	Reserved	
79-80	PC segment address of where SP coded is loaded in PC	
81-82	Reserved	
83	Connection count for function set	
84	Optimize count for function set	

Resource Management

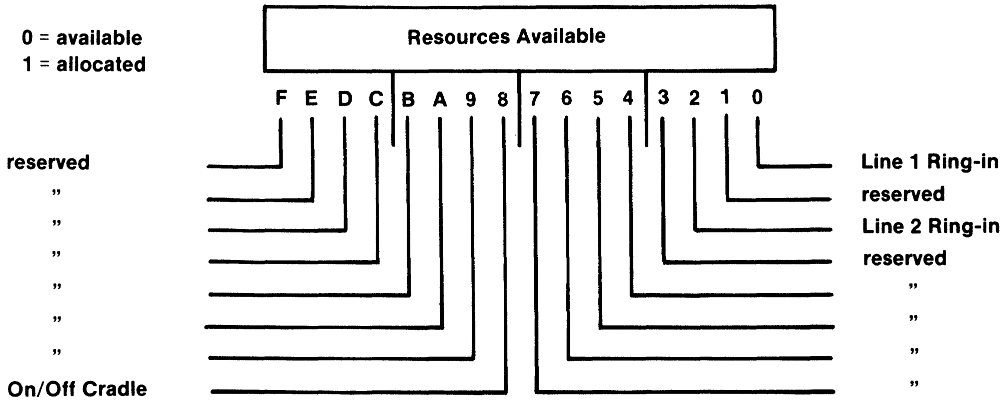
The BCB is used to keep track of the allocation and interconnection of all resources owned by the subsystem, while each RCB keeps track, in a set of similar structures, the allocation and interconnection of the resources owned by the application.

BCB/RCB Resource Allocation Flags

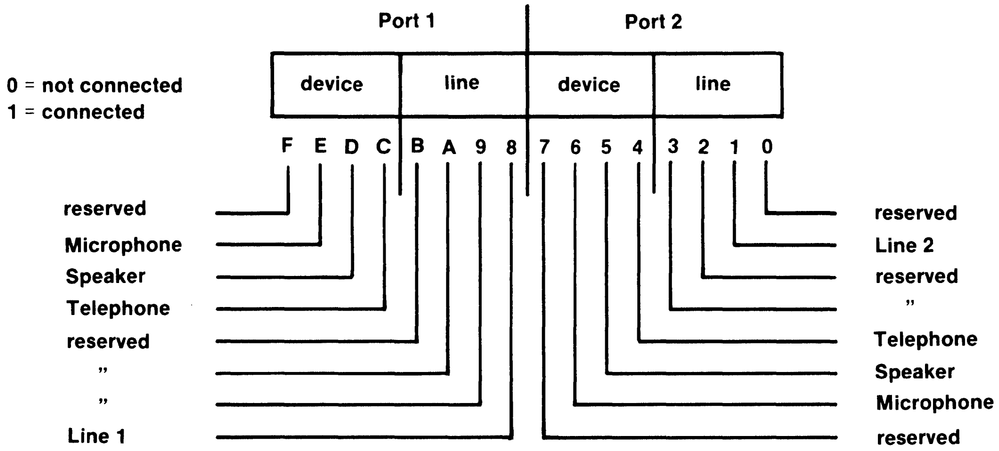
Hardware



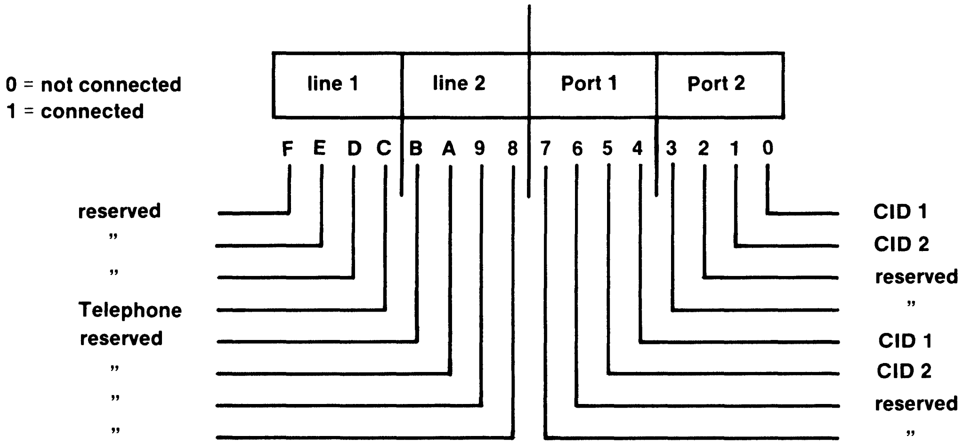
Interrupts



Device to Port Connections



Device to Line and Partition to Port connections



Linkage to the VCAPI

Assembly Linkage

```
        OPEN                         Others

In        AH = 11H                     AH = 11H
          AL = 11H                     AL = Fct-Code
          DX = 021FH                   DX = Base ID or CID (from OPEN)
       ES.BX = A(PLIST)             ES.BX = A(PLIST)
         INT 14H                      INT 14H


Out       AX = Return Code             AX = Return Code
       PLIST = Return Code          PLIST = Return Code
               RCB_ID                             :
               BASE_ID                      function dependent
               CID1
               CID2
```

High-Level Language Linkage

In order to facilitate the use of the interface from high-level languages, an interface module should be provided for each compiler used. (Actually a module is required for each high-level language linkage convention; if two compilers use the same linkage convention, they can share the same interface module.) For example:

BASIC Interpreter

```
OPEN      CALL BAPIFCT(CARD#, FCT_CODE, PLIST)
Others    CALL BAPIFCT(FS_ID, FCT_CODE, PLIST)
```

BASIC Compiler

```
OPEN      CALL ABSOLUTE(CARD#, FCT_CODE, PLIST, BAPIFCT)
Others    CALL ABSOLUTE(FS_ID, FCT_CODE, PLIST, BAPIFCT)
```

"C"

```
OPEN     ..CAPIFCT(CARD#, FCT_CODE, SIZE, &PLIST)...
Others   ..CAPIFCT(FS_ID, FCT_CODE, SIZE, &PLIST)...
```

Where:

FS_ID   Is a 16-bit integer used to identify function set. (Either Base ID or a Connection ID.)

FCT_CODE   Is a 16-bit integer whose low-order eight bits identify the function relative to the function set.

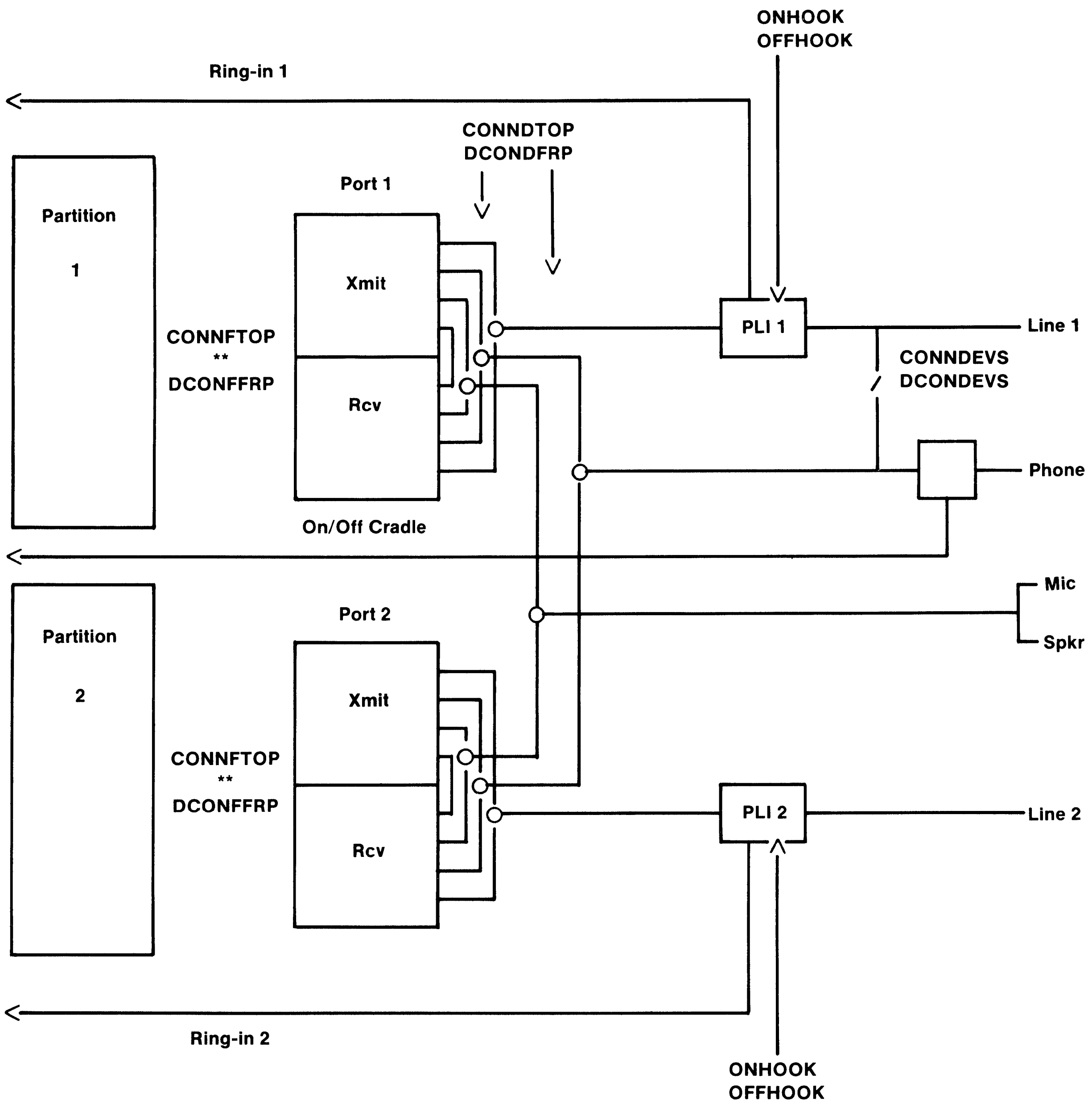
CARD#   Is a 16-bit integer containing the value 021FH

SIZE   Is a 16-bit integer whose value indicates the address length of &PLIST

  1 = 16-bit offset relative to DS
  2 = 32-bit address

PLIST   In general, a structure containing the parameters of the function being called. Each function defines its own parameters, but the first parameter in all cases is a 16-bit integer in which will be returned a return code. (Note that this return code is also returned in the AX register.)

