SRIO Driver

Release Notes

Applies to Product Release: 01.00.01.10
Publication Date: September 13, 2012

Texas Instruments, Incorporated
20450 Century Boulevard
Germantown, MD 20874 USA

Contents

Release Notes

SRIO Driver version 01.00.01.10

Overview

This document provides the release information for the latest SRIO driver which should be used by drivers and application that interface with SRIO IP.

SRIO Driver module includes:

- Compiled library (Big and Little) Endian of SRIO Driver.
- Source code.
- API reference guide
- Design Documentation

LLD Dependencies

LLD is dependent on following external components delivered in PDK package:

- CSL
- CPPI LLD
- QMSS LLD

New/Updated Features and Quality

Release 1.0.1.10:

- Bug fixes. Refer Resolved IRs section below.

Release 1.0.1.9:

- Padding added to global data structures to resolve cache coherency issues. [SDOCM00091801]

Release 1.0.1.8:

- Added ability to the driver to specify having the hardware use all available mailboxes on receive side. [SDOCM00084668]
- Fixed the issue with enabling TSC timing functions in Loopback DIO ISR example project. [SDOCM00090657]
- Added a SRIO LLD benchmarking application to measure throughput & latency for Type-11, DIO NWRITE and DIO NREAD transfers. [SDOCM00084008]

- Added a check in `Srio_allocTransmitBuffer()` to validate returned buffer pointer in Multicore Example project. [SDOCM00083561]

Release 1.0.1.7:

- Fixed an issue which was causing SRIO Loopback DIO ISR example project to fail on Big Endian across all devices. [SDOCM00088341]
- Removed un-used code from SRIO Loopback DIO ISR example project.
- Reduced TXU priority from High to Low in the SRIO Loopback test project as message response priority has to be 1 higher than the transmit message to prevent deadlock.
- Fixed the mismatched types issue (`uint16_t` vs. `uint32_t`) in extraction of doorbell "reg" and "bit" fields. [SDOCM00085440]

Release 1.0.1.6:

- Enabled all 16 Rx channels at init time because the SRIO picks a channel in a round-robin fashion.
- Added a configuration parameter to set scheduling priority for SRIO PktDma Tx channel.
- Updated documentation to indicate that driver expects accumulator lists to be allocated from local (un-cached) memory due to performance reasons.
- Removed the hard coding of byte size in the accumulator list address calculation and made it a function of list entry size.

Release 1.0.1.5:

- Added an include file in example project to provide platform specific configurations.
- Modified `SRIOLoopbackDioIsr` example project to support devices with fewer number of cores.

Release 1.0.1.4:

- Additional device support

Release 1.0.1.3:

- Fix for driver not performing writeback of descriptor sitting in cache when using App Managed config

Release 1.0.1.2:

- SRIO modes (Type11, Type9 and DIO) specific code in common functions are separated into individual mode specific functions
- Enhanced driver to support processing of DIO ISR to get transaction completion code

Release 1.0.1.1:

- Driver support for hardware assigned Letter field for Type11 message
- Support for same TX queue for multiple driver instances

- Added `#pragma CODE_SECTION` to driver functions to allow code placements in different memory sections

Release 1.0.1.0:

- Added a new example demonstrating interrupt at the end of Direct IO write/read
- Bug fixes (refer Resolved IRs section)

Release 1.0.0.14:

- Build Infrastructure support for Makefiles.

Release 1.0.0.13:

- Deprecated support for C64P ELF and COFF. Only C66 ELF is supported now.
- Extended DIO socket support
    - Deprecated the `Srio_sockSendDoorbell` API. Use the `Srio_sockSend` API for DIO sockets to send doorbells
    - Use the `Srio_sockRecv` API to receive doorbells.
    - Added a new handler for handling DIO completion interrupts `Srio_dioCompletionIsr`. Applications need to ensure that this is plugged with their interrupt managed routines or can be called in polling mode.
    - Blocking and Non-blocking support for DIO sockets.
    - New socket options Srio_Opt_DIO_SOCK_COMP_CODE & Srio_Opt_REGISTER_DOORBELL are added.
- OSAL extensions to ensure descriptors are invalidated & written back if they are modified.
    - `Srio_osalBeginDescriptorAccess`
    - `Srio_osalEndDescriptorAccess`
- Changes for limiting doxygen requirement only during the release
- Copyright modification to TI BSD
- SIMULATOR_SUPPORT is disabled by default for the library being included for examples to run on EVM.

Release 1.0.0.12:

- Renamed the test and example project files to be compliant to execute with the PDK Project creation script.
- OSAL Fixed in the Test and Example to ensure that BIOS Memory_alloc is not invoked from ISR context.
- Fixed a bug in the DIO socket binding to ensure that the correct status flag was updated.

