Setup

Figures

Notes:

The integrated I/O functions of the system board use POS information during the setup procedure. The bit assignments and functions may vary from system to system (refer to the system board setup information in the system-specific technical reference for the system you are dealing with). The setup routine (Automatic Configuration) obtains adapter information from adapter description files and programs.

Reference Diskette

Each system has a Reference Diskette containing the System Configuration and diagnostic utilities. The diagnostic utilities perform customer and service testing on the system, its components, and adapters made by IBM. The System Configuration utilities identify the installed hardware and interpret the system resources (I/O ports, memory, interrupt levels, and arbitration levels) for each device.

All the files on the Reference Diskette must be copied onto a backup copy, and the backup copy is used for configuring the system and running diagnostic routines. As options are added, the files needed for configuration and problem determination of the new options are merged onto this backup copy.

The following table shows the naming convention for files used by adapters during configuration. The CardID is the POS adapter ID in hex.

File Name	Description	
@CardID.ADF	Adapter Description File	Required
CCardID.ADF	Adapter Description Program	Optional
ICardID.ADF	Initialization Program	Optional

Figure 1. Files with the ADF Extension

Option Diskette

The Option Diskette supplied with the adapter should contain the adapter description file (ADF) and, if needed by the adapter, the adapter description program (ADP) or the initialization program.

Before a new adapter is installed, its ADF and other associated files are merged onto a backup copy of the Reference Diskette. These files are merged by selecting the Copy an Option Diskette program from the Main Menu and following the instructions on the screen.

Additional files may be required on the Option Diskette to support keywords or functions used in the ADF or ADP. For more information on the content of an Option Diskette, contact Developer Assistance at 1-800-IBM-7763 (this number is for developers who are registered with IBM).

Note: Developers of adapters and their customers are granted permission to use, but not to modify, the Set Configuration program to support their adapters.

System Configuration Utilities

The System Configuration utilities are contained within the Set Configuration program. This program enables the user to configure the system in one of two ways:

- Automatically after a configuration error is displayed
- Manually by selecting Set Configuration from the Main Menu.

The Set Configuration program uses information contained in the adapter description file (ADF) to track and allocate system resources. Each ADF describes the resources that can be allocated to a specific adapter and the POS setting used to indicate those resource assignments. When more than one device is configured to the same resource and that resource cannot be shared, only one of the conflicting devices is enabled.

Each adapter contains a 16-bit adapter ID and four 8-bit POS registers. Adapter IDs and the POS information are stored in CMOS or nonvolatile RAM by the Set Configuration program. The Set Configuration program determines the system type and handles differences between systems such as the nonvolatile RAM storage and the number of adapter slots available.

Automatic Configuration Utility

Automatic Configuration can be run after a configuration error has occurred or by selecting Run Automatic Configuration from the Set Configuration menu. Each time the system is powered on, POST compares the configuration of the system to the configuration indicated by nonvolatile RAM. If differences between the two are detected, an error is displayed and logged in system RAM. If the Reference Diskette is installed, an expanded message is displayed that further explains the POST error.

If a configuration error is caused by a loss o f configuration data, Automatic Configuration is run immediately after the POST error message is displayed. If the error is caused by a change to the configuration, the user may be given a choice to either run Automatic Configuration, restore configuration, or continue to the Main Menu. If the user continues with the Main Menu, the changed areas of the system are configured and nonvolatile RAM is updated when Set Configuration is selected from the Main Menu.

Depending on the cause of the error, Automatic Configuration either reconfigures the entire system or configures only the areas of the system that have been changed since the last time a configuration was performed. The following POST errors cause the system to be completely reconfigured:

- 161 (battery failure)
- 162 (CRC error in nonvolatile RAM).
- 169 (processor complex error).
- 173 (CRC error in nonvolatile RAM).

The following POST errors cause only the areas of the system that have changed to be reconfigured:

- 162 (system configuration error not caused by a CRC error)
- 164 (memory configuration)
- 165 (adapter configuration).
- 174 (adapter configuration).

An adapter is considered previously configured when the POS data stored in nonvolatile RAM matches a set of POS settings in the appropriate ADF. If an ADF for an installed adapter cannot be found, the user is prompted to insert the Option Diskette. If system configuration is continued without its ADF being found, the adapter is not configured and is not enabled.

During Automatic Configuration, devices are configured to the first nonconflicting values as defined in the ADF. Adapters are configured in the order of the channel position in which they are installed. The system board is configured first, followed by each adapter slot starting with slot 1. Interrupt levels are a shared resource and are not considered to be in conflict if more than one device is assigned the same level. If a named item in the ADF contains a list of interrupt levels to choose from, the Automatic Configuration program assigns the level that is assigned to the fewest devices at that time.

Automatic Configuration will backtrack to previously configured adapters to resolve conflicts in nonsystem memory resources. If Automatic Configuration cannot resolve a conflict, the conflict must be resolved by choosing a nonconflicting resource option through the Change Configuration utility. Any adapter having a resource conflict that cannot be resolved by the Set Configuration program is disabled; the program sets bit 0 of POS Register 2 (hex 0102) to 0 in nonvolatile RAM.

Change Configuration Utility

The Change Configuration utility allows the user to change the current configuration settings to resolve unusual conflicts or to customize configuration.

Resource conflicts are indicated by an asterisk (*) next to the conflicting items and also by the "* Conflicts" string in the upper right corner of the Change Configuration window. Conflicts with fixed resources have the asterisk (*) to the left of the slot number of the adapter. Adapters with conflicts are disabled; the program sets bit 0 of POS Register 2 (hex 0102) to 0.

The user can scroll or page through the various screens. Changes are made by rotating field value names through a set of choices using the F5 (Previous) and F6 (Next) keys. Changes are not saved in nonvolatile RAM until the F10 (Save) key is pressed. Help text is provided for each item by pressing the F1 (Help) key.

Note: Automatic Configuration will restore all customized settings to the first setting listed in the ADF that does not conflict.

View Configuration Utility

The View Configuration utility is provided to view the configuration. This is the Change Configuration utility with the change capabilities disabled.

Backup and Restore Configuration Utilities

The Backup Configuration utility provides a means to back up the configuration data to a file on the Reference Diskette. If the battery fails or the battery is changed, the user can use the Restore utility to restore the configuration.

Note: A copy of the Reference Diskette that is not write protected is needed for this backup and restore process.

Copy an Option Diskette Utility

The Copy an Option Diskette utility is a separate program from the Set Configuration program and is accessible through the Main Menu. This utility is used to merge the files with the extension of ADF[1] from an Option Diskette onto a backup copy of the Reference Diskette. The files found on this diskette are compared to the files on the backup copy of the Reference Diskette. If the file does not exist on the Reference Diskette, it is copied. If the file already exists on the Reference Diskette, the dates of the two files are compared, and the file is copied if it has a date later than, or the same as, the corresponding file on the backup diskette.

The Option Diskette must be a DOS formatted diskette.

---

1 The following files are also copied by this utility; however, the file names are reserved for use by IBM: *.DGS, *.PEP, COMMAND.COM, DIAGS.COM, CMD.COM, SC.EXE, UPDATE.EXE, and SYSCONF.

Adapter Description Files

Adapter description files (ADFs) provide POS information and system resource information for system configuration. The ADFs also provide text for the System Configuration utilities, help screens, and prompts. This section provides guidelines for developing the adapter description files.

Format

- File names: @CardID.ADF (high byte of the adapter ID first).
- Type of file: ASCII text.
- Not case sensitive: Key words can be lowercase, uppercase, or mixed. The case is preserved within the user interface text strings.
- Blanks, tabs, new lines: These are considered as white space and ignored, except when in text strings for the Change Configuration user interface.
- Comments: Lines beginning with semicolons are comments and are ignored.

Syntax

The following table shows the meaning of special symbols used in the adapter description file syntax. Values shown as hexadecimal can be specified as decimal or hexadecimal using the Intel format. Values shown as binary must be specified as binary.

