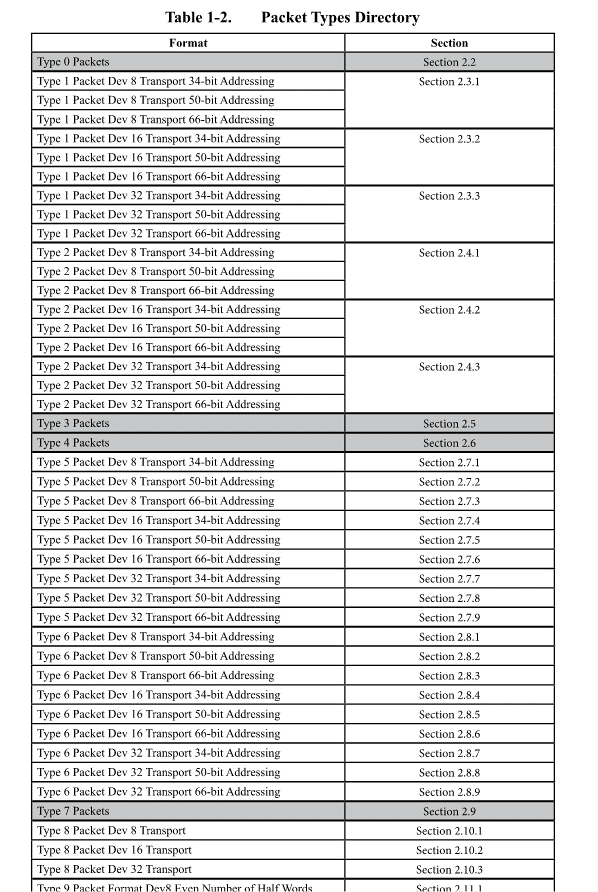
[ cover sheet, Table of Contents, etc.]

# Chapter 1 Introduction

## 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 1-2 ALLIE REPLACE CROSS REF (I only copied part of the table below, just to show which table I’m talking about) provides hyperlinks to facilitate finding the layout diagrams for a specific packet format.



## 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 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 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 1-1 shows a cross reference for each packet type and where in the standard each portion of the packet is defined.

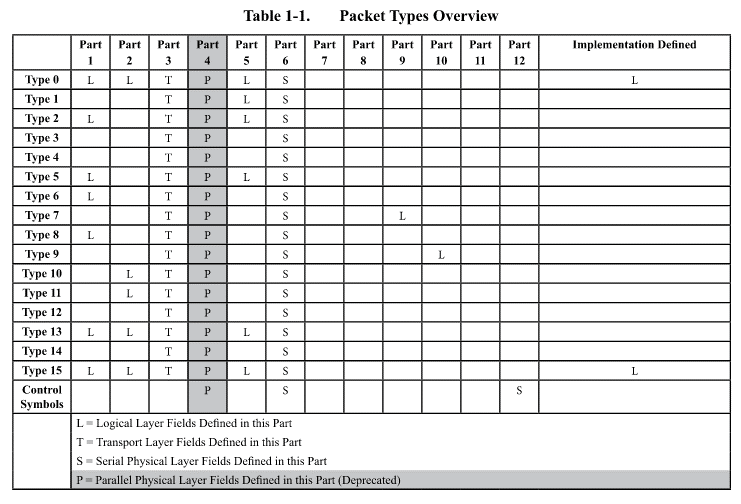
The RapidIO protocol uses data symbols and control symbols to transfer packets across a link. The protocol also uses control symbols to demark packets, acknowledge packets, and perform the various fault, link, and time management functions defined in the standard. Chapter 3 provides an overview of the 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.