Release 1.0.0.11:

- The definition SIMULATOR_SUPPORT has been added to differentiate between the driver dependencies between the simulator and the device. Please ensure that all test and example code is built with this definition. All pre-built libraries are compiled with this flag switched off so they will work by default on the simulator.

Release 1.0.0.10:

- The `csl_srioAuxTundra.h` was renamed to `csl_srioAuxPhyLayer.h`.

- The function `Srio_processReceivedBD` has now been exposed to the application and can be now used by applications which handle SRIO interrupt by themselves.

Release 1.0.0.9:

- C66 Target support
- SRIO Driver has been validated on QT for the following features
    - Type11
    - Type9
    - DIO

  The driver should be recompiled with the QT_DEBUG compilation flag to build the SRIO driver for QT.
- Modifications to support the new CPPI (1.0.0.11) and QMSS (1.0.0.11) LLD

Release 1.0.0.8:

- Added ELF & COFF support.
- OSAL API have been extended:
    - Cache Hooks added to the driver for CX Simulator
    - Critical Section Hooks have been modified to differentiate between
        - Single Core
          This OSAL hook is required to protect the resources from access on a single core but between multiple threads.
        - Multi Core
          This OSAL hook is required to protect shared resources from access across multiple cores.
    - Memory Allocation/Cleanup hooks have been modified to differentiate between
        - Control Path
          These allocations are done during initialization and control path
        - Data Path
          This is applicable only for Driver Managed configuration and is used in the data path

      The hooks will allow application developers to plug a fast OSAL implementation for data path allocations.
- Driver Managed Configuration now exposes the accumulator configuration to the application.
- Added new RAW exported cleanup API in the Application Managed Configuration which needs to be provided by the application to free received data.
- SRIO Device Initialization code has been removed from the prebuilt library. Applications now need to ensure that they initialize the SRIO IP block before calling the SRIO driver API's.
- Test and Example code updated to use the Cache hooks for the CX Simulator.
- Updated to use the new CPPI and QMSS Library 1.0.0.10

Release 1.0.0.7:

- Added ELF support. Prebuilt driver libraries are ELF only.
- Fixed compilation warnings in the test project.
- Fixed compilation error in SRIO Initialization sequence for QT builds.

Release 1.0.0.6:

- Direct IO Support added
    - The SRIO driver is extended to handle the DIO sockets. The support has NOT been tested since the simulator does not support this functionality. DIO support in the driver is experimental and is subject to change in the future.
- Type9 Support added
    - The SRIO driver is extended to handle the Type9 sockets. The support has NOT been tested since the simulator does not support this functionality. Type9 support in the driver is experimental and is subject to change in the future.
- C99 Types
    - The SRIO driver has been modified to use the C99 types from the previous implementation which used XDC types.
- SRIO Driver modified to reflect CSL include path change
- Modifications to support the new CPPI & QMSS Version 1.0.0.8 Libraries.
- Updated SRIO Driver Initialization sequence for QT.
- SRIO Driver was tested on QT. The driver test works in polled mode. The driver has *not* been verified for interrupt support & Multicore.
- Fixed IR - SDOCM00068684 NySh SRIO LLD: Receive configuration errors in `srio_drv.c`

Release 1.0.0.5:

- Modifications to support the new CPPI & QMSS Version 1.0.0.5 Libraries.

Release 1.0.0.4:

- Modifications to the "test" & "example" configuration files to support the whole program build profile.

Release 1.0.0.3:

- Modifications to the driver to support the new CPPI specification (4.2.9)
- Support for RAW Sockets.
- Support for Interrupts.
- Extended configuration support for applications.

Release 1.0.0.2:

- Modifications to the driver to support the new CPPI specification (4.2.7)

Release 1.0.0.1:

- Multi-core support

Release 1.0.0.0:

- Initial Release

Resolved Incident Reports (IR)

Table 1 provides information on IR resolutions incorporated into this release.

Table 1 Resolved IRs for this Release

IR Parent/ Child Number	Severity Level	IR Description
SDOCM00093407	Major	OSAL defines memcpy and so on by itself
SDOCM00095787	Major	C6657: SRIO projects do not link properly

Known Issues/Limitations

Table 2 Known Issue IRs for this Release

IR Parent/ Child Number	Severity Level	IR Description
SDSCM00036978	Major	The SRIO driver test and example projects do not work correctly with the CX simulator. There is an issue with the CX simulator where on reception data from one core is placed into the receive queue of another core.

Licensing

Please refer to the software Manifest document for the details.

