Hotline™ Command-Response Transaction (HCrt) Specification

HCrt Features

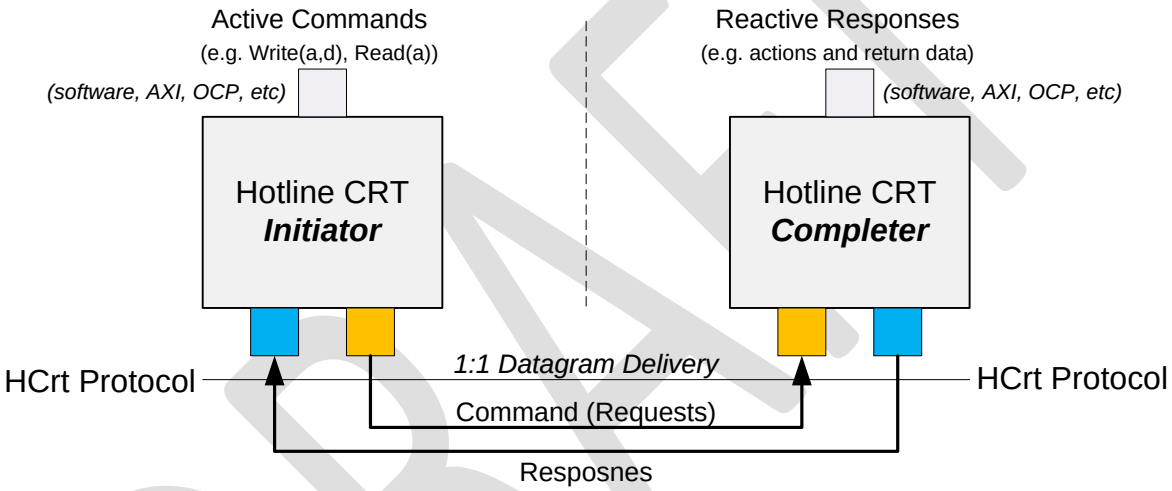
- A datagram-based protocol between one Initiator and one Completer
- Efficient Encapsulation of 32- and 64-bit Load/Store (Read/Write) Transactions
    - Commands are programmable-length, address-sequential read or write bursts
    - Multiple Commands may be embedded in a single Message (Packet or Frame)
- Initiator and Completer each may be implemented in either software or hardware
- Use-Cases Include
    - Software Initiator issuing commands to Software Completer
    - Software Initiator issuing command to Hardware Completer
    - Hardware Initiator issuing commands to Software Completer
    - Hardware Initiator issuing commands to Hardware Completer
- Protocol tolerates an underlying undelivered and/or out-of-order datagram delivery service
    - Datagrams may arrive in any order, or not at all; but must not be corrupt
- Common underlying datagram interchange between Initiator and Completer Include
    - Ethernet Layer 2 PDU Datagrams
    - IP / UDP Datagram Layer 3
    - Wire, full- or half-duplex, request-response channels (e.g. FIFOs)
    - Software Pipes, such as functional simulator request/response channels
- Facilitates communications between Initiators and Completers that are not co-located
- Facilitates interchanging different Initiators and/or Completers
    - Linear Effort, Multiplicative Value – due to common, reused HCrt Protocol
    - Write one Initiator…
        - Talk to software Completer
        - Talk to functional simulator Completer
        - Talk to Hardware (FPGA) Completer X by means Y
        - Talk to Hardware (FPGA) Completer X' by means of Y', etc.
        - All with no change to software calls to Initiator
    - Write one Completer…
        - Ibid (dual of above) Talk to X Initiator
- HCrt Bridge Components allow the physical separation of AXI/OCP Master and Slave devices
    - Logical extension of AXI/OCP infrastructures beyond one physical device
    - HCrt is AXI/OCP fabric-agnostic, providing transparent bridged inter-conversion

Background

This document is the specification of the Hotline Command-Response Transaction (HCrt, or just CRT) protocol. The CRT protocol provides the ability for a CRT *Initiator* to command operations upon and receive responses from CRT *Completer*.

A CRT Initiator generates requests for a CRT Completer. A CRT Transaction is performed as follows: A CRT Initiator generates a CRT Command Message; in response to the received request, a CRT Completer generates a CRT Response Message. The command and response taken together comprise a transaction.

The diagram below shows the one-to-one correspondence between a CRT initiator and completer when both are acting to bridge to or from an addressable, upper-layer, transaction-level protocol or fabric.



