Application Report

SPRACE9A–May 2018–Revised January 2019

PRU-ICSS / PRU_ICSSG Getting Starting Guide on Linux

Nick Saulnier and Jason Reeder

Catalog Processors

ABSTRACT

This application report provides a getting started guide for the PRU-ICSS / PRU_ICSSG of the Sitara™ family of devices while running Linux on the Arm® cores.

Contents

Trademarks

1 Overview

1.1 PRU-ICSS Description

The Programmable Real-time Unit and Industrial Communication SubSystem (PRU-ICSS) consists of dual 32-bit RISC cores (Programmable Real-time Units, or PRUs), shared data, instruction memories, internal peripheral modules, and an interrupt controller (INTC). The programmable nature of the PRU, along with its access to pins, events and all system-on-chip (SoC) resources, provides flexibility in implementing fast real-time responses, specialized data handling operations, custom peripheral interfaces, and in offloading tasks from the other processor cores of the SoC.

Figure 1 shows the details of the PRU-ICSS found in the AM335x device.

The PRU cores are programmed with a small, deterministic instruction set. Each PRU can operate independently or in coordination with each other and can also work in coordination with the device-level host CPU. The interaction between processors is determined by the nature of the firmware loaded into the PRU's instruction memories.

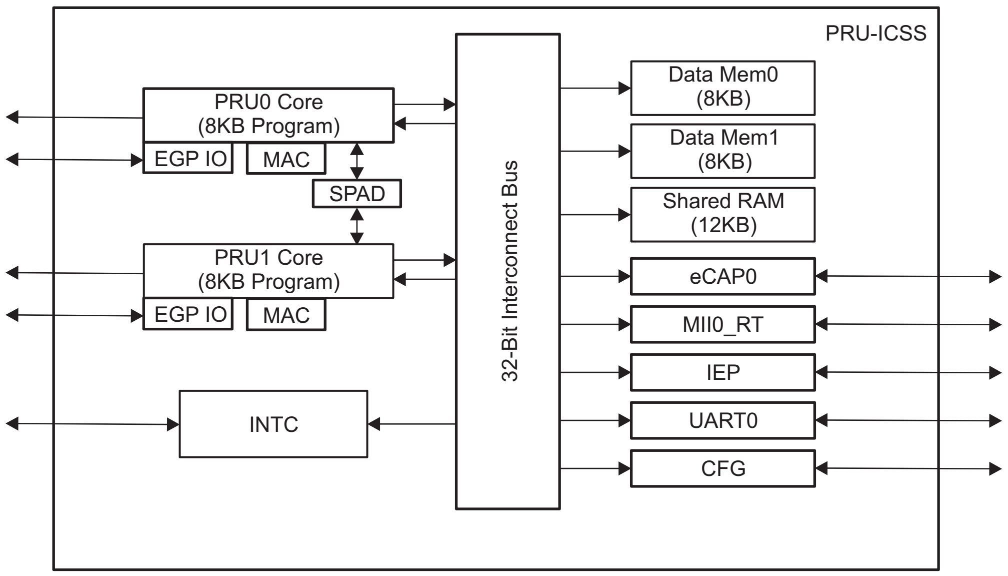

Figure 1. PRU-ICSS Block Diagram

1.2 PRU_ICSSG Description

The Programmable Real-time Unit and Industrial Communication SubSystem - Gigabit (PRU_ICSSG) can be considered a superset of the PRU-ICSS. In addition to all PRU-ICSS features, the PRU_ICSSG adds two Auxiliary Programmable Real-Time Unit (RTU) cores, broadside memories, improved event management with the task manager, data processing and data movement accelerators, and new peripherals such as PWM.

Figure 2 shows a simplified block diagram of a PRU_ICSSG. The PRU and RTU cores and resources are divided between two identical slices within the PRU_ICSSG.

