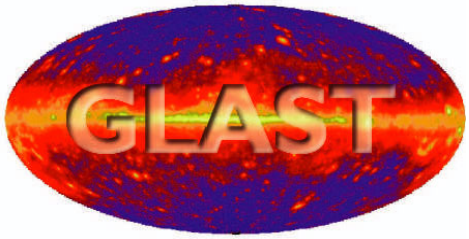

LAT Flight Software

DAB Programmers' Guide

Type: Programmers' Guide
Version: V0-0-0
Author: J. Swain
Created: 30 October 2003
Updated: 10 November 2003
Printed: 22 September 2004

This document provides an introduction to the DAQ Board driver package, DAB.

Contents

0 Introduction

0.0 Intended Audience

This document is intended to provide a brief introduction to the DAB package for programmers who will build on top of the DAB libraries as well as anyone who must maintain this package.

0.1 References

LAT-TD-00606, “LAT Inter-module Communications: A reference manual”.

LAT-TD-01380, “LATC Communication Board Driver: Software Architecture and Interfaces”.

LAT-TD-01545, “The GLT Electronics Module: Programming ICD specification”.

LAT-TD-01546, “The Event Builder Module: Design Specification”.

LAT-TD-01547, “The Command/Response Unit: Programming ICD specification”.

LAT-TD-01545, “The Power Distribution Unit: Programming ICD specification”.

DAB Automatically Generated Documentation, http://www.slac.stanford.edu/exp/glast/flight/doxygen/Doxyidx.htm.

GLAST Acronym List.

0.2 Request for Comments

Please post corrections or questions to the SNITZ release announcement for the relevant DAB version. Failing this, email jswain@slac.stanford.edu

0.3 Overview

The DAB package provides the command and response facility for the CRU, GEM, EBM and PDU using either an LCB or a COMM I/O board. In addition to the pure LCB and COMM I/O implementations there is also a single item list version of the LCB driver, which presents the same interface as the COMM I/O version, for backwards compatibility. This document briefly describes how to use the libraries before giving a quick overview of the design and layout of the package.

1 DAB API

Although linux-x86 and Sun-SPARC versions of the libraries exist they are intended for use exclusively on VxWorks-PowerPC machines, thus the examples presented are all in the form of VxWorks scripts.

1.0 Initialisation

Access to the hardware (COMM I/O board or LCB) is through a pointer to a dab structure. Memory must be allocated for this structure and then the structure initialised, which configures the hardware.

1.0.0 COMM I/O

```
    ld < libcmx_asBuilt.o
    ld < libgnat.o
    ld < libgdab.o
    __DAB = MBA_alloc(DABsizeOf());
    DAB_LATP_SOURCE_ADDR=5
    DAB_VME_ADDRESS     =0x08000000
    DAB_VME_IRQ_LEVEL   =4
    DAB_VME_IRQ_VECTOR  =200
    gdabOpen(__DAB,
             DAB_LATP_SOURCE_ADDR,
             DAB_VME_ADDRESS,
             DAB_VME_IRQ_LEVEL,
             DAB_VME_IRQ_VECTOR);
```

Note that the DAB_LATP and DAB_VME parameters are designated by the hardware.

1.0.1 LCB

```
# Load libraries
ld < libpbs.o
ld < libmsg_mt.o
ld < libmsg_init.o
ld < liblcbd.o
ld < libliox.o
ld < lib/libdab_to.o
ld < lib/libldab.o
ld < lib/libgcru_lcb.o
ld < lib/libggem_lcb.o

# Initialise PBS and MSG
PBS_initialize(0,0);
msg_init();

# Initialise the LCB
lcb_addr = 0x3C
lcb = MBA_alloc(LCB_sizeOf());
LCB_create(lcb);
LCB_IO_CSR_Write( lcb, 0x0);

errFCB = MBA_alloc(FORK_fcb_sizeof(1));
errFPA = MBA_alloc(FPA_fcb_sizeof());
fpaBuf = MBA_alloc(16*100);
LCB_errQueueInit(lcb, errFCB, errFPA, fpaBuf, 16*100);
reqFCB = MBA_alloc(FORK_fcb_sizeof(1));
rstFCB = MBA_alloc(FORK_fcb_sizeof(1));
reqSysMsg = 1000;
LIOX_sysInit( lcb, rstFCB, reqFCB, reqSysMsg);

# Associate the LCB with the DAB
__DAB = MBA_alloc(DABsizeOf());
DABopen( __DAB, lcb);
```

1.1 Address Assignment

Once the DAB handle has been opened the interface for the COMM I/O board and the single-item command list LCB implementations are identical (hereafter labled COMM I/O). As far as possible the remaining differences between the multi-item command list LCB and the COMM I/O implementations have been concealed from the user.

Before an item on the LATp Command/Response fabric can be manipulated it must be assigned a LATp address. This is done by setting the command enable mask within the CRU to only pass commands to the target node and broadcasting a register load to set the address. The following VxWorks script fragment performs this procedure for the GEM and EBM[1]. At the same time the variables in the `dab` structure are set to the same addresses.