Base Function Set



** A partition can access either port
Both partitions can access the same port

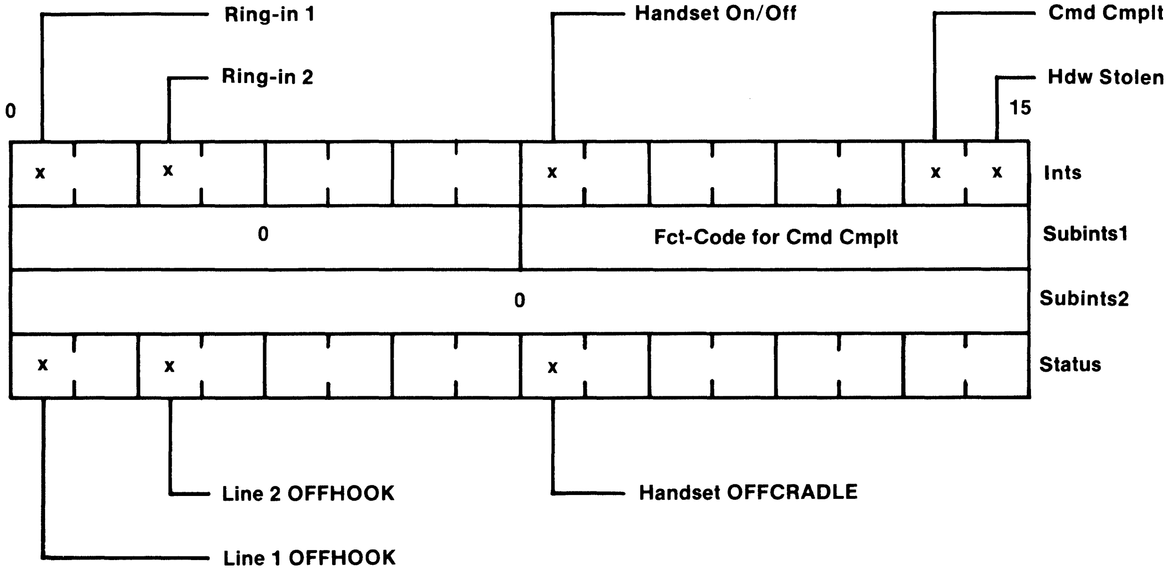
Command Summary

AL xxH	Command Name	Function
11	OPEN	Obtain access to the subsystem
12	CLOSE	Relinquish access to the subsystem
16	ESTINT	Establish ISR for some specified base interrupts
17	MASKINT	Enable/Disable base interrupts
18	READSTAT	Read/Poll base interrupt status
19	READCONN	Examine hardware connection status
1A	CLAIMHDW	Seize specified hardware/interrupt resources
1B	FREEHDW	Free specified hardware/interrupt resources
1C	READHDW	Test availability of hardware/interrupt resources
1D	OPTSET	Pre-load a function set
1E	DEOPTSET	Release a function set
1F	CONNFTOP	Activate a function set for operation at a specified port
20	DCONFFRP	De-activate a function set
21	CONNDTOP	Connect a device/line to a port or wrap port
22	CONNDEVS	Connect telephone to line 1
23	DCONDEVS	Disconnect telephone from line 1
25	ONHOOK	Place specified line "On-Hook"
26	OFFHOOK	Place specified line "Off-Hook"

Interrupt Summary

Mask Bit	Name	Event
0	Ring-in 1	One interrupt for each ring-in signal on line 1
2	Ring-in 2	One interrupt for each ring-in signal on line 2
8	Handset	Handset placed on or removed from cradle
14	Cmd Cmplt	An asynchronous command completion
15	Hdw Stolen	A hardware/interrupt resource has been stolen

Interrupt Status



Line Monitoring

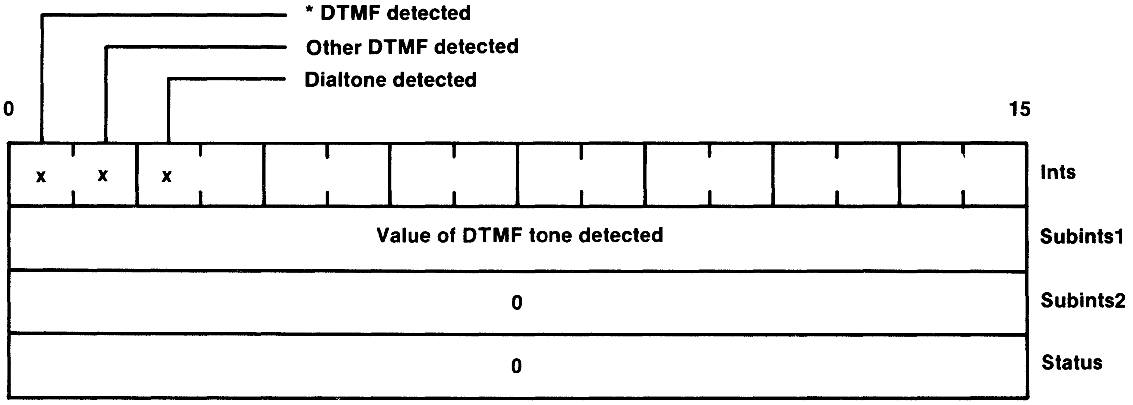
Command Summary

AL xxH	Command Name	Function
13	DCONFIG	Establish function set's default configuration
14	RCONFIG	Read function set's current configuration
15	CCONFIG	Change function set's current configuration
16	ESTINT	Establish ISR for some specified interrupts
17	MASKINT	Enable/Disable specified interrupts
18	READSTAT	Read/Poll interrupt status
19	GENTONES	Generate a pair of tones

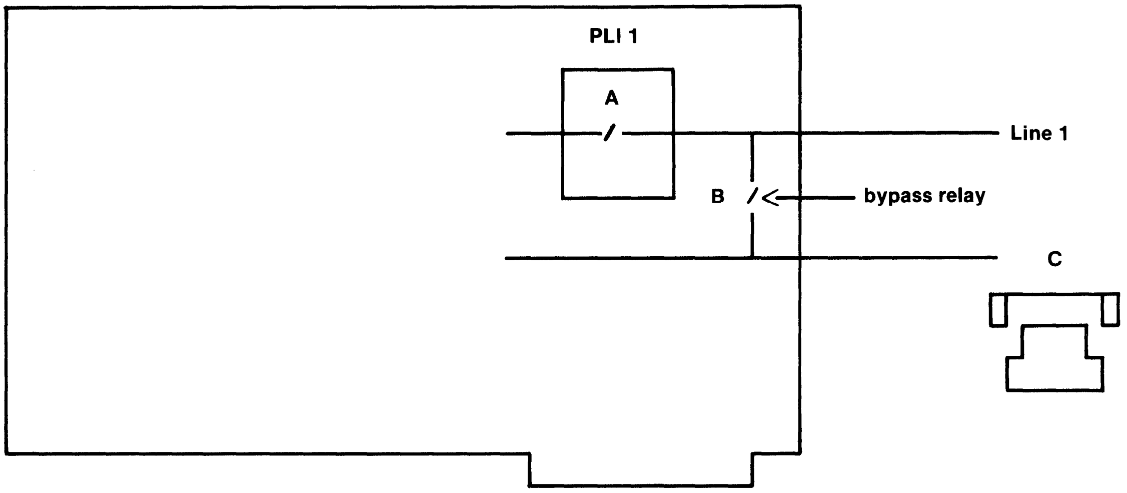
Interrupt Summary

Mask Bit	Name	Event
0	* DTMF	* DTMF tone detected
1	DTMF	Any DTMF tone detected
2	Dial tone	Remote dial tone detected

Interrupt Status



Telephony



	Object	States	Actions
A	PLI 1	ONHOOK OFFHOOK	ONHOOK command OFFHOOK command
B	Phone connection	ONLINE OFFLINE	CONNDEVS command DCONDEVS command
C	Handset	ONCRADLE OFFCRADLE	Handset interrupt Handset interrupt

PBX connection: IF A is OFFHOOK and/or (B is OFFLINE and C is OFFCRADLE)

