RapidIO™ Interconnect Specification Annex 3: Consolidated Packet Formats

4.2, 3/2021

RapidIO.org

Revision History

Revision	Description	Date
1.0	First release	03/28/2021

Questions regarding RapidIO.org, specifications, or membership should be forwarded to:
RapidIO.org
8650 Spicewood Springs #145-515
Austin, TX 78759
512-827-7680 Tel.

Table of Contents

Chapter 1 Introduction

1.1 Scope

This document captures the packet formats of all packet types defined in the RapidIO standard. This annex is informative. If there is a conflict between this annex and Part 1, Part 2, Part 3, Part 5, Part 6, Part 9, Part 10, or Part 12, the field layouts required by that part of the standard takes precedence.

All RapidIO packets contain information from three layers; the Physical Layer defined in Part 6 of the Standard, the Transport Layer defined in Part 2, and the various Logical Layers defined in Parts 1, 3, 5, 9, and 10. The RapidIO standard is organized in a modular fashion in that each Part in the standard defines only the requirements associated with that function. The advantage of this organization is that each Logical Layer protocol is described independently of the others, and thus an implementer can focus on the logical layer protocols that are of interest to them without being distracted by other logical layer protocols that may not be of interest. Because of this, however, each Part in the standard does not show all of the fields from all parts of the standard and how they are laid out with respect to each other. This Annex solves that problem, and Chapter 2 explicitly documents the layout of each packet type with all of the fields from the physical, transport, and logical layers together in easy to read diagrams. Table 1-1 provides hyperlinks to facilitate finding the layout diagrams for a specific packet format.

Table 1-1. Packet Types Directory

Format	Section
Type 0 Packets	Section 2.1
Type 1 Packet Dev 8 Transport 34-bit Addressing	Section 2.2.1
Type 1 Packet Dev 8 Transport 50-bit Addressing	
Type 1 Packet Dev 8 Transport 66-bit Addressing	
Type 1 Packet Dev 16 Transport 34-bit Addressing	Section 2.2.2
Type 1 Packet Dev 16 Transport 50-bit Addressing	
Type 1 Packet Dev 16 Transport 66-bit Addressing	
Type 1 Packet Dev 32 Transport 34-bit Addressing	Section 2.2.3
Type 1 Packet Dev 32 Transport 50-bit Addressing	
Type 1 Packet Dev 32 Transport 66-bit Addressing	
Type 2 Packet Dev 8 Transport 34-bit Addressing	Section 2.3.1
Type 2 Packet Dev 8 Transport 50-bit Addressing	
Type 2 Packet Dev 8 Transport 66-bit Addressing	
Type 2 Packet Dev 16 Transport 34-bit Addressing	Section 2.3.2
Type 2 Packet Dev 16 Transport 50-bit Addressing	
Type 2 Packet Dev 16 Transport 66-bit Addressing	
Type 2 Packet Dev 32 Transport 34-bit Addressing	Section 2.3.3
Type 2 Packet Dev 32 Transport 50-bit Addressing	
Type 2 Packet Dev 32 Transport 66-bit Addressing	
Type 3 Packets	Section 2.4
Type 4 Packets	Section 2.5
Type 5 Packet Dev 8 Transport 34-bit Addressing	Section 2.6.1
Type 5 Packet Dev 8 Transport 50-bit Addressing	Section 2.6.2
Type 5 Packet Dev 8 Transport 66-bit Addressing	Section 2.6.3
Type 5 Packet Dev 16 Transport 34-bit Addressing	Section 2.6.4
Type 5 Packet Dev 16 Transport 50-bit Addressing	Section 2.6.5
Type 5 Packet Dev 16 Transport 66-bit Addressing	Section 2.6.6
Type 5 Packet Dev 32 Transport 34-bit Addressing	Section 2.6.7
Type 5 Packet Dev 32 Transport 50-bit Addressing	Section 2.6.8
Type 5 Packet Dev 32 Transport 66-bit Addressing	Section 2.6.9
Type 6 Packet Dev 8 Transport 34-bit Addressing	Section 2.7.1
Type 6 Packet Dev 8 Transport 50-bit Addressing	Section 2.7.2
Type 6 Packet Dev 8 Transport 66-bit Addressing	Section 2.7.3
Type 6 Packet Dev 16 Transport 34-bit Addressing	Section 2.7.4
Type 6 Packet Dev 16 Transport 50-bit Addressing	Section 2.7.5
Type 6 Packet Dev 16 Transport 66-bit Addressing	Section 2.7.6
Type 6 Packet Dev 32 Transport 34-bit Addressing	Section 2.7.7
Type 6 Packet Dev 32 Transport 50-bit Addressing	Section 2.7.8
Type 6 Packet Dev 32 Transport 66-bit Addressing	Section 2.7.9
Type 7 Packets	Section 2.8
Type 8 Packet Dev 8 Transport	Section 2.9.1
Type 8 Packet Dev 16 Transport	Section 2.9.2
Type 8 Packet Dev 32 Transport	Section 2.9.3
Type 9 Packet Format Dev8 Even Number of Half Words	Section 2.10.1
Type 9 Packet Format Dev8 Odd Number of Half Words	Section 2.10.2
Type 9 Extended Packet Format Dev8	Section 2.10.3

Table 1-1 Packet Types Directory (Continued)

Format	Section
Type 9 Packet Format Dev16 Even Number of Half Words	Section 2.10.4
Type 9 Packet Format Dev16 Odd Number of Half Words	Section 2.10.5
Type 9 Extended Packet Format Dev16	Section 2.10.6
Type 9 Packet Format Dev32 Even Number of Half Words	Section 2.10.7
Type 9 Packet Format Dev32 Odd Number of Half Words	Section 2.10.8
Type 9 Extended Packet Format Dev32	Section 2.10.9
Type 10 Packet Dev 8 Transport	Section 2.11.1
Type 10 Packet Dev 16 Transport	Section 2.11.2
Type 10 Packet Dev 32 Transport	Section 2.11.3
Type 11 Packets	Section 2.12
Type 12 Packets	Section 2.13
Type 13 Packet Dev 8 Transport	Section 2.14.1
Type 13 Packet Dev 16 Transport	Section 2.14.2
Type 13 Packet Dev 32 Transport	Section 2.14.3
Type 14 Packets	Section 2.15
Type 15 Packets	Section 2.16

1.2 Overview

The RapidIO specification organizes transactions into transaction classes. All RapidIO packets are part of a transaction class identified by the Format Type field (ftype) in Byte 1. There is a one-to-one mapping between a transaction class and its Format Type. The ftype field is used to determine the general format of the packet. Some transaction classes have multiple transaction types identified by a transaction type field, also called ttype. The transaction type can impose restrictions on how that transaction class can be used. Table 1-2 lists the transaction classes with their associated ftype, and ttypes when applicable. Whether or not a packet includes an embedded data payload also impacts its format, and that is listed in Table 1-2 as well. In some cases, multiple logical layer protocols use the same transaction class to define transactions (e.g. both Part 1 and Part 5 define transactions using Type 5 packets). Table 1-3 shows a cross reference for each packet type and where in the standard each portion of the packet is defined.

The RapidIO protocol uses data symbols and control symbols to transfer packets across a link. The protocol also uses control symbols to demark packets, acknowledge packets, and perform the various fault, link, and time management functions defined in the standard. Chapter 3 provides an overview of the control symbols involved in packet formation and how they relate to the packet information described in Chapter 2.

This Annex does not attempt to describe the semantics of how packets are used to perform transactions. For that information, please see the Part that is the Specification for that protocol.

Table 1-2. RapidIO Format and Transaction Types

ftype	Class Description	ttype	Transaction Description	Payload
0	Implementation-Defined			
1	Intervention-Request Class	0	READ_OWNER	No
		1	READ_TO_OWN_OWNER	
		2	IO_READ_OWNER	
			Reserved	
2	Request class	0	READ_HOME	No
		1	READ_TO_OWN_HOME	
		2	IO_READ_HOME	
		3	DKILL_HOME	
		4	NREAD transaction	
		5	IKILL_HOME	
		6	TLBIE	
		7	TLBSYNC	
		8	IREAD_HOME	
		9	FLUSH without data	
		10	IKILL_SHARER	
		11	DKILL_SHARER	
		12	ATOMIC inc: post-increment the data	
		13	ATOMIC dec: post-decrement the data	
		14	ATOMIC set: set the data (write 0b11111...)	
		15	ATOMIC clr: clear the data (write 0b00000...)	
3	Reserved			
4	Reserved			
5	Write Class	0	CASTOUT	Yes
		1	FLUSH with data	
		2-3	Reserved	
		4	NWRITE transaction	
		5	NWRITE_R transaction	
		6-11	Reserved	
		12	ATOMIC swap: read and return the data, unconditionally write with supplied data.	
		13	ATOMIC compare-and-swap: read and return the data, if the read data is equal to the first 8 bytes of data payload, write the second 8 bytes of data to the memory location	
		14	ATOMIC test-and-swap: read and return the data, compare to 0, write with supplied data if compare is true	
		15	Reserved	
6	Streaming-Write Class	N/A		Yes
7	Flow Control Class	N/A		No
8	Maintenance Class	0	Maintenance read request	No
		1	Maintenance write request	Yes
		2	Maintenance read response	Yes
		3	Maintenance write response	No
		4	Maintenance port-write request	Yes
		5-15	Reserved	

Table 1-2 RapidIO Format and Transaction Types (Continued)

ftype	Class Description	ttype	Transaction Description	Payload
9	Data-Streaming Class	N/A		Yes
10	Doorbell Class	N/A		Yes
11	Message Class	N/A		Yes
12	Reserved			
13	Response Class	0	RESPONSE transaction with no data payload, including DOORBELL response	No
		1	MESSAGE RESPONSE transaction	No
		2-7	Reserved	
		8	RESPONSE transaction with data payload	Yes
		9-15	Reserved	
14	Reserved			
15	Implementation-Defined			

Table 1-3. Packet Types Overview

	Part 1	Part 2	Part 3	Part 4	Part 5	Part 6	Part 7	Part 8	Part 9	Part 10	Part 11	Part 12	Implementation Defined
Type 0	L	L	T	P	L	S							L
Type 1			T	P	L	S							
Type 2	L		T	P	L	S							
Type 3			T	P		S							
Type 4			T	P		S							
Type 5	L		T	P	L	S							
Type 6	L		T	P		S							
Type 7			T	P		S			L				
Type 8	L		T	P		S							
Type 9			T	P		S				L			
Type 10		L	T	P		S							
Type 11		L	T	P		S							
Type 12			T	P		S							
Type 13	L	L	T	P	L	S							
Type 14			T	P		S							
Type 15	L	L	T	P	L	S							L
Control Symbols				P		S						S	
	L = Logical Layer Fields Defined in this Part												
	T = Transport Layer Fields Defined in this Part												
	S = Serial Physical Layer Fields Defined in this Part												
	P = Parallel Physical Layer Fields Defined in this Part (Deprecated)												

1.2.1 Packet Header Variability

There are three different device id widths and three different address widths defined by the RapidIO standard. The width of the device ids in a packet is determined by the tt header field defined in the transport layer; tt=0 indicates 8-bit device ids, tt=1 indicates 16-bit device ids, and tt=2 indicates 32-bit device ids. A RapidIO system will often be designed to use a single device id width for all devices in the system. However, the tt field allows a system to have a mix of device id widths. Since all packets include the transport layer fields, every transaction class has at least 3 potential packet formats; one each for 8-bit, 16-bit, and 32-bit device ids.

There are also three address widths supported by RapidIO (34, 50, and 66-bit address widths). The address width supported by a device also impacts the format of the packets created by that device. However, there is no field in the packet header that defines the width of that packet's embedded address. Thus, devices that support the transaction classes that have embedded addresses are typically designed to support a single address width, and the system designer must ensure that all devices that communicate with each other support the same address width. It is possible, however, to have devices that support different address widths in the same system if a bridge is used to perform the translation, if devices are constrained to only communicate with other devices that support the same address width, or if a proprietary mechanism is used by a target to associate an address width with one or more Source or Target Device Identifiers.

A packet's header format is primarily determined by its ftype. However, since there are three possible device id widths and three possible address widths that are used independently, packet classes that include an address in their packet definition have nine possible packet formats. In practice, a single device usually only supports a subset of the permutations. Table 1-4 shows the header sizes for all legal permutations of each transaction class. If only a single header size is listed for a particular Device Id width, then there is no address embedded in the packet header.

