Using SpaceWire with LabVIEW

SpaceWire Test and Verification, Short Paper

Alex Mason, Steve Parkes
STAR-Dundee Ltd.
Dundee, Scotland, UK
alex.mason@star-dundee.com, steve.parkes@star-dundee.com

***Abstract*— To support customers using the National Instruments LabVIEW software development environment, STAR-Dundee Ltd. have developed LabVIEW libraries and drivers allowing for the rapid integration of STAR-Dundee SpaceWire interface devices into EGSE or test and verification applications. Customers familiar with STAR-Dundee's STAR-System API suite can use a wrapper library to control and configure any supported SpaceWire interface device under the Windows operating system. Using a native LabVIEW NI-VISA driver, users can interface to STAR-Dundee SpaceWire PCI and cPCI, boards on any platform supported by LabVIEW, including National Instruments real-time targets.**

In this paper, the LabVIEW solutions provided by STAR-Dundee are described, including an overview of the APIs, and example usage demonstrating solutions to common tasks.

***Index Terms*— SpaceWire, LabVIEW, NI-VISA, VISA**

I. Introduction

The design of SpaceWire electronic check-out and ground support equipment can be both costly and time consuming. To help alleviate this problem, STAR-Dundee supplies a number of test and development devices that can be used to transmit and receive SpaceWire traffic and configure and monitor devices on a network. Users can write their own custom applications using a provided powerful API.

National Instruments LabVIEW can be used to rapidly develop test and measurement systems with custom graphical user interfaces.

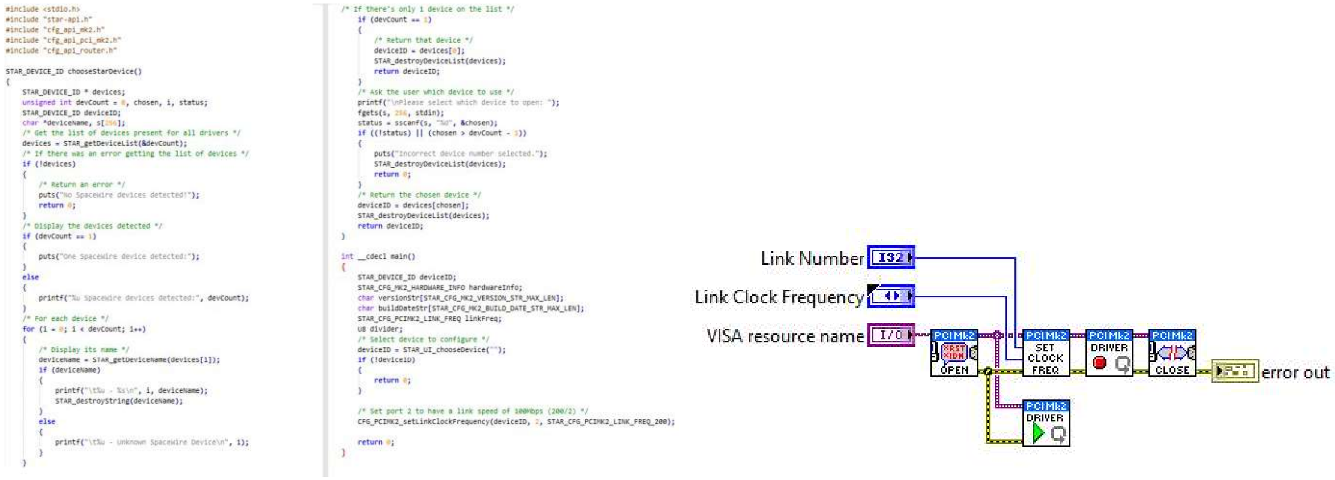
Combining STAR-Dundee equipment with LabVIEW provides a means of rapidly developing SpaceWire test applications.

II. LabVIEW

LabVIEW is a software development environment provided by National Instruments Corporation [1]. The environment provides a visual dataflow programming language in which functions are laid out in a flow chart style, with 'wires' connecting the output of one node to the input of another. Data is operated on at each node immediately as it becomes available, and the compiler identifies segments of code that can run in parallel and automatically splits the application into multiple threads.