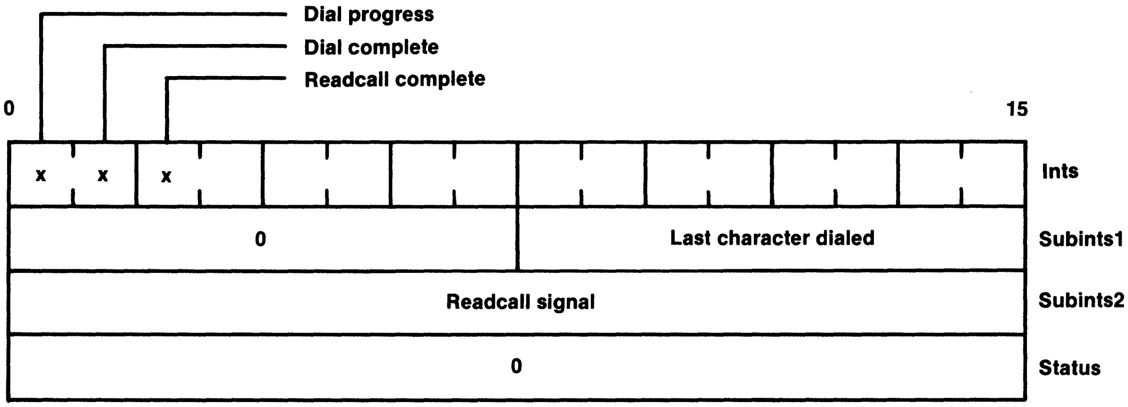
Command Summary

AL xxH	Command Name	Function
13	DCONFIG	Establish function set's default configuration
14	RCONFIG	Read function set's current configuration
15	CCONFIG	Change function set's current configuration
16	ESTINT	Establish ISR for some specified interrupts
17	MASKINT	Enable/Disable specified interrupts
18	READSTAT	Read/Poll interrupt status
19	GENTONES	Generate a pair of tones
1A	DIAL	Dial a string of digits
1B	READCALL	Analyze response from PBX
1C	TAPHOOK	Flash Hook (momentary ONHOOK)

Interrupt Summary

Mask Bit	Name	Event
0	Dial progress	After each digit
1	Dial complete	End of DIAL string
2	Readcall cmplt	Indicate signal from PBX

Interrupt Status



Readcall Signal

0 = Silence
1 = Busy
2 = Fast Busy
3 = Dial Tone
4 = Unrecognized
* 5 = Ringback
6 = Alternate long distance carrier
7 = Possible speech (Hello detect)
8 = Carrier signal

* A readcall interrupt normally terminates the analysis of PBX signals. However, in the case of Ringback, analysis continues. Further Ringbacks will be reported until the remote phone stops ringing or the interrupt is disabled. After the last Ringback, there will be one more readcall interrupt. This will normally be one of the following:

- silence
- hello detect
- alternate long distance carrier
- carrier signal

Asynchronous Communications/Modem

Configuration Parameters

The following describes the asynchronous communications and modem parameters that can be configured. Each parameter is a 16-bit integer and appears in the order shown for the DCONFIG:

Mode	0 - Originate	(default)
	1 - Answer	
Rate	1 - 110	
	3 - 300	(default)
	12 -1200	
Length	7 - Seven bits per byte	(default)
	8 - Eight bits per byte	
Parity	0 - None	
	1 - Odd Parity	
	2 - Even Parity	(default)
	3 - Mark Parity	
	4 - Space Parity	
Stop	1 - ONE stop bit	(default)
	2 - TWO stop bits	

Break The length of time, in 100ms, that a break signal should persist on the line. The valid range is 1 - 50; default is 4.

Detect The time, in seconds, that the modem will attempt to detect a carrier signal during START, before reporting a failure to establish a communications session. The valid range is 1 - 90; the default value is 30.

Loss The time, in tenths of a second, that the modem will continue to attempt carrier signal detection after loss of carrier, before reporting link status change to DOWN. If the carrier signal resumes before this time expires, no state change is indicated. Valid range is 1 - 255; default is 7. Loss must be greater than Present. The default value is 7.

Present The time, in tenths of a second, that a carrier signal must be present during START before it can be recognized as a carrier. Valid range is 1 - 255; default is 6.

The Rate, Length, Parity, and Stop parameters are collectively referred to as the asynchronous parameters.

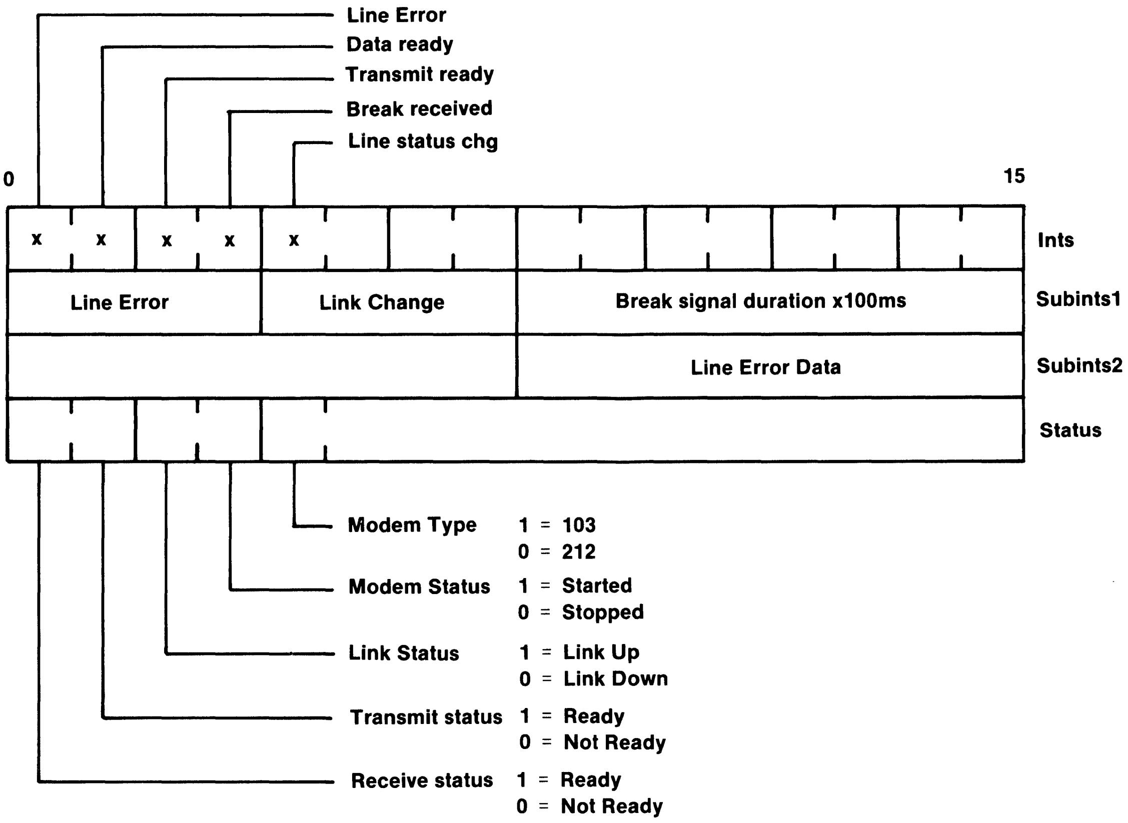
Command Summary

AL xxH	Command Name	Function
13	DCONFIG	Establish function set's default configuration
14	RCONFIG	Read function set's current configuration
15	CCONFIG	Change function set's current configuration
16	ESTINT	Establish ISR for some specified interrupts
17	MASKINT	Enable/Disable specified interrupts
18	READSTAT	Read/Poll interrupt status
1B	START	Start Carrier Exchange
1C	STOP	Stop Carrier Exchange
1D	RECEIVE	Receive one character
1E	SEND	Send one character
1F	BREAK	Send Break Signal

Interrupt Summary

Mask Bit	Name	Event
0	Line Error	Overrun or bad data received
1	Data Ready	Data available to Receive
2	Transmit Ready	Ready to Send
3	Break received	Interrupt at beginning and end of Break
4	Link status chg	The status of the link has changed

Interrupt Status



Line Error

Bit 0	1 = Overrun
	0 = OK
Bit 1	1 = Parity Error
	0 = OK
Bit 2	1 = Framing Error
	0 = OK

