ABSTRACT

This application report provides a getting started guide for the PRU-ICSS of the Sitara™ family of devices, while running TI-RTOS on the ARM core. The application report is a companion document of PRU-ICSS Getting Started Guide on Linux, intended to help new users ramp up PRU-ICSS-based development quickly. The application report focuses on TI-RTOS-specific topics, and does not cover the basics discussed in the PRU-ICSS Getting Started Guide on Linux, which includes:

- PRU-ICSS description
- Devices containing the PRU-ICSS
- What can the PRU-ICSS do
- PRU-ICSS usage environments
- PRU C compiler and assembler

Contents

1 PRU-ICSS TI-RTOS Drivers ........................................................................................................... 2
2 PRU-ICSS TI-RTOS Examples ....................................................................................................... 3
3 More Information .......................................................................................................................... 5
4 Frequently Asked Questions ......................................................................................................... 5
5 References ....................................................................................................................................... 6

List of Tables

1 ICSS_EMAC Modules and Locations .............................................................................................. 3

Trademarks

Sitara is a trademark of Texas Instruments.
Linux is a registered trademark of Linus Torvalds.
All other trademarks are the property of their respective owners.
1 PRU-ICSS TI-RTOS Drivers

1.1 PRUSS Driver

The Programmable Real-time Unit SubSystem (PRUSS) on Texas Instruments’ Sitara processors is firmware-programmable, and can execute various functionalities as designed: for example, Ethernet MAC, Ethernet switch, and real-time industrial protocols. The PRUSS driver from the Processor SDK RTOS provides well-defined APIs, which allow an application to control the subsystem.

The PRUSS driver module available from <PDK>\packages\ti\drv\pruss includes:
• Compiled library
• Sources, unit test code
• API reference guide

The PRUSS driver is featured with the following:
• Control to enable, disable, or reset a PRU
• Helper functions to load and execute firmware in PRU
• Memory mapping of PRU, L3, or external memories
• PRU and host event management to map sys_evt, channel, or hosts in the PRU Interrupt Controller (INTC) to generate interrupts, wait for occurrence of an event, and acknowledge interrupts
• Interrupt management for A15/C66x CPU targets

A typical PRUSS API call flow, for a use case when sending an INTC event from the host to the PRU, is described in Processor SDK RTOS driver section.

All the board-specific configurations, for example, enabling the clock and pin-mux of UART/GPIO/PRUSS pins should be performed before calling any of the driver APIs.

The PRUSS configuration structure (pruss_config) must be provided to the driver when the PRUICSS_create() API is called to create the PRUICSS_Handle. This handle is subsequently required to make any PRUSS LLD API call. A PRUSS example Code Composer Studio (CCS) project with the above call flow is available in the Processor SDK. Refer to the PDK Example and Test Project Creation procedure to get familiar with the example project “PRUSS_BasicExample” creation and PRUSS APIs.

PRUICSS_setPRUBuffer() sets the firmware buffer pointer for the PRU. PRUICSS_pruExecProgram() implements firmware array loading to Instruction RAM (IRAM) using PRUICSS_pruWriteMemory(), PRU reset and enabling to execute the program.

PRUICSS_pruWriteMemory() is the function to write the given data (firmware) to PRU memory, including IRAM and data RAM (DRAM). For example:

PRUICSS_pruWriteMemory(ICSS_EMAC_testPruIcssHandle,PRU_ICSS_DATARAM(0)...)  
PRUICSS_pruWriteMemory(ICSS_EMAC_testPruIcssHandle,PRU_ICSS_IRAM(0)...)  

The PRU Code Generation Tool (CGT) 2.2.1 has added an --array option that uses the hexpru utility to take an *.out file and create a 32-bit array of PRU firmware.

1.2 ICSS_EMAC Driver

The Industrial Communications Sub System Ethernet Media Access Controller (ICSS_EMAC) driver provides APIs to transmit and receive packets with a PRUSS firmware-based Ethernet switch implemented on the PRU-ICSS 32-bit RISC cores.

The ICSS EMAC low level driver can be partitioned into:
• Driver software running on the host processor that provides a well-defined set of APIs to configure the driver, send packets to the firmware, and receive packets from the firmware.
• Firmware running on PRU-ICSS cores that implements a 2-port Ethernet switch supporting 802.1d at 100 Mbps.

The ICSS_EMAC driver is featured with the following:
• Rx – Copying packet received from firmware and providing it to the TCP/IP stack
• Tx – Providing packet from TCP/IP stack to firmware
• Learning/forwarding data base
• Storm prevention implementation
• Host statistics implementation
• TCP/IP stack-related initialization
• Configuring IP address
• ARM interrupt management

A typical use case of the ICSS_EMAC driver APIs for an EMAC example which implements a single Ethernet MAC using PRU-ICSS Instance 2, ETH0 is described in the Processor SDK RTOS driver section. All the board-specific configurations, for example, enabling the clock and pin-mux of GPIO, MDIO, and IEP pins should be performed before calling any of the driver APIs.