Table 1-4. Header Size by Format Type

		Header Size in Bytes								
		tt=0, 8-bit Dev IDs			tt=1, 16-bit Dev IDs			tt=2, 32-bit Dev IDs		
		Address Width			Address Width			Address Width		
	Ftype	**34**	**50**	**66**	**34**	**50**	**66**	**34**	**50**	**66**
Requests	0	DBD[1]	DBD	DBD	DBD	DBD	DBD	DBD	DBD	DBD
	1	10	12	14	12	14	16	16	18	20
	2	10	12	14	12	14	16	16	18	20
	5	10	12	14	12	14	16	16	18	20
	6	8	10	12	10	12	14	14	16	18
	7	6			8			12		
	8	10			12			16		
	9 Start/End/Single)	8			10			14		
	9 (Continuation)	6			8			12		
	9 (Traffic Mngmnt)	12			14			18		
	10	8			10			14		
	11	6			8			12		
Responses	8	10			12			16		
	13	6			8			12		
	15	DBD			DBD			DBD		

[1] DBD = Packet format is “Defined by Device”. The packet may or may not have an address.

1.2.2 Max Packet Sizes

The maximum number of data symbols transmitted over the physical layer for a single packet is based on its format type and device id width, and its address width and payload length when appropriate. The maximum sized payload in any RapidIO packet is 256 bytes, although some transaction classes further restrict the allowed payload size. This maximum payload size reduces the buffering required in switches and reduces the jitter through the network. Table 4 lists the maximum size of each packet format including all data bytes. These sizes include the header, the embedded CRCs, the number of bytes in that format's max sized payload, and any bytes required for pad at the end to satisfy alignment rules. The control symbols that are required at the beginning and end of packets are not included in this table, since they can vary based on the efficiency of the implementation.

If only a single size is listed for a particular Device Id width, then there is no address embedded in the packet header.

Table 1-5. Max Packet Sizes by Format Type

		Size in Bytes								
		tt=0, 8-bit Dev IDs			tt=1, 16-bit Dev IDs			tt=2, 32-bit Dev IDs		
		Address Width			Address Width			Address Width		
	Ftype	**34**	**50**	**66**	**34**	**50**	**66**	**34**	**50**	**66**
Requests	0	DBD[2]	DBD	DBD	DBD	DBD	DBD	DBD	DBD	DBD
	1	12	16	16	16	16	20	20	20	24
	2	12	16	16	16	16	20	20	20	24
	5[3]	272[4]	272	276	272	276	276	276	280	280
	6	268	272	272	272	272	276	276	276	280
	7[5]	8			12			16		
	8[6]	76			80			84		
	9 (Start/End/Single)	268			272			276		
	9 (Continuation)	268			268			272		
	9 (Traffic Mngmnt)	16			16			20		
	10	12			12			16		
	11	268			268			272		
Responses	8	76			80			84		
	13	268			268			272		
	15	DBD			DBD			DBD		

[2] DBD = Packet format is “Defined by Device”. The packet may or may not have an address.

[3] Different Type 5 transaction types have different maximum packet sizes. The max sizes shown here are for the largest transaction formats, which include at least NWRITE and NWRITE_R.

[4] The maximum payload size is 256 bytes for packet Format Types 5, 6, 11, 13, and 9 (except Traffic Management).

[5] There are no addresses in packet Format Types 7, 9, 10, 11, and 13.

[6] Max sized payload for Type 8 transactions is 64 bytes.

Chapter 2 Packet Formats

This chapter explicitly lays out every bit in each packet format, including all fields from all layers color-coded to highlight the layer in which they are defined. A legend describing the color coding is shown in Figure 2-1.

Figure 2-1. Field Type Legend

There is a section in this chapter devoted to each transaction class. Each transaction class section has a subsection for each device id width that lays out how all fields align when that device id width is used. Since the address width also impacts the location of the fields that follow, if the transaction type contains an address, then that subsection contains at least 3 diagrams.

Each diagram is laid out in 64-bit rows, with each column representing 1 bit. Since the RapidIO standard is big-endian, byte 0 is on the left of each diagram, and byte 7 is on the right. The field locations and widths align with the bit positions in the ruler at the top of each section. The byte number of the first byte of the row is shown on the left side of the diagram.

The physical layer requires either one or two CRCs in each packet. If a packet contains more than 80 total bytes, then two CRCs are required, an Early CRC and a Final CRC. In order to keep the byte number equations in each diagram simple, a diagram is included for each of the boundary cases associated with the Early CRC (see Types 5, 6, and 13). The physical layer also requires that every packet contain an integer multiple of 4 bytes. Thus, where it is required, the diagrams show a Logic 0 Pad at the end that fills to the nearest 32-bit boundary.

2.1 Type 0 Request Packets

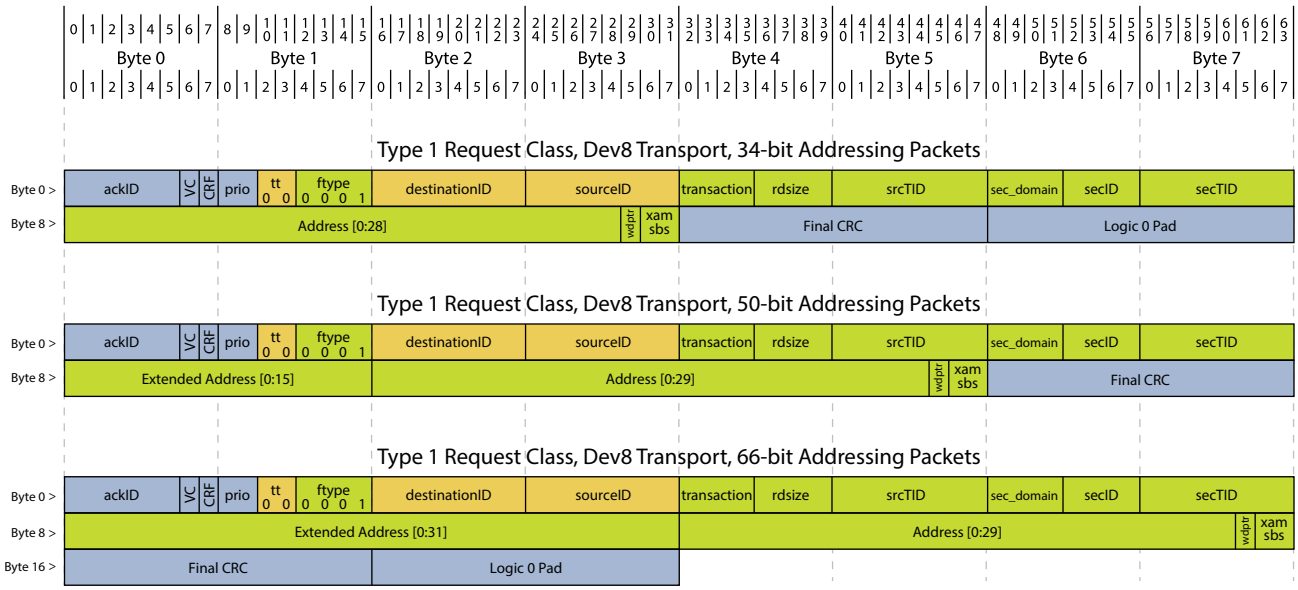
The Type 0 request packet format is reserved for implementation-defined functions. The Logical Layer fields are implementation defined.

2.2 Type 1 Request Packets

The Type 1 request packet format is used for the Intervention-Request Class. The Logical Layer fields for the Intervention-Request Class are defined in Part 5.

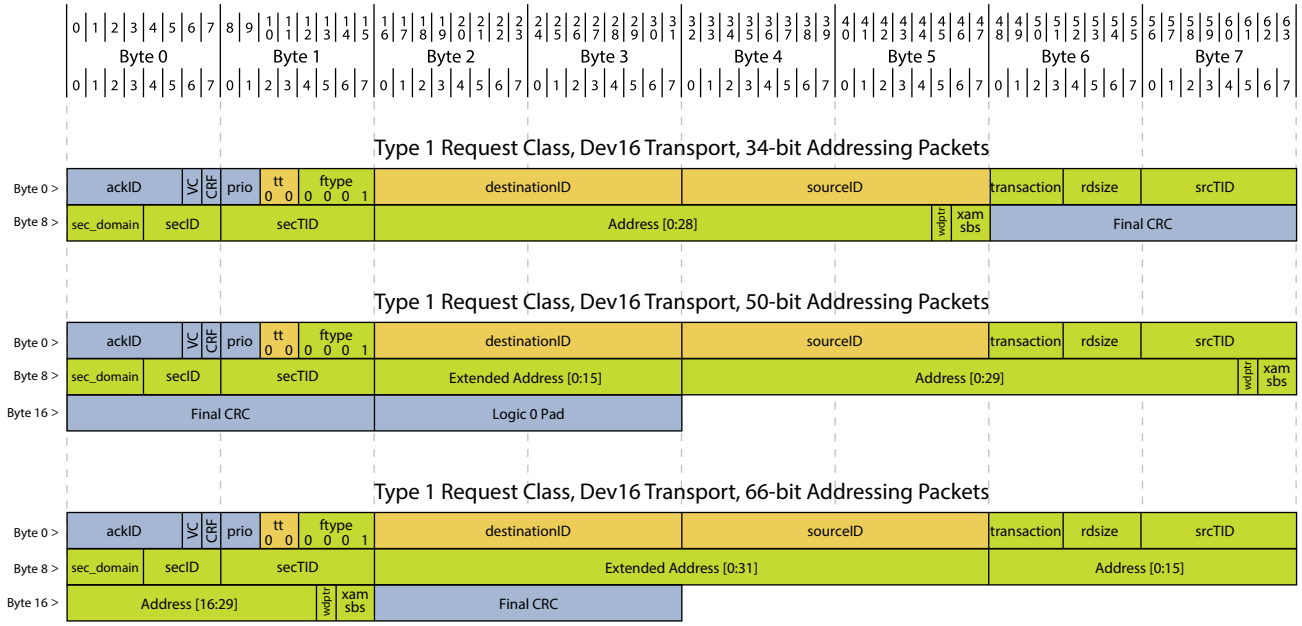
2.2.1 Type 1 Request Packet: Dev8 Transport Format

Figure 2-2. Type 1 Request Packet: Dev8 Transport Format



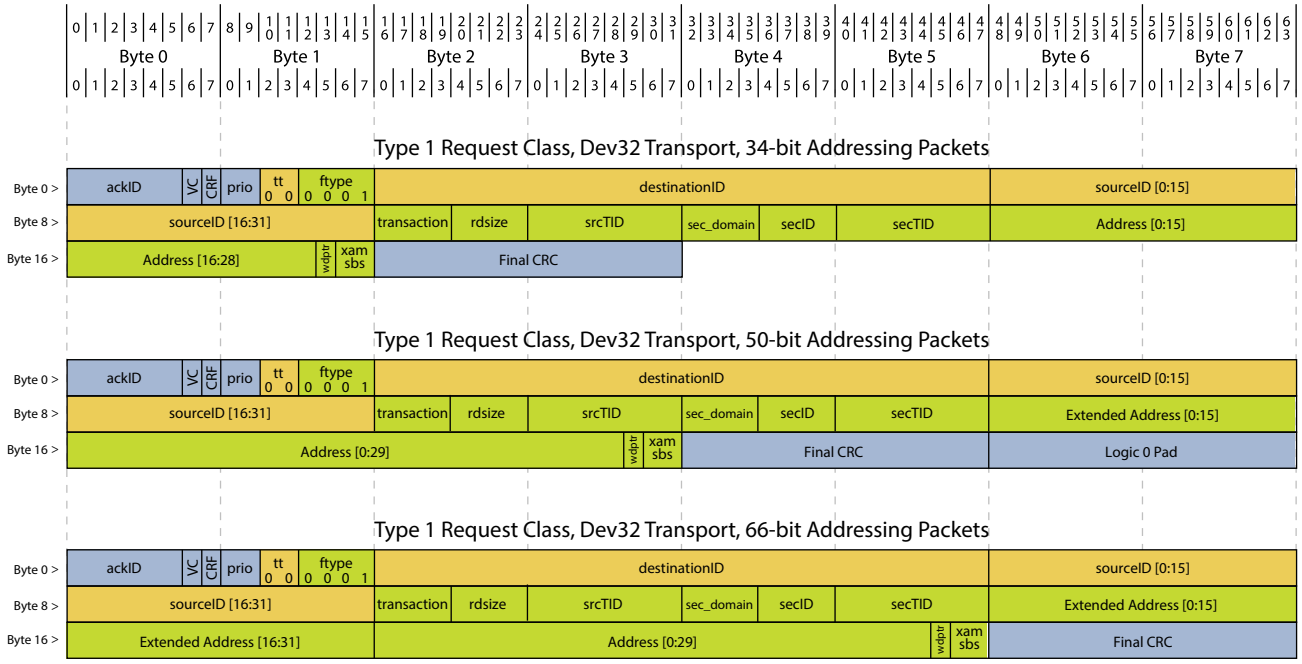
2.2.2 Type 1 Request Packet: Dev16 Transport Format