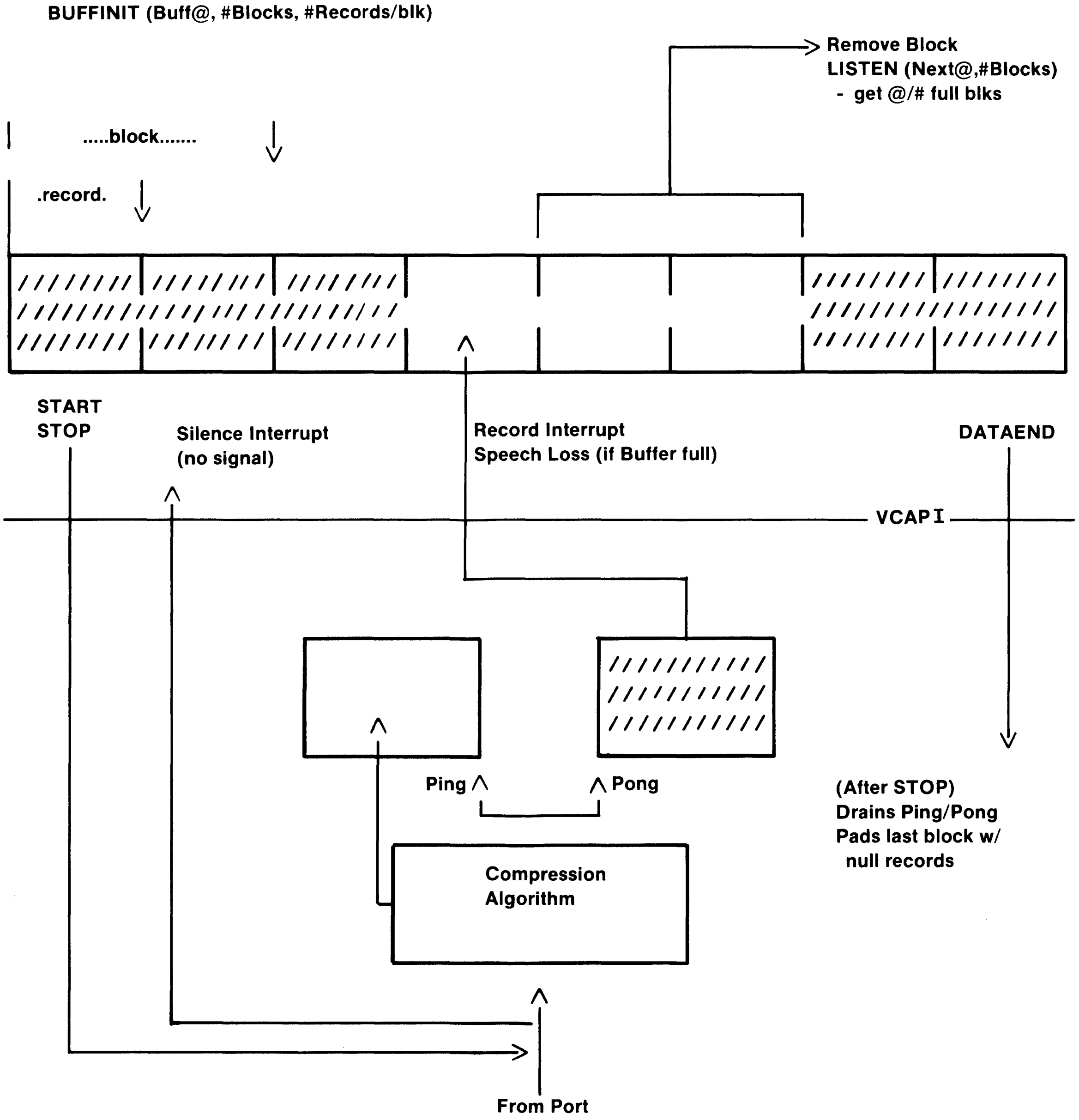
Link Change

Bits 4-6	000 = START complete
	001 = START failed
	010 = STOP complete
	100 = Line Dropped
	111 = START reconfig complete

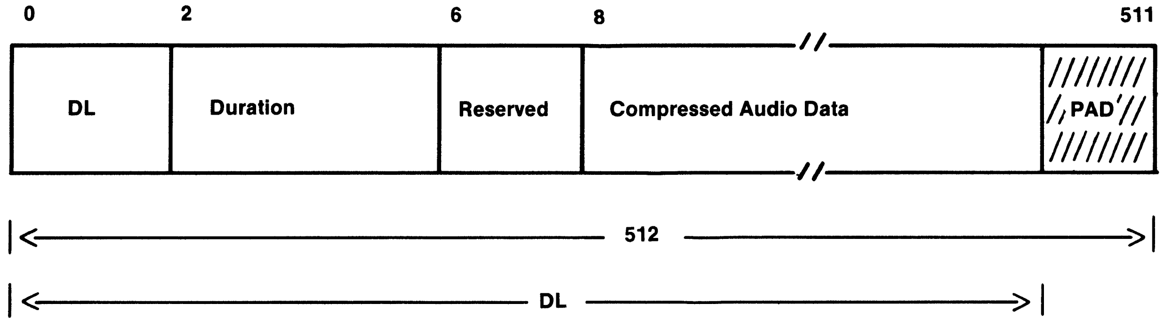
Valid Asynchronous States/Commands

Modem/Link States	Valid Commands
STOPPED/DOWN - state after CONNFTOP or - state after STOP complete	START RECEIVE (if Receive status READY)
STARTED/DOWN - START in progress or - Link dropped	STOP RECEIVE (if Receive status READY)
STARTED/UP - normal running state - but START reconfig could be in progress	START (reconfig) unless in progress STOP RECEIVE (if Receive status READY) SEND (if Transmit status READY) BREAK
STOPPED/UP - STOP in progress	RECEIVE (if Receive status READY)

Audio Record



Audio Record Structure



Where:

Record Length is fixed at 512 bytes

Bytes 0 - 1	=	Length of data in record (could be zero)
2 - 5	=	Time taken to record data in millisecs.
6 - 7	=	Reserved
8 - (DL-1)	=	Compressed Audio Data
DL - 511	=	Padding (of undefined value)

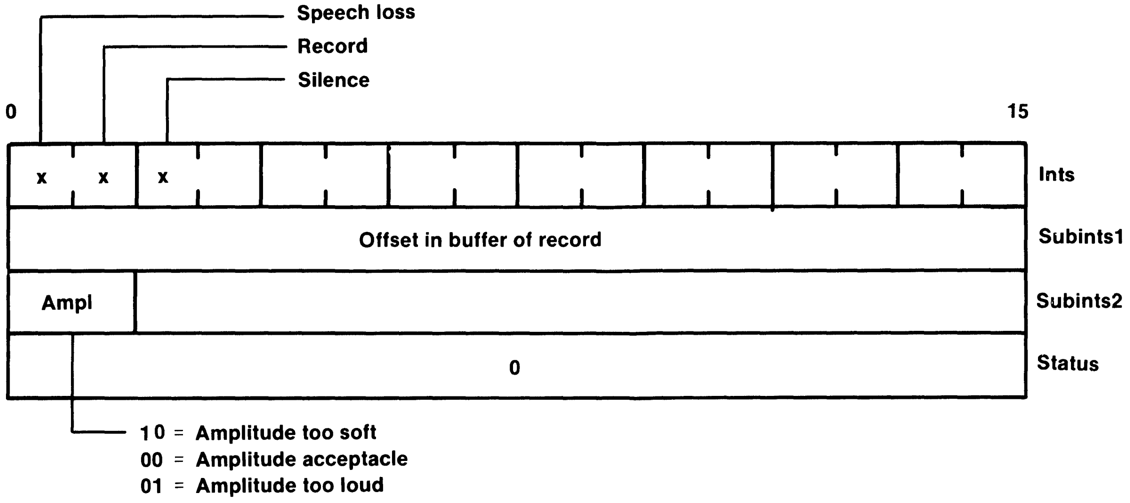
Command Summary

AL xxH	Command Name	Function
13	DCONFIG	Establish function set's default configuration
14	RCONFIG	Read function set's current configuration
15	CCONFIG	Change function set's current configuration
16	ESTINT	Establish ISR for some specified interrupts
17	MASKINT	Enable/Disable specified interrupts
18	READSTAT	Read/Poll interrupt status
19	BUFFINIT	Initialize input ring buffer
1A	DATAEND	Complete the last block of audio input data after STOP
1B	START	Start recording
1C	STOP	Stop recording
1D	LISTEN	Acknowledge block of data and get address of next

