Technical Committee

Specification of the Device Control Protocol (DCP)

Version 1.0

AF-PHY-0138.000

February, 2000

The ATM Forum

Worldwide Headquarters

2570 West El Camino Real, Suite 304

Mountain View, CA 94040-1313

Tel: +1-650-949-6700

Fax: +1-650-949-6705

TABLE OF CONTENTS

1 INTRODUCTION 5

1.1 DCP and its uses 5
1.2 DCP operating in a network element 7
1.3 DCP specification and beyond 10
1.4 About this specification 11

2 DCP PROTOCOL OVERVIEW 11

2.1 DCP Capabilities 11
2.2 DCP Minimum set 12
2.3 DCP Basic set 12
2.4 DCP Options 13

3 DEFINING THE VIRTUAL REGISTERS FOR DCP 14

3.1 How to define the virtual register map for a device 14
3.2 Virtual Register Bit Layout 16
3.3 Common Virtual Registers 16

4 DEFINING DEVICE BEHAVIOUR 17

5 OPERATION OF DCP 20

5.1 Minimum Set Procedures: 20
5.1.1 Discover Procedures 20
5.1.2 Reset Procedures 22
5.2 DCP Basic Set 23
5.2.1 Virtual Register Procedures 23
5.2.2 Event Notification 24
5.2.3 Connect Procedures 25
5.2.4 Device-specific Virtual Register Operations 26

6 DCP MESSAGES 27

6.1 General Message Format 27
6.2 Message coding 27
6.2.1 ATM Header 27
6.2.2 Destination bit 28
6.2.3 ACK request bit 29
6.2.4 Acknowledgement bit 29
6.2.5 Message Type 29
6.2.6 Device ID 31
6.2.7 Data 33
6.2.8 Transaction Correlation ID 33
6.2.9 Message Trailer 33

7 MESSAGE FORMATS 34

7.1 Initiate Discover Message 34
7.2 Discover Message and Discover ACK message 34
7.2.1 Device type and Class 35
7.2.2 Model Number 36
7.2.3 Number of Ports 36
7.3 RESET Message 36
7.4 CONNECT and CONNECT ACK Messages 37

7.5 Virtual Register Read Message 38
7.6 Virtual Register Write Message 39
7.7 Event Notification 40
7.8 Device-specific Virtual Register Operations Messages 42

8 DCP NOTES AND MACROS 42

8.1 Downloading software loads 42
8.2 Acknowledgements 43
8.3 Outstanding transactions 43
8.4 Multi-referenced registers 43
8.5 Soft Resets 44

APPENDIX A. USING DCP FOR TOPOLOGY-BASED SOURCE ROUTING 44

Editor Contact:

Diana Wilhelm
P.O. Box 3511 Station C, Ottawa, Ontario Canada, K1Y 4H7.
Phone: 1 613 763 3539; Fax: +1 613 763 3988;
Email: wilhelm@nortelnetworks.com

1 Introduction

1.1 DCP and its uses

The Device Control Protocol (DCP) allows a simple device to be controlled within an ATM Network Element (e.g. an ATM switch) with a minimum of needed functionality in the device itself.

DCP is targeted at controlling devices within an ATM Network Element. For example, DCP can be used between a master processor card and ATM port cards on a shelf of equipment. The protocol is used for communication between a device and a Device Controller within an ATM Network Element.

- **The Device Controller is the master of the device and is able to detect and control the device.**
- **A device is any component of the ATM Network Element.**

The DCP protocol is able to support all of the requirements for control and management functions. It will support the real-time responsiveness needed for controlling the device through the single cell mechanism, and it will support information retrieval and configuration of the device needed for managing the device through the virtual register operations.

The rationale for creating yet another protocol is simple. Management and Control must meet at the device. Some devices need only a very bare minimum of functionality to operate, since they only need simple connectivity. DCP is intended to address the management and control of such devices.

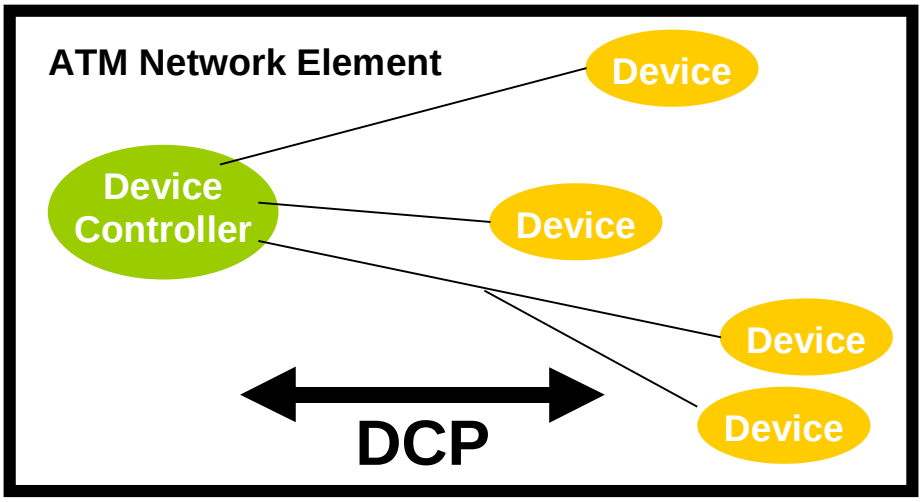


Figure 1 – Device Controller relationship to Devices

The protocol uses the concept of the virtual register. A virtual register is an addressable location on the device that can be directly mapped to such things as: memory, control registers, command registers, or counters.

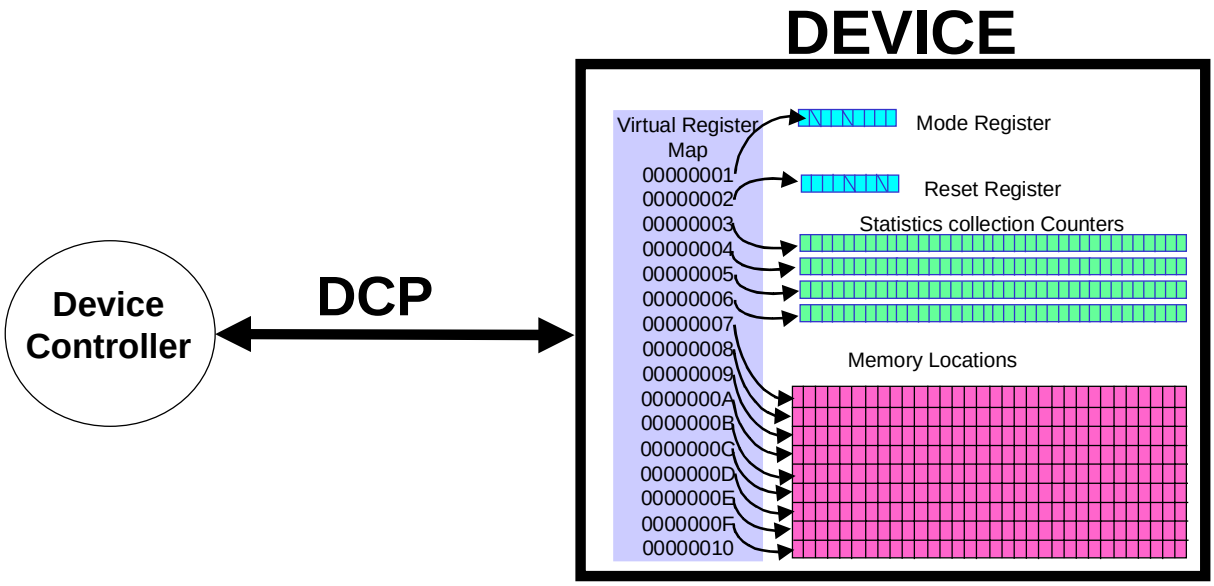


Figure 2 – Example of a Virtual Register map

The use of virtual registers creates a very powerful mechanism for DCP to adapt to a wide variety of applications.

DCP supports the following capabilities:

- **start-up (boot-up) of devices including device identification**
- **configuration of the devices**

- **hardware control e.g. reset**
- **information exchange**
- **simple event notification i.e. device status changes**
- **keep-alive/heart-beat.**

As well, DCP

- **can operate over either point-to-point or shared media connections**
- **supports the verification of the message payload.**

DCP also contains a set of optional functions that can further support device requirements. These functions include:

- **requesting another channel for communication**
  - **This is useful for either running DCP outside of the well-known virtual channel, or for running other protocols to the device (such as signalling or management protocols) or for re-homing the device on a different Device Controller for processor load-sharing.**
- **topology-based source routing**
  - **This option provides the means to address messages according to topology (using source routing techniques). It requires the switches to support the topology-based source routing mechanisms.**

1.2 DCP operating in a network element

When a DCP device appears within a network element (e.g. after a reset), it initiates the discovery process by sending Discover messages out each of its ports, on the well-known DCP VPI/VCI.

If there are switch fabrics between the new device and the Device Controller, then the DCP Discover message will be either source routed or switched to particular VCs back to the Device Controller, depending on the network element configuration.

In the case where topology-based source routing is used, each switch

fabric is aware of its own upward-facing port that reaches the Device Controller. The incoming DCP message is modified to add source routing information, and passed out this port.

In the case where topology-based source routing is not used, each port on a switch fabric has a virtual connection back to the Device Controller. This connection is set up as part of the discovery process for the switch fabric itself. When a DCP message is received, the cell is simply switched on its way to the Device Controller. In this manner, each device's messages will arrive at the Device Controller on a separate channel. See Figure 3 - DCP Connections. In the case where shared media connections appear in the path between the device and the Device Controller, the messages from individual devices are distinguished by the Device ID contained in all messages. Note that shared media connections can not be supported when source routing is used.

