draft proposed X3T10/792D
American National Standard

Revision 12b
20-MAY-95

Information technology - SCSI-2 Common access method transport and SCSI interface module

This is a draft proposed American National Standard of Accredited Standards Committee X3. As such this is not a completed standard. The X3T10 Technical Committee may modify this document as a result of comments received during public review and its approval as a standard.

ASC X3T10 Technical Editor:

William D. Dallas
Digital Equipment Corporation
110 Spit Brook Road
Nashua N.H. 03060
USA

Telephone: 603-881-2508
Facsimile: 603-881-2257
Email:
dallas@wasted.enet.dec.com

Reference number
ISO/IEC **** : 199x
ANSI X3.232 - 199x

Contents

Tables

Figures

Annexes

Foreword

The purpose of this standard is to define a software architecture for software peripheral device drivers, the transport, and the software interface modules to the host hardware (e.g., host bus adaptor), for control of SCSI peripheral devices in a common environment. Peripheral device drivers and software interface modules developed to this standard may easily be ported between different operating systems that have implemented this standard.

As with any other technical document, there may arise questions of interpretation of this standard. The X3 Committee has established procedures to issue technical opinions concerning the standards developed by the X3 organization. These procedures may result in a SCSI Technical Information Bulletins (TIBs) being published by X3.

These Technical Information Bulletins, while reflecting the opinion of the Technical Committee which developed the standard, are intended solely as supplementary information to other users of the standard. This standard ANSI X3.xxx-199x, as approved through the publication and voting procedures of the American National Standards Institute, is not altered by these bulletins. Any subsequent revision to this standard may or may not reflect the contents of these Technical Information Bulletins.

Introduction

SCSI provides a diverse range of peripherals for attachment to a wide range of computing equipment. Some system manufacturers have developed approaches for SCSI attachment which are widely followed, increasing the applications available for the attachment of SCSI peripherals. In markets where no standard method of attachment exists, however, variations between third party sellers have made it nearly impossible for end users to attach more than one SCSI peripheral to one host bus adapter.

In an effort to broaden the application base for SCSI peripherals, an ad hoc industry group of companies representing system integrators, controllers, peripherals, and semiconductors decided to address the issues involved. That effort has evolved into this standard.

Information processing systems -- SCSI-2 common access method transport and SCSI interface module

1 Scope

This standard defines the Common Access Method (CAM) for the Small Computer Systems Interface (SCSI).

The purpose of this standard is to define a method whereby multiple environments may adopt a common procedure for the support of SCSI devices.

The CAM provides a structured method for supporting peripherals with the software (e.g., device driver) and hardware (e.g., host bus adapter) associated with any computer.

2 Normative references

ANSI X3.131-1994, *Small Computer Systems Interface - 2*

3 Definitions and conventions

3.1 Definitions

For the purposes of this standard the following definitions apply

3.1.1 **block**: This defines an action to prevent access, (e.g., to obstruct the action of or the continuation of a process thread).

3.1.2 **CCB (CAM control block)**: The data structure provided by peripheral drivers to the XPT to control execution of a function by the SIM.

3.1.3 **Immediate CCB**: Provides valid completion status when the call to xpt_action () returns (e.g., path inquiry).

3.1.4 **Queued CCB**: Provides valid completion status when the completion callback routine is called, or the CAM Status field in the CCB changes from Request In Progress to another valid CAM Status.

3.1.5 **CDB (command descriptor block)**: A data structure containing the SCSI opcode, parameters, and control bits for that operation.

3.1.6 **DMA (direct memory access)**: A means of data transfer between peripheral and host memory without processor intervention.

3.1.7 **freeze**: This defines a software action to quiesce activity (e.g., freeze the queue).

3.1.8 **HBA (host bus adapter)**: The hardware and microcode which provides the interface between system memory and any number of SCSI buses.

3.1.9 **null**: A value which indicates that the contents of a field have no meaning. This value is typically, though not necessarily, zero.

3.1.10 **optional**: This term describes features which are not required by this standard. However, if any feature defined by this standard is implemented, it shall be done in the same way as defined by the standard. Describing a feature as optional in the text is done to assist the reader. If there is a conflict between text and tables on a feature described as optional, the table shall be accepted as being correct.

3.1.11 OSD (Operating System Dependent): This term describes a capability, method of operation, or feature that depends on the specific operating system on which CAM is implemented.

3.1.12 path: This term describes the address of the XPT or a SIM/HBA SCSI bus combination.

3.1.13 reserved: Where this term is used for bits, bytes, fields, and code values; the bits, bytes, fields, and code values are set aside for future standardization. The default value shall be zero. The originator is required to define a reserved field or bit as zero, but the receiver should not check reserved fields or bits for zero.

3.1.14 SIM (SCSI interface module): A module designed to accept the CAM control blocks routed through the XPT in order to execute SCSI commands and perform other functions.

3.1.15 VU (vendor unique): This term is used to describe bits, bytes, fields, code values, and features which are not described in this standard, and may be used in a way that varies among vendors.

3.1.16 XPT (transport): A layer of software which peripheral drivers use to originate the execution of CAM functions.

3.2 Conventions

Within the tables, there is a direction bit which indicates in or out. The direction is from the view point of the peripheral driver (i.e., information is out to the SIM from the peripheral driver and in to the peripheral driver from the SIM).

Certain terms used herein are the proper names of signals. These are printed in uppercase to avoid possible confusion with other uses of the same words (e.g., ATTENTION. Any lower-case uses of these words have the normal American-English meaning).

There are places in this standard where programming language is used to define or express a concept in order to assist the reader. These are not copyrighted program steps and implementors are encouraged to use the code wherever it suits their application.

4 Conformance

An implementation claiming conformance to the transport layer (XPT) for a specified operating system and language environment shall:

- provide all the mandatory XPT functions and services specified in this standard.
- correctly interoperate with any conforming SCSI Interface Module (SIM) for the specified environment.
- provide the necessary interface specifications that a conforming SIM requires to interface with the XPT.

An implementation claiming conformance to the SIM for a specified operating system and language environment shall:

- provide all the mandatory SIM functions and services specified in this standard.
- correctly interoperate with any conforming XPT for the specified environment.
- provide the necessary interface specifications that a conforming XPT requires to interface with SIMs.

A conforming implementation shall execute all function codes as required by this standard, and in response to these codes shall only return specified status, and return codes. A conforming implementation may provide additional capabilities via Vendor Unique function codes.

If an operating system is not specified in this standard, then that operating system shall conform to clause 8.1 in this standard. (See also annex C.)

Claims of conformance to this standard shall state:

- whether conformance is claimed with the XPT or the SIM or both.
- which operating systems and environments are supported.
- whether the optional capabilities of target mode or Host Bus Adaptor (HBA) engines are supported.

5 General description

The application environment for CAM is any computer addressing a SCSI peripheral through a protocol chip on a motherboard or a host bus adapter.

SCSI is a widely-used interface which provides common attachment for a variety of peripherals.

The purpose of the CAM is to define a standard for the support of Host Bus Adapters (HBA) and the like by device driver software.

Software in the operating system dispatches I/O requests to the SCSI peripherals in a number of different ways depending on the software architecture. The operating system dependencies (OSD) are defined in clause 8 for certain named software and hardware platforms.

5.1 Environment

A model of the CAM usage environment is illustrated in figure 1. Multiple applications are shown accessing a variety of SCSI devices. Several device drivers, both peripheral drivers and SIMs, are present to support the peripherals on the system.

The choice of XPT and SIM packaging is an operating system dependency. Clause 8 defines this dependency for certain named software and hardware platforms.

Requests for SCSI I/O are made through the CAM XPT interface. The XPT may execute them directly or pass them to a SIM for execution.

The XPT function is illustrated as a separate element. In some operating systems, the XPT operations is incorporated into a single module which integrates both XPT and SIM functionality. The logical separation between the two is maintained because there may be more than one SIM loaded. The XPT function may be provided by the operating system, or can be achieved through associating multiple separately loaded XPTs

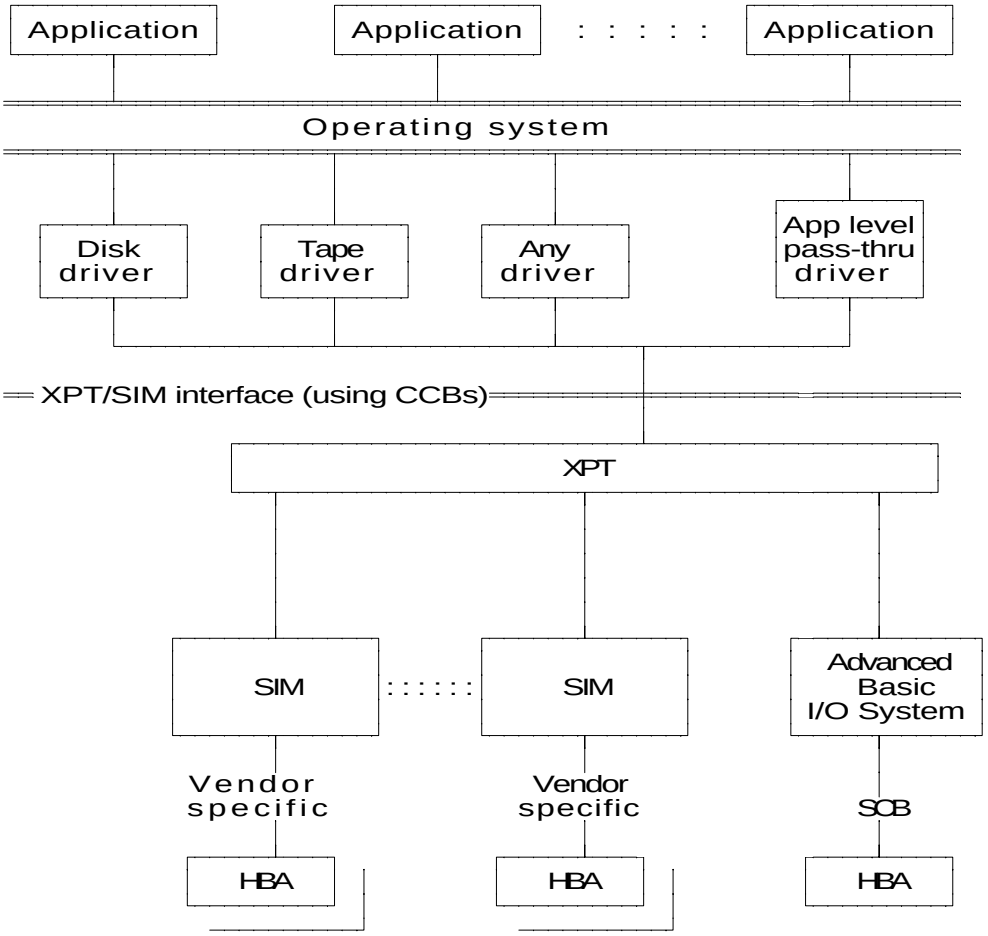

Figure 1 - CAM environment model

5.2 Peripheral driver functions

Peripheral drivers provide the following functionality:

a) Interpreting of application or system level requests;
b) Mapping of application level requests to XPT/SIM control blocks;
c) Requesting of resources to initiate a CAM request:
   1) CAM control blocks and supporting blocks that may be needed,
   2) Buffer requirements;
d) Handling of exception conditions not managed transparently by SCSI (e.g., check condition status, unexpected bus free, resets, etc.);
e) Logging of exception conditions for maintenance analysis programs;
f) Format utility or services required by format utilities;
g) Establishing parameters for HBA operation;
h) Set up routing of SCSI requests to the correct path/bus, target, and LUN;
i) Initialization and configuration functions of a target not handled by a utility at installation and formatting time;
j) Establishing a timeout value for a task and passing this value in the CCB.

5.3 XPT functions

XPT services provide the following functionality to process CCBs:

a) Routing of the CCB to the proper SIM;
b) OSD management of CCB resources (e.g., get_CCB, free_CCB);
c) Maintenance of the SCSI device table. (This consists of owning the table and servicing requests to read and write the table.);
d) Providing properly formatted control blocks and priming the fields needed to accomplish a request;
e) Routing of asynchronous events back to peripheral drivers;

5.4 SIM functions

SIM services provide the following functionality to process CCBs:

a) Perform all interface functions to the SCSI HBA;
b) Manage or delegate (as appropriate) all the SCSI HBA protocol steps;
c) Distinguish abnormal behavior and perform error recovery, as required;
d) Manage data transfer path hardware, including DMA circuitry and address mapping, and establish DMA resource requests (if necessary);
e) Queuing of multiple operations for different LUNs as well as the same LUN and assign tags for tag queuing (if supported);
f) Freeze and unfreeze the queue as necessary to accomplish queue recovery;
g) Post the completed operation to the initiating device driver;
h) Manage selection, disconnection, reconnection, and data pointers of the SCSI HBA protocol;
i) Implement a timer mechanism, using values provided by the peripheral driver;

6 Background

SCSI is a peripheral interface designed to permit a wide variety of devices to coexist. These peripherals are typically, but not necessarily, attached to the host by a single SCSI HBA.

6.1 Software

OS (operating system) support for peripheral devices is normally achieved through peripheral drivers or utility programs. No single driver or program can reasonably support all possible SCSI peripherals, so separate drivers are needed for each class of installed SCSI device. These drivers need to be able to share the SCSI HBA hardware. These drivers also have to work with a broad range of HBA hardware, from highly intelligent coprocessors to the most primitive, including a SCSI chip on a motherboard. A standard SCSI programming interface layer is essential to insulate SCSI peripheral drivers and utilities from the HBA hardware implementation, and to allow multiple drivers to share a single SCSI hardware interface.

6.2 CAM (Common Access Method)

This standard describes the general definition of the CAM (Common Access Method). CAM functionality has been separated into a few major elements.
- XPT (Transport)

The XPT (transport) defines a protocol for SCSI peripheral drivers and programs to submit I/O requests to the HBA specific SIM module(s). Routing of requests to the correct HBA and posting the results of a request back to the driver are responsibilities of the transport.

- SIM (SCSI Interface Module)
  The SIM (SCSI interface module) manages HBA resources and provides a hardware-independent interface for SCSI applications and drivers. The SIM is responsible to process and execute SCSI requests, and manage the interface to the HBA hardware.

  There are no requirements on how the SIM is implemented, in RAM (random access memory) or ROM (read only memory), provided the XPT is properly supported. A ROM-based SIM may need a transparent (to the user) software layer to match the SIM-required services to the specific manner in which they are requested of the OS.

- CCB (CAM Control Block)
  The CAM control block is a data structure passed from the peripheral driver to the XPT. The contents of the data structure describe the action required and provides the fields necessary for successful processing of a request.

6.3 OSD (Operating System Dependencies)

The system environment in which the CAM is operating is a function of the hardware platform and the operating system being executed. The byte ordering is different between an Intel-based and a Motorola-based machine for example, and the calling structure differs greatly between operating systems.

Although the fields of a CCB have a common meaning, the contents may vary by platform and OS. These dependencies cause differences in operation and implementation, but do not prevent interoperation on the same platform of two CAM modules implemented by different manufacturers.

Some OSD issues are discussed in Clause 8.

6.4 Architectural considerations

Ideally, a single XPT structure would suffice for all OS environments, but this is impractical in light of the wide architectural differences between the various processor architectures.

Programming effort has been minimized by making the interfaces as similar as possible across OS platforms, and customizing the SIM for each HBA to maximize performance under each OS. HBAs vary widely in the capability and functions they provide so there may be an internal (transparent) interface to isolate hardware interface routines from routines which make use of OS resources.

In order to prevent each peripheral driver from having to scan the SCSI bus for devices at initialization, the XPT ascertains all installed SCSI devices and constructs an internal table. A XPT function is used by drivers and programs to access this table.

Peripheral drivers need to be developed with documentation provided by the operating system vendor in addition to that supplied by this standard.

Under a UNIVOS (Universal Operating System), the XPT and SIMs would typically be compiled with the kernel at system generation time, so that entry points would be resolved during linkage-editing. Third party attachments may be supported without the need for a sysgen if suitable routing facilities are provided by the system vendor. One

example of a universal operating system is UNIX (a registered trademark of X/Open Ltd.) and as described in this standard, software developed for a UNIVOS could be compatible within a Unix system.

Under a LANOS (Local Area Networking Operating System), the XPT can be loaded as a loadable device driver. SIMs are also implemented as loadable device drivers. One example of a local area network operating system is Netware 386 (a Trademark of Novell) and as described in this standard, software developed for a LANOS could be compatible within a Netware 386 system.

Under DOS, there is one logical XPT with one entry point, but it may consist of a number of separate XPT/SIM modules (perhaps supplied for each HBA in the system).

XPT routing is a mechanism to support concurrent multiple co-resident SIMs so that different HBAs can be present in the same system. This task is handled by the XPT logical entity. The XPT is implemented differently under each operating system, but the logical functionality is the same for all operating systems.

Once one or more SIMs are loaded, the peripheral drivers integrate each type of SCSI device into the OS through the XPT, independent of the installed HBA hardware.

This is a list of requirements that CAM is designed to meet:

a) The ability to use any SCSI command (including vendor unique);
b) No restrictions on the size or format of transferred data;
c) Allow all the capabilities of high end host adapters to be fully utilized and accommodating HBAs which do most of the SCSI processing on board (this precludes interfaces which expect to control SCSI phases);
d) Allow the calling peripheral driver or program to interpret sense data returned by SCSI devices;
e) Fully re-entrant code;

NOTE: This is an obvious requirement for multi-tasking environments such as UNIVOS but even in single tasking DOS applications, multi-threaded I/O is required to achieve maximum performance. SCSI devices such as printers, communication ports, and LAN interfaces are often serviced in the background under DOS. If an HBA cannot support multi-threading, requests can be queued, and serialized within the SIM module transparently to the XPT.

f) Support of multiple HBAs;
g) Peripheral drivers may ascertain which optional features are available;
h) Define a single XPT-based SCSI Bus scanning algorithm (so that each peripheral driver need not use bus bandwidth to perform this function;
i) Provide a mechanism to abort I/O requests (at request of peripheral driver);
j) Ability to issue multiple I/O requests from one or more peripheral drivers to a single target/LUN;
k) Providing peripheral drivers with a mechanism for allocating a sense data area and for specifying the number of sense bytes to be automatically requested on a CHECK CONDITION.

7 Principles of operation

7.1 Accessing the XPT

The OS peripheral drivers access the XPT through a software call to a single entry point. The method for obtaining and using the entry point differs between operating systems.

The XPT is responsible for routing a CCB to the SIM indicated by the Path ID field.

The XPT is not involved in the reverse process of advising the peripheral driver of the completion of a queued request. The completion callback permits a direct return from the SIM to the peripheral driver (the exact method employed in callback is operating system dependent). Completion notification for immediate requests is accomplished by the normal function call return mechanism.

The XPT is responsible for notifying peripheral drivers of asynchronous events via the asynchronous callback mechanism.

7.2 Initialization

The XPT is responsible for determining the SCSI Bus configuration at initialization for each SIM.

The scan by the XPT/SIM would follow a pattern such as the following:

```
for all SCSI buses
    for all SCSI IDs (excluding the HBA SCSI ID)
        find the device
        if device exists
            for all LUN's
                send an INQUIRY command and save returned information
            end for
        end if
    end for
end for
```

7.3 CCB completion

CCB completion is either immediate or queued.

7.3.1 Completion of immediate CCB

All CCBs are complete when the xpt_action() call returns except:

- Execute SCSI I/O
- Execute Target I/O
- Accept Target I/O
- Continue Target I/O
- Immediate Notify
- Execute Engine Request

7.3.2 Completion of queued CCB

The XPT/SIM shall set all appropriate queued CCB fields marked as IN or IN/OUT and shall provide autosense information as specified in this standard before CCB completion notification is done. See Clause 7.7 (autosense) and the appropriate CCB tables in Clauses 9, 10, 11, and 12 for further information.

A peripheral driver is notified of the completion of a queued CCB by using one of the following mechanisms, as directed by the CAM Flags field in the CCB:

- CCB Callback on completion.
  The SIM/XPT calls the peripheral driver's callback routine when execution of the queued CCB completes. The queued CCB includes a pointer to the peripheral driver's callback routine (in the Callback on Completion field).

  The peripheral driver callback routine is used much like a hardware interrupt handler. The callback routine has the same privileges and restrictions as an interrupt handler.

  The address of the specific CCB that completed is passed to the peripheral driver's callback routine.

- CCB CAM Status field change from Request in Progress to another CAM Status.
  If the CCB CAM Flags has the Disable Callback of Completion set the XPT/SIM shall notify the peripheral driver of a completed CCB by changing the CAM Status field of the CCB from Request in Progress to another valid CAM Status. The peripheral driver shall be responsible for monitoring the CCB CAM Status field for completion.

7.4 SCSI request queues

Queues are used in systems where there is a need to manage multiple outstanding requests. There are various types of queues and each has different support needs.

A SCSI request can be queued in any of the following places:

a) in the SIM
b) in the target/LUN
c) in the peripheral driver

The SIM keeps a queue of all the CCB requests from the various peripheral drivers that access a Logical Unit.

SCSI-2 permits a Logical Unit to keep a queue using tag queues, or a simple queue of one element.

A peripheral driver may also keep a queue (e.g., to perform a simple elevator sort).

7.4.1 The logical unit and the peripheral driver

Based on needs outside the scope of CAM, a peripheral driver may maintain a list of unfinished queued CCBs. The SIM, acting on behalf of the peripheral driver, sends the appropriate commands or messages to manage the queues in the Logical Unit. When the Logical Unit completes an operation, the peripheral driver is advised by the SIM via a callback or by checking CAM Status for completion (see Clause 7.3 for additional information).

The peripheral driver should be coded as if there are other peripheral drivers and other initiators working with the same Logical Unit.

7.4.2 SIM queuing

The SIM maintains a request queue for each Logical Unit. The request queue is logically shared by all peripheral drivers. The request queue may support tagged commands. Queue priority shall be supported.

7.4.2.1 SIM queue priority

When a CCB has a SIM queue priority=1, the SIM places the CCB ahead of all CCBs that have a SIM queue priority=0 for the Logical Unit and at the end (tail) of all other CCBs having a SIM queue priority=1. One use of this CAM Flag is during error handling. If the queue is frozen and a CCB with SIM queue priority=1 is received, the CCB shall be placed at the head of the queue of CCBs that have SIM priority=0 and the queue remains frozen. When the SIM queue is released, any CCBs with SIM queue priority=1 are executed individually first (in FIFO sequence), followed by CCBs with a SIM priority=0 (in FIFO sequence). See Clause 7.4.2.3 on how to release the SIM queue.

To force step-by-step execution, the peripheral driver can set SIM queue freeze=1, so that when the queue is released and a CCB with SIM queue priority=1 is executed, the queue is re-frozen by the SIM at completion.

7.4.2.2 Tag recognition

To support tagged queuing recognition the SIM maintains a reference between the CCB pointers and the queue tags for a Logical Unit. By this means, the SIM can handle both the queue tag resource allocation and reconnection of the I_T_L_Q nexus (see SCSI-2 for further information on I_T_L_Q nexus) for the CCB from a peripheral driver.

The peripheral driver shall allow the SIM/XPT to handle the assignment of the queue tag ID for the request. The SIM shall assign unique TAG Ids for the Logical Unit operation based on its internal reference table.

When a Logical Unit that supports tagged queuing reconnects to the initiator (SIM/HBA pair), it sends the SIMPLE QUEUE TAG message with the queue tag value for the I_T_L_Q nexus. Using the returned queue tag ID, the SIM restores what is necessary to continue the SCSI transaction. The queue tag ID shall be freed by the SIM at the completion of the SCSI request.

7.4.2.3 Error conditions and queues within the subsystem

The SIM shall place its internal queue for a Logical Unit into the frozen state for any status other than Request Completed without Error or Request in Progress for an EXECUTE SCSI I/O REQUEST CCB, unless the SIM queue freeze disable bit has been set in the CCB. If the SIM queue freeze disable bit is set, the queue shall not be frozen and CCB execution continues from the SIM queue.

Note: Immediate CCB function types, Target Mode CCBs, and EXECUTE ENGINE REQUEST CCBs never result in a frozen SIM queue. The action of some immediate CCBs (e.g., Abort SCSI Command, Terminate I/O Process Request) may result in an EXECUTE SCSI I/O REQUEST CCB being returned which does freeze the SIM queue.

The SIM shall maintain a count of the number of CCBs returned with the indication of SIM queue frozen. The SIM shall decrement the SIM queue frozen count for each CCB received with a Release SIM Queue function code with the SIM Queue Freeze bit set to a one. The SIM/HBA shall not release its internal queue until the SIM queue frozen count is zero. The SIM shall not allow the SIM queue frozen count to have a negative value. The SIM shall not consider it an error when a CCB is received with a Release SIM Queue function code that would decrement the SIM queue frozen count past zero (negative).

In the event that a SIM encounters an error condition which can not be associated with a CCB the SIM shall not freeze the queue. The SIM should attempt to continue operation. Failing that the SIM shall take corrective action leaving the SCSI bus in a usable state.

When a SIM's Logical Unit's queue is in the frozen state (a positive value in the SIM queue frozen count), the SIM shall not dispatch any CCBs to that Logical Unit except to retrieve autosense information. Peripheral drivers can still send CCBs to the SIM for the Logical Unit, or any other Logical Unit. Any new CCBs received by the SIM shall be placed in the queue according to the rules specified in Clause 7.4.2.1

Note: Since the SIM is the initiator, the SIM's internal queue goes into a frozen state so that the pending sense information in the Logical Unit is not discarded. The SIM holds it's internal Logical Unit queue in the frozen state until RELEASE SIM QUEUE CCB(s) are received that decrement the SIM queue frozen count to zero.

Note: The SIM queue frozen count can be greater then one for a number of conditions. An example of two of these conditions are SCSI BUS RESET, and a target operating in tagged queue mode.

Using the CCB, the SIM returns in the CAM Status field an indication of the frozen queue condition and other information. The peripheral driver acts upon the information returned via the CCB. The setting of the Autosense bit in the CAM Flags field does not affect how the SIM handles freezing the SIM's internal queue (i.e., the REQUEST SENSE command issued by the SIM to recover the sense data does not release the SIM queue). See Clause 7.7 for further information on autosense.

7.5 SIM handling of SCSI resets

CAM defines the SIM/HBA mandatory handling of the SCSI hard reset alternative and SCSI BUS DEVICE RESET messages. The optional handling of the SCSI soft reset alternative for SCSI bus resets is SIM/HBA vendor unique. CAM defines some operational functionality that shall be adhered to if the SIM/HBA supports the SCSI bus soft reset alternative.

The SIM/HBA handling of the SCSI hard bus reset shall be as follows:

- Upon detecting a SCSI bus reset, the SIM/HBA shall perform the following actions in sequence:
  a) block Path ID to the reset bus, all new received CCBs for this bus are rejected with status of CAM busy;
  b) return all CCBs within the SIM/HBA for this bus with a CAM status of SCSI Bus Reset Sent/Received;
  c) call: xpt_async(opcode=reset,
     path_id=bus that was reset,
     target_id=-1,
     lun=-1,
     buffer_ptr=null,
     data_cnt=0)
  d) unblock Path ID for the bus;
  e) resume normal processing of CCBs;

For the operational functionality for SCSI soft reset alternative, the SIM/HBA shall implement in a vendor unique manner as follows:

- The SIM/HBA shall check whether all Logical Units for a SCSI bus it controls implement the SCSI soft reset alternative.
- If all Logical Units support the SCSI soft reset alternative for a SCSI bus, the SIM/HBA shall indicate its support for the soft reset alternative by setting the Soft Reset flag in the PATH INQUIRY CCB.
- Upon detecting a SCSI bus reset for a SIM/HBA that has indicated support of the SCSI soft reset alternative in the PATH INQUIRY CCB, the SIM/HBA shall perform the following actions in sequence:

a) Handle the SCSI soft reset alternative as defined in ANSI X3.131-1994;
b) call: xpt_async(opcode=reset,
path_id=bus that was reset,
target_id=-1,
lun=-1,
buffer_ptr=null,
data_cnt=0)
c) resume normal processing of CCBs;

- If not all Logical Units for a SCSI bus implement the SCSI soft reset alternative, the SIM/HBA shall ensure in a vendor unique manner that all Logical Units operate in consistent manner with either the SCSI soft reset alternate or the hard reset alternative. The SIM/HBA shall also indicate the alternative chosen by the PATH INQUIRY CCB Soft Reset flag setting.

The SIM/HBA handling of the SCSI BUS DEVICE RESET message shall be as follows:

- Upon successfully issuing a SCSI BUS DEVICE RESET message to a target, the SIM/HBA shall perform the following actions in sequence:
  a) block Target Id for the bus, all new received CCBs for this target are rejected with status of CAM busy;
  b) return all CCBs within the SIM/HBA for this target with CAM status of Bus Device Reset sent;
  c) call: xpt_async(opcode=reset,
  path_id=bus that issued the SCSI BUS DEVICE RESET message,
  target_id= target id of device that received the SCSI BUS DEVICE RESET message,
  lun=-1,
  buffer_ptr=null,
  data_cnt=0)
  d) unblock Target ID for the bus;
  e) resume normal processing of CCBs;

7.6 Asynchronous event callback

In an event such as a SCSI bus reset or an asynchronous event notification the XPT/SIM has to be able to make a callback to the peripheral driver(s), even though there may be no CCBs active for the peripheral driver(s).

Callback routines have the same privileges and restrictions as hardware interrupt service routines. The peripheral driver shall return from the callback.

During system startup or peripheral driver initialization, the peripheral driver should register for an asynchronous event callbacks for each Logical Unit with which it is working and once with the XPT. The Asynchronous Events for Logical Units registration are the following:

- Sent Bus Device Reset to target.
- SCSI AEN.
- Unsolicited reselection.
- SCSI Bus Reset.

The Asynchronous Events for the XPT registration are the following:

- New Logical Units found during a rescan.
- Path ID de-registered.
- Path ID registered.

In order for a peripheral driver to receive asynchronous event callbacks for a Logical Unit, it shall issue a SET ASYNCHRONOUS CALLBACK CCB addressed to the Logical Unit with the Asynchronous Event Enables fields set to a 1 for those events the peripheral driver wishes to be notified of through an asynchronous callback. For the XPT asynchronous event callbacks, the peripheral driver shall issue a SET ASYNCHRONOUS CALLBACK CCB

addressed to the XPT (Path ID 0xFF) with the Asynchronous Event Enables fields set to a one for those events the peripheral driver wishes to be notified of through an asynchronous callback. The SET ASYNCHRONOUS CALLBACK CCB shall apply only to a single Logical Unit or the XPT. The use of wildcards shall not be supported for the SET ASYNCHRONOUS CALLBACK CCB.

The peripheral driver can change its asynchronous events callbacks for a particular SCSI Logical Unit or the XPT by issuing the SET ASYNCHRONOUS CALLBACK CCB to a Logical Unit or the XPT, with the Asynchronous Event Enables field set to the replacement value, an updated Peripheral Driver Buffer Pointer field, an updated Size of Allocated Peripheral Buffer field, and the Asynchronous Callback Pointer field containing the original registered value.

The peripheral driver can de-register for asynchronous events callbacks for a Logical Unit or the XPT by issuing the SET ASYNCHRONOUS CALLBACK CCB to the Logical Unit or the XPT with the Asynchronous Event Enables field set to zero and the Asynchronous Callback Pointer field containing the original registered value. When a peripheral driver wishes to change its asynchronous event callback routine, it shall do so by de-registering and then shall follow the registration procedure.

The XPT shall pass all SET ASYNCHRONOUS CALLBACK CCBs to the addressed (Path ID) SIM in addition to performing XPT table maintenance. This allows the SIM to perform its own asynchronous callbacks to peripheral drivers, foregoing the available (and required) XPT services. The callback interface to the peripheral driver is the same whether the asynchronous callback is from the XPT or directly from the SIM.