Figure 2-3. Type 1 Request Packet: Dev16 Transport Format



2.2.3 Type 1 Request Packet: Dev32 Transport Format

Figure 2-4. Type 1 Request Packet: Dev32 Transport Format

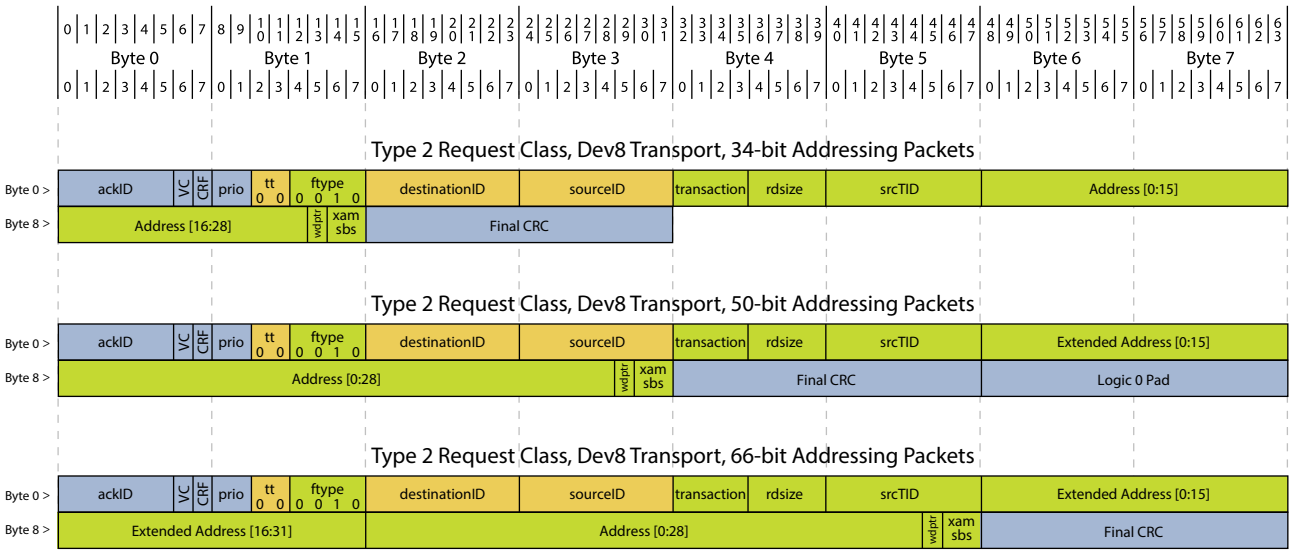


2.3 Type 2 Request Packets

The Type 2 request packet format is used for the Request Class. The Logical Layer fields for the Request Class are defined in Parts 1 and 5.

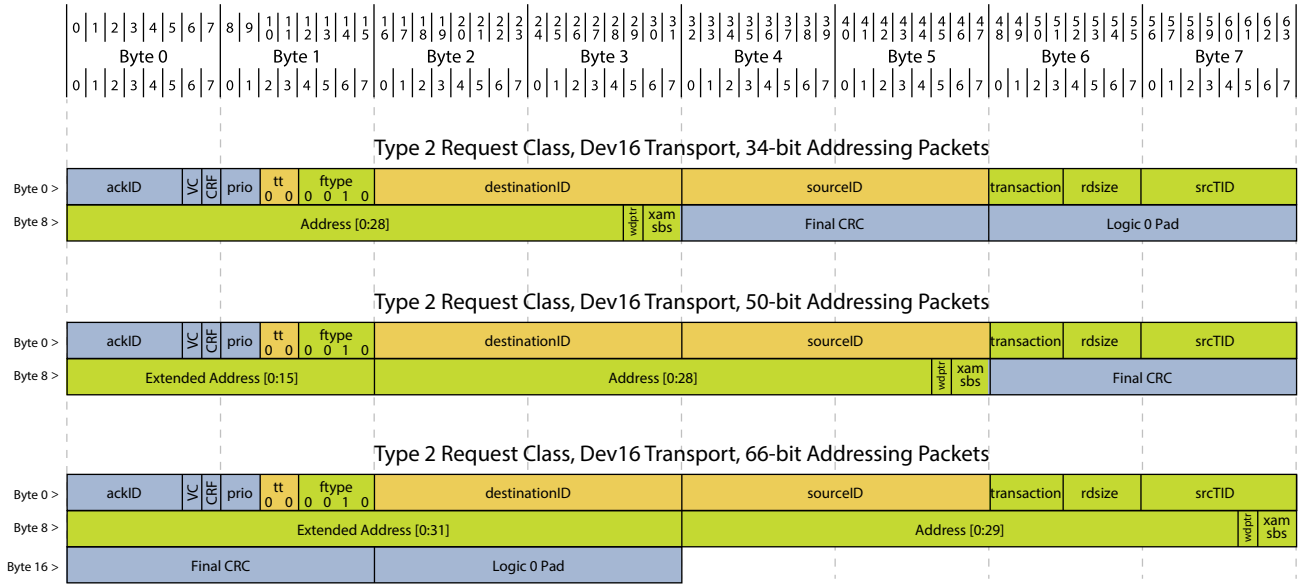
2.3.1 Type 2 Request Packet: Dev8 Transport Format

Figure 2-5. Type 2 Request Packet: Dev8 Transport Format



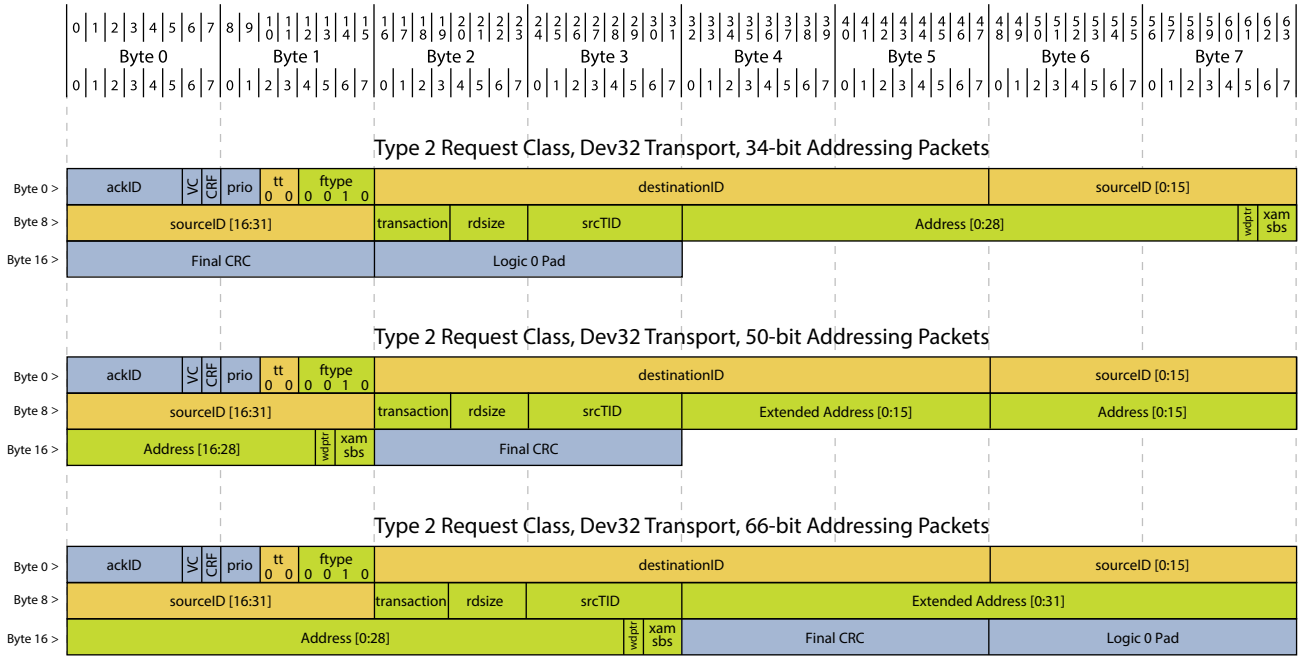
2.3.2 Type 2 Request Packet: Dev16 Transport Format

Figure 2-6. Type 2 Request Packet: Dev16 Transport Format



2.3.3 Type 2 Request Packet: Dev32 Transport Format

Figure 2-7. Type 2 Request Packet: Dev32 Transport Format



2.4 Type 3 Request Packets

The Type 3 request packet format is reserved and not yet defined.

2.5 Type 4 Request Packets

The Type 4 request packet format is reserved and not yet defined.

2.6 Type 5 Request Packets

The Type 5 request packet format is used for the Write Class. The Logical Layer fields for the Write Class are defined in Parts 1 and 5.

