

# **Programmer's Manual**

# **Digital Gamma Finder (DGF)**

## **Model Polaris**

June 2003

### **X-Ray Instrumentation Associates**

8450 Central Ave  
Newark, CA 94560 USA

Phone: (510) 494-9020; Fax: (510) 494-9040  
<http://www.xia.com>



### **Disclaimer**

Information furnished by XIA is believed to be accurate and reliable. However, XIA assumes no responsibility for its use, or for any infringement of patents, or other rights of third parties, which may result from its use. No license is granted by implication or otherwise under the patent rights of XIA. XIA reserves the right to change the DGF product, its documentation, and the supporting software without prior notice.

## 1 Overview

This manual is a short description of the Polaris C Library which is currently used by the Polaris Viewer. Advanced users can build their own user interface using the user accessible functions in the library. In order for the users to be familiar with this library quickly, a sample code using this library to initialize Polaris is also being released together with this manual.

## 2 Polaris C Library

The Polaris C Library contains a group of C functions used to control the Polaris. It can be compiled as either a XOP file currently used by the Polaris Viewer, or a dynamic link library (DLL) or static library used by customized user interface or applications. The library provides users six functions which can be used to fully control the Polaris. These six functions were described in sections 2.1 to 2.6. Section 2.7 gives the different options of compiling the Polaris C Library.

### 2.1 C\_Hand\_Down\_File\_Names

#### Function prototype

```
long C_Hand_Down_File_Names(char *File_Names[])
```

#### Parameter description

File\_Names: A two dimensional string array containing the names of system configuration files or settings file. Currently six files need to be downloaded: system FPGA file, Fippi file, DSP code binary file, DSP I/O parameter values file, DSP code I/O variable names file, and DSP code all variable names file. All file names should contain the complete path name.

#### Function description

This function can be used to download the names of files which are required for booting the Polaris.

### 2.2 C\_Boot\_System

#### Function prototype

```
long C_Boot_System(long Boot_Pattern)
```

#### Parameter description

Boot\_Pattern is a bit mask:  
Bit 0: Boot system FPGA  
Bit 1: Boot FIPPI

- Bit 2: Boot DSP
- Bit 3: Load DSP parameters
- Bit 4: Apply DSP parameters (call Set\_DACs and Program\_FIPPI)
- Bit 5: OFFLINE
- Bit 6: EPP\_IO
- Bit 7: USB\_IO

#### Function description

This function can be used to boot the Polaris. Under most of the circumstances, all the tasks listed in bit 0 to bit 4 should be executed to initialize the Polaris, i.e. the Boot\_Pattern should be 0x9F if using USB or 0x5F if using EPP.

### **2.3 C\_User\_Par\_IO**

#### Function prototype

```
long C_User_Par_IO(char *User_Par_Names[], double *User_Par_Values, char *User_Par_Name,  
long direction)
```

#### Parameter description

User\_Par\_Names: A two dimensional string array which contains user parameter names. Currently it can hold 64 names. If less than 64 names are needed (which is the current case), the remaining names should be defined as empty strings.

User\_Par\_Values: A double precision array containing the User Values to be transferred.

User\_Par\_Name: A string variable which contains the user parameter name.

direction:           0 - download from the host to this library.  
                      1 - upload from this library to the host.

#### Function description

This function downloads or uploads user values between the host and the library. For those DSP I/O parameters, this function calls other functions to convert these parameter values to DSP recognizable values or verse versa.

### **2.4 C\_Acquire\_Data**

#### Function prototype

```
long C_Acquire_Data(long Run_Type, unsigned int *User_data, char *file_name)
```

#### Parameter description

User\_data: An unsigned 32-bit integer array which holds data transferred between the modules and the host.

Run\_Type: An integer whose bits 0-3 specifies common run types, bits 4-7 specifies actions(start\stop\poll), and bits 8-11 specifies special control runs.

bits 0-3: 1 Get\_Traces()  
2 MCA\_Run\*  
3 List\_Mode\_run

bits 4-7: 1 Start new run  
2 Resume run  
3 Stop  
4 Poll

bits 8-11: Special control run

file\_name: A string variable which specifies the name of the output file. file\_name needs to have complete path.

#### Function description

This function acquires ADC traces, MCA spectrum, or list mode data. The string variable file\_name needs to be specified when stopping a MCA run or list mode run in order to save the MCA spectrum data or list mode data into a file. In all other cases, file\_name can be specified as an empty string. The unsigned 32-bit integer array User\_data is used only for acquiring ADC traces (run type 0x1) or reading out MCA spectrum. In all other cases, User\_data can be any unsigned integer array with arbitrary size. Make sure that the User\_data has the correct size and data type before reading out ADC traces or MCA spectrum.

## 2.5 C\_Set\_Current\_Module

#### Function prototype

long C\_Set\_Current\_Module(unsigned short ModuleNumber)

#### Parameter description

Module is an unsigned 16-bit integer which specifies the current module to be set. Module should be in the range of 1 to MAX\_NUMBER\_OF\_MODULES.

#### Function description

This function sets the current module number.

## 2.6 C\_Buffer\_IO

#### Function prototype

long C\_Buffer\_IO(unsigned short \*Values, unsigned short type, unsigned short direction)

Parameter description

The parameter Values is an unsigned 16-bit integer array used for data transfer between the host and the library. The parameter type specifies the I/O type. The parameter direction indicates the data flow direction. Table 1 lists the functionality of C\_Buffer\_IO for different combinations of type and direction.

**Table 1: The Description of Function C\_Buffer\_IO.**

Type	Direction	Values	I/O Operation
0	0	DSP I/O variable values	Write DSP I/O variable values to modules
	1		Read DSP I/O variable values from modules
1	1	DSP all variable values	Read all DSP variable values from modules

Function description

This function downloads or uploads DSP settings between the host, the library and the Polaris, or read out the whole data memory.

**2.7 Polaris C Library Compile Options**

Polaris C Library can be compiled as either an Igor XOP used in the Polaris Viewer, or a standalone C-Library free of Igor stuff. The latter can be used by advanced users to develop their own data acquisition systems.

The following table lists the required files for these two options.

**Table 2: Compilation Options of the DGF-4C C Library.**

Compile Option	Required Files		
	C source files	C header files	Library files
Standalone C-Library	boot.c, epp.c, polaris_c.c, usb.c, utilities.c	boot.h, Dlportio.h, epp.h, globals.h, Main.h, sharedfiles.h, usb.h, utilities.h	Dlportio.lib
Igor XOP	boot.c, epp.c, PolarisWinCustom.rc, polaris_c.c, polaris_iface.c, polaris_igor.c, usb.c, utilities.c	boot.h, Dlportio.h, epp.h, globals.h, Main.h, polaris_iface.h, sharedfiles.h, usb.h, utilities.h	Dlportio.lib

The Igor XOP option also needs the following files in the Igor XOP Library provided by WaveMetrics.

IgorXOP.h, VCExtraIncludes.h, Xop.h, XOPResources.h, XOPStandardHeaders.h, XOPSupport.h, XOPSupportWin.h, XOPWinMacSupport.h, XOPSupport x86.lib, and IGOR.lib.