EUDET-Memo-2008-18

A trigger distributor box for the LP-TPC

G. De Lentdecker, J.-P. Dewulf, X. Janssen, Y. Yang [*]

November 27, 2008

Abstract

This memo describes the trigger distributor box (TDBOX) with is part of the EUDET JRA2 readout system for the Large Prototype TPC (LP-TPC) for the Linear Collider. The TDBOX aims to interface the different LP-TPC front-end electronics and data acquisition systems to the EUDET Trigger logic Unit (TLU). In particular, it redistributes the trigger information and event number.

[*]IIHE (ULB-VUB), Brussels, Belgium

1 Introduction

The detector at the future linear electron-positron collider ILC (International Linear Collider) will include a high precision tracking device with a track momentum resolution of $\delta p_t/p_t^2 \sim 5.0\ 10^{-5}$ GeV^{-1}c^{-1}. A large Time Projection Chamber (TPC) is a candidate for the central tracker system. Within the EUDET framework, a field-cage for a large prototype TPC (LP-TPC), 60 cm long, with an outer radius of 77 cm has been builded [1] and installed in the 6 GeV electron test beams area of DESY in October 2008. This research infrastructure offers some modularity to investigate various TPC endplate amplification systems, pad sizes and geometries as well as different readout systems.
Among other possibilities, a readout system based on the ALTRO electronic used in the ALICE experiment at LHC [2], as well as a readout system based on the AFTER electronic of the T2K experiment electronic [3] have been developed. They both need to be interfaced to the EUDET Trigger Logic Unit (TLU) [4] which provides a common trigger source and an event synchronisation mechanism to all subdetectors under test beams. This memo describes the trigger distributor box (TDBOX) which provides this interface.

2 TDBOX requirements

2.1 Requirements from the TLU

For our purpose, the TLU provides a LVDS interface through a RJ45 connector with the following signals:

- TRIGGER (output)
- BUSY (input)
- TLU-CLOCK (input)
- TLU-RESET (output)

which allow to use a trigger data handshake mechanism with the data acquisition systems of each detector under test (DUT) to ensure a synchronisation based on a single event number. In case a trigger has to be delivered by the TLU, the handshake mechanism proceeds as following:

1. TLU asserts TRIGGER;
2. On receipt of TRIGGER going high, the DUT should assert BUSY;
3. On receipt of BUSY going high, TLU de-asserts TRIGGER and switches the TRIGGER line to the output of a 32 bits shift register holding the event number;
4. The DUT clocks the event number out of this shift register by toggling the TLU-CLOCK;
5. After reading the event number from the TLU and being ready to read more data, the DUT de-asserts BUSY;
6. The system is hence ready for next trigger.

The first aim of the TDBOX is to comply to this handshake mechanism in the case of the LP-TPC without having to implement it directly in each different readout electronic system. In the followings, the TLU-RESET signal is not used.

The TLU also provides, via an USB interface, a logging of the received trigger together with an associated timestamp. This information can be integrated in the data stream and used a posteriori to cross check the event synchronisation between several DUT's.

2.2 Requirements from the readout systems

The ALTRO readout based system is existing in 2 versions:

1. A full scaled system based on a PCI-64bits bus interface to data acquisition PC;
2. A lightweight system with an USB interface to data acquisition PC allowing to readout a limited number of channels.

The first system will be used in the DESY testbeam with the LP-TPC and the second one for tests with a small TPC.

The PCI-64 system requires the following input signals:

- 4 LVTTL 3.3 V triggers (1 signal for each of the 4 crates);
- 4 LVTTL 3.3 V clocks at 40 MHz (1 signal for each of the 4 crates).

The USB system requires only a NIM trigger input.

The AFTER readout based system requires the following input signals:

- 1 LVTTL 3.3 V trigger;
- 1 LVTTL 3.3 V reset.

None of these readout systems are able to receive a LVDS TRIGGER signal, provide a BUSY signal or read the event number directly from the TLU. Hence the TDBOX should ensure a conversion from LVDS to LVTTL and NIM of the trigger signal, generate the BUSY signal and provide another way to include the event number in the data stream. It should as well provide the common 40 MHz clock for the ALTRO readout and the reset signal for the AFTER readout.

2.3 Other requirements

To allow an offline cross check of the LP-TPC readout synchronisation with the TLU, the TDBOX should associate to each TPC event a timestamp which can be compared to the TLU event timestamp afterwards.