2.6.1 Type 5 Request Packet: Dev8 Transport: 34-bit Addressing Format

Figure 2-8. Type 5 Request Packet: Dev8 Transport: 34-bit Addressing Format

Notes: N is the number of double-words in the payload.
n = N-1

2.6.2 Type 5 Request Packet: Dev8 Transport: 50-bit Addressing Format

Figure 2-9. Type 5 Request Packet: Dev8 Transport: 50-bit Addressing Format

Notes: N is the number of double-words in the payload.
n = N-1

2.6.3 Type 5 Request Packet: Dev8 Transport: 66-bit Addressing Format

Figure 2-10. Type 5 Request Packet: Dev8 Transport: 66-bit Addressing Format

Notes: N is the number of double-words in the payload.
n = N-1

2.6.4 Type 5 Request Packet: Dev16 Transport: 34-bit Addressing Format

Figure 2-11. Type 5 Request Packet: Dev16 Transport: 34-bit Addressing Format

Notes: N is the number of double-words in the payload.
n = N-1

2.6.5 Type 5 Request Packet: Dev16 Transport: 50-bit Addressing Format

Figure 2-12. Type 5 Request Packet: Dev16 Transport: 50-bit Addressing Format

Notes: N is the number of double-words in the payload.
n = N-1

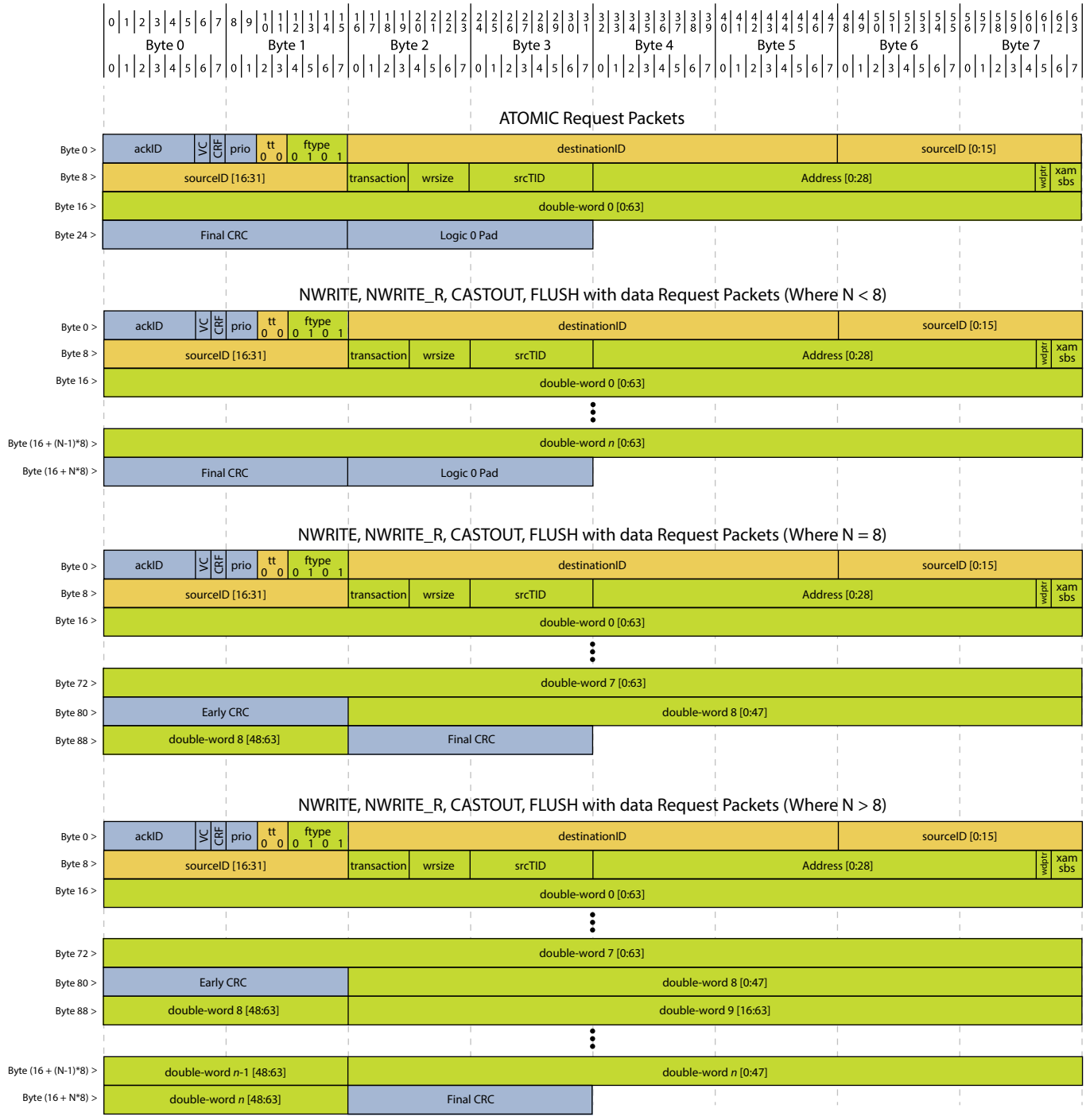
2.6.6 Type 5 Request Packet: Dev16 Transport: 66-bit Addressing Format

Figure 2-13. Type 5 Request Packet: Dev16 Transport: 66-bit Addressing Format

Notes: N is the number of double-words in the payload.
n = N-1

2.6.7 Type 5 Request Packet: Dev32 Transport: 34-bit Addressing Format

Figure 2-14. Type 5 Request Packet: Dev32 Transport: 34-bit Addressing Format



Notes: N is the number of double-words in the payload.
n = N-1

2.6.8 Type 5 Request Packet: Dev32 Transport: 50-bit Addressing Format

Figure 2-15. Type 5 Request Packet: Dev32 Transport: 50-bit Addressing Format

Notes: N is the number of double-words in the payload.
n = N-1

2.6.9 Type 5 Request Packet: Dev32 Transport: 66-bit Addressing Format

Figure 2-16. Type 5 Request Packet: Dev32 Transport: 66-bit Addressing Format

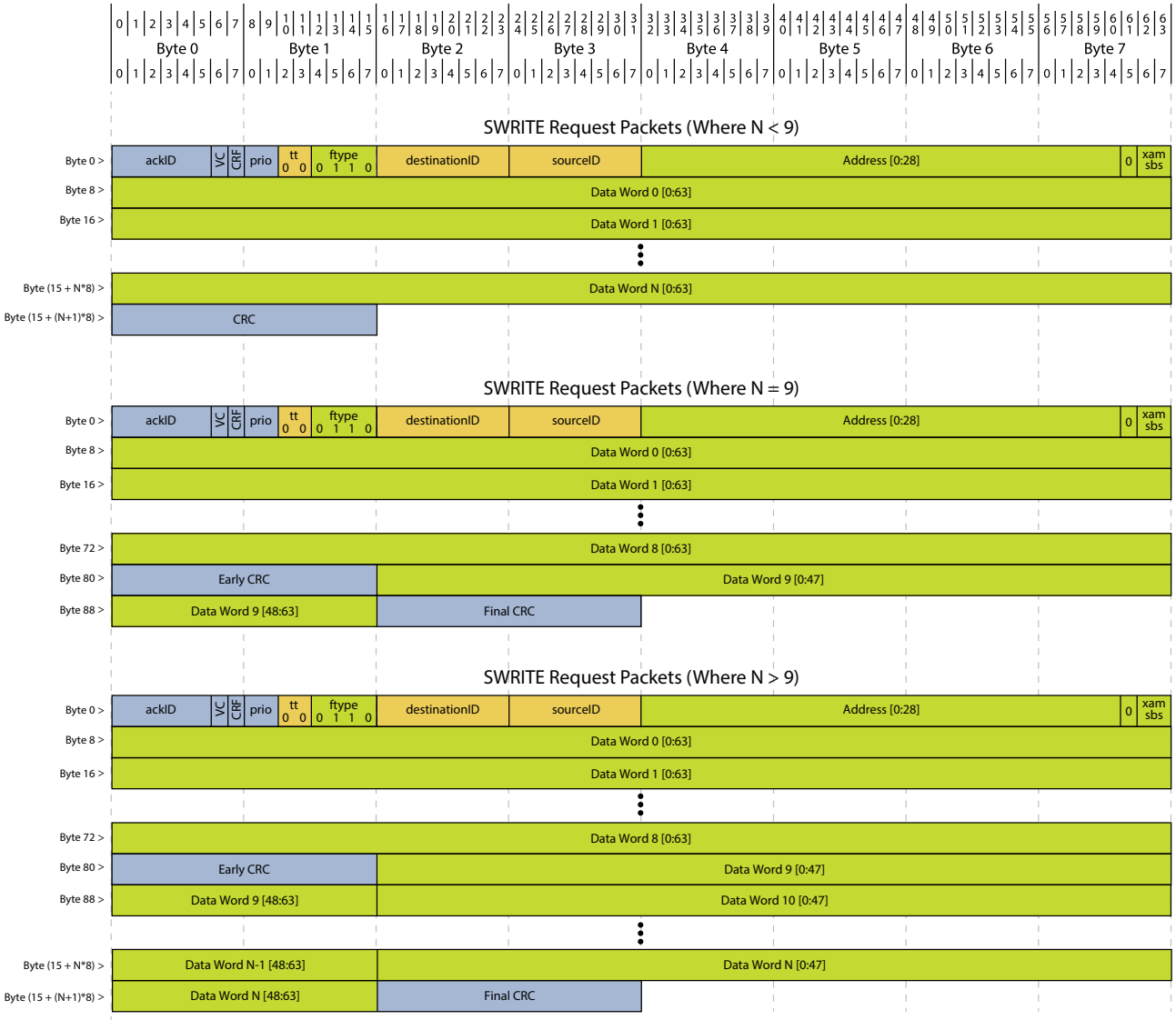
Notes: N is the number of double-words in the payload.
n = n-1

2.7 Type 6 Request Packets

The Type 6 request packet format is used for the Streaming Class. The Logical Layer fields for the Streaming Class are defined in Part 1.

2.7.1 Type 6 Request Packet: Dev8 Transport: 34-bit Addressing Format

Figure 2-17. Type 6 Request Packet: Dev8 Transport: 34-bit Addressing Format



2.7.2 Type 6 Request Packet: Dev8 Transport: 50-bit Addressing Format

Figure 2-18. Type 6 Request Packet: Dev8 Transport: 50-bit Addressing Format

2.7.3 Type 6 Request Packet: Dev8 Transport: 66-bit Addressing Format

Figure 2-19. Type 6 Request Packet: Dev8 Transport: 66-bit Addressing Format

2.7.4 Type 6 Request Packet: Dev16 Transport: 34-bit Addressing Format

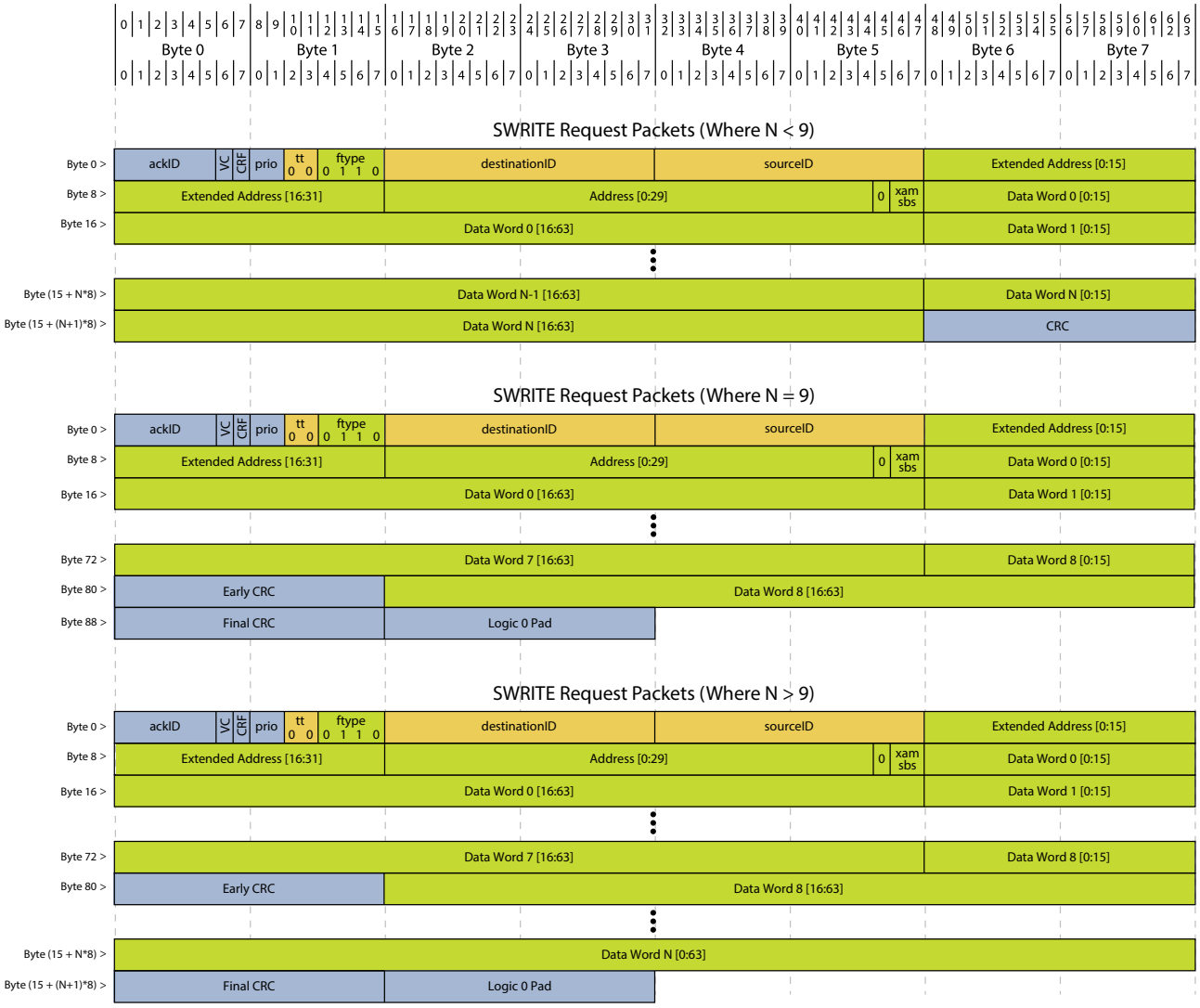
Figure 2-20. Type 6 Request Packet: Dev16 Transport: 34-bit Addressing Format