Upon detection of a supported enabled event, the SIM shall do the following once for each detected event:

a) Classify the event:
   ascertain the opcode. (See Table 1)
b) Format the associated data within an internal (to the SIM) buffer, (e.g., the sense data received from an AEN).
c) Perform the XPT reverse routing required by the event. The SIM calls the async callback entry point in the XPT:
   long xpt_async(opcode, path_id, target_id, lun, buffer_ptr, data_cnt)

The XPT shall be responsible for ensuring that requests to the xpt_async() routine are processed in serial fashion. Alternatively, the SIM shall call the peripheral driver directly. In this case the SIM shall be responsible for serialization.

All of the arguments, other than the pointer, are long values of 32 bits. The value of -1 in path_id, target_id, and lun fields can be used as a wildcard. A null buffer pointer value and a count of 0 are valid for opcodes that do not require any data transfer.

Using the path_id, target_id, lun fields, and event opcode information available directly to the SIM or to the XPT from the xpt_async() call, the XPT or SIM scans its internal tables looking for "matches" with the registered asynchronous callback peripheral drivers (see Clause 9.2.7). When a match is found, either exactly or with a wildcard of "-1," the XPT or SIM shall copy the data for the opcode, if available, into the area reserved by the peripheral driver and then call the peripheral driver's async callback routine.

The arguments to the peripheral driver's async callback routine are the same as the xpt_async() routine.

```
void (*cam_async_func) (opcode,path_id,target_id,lun,buffer_ptr,data_cnt)
```

The buffer_ptr value shall be the peripheral driver's buffer. The data_cnt value shall be what the XPT has to transfer from the SIM's buffer up to the limit of the peripheral driver's buffer.

For most opcodes the path_id, target_id, and lun arguments are the only ones needed. The only opcodes that require an additional buffer are AEN, Path ID registered, and Path ID de-registered. Table 1 lists the opcodes and the expected data requirements for the number of bytes to be transferred.

Table 1 - Asynchronous event callback opcode data requirements

Event	Opcode	Path ID	Target	LUN	Data cnt
Unsol. SCSI bus reset	0x0001	Valid	n/a	n/a	n/a
Unsol. reselection	0x0002	Valid	Valid	Valid	n/a
reserved	0x0004				
SCSI AEN	0x0008	Valid	Valid	Valid	Min. 22
Sent BDR to target	0x0010	Valid	Valid	n/a	n/a
Path ID registered	0x0020	XPT ID	n/a	n/a	Min. 1
Path ID de-registered	0x0040	XPT ID	n/a	n/a	Min. 1
New devices found	0x0080	Valid	n/a	n/a	n/a

The AEN requires a minimum of 22 bytes of buffer space. This space includes the 4 bytes required by the AEN data format and 18 bytes defined by the sense data format (see SCSI-2).

The Path ID registered and the Path ID de-registered data requirements shall be a minimum of 1 byte. This byte shall contain the Path ID needed to access the SIM. Since Path ID registered and Path ID de-registered Asynchronous Callback Requests are directed to the XPT, the Path ID argument is the XPT's Path ID (0xFF).

The new devices found opcode shall be returned whenever the XPT/SIM issues an inquiry which detects that a device is attached which was not previously found (e.g., a printer powered on after system initialization was completed).

NOTE: Some devices provide minimal information at power-on and cannot provide complete inquiry information until after some delay. A XPT/SIM may scan the bus after initialization to update its tables with the complete inquiry information.

If there is valid data placed in the peripheral driver's data buffer by the XPT/SIM, the peripheral driver is required to save or discard that data before returning control to the XPT/SIM.

7.7 Autosense

Autosense causes sense data to be retrieved automatically if a CHECK CONDITION or COMMAND TERMINATED status is reported in the SCSI Status field of the CCB. Unless the Disable Autosense CAM Flag is a 1, the SIM shall perform an Autosense function as described in this Clause.

The SIM shall construct a REQUEST SENSE command and send it to the same Logical Unit. The location and amount of sense data are specified in the Sense Info Buffer Pointer and Sense Info Buffer Length fields respectively of the SCSI I/O Request CCB. If the length field is 0 or the Sense Info Buffer Pointer field is null, the REQUEST SENSE command shall be issued, but with a data allocation length of 0.

After completing the autosense sense sequence without failure the CAM Status and SCSI Status fields shall contain the status of the original command that caused the check condition.

The target can return fewer than the number of sense bytes requested. This is not reported as an error, sense status shall be flagged as valid and the Autosense Residual Length field shall be set correctly.

7.8 Loadable modules

Some operating system environments provide the ability to load or unload software drivers, thus peripheral drivers or SIM modules can be loaded dynamically. In such systems, the XPT module (typically supplied by the OS vendor) is either part of the system or must be loaded first.

The XPT, as part of a loadable OS, exports its "label," which is to used as a reference by the other loadable modules. The XPT manages the loading of SIMs and provides the common access point for peripheral drivers to register a loaded or unloaded SIM.

When a peripheral driver is loaded, it can go through its initialization process (see OSD initialization), call the XPT initialization point and then query the XPT for the HBAs (Path IDs) that are present in the system and targets that have been identified as being on those HBAs.

When a SIM is loaded, the SIM and XPT have to work together to have the SIM-supported HBAs registered as addressable Path IDs. The SIM shall call the XPT once for each supported bus in order to obtain the Path ID for that bus.

```
long xpt_bus_register(CAM_SIM_ENTRY *)
```

The argument is the pointer for the data structure defining the entry points for the SIM. The value returned is the assigned Path ID; a value of -1 indicates that registration was not successful.

The CAM_SIM_ENTRY table is used to define the entry points for the SIMs.

```
typedef struct
{
    long (*sim_init)();          /* pointer to the SIM init routine */
    long (*sim_action)();        /* pointer to the SIM CCB go routine */
} CAM_SIM_ENTRY;
```

When the xpt_bus_register function is called, the XPT shall update its internal tables and then call the sim_init(path_id) function pointed to by the CAM_SIM_ENTRY structure. The initialization for the loaded SIM is no different than for a SIM statically included in the kernel at boot time. After the SIM has gone through the initialization process, the XPT shall scan the SCSI bus in order to update its internal tables containing inquiry information.

The SIM shall call the XPT once to de-register the bus for a given Path ID:

```
long xpt_bus_deregister(path_id)
```

The argument is the Path ID for the bus being de-registered. A return value of zero indicates the bus is no longer registered, any other value indicates the call was unsuccessful.

When the xpt_bus_deregister function is called, the XPT shall update its internal tables to reflect that the path_id (HBA) is not present.

Peripheral drivers can request to be informed when a Path ID is registered or de-registered via the async callback feature (see Clauses 7.6 and 9.2.7).

8 OSD (operating system dependent) operation

8.1 UNIVOS operating system

The CAM subsystem is intended to provide a set of services for third-party vendors.

There are several sets of modules for UNIVOS:

- peripheral drivers that are device class specific
- a configuration_driver for initialization
- the XPT
- SIMs that are HBA-specific

Each member of these sets is treated as a UNIVOS driver and is linked into the kernel. The XPT and configuration_driver (which is responsible for initialization) are OS-vendor specific; other drivers may come from any source.

At kernel configuration and link time the cam_conftbl[] is created and contains entry points for the SIMs, which are used by the XPT.

The cam_conftbl[] is used by the XPT/configuration_driver to call routines and pass CAM parameters between them (e.g., the Path ID contained in the CCB created by the peripheral driver) is used to index into the cam_conftbl[]. The entry point for the selected SIM, sim_action() is called with a pointer to the CCB as an argument.

The cam_edt[] data structure is used and created during the initialization process to contain the necessary information of all the targets found on all the HBAs during the init sequence.

The CAM Flags used are as described in table 18.

The success of a function is reported with a CAM Status of Request Completed without Error.

The failure of a function is reported with any other CAM Status except Request In Progress.

8.1.1 Initialization

The initialization of the XPT and SIMs is under the control of the configuration_driver.

Due to the different UNIVOS-based systems, there is no common initialization process that can control the order of calls to the peripheral driver's and configuration_driver's init() routines. It is necessary to make sure that the subsystem is initialized before any requests can be serviced from the peripheral drivers. Due to this constraint when the peripheral driver's initialization routines are called the driver shall call the xpt_init() routine. If the subsystem is not yet initialized, the XPT shall call the configuration_driver to formally initialize the subsystem. Once the

subsystem is set up, either from a previous xpt_init call or the configuration_driver being called, all subsequent xpt_init calls shall simply return.

When the configuration_driver is called for initialization, it uses the cam_conftbl[] entry structures. The configuration_driver makes the init() routine calls, to the XPT, and to each SIM in turn, allowing them to initialize. The initialization routine for the SIM is called with its Path ID as the argument. Interrupts shall be disabled or blocked by the configuration_driver during the initialization process.

After the initialization process has been completed, the configuration_driver obtains information about each SIM, HBA, and Logical Unit detected, and maintains a table, the cam_edt[], of these devices. The information is obtained by using CCBs through the CAM interface.

Once the CAM subsystem is initialized and the cam_edt[] set, the peripheral drivers can use the subsystem. This allows them to ascertain what devices are known and make appropriate memory allocations and resource requests of the XPT.

The SCSI-2 inquiry command shall be issued to all Logical Units on the attached interfaces, and shall contain an allocation length of 36 bytes, which is sufficient to transfer the device information and the product information. The EVPD and page code fields in the inquiry command shall be set to 0. It is assumed that the responding devices return the inquiry data, even though the device may not be ready for other commands. A limited number of retries are done for devices that return busy status following the inquiry command. If the retry limit is reached, the status of the device in the XPT is set to "not found". The inquiry command shall be the only command issued by the XPT to the devices during initialization.

8.1.2 Accessing the XPT

8.1.2.1 From the peripheral driver

The XPT provides functions to obtain CAM system resources for the peripheral driver. These functions are used to allocate and free CCB resources.

There are two routines used in the handling the CCB resources. The two routines are:

CCB *xpt_ccb_alloc() and
void xpt_ccb_free(CCB *):

- The xpt_ccb_alloc() routine returns a pointer to the allocated CCB. The peripheral driver can now use this CCB for it's SCSI/XPT requests.
- The xpt_ccb_free() routine takes a pointer to the CCB that the peripheral driver has finished with, and can now be returned to the CAM subsystem CCB pool.
- The pointer to the CCB returned from the xpt_ccb_alloc() call shall be large enough to contain any of the possible XPT/SIM function request CCBs.
- The CCB can only be used (i.e., sent to the XPT), once. Once the CCB has completed it shall be returned using the xpt_ccb_free() routine.
- The xpt_ccb_alloc() routine shall return a null if memory resources are not immediately available.

8.1.2.2 From the SIM

The SIMs obtain requests from the XPT as they are passed across from the peripheral driver, via a routine included in the SIM's configuration information. The field in the configuration table is declared as "long (* sim_action)(CCB *)." The XPT does not modify CCBs or CDBs. The XPT shall intercept those CCBs which must be redirected to the configuration_driver (get device type, and set device type).

8.1.3 Callback on completion

The callback on completion field in the CCB is a structure that is platform specific, but always contains at least a callback function pointer, named cbfcnp, and declared as "void (*cbfcnp)(CCB *)." The argument to cbfcnp shall be the address to the CCB.

The disable callback on completion feature should not be used. Peripheral drivers should not poll the CAM Status field.

8.1.4 Pointer definition in the UNIVOS environment

Pointers in the CAM environment are treated as any other pointer in a given UNIVOS implementation. For the Intel 80386 platforms, pointers are 32-bit virtual addresses into a flat address space.

8.1.5 Request mapping information

This field is expected to contain a pointer to the buf structure that the SCSI I/O CCB was created for. This copy of the buf structure pointer, bp, is used by the SIM to get to the I/O mapping information needed to access the data buffers allocated by the application program. A value of null is allowed if there is no need for the SIM to map the data buffer addresses (i.e., data count is zero), the buffer is internal to the kernel, or the addresses are physical.

8.1.6 XPT interface

The XPT interface provides functions that peripheral drivers and SIM modules can access in order to transfer information and process user requests. The following defines the entry points, and describes the required arguments and return values.

8.1.6.1 Functions for peripheral driver support

a) long xpt_init()

This routine is called by the peripheral driver to request that the XPT and sub-layers be initialized. Once the sub-layers are initialized any subsequent calls by other peripheral drivers shall quickly return.

There are no arguments and the return value is either success or failure.

b) CCB *xpt_ccb_alloc()

This routine is used whenever a peripheral driver needs a CCB (the common data structure for processing SCSI requests). It returns a pointer to the allocated CCB which the peripheral driver can now use as the CCB for its SCSI/XPT requests. The returned CCB shall be properly initialized for use as a SCSI I/O request CCB. The SIM private data area shall have been already set up to be used by the XPT and SIM, and shall not be modified by the peripheral driver.

There are no arguments and the return value is a pointer to an initialized CCB.

c) void xpt_ccb_free(CCB *)

This routine takes a pointer to the CCB that the peripheral driver has finished with so it can be returned to the CAM subsystem CCB pool.

The argument is the pointer to the CCB to be freed, there is no CAM Status.

d) long xpt_action(CCB *)

All CAM/SCSI CCB requests to the XPT/SIM are placed through this function call. All returned CAM Status information is obtained at the callback point via the CAM and SCSI status fields.

The argument is a pointer to the CCB, and the return value is either success or failure.

8.1.6.2 Functions for SIM module support

a) See 7.8 for loadable module support:

long xpt_bus_register(CAM_SIM_ENTRY *)

long xpt_bus_deregister(path_id)

b) void xpt_async(opcode, path_id, target_id, lun, buffer_ptr, data_cnt)

The SIM calls this routine to inform the XPT that an async event has occurred and that there may be peripheral drivers which need to be informed.

- The opcode, path_id, target_id, lun, and data_cnt arguments are long 32-bit values.
- The path_id, target_id, and lun define a Logical Unit for the async callback.
- The opcode contains the value for what has happened.
- The buffer_ptr and data_cnt are used to inform the XPT where and how much data is associated with the opcode.

8.1.7 SIM interface

The SIM interface provides functions to the XPT, and should never be accessed directly by the peripheral driver. Each vendor's SIM should provide a publicly-defined entry structure such as CAM_SIM_ENTRY cse_vendorname.

The following defines the entry points, and describes the required arguments and return values.

a) long sim_init(path_id)

This routine is called by the XPT to request that the SIM be initialized. The argument is the Path Id assigned to the SIM and the returned value is either success or failure.

b) long sim_action(CCB *)

All CCB requests to the SIM are placed through this function call. All returned CAM Status information is obtained at the callback point via the CAM and SCSI status fields.

The argument is a pointer to a CCB, and the returned value is either success or failure.

8.2 LANOS

LANOS drivers are called NLMs (LANOS loadable modules). These modules are registered and linked dynamically with LANOS: they are loaded after the operating system is initialized and may be unloaded at any time.

The LANOS CAM subsystem consists of 3 sets of NLMs:

- peripheral drivers (NLMs) that are device class specific
- the XPT router and SIM maintenance NLM
- SIM NLMs that are HBA-specific

The peripheral drivers and SIMs communicate with the XPT through labels exported by the XPT when it is loaded.

The CAM Flags used are as described in table 18.

8.2.1 Initialization

As the LANOS dynamic linker does not allow an NLM to load if it makes references to a label it cannot resolve, the order in which the NLMs load is important. The XPT module exports four entry points when it is loaded, and both peripheral drivers and SIM modules make references to them. The XPT shall be loaded first, after which either peripheral drivers or SIMs may be loaded.

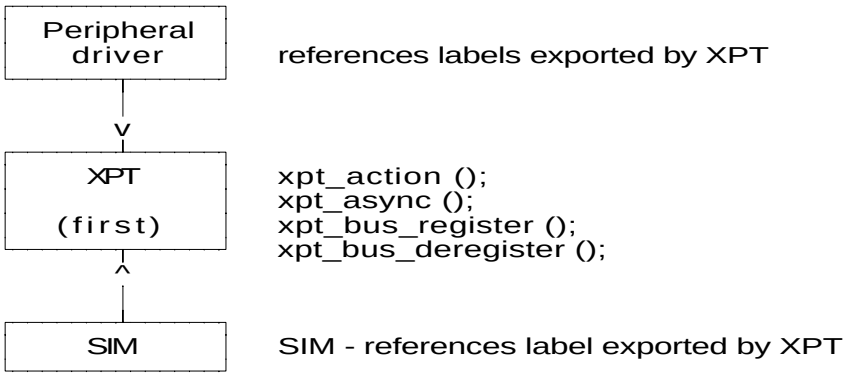

Figure 2 - CAM layers

For an overview of SIM SCSI registration with the XPT see 7.8. For an overview of peripheral driver registration with the XPT see 7.6 and 9.2.6.

When LANOS loads a SIM, it shall call the initialization routine specified in the LANOS linker definition file. At this point the SIM can perform its initialization functions.

As part of initialization the SIM shall call the xpt_bus_register function once for each HBA it supports, to register the address of its entry point with the XPT and to get a Path ID for each HBA from the XPT. The XPT then adds this SIM to its internal tables so it can route requests to the new SIM. The XPT also notifies all peripheral drivers that registered an asynchronous callback routine with the XPT (with the SIM module registered bit set), that a new Path ID exists. Upon receiving this message the peripheral drivers can check for new devices on this path.

When LANOS loads a peripheral driver, the initialization routine specified in the linker definition file shall be called. At this time, the driver needs to ascertain which, if any, SIMs are registered.

The peripheral driver sends a PATH INQUIRY CCB to each path to ascertain if a SIM is registered. If a valid response is returned the peripheral driver checks for devices that it supports on that path. If the peripheral driver supports any devices on this path, it shall register an asynchronous callback routine and specify the SIM registration in the opcode field so that if the SIM is de-registered, the peripheral driver shall be notified. In addition, a peripheral driver should also register for SIM registration to alert the driver of the need to locate devices on a newly added SIM module.

8.2.2 SIM and peripheral driver unloading

Before a SIM unloads, it shall call the xpt_bus_deregister() function once for each path the SIM supports. The XPT then calls every peripheral driver that has registered an asynchronous callback routine with the SIM module de-registered bit set on this path. Peripheral drivers then notify LANOS that the drives on this path are in an inactive state. The XPT then removes the path from its internal tables and further peripheral driver requests on this path shall return CAM Status of Invalid Path ID.

Before a peripheral driver unloads, it needs to notify the XPT module so that the dependency tables can be updated. This is done by registering an asynchronous callback routine with the opcode set to zero. The XPT then removes this driver from its callback tables.

The XPT can only be unloaded after all peripheral drivers and SIM modules have been unloaded. LANOS does not allow an NLM to unload if it has exported labels that other NLMs are using. As all SIM and peripheral drivers refer to labels exported by the XPT, LANOS does not allow the XPT to unload until all the SIMs and peripheral drivers have been unloaded, at which point there is nothing left for the XPT to support and it can be safely unloaded.

8.2.3 Accessing the XPT

LANOS allows an NLM to export functions which NLMs loaded at a later time can reference. An NLM calls an exported function in the same way it calls any other function. The C language calling convention is assumed. In order for communication between the peripheral drivers, XPT, and SIM modules to work correctly the names of the XPT entry points have to be constant.

The entry points in the XPT module are:

- xpt_action () accepts CAM control blocks from the peripheral driver and routes them to the correct SIM
- xpt_async () is used by the SIM module to notify the XPT when an asynchronous event occurs.
- xpt_bus_register () is used by the SIM to register a SCSI bus with the XPT and obtain a Path ID for the SCSI bus.
- xpt_bus_deregister () is used to de-register the passed Path ID and associated SCSI bus.

8.2.4 Hardware registration

The SIM module needs to do the actual registration of the host adapter with LANOS. Since only one SIM may support a given host adapter this prevents any hardware options from being registered twice. The SIM does not register any devices with LANOS, only the hardware options used by the card (e.g., interrupt line, base address, DMA etc).

Interrupts generated by the host adapter are handled by the SIM module, so the SIM must also register its interrupt service routine with LANOS.

A peripheral driver registers a logical card with LANOS for each path_id it supports. This logical card uses no hardware resources, but does have entry points for I/O and IOCTL requests from LANOS. The peripheral driver also reports the devices that it supports to LANOS.

The XPT does not register any hardware or devices with LANOS. It loads as a driver, but does not register any IOPOLL or IOCTL entry points.

8.2.5 Miscellaneous

It is the responsibility of the peripheral driver to allocate memory for its CCB blocks. Normally the peripheral driver needs to keep one CCB structure for each device it supports, so the memory can be allocated in the data structure provided by LANOS when a device is added to the system.

Since fast disk channels are essential for a LANOS server, peripheral drivers should never poll the CAM Status field to wait for completion. The driver should send the CCB to the XPT module and then either do more work, or exit immediately. The SIM module calls the function whose address is in the callback field of the CCB block when the request is finished. The callback function runs at interrupt level, so it cannot call any LANOS routines that are "blocking" or the file server abends.

8.3 DOS (disk operating system)

Under DOS, a software interrupt is used to access any of the XPT or SIM functions, which are combined into a single module.

The routing functions of the XPT are performed by the DOS concept of "interrupt vector chaining." During execution, an XPT/SIM module ascertains if a particular CCB is one that it should handle. If not, it routes the CCB to the previous "owner" of the interrupt vector.

The CAM Flags used by the DOS XPT/SIM are described in table 18.

8.3.1 Initialization

During initialization, the XPT/SIM modules should be loaded as character device drivers.

As character device drivers are required by DOS to have unique names, the 8-character device name should be "$$CAMxxx", where xxx is the ASCII decimal numeric value of the lowest Path ID supported by this XPT/SIM module.

The programming examples in this clause are used to assist the reader's understanding. Implementations do not need to use the same code, but they are required to accomplish the same goals.

8.3.1.1 Multiple XPTs

The pseudocode for the XPT initialization sequence is as follows:

```
Get INT 4Fh interrupt vector;
Save this address for chaining;
IF there is a CAM XPT already installed (see 8.3.2.1)
    Perform Path Inquiry (Path ID=0FFh) to get highest Path ID;
    First Path ID = highest Path ID + 1;
ELSE
    First Path ID = 0;
END IF;
Count number of Path IDs needed;
IF no HBAs to support (count = 0)
    Exit initialization without installing driver;
END IF;
Set INT 4Fh interrupt vector to point to CAM entry point;
Save highest Path ID used (first Path ID + count - 1);
Set character device name to "$$CAMxxx",
    where xxx=first Path ID;
```

```
Perform all necessary HBA initialization;
FOR each SCSI bus supported:
    FOR each SCSI ID (excluding initiator)
        IF device exists
            FOR each LUN
                Perform INQUIRY to get PDT for table;
            END FOR;
        END IF;
    END FOR;
END FOR;
```

8.3.1.2 Device table handling

The XPT/SIM is only required to keep the peripheral device type of the devices connected to the supported SCSI bus(es).

8.3.2 Accessing the XPT

There are various mechanisms used to access XPT or SIM functions from peripheral drivers or application programs.

8.3.2.1 Testing for the presence of the XPT/SIM

Peripheral drivers and applications can check for the presence of an XPT/SIM module (e.g., by performing a "check install") function such as:

On entry:
- AX = 8200h
- CX = 8765h
- DX = CBA9h

On return:
- AH = 0 (if successful)
- CX = 9ABCh
- DX = 5678h
- ES:DI = address of character string "SCSI_CAM"
- All other registers unaffected.

The following routine checks for the presence of an XPT/SIM module. It returns a value of 1 if a module is found and a value of 0 if not found.

```
CHK_FOR_CAM  PROC   NEAR
             MOV    CX,8765H        ; load l.s.w. of signature
             MOV    DX,0CBA9H       ; load m.s.w. of signature
             MOV    AX,8200H        ; load "check install" code
             INT    4FH             ; perform "check install"
             CMP    AH,0            ; function supported?
             JNE    NOT_THERE       ; if not, no xpt/sim
             CMP    DX,5678H        ; check m.s.w. of signature
             JNE    NOT_THERE       ; if invalid, no xpt/sim
             CMP    CX,9ABCH        ; check l.s.w. of signature
             JNE    NOT_THERE       ; if invalid, no xpt/sim
             CLD                    ; set direction flag
```

```
                MOV     CX,8                    ; load string length
                MOV     SI,OFFSET SCSI_CAM      ; get string address
                REPE        CMPSB                   ; compare strings
                JNE     NOT_THERE               ; if strings differ, no xpt/sim
                MOV     AX,1                    ;load "found" status
                RET                             ; return to caller
NOT_THERE:      MOV     AX,0                    ; load "not found" status
                RET                             ; return to caller
CHK_FOR_CAM     ENDP
SCSI_CAM        DB      'SCSI_CAM'              ; string to find
```

8.3.2.2 Sending a CCB to the XPT

Once it is ascertained that an XPT/SIM module is present, the peripheral driver or application can access the XPT/SIM functions by sending a CCB to the XPT/SIM e.g.

On entry:

ES:BX = address of the CCB
AX = 8100h

On return:

AH = 0 if successful
= any other value if an error occurred
All other registers unaffected.

NOTE: The SIM may complete and return control to the location pointed to by the callback on completion field in the CCB before the software interrupt returns.

The following routine sends a CCB to the XPT/SIM module. It returns a value of 0 if successful and 1 if not.

```
SEND_CCB    PROC    NEAR
            MOV     AX,8100H            ; load "send CCB" function
            MOV     ES,SEGMENT CCB      ; load segment of CCB
            MOV     BX,OFFSET CCB       ; load offset of CCB
            INT     4FH                 ; call XPT/SIM module
            SHR     AX,8                ; put return code in al
            RET                         ; return to caller
SEND_CCB    ENDP
```

8.3.3 Callback on completion

When an I/O operation has completed, a XPT/SIM module shall make a FAR call to the routine which had its address passed in the callback on completion field of the CCB. The first 4 bytes of this field are used to indicate the routine's address in the Intel segment:offset format. When the callback is made, all hardware interrupts shall be disabled and ES:BX shall point to the completed CCB.

8.3.4 Asynchronous event callbacks

There are some differences in the DOS XPT/SIM implementation of asynchronous callbacks as compared with the description in 7.6.

The DOS XPT/SIM does not support the SIM module loaded and SIM module unloaded opcodes reported by the XPT/SIM module when the asynchronous callback routine is called.

The SET ASYNCHRONOUS CALLBACK CCB is held by the XPT/SIM until it is "de-registered." This is accomplished by sending another SET ASYNCHRONOUS CALLBACK CCB to the XPT/SIM with all of the Asynchronous Event Enables field reset and the address of the original SET ASYNCHRONOUS CALLBACK CCB in the Peripheral Driver Buffer Pointer field. At that point the original CCB shall be dequeued and both CCBs shall be returned to the peripheral driver or application.

NOTE: There is an implication here that a peripheral driver or application which wishes to be notified when the specified asynchronous event occurs, has to register separately with each Path ID.

The peripheral driver buffer pointer and size of allocated peripheral buffer fields in the SET ASYNCHRONOUS CALLBACK CCB are considered as private data by the XPT/SIM, to be used for CCB queuing.

When an asynchronous event occurs that is enabled by the bits in the Asynchronous Event Enables field of the SET ASYNCHRONOUS CALLBACK CCB, the virtual address specified by the Asynchronous Callback Pointer field shall be called with the following registers:

On entry:

AH = opcode as specified in table 1.
AL = Path ID that generated the callback.
DH = Target ID that caused event (if applicable).
DL = LUN that caused event (if applicable).
CX = data byte count (if applicable).
ES:BX = address of data buffer (if applicable).

On return:

All registers shall be preserved.

It is the responsibility of the peripheral driver or application to copy any or all required data out of the data buffer into a local buffer before returning from the asynchronous callback routine.

8.3.5 Pointer definition

All pointers shall be passed to the XPT/SIM as segment:offset type virtual addresses.

9 CAM control blocks

The CCBs used by peripheral drivers and applications to request functions of the XPT and SIM have a common header, as shown in table 2.

Table 2 - CAM control block header

Size	Dir	
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID
1	O	Target ID
1	O	LUN
4	O	CAM Flags

The sequence of the fields in the data structures shall be consistent between vendors (i.e., the binary offset shall be the same for every field). The binary contents of fields may vary according to the memory addressing protocol of the processor.

The definition of the fields in the data structures can vary among operating systems and hardware platforms, but the vendors are expected to provide compiler definitions which can be used by third-party attachments. Several fields in the CCB are pointers, and their meaning is dependent on the OS which is being supported. In general, these pointers are interpreted as either virtual or physical addresses.

Additional bytes beyond the CCB header are dependent on the function code.

A peripheral driver, the XPT, or a SIM that allocates CCB(s) shall be responsible for freeing those CCBs it has allocated after the allocating entity is finished with the CCB(s). A peripheral driver, the XPT, or a SIM shall not free any CCB that it has not allocated. A peripheral driver, the XPT, or a SIM shall not free a CCB that has been sent to xpt_action() and has not completed (see Clause 7.3 for further information on CCB completion). A peripheral driver, the XPT, or a SIM shall not free a CCB more then once per allocation of that CCB and shall be responsible for keeping track of the CCB(s) it is using in a vendor unique manner.

Most SCSI messages are handled transparently by the SIM, but in some cases, the peripheral driver has been given the ability to force the SIM to issue a message. Table 3 summarizes the message support.

Table 3 - Support of SCSI messages

ABORT	Discretely supported by function codes
ABORT TAG	Discretely supported by function codes
BUS DEVICE RESET	Discretely supported by function codes
CLEAR QUEUE	Not supported
COMMAND COMPLETE	Transparently supported by SIM
DISCONNECT	Transparently supported by SIM *
IDENTIFY	Transparently supported by SIM
IGNORE WIDE RESIDUE	Transparently supported by SIM
INITIATE RECOVERY	Not supported
INITIATOR DETECTED ERROR	Transparently supported by SIM
LINKED COMMAND COMPLETE	Transparently supported by SIM
MESSAGE PARITY ERROR	Transparently supported by SIM
MESSAGE REJECT	Transparently supported by SIM
MODIFY DATA POINTER	Transparently supported by SIM
NO OPERATION	Transparently supported by SIM
Queue tag messages	
HEAD OF QUEUE TAG	Discretely supported by function codes
ORDERED QUEUE TAG	Discretely supported by function codes
SIMPLE QUEUE TAG	Discretely supported by function codes
RELEASE RECOVERY	Not supported
RESTORE POINTERS	Transparently supported by SIM
SAVE DATA POINTERS	Transparently supported by SIM
SYNCH DATA TRANSFER REQUEST	Transparently supported by SIM *
TERMINATE I/O PROCESS	Discretely supported by function codes
WIDE DATA TRANSFER REQUEST	Transparently supported by SIM
* Issuing this message influenced by peripheral driver via CAM Flags	

9.1 CCB header fields

9.1.1 Address of this CCB

Pointer containing the physical address of this CCB.

9.1.2 CAM control block length

This field contains the length in bytes of the CCB, including this field and the address of this CCB in the total.

9.1.3 XPT function code

The function codes used to identify the XPT service being requested are listed in table 4.

Table 4 - XPT function codes

Code	
00-0F	Common functions
00h	NOP
01h	Execute SCSI I/O (see 10.x)
02h	Get Device Type
03h	Path Inquiry
04h	Release SIM Queue
05h	Set Asynchronous Callback
06h	Set Device Type
07h	Scan SCSI Bus
08-0F	Reserved
10-1F	SCSI control functions
10h	Abort SCSI Command
11h	Reset SCSI Bus
12h	Reset SCSI Device
13h	Terminate I/O Process
14h	Scan Logical Unit
15-1F	reserved
20h	Engine Inquiry (see 12.x)
21h	Execute Engine
22-2F	reserved
30-3F	Target mode (see 11.x)
30h	Enable LUN
31h	Execute Target I/O
32h	Accept Target I/O
33h	Continue Target I/O
34h	Immediate Notify
35h	Notify Acknowledge
36-3F	reserved
40-7F	reserved
80-FF	Vendor Unique

If a function code which is not supported is issued to the XPT, the XPT shall complete the request and post CAM Status of Invalid Request.