Symbol	Meaning
{ }	an optional item
{ }*	0,1,2,... items allowed
{ }+	1,2,3,... items allowed
x | y	either x or y allowed
x{n}	n x's required
[0-9]+	one or more decimal digits

Figure 2. Syntax Symbol Key

```
adf_file => card_id card_name nbytes {fixed_resources}
            {system_memory} {initial_program} {lock_enable}
            {address_width} {programmed_io} {items}*
```

This statement defines the contents of an ADF. The following definitions describe each parameter in detail.

```
card_id => AdapterId number
```

Each ADF must contain the card_id parameter. The character string AdapterId is a keyword and must be present. The Set Configuration program looks for this ID, which matches the ID used in the filename.
Example: `AdapterId 0DEFFh`

```
card_name => AdapterName string
```

Each ADF must contain the card_name parameter. The character string AdapterName is a keyword and must be present.
The string following the AdapterName keyword is displayed as the adapter name in the Change Configuration and View Configuration screens. The length of the AdapterName string is limited to (74 - (length of 'SlotX - ')) characters.
(The length for US English is 66 characters)
Example: `AdapterName "IBM Multiprotocol Communications Adapter"`

```
nbytes => NumBytes number
```

Each ADF must contain the nbytes parameter; it defines the number of POS bytes used by the adapter. The character string NumBytes is a keyword. The number of POS bytes used includes all bytes from POS [0] to the last POS byte used. (If POS [3] is the only POS byte defined, NumBytes should be set to 4.)
Example: `NumBytes 4`

```
fixed_resources => FixedResources pos_setting resource_setting
```

The fixed_resources parameter is optional, and it defines those resources required by an adapter and the corresponding POS data. The character string FixedResources is a keyword and must be present if the adapter has a specific system resource that it must use, such as a hard-wired I/O address or interrupt level.
Note:
Using fixed resources is not recommended because conflicts cannot be resolved automatically and can result in the adapter being disabled.
Example: `FixedResources  POS[1]=XXXX01XXb int 3`

Figure 3 (Part 1 of 9). Adapter Description File Syntax

```
system_memory => SysMem granularity {address_size}
```

The system_memory parameter is required for adapters that use an ADP to configure their system memory. The character string SysMem is the keyword.
The granularity field is required and specifies the smallest block of memory on the adapter that can be disabled. This block size must be divisible by 4KB. The value used in this field specifies the number of 1KB blocks.
The address_size field specifies the number of address bits supported by the adapter, 24 or 32 bits. The default size is 24.
Note:
This keyword requires an ADP.
Example: `SysMem 1024 32`

```
initial_program => InitProg attribute
```

The initial_program parameter is required for adapters that have an initialization program associated with them. The character string InitProg is the keyword. The initialization programs are loaded onto the fixed disk and executed before the operating system is booted from the fixed disk.
The attribute is a number from 0 to 127 and determines the order in which the initialization programs are run. The lower numbers are run first. Values outside the range from 30 to 80 are reserved.
Example: `InitProg 30`
(See "Initialization Programs" on page 49 for more information.)

```
lock_enable => Lock
```

The lock_enable parameter is optional. The character string Lock is the keyword, and its presence enables the Lock function.
An adapter that wishes to communicate with other masters through a memory semaphore protocol (an indivisible test and set operation, such as the Exchange instruction) must indicate this to the Set Configuration program by the Lock keyword.
Example: `Lock`

```
items => {{named_item | input_item} | {begin_device}}*
```

The items parameter is optional and can be any combination of the named_item and input_item parameters, or it can be any number of begin_device parameters.

Figure 3 (Part 2 of 9). Adapter Description File Syntax

```
named_item => NamedItem prompt {named_choice}+  help
```

The named_item parameter is optional, and it defines a field that provides a choice of one or more resources. Each Choice field sets specified POS bits to a unique setting used to identify resources assigned to the adapter. The character string NamedItem is a keyword and must be present if the adapter can be configured to use different resources. The adapter determines the resources to which it is configured by how the POS bytes are set. When a NamedItem is defined, it must be accompanied by a prompt, at least one named_choice, and a help field.

To provide flexibility during configuration, each named_item should contain more than one choice.

Note:

Each ADF is limited to 10 named_items per adapter.

Example:

```
NamedItem
Prompt "Communications Port"
choice "SDLC_1"    pos[0]=XXX1000Xb  io 0380h-038ch  int 3 4
choice "SDLC_2"    pos[0]=XXX1001Xb  io 03a0h-03ach  int 3 4
choice "BISYNC_1" pos[0]=XXX1100Xb  io 0380h-0389h  int 3 4
choice "BISYNC_2" pos[0]=XXX1101Xb  io 03a0h-03a9h  int 3 4
choice "SERIAL_1" pos[0]=XXX0000Xb  io 03f8h-03ffh  int 4
choice "SERIAL_2" pos[0]=XXX0001Xb  io 02f8h-02ffh  int 3
choice "SERIAL_3" pos[0]=XXX0010Xb  io 3220h-3227h  int 3
choice "SERIAL_4" pos[0]=XXX0011Xb  io 3228h-322fh  int 3
choice "SERIAL_5" pos[0]=XXX0100Xb  io 4220h-4227h  int 3
choice "SERIAL_6" pos[0]=XXX0101Xb  io 4228h-422fh  int 3
choice "SERIAL_7" pos[0]=XXX0110Xb  io 5220h-5227h  int 3
choice "SERIAL_8" pos[0]=XXX0111Xb  io 5228h-522fh  int 3
Help
"This port can be assigned as the primary (SDLC1) or
secondary (SDLC2) sdlc, the primary (BISYNC1) or secondary
(BISYNC2) bisync, or as a serial port (Serial 1 through Serial
8).  Use the F5=Previous and the F6=Next keys to change this
assignment in the 'Change configuration' window.  Conflicting
assignments are marked with an asterisk and must be changed
to use the adapter."
```

Figure 3 (Part 3 of 9). Adapter Description File Syntax

```
input_item => Input prompt pos_byte_setting range help
```

The input_item parameter is optional; it defines a field that accepts operator-entered data to initialize a single POS byte, or to define specific bits within that byte. The character string Input is the keyword. The operator can enter up to a three-digit positive integer. The bits to be altered by the input are defined using 0 or 1. The X specifies bits that remain unchanged by the input. The operator input is aligned with the least-significant bit of the pos_byte_setting field not defined as X. The two values are then ORed before the mask is applied and the result is stored. The range defines the limit of the values acceptable as user input. The first number listed in the range will be used as the input for the adapter during automatic configuration.

Example:
```
Input prompt "Enter value"pos[0]=XXXX1111b 0 - 15
      help "Enter a value from 0 - 15."
```

```
prompt => Prompt string
```

The prompt is required when a named_item or input_item is defined. The prompt is used to define a title for a NamedItem or an Input field. The character string Prompt is a keyword and must be present when a NamedItem or Input parameter is present. The string following the Prompt keyword appears after the adapter name in the Change Configuration and View Configuration screens.

Following the prompt string is a field that can be toggled in the Change Configuration screen if two or more named_choices are defined. The length of the prompt cannot exceed 38 characters.

Example: (See the example for named_item).

```
named_choice => Choice {valid_check} choice_name pos_setting
      resource_setting
```

At least one named_choice is required when a NamedItem is defined. The character string Choice is a keyword and must be present when a NamedItem is present. The named_choice field is specified after the prompt. Each Choice must contain the choice_name, which describes the current choice. The choice_name string must not exceed 28 characters. Each Choice must define a pos_setting for at least one POS byte that uniquely identifies the resource_setting defined in the named_choice.

Example:
```
Choice "IO 3F80" pos[1]=00111111b pos[2]=10000000b
      io 3f80-3f87
```

Figure 3 (Part 4 of 9). Adapter Description File Syntax

```
valid_check => Vcheck
```

The valid_check parameter invokes the ADP to check the specified choice for validity. The character string Vcheck is the keyword. The ADP ensures that the choice is valid for the system configuration. If the Set Configuration program is unable to load the ADP, the choice is treated as invalid. If all the choices within a named_item are invalid, the Set Configuration program deletes the named_item from the list of items for that adapter.

```
help => Help string
```

The help field is a string of text used to give the user assistance at a prompt. This text is displayed in the Change Configuration and View Configuration screens when the cursor is at the associated prompt and the F1 key is pressed. The character string Help is a keyword and must be present when the NamedItem or Input parameter is present. The string following the Help keyword is the text that describes the prompt. The length of the help string is limited to 1000 characters.
Example: (See the example of named_item.)

```
pos_setting => {pos_byte_setting}+
```

The pos_setting parameter must contain at least one pos_byte_setting. See the definition of pos_byte_setting.

```
pos_byte_setting => POS[number]=pos_bit{8}b
```

Each pos_byte_setting field defines one POS byte.
The character string POS is a keyword and precedes each POS byte defined. The number of the POS byte is specified in brackets and refers to the POS bytes as shown below:

- number = 0, POS byte 0 at port 102h
- number = 1, POS byte 1 at port 103h
- number = 2, POS byte 2 at port 104h
- number = 3, POS byte 3 at port 105h

The closing bracket must be followed by an equal sign, and then a bit definition of the POS byte (See pos_bit for more). The bit definition must define all 8 bits of the byte. Bit 0 of pos[0] and bit 7 of pos[3] should always be defined as X.
Example: `pos[0]=XXX1001Xb`

```
pos_bit => x | X | 0 | 1
```

A pos_bit can be defined as a mask bit (x or X), a clear bit (0), or a set bit (1).
Example: `pos[0]=XXX1001Xb`

Figure 3 (Part 5 of 9). Adapter Description File Syntax

```
resource_setting => {ioblock_list} {interrupt_list} {arb_list}
            {memaddr_list} {exec_adp}
```

The resource_setting parameter defines a list of system resources. The resources can be fixed resources required by the adapter, or they can be resources the adapter uses when configured to a specific choice in a named_item. The resources can consist of the following:

- Range of I/O addresses (limited to 16 ranges).
- List of interrupt levels (limited to 16).
- List of arbitration levels (limited to 16).
- Range of memory addresses (limited to 2 ranges).
- An ADP call for nonsystem memory allocation.

