Pixel DTB Ethernet Communication Protocol (DECC)

Caleb Fangmeier
University of Nebraska, Lincoln

February 6, 2015

Contents

1 Introduction

This paper gives a description of the DTB Ethernet Communication and Control (DECC) Protocol. The CMS experiment at CERN's LHC uses a tracker at its innermost later. This tracker is made up of individual silicon detectors. For the testing and comissioning of these detectors, a data aquisition(DAQ) device was created. The current incarnation of the DAQ card is known as the Digital TestBoard or DTB.

The DTB can communicate directly with a silicon detector for purposes such as reading out data, or configuration. The DTB is not designed to interact with humans directly. Rather, it is connected to a PC via either High Speed USB(USB 2.0) or Gigabit Ethernet. The PC then has software that allows a user to request for a test to be run, and then communicates that request to the DTB.

The protocol that was chosen to facilitate communication between a PC and a DTB was a custom flavor of Remote Procedure Call(RPC). The main idea of RPC is to abstract the actual serial communication between a master(PC) and slave(DTB) as a series of function calls. In other words, a user will call a function with possible arguments and a return type on a the PC, then a matching function will execute on the DTB.

For the purposes of developing DECC, what matters about RPC is that it is just a bidirectional serial stream of bytes. The details of how these bytes are organized has no implications on the design of DECC.

2 System Requirements

Based on the above considerations, the following requirements for DECC were laid out.

1. Speed is paramount. Therefore, DECC must have a minimal overhead. This is interpreted to mean two things.
   (a) The percent of data packets that is made up of bytes dedicated to DECC must be small, at least for large packets(>100 bytes).
   (b) The computational overhead, especially on the DTB, must be minimal.
2. DECC must allow for communication over a local subnet(eg. through an Ethernet switch or router). However, communication over a WAN or the Internet is not required. Therefore, an implementation of TCP/IP or UDP/IP is not necessary. However,

(a) Multiple DTBs must be able to exist on a subnet without interfering with each other's operations.

(b) Multiple host PCs must also be able to exist on a single subnet without interfering with each other.

(c) Finally, a PC may control multiple DTBs on a subnet, but a DTB can only be controlled by a single PC at a time.

3 The DECC Protcol

An Ethernet packet utilizing the DECC protocol will look like the following

Field	DST-MAC	SRC-MAC	Ethertype	ProcNum	T-Byte	DatSize	data
Size in Bytes	6	6	2	2	1	2	1-1500

- **DST-MAC** The MAC address of the destination device. Set to `0xFFFFFFFFFFFF` for broadcast packets.
- **SRC-MAC** The MAC address of the source device.
- **Ethertype** This identifies the protocol that is contained by the packet. For DECC, the Ethertype is `0x0809`.
- **ProcNum** This is the process number of the PC application.
- **T-Byte** This is used by DECC to identify what this packet is trying to do. See below.
- **DatSize** The size in bytes of the RPC packet.
- **data** The data that DECC is responsible for sending. Normally limited to 1500 bytes by the Ethernet standard.

The Type-Byte, or T-Byte, is responsible for identifying the type of a DECC packet. The different possible values for T-Bytes and their meanings are summarized in the following table.

T-Byte	Sent from PC	Sent from DTB
0x0	RPC Packet	RPC Packet
0x1	Hello: Query DTBs on subnet for status	status: unclaimed, or operation: sucessful
0x2	Claim: Attempt to claim targeted DTB	status: claimed, or operation: failure
0x3	Unclaim: Attempt to free the targeted DTB	UNUSED
0x4	Force Claim: Claim DTB, even if not unclaimed	UNUSED

The way the DECC protocol works is the following. The initial state of the system is there are a set of DTBs and a set of PCs on a local subnet. All of the DTBs are unclaimed. PC A sends out a "Hello" DECC packet which is broadcast throughout the subnet. Every DTB will then send a response with it's status: either claimed or unclaimed. Since all DTBs are unclaimed, the T-Byte of all response packets will be 0x1, meaning "unclaimed". The user at PC A can then select from a list of available DTBs one to connect to, or "claim". PC A will then send out another DECC packet with T-Byte 0x2 to the selected DTB. If the DTB hasn't been claimed in the meantime by another PC, the DTB will accept the claim and send back a response packet with T-Byte 0x1, meaning the claim was sucessful. If the claim was unsucessful, the T-Byte will be 0x2. If the claim was sucessful, PC A and the claimed DTB can now communicate RPC commands until finished. When finished, PC A will send an unclaim packet to the DTB. It is then available to be claimed by another PC.

As another example, say there are two DTBs and a single PC on a subnet. The operator wants to run tests with both DTBs simultaneously with a single PC. This is accomplished using the ProcNum field. When a DTB is claimed, it remembers both the MAC address of the claiming PC as well as the ProcNum of the claiming program on that PC. The process number is simply the default, ProcNum can be any identifying number. (For example, in the case where multiple DTBs need to be controlled by a single instance of a program.) Now, when a DTB sends data to it's controlling *process*, it addresses the packet both to the MAC address of the controlling machine and the process number on that machine. Together this data uniquely identify the recipient of the packet.