2.7.5 Type 6 Request Packet: Dev16 Transport: 50-bit Addressing Format

Figure 2-21. Type 6 Request Packet: Dev16 Transport: 50-bit Addressing Format

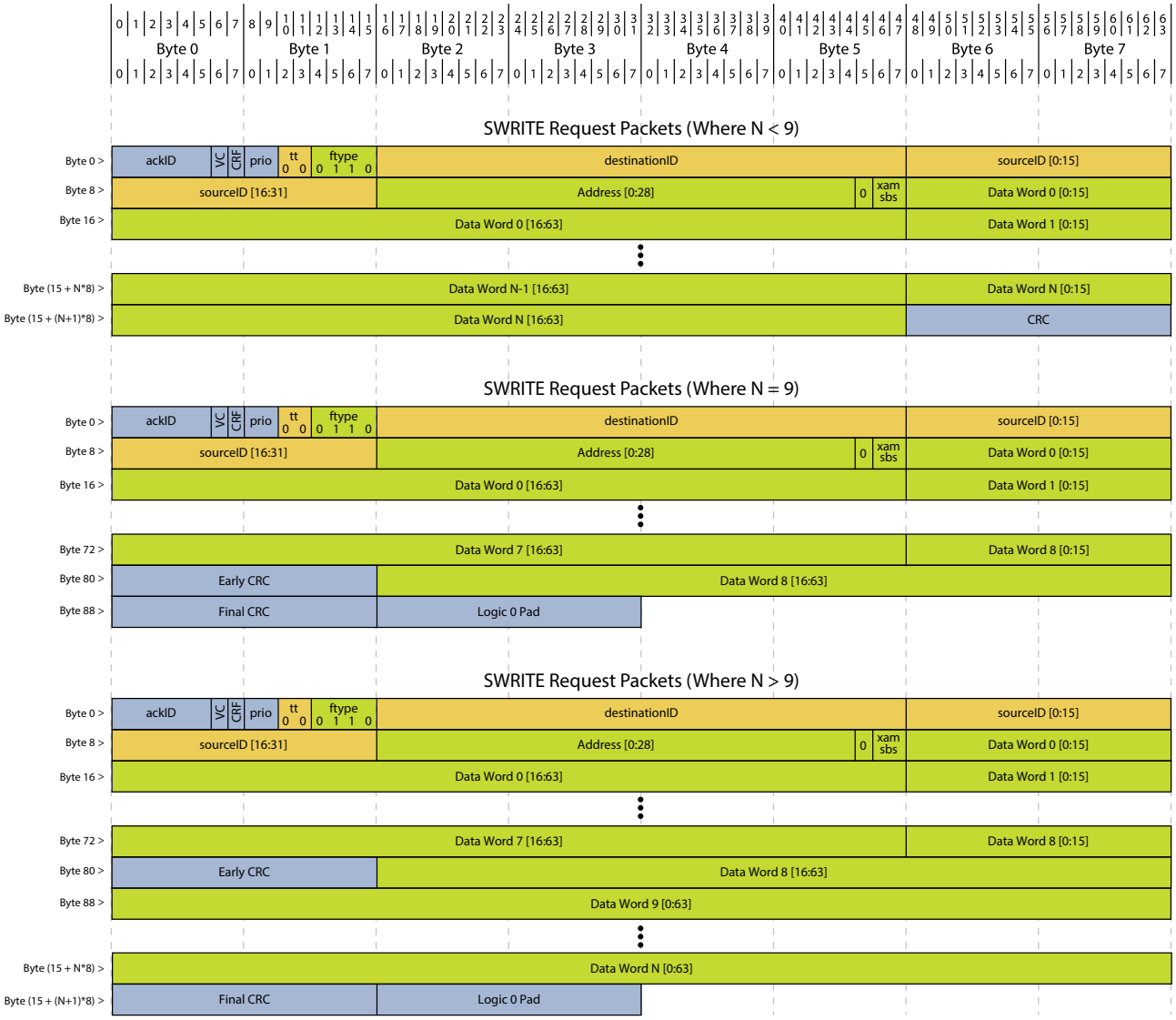
2.7.6 Type 6 Request Packet: Dev16 Transport: 66-bit Addressing Format

Figure 2-22. Type 6 Request Packet: Dev16 Transport: 66-bit Addressing Format



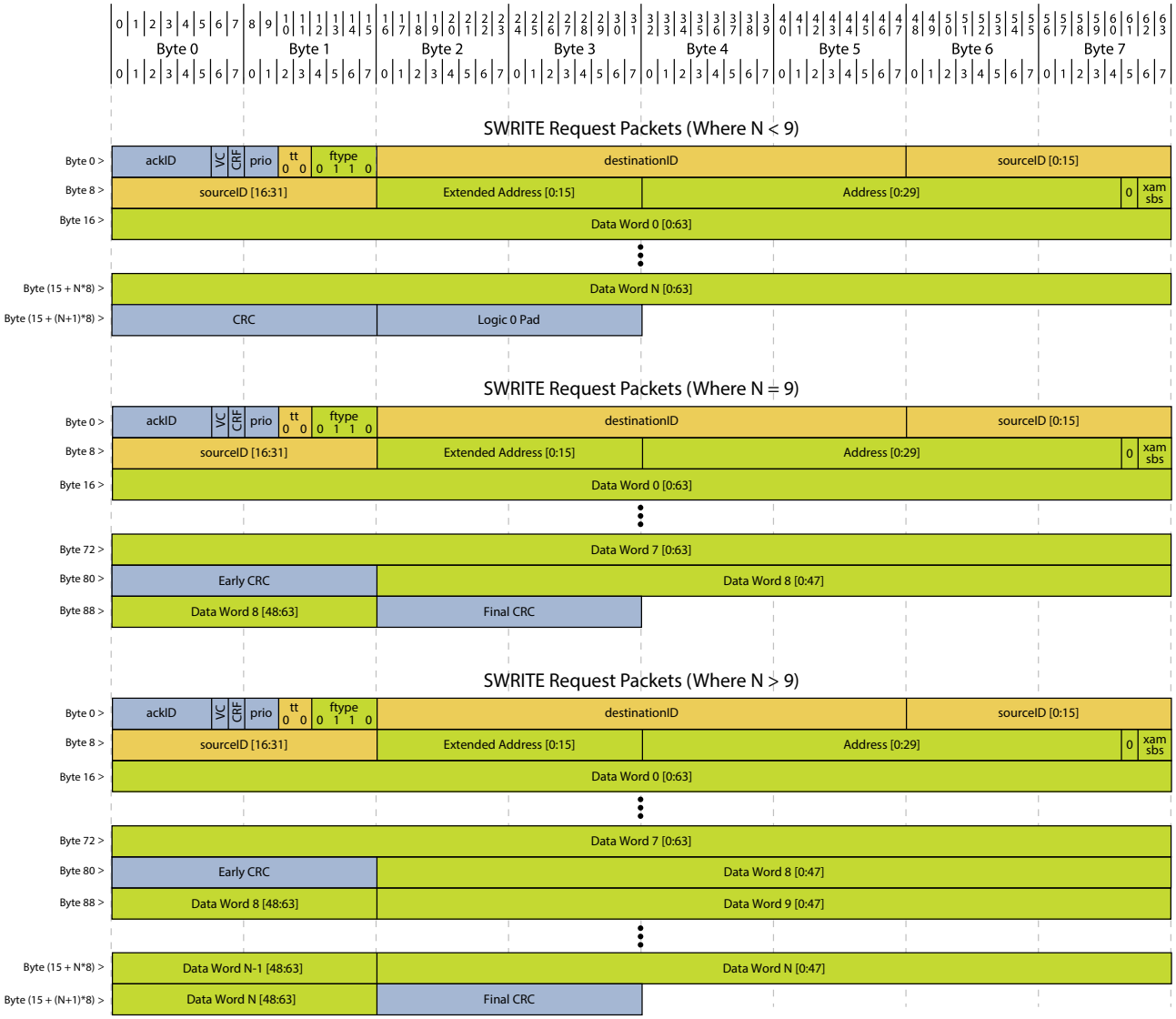
2.7.7 Type 6 Request Packet: Dev32 Transport: 34-bit Addressing Format

Figure 2-23. Type 6 Request Packet: Dev32 Transport: 34-bit Addressing Format



2.7.8 Type 6 Request Packet: Dev32 Transport: 50-bit Addressing Format

Figure 2-24. Type 6 Request Packet: Dev32 Transport: 50-bit Addressing Format



2.7.9 Type 6 Request Packet: Dev32 Transport: 66-bit Addressing Format

Figure 2-25. Type 6 Request Packet: Dev32 Transport: 66-bit Addressing Format

2.8 Type 7 Request Packets

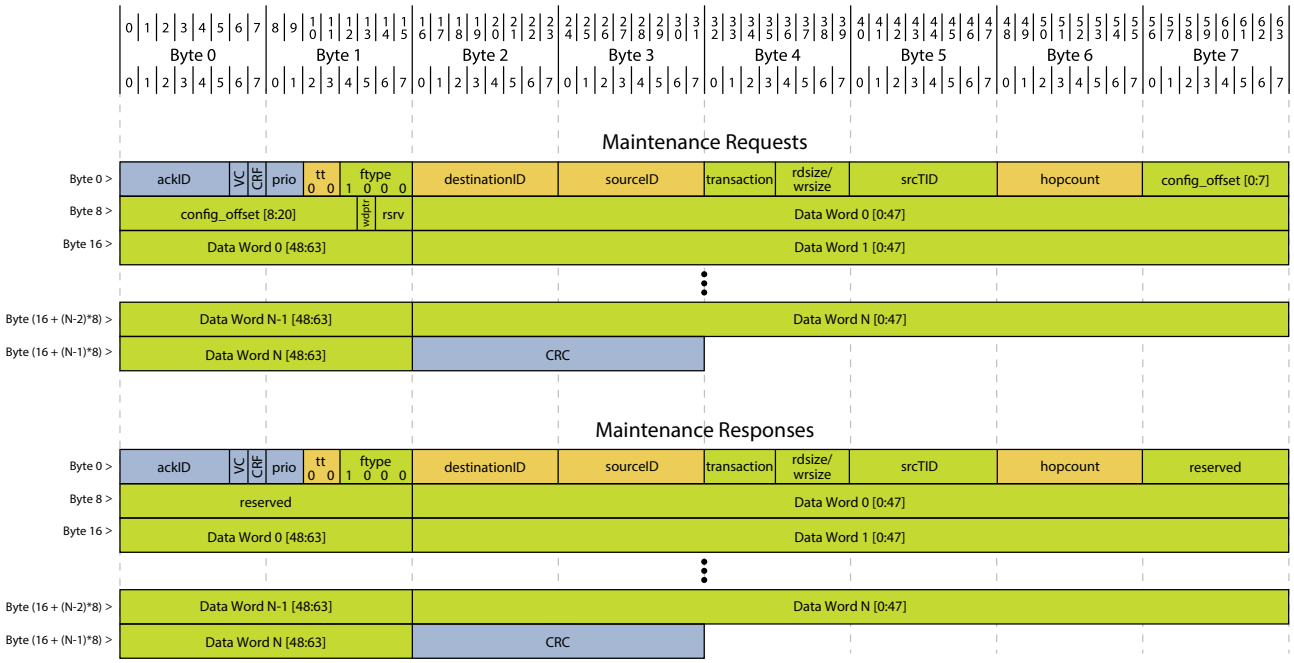
The Type 7 request packet format is used for the Flow Control Class. The Logical Layer fields for the Flow Control Class are defined in Part 9.

2.9 Type 8 Request and Response Packets

The Type 8 packet format is used by the Maintenance protocol for both request and response transactions. The Type 8 request and response transactions are described in Part 1: IO Logical Layer Section 4.1.10. Each Maintenance request type (Read or Write) is paired with a corresponding Maintenance response type (Read response or Write response) except the Port Write request transaction does not expect a response.