9.1.4 CAM status

This field is returned by the SIM after the function is completed. A zero status indicates that the request is still in progress or queued. CAM Status is defined in Table 5.

If autosense information is available, the code returned shall be incremented by 80h (e.g., 04h indicates an error occurred, and 84h indicates that an error occurred and autosense information is available for analysis).

Table 5 - CAM status

00h	Request in Progress
01h	Request Completed without Error
02h	Request Aborted by Host
03h	Unable to Abort Request
04h	Request Completed with Error
05h	CAM Busy
06h	Invalid Request
07h	Invalid Path ID
08h	SCSI Device Not Installed
09h	Unable to Terminate I/O Process
0Ah	Target Selection Timeout
0Bh	Command Timeout
0Ch	Reserved
0Dh	Message Reject Received
0Eh	SCSI Bus Reset Sent/Received
0Fh	Uncorrectable Parity Error Detected
10h	Autosense Request Sense Cmd Failed
11h	No HBA Detected
12h	Data Overrun
13h	Unexpected Bus Free
14h	Target Bus Phase Sequence Failure
15h	CCB Length Inadequate
16h	Cannot Provide Requested Capability
17h	Bus Device Reset Sent
18h	Terminate I/O Process
19h	Unrecoverable Host Bus Adaptor Error
1Ah	SCSI Bus Reset Denied
1B-32h	Reserved
	Target mode only status
33h	Initiator Detected Error Received
34h	Resource Unavailable
35h	Unacknowledged Event by Host
36h	Message Received
37h	Invalid CDB
38h	Invalid LUN
39h	Invalid Target ID
3Ah	Function Not Implemented
3Bh	Nexus Not Established
3Ch	Invalid Initiator ID
3Dh	SCSI CDB Received
3Eh	LUN Already Enabled
3Fh	SCSI Bus Busy
+40H	to indicate that SIM queue is frozen
+80h	to indicate that autosense is valid

- 00h Request in Progress: the request is still in process.
- 01h Request Completed without Error: the request has completed and no error condition was encountered.
- 02h Request Aborted by Host: the request was aborted by the SIM/HBA.
- 03h Unable to Abort Request: the SIM/HBA was unable to abort the request as instructed by the peripheral driver.
- 04h Request Completed with Error: the request has completed and an error condition was encountered.
- 05h CAM Busy: CAM unable to accept request at this time.
- 06h Invalid Request: the request has been rejected because it is invalid.
- 07h Invalid Path ID: indicates that the Path ID is invalid.
- 08h SCSI Device Not Installed: peripheral device type field is not valid.
- 09h Unable to Terminate I/O Process: the SIM/HBA was unable to terminate the request as instructed by the peripheral driver.
- 0Ah Target Selection Timeout: The target failed to respond to selection.
- 0Bh Command Timeout: the specified command did not complete within the timer value specified in the CCB. Prior to reporting this status the SIM/HBA shall ensure the command is no longer active in the target.

- 0Dh Message Reject Received: The SIM/HBA received a SCSI MESSAGE REJECT message.
- 0Eh SCSI Bus Reset Sent/Received: The SCSI operation was terminated at some point because the SCSI bus was reset.
- 0Fh Uncorrectable Parity Error Detected: An uncorrected SCSI bus parity error was detected.
- 10h Autosense Request Sense Command Failed: The SIM/HBA attempted to obtain sense data and failed.
- 11h No HBA Detected: HBA no longer responding to SIM (assumed to be a hardware problem).
- 12h Data Overrun: target transferred more data bytes than peripheral driver indicated in the CCB.
- 13h Unexpected Bus Free: an unexpected bus free condition occurred.
- 14h Target Bus Phase Sequence Failure: the Logical Unit failed to operate in compliance with ANSI X3.131-1994.
- 15h CCB Length Inadequate: more private data area is required in the CCB (see Clause 9.2.3 for further information).
- 16h Cannot Provide Requested Capability: resources are not available to provide the capability requested in the CAM Flags.
- 17h Bus Device Reset Sent: this CCB was terminated because a BUS DEVICE RESET message was sent to the target.
- 18h Terminate I/O Process: this CCB was terminated because a Terminate I/O Process function was specified for this CCB and the CCB was not an I/O process within the Logical Unit.
- 19h Unrecoverable Host Bus Adaptor Error: this CCB was terminated because of a hardware error detected by the HBA. The error does not indicate a SCSI bus problem but an error within the HBA or host.
- 1Ah SCSI Bus Reset Denied: this CCB was not accepted by the SIM/HBA due to a hung SCSI bus condition. This CAM Status is a result of an intelligent HBA detecting a hung SCSI bus and with a SIM/HBA specific mechanism, requesting permission from the controlling SIM software to perform a SCSI bus reset. The permission to perform a SCSI bus reset was denied by the SIM.

Note: This CAM Status is forecasted as being deleted in future versions of CAM due to the following reasons:
  - An intelligent HBA that reports (in a SIM/HBA specific manner) a hung SCSI bus to the controlling SIM software is always be granted the permission to reset the SCSI bus.
  - Multiple SCSI bus resets could occur to clear the condition if the decision to reset the bus is left to multiple peripheral drivers.

- 33h Initiator Detected Error: indicates the the SIM/HBA has received an INITIATOR DETECTED ERROR message.
- 34h Resource Unavailable: indicates that the SIM/HBA has run out of resources for processing connections (Host Target Mode only).
- 35h Unacknowledged Event: indicates that the Host Target Mode peripheral driver has not acknowledged an event.
- 36h Message Received: indicates that a message has been received by the SIM/HBA that requires attention.
- 37h Invalid CDB: indicates that the SIM/HBA has detected an error condition on reception of a CDB.
- 38h Invalid LUN: indicates that the Logical Unit specified is outside the supported range of the SIM/HBA.
- 39h Invalid Target ID indicates that the Target ID does not match that used by the HBA specified by the Path ID field.
- 3Ah Function Not Implemented: indicates that target mode is not supported.
- 3Bh Nexus Not Established: there is currently no connection established between the specified Target ID and target LUN with any initiator.
- 3Ch Invalid Initiator ID: the initiator ID specified is outside the valid range that is supported.

NOTE: This status can also be returned if the target tries to reselect an initiator other than the one to which it was previously connected.

- 3Dh SCSI CDB Received: indicates that the target has been selected and that the SCSI CDB is present in the CCB.
- 3Eh LUN Already Enabled: the LUN identified in enable LUN was previously enabled.
- 3Fh SCSI Bus Busy: the SIM failed to win arbitration for the SCSI bus during several different bus free phases.

9.1.5 Connect ID

The connect ID consists of 4 separate fields, of which the first is reserved.

- Path ID
  The Path ID specifies the SCSI bus on the installed HBA to which the request is addressed. Path IDs are assigned by the XPT, begin with zero, and need not be consecutive. The Path ID of FFh is assigned for the XPT. A HBA may have more than one SCSI bus. A SIM may support more than one HBA.

- Target ID
  This field identifies the SCSI target which is to be selected for execution of the CCB request.

- LUN
  This field identifies the SCSI LUN to which this CCB is being directed.

9.1.6 CAM flags

The CAM flags qualify the function to be executed, and vary by function code (see Clause 10.1.2).

9.2 Function codes

9.2.1 NOP

A peripheral driver can execute this function at any time. The XPT shall call sim_action() passing this CCB if the Path ID is valid. The SIM shall return immediately.

Table 6 - NOP CCB

Size	Dir	NOP CCB
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID
1		Target ID
1		LUN
4		CAM Flags

This function shall return a CAM Status of:

- CAM Status of Request Completed without Error.
- CAM Status of Invalid Path ID indicates that the specified Path ID is not installed.

9.2.2 Get device type

For a given Logical Unit this function returns the Peripheral Device Type of the INQUIRY response data, and optionally the first 36 bytes of inquiry data.

Table 7 - Get device type CCB

Size	Dir	Get device type
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID
1	O	Target ID
1	O	LUN
4	O	CAM Flags
4	O	Inquiry Data Pointer
1	I	Peripheral Device Type of Logical Unit

The information on attached SCSI devices is gathered at power-on by the XPT/SIM (to eliminate the need for each driver to duplicate the effort of scanning the SCSI bus for devices).

The Peripheral Device Type of the Logical Unit field is the Peripheral Device Type from the INQUIRY response data. The XPT/SIM shall generate this data by taking byte 0 of the INQUIRY response data and setting bits 7-5 to zero.

If the Inquiry Data Pointer field contains a value other then null, it shall point to a buffer of at least 36 bytes. When the Inquiry Data Pointer field is not null the XPT/SIM shall copy the first 36 of INQUIRY response data from its internal tables to the identified buffer, if the Logical Unit responded with INQUIRY response data to the INQUIRY command as defined by SCSI-2.

This function shall return a CAM Status other than Request in Progress. The CAM Status shall be one of the following:

- Request Completed without Error indicates that the specified device is installed and the peripheral device type field is valid.
- SCSI Device Not Installed indicates that the peripheral device type field is not valid.
- Invalid Path ID indicates that the Path ID is invalid.

9.2.3 Path inquiry

This function shall return information on the installed addressed HBA/SCSI bus(es) hardware, or the XPT. To obtain further information on a specific HBA(s)/SCSI bus(es) attached, this function can be issued for each assigned Path ID.

If the Path ID field of the CCB has a value of FFh (the XPT Path ID), then the only fields that shall be valid upon return to the caller are the Highest Path ID Assigned field and the Version Number field. The highest Path ID assigned field shall not be valid if the Path ID field in the CCB contains a value other than FFh.

In some operating system environments it may be possible to dynamically load and unload SIMs, so Path IDs may not be consecutive from 0 to the highest Path ID assigned.

Table 8 - PATH INQUIRY CCB - Part 1 of 2

Size	Dir	Path inquiry
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID
1	O	Target ID
1	O	LUN
4	O	CAM Flags
		Features Supported
1	I	Version Number (SIM/HBA or XPT)
		00-07h prior to rev. 1.7
		08h implementation version 1.7
		09-39h rev. no. e.g. 31h = 3.1
		40-ffh (see rev. no. explanation)
1	I	SCSI Capabilities
		7 Modify Data Pointers
		6 Wide Bus 32
		5 Wide Bus 16
		4 Synchronous Transfers
		3 Linked Commands
		2 Reserved
		1 Tagged Queuing
		0 Soft Reset

Table 8 - PATH INQUIRY CCB - Part 2 of 2

Size	Dir	Path inquiry
1	I	Target Mode Support
		7 Host Target Mode
		6 Phase Cognizant Mode
		5 Target Mode Disconnects
		4 Terminate I/O Process
		3 Group 6 Commands
		2 Group 7 Commands
		1 - 0 Reserved
1	I	Miscellaneous
		7 0=Scan Low to High
		1=Scan High to Low
		6 0=Removables Included in Scan
		1=Removables not Included
		5 1=Inquiry Data not Kept by XPT
		4-0 Reserved
		HBA Capabilities
2	I	Engine Count (see Clause 12)
14	I	Vendor Unique
4	I	Size of Private Data Area
4	I	Asynchronous Event Capabilities
		31-24 Vendor Unique
		23- 8 Reserved
		7 New Devices Found During Rescan (XPT)
		6 SIM Module De-registered (XPT)
		5 SIM Module Registered (XPT)
		4 Sent Bus Device Reset to Target (SIM)
		3 SCSI AEN (SIM)
		2 Reserved
		1 Unsolicited Reselection (SIM)
		0 Unsolicited SCSI Bus Reset (SIM)
1	I	Highest Path ID Assigned
1	I	SCSI Device ID (of initiator)
1		Reserved
1		Reserved
16	I	Vendor ID of SIM Supplier
16	I	Vendor ID of HBA Supplier
4	O	OSD Usage Pointer

If no Path IDs exist (i.e., no SCSI buses are registered), then the highest Path ID Assigned field shall be FFh, the ID of the XPT.

NOTE: For CCB's other than PATH INQUIRY CCB with a Path ID of the XPT the CCB is returned with a CAM Status of Invalid Path ID.

The Version Number field shall identify the revision number of CAM the SIM/HBA or the XPT conforms to. PATH INQUIRY CCBs addressed to a valid SIM/HBA or the XPT shall respectively place in the Version Number field the CAM revision number it conforms to. A version number of 00 to 07h represents revisions prior to revision 1.7 (i.e., 01h represents revision 1.0 and 07h represents revision 1.6). A version number of 08h represents revision number 1.7 and version numbers 09h to 39h represents the CAM revision numbers 1.9 to 3.9 (i.e., a version number of 20h represents revision 2.0 and a version number of 31h represents revision 3.1). Starting with CAM revision 4 the version number is based on a 40h plus the revision number (i.e., a version number of 44h represents the CAM revision number of 4 (40h + decimal rev number converted to hex) and a version number of 47h represents CAM revision 7). The letter suffix usage in the CAM revision numbers denotes editorial changes in the document and shall not be used in representing the CAM revision number (e.g., CAM revision 12a is represented in the Version Number field as 4Ch).

The current CAM revision number is 12b.

The SCSI Capabilities field is a duplicate of byte 7 field in inquiry data.

The OSD Usage Pointer field is provided for OS-specific or platform-specific functions to be executed by the SIM. The contents of this field are vendor-specific and are not defined in this standard.

In some environments, the Size of Private Data Area field value returned may be zero because the OSD has central allocation of private data requirements, or it is a fixed size as defined by the OSD vendor. See the vendor specification for the definition of vendor unique HBA capabilities peculiar to a particular HBA implementation.

The Asynchronous Event Capabilities field indicate what reasons can cause the XPT/SIM to generate an asynchronous event callback.

This function shall return a CAM Status other than Request in Progress. The CAM Status shall be one of the following:

- Request Completed without Error indicates that the other returned fields are valid.
- No HBA Detected indicates that the HBA is no longer responding to the SIM.
- Invalid Path ID indicates that the specified Path ID is not installed.

9.2.4 Release SIM queue

This function is provided so that the peripheral driver can decrement a SIM queue frozen count or ascertain the SIM queue frozen count without modification for the addressed Logical Unit. Determining the current SIM queue frozen count without modification is accomplished by issuing this function with the CAM Flag of SIM Queue Freeze bit set to a one. With the CAM Flag of SIM Queue Freeze bit set to zero, the SIM/HBA shall decrement the SIM queue frozen count by one and shall release the SIM queue for the addressed Logical Unit when the count reaches zero. The SIM/HBA shall return the current SIM queue freeze count for the Logical Unit in the SIM Queue Frozen Count Field for either value of the CAM Flag of SIM Queue Freeze bit. See Clause 7.4.2.3 for further clarification on the SIM queue frozen state.

The SIM Queue Frozen Count field shall be zero or a positive value and shall be ascertained by the following:

- If the CAM Flag of SIM Queue Freeze bit is set to a one, the SIM/HBA shall place the current SIM queue frozen count without any modifications into the SIM Queue Frozen Count field.
- If the CAM Flag of SIM Queue Freeze bit is set to a zero, the SIM/HBA shall decrement by one the current SIM queue frozen count and shall place that value if positive or a zero if negative into the SIM Queue Frozen Count field.

Table 9 - Release SIM queue

Size	Dir	Release SIM queue
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID
1	O	Target ID
1	O	LUN
4	O	CAM Flags
4	I	SIM Queue Frozen Count

This function shall return a CAM Status other than Request in Progress. The CAM Status shall be one of the following:

- Request Completed without Error.
- Invalid Path ID indicates that the Path ID is invalid.

9.2.5 Scan SCSI bus

This function shall cause the XPT/SIM to update its internal tables on the installed devices on the identified Path ID. The target and LUN fields shall be ignored. The XPT/SIM shall scan each Logical Unit address on the SCSI bus and update its tables with the inquiry data provided by each Logical Unit that responds.

Table 10 - SCAN SCSI BUS CCB

Size	Dir	Scan SCSI bus
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID
1		Target ID
1		LUN
4	O	CAM Flags

The information on attached SCSI devices is gathered at initialization by the XPT/SIM. This function is provided to force an update of the table contents. Using this function at any other time is not recommended because its execution can take a long time. Also execution of this function severely degrades SCSI Bus performance. Furthermore the thread that calls this function can do no further work until this function completes. Any new devices detected during the scan shall generate asynchronous callbacks to peripheral drivers registered for new devices found.

This function shall return a CAM Status other than Request in Progress. The CAM Status shall be one of the following:

- Request Completed without Error indicates that the devices have been scanned and the table updated.
- Invalid Path ID indicates that the Path ID is invalid.

9.2.6 Scan logical unit

This function shall cause the XPT/SIM to update its internal tables on the installed device on the identified Path ID, Target ID, and LUN. The XPT/SIM shall scan the Logical Unit addressed on the SCSI bus and update its tables with the inquiry data provided by the Logical Unit that responds.

Table 11 - SCAN LOGICAL UNIT CCB

Size	Dir	Scan logical unit
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID
1	O	Target ID
1	O	LUN
4	O	CAM Flags

This function is provided to force an update of the table contents for a Logical Unit not present or configured when the Scan SCSI Bus function was invoked. Execution of this function can severely degrade SCSI Bus performance if the scanned Logical Unit does not respond to selections. Furthermore the thread that calls this function can do no further work until this function completes. A new Logical Unit detected during the scan shall generate asynchronous callbacks to peripheral drivers registered for new devices found.

This function shall return a CAM Status other than Request in Progress. The CAM Status shall be one of the following:

- Request Completed without Error indicates that the devices have been scanned and the table updated.
- Invalid Path ID indicates that the Path ID is invalid.

9.2.7 Set asynchronous callback

Asynchronous event callbacks are described in Clause 7.6.

This function is provided so that a peripheral driver can register a callback routine for the selected Logical Unit. The function shall register a routine for receipt of callbacks for selected asynchronous events that occur on the selected Logical Unit. It is required that the asynchronous callback field be filled in with the callback routine address if any of the asynchronous events enabled bits are set.

Table 12 - SET ASYNCHRONOUS CALLBACK CCB

Size	Dir	Set asynchronous callback
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code
1	I	CAM Status
		Connect ID
		Reserved
1	O	Path ID
1	O	Target ID
1	O	LUN
4	O	CAM Flags
4	O	Asynchronous Event Enables
		31-24 Vendor Unique
		23- 8 Reserved
		7 New Devices Found During Rescan
		6 Path ID De-registered
		5 Path ID Registered
		4 Sent Bus Device Reset to LUN
		3 SCSI AEN (LUN)
		2 Reserved
		1 Unsolicited Reselection (LUN)
		0 Unsolicited SCSI Bus Reset(LUN)
4	O	Asynchronous Callback Pointer
4	O	Peripheral Driver Buffer Pointer
1	O	Size of Allocated Peripheral Buffer

This function shall return a CAM Status other than Request in Progress. The CAM Status shall be one of the following:

- Request Completed without Error indicates that the registration of the callback routine was accepted.
- Request Completed with Error indicates that the registration was rejected (possibly due to invalid parameter settings).
- Invalid Path ID indicates that the Path ID is invalid.

9.2.8 Set device type

In response to this function, the XPT/SIM shall add the Target ID, LUN, and peripheral type to the table of attached peripherals built during CAM initialization.

Table 13 - SET DEVICE TYPE CCB

Size	Dir	Set device type
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID
1	O	Target ID
1	O	LUN
4	O	CAM Flags
1	O	Peripheral Device Type of LUN

The XPT/SIM does not check the validity of the information supplied by the peripheral driver.

NOTE: Insertion of device type information may corrupt the table, and the results would be unpredictable.

This function shall return a CAM Status other than Request in Progress. The CAM Status shall be one of the following:

- Request Completed without Error indicates that the specified information was inserted into the table of SCSI devices.
- Request Completed with Error indicates a problem (e.g., not enough room in the table to add the device information).

9.3 SCSI control functions

9.3.1 Abort SCSI command

This function requests that a SCSI command be aborted by identifying the CCB associated with the request. It should be issued on any I/O request that has not completed that the driver expects to abort. The SIM/HBA shall issue an ABORT message or an ABORT TAG message on the SCSI bus based on the following conditions:

- ABORT message:
  - The CCB identified in the CCB to be Aborted Pointer field is not an established I_T_L_Q nexus I/O process and is an established I_T_L nexus I/O process (untagged command currently active).

- ABORT TAG message:
  - The CCB identified in the CCB to be Aborted Pointer field is an established I_T_L_Q nexus I/O process.

If a contingent allegiance condition is in effect for the specified Execute I/O Request CCB (e.g., a SCSI status has been returned from the Logical Unit of CHECK CONDITION or COMMAND TERMINATED), then the EXECUTE SCSI I/O REQUEST CCB shall complete normally as specified by this standard. If autosense is specified for the EXECUTE SCSI I/O REQUEST CCB then the SIM/HBA shall retrieve the autosense data.

This request does not necessarily result in an ABORT message or an ABORT TAG message being issued over the SCSI Bus if the CCB identified is not an established I/O process.

The specified Execute I/O Request CCB may be in one of the following states which ascertains the ultimate results of the Abort SCSI Command function.

- Not contained within the SIM/HBA queues for the addressed Logical Unit.
  - The SIM/HBA shall not be responsible for any further processing for the specified CCB.

- Contained within the SIM/HBA queues for the addressed Logical Unit, but is not an I/O process within the Logical Unit.
  - The SIM/HBA shall set the CAM Status field in the specified CCB to Request Aborted by Host and return the CCB by the mechanisms specified within the CCB.

- Contained within the SIM/HBA queues for the addressed Logical Unit, and is an I/O process within the Logical Unit.
  - The SIM/HBA detects a SCSI bus phase other than BUS FREE in response to the initiators ABORT or ABORT TAG message. The SIM/HBA shall set the CAM Status field for the specified CCB to Unable to Abort Request and shall complete processing for the specified CCB as specified in this standard.
  - The SIM/HBA detects a SCSI bus phase of BUS FREE in response to the initiators ABORT or ABORT TAG message. The SIM/HBA shall set the CAM Status field for the specified CCB to Request Aborted by Host and shall complete processing for the specified CCB as specified in this standard.

Table 14 - ABORT SCSI COMMAND CCB

Size	Dir	Abort SCSI command
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID
1	O	Target ID
1	O	LUN
4	O	CAM Flags
4	O	CCB to be Aborted Pointer

This function shall return a CAM Status other than Request in Progress. The CAM Status shall be one of the following:

- Request Completed without Error indicates that the Path ID is valid.
- Invalid Path ID indicates that the Path ID is invalid.

9.3.2 Reset SCSI bus

This function is used to reset the specified SCSI bus. This function should not be used in normal operation. This request shall always result in the SCSI RST signal being asserted (see Clause 7.5).

Table 15 - RESET SCSI BUS CCB

Size	Dir	Reset SCSI bus
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID
1	O	Target ID
1	O	LUN
4	O	CAM Flags

This function shall return a CAM Status other than Request in Progress. The CAM Status shall be one of the following:

- Request Completed without Error indicates that the Path ID is valid.
- Invalid Path ID indicates that the Path ID is invalid.

The actual failure or success of the Reset SCSI Bus function is indicated by the asynchronous callback information.

9.3.3 Reset SCSI device

This function is used to reset the specified SCSI target. This function should not be used in normal operation. This request shall always result in a BUS DEVICE RESET message being issued over the SCSI Bus (see Clause 7.5).

Table 16 - RESET SCSI DEVICE CCB

Size	Dir	Reset SCSI device
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID
1	O	Target ID
1	O	LUN
4	O	CAM Flags

This function shall return a CAM Status other than Request in Progress. The CAM Status shall be one of the following:

- Request Completed without Error indicates that the Path ID is valid.
- Invalid Path ID indicates that the Path ID is invalid.

The actual failure or success of the Reset SCSI Device function is indicated by the asynchronous callback information.

9.3.4 Terminate I/O process

This function requests that a SCSI I/O request be terminated by identifying the CCB associated with the request. It should be issued on any I/O request that has not completed that the driver expects to terminate. The SIM/HBA shall issue an TERMINATE I/O PROCESS message on the SCSI bus based on one of the following conditions:

- The CCB identified in the CCB to be Terminated Pointer field is an established I_T_L nexus I/O process (untagged command currently active).

- The CCB identified in the CCB to be Terminated Pointer field is an established I_T_L_Q nexus I/O process.

If a contingent allegiance condition is in effect for the specified Execute I/O Request CCB (e.g., a SCSI status has been returned from the Logical Unit of CHECK CONDITION or COMMAND TERMINATED), then the EXECUTE SCSI I/O REQUEST CCB shall complete normally as specified by this standard. If autosense is specified for the EXECUTE SCSI I/O REQUEST CCB then the SIM/HBA shall retrieve the autosense data.

This request does not necessarily result in an TERMINATE I/O PROCESS message being issued over the SCSI Bus if the CCB identified is not an established I/O process.

Table 17 - TERMINATE I/O PROCESS CCB

Size	Dir	Terminate I/O process
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID
1	O	Target ID
1	O	LUN
4	O	CAM Flags
4	O	CCB to be Terminated Pointer

This function shall return a CAM Status other than Request in Progress. The CAM Status shall be one of the following:

- Request Completed without Error indicates that the Path ID is valid.
- Invalid Path ID indicates that the Path ID is invalid.

The specified Execute I/O Request CCB may be in one of the following states which ascertains the ultimate results of the Terminate I/O Process function.

- Not contained within the SIM/HBA queues for the addressed Logical Unit.
  - The SIM/HBA shall not be responsible for any further processing for the specified CCB.

- Contained within the SIM/HBA queues for the addressed Logical Unit, but is not an I/O process within the Logical Unit.
  - The SIM/HBA shall set the CAM Status field in the specified CCB to Terminate I/O Process and return the CCB by the mechanisms specified within the CCB.

- Contained within the SIM/HBA queues for the addressed Logical Unit, and is an I/O process within the Logical Unit.
  - The SIM/HBA receives a MESSAGE REJECT message in response to the initiators TERMINATE I/O PROCESS message. The SIM/HBA shall set the CAM Status field for the specified CCB to Unable to TERMINATE I/O Process when the I/O Process is completed as specified in ANSI X3.131-1994 and shall complete processing for the specified CCB as specified in this standard.
  - The SIM/HBA receives a SCSI-2 status byte other than COMMAND TERMINATED for the specified CCB. The SIM/HBA shall set the CAM Status field for the specified CCB to Unable to TERMINATE I/O Process when the I/O Process is completed as specified in ANSI X3.131-1994 and shall complete processing for the specified CCB as specified in this standard.
  - The SIM/HBA receives a SCSI-2 status byte of COMMAND TERMINATED for the specified CCB. The SIM/HBA shall set the CAM Status field to Request Completed with Error when the I/O Process is completed as specified in ANSI X3.131-1994 and shall complete processing for the specified CCB as specified in this standard (e.g., Autosense mechanisms).

10 CAM control block to request I/O

Peripheral drivers should make all of their SCSI I/O requests using this CCB, which is designed to take advantage of all features of SCSI that can be provided by virtually any HBA/SIM combination. The CCB is common in format and structure for the following function codes:

- Execute SCSI I/O (see Clause 10 and Table 20 for further information)
- Execute Target I/O (see Clause 11 and Table 23 for further information)
- Accept Target I/O (see Clause 11 and Table 27 for further information)
- Continue Target I/O (see Clause 11 and Table 28 for further information)
- Execute Engine Request (see Clause 12 and Table 30 for further information)

10.1 Field descriptions for CAM control blocks to request I/O

10.1.1 Autosense Residual Length

This field contains the difference in twos complement form of the number of autosense data bytes transferred by the HBA compared with the number of autosense bytes requested by the CCB. This is calculated by the total number of autosense bytes requested to be transferred by the CCB minus the actual number of autosense bytes transferred by the HBA.

10.1.2 Callback on completion

See Clause 7.3.2 for callback on completion of a queued CCB.

10.1.3 CAM flags

This field contains bit settings as described in table 18 to indicate special handling of the requested function.

Table 18 - CAM flags - Part 1 of 2

Size	Bit	CAM flags
1	7-6	Direction

Bit 7	Bit 6	Direction
0	0	Reserved
0	1	Data in (e.g., Read from Logical Unit)
1	0	Data out (e.g., Write to Logical Unit)
1	1	No data transfer

	5	1=Disable Autosense
	4	1=Scatter/Gather
	3	1=Disable Callback on Comp
	2	1=Linked CDB
	1	1=Tagged Queue Action Enable
	0	1=CDB is a Pointer
1	7	1=Disable Disconnect
	# 6	1=Initiate Synchronous Transfers
	# 5	1=Disable Synchronous Transfers
	4	SIM Queue Priority 1=Priority Insertion 0=Normal (tail insertion)
	# 3	SIM Queue Freeze
	# 2	SIM Queue Freeze Disable

Bit 3	Bit 2	Queue Action
0	0	SIM Queue Frozen if CAM Status not Request Complete w/o Error
0	1	SIM Queue not Frozen for all CAM Statuses
1	0	SIM Queue Frozen when CCB completes
1	1	Invalid setting of CAM Flags

	1	Engine Synchronize
	0	Reserved
1	7	SG List/Data 0=host 1=engine
	6	CDB Pointer 0=VA 1=PA
	5	SG List/Data 0=VA 1=PA
	4	Sense Buffer 0=VA 1=PA
	3	Message Buffer 0=VA 1=PA
	2	Next CCB 0=VA 1=PA
	1	Callback on Comp 0=VA 1=PA
	0	SG List Pointers 0=VA 1=PA

These bits are mutually exclusive

Table 18 - CAM Flags - Part 2 of 2

Size	Bit	CAM flags		
1	7-0	Target mode-specific CAM Flags		
		Bit	**Host Target**	**Phase Cognizant**
		7	Send Status	Data Buffer Valid
		6	Disconnects Mandatory	Status Buffer Valid
		5	Terminate I/O	Message Buffer Valid
		4	Reserved	Reserved
		3	0 = Host Target Mode	1 = Phase Cognizant Mode
		2	Reserved	1 = Target CCB Available
		1	Reserved	1 = Disable AutoDisconnect
		0	Reserved	1 = Disable AutoSave/Restore

10.1.3.1 Byte 1 bits

7-6 Direction - These encoded bits identify the expected direction of data movement during data transfer. In the Execute Engine Request CCB they have different meaning (See Clause 12.2).
- a setting of 01 indicates a read operation (data transfer from target to initiator).
- a setting of 10 indicates a write operation (data transfer from initiator to target).
- a setting of 11 indicates there is to be no data transfer.

5 Disable Autosense - When set to 1, this bit disables autosense.
NOTE: It is expected that Autosense will be mandatory in future versions of CAM and this bit may be reserved.

4 Scatter/Gather - When set to 1, this bit indicates that data is not to be transferred to/from a single location in memory but to/from several. In this case the Data Buffer Pointer refers to a list of addresses and lengths in bytes at each address to which the data is to be transferred. The format of the SG List is defined in table 19.

Table 19 - Scatter Gather List

Size	
4	Data Address 1
4	Data Length 1
4	Data Address 2
4	Data Length 2
	:
4	Data Address n
4	Data Length n

3 Disable Callback on Completion - When set to 1, the peripheral driver does not want the SIM to callback automatically when the request is completed. This implies that the caller is polling for a CAM Status other than Request in Progress status, which indicates completion of the request.

2 Linked CDB - When set to 1, this CDB is a SCSI linked command. If this bit is set, then the Control field in the CDB shall have bit 0=1. If not, the results are unpredictable. See Clause 10.3 for further information.

