DYNAMIC ENGINEERING
150 DuBois St. Suite B&C Santa Cruz CA 95060
831-457-8891
https://www.dyneng.com
sales@dyneng.com
Est. 1988

Software User's Guide (Linux)

SpaceWire

Four-Channel SpaceWire Interface
K, BK, BK-DDR Models

SpaceWire

Dynamic Engineering
150 DuBois St Suite B&C
Santa Cruz, CA 95060
831-457-8891

Revised 06/13/24

Dynamic Engineering has made every effort to ensure that this manual is accurate and complete. Still, the company reserves the right to make improvements or changes in the product described in this document at any time and without notice. Furthermore, Dynamic Engineering assumes no liability arising out of the application or use of the device described herein.

The electronic equipment described herein generates, uses, and can radiate radio frequency energy. Operation of this equipment in a residential area is likely to cause radio interference, in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

Dynamic Engineering's products are not authorized for use as critical components in life support devices or systems without the express written approval of the president of Dynamic Engineering.

Connection of incompatible hardware is likely to cause serious damage.

Table of Contents

Product Description

The SpaceWire I/O card is available in multiple formats. All variants support four channels or ports of the SpaceWire protocol. Various configurations are available including addition of external FIFOs to support further burst capability.

This driver supports SpaceWire Xilinx Design Rev 10 or greater, and all BK revisions.

For a detailed description of the hardware including register definitions, see HW User Manual, SpaceWire.

Software Description

The SpaceWire driver supports simultaneous operation of all ports independently. The driver auto-detects the presence of external FIFOs and DDR on a per port basis and controls the HW accordingly.

The version of this driver is v1.1.9. The driver has been validated on an i7 Ubuntu Desktop running 5.4.0-150-generic (64 bit) SMP.

Installation

To install the de_SpaceWireLnxDriver, first navigate to the SpaceWireDriver/build directory within the downloaded repository.

You can do this by opening a terminal and using the command **cd path_to_downloaded_directory/de_SpaceWireLnxDriver/SpaceWireDriver/build**, replacing path_to_downloaded_directory with the actual path where the repository was downloaded.

Once you are in the build directory, compile the driver by executing the **sudo make** command. After the compilation is complete, install the driver by running the command **sudo ./Install_sw**. This will finalize the installation of the **de_SpaceWireLnxDriver,** and will create a device node in /dev/deSpWr_x with 1 for each port (note the script arbitrarily makes 11 device nodes, but only 0-3 work with a single card, if there are multiple cards, then the next 4 work for the second card, etc.

Three PLL files are defined in the driver. A module parameter, pll_freq determines which file is programmed when driver is installed. If the value of this parameter is not set, or value is greater than 2, 200 MHz file will be programmed. The delivered bnm script sets pll_freq = 0. Valid values are as follows:

pll_freq = 0, frequency = 200 MHz
pll_freq = 1, frequency = 180 MHz
pll_freq = 2, frequency = 100 MHz

Application Programming model

The following is the applicable section from the SpaceWire specification ECSS-E-ST-50-12C:

4.8 Application programming interface

The application programming interface (API) is not defined in this Standard. However, a typical application interface comprises the following services:

- **Open link**: Starts a link interface and attempts to establish a connection with the link interface at the other end of the link.
- **Close link**: Stops a link and breaks the connection.
- **Write packet**: Sends a packet out of the link interface.
- **Read packet**: Reads a packet from the link interface.
- **Status and configuration**: Reads the current status of the link interface and sets the link configuration.

The Dynamic Engineering driver implements this functionality as follows:

1) Upon first Linux open, a default configuration is applied. The port is configured in internal loopback, thus external link access is disabled. Link will remain disconnected.
2) Port must be configured via DE_CONFIG_PT ioctl to enable link and read/write access to remote end of the link. If the link connection is not established in 4 seconds, the ioctl will return failure. Failure status may be interrogated via the Linux perror invocation.
3) Link status may be interrogated via DE_GET_STATS ioctl. Current link state as well as accumulated error counts and I/O byte counts are returned.
4) Read/write implemented via standard Linux read/write APIs. If either fails, cause may be determined via Linux perror invocation.
5) Upon last Linux close of a port, the port is reset and link is disconnected.

Blocking and non-blocking modes of operation are supported. This behavior is set via the standard file flags upon open. Further, a blocking read timeout maybe specified via the config ioctl.

Besides blocking read timeouts, the following parameters are set via the config ioctl: time code generation, DMA priority, packet mode (enable/disable), auto start enable. and link speed in MHz.

Configuration Considerations

Reads and writes up to the size of the port FIFOs are supported by default. Larger packet sizes are no longer supported for either packet or non-packet modes as this functionality is no longer deemed useful. One needed to know the expected receive packet size in advance.