The processing core of an RTU is the same as a PRU, but RTUs have different resources, connections, and accelerators than PRUs. For example, PRUs have access to external general purpose input and general purpose output (GPI/GPO) pins, while RTUs do not. The rest of this document refers to all of the PRU_ICSSG cores as "PRU" rather than specifying "PRU and RTU".

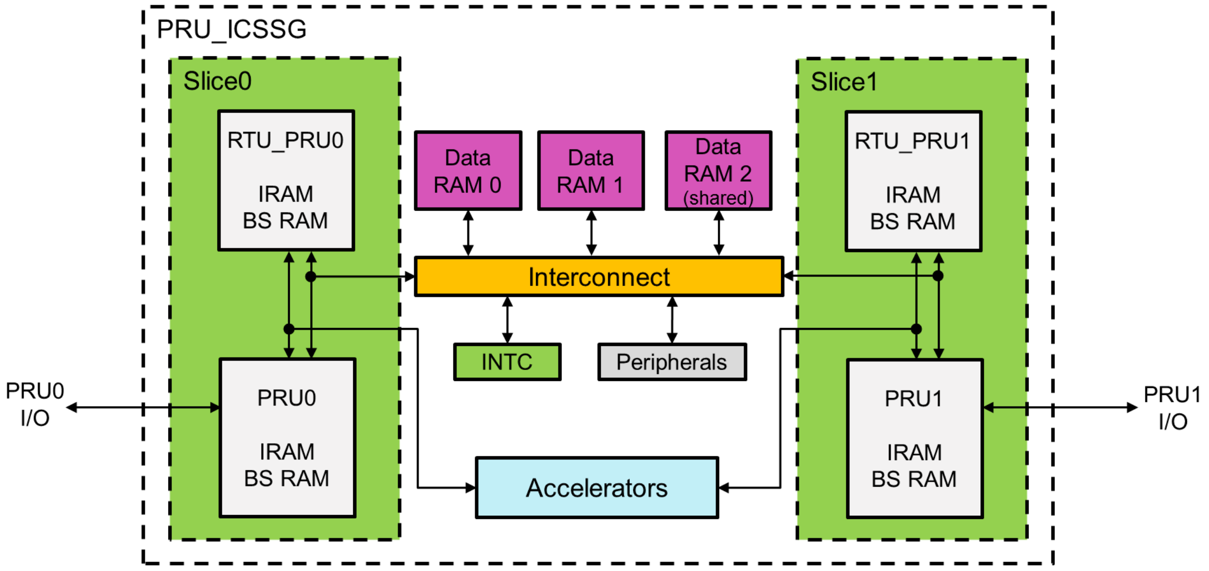

Figure 2. PRU_ICSSG Block Diagram

1.3 *Devices Containing the PRU-ICSS / PRU_ICSSG*

The PRU-ICSS / PRU_ICSSG can be found throughout the Sitara family of devices from Texas Instruments (TI). The PRU-ICSS / PRU_ICSSG is slightly different in each device. An overview of the differences between PRU-ICSS / PRU_ICSSG in each device can be found in the *PRU-ICSS / PRU_ICSSG Feature Comparison*. Detailed analysis of the differences between PRU-ICSS / PRU_ICSSG in each device can be found in the *PRU-ICSS / PRU_ICSSG Migration Guide*.

- The number of PRU-ICSS / PRU_ICSSG's in a device (some devices have more than one)
- The size of the internal memories (instruction RAM, data RAM, and shared RAM)
- The number of general purpose input/output pins available externally

1.4 *What Can the PRU-ICSS / PRU_ICSSG Do?*

1.4.1 Real-Time Industrial Communication

The PRU-ICSS was designed with real-time industrial communications in mind. Ethernet, Fieldbus, interface, and redundancy protocols are available for the PRU-ICSS, including certified solutions for EtherCAT®, PROFIBUS®, PROFINET®, HSR and PRP among others. The PRU-ICSS can support multiple protocols on the same hardware using firmware supplied by TI and stacks from industry partners. The PRU_ICSSG adds gigabit industrial communication capabilities, in addition to the capabilities offered by PRU-ICSS.