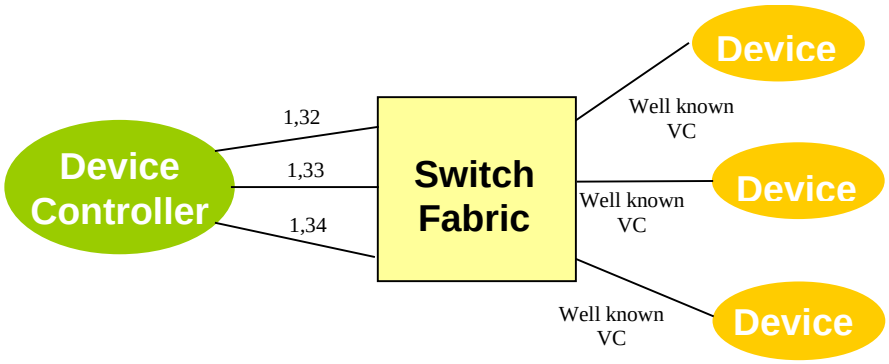


Figure 3 - DCP Connections

The Discover message contains information identifying the specific device type and version, which the Device Controller uses to identify the DCP message set and virtual register map used by the device. DCP can then be used to configure, monitor, and control the device. Figure 4 depicts a typical scenario of DCP message flows between a device and a Device Controller.

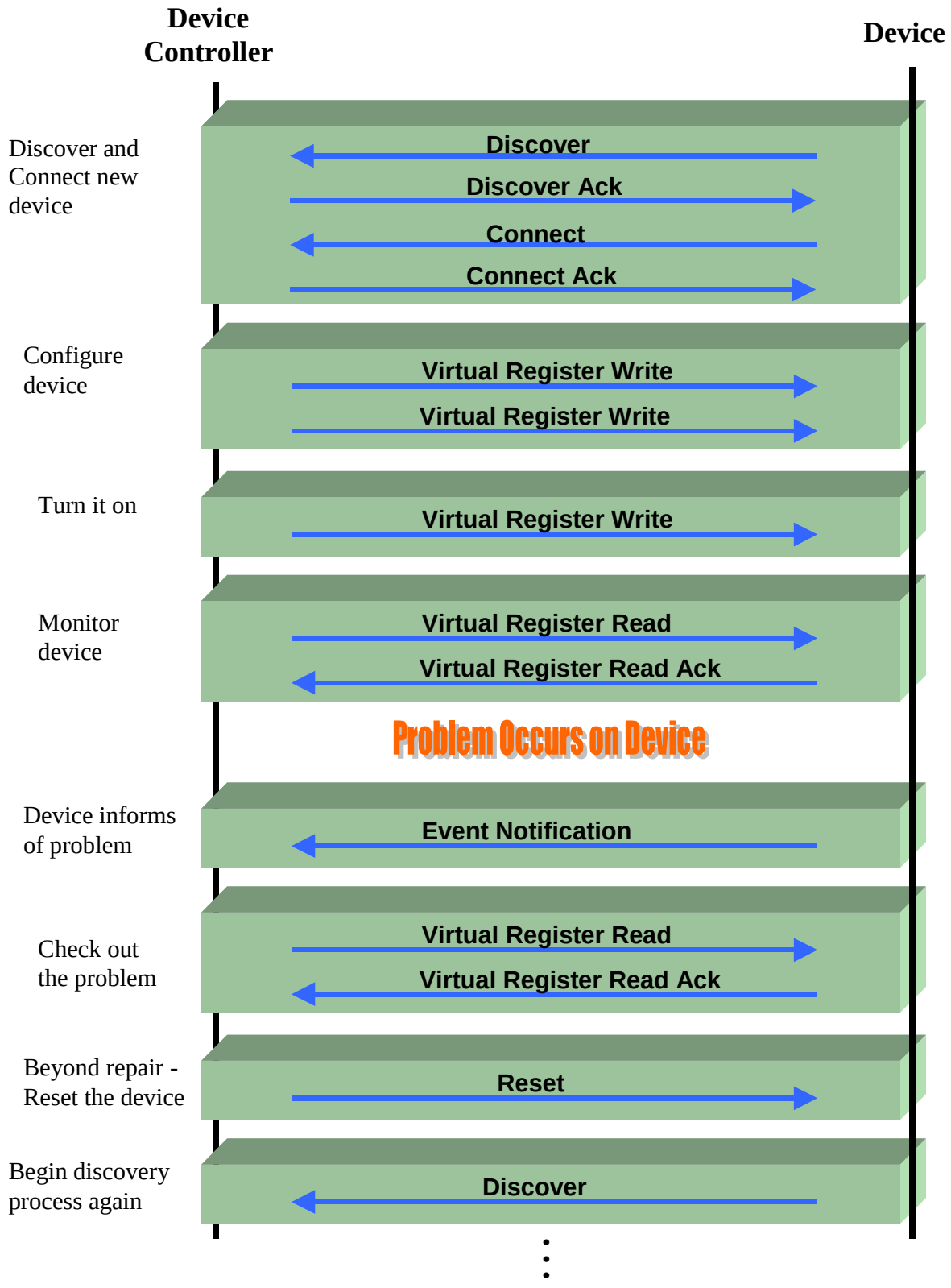


Figure 4- Typical use of DCP

In this typical scenario, the device appears to the network element and uses the Discover message to make itself known to the Device Controller, and then uses the Connect message to establish a dedicated virtual connection with the Device Controller.

Further DCP communication with this device will occur over the new dedicated VC. Based on the device information from within the Discover message, the Device Controller configures the new device by writing information into virtual registers on the device.

At the device, each of these virtual registers may correspond to a register, memory location, or a local abstracted virtual register. After configuration is complete, the device is instructed to begin normal execution.

The Device Controller periodically monitors the device, by reading its set of virtual registers. If necessary, the Device Controller could also write virtual registers to provide updated information and instructions to the device.

At some point later, the device experiences some sort of fault, and informs the Device Controller by sending an Event Notification message. The Device Controller examines the virtual registers of the device to determine the cause and severity of the fault. In some instances, the device may be instructed to run additional diagnostics, or take specific components out of service.

In this example scenario, the Device Controller decides that the state of the device is beyond repair, and elects to reset the device by sending a Reset message. Upon resetting, the device issues a Discover message, to begin the discovery process again.

1.3 DCP specification and beyond

The DCP specification contained in this document specifies the messages, coding and procedures of the protocol. These represent the tools provided by the protocol.

Beyond this, for each device type, it is necessary to define the virtual register map of the device and the behavior of the device (e.g. control registers, reset registers, messages supported). By allowing different

devices to define their own virtual register map, DCP is very flexible. It can support a wide variety of different device types while still maintaining a common set of functions.

DCP can also support more complex functions through the definition of specific command registers in the virtual register map e.g. a command for the device to calculate a checksum over a range of memory. In this way, there are means to use DCP to address device-specific requirements without changing the DCP protocol. These are called DCP-macros in this document.

1.4 About this specification

This specification includes:

- the DCP capabilities and option sets available
- the rules and recommended practices for defining the virtual register map
- a template for defining device behavior
- the DCP message set and coding
- the DCP procedures

In addition, Section 8 DCP Notes and Macros includes some of the options for DCP-macros that can be defined and the Appendices include a variety of example applications of the protocol.

2 DCP Protocol Overview

2.1 DCP Capabilities

The DCP protocol consists of three categories of functions:

Minimum set:

- the Minimum set of functions which all devices must support

Basic set:

- a Basic set of functions which includes the Minimum set above. This set is the most commonly used set of functions. A device usually includes all of the messages in the Basic set.

Optional sets:

- **a set of optional functions which enhance DCP's functionality. The set of optional functions include:**
  - **device-specific and vendor-specific messages**
  - **assignment of another channel over which the device can communicate. This new channel could run either DCP or some other protocol as required by the device type.**
  - **topology-based source routing**

2.2 DCP Minimum set

The DCP Minimum set includes the following functions:

- **operation over a well-known channel (ATM VPI/VCI) to begin communication**
- **reset of the device (hard)**
- **support for the Initiate Discover request from the Device Controller**
- **support for the Discover request by the device**
  - **including device type, device class, vendor model, version, and number of ports.**
  - **support for a device id (note: device id may be null for devices with no unique ID)**
- **devices can ignore all DCP messages that are not part of the Minimum set.**

The DCP messages supported for the Minimum set include:

Initiate Discover
Discover
Discover ACK
Reset

2.3 DCP Basic set

The DCP Basic set includes the DCP Minimum set plus the following functions:

- **virtual register read, write capabilities**
- **event notification**

The Basic set provides the means to operate on virtual registers. Virtual registers can index a wide variety of functions and features on a device including: control registers, memory locations, counters, and command registers as well as registers to be used for DCP macros (e.g. perform checksum operation). In addition, event notification provides the means for a device to indicate to its Device Controller that there is something needing attention in the device.

The additional DCP messages supported for the Basic Set include:

Virtual Register Read
Virtual Register Read ACK
Virtual Register Write
Virtual Register Write ACK
Event Notification
Event Notification ACK (optional)

2.4 DCP Options

DCP further supports additional optional functions. These are covered in this specification. These functions include:

1. **Device-specific read, write**
   - **These messages allow for device-specific register read/write. A device can use any format for the data payload of the message.**
2. **Device-specific messages**
   - **These messages allow for device-specific operations. A set of message types have been reserved for this purpose.**
3. **Connection assignment**
   - **This allows the device to assign another channel for communication. This additional channel need not carry DCP nor does it have to terminate at the Device Controller, but could if needed. As well, multiple channels can be requested through this method.**

4. Topology-based source routing

- This allows the Device Controller to address messages according to the topology (using topology-based source routing techniques).

3 Defining the Virtual registers for DCP

3.1 How to define the virtual register map for a device

DCP requires that a Virtual Register Map be defined for each device type. To do this requires some careful planning and specification to ensure interoperability between the device and the Device Controller.