Two types of link connection/establishment are supported, manual and auto-start. The value of this parameter is determined by the requirements/implementation of the external device. Nominally, this parameter should be set to 0 (manual start). If problems establishing the link are encountered, set this parameter to 1 (auto-start). This allows the other end of the link to initiate the link start.

An option for an interrupt driven time code C ioctl has been implemented. The opt field in the de_tc_cmd_t determines behavior. If opt is to zero, the current TC value is returned. If opt is set to one, the TC interrupt is enabled, and the TC value is read upon TC interrupt.

Please see de_SpwrDrv.h for details of the parameters for this ioctl.

Sample application

Two sample applications (de_IoApp.c, de_IoctlApp.c) are provided to demonstrate configuration, ioctl invocation, and I/O in the supported modes.

1) Compile the sample application for your platform, the output executable for these examples are dyn_io and dyn_ioctl.
    a. Nominal compilation gcc
        gcc –Wall –o dyn_io de_IoApp.c -pthread
        gcc –Wall –o dyn_ioctl de_IoctlApp.c
        The apps should compile without warnings, it is assumed de_SpwrDrv.h, de_SpwrCfg.h, and de_PllDefs.h are resident in the same directory as the applications for these examples.

Invocation parameters

I/O application invocation is as follows:

./dyn_io e(xternal)|i(nternal) port(0-3) frame_len(bytes) r(eader)|w(riter)

The first parameter specifies the I/O mode (external I/O or internal loopback). The second parameter specifies the port to be exercised. Frame length is specific in bytes. The final parameter (r|w) is only applicable to external I/O.

For internal loopback the application will transmit and receive on the same port. Thus, only one instance of the app is required to exercise both Tx and Rx. The last parameter is ignored in the case of internal loopback.

In external I/O mode 2 instances of the app must be invoked to exercise both the read and write ports. It is assumed the two ports are connected directly or via a switch.

In either mode, Rx data is compared to Tx data upon read completion for each iteration. The same frame size must be specified for the port pair being exercised.

Optional parameters:
In addition to parameters listed above, two optional parameters may be specified. By default 100,000 test iterations are completed. Optional parameter 5 overrides the default value.

Default link speed is 200 MHz, optional parameter 6 overrides the default speed. Link speed cannot be set higher than the PLL file loaded upon installation of driver. Further link speed must be greater than (PLL frequency) /16. For example, if 200 MHz PLL was programmed, then minimum link speed is 13 MHz. If link speed is too high or low, config ioctl will fail.

Link speed specifies the transmit speed, Rx speed is determined by transmitter on the other end of link.

Ioctl application invocation is as follows:
./dyn_ioctl

A menu will be displayed:
Enter p(ll program)||r(eg ops)|l(ed ops)|e(xit)

The sample jed file (fpcisw_180.jed) is included in this distribution. The sample app assumes this file is resident in the same directory as the executable.

The Ioctl application demonstrates pll programming, register R/W/RMW operations, and lane steering control.

Debug

If an error is encountered while running Dynamic Engineering applications, please perform the following steps. This information is required by Technical Support to resolve any issues.

1) Execute modinfo de_SpwrDrv.ko
   Driver version as well as other information will be displayed.
   Note the version.
2) Execute dmesg
   This command will display error messages logged by the driver.
   Dynamic Engineering drivers are quite verbose logging errors and cause.
   Take a screen shot of the output.
3) Contact Technical Support with information collected in steps 1 and 2.

Warranty and Repair

Please refer to the warranty page on our website for the current warranty offered and options.

https://www.dyneng.com/warranty.html

Service Policy

Before returning a product for repair, verify as well as possible that the suspected unit is at fault. Then call the Customer Service Department for a RETURN MATERIAL AUTHORIZATION (RMA) number. Carefully package the unit, in the original shipping carton if this is available, and ship prepaid and insured with the RMA number clearly written on the outside of the package. Include a return address and the telephone number of a technical contact. For out-of-warranty repairs, a purchase order for repair charges must accompany the return. Dynamic Engineering will not be responsible for damages due to improper packaging of returned items. For service on Dynamic Engineering Products not purchased directly from Dynamic Engineering contact your reseller. Products returned to Dynamic Engineering for repair by other than the original customer will be treated as out-of-warranty.

Out of Warranty Repairs

Out of warranty repairs will be billed on a material and labor basis. Customer approval will be obtained before repairing any item if the repair charges will exceed one half of the quantity one list price for that unit. Return transportation and insurance will be billed as part of the repair and is in addition to the minimum charge.

For Service Contact:

Customer Service Department
Dynamic Engineering
150 DuBois St. Suite B&C
Santa Cruz, CA 95060
831-457-8891
InterNet Address support@dyneng.com