For industrial communication protocols, TI, or one of its industry partners, will provide the PRU firmware to be loaded. In this type of use case, the PRU-ICSS / PRU_ICSSG appears to the high level operating system (Linux or RTOS) as another Ethernet peripheral. Refer to *Industrial Communication Protocols Supported on Sitara™ Processors* for a list of all protocols that are currently supported on the PRU-ICSS / PRU_ICSSG.

1.4.2 General Purpose Use Cases

The PRU cores were created to be non-pipelined, single-cycle execution engines. This means that all instructions (except for memory reads and writes) complete in a single cycle. Since there is no pipeline, the PRU cores are completely deterministic: the PRU cores will not rearrange instructions during execution. If the PRU firmware has a critical loop that was written to complete in 15 cycles, it always completes in 15 cycles. The PRU_ICSSG adds a Task Manager, which can preempt currently running code. The Task Manager allows firmware to meet timing requirements for high priority tasks by interrupting lower priority tasks.

The PRU-ICSS / PRU_ICSSG can also read from or write to external pins in a single cycle using a PRU core register. With a PRU core running at 200 MHz, this means that an external signal change can be detected in as little as 5 nanoseconds! This also means that the PRU can toggle external pins with a granularity of 5 nanoseconds. PRU_ICSSG cores running at 250 MHz can toggle external pins with a granularity of 4 nanoseconds.

The combination of deterministic PRU cores along with direct access to GPIO means that the PRU-ICSS / PRU_ICSSG opens up many interesting uses cases to Sitara SoCs.

1.5 PRU-ICSS / PRU_ICSSG Usage Environments

In the context of the SoC, there are three environments in which the PRU-ICSS / PRU_ICSSG is used: Code Composer Studio™ (CCS), TI RTOS running on the Arm core(s), or Linux running on the Arm core(s).

1.5.1 Code Composer Studio

Code Composer Studio (CCS) is an Integrated Development Environment (IDE) that supports TI's Microcontroller and Embedded Processors portfolio. Using CCS, the PRU-ICSS / PRU_ICSSG can be loaded, run, and fully debugged just like any other core in the SoC. This environment is very useful during the development and debug stages of creating a stand-alone PRU firmware.

1.5.2 TI RTOS

TI RTOS, based on SYS/BIOS, is Texas Instrument's Real-Time Operating System (RTOS) that can be used as the high-level operating system on the processors of Sitara devices (Cortex-A or C66x DSP). In this environment, an RTOS driver is provided to load the PRU-ICSS / PRU_ICSSG memories and run the PRU cores. The PRU firmware files are expected to be built and then converted into an array format that the RTOS driver can use.

1.5.3 Linux

TI creates its own Linux distribution and packages it into the Processor SDK Linux. In this environment, the Linux RemoteProc framework treats the PRU cores inside the PRU-ICSS / PRU_ICSSG as remote processors. The PRU firmware files are expected to be built into elf binaries and placed into the Linux file system. The remainder of this application report will focus on using the PRU-ICSS / PRU_ICSSG in the Linux environment.

2 PRU-ICSS / PRU_ICSSG Getting Started With Linux

As mentioned previously, there are two types of use cases for the PRU-ICSS / PRU_ICSSG while running Linux on the Arm core:

- The first type of use case utilizes TI-provided PRU firmware for industrial communication protocols, such as HSR or PRP.
- The second type of use case utilizes building block examples to create general purpose applications that take advantage of the strengths of the PRU-ICSS / PRU_ICSSG.

Both of these types of use cases require the Processor SDK Linux.

2.1 TI Provided PRU-ICSS / PRU_ICSSG Firmware

In the Processor SDK Linux, TI provides prebuilt PRU-ICSS firmware to implement 100Mbps Ethernet ports as well as the redundancy protocols HSR and PRP. In addition to the PRU firmware provided, TI has also created and provided the Linux drivers necessary to use the PRU Ethernet ports from user space applications.