Virtual Registers are identified by a Virtual Register Index number. This number is 4 bytes long and thus allows for up to 4,000 Meg of Virtual registers on the device, each of which can be up to 32 bits long.

A Virtual Register Map consists of all the virtual registers, their indices and the definition of their use and codepoints for a device.

A virtual register is defined as being 32-bits long. If the device only needs to use a subset of the bits, then the least significant bits of the 32-bits are used. It is also possible to combine several functions into a single virtual register as needed. This would allow different bits in the register to be designated for different purposes. However, there are no masking utilities provided in the standardized DCP message set. Operations to mask bits would be handled by the Device Controller.

For devices, which require 64-bit or longer register operations, the virtual register map will use big endian format (MSW and then LSW). It is recommended that these be numbered sequentially with the MSW being an even value Virtual Register index number.

To lay out the VR Map, identify all of the control registers, mode registers, hardware specific registers, memory locations, status registers, etc. that need to be accessed by the Device Controller.

Then, identify the actions and commands that need to be performed on the device. These could include such things as software download, checksum calculations, diagnostic tests to run, looparound activation,

etc.

It is important to consider how the device operation will function because it is very useful if groups of registers relating to a single function can be accessed in a single cell. This would require that their Virtual Register indices are contiguous.

To facilitate the ease of grouping Virtual Registers, it is possible for a device register to be identified by more than one Virtual Register Index. This allows some flexibility to have a single device location included in different contiguous virtual register groups.

Note that Virtual Register index value = 00 00 00 00 is reserved.

Virtual Registers Application Notes:

- **Identify the complete range of Virtual Register indices to be used.**
- **Layout the virtual register map in a table format.**
- **For each register, decide on which Virtual Register Operations can be performed on the Virtual Register**
  - **read and/or write**
  - **if device-specific messages are used to operate on the Virtual register, then this should be identified.**
- **Do not use VR = 00 00 00 00 - reserved to indicate problems**
- **Make a list of device specific registers such as:**
  - **reset register**
    - **define how the value written to this register selects different reset types**
  - **connect procedures activation register**
    - **this allows for the Device Controller to turn on/off the Connect Procedures**
  - **alternative VPI/VCI used for s/w download**
  - **on/off capabilities**
    - **is it possible to turn subcomponents of the device on and off e.g. switch ports**
  - **general operating status register**
  - **mode registers needed to configure the device before the MIB for the device is configured**

- command registers needed for specific actions such as:
  - check sum calculation including the range of addresses to perform the checksum on
  - command to set up a connection in the hardware e.g. across the device fabric
  - run diagnostic (s)
  - run test
  - clear memory including the range of addresses to be cleared
- Define the MIB (Management information Block) information for the device
  - This is used by management functions to access information about the device and to configure the device including:
    - status registers
    - counters for measurements e.g. operational measurements such as traffic statistics
  - define the configuration data for the MIB which could include
    - VPI/VCI configuration data of any switch fabric
    - enable hardware, specific features
- Memory locations on the device that need to be addressed to be used for such things as:
  - software load location
  - flash ROM location

3.2 Virtual Register Bit Layout

Virtual Registers contain 32 bits. The bits are numbered as follows:

Table 1 – Virtual Register Bit Layout

bit #	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

The bytes and words for DCP are always arranged in big endian.

3.3 Common Virtual Registers

The only common virtual register that should be supported is:

Table 2 – Common Virtual Register

Virtual Register Index	Name	Coding	
00 00 00 01	Basic Device Status	bit 0	0 – indicates device is off 1 – indicates device is on
		bit 1	0 – indicates device is operating correctly 1 – indicates device is not operating correctly

Note that Virtual Register 00 00 00 00 is reserved.

4 Defining Device Behavior

Device behavior needs to be defined in order to ensure interoperability between the Device Controller and the device. This is done for each message type.

The following outlines the aspects that need to be defined for each message type for a device.

Table 3 – Device Behavior Template

Message Type	Behavior	Device Support (Y/N)
Initiate Discover	Supported?	Y
	Receipt of Initiate Discover causes a Discover message to be sent?	Y
Discover and Discover ACK	Supported?	Y
	Reply to Initiate Discover?	Y
	Sent on hard reset?	Y
	On power-up, random offset timer used to avoid congestion at Device Controller? (define maximum timer value)	(Y/N) DEFINE

	Timer for ACK used?	**(Y/N)**
	After timeout for ACK, device sends another Discover?	**(Y/N)**
	Device discards all other messages (except Initiate Discover and Reset) until Discover ACK is received?	**(Y/N)**
	Define the contents of the Discover message information field	**DEFINE**
	Define the values of the Discover message code points	**DEFINE**
Reset	**Supported?**	**Y**
	Reset causes complete return to initial state?	**(Y/N)**
	Clears all volatile memory upon reset?	**(Y/N)**
	Device sends Discover message upon reset completion?	**Y**
Virtual Register operations (Read, Write)	**Supported?**	**(Y/N)**
	Virtual Register Read supported?	**(Y/N)**
	Virtual Register Write supported?	**(Y/N)**

	Unrecognized register behavior defined?
	- **options include:**
	  - **send back VR = 00 00 00 00**
	  - **event notification sent back**
	  - **change status register**
	  - **do nothing**
		DEFINE

Event Notification	**Supported?**	**(Y/N)**
	Define the list of events and their	**DEFIN**

	coding for DCP protocol problems	E
	Define the list of events and their coding for device problems.	DEFINE
	Is the Event Notification ACK used?	(Y/N)
	Is a timer used for the ACK?	(Y/N)
	Is there a maximum outstanding event counter?	(Y/N)
	Define the contents of the Event Notification information field	DEFINE
Connect Message	Supported?	(Y/N)
	Define when the Connect message would be sent: - After the initial Discover message sequence? - Activated by the Device Controller writing to a virtual register which causes the Connect to be sent?	DEFINE
	Is DCP supported on the new connection?	(Y/N)
	If not, what is the new connection used for?	DEFINE
	Are multiple additional connections supported?	(Y/N)
	If multiple connections are supported, define the Connection Numbers and the uses of each of these connections?	DEFINE
	Will the well-known VPI/VCI continue to be used? For what messages?	DEFINE
	Define the contents of the	DEFIN

	Connection Binding information field.	E
Other Messages	Supported?	(Y/N)
	What messages are supported?	DEFINE
	Define their format and their coding.	DEFINE
	How are message errors handled?	DEFINE
	Are Acknowledgements needed for these messages?	(Y/N)
	What is the Message flow for these messages?	DEFINE
	Will a timer be used for the message?	(Y/N)

5 Operation of DCP

5.1 Minimum Set Procedures:

There are only two procedures that a device must support: Discover and Reset.

5.1.1 Discover Procedures

The Discover Procedures are supported through three messages: the Initiate Discover message, the Discover message and the Discover ACK message.

The Initiate Discover message is sent from the Device Controller to the device. It can be used to both discover a device and to ensure that the device is still attached (i.e. as a heartbeat message).

The Discover Message is always sent from the Device when it attaches to the network element for the first time or after a Reset message. It is also always sent in response to an Initiate Discover message from the Device Controller.

Key points to consider with the Discover Message:

- **The Discover Message contains information about the device**

that the Device Controller uses to uniquely identify the device. Using this information, the Device Controller will know which other DCP messages and procedures are supported by the device i.e. what device driver and virtual register map layout to use for this device.

- **A random offset timer may be used at the Device to delay sending the Discover message so as to avoid many devices simultaneously sending Discover message to the Device Controller upon reset. This is implementation dependent.**
- **The Initiate Discover message is also used by the Device Controller to maintain constant communication with the Device as a form of heartbeat. The device is expected to respond to this message with a Discover message that implies that communication is working. A Discover ACK message will then be returned by the Device Controller.**
- **Receipt of the Initiate Discover message does not cause any reset functions to occur at the device. It simply prompts the Discover procedures from the device.**
- **Devices which have more than one outgoing port (e.g. switch) may send Discover messages out of every port on the device. This is expected to happen for the first attach, at reset, and under recovery procedures.**
- **Devices may optionally support topology-based source routing. The Discover procedures support this. See APPENDIX A Using DCP for Topology-based Source Routing.**

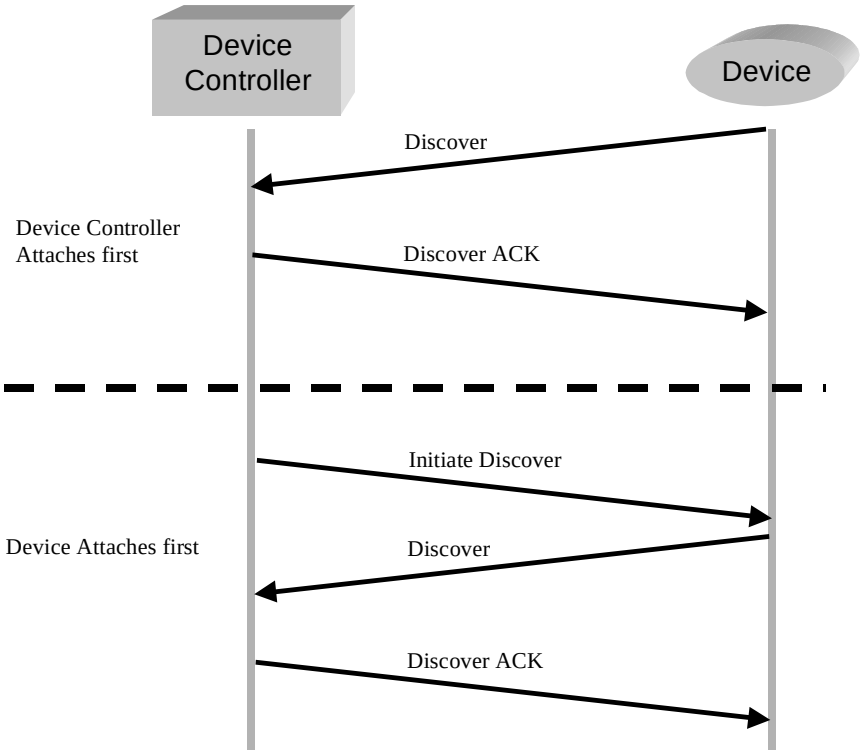


Figure 5 – Discover Operation Message Flows

5.1.2 Reset Procedures

The Reset message is always sent from the Device Controller to the Device.