LabVIEW offers the ability to work at a higher layer of abstraction than typical text based programming languages like C. For example, no manual memory allocation is required by the user, there are many included libraries hiding the implementation of File I/O and network connectivity, and graphical user interfaces are created in a drag and drop manner.

As an example of the way LabVIEW allows rapid development, compare the volume of C code required to select a desired SpaceWire device and configure its link speed to the code required to perform the same operation with the STAR-Dundee VISA Driver: (Fig. 1)



```c
#include <stdio.h>
#include "star-api.h"
#include "cfg_api_mk2.h"
#include "cfg_api_pci_mk2.h"
#include "cfg_api_router.h"

STAR_DEVICE_ID chooseStarDevice()
{
    STAR_DEVICE_ID * devices;
    unsigned int devCount = 0, chosen, i, status;
    STAR_DEVICE_ID deviceID;
    char *deviceName, s[256];
    /* Get the list of devices present for all drivers */
    devices = STAR_getDeviceList(&devCount);
    /* If there was an error getting the list of devices */
    if (!devices)
    {
        /* Return an error */
        puts("No SpaceWire devices detected!");
        return 0;
    }
    /* Display the devices detected */
    if (devCount == 1)
    {
        puts("One SpaceWire device detected:");
    }
    else
    {
        printf("%u SpaceWire devices detected:", devCount);
    }
    /* For each device */
    for (i = 0; i < devCount; i++)
    {
        /* Display its name */
        deviceName = STAR_getDeviceName(devices[i]);
        if (deviceName)
        {
            printf("\t%u - %s\n", i, deviceName);
            STAR_destroyString(deviceName);
        }
        else
        {
            printf("\t%u - Unknown SpaceWire Device\n", i);
        }
    }
/* If there's only 1 device on the list */
    if (devCount == 1)
    {
        /* Return that device */
        deviceID = devices[0];
        STAR_destroyDeviceList(devices);
        return deviceID;
    }
    /* Ask the user which device to use */
    printf("\nPlease select which device to open: ");
    fgets(s, 256, stdin);
    status = sscanf(s, "%d", &chosen);
    if ((!status) || (chosen > devCount - 1))
    {
        puts("Incorrect device number selected.");
        STAR_destroyDeviceList(devices);
        return 0;
    }
    /* Return the chosen device */
    deviceID = devices[chosen];
    STAR_destroyDeviceList(devices);
    return deviceID;
}

int __cdecl main()
{
    STAR_DEVICE_ID deviceID;
    STAR_CFG_MK2_HARDWARE_INFO hardwareInfo;
    char versionStr[STAR_CFG_MK2_VERSION_STR_MAX_LEN];
    char buildDateStr[STAR_CFG_MK2_BUILD_DATE_STR_MAX_LEN];
    STAR_CFG_PCIMK2_LINK_FREQ linkFreq;
    U8 divider;
    /* Select device to configure */
    deviceID = STAR_UI_chooseDevice("");
    if (!deviceID)
    {
        return 0;
    }

    /* Set port 2 to have a link speed of 100Mbps (200/2) */
    CFG_PCIMK2_setLinkClockFrequency(deviceID, 2, STAR_CFG_PCIMK2_LINK_FREQ_200);

    return 0;
}
```

Fig. 1. LabVIEW source compared to text based code.

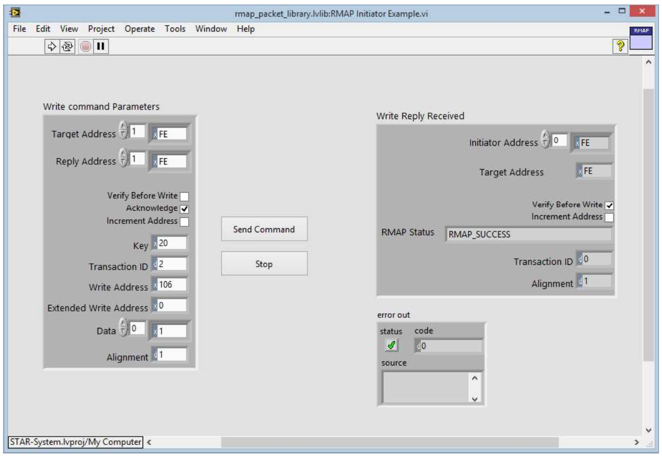
III. STAR-DUNDEE LABVIEW SOLUTIONS

STAR-Dundee provides two separate LabVIEW solutions: a LabVIEW wrapper around the existing STAR-System libraries (currently provided only for Windows based hosts), and a native LabVIEW NI-VISA driver that can be used on all targets supported by LabVIEW.