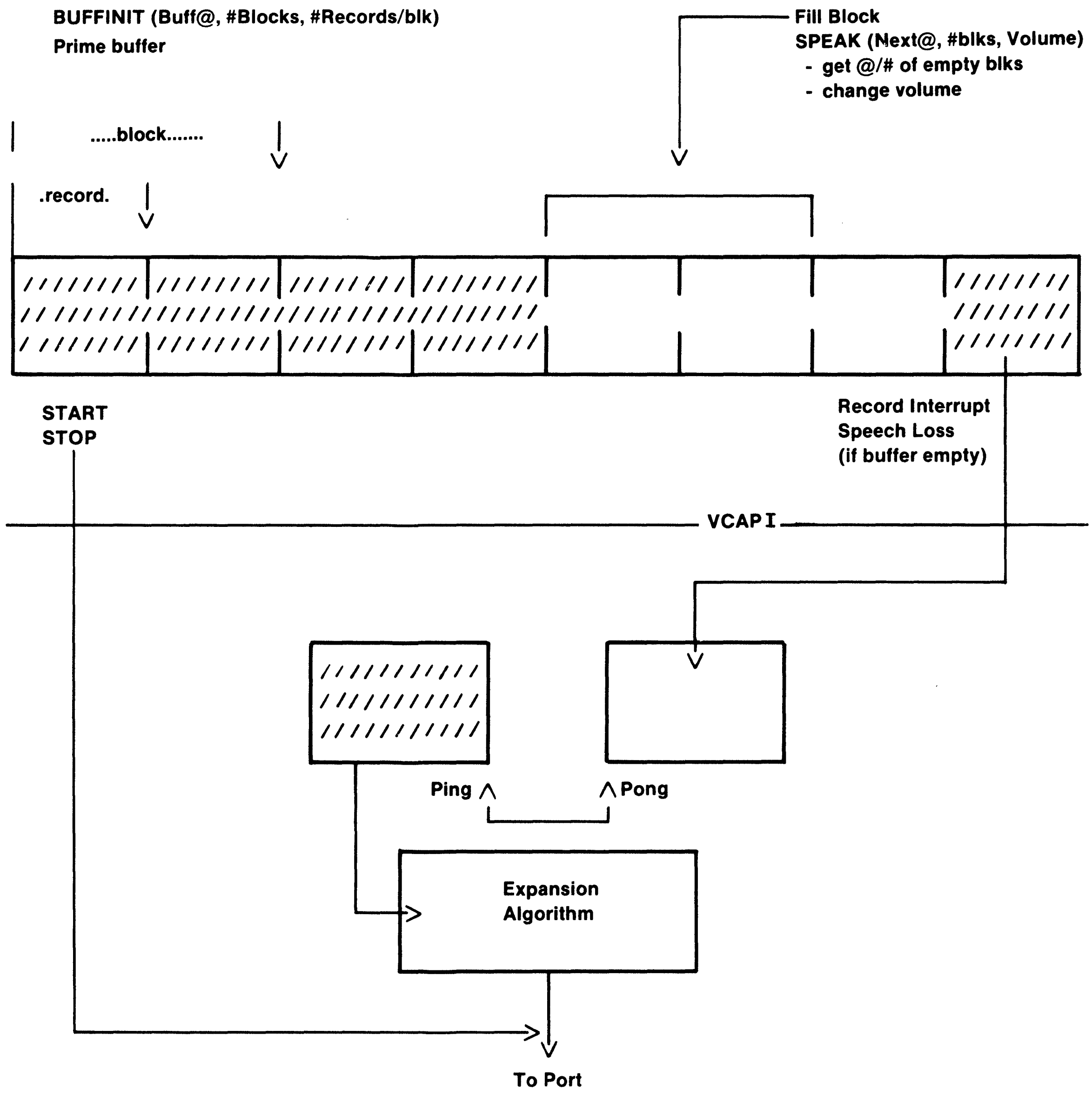
Interrupt Summary

Mask Bit	Name	Event
0	Speech loss	Ring buffer full and function set not stopped
1	Record	One more record processed
2	Silence	No signals have been received for specified time

Interrupt Status



Audio Playback



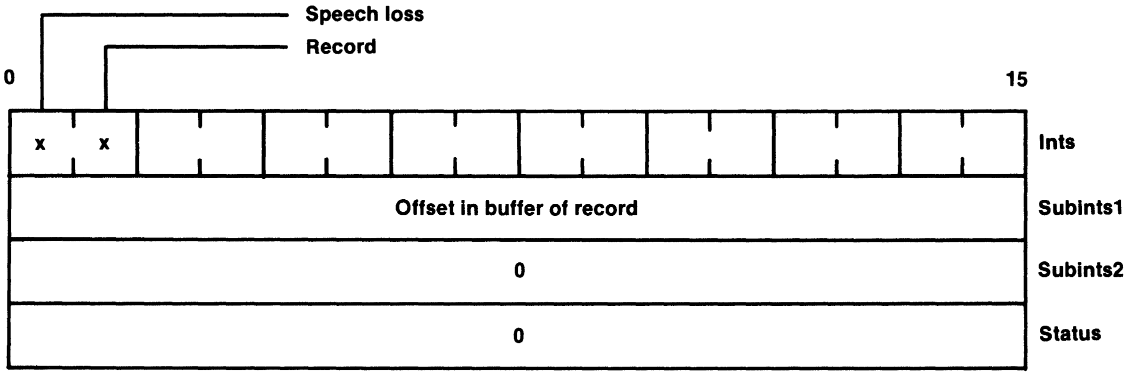
Command Summary

AL xxH	Command Name	Function
13	DCONFIG	Establish function set's default configuration
14	RCONFIG	Read function set's current configuration
15	CCONFIG	Change function set's current configuration
16	ESTINT	Establish ISR for some specified interrupts
17	MASKINT	Enable/Disable specified interrupts
18	READSTAT	Read/Poll interrupt status
19	BUFFINIT	Initialize output ring buffer
1B	START	Start playback
1C	STOP	Stop playback
1E	SPEAK	Indicate new block full and get address of next

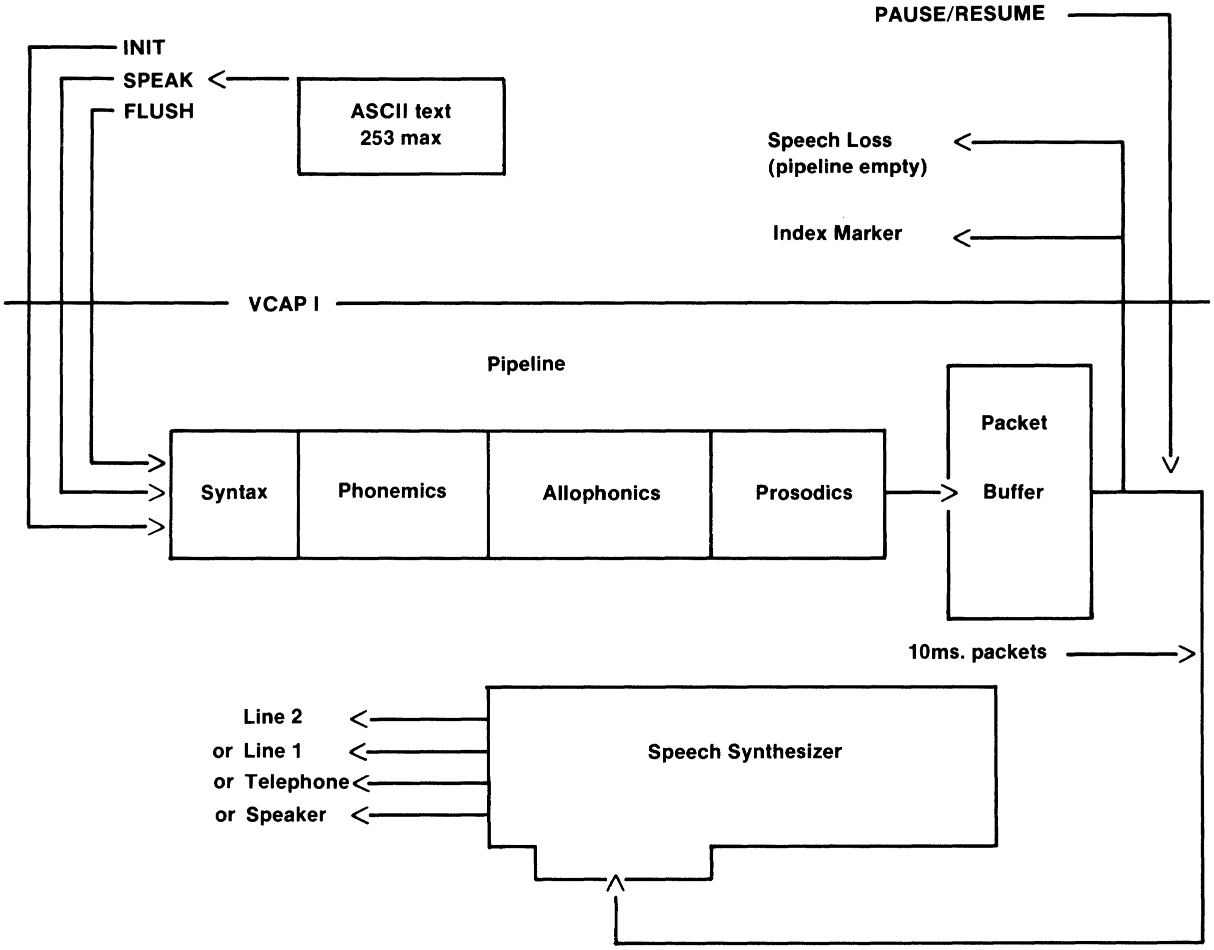
Interrupt Summary

Mask Bit	Name	Event
0	Speech loss	Ring buffer empty and function set not stopped
1	Record	One more record processed

Interrupt Status



Speech Synthesis



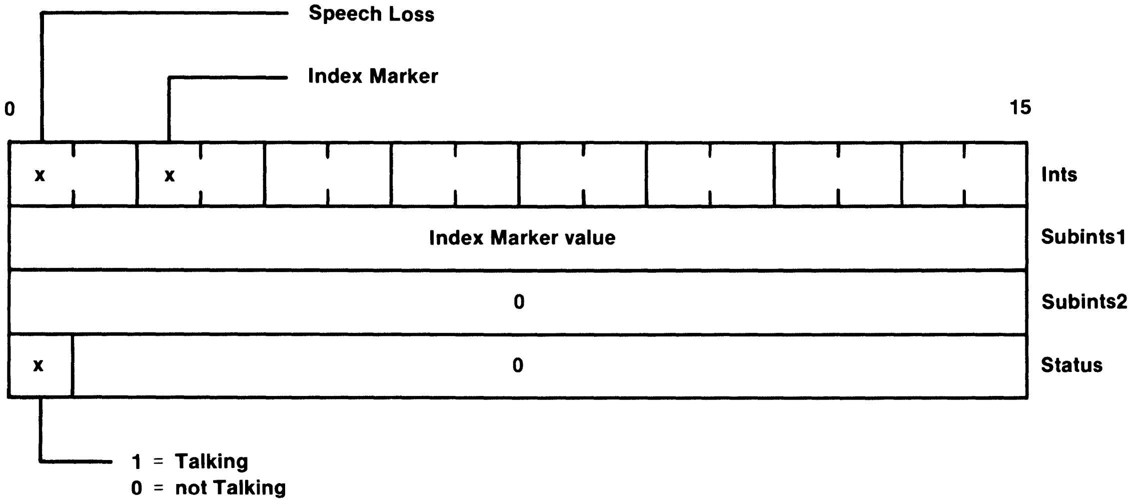
Command Summary

AL xxH	Command Name	Function
13	INIT	Initialize the data structures of the function set
16	ESTINT	Establish ISR for some specified interrupts
17	MASKINT	Enable/Disable specified interrupts
18	READSTAT	Read/Poll interrupt status
19	FLUSH	Flush any buffered text
1B	RESUME	Resume production of speech
1C	PAUSE	Suspend the production of speech
1E	SPEAK	Supply more text to speak (and start speaking)

