DAΦNE TECHNICAL NOTE

INFN - LNF, Accelerator Division

Frascati, November 30, 1993

Note: **C-8**

LVLibrary: a set of FORTRAN subroutines for accessing the DANTE HLS interface.

Alessandro Stecchi

1. Introduction

This document presents materials that will be part of an extensive user manual for the physicists involved in the High Level Software development. Even though it will be likely updated in order to satisfy new requirement it contains enough material to start testing and discussion on the matter.

For a better understanding of what follows refer to:

- Control System RTDB architecture (Control Group);
- use of FORTRAN STRUCTURE and RECORDS (any FORTRAN user manual i.e. Language System [1]);
- Protocol for Accelerator Applications (Control System Status Report 13/9/93).

The following pages describe a set of subroutines developed in order to access the Control System First Level Interface which is implemented in LabVIEW [2] from an external FORTRAN application (we will refer to such interface as HLS interface). This set constitutes the LVLibrary. The LVLibrary contains *User subroutines* and *Low Level subroutines* even though a programmer will likely deal only with the User subroutines within its application.

All presented subroutines as well the corrispondent HLS LabVIEW interface have been tested.

This document will not go into detail of the software methods used for the development of the routines themselves.

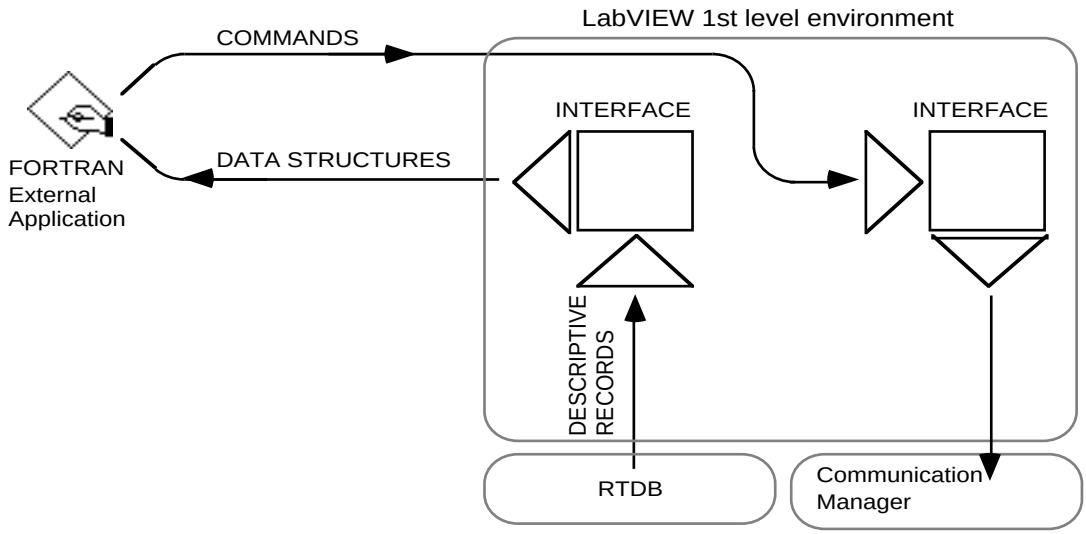


Fig.1 - The data and commands flow through the HLS interface.

2. User subroutines

In this first release the HLSError codes reported by each routine have to be defined yet.

`initLV(targetPortName,HLSUserID,HLSUnit,HLSErr)`

targetPortName	INPUT	CHARACTER*32	The LV PPC port name which we want to communicate with.
HLSUserID	INPUT	INTEGER*4	User Identification number. I used as password.
HLSUnit	OUTPUT	RECORD /HLSCommRec/	Record containing all the communication parameters neede in the following calls. The corresponding STRUCTURE is defined in the include file LVLibrary.h.
HLSError	OUTPUT	INTEGER*2	Error result code.

This subroutines initializes the communication with the LabVIEW HLS interface. The returned parameter HLSUnit must be used as a sort of *logical unit number* in the following calls.