Example: (See the specific resource definition).

```
ioblock_list => IO {range}+
```

The ioblock_list parameter is a list of one or more ranges of I/O addresses. The character string IO is a keyword and must be present in the ioblock_list.
Example: `IO 4220h-4227h`

```
interrupt_list => Int {number}+
```

The interrupt_list parameter is a list of one or more interrupt levels that are used concurrently. The character string Int is a keyword and must be present in the interrupt_list.
Example: `Int 3 4`

```
arb_list => Arb {shared_arb}{number}+
```

The arb_list parameter is a list of one or more arbitration levels that are used concurrently. The character string Arb is a keyword and must be present in the arb_list.
Example: `Arb 1`

```
shared_arb => SharedArb
```

The character string SharedArb is the keyword and indicates that, in a controlled environment, the arbitration level in the named_item parameter can be shared with the arbitration level of other devices that support sharing of arbitration levels.

Example: `Arb ShareArb 1`

Figure 3 (Part 6 of 9). Adapter Description File Syntax

```
memaddr_list => Mem {range}+
```

The memaddr_list parameter is a list of one or more ranges of RAM or ROM addresses. The character string Mem is a keyword and must be present in the memaddr_list. This keyword is used to allocate the memory address space in the hex 000C0000 through hex 000DFFFF and the nonsystem memory above 1MB.
Include the address for each Choice in the choice_name field so that the address is also displayed during Change Configuration.
Note:
Because of the limited space available, each adapter should limit the address space used in segments 0C000 and 0D000 to 16KB.
Example: `Mem 0CC000h - 0CDFFFh  0E00000h - 0EFFFFFh`

```
exec_adp => Exec
```

The exec_adp parameter is required for adapters containing nonsystem memory that can be initialized by an ADP. The character string Exec is the keyword that defines the ADP and should be the last Choice in the list of named_choices.
The Exec keyword indicates to the Set Configuration program that the adapter has an ADP.
Note:
This keyword requires an ADP.
See the example on page 22.

```
address_width => Address 24|32
```

The address_width field specifies the number of address bits supported by the adapter, 24 or 32 bits. The character string Address is the keyword and the default size is 24.
Example: `Address 32`

```
programmed_I/O => PIO
```

The programmed_I/O parameter is used to specify whether an adapter includes address bits 24 through 31 in its address decode. If this parameter is not specified, the adapter does respond to these address bits.
Note:
A device is address limited if the ADF does not have the PIO parameter and the slot or the adapter is address limited.
Example: `PIO`

Figure 3 (Part 7 of 9). Adapter Description File Syntax

```
begin_device =>
  Begin
  {device} {rs422} {DMA_not_used}
  {named_item |input_item}*
  End
```

The begin_device parameter is optional, and is used to define a device block on an adapter. A device block is a series of items that define POS settings for like devices on an adapter, but are not associated with any other devices that might be on the adapter.

```
device =>  Device device_id {implement_id {revision_level}}
```

The device parameter is optional and defines the device types in ABIOS for the individual devices on an adapter or the system board. The character string Device is the keyword. The device_id parameter is the ABIOS device ID, which is described in the *IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference*.

The implement_id parameter is the implementation identifier, and the revision_level parameter is the implementation revision level.
The implementation identifier and revision level are optional and default to 0 if not specified; however, the revision level cannot be specified without the implementation identifier.
Example: Device 2 1

```
rs422 => RS422
```

The rs422 parameter is optional, and indicates that the serial device uses a RS422-type interface. The character string RS422 is the keyword.
Example: RS422

```
DMA_not_used => NoDMA
```

The DMA_not_used parameter indicates that arbitration levels for the named_items within the begin_device do not require a DMA channel. The character string NoDMA is the keyword. If this parameter is specified, the Set Configuration program does not indicate that a DMA channel is required when an arbitration level is used. If this parameter is not specified, the Set Configuration program indicates that a DMA channel is required when an arbitration level is used.
Note:
Bus masters should use the NoDMA keyword.

Figure 3 (Part 8 of 9). Adapter Description File Syntax

```
range => number - number

number => [0-9]+ {d} |  [0-9a-f]+ h |  [0-9A-F]+ H

string => " [ascii except for "]+ "
```

The string field is a set of ASCII characters beginning with a double quote (") and ending with a double quote.
Example:

```
"This port can be assigned as:  primary (SDLC1) or
secondary (SDLC2) sdlc port, primary (BISYNC1) or secondary
(BISYNC2) bisync port, or as a serial port (Serial 1 through
Serial 8).  Use the F5=Previous and the F6=Next keys to change
this assignment in the 'Change configuration' window.
Conflicting assignments are marked with an asterisk and must
be changed to use the adapter."
```

Figure 3 (Part 9 of 9). Adapter Description File Syntax

Example

The following is an example of an adapter description file for the IBM Personal System/2 Multiprotocol Communications Adapter/A. The name of the file for this adapter is @DEFF.ADF. An explanation of each numbered item begins on page 18.

```
AdapterId 0DEFFh 1

AdapterName "IBM Multiprotocol Communications Adapter" 2

NumBytes 2 3

NamedItem 4

Prompt "Communications Port"

choice "SDLC_1"   pos[0]=XXX1000Xb  io 0380h-038ch  int 3 4
choice "SDLC_2"   pos[0]=XXX1001Xb  io 03a0h-03ach  int 3 4
choice "BISYNC_1" pos[0]=XXX1100Xb  io 0380h-0389h  int 3 4
choice "BISYNC_2" pos[0]=XXX1101Xb  io 03a0h-03a9h  int 3 4
choice "SERIAL_1" pos[0]=XXX0000Xb  io 03f8h-03ffh  int 4
choice "SERIAL_2" pos[0]=XXX0001Xb  io 02f8h-02ffh  int 3
choice "SERIAL_3" pos[0]=XXX0010Xb  io 3220h-3227h  int 3
choice "SERIAL_4" pos[0]=XXX0011Xb  io 3228h-322fh  int 3
choice "SERIAL_5" pos[0]=XXX0100Xb  io 4220h-4227h  int 3
choice "SERIAL_6" pos[0]=XXX0101Xb  io 4228h-422fh  int 3
choice "SERIAL_7" pos[0]=XXX0110Xb  io 5220h-5227h  int 3
choice "SERIAL_8" pos[0]=XXX0111Xb  io 5228h-522fh  int 3

Help

"This port can be assigned as the primary (SDLC1) or
secondary (SDLC2) sdlc port, the primary (BISYNC1) or secondary
(BISYNC2) bisync port, or as a serial port (Serial 1 through
Serial 8). Use the F5=Previous and the F6=Next keys to
change this assignment. Conflicting assignments are marked
with an asterisk and must be changed."
```

```
NamedItem 5
Prompt "Arbitration Level for SDLC"

choice "Level_1"   pos[1]=XXXX0001b  arb  1
choice "Level_0"   pos[1]=XXXX0000b  arb  0
choice "Level_2"   pos[1]=XXXX0010b  arb  2
choice "Level_3"   pos[1]=XXXX0011b  arb  3
choice "Level_4"   pos[1]=XXXX0100b  arb  4
choice "Level_5"   pos[1]=XXXX0101b  arb  5
choice "Level_6"   pos[1]=XXXX0110b  arb  6
choice "Level_7"   pos[1]=XXXX0111b  arb  7
choice "Level_8"   pos[1]=XXXX1000b  arb  8
choice "Level_9"   pos[1]=XXXX1001b  arb  9
choice "Level_10"  pos[1]=XXXX1010b  arb 10
choice "Level_11"  pos[1]=XXXX1011b  arb 11
choice "Level_12"  pos[1]=XXXX1100b  arb 12
choice "Level_13"  pos[1]=XXXX1101b  arb 13
choice "Level_14"  pos[1]=XXXX1110b  arb 14

Help

"This assignment need only be changed if it is in conflict
with another assignment.  Conflicting assignments are marked
with an asterisk.  Use the F5=Previous and the F6=Next keys
to change arbitration level assignments.  Using arbitration
levels, this adapter accesses memory directly without
burdening the computer's main microprocessor.  An
arbitration level of 0 has the highest priority, and
increasing levels have corresponding decreased priority"
```

1 The card_id for this adapter is hex 0DEFF. This is an ASCII representation of the ID generated by the adapter. The high byte is followed by the low byte. The card_id is required for all ADFs.