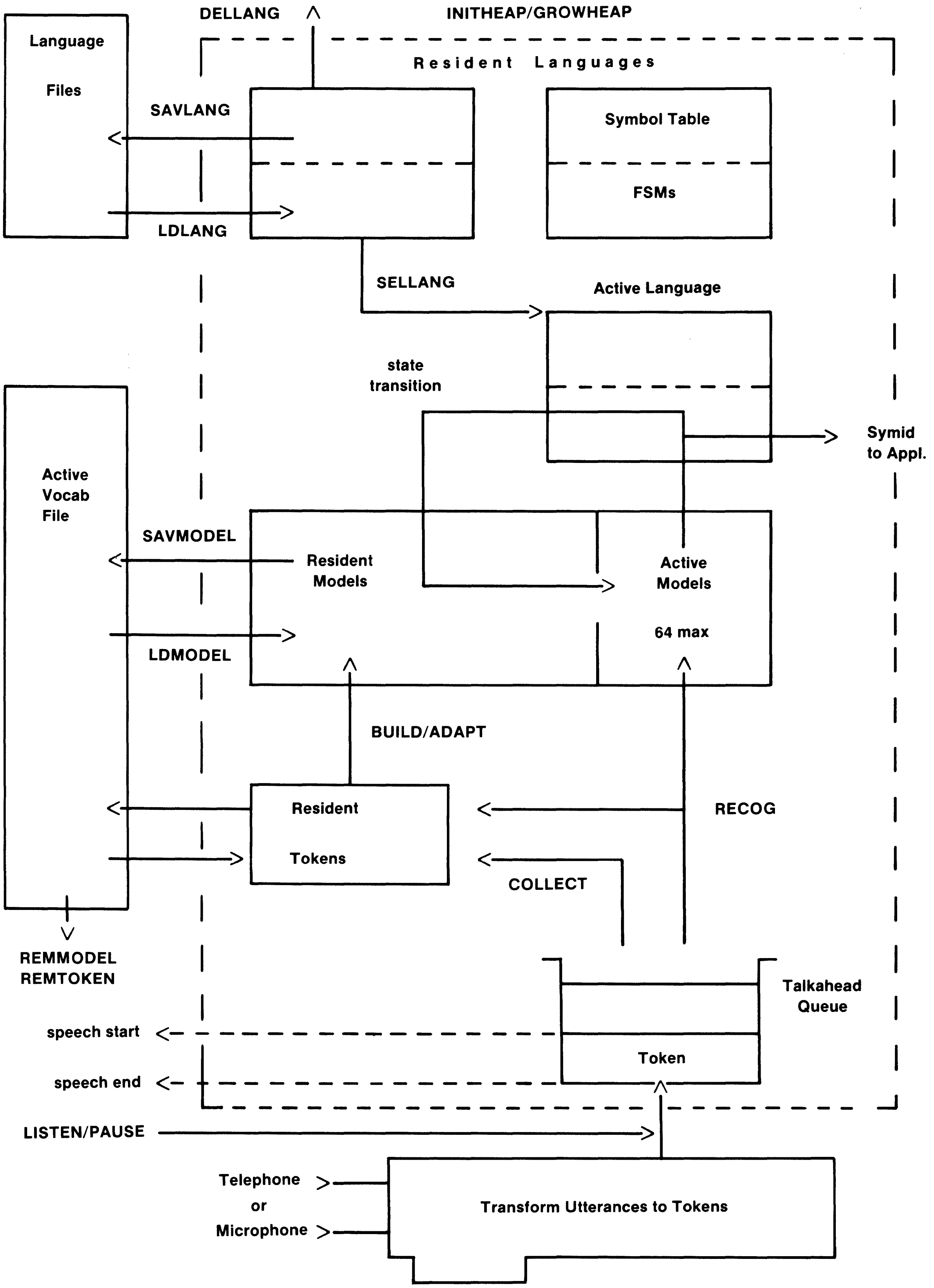
Interrupt Summary

Mask Bit	Name	Event
0	Speech loss	All text has been spoken, and function set not stopped
2	Index Marker	Index Marker control reached

Interrupt Status



Speech Recognition



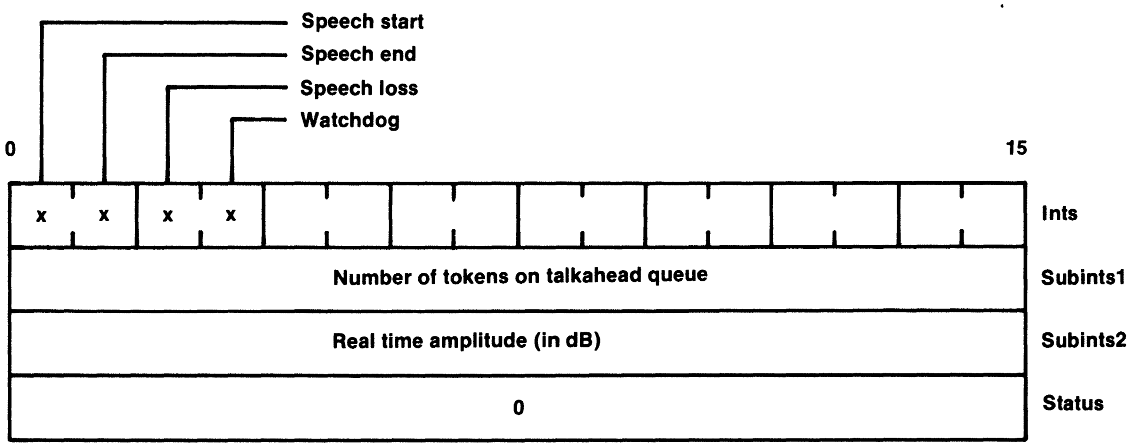
Command Summary

AL	Command	Function
xxH	Name	
13	DCONFIG	Establish function set's default configuration
14	RCONFIG	Read function set's current configuration
15	CCONFIG	Change function set's current configuration
16	ESTINT	Establish ISR for some specified interrupts
17	MASKINT	Enable/Disable specified interrupts
18	READSTAT	Read/Poll interrupt status
19	INITHEAP	Initialize function set's working space
1A	GROWHEAP	Inform function set of larger heap
1B	CALIB	Set acoustical parameters in heap for new vocabulary
1C	LEVELS	Measure input signal levels
1D	LDLANG	Load a command language into heap and make active
1E	SELLANG	Select new active language from resident languages
1F	DELLANG	Delete language from the heap
20	SAVLANG	Save updated language to disk
21	INITVOC	Initialize new vocabulary file
22	SELVOC	Select a file as the active vocabulary
23	CLOSEVOC	Save all changes to a vocabulary and close file
25	PUTLABEL	Write a label to a vocabulary file
26	LDMODEL	Load a model or set of models from a vocabulary
27	SAVMODEL	Save a model or set of models in a vocabulary file
28	DELMODEL	Delete a model or set of models from the heap
29	REMMODEL	Delete a model or set of models from a vocabulary file
2A	DIRMODEL	Prepare to read the directory of models from a vocabulary
2B	NXTMODEL	Read the text name of the next model in the directory
2C	LDTOKEN	Read a token into the heap from the vocabulary file
2D	SAVTOKEN	Write a token from the heap to the vocabulary file
2E	DELTOKEN	Delete a token from the heap
2F	REMTOKEN	Delete a token from the vocabulary file
30	DIRTOKEN	Prepare to read the directory of tokens from a vocabulary
31	NXTTOKEN	Read the ID of the next token in the directory
32	LISTEN	Enable listening for utterances
33	PAUSE	Disable listening for utterances
34	QUIT	Terminate the current RECOG or COLLECT function
35	COLLECT	Collect the next utterance (token) for training
36	BUILD	Contruct a new model from a number of tokens
37	ADAPT	Modify a model with a number of tokens
38	RECOG	Recognize the next utterance
39	DIRSYMS	Prepare to read symbols from the active language
3A	NEXTSYM	Read ID of next symbol in directory
3B	SYMNAME	Get name of symbol given the ID
3C	SYMCHNAM	Change the name of a symbol
3D	SYMID	Get ID of symbol given the name
3E	SYMTYPE	Get the type of a symbol
3F	OUTPUT	Get the character data associated with a symbol
40	CHOUTPUT	Change the character data associated with a symbol
41	TRMACT	Activate a symbol
42	TRMDEACT	Deactivate a symbol
43	TRMISACT	Test if a symbol is active
44	TRMISMOD	Test if model resident for symbol

Interrupt Summary

Mask Bit	Name	Event
0	Speech start	Start of utterance
1	Speech end	End of utterance
2	Speech Loss	Talkahead queue full - utterance lost
3	Watchdog	250ms interrupts while waiting for next utterance

Interrupt Status



Sample Programs

Overview of an Application Program

An application must take the following actions to use the VCAPI:

1. Use the OPEN command to obtain a Resource Control Block (RCB) and Connection IDs (CIDs).

2. Claim hardware resources for the RCB using the CLAIMHDW command.

   To use a particular function set, the application must claim the appropriate partition for the function set, a port and the device(s) it needs.