An underlying transmission layer is responsible for providing a one-to-one association between a specific initiator and a specific completer. CRT Initiators may issue three kinds of Command Messages: Write Requests, which move write data from Initiator to Completer. Read Requests, which request that the Completer deliver read data. And NOP Requests, which assist in device discovery. Similarly, CRT Completers deliver Response Messages, of which there are three kinds: Write, Read, and NOP.

The CRT protocol is designed to operate reliably over a transmission layer which may have unreliable service such that the underlying transmission layer may not guarantee delivery of or order of delivery of complete messages. CRT requires that the transmission layer deliver a message correctly when it does deliver it. Anticipated use-cases for linking CRT Initiators and Completers (the Transmission Layer) include software, hardware direct-connect, Ethernet Layer 2, UDP datagrams, FIFO pipelines of bit widths 2^m, and other transports lacking a link layer, as CRT does not require one.

CRT Messages are comprised of one or more commands or responses. The protocol specifies that all commands or responses within a message must be of the same kind. For example, a message may have more than one write request, but must not contain a write request and a read request.[1]

Each transaction begins with a command from the Initiator to the Completer. For example, a request write command would have command arguments in the request command that include a DWORD (DW) for the starting write address and additional DWs of write data. A field in the command header, Argument DW Length (ADL), provides indication as to how many DWs the request applies to. In all cases a First DW Byte Enable indicates byte-enables for the first DW. In cases where (ADL > 1), a Last DW Byte Enable indicates the byte-enables for the last DW. For ADL >2, all intermediate DWs are fully enabled.

The CRT Initiator generates an incrementing unsigned 4b rolling ID “tag” for each transaction. The lifetime of a transaction tag begins with Initiator, travels through to the Completer on the command request message, and is returned to the Initiator in the response message. The purpose of the tag is to identify which response belongs to which request when more than one are simultaneously pipelined in flight between a paired initiator and completer. The scope of the 4b tag is limited to a specific CRT pair.

The CRT protocol contains discovery features to allow a single initiator to discover and converse with a single completer. Following discovery, an initiator and completer may optionally decide to “pair”, and carry on a tag-based sequence of command-response transactions between themselves. This is trivial in the case where the two are connected uniquely by wire; but more complex where the datagram service underneath CRT is used to establish this pairing. The CRT protocol contains no steering information to allow simultaneous outstanding communication in a many to many mode. If this is necessary, other protocol layers must be involved to acquire and release an exclusive lock on the channel.

[1] The “homogeneous command constraint” is a normative part of this specification. The rationale is that it may provide for some optimization of transaction processing in certain implementations, in exchange for the modest restriction in functionality. It is conceivable that this constraint could be lifted in a future revision without substantial changes to the protocol.

Terms and Definitions

The following terms are defined for use in the context of this CRT specification

- DWORD (DW): A 32b, little-endian storage unit composed of four Bytes
  - Little-Endian Rules apply throughout this specification

- Command: The building block of Request Messages, made of one or more DWORDs
  - A single DW at the head of each command, the Command-Response Header (CRH) describes the command, its length in DW, and if it is the last command in a message
  - A variable amount of data as described by the command's CRH
  - One or more commands may be grouped into a command message

- Command Message: One or more Commands (see Request Message)

- Response: The building block of Response Messages, made of one or more DWORDs
  - A single DW at the head of each response, the Command-Response Header (CRH) describes the response, its length in DW, and if it is the last response in a message
  - A variable amount of data as described by the responses' CRH
  - One or more responses may be grouped into a response message

- Initiator: The source of Request Messages to the Completer

- Completer: The source of Response Messages to the Initiator

- Request Message: A message from Initiator to Completer
  - One or more homogenous request commands

- Response Message: A message from Completer to Initiator
  - The response to a request message
  - Contains all responses to a command message up to the last one or first error

- Transaction: The combination of the request and response messages taken together
  - A uniquely tagged combination of a request message and its response message

- Transmission Layer: The layer below this protocol which provides datagram service

Command-Response Header (CRH)

This table describes the four Bytes in the Command-Response Header (CRH). The CRH precedes every command or response within a message. Byte 0 is the first byte on the wire on Byte-serialized transports. Bit 0 is the first bit on the wire on bit-serial transports.