To learn more about the PRU Ethernet firmware and how to test it, see PRU-ICSS Ethernet.

To learn more about the HSR and PRP redundancy implementation, see Processor SDK Linux HSR PRP.

2.2 General Purpose Firmware

Before creating general purpose Linux PRU firmware, you must understand the basics of how the Linux RemoteProc framework works. The RemoteProc Linux driver is responsible for taking the PRU firmware from the Linux filesystem, parsing it for any resources that it must provide for the PRU (for example, interrupts or shared memory), loading the firmware into PRU instruction memory and data memory, and then running the PRUs.

For more information on the RemoteProc driver and how it works, see RemoteProc. A key thing to understand is that the resource table is necessary for the RemoteProc driver to work, even if it is empty.

User applications can require communication between the PRUs and the Arm core. Linux provides a method for this communication called RemoteProc Messaging (RPMsg). RPMsg fits into the RemoteProc framework. TI provides a RPMsg library and getting started examples to demonstrate how to use RPMsg. These examples showcase the PRU firmware using its resource table to request specific interrupt mappings and shared memory to implement the communication channel.

To learn more about how RPMsg works, see RPMsg.

For a quick start guide that walks through the RPMsg out-of-box example as well as how to rebuild/reload the example, see RPMsg Quick Start Guide wiki.

Once RemoteProc and RPMsg are understood, a good next step is to check out the TI-provided PRU-ICSS Hands-on Labs. These labs are written for a BeagleBone Black with a PRU Cape attached, but the concepts apply to the PRU-ICSS / PRU_ICSSG across the Sitara family. Labs 1-3 require CCS and do not use Linux. If CCS is of no interest, Labs 1-3 can be skipped and you can start from Lab4.

The PRU Software Support Package is the central location for all the previously mentioned general purpose firmware source code and libraries. The package is located in the ‘example-applications/pru-icss-x.y.z’ folder of the Processor SDK Linux installation. The git repository for the package is also available at PRU Software Support Package Git Repository. This package contains building block PRU examples for Sitara devices along with the necessary header files and libraries. The examples in the PRU Software Support Package may be used as a starting point for developing new general purpose firmware.

3 More Information

3.1 PRU C Compiler and Assembler

PRU firmware can be written using C code or assembly. C code is more approachable to new users, but it may not get the absolute best performance achievable. The recommendation is to use C code in your firmware as much as possible, and then see how well the provided C compiler optimizes your code. If further optimizations are necessary, then assembly code can also be used.

TI provides the PRU C compiler in the PRU Code Generation Tools package available in the ‘linux-devkit/sysroots/x86_64_arago-linux/usr/share/ti/cgt-pru/’ directory of the Processor SDK Linux. The C compiler is available as an add-on package for CCS. The compiler may also be downloaded from this link: PRU Code Generation Tools.

The PRU C compiler user guide is available at: *PRU Optimizing C/C++ Compiler User's Guide*.

The PRU assembler user guide is available at: *PRU Assembly Language Tools User's Guide*.

The PRU assembly instructions list is available at: *PRU Assembly Instruction User's Guide*.

3.2 *TI Designs*

TI Designs are system-level hardware and software references for specific use cases. The following TI Designs are PRU-ICSS examples where Linux is running on the Arm core.

3.2.1 Industrial Protocol TI Designs (TI Provided Firmware)

High-Availability Seamless Redundancy Ethernet Reference Design for Substation Automation for Linux® – This reference design implements a solution for high-reliability, low-latency network communications for substation automation equipment in Smart Grid transmission and distribution networks. It supports the high-availability seamless redundancy (HSR) specification in the IEC 62439 standard.

Parallel Redundancy Protocol Ethernet Reference Design for Substation Automation on Linux – This Linux®-based reference design demonstrates high-reliability, low-latency network communications for substation-automation equipment in smart grid transmission and distribution networks. This reference design supports the Parallel Redundancy Protocol (PRP) specification in the IEC 62439 standard.