The Reset message indicates to the device that a complete device reset (hard) is required and that the device should reinitialize itself to its starting state.

Other reset types (i.e. soft) can be supported through the BASIC DCP set by using virtual register writes to specific registers in the device defined for that purpose.

No acknowledgement is sent in reply to a RESET message. Under normal operation, the device will start the Discover process or the Device Controller may initiate it.

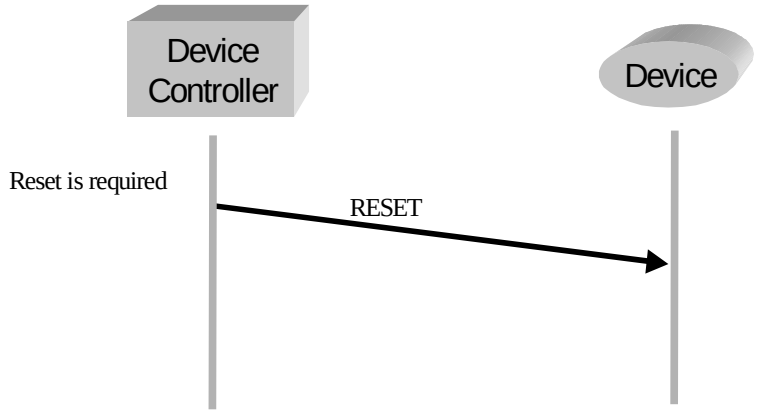


Figure 6 Reset Operation Message Flows

5.2 DCP Basic Set

The DCP Basic set provides most devices with all the functions they may require. The Basic set includes the Virtual Register read-write functions (read, write). In addition, the DCP Basic set includes the Event Notification messages to allow the device to notify the Device Controller of events that need further attention (e.g. error conditions).

5.2.1 Virtual Register Procedures

Each device must have a map of its Virtual Registers to use the Virtual Register procedures. Virtual registers are a means of numbering the various device-specific memory, register, counters and control/mode registers into a single common map. In this way, the Device Controller can access each memory location or hardware register on a device.

Virtual Register operations allow for multiple sequential virtual registers to be read through a single Virtual Register message. The maximum number of 32-bit registers is limited only by the single cell length.

Virtual Register operations include:

- **read virtual registers – used to read the value of one or more virtual register**
- **write to virtual registers – used to write value(s) contained in the message to the virtual register(s)**

Notes on Virtual Register procedures:

- **All operations are performed using 32-bit registers. If a virtual register identifies information that is shorter than this e.g. 8-bit or 16-bit, then the least significant bits of the 32-bit virtual register value are used.**
- **For devices, which support 64-bit or longer registers, the device should perform its operation (read or write) on the complete register. This must be supported in the device implementation.**
- **There may be a need to assign and map a single device register to more than one Virtual Register indices. This should be very carefully defined however to ensure it does not cause any failure/error conditions.**

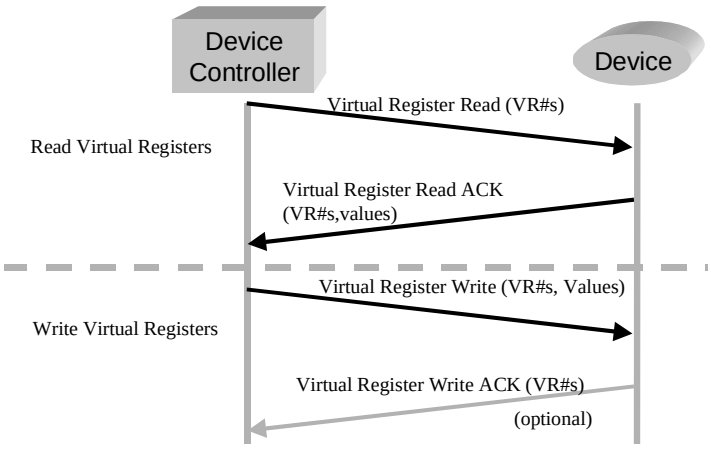


Figure 7 Virtual Register Operations Message Flows

5.2.2 Event Notification

The Event Notification Message is passed from a device to the Device Controller when some condition exists which requires Device Controller intervention.

The Event Notification can be used with or without an Acknowledgement, depending on the purpose. The DCP message header will identify if an acknowledgement is required.

The Event Notification contains information relating to the event to distinguish the event type. This information is device specific and must be specified. A common set of events for all devices is defined in this specification.

Notes on Event Notification

- **This message can be replaced by using some form of polling of status registers from the Device Controller if there is concern about focused loads on the Device Controller on failure conditions.**
- **There may be a need to have a maximum outstanding count of uncleared event notifications in each device to avoid excessive messaging under compound error situations. This counter could be reset at regular intervals as required.**
- **The content of the data field contained in the Event Notification is copied into the Event Notification ACK. This is used by the device to correlate the ACK message (if it is used.)**

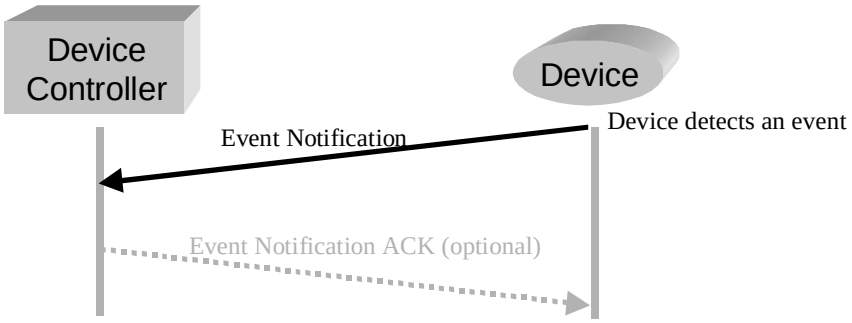


Figure 8 Event Notification Operation Message Flows

5.2.3 Connect Procedures

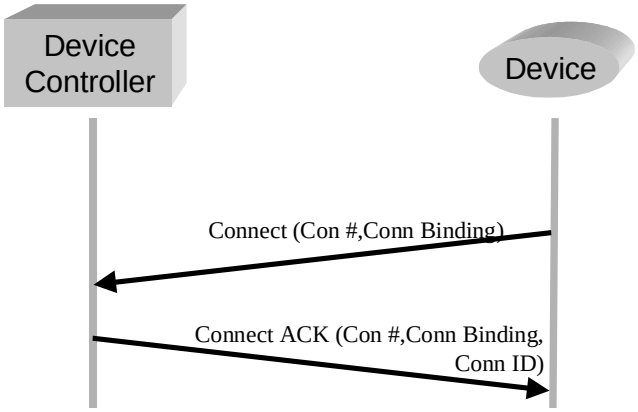


Figure 9 Connect Procedures

The Connect procedures provide the means for a device to request another channel. These procedures are optional for all devices.

A device may request more than one channel. Each request is sent in a separate message.

The Connect ACK contains the Connection ID of the assigned channel.

The Connect message contains the Connection Binding field which is used to further identify the requested endpoint for the device request. The definition of this field is implementation dependent. This field is also returned to the device in the Connect ACK.

The new channel may be used to carry DCP or it may be used for some other purpose. A value for the Connection binding field is defined to indicate that DCP is to be used over the newly assigned connection.

Notes on the Connect Message:

- **It is possible for the Device Controller to return the well-known DCP VPI/VCI in the Connect ACK message. The device itself shouldn't care. It depends on the topology of the configuration.**

5.2.4 Device-specific Virtual Register Operations

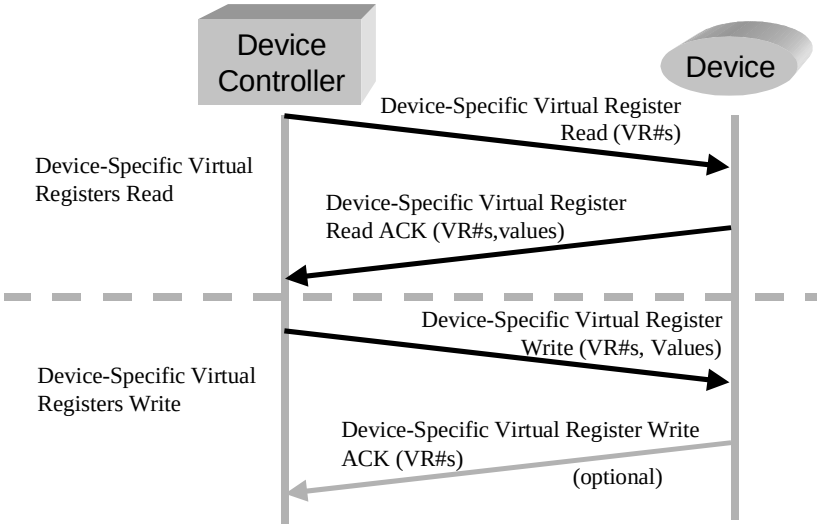


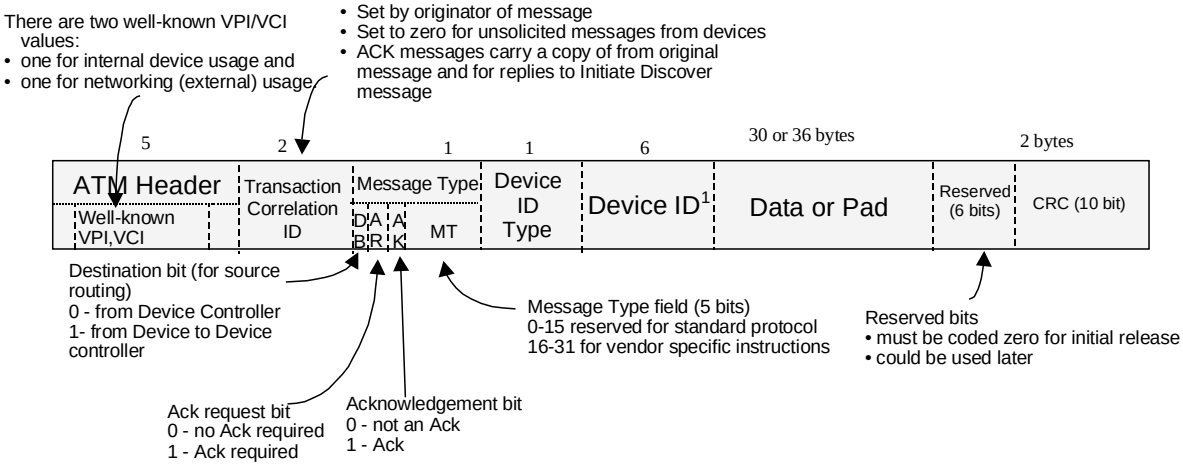
Figure 10 Device-specific Virtual Register Operations

Device-Specific Virtual Register operations include: read, write. The message sequences for these messages are identical to the standard Virtual Register operations of read and write.