1 Tag Queue Action Enable - When set to a 1, the SCSI CDB contained or pointed to within the SCSI I/O REQUEST CCB shall have Tag Queuing attributes. See Clause 10.1.21 for further information.

0 CDB is a Pointer - When set to a 1, the first four bytes of the CDB field shall contain a pointer to the location of the CDB.

10.1.3.2 Byte 2 bits

7 Disable Disconnect - When set to a 1, the disconnect capability of SCSI is disabled. The default of 0 sets bit 6=1 in the SCSI IDENTIFY message.

6 Initiate Synchronous Transfers - When set to a 1, indicates the SIM shall negotiate for the best transfer parameters it is capable of with the target, and wherever possible execute the negotiated transfer parameters (synchronous, fast, wide transfers).

5 Disable Synchronous Transfers - When set to a 1, indicates the SIM shall negotiate for the least transfer parameters (asynchronous, narrow transfers) if the SIM previously negotiated synchronous. If unable to negotiate synchronous (best transfer parameters) or negotiation has not yet been attempted, the SIM shall not initiate negotiation.

4 SIM Queue Priority - When set to a 1, the SIM shall place this CCB ahead of all CCB operations with normal priority sent to the Logical Unit and at the tail of the Logical Unit's (FIFO order) priority internal queue.

3 SIM Queue Freeze - When set to a 1, the SIM shall place its internal Logical Unit queue into the frozen state. Upon callback, the CAM Status for this CCB shall have the SIM Queue Freeze flag set. This bit should only be set for SIM error recovery and should be used in conjunction with the SIM Queue Priority bit and the release SIM queue command.

2 SIM Queue Freeze Disable - When set to a 1, the SIM queue freeze mechanism shall be disabled (i.e., the SIM queue shall not be frozen for the Logical Unit addressed in this CCB in the event of a CAM Status other than Request Complete without Error).

1 Engine Synchronize - This bit is used in conjunction with the Direction in or out settings to flush any residual bits before terminating engine processing (see Clause 12 for further information).

0 Reserved

10.1.3.3 Byte 3 bits

The Pointer fields are set up to have one characteristic. If a bit is set to 1 it shall indicate the pointer field described contains a Physical Address. If set to 0 it shall indicate the pointer contains a Virtual Address. If the SIM needs an address in a different form to that provided, it shall be converted by the SIM (using OSD facilities) and stored in Private Data. See Clause 12.2 for a description of bit 7.

10.1.3.4 Byte 4 bits for phase-cognizant mode

The Phase-Cognizant mode only flags are only active on ENABLE LUN or EXECUTE TARGET I/O CCBs. See Clause 11.2 for complete details.

7-5 The buffer valid bits identify which buffers have contents. In the event that more than one bit is set, they shall be transferred in the sequence of data buffer, status, message buffer.

3 Phase-Cognizant Mode - If target operations are supported, when set to 1, the SIM shall operate in Phase-Cognizant Mode, otherwise it shall operate in Host Target Mode.

2 Target CCB Available - When set to 1, this bit indicates that the XPT/SIM can use this CCB to process this request. A value of 0 indicates that this CCB is not available to the XPT/SIM.

1 Autodisconnect - When set to 1, this bit disables autodisconnect. The default of 0 causes the XPT/SIM to automatically disconnect, if the IDENTIFY message indicates discpriv is set.

0 Autosave - When set to 1, this bit disables autosave feature for Phase-Cognizant Mode. The default of 0 causes the XPT/SIM to automatically send a SAVE DATA POINTER message on an autodisconnect.

10.1.3.5 Byte 4 bits for host target mode

The Host Target Mode flags are shall only be valid for the ENABLE LUN, ACCEPT TARGET I/O and CONTINUE TARGET I/O CCBs. See Clause 11.3 for complete details.

7 Send Status - When set to a 1, this bit directs the SIM/HBA that it shall go to status phase after data phase (if there is a data phase for this CCB) and send the SCSI status byte contained within this CCB.

6 Disconnects Mandatory - When set to a 1, this bit directs the SIM/HBA that it shall disconnect from the SCSI bus for each CCB processed for the enabled LUN.

5 Terminate I/O - When set to a 1, this bit informs the SIM/HBA that the Host Target Mode peripheral driver is supporting the TERMINATE I/O PROCESS SCSI message.

3 Phase-Cognizant Mode - If target operations are supported, when set to 1, the SIM shall operate in Phase-Cognizant Mode, otherwise it shall operate in Host Target Mode.

10.1.4 CDB

This field either contains the SCSI CDB (command descriptor block), or a pointer to the CDB, to be dispatched.

10.1.5 CDB length

For EXECUTE SCSI I/O REQUEST CCBs this field shall contain the length in bytes of the CDB. For ACCEPT TARGET I/O CCBs and EXECUTE TARGET I/O CCBs this field shall contain the length in bytes of the buffer for CDB placement.

10.1.6 Data transfer length

This field contains the length in bytes of the data to be transferred.

10.1.7 Function code

This field shall contain one of the following function codes:
- Execute SCSI I/O
- Execute Engine Request
- Execute Target I/O
- Accept Target I/O
- Continue Target I/O

10.1.8 Initiator ID

This field indicates which SCSI initiator this CCB is in response to. This field is only valid for target mode operation.

10.1.9 Message buffer length (target-only)

This field contains the length in bytes of the field which is to be used to hold message information in the event that the peripheral drivers needs to issue any MSGs. This field is exclusive to target mode operation (see Clause 11.2.3 for further information).

10.1.10 Message buffer pointer (target-only)

This field contains a pointer to a buffer containing messages. This pointer is only valid for use in target mode (see Clause 11.2.3 for further information).

10.1.11 Next CCB pointer

This field contains a pointer to the next command block in a chain of command blocks. A value of 0 indicates the last command block on the chain. This field is used for linking commands.

10.1.12 Number of scatter/gather entries

This field contains the number of entries in the SG List.

10.1.13 Peripheral driver pointer

This field contains a pointer which is for the exclusive use of the peripheral driver, the use of which is not defined by this standard.

10.1.14 Private data

This field is used to contain whatever fields the SIM module needs to execute the request. As such it constitutes a scratchpad of working space needed by the SIM. The size of this area is an OSD as it may differ between SIMs, by environment or by vendor implementation. The XPT shall use the Path Inquiry function to obtain the minimum required size of the private data area.

10.1.15 Request mapping information

This field is a pointer to an OSD data structure which is associated with the original I/O request.

10.1.16 Residual length

This field contains the difference in twos complement form of the number of data bytes transferred by the HBA compared with the number of bytes requested by the CCB. This is calculated by the total number of bytes requested to be transferred by the CCB minus the actual number of bytes transferred by the HBA.

10.1.17 SCSI status

This field contains the status byte returned by the SCSI Logical Unit after the command is completed as defined in ANSI X3.131-1994. This field shall be valid for the CAM Status of Request Complete without Error and Request Complete with Error.

10.1.18 Sense info buffer length

This field contains the length in bytes of the buffer which is to be used to hold sense data in the event that a autosense is invoked.

10.1.19 Sense info buffer pointer

This field contains a pointer to the buffer for autosense data.

10.1.20 SG list/data buffer pointer

This field contains a pointer to either the data buffer to be used for the transfer, or to the SG List which contains the list of scatter/gather addresses to be used for the transfer.

10.1.21 Tag ID

This field indicates the SCSI TAG QUEUE ID that this CCB is in response to if tagged queue operation. This field is valid only for Host Target Mode operation.

10.1.22 Tagged queue action

SCSI provides the capability of tagging commands to force execution in a specific sequence, or of letting the target optimize the sequence of execution to improve performance. For a description of the tagged command queuing philosophy see SCSI-2.

When the Tag Queue Action Enable bit in the CAM Flags is set, the CDB issued by the SIM shall be associated with the queue action specified as:

20h = SIMPLE QUEUE TAG request
21h = HEAD of QUEUE TAG request
22h = ORDERED QUEUE TAG request

10.1.23 Timeout value

This field contains the maximum period in seconds that an issued SCSI command request can remain outstanding. If this value is exceeded then the CAM Status shall report the timeout condition. A value of 00h in the CCB means the peripheral driver accepts the SIM default timeout. A value of F...Fh in the CCB specifies an infinite period.The Timeout Value field is on a per CCB basis, and is measured from successful selection to command completion. If the CCB has timed out and the command has not completed (e.g., COMMAND COMPLETE or LINKED COMMAND COMPLETE message has not been received for the CCB), the SIM/HBA shall reselect the addressed Logical Unit and issue an ABORT message or an ABORT TAG message. If the command that timed out is an I_T_L I/O process then an ABORT message shall be issued by the SIM/HBA. If the command that timed out is an I_T_L_Q I/O process then an ABORT TAG message for the identified I_T_L_Q I/O process shall be issued by the SIM/HBA.

10.1.24 VU field

The uses for this field are defined in the SIM vendor specification.

10.1.25 VU flags

The uses for this field are defined in the SIM vendor specification.

10.2 Execute SCSI I/O

This function typically returns a CAM Status of Request in Progress, indicating that the request was queued successfully. Request completion can be ascertained by polling for a CAM status other than Request in Progress or through use of the Callback on Completion field. Polling for completion of a CCB is not recommended. The CCB is defined in table 20.

Table 20 - EXECUTE SCSI I/O REQUEST CCB

Size	Dir	Execute SCSI I/O Request	
4	O	Address of this CCB	
2	O	CAM Control Block Length	
1	O	Function Code	
1	I	CAM Status	
		Connect ID	
1		Reserved	
1	O	Path ID	
1	O	Target ID	
1	O	LUN	
4	O	CAM Flags	
4	O	Peripheral Driver Pointer	
4	O	Next CCB Pointer	
4	O	Request Mapping Information	
4	O	Callback on Completion	
4	O	SG List/Data Buffer Pointer	
4	O	Data Transfer Length	
4	O	Sense Info Buffer Pointer	
1	O	Sense Info Buffer Length	
1	O	CDB Length	
2	O	Number of Scatter/Gather Entries	
4		VU field	
1	I	SCSI Status	
1	I	Autosense Residual Length	
2		Reserved	
4	I	Residual Length	
12	O	CDB	
4	O	Timeout Value	
4		Message Buffer Pointer	(target-only)
2		Message Buffer Length	(target-only)
2	O	VU flags	
1	O	Tag Queue Action	
1		Tag ID	(target only)
1		Initiator ID	(target only)
1		Reserved	
n	O	Private Data	

The final CAM Status shall be one of the following:

- Request Completed without Error: the request has completed and no error condition was encountered.
- Request Aborted by Host: the request was aborted by the SIM/HBA as instructed by the peripheral driver.
- Unable to Abort Request: the SIM was unable to abort the request as instructed by the peripheral driver.
- Request Completed with Error: the request has completed and an error condition was encountered.
- CAM Busy: CAM unable to accept request at this time.
- Invalid Request: the request has been rejected because it is invalid.
- Invalid Path ID: indicates that the Path ID is invalid.
- Unable to Terminate I/O Process: the SIM was unable to terminate the request as instructed by the peripheral driver.
- Target Selection Timeout: The target failed to respond to selection.
- Command Timeout: the specified command did not complete within the timer value specified in the CCB. Prior to reporting this status the SIM/HBA shall ensure the command is no longer active in the target.

- Message Reject Received: The SIM/HBA received a SCSI MESSAGE REJECT message.
- SCSI Bus Reset Sent/Received: The SCSI operation was terminated at some point because the SCSI bus was reset.
- Uncorrectable Parity Error Detected: An uncorrectable SCSI bus parity error was detected.
- Autosense Request Sense Command Failed: The SIM/HBA attempted to obtain sense data and failed.
- No HBA Detected: HBA no longer responding to SIM (assumed to be a hardware problem).
- Data Overrun: target transferred more data bytes than peripheral driver indicated in the CCB.
- Unexpected Bus Free: an unexpected bus free condition occurred.
- Target Bus Phase Sequence Failure: the Logical Unit failed to operate in compliance with ANSI X3.131-1994.
- CCB Length Inadequate: more private data area is required in the CCB (refer to Clause 9.2.3 for further clarification) .
- Cannot Provide Requested Capability: resources are not available to provide the capability requested in the CAM Flags.
- Bus Device Reset Sent: this CCB was terminated because a BUS DEVICE RESET message was sent to the target.
- Terminate I/O Process: this CCB terminated due to a Terminate I/O Process Request CCB was received by the SIM/HBA and the CCB was not an I/O Process within the Logical Unit.
- Unrecoverable Host Bus Adaptor Error: this CCB was terminated because of a hardware error detected by the HBA. The error does not indicate a SCSI bus problem but an error within the HBA or host.
- SCSI Bus Reset Denied: this CCB was not accepted by the SIM/HBA due to a bus hung condition. This CAM Status is a result of an intelligent HBA detecting a hung SCSI bus and requesting permission from the controlling SIM software to perform a SCSI bus reset. The permission to perform a SCSI bus reset was denied by the SIM.

  Note: This CAM Status is forecasted as being deleted in future versions of CAM due to the following reasons:
  - An intelligent HBA that reports (in a SIM/HBA specific manner) a hung SCSI bus to the controlling SIM software is always be granted the permission to reset the SCSI bus.
  - Multiple SCSI bus resets could occur to clear the condition if the decision to reset the bus is left to multiple peripheral drivers.

10.3 Command linking (optional)

The SIM/HBA supports SCSI's ability to link commands in order to guarantee the sequential execution of several requests. This function requires that both the SIM/HBA and the involved Logical Unit support the SCSI link capability. The SIM/HBA shall indicate its support for command linking by setting the Linked Commands bit in response to a PATH INQUIRY CCB (see Clause 9.2.3 Path Inquiry for further details).

To utilize linking, a chain of CCBs is built with the next CCB pointer being used to link the CCBs together. The CAM Linked CDB flag bit shall be set in all CCBs but the last in the chain. The first CCB in the linked list of CCBs shall have a valid Callback on Completion field set and the CAM Flag of Disable Callback on completion cleared. When a SCSI Logical Unit returns the LINKED COMMAND COMPLETE message or LINKED COMMAND COMPLETE (WITH FLAG) message, the next CCB is processed, and its associated CDB is dispatched. Any SCSI status other than INTERMEDIATE or INTERMEDIATE CONDITION MET returned by the Logical Unit on a linked command shall break the chain. The SIM/HBA shall callback the peripheral driver using the first CCB's Callback on Completion field when the linked list of CCBs is completed or when the chain is broken.

The peripheral driver shall:

- Build a valid list of EXECUTE I/O CCBs with the CAM Linked CDB flag bit set except for the last CCB in the linked CCB chain.

  Note: The peripheral driver is responsible for the correct settings of the Flag and Link bits in the control field of all CDBs within CCBs.
- Call xpt_action() with the address of the first CCB as the argument.
- Wait for completion of the linked CCB list.
- On callback ascertain completion status by starting with the first CCB in the linked list and examining the CAM Status field, then proceeding to the next in the list. The following CCB statuses shall be used to ascertain completion of a individual CCB and the list.

- The CAM Status of Request completed without Error indicates that this CCB completed successfully and if the CAM Linked CDB flag bit is clear the CCB linked list completed without error.
- The CAM Status of Request In Progress indicates the chain was broken and this CDB contained within this CCB was not issued to the Logical Unit.
- The CAM Status of Request Completed with Error indicates that the Logical Unit sent a SCSI status other than INTERMEDIATE or INTERMEDIATE CONDITION MET for this CCB.
- All other CAM Statuses indicate that another CAM condition occurred while processing this CCB and the chain was broken at this point.

The peripheral driver may:
- Monitor chain CCB processing by examining the CAM Status field of each CCB within the list but shall not attempt to modify any CCB within the list until callback on completion by the SIM/HBA.
- Abort the linked CCB chain by issuing an Abort SCSI Command function or Terminate I/O Process Request function to the first CCB within the list.

The SIM/HBA shall for linked CCB lists:
- Validate each CCB within the list (e.g. the first CCB in the linked list of CCBs shall have a valid Callback on Completion field set and the CAM flag of Disable Callback on completion cleared and the CAM Linked CDB flag bit shall be set in all CCBs but the last in the chain).
- Establish an I/O Process for the CCB list (e.g. issue the first CCB then proceeding to the next CCB when a LINKED COMMAND COMPLETE message or a LINKED COMMAND COMPLETE (WITH FLAG) message) is received.
- A COMMAND COMPLETE message for the any CCB but the last CCB shall break the chain.
- A COMMAND COMPLETE message for the last CCB shall indicate successful completion of the chain.
- When each CCB within the chain completes, set all appropriate fields within the CCB.
- When the chained CCB list is completed or broken, callback the peripheral driver using the first CCB Callback on Completion field.
- If the peripheral driver terminates or aborts the first CCB within the chain:
  - If the CCB chain to be aborted or terminated has a current I/O process, it shall abort or terminate the CCB as specified by this standard and break the chain (e.g., the current I/O process CCB has a CAM Status of Request Aborted by Host or Terminate I/O Process).
  - If the CCB chain to be aborted or terminated does not have a current I/O Process, it shall break the chain.
  - If the CCB specified for a CCB chain is not the first CCB of a chain, it shall ignore the request.
  - The SIM/HBA shall ensure that the SCSI bus and the Logical Unit are not left in a hung state as specified by ANSI X3.131-1994.

11 Target mode (optional)

The Target Mode functionality causes the HBA associated with the specified SCSI bus to be set up so that it may be selected as a target. The Target Mode model allows a peripheral driver to mimic any SCSI device type.

If a Target Mode function is specified by a CCB and this functionality is not provided by a particular SIM implementation, then a CAM Status of Function Not Implemented shall be returned in the CCB.

11.1 Target mode overview

There are two different modes of target operation, either or both of which may be supported by the SIM/HBA as defined by the Target Mode Support flags in the PATH INQUIRY CCB:
- Phase-Cognizant Mode
- Host Target Mode

For both Phase-Cognizant Mode and Host Target Mode the CDB group codes of 6 and 7 (vendor unique) shall only be of one size for each group code (e,g., group code 6 having a size of 20 and group code 7 having a size of 15).

Phase-Cognizant mode permits a target peripheral driver tight control over what takes place when a SCSI command is received by the SIM. When a Phase-Cognizant application registers itself and a command is received, the XPT/SIM does an immediate Callback on Completion after placing the SCSI command in an available CCB. The Phase-Cognizant peripheral driver is responsible for setting up data, message, status fields, and CAM Flags in the CCB. It then reissues the CCB with an Execute Target I/O function code so that the XPT/SIM knows which phase it should execute. The "callback-reissue CCB" cycle may happen multiple times before a command completes execution.

In summary, Phase-Cognizant peripheral drivers get a callback immediately after the SCSI command block is received and they are expected to instruct the XPT/SIM which phases to go through to perform the command.

Host Target Mode permits a peripheral driver to register itself as a LUN and provide a set of one or more ACCEPT TARGET I/O CCBs that the SIM/HBA can use for Target Mode command processing. In this mode, when the adapter is selected and the XPT/SIM receives an IDENTIFY message for a LUN that has registered as a Host Target LUN, the SIM/HBA may accept any target mode command, based on conditions specified in this document. Using one of the available ACCEPT TARGET I/O CCBs, the SIM/HBA shall pass the received Host Target mode command to the Peripheral Driver CDB Received Completion callback function. The peripheral driver shall interpret the command and based on its internal knowledge shall decide how to respond. Response shall be in the form of one or more CONTINUE TARGET I/O CCBs to the SIM/HBA. The SIM/HBA shall callback as specified by the peripheral driver for each completed CONTINUE TARGET I/O CCB.

Some SCSI bus message processing and event notification is handled both in the SIM/HBA and the Host Target Mode peripheral driver. An example of this is the ABORT message. Other messages are handled transparently by the SIM/HBA. An example of this is the SYNCHRONOUS DATA TRANSFER REQUEST message. For optional messages that require notification from the SIM/HBA, the peripheral driver decides which optional messages it shall support. If the Host Target Mode peripheral driver has indicated that the message is not to be supported, the SIM/HBA shall reject the message. On receipt of a supported target message that is not handled transparently, the SIM/HBA shall immediately notify the Host Target Mode peripheral driver using the mechanisms provided by the IMMEDIATE NOTIFY CCB. The IMMEDIATE NOTIFY CCB ownership shall always be with the SIM/HBA until the LUN is no longer enabled.

In summary, Host Target Mode peripheral drivers can be called back multiple times for a command received by the SIM/HBA, once for each command received and an additional number depending on the command and how the Host Target Mode peripheral driver has been implemented. The model allows all phase handling and SCSI command processing nuances to be performed by the SIM/HBA but allows the peripheral driver enough functionality for emulated device control.

11.2 Phase-cognizant mode

The following SCSI-2 functionality is not supported by Phase-Cognizant mode:
- Command Tagged queuing

The following SCSI-2 messages shall be handled transparently by the SIM/HBA:
- ABORT TAG
- CLEAR QUEUE
- COMMAND COMPLETE
- DISCONNECT
- IDENTIFY
- MESSAGE PARITY ERROR

- MESSAGE REJECT (for the specified conditions in Clauses 11.2 through 11.2.5)
- NO OPERATION
- Queue Tag Messages
- SAVE DATA POINTER (for the specified conditions in Clauses 11.2 through 11.2.5)
- SYNCHRONOUS DATA TRANSFER REQUEST
- WIDE DATA TRANSFER REQUEST

The following SCSI-2 messages shall be received by SIM/HBA and provided to the target peripheral driver by the mechanisms specified by this standard:
- ABORT
- BUS DEVICE RESET
- TERMINATE I/O PROCESS
- INITIATOR DETECTED ERROR

For the following messages received by the SIM/HBA for an enabled Phase-Cognizant LUN, the SIM/HBA shall issue a MESSAGE REJECT as a response:
- ABORT TAG
- CLEAR QUEUE
- All Queue Tag Messages

11.2.1 Enable LUN for phase cognizant mode

The specified Target ID shall match that returned by the HBA Path Inquiry Function for the specified SCSI bus. The specified LUN is the one enabled for selection. If the HBA is to respond as an additional LUN, another Enable LUN is required.

In addition to providing a hook into the target peripheral driver, this function is intended to provide an area that the SIM/HBA can use as working space when the HBA is selected.

Table 21 lists the fields of the ENABLE LUN CCB. The absence of any entry under the "Dir" heading indicates that this field is not being used for the Enable LUN function.

Table 21 - ENABLE LUN CCB for phase cognizant mode

Size	Dir	Enable LUN
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code (Enable LUN)
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID
1	O	Target ID
1	O	LUN
4	O	CAM Flags
2	O	Group 6 Vendor Unique CDB Lengths
2	O	Group 7 Vendor Unique CDB Lengths
4		Pointer to Immediate Notify CCB List
4		Number of Immediate Notify CCBs
4	O	Pointer to Target CCB list
2	O	Number of Target CCBs
2		Reserved
n		SIM private

If the Number of Target CCBs is zero, then Target Mode is disabled, otherwise the Pointer to Target CCB List field refers to a list of addresses of CCBs to which the data is to be transferred (see Table 22).

Table 22 - CCB List

Size	Target CCB list
4	CCB Address 1
4	CCB Address 2
	:
4	CCB Address n

Target CCB(s) shall be EXECUTE SCSI I/O CCB(s) and the target peripheral driver shall not alter the list until a successful disable of the LUN is accomplished.

The SIM/HBA shall place the pointer to the list of CCBs in a list until the specified Target ID and LUN is disabled on the SCSI bus specified by the Path ID field. While the LUN is enabled for Phase-Cognizant Mode operation, the CAM Status field of each Target CCB shall be set to Request in Progress. The target peripheral driver is required to poll the CAM Status field of the Target CCB or provide a Completion Callback routine in the Target CCB.

The CCB(s) provided to the SIM/HBA by the ENABLE LUN CCB shall only be used for CDB reception and continuation of an I/O process (linked commands). The target peripheral driver shall use other allocated EXECUTE TARGET I/O CCBs to receive/transmit data, messages, and status to the selecting initiator. It is recommended that the target peripheral driver supply at least 1 CCB per initiator it expects to communicate with when the Phase-Cognizant LUN is enabled.

The SIM/HBA shall keep an indication of whether a single CCB or list of CCBs was provided on the ENABLE LUN service.

The SIM/HBA shall set the following in each Target CCB when they are first provided:
- CAM Status to Request in Progress.
- CAM Flags shall be set to the values of the CAM Flags in the ENABLE LUN CCB.
- CAM Flags shall be set with the Target CCB Available bit.

Within the Target CCB(s) provided, the following information shall be present and valid:
- CDB field is valid for the Command Blocks that may be received. That is, either CDBs are embedded in the CCB, or a pointer to a CDB buffer area is provided in the CDB field.
- Timeout Value field shall be set to the infinity value.

If the target peripheral driver supports Vendor Unique CDB(s), then the CDB Length field of the CCB shall reflect the largest supported CDB. If a CDB greater than the size of the CDB field is desired, then the CDB field shall contain a pointer to a CDB buffer.

To disable the selection of a specific LUN, the target peripheral driver performs an Enable LUN with a zero value for the Number of Target CCBs. Upon disabling the specified LUN, the target peripheral driver may free all allocated Target CCBs that were associated with the enabled LUN.

When a CDB is received by a SIM/HBA for a LUN that is not enabled, one of the following sequences shall occur depending on the command received:

A) INQUIRY Command
   If the SIM receives a CDB for the INQUIRY command for a non-enabled LUN, the SIM shall return only byte 0 of the inquiry data set to 23H:
   - The peripheral qualifier is set to 01B indicating the target is capable of supporting a physical device on this logical unit, however the physical device is not currently connected to this LUN.
   - The peripheral device type set to 3H indicating Processor device type.

B) REQUEST SENSE Command
If a REQUEST SENSE command is received for a non-enabled LUN, the SIM shall return sense data in which the sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to LOGICAL UNIT NOT SUPPORTED.

C) All other commands
If a command other than INQUIRY or REQUEST SENSE is received for a non-enabled LUN, the SCSI status returned shall be CHECK CONDITION. Any subsequent REQUEST SENSE command shall behave as in Item B.

The Enable LUN function shall return CAM Status of:
- Request Completed without Error indicates that the Enable LUN was completed successfully.
- LUN Already Enabled indicates that this LUN is already enabled.
- Terminate I/O Process indicates that there is currently a nexus established with an initiator that shall be terminated first.
- Invalid Request indicates that one or more of the CCB(s) supplied is invalid.
- Invalid Path ID indicates that the Path ID is invalid.
- Invalid Target ID indicates that the Target ID does not match that used by the HBA specified by the Path ID field.
- Invalid LUN indicates that the LUN specified is outside the supported range of the SCSI bus.
- Function Not Implemented indicates that Phase Cognizant Target mode is not supported by this SIM/HBA.

11.2.2 I/O process creation for phase cognizant mode

If the HBA is selected with the SCSI bus signal of ATN being false the SIM/HBA shall go BUS FREE. When the HBA is selected, the SIM/HBA automatically sets the HBA to the MESSAGE OUT phase to receive the IDENTIFY, SYNCHRONOUS DATA TRANSFER REQUEST, and other messages that may be sent by the Initiator. The SIM/HBA response to these messages shall be as defined in ANSI X3.131-1994 and this standard.

The SIM/HBA shall maintain an indication (in a vendor unique manner) that an I/O process is present for each enabled Phase-Cognizant LUN on a per initiator basis. The indication is maintained from I/O process creation (CDB received and passed to target peripheral driver) to the COMMAND COMPLETE message for the I/O process or one of the SCSI control messages that terminate I/O process(es) (Refer to ANSI X3.131-1994 for control messages). The indication is maintained for correct coordination of the LUN behavior between the target peripheral driver and the SIM/HBA for the ABORT, TERMINATE I/O PROCESS, and BUS DEVICE RESET messages and bus reset. See Clause 11.2.4 for further details.

If the LUNTAR bit (or any of the reserved bits) of the IDENTIFY message is set to 1, then the SIM/HBA shall send a MESSAGE REJECT message back to the initiator, and go to BUS FREE phase.

The LUN shall be extracted from the IDENTIFY message and the SIM/HBA shall scan the CAM Flags in the CCB(s) provided with Enable LUN. If none of them have the Target CCB Available bit set, the SIM/HBA shall post BUSY status and then send a COMMAND COMPLETE message. Otherwise, the SIM/HBA shall select a CCB with the Target CCB Available flag set. The Initiator ID field shall be set to the ID of the initiator that performed the selection. This field shall be used by the target peripheral driver for subsequent functions, such as reselect, to ascertain the Initiator's ID.

Note: The target peripheral driver should ensure that there are always CCBs with the Target CCB Available bit set especially for linked commands.

If the DiscPriv bit in the IDENTIFY message was set, which results in the Disable Disconnect bit of the CAM Flags being cleared, and the Disable AutoDisconnect bit of the CAM Flags field is cleared, the SIM/HBA shall

automatically disconnect upon receipt of the command block. The disconnect shall be performed before clearing the CCB Available bit in the CAM Flags and the callback of the target peripheral driver.

The Disable Disconnect bit in the CAM Flags field shall be updated to indicate the state of the DiscPriv bit in the IDENTIFY message that was received from the initiator. If the DiscPriv bit was set in the IDENTIFY message, then the Disable Disconnect bit shall be cleared, and vice-versa.

NOTE: The default state of the Disable Disconnect bit in the CAM Flags is cleared, implying that disconnect is enabled.

If an ABORT, TERMINATE I/O PROCESS or BUS DEVICE RESET message was received in the initial MESSAGE OUT Phase the SIM/HBA shall handle these messages as specified in Clause 11.2.4. Once the initial MESSAGE OUT Phase is complete, the SIM/HBA automatically sets the HBA to the Command Out Phase to request the SCSI CDB. After receiving the SCSI CDB bytes, the SIM/HBA shall set the CAM Status field to CAM Status of SCSI CDB received. The SIM/HBA shall clear the CCB Available bit in the CAM Flags and if the CAM flag of Disable Callback on Completion is a zero for this CCB the SIM/HBA shall callback the target peripheral driver.

If the Group Code of the Operation Code of the CDB is Vendor Unique, the SIM/HBA shall transfer the number of CDB bytes specified in the ENABLE LUN CCB for this LUN. The Group Code in the incoming CDB (either 6 or 7) shall select the Vendor Unique CDB size from the ENABLE LUN CCB. If the selected CDB size (specified in the ENABLE LUN CCB) is zero, the SIM/HBA shall transfer only the CDB Operation Code. If the required number of bytes is not transferred or the specified size for this Group Code is zero, then the SIM shall set in the selected CCB the CDB bytes transferred in the area provided and shall set the CAM Status to Invalid CDB. The SIM shall then clear the CCB Available bit in the CAM Flags and if the CAM flag of Disable Callback on Completion is a zero for this CCB the SIM/HBA shall callback the target peripheral driver.

The target peripheral driver for CDB reception on the callback or upon noticing the CCB Available bit is clear for a Target mode CCB shall process the CCB (e.g, ascertain if valid CDB, copying pertinent data from the Target CCB). After processing the CDB from a Target CCB, the target peripheral driver shall set CCB Available in the CAM Flags, which passes the CCB back to the SIM/HBA.

The target peripheral driver shall be responsible for completing all I/O process(es) as defined in ANSI X3.131-1994. The SIM/HBA shall pass a Target CCB to a target peripheral driver with a CAM Status of CDB Received or Invalid CDB. All other events relating to the creation of an I/O process shall be handled transparently unless otherwise specified.

11.2.3 Continuation and completion of an I/O process for phase cognizant mode

The SIM/HBA shall receive an EXECUTE TARGET I/O CCB from the target peripheral driver, by the target peripheral driver calling xpt_action() with the address of the CCB as the argument. The SIM/HBA shall reject the CCB based on the conditions specified in Clauses 11.2.3, 11.2.4, and 11.2.5.