A. STAR-System Wrapper

STAR-System is the driver and API system provided with all new and future STAR-Dundee interface and router devices [2]. STAR-System provides high bandwidth and low latency packet transmission and reception, and a consistent API interface to numerous device types. Supported devices include the SpaceWire USB Brick Mk2 and Router Mk2s, and the PCI Mk2 and PCIe boards.

The STAR-System LabVIEW wrapper library [3] provides access to every function exported by the STAR-System C API, and includes a number of example VIs (Virtual Instruments) that provide implementations of commonly performed SpaceWire tasks, such as setting up routing tables, sending and receiving time-codes and configuring link speed and status. Also provided are the RMAP packet library and example implementations of an RMAP Target and Initiator (shown in Fig. 2).


rmap_packet_library.lvlib:RMAP Initiator Example.vi
File
Edit
View
Project
Operate
Tools
Window
Help
RMAP
Send Command
Stop
STAR-System.lvproj/My Computer

Write command Parameters

Target Address 1 FE

Reply Address 1 FE

Verify Before Write □

Acknowledge ☑

Increment Address □

Key 20

Transaction ID 2

Write Address 106

Extended Write Address 0

Data 0 1

Alignment 1

Write Reply Received

Initiator Address 0 FE

Target Address FE

Verify Before Write ☑

Increment Address □

RMAP Status RMAP_SUCCESS

Transaction ID 0

Alignment 1

error out

status ✓ code 0

source

Fig. 2. RMAP Initiator example front panel.

Using the STAR-System wrapper allows LabVIEW applications to share data with other STAR-System processes running on the host. For example, device names set up using the STAR-System Device Configuration GUI can be viewed or modified with changes propagated across all running processes. This can help a user quickly identify and select a desired device to work with without looking up serial numbers.

The complexity of the C API has been abstracted away where possible. No manual memory allocation is required to transmit and receive packets; this is handled by the wrapper with packet data buffers provided as LabVIEW byte arrays. LabVIEW events are used to implement device listeners and transfer completion events.

Performance of the LabVIEW wrapper compares favorably with that of unwrapped STAR-System performance [4] with performance figures roughly the same when transmitting and receiving packets of length above around 60 bytes (Fig. 3). These figures are for a 200Mbit/s link speed, and show performance is close to the maximum theoretical data rate (160 Mbits/s).

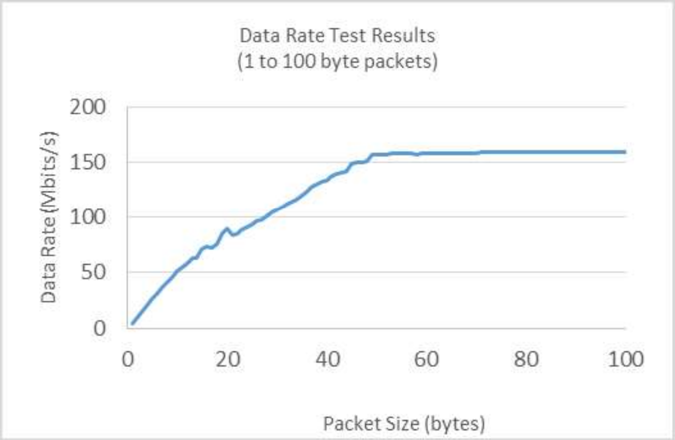


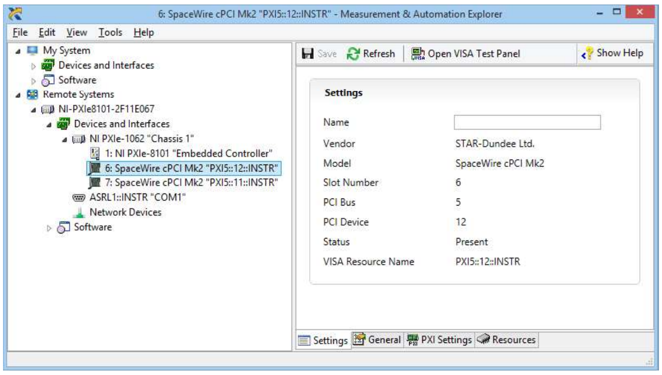
Fig.3. STAR-System wrapper loopback performance.

B. NI-VISA Driver

National Instruments VISA (NI-VISA) provides a standard programming interface between hardware and development environments such as LabVIEW [5]. NI-VISA is supported across the National Instruments product line.

The STAR-Dundee SpaceWire NI-VISA driver has been implemented as a native LabVIEW driver, providing support for the STAR-Dundee PCI family of devices. Software written to control these devices may be deployed on any hardware platform that supports cPCI/PCI and NI-VISA, including both Windows based hosts and LabVIEW Real-Time targets, without requiring modifications to source code. The software is provided as LabVIEW source with password protected block diagrams, allowing users to compile for any target.

The driver allows STAR-Dundee SpaceWire PCI cards to be detected with and controlled by National Instruments' MAX (Measurement and automation explorer) tool (Fig. 4).



My System
- Devices and Interfaces
- Software

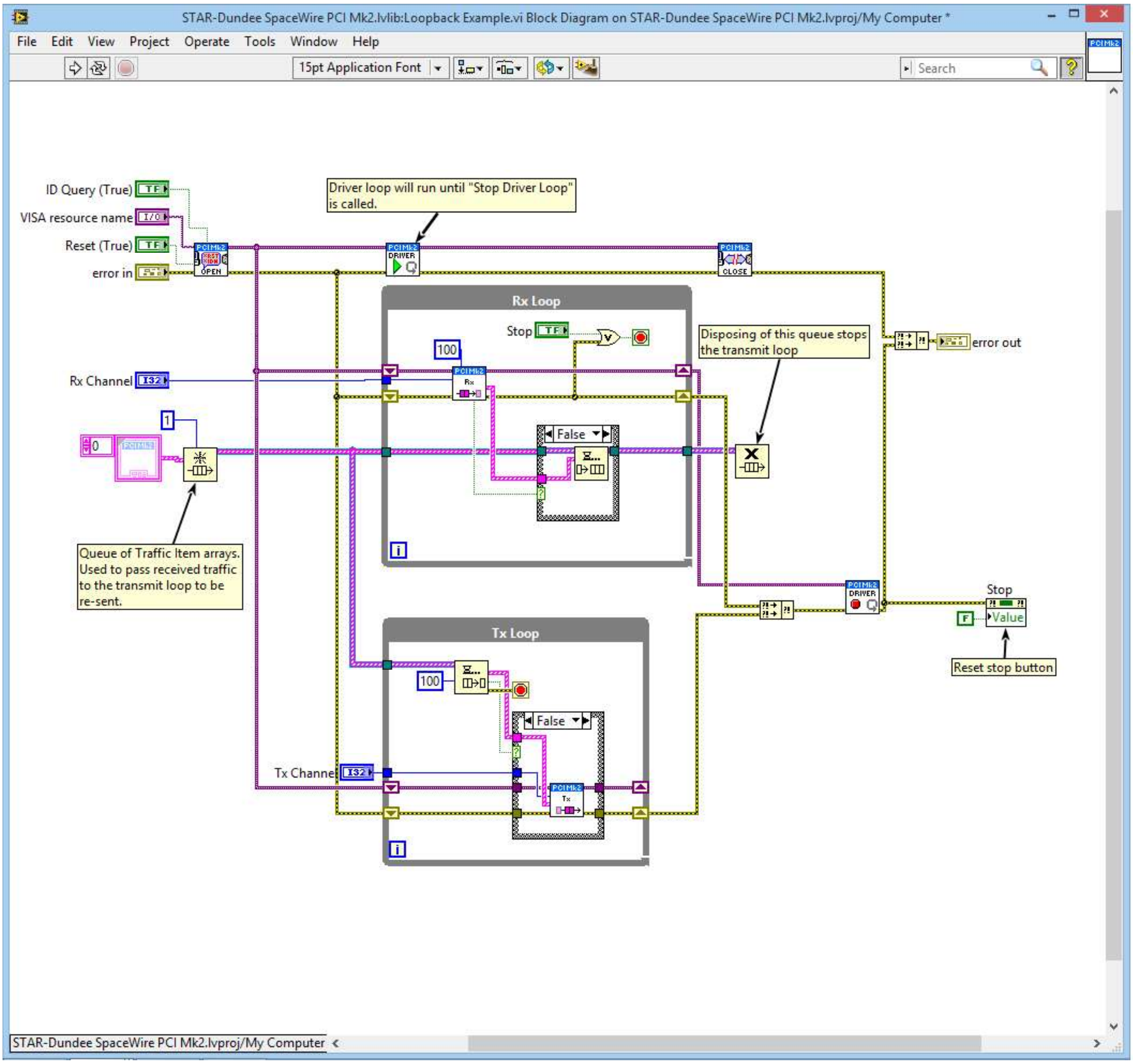
Remote Systems
- NI-PXIe8101-2F11E067
  - Devices and Interfaces
    - NI PXIe-1062 "Chassis 1"
      - 1: NI PXIe-8101 "Embedded Controller"
      - 6: SpaceWire cPCI Mk2 "PXI5::12::INSTR"
      - 7: SpaceWire cPCI Mk2 "PXI5::11::INSTR"
    - ASRL1::INSTR "COM1"
    - Network Devices
  - Software

Settings

Name	
Vendor	STAR-Dundee Ltd.
Model	SpaceWire cPCI Mk2
Slot Number	6
PCI Bus	5
PCI Device	12
Status	Present
VISA Resource Name	PXI5::12::INSTR

Fig. 4. MAX displaying chassis with cPCI cards

The driver has been designed to be intuitive to work with for LabVIEW users. For example, device access follows the familiar "Open, Perform Action, Close" architecture, with LabVIEW arrays used to pass SpaceWire data to transmit and receive functions. Figure 5 demonstrates the ease of use of this API. This example implements a software loopback device: packets are received on one port of the device, and are then looped back out of another. One could easily extend this example into a useful tool by inspecting the received traffic and permuting it in some way, perhaps by inserting or removing time codes, or injecting errors, before re-transmitting out the other port.



STAR-Dundee SpaceWire PCI Mk2.lvlib:Loopback Example.vi Block Diagram on STAR-Dundee SpaceWire PCI Mk2.lvproj/My Computer *

Driver loop will run until "Stop Driver Loop" is called.

Disposing of this queue stops the transmit loop

Queue of Traffic Item arrays. Used to pass received traffic to the transmit loop to be re-sent.

Reset stop button

STAR-Dundee SpaceWire PCI Mk2.lvproj/My Computer

Fig. 5. LabVIEW source code example showing a software loopback application.

IV. Future work

The STAR-System wrapper for LabVIEW supports all functionality provided by the current STAR-System libraries. This wrapper will be continuously upgraded to support any new functionality and released at the same time as new STAR-System releases

The NI-VISA driver is currently capable of transmitting and receiving SpaceWire packets, and configuring SpaceWire links. The RMAP packet library (already provided with STAR-System) will be ported to native LabVIEW code allowing it to be used with the NI-VISA driver on LabVIEW RT targets. Error injection support will also be added, allowing a user to inject, for example, a parity error on a given byte in a data stream, along with all the device configuration operations offered by the STAR-System API. Currently only the cPCI/PCI Mk2 cards are supported by this driver, but a USB driver could be quickly developed by re-using the existing top level API.

V. Conclusion

LabVIEW is a software development platform that allows for rapid development of test and measurement applications. Users of STAR-Dundee SpaceWire equipment can leverage the features of LabVIEW by using ready-built SpaceWire wrapper libraries and drivers in order to reduce the time and cost of developing test and verification tools.

References

[1] National Instruments, http://www.ni.com/labview/ LabVIEW, National Instruments Website.

[2] STAR-Dundee, http://star-dundee.com/sites/default/files/STAR-System.pdf, STAR-System Datasheet, STAR-Dundee Website.

[3] STAR-Dundee, http://www.star-dundee.com/products/star-system-labview, STAR-System for LabVIEW, STAR-Dundee Website.

[4] STAR-Dundee, http://star-dundee.com/knowledge-base/star-system-performance, STAR-System Performance, STAR-Dundee Website.

[5] National Instruments, http://www.ni.com/visa/ National Instruments VISA, National Instruments Website.