2 The card_name is "IBM Multiprotocol Communications Adapter." The card_name is required for all ADFs.

3 The nbytes in this file is 2, which indicates the adapter uses two POS bytes located at hex 0102 and 0103.

4 This is the first named_item for the adapter. The title of the field is "Communications Port." The user can toggle between the 12 named_choices. Each named_choice has a unique pos_setting assigned to it in bit locations 1 through 4 of POS byte hex 0102 (pos [0]). Also shown is a resource_setting that corresponds to the

pos_setting of the named_choice. The resources allocated in this named_item are I/O addresses and interrupt levels. A help string for this named_item is provided below the last named_choice.

5 This is the second named_item for the adapter. The title of the field is "Arbitration Level for SDLC." The user can toggle between the 15 named_choices. Each named_choice has a unique pos_setting assigned to it in bit locations 0 through 3 of POS byte hex 0103 (pos [1]). Also shown is a resource_setting that corresponds to the pos_setting of the named_choice. The resources allocated in this named_item are arbitration levels. A help string for this named_item is provided below the last named_choice.

Adapter Description Program

The adapter description program (ADP) is program used to configure memory space on an adapter that cannot be handled by normal ADF syntax. Memory adapters that require initialization of more than their POS values, however, should use an initialization program, not an ADP. The ADP is called when the item selected in the ADF contains the appropriate keyword.

Two basic types of memory can be assigned addresses within the memory address space: system memory and nonsystem memory.

System memory is memory that is managed and allocated by the primary operating system. It remains assigned to or fixed in the 4GB physical address space, and its contents can be accessed or modified only by an independent master (bus master, system master, or DMA controller).

Nonsystem memory is memory that is not managed or allocated by the primary operating system. It is made up of memory-mapped I/O devices, memory on an adapter that can be directly modified by the adapter, or memory that can be relocated within the address space, such as bank switched and EMS (expanded memory specifications).

The ADP initializes its POS data in the POS array and updates the parameter block. It must be in EXE program format, should not exceed 10KB, and start its code at offset hex 100. The filename for the ADP is CCardID.ADF (CardID is the adapter ID in hex).

Notes:

1. Memory address space is the only system resource that can be allocated by an ADP.
2. The Set Configuration program does not include support for memory adapters that have translation tables.
3. ADPs should not output directly to the display or prompt the user for input; unpredictable results will occur.
4. To ensure system integrity, ADPs must not write directly to locations in nonvolatile RAM.

Syntax for ADPs

There are three keywords in the ADF syntax that define an ADP: SysMem, Exec, and Vcheck. SysMem is used to assign address space to system memory, and Exec is used to assign address space to nonsystem memory.

System Memory ADPs

The SysMem keyword in the ADF indicates that the adapter contains system memory and that an ADP is used to configure the adapter's POS data and the amount of its system memory.

Adapters should not split their memory between the 1MB to 16MB range and the 16MB to 4GB range unless the memory is contiguous across the 16MB address.

System memory is the first resource allocated by the Set Configuration program. The system memory ADPs are run before other resources are allocated and in the following priority:

1. System board memory
2. Memory address size (24-bit address followed by 32-bit)
3. Memory granularity (larger block size first)
4. Slot number starting with connector 1.

If the ADP is not found by the Set Configuration program, the user is prompted to insert the appropriate diskette into drive A. If the ADP still is not found, the adapter is disabled.

Nonsystem Memory ADPs

An ADP can be used by adapters that want to map nonsystem memory into the address space above 1MB. The memory on these adapters does not get included as part of the top of memory value stored in CMOS RAM. This memory must be mapped starting at the highest addressable location and counting down.

The Exec keyword, included within a NamedItem in the ADF, indicates that the adapter has an ADP. It should be the last Choice of a NamedItem list. The ADP is run at the time that Choice is selected. If the ADP cannot be found by the Set Configuration program, the user is prompted to insert the appropriate diskette into drive A. If the ADP still cannot be found, the adapter will be disabled if that Choice is selected. When the Exec keyword is used, POS bits that are set by the ADP are indicated by X's. The following is an example of how the Exec keyword is used.

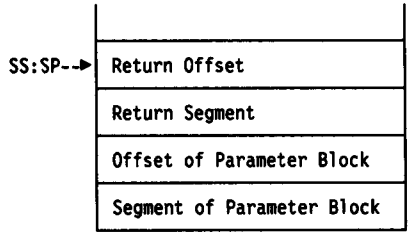
```
NamedItem Prompt "Adapter RAM"
     Choice "C000"
          pos[2]=XXXX0000b
          pos[1]=11000000b  mem 0c0000h-0c7fffh

     Choice "C800"
          pos[2]=XXXX0000b
          pos[1]=11001000b  mem 0c8000h-0cffffh

     Choice "D000"
          pos[2]=XXXX0000b
          pos[1]=11010000b  mem 0d0000h-0d7fffh

     Choice "D800"
          pos[2]=XXXX0000b
          pos[1]=11011000b  mem 0d8000h-0dffffh

     Choice "Extended Memory"
          pos[2]=XXXXXXXXb
          pos[1]=XXXXXXXXb  Exec

     Help
          "Select a memory address space."
```

Program Call

When the Set Configuration program calls the ADP, the address of the ADP parameter block is passed on the stack. This parameter block contains pointers to data structures that are used by the ADP.

The Set Configuration program checks the parameter block before and on return from the ADP; however, the ADP must maintain the integrity of these data structures during its execution.

When the ADP is given control, the stack is set up as follows:



The ADP must preserve the contents of the DS, DI, SI, and BP registers as well as the stack and stack pointers. The ADP ends using a Far Return with the stack set up as shown.

The following figure shows the format of the parameter block.

00H	ADP Function	Slot
02H	Offset to System Memory (< 16MB)	
04H	Segment to System Memory (< 16MB)	
06H	Offset to System Board Memory (< 16MB)	
08H	Segment to System Board Memory(< 16MB)	
0AH	Offset to POS Array	
0CH	Segment to POS Array	
0EH	Offset to POS Mask	
10H	Segment to POS Mask	
12H	Offset to Adapter ID List	
14H	Segment to Adapter ID List	
16H	Offset to ADP Function Call Entry	
18H	Segment to ADP Function Call Entry	
1AH	Offset to Resource Query Parameters	
1CH	Segment to Resource Query Parameters	
1EH	Offset to System Memory (16MB to 4GB)	
20H	Segment to System Memory (16MB to 4GB)	
22H	Offset of System Board Memory (16MB to 4GB)	
24H	Segment of System Board Memory (16MB to 4GB)	

Figure 4. ADP Parameter Block

The following is a description of the parameter block values and the data structures that are pointed to.

ADP Function: This byte specifies the operation that the ADP will perform. The functions are listed on page 28. The ADP uses this byte to return the results of the operation, successful (0) or unsuccessful (not 0). The ADP returns a hex 01 in this byte if the operation was not successfully completed, and returns a hex FF if the function is not supported.

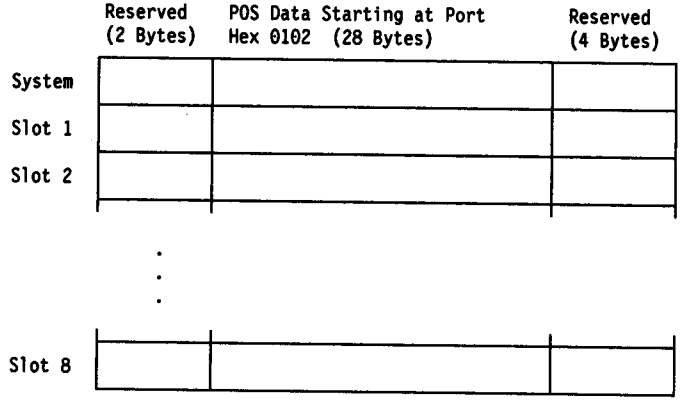
Slot: This is the slot number of the adapter to be configured. The value is not required for all ADP functions.

System Memory Below 16MB: This is a pointer to the current count of all system memory below 16MB. The count is a doubleword value and is the number of 1KB blocks of memory below 16MB already allocated as system memory. (This count includes the count in System Board Memory Below 16MB.) This value is adjusted by ADPs as they map their system memory into the system address space.

The value is initially set to 0 before any configuring is started. An ADP must not increase this value to the point that memory gets mapped beyond the 16MB address.

System Board Memory Below 16MB: This is a pointer to the count of system memory on the system board below 16MB. The count is a doubleword value and is the number of 1KB blocks of system board memory below 16MB. The memory from 640KB to 1MB is included in this count. This value is set by the system description program and must not be adjusted by any other program.

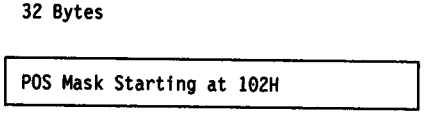
POS Array: This is a pointer to a table of the POS data for the system board and the I/O slots. The ADP should initialize only those locations that correspond to its adapters, and only the four POS data bytes from hex 0102 to 0105. The following is the format of the table and each entry.



POS Mask: This is a pointer to the 32-byte field containing the POS mask for adapter POS information. The POS mask data is initialized by an ADP, and is applied to POS data for all slots with that adapter ID.