[1] The CRU address is fixed to be 0x0 and there are two PDUs so the LATp address is one of the arguments to the C/R functions for the PDU.

```
    CRUconLoad   (__DAB, CRU_COMMAND_ENABLE, CRU_GEM_MASK);
    GEMconLoad   (__DAB, BCAST_ADDR,         GEM_LATP_ADDR);
    DABlam       (__DAB, GEM_LATP_ADDR);
    DABsetGEMAddr(__DAB, GEM_LATP_ADDR);

    CRUconLoad   (__DAB, CRU_COMMAND_ENABLE, CRU_EBM_MASK);
    EMBconLoad   (__DAB, BCAST_ADDR,         EBM_LATP_ADDR);
    DABlam       (__DAB, EBM_LATP_ADDR);
    DABsetEBMAddr(__DAB, EBM_LATP_ADDR);
```

1.2 Commanding

The DAB package facilities the reading and writing of front end registers for the GEM, CRU, PDU and EBM. The functions that are used to perform these actions using both a COMM I/O board and an LCB are presented below.

1.2.0 COMM I/O

```
payload = MBA_alloc(14);
value   = 0;

CRUconCmd   (__DAB, opCode);
CRUconLoad  (__DAB, regId, value);
CRUconRead  (__DAB, regId, &value);

GEMconCmd   (__DAB, opCode);
GEMconLoad  (__DAB, regId, value);
GEMconRead  (__DAB, regId, &value);
GEMieLoad   (__DAB, regId, payload);
GEMieRead   (__DAB, regId, payload);
GEMtamLoad  (__DAB, regId, payload);
GEMtamRead  (__DAB, regId, payload);
GEMschedLoad(__DAB, regId, payload);
GEMschedRead(__DAB, regId, payload);
GEMstatCmd  (__DAB, opCode);
GEMstatRead (__DAB, regId, payload);
GEMroiLoad  (__DAB, regId, payload);
GEMroiRead  (__DAB, regId, payload);
GEMwinLoad  (__DAB, regId, value);
GEMwinRead  (__DAB, regId, &value);

PDUconCmd   (__DAB, pduAddr, opCode);
PDUconLoad  (__DAB, pduAddr, regId, value);
PDUconRead  (__DAB, pduAddr, regId, &value);
PDUenvLoad  (__DAB, pduAddr, regId, payload);
PDUenvRead  (__DAB, pduAddr, regId, payload);

EBMconCmd   (__DAB, opCode);
EBMconLoad  (__DAB, regId, value);
EBMconRead  (__DAB, regId, &value);
EBMstatCmd  (__DAB, opCode);
EBMstatRead (__DAB, regId, &value);
```

Note that the size of payload does not always need to be fourteen bytes. Some of these functions take smaller payloads as specified in the GEM ICD.

1.2.1 LCB

```
payload = MBA_alloc(14);
value   = 0;

CRUconCmd   (__DAB, opCode);
CRUconLoad  (__DAB, regId, value);
CRUconRead  (__DAB, regId);

GEMconCmd   (__DAB, opCode);
GEMconLoad  (__DAB, regId, value);
GEMconRead  (__DAB, regId);
GEMieLoad   (__DAB, regId, payload);
GEMieRead   (__DAB, regId);
GEMtamLoad  (__DAB, regId, payload);
GEMtamRead  (__DAB, regId);
GEMschedLoad(__DAB, regId, payload);
GEMschedRead(__DAB, regId);
GEMstatCmd  (__DAB, opCode);
GEMstatRead (__DAB, regId, payload);
GEMroiLoad  (__DAB, regId, payload);
GEMroiRead  (__DAB, regId, payload);
GEMwinLoad  (__DAB, regId, value);
GEMwinRead  (__DAB, regId);

PDUconCmd   (__DAB, opCode);
PDUconLoad  (__DAB, regId, value);
PDUconRead  (__DAB, regId);
PDUenvLoad  (__DAB, regId, payload);
PDUenvRead  (__DAB, regId);

EBMconCmd   (__DAB, opCode);
EBMconLoad  (__DAB, regId, value);
EBMconRead  (__DAB, regId, &value);
EBMstatCmd  (__DAB, opCode);
EBMstatRead (__DAB, regId, &value);
```

The only difference between the LCB and COMM I/O interfaces is that the read commands for the LCB do not fill in a payload, the simply add a command item to the LCB command list. The results of the read will appear in the LCB results buffer after the command list has been queued and dispatched. Users should consult the LCB driver documentation for more details.

1.3 Testing

There are several functions provided for use during testing. These deliberately reverse the parity of various parts of the LATp communications packets to verify the parity testing in other parts of the system.

```
    DABsetCmdStrParity    (__DAB, 0/1);
    DABsetAccessDescParity(__DAB, 0/1);
    DABsetCmdPayloadParity(__DAB, 0/1);
    DABsetCellHeaderParity(__DAB, 0/1);
    DABsetCellBodyParity  (__DAB, 0/1);
```

1.4 Reset

The dab driver can issue a LAT reset command.

```
    DABreset(__DAB);
```

2 Software Architecture

The rest of this document is aimed at developers responsible for maintaining this package.

2.0 Libraries

The DAB package provides the C/R interface for four LAT components, GEM, EBM, CRU and PDU. For each of these the package provides libraries that use a COMM I/O board, an LCB, or an LCB in a manner that makes it look like a COMM I/O board. Thus there are a total of twelve component specific libraries. They are listed in the table below.

	COMM I/O	LCB	LCB (COMM I/O interface)
GEM	ggem	lgem	ggem_lcb
EBM	gebm	lebm	gebm_lcb
CRU	gcru	lcru	gcru_lcb
PDU	gpdu	lpdu	gpdu_lcb

As far as possible these libraries share interface files (C headers and CMX cid files) and reuse code. They are all built on top of one of two base libraries called gdab (COMM I/O) and ldab (LCB).