`fetchData(HLSUnit,dataIdentificator,dataContainerPtr,dataSize,HLSError)`

HLSUnit	INPUT	RECORD /HLSCommRec/	As obtained from the LVInit routine. The corresponding STRUCTURE is defined in the include file LVLibrary.h.
dataIdentificator	INPUT	CHARACTER*8	ElementName, MachinePartName, VariableName and -in general- identificator recognized by th Control System.
dataContainerPtr	INPUT	INTEGER*4	FORTRAN pointer to a RECORD th matches the STRUCTURE corresponding to the wanted dataIdentificator. The pointer be calculated directly withi parameter list applying the operator %LOC() to the RECORD (see example below)
dataSize	INPUT	INTEGER*4	Size (in bytes) of the RECORD matches the STRUCTURE corresponding to the wanted dataIdentificator. DataSize ca calculated directly within th pameter list applying the ope SIZEOF() to the RECORD (see example below)
HLSError	OUTPUT	INTEGER*2	Error result code.

This subroutine sends a command of type 'SEND' (send a block of this type back to me) to the LV HLS interface. After this fetchData waits for the incoming data and then return those in the dataContainer RECORD variable.

Pointer to RECORD and size of a RECORD.

The fetchData subroutine requires -among its parameters- a pointer to the record we want to hold the fetched data and the length (in bytes) of the record itself. Making reference to a data STRUCTURE of type "example" and to a corresponding RECORD variable "misterX" we can use either:

```
c         declarations
          .................
          RECORD/example/misterX
          POINTER/example/misterXPtr
          INTEGER*4  lengthOfMisterX
          .................
c         statements
          .................
          misterXPtr=  %LOC(misterX)
          sizeOfmisterX = SIZEOF(misterX)
          CALL  fetchData(sessRefNum,dataID,misterXPtr,lengthOfMisterX,err)
          .................
or:
c         declarations
          .................
          RECORD/example/misterX
          .................
c         statements
          .................
          CALL  fetchData(HLSUnit,dataID,%LOC(misterX),SIZEOF(misterX),HLSError)
          .................
```

issueCmd(HLSUnit,commandString,HLSError) {*not implemented yet*}

HLSUnit	INPUT	RECORD /HLSCommRec/	As obtained from the LVInit routine. The corresponding STRUCTURE is defined in the include file LVLibrary.h.
commandString	INPUT	CHARACTER*32	Any System command.
HLSError	OUTPUT	INTEGER*2	Error result code.

This subroutine sends a command of type 'FRWD' (forward this command string) to the LV HLS interface.

quitLV(HLSUnit,HLSError)

HLSUnit	INPUT	RECORD /HLSCommRec/	As obtained from the LVInit routine. The corresponding STRUCTURE is defined in the include file LVLibrary.h.
HLSError	OUTPUT	INTEGER*2	Error result code.

This subroutine sends a command of type 'QUIT' (I am quitting; do not care about me longer) to the LV HLS interface. Then ends the PPC session and closes the PPC port.

3. Low level subroutines

```
openLV(targetPortName,MyPortRefNum,err)
```

targetPortName	INPUT	STRING*32	LabVIEW HLS interface port whi we want to communicate with (a the moment the name 'bucket' MUST be specified in the application.
MyPortRefNum	OUTPUT	INTEGER*4	PPC port reference number oper by the subroutine.
err	OUTPUT	INTEGER*2	Two components array

This subroutine checks for the existence of a LabVIEW port of type 'PPCToolBox' and then opens from scratch a PPC Port of the same type.

```
startSessionLV(targetPortName,MyportRefNum,MySessUserData,MySessRefNum,err)
```

targetPortName	INPUT	CHARACTER*32	The LV PPC port which we want communicate with.
MyportRefNum	INPUT	INTEGER*2	As obtained from the openLV subroutine.
MySessUserData	INPUT	INTEGER*4	It is copied in the correspond parameter of the PPCInform running on LV. It can be used a starting information.
MySessRefNum	OUTPUT	INTEGER*4	To be passed to subsequent HLS routines.
err	OUTPUT	INTEGER*2	Two components array

This subroutine opens a PPC session.

```
writeOnlyLV(MySessRefNum,MyUserData,MyBlockCreator,MyBlockType,
                MyBufferPtr,MyBufferLength,err)
```

MySessRefNum	INPUT	INTEGER*4	As obtained from the startSessionLV subroutine.
MyUserData	INPUT	INTEGER*4	Block header first field.
MyBlockCreator	INPUT	CHARACTER*4	Block header second field.
MyBlockType	INPUT	CHARACTER*4	Block header third field.
MyBufferPtr	INPUT	INTEGER*4	Pointer to the buffer that we to transmit.
MyBufferLength	INPUT	INTEGER*4	Length of the buffer that we w to transmit.
err	OUTPUT	INTEGER*2	Two components array

This subroutine sends to the LabVIEW target port a block of bytes.

```
readOnlyLV(MySessRefNum,MyUserData,MyBlockCreator,MyBlockType,
MyBufferPtr,MyBufferLength,err)
```

MySessRefNum	INPUT	INTEGER*4	As obtained from the startSessionLV subroutine.
MyUserData	INPUT	INTEGER*4	Block header first field.
MyBlockCreator	INPUT	CHARACTER*4	Block header second field.
MyBlockType	INPUT	CHARACTER*4	Block header third field.
MyBufferPtr	INPUT	INTEGER*4	Pointer to the buffer that we to transmit.
MyBufferLength	INPUT	INTEGER*4	Length of the buffer that we w to transmit.
err	OUTPUT	INTEGER*2	Two components array

This subroutine performs a PPCRead: receives a block of bytes from LabVIEW and then ends the session.

```
endSessionLV(MySessRefNum,err)
```

MySessRefNum	INPUT	INTEGER*4	As obtained from the startSessionLV subroutine.
err	OUTPUT	INTEGER*2	Two components array

```
closeLV(MyportRefNum,err)
```

MyPortRefNum	OUTPUT	INTEGER*4	PPC port reference number oper by the subroutine openLV.
err	OUTPUT	INTEGER*2	Two components array

This subroutine closes a PPCPort already opened by openLV.

4. An example of data exchange

The following example shows how to write a FORTRAN application that gets data from the LabVIEW HLS interface. In the example there is no error handling due to the fact that the error codes have not been defined yet.

```
          program HLS
          implicit none
          include 'LVLibrary.h'

          integer*2  error
          integer*4  MySessUserData
          string*32  targetPortName
          record/HLSCommRec/theUnit
          character*8  dataID

          structure/example/  ! • This structure is only an example.
            integer*4 a       !   The real descriptive records will be
            real*4 b1         !   included with the file DAFNETypes.h
            real*4 b2
            real*4 b3
          end structure

          record/example/theExample
          real*4  result

c         ------------------------------------------------------------------------
c                                          Start of MAIN
c         ------------------------------------------------------------------------
          targetPortName = 'bucket'                ! • Mandatory at the moment
          MyID = 0                                 ! • Password: any value
                                                   !   allowed at the moment.

c         Set up the communication with the HLS interface
   100 CALLinitLV(targetPortName,MyID,theUnit,error)

c         Read a block of type example
          dataID = 'NNNLLXXX'                      ! • Name of an element of
                                                   !   type "example"
   200 CALL fetchData(theUnit,dataID,%LOC(theExample), SIZEOF(theExample),error)

c         An example of calculations with the obtained values
          result = theExample.b1 + theExample.b2 + theExample.b3
          result = result ** theExample.a
          Write(*,*) ' The result is: ',result

c         Issue a command to the System
          theCommand = 'SET KCKA1001 DAC1 30'
   300 CALL issueCmd(theUnit,theCommand,error)

c         Terminate the session & close port
   400 CALL quitLV(theUnit,error)

          Write(*,*) ' All done.'

          stop
          end
```

Appendix A: table of CALLs

The following table reports for each subroutine the corresponding called subroutines and the used data structures.

Subroutine	Called Subroutine	Used Structures	Custom Include files
initLV	openLV startSessionLV	HLSCommRec	LVlibrary.h
fetchData	writeOnlyLV readOnlyLV	HLSHeaderRec	LVlibrary.h
issueCmd	writeOnlyLV	HLSCommRec HLSHeaderRec	LVlibrary.h
quitLV	writeOnlyLV endSessionLV closeLV	HLSCommRec HLSHeaderRec	LVlibrary.h
openLV	Gestalt PPCInit PPCOpen IPCListPorts PPCClose	PPCOpenPBRec PPCPortRec IPCListPortsPBRec PortInfoRec	
startSessionLV	PPCStart	PortInfoRec PPCStartPBRec OSErr	
writeOnlyLV	PPCWrite	PPCWritePBRec OSErr	
readOnlyLV	PPCRead	PPCReadPBRec OSErr	
endSessionLV	PPCEnd	PPCEndPBRec OSErr	
closeLV	PPCClose	PPCClosePBRec OSErr	

Appendix B: LabVIEW HLS interface VI

The HLS interface implemented in the first level behaves as a *server*. An external application using the routines of the LVLibrary behaves as a *client* and is in charge of starting the session with the inilLV routine. Once the HLS interface has accepted the session, it polls for any incoming request from the external application.

Such requests are characterized by an header made of three fields; in particular the last field *block type* (of 4 bytes) specifies what the interface has to do. The following 4-byte commands are recognized by the interface:

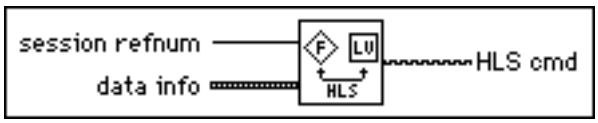
'SEND'	Send a record back to me.	The record type is specified in remaining fields of the header.
'FRWD'	Forward this command.	The command string is in the blo annexed to the header and its le specified in the remaining field header.
'PLOT'	Plot this array.	The byte stream and the array da type as well are in the block ar to the header and its length is specified in the remaining field header.
'QUIT'	I am quitting. Do not care any mo about me.	No further information for this

A 4-byte command (HLS command) with its eventual parameters and data is interpreted by a proper subVI which executes it by calling a dedicated subVI.

It follows a description of the HLSinterpreter.vi and of the dumpHLSrecord.vi which executes HLS commands of type 'SEND'.

HLS commands of type 'QUIT' are executed directly by the interface top level VI whilst HLS commands of type 'PLOT' have not been implemented yet.

HLSinterpreter.vi



I32 **session refnum** is the PPC session reference number relative to connection with the external FORTRAN application.

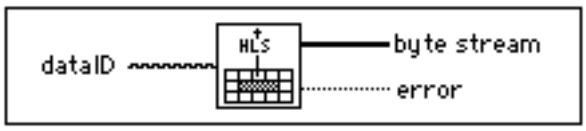
data info is a cluster containing the following parameters in t below. This cluster is the PPC block header.

U32 **user data**
U32 **block creator**
U32 **block type**

User data and **block creator** are interpreted as a 8 character

abc **HLS cmd** is a 4 characters string obtained from **block type**. The strings are recognized and returned:
'SEND'
'FRWD'
'PLOT'
'QUIT'
If **block type** is not a valid 4-byte pattern an empty string is

dumpHLSrecord.vi



abc **dataID** is a 8 characters string which specifies the descriptive want to fetch. It can be any element name recognized by the System

[U8] **byte stream** is a sequence of bytes which can be mapped on the re corresponding to the wanted data structure.

TF **error** is a boolean indicating whether a bus error occourred duri gathering or not.

REFERENCE

[1] Language System FORTRAN 3.0 Reference Manual,.Language System Corporation, 441 Charlisle Drive, Herndon, VA 22070-4802.

[2] LabVIEW® National Instrument Corporation, 6504 Bridge Point Parkway, Austin, TX 78730-5039.