Delivery Package

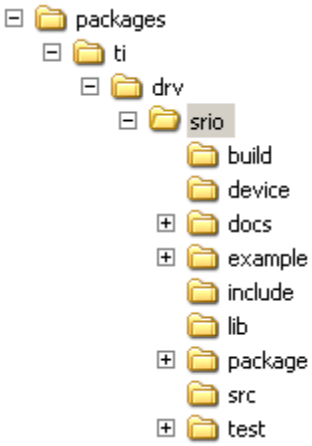
There is no separate delivery package. The SRIO Driver is being delivered as part of PDK.

Installation Instructions

The LLD is currently bundled as part of Platform Development Kit (PDK). Refer installation instruction to the release notes provided for PDK.

Directory structure

The following is the directory structure after the SRIO driver package has been installed:



The following table explains the contents of the SRIO package:-

Directory Name	Description
`ti/drv/srio`	The top level directory contains the following:- 1. *XDC Build and Package files* These files (`config.bld, package.xdc` etc) are the XDC build files which are used to create the SRIO package. 2. *Exported Driver header file* Header files which are provided by the SRIO driver and should be used by the application developers for driver customization and usage.
`ti/drv/srio/build`	The directory contains internal XDC build related files which are used to create the SRIO Driver package.
`ti/drv/srio/device`	The directory contains the device specific files for the SRIO device driver.
`ti/drv/srio/docs`	The directory contains the SRIO driver documentation.
`ti/drv/srio/example`	The "`example`" directory in the SRIO driver has a usage example which explains how the SRIO driver API's are used to send and receive data.
`ti/drv/srio/include`	The "`include`" directory has private SRIO driver header files. These files should not be used by application developers.
`ti/drv/srio/lib`	The "`lib`" folder has pre-built Big and Little Endian libraries for the SRIO driver along with their *code/data size information*.
`ti/drv/srio/package`	Internal SRIO driver package files.
`ti/drv/srio/src`	Source code for the SRIO Driver.

Test and Example

The section documents information about the test and example code located in the SRIO driver.

SRIO Loopback Test

The unit test project provided in the SRIO driver is used by the development teams for validating the SRIO driver. The test code runs on all 4 cores and executes on a single Nyquist by configuring the SRIO to operate in loopback mode.

The test code tests the following functionality of the SRIO driver

- *Non Blocking RAW Sockets in polled mode*
  The test case verifies data transfers using Type11 messages over RAW sockets in non-blocking mode. The driver instance is configured to be operating in polled mode. The test case polls for received data and validates the data to ensure correctness.
- *Normal Non blocking Sockets in interrupt mode*
  The test case verifies data transfers using Type11 messages over Normal sockets in non blocking mode. The driver instance is configured to be operating in interrupt mode. The test case ensures that the data is received and validated to ensure correctness.
- *Normal Blocking Sockets in interrupt mode*
  The test case verifies data transfers using Type11 message over Normal sockets configured in blocking mode. The test case starts a producer and consumer thread in which the consumer thread is blocked waiting for data to be received. The producer thread sends a block of data using Type11 message and the test case ensures that the driver wakes up the consumer thread on the reception of the data. The consumer is responsible for data verification.
- *Multicore test*
  The test case runs on 4 cores. Each core is executing a SRIO driver instance and participates in sending and receiving data. Data is sent as per the following chain
  CORE 1 → CORE 2 → CORE 3 → CORE 0 → CORE 1
  The test case ensures that the SRIO Driver API can be used across multiple cores. The test case uses Normal Non-blocking sockets in interrupt mode & Type11 messages for data transfers. Each core ensures that the received data is validated.

Multicore tests are selected by ensuring that the `TEST_MULTICORE` option is defined in the pre-defined symbols. Multicore tests can only be run if 4 cores are synchronized and the resulting image file is loaded on 4 cores.

Note: To execute on the EVM; please ensure that you power cycle the EVM for every run of the test.

SRIO Example

The example project is provided to test the pre-built libraries which are provided by the SRIO LLD and to ensure that these libraries are validated.

The example project runs on 4 cores. Each core is executing a SRIO driver instance and participates in sending and receiving data. Data is sent as per the following chain

CORE 1 → CORE 2 → CORE 3 → CORE 0 → CORE 1

The test case ensures that the SRIO Driver API can be used across multiple cores. The test case uses Normal Non-blocking sockets in interrupt mode & Type11 messages for data transfers. Each core ensures that the received data is validated.

> **Note:** To execute on the EVM; please ensure that you power cycle the EVM for every run of the example.

Customer Documentation List

Table 3 lists the documents that are accessible through the **/docs** folder on the product installation CD or in the delivery package.

Table 3 Product Documentation included with this Release

Document #	Document Title	File Name
1	API documentation (generated by Doxygen)	docs/srioDocs.chm
2	Design Document	docs/SRIO_SDS.pdf
3	Software Manifest document	Docs/ SRIO_LLD_SoftwareManifest.pdf