For test purpose, the TDBOX should provide a standalone mode, i.e. without connection to the TLU, in which case the trigger source can be selected among the following options:

- 5 NIM inputs;
- An internal "random" trigger with adjustable rate;
- A software trigger.

In this mode, the event number should be generated by the TDBOX directly.

3 TDBOX implementation

3.1 Hardware

The LP-TPC trigger distributor box is made of two parts:

- The distributor box in itself;
- A small fan-out card to be placed close to the detector readout electronic.

This separation was needed to insure that only components which are non magnetisable are close to the LP-TPC magnet. They are connected together via LVDS signals.

The distributor box is based around a commercial FPGA board [5] to which dedicated LVDS, NIM and ethernet interfaces have been added. The FPGA board include a 50 MHz clock quartz oscillator which is divided by a PLL in a 40 MHz clock feeded through the fan-out card to the TPC frontend readout as needed for the ALTRO electronic. The ethernet interface is used to configure the TDBOX and to ship to the data acquisition PC the event number fetched from the TLU as well an internal TDBOX event number and an internal timestamp which is derived from a 1 MHz clock. The embedded FPGA logic will be described in section 3.2 and the configuration and operation of the TDBOX will be explained in section 4.

The fan-out card provides several copy of the trigger signal to the detector readout (1 NIM and 4 LVTTL 3.3 V signals), a LVTTL 3.3 V reset signal for the AFTER readout, as well as 4 copy of the ALTRO needed 40 MHz clock which is distributed via a PLL to correct for possible signal distortion.

3.2 FPGA logic

As soon as the TDBOX is configurated and that a run has been started (see section 4), the embedded FPGA will proceed in the following order:

1. Wait for a trigger from the selected source (TLU, local NIM, random or soft trigger).
2. Once a trigger has been received, perform the following tasks in parallel:
    - asserts BUSY;
    - send the trigger to the detector frontend via the fan-out card;
    - increment the internal TDBOX event number and flag this event with a TDBOX timestamp;
    - and in TLU slave mode, read the TLU event number by toggling the TLU-CLOCK.
3. Send the TLU event number, the TDBOX event number and the TDBOX timestamp via the ethernet interface to all clients (data acquisition PC's).

4. De-asserts BUSY if one of the following conditions occurs:
    - the TPC data acquisition PC notify by ethernet that the frontend is ready for next event;
    - this frontend acknowledge has not happen before a given timeout.
5. Wait for next trigger or run stop.

3.3 Connectors and LED's

On the **TDBOX front panel**, the following connectors are provided:

- 1 RJ45 connector providing the interface to the TLU (named TLU) which should be connected via a standard ethernet like cable to the corresponding TLU RJ45 connector if the TLU slave mode is used;
- 1 RJ45 connector providing the interface to the fan-out card (named fan-out) which should be connected via a standard ethernet like cable to the fan-out card

On the **TDBOX rear panel**, the following connectors are provided:

- 1 RJ45 connector providing the ethernet interface (named ETH) which should be connected to the LAN via a standard ethernet cable;
- 5 LEMO connectors providing NIM trigger inputs (named TRG1 to TRG5) for the TDBOX standalone mode;
- 1 USB (type B) connector to be used only to reprogram the FPGA by "experts";
- 1 connector to provide DC voltage to the fan-out card (named DCFO);
- 3 power plugs (named POW1 to POW3) to be connected to the 3 corresponding DC adaptators (named POW1 to POW3). **BE CAREFUL TO CONNECT THEM IN CORRECT ORDER NOT TO DAMAGE ELECTRONIC.**

On the **TDBOX front panel**, the following LED's are provided:

- 1 LED showing that a trigger has been received from the TLU (named TRGIN);
- 1 LED showing that a trigger has been sent to the fan-out card (named TRGOUT);
- 1 LED showing the BUSY status (named BUSY).

On the **Fan-Out card**, the following connectors are provided:

- 1 RJ45 connector providing the LVDS interface to the TDBOX which should be connected via a standard ethernet like cable to the TDBOX;
- 1 connector to provide DC voltage to the fan-out card;
- 1 LEMO connector providing a LVTTL 3.3 V busy signal output (named BUSY);
- 4 LEMO connectors providing LVTTL 3.3 V 40 MHz clock output (named CLK);
- 4 LEMO connectors providing LVTTL 3.3 V trigger signal output (named LTRG);
- 1 LEMO connector providing a LVTTL 3.3 V reset signal output (named LRES);
- 2 LEMO connectors providing NIM trigger signal output (named NTRG);

4 TDBOX software manual

After connecting the TDBOX and the fan-out card according to the informations in section 3.3, the steps described in the following subsection have to be followed to configure and take data. Everything happens through the exchange of ethernet messages.
Section 4.3, 4.4 and 4.6 and 4.7 are intended only to those PC/program who will act as a run control for the TDBOX and may be skipped if only the event numbers and timestamp have to be received.

4.1 Configuring the TDBOX IP address

This has to be done only once at the initial installation and for the moment help should be asked to us[1] as it unfortunately involve proprietary software.

4.2 Establishing a TCP/IP client connection to the TDBOX server

The ethernet TDBOX interface is implemented as a TCP/IP server up to which maximum 4 clients can connect together. During data taking the clients have to keep the connection open in order to receive the event data from the TDBOX. A client can be passive and only read the event data from the TCP/IP message buffer or active and also send commands to the TDBOX, i.e. act as a config and run control program. It is however highly recommended that only one client take the control of the TDBOX. The TCP/IP server port of the TDBOX is 4000, the separator character is ; and the message terminator character is #.

4.3 Setting the TDBOX mode and parameters

Once the client controlling the TDBOX has established connection, some configuration has to be performed before starting the first run.

BUSY timeout: To set the timeout after which the BUSY is automatically de-asserted (if this mode is selected), the following 13 characters TCP/IP message has to be send:

BSY;WORDTOUT#

where WORDTOUT is a 32 bits word in hexadecimal giving the timeout in microseconds.

Random rate: To set the time interval between 2 pseudo-random trigger (if this mode is selected), the following 13 characters TCP/IP message has to be send:

RND;WORDRNDM#

where WORDRNDM is a 32 bits word in hexadecimal giving the timeout in microseconds.

[1] xavier.janssen.AT.ulb.ac.be

TDBOX mode: To set the TDBOX mode, the following 13 characters TCP/IP message has to be send:

TRG;WORDCONF#

where WORDCONF is a 32 bits word in hexadecimal format with the following meaning:

Bit(s)	Value	Meaning
0	0/1	off/on the possibility to de-asserts BUSY via ethernet
1	0/1	off/on the possibility to de-asserts BUSY after a fixed timeout
2-4		Trigger source selection:
	000	TLU
	001	local NIM trigger inputs on the TDBOX rear panel
	010	pseudo random trigger according to selected rate
	011	ethernet software trigger (see section 4.7)
	100	push button #0 on the FPGA board, i.e. inside the TDBOX
	others	meaningless
5-6		ALTRO 40 MHz clock source:
	01	FPGA board oscillator
	10	External clock (not implemented)
		NB: Shortcut to 01 in the FPGA for the time being
7-8		Action to be taken on run init:
	00	only reset the TDBOX event register and time counter
	01	... and send an ALTRO reset (on trigger output of fan-out card)
	10	... and send an AFTER reset (on reset output of fan-out card)
	11	meaningless
9-13		OR mask for NIM trigger inputs
others	0	reserved for future use

A least one of the two first bits has to be set to 1 or the TDBOX would be stuck in BUSY mode after the first trigger.

4.4 Initialising, starting and stopping a run

Start run Before starting each run, the FPGA register holding the TDBOX event number and the TDBOX timestamp counter should be cleared by issuing the following 4 characters TCP/IP message:

IOR#

This will as well send the reset signal to the fronend electronic according to the configuration setted in WORDCONF.

To start the run itself, the following 4 characters TCP/IP message has to be send:

SOR#

This will as well de-assert the BUSY which is kept high to serve as veto for the TLU in between runs such that triggers can be send by the TLU.

Stop run To stop the run, the following 4 characters TCP/IP message has to be send:

EOR#

This will as well assert the BUSY which is kept high to serve as veto for the TLU in between runs such that triggers can not be send by the TLU.

4.5 Receiving event information

Onec a trigger has been processed, the TDBOX will send to all clients the following 30 characters TCP/IP message:

TRG;DBOXTIME;DBOXEVTN;TLUEVTNM#

where:

DBOXTIME	TDBOX time stamp in 0.1 millisecond units 32 bits word in hexadecimal format
DBOXEVTN	TDBOX internal event number 32 bits word in hexadecimal format
TLUEVTNM	TLU event number 32 bits word in hexadecimal format

NB: If the trigger source is not the TLU (see WORDCONF option), the TLUEVTNM is set to the DBOXEVTN word.

4.6 Acknowledging a trigger

To notify to the TDBOX that the frontend electronic and/or DAQ software is ready for next event the following 4 characters TCP/IP message has to be send:

ACK#

after receiving the TDBOX trigger data. This will de-assert the BUSY if the corresponding option is selected in WORDCONF.

4.7 Software (ethernet) trigger

To send an software trigger (if the corresponding option is selected in WORDCONF), the following 4 characters TCP/IP message has to be send:

SFT#

4.8 Simple example software

The following test software example will configure the TDBOX with pseudo random trigger source and acquire 10 events.

```
#include <stdio.h>              /* Input/output                          */
#include<stdlib.h>              /* number conversions and allocations   */
#include <errno.h>
#include <arpa/inet.h>
#define BUFFER_SIZE 40

int main(void)
{
   int ClientSock ;
   int errnos ;
   int res;
   int counter;
   int len ;
   char reponse[BUFFER_SIZE];
   uint data;
   char cc ;
   struct sockaddr_in address;
   char *ServerAddr ;

   /*  Create client socket */
   ClientSock = socket(AF_INET,SOCK_STREAM,0);
   errnos  = errno;
   if( ClientSock <0 ) {
      printf( "%s:%s","InitClient:socket\n",strerror(errnos));
   }
   printf("InitClient:socket %d opnened\n",ClientSock);

   /*  Name socket: */
   address.sin_family      = AF_INET;
   address.sin_port        = htons(4000);
   ServerAddr              = "193.190.246.191" ;
   inet_pton(AF_INET,ServerAddr,&address.sin_addr);
   printf("InitClient:Addr %s Port %d\n",ServerAddr,4000);

   /*  Connect to Server */
   res = connect(ClientSock,(struct sockaddr *)&address,sizeof(address));
   errnos  = errno;
   if ( res == 0 ) {
      printf( "%s->%s","InitClient:connect \n ",inet_ntoa(address.sin_addr));
   }
   else {
      printf("%s->%s","InitClient:connect",strerror(errnos));
   }

   // STOP RUN (Reset needed if program was previously intteruped by ctrl-c)
   send(ClientSock,"EOR#",4,0);

   // ********** Config DBOX ***********

   // Random trigger rate
   send(ClientSock,"RND;000186a0#",13,0);

   // Busy timeout
   send(ClientSock,"BSY;00002710#",13,0);

   // Trigger source: Random Trigger
   send(ClientSock,"TRG;0000000B#",13,0);

   // ********* TAKE 1 RUN ************

   // Init (reset dbox evt/time counters) + Start Run
   send(ClientSock,"IOR#",4,0);
   send(ClientSock,"SOR#",4,0);

   // Accept 10 events
   int i = 0;
   while(i < 10 ) {
      i++;
      printf(" Waiting for data : \n");
      res = recvfrom(ClientSock,reponse,BUFFER_SIZE,0,NULL,0);
      send(ClientSock,"ACK#",4,0);
      printf("TRIG RECEIVED: %s \n",reponse);
   }

   // Stop Run
   send(ClientSock,"EOR#",4,0);

   /* Close client */
   close(ClientSock);

   return 0;
}
```

5 Conclusion

Acknowledgement

This work is supported by the Commission of the European Communities under the 6th Framework Programme ”Structuring the European Research Area”, contract number RII3-026126, and by the Belgian ”Fonds de la Recherche Scientifique - FNRS”.

References

[1] T.Behnke, K.Dehmelt, R.Diener, ”Integration Issues Around the TPC, T.Behnke”, EUDET-Memo-2007-42;
T.Behnke, R.Diener, L.Hallermann, P.Schade, ”Status and Plans of the Large TPC Prototype for the ILC”, EUDET-Memo-2007-37;
L.Jönsson, ”The large prototype TPC”, EUDET-Memo-2006-06.

[2] D.Bertrand, G.De Lentdecker J.P.Dewulf, X.Jansen et al., ”A prototype readout system for the LCTPC”, EUDET-Memo-2008-06.

[3] D. Calvet, ”T2K TPC Readout Electronics”, September 2008.

[4] D. Cussans, ”Description of the JRA1 Trigger Logic Unit (TLU)”, EUDET-Memo-2007-02.

[5] Cyclone III FPGA Starter Kit, http://www.altera.com/products/devkits/altera/kit-cyc3-starter.html.