3. Connect the devices to be used by the function set to the port using the CONNDTOP command.

4. Load the function set into a partition and connect it to the port using the CONNFTOP command.

5. Establish interrupt routines and enable interrupts for the function set using the ESTINT and MASKINT commands. (This step is optional, depending on how the application chooses to handle interrupts).

6. When an application is finished, it must either use the CLOSE command to free all resources, or selectively free resources using the FREEHDW command.

The rest of this chapter contains two programs. The first is written in assembly language, and the second in BASIC. Each program accomplishes the same task: to convert text to speech. The programs prompt the user to input text from the keyboard. When finished, the user enters "q" to quit.

Assembly Language Example

```
          TITLE text to speech
          PAGE     60,132

          include  macro.lib

COMMENT   *
             The user is prompted to enter any text. However, the text
             must end with either a period, question mark, or an
             exclamation point.
          *
SSEG      SEGMENT  PARA STACK
          DB       100 DUP('STACK')
SSEG      ENDS
;
DSEG      SEGMENT  PARA PUBLIC

crlf      db       0dh, 0ah, '$'               ;parameter list for OPEN
text      db       255,?,255 dup(0)            ;input text
prompt    db       'text:  $'                  ;prompt for input
errormsg  db       'syntax error$'
instr1    db       'Enter text followed by terminator (. ? !).$'
instr2    db       'Enter "q" to quit$'


plist     label    word                ;parameter list for OPEN
ret       dw       ?
rcb       dw       ?
bid       dw       ?
cid1      dw       ?
cid2      dw       ?
cid3      dw       ?
cid4      dw       ?
p1        dw       8 dup (?)           ;parameter list for commands

hdwset    dw       ?                   ;port 1, speaker/mic, partition 2
intset    dw       ?                   ;interrupt conditions
port      dw       ?                   ;port #
fctset    dw       ?                   ;function set
device    dw       ?                   ;device(s)

addrsize  dw       ?                   ;address size
                                       :for text to speech function set


DSEG      ENDS
;
CSEG      SEGMENT  PARA PUBLIC
          ASSUME   CS:CSEG,SS:SSEG,DS:DSEG,ES:DSEG
;
MAIN      PROC     FAR                 ;set up return to DOS
          PUSH     DS
          XOR      AX,AX
          PUSH     AX
          MOV      AX,DSEG
          MOV      DS,AX
          MOV      ES,AX
```

```
;
;PROGRAM STARTS HERE

;CLS
          MOV      AX,0002H
          INT      10H
          WRITELIN INSTR1
          WRITELIN INSTR2
          WRITELIN

;initialize parameters (values here are specific to application)

          MOV      AX,2601H             ;ax = hardware resources
                                        ;2601 = port1, spkr/mic, partition 2
          MOV      HDWSET,AX            ;hdwset = AX

          MOV      AX,0                 ;AX = no interrupt conditions
          MOV      INTSET,AX            ;Intset = AX

          MOV      AX,1                 ;AX = port # 1
                                        ;1 = port #1
          MOV      PORT,AX              ;port = AX

          MOV      AX,0600H             ;AX = devices
                                        ;0600 = speaker/mic
          MOV      DEVICE,AX            ;device = AX

          MOV      AX,10                ;AX = function set
                                        ;10 = text-to-speech
          MOV      FCTSET,AX            ;fctset + AX

          MOV      AX,1                 ;AX = address size
                                        ;for text-to-speech function set
          MOV      ADDRSIZE,AX          ;addrsize = AX


;
;OPEN
          MOV      AH,11H               ;VCAPI id
          MOV      AL,11H               ;function code for OPEN
          MOV      DX,21FH              ;card I/O address
          LEA      BX,PLIST             ;address of parameter list
          INT      14H                  ;invoke OPEN
          MOV      RET,AX
          CMP      AX,0                 ;test return code
          JE       CONT1                ;continue if no error
          JMP      ERROR
```

```
;
;CLAIMHDW
CONT1     MOV      AH,11H             ;VCAPI id
          MOV      AL,1AH             ;function code for CLAIMHDW
          LEA      BX,P1              ;address of parameter list
          MOV      DI,BX
          MOV      DX,RCB             ;DX = RCB
          MOV      (DI+2),DX          ;move RCB into parameter list
          MOV      DX,HDWSET          ;DX = hardware resources
          MOV      (DI+4),DX          ;move hdwset into parameter list
          MOV      DX,INTSET          ;DX = interrupt conditions
          MOV      (DI+6),DX          ;move intset into parameter list
          MOV      DX,BID             ;dx = BASE id (for base cmds)
          INT      14H                ;invoke CLAIMHDW
          MOV      RET,AX
          CMP      RET,0              ;test return code
          JE       CONT2              ;continue if no error
          JMP      ERROR


;
;CONNDTOP
CONT2     MOV      AH,11H             ;VCAPI ID
          MOV      AL,21H             ;function code for CONNDTOP
          LEA      BX,P1              ;address of parameter list
          MOV      DI,BX
          MOV      DX,RCB             ;DX = RCB
          MOV      (DI+2),DX          ;move RCB to parameter list
          MOV      DX,PORT            ;DX = Port
          MOV      (DI+4),DX          ;move port to parameter list
          MOV      DX,DEVICE          ;DX = speaker/mic
          MOV      (DI+6),DX          ;move device to parameter list
          MOV      DX,BID             ;DX = BASE ID (for base cmds)
          INT      14H                ;invoke CONNDTOP
          MOV      RET,AX
          CMP      AX,0               ;test return code
          JE       CONT3              ;continue if successful
          JMP      ERROR

;
;CONNFTOP
CONT3     MOV      AH,11H            ;VCAPI ID
          MOV      AL,1FH             ;function code for CONNFTOP
          LEA      BX,P1              ;address of parameter list
          MOV      DI,BX
          MOV      DX,CID2            ;DX = connection ID corresponding
                                      ;to the partition required by the
                                      ;function set being connected
          MOV      (DI+2),DX          ;move CID to parameter list
          MOV      DX,PORT            ;DX = Port
          MOV      (DI+4),DX          ;move port to parameter list
          MOV      DX,FCTSET          ;DX = text-to-speech function set
          MOV      (DI+6),DX          ;move device to parameter list
          MOV      DX,BID             ;DX = BASE ID (for base cmds)
          INT      14H                ;invoke CONNFTOP
          MOV      RET,AX
          CMP      AX,0               ;test return code
          JE       CONT4              ;continue if successful
          JMP      ERROR
```

```
;
;Text-to-Speech Application

;Initialize Buffers
CONT4     MOV      AH,11H             ;VCAPI ID
          MOV      AL,13H              ;function code for INIT
          LEA      BX,P1               ;address of parameter list
          MOV      DI,BX
          MOV      DX,CID2             ;DX = connection ID corresponding
                                       ;to the partition containing
                                       ;the text-to-speech function set
          MOV      (DI+2),DX           ;move CID to parameter list
          INT      14H                 ;invoke INIT
          MOV      RET,AX
          CMP      AX,0                ;test return code
          JE       ITEXT               ;continue if successful
          JMP      ERROR

;
;get text from keyboard
ITEXT     WRITE    PROMPT              ;Display "text:"
          READ     TEXT                ;input text
          MOV      CL,TEXT+1           ;length of input text
          MOV      CH,0                ;Allow only 255 characters
          LEA      BX,TEXT+2           ;put 0 at end of string
          ADD      BX,CX
          MOV      BYTE PTR (BX),0
          CMP      CL,1                ;length of 1 (possible 'q')?
          JNE      CONT6               ;no; continue
          MOV      AL,TEXT+2           ;AL = first letter of text
          CMP      AL,'q'              ;QUIT?
          JE       CLOSE               ;YES, close VCAPI and end.

;
;SPEAK
CONT6     MOV      AH,11H             ;VCAPI ID
          MOV      AL,1EH              ;function code for SPEAK
          LEA      BX,P1               ;address of parameter list
          MOV      DI,BX
          MOV      DX,CID2             ;DX = connection ID corresponding
                                       ;to the partition containing
                                       ;the text-to-speech function set
          MOV      (DI+2),DX           ;move CID to parameter list
          MOV      DX,ADDRSIZE         ;DX = address size
          MOV      (DI+4),DX           ;move addrsize into parameter list
          LEA      DX,TEXT+2           ;address of buffer
          MOV      (DI+6),DX           ;move address into parameter list
          INT      14H                 ;invoke SPEAK
          MOV      RET,AX
          CMP      AX,0                ;test return code
          JE       ITEXT               ;continue if successful
          WRITELN  ERRORMSG            ;display error message
          JMP      ITEXT               ;continue
```

```
;
;CLOSE
CLOSE     MOV      AH,11H              ;VCAPI ID
          MOV      AL,12H              ;function code for CLOSE
          LEA      BX,PLIST            ;address of parameter list
          MOV      DX,BID              ;DX = base id (for base commands)
          INT      14H                 ;invoke CONNFTOP
          MOV      RET,AX
          CMP      AX,0                ;test return code
          JNE      ERROR               ;process error
          JMP      EXIT                ;exit mainline program

;
;ERROR
ERROR:    NOP                          ;Error handling routine goes here


;PROGRAM ENDS HERE
EXIT:     RET                          ;Return to DOS
MAIN      ENDP
CSEG      ENDS
          END      MAIN
```

```
;MACRO LIBRARY.....
;=======================================================================

READ      MACRO    BUFFER

COMMENT   *
          buffer..........buffer for input (size = max size)
          in dseg:
             byte 1.......max size of buffer
             byte 2.......number of characters read
             remaining....buffer
          *

          PUSH     DX
          PUSH     AX
          LEA      DX,BUFFER
          MOV      AH,0AH
          INT      21H
          POP      AX
          POP      DX
          WRITELN
          ENDM
;=======================================================================
WRITE     MACRO    STRING

COMMENT   *
          String...........string for output
          in dseg, string must be followed by '$'
          *

          PUSH     DX
          PUSH     AX
          LEA      DX,STRING
          MOV      AH,09H
          INT      21H
          POP      AX
          POP      DX
          ENDM
;=======================================================================
WRITELN   MACRO    STRING

COMMENT   *
          String...........address of string for output
          in dseg, string must be followed by '$'
          *

IFNB  <string>
          WRITE     STRING
ENDIF
          WRITE     CRLF          ;prints carriage return, linefeed
                                  ;crlf   db    0dh,0ah,'$'  (in dseg)

          ENDM
```