Hardware attributes, including the base address of various sub-subsystems required by the ICSS_EMAC driver, are required and must be passed in as part of the ICSS_EMAC handle when calling the icss_emacInit() API, which initializes and configures PRU-ICSS in EMAC/switch mode. For details on EMAC/switch mode, refer to ICSS_EMAC LLD developers guide.

2 PRU-ICSS TI-RTOS Examples

2.1 PRU Software Support Package

As explained in the PRU-ICSS Getting Started Guide on Linux, the PRU Software Support Package hosted in the git repository contains building block PRU examples for Sitara devices, along with the necessary header files and libraries. The PRU Software Support Package is designed in the context of RemoteProc Linux driver, which is responsible for taking the PRU firmware from the Linux® filesystem, parsing it for any resources that it needs to provide for the PRU (interrupts or shared memory), loading it into PRU instruction memory and data memory, and then running the PRU. A resource table is necessary for the RemoteProc driver to work, even if it is empty.

Except for the RPMsg examples in the PRU Software Support Package, which must use the resource table to request that Linux creates virtio vrings in DDR and configure the PRU system events to be used as notifications, a lot of the examples that have empty resource tables which can be removed or excluded, and can be re-built and run with an ARM TI-RTOS use case.

Similarly, as PRU development on ARM Linux, TI recommends referring to the PRU Software Support Package, and using one of the examples in the PRU Software Support Package as a starting point for developing a new general purpose firmware on TI-RTOS.

2.2 Dual EMAC and Switch

The PRU-ICSS firmware of a dual EMAC and switch is a part of the ICSS_EMAC lower level driver. The dual EMAC firmware realizes a single port Ethernet MAC, which can be used independently on two PRUs to implement two independent MACs with two different MAC addresses and two different IP addresses. The switch firmware is a 3-port learning Ethernet switch, and implements a 2-port cut through Ethernet switch. The dual EMAC and switch PRU-ICSS firmware source code is available in TI’s Processor SDK, as shown in Table 1.

<table>
<thead>
<tr>
<th>Module</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common baseline code</td>
<td>&lt;PDK&gt;/packages/ti/drv/icss_emac/firmware/icss_dualemac</td>
</tr>
<tr>
<td>Dual_EMAC</td>
<td>&lt;PDK&gt;/packages/ti/drv/icss_emac/firmware/icss_dualemac</td>
</tr>
<tr>
<td>Switch</td>
<td>&lt;PDK&gt;/packages/ti/drv/icss_emac/firmware/icss_switch</td>
</tr>
</tbody>
</table>
For details of building and running the dual EMAC and switch firmware and example, refer to Processor SDK PRU-ICSS firmware section

The Processor SDK provides a variety of ARM/DSP/M4 core-based application examples of dual EMAC and switch, such as:
- ICSS_EMAC_BasicExample
- ICSS_EMAC_SwitchExample
- NIMI_ICSS_BasicExample
- NIMU_ICSS_FtpExample
- NIMU_ICSS_CCLinkMaster
- NIMU_ICSS_CCLinkSlave

### 2.3 Simple Open Real-Time Ethernet Protocol (SORTET)

The Simple Open Real-time Ethernet protocol (SORTET) is a TI-developed industrial Ethernet protocol that supports 4-µs cycle time and operates exclusively on the PRU-ICSS. The protocol is fully documented and released in source code. It is open to customers to learn, adapt, and enhance the protocol for their application requirements.

THE SORTE ARM application and firmware sources can be found under the following directory:

`<PDK>/packages/ti/drv/pruss/example/apps/sorte/`

Refer to the README.txt for high-level overview of how the protocol is implemented for master and slave device network components:

`<PDK>/packages/ti/drv/pruss/example/apps/sorte/firmware/src/master/README.txt`

`<PDK>/packages/ti/drv/pruss/example/apps/sorte/firmware/src/slave/README.txt`

For details of the SORTE building and running procedure, refer to Processor SDK PRU-ICSS firmware section.

### 2.4 I2C

PRU-ICSS I2C provides additional I2C instances for Sitara processors. The firmware supports standard two-pin I2C interface through three GPIO pins from PRU-ICSS peripheral. The I2C SCL pin for firmware is implemented with a single GPIO configured in GPI mode. While the I2C SDA pin is implemented with two GPIO pins, one pin configured in GPI mode for taking input sample and a second pin configured in GPO mode for driving the line. Depending on Baud rate, firmware can emulate multiple instances of the I2C interface from a single PRU core. The I2C firmware source code is available from `<PDK>/packages/ti/drv/i2c/firmware/icss_i2c/src`. For details of the I2C firmware build procedure and its example application, refer to Processor SDK PRU-ICSS firmware section.

### 2.5 Industrial SW

The PRU-ICSS industrial software for Sitara processors contains many optimized real-time industrial communication protocols firmware including EtherCAT, Profinet, EtherNet/IP, Profibus, HSR/PRP, and industrial drive interfaces. The PRU-ICSS firmware runs on the PRU cores, offloading the time-critical link layer processing from the main ARM processor running TI-RTOS. The application examples in the software illustrate how to integrate low-level firmware with the protocol stack and application software.

The industrial SW features:
- PRU-ICSS firmware binary images and driver sources
- Third-party stacks and evaluation libraries
- Scripts to generate CCS projects
- Example application for evaluation
- Documentation (release notes, protocol data sheets, user guides, porting guides, and so forth)
3 More Information

3.1 Training Materials
The PRU training series covers:
• Sitara Processors Building Blocks for PRU Development
• Tools for PRU development
• Running Industrial Protocols on PRU

3.2 TI Designs
SORTE Master with PRU-ICSS Reference Design – This design implements a Simple Open Real-Time Ethernet (SORTE) master with PRU-ICSS. SORTE protocol enables customer applications to exchange process data between the master and devices in a 4-μs cycle time. The PRU firmware in source code enables customers to differentiate their products.

SORTE Device With PRU-ICSS Reference Design – This design shows the different industrial Ethernet operation modes, including auto-forwarding (AF), host receive (HR), cut-through (CT), and time-triggered-send (TTS). The application example on top of the SORTE protocol controls the digital 8-bit output LEDs on the TMDSICE3359 evaluation module (EVM).

DDR-less EtherCAT® Slave on AMIC110 Reference Design – This design shows how to run a full EtherCAT slave stack entirely on the internal memory of the Sitara AMIC110 SoC, while offloading the EtherCAT data link layer to PRU-ICSS.

EnDat 2.2 System Reference Design – This design implements the EnDat 2.2 Master protocol stack on PRU-ICSS, and a hardware interface solution based on the HEIDENHAIN EnDat 2.2 standard for position or rotary encoders.

3.3 E2E Forum Support
Any questions not addressed by this guide can be posted on the Sitara E2E Forums.

4 Frequently Asked Questions
This section includes a few of the most common questions regarding using the PRU-ICSS with TI-RTOS on the ARM core. For a comprehensive list of Frequently Asked Questions on the PRU-ICSS, see the PRU-ICSS FAQ wiki.

4.1 How can I load PRU firmware from a host core with TI-RTOS?
Refer to Section 1.1.

4.2 I'm setting a GPIO pin in PRU-ICSS to direct input mode; why is the value reflecting the GPIO pin status read from R31 register unchanged?
In addition to the setting to the pin mux mode to GPIO, ensure that the INPUTENABLE bit is set to get ‘receive mode’ enabled in the pin’s CTRL_CORE_PAD register. Because the PRU does not have privileges to edit the pinmux or pad config settings in the device-level control module, the INPUTENABLE bit in CTRL_CORE_PAD register can only be set from a host core.

4.3 How can I establish the communication between a host core and PRU?
Shared memory (DDR, OCMC RAM, or PRU data RAM) with an interrupt between a host core and PRU can be used to implement the functionality.

For an ARM core, a memory fence instruction must be used to insure that all writes have completed before generating an interrupt and any access from the PRU. While using the HW_WR_REG32_RAW() function, which has HW_MEM_BARRIER() embedded, ensure that __ARMv7 or __TI_ARM_V7__ is defined in your project to get the barrier taking effect; see the implementation of HW_MEM_BARRIER below.
static inline void HW_MEM_BARRIER(void)
{
    #if (defined(__ARMv7) || defined(__TI_ARM_V7__))
    #ifndef MEM_BARRIER_DISABLE
        asm(" dsb");
    #endif
    #endif
}

While using the DDR as shared memory, ensure the MMU has the memory attribute set to shareable, as in the following example:

```c
// descriptor attribute structure
var attrs1 = new Mmu.DescriptorAttrs();
Mmu.initDescAttrsMeta(attrs1);
attrs1.type = Mmu.DescriptorType_BLOCK;  // BLOCK descriptor
attrs1.shareable = 2;                // shareable
attrs1.attrIndx = 2;                 // Cached, normal memory

// Set the descriptor for each entry in the address range
for (var i=0x80000000; i < 0xA0000000; i = i + 0x00200000) {
    Mmu.setSecondLevelDescMeta(i, i, attrs1);
}
```

The PRU can interrupt the ARM by writing to R31 and generating a system event. The PRU INTC should be pre-configured to map this system event to a host interrupt connected to the ARM. The ARM can interrupt a PRU by writing to the PRU INTC SRSRx register and setting a pr<k>_pru_mst_intr<x>_intr_req system event. The PRU INTC should be pre-configured to map this system event to a host interrupt connected to the PRU. The PRU can poll R31 bit 30 or 31 to detect an interrupt on host 0 or 1, respectively.

The example PRU_ARMtoPRU_Interrupt and PRU_PRUtoARM_Interrupt projects in the PRU Software Support Package can be used as reference for interrupt configuration and handling.

5 References

- RU-ICSS Getting Started Guide on Linux
- Processor SDK Device Drivers
- SORTE Master with PRU-ICSS Reference Design
- SORTE Device With PRU-ICSS Reference Design
- DDR-less EtherCAT® Slave on AMIC110 Reference Design
- EnDat 2.2 System Reference Design
- PRU-ICSS FAQ
IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI') technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT. AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include, without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2018, Texas Instruments Incorporated