CRH Byte	Bit (B)	Bit (DW)	Description
0	[3:0]	[3:0]	CRT Message Tag (Generated by requester; returned in response by completer) The tag shall be identical in all message headers within the same message. The tag shall be a rolling 4b unsigned count.
	[5:4]	[5:4]	CRT Message Type **NOP:** 0x0 = Command Request No Operation (Command) **Write:** 0x1 = Command Request Write Operation (Command) **Read:** 0x2 = Command Request Read Operation (Command) **Response:** 0x3 = Response Completion Operation (Response)
	[6]	[6]	64b Addressing Mode (AM64): '0'=32b (1DW Address); '1'=64b (2DW Address) Transaction
	[7]	[7]	Discovery Operation (DO): '1'=DO; '0'=Normal (not DO)
1	[3:0]	[11:8]	On Request: First DWORD Byte Enables 1=Enable 0=Disable 0xF=4B; 0xC=2B(MS); 0x3=2B(LS); 0x8=1B(B3); 0x4=1B(B2); 0x2=1B(B1); 0x1=1B(B0) On Response: Response Code 0x0=OK; 0x1=Timeout; 0x2=Error; 0x3-0xF=Reserved
	[7:4]	[15:12]	On Request: Last DWORD Byte Enables 1=Enable 0=Disable 0xF=4B; 0xC=2B(MS); 0x3=2B(LS); 0x8=1B(B3); 0x4=1B(B2); 0x2=1B(B1); 0x1=1B(B0) On Response: Must be 0
2	[7:0]	[23:16]	Arg DWORD Length (ADL) (excludes CRH and Addr) 12b Unsigned Bits [7:0]
3	[3:0]	[27:24]	Arg DWORD Length (ADL) (excludes CRH and Addr) 12b Unsigned Bits [11:8]
	[6:4]	[30:28]	Reserved – Must be 000
	[7]	[31]	LAST Indication: '1' indicates LAST command in message; '0' otherwise

Requirement: Byte 0 of the CRH shall be identical for all commands within a message, request or response. The Tag and DO bits shall be identical for all commands and responses within a transaction.

The figure below shows the information from the table above in graphical, little-endian form:

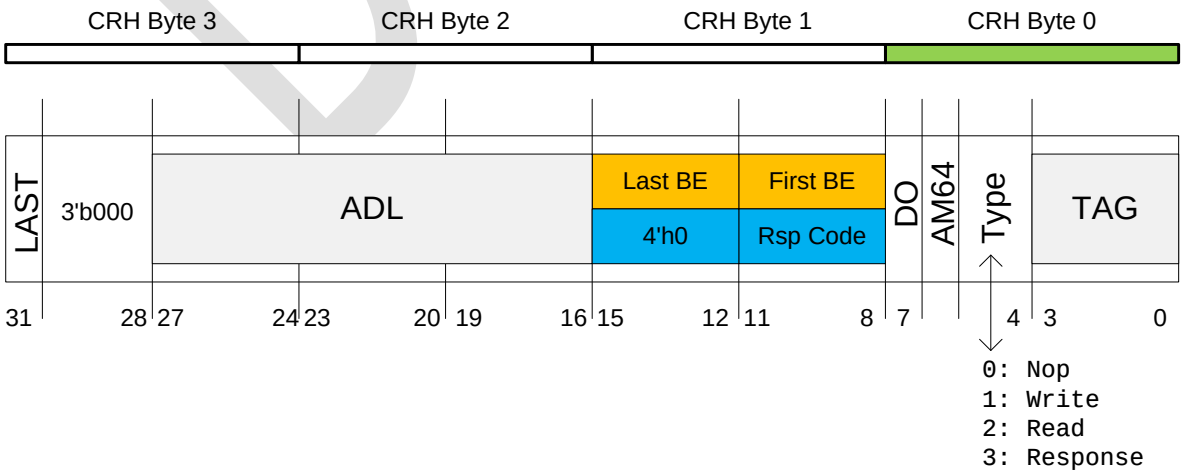


Table of CRH Command-Response Types

The table below lists six types of CRT commands and responses

CRT Command Type	# of DWORDs Follow CRH	Description (what data follows CRT CRH in each command)
Cmd:NOP	ADL	Command No Operation (+ADL Initiator Advertisement)
Cmd:Write	{1|2} + ADL	Command Write (+{1|2} DW Write Address, +ADL DW Write Data)
Cmd:Read	{1|2}	Command Read (+{1|2} DW Read Address)
Resp:NOP	ADL	Response NOP Request (+ADL Completer Advertisement)
Resp:Write	0	Response Write Request
Resp:Read	ADL	Response Read Request (+ADL DW Read Response Data)

Multiple commands-responses may be conveyed in a message so long as CRH byte 0 remains constant.