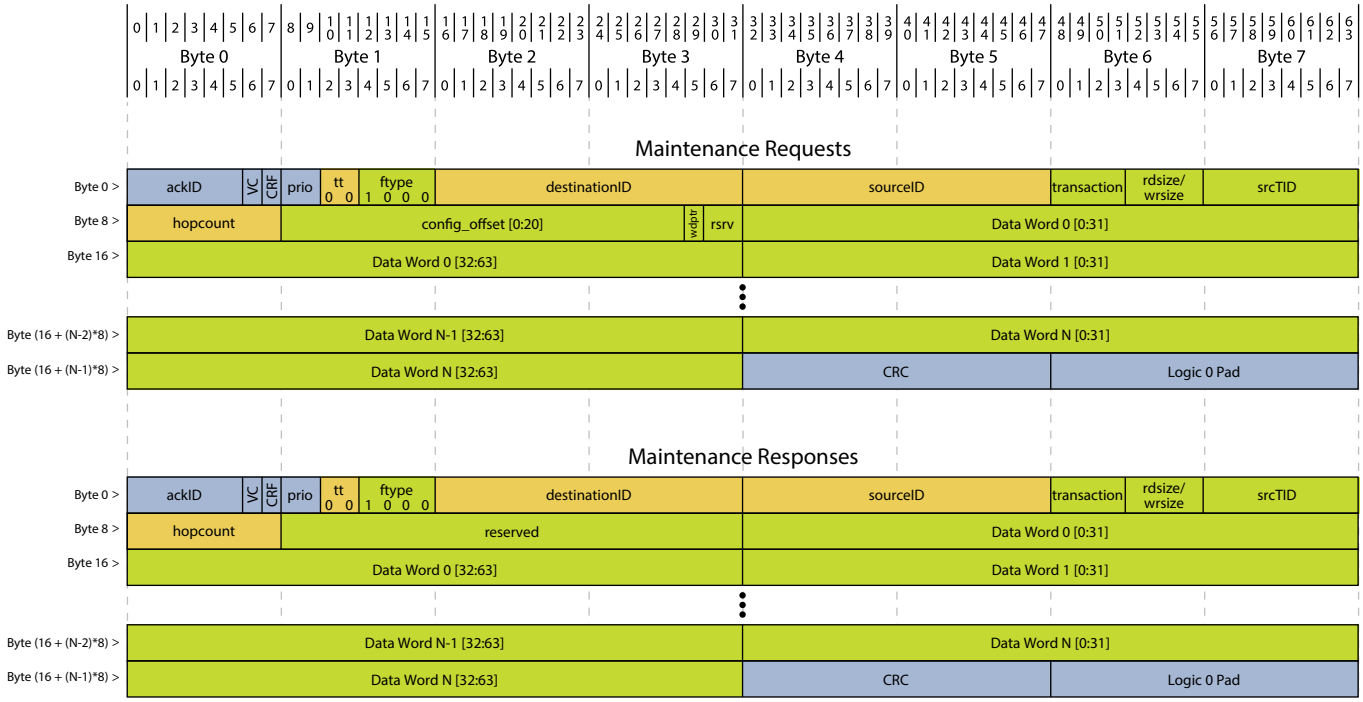
2.9.1 Type 8 Request and Response Packet: Dev8 Transport Format

Figure 2-26. Type 8 Request and Response Packet: Dev8 Transport Format



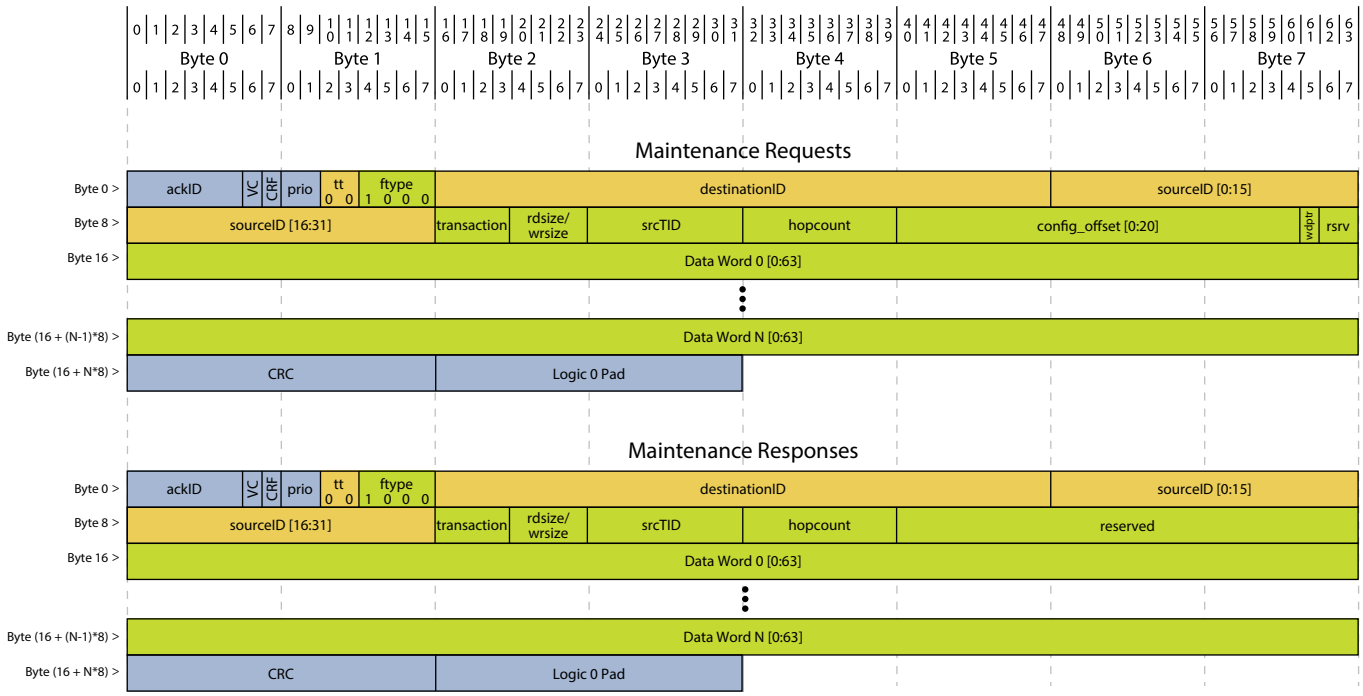
2.9.2 Type 8 Request and Response Packet: Dev16 Transport Format

Figure 2-27. Type 8 Request and Response Packet: Dev16 Transport Format



2.9.3 Type 8 Request and Response Packet: Dev32 Transport Format

Figure 2-28. Type 8 Request and Response Packet: Dev32 Transport Format

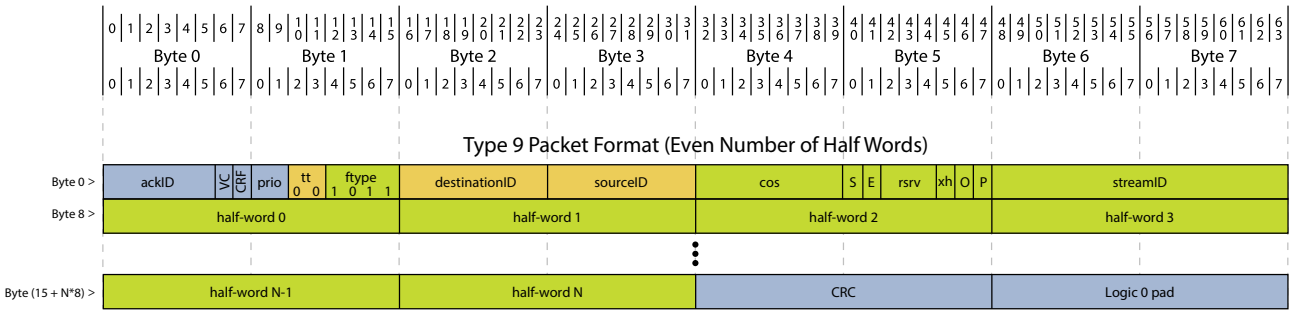


2.10 Type 9 Request Packets

The Type 9 request packet format is used for the Data-Streaming Class. The Logical Layer fields for the Data-Streaming Class are defined in Part 10.

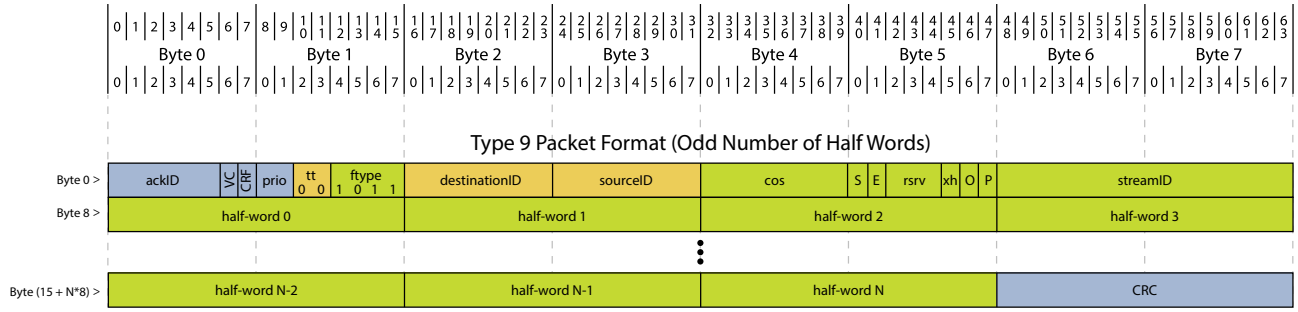
2.10.1 Type 9 Packet Format: Dev8 Even Number of Half Words

Figure 2-29. Type 9 Packet Format: Dev8 Even Number of Half Words



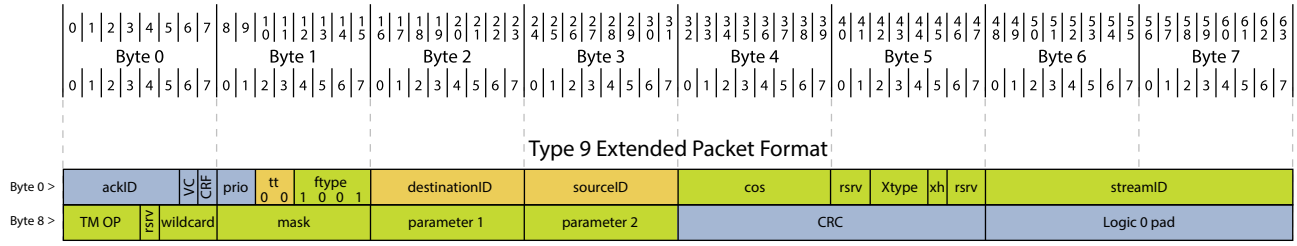
2.10.2 Type 9 Packet Format: Dev8 Odd Number of Half Words

Figure 2-30. Type 9 Packet Format: Dev8 Odd Number of Half Words



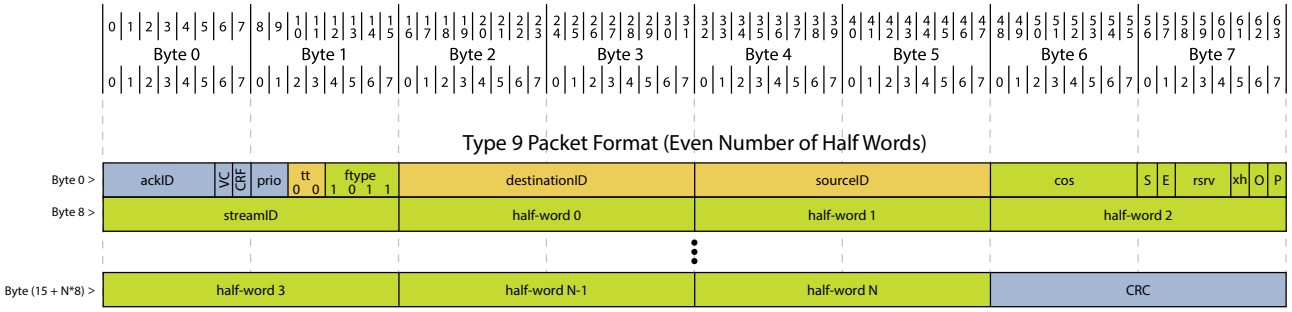
2.10.3 Type 9 Extended Packet Format Dev8

Figure 2-31. Type 9 Extended Packet Format Dev8



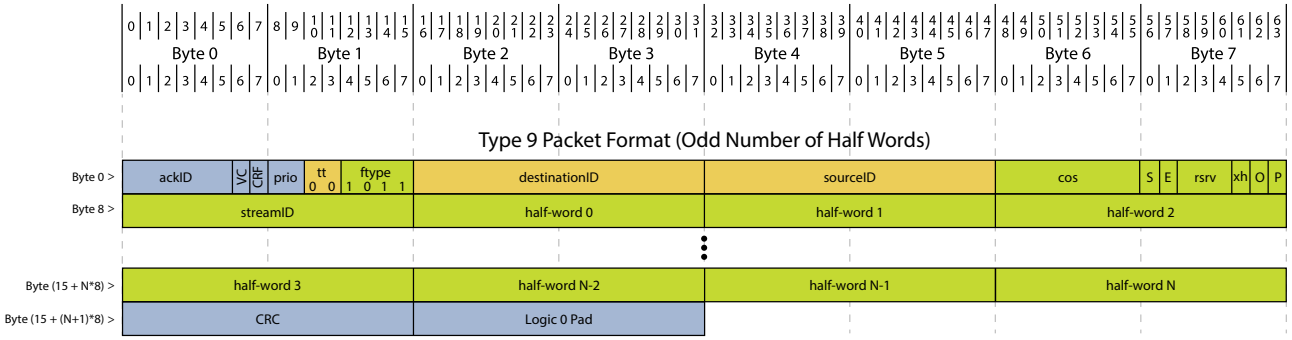
2.10.4 Type 9 Packet Format: Dev16 Even Number of Half Words

Figure 2-32. Type 9 Packet Format: Dev16 Even Number of Half Words



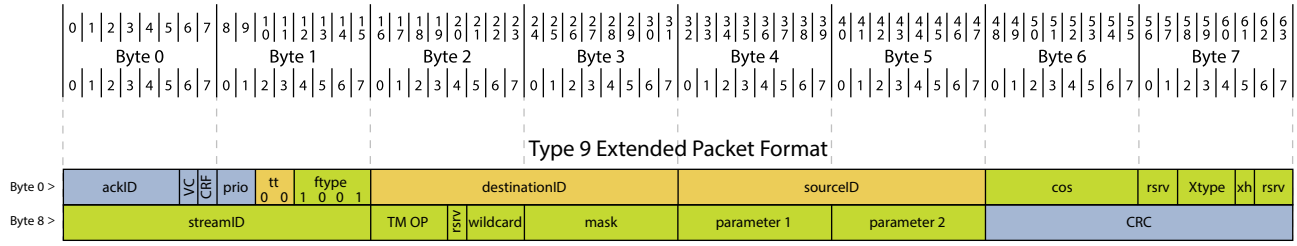
2.10.5 Type 9 Packet Format: Dev16 Odd Number of Half Words

Figure 2-33. Type 9 Packet Format: Dev16 Odd Number of Half Words



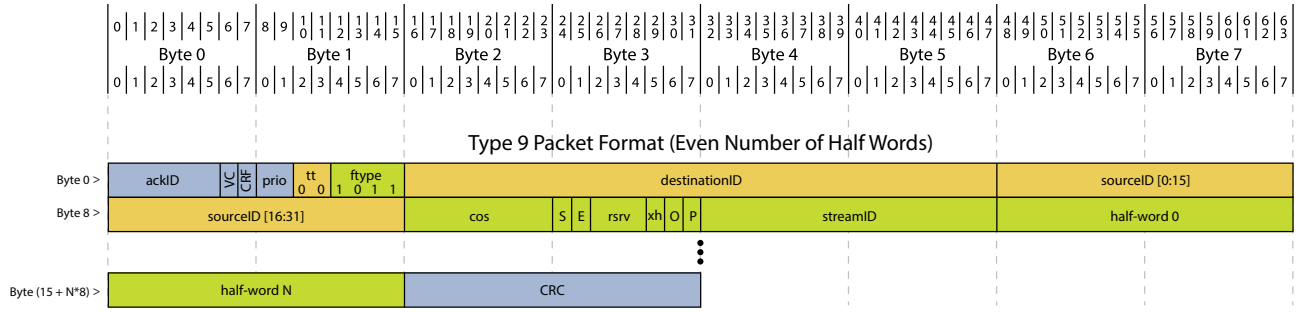
2.10.6 Type 9 Extended Packet Format Dev16

Figure 2-34. Type 9 Extended Packet Format Dev16



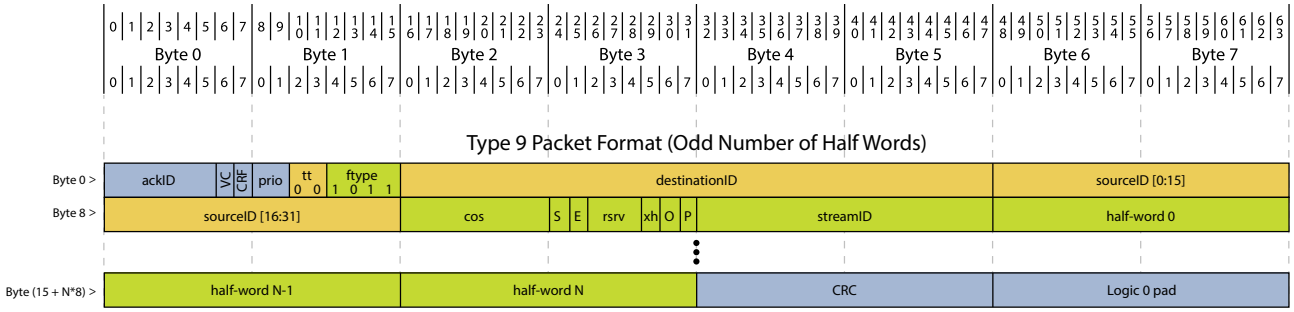
2.10.7 Type 9 Packet Format: Dev32 Even Number of Half Words

Figure 2-35. Type 9 Packet Format: Dev32 Even Number of Half Words



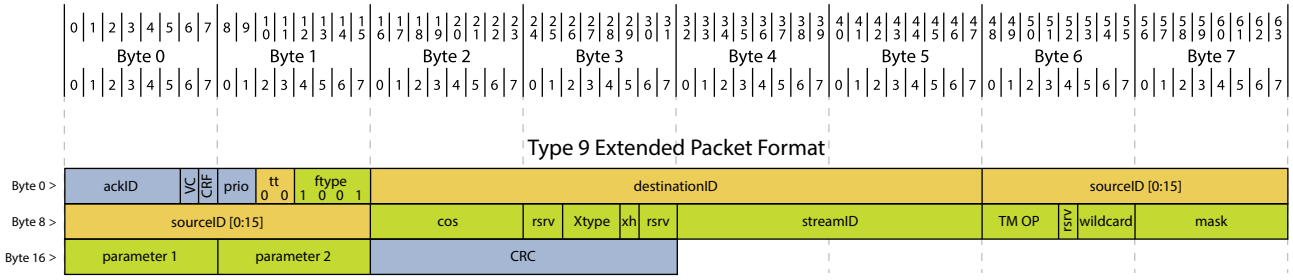
2.10.8 Type 9 Packet Format: Dev32 Odd Number of Half Words

Figure 2-36. Type 9 Packet Format: Dev32 Odd Number of Half Words



2.10.9 Type 9 Extended Packet Format Dev32

Figure 2-37. Type 9 Extended Packet Format Dev32



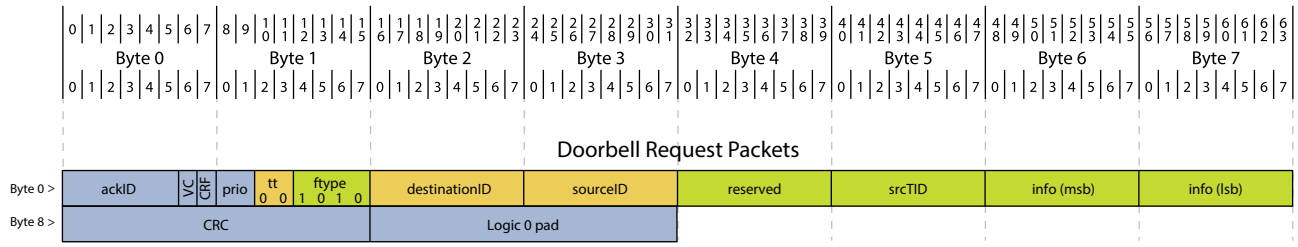
Notes: N is the number of double-words in the payload.
n = N-1

2.11 Type 10 Request Packets

The Type 10 request packet format is used for the Doorbell Class. The Logical Layer fields for the Doorbell Class are defined in Part 2.

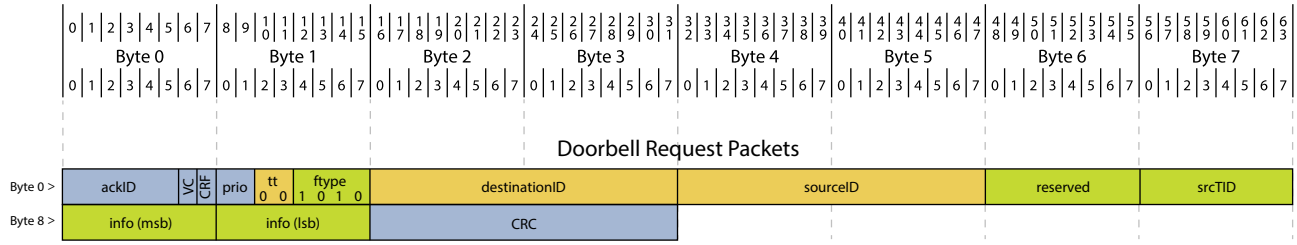
2.11.1 Type 10 Request Packet: Dev8 Addressing Format

Figure 2-38. Type 10 Request Packet: Dev8 Addressing Format



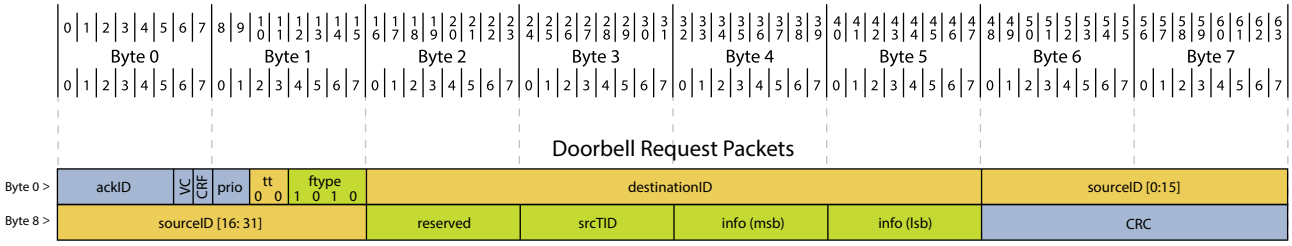
2.11.2 Type 10 Request Packet: Dev16 Addressing Format

Figure 2-39. Type 10 Request Packet: Dev16 Addressing Format



2.11.3 Type 10 Request Packet: Dev32 Addressing Format

Figure 2-40. Type 10 Request Packet: Dev32 Addressing Format



2.12 Type 11 Request Packets

The Type 11 request packet format is used for the Message Class. The Logical Layer fields for the Message Class are defined in Part 2..

2.13 Type 12 Response Packets

The Type 12 response packet format is reserved and not yet defined.

2.14 Type 13 Response Packets

The Type 13 Response Packet is used for specified Logical Layer protocols that define request transactions that expect a response transaction. There are two general versions of Type 13 Response Packets, distinguished by the ‘transaction’ header field, response packets containing data (transaction = 0b1000) and response packets without data (transaction = 0b0000). An additional response packet without data (transaction = 0b0001) is dedicated for responding to Type 11 Message request transactions.