BASIC Example

BASIC source code for text-to-speech example:

```
  1  CLS
  2  PRINT "Enter text followed by a terminator (. ? !)"
  3  PRINT "Enter "q" to quit"
  4  PRINT
  5  DEF SEG = &H1000
 10  DIM
        ID%                      'parameter for base ID or connection ID
 15  DIM
        FCODE%                   'parameter for function code
 20  DIM
        PLIST%(10)               'parameter for address of parameter list
 25  DIM
        RCB%                     'resource control block ID
 30  DIM
        BID%                     'Base ID
 45  DIM
        CID1%,
        CID2%                    'connection ID
 50  PORT%=1                     'port 1
 55  DEVICE%=&H600               'device = speaker/microphone
 60  HDWSET%=&H2601              'hardware set = port 1, partition 2
 61                              'speaker/microphone
 65  INTSET%=0                   'No interrupts to claim
 70  FCTSET%=10                  'function set = text-to-speech
 75  ADDRSIZE%=1                 'address size = 16 bits
 80  DIM
        BUFFER%,
        BUFF                     'address of text buffer
 85  API = 0

 88  REM poke API link code into memory

 90  READ J$  :
     FOR I = API TO API + &H1B :
     :  J = VAL("&h" + MID$(J$,2*I,2)) :
     :  POKE I,J :
     NEXT I
 95  DATA
        558bec061e078b7e0a8b158b5e068b7e088b05b411cd14075dca600

100  REM OPEN

110  ID%=&H21F                         'card I/O address
120  FCODE%=&H11                       'function code for OPEN
130  CALL API(ID%,FCODE%,PLIST%(0))    'CALL OPEN
140  RCB%=PLIST%(1)                    'resource control block ID
150  BID%=PLIST%(2)                    'Base ID
160  CID1%=PLIST%(3)                   'Connection ID # 1
170  CID2%=PLIST%(4)                   'Connection ID # 2
```

```
200  REM CLAIMHDW

210  ID%=BID%                          'For Base Function s
220  FCODE%=&H1A                       'Function code for CLAIMHDW
230  PLIST%(1)=RCB%                    'RCB ID
240  PLIST%(2)=HDWSET%                 'Hardware Set
250  PLIST%(3)=INTSET%                 'Interrupt Set
260  CALL API(ID%,FCODE%,PLIST%(0))    'CALL CLAIMHDW

300  REM CONNDTOP

310  ID%=BID%                          'For Base Function s
320  FCODE%=&H21                       'Function code for CONNDTOP
330  PLIST%(1)=RCB%                    'RCB ID
340  PLIST%(2)=PORT%                   'Port #
350  PLIST%(3)=DEVICE%                 'Speaker/Mic
360  CALL API(ID%,FCODE%,PLIST%(0))    'CALL CONNDTOP

400  REM CONNFTOP

410  ID%=BID%                          'For Base Functions
420  FCODE%=&H1F                       'Function code for CONNFTOP
430  PLIST%(1)=CID2%                   'Connection ID for function set
440  PLIST%(2)=PORT%                   'Port #
450  PLIST%(3)=FCTSET%                 'Function Set ID (text-to-speech)
460  CALL API(ID%,FCODE%,PLIST%(0))    'CALL CONNFTOP

500  REM INITIALIZE TEXT-TO-SPEECH BUFFERS

510  ID%=CID2%                         'For Text-to-Speech functions
520  FCODE%=&H13                       'Function code for INIT
530  PLIST%(1)=CID2%                   'Connection ID for function set
540  CALL API(ID%,FCODE%,PLIST%(0))    'CALL INIT

600  REM GET INPUT TEXT

610  DEF SEG                           'BASIC data segment
620  LINE INPUT "text:  ";TEXT$
630  IF
        TEXT$="q"
           THEN
             ( GOTO ) 800             'quit?
640  TEXT$=TEXT$+CHR$(0)              'put 0 at end of text

650  REM FIND ADDRESS OF TEXT BUFFER

660  BUFF=PEEK(VARPTR(TEXT$)+1)+256*PEEK(VARPTR(TEXT$)+2)
670  IF
       BUFF < &H8000
          THEN
             BUFFER% = BUFF
          ELSE
             BUFFER% = BUFF - 65536!
680  DEF SEG=&H1000                   'segment with API code
```

```
700  REM SPEAK input text

710  ID%=CID2%                          'For Text-to-Speech functions
720  FCODE%=&H1E                        'Function code for SPEAK
730  PLIST%(1)=CID2%                    'Connection ID for function set
740  PLIST%(2)=ADDRSIZE%                'address size
750  PLIST%(3)=BUFFER%                  'Address of Buffer
760  CALL API(ID%,FCODE%,PLIST%(0))     'CALL SPEAK
770  IF
       PLIST%(0)<>0
          THEN
             PRINT"syntax error"
             GOTO 600
780  GOTO 600                           'continue until user quits

800  REM CLOSE
810  DEF SEG = &H1000                   'segment with API code
820  ID%=BID%                           'For Base commands
830  FCODE%=&H12                        'Function code for CLOSE
840  PLIST%(1)=RCB%                     'Resource Control Block ID
850  CALL API(ID%,FCODE%,PLIST%(0))     'CALL CLOSE
        TITLE      linkage to API
        PAGE       60,132

COMMENT *



        *

;
CSEG    SEGMENT    PARA PUBLIC 'code'
        ASSUME     CS:SEG
;
        PUBLIC     API
API     PROC       FAR

        PUSH       BP                   ;save BP
        MOV        BP,SP                ;BP points to stack
        PUSH       ES                   ;save ES
        PUSH       DS                   ;save DS
        POP        ES                   ;ES = data seg of calling program
        MOV        DI,(BP+10)           ;DX = BID or CID
        MOV        DX,(DI)
        MOV        BX,(BP+6)            ;BX = address of parameter list
        MOV        DI,(BP+8)            ;AL = function code
        MOV        AX,(DI)
        MOV        AH,11H               ;AH = VCAPI ID
        INT        14H                  ;invoke VCAPI command
        POP        ES                   ;restore ES
        POP        BP                   ;restore BP
        RET        6                    ;FAR return

API     ENDP
CSEG    ENDS
        END        API
```

IBM Personal Computer Seminar Proceedings

Publication Number	Volume	Topic
	V1.1	Contains identical information as V1.2
(G320-9307)	V1.2	DOS 2.0 and 1.1 Comparison Compatibility Guidelines for Application Development 8087 Math Co-Processor IBM Macro Assembler
(G320-9308)	V1.3	DOS 2.1, 2.0 and 1.1 Comparison Disk Operation System 2.1 IBM PC*jr* Architecture IBM PC*jr* Compatibility Overview Cartridge BASIC IBM Personal Communications Manager-Modem Drivers
(G320-9309)	V2.1	Contains identical information as V2.2
(G320-9310)	V2.2	IBM Software Support Center International Compatibility Requirements IBM Personal Computer Cluster Program
(G320-9311)	V2.3	IBM Personal Computer Cluster Program Sort, Version 1.00 FORTRAN and Pascal Compiler, Version 2.00 PC*jr* Cartridge Tips and Techniques
(G320-9312)	V2.4	IBM Personal Computer AT Architecture ROM BIOS Compatibility DOS 3.0 Software Compatibility
(G320-9313)	V2.5	IBM PC Network Overview IBM PC Network Hardware IBM PC Network BIOS (NETBIOS) Architecture IBM PC Network Program
(G320-9314)	V2.6	TopView
(G320-9315)	V2.7	IBM Personal Computer Resident Debug Tool
(G320-9319)	V2.8	IBM PC Network SMB Protocol
(G320-9316)	V2.9	IBM PC XENIX
(G320-9317)	V2.10	IBM PC Professional Graphics Software IBM PC Graphical Kernel System IBM PC Graphical File System IBM Plotting System Library IBM Professional FORTRAN IBM PC Data Acquisition & Control Adapter & Software IBM General Purpose Interface Bus Adapter & Software
(G320-9318)	V2.11	IBM Enhanced Graphics Adapter
(G320-9320)	V3.1	IBM PC Information Panel (3295 Plasma Display)
(G320-9321)	V3.2	IBM BASIC Compiler 2.00
(G320-9322)	V3.3	IBM Personal Computer C Compiler
(G320-9323)	V3.4	IBM Asynchronous Communications Server Protocol
(G320-9324)	V3.5	IBM Personal Computer Voice Communications Option

Notes

G320-9324

IBM Corporation
Editor, IBM Personal Computer Seminar Proceedings
Internal Zip 4629
Post Office Box 1328
Boca Raton FL 33432