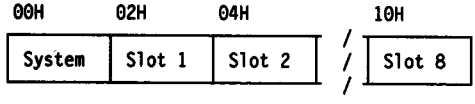
The mask bits are set to 1 for all POS bit positions that are initialized by the ADP. The mask bits are set to 0 for those POS bits that are not used, or that are initialized by statements in the ADF.

The following is the format of the POS mask data pointed to by this field.



The POS mask must be set every time the ADP is called.

Adapter ID List: This is a pointer to a list containing the system and adapter IDs that are currently installed in the system. The adapter IDs are stored as words at the following offsets.



ADP Function Call Entry: This pointer is the program entry point for calls by the ADP to the Set Configuration program when using the Resource Query function calls (see "Resource Query Functions" on page 34). The ADP does a Far Call when calling the Set Configuration program.

Resource Query Parameters: This pointer is the address for the Resource Query parameter block that the ADP uses to pass parameters when calling a Resource Query function. The address for this parameter block is allocated by the Set Configuration program and should not be changed by the ADP.

System Memory Between 16MB and 4GB: This is a pointer to the current count of all system memory between 16MB and 4GB. The count is a doubleword value and is the number of 1KB blocks from 16MB to below 4GB already allocated as system memory. This value is adjusted by ADPs as they map their system memory into the system address space.

The value is initially set to 0 before any configuring is started. An ADP must not increase this value to the point that system memory gets mapped to the 4GB address (must be less than 4GB).

System Board Memory Between 16MB and 4GB: This is a pointer to the count of system memory on the system board between 16MB and 4GB. The count is a doubleword value and is the number of 1KB blocks of system memory on the system board from 16MB to 4GB. This value is set by the system description program and should not be adjusted by other description programs.

ADP Functions

When control is passed to an ADP, a function code is also passed. This function code indicates the operation to be performed by the ADP. The ADPs do not need to support all functions. If a function code is passed to an ADP that is not supported, the ADP should return a hex FF in the Function byte.

The following functions are supported by the Set Configuration program.

Function (Hex)	Description
01	Configure the adapter in the given slot.
02	Configure all adapters corresponding to the ADP.
03	Deallocate resources for the adapter in the given slot.
04	Deallocate resources for all adapters corresponding to the ADP.
05	Deduce the configuration of the adapter in the given slot.
06	Disable a block of system memory.
07	Remap the system memory to above the 16MB address.
08	Remap the system memory to below the 16MB address.
09	Configure the system memory on the adapter in the given slot.
0A	Configure the system memory on all adapters corresponding to the ADP.
0B	Deallocate the system memory on the adapter in the given slot.
0C	Deallocate the system memory on all adapters corresponding to the ADP.
0D	Deduce the configuration of the system memory on the adapter in the given slot.
80	Check the validity of an ADF choice

Figure 5. ADP Functions

Configure the Adapter

This function is indicated by 01 in the Function byte. The ADP configures the adapter in the slot specified, and updates the appropriate system memory and POS array values in the parameter table before returning to the Set Configuration program.

All ADPs defined by the SysMem or Exec keywords must support this function.

Configure All Adapters

This function is indicated by 02 in the Function byte. All adapters associated with that ADP are configured. If more than one adapter is being configured during this function, they are configured in the ascending order of their slot number. The ADP updates the appropriate system memory and POS array values in the parameter table before returning to the Set Configuration program.

All ADPs defined by the SysMem keyword must support this function.

Deallocate Resources for an Adapter

This function is indicated by 03 in the Function byte. The resources allocated to the slot specified are deallocated. An ADP can deallocate only those resources it has allocated. The ADP updates the appropriate system memory and POS array values in the parameter table before returning to the Set Configuration program.

All ADPs defined by the SysMem or Exec keywords must support this function.

Note: The Set Configuration program can deallocate memory space, previously allocated by an ADP, without calling the ADP.

Deallocate Resources for All Adapters

This function is indicated by 04 in the Function byte. The ADP deallocates all resources for all its adapters. The ADP can affect only those resources and adapters that it changed during allocation. The ADP updates the appropriate system memory and POS array values in the parameter table before returning to the Set Configuration program.

All ADPs defined by the SysMem or Exec keywords must support this function.

Deduce Adapter Configuration

This function is indicated by 05 in the Function byte. The ADP interprets the data in the POS array for the slot specified and determines what memory space is allocated to the adapter.

If the ADP was defined by the Exec keyword, it uses the Request Memory Block function to request the address space again. If the address space is not allocated, the ADP leaves the POS data unchanged and returns with an error in the Function byte.

| If the ADP was defined by the SysMem keyword, the current top of
| memory is checked to determine if this adapter was previously
| assigned the next address space. If not, memory on another adapter
| may have been deallocated, and the ADP should reconfigure its
| adapter to the current top of memory. The ADP then updates the
| current count of system memory in the parameter block before
| returning.

| When the ADP detects a difference between the actual memory on the
| adapter and its configuration, the ADP calls the Enable Memory
| Prompt function and handles memory according to the user response.

| All ADPs defined by the SysMem or Exec keywords must support this
| function.

| If the Set Configuration program is deducing the configuration of an
| adapter and a match cannot be found between the data stored in
| CMOS/NVRAM and any of the ADF choice POS definitions, the
| adapter will be configured to the first choice in the list and the
| CMOS/NVRAM will be updated to reflect this. If the adapter is in
| conflict with any other adapters when configured to the first choice,
| the adapter will be disabled. The only indication that the user will
| have of this change is by looking at the Change or View Configuration
| screen. A message indicating that changes were made will also be
| displayed when the user exits the Set Configuration menu.

| Disable Block of System Memory

| This function is indicated by 06 in the Function byte. The ADP
| disables the smallest amount of memory that it possibly can for the
| specified slot. The ADP also decreases the current count of memory
| value affected (either in the below 16MB value or the 16MB to 4GB
| value), and updates the POS array to reflect the disabled memory.

| The ADP can interpret the POS data to determine where its memory
| was previously mapped. If its memory is not currently mapped as the
| top of memory, the ADP does not change any values and indicates a
| successful operation. If its memory is currently mapped as the top of
| memory, but the ADP cannot disable the memory, the ADP returns a
| hex 01 in the ADP Function byte indicating the operation was not
| successfully completed.

| The ADP should support multiple calls to disable memory. For each
| call, an additional block of memory should be disabled until all
| memory on the adapter is disabled. If all memory has been disabled

| and the ADP is called again, the ADP returns an error indicating that
| all memory for the slot specified is already disabled.

| All ADPs defined by the SysMem keyword must support this function.

| Remap System Memory to Above 16MB

| This function is indicated by 07 in the Function byte. All memory is
| remapped to the area above 16MB. The ADP updates its POS values
| in the POS array and also updates both top of memory values to
| indicate the changes. If its memory is not currently mapped as the
| top of memory, the ADP does not change any values and indicates a
| successful operation.

| All ADPs defined by the SysMem keyword must support this function;
| adapters that have specified 24-bit addressing will not receive this
| request.

| Remap System Memory to Below 16MB

| This function is indicated by 08 in the Function byte. All memory for
| this adapter is remapped to the area below 16MB. The ADP updates
| its POS values in the POS array and also updates both top of memory
| values to reflect the changes. If its memory is not currently mapped
| as the top of memory, the ADP does not change any values and
| indicates a successful operation.

| All ADPs defined by the SysMem keyword must support this function.

| Configure System Memory on an Adapter

| This function is indicated by 09 in the Function byte and is used only
| when the ADP is defined with both the SysMem and Exec keywords.
| The ADP configures only the system memory on the adapter in the
| slot specified. The ADP updates the appropriate system memory and
| POS array values in the parameter table before returning to the Set
| Configuration program.

| Configure System Memory on All Adapters

| This function is indicated by hex 0A in the Function byte and is used
| only when the ADP is defined with both the SysMem and Exec
| keywords. The ADP configures only the system memory on all
| adapters associated with the ADP. If more than one adapter is being
| configured, they are configured in the ascending order of their slot
| number. The ADP updates the appropriate system memory and POS

| array values in the parameter table before returning to the Set
| Configuration program.

| Deallocate System Memory on an Adapter

| This function is indicated by hex 0B in the Function byte and is used
| only when the ADP is defined with both the SysMem and Exec
| keywords. The system memory previously allocated to the adapter
| by the ADP is deallocated. The ADP updates the appropriate system
| memory and POS array values in the parameter table before
| returning to the Set Configuration program.

| Deallocate System Memory on All Adapters

| This function is indicated by hex 0C in the Function byte and is used
| only when the ADP is defined with both the SysMem and Exec
| keywords. The ADP deallocates system memory that it previously
| allocated for all its adapters. The ADP updates the appropriate
| system memory and POS array values in the parameter table before
| returning to the Set Configuration program.

| Deduce System Memory Configuration on an Adapter

| This function is indicated by hex 0D in the Function byte and is used
| only when the ADP is defined with both the SysMem and Exec
| keywords. The ADP checks the current top of memory to determine if
| the adapter was previously assigned the next address space. If not,
| memory on another adapter may have been deallocated and the ADP
| should reconfigure the starting address of its adapter to the current
| top of memory. The ADP then updates the current count of system
| memory in the parameter table before returning.

| When an ADP detects a difference between the actual memory on the
| adapter and its configuration, the ADP uses the Enable Memory by
| Prompt function and handles memory according to the user response.

| Check Validity of Choice

| This function is indicated by hex 80 in the function byte. It indicates
| whether the choice is valid or not for the current system
| configuration. The parameter block for this function is different than
| those defined previously. The following figure shows the parameter
| block for this function.