OPC UA Data Access Server for AM572x Reference Design – OPC UA is an industrial machine-to-machine protocol designed to allow interoperability and communication between all machines connected under Industry 4.0.

3.2.2 General Purpose TI Designs

Flexible Interface (PRU-ICSS) Reference Design for Simultaneous, Coherent DAQ Using Multiple ADCs – This TI design showcases the PRUs ability to implement parallel SPI ports to connect to (6) 8-channel ADCs simultaneously.

Simple PID Control Reference Design With PRU®-ICSS Through Web Interface – This design implements a simple PID loop for a small motor in the PRU. The data (set point, current speed, constants, etc.) is communicated to the Arm core through RPMsg and displayed on a webpage.

Thermal Printing with the PRU-ICSS on the BeagleBone Black Reference Design – A thermal printer driver is made using the GPIO of the PRU.

3.3 *E2E Forum Support*

Any questions not addressed by this PRU-ICSS / PRU_ICSSG Getting Started Guide for Linux can be posted on the Sitara E2E Forums.

4 Frequently Asked Questions

This section includes a few of the most common questions while using the PRU-ICSS / PRU_ICSSG with Linux on the Arm core. For a comprehensive list of Frequently Asked Questions on the PRU-ICSS / PRU_ICSSG, see the *PRU-ICSS FAQ* wiki.

4.1 *Do you Support the BeagleBoard Community Linux Distribution, the PASM Tool, or the Linux UIO Driver for PRU General Purpose Use Cases?*

No, TI does not support these.

The BeagleBoard Linux Distribution is supported by the BeagleBoard community on their forums: BeagleBoard Community Forums.

The PASM tool has been deprecated in favor of the PRU Code Generation Tools mentioned earlier.

The UIO driver for general purpose use cases has also been deprecated and is replaced by the RemoteProc driver.

4.2 Does the PRU-ICSS / PRU_ICSSG Support Interrupts?

Yes, but not in the same way that most cores support interrupts.

The PRU-ICSS / PRU_ICSSG contains an interrupt controller that can map 64 system events down to two flags that are set in a PRU core register (bits 30 and 31 in core register R31). The PRU core can then check each of these flags in a single cycle to see if an event has occurred. These flags can either be polled upon or checked periodically (dependent on what makes the most sense for the use case).

The PRU-ICSS / PRU_ICSSG interrupt controller does not support jumping the program counter of the PRU core to a pre-determined function when an event occurs.

4.3 What are the Read and Write Latencies to Different Memories?

The latencies associated with reading from and writing to different SoC memories are found in the *PRU Read Latencies* wiki.

5 References

- BeagleBoard Community Forums
- *PRU Optimizing C/C++ Compiler User's Guide*
- *PRU Assembly Language Tools User's Guide*
- *PRU Assembly Instruction User's Guide*
- *Flexible Interface (PRU-ICSS) Reference Design for Simultaneous, Coherent DAQ Using Multiple ADCs*
- *Parallel Redundancy Protocol Ethernet Reference Design for Substation Automation on Linux*
- *Thermal Printing with the PRU-ICSS on the BeagleBone Black Reference Design*
- *Flexible Interface (PRU-ICSS) Reference Design for Simultaneous, Coherent DAQ Using Multiple ADCs*
- *Simple PID Control Reference Design With PRU®-ICSS Through Web Interface*
- *Thermal Printing with the PRU-ICSS on the BeagleBone Black Reference Design*
- *PRU-ICSS FAQ* wiki
- *PRU Read Latencies* wiki

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Original (June 2018) to A Revision	Page
• Added PRU_ICSSG information.	2
• Added PRU_ICSSG Description section.	2
• Updated Real-Time Industrial Communication section.	3
• Updated General Purpose Use Cases section.	4
• Updated References.	7