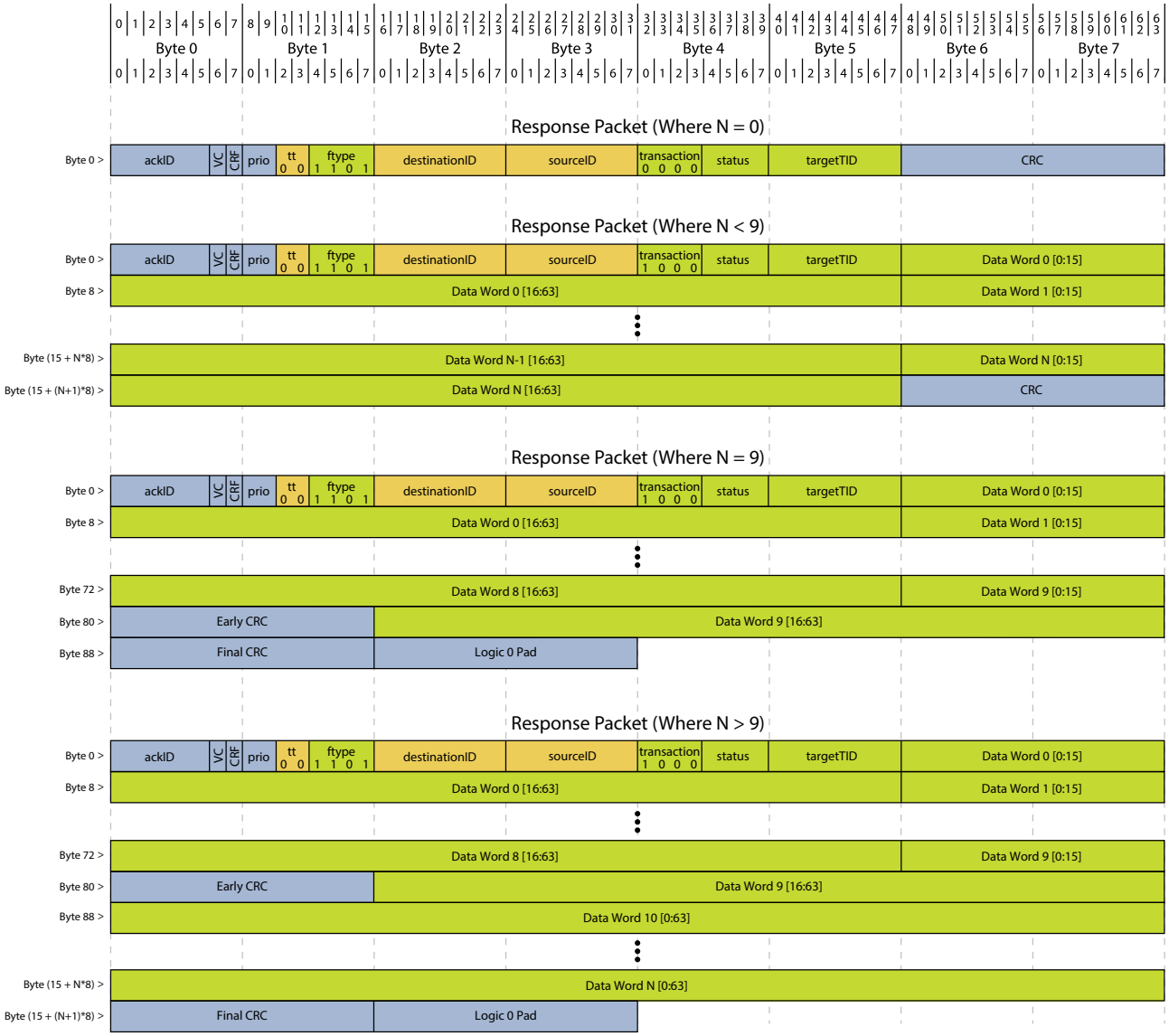
Because the transaction field value for response packets is common to multiple request packet types, implementations must track the packet Transaction ID field to correlate each Type 13 Response packet with the corresponding Type X (X = 1, 2, 5, 10) Request packet. Note that Transaction ID tracking cannot be used for message Type 13 responses (transaction = 0b0001) because message packet headers don’t include the Transaction ID field.

The Type 13 Response Packet is used to respond to the request packet types as defined by the following Sections of the specification.

1. The Type 1 request transactions as described in Part 5: Globally Shared Memory Logical Layer Section 4.2.
2. The Type 2 request transactions as described in Part 1: IO Logical Layer Section 4.2 and in Part 5: Globally Shared Memory Logical Layer Section 4.2.
3. The Type 5 request transactions as described in Part 1: IO Logical Layer Section 4.2 (the Type 5 NWRITE request transaction does not expect a response) and in Part 5: Globally Shared Memory Logical Layer Section 4.2.
4. The Type 10 Doorbell request transaction as described in Part 2: Message Passing Logical Layer Section 4.2.
5. The Type 11 Message Passing request transaction as described in Part 2: Message Passing Logical Layer Section 4.2. The Type 13 response has a dedicated transaction field value of 0b0001.

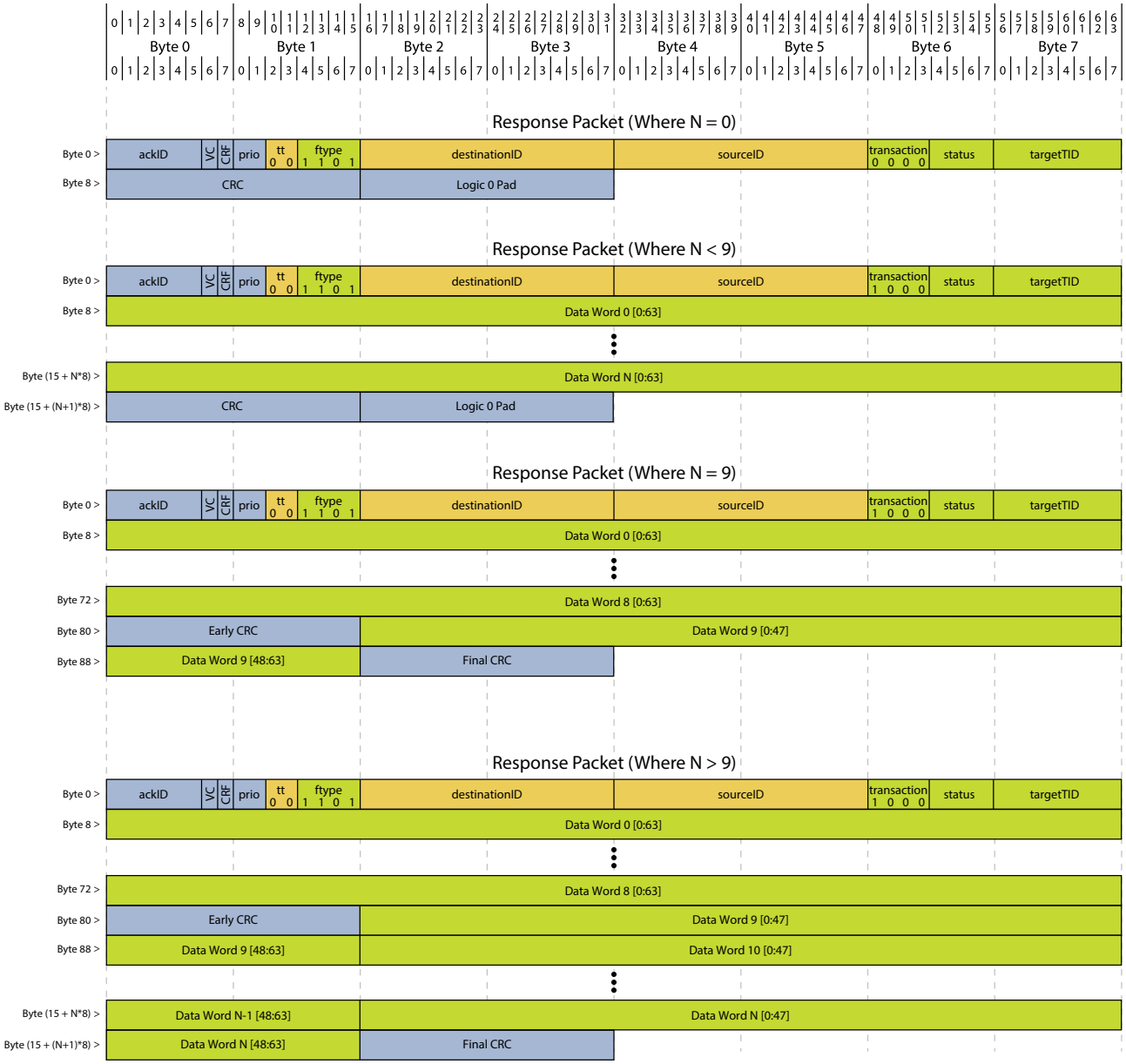
2.14.1 Type 13 Response Packet: Dev8 Transport Format

Figure 2-41. Type 13 Response Packet: Dev8 Transport Format



2.14.2 Type 13 Response Packet: Dev16 Addressing Format

Figure 2-42. Type 13 Response Packet: Dev16 Addressing Format



2.14.3 Type 13 Response Packet: Dev32 Addressing Format

Figure 2-43. Type 13 Response Packet: Dev32 Addressing Format

2.15 Type 14 Response Packets

The Type 14 response packet format is reserved and not yet defined.

2.16 Type 15 Response Packets

The Type 15 response packet format is reserved for implementation-defined functions. The Logical Layer fields are implementation defined.

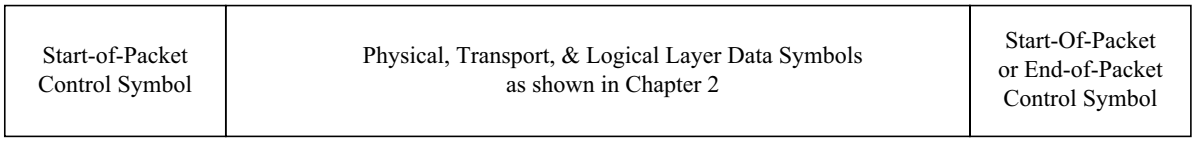
Chapter 3 Control Symbols

The Serial RapidIO protocol uses control symbols and data symbols to transfer information across the physical medium. Control symbols help keep a link alive reliably, demark the start and end of packets, represent time syncs, acknowledge packets, and are used to recover from a variety of asynchronous behaviors on the link. They are discussed in depth in the physical layer requirements in Part 6 of the standard. The Serial RapidIO control symbols related to packets are discussed in this section.

3.1 Complete Packets

Every complete packet must have a Start-of-Packet control symbol (CS), stype1 = 0, prepended to the data bytes defined in Chapter 2, and either a Start-of-Packet or End-of-Packet control symbol, stype1 = 2, appended to the end of the data bytes defined in Chapter 2. Thus, a complete packet that is fully delimited is depicted in Figure 3-1 Packet Delimiter Control Symbols. The full length is determined by the length of the control symbols added to the number of data bytes in the packet as defined in Chapter 2. A packet can be terminated by a Start-Of-Packet CS if it is immediately followed by the data associated with the next packet. Otherwise, the End-Of-Packet CS must immediately follow the packet data. RapidIO is a bidirectional protocol, and packets in one direction on a link result in acknowledgement control symbols in the opposite direction.

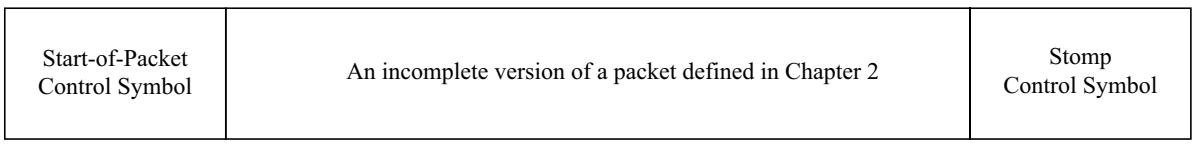
Figure 3-1. Packet Delimiter Control Symbols



3.2 Incomplete Packets

There are a variety of situations that can cause a sender to prematurely terminate a packet in progress. The Stomp control symbol, stype1 = 1, is used to prematurely end a packet.

Figure 3-2. Terminated Packet Delimiters



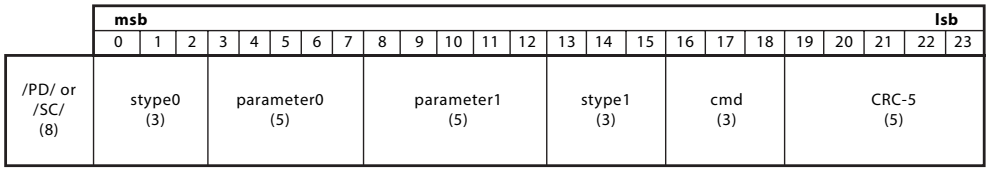
3.3 Other Interactions

The other control symbols defined in Part 6, Chapter 3, can be interleaved with the above sequences at any time. A full description of the semantics of control symbols can be found in Part 6, Chapter 3.

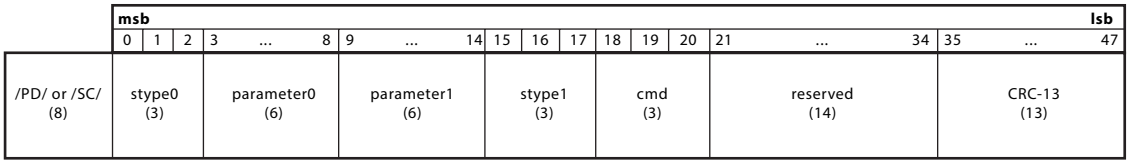
3.4 Control Symbol Formats

See Part 6, Table 3-2 and Table 3-3 for the list of the stype0 control symbol functions. See Part 6, Table 3-17 and Table 3-18 for the list of stype1 control symbol functions. Stype0 control symbols are associated with received data. Stype1 control symbols are either associated with transmitted data, are requests directly to the link partner, or are broadcast events. Stype0 and stype1 control symbol functions can be merged into the same control symbol.

3.4.1 24-bit Control Symbol



3.4.2 48-bit Control Symbol



3.4.3 64-bit Control Symbol