00H	Hex 80	Hex 00
02H	Starting Address of the First I/O Address Range (0 if no I/O Address Range)	
04H	First Interrupt Level (0 if no Interrupt Level)	
06H	First Arbitration Level (Hex 0FFFF if no Arbitration Level)	
08H	Starting Address of the First Memory Address Range (0 if no Memory Address Range)	
0CH	Offset of the Adapter IDs List.	
0EH	Segment of the Adapter IDs List.	
10H	Offset of the ADP Function Call Entry Point	
12H	Segment of the ADP Function Call Entry Point	
14H	Offset of the Resource Query Parameters	
16H	Segment of the Resource Query Parameters	

Figure 6. Check Validity of ADF Choice Function.

If the choice is valid, the ADP returns a 0 in the function byte. If the choice is not valid, the named_item is not included in the list of available settings during the configuration process.

All ADPs that use the Vcheck keyword must support this function.

Resource Query Functions

The Resource Query functions allows the ADP to communicate with the Set Configuration program. The ADP can query the memory resource-allocation table and request a block of addresses for an adapter. Before calling, the ADP loads the specific values into the Resource Query parameter block for that function and restores the DS register if it was changed.

A Resource Query function is initiated by making a far call to the program entry point passed to the ADP in the parameter block. Most Resource Query functions are designed to allocate address space for nonsystem memory. ADPs configuring system memory are generally concerned with only the current top of memory value.

Code (Hex)	Description
00	Request Memory Block
01	Check Memory Block
02	De-allocate Memory Block
03	Enable an Adapter
04	Disable an Adapter
05	Enable Memory with a Prompt
06	Get Number of Slots
07	Get Slot Information

Figure 7. Resource Query Function Calls.

The allocation of nonsystem is handled by the Set Configuration program. The ADP simply requests a block of memory within a specified range. The ADP should try to map its memory into the highest block possible.

Note: The size of the block and high end of the range plus 1 must be divisible by 4KB.

The Set Configuration program responds with a return code that indicates whether the request was successful or unsuccessful. If the request is successful, the Set Configuration program has also returned the starting address of the memory block allocated. The ADP then sets the appropriate bits in the POS array.

The Set Configuration program attempts to allocate the highest block of memory within the requested range. If the highest block is not available, the starting address is decreased by the size of the block requested until either a block of memory space is available, or the starting address is below the requested range.

When the ADP returns control to the Set Configuration program, it must indicate in the Function byte whether or not the configuration was successful.

The parameter block for a Resource Query function is defined as follows:

00H	Function	Slot
02H	Start_add_high	
04H	Start_add_low	
06H	End_add_high	
08H	End_add_low	
0AH	Size_high	
0CH	Size_low	

Figure 8. Resource Query Parameter Block

Request Memory Block

This function is used to request a block of memory space for adapters with nonsystem memory. The ADP provides the range where its memory can be mapped and the number of bytes to be mapped.

The ADP can only allocate memory to an adapter with its ID. Once a memory block has been allocated, it remains allocated until that ADP deallocates the space, or until automatic configuration is run.

The following figure shows the values for the parameter passed between the ADP and the Set Configuration program.

```
        Function = 00
        Slot = Slot number of adapter
        Start_addr_high = Beginning of range (High Word)
        Start_addr_low = Beginning of range (Low Word)
        End_addr_high = End of range (High Word)
        End_addr_low = End of range (Low Word)
        Size_high = Requested block size in bytes (High Word)
        Size_low = Requested block size in bytes (Low Word)

    Return:
        Successful:
            Function = 00
            Start_addr_high = Beginning of the block allocated
            Start_addr_low = Beginning of the block allocated
        Unsuccessful:
            Function = Error code (01, 02, 03, 06, 08 or 09)
                       (refer to "Error Codes" on page 40)
            All other values unchanged.
```

Figure 9. Request Memory Block Function

Check Memory Block

This function allows the ADP to get the current memory block assigned to an adapter in a given slot. An ADP can check the allocated memory blocks for any slot.

Because a list of memory blocks can be allocated to an adapter, the ADP must specify which block in the list to use. An ADP can check the entire list by starting at number 0 and checking the next block until an error code is returned.

```
        Function = 01
        Slot = Slot number of adapter
        End_addr_high = Block number

    Return:
        Successful:
            Function = 00
            Start_addr_high = Beginning of block (High Word)
            Start_addr_low = Beginning of block (Low Word)
            Size_high = Allocated block size in bytes (High Word)
            Size_low = Allocated block size in bytes (Low Word)
        Unsuccessful:
            Function = Error code (01, 05 or 09)
                       (refer to "Error Codes" on page 40)
            All other values unchanged.
```

Figure 10. Check Memory Block Function

Deallocate Memory Block

This function allows the ADP to deallocate a block of memory for a given slot. The ADP can deallocate only the memory space assigned to adapters with its ID. The ADP specifies only the beginning address of the memory block.

```
        Function = 02
        Slot = Slot number of adapter
        Start_addr_high = Beginning of memory block (High Word)
        Start_addr_low = Beginning of memory block (Low Word)

    Return:
        Successful:
            Function = 00
        Unsuccessful:
            Function = Error code (01, 04 or 09)
                       (refer to "Error Codes" on page 40)
            All other values unchanged.
```

Figure 11. Deallocate Memory Block Function

Enable an Adapter

This function allows the ADP to enable and temporarily configure one of its adapters. Because the data in the POS mask array is written when the adapter is enabled, the ADP must initialize the POS mask before enabling the adapter.

```
        Function = 03
        Slot = Slot number of the adapter

    Return:
        Successful:
            Function = 00
        Unsuccessful:
            Function = Error code (01)
                       (refer to "Error Codes" on page 40)
            All other values unchanged.
```

Figure 12. Enable an Adapter Function

Disable an Adapter

This function disables the adapter. Configuration information is not required because only the adapter enable bit is changed.

```
        Function = 04
        Slot = Slot number of the adapter

    Return:
        Successful:
            Function = 00
        Unsuccessful:
            Function = Error code (01)
                       (refer to "Error Codes" on page 40)
            All other values unchanged.
```

Figure 13. Disable an Adapter Function

Enable Memory by Prompt

This function allows an ADP to prompt the user for a decision when the amount of memory on the adapter does not match configuration. When this function is called, the Set Configuration program displays a message and waits for the user to type a Y or N. The message asks the user if the memory should be enabled and the adapter reconfigured. If a Y is typed, the Set Configuration program returns a 01 in the Function byte and the ADP enables memory and reconfigures the adapter. If an N is typed, the Set Configuration program returns a 00 in the Function byte, the ADP disables the changed memory and does not change the configuration.

This function should be used only when the ADP was called with a Deduce function. An example of when this function is used is if memory was added to a configured adapter.

Note: The return code specified in the Function byte for Enable Memory by Prompt function is different from the error code described in "Error Codes" on page 40.

Architecture Note:

If the slot number specified in the parameter block is not valid, the Set Configuration program also returns a 01 in the Function byte.

```
        Function = 05
        Slot = Slot number of the adapter

    Return:
        Enable memory:
            Function = 01
        Leave memory disabled:
            Function = 00
```

Figure 14. Enable Memory by Prompt Function

Get Number of Slots

This function allows the ADP to determine the number of channel connectors in the system, which can be used to determine the size of the POS array, adapter ID list, and other arrays.

```
        Function = 06

   Return:
        Function = 00
        Slot = Number of channel connectors in the system
```

Figure 15. Get Number of Slots Function

Get Slot Information

This function requests information about a particular channel connector. The ADP can use this information to determine the address size of the connector and whether or not the connector has a video extension.

```
        Function = 07
        Slot = Slot number to get information about

   Return:
        Successful:
             Function = 00
             Slot = address size of slot
             Start_addr_high:
                bit 0 = 1, slot contains video extension
                        0, slot does not contain video extension
                bits 1 - 15 = reserved (set to 0)
        Unsuccessful:
             Function = Error code (01)
                      (refer to "Error Codes" on page 40)
             All other values unchanged.
```

Figure 16. Get Slot Information Function

Error Codes

The following is a list of the error codes used by the ADP and Set Configuration program. The error codes are the same for the ADP functions and the Resource Query functions.