These messages use device-specific message formats.

6 DCP Messages

6.1 General Message Format



[1]Device ID is always contained in all of the standardized messages (Message type values from 00 to 08). For device-specific messages, Device ID is optional and its inclusion will depend on the application. Device ID Type field will be coded to indicate when Device ID is not included.

Figure 11 – General Cell Format.

All standardized DCP messages use the same message format as shown in Figure 11 – General Cell Format.

Device-specific messages use the same message format but may exclude the Device ID field.

6.2 Message coding

6.2.1 ATM Header

Devices can use the well-known VPI/VCI specifically reserved for DCP. Alternatively, the implementers may select some specific channel to use.

There is one well-known VPI/VCI reserved for this purpose.

Table 4 DCP Well-known VPI/VCI value

	VPI	VCI

DCP VPI/VCI	**0**	**19**[1]

The ATM header does not require any special handling.

The PT bit is always set to indicate last cell in frame.

The device will always use its well-known VPI/VCI. However, there could be many DCP protocol channels connected to the Device Controller, each with a different VPI/VCI value. This is because there can be switches between the device and the Device Controller. In that case, the well-known VPI/VCI value would get translated before it reaches the Device Controller.

Since DCP can operate over shared media, there may also be more than one device attached to a single channel.

DCP is currently for use internal to a network element.

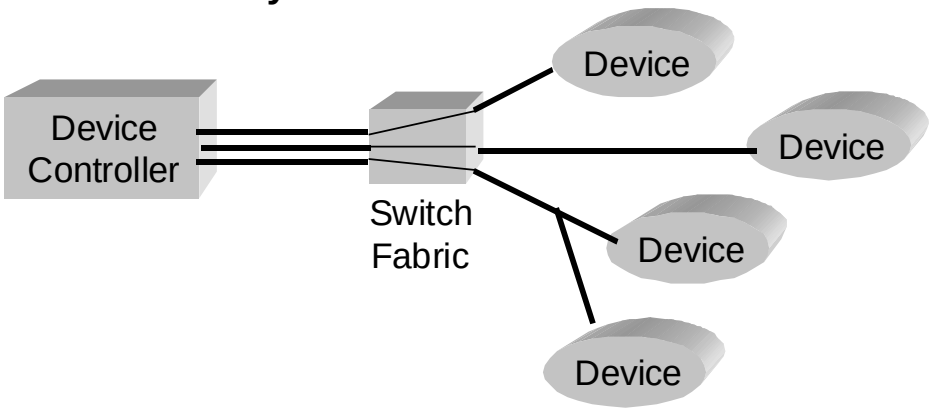


Figure 12 – DCP protocol channels from Devices to Device Controller

6.2.2 Destination bit

The destination bit is only needed by the topology-based Source routing option. However, for consistency, the bit should be coded as follows:

For messages from the Device Controller going to the device, this bit is set to "1".

For messages from the device going to the Device Controller, this bit

[1] **Subject to ratification by the ATM Forum.**

is set to “0”.

See APPENDIX A Using DCP for Topology-based Source Routing for more details on topology-based Source Routing.

6.2.3 ACK request bit

This bit is used to request an acknowledgement from the receiver of the message. If an Acknowledgement message is required, then this bit is set to “1”.

See Tables 5, 6, 7, and 8 for details on which message types support the ACK request bit.

6.2.4 Acknowledgement bit

This bit is used to indicate that this DCP message is an acknowledgement message.

An acknowledgement message should contain:

- **The same message type value as the requesting message that prompted this acknowledgement (except in the case of the Discover message where it is acknowledging the Initiate Discover message.)**
- **The correct contents in the Data field as required for the message.**
- **The identical value for the Transaction ID as was contained in the requesting message. This allows the receiver to correlate the Acknowledgement with the original request.**

See Tables 5, 6, 7, and 8 for details on which message types support the ACK request bit.

6.2.5 Message Type

The message type field is a 5 bit long field that is used to indicate the operation of the message. The message types are very closely linked to the ACK request and the ACK bit. The following tables defines how they are to be coded:

Table 5 Minimum Set Message type codings

Destinati	Ack	Ack bit	Messa	Message Type

on Bit	Request bit		ge Type Value	
X	0	0	00	not used (indicates invalid message)
1	1	0	01	Initiate Discover
0	1	0 – device initiated 1 – response to Initiate Discover message	02	Discover
1	0	1	02	Discover ACK
1	0	0	03	Reset

Table 6 Basic Set Message type codings

Destination Bit	Ack Request bit	Ack bit	Message Type Value	Message Type
1	1	0	05	Virtual Register Read
0	0	1	05	Virtual Register Read ACK
1	0 or 1	0	06	Virtual Register Write
0	0	1	06	Virtual Register Write ACK
0	0 or 1[2]	0	08	Event Notification
1	0	1	08	Event Notification ACK

[2] The use of acknowledgments for the Event Notification may cause focused message loading on the Device Controller and should be carefully designed. Polling the device can provide equivalent functionality to the event notification if required.

Table 7 Optional Message type codings

Destination Bit	Ack Request bit	Ack bit	Message Type Value	Message Type
0	1	0	04	Connect
1	0	1	04	Connect ACK
1	1	0	09	Device-specific Virtual Register Read
0	0	1	09	Device-specific Virtual Register Read ACK
1	1	0	0A	Device-specific Virtual Register Write
0	0	1	0A	Device-specific Virtual Register Write ACK
			0C-0F	Reserved for DCP
			10-1F	For device –specific messaging

Table 8 Reserved codings - Message type codings

Destination Bit	Ack Request bit	Ack bit	Message Type Value	Message Type
			0C-0F	Reserved for DCP
			10-1F	For device –specific messaging

6.2.6 Device ID

The Device ID field is 7 bytes long. It is formatted as follows:

Table 9 Device ID Field

Device ID type	Device ID Value
1 byte	6 bytes

The Device ID is used to identify a device.

The Device ID type is coded as follows:

Table 10 Device ID type coding

Code	Definition
00	not used. This indicates that the message is likely invalid (accidentally coded all zeros)
01	null id (all devices should recognize and process this message)
02	user-defined device numbering. It has local significance only e.g. X10 standard
03	Topology-based source routing Device ID is being used (see further in document for details)
04	IEEE MAC Id
05	no Device ID field included in message (used for device-specific messages only)
06	ITU G.983[3],[4] - ATM PON specification
07-127	Reserved for DCP
128-255	Reserved for vendor specific applications (non-standard)

Each of the Device ID coding types has its own set of rules.

The null id value (coded "01") can be used by the Initiate Discover message if no Device ID is known. All devices should accept and process messages with this value.

The user-defined device numbering (coded "02") can be used in any way. The use of this numbering plan is at the discretion of the implementation.

Topology-based source routing device ID (coded "03) is used as a binary map of the path to the destination of the message. Topology-based source routing device IDs can be said to "identify" a device by the path to get to that device. It is important to note that topology-based source routing device IDs are modified as they transit

[3] Note: For ITU G.983 application, the complete message type field range can be used (except for the minimum set message type values). The message type values from 0X04 to 0X1F are used.

[4] Note: For ITU G.983, no Device ID field is carried and the bytes are reused for other purposes.

intermediary switch devices. This is required to accomplish the routing of the message. For more details on topology-based source routing see APPENDIX A Using DCP for Topology-based Source Routing.

The Device ID coded as "04" means that the IEEE Media Access Control (MAC) ID is contained in the field. This is a 6-byte field that identifies devices uniquely.

6.2.7 Data

The contents of the data field is a function of the message itself. See the individual messages for the coding of this field. In all cases, PAD field are coded with all "0" bits.

6.2.8 Transaction Correlation ID

The Transaction Correlation ID field is a 2 byte field which is used to identify the response to a DCP transaction that is ongoing. The Transaction Correlation ID is thus used to correlate messages requiring some action and their corresponding reply. In this way, more than one outstanding DCP message can be sent to a single device without waiting for Acknowledgements.

Transaction Correlation ID notes:

- **A device initiating a message on its own (such as Discover or an Event Notification message) will code this field as 00 00.**
- **The Transaction Correlation ID is typically set by the Device Controller to a value (except zero "0000"). The Device responding to the message will copy this value into the replying DCP message (ACK or Discover message).**
- **It is up to the Device Controller to manage the values of the Correlation IDs and to not reuse the same value again for some amount of time.**

6.2.9 Message Trailer

The message trailer uses the standard ATM CRC-10 used for OAM cells and AAL3/4.