NOP Commands and Advertisements

The NOP command allows an initiator to both send and request information to the completer. It may be thought of as a command to share information not accessible by command write and reads. The NOP command provides utility in several ways:

- The successful interchange of a NOP command request-response is evidence of communication between a CRT initiator and completer. This is analogous to a “ping” or “echo” command.
- The NOP command does not require any interaction with an upper-layer entity. For example, a NOP completion can be returned even if an attached fabric bridge is non-functional.
- The payload contained in the NOP command and response for ADL>0 is used to exchange capability data between a CRT initiator and completer without any involvement of an upper-layer entity.

The initiator adds to the NOP command (ADL) DW of Initiator Advertisement which describes to the completer the initiator’s capabilities. In response, the completer acknowledges the NOP with a response containing (ADL) DW of Completer Advertisement.

The next two tables describe the capability information sent from initiator to completer and completer to initiator respectively within 1 DW. Future revisions of the specification may extend these tables

Initiator Advertisement

DW	Bits	Description
0	[31:0]	Size of Initiator’s Response Buffer in Bytes, 32b Unsigned Integer
1	[31:0]	

147 **Completer Advertisement**

DW	DW Bits	Description
0	[31:0]	Size of Completer's Response Buffer in Bytes, 32b Unsigned Integer
1	[31:0]	

148

149 **Argument DWORD Length (ADL)**

150 The 12b ADL field in the header is used to describe how many DWORDs, in addition to the CRH and any
151 necessary addressing, the command applies to. For NOP commands and response, the ADL indicates the
152 size of the advertisements. For Writes, the write command ADL says how many DWORDs follow the
153 address; the write response ADL is always zero. Write responses for each write command are just a
154 single CRH response DWORD. For Reads, the read command ADL says how many DWORDs are being
155 read. In the read command, the read command's CRH ADL says the read length in DWORDs, yet no data
156 follows the read address. Read responses contain the length of the commanded read in the response
157 message's CRH ADL.

158

159 **32 and 64 bit Addressing (AM64)**

160 By setting or clearing the AM64 bit in the CRH Byte 0, the CRT Initiator can indicate either a 32 or 64 bit
161 addressing mode for Write and Read command requests. When this bit is clear, 32b Addressing is used
162 and exactly one 32b DWORD follows after the CRH for addressing. When this bit is set, 64b Addressing is
163 used and exactly two 32b DWORDs, LS first, follow the CRH for addressing. Completers must examine
164 this bit to understand how to decode the request. Like all other bits in CRH Byte 0, this but must be
165 constant across all commands in a message.

166

167 **Discovery Operation (DO)**

168 An initiator can set the "Discovery Operation" (DO) bit to discover properties and capabilities of a
169 completer device without disrupting a conversation that may be underway with another initiator. To
170 achieve this goal, requests received with the "Discovery Operation" bit set to one will have the following
171 behaviors:

172 i) The CRT command tag match will be ignored and the request processed in any event.
173 ii) The state of the response replay buffer (a.k.a. "cache") will be unaffected by the operation.
174 iii) The completer's response to the Discovery Operation (DO) will also have the DO bit set.

175 Discovery may be used a precursor to "pairing", ensuring that a single conversion exists between an
176 Initiator and Target. The DO bit may also be used, typically during debug, to "force" a command to take
177 place without regard for the current tag state.

First and Last Byte Enable (BE) Rules

The Byte Enables provide a means for providing byte-level resolution for accesses at the start and end of a DWORD sequence. The CRT BE rules are *identical* to that of the "*PCIExpress Specification Revision 3.0*" BE rules. Our rationale is to avoid confusion by piggybacking upon a ubiquitous spec with a similar use-case and avoid confusion. These Byte Enable rules include, but are not limited to:

- Byte enable bits are high true. "0" indicates the corresponding byte should not be written.
- If the valid data transferred is all within a single DWORD, the Last BE field must be 4'h0

Response Success Aggregation

Each command CRH in a request message is replied to with a response CRH in the response message. For example, if a command request has *one* command accessing any number of DWORDs, there will be a same-tagged response with *one* response CRH which will indicate OK if all the access succeeded and the first error status otherwise. If a command request had *N* commands each accessing any number of DWORDs, there would be a same-tagged response with *N* response CRHs, each CRH showing the status of its respective command.

Overall Responsibilities:

The first byte of the each command-response header (CRH) shall be constant over all headers in a message.

Each message shall have exactly one command marked as the "LAST"; and that command or response shall be the last in a sequence of more than one commands or responses within a message.

Initiator Responsibilities:

The initiator is responsible for transmitting commands and receiving responses.

When the initiator initiates a NOP command, it must correctly populate the 4B Initiator Advertisement field with its capabilities.

Completer Responsibilities:

The completer shall store the tag from the last command request and the response of at least one previous response message.

On reception of a request whose command tag appears to be original, the completer shall perform the operation requested. On reception of a request whose command tag appear to be a retransmission of a previous command, no action is taken; but the stored response is (re)sent. This clause highlights the understanding that transmission layer may be unreliable and the mechanism by which that unreliability is addressed.

Example Conversations

This section provides some examples of conversations between Initiator and Completer.

```
// Initiator Discovery Operation (NOP)
```

In I0, the initiator sends a NOP CRH of tag 0, with Discover Option set, ADL=1, and LAST.
In I1, the initiator provided the 1 ADL worth of Initiator Advertisement. (4B in this example)
Since I1 is the LAST DWORD of the frame, an EOF or LAST is signaled implicitly by the datagram.

```
I0: Initiator Sends 0x80010080
I1: Initiator Sends 0x00000004  (EOF)
```

In response to receiving the above Command, the completer will respond with the following.
In C0, the completer sends its Response to the NOP.
In C1, the completer sends its 1 ADL worth of Completer Advertisement. (8B in this example)

```
C0: Completer Sends 0x800100B0
C1: Completer Sends 0x00000008  (EOF)
```

```
// Forced Read Operation
```

In this example we assume that the completer provides a bridge to a fabric which has the value 0xF00DFACE stored in memory at address 0x00000010.

In I0, the initiator sends a Read CRH of tag 0, with Discover Option set, ADL=1, and LAST.
In I1, the initiator sends the starting read address. (0x00000010 in this example)
Since I1 is the LAST DWORD of the frame, an EOF or LAST is signaled implicitly by the datagram.

```
I0: Initiator Sends 0x800100A0
I1: Initiator Sends 0x00000010  (EOF)
```

In response to receiving the above Command, the completer will take whatever action is needed to retrieve (read) the value at the address provided. Assuming the action takes place successfully:
In C0, the completer sends its “OK” Response to the Read.
In C1, the completer sends the 1 DW of read data

```
C0: Completer Sends 0x800100B0
C1: Completer Sends 0xF00DFACE  (EOF)
```

```
// Forced Write Operation
```

In this example we assume that the completer provides a bridge to a fabric which we can write. We will be writing the value 0xFEEDC0DE to fabric address 0x00000004

In I0, the initiator sends a Write CRH of tag 0, with Discover Option set, ADL=1, and LAST. Note that the FirstBE is “F” to enable all 4 Bytes for writing; while LastBE must be “0” for this ADL=1 operation. Note that the ADL=1 is indicating the number of DWORDs that follow the CRH and Address.
In I1, the initiator sends the starting write address. (0x00000004 in this example)
In I2, the initiator sends the DWORD data to be written (0xFEEDC0DE in this example)
Since I2 is the LAST DWORD of the frame, an EOF or LAST is signaled implicitly by the datagram.

```
I0: Initiator Sends 0x80010F90

I1: Initiator Sends 0x00000004

I2: Initiator Sends 0xFEEDC0DE  (EOF)
```

In response to receiving the above Command, the completer will take whatever action is needed to store (write) the data to the address provided. Assuming the action takes place successfully:
In C0, the completer sends its “OK” Response to the Write. Note that the returned ADL=0 as the write response has no data following the response CRH. Since C0 is the LAST DWORD of the frame, an EOF or LAST is signaled implicitly by the datagram.

```
C0: Completer Sends 0x800000B0  (EOF)
```

Remaining Work

This section describes outstanding known issues with the specification

1. Describe response behavior when there are read errors for greater than 1 DWORD.
2. Describe processing after an error is encountered.
3. Simplify NOP advertisement perhaps to max ADL (not Bytes).
4. First error in response completes with ERROR and LAST