Error Code	Description
01	The slot number specified was not valid.
02	The size specified is not valid.
03	The resource is already allocated to an adapter.
04	Attempted to deallocate resources that were not allocated.
05	A list value is not valid.
06	The memory range specified is not valid.
07	The function passed by the ADP is not valid.
08	Cannot allocate any more memory blocks to this adapter.
09	All memory for this adapter has already been disabled.
FF	Operation is not supported.

Figure 17. Error Codes

ADP Example

The following is an example of an adapter description program that is defined by the Exec keyword.

```
;-------------------------------------------------------------------;
;                                                                   ;
;  Sample ADP defined by the EXEC keyword.                          ;
;                                                                   ;
;  Adapter specifications:                                          ;
;       Adapter has a 1M buffer of nonsystem memory that can be     ;
;       mapped on any 1M boundary in the area from 2G to 4G - 1.    ;
;                                                                   ;
;       Adapter POS registers:                                      ;
;               register 100h = 77h                                 ;
;               register 101h = 77h                                 ;
;               register 102h =                                     ;
;                       bit 0 = adapter enable bit                  ;
;                       bit 1 = nonsystem memory enable             ;
;               register 103h = low byte of high word of assigned   ;
;                               memory block.                       ;
;               register 104h = high byte of high word of assigned  ;
;                               memory block.                       ;
;                       (Low word of memory block always 0).        ;
;                                                                   ;
;-------------------------------------------------------------------;
.286C
.286p

;-------------------------------------------------------------------;
;  Constants:
;-------------------------------------------------------------------;

adapter_id              equ     7777h   ;ID for this adapter
begin_block_h           equ     8000h   ;high word for start of 2G
begin_block_l           equ     0       ;low word for start of 2G
end_block_h             equ     0ffffh  ;high word for end of 4G - 1
end_block_l             equ     0ffffh  ;low word for end of 4G - 1
size_h                  equ     10h     ;high word for size of 1M
size_l                  equ     0       ;low word for size of 1M
mem_enable_mask         equ     2       ;mask for memory enable bit in POS 102h
not_mem_enable          equ     0fdh    ;complement of mem_enable_mask

max_pos_bytes           equ     32      ;number of POS bytes in pos array
pos_array_size          equ     34      ;number of total bytes in pos array

;ADP functions
configure               equ     1       ;configure an adapter
deallocate              equ     3       ;deallocate resources for an adapter
deallocate_all          equ     4       ;deallocate resources for all adapters
deduce                  equ     5       ;deduce the adapter configuration
```

```
;ADP function calls - resource query
request_mem_block       equ     0       ;request a block of nonsystem memory
check_mem_block         equ     1       ;check memory block allocated to adapter
deallocate_mem_block    equ     2       ;de-allocate a block of nonsystem memory

;ADP parameter block
adp_pb          struc
function        db      ?               ;ADP function
slot            db      ?               ;slot being configured
top_mem_os      dw      ?               ;pointer to top of system memory below
top_mem_seg     dw      ?               ;       16M
top_pmem_os     dw      ?               ;pointer to top of planar system memory
top_pmem_seg    dw      ?               ;       below 16M
pos_array_os    dw      ?               ;pointer to POS array
pos_array_seg   dw      ?
pos_mask_os     dw      ?               ;pointer to POS mask
pos_mask_seg    dw      ?
id_list_os      dw      ?               ;pointer to list of adapter IDs
id_list_seg     dw      ?
res_query_os    dw      ?               ;resource query entry point
res_query_seg   dw      ?
adp_pb_os       dw      ?               ;pointer to ADP function call parameter
adp_pb_seg      dw      ?               ;       block
top_mem32_os    dw      ?               ;pointer to top of system memory from
top_mem32_seg   dw      ?               ;       16M to below 4G
top_pmem32_os   dw      ?               ;pointer to top of planar system memory
top_pmem32_seg  dw      ?               ;       from 16M to below 4G
adp_pb          ends

;Resource Query parameter block
res_query       struc
rqfunction      db      ?               ;ADP function call
rqslot          db      ?               ;slot to perform function on
start_addr_high dw      ?               ;high word of starting address
start_addr_low  dw      ?               ;low word of starting address
end_addr_high   dw      ?               ;high word of ending address
end_addr_low    dw      ?               ;low word of ending address
size_high       dw      ?               ;high word of memory block size
size_low        dw      ?               ;low word of memory block size
res_query       ends
```

```
;--------------------------------------------------------------------;
;  Begin program:
;--------------------------------------------------------------------;

code    segment para    public 'code'
start   proc    far
        assume  cs:code,ds:code,ss:code
        org     100h                    ;program starts at offset 100h

        push    bp                      ;save base pointer on stack
        mov     bp,sp                   ;initialize base pointer to stack pointer
        push    es                      ;save registers
        push    ds
        push    si
        push    di
        jmp     prog                    ;jump around data area

;--------------------------------------------------------------------;
;  Data area:
;--------------------------------------------------------------------;

data_seg        dw      ?               ;holds value of DS for Setup program


prog:   mov     cs:data_seg,ds          ;save value of DS from Setup program
        mov     ax,[bp+8]               ;get segment of ADP parameter block
        mov     ds,ax                   ;put ADP parameter block segment in DS
        mov     di,[bp+6]               ;put offset of ADP parameter block in DI

;--------------------------------------------------------------------;
;  Determine what ADP function is to be performed.
;--------------------------------------------------------------------;

        mov     al,[di].function        ;get ADP function
        cmp     al,configure            ;is function configure?
        je      do_configure            ;yes, branch
        cmp     al,deallocate           ;is function deallocate?
        je      do_deallocate           ;yes, branch
        cmp     al,deallocate_all       ;is function deallocate all?
        je      do_deallocate_all       ;yes, branch
        cmp     al,deduce               ;is function deduce?
        jne     bad_command
        jmp     do_deduce               ;yes, branch
bad_command:
        mov     al,0ffh                 ;function not supported return code
        jmp     exit                    ;exit the ADP
```

```
;-------------------------------------------------------------------;
;  Configure the adapter.
;-------------------------------------------------------------------;

do_configure:
        call    get_rq_pb               ;point ES:SI to resource query param blk
        mov     es:[si].rqfunction,request_mem_block    ;function to perform
        mov     al,[di].slot            ;get slot of adapter being configured
        mov     es:[si].rqslot,al                       ;slot to configure
        mov     es:[si].start_addr_high,begin_block_h   ;beginning of allowable
        mov     es:[si].start_addr_low,begin_block_l    ;       range
        mov     es:[si].end_addr_high,end_block_h       ;end of allowable range
        mov     es:[si].end_addr_low,end_block_l
        mov     es:[si].size_high,size_h                ;size of block required
        mov     es:[si].size_low,size_l
        call    rqfunction_call         ;do the resource query function call
        cmp     es:[si].rqfunction,0    ;was the function successful?
        je      set_pos_bits            ;yes, set the appropriate POS bits
        jmp     unsuccessful

set_pos_bits:
        mov     bx,es:[si].start_addr_high              ;get assigned address
        mov     al,[di].slot            ;slot value to get pos array pointer for
        call    point_to_pos            ;point ES:SI to pos array
        mov     es:[si+3],bl            ;set POS value for register 103h
        mov     es:[si+4],bh            ;set POS value for register 104h
        jmp     successful

;-------------------------------------------------------------------;
;  De-allocate resources for the adapter.
;-------------------------------------------------------------------;

do_deallocate:
        mov     al,[di].slot            ;get slot of adapter being configured
        call    deallocate_mem          ;call proc to de-allocate the memory
        call    point_to_pos            ;get pointer to pos array for adapter
        and     byte ptr es:[si+2],not_mem_enable       ;clear memory enable bit
        mov     al,0                    ;successful error code
        jmp     exit                    ;exit ADP
```

```
;---------------------------------------------------------------------;
;  De-allocate resources for all adapters.
;---------------------------------------------------------------------;

do_deallocate_all:
        mov     cx,8                    ;max number of adapter slots per system
        mov     ax,[di].id_list_seg     ;get segment of list of adapter IDs
        mov     es,ax
        mov     si,[di].id_list_os      ;get offset of list of adapter IDs
        mov     al,1                    ;first slot to check

deallocate_lup:
        add     si,2                    ;point to next ID in list (skip planar)
        cmp     es:[si],adapter_id      ;does current slot have this adapter?
        jne     no_adapter              ;no, skip dealloction of memory
        call    deallocate_mem          ;call proc to deallocate the memory
        call    point_to_pos            ;get pointer to pos array for adapter
        and     byte ptr es:[si+2],not_mem_enable       ;clear memory enable bit

no_adapter:
        inc     al                      ;next slot to check
        loop    deallocate_lup

        mov     al,0                    ;successful return code
        jmp     exit                    ;exit ADP

;---------------------------------------------------------------------;
;  Deduce current configuration of adapter.
;---------------------------------------------------------------------;

do_deduce:
        mov     al,[di].slot
        call    point_to_pos            ;point ES:SI to pos array
        mov     al,es:[si+2]            ;read value of POS register 102h
        and     al,mem_enable_mask      ;see if the memory is enabled
        jz      unsuccessful            ;if mem enable bit not set, branch

        mov     bl,es:[si+3]            ;read value of POS register 103h
        mov     bh,es:[si+4]            ;read value of POS register 104h
        call    get_rq_pb               ;point ES:SI to resource query param blk
        mov     es:[si].rqfunction,request_mem_block    ;function to perform
        mov     al,[di].slot            ;get slot of adapter being deduced
        mov     es:[si].rqslot,al                       ;slot to deduce
        mov     es:[si].start_addr_high,bx              ;beginning of allocated
        mov     es:[si].start_addr_low,begin_block_l    ;       range
        add     bx,size_h - 1           ;determine end value for this request
        mov     es:[si].end_addr_high,bx                ;end of allocated range
        mov     es:[si].end_addr_low,end_block_l
        mov     es:[si].size_high,size_h                ;size of block required
        mov     es:[si].size_low,size_l
        call    rqfunction_call         ;do the resource query function call
        cmp     es:[si].rqfunction,0    ;was the function successful?
        je      successful              ;yes, branch
        jmp     unsuccessful
```

```
;-------------------------------------------------------------------;
;  ADP function was successful.
;-------------------------------------------------------------------;

successful:
        mov     al,[di].slot
        call    point_to_pos            ;get pointer to pos array
        or      byte ptr es:[si+2],mem_enable_mask      ;set memory enable bit
        mov     al,0                    ;successful return code for ADP function
        jmp     exit                    ;exit the ADP

;-------------------------------------------------------------------;
;  ADP function was unsuccessful.
;-------------------------------------------------------------------;

unsuccessful:
        mov     al,[di].slot
        call    point_to_pos            ;get pointer to pos array
        and     byte ptr es:[si+2],not_mem_enable       ;clear memory enable bit
        mov     al,1                    ;unsuccessful return code for ADP function

;-------------------------------------------------------------------;
;  Exit the ADP.
;-------------------------------------------------------------------;

exit:   mov     [di].function,al        ;save return code of ADP function
        call    set_pos_mask            ;initialize the pos mask bits
        pop     di                      ;restore registers
        pop     si
        pop     ds
        pop     es
        pop     bp
        ret
start   endp
```

```
;--------------------------------------------------------------------;
;  DEALLOCATE_MEM
;       Perform deallocate ADP function on the slot defined in the
;       AL register.
;--------------------------------------------------------------------;

deallocate_mem  proc    near
        push    ax
        push    es
        push    cx
        push    si
        mov     bl,al                   ;save slot number in BL register

del_begin:
        call    get_rq_pb               ;point ES:SI to resource query param blk
        mov     es:[si].rqfunction,check_mem_block      ;function to perform
        mov     es:[si].rqslot,bl       ;slot to check
        mov     es:[si].end_addr_high,0 ;always check first block in list
        call    rqfunction_call         ;do the resource query function call
        cmp     es:[si].rqfunction,0    ;is there a memory block assigned?
        jne     del_done                ;no, done de-allocating memory

        call    get_rq_pb               ;point ES:SI to resource query param blk
        mov     es:[si].rqfunction,deallocate_mem_block ;function to perform
        call    rqfunction_call         ;do the resource query function call
        jmp     del_begin

del_done:
        pop     si
        pop     cx
        pop     es
        pop     ax
        ret
deallocate_mem  endp

;--------------------------------------------------------------------;
;  GET_RQ_PB
;       Initializes ES:SI to point to the ADP function call
;       (resource query) parameter block.
;--------------------------------------------------------------------;

get_rq_pb       proc    near
        push    ax
        mov     ax,[di].adp_pb_seg      ;put the segment of the ADP function
        mov     es,ax                   ;       call parameter block in ES
        mov     si,[di].adp_pb_os       ;put the offset in SI
        pop     ax
        ret
get_rq_pb       endp
```

```
;--------------------------------------------------------------------;
;  RQFUNCTION_CALL
;       Makes an ADP function call.
;--------------------------------------------------------------------;

rqfunction_call proc    near
        push    ds                      ;save registers
        push    es
        push    si
        push    di
        push    bx
        push    cx

        mov     bx,di                   ;set BX to start of ADP parameter block
        add     bx,res_query_os         ;ES:BX points to ADP function call entry
        mov     ax,cs:data_seg          ;restore the Setup data segment
        mov     ds,ax
        call    dword ptr es:[bx]       ;make the ADP function call

        pop     cx                      ;restore the registers
        pop     bx
        pop     di
        pop     si
        pop     es
        pop     ds
        ret
rqfunction_call endp

;--------------------------------------------------------------------;
;  POINT_TO_POS
;       Initializes ES:SI to point to the pos array for the slot
;       number defined in the AL register.
;--------------------------------------------------------------------;

point_to_pos    proc    near
        push    ax
        push    bx

        mov     bl,al                   ;slot number to get pos array pointer for
        mov     ax,[di].pos_array_seg   ;put the pointer to the beginning of the
        mov     es,ax                   ;       pos array into ES
        mov     si,[di].pos_array_os    ;put pointer to beginning in SI
        mov     ax,pos_array_size       ;get the size of each adapter pos array
        mul     bl                      ;multiply the size by the slot number
        add     si,ax                   ;set SI to the start of the pos array
                                        ;       for the specified slot
        pop     bx
        pop     ax
        ret
point_to_pos    endp
```

```
;--------------------------------------------------------------------;
;  SET_POS_MASK
;       Initializes the bits in the POS mask.
;--------------------------------------------------------------------;

set_pos_mask    proc    near
        mov     ax,[di].pos_mask_seg    ;put the segment of the pos mask into
        mov     es,ax                   ;       ES
        mov     si,[di].pos_mask_os     ;put segment of pos mask in SI
        mov     cx,max_pos_bytes        ;set CX to the number of POS bytes
        push    si
lup2:   mov     byte ptr es:[si],0      ;set all pos mask values to 0
        inc     si
        loop    lup2
        pop     si
        mov     byte ptr es:[si],mem_enable_mask        ;set mem enable bit for
                                                        ;   POS register 102h
        mov     byte ptr es:[si+1],0ffh ;set all bits for POS register 103h
        mov     byte ptr es:[si+2],0ffh ;set all bits for POS register 104h
        ret
set_pos_mask    endp

code    ends
        end     start
```