7 Message Formats

7.1 Initiate Discover Message

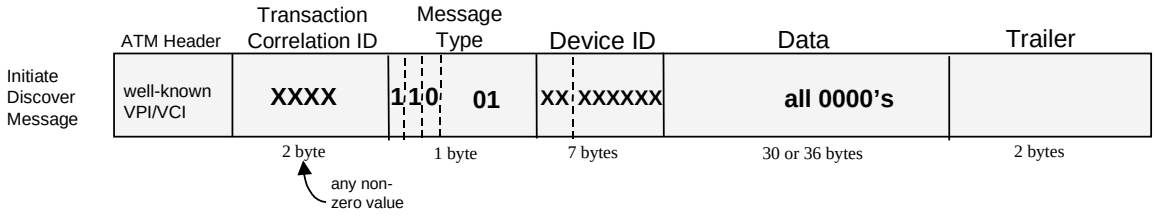


Figure 13- Initiate Discover Message Format

7.2 Discover Message and Discover ACK message

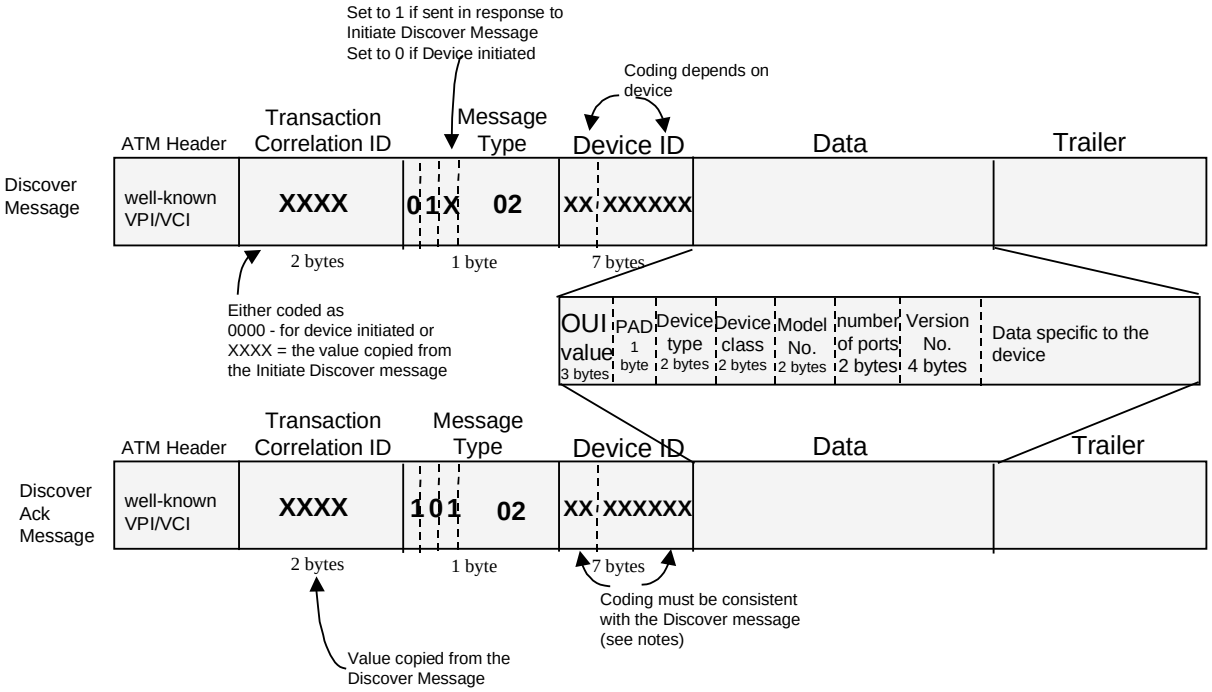


Figure 14- Discover and Discover ACK Message Format

The Data field of the Discover Messages is coded as follows:

Table 11 Discover Message Parameters

Field	Field description	Coding	Length
OUI	Organization Unique Identifier	ISO standard ISO/IEC 8802 available	3 bytes

		through IEEE (IEEE Std 802-1990)	
Device Type	DCP specific	see below for coding	2 bytes
Device Class	DCP specific	see below	2 bytes
Model No.	Vendor specified	see below	2 bytes
Number of ports	Number of ports on the device	see below	2 bytes
Version no.	Vendor specified	see below	4 bytes

7.2.1 Device type and Class

Table 12 Device Type Coding

Device Type	Value	Example
reserved	00	
ATM Switch Fabric	01	
ATM label translator	02	
Interworking Unit	03	Bridges to another Device controller domain
Internal Device – subcomponent	04	Integrated Circuit, Printed Circuit board

The Device Type and Class fields will be used to identify that the device belongs to a particular group of devices.

The Device Type field identifies, generally, what the device is, e.g. the specific device is a switch fabric.

The Device Class is an optional field that may be used to further refine the categorization of the device, e.g. the device is a uni-directional switch fabric.

The specific Device Types and Device Classes defined and their specific codings is a matter for further work.

7.2.2 Model Number

Model Number is used by each organization to differentiate the version of a particular device type and class.

For certain applications, there will be standards for virtual register maps and behaviors of a particular device type and class. To indicate that a device is using the standard for that device type and class, the coding value "00" is used.

Model Number	value
Industry standard model	0000
vendor-specific model number	0001-FFFF

7.2.3 Number of Ports

Each device will have from 1 to n ports. The value returned in this field will indicate the number of ports. This value is only for ports that are involved with the DCP protocol.

For devices which may interwork between two separate subsystems of a network element, the ports going into the other subsystem (for which this device controller is not controlling), are not counted.

The value of "00" is invalid in this field.

7.3 RESET Message

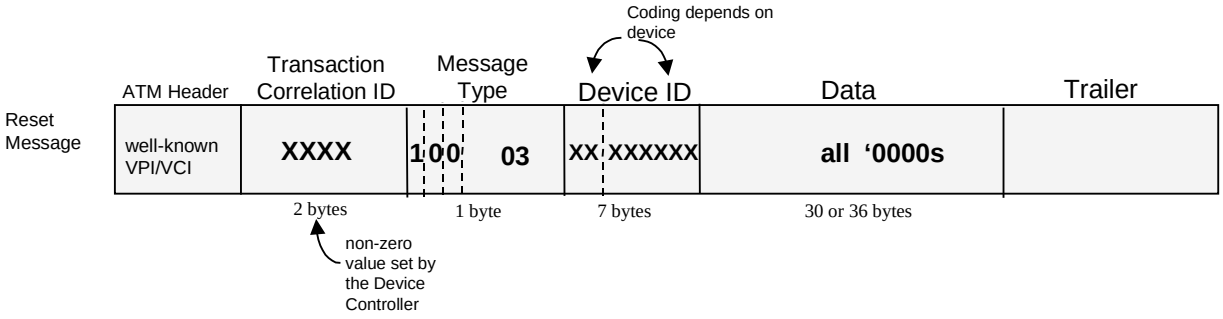


Figure 15- Reset Message Format

The Reset Message is sent from the Device Controller to the device to make the device perform a hard reset. This reset should put the device into its initial start up procedures. As part of these start up

procedures, the device should send a Discover message to the Device Controller.

Notes on Reset Message

- **only hard resets are supported with this message**
- **other types of resets should be supported through the use of a virtual register that is defined to enable other types of resets.**

7.4 CONNECT and CONNECT ACK Messages

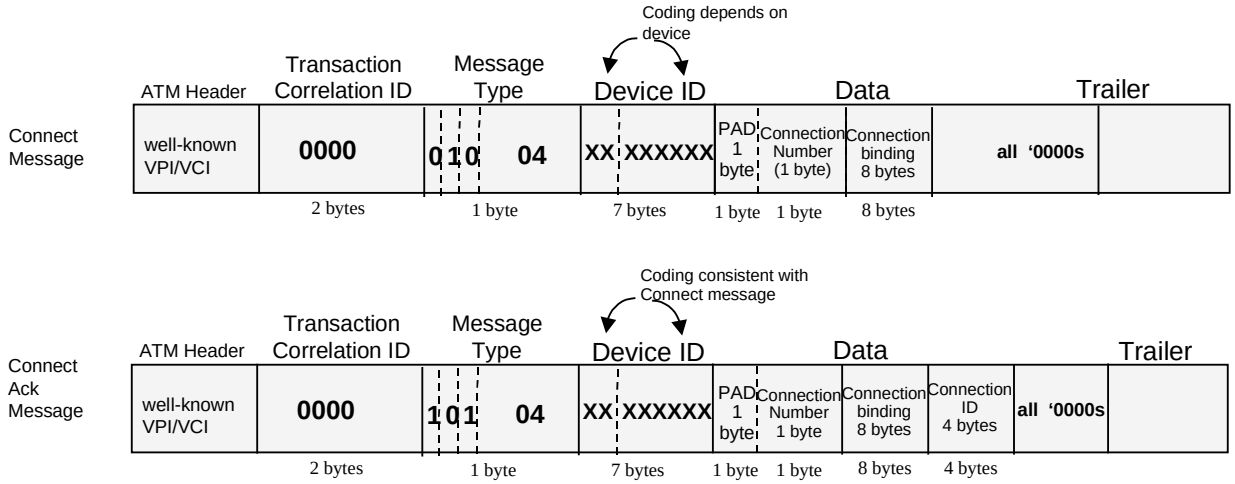


Figure 16- Connect and Connect ACK Message Format

The Connect message contains the Connection Number field. This field is used by the device to number the connections that it requires.