The SIM shall reject any CCB which has a Timeout Value of other than infinity with a CAM Status of Invalid Request.

The SIM/HBA shall perform an automatic reselect if the SIM/HBA had disconnected after the receipt of the CDB, or had disconnected upon completion of a previous Execute Target I/O (within the same I/O process).

If the Data Buffer Valid bit is set, the SIM/HBA shall enter the data phase indicated by the direction bits in the CAM Flags field (i.e., DATA IN or DATA OUT). It shall send/receive data indicated by the CAM Direction Flags to/from the buffer(s) indicated in the CCBs Scatter Gather List or Data Pointer. The CAM Direction flags shall be interpreted as the following:

- Bit 7 = 0 and 6 = 0: Reserved
- Bit 7 = 0 and 6 = 1: Data in to initiator (data from target to initiator)

- Bit 7 = 1 and 6 = 0: Data out from initiator (data from initiator to target)
- Bit 7 = 1 and 6 = 1: No data transfer

The SIM shall reject any CCB with the Data Buffer Valid bit set and the CAM Direction flags indicating No data transfer or Reserved with a CAM Status of Invalid Request.

If the Status Buffer Valid bit is set, the SIM/HBA shall send the status byte specified in the SCSI Status field to the current initiator and then send the COMMAND COMPLETE message if the Message Buffer Valid bit is cleared and go to BUS FREE phase. If the Status Buffer Valid bit and Message Buffer Valid bit are both set then the SIM/HBA shall send the status byte specified in the SCSI Status field to the current initiator and then send the message contained in the Message buffer. The Message buffer shall contain a single message being one of the following:
- LINKED COMMAND COMPLETE
- LINKED COMMAND COMPLETE (WITH FLAG)

The SIM/HBA shall perform the following after successfully transmitting the LINKED COMMAND COMPLETE or the LINKED COMMAND COMPLETE (WITH FLAG) message:
- Post the required CAM Status in the EXECUTE TARGET I/O CCB and call back the target peripheral if specified.
- If a Target CCB is available to the SIM/HBA, go to COMMAND phase and receive the CDB bytes for the next linked command.
- If a Target CCB is not available to the SIM/HBA, the SIM/HBA shall not change phase and shall wait until a Target CCB is available. Once a Target CCB is available the SIM/HBA shall go to COMMAND phase and receive the CDB bytes for the next linked command.

  Note: A target peripheral driver can hang the SCSI bus if Target CCBs are not available to the SIM/HBA. Care should be taken if linked commands are supported to prevent this situation.

If the Message Buffer Valid bit is set and the Status Buffer Valid bit is clear, the SIM/HBA shall enter the MESSAGE phase and transfer the contents of the Message buffer. The target peripheral driver shall not place a DISCONNECT message into the Message buffer and set the Message Buffer Valid bit. The SIM/HBA shall be able to detect a DISCONNECT message in the Message Buffer and if the SIM/HBA detects this condition the SIM/HBA shall not enter MESSAGE phase to transmit the message. The SIM/HBA shall process all other phases indicated by the EXECUTE TARGET I/O CCB as defined by this standard.

The SIM/HBA shall receive and respond to any messages resulting from ATN being asserted by the initiator, in addition to any messages it sends to the initiator (see Clause 11.2.4 for further details).

The SIM/HBA shall be able to execute all the phases indicated by the Buffer Valid bits of the CAM Flags, within a single invocation of the Execute Target I/O (i.e., if more than one bit is set), the order of execution of the phases shall be data, status, and message.

If either the Status Buffer Valid bit or the Message Buffer Valid bit of the CAM Flags field are set for an invocation of Execute Target I/O, the AutoDisconnect and AutoSave features shall be disabled.

Going to BUS FREE phase or disconnecting from the SCSI bus (e.g., sending a DISCONNECT message) and going to BUS FREE phase shall be governed by the following:
- If a COMMAND COMPLETE message is successfully transmitted on the SCSI bus, the SIM/HBA shall go to BUS FREE phase.
- If the Disable AutoDisconnect bit of the CAM Flags is cleared, and the Disable Disconnect of the CAM Flags bit is cleared, then the SIM/HBA shall disconnect on the completion of a data transfer specified by the Data Buffer Valid bit set. If the Disable AutoSave bit of the CAM Flags is cleared, then the SIM/HBA shall send a SAVE DATA POINTERS message to the initiator prior to sending the DISCONNECT message.

Upon completion of the function(s) specified in the EXECUTE TARGET I/O CCB, the SIM/HBA shall post the required CAM Status in the CCB and call back the target peripheral driver if specified.

Upon the last Execute Target I/O, the target peripheral driver should consider setting the Disable AutoSave bit, which shall disable the sending of the Save Data Pointers.

11.2.4 Non-transparent event handling for phase cognizant mode

When the SIM/HBA receives an ABORT message from an initiator for an enabled Phase-Cognizant LUN it shall perform the following actions:

- Go to BUS FREE phase.
- If an I/O process exists for this I_T_L and EXECUTE TARGET I/O CCB(s) are held by the SIM/HBA for this I_T_L with CAM Status of Request in Progress, the SIM/HBA shall return the CCB(s) to the target peripheral driver with the CAM Status field set to Request Aborted by Host.
- If an I/O process exists for this I_T_L it shall reject all EXECUTE TARGET I/O CCBs received for this I_T_L with a CAM status of Request Aborted by Host set into the CAM Status field and a return status of Request Aborted by Host, until the establishment of a new I/O process for this I_T_L.
- If an I/O process does not exist for this I_T_L then the only action taken is the BUS FREE response to the message.

When the SIM/HBA receives a BUS DEVICE RESET message from an initiator, it shall perform the following actions for each enabled Phase-Cognizant LUN that uses the SCSI bus over which the message was received:

- Go to BUS FREE phase.
- Return all EXECUTE TARGET I/O CCBs with a CAM Status of Bus Device Reset Sent.
- Perform an asynchronous event callback as specified in Clause 7.6.
- Ascertain if a I/O process(es) exists for this target for all enabled Phase-Cognizant LUNs (in a vendor unique manner).
- If an I/O process exists for this target it shall reject all EXECUTE TARGET I/O CCBs sent to an enabled Phase-Cognizant LUN with a CAM status of Bus Device Reset Sent set into the CAM Status field and a return status of Bus Device Reset Sent, until the establishment of a new I/O process for a particular I_T_L.

When the SIM/HBA receives a TERMINATE I/O PROCESS message from an initiator for an enabled Phase-Cognizant LUN it shall perform the following actions:

- If an I/O process exists for this I_T_L and no EXECUTE TARGET I/O CCB(s) are held by the SIM/HBA for this I_T_L.
  - If the IDENTIFY message had the DiscPriv bit set the SIM/HBA shall disconnect from the SCSI bus.
  - The SIM/HBA shall reject the next EXECUTE TARGET I/O CCB received for this I_T_L with a CAM status of Terminate I/O Process set into the CAM Status field and a return status of Terminate I/O Process, all subsequent EXECUTE TARGET I/O CCBs received for the I_T_L shall be processed normally.
- If an I/O process exists for this I_T_L and EXECUTE TARGET I/O CCB(s) are held by SIM/HBA for this I_T_L that have a CAM Status of Request in Progress and SCSI status has not been sent.
  - If the I/O process is the current I/O process the SIM/HBA shall examine the CAM flags to ascertain if disconnecting from the SCSI bus is allowed. If the Disable AutoDisconnect bit of the CAM Flags is cleared, and the Disable Disconnect of the CAM Flags bit is cleared, then the SIM/HBA shall disconnect from the SCSI bus. If a data transfer is specified by the Data Buffer Valid bit set and the Disable AutoSave bit of the CAM Flags is cleared, then the SIM/HBA shall send a SAVE DATA POINTERS message to the initiator prior to sending the DISCONNECT message.
  - The SIM/HBA shall properly set the Residual Length field of the CCB if a data phase is specified and return the CCB(s) to target peripheral driver with a CAM Status of Terminate I/O Process.
  - The SIM/HBA shall not reject any subsequent EXECUTE TARGET I/O CCBs for the I_T_L.
- If an I/O process exists for this I_T_L and an EXECUTE TARGET I/O CCB is held by SIM/HBA for this I_T_L and SCSI status has been sent, the SIM/HBA shall continue normal processing of this CCB.

- If an I/O process does not exist for this I_T_L then the only action taken is the BUS FREE response to the message.

When the SIM/HBA receives an INITIATOR DETECTED ERROR message from an initiator for an enabled Phase-Cognizant LUN it shall perform the following actions:

- If an I/O process exists for this I_T_L and no EXECUTE TARGET I/O CCB(s) are held by the SIM/HBA for this I_T_L.
  - If the IDENTIFY message had the DiscPriv bit set the SIM/HBA shall disconnect from the SCSI bus.
  - If an I/O process exists for this I_T_L it shall reject the next EXECUTE TARGET I/O CCB received for this I_T_L with a CAM status of Initiator Detected Error set into the CAM Status field and a return status of Initiator Detected Error, all subsequent EXECUTE TARGET I/O CCBs received for the I_T_L shall be processed normally.
- If an I/O process exists for this I_T_L and EXECUTE TARGET I/O CCB(s) are held by SIM/HBA for this I_T_L that have a CAM Status of Request in Progress.
  - The SIM/HBA shall examine the CAM flags to ascertain if disconnecting from the SCSI bus is allowed. If the Disable AutoDisconnect bit of the CAM Flags is cleared, and the Disable Disconnect of the CAM Flags bit is cleared, then the SIM/HBA shall disconnect from the SCSI bus.
  - The SIM/HBA shall properly set the Residual Length field of the CCB if a data phase is specified and return the CCB(s) to target peripheral driver with a CAM Status of Initiator Detected Error.
  - The SIM/HBA shall not reject any subsequent EXECUTE TARGET I/O CCBs for the I_T_L.
- If an I/O process does not exist for this I_T_L then the only action taken is the BUS FREE response to the message.

When the SIM/HBA detects a SCSI bus reset, it shall perform the following actions for each enabled Phase-Cognizant LUN that uses the SCSI bus over which the SCSI bus reset was detected:

- Return all EXECUTE TARGET I/O CCBs held by SIM/HBA with a CAM Status of SCSI Bus Reset Sent/Received.
- Perform an asynchronous event callback as specified in Clause 7.6 (e.g., return all EXECUTE TARGET I/O CCBs held by SIM/HBA with a CAM Status of SCSI Bus Reset Sent/Received).
- Ascertain if a I/O process(es) exists for this target for all enabled Phase-Cognizant LUNs (in a vendor unique manner).
- If an I/O process exists for this target it shall reject all EXECUTE TARGET I/O CCBs sent to an enabled Phase-Cognizant LUN with a CAM status of SCSI Bus Reset Sent/Received set into the CAM Status field and a return status of SCSI Bus Reset Sent/Received, until the establishment of a new I/O process for a particular I_T_L.

If the target peripheral driver receives an EXECUTE TARGET I/O CCB with a CAM Status of either Terminate I/O Process or Initiator Detected Error then an I/O process is still in existence for the I_T_L specified by the CCB. It shall be the target peripheral driver's responsibility to complete the I/O process as specified in ANSI X3.131-1994.

11.2.5 EXECUTE TARGET I/O CCB

This function typically returns a CAM Status of Request in Progress, indicating that the request was queued successfully. Request completion can be ascertained by polling for a CAM status other than Request in Progress or through use of the Callback on Completion field. Polling for completion of a CCB is not recommended.

The following lists the fields of the EXECUTE TARGET I/O CCB. No entry under the "Dir" heading indicates that this field is not being used for the Execute Target I/O function.

Table 23 - EXECUTE TARGET I/O CCB

Size	Dir	Execute target I/O
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code (Execute Target I/O)
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID (Bus no. of SIM/HBA)
1	O	Target ID (Target ID of SIM/HBA)
1	O	LUN
4	I/O	CAM Flags
4	O	Peripheral Driver Pointer
4		Next CCB Pointer
4	O	Request Mapping Information (If needed)
4	O	Callback on Completion
4	O	SG List/Data Buffer Pointer
4	O	Data Transfer Length
4		Sense Info Buffer Pointer
4		Sense Info Buffer Length
1	O	CDB Length
2	O	Number of Scatter/Gather Entries
4		VU Field
1	O	SCSI Status
1		Autosense Residual Length
2		Reserved OSD
4	I	Residual Length
12	I/O	CDB
4		Timeout Value
4	O	Message Buffer Pointer
2	O	Message Buffer Length
2	O	VU Flags
1		Tag Queue Action
1		Tag ID
1	I	Initiator ID
1		Reserved
n		Private Data

The final CAM Status shall be one of the following:

- Request Completed without Error: the request has completed and no error condition was encountered.
- Request Aborted by Host: the request was aborted by the SIM/HBA as instructed by an initiator.
- Request Completed with Error: the request has completed and an error condition was encountered.
- CAM Busy: CAM unable to accept request at this time.
- Invalid Request: the request has been rejected because it is invalid.
- Invalid Path ID: indicates that the Path ID is invalid.
- Target Selection Timeout: the specified initiator failed to respond to reselection.
- Message Reject Received: The SIM/HBA received a SCSI MESSAGE REJECT message in response to a message sent contained in the Message Buffer.
- SCSI Bus Reset Sent/Received: The SCSI operation was terminated at some point because the SCSI bus was reset.
- Uncorrectable Parity Error Detected: An uncorrectable SCSI bus parity error was detected.
- No HBA Detected: HBA no longer responding to SIM (assumed to be a hardware problem).
- CCB Length Inadequate: more private data area is required in the CCB (refer to Clause 9.2.3 for further clarification) .
- Cannot Provide Requested Capability: resources are not available to provide the capability requested in the CAM Flags.
- Bus Device Reset Sent: this CCB was terminated because a BUS DEVICE RESET message was sent to the target.

- Terminate I/O Process: this CCB terminated due to a TERMINATE I/O PROCESS message was received by the SIM/HBA for the specified I_T_L.
- Unrecoverable Host Bus Adaptor Error: this CCB was terminated because of a hardware error detected by the HBA. The error does not indicate a SCSI bus problem but an error within the HBA or host.
- Initiator Detected Error: this CCB terminated due to a INITIATOR DETECTED ERROR message was received by the SIM/HBA for the specified I_T_L.
- Invalid CDB: indicates that the SIM/HBA has detected an error condition on reception of a CDB.
- Invalid LUN: indicates that the Logical Unit specified is outside the supported range of the SCSI bus.
- Invalid Target ID indicates that the Target ID does not match that used by the HBA specified by the Path ID field.
- Nexus Not Established: there is currently no connection established between the specified Target ID and target LUN with any initiator.
- Invalid Initiator ID: the initiator ID specified is outside the valid range that is supported.

  NOTE: This status can also be returned if the target tries to reselect an initiator other than the one to which it was previously connected.

- SCSI CDB Received: indicates that the target has been selected and that the SCSI CDB is present in the CCB.
- SCSI Bus Busy: the SIM failed to win arbitration for the SCSI bus during several different bus free phases.

11.3 Host target mode

11.3.1 Host target mode functionality not specified

This standard does not address the following SCSI functionality for Host Target Mode operation:

A) LUNTAR
B) Linked Commands
C) Extended Contingent Allegiance
D) Soft Reset

11.3.2 Host target mode messages

The SCSI messages outlined below are in two main categories for Host Target Mode. The categories are transparent message handling and notification message handling. These 2 main categories can further be divided into mandatory messages and optional messages. Transparent message handling shall mean that the SIM/HBA shall handle the message and that there is no notification of the message to a peripheral driver. Notification message handling shall mean that the SIM/HBA shall receive the message and notify the Host Target Mode peripheral driver as specified by this standard. For notification message handling the SIM/HBA shall maintain certain state information about the message, but how that is accomplished is left to the discretion of the SIM/HBA implementer.

A) Mandatory Transparent Message Handling

   1) COMMAND COMPLETE
   2) IDENTIFY
   3) INITIATOR DETECTED ERROR
   4) MESSAGE PARITY ERROR
   5) MESSAGE REJECT
   6) NO OPERATION

B) Optional Transparent Message Handling at the discretion of SIM/HBA.

   1) DISCONNECT
   2) IGNORE WIDE RESIDUE

3) MODIFY DATA POINTER
4) RESTORE POINTERS
5) SAVE DATA POINTERS
6) SYNCHRONOUS DATA TRANSFER REQUEST
7) WIDE DATA TRANSFER REQUEST

C) Mandatory Message Handling that requires notification to the Host Target Mode peripheral driver.

1) ABORT message
   Host Target Mode peripheral driver shall be notified by the IMMEDIATE NOTIFY CCB mechanism.

2) BUS DEVICE RESET message
   Host Target Mode peripheral driver shall be notified by the Asynchronous Event mechanism.

D) Optional Message Handling that requires notification to the Host Target Mode peripheral driver.

1) ABORT TAG
2) CLEAR QUEUE
3) HEAD OF QUEUE TAG (see 11.3.8)
4) ORDERED QUEUE TAG (see 11.3.8)
5) SIMPLE QUEUE TAG (see 11.3.8)
6) TERMINATE I/O PROCESS

Note: See Clause 11.3.3 (Use of the IMMEDIATE NOTIFY CCB), Clause 11.3.4 (Enable Target Mode LUN for Host Target Mode), and Clause 11.3.8 (ACCEPT TARGET I/O and CONTINUE TARGET I/O CCB Operation) for clarification.

11.3.3 Use of the IMMEDIATE NOTIFY CCB

The IMMEDIATE NOTIFY CCB is for Host Target Mode use only. It is used to notify the Host Target Mode peripheral driver of events detected by the SIM/HBA. The IMMEDIATE NOTIFY CCBs, once passed to the SIM/HBA by the ENABLE LUN CCB, shall be maintained by the SIM/HBA for its exclusive use. Ownership of IMMEDIATE NOTIFY CCB(s) shall not be returned to the Host Target Mode peripheral driver until the successful completion of a DISABLE TARGET MODE LUN. The IMMEDIATE NOTIFY CCB contents shall be valid to the Host Target Mode peripheral driver only at the point where the SIM/HBA does the callback for notification of an event. The Host Target Mode peripheral driver should at that point read its contents to ascertain event and sense data. Upon callback return, the contents of IMMEDIATE NOTIFY CCB shall no longer be considered valid.

The Immediate Notify mechanism from the SIM/HBA has a corresponding Notify Acknowledgement mechanism. The IMMEDIATE NOTIFY CCB shall contain a unique sequence identifier for a Host Target Mode LUN. For each event/message delivered to the Host Target Mode peripheral driver by the callback mechanism, the Host Target Mode peripheral driver shall do the following:

A) Any processing needed to comply with this standard and ANSI X3.131-1994.

B) Issue a NOTIFY ACKNOWLEDGE CCB with the Sequence Identifier field set to the value from Sequence Identifier field of the IMMEDIATE NOTIFY CCB for the event/message being processed.

There shall be a one to one correspondence between the Immediate Notify and the Notify Acknowledgement from the Host Target Mode peripheral driver. For each Immediate Notify for a LUN there shall be a Notify Acknowledgement.

Sequence identifiers shall be unique for each Host Target Mode LUN. There shall not be two identical sequence identifiers in use at the same time for a Host Target Mode LUN. A sequence identifier is "in use" from the time the

peripheral driver callback is invoked until the receipt of the Notify Acknowledgement from the peripheral driver. Sequence identifiers shall be non zero value(s).

If an event/message is detected by the SIM/HBA that requires a notification back to the Host Target Mode peripheral driver and there are no IMMEDIATE NOTIFY CCBs available, the SIM/HBA shall do the following:

A) Record all information about the event/message as specified later in this section, and preserve ordering of the event/message, in FIFO fashion.

B) When an IMMEDIATE NOTIFY CCB becomes available for use the SIM/HBA shall notify the Host Target Mode peripheral driver using the mechanisms specified later in this section.

The ordering of Host Target Mode peripheral driver notification and when the SIM/HBA releases the SCSI bus to the BUS FREE phase is not specified. The change to Bus Free phase and the peripheral driver callback in clause 11.3.3.1 may occur in either order, but both steps are required. In all other cases, the order in which clause 11.3.3.1 lists operations is the order in which those operations shall be performed.

The order in which the SIM/HBA places the Extended message arguments into the IMMEDIATE NOTIFY CCB Message Arguments field for an Extended message is specified. After the Extended message code is received all Extended message bytes received for the Extended message shall be placed into the IMMEDIATE NOTIFY CCB Message Arguments field in ascending order in the order that they were received. The first Extended message argument byte received after the Extended message code shall be placed into the Message Arguments array[0] field. The next Extended message argument byte received shall be placed into the Message Arguments array[1] field.

For the SCSI ABORT and CLEAR QUEUE messages, the order in which the CCBs are returned to the Host Target Mode peripheral driver and the IMMEDIATE NOTIFY CCB callback is done to the driver is specified. The order shall be as follows:

A) All CONTINUE TARGET I/O CCBs

B) IMMEDIATE NOTIFY CCB callback to the Host Target Mode peripheral driver

11.3.3.1 The events/messages that use the immediate notify mechanism

11.3.3.1.1 Sense data preservation where no nexus has been established

There are certain events that require or optionally provide for sense data preservation by the target. These events are specified in ANSI X3.131-1994. If one of these events is detected by the SIM/HBA, the SIM/HBA shall respond as follows for each enabled Host Target Mode LUN:

A) Set the Path ID of an IMMEDIATE NOTIFY CCB to the bus number of this bus and target id of the SIM/HBA.

B) Form the SCSI-2 required 18 bytes of correct sense data for the event and place the sense data in the sense buffer provided in the IMMEDIATE NOTIFY CCB. It shall not be considered an error if the sense buffer bytes are zero, indicating NOSENSE. However the Host Target Mode peripheral driver shall not be required to preserve NOSENSE data.

C) Set the CAM Status to Nexus Not Established in the IMMEDIATE NOTIFY CCB indicating that a nexus was not yet established.

D) Indicate in the CAM Status field of the IMMEDIATE NOTIFY CCB that Autosense is valid. Autosense valid indicates to the Host Target Mode peripheral driver that the sense buffer data is valid and can be copied.

The Host Target Mode peripheral driver shall save the copied sense data for use if the next received command is request sense.

E) Form a unique sequence identifier and place it in the IMMEDIATE NOTIFY CCB's Sequence Identifier field.

F) Transition the SCSI Bus to BUS FREE. Callback the Host Target Mode peripheral driver using the callback notify field in the IMMEDIATE NOTIFY CCB. The exact order of these two operations is not specified.

G) For all initial connections, the SIM/HBA for this enabled Host Target Mode LUN shall transparently respond with a SCSI Status of BUSY until all events are acknowledged by the Host Target Mode peripheral driver.

H) All CONTINUE TARGET I/O CCBs received by the SIM/HBA for this enabled Host Target Mode LUN shall be rejected until all events are acknowledged by the Host Target Mode Peripheral driver. The rejected CCBs shall have:

1) A CAM Status field set to Unacknowledged Event by Host

2) A return status of Unacknowledged Event by Host.

Acknowledgment of this event by the Host Target Mode peripheral driver shall be accomplished when the SIM/HBA receives a NOTIFY ACKNOWLEDGE CCB for this LUN with the Sequence Identifier field equal to the sequence identifier of the IMMEDIATE NOTIFY CCB for this event.

The Host Target Mode peripheral driver shall be responsible for preserving Contingent Allegiance conditions.

11.3.3.1.2 Mandatory messages

This clause describes the handling of mandatory messages that are handled jointly between the SIM/HBA and the corresponding Host Target Mode peripheral driver via the IMMEDIATE NOTIFY CCB.

Note: The ABORT message is the only mandatory message that is handled with the IMMEDIATE NOTIFY CCB. BUS DEVICE RESET messages are handled by the Asynchronous Event mechanism (see Clause 11.3.12.2).

11.3.3.1.2.1 ABORT message

When an ABORT message is received for an enabled Host Target Mode LUN by the SIM/HBA, the SIM/HBA shall:

A) Accept the message.

B) Set the Path ID of an IMMEDIATE NOTIFY CCB to the bus number of this bus. Set the Target ID of the SIM/HBA, and LUN ID from the IDENTIFY message. Set the Initiator ID field of the IMMEDIATE NOTIFY CCB to the ID of the Initiator that selected this SIM/HBA.

C) Set the IMMEDIATE NOTIFY CCB CAM Status to Message Received.

D) Form a unique sequence identifier and place it in the IMMEDIATE NOTIFY CCB's Sequence Identifier field.

E) Set the IMMEDIATE NOTIFY CCB Message Code field to the ABORT Message code.

F) All CONTINUE TARGET I/O CCBs for this I_T_L or I_T_L_Q nexus shall have the CAM Status set to Request Aborted by Host and shall be returned to the Host Target Mode peripheral driver by the CONTINUE TARGET I/O CCB callback mechanism.

G) Transition the SCSI Bus to BUS FREE. Callback the Host Target Mode peripheral driver using the callback notify field in the IMMEDIATE NOTIFY CCB. The exact order of these two operations is not specified.

H) For all initial connections the SIM/HBA for this I_T_L or I_T_L_Q shall transparently respond with a SCSI Status of BUSY until all events are acknowledged by the Host Target Mode peripheral driver.

I) All CONTINUE TARGET I/O CCBs received by the SIM/HBA for this I_T_L or I_T_L_Q shall be rejected until all events are acknowledged by the Host Target Mode Peripheral driver. The rejected CCBs shall have:

1) A CAM Status field set to Unacknowledged Event by Host

2) A return status of Unacknowledged Event by Host.

Acknowledgment of this event by the Host Target Mode peripheral driver shall be accomplished when the SIM/HBA receives an NOTIFY ACKNOWLEDGE CCB for this I_T_L or I_T_L_Q with the Sequence Identifier field equal to the sequence identifier of the IMMEDIATE NOTIFY CCB for this event.

11.3.3.1.3 Optional messages

This clause describes optional messages that are handled by the SIM/HBA with or without notification to the corresponding Host Target Mode peripheral driver.

11.3.3.1.3.1 Optional messages that are not supported

For all messages in this category which are not supported:

- If the SIM/HBA can not provide support or the Host Target Mode peripheral driver has indicated through the ENABLE TARGET LUN CCB that a message is not supported, the SIM/HBA shall reject the message and shall continue normal SCSI bus processing. The Host Target Mode peripheral driver shall not be notified of the event.

11.3.3.1.3.2 ABORT TAG message

The ABORT TAG message shall be supported if Tagged Queue Operation is active for this LUN. When an ABORT TAG message is received for an enabled Host Target Mode LUN by the SIM/HBA, the SIM/HBA shall:

A) Accept the message.

B) If the current I/O process is not fully identified (e.g., no Queue Tag message) then the SIM/HBA shall:

1) Go to BUS FREE phase.

2) No other processing is required (see ABORT TAG message in ANSI X3.131-1994 for further information).

C) Set the Path ID of an IMMEDIATE NOTIFY CCB to the bus number of this bus. Set the Target ID of the SIM/HBA, and LUN ID from the IDENTIFY message. Set the Initiator ID field of the IMMEDIATE NOTIFY CCB to the ID of the initiator that selected this SIM/HBA.

D) Set the IMMEDIATE NOTIFY CCB CAM Status to Message Received.

E) Set the IMMEDIATE NOTIFY CCB Message Code field to the ABORT TAG Message code. Place the Additional Arguments to the message (Queue Tag) in the Message arguments array of the IMMEDIATE NOTIFY CCB.

F) Form a unique sequence identifier and place it in the IMMEDIATE NOTIFY CCB's Sequence Identifier field.

G) Search the list of CONTINUE TARGET I/O CCB(s) for this I_T_L_Q nexus looking for a match between the TAG ID field of the CCB and the Queue Tag of the I_T_L_Q nexus. For each match found, that CONTINUE TARGET I/O CCB's CAM Status shall be set to Request Aborted by Host and shall be returned to the Host Target Mode peripheral driver by the CONTINUE TARGET I/O CCB callback mechanism.

H) Transition the SCSI Bus to BUS FREE. Callback the Host Target Mode peripheral driver using the callback notify field in the IMMEDIATE NOTIFY CCB. The exact order of these two operations is not specified.

I) For all initial connections, the SIM/HBA for this I_T_L or I_T_L_Q shall transparently respond with a SCSI Status of BUSY until all events are acknowledged by the Host Target Mode peripheral driver.

J) All CONTINUE TARGET I/O CCBs received by the SIM/HBA for this I_T_L or I_T_L_Q shall be rejected until all events are acknowledged by the Host Target Mode peripheral driver. The rejected CCBs shall have:

   1) A CAM Status field set to Unacknowledged Event by Host

   2) A return status of Unacknowledged Event by Host.

Acknowledgment of this event by the Host Target Mode peripheral driver shall be accomplished when the SIM/HBA receives a NOTIFY ACKNOWLEDGE CCB for this I_T_L or I_T_L_Q nexus with the Sequence Identifier field equal to the sequence identifier of the IMMEDIATE NOTIFY CCB for this event.

11.3.3.1.3.3 CLEAR QUEUE message

The CLEAR QUEUE message shall be supported if Tagged Queue Operation is active for this LUN. When an CLEAR QUEUE message is received for an enabled Host Target Mode LUN by the SIM/HBA, the SIM/HBA shall:

A) Accept the message.

B) Set the Path ID of an IMMEDIATE NOTIFY CCB to the bus number of this bus. Set the Target ID of the SIM/HBA, and LUN ID from the IDENTIFY message. Set the Initiator ID field of the IMMEDIATE NOTIFY CCB to the ID of the initiator that selected this SIM/HBA.

C) Set the IMMEDIATE NOTIFY CCB CAM Status to Message Received.

D) Set the IMMEDIATE NOTIFY CCB Message Code field to the CLEAR QUEUE Message code.

E) Form a unique sequence identifier and place it in the IMMEDIATE NOTIFY CCB's sequence identifier field.

F) All CONTINUE TARGET I/O CCB(s) for this enabled Host Target Mode LUN, shall have the CAM Status set to Request Aborted by Host and shall be returned to the Host Target Mode peripheral driver by the CONTINUE TARGET I/O CCB callback mechanism. Refer to ANSI X3.131-1994 clause 5.6.4 for any further clarification.

G) Transition the SCSI Bus to BUS FREE. Callback the Host Target Mode peripheral driver using the callback notify field in the IMMEDIATE NOTIFY CCB. The exact order of these two operations is not specified.

H) For all initial connections the SIM/HBA for this LUN shall transparently respond with a SCSI Status of BUSY until all events are acknowledged by the Host Target Mode peripheral driver.

I) All CONTINUE TARGET I/O CCBs received by the SIM/HBA for this LUN shall be rejected until all events are acknowledged by the Host Target Mode Peripheral driver. The rejected CCBs shall have:

1) A CAM Status field set to Unacknowledged Event by Host.

2) A return status of Unacknowledged Event by Host.

Acknowledgment of this event by the Host Target Mode peripheral driver shall be accomplished when the SIM/HBA receives a NOTIFY ACKNOWLEDGE CCB for this LUN with the Sequence Identifier field equal to the sequence identifier of the IMMEDIATE NOTIFY CCB for this event.

11.3.3.1.3.4 HEAD OF QUEUE, ORDERED QUEUE and SIMPLE QUEUE TAG messages

The handling of these messages is described in detail in Clause 11.3.8.

11.3.3.1.3.5 TERMINATE I/O PROCESS message

When the supported TERMINATE I/O PROCESS message is received for an enabled Host Target Mode LUN by the SIM/HBA, the SIM/HBA shall:

A) If there is no matching I/O process, reject the message, and no further processing is required.

B) If there is a matching I/O process, accept the message.

C) If disconnects are mandatory the SIM/HBA shall disconnect from the bus. If disconnects are allowed the SIM/HBA should disconnect from the bus.

D) Set the Path ID of an IMMEDIATE NOTIFY CCB to the bus number of this bus. Set the Target ID of the SIM/HBA, and LUN ID from the IDENTIFY message. Set the Initiator ID field of the IMMEDIATE NOTIFY CCB to the ID of the initiator that selected this SIM/HBA.

E) Form a unique sequence identifier and place it in the IMMEDIATE NOTIFY CCB's Sequence Identifier field.

F) Set the IMMEDIATE NOTIFY CCB CAM Status to Message Received.

G) Set the IMMEDIATE NOTIFY CCB Message Code field to the TERMINATE I/O PROCESS message code.

H) Each CONTINUE TARGET I/O CCB for this I_T_L or I_T_L_Q nexus shall have the CAM Status set to Terminate I/O Process. The Residual Length field shall be set to valid number of bytes not transferred for this CCB. The CCB(s) shall be returned to the Host Target Mode peripheral driver by the CONTINUE TARGET I/O CCB callback mechanism.

I) Call back the Host Target Mode peripheral driver by the mechanism provided by the IMMEDIATE NOTIFY CCB.

J) For all initial connections the SIM/HBA for this I_T_L or I_T_L_Q shall transparently respond with a SCSI Status of BUSY until all events are acknowledged by the Host Target Mode peripheral driver.

K) All CONTINUE TARGET I/O CCBs received by the SIM/HBA for this I_T_L or I_T_L_Q shall be rejected until all events are acknowledged by the Host Target Mode peripheral driver. The rejected CCBs shall have:

1) A CAM Status field set to Unacknowledged Event by Host

2) A return status of Unacknowledged Event by Host.

L) Acknowledgment of this event by the Host Target Mode peripheral driver shall be accomplished when the SIM/HBA receives an NOTIFY ACKNOWLEDGE CCB for this I_T_L nexus with the Sequence Identifier field equal to the Sequence Identifier of the IMMEDIATE NOTIFY CCB for this event.

The Host Target Mode peripheral driver shall acknowledge the event and properly terminate the I/O process as specified in ANSI X3.131-1994.

11.3.3.1.4 Resource unavailable to SIM/HBA

If the SIM/HBA receives a CDB on the SCSI bus for an enabled LUN in Host Target Mode that does not have any ACCEPT TARGET I/O CCBs available for use, it shall do the following:

A) Set the Path ID of an IMMEDIATE NOTIFY CCB to the bus number of this bus. Set the Target ID of the SIM/HBA, and LUN ID from the IDENTIFY message. Set the Initiator ID field of the IMMEDIATE NOTIFY CCB to the ID of the initiator that selected this SIM/HBA.

B) Set the IMMEDIATE NOTIFY CCB CAM Status to Unavailable Resource.

C) Form a unique sequence identifier and place it in the IMMEDIATE NOTIFY CCB's sequence identifier field.

D) Transition the SCSI bus to status phase and return BUSY status to the initiator and then go to BUS FREE phase. Callback the Host Target Mode peripheral driver using the callback notify field in the IMMEDIATE NOTIFY CCB. These operations shall be performed in the order shown here.

E) For all initial connections the SIM/HBA for this LUN shall transparently respond with a SCSI Status of BUSY until all events are acknowledged by the Host Target Mode peripheral driver.

Note: ACCEPT and CONTINUE TARGET I/O CCBs are allowed to be received and processed by the SIM/HBA for this event. The Host Target Mode peripheral driver should send a number of ACCEPT TARGET I/O CCBs to the enabled LUN and then acknowledge the event to replenish the resource.

Acknowledgment of this event by the Host Target Mode peripheral driver shall be accomplished when the SIM/HBA receives a NOTIFY ACKNOWLEDGE CCB for this LUN with the Sequence Identifier field equal to the sequence identifier of the IMMEDIATE NOTIFY CCB for this event.

11.3.3.1.5 HBA faults

If a controlling SIM detects that a HBA has faulted/failed in a way that causes the HBA to be not usable in its current state (e.g., has ceased responding, declared itself insane, needs to be initialized/restarted), it shall do the following for each Host Target Mode enabled LUN for the faulted/failed HBA:

A) Set the Path ID of an IMMEDIATE NOTIFY CCB to the XPT assigned SCSI bus number. Set the Target ID of the HBA for this SCSI bus, and the LUN to the LUN number being reported.

B) Set the IMMEDIATE NOTIFY CCB CAM Status to No HBA Detected.

C) Form a unique sequence identifier and place it in the IMMEDIATE NOTIFY CCB's sequence identifier field.

D) All CONTINUE TARGET I/O CCBs for this Host Target Mode LUN shall have the CAM Status set to No HBA Detected and shall be returned to the Host Target Mode peripheral driver by the CONTINUE TARGET I/O CCB callback mechanism.

E) Clear the unacknowledged event list of all other unacknowledged events for this LUN (e.g., the only unacknowledged event for this LUN is this event).

F) Callback the Host Target Mode peripheral driver using the callback notify field in the IMMEDIATE NOTIFY CCB.

G) All CONTINUE TARGET I/O CCBs received by the SIM for this LUN until all events are acknowledged by the Host Target Mode peripheral driver shall be rejected with:

   1) A CAM Status field set to Unacknowledged Event by Host

   2) A return status of Unacknowledged Event by Host.

H) The SIM may rectify fault/failure for the HBA and bring the HBA to a usable state once again. The recovery actions that the SIM employs are vendor specific but the SIM/HBA responses to connects are specified for the fault/failure recovery period. The term "fault/failure recovery period" shall mean from the SIMs initial recovery action (e.g., the first step done by the SIM to correct the HBA fault/failure) to the SIM/HBA being able to comply to this standard for an enabled Host Target Mode LUN.

   During the fault/failure recovery period the SIM/HBA shall respond to connects in one of the following ways:

   1) No response to selections (selection timeout) until the recovery period ends.

   2) SCSI-2 BUSY status in response to connections until the recovery period ends.

   3) No response to selections (selection timeout) for a period of time then SCSI-2 BUSY status in response to connections until the recovery period ends.

I) The state of the SIM/HBA shall be reflected in the response to PATH INQUIRY CCBs (see Clause 9.2.3 Path Inquiry for further details).

Note: The Host Target Mode LUN is not disabled unless the Host Target Mode peripheral driver explicitly does a disable LUN function.

Acknowledgment of this event by the Host Target Mode peripheral driver shall be accomplished when the SIM receives a NOTIFY ACKNOWLEDGE CCB for this LUN with the Sequence Identifier field equal to the sequence identifier of the IMMEDIATE NOTIFY CCB for this event.

The Host Target Mode peripheral driver shall recognize that this event is analogous to a power off of the HBA. The Host Target Mode peripheral driver may disable the LUN or may try to continue processing (e.g., ascertain if the SIM has brought the HBA to a usable state) or may try to continue processing and then disable the LUN.

The Host Target Mode peripheral driver shall ascertain through the Path Inquiry function if the SIM has brought the HBA to a usable state. If the SIM has brought the HBA to usable state again the Host Target Mode peripheral driver shall treat this condition as a power on/reset of the I_T_L or I_T_L_Q.

Note: The Host Target Mode peripheral driver should respond with CHECK CONDITION status and UNIT ATTENTION sense key in response to REQUEST SENSE command for the first command received except for INQUIRY and REQUEST SENSE commands for this I_T_L or I_T_L_Q. The Host Target Mode peripheral driver should ensure that there are ACCEPT TARGET I/O CCBs available to the SIM/HBA before acknowledging the event if processing with the SIM/HBA is to be resumed.

11.3.4 IMMEDIATE NOTIFY CCB

The following lists the fields of the IMMEDIATE NOTIFY CCB. No entry under the "Dir" heading indicates that this field is not being used for the IMMEDIATE NOTIFY CCB.

Table 24 - IMMEDIATE NOTIFY CCB

Size	Dir	Immediate notify
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code (IMMEDIATE NOTIFY)
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID (Bus no. of SIM/HBA)
1	I	Target ID (Target ID of SIM/HBA)
1	O	LUN
4	O	CAM Flags
4	O	Pointer to the sense buffer
1	O	Length of the sense buffer
3		Reserved
4	I	Callback on notification
1	I	Initiator ID (ID of Initiator that selected SIM/HBA)
1		Reserved
2	I	Sequence Identifier
1	I	Message Code
7	I	Message Arguments

The following are the only possible CAM Status values for the IMMEDIATE NOTIFY CCB passed to the SIM/HBA from the Host Target Mode peripheral driver (e.g., rejected by the SIM/HBA):

- Invalid Request - Indicates that the CCB has invalid field(s).
- Invalid Path ID - Indicates the Path ID is not known.
- Invalid Target ID - Indicates the Target ID is not that of the target device.
- Invalid LUN ID - Indicates the Target LUN is not in the valid range for LUNs.

The following are the only possible CAM Status values for the IMMEDIATE NOTIFY CCB passed from the SIM/HBA to the Host Target Mode peripheral driver (e.g., event notification by the SIM/HBA):

- No HBA Detected
- Nexus Not Established
- Message Received
- Unavailable Resource

Fields Described:
The Pointer to Sense Buffer field shall contain a pointer to a buffer having minimum of 18 bytes.

The Sense Buffer Length field shall be the length of the sense buffer. The length shall be at least 18.

Initiator ID field is the SCSI BUS ID of the Initiator that selected the SIM/HBA.

The Message Code field is used to store the SCSI message code for received messages.

The Message Arguments field is used to store SCSI message arguments received.

11.3.5 NOTIFY ACKNOWLEDGE CCB

The following lists the fields of the NOTIFY ACKNOWLEDGE CCB. No entry under the "Dir" heading indicates that this field is not being used for the NOTIFY ACKNOWLEDGE CCB.

Table 25 - NOTIFY ACKNOWLEDGE CCB

Size	Dir	Notify Acknowledge
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code (NOTIFY ACKNOWLEDGE)
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID (Bus no. of SIM/HBA)
1	O	Target ID (Target ID of SIM/HBA)
1	O	LUN
4	O	CAM Flags
2	O	Sequence Identifier
1	O	Event
	O	7 Reset Cleared
		6 - 0 Reserved
1		Reserved

The following are the only possible CAM Status values for the NOTIFY ACKNOWLEDGE CCB passed to the SIM/HBA from the Host Target Mode peripheral driver:

- Request Complete without Error - Indicates that the event has been acknowledged by the SIM/HBA.
- Invalid Request - Unknown Sequence Identifier
- Invalid Path ID - Indicates the Path ID is not known.
- Invalid Target ID - Indicates the Target ID is not that of the target device.
- Invalid LUN ID - Indicates the Target LUN is not in the valid range for LUNs.

Fields Described:
The Sequence Identifier field is the sequence event identifier which is being acknowledged.

The Event field is used for acknowledgement of events whose notifications are delivered by the Asynchronous Event mechanism (BUS RESET and BUS DEVICE RESET message).

11.3.6 Enable target mode LUN for host target mode

It is recommended that the SIM/HBA reserve LUN 0 for Asynchronous Event Notification. While it is not a requirement, it is a suggestion due to the increased complexity required to handle both Host Target Mode and AENs. If the SIM/HBA reserves LUN 0 for AENs then it shall return the CAM Status of LUN Already Enabled for all ENABLE LUN CCBs for LUN 0.

When a peripheral driver wishes to enable a LUN for Host Target Mode it shall perform the following tasks:

A) Issue a Path Inquiry to the SIM/HBA to ascertain what features and options it supports.

B) If the Host Target Mode peripheral driver requires disconnecting from the SCSI bus after receiving a CDB, supports optional immediate notify message processing or the peripheral driver wishes to support tagged commands, it shall use the information returned from the PATH INQUIRY CCB to ascertain whether the SIM/HBA supports these capabilities. It is recommended that the Host Target Mode peripheral driver return for the INQUIRY command data that reflects the features of the SIM/HBA. These features include Wide Bus

32, Wide Bus 16, and Synchronous Data Transfers and are obtained from the PATH INQUIRY CCB. If the SIM/HBA supports Tag Queuing and the Host Target Mode peripheral driver wishes to support the feature, it shall be reflected in INQUIRY data.

C) The Host Target Mode peripheral driver shall issue the SET ASYNCHRONOUS CALLBACK CCB for each LUN that it supports to register for bus reset and bus device reset.

Note: The peripheral driver should prepare, if necessary, to handle the Contingent Allegiance condition on a per initiator basis. Therefore the peripheral driver may need to do some initial setup for this.

D) The Host Target Mode peripheral driver shall issue the ENABLE LUN CCB to the SIM to register for Host Target Mode with the SIM/HBA. The ENABLE LUN CCB shall be setup as follows:

1) If the Host Target Mode peripheral driver requires disconnects and the SIM/HBA supports disconnects (as ascertained by the Disconnects Supported field in the PATH INQUIRY CCB), the Disconnect Mandatory bit in the target mode specific CAM Flags field in the ENABLE LUN CCB shall be set.

2) If the Host Target Mode peripheral driver supports tagged commands, and if the SIM/HBA supports Tagged Queuing, the Tag Queue Enable bit shall be set in the CAM Flags field of the ENABLE LUN CCB.

3) A Host Target Mode peripheral driver shall clear the Host Target Mode flag indicating this is a Host Target Mode ENABLE LUN CCB. The Host Target Mode peripheral driver shall set in the Target Mode bits of the CAM Flags all other optional settings it wishes to support.

4) The Immediate Notify CCB list pointer field shall point to a list of CCBs (of at least one) available to the SIM/HBA to use for Host Target Mode event/message notification. It is recommended that there should at least one CCB for each and every initiator that this Host Target Mode peripheral driver expects to have a connection with. These CCBs are pre-allocated CCBs from the XPT layer which are empty (zeroed) except for the following:

a) The Address of this CCB and CAM Control Block Length. These fields shall contain the proper information as defined previously in this standard.

b) The Connect ID field shall be identical to the ENABLE LUN CCB Connect ID field.

c) The completion callback function shall be set to the Immediate Notify callback function. This is the function in the Host Target Mode peripheral driver to be called when a event is detected by the SIM/HBA. These events are described in Clause 11.3.3.

d) The function code shall be set to IMMEDIATE NOTIFY.

e) The pointer to the sense buffer, and the length of the sense buffer shall be set. The sense buffer length shall be a minimum of 18 bytes.

Note: See clause 11.3.11 (Disable of Host Target Mode LUN) for the mechanism on how to retrieve ownership of the IMMEDIATE NOTIFY CCBs.

5) The Accept Target I/O CCB List Pointer field shall point to a list of CCBs (of at least one) available to the SIM to use for Host Target Mode operation. These CCBs are pre-allocated CCBs from the XPT layer which are empty (zeroed) except for the following:

a) The Address of this CCB and CAM Control Block Length. These fields shall contain the proper information as defined previously in this standard.

b) The completion callback function shall be set to the CDB Received Completion callback function. This is the function in the Host Target Mode peripheral driver to be called when a CDB is received error free from an initiator by the SIM.

c) The function code shall be set to Accept Target I/O.

d) If the CDB Pointer bit is set in the flags field of the ACCEPT TARGET I/O CCB, the cdb pointer and cdb length field shall be set.

e) The pointer to the sense buffer, and the length of the sense buffer shall be set. The sense buffer length shall be a minimum of 18 bytes.

Note: For error conditions detected by the SIM/HBA, the sense buffers in both the ACCEPT TARGET I/O and CONTINUE TARGET I/O CCBs are used to report proper sense data back to the Host Target Mode peripheral driver (i.e., Memory/RAM failures that the Host Target Mode peripheral driver has no knowledge of unless reported by the SIM/HBA).

6) If Group 6 and/or 7 commands are to be supported by the Host Target Mode peripheral driver, then the Group 6 and/or 7 Vendor Unique CDB Length fields shall contain the number of bytes for these CDB group codes. If Group 6 and/or 7 commands are not supported by the Host Target Mode peripheral driver, then the Group 6 and/or 7 Vendor Unique CDB Length fields shall be equal to zero. If the Host Target Mode peripheral driver supports Group 6 and/or 7 commands, then the Host Target Mode peripheral driver shall ensure that all ACCEPT TARGET I/O CCBs passed to the SIM/HBA contain sufficient storage for the received CDB. If a vendor unique command CDB is larger than the CDB field for the Accept Target I/O, then the CDB field shall contain a pointer to a buffer of sufficient length.

E) The Host Target Mode peripheral driver shall verify that the Enable LUN succeeded by checking that the CAM Status in the ENABLE LUN CCB is set to Request Completed without Error.

F) The Host Target Mode peripheral driver may add to the list of ACCEPT TARGET I/O CCBs by issuing an ACCEPT TARGET I/O CCB to the SIM/HBA anytime after the LUN has been enabled.

G) The Host Target Mode peripheral driver may add to the list of IMMEDIATE NOTIFY CCBs by issuing an IMMEDIATE NOTIFY CCB to the SIM/HBA anytime after the LUN has been enabled.

Note: Once an IMMEDIATE NOTIFY CCB is given to an enabled Host Target Mode SIM/HBA, ownership of the CCB remains with the SIM/HBA until the LUN is disabled.

When the SIM/HBA receives a Host Target Mode ENABLE LUN CCB with a non-zero number of Target CCBs from the Host Target Mode peripheral driver, it shall do the following:

A) If the LUN is already enabled, the ENABLE LUN CCB shall be returned with a CAM Status of LUN Already Enabled.

B) If the Path ID, Target ID or LUN specified in the ENABLE LUN CCB are invalid then the Enable LUN shall be failed with the proper CAM Status (see Clause 11.3.7).

C) The SIM/HBA shall check the Host Target Mode options in the target mode specific CAM Flags field of the ENABLE LUN CCB. If specific target mode options are set in this field and the SIM does not support these options, the ENABLE LUN CCB shall be returned with a CAM Status of Cannot Provide Requested Capability.

D) The SIM/HBA shall check whether the Tagged Queue Enable bit is set in the CAM Flags field of the ENABLE LUN CCB. If the Tagged Queue Enable bit is set but the SIM does not support tagged commands, the ENABLE LUN CCB shall be returned with a CAM Status of Cannot Provide Requested Capability.

E) The SIM/HBA shall check that the ENABLE LUN CCB Immediate Notify CCB Pointer field is non null and for each IMMEDIATE NOTIFY in the list, the Immediate Notify CCB Callback field is set. The SIM/HBA shall check that the pointer to the sense buffer is non null and length of the sense buffer is a minimum of 18 bytes for each IMMEDIATE NOTIFY CCB. If either the ENABLE LUN CCB or any IMMEDIATE NOTIFY CCB is found to be in error, then the ENABLE LUN CCB shall fail with Invalid Request.

F) The SIM/HBA shall check that the ENABLE LUN CCB Target CCB pointer field is non null and for each ACCEPT TARGET I/O CCB in the list, the Accept Target I/O CCB Callback on Completion field is set. The SIM/HBA shall check that the pointer to the sense buffer is non null and length of the sense buffer is a minimum of 18 bytes for each ACCEPT TARGET I/O CCB. If either the ENABLE LUN CCB or an ACCEPT TARGET I/O CCB is found to be in error, then the ENABLE LUN CCB shall fail with Invalid Request.

G) If preceding checks complete without error, the CAM Status of the ENABLE LUN CCB shall be set to Request Completed without Error and the CCB returned.

11.3.7 ENABLE LUN CCB for host target mode

The following lists the fields of the ENABLE LUN CCB. No entry under the "Dir" heading indicates that this field is not being used for the ENABLE LUN CCB.

Table 26 - ENABLE LUN CCB for host target mode

Size	Dir	Enable LUN
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code (Enable LUN)
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID (Bus no. of SIM/HBA)
1	O	Target ID (Target ID of SIM/HBA)
1	O	LUN
4	O	CAM Flags
2	O	Group 6 Vendor Unique CDB Lengths
2	O	Group 7 Vendor Unique CDB Lengths
4	O	Pointer to Immediate Notify CCB List
4	O	Number of Immediate Notify CCBs
4	O	Pointer to Target CCB list
2	O	Number of Target CCBs
2		Reserved
n		SIM private

The following are the only possible CAM Status return values for the ENABLE LUN CCB:

- Request Completed without Error - Completed without error.
- LUN Already Enabled - The specified target mode LUN is already enabled.
- Invalid Path ID - Indicates the Path ID is not known.
- Invalid Target ID - Indicates the Target ID is not that of the peripheral device.
- Invalid LUN ID - Indicates the Target LUN is not in the valid range for LUNs.
- Invalid Request - For the ENABLE LUN CCB with a list count greater than zero, this CAM Status indicates invalid field(s) within the CCB(s). For ENABLE LUN CCB with a list count of zero (DISABLE LUN) see section 11.3.11.

- Cannot Provide Requested Capability
  - A) The SIM does not support target mode.
  - B) The Host Target Mode peripheral driver requires disconnects but the SIM does not support disconnects.
  - C) The Host Target Mode peripheral driver requested the ability to run tagged but the SIM does not support this feature.
- CCB Length Inadequate - Indicates more private data area is required in the CCB.

The following are the only possible CAM Flags bits which are valid for the ENABLE LUN CCB:
- Tagged Queue Action Enable - Indicates to the SIM/HBA that the Host Target Mode peripheral driver requires tagged queued operation.
- Disconnects Mandatory - Indicates to the SIM/HBA that any IDENTIFY message received shall have the DiscPriv bit set. Clause 11.3.8.2 describes the SIM/HBA response if the DiscPriv bit is not set when Disconnect Mandatory CAM Flag indicates that it is required.

Fields Described:
The Pointer to Target CCB List field shall contain a pointer to a list of ACCEPT TARGET I/O CCB(s) for the Enable LUN request (see Table 22 for the format of the list). For the Disable LUN request, this field shall be null.

The Number of Target CCBs shall be set to the number of CCBs (greater than zero) pointed to by the Pointer to Target CCB List field for the Enable LUN request. For the Disable LUN request this field shall be zero.

The Pointer to Immediate Notify CCB List field shall contain a pointer to a list of IMMEDIATE NOTIFY CCBs for the Enable LUN request (see Table 22 for the format of the list).

The Number of Immediate Notify CCBs shall be set to the number of CCBs (greater than zero) pointed to by the Pointer to Immediate Notify CCB List field for the Enable LUN request.

11.3.8 ACCEPT TARGET I/O and CONTINUE TARGET I/O CCB operation

11.3.8.1 SIM/HBA ACCEPT TARGET I/O CCB acceptance

When the SIM/HBA receives an ACCEPT TARGET I/O CCB from a Host Target Mode peripheral driver, it shall do the following:

A) Check that the Path ID, Target ID, and LUN specified in the ACCEPT TARGET I/O CCB is that of an enabled LUN. If the LUN is not enabled, the CCB CAM Status and xpt_action() return status shall be Path Invalid.

B) Check that the CDB Received Completion callback function is set in the ACCEPT TARGET I/O CCB. If it is not set, the CCB CAM Status and xpt_action() return status shall be Request Completed with Error.

C) Check that the pointer to the sense buffer is set, and that the sense buffer length is a minimum of 18 bytes. If not correct, the CCB CAM Status and xpt_action() return status shall be Request Completed with Error.

D) Otherwise the xpt_action() return status shall be Request in Progress.

11.3.8.2 SIM/HBA CDB reception

Error condition handling for CDB reception for Accept Target I/O functions are described in Clause 11.3.8.6.

When the SIM/HBA requests a CDB on the SCSI bus for an enabled LUN in Host Target Mode, it shall do the following:

A) If the DiscPriv bit is not set in the IDENTIFY message and the Disconnect Mandatory bit was set in the target mode specific CAM Flags field of the ENABLE LUN CCB, then the SIM shall go to BUS FREE.

Note: Refer to Clause 11.3.3 (Use of the IMMEDIATE NOTIFY CCB) for reference in handling this condition. Additional information can be found in ANSI X3.131-1994 regarding BUS FREE.

B) Remove a CCB from the SIM's list of available ACCEPT TARGET I/O CCBs which were made available with the Enable LUN or Accept Target I/O function. If there are no CCBs available, then the SIM shall go to status phase and return BUSY SCSI status to the initiator. The SIM/HBA shall notify the Host Target Mode peripheral driver of this event by the Immediate Notify mechanism described in Clause 11.3.3.

C) The ACCEPT TARGET I/O CCB shall be filled in with the following information:
   1) The Target ID of this SIM/HBA.
   2) Bus ID of the SIM/HBA.
   3) LUN ID.
   4) The CDB data received from the initiator.
      - If the Group Code of the Operation Code of the CDB is Vendor Unique, the SIM/HBA shall transfer the number of CDB bytes specified in the ENABLE LUN CCB for this LUN. The Group Code in the incoming CDB (either 6 or 7) shall select the Vendor Unique CDB size from the ENABLE LUN CCB. If the selected CDB size (specified in the ENABLE LUN CCB) is zero, the SIM/HBA shall transfer only the CDB Operation Code. If the required number of bytes is not transferred or the specified size for this Group Code is zero, then the SIM shall set in the selected CCB the CDB bytes transferred in the area provided (see Clause 11.3.8.6 for further details).

Note: The SIM/HBA does not set the CDB Length field in the ACCEPT TARGET I/O CCB with number of CDB bytes transferred. The Group Code of the CDB is used by both the SIM/HBA and the Host Target Mode peripheral driver to ascertain the length of the CDB. The CDB Length field is used to indicate the amount of buffer area is available for CDB bytes.

   5) The SCSI Bus ID of the initiator that selected this SIM/HBA in the Initiator ID field.

D) The CAM Status shall be set to CDB Received.

E) If this is a tagged request then the SIM/HBA shall do the following:

   1) The SIM/HBA shall check whether the Host Target Mode peripheral driver has enabled tagged queuing with the Enable LUN command for this LUN. If the Host Target Mode peripheral driver does not support tagged commands then the SIM/HBA shall send the MESSAGE REJECT message for the Queue Tag message and continue as specified in ANSI X3.131-1994.

   2) If the Host Target Mode peripheral driver does support tagged commands, the following shall occur:

      a) The Queue Tag message (HEAD OF QUEUE TAG, ORDERED QUEUE TAG, or SIMPLE QUEUE TAG) shall be placed in the Tag Queue Action field of the ACCEPT TARGET I/O CCB.

      b) The queue tag value shall be placed in the tag id field of the ACCEPT TARGET I/O CCB.

F) If the Host Target Mode peripheral driver requires disconnects, the SIM/HBA shall send the DISCONNECT message and go to BUS FREE phase. If disconnects are not required (ENABLE LUN CCB) but are allowed (IDENTIFY message) the SIM/HBA should send the DISCONNECT message and go to BUS FREE phase.

G) Call the Host Target Mode peripheral driver CDB Received Completion callback function provided in the ACCEPT TARGET I/O CCB Callback on Completion field.

Note: The SIM/HBA handle SDTR and WDTR messages transparently.

11.3.8.3 Host peripheral driver CDB Received Completion callback function

When the Host Target Mode peripheral driver's CDB Received Completion callback function is called, it shall do the following:

A) Ascertain how it responds to the SCSI command specified in the CDB. The Host Target Mode peripheral driver may use the same CCB (although this is not a requirement) for the Continue Target I/O function, filling in the connect id, initiator id, function code, the data pointer (if needed), data count (if needed), data direction (if any), the callback completion function, and the queue tag id if running tagged. If a data transfer is required then the Direction bits shall be set in the CAM Flags field of the CONTINUE TARGET I/O CCB. If disconnects are enabled, the Host Target Mode peripheral driver may require a SAVE DATA POINTERS message and DISCONNECT message after transferring some of the data. If the request is to be completed then the SEND_STATUS bit shall be set in the CAM Flags field of the CONTINUE TARGET I/O CCB and the SCSI status field shall contain the SCSI status to be returned to the initiator.

B) If a CHECK CONDITION SCSI status is to be returned to the initiator, then the Host Target Mode peripheral driver shall setup and maintain the CONTINGENT ALLEGIANCE condition as specified in ANSI X3.131-1994.

Note: Host Target Mode peripheral driver sense data always represents the current status of the Host Target Mode peripheral driver.

C) The Host Target Mode peripheral driver shall then call the SIM/HBA via xpt_action() passing the CONTINUE TARGET I/O CCB.

D) On return from the xpt_action() call, the Host Target Mode peripheral driver shall check that the return value from xpt_action() is Request in Progress. If a status other than Request in Progress is returned, then the command could not be transferred to the SIM/HBA (possibly due to a bus reset or the receipt of a message). It is the responsibility of the Host Target Mode peripheral driver to handle the condition correctly.

Note: Refer to Clause 13.3.3 (Use of the IMMEDIATE NOTIFY CCB) for additional information.

11.3.8.4 SIM/HBA CONTINUE TARGET I/O CCB acceptance

Error condition handling for Data phase and Message phase for Continue Target I/O functions are described in Clause 11.3.8.6.

When the SIM/HBA receives a CONTINUE TARGET I/O CCB it shall do the following:

A) If the SIM/HBA is recovering from a bus reset or bus device reset, or an Immediate Notify Condition the CAM Status and the xpt_action() return status shall be set appropriately.

B) Check whether there are outstanding unacknowledged events for this I_T_L nexus, if there are any the SIM/HBA shall set the CAM Status and the xpt_action() return status to Unacknowledged Event by Host and return.

C) Check that the Path ID, Target ID, and LUN specified in the CONTINUE TARGET I/O CCB is that of an enabled LUN. If the LUN is not enabled, the CCB CAM Status and xpt_action() return status shall be Path Invalid.

D) Check that the callback on completion function is set in the CONTINUE TARGET I/O CCB. If it is not set, the CCB CAM Status and xpt_action() return status shall be Request Completed with Error.

E) Check that the pointer to the sense buffer is set, and that the sense buffer length is a minimum of 18 bytes. If not correct, the CCB CAM Status and xpt_action() return status shall be Request Completed with Error.

F) Set the CAM Status and xpt_action() return value to Request in Progress.

G) If disconnects are not enabled, the SIM/HBA shall perform the necessary phase transitions. If disconnects are enabled, the SIM/HBA shall reselect the initiator, when possible, to perform the necessary phase transitions and reestablish the I_T_L or I_T_L_Q nexus. The order of the operations that the SIM/HBA shall perform is specified as follows:

1) If the Direction Bits are set specifying DATA In (data to initiator) or DATA Out (data from the initiator) in the CAM Flags field of the CONTINUE TARGET I/O CCB, then data shall be transferred (Data phase).

2) If the Send Status bit is set in the CAM Flags field of the CONTINUE TARGET I/O CCB, the SIM/HBA shall go to STATUS phase and complete the request with a COMMAND COMPLETE message and go to BUS FREE phase.

3) If disconnects are enabled and the SEND_STATUS bit is not set in the CAM Flags field of the CONTINUE TARGET I/O CCB, the SIM/HBA shall send a SAVE DATA POINTER message and a DISCONNECT message followed by a BUS FREE phase.

4) After all operations specified in the CCB have been transferred successfully, the Host Target Mode peripheral driver's callback completion function in the CONTINUE TARGET I/O CCB shall be called with a CAM Status of Request Completed without Error.

11.3.8.5 Host target mode peripheral driver continue target I/O callback

The Host Target Mode peripheral driver's Continue Target I/O callback completion function shall:

A) If the CAM Status is not Request Complete Without Error, then an error has occurred. The Host Target Mode peripheral driver shall be responsible for forming sense data, if applicable and for maintaining it. The sense data conditions are defined in ANSI X3.131-1994. The Host Target Mode peripheral driver also shall be responsible for maintaining the CONTINGENT ALLEGIANCE condition associated with the sense data.

B) If the request completed and the Host Target Mode peripheral driver has more data and/or status to send to the initiator then a CONTINUE TARGET I/O CCB shall be sent to the SIM with the proper fields set.

C) If status has been sent to the initiator then the Host Target Mode peripheral driver may reissue the same CCB by calling the SIM/HBA passing the same CCB. The CCB shall have all fields properly set for the ACCEPT TARGET I/O CCB as specified in this document. The SIM/HBA can now reuse this CCB to accept a new connection for the CDB received.

D) If status has not been sent, then the Host Target Mode peripheral driver can reissue the CCB as a CONTINUE TARGET I/O CCB after it has been set up.

11.3.8.6 Command reception errors and data phase errors handling.

If the command Group code is not supported (Groups 3,4,6 or 7), or there were CDB length errors, or there were errors in the incoming CDB that could not be handled transparently, the SIM/HBA shall do the following:

A) Set the proper Connect ID in the ACCEPT TARGET I/O CCB, and the Target ID of the initiator that selected this SIM/HBA in the Initiator ID field.

B) Form the SCSI-2 required 18 bytes of correct SCSI sense data and place it in the ACCEPT TARGET I/O CCB sense buffer. The SIM/HBA shall also indicate that sense data is valid by setting the Autosense flag in CAM Status and set the SCSI Status field to CHECK CONDITION.

C) For a Tagged Queue I/O process:

Note: If the SIM/HBA or the Host Target Mode peripheral driver has indicated that Queue Tags are not supported, the Queue Tag message would have been rejected.

1) The Queue Tag message (HEAD OF QUEUE TAG, ORDERED QUEUE TAG, or SIMPLE QUEUE TAG) is placed in the Tag Queue Action field of the ACCEPT TARGET I/O CCB.

2) The Queue Tag value shall be placed in the Tag ID field of the ACCEPT TARGET I/O CCB.

3) The Tagged Queue Action Enable bit shall be set in the CAM Flags field of the ACCEPT TARGET I/O CCB.

D) Due to the variety of SIM/HBA functionality and bus conditions, the SIM/HBA shall release the SCSI BUS in one of the following ways:

1) If the Host Target Mode peripheral driver required disconnects, the SIM/HBA shall send a DISCONNECT message and go to BUS FREE phase. If disconnects are not required but are allowed (IDENTIFY message), the SIM/HBA should send a DISCONNECT message and go to BUS FREE phase.

   If the SIM/HBA disconnects from the bus, then the CAM Status in the ACCEPT TARGET I/O CCB shall be set to Invalid CDB.

2) Cause an unexpected bus free. See ANSI X3.131-1994 for further details (BUS FREE).

   If the SIM/HBA causes an unexpected BUS FREE condition, then the CAM Status in the ACCEPT TARGET I/O CCB shall be set to Unexpected Bus Free.

Note: The Unexpected Bus Free Cam status ends the I/O process and the Host Target Mode peripheral driver should not try to terminate it.

E) Call back the Host Target Mode peripheral driver using the callback field within the ACCEPT TARGET I/O CCB.

The Host Target Mode peripheral driver shall be responsible for the creation and preservation of the Contingent Allegiance condition if the CAM Status is not Unexpected Bus Free. If the CAM Status is not Unexpected Bus Free, it shall also complete the bus transaction by sending a CONTINUE TARGET I/O CCB to properly terminate this connection.

If the SIM/HBA detects Data phase, Message Phase, or STATUS phase errors that cannot be handled transparently or any other unrecoverable errors (except timeouts) while processing a CONTINUE TARGET I/O CCB, the SIM/HBA shall:

A) Set the proper Connect ID in the CONTINUE TARGET I/O CCB, and the Target ID of the initiator that selected this SIM/HBA in the Initiator ID field.

B) The SIM/HBA shall form the SCSI-2 required 18 bytes of correct SCSI sense data and place it in the CONTINUE TARGET I/O CCB sense buffer. The SIM/HBA shall also indicate that sense data is valid by setting the Autosense flag in CAM Status and set the SCSI Status field to CHECK CONDITION.

C) For a Tagged Queue I/O process:

1) The queue tag value shall be placed in the Tag ID field of the CONTINUE TARGET I/O CCB.

2) The Tagged Queue Action Enable bit shall be set in the CAM Flags field of the CONTINUE TARGET I/O CCB.

D) Due to the variety of SIM/HBA functionality and bus conditions, the SIM/HBA shall release the SCSI BUS in one of the following ways:

1) If the Host Target Mode peripheral driver required disconnects, the SIM/HBA shall send a DISCONNECT message and go to BUS FREE phase. If disconnects are not required but are allowed (IDENTIFY message), the SIM/HBA should send a DISCONNECT message and go to BUS FREE phase. If the SIM/HBA disconnects from the bus, then the CAM Status in the CONTINUE TARGET I/O CCB shall be set to one of the following:

a) Target Bus Phase Sequence Failure

b) Uncorrectable Parity Error Detected

c) Initiator Detected Error

2) Cause an unexpected bus free. See ANSI X3.131-1994 for further details (BUS FREE).

If the SIM/HBA causes an unexpected BUS FREE condition, then the CAM Status in the CONTINUE TARGET I/O CCB shall be set to Unexpected Bus Free.

Note: The Unexpected Bus Free Cam status ends the I/O process and the Host Target Mode peripheral driver should not try to terminate it.

E) Call back the Host Target Mode peripheral driver using the callback field within the CONTINUE TARGET I/O CCB.

The Host Target Mode peripheral driver shall be responsible for the creation and preservation of the Contingent Allegiance condition if the CAM Status is not Unexpected Bus Free. If the CAM Status is not Unexpected Bus Free, it shall also complete the bus transaction by sending a CONTINUE TARGET I/O CCB to properly terminate this connection.

11.3.8.7 ACCEPT and CONTINUE TARGET I/O CCB timeouts

For connections that have not been fully identified the SIM/HBA shall use its default timeout value. Fully identified connections shall mean the following:

- Initial connection:
  - The IDENTIFY message has been processed for this connection without error with the ATN signal false.
  - The IDENTIFY message and the Queue Tag messages have been processed for this connection without error with the ATN signal false.

- Reconnection:
  - Shall be fully identified by a valid CONTINUE TARGET I/O CCB.

For initial connections if the SIM/HBAs default timeout period expires without full identification of the connection, the SIM/HBA shall behave as specified in Clause 11.3.3.1.1. When the initial connection has been fully identified

the SIM/HBA shall use the ACCEPT TARGET I/O CCB Timeout field value minus in seconds (if any) the selection time to full connection identification.

The reconnection timeout value shall be the valid CONTINUE TARGET I/O CCBs Timeout Value field.

Timeout periods specified in the CCB(s) shall be measured in seconds and shall be handled in the following manner when the connection has been fully identified:

A) ACCEPT TARGET I/O CCB Timeouts for Initial Connections

1) The timeout period shall be from SIM/HBA selection to just before Host Target Mode peripheral driver callback.

Note: For this example Disconnects are used.
- When the SIM/HBA is selected by an initiator, the timeout period starts.
- After the IDENTIFY message(s) and optional Queue Tag messages are received the SIM/HBA goes to COMMAND phase.
- The CDB is received and SIM/HBA disconnects.
- When the SIM/HBA disconnects from the bus (DISCONNECT message and BUS FREE phase).
- CCB Timeout Value - connection identification time (seconds) is positive, timeout period ends, no action is taken.
- The SIM/HBA calls back the Host Target Mode peripheral driver.
- If the timer expired before the callback then this ACCEPT TARGET I/O CCB represents a timeout condition.

For this example Disconnects are not used.
- When the SIM/HBA is selected by an initiator, the timeout period starts.
- After the IDENTIFY message(s) and optional Queue Tag messages are received the SIM/HBA goes to COMMAND phase.
- The CDB is received but SIM/HBA is not allowed to disconnect.
- CCB Timeout Value - connection identification time (seconds) is positive, timeout period ends, no action is taken.
- The SIM/HBA calls back the Host Target Mode peripheral driver.
- If the timer expired before the callback then this ACCEPT TARGET I/O CCB represents a timeout condition.

2) If the timeout period expires for the ACCEPT TARGET I/O CCB, the SIM/HBA shall set the CAM Status to Command Timeout.

3) The proper Connect ID shall be set in the ACCEPT TARGET I/O CCB. The SCSI BUS ID of the initiator that selected this SIM/HBA shall be set in the Initiator ID field.

4) The SIM/HBA shall cause an unexpected bus free. See ANSI X3.131-1994 for further details (BUS FREE).

5) Call back the Host Target Mode peripheral driver using the callback field within the ACCEPT TARGET I/O CCB.

B) CONTINUE TARGET I/O CCB Timeouts for Reconnections

1) The time period shall be measured from SIM/HBA reselection of the initiator to just before Host Target Mode peripheral driver call back.

2) If the timeout period expires for the CONTINUE TARGET I/O CCB, the SIM/HBA shall set the CAM Status to Command Timeout.

3) If a data transfer has been specified for this CCB, the Residual Length shall be set to the number of bytes not transferred.

4) The SIM/HBA shall cause an unexpected bus free. See ANSI X3.131-1994 regarding unexpected bus free.

5) Call back the Host Target Mode peripheral driver using the callback field within the CONTINUE TARGET I/O CCB.

The Host Target Mode peripheral driver shall be responsible for forming and maintaining sense data for all timeouts.

11.3.9 ACCEPT TARGET I/O CCB

The following lists the fields of the ACCEPT TARGET I/O CCB. No entry under the "Dir" heading indicates that this field is not being used for the Accept Target I/O function.

Table 27 - ACCEPT TARGET I/O CCB

Size	Dir	Accept Target I/O
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code (Accept Target I/O)
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID (Bus no. of SIM/HBA)
1	O	Target ID (Target ID of SIM/HBA)
1	O	LUN
4	I/O	CAM Flags
4	O	Peripheral Driver Pointer
4		Next CCB Pointer
4	O	Request Mapping Information (If needed)
4	O	Callback on Completion
4		SG List/Data Buffer Pointer
4		Data Transfer Length
4	O	Sense Info Buffer Pointer
1	O	Sense Info Buffer Length
1	O	CDB Length
2		Number of Scatter/Gather Entries
4		VU Field
1	I	SCSI Status
1		Autosense Residual Length
2		Reserved OSD
4		Residual Length
12	I/O	CDB
4	O	Timeout Value
4		Message Buffer Pointer
2		Message Buffer Length
2		VU Flags
1	I	Tag Queue Action
1	I	Tag ID
1	I	Initiator ID
1		Reserved
n		Private Data

The following lists the possible CAM Status values of the ACCEPT TARGET I/O CCB:

- Invalid Request - Indicates that the CCB was sent to a disabled LUN.
- CDB Received - Indicates that the CCB contains a CDB received from an initiator.
- Invalid CDB - Target Mode CDB error
- Request Aborted by Host
- SCSI Bus Reset Sent/Received - This SIM/HBA is recovering from a bus reset.
- Bus Device Reset Sent - This SIM/HBA is recovering from a BUS DEVICE RESET message.
- Command Timeout - Time period specified expired
- Path Invalid

- Unexpected Bus Free

The only valid bits for the CAM Flags field are the following:
- CDB is a Pointer - Indicates the CDB contained in the ACCEPT TARGET I/O CCB is a pointer. The CDB Length field shall indicate the size of the CDB buffer.
- CDB Physical - Indicates whether the pointer address is virtual or physical.
- Sense Buffer - Indicates whether the pointer address is virtual or physical.
- Callback on Comp.

11.3.10 CONTINUE TARGET I/O CCB

The following lists the fields of the CONTINUE TARGET I/O CCB. No entry under the "Dir" heading indicates that this field is not being used for the Continue Target I/O function.

Table 28 - CONTINUE TARGET I/O CCB

Size	Dir	Continue Target I/O
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code (Continue Target I/O)
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID (Bus no. of SIM/HBA)
1	O	Target ID (Target ID of SIM/HBA)
1	O	LUN
4	I/O	CAM Flags
4	O	Peripheral Driver Pointer
4		Next CCB Pointer
4	O	Request Mapping Information (if Needed)
4	O	Callback on Completion
4	O	SG List/Data Buffer Pointer
4	O	Data Transfer Length
4	O	Sense Info Buffer Pointer
1	O	Sense Info Buffer Length
1	O	CDB Length
2	O	Number of Scatter/Gather Entries
4		VU Field
1	I/O	SCSI Status (SCSI status to be sent to initiator)
1		Autosense Residual Length
2		Reserved OSD
4	I	Residual Length
12		CDB
4	O	Timeout Value
4		Message Buffer Pointer
2		Message Buffer Length
2		VU Flags
1	O	Tag Queue Action
1	O	Tag ID (Original tag value)
1	O	Initiator ID
1		Reserved
n		Private Data

The following lists the possible CAM Status values of the CONTINUE TARGET I/O CCB:
- Invalid Request - Indicates that the CCB was sent to a disabled LUN.
- Request in Progress - Request accepted by SIM/HBA.
- Request Completed with Error - Indicates that the CCB is not properly setup.
- Request Aborted by Host.
- Request Completed without Error - Request completed successfully.
- SCSI Bus Reset Sent/Received - This SIM/HBA is recovering from a bus reset.

- No HBA Detected - The HBA is no longer responding.
- Bus Device Reset Sent - This SIM is recovering from a BUS DEVICE RESET message.
- Target Bus Phase Sequence Failure
- Uncorrectable Parity Error Detected
- Initiator Detected Error
- Unexpected Bus Free
- Target Selection Timeout - Initiator failed to response to selection
- Command Timeout - Time period specified expired
- Path Invalid
- Unacknowledged Event by Host - An event has not been acknowledged by the Host Target Mode peripheral driver.

The following bits are valid for the CAM Flags field of the CONTINUE TARGET I/O CCB:

- Direction Out - Sending data from the target peripheral device to the initiator.
  + SG List/Data Buffer Pointer and Data Transfer Length fields shall be filled in.
- Direction In - Receiving data from the initiator to the target peripheral device.
  + SG List/Data Buffer Pointer and Data Transfer Length fields shall be filled in.
- No Direction - No data phase required.
- Scatter/Gather - Indicates that the Data Buffer Pointer is a pointer to a scatter/gather list.
- Tagged Queue Action Enable - Indicates this is a tagged request and the Tag ID shall be filled in.
- Sense Buffer - Indicates whether the pointer address is virtual or physical.
- Callback on Completion.

11.3.11 Disable of a host target mode LUN

When a Host Target Mode peripheral driver wishes to disable Host Target Mode for an enabled LUN, it shall perform the following tasks:

A) Issue an ABORT CCB for all ACCEPT TARGET I/O CCBs which were sent to the SIM that have not been returned.

B) Wait for processing of all CONTINUE TARGET I/O CCBs sent to the SIM to be completed.

C) Once all CCBs are owned by the Host Target Mode peripheral driver, it shall issue an ENABLE LUN CCB to the SIM/HBA with the Number of Target CCBs equal to zero (indicating the LUN should be disabled).

D) Upon successful completion of the disable for the Host Target Mode LUN (indicated by a CAM Status of Request Completed without Error) the Host Target Mode peripheral driver shall own all IMMEDIATE NOTIFY CCBs it sent to the SIM/HBA. The Host Target Mode peripheral driver may now free the CCBs if it so desires.

When the SIM receives an ENABLE LUN CCB with the Number of Target CCBs equal to zero (disable LUN) it shall perform the following tasks:

A) If the LUN was never enabled, then the ENABLE LUN CCB shall be returned with a CAM Status of Invalid Request.

B) If there are ACCEPT TARGET I/O CCBs or CONTINUE TARGET I/O CCBs still owned by the SIM, then the disable LUN request shall fail with Request Completed with Error.

C) The SIM/HBA unacknowledged event list shall be cleared.

D) Upon successful completion of the disable for the Host Target Mode LUN, the ENABLE LUN CCB shall be returned with the CAM Status field set to Request Completed without Error.

E) All subsequent ACCEPT TARGET I/O and CONTINUE TARGET I/O CCBs received by the SIM prior to the receipt of another ENABLE LUN CCB that enables the LUN shall be returned with a CAM Status of Invalid Request.

11.3.12 Exception conditions

11.3.12.1 BUS RESET

When a bus reset (hard reset) is sent or received, the following sequence of events shall occur:

A) The SIM/HBA shall:

1) Clear any outstanding target I/O's owned by the SIM/HBA for the bus that suffered the reset by calling the callback completion function with a CAM Status of SCSI Bus Reset Sent/Received. Requests owned by the SIM/HBA are:

a) Any I/O currently on the bus.

b) Any Continue Target I/O requests which are currently queued in the SIM/HBA.

2) Cause the Host Target Mode peripheral driver asynch callback function to be called as part of the normal asynch callback notification.

3) All initial connections shall complete with a SCSI status of BUSY, until the SIM/HBA receives an applicable NOTIFY ACKNOWLEDGE CCB.

4) All CONTINUE TARGET I/O CCBs that are received by the SIM while the reset recovery is in progress shall be returned with a CAM Status of SCSI Bus Reset Sent/Received.

5) Reset recovery shall be complete when the Host Target Mode peripheral driver issues a NOTIFY ACKNOWLEDGE CCB with the Reset Cleared field set, and the Sequence Identifier equal to zero.

6) The SIM/HBA unacknowledged event list shall be cleared for all enabled LUNs on the bus which experienced the bus reset.

B) The Host Target Mode peripheral driver asynch callback routine shall:

1) Clear all pending sense data.

2) Cease processing on any outstanding requests owned by the Host Target Mode peripheral driver. These are ACCEPT TARGET I/O or CONTINUE TARGET I/O CCBs being processed by the Host Target Mode peripheral driver.

3) For all initiators, sense data shall be formed and saved indicating the UNIT ATTENTION condition caused as a result of the bus reset. See ANSI X3.131-1994 for further details on forming and clearing the contingent allegiance condition.

4) Issue a NOTIFY ACKNOWLEDGE CCB with the Reset Cleared field set and the Sequence Identifier equal to zero.

5) If necessary, the Host Target Mode peripheral driver shall issue/reissue ACCEPT TARGET I/O CCB(s) so that the SIM/HBA can resume normal processing.

11.3.12.2 BUS DEVICE RESET message

When a BUS DEVICE RESET message is sent/received, the following sequence of events shall occur:

A) The SIM/HBA shall:

1) Clear any outstanding target I/O's owned by the SIM/HBA for the target that received the BUS DEVICE RESET message by calling the callback completion function with a CAM Status of Bus Device Reset. Requests owned by the SIM/HBA are:

a) Any I/O currently on the bus.

b) Any Continue Target I/O requests which are currently queued in the SIM/HBA.

2) Cause the Host Target Mode peripheral driver asynch callback function to be called as part of the normal asynch callback notification.

3) All initial connections shall complete with a SCSI status of BUSY, until the SIM/HBA receives an applicable NOTIFY ACKNOWLEDGE CCB.

4) All CONTINUE TARGET I/O CCBs that are received by the SIM while the reset recovery is in progress shall be returned with a CAM Status of Bus Device Reset.

5) Reset recovery shall be complete when the Host Target Mode peripheral driver issues a NOTIFY ACKNOWLEDGE CCB with the Reset Cleared field set and the Sequence Identifier equal to zero.

6) The SIM/HBA unacknowledged event lists shall be cleared for all enabled LUNs on the target that received the BUS DEVICE RESET message.

B) The Host Target Mode peripheral driver asynch callback routine shall:

1) Clear all pending sense data.

2) Cease processing on any outstanding requests owned by the Host Target Mode peripheral driver. These are ACCEPT TARGET I/O or CONTINUE TARGET I/O CCBs being processed by the Host Target Mode peripheral driver.

3) For all initiators, sense data shall be formed and saved indicating the UNIT ATTENTION condition caused as a result of the bus reset. See ANSI X3.131-1994 for further details on forming and clearing the contingent allegiance condition.

4) Issue a NOTIFY ACKNOWLEDGE CCB with the Reset Cleared field set and the Sequence Identifier equal to zero.

5) If necessary, the Host Target Mode peripheral driver shall issue/reissue ACCEPT TARGET I/O CCB(s) so that the SIM/HBA can resume normal processing.

11.3.13 CDB reception on a non enabled LUN

When a CDB is received by a SIM/HBA for a LUN that is not enabled, one of the following sequences shall occur depending on the command received:

A) INQUIRY Command
If the SIM receives a CDB for the INQUIRY command for a non-enabled LUN, the SIM shall return only byte 0 of the inquiry data set to 23H:
- The peripheral qualifier is set to 01B indicating the target is capable of supporting a physical device on this logical unit, however the physical device is not currently connected to this LUN.
- The peripheral device type set to 3H indicating Processor device type.

B) REQUEST SENSE Command
If a REQUEST SENSE command is received for a non-enabled LUN, the SIM shall return sense data in which the sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to LOGICAL UNIT NOT SUPPORTED.

C) All other commands
If a command other than INQUIRY or REQUEST SENSE is received for a non-enabled LUN, the SCSI status returned shall be CHECK CONDITION. Any subsequent REQUEST SENSE command shall behave as in Item B.

11.3.14 Retrieving unused ACCEPT TARGET I/O CCBs from the SIM

If a Host Target Mode peripheral driver wishes to retrieve an ACCEPT TARGET I/O CCB which was sent to the SIM (probably due to the lack of resources) the Host Target Mode peripheral driver may issue an ABORT CCB. This shall result in the CDB Received Completion callback function of the ACCEPT TARGET I/O CCB being called with a CAM Status of Request Aborted by Host.

12 HBA engines

An engine is a hardware device implemented in an HBA to perform time-intensive functions not available on target devices. Generally, these engines are required to process data prior to building a CDB and submitting it to a Logical Unit. There may be more than one engine in a peripheral HBA.

One use of engines is to compress data. In this mode, a device driver first submits data to the engine using the EXECUTE ENGINE REQUEST CCB. Once the engine has completed processing the data, an EXECUTE SCSI I/O REQUEST CCB then can be built for the compressed data and sent to a SIM/HBA.

The engine model allows for the addressing of buffer memory located on the HBA. The buffer addressing appears to the host as contiguous space. Using this model, it is possible to submit multiple EXECUTE ENGINE REQUEST CCBs until the engine buffer is full. Once the full condition is met, an EXECUTE SCSI I/O REQUEST CCB then can be built for the compressed data and sent to a SIM/HBA.

When the full condition occurs (as defined by the destination data length equalling the destination data maximum length), the amount of unprocessed source data is reported in the source residual length. The residual data may then be re-submitted at a later time.

12.1 Engine inquiry

This function is used to gather information about the data processing engines installed in the HBA hardware.

Table 29 - ENGINE INQUIRY CCB

Size	Dir	Engine inquiry
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function code
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID
1		Target ID
1		LUN
4	O	CAM Flags
2	O	Engine number
1	I	Engine type 0=Buffer memory 1=Lossless compression 2=Lossy compression 3=Encryption 4-FF reserved
1	I	Engine algorithm ID 0=Vendor Unique 1=LZ1 variation 1 (STAC) 2=LZ2 variation 1 (HP DCZL) 3=LZ2 variation 2 (infochip) 4-FF reserved
4	I	Engine memory size

The engine type reports the generic function the addressed engine is capable of supporting.

The engine algorithm ID reports the specific capability the addressed engine supports.

The amount of buffer memory provided for an engine is reported in the engine memory size.

This function shall return a CAM Status other than Request in Progress.

- CAM Status of Request Completed without Error indicates that the other returned fields are valid.
- CAM Status of Invalid Request indicates that the specified engine number is not installed.
- CAM Status of Cannot Provide Requested Capability indicates that the SIM/HBA does not support this function.

12.2 Execute engine (optional)

To accommodate buffering associated with the engine, the CAM Flag SG list/data (bit 7) set to 1=engine is used to specify that the normal data buffer pointer is actually a physical address in the buffer space of the engine. The CAM Flag SG List/Data (bit 5) shall signify whether the address(es) is Virtual or Physical.

There are four modes associated with engine processing established by CAM Flags:

- A direction setting of out is used to encrypt or compress the data
- A direction setting of in is used to decrypt or decompress the data
- Synchronize is used in conjunction with in or out to flush any residual bits prior to terminating engine processing.

The EXECUTE ENGINE REQUEST CCB activates the engine to perform the requested function. Some functions change the data size (e.g., a compression engine reduces the size of data prior to transmission over the SCSI bus).

Table 30 - EXECUTE ENGINE REQUEST CCB

Size	Dir	Execute Engine
4	O	Address of this CCB
2	O	CAM Control Block Length
1	O	Function Code
1	I	CAM Status
		Connect ID
1		Reserved
1	O	Path ID
1	O	Target ID
1	O	LUN
4	O	CAM Flags
4	O	Peripheral driver pointer
4	O	Reserved
4	O	Request mapping information
4	O	Callback on completion
4	O	SG list/data buffer pointer
4	O	Data transfer length
4	O	Engine buffer data pointer
1		Reserved
1		Reserved
2	O	Number of scatter/gather entries
4	O	Destination data maximum length
4	I	Destination data length
4	I	Source residual length
12		Reserved
4	O	Timeout value
4		Reserved
2	O	Engine number
2	O	VU flags
1		Reserved
3		Reserved
n	O	Private data

This function typically returns with CAM Status of In Progress indicating that the request was queued successfully. Function completion can be ascertained by polling for non-zero status (non Request in Progress status) or through use of the Callback on Completion field.

The final CAM Status shall be one of the following:

- Request Completed without Error: the request has completed and no error condition was encountered.
- Request Aborted by Host: the request was aborted by the SIM/HBA.
- Unable to Abort Request: the SIM/HBA was unable to abort the request as instructed by the peripheral driver.
- Request Completed with Error: the request has completed and an error condition was encountered.
- CAM Busy: CAM unable to accept request at this time.
- Invalid Request: the request has been rejected because it is invalid.
- Invalid Path ID: indicates that the Path ID is invalid.
- Unable to Terminate I/O Process: the SIM/HBA was unable to terminate the request as instructed by the peripheral driver.
- Command Timeout: the specified command did not complete within the timer value specified in the CCB. Prior to reporting this status the SIM/HBA shall ensure the command is no longer active in the HBA.
- No HBA Detected: HBA no longer responding to SIM (assumed to be a hardware problem).
- Data Overrun: target transferred more data bytes than peripheral driver indicated in the CCB.

- CCB Length Inadequate: more private data area is required in the CCB (see Clause 9.2.3 for further information).
- Cannot Provide Requested Capability: resources are not available to provide the capability requested in the CAM Flags.
- Terminate I/O Process: this CCB was terminated because a Terminate I/O Process function was specified for this CCB and the CCB was not an I/O process within the Logical Unit.
- Unrecoverable Host Bus Adaptor Error: this CCB was terminated because of a hardware error detected by the HBA.

Annex A
(informative)
Physical/logical translation in 80x86 environment

A.1 OSD formatting of disk drives

The DOS physical address to/from logical block address conversion algorithms to map SCSI disks into int 13h head-cylinder-sector format vary widely between suppliers of software to support third party disks.

The following "C" routines have been adopted by CAM as representing the most efficient utilization of capacity. The following code is ANSI "C" that can be compiled using the Microsoft C compiler, version 5.1.

a) SETSIZE converts a read capacity value to int 13h head-cylinder-sector requirements. It minimizes the value for number of heads and maximizes the number of cylinders. This supports very large disks before the number of heads will not fit in 4 bits (or 6 bits). This algorithm also minimizes the number of sectors that are unused at the end of the disk while allowing for large disks to be accommodated. This algorithm does not use physical geometry.

b) LTOP does logical to physical conversion

c) PTOL does physical to logical conversion

d) MAIN is a test routine for a, b and c.

A.1.1 SETSIZE

```
typedef unsigned int UINT;
typedef unsigned long ULNG;
/*
 *  Convert from logical block count to cylinder, sector and head (int 13)
 */

 int setsize(ULNG capacity,UINT *cyls,UINT *hds,UINT *secs)

{
  UINT rv = 0;
  ULNG heads, sectors, cylinders, temp;

  cylinders = 1024L;              /* Set number of cylinders to max value */
  sectors = 62L;                  /* Max out number of sectors per track */

  temp = cylinders * sectors;     /* Compute divisor for heads */
  heads = capacity / temp;        /* Compute value for number of heads */
  if (capacity % temp) {          /* If no remainder, done! */
    heads++;                      /* Else, increment number of heads */
    temp = cylinders * heads;     /* Compute divisor for sectors */
    sectors = capacity / temp;    /* Compute value for sectors per track */
    if (capacity % temp) {        /* If no remainder, done! */
      sectors++;                  /* Else, increment number of sectors */
```

```
      temp = heads * sectors;          /* Compute divisor for cylinders */
      cylinders = capacity / temp;     /* Compute number of cylinders */
      }
    }
  if (cylinders == 0) rv=1;          /* Give error if 0 cylinders */

  *cyls = (UINT) cylinders;          /* Stuff return values */
  *secs = (UINT) sectors;
  *hds  = (UINT) heads;
  return(rv);
}
```

A.1.2 LTOP

```
/*
 *  logical to physical conversion
 */

void ltop(ULNG block,UINT hd_count,UINT sec_count,UINT *cyl,UINT *hd,UINT
*sec)

{
  UINT spc;
  spc = hd_count * sec_count;
  *cyl = block / spc;
  *hd =  (block % spc) / sec_count;
  *sec = (block % spc) % sec_count;
}
```

A.1.3 PTOL

```
/*
 *     Physical to logical conversion
 */

ULNG ptol(UINT cyl,UINT hd,UINT sec,UINT cyl_count,UINT hd_count,UINT
sec_count)

{
  ULNG cylsize;
  cylsize = sec_count * hd_count;
  return((cyl * cylsize) + (hd * sec_count) + sec);
}
```

A.2 Backwards compatibility

The selection of a new algorithm for CAM solves the problem of future compatibility, but it does not solve the problem of the installed base. The following techniques are an example of how a supplier can update the installed base to CAM-compliant operation but not require users to reformat their drives. These techniques are suitable for support of more than one device, as long as the number of sectors per track is the same on all devices.

A.2.1 ROM-based

The one sector that is independent of the algorithm is sector 00. Under DOS and many other operating systems this sector is used for the boot sector and contains the partition table for a fixed disk.

If the partition table is structured according to MS DOS and IBM DOS standards, partitions end on cylinder boundaries e.g.

Offset from start of partition	Table entry
00h Boot indicator	80h
01h Beginning or start head	01h
02h Beginning or start sector	01h
03h Beginning or start cylinder	00h
04h System indicator	04h
05h Ending head	07h
06h Ending sector	91h
07h Ending cylinder	7Ch
08h Starting sector (relative to beginning of disk)	
0Ch Number of sectors in partition	

The ending head 07h indicates a device with 8 heads (0 to 7). The ending sector 91h contains 2 bits of high cylinder so it has to be masked to obtain ending sector = 11h (17 decimal).

To verify these values calculate:

Logical ending sector (from beginning head, cylinder, and sector)

and compare it to:

(Starting sector + number of sectors in partition)

This leaves number of cylinders as the one unresolved parameter. This is obtained by:

Read capacity divided by (heads * sectors).

All of this can be done by the BIOS in ROM or RAM. To be capable of booting from any drive or cartridge regardless of the algorithm used to partition and format the media, the BIOS would need to respond to int 13 function 8 with the head, sector, and cylinder values obtained from this information. In addition, the BIOS would need to use those values in its calculation from physical to logical sectors.

Example of pseudocode:

```
For each drive
    Read boot sector (LBA 0)
    Validate the signature at end of sector (55AA)
    Find partition with largest logical start cyl

        If no partitions found
        Use defaults
        Exit

      SECS = Ending sector (from partition table)
      Heads = Ending head+1  (from partition table)
```

```
Logical_end =  end_cyl * (end_head+1 * end_sector) +
               (end_head * end_sector) + end_sector

Compare logical_end to starting_sec + number_sec
   If not equal
   Use defaults
   Exit

Cyls = capacity / (end_head+1 * end_sector)
```

A.2.2 RAM-based

Under DOS it is possible to modify the code of the boot sector to accomplish bootability. Access to other partitions is dependent on the device driver to do a translation.

This method is a patch just prior to jumping to code loaded in memory at segment 00 offset 7C00h.

```
PUSH   AX            ; Save registers used in patch
PUSH   DX
MOV      AH,08         ; set function code = 8 get drive parameters
INT    13            ; do INT 13 call
INC    DH            ; inc head number to convert from zero based
MOV      [7C1A],DH     ; fix value of heads in BPB table
AND    CL,3F         ; Mask off non-sector information
MOV      [7C18],CL     ; fix value of sectors in BPB table
POP    DX
POP    AX            ; Restore registers used in patch
JMP    7C00          ; jump to partition boot loader
```

```
01B0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 80 01
01C0  01 00 06 07 91 7C 11 00-00 00 57 52 01 00 00 00
01D0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
01E0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
01F0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 55 AA
```

7C00	7C03	7C0B	7C0D	7C0E
jump nop	Name I B M 4 . 0	Bytes/ sector	Sectors/ cluster	Reserved sectors
EB 3C 90	49 42 4D 20 20-34 2E 30	00 02	04	01 00

7C10	7C11	7C13	7C15	7C16	7C18	7C1A
# FATs 02	# DIR entries 00 02	# Log'l sectors 00 00	Media descrip F8	# FAT sectors 55 00	# Sectors 11 00	# Heads 08 00

Annex B
(informative)
Target peripheral driver example

The following are examples of how a target peripheral can operate the target mode capabilities defined in clause 11.

Phase-cognizant examples

Initialization sequence with single target CCB provided

- fill target1CCB with required info
  target1CCB.callbackPointer = callback routine address #1
- targetCCBList [0] = pointer to target1CCB

NOTE: where targetCCBList is an array of pointers

- fill enable CCB with the required information
  enableCCB.functionCode = function code for enable lun
  enableCCB.targetid = the id of the target
  enableCCB.targetLUN = the lun to enable
  enableCCB.group6VULength = vendor unique length for group 6 (IF required)
  enableCCB.group7VULength = vendor unique length for group 7 (IF required)
  enableCCB.targetCCBListLength = 1
  enableCCB.targetCCBPointer = &targetCCBList
- Enable LUN (&enableCCB)
- EXIT

Initialization sequence with multiple target CCBs provided

- fill target CCB #1 with required info
  target1CCB.callbackPointer = callback routine address #1
- fill target CCB #2 with required info
  target2CCB.callbackPointer = callback routine address #2
- fill target CCB #n with required info
  targetnCCB.callbackPointer = callback routine address #n
- targetCCBList [0] = pointer to target1CCB
- targetCCBList [1] = pointer to target2CCB
- targetCCBList [n-1] = pointer to targetnCCB

NOTE: where targetCCBList is an array of pointers

- fill enable CCB with the required information
  enableCCB.functionCode = function code for enable lun
  enableCCB.targetid = the id of the target
  enableCCB.targetLUN = the lun to enable
  enableCCB.group6VULength = vendor unique length for Group 6 (IF required)
  enableCCB.group7VULength = vendor unique length for Group 7 (IF required)
  enableCCB.targetCCBListLength = n, where n is the number of target CCBs

enableCCB.targetCCBPointer = &targetCCBList
- Enable LUN (&enableCCB)
- EXIT

Peripheral driver sequence with single execute target I/O

- loop until target1CCB.camFlags TargetCCB Available bit is reset, OR callback routine called from SIM/HBA
- process fields in target1CCB

/* return target CCB to pool */

- set target1CCB.camFlags TargetCCB Available bit
- fill executetarget1CCB with required information
  executetarget1CCB.functionCode = function code for execute target io
  executetarget1CCB.camFlags = data phase and status phase
  executetarget1CCB.dataBufferPointerLength = length of data
  executetarget1CCB.dataBufferPointer = pointer to data buffer
  executetarget1CCB.scsiStatus = whatever status is appropriate
- Execute Target I/O (&executetargetCCB)

Peripheral driver sequence with multiple execute target I/O

- loop until targetxCCB.camFlags TargetCCB Available bit is reset, OR callback routine called from SIM/HBA (where x is one of the targetCCBs provided in targetCCBList)
- process fields in targetxCCB

/* return target CCB to pool */

- set targetxCCB.camFlags TargetCCB available bit
- loop until all data transferred
  fill executetargetxCCB with required information
  executetargetxCCB.functionCode = function code for execute target io
  executetargetxCCB.camFlags = data phase
  executetargetxCCB.dataBufferPointerLength = length of data
  executetargetxCCB.dataBufferPointer = pointer to data buffer

  IF (last data block)
  executetargetxCCB.camFlags = data phase AND status phase
  executetargetxCCB.scsiStatus = whatever status is appropriate

  Execute Target I/O (&executetargetxCCB)
- end loop

Annex C
(informative)
UNIVOS OSD data structures

This Annex contains an example of definitions and data structures for the CAM Subsystem interface. The contents of this example should match the data structures and constants that are specified in this standard.

```
/* ---------------------------------------------------------------------- */

/* Defines for the XPT function codes in the CAM spec. */

/* Common function commands,        0x00 - 0x0F */
#define XPT_NOOP                    0x00  /* Execute nothing */
#define XPT_SCSI_IO                 0x01  /* Execute the requested SCSI IO */
#define XPT_GDEV_TYPE               0x02  /* Get the device type information */
#define XPT_PATH_INQ                0x03  /* Path Inquiry */
#define XPT_REL_SIMQ                0x04  /* Release the SIM queue that is frozen */
#define XPT_SASYNC_CB               0x05  /* Set Asynchronous Callback parameters */
#define XPT_SDEV_TYPE               0x06  /* Set the device type information */
#define XPT_SCAN_BUS                0x07  /* Scan the SCSI Bus */

/* XPT SCSI control functions,      0x10 - 0x1F */
#define XPT_ABORT                   0x10  /* Abort the selected CCB */
#define XPT_RESET_BUS               0x11  /* Reset the SCSI bus */
#define XPT_RESET_DEV               0x12  /* Reset the SCSI device, BDR */
#define XPT_TERM_IO                 0x13  /* Terminate the I/O process */
#define XPT_SCAN_LUN                0x14  /* Scan Logical Unit */

/* HBA engine commands,             0x20 - 0x2F */
#define XPT_ENG_INQ                 0x20  /* HBA engine inquiry */
#define XPT_ENG_EXEC                0x21  /* HBA execute engine request */

/* Target mode commands,            0x30 - 0x3F */
#define XPT_EN_LUN                  0x30  /* Enable LUN, target mode support */
#define XPT_TARGET_IO               0x31  /* Execute the target IO request */
#define XPT_ACCEPT_TARGET_IO        0x32  /* Accept Host Target Mode CDB */
#define XPT_CONT_TARGET_IO          0x33  /* Continue Host Target I/O Connection */
#define XPT_IMMED_NOTIFY            0x34  /* Notify Host Target driver of event */
#define XPT_NOTIFY_ACK              0x35  /* Acknowledgement of event */

#define XPT_FUNC                    0x7F  /* TEMPLATE */
#define XPT_VUNIQUE                 0x80  /* All the rest are vendor unique commands */

/* ---------------------------------------------------------------------- */

/* General allocation length defines for the CCB structures. */

#define IOCDBLEN            12      /* Space for the CDB bytes/pointer */
```

```
#define VUHBA           14    /* Vendor Unique HBA length */
#define SIM_ID          16    /* ASCII string len for SIM ID */
#define HBA_ID          16    /* ASCII string len for HBA ID */
#define SIM_PRIV        50    /* Length of SIM private data area */

/* Structure definitions for the CAM control blocks, CCB's for the subsystem. */

/* Common CCB header definition. */
typedef struct ccb_header
{
    struct ccb_header *my_addr;   /* The address of this CCB */
    u_short cam_ccb_len;          /* Length of the entire CCB */
    u_char cam_func_code;         /* XPT function code */
    u_char cam_status;            /* Returned CAM subsystem status */
    u_char cam_hrsvd0;            /* Reserved field */
    u_char cam_path_id;           /* Path ID for the request */
    u_char cam_target_id;         /* Target device ID */
    u_char cam_target_lun;        /* Target LUN number */
    u_long cam_flags;             /* Flags for operation of the subsystem */
} CCB_HEADER;

/* Common SCSI functions. */

/* Union definition for the CDB space in the EXECUTE SCSI I/O REQUEST CCB*/
typedef union cdb_un
{
    u_char *cam_cdb_ptr;                    /* Pointer to the CDB bytes to send */
    u_char cam_cdb_bytes[ IOCDBLEN ];       /* Area for the CDB to send */
} CDB_UN;

/* GET DEVICE TYPE CCB*/
typedef struct ccb_getdev
{
    CCB_HEADER cam_ch;              /* Header information fields */
    u_char *cam_inq_data;           /* Ptr to the inquiry data space */
    u_char cam_pd_type;             /* Periph device type from the TLUN */
} CCB_GETDEV;
```

```
/* PATH INQUIRY CCB */
typedef struct ccb_pathinq
{
    CCB_HEADER cam_ch;              /* Header information fields */
    u_char cam_version_num;         /* Version number for the SIM/HBA */
    u_char cam_hba_inquiry;         /* Mimic of INQ byte 7 for the HBA */
    u_char cam_target_sprt;         /* Flags for target mode support */
    u_char cam_hba_misc;            /* Misc HBA feature flags */
    u_short cam_hba_eng_cnt;        /* HBA engine count */
    u_char cam_vuhba_flags[ VUHBA ];/* Vendor Unique capabilities */
    u_long cam_sim_priv;            /* Size of SIM private data area */
    u_long cam_async_flags;         /* Event cap. for async callback */
    u_char cam_hpath_id;            /* Highest Path ID in the subsystem */
    u_char cam_initiator_id;        /* ID of the HBA on the SCSI bus */
    u_char cam_prsvd0;              /* Reserved field, for alignment */
    u_char cam_prsvd1;              /* Reserved field, for alignment */
    char cam_sim_vid[ SIM_ID ];     /* Vendor ID of the SIM */
    char cam_hba_vid[ HBA_ID ];     /* Vendor ID of the HBA */
    u_char *cam_osd_usage;          /* Ptr for the OSD specific area */
} CCB_PATHINQ;

/* RELEASE SIM QUEUE CCB */
typedef struct ccb_relsim
{
CCB_HEADER cam_ch;                  /* Header information fields */
u_long cam_frzn_cnt;                /* Frozen count of the queue */
} CCB_RELSIM;
```

```
/* EXECUTE SCSI I/O REQUEST CCB*/
typedef struct ccb_scsiio
{
CCB_HEADER cam_ch;                  /* Header information fields */
u_char *cam_pdrv_ptr;               /* Ptr used by the peripheral driver */
CCB_HEADER *cam_next_ccb;           /* Ptr to the next CCB for action */
u_char *cam_req_map;                /* Ptr for mapping info on the Req. */
void (*cam_cbfcnp)();               /* Callback on completion function */
u_char *cam_data_ptr;               /* Pointer to the data buf/SG list */
u_long cam_dxfer_len;               /* Data xfer length */
u_char *cam_sense_ptr;              /* Pointer to the sense data buffer */
u_char cam_sense_len;               /* Num of bytes in the autosense buf */
u_char cam_cdb_len;                 /* Number of bytes for the CDB */
u_short cam_sglist_cnt;             /* Num of scatter gather list entries */
u_long cam_osd_rsvd0;               /* OSD reserved field, for alignment */
u_char cam_scsi_status;             /* Returned SCSI device status */
u_char cam_sense_resid;             /* Autosense resid length: 2's comp */
u_char cam_osd_rsvd1[2];            /* OSD reserved field, for alignment */
u_long cam_resid;                   /* Transfer residual length: 2's comp */
CDB_UN cam_cdb_io;                  /* Union for CDB bytes/pointer */
u_long cam_timeout;                 /* Timeout value */
u_char *cam_msg_ptr;                /* Pointer to the message buffer */
u_short cam_msgb_len;               /* Num of bytes in the message buf */
u_short cam_vu_flags;               /* Vendor Unique flags */
u_char cam_tag_action;              /* What to do for tag queuing */
u_char cam_tag_id;                  /* tag id from initiator (target mode) */
u_char cam_init_id;                 /* initiator id of who selected */
u_char cam_iorsvd0[1];              /* Reserved field, for alignment */
u_char cam_sim_priv[ SIM_PRIV ];    /* SIM private data area */
} CCB_SCSIIO;

/* SET ASYNCHRONOUS CALLBACK CCB */
typedef struct ccb_setasync
{
    CCB_HEADER cam_ch;              /* Header information fields */
    u_long cam_async_flags;         /* Event enables for callback resp */
    void (*cam_async_func)();       /* Async callback function address */
    u_char *pdrv_buf;               /* Buffer set aside by the per. drv */
    u_char pdrv_buf_len;            /* The size of the buffer */
} CCB_SETASYNC;

/* SET DEVICE TYPE CCB */
typedef struct ccb_setdev
{
    CCB_HEADER cam_ch;          /* Header information fields */
    u_char cam_dev_type;        /* Val for the dev type field in EDT */
} CCB_SETDEV;
```

```
    /* SCSI control functions. */

    /* Abort XPT request CCB */
    typedef struct ccb_abort
    {
        CCB_HEADER cam_ch;              /* Header information fields */
        CCB_HEADER *cam_abort_ch;       /* Pointer to the CCB to abort */
    } CCB_ABORT;

    /* RESET SCSI BUS CCB */
    typedef struct ccb_resetbus
    {
        CCB_HEADER cam_ch;              /* Header information fields */
    } CCB_RESETBUS;

    /* RESET SCSI DEVICE CCB */
    typedef struct ccb_resetdev
    {
        CCB_HEADER cam_ch;              /* Header information fields */
    } CCB_RESETDEV;

    /* TERMINATE I/O PROCESS REQUEST CCB */
    typedef struct ccb_termio
    {
        CCB_HEADER cam_ch;              /* Header information fields */
        CCB_HEADER *cam_termio_ch;      /* Pointer to the CCB to terminate */
    } CCB_TERMIO;

    /* Target mode structures. */

    typedef struct ccb_en_lun
    {
        CCB_HEADER cam_ch;              /* Header information fields */
        u_short cam_grp6_len;           /* Group 6 VU CDB length */
        u_short cam_grp7_len;           /* Group 7 VU CDB length */
        u_char *cam_notify_listptr;     /* Pointer to Immediate Notify List */
        u_long cam_notify_listcnt;      /* Count of Immediate Notify CCBs */
        u_char  *cam_ccb_listptr;       /* Pointer to the target CCB list */
        u_short cam_ccb_listcnt;        /* Count of target CCBs in the list */
        u_char Reserved[2];             /* Reserved */
        u_char cam_sim_priv[ SIM_PRIV ];/* SIM private data area */
} CCB_EN_LUN;
```

```
typedef struct ccb_immed_notify
    {
        CCB_HEADER cam_ch;              /* Header information fields */
        u_char *cam_sense_ptr;          /* Pointer to the sense data buffer */
        u_char cam_sense_len;           /* Num of bytes in the sense buf */
        u_char Reserved[3];             /* Reserved bytes */
        void (*cam_cbfcnp)();           /* Callback on notification function */
        u_char cam_initiator_id;        /* Id of initiator that selected */
        u_char Reserved;                /* Reserved */
        u_short cam_seq_id;             /* Sequence Identifier */
        u_char cam_message_code;        /* Message Code */
        u_char cam_message_args[7];     /* Message Arguments */
} CCB_IMMED_NOTIFY;


typedef struct ccb_notify_ack
    {
        CCB_HEADER cam_ch;              /* Header information fields */
        u_short cam_seq_id;             /* Sequence Identifier */
        u_char cam_event;               /* Event flags */
        u_char Reserved;                /* Reserved */
} CCB_NOTIFY_ACK;

    /* HBA engine structures. */

    typedef struct ccb_eng_inq
    {
        CCB_HEADER cam_ch;              /* Header information fields */
        u_short cam_eng_num;            /* The number for this inquiry */
        u_char cam_eng_type;            /* Returned engine type */
        u_char cam_eng_algo;            /* Returned algorithm type */
        u_long cam_eng_memory;          /* Returned engine memory size */
    } CCB_ENG_INQ;
```

```
typedef struct ccb_eng_exec             /* This must match SCSIIO size */
{
    CCB_HEADER cam_ch;                  /* Header information fields */
    u_char *cam_pdrv_ptr;               /* Ptr used by the peripheral driver */
    u_long cam_engrsvd0;                /* Reserved field, for alignment */
    u_char *cam_req_map;                /* Ptr for mapping info on the req. */
    void (*cam_cbfcnp)();               /* Callback on completion function */
    u_char *cam_data_ptr;               /* Pointer to the data buf/SG list */
    u_long cam_dxfer_len;               /* Data xfer length */
    u_char *cam_engdata_ptr;            /* Pointer to the engine buffer data */
    u_char cam_engrsvd1;                /* Reserved field, for alignment */
    u_char cam_engrsvd2;                /* Reserved field, for alignment */
    u_short cam_sglist_cnt;             /* Num of scatter gather list entries */
    u_long cam_dmax_len;                /* Destination data maximum length */
    u_long cam_dest_len;                /* Destination data length */
    long cam_src_resid;                 /* Source residual length: 2's comp */
    u_char cam_engrsvd3[12];            /* Reserved field, for alignment */
    u_long cam_timeout;                 /* Timeout value */
    u_long cam_engrsvd4;                /* Reserved field, for alignment */
    u_short cam_eng_num;                /* Engine number for this request */
    u_short cam_vu_flags;               /* Vendor Unique flags */
    u_char cam_engrsvd5;                /* Reserved field, for alignment */
    u_char cam_engrsvd6[3];             /* Reserved field, for alignment */
    u_char cam_sim_priv[ SIM_PRIV ];    /* SIM private data area */
} CCB_ENG_EXEC;

/* The CAM_SIM_ENTRY definition is used to define the entry points for the SIMs contained in the SCSI CAM
subsystem.  Each SIM file contains a declaration for it's entry.  The address for this entry is stored in the
cam_conftbl[] array along with all the other SIM entries. */

typedef struct cam_sim_entry
{
    long (*sim_init)();                 /* Pointer to the SIM init routine */
    long (*sim_action)();               /* Pointer to the SIM CCB go routine */
} CAM_SIM_ENTRY;

/* ---------------------------------------------------------------------- */

/* Defines for the CAM Status field in the CCB header. */

#define CAM_REQ_INPROG          0x00  /* CCB request is in progress */
#define CAM_REQ_CMP             0x01  /* CCB request completed w/out error */
#define CAM_REQ_ABORTED         0x02  /* CCB request aborted by the host */
#define CAM_UA_ABORT            0x03  /* Unable to abort CCB request */
#define CAM_REQ_CMP_ERR         0x04  /* CCB request completed with an err */
#define CAM_BUSY                0x05  /* CAM subsystem is busy */
#define CAM_REQ_INVALID         0x06  /* CCB request is invalid */
#define CAM_PATH_INVALID        0x07  /* Path ID supplied is invalid */
#define CAM_DEV_NOT_THERE       0x08  /* SCSI Device Not Installed/there */
#define CAM_UA_TERMIO           0x09  /* Unable to terminate I/O CCB req */
#define CAM_SEL_TIMEOUT         0x0A  /* Target Selection Timeout */
#define CAM_CMD_TIMEOUT         0x0B  /* Command Timeout */
```

```
#define CAM_MSG_REJECT_REC    0x0D  /* Message Reject Received */
#define CAM_SCSI_BUS_RESET    0x0E  /* SCSI Bus Reset Sent/Received */
#define CAM_UNCOR_PARITY      0x0F  /* Uncorrectable parity err occurred */
#define CAM_AUTOSENSE_FAIL    0x10  /* Autosense: request sense cmd fail */
#define CAM_NO_HBA            0x11  /* No HBA Detected error */
#define CAM_DATA_RUN_ERR      0x12  /* Data Overrun error */
#define CAM_UNEXP_BUSFREE     0x13  /* Unexpected Bus Free */
#define CAM_SEQUENCE_FAIL     0x14  /* Target Bus Phase Sequence Failure */
#define CAM_CCB_LEN_ERR       0x15  /* CCB length supplied is inadequate */
#define CAM_PROVIDE_FAIL      0x16  /* Unable to provide requ. capability */
#define CAM_BDR_SENT          0x17  /* A SCSI BDR msg was sent to target */
#define CAM_REQ_TERMIO        0x18  /* CCB request terminated by the host */

#define CAM_IDE               0x33  /* Initiator Detected Error */
#define CAM_RESRC_UNAVAIL     0x34  /* Resource Unavailable */
#define CAM_UNACKED_EVENT     0x35  /* Unacknowledged Event by Host */
#define CAM_MESSAGE_RECV      0x36  /* Message Received in Host Target Mode */
#define CAM_INVALID_CDB       0x37  /* Invalid CDB received in Host Target Mode */
#define CAM_LUN_INVALID       0x38  /* LUN supplied is invalid */
#define CAM_TID_INVALID       0x39  /* Target ID supplied is invalid */
#define CAM_FUNC_NOTAVAIL     0x3A  /* The requ. func is not available */
#define CAM_NO_NEXUS          0x3B  /* Nexus is not established */
#define CAM_IID_INVALID       0x3C  /* The initiator ID is invalid */
#define CAM_CDB_RECVD         0x3D  /* The SCSI CDB has been received */
#define CAM_LUN_ALRDY_ENA     0x3E  /* The LUN is already enabled */
#define CAM_SCSI_BUSY         0x3F  /* SCSI Bus Busy */

#define CAM_SIM_QFRZN         0x40  /* The SIM queue is frozen w/this err */
#define CAM_AUTOSNS_VALID     0x80  /* Autosense data valid for target */

#define CAM_STATUS_MASK       0x3F  /* Mask bits for just the status # */

/* ---------------------------------------------------------------------- */

/* Defines for the CAM Flags field in the CCB header. */

#define CAM_DIR_RESV          0x00000000 /* Data direction (00: reserved) */
#define CAM_DIR_IN            0x00000040 /* Data direction (01: DATA IN) */
#define CAM_DIR_OUT           0x00000080 /* Data direction (10: DATA OUT) */
#define CAM_DIR_NONE          0x000000C0 /* Data direction (11: no data) */
#define CAM_DIS_AUTOSENSE     0x00000020 /* Disable autosense feature */
#define CAM_SCATTER_VALID     0x00000010 /* Scatter/gather list is valid */
#define CAM_DIS_CALLBACK      0x00000008 /* Disable callback feature */
#define CAM_CDB_LINKED        0x00000004 /* The CCB contains a linked CDB */
#define CAM_QUEUE_ENABLE      0x00000002 /* SIM queue actions are enabled */
#define CAM_CDB_POINTER       0x00000001 /* The CDB field contains a pointer */

#define CAM_DIS_DISCONNECT    0x00008000 /* Disable disconnect */
#define CAM_INITIATE_SYNC     0x00004000 /* Attempt sync data xfer, and SDTR */
#define CAM_DIS_SYNC          0x00002000 /* Disable sync, go to async */
#define CAM_SIM_QHEAD         0x00001000 /* Place CCB at the head of SIM Q */
#define CAM_SIM_QFREEZE       0x00000800 /* Return the SIM Q to frozen state */
#define CAM_SIM_QFRZDIS       0x00000400 /* Disable the SIM Q frozen state */
```

```
#define CAM_ENG_SYNC            0x00000200 /* Flush resid bytes before cmplt */
#define CAM_SOFT_RST_OP         0x00000100 /* Soft reset alternative queue operation */
#define CAM_ENG_SGLIST          0x00800000 /* The SG list is for the HBA engine */
#define CAM_CDB_PHYS            0x00400000 /* CDB pointer is physical */
#define CAM_DATA_PHYS           0x00200000 /* SG/buffer data ptrs are physical */
#define CAM_SNS_BUF_PHYS        0x00100000 /* Autosense data ptr is physical */
#define CAM_MSG_BUF_PHYS        0x00080000 /* Message buffer ptr is physical */
#define CAM_NXT_CCB_PHYS        0x00040000 /* Next CCB pointer is physical */
#define CAM_CALLBCK_PHYS        0x00020000 /* Callback func ptr is physical */
#define CAM_SG_LIST_PHYS        0x00010000 /* SG list pointers physical */

/* Phase cognizant mode flags */
#define CAM_DATAB_VALID         0x80000000 /* Data buffer valid */
#define CAM_STATUS_VALID        0x40000000 /* Status buffer valid */
#define CAM_MSGB_VALID          0x20000000 /* Message buffer valid */
#define CAM_TGT_PHASE_MODE      0x08000000 /* The SIM runs in phase mode */
#define CAM_TGT_CCB_AVAIL       0x04000000 /* Target CCB available */
#define CAM_DIS_AUTODISC        0x02000000 /* Disable autodisconnect */
#define CAM_DIS_AUTOSRP         0x01000000 /* Disable autosave/restore ptrs */


/* Host Target Mode flags */
#define CAM_SEND_STATUS         0x80000000 /* Send status after data phase (if any) */
#define CAM_DISCONNECT          0x40000000 /* Disconnects are mandatory after cdb recv */
#define CAM_TERM_IO             0x20000000 /* Terminate I/O Message supported */
#define CAM_TGT_PHASE_MODE      0x08000000 /* The SIM runs in phase mode */


/* ---------------------------------------------------------------------- */

/* Defines for the SIM/HBA queue actions.  These value are used in the SCSI I/O CCB, for the queue action
field. [These values should match the defines from some other include file for the SCSI message phases.  We
may not need these definitions here. ] */

#define CAM_SIMPLE_QTAG     0x20    /* Tag for a simple queue */
#define CAM_HEAD_QTAG       0x21    /* Tag for head of queue */
#define CAM_ORDERED_QTAG    0x22    /* Tag for ordered queue */

/* ---------------------------------------------------------------------- */

/* Defines for the timeout field in the SCSI I/O CCB.  At this time a value of 0xF-F indicates a infinite timeout.
A value of 0x0-0 indicates that the SIM's default timeout can take effect. */

#define CAM_TIME_DEFAULT    0x00000000  /* Use SIM default value */
#define CAM_TIME_INFINITY   0xFFFFFFFF  /* Infinite timeout for I/O */

/* ---------------------------------------------------------------------- */

/* Defines for the path inquiry CCB fields. */

#define CAM_VERSION         0x4C /* hex value for the current ver */

#define PI_MDP_ABLE         0x80 /* Supports MDP message */
```

```
#define PI_WIDE_32          0x40  /* Supports 32 bit wide SCSI */
#define PI_WIDE_16          0x20  /* Supports 16 bit wide SCSI */
#define PI_SDTR_ABLE        0x10  /* Supports SDTR message */
#define PI_LINKED_CDB       0x08  /* Supports linked CDBs */
#define PI_TAG_ABLE         0x02  /* Supports tag queue message */
#define PI_SOFT_RST         0x01  /* Supports soft reset */

#define PIT_PROCESSOR       0x80  /* Target mode processor mode */
#define PIT_PHASE           0x40  /* Target mode phase cog. mode */
#define PIT_DISCONNECT      0x20  /* Disconnects supported in target mode */
#define PIT_TERM_IO         0x10  /* Terminate I/O message support in target mode */
#define PIT_GRP_6           0x08  /* Group 6 commands supported */
#define PIT_GRP_7           0x04  /* Group 7 commands supported */

#define PIM_SCANHILO        0x80  /* Bus scans from ID 7 to ID 0 */
#define PIM_NOREMOVE        0x40  /* Removable dev not included in scan */
#define PIM_NOINQUIRY       0x20  /* INQUIRY data not kept by XPT */

/* ---------------------------------------------------------------------- */

/* Defines for asynchronous callback CCB fields. */

#define AC_FOUND_DEVICES    0x80  /* During a rescan new device found */
#define AC_SIM_DEREGISTER   0x40  /* A loaded SIM has de-registered */
#define AC_SIM_REGISTER     0x20  /* A loaded SIM has registered */
#define AC_SENT_BDR         0x10  /* A BDR message was sent to target */
#define AC_SCSI_AEN         0x08  /* An SCSI AEN has been received */
#define AC_UNSOL_RESEL      0x02  /* An unsolicited reselection occurred */
#define AC_BUS_RESET        0x01  /* An SCSI bus RESET occurred */

/* ---------------------------------------------------------------------- */

/* Typedef for a scatter/gather list element. */

typedef struct sg_elem
{
    u_char *cam_sg_address;         /* Scatter/gather address */
    u_long cam_sg_count;            /* Scatter/gather count */
} SG_ELEM;

/* ---------------------------------------------------------------------- */

/* Defines for the HBA engine inquiry CCB fields. */

#define EIT_BUFFER          0x00  /* Engine type: buffer memory */
#define EIT_LOSSLESS        0x01  /* Engine type: lossless compression */
#define EIT_LOSSY           0x02  /* Engine type: lossy compression */
#define EIT_ENCRYPT         0x03  /* Engine type: encryption */

#define EAD_VUNIQUE         0x00  /* Eng algorithm ID: vendor unique */
#define EAD_LZ1V1           0x01  /* Eng algorithm ID: LZ1 var. 1*/
#define EAD_LZ2V1           0x02  /* Eng algorithm ID: LZ2 var. 1*/
#define EAD_LZ2V2           0x03  /* Eng algorithm ID: LZ2 var. 2*/
```

```
/* ---------------------------------------------------------------------- */
/* ---------------------------------------------------------------------- */

/* UNIVOS OSD defines and data structures. */

#define INQLEN  36                  /* Inquiry string length to store. */

#define CAM_SUCCESS         0       /* For signaling general success */
#define CAM_FAILURE         1       /* For signaling general failure */

#define CAM_FALSE           0       /* General purpose flag value */
#define CAM_TRUE            1       /* General purpose flag value */

#define XPT_CCB_INVALID     -1      /* for signaling a bad CCB to free */

/* General union for kernel space allocation.  Contains all the possible CCB structures.  This union should never
be used for manipulating CCB's its only use is for the allocation and deallocation of raw CCB space. */

typedef union ccb_size_union
{
    CCB_SCSIIO    csio;     /* Please keep this first, for debug/print */
    CCB_GETDEV    cgd;
    CCB_PATHINQ   cpi;
    CCB_RELSIM    crs;
    CCB_SETASYNC  csa;
    CCB_SETDEV    csd;
    CCB_ABORT     cab;
    CCB_RESETBUS  crb;
    CCB_RESETDEV  crd;
    CCB_TERMIO    ctio;
    CCB_EN_LUN    cel;
    CCB_IMMED_NOTIFY cin;
    CCB_NOTIFY_ACK cna;
    CCB_ENG_INQ   cei;
    CCB_ENG_EXEC  cee;
} CCB_SIZE_UNION;

/* The typedef for the async callback information.  This structure is used to store the supplied info from the set
async callback CCB, in the EDT table in a linked list structure. */

typedef struct async_info
{
    struct async_info *cam_async_next;      /* pointer to the next structure */
    u_long cam_event_enable;                /* Event enables for callback resp */
    void (*cam_async_func)();               /* Async callback function address */
    u_long cam_async_blen;                  /* Length of "information" buffer */
    u_char *cam_async_ptr;                  /* Address for the "information */
} ASYNC_INFO;
```

```
/* The CAM EDT table contains the device information for all the devices, SCSI ID and LUN, for all the SCSI
busses in the system.  The table contains a CAM_EDT_ENTRY structure for each device on the bus.*/

typedef struct cam_edt_entry
{
    long cam_tlun_found;            /* Flag for the existence of the target/LUN */
    ASYNC_INFO *cam_ainfo;          /* Async callback list info for this B/T/L */
    u_long cam_owner_tag;           /* Tag for peripheral driver's ownership */
    char cam_inq_data[ INQLEN ];    /* storage for the inquiry data */
} CAM_EDT_ENTRY;
/* ---------------------------------------------------------------------- */
```