Initialization Programs

An initialization program is a memory image program that is loaded onto track 0, head 0 of drive hex 80. The master boot record executes the initialization programs before booting the operating system from the fixed disk (the initialization program is bypassed if the operating system is booted from the diskette drive). It is an alternate method of initializing an adapter that does not require the use of ROM or jumpers and switches on the adapter.

The Set Configuration program, selected from the Main Menu of the Reference Diskette, installs the initialization programs onto drive C. (Automatic Configuration does not install these programs.) The Set Configuration program determines that an adapter has an initialization program by checking the adapter ADF. An adapter with a corresponding initialization program has the statement, InitProg <attribute>, in its ADF. The InitProg statement is placed after the FixedResources statement and before the NamedItem statement.

The attribute is a value from 0 to 127 that selects the order in which the initialization programs are run; the lowest value is run first. Attributes from 30 to 80 are available; attributes outside of this range are reserved. An example of the use of this attribute is to ensure that a specific adapter is initialized before another if both are installed in the same system.

The following rules apply to all initialization programs:

- The initialization program is always loaded beginning at offset hex 7C00. Any data variables referenced in the code may need their offsets adjusted to hex 7C00.
- The initialization program must begin with a 20 byte header. Execution of the initialization program begins on the 21st byte.
- The 16th byte of the program header is the revision level of the initialization program. The value in this byte should be incremented if a new revision of the program is released. When Set Configuration is run, the installation process checks this byte to ensure that the program with the latest revision level is installed. The other bytes in the header are set by the Set Configuration program.
- The initialization program cannot be more than 1024 bytes in size, including the 20 byte header.
- While executing, the initialization program should not assume that memory is available beyond its own programming size.
- The initialization program is shipped as a DOS file so that it can be merged onto the Reference Diskette in the same way as the ADFs. The name of the file is based on the ID of the adapter, and is in the form ICardID.ADF (CardID is the adapter ID in hex).
- When an initialization program finishes executing, it performs a Near Return to pass control back to the calling program. The stack will then be in the same state it was when the initialization program was called.

A revision to an initialization program is installed the same way as the original is installed. The revision to the program has a higher value in the revision level byte than previous versions. The Set Configuration program compares the revision levels of initialization programs installed on the fixed disk drive with those on the Reference Diskette. If the program on the Reference Diskette has a greater revision level than the one on the fixed disk drive, it updates the program on the fixed disk drive.

Index

G

H

I

L

M

N

O

P

Q

R

S

V

Notes: