Programmable Real-Time Unit Subsystem and Industrial Communication Subsystem

Topic **Page**

30.1 Programmable Real-Time Unit Subsystem and Industrial Communication Subsystem AM572x SR2.0

This section describes the Programmable Real-Time Unit and Industrial Communication Subsystems (PRU-ICSS) in the SR2.0 of this device.

30.1.1 PRU-ICSS Overview

The Programmable Real-Time Unit Subsystem and Industrial Communication Subsystem (PRU-ICSS) consists of:

- Two 32-bit load/store RISC CPU cores - Programmable Real-Time Units (PRU0 and PRU1)
- Data RAMs per PRU core
- Instruction RAMs per PRU core
- Shared RAM
- Peripheral modules
- Interrupt controller (PRUSS_INTC)

The programmable nature of the PRU cores, along with their access to pins, events and all device 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 device.

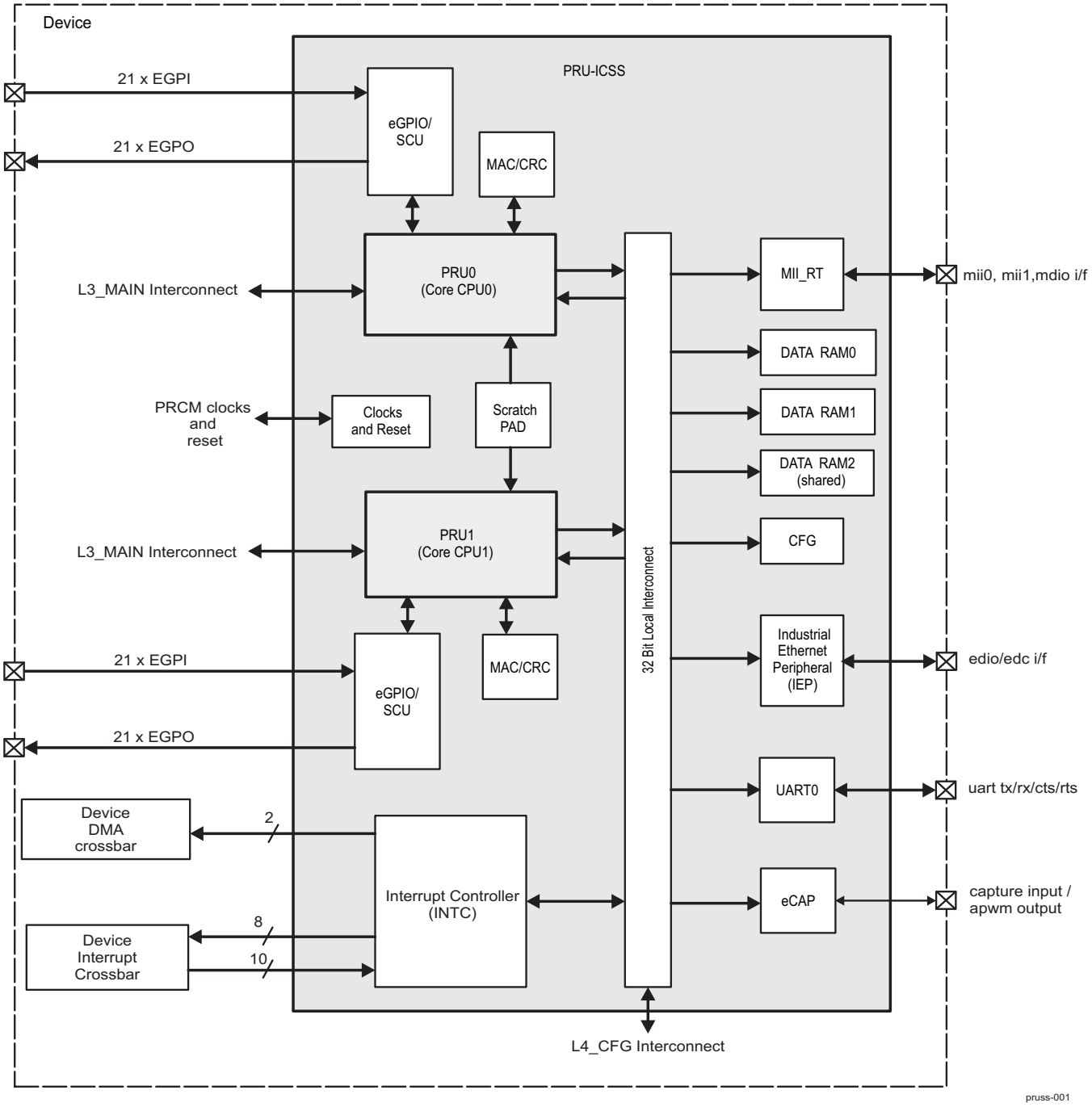
The device has integrated two identical PRU subsystems (PRU-ICSS1 and PRU-ICSS2). The PRUs have access to all resources on the device through a master port on the L3_MAIN interconnect, and vice versa, the external host processors can access the PRU-ICSS resources through a L3_MAIN slave port.

