

MicroTCA AMC523 Bare DDCP

Hardware User Guide



Fermilab, United States Department of Energy

Ryan Rivera

Jonathan Eisch

beams-doc-8194

Created: 24 August 2020

Last Update: 23 September 2020

Document Change Log

DocDB Version	Date	Pages	Editor	Description of Changes
V1	24 Aug 2020	All	rrivera	Created.
V2	23 Sep 2020	All	rrivera	Migrated from 64-bit to 32-bit DDCP handling. Standardized feature interpretation.

1. Introduction

The AMC523 Bare DDCP module provides an interface between the AMC523 Advanced Mezzanine Card (with MRT523b MTCA.4 Rear Transition Module) and DDCP clients. The AMC523 and MRT523b are both VadaTech cards; more information on the MicroTCA evaluation and demonstration effort can be found at beams-doc-7977.

The key features of the AMC523, as related to the MicroTCA evaluation and demonstration effort, include the following:

- 1000Base-X gigabit Ethernet port on AMC port 0.
- DDR3 2GB of memory.

The key features of the MRT523b, as related to the MicroTCA evaluation and demonstration effort, include the following:

- 12-channels (3 chips each with 4-channels) of ADC with 16-bit samples at 125MHz.

The source code repository and wiki for the AMC523 Bare DDCP project can be found here:

<https://cdcv.s.fnal.gov/redmine/projects/pipii-controls/wiki>

2. Functional Description

The AMC523 Bare DDCP module firmware is based on the VadaTech amc523_xxx_42x_xxx_mrt523b Reference Design targeted for the Xilinx Kintex-7 FPGA xc7k410tffg900-2. For the Bare DDCP functionality project, all features except for the 1000Base-X gigabit Ethernet port on AMC port 0 were commented out. The pure RTL otsdaq ethernet controller was inserted, and a pure RTL DDCP data manager was developed to provide the DDCP network layer to act as a DDCP server. The otsdaq ethernet controller interfaces to the 1000Base-X Xilinx IP block using GMII.

The DDCP server functionality is intended to implement a standard DDCP server at slot 15. Standard features 0, 1, 2, 5 are implemented with query, set, and read as appropriate. The MTU feature 5 will report 1500 bytes.

Feature 256 reaches the user firmware address space and for the Bare DDCP project will return 32-bit words for every read that mirror the target index (for 24-bits of index). Feature 256 writes are not implemented.

Feature 10, index 12, is a throttling register with read and write access. The throttle value become the delay between ethernet packet transmissions in units of 8 ns. The default value is 96ns which is the ethernet standard.

3. External Connections

The AMC523 module is equipped with a custom MTCA.4 RTM card, the MRT523b. For the Bare DDCP project, the MRT523b is ignored.

The Ethernet port on AMC port 0 is routed over the backplane of the MicroTCA crate to the MCH for external network handling. It is expected that the network supports the AMC523 module having an IP address of 10.0.0.15 and MAC address of 008055EC00010F.

4. Operation

ARP Operation

The otsdaq ethernet controller handles ARP replies and gratuitous ARPs on reset. It is common for network infrastructure to automatically make ARP requests to locate the MAC address of a target IP address.

ICMP Ping Operation

The otsdaq ethernet controller handles ICMP ping replies completely internally.

For example, to ping the AMC with an IP address of 10.0.0.15 use the following command:

```
ping 10.0.0.15
```

A successful response will look like this:

```
PING 10.0.0.15 (10.0.0.15) 56(84) bytes of data.  
64 bytes from 10.0.0.15: icmp_seq=1 ttl=128 time=0.104 ms  
64 bytes from 10.0.0.15: icmp_seq=2 ttl=128 time=0.063 ms
```

DDCP Read Operation

From the DDCP client point of view, reads are in units of 32-bit words. The internal address space for the firmware is comprised of 16-bits for feature and 24-bits for index.

A read from feature 256 should expect the index mirrored back in the data.

For example, using the ddcprw tool, one can read (option -r) a count of 4 32-bit words (option -c4 and -d32) from an AMC at 10.0.0.15 (option -@10.0.0.15), MTU of 1500 bytes (option -u1500), virtual slot 15 (options -s15), starting at feature 256 (option -f256) and index 8 (option -i8), by using the following command:

```
ddcprw -@10.0.0.15 -s15 -u1500 -f256 -i8 -c4 -d32 -r
```

A successful response will look like this:

```
Updated DDCP environment...
  server: 10.0.0.15, MTU: 1500, slot: 15, feature: 256, index: 8, count: 4,
width: d32
Read 4 feature #256 d32 location(s)
Note: indices are displayed in decimal, data are displayed in hexadecimal.
      8: 00000008 00000009 0000000a 0000000b  *.....*
```

DDCP Query Operation

From the DDCP client point of view, queries are defined for any read address. The query response will always indicate the supported number elements and operations for the target feature.

For example, using the `ddcprw` tool one can query (option `-q`) an AMC at 10.0.0.15 (option `-@10.0.0.15`), for supported operations of virtual slot 15 (option `-s15`), feature 1 (option `-f1`), and index 0 (option `-i0`), by using the following command:

```
ddcprw -@10.0.0.15 -s15 -f1 -i0 -c1 -d32 -q
```

A successful response will look like this:

```
Updated DDCP environment...
  server: 10.0.0.15, MTU: 1500, slot: 15, feature: 1, index: 0, count: 1,
width: d32
Query node 10.0.0.15, slot #15, feature #1...
supportedOps: Query Read Set, elementBytes: 4, elements: 1
```

This response indicates feature 1, which is the standard interrupter feature, supports query, read, and set operations; this response also indicates that the module supports 1 element (i.e. 1 interrupt handler destination is supported).

DDCP Write Operation

From the DDCP client point of view, writes are in units of 32-bit words.

For example, using the `ddcprw` tool one can write/poke (option `-p`) a count of 2 words (option `-c2` and `-d64`), with value `0x400` and `0xA00`, to an AMC at 10.0.0.15 (option `-@10.0.0.15`), virtual slot 15 (options `-s15`), starting at feature 0 (option `-f1`) and index 0 (option `-i0`), by using the following command:

```
ddcprw -@10.0.0.15 -s15 -f0 -i0 -c2 -d32 -p 0x400 0xA00
```

A successful response will look like this:

```
Updated DDCP environment...
  server: 10.0.0.15, MTU: 1500, slot: 15, feature: 0, index: 0, count: 2,
width: d32

Patch 2 feature #0 d32 location(s)
User provided values: 00000400 00000a00
```

DDCP Interrupt Operation

DDCP interrupts are sent from the AMC523 module as though the AMC523 is a DDCP client to a predetermined DDCP server (i.e. a UDP listener on port 65000). Note that all other DDCP operations (i.e. Read, Query, and Write) are usually conducted from the perspective that the AMC523 is a DDCP server and the interacting UDP process is the DDCP client.

For an interrupt to be successful, the interrupt destination must be setup in advance. This is done by writing a word to feature 1, index 0 indicating the 16-bit slot and 16-bit feature interrupt handler destination. Note that the interrupt destination MAC and IP address are automatically captured during this write operation (i.e. they are not specified in the DDCP data payload), and the DDCP interrupt destination port is always 65000 (since the destination is considered a DDCP server). The bit definition of the word are as follows:

Bits	Description of virtual slot 1, feature 0, index 13 (0xD) bitfields
31:16	16-bit interrupt destination slot
15:0	16-bit interrupt destination feature

Following the setup of the interrupt destination, any write of a 1 to bit-0 of feature 2, the standard software interrupt generation features, will trigger an interrupt.

For example, using the `ddcprw` tool one can force an interrupt from an AMC at 10.0.0.15 (option `-@10.0.0.15`), to be sent to the commanding host's slot 4 and feature 10, by using the following commands:

```
ddcprw -@10.0.0.15 -s15 -f1 -i0 -c1 -d32 -p 0x0004000a #setup handler
ddcprw -@10.0.0.15 -s15 -f2 -i0 -c1 -d32 -p 0x1 #trigger interrupt
```

5. Firmware Programming

The AMC523 module uses the front-panel JTAG connector for programming .bit format files to the FPGA, and for accessing the internal logic analyzer. The Flash PROM can also be written over the front-panel JTAG (using an .mcs format file) for persistent programming, which will survive power cycles.