The Connection Binding field is 8 bytes. It is used to identify the connection endpoint required. Its coding is as follows:

Table 13 – Connection Binding Coding

Value	Meaning
00 00 00 00	not used
00 00 00 01	DCP protocol used over assigned connection
00 00 00 02 – FF FF FF FF	device-specific coding

The Connection ID field is coded as follows:

Table 14 Connection ID Coding

	PAD	VPI	VCI

UNI format	**8 bits coded "0"**	**8 bits**	**16 bits**
NNI format	**4 bits coded "0"**	**12 bits**	**16 bits**

Notes on Connect Message

- **if a device only needs one additional connection, then the value of the Connection number is always 1.**
- **if multiple connections are requested, the Connection Number field should be numbered sequentially starting at 1. The numbering is static and does not change.**

7.5 Virtual Register Read Message

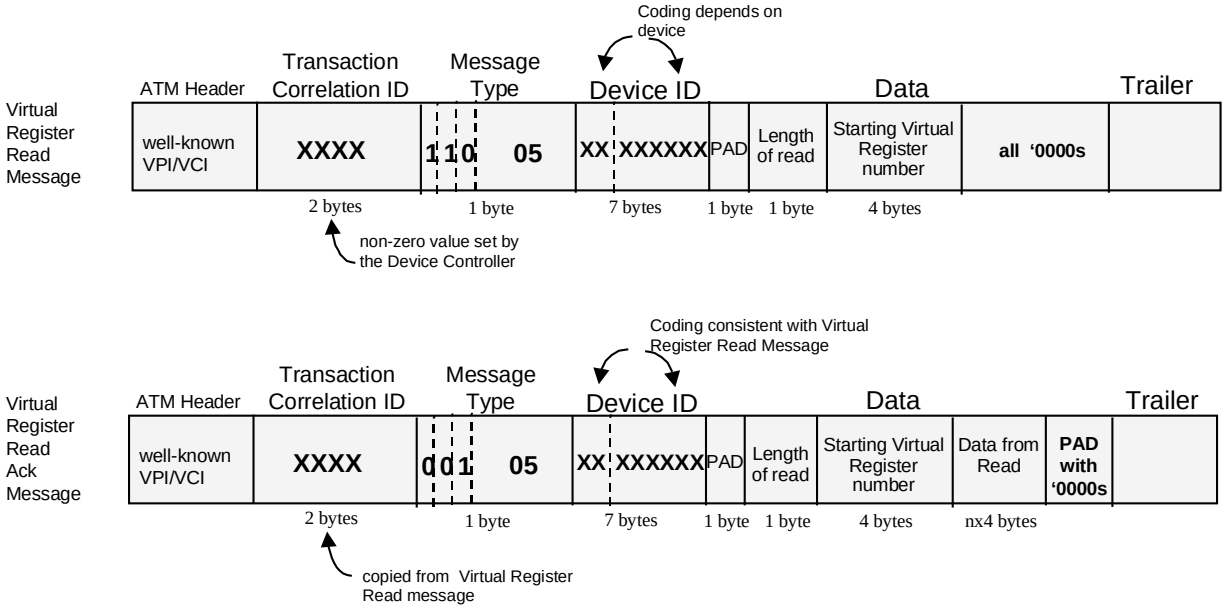


Figure 17- Virtual Register Read and Read ACK Message Format

The Virtual Register Read message is used to read virtual registers.

The Virtual Register Read message is sent from the Device Controller to the Device.

The message contains a length field to indicate the number of virtual registers to be read.

Virtual registers are always 4 bytes long. This means that the total number of bytes to read is equal to 4 times the length field. The length

specified must fit into the single cell Virtual Register Read ACK message.

The Virtual Register Read ACK is sent back to the Device Controller with the virtual register data in the message.

Notes on the Virtual Register Read Procedures:

- **Coding of zero length is not permitted.**
- **The device should read the registers in ascending order.**
- **An unrecognized Virtual Register index operation can be acknowledged with the Virtual Register Read ACK message with the Virtual Register number set to 00 00 00 00.**
- **a length exceeding a single cell size for the acknowledgement will send back the length of read = 00 and will include all 0000's for the data.**

7.6 Virtual Register Write Message

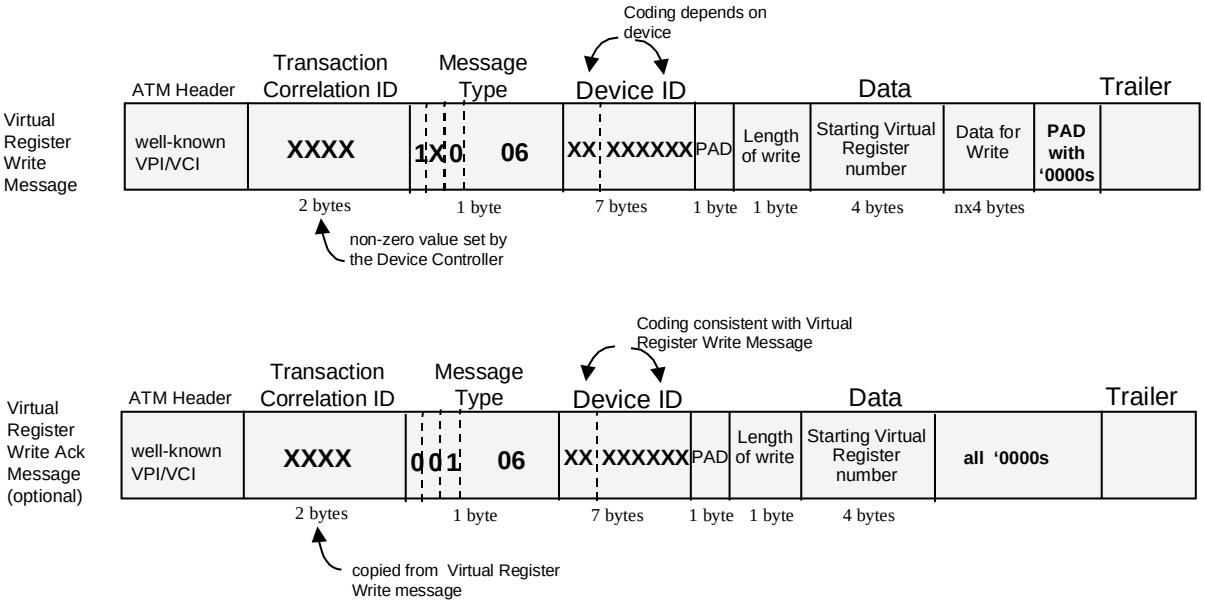


Figure 18- Virtual Register Write and Write ACK Message Format

The Virtual Register Write message provides the means to write data to the virtual registers.

The Virtual Register Write Message is sent from the Device Controller to the Device.

The fields are coded in the same way as for the Virtual Register Read procedures but include the data that is to be written to the registers.

The Virtual Register Write ACK message is sent back to the Device Controller from the device to acknowledge that the function has been performed.

The Virtual Register Write ACK message is optional. For such things as software downloads where a checksum operation is used, there may be no need for the acknowledgements.

Notes on the Virtual Register Write Procedures:

- **Coding of zero length is not permitted.**
- **The device should write the registers in ascending order.**
- **An unrecognized Virtual Register index can be acknowledged with the Virtual Register Write ACK message with the Virtual Register number set to 00 00 00 00. This may be detected partially through a write operation which would indicate a more severe problem (i.e. the Device Controller has an incorrect Virtual Register Map for the device).**

7.7 Event Notification

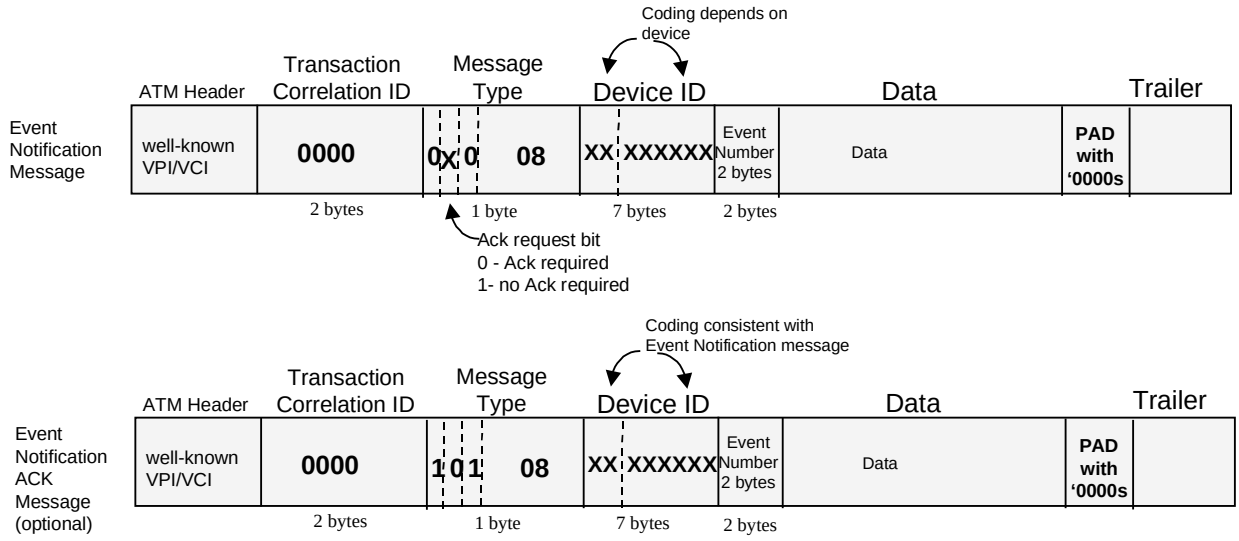


Figure 19 – Event Notification and ACK message format

The Event Notification message provides a means for the device to indicate to the Device Controller that there is something needing

attention.

The Event Notification messages are optional.

Where an Event Notification ACK is used, the content of the data field is copied into the Event Notification ACK directly from the Event Notification message. This is used by the device to correlate the messages.

The Event number is coded as follows:

Table 15 Event Number Coding

Code	Value	Data
00	not used	
01	unrecoverable error	
02	Virtual Register read/write failure	Virtual register number (4 bytes), Virtual Register value
03-FF	Reserved for common DCP event values	
100-FFFF	Reserved for device-specific events	

Notes on Event Notification procedures:

- **As an alternative to using Event Notification, it is possible to use polling techniques instead. The Device Controller would use a Virtual Register Read to read status registers from the device. This may be required in some applications to avoid the use of unsolicited messages from the device to the Device Controller.**
- **The use of the Event Notification ACK is optional and is not recommended.**

7.8 Device-specific Virtual Register Operations Messages

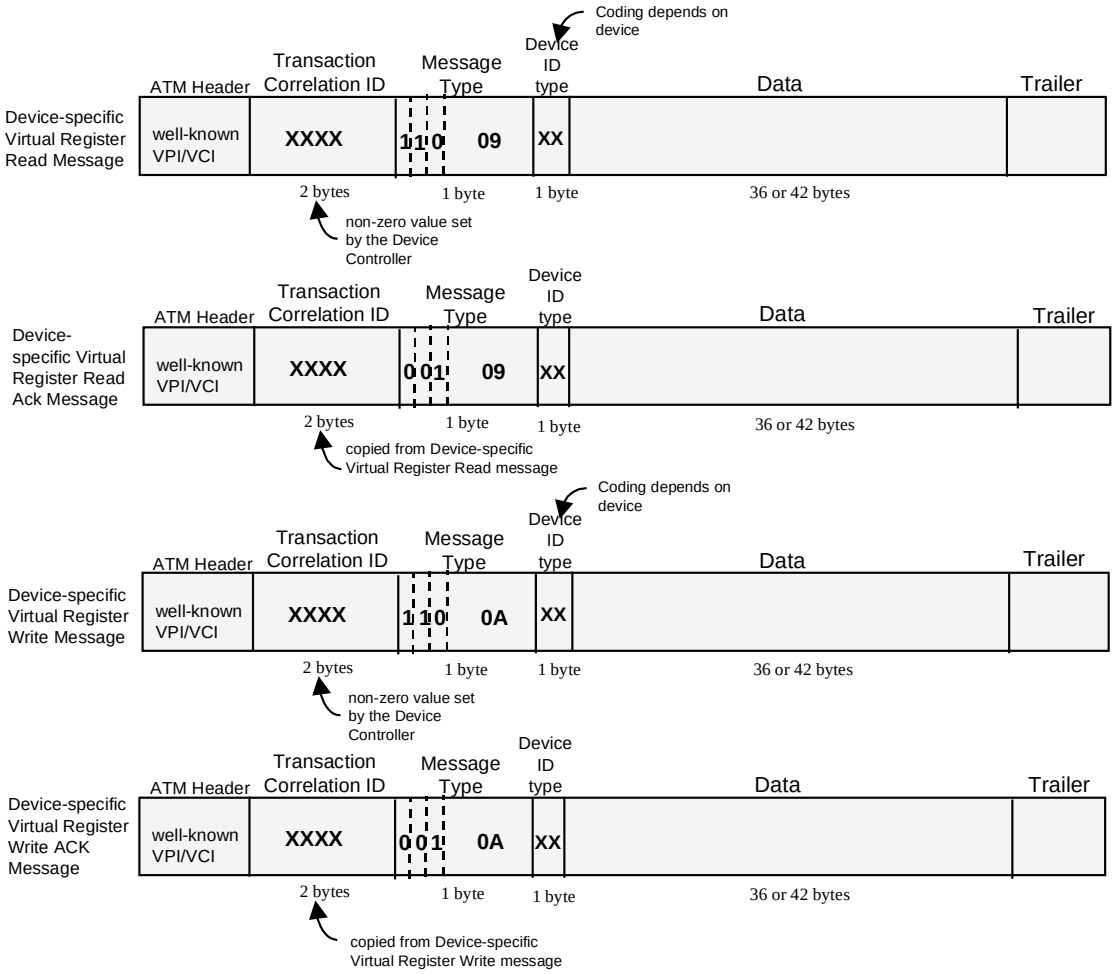


Figure 20 Device-Specific Virtual Register Ops Message Formats

There are no standard formats for these messages.

Notes on Device-Specific Virtual Register operations:

- **For devices which do not require a Device ID, the Device ID type shall be coded “05” to indicate that there is no Device ID value contained in the message.**

8 DCP Notes and Macros

8.1 Downloading software loads

To download software to the device, a block of Virtual Registers write operations can be performed.

To verify that the complete load has been downloaded, some type of

checksum should be performed. This can be performed through the use of a virtual register designated as a command register. The checksum can be performed on the complete software load, or it can be performed on subsets of the complete load. This is up to the individual device and how its virtual registers are defined.

8.2 Acknowledgements

Acknowledgement messages are sent in reply to messages that request an ACK.

The acknowledgement is verified by matching the Transaction Correlation ID with the Transaction Correlation ID of the original message.

For shared media and topology-based source routing, the Transaction Correlation ID and the Device ID value are checked to ensure that a correct acknowledgement has been received.

For point-to-point connections, the VPI/VCI in the header of the message and the Transaction Correlation ID are checked to ensure that a correct acknowledgement has been received.

Failure to receive an acknowledgement at the Device Controller will be detected at the Application layer. Timers may be used to detect this.

Failure to receive an acknowledgement at the device may require a timer at the device. This is implementation dependent.

8.3 Outstanding transactions

There is no defined limit on the number of outstanding transactions. However, there may be a need for the Device Controller to maintain a limit on transactions to the device if there are processing limitations at the device. The method to do this is implementation dependent.

8.4 Multi-referenced registers

A device virtual register map may index more than once, the same device component e.g. memory location. This would mean that multiple virtual register values in the map point to the same device component. This may be necessary to simplify or group virtual registers based on the operations that need to be performed on the

device.

8.5 Soft Resets

For soft resets, it is recommended that a virtual register be defined which acts as a command register to perform a soft reset on the device. There could be different levels of reset and these could be identified by the value that was written to this command register.

APPENDIX A.Using DCP for Topology-based Source Routing

When the first byte of the Device ID field – the Device ID type field - indicates that the address format is topology-based Source Routing (codepoint = 03), the remaining six bytes are not an explicit Device ID as such, but instead are a bit map describing how to route the cell to the destination. This is known as ‘topology-based Source Routing’ and also known elsewhere as ‘source-routing’. The routing information is removed from the cell as it is consumed on the way from the controller to device and is added in the reverse manner for cells in the reverse direction.

The Destination bit indicates the direction of travel, which is a ‘one’ for cells from the controller to the device. Intermediate devices forwarding the cells either add or remove routing information depending on the direction of travel. A device knows that a cell is destined to it when the Destination Bit is set to `one' and there is no routing information to interpret (the value will actually be 00 00 00 00 00 01 because a marker bit is used to indicate the boundary of the bit map).

For intermediate devices involved in topology-based Source Routing, the number of routing bits required at that device is known, both by the device and by the controller, since the controller knows the Device type of the intermediate device. The number of routing bits used by a device is lower-bounded by the nearest integer above or equal to the log to the base 2 of the number of ports. However, a particular device might use more bits than this to support expansion in various ways, or perhaps to provide multicast of control cells or any future application.

For control cells moving from the device to the controller, the routing operation is to add to the Device ID field the identification of the input port that the control cell arrived on and to forward the cell on the port which is directed towards the controller. For control cells moving from the controller to the device, the routing operation is to remove from the Device ID field the identification of the port that the cell is to be forwarded on and to forward the cell on that port. The information inserted or removed at a hop is a routing component.

The details of the handling of the six-byte Device ID field when topology-based Source Routing are as follows. The six-byte field is treated as a 48-bit field with the most significant bit of the first byte being the first bit and the least significant bit of the last byte being the last. The field is not self-describing in that no bits are used to delimit the individual routing components. Instead, each device knows how many bits to insert or extract at each hop.

A value of all zeroes in the routing field is not used. A null route is where all bits are zero except the least significant bit. (Do not confuse a null route with null device ID). A device generating a cell for the controller should start it with the null route. To add a routing component to the 48 bit routing field, the existing data is shifted left the appropriate number of bits and the new information is inserted at the right. To remove a routing component from the 48 bit routing field, the required number of bits should be extracted from the right hand end and the data shifted right accordingly, with zeros added at the left.

In the case of a misdirected control cell in the direction from the controller to the device, it may be that the identification for the port to forward on does not give a valid port. Such cells should be discarded.

In the other direction, in the case of a misconfigured network element or a bad topology map for the network element, control cells may be routed in what is believed by the intermediate nodes to the correct direction towards the controller, but in fact the route is too long or incorrect. In this case, the routing field will become full and it will not be possible to add on the ingress port identification. Such cells should be discarded by the intermediate node.

For example, consider an 8 port switch connected between a device and a controller with the device on port 3 and the controller on port 7. This switch simply uses a three-bit routing component directly as a port number. If it receives a control cell with null route in the Device ID field from the device with the Destination Bit indication set as towards the controller, then it will convert the null route to 00 00 00 00 00 0B and send the cell out on port 7. If it instead received a cell from the controller on port 7 with the reverse direction and routing data in the Device ID field is 00 00 00 00 00 0B, then it would remove the three-bit routing component, converting it back to the null route and forward it on port 3 to the device. If it now received a cell with null route in the Device ID field from the controller on port 7 and direction towards the device, it would realize that the control cell was intended for it and it might reply with a similar control cell on port 7 which would have the reverse direction, but the same null route.

Note that is also possible for the device to communicate with the controller by routing the protocol over a standard PVC or SVC set up in the switch, in which case the switch would operate normally and not modify the cell payload. This just depends on whether the VCIs are DCP VCI value or normal VCC values.

DCP devices which do not support topology-based Source Routing should pass on all self-routed cells without modification and the output port selected for this is either obvious, since there might only be two ports, or it has been configured using some other means (e.g. DCP Virtual Register Write).