ALLIE, PUT THESE TABLES WHEREVER YOU SEE FIT. I INCLUDE THEM HERE SO YOU KNOW WHICH ONES I’m referencing above.

| **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 |
| 3-15 | 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 |  |
| 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 RapidIO Format and Transaction Types



### 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 3 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.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **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** | | |
| Requests | **Ftype** | **34** | **50** | **66** | **34** | **50** | **66** | **34** | **50** | **66** |
| 0 | DBD[[1]](#footnote-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 | | |

Table 3 – Header Size by Format Type

### 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.

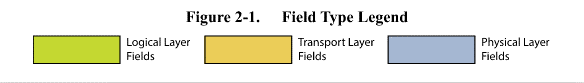
If only a single size is listed for a particular Device Id width, then there is no address embedded in the packet header.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **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** | | |
| Requests | **Ftype** | **34** | **50** | **66** | **34** | **50** | **66** | **34** | **50** | **66** |
| 0 | DBD[[2]](#footnote-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]](#footnote-3) | 272[[4]](#footnote-4) | 272 | 276 | 272 | 276 | 276 | 276 | 280 | 280 |
| 6 | 268 | 272 | 272 | 272 | 272 | 276 | 276 | 276 | 280 |
| 7[[5]](#footnote-5) | 8 | | | 12 | | | 16 | | |
| 8[[6]](#footnote-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 | | |

Table 4 – Max Packet Sizes by Format Type

# 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.



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.

## Type 0 Request Packets

**ALLIE, THE REST OF YOUR CHAPTER 2 HERE**

**NEW CHAPTER 3**

# 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.

## 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 1‑1 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.

|  |  |  |
| --- | --- | --- |
| Start-of-Packet Control Symbol | Physical, Transport, & Logical Layer Data Symbols as shown in Chapter 2 | Start-Of-Packet or End-of-Packet Control Symbol |

Figure 1‑1 Packet Delimiter Control Symbols

## 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.

|  |  |  |
| --- | --- | --- |
| Start-of-Packet Control Symbol | An incomplete version of a packet defined in Chapter 2 | Stomp Control Symbol |

Figure 1‑2 Terminated Packet Delimiters

## 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.

## 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.

### 24-bit Control Symbol

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **MSB** | |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **LSB** | |
|  | 0 | 1 | | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| /PD/ or /SC/ (8) | stype0 (3) | | | | parameter0 (5) | | | | | parameter1 (5) | | | | | stype1 (3) | | | cmd (3) | | | CRC-5 (5) | | | | | |
|

### 48-bit Control Symbol

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **MSB** | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **LSB** | |
|  | 0 | 1 | 2 | 3 | … | | | 8 | 9 | … | | | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | … | | | | | | 34 | 35 | … | | | | | 47 |
| /PD/ or /SC/ (8) | stype0 (3) | | | parameter0 (6) | | | | | parameter1 (6) | | | | | stype1 (3) | | | cmd (3) | | | reserved (14) | | | | | | | | CRC-13 (13) | | | | | | |
|

### 64-bit Control Symbol

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **MSB** | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **LSB** | |
|  | 0 | … | 3 | 4 | … | | | 15 | 16 | … | | | 27 | 28 | 29 | 30 | 31 | 32 | … | 37 | 38 | … | | | | | | | | | | | 62 | 63 |
| /PD/ or /SC/ (8) | stype0 (4) | | | parameter0 (12) | | | | | parameter1 (12) | | | | | stype1[0:1] (2) | | alignment (2) | | stype[2:7] (6) | | | CRC-24 (24) | | | | | | | | | | | | | alignment |

1. DBD = Packet format is “Defined by Device”. The packet may or may not have an address. [↑](#footnote-ref-1)
2. DBD = Packet format is “Defined by Device”. The packet may or may not have an address. [↑](#footnote-ref-2)
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. [↑](#footnote-ref-3)
4. The maximum payload size is 256 bytes for packet Format Types 5, 6, 11, 13, and 9 (except Traffic Management). [↑](#footnote-ref-4)
5. There are no addresses in packet Format Types 7, 9, 10, 11, and 13. [↑](#footnote-ref-5)
6. Max sized payload for Type 8 transactions is 64 bytes. [↑](#footnote-ref-6)