The PRU-ICSS L2 interconnect, provides access to the various internal and external masters to the resources inside the PRU-ICSS. A subsystem local Interrupt Controller - PRUSS_INTC handles system input events and posts events back to the device-level host CPUs.

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. This interaction between processors is determined by the nature of the firmware loaded into the PRU's instruction memory.

Figure 30-1 shows an overview of the PRU subsystem.

Figure 30-1. PRU-ICSS Overview



30.1.1.1 PRU-ICSS Key Features

The PRU subsystem includes the following main features:

- Two PRU CPUs
  - 21 Enhanced General-Purpose Inputs (EGPI) and 21 Enhanced General-Purpose Outputs (EGPO)
  - Multiplier with optional accumulation (MAC)
  - CRC16/CRC32
  - 12-KiB program RAM per PRU CPU (signified IRAM0 for PRU0 and IRAM1 for PRU1)

  - 8-KiB data RAM per PRU CPU (signified RAM0 for PRU0 and RAM1 for PRU1)
  - Two high-performance master (initiator) ports on the L3_MAIN interconnect - one per PRU
- 32-KiB general purpose memory RAM (signified RAM2) shared between PRU0 and PRU1
- One Scratch-Pad (SPAD) memory
  - 3 Banks of 30 × 32-bit registers
- Broadside direct connect between PRU cores within subsystem. Optional address translation for PRU transaction to External Host
- 16 software events generated by two PRUs
- One Ethernet MII_RT module (PRUSS_MII_RT_CFG) with two MII ports and configurable connections to PRUs
- MDIO Port (PRUSS_MII_MDIO) to control extenal Ethernet PHY
- Industrial Ethernet Peripheral (IEP) to manage/generate Industrial Ethernet functions
- 16550-compatible UART with a dedicated 192-MHz clock to support 12-Mbps PROFIBUS
- Industrial Ethernet timer with 7/9 capture and 16 compare events
- Enhanced Capture Module (ECAP)
- Interrupt Controller (PRUSS_INTC)
  - Up to 64 input events supported
  - Interrupt mapping to 10 interrupt channels via an interrupt crossbar
  - 10 Host interrupts (2 to PRU0 and PRU1, 8 outputs to device level)
  - Each system event can be enabled and disabled
  - Each host event can be enabled and disabled
  - Hardware prioritization of events
  - Two level-sensitive DMA requests generated by the local PRUSS INTC to the device DMA Crossbar
- One Slave (target) port for memory mapped register and internal memories access through device L3_MAIN
- Two (master and slave) 32-bit ports for low-latency interface between PRU-ICSS subsystems
- Flexible power management support
- Integrated 32-bit interconnect
- Parity control supported by all memories

NOTE: Use of IEP Sync and WatchDog features are only supported via the TI Industrial software development kit (SDK).

PRU-ICSS unsupported features:

- Only 8 bits are supported of the 32-bit ECAT Digital Data Input
- Only 8 bits are supported of the 32-bit ECAT Digital Data Output
- UART Modem interface is not supported

30.1.2 PRU-ICSS Environment

This section specifies the PRU-ICSS subsystem (top) interface signals to the device environment components.

30.1.2.1 PRU-ICSS I/O Interface

The PRU-ICSS1 and PRU-ICSS2 external interface signals are described in Table 30-1 and Table 30-2, respectively. The PRU-ICSS has a large number of available I/O signals. Most of these are multiplexed with other functional signals at the device level.

Table 30-1. PRU-ICSS1 I/O Signals

Device Level Signal Name	I/O	Description	Value at Reset
pr1_pru0_gpo[20:0]	O	PRU0 Register R30 Outputs	0
pr1_pru0_gpi[20:0]	I	PRU0 Register R31 Inputs	HiZ
pr1_pru1_gpo[20:0]	O	PRU1 Register R30 Outputs	0
pr1_pru1_gpi[20:0]	I	PRU1 Register R31 Inputs	HiZ
pr1_mii_mr0_clk	I	MII0 Receive Clock	HiZ
pr1_mii0_rxdv	I	MII0 Receive Data Valid	HiZ
pr1_mii0_rxd[0:3]	I	MII0 Receive Data	HiZ
pr1_mii0_rxlink	I	MII0 Receive Link	HiZ
pr1_mii0_rxer	I	MII0 Receive Data Error	HiZ
pr1_mii0_crs	I	MII0 Carrier Sense	HiZ
pr1_mii0_col	I	MII0 Carrier Sense	HiZ
pr1_mii_mt0_clk	I	MII0 Transmit Clock	HiZ
pr1_mii0_txen	O	MII0 Transmit Enable	0
pr1_mii0_txd[0:3]	O	MII0 Transmit Data	0
pr1_mii_mr1_clk	I	MII1 Receive Clock	HiZ
pr1_mii1_rxdv	I	MII1 Receive Data Valid	HiZ
pr1_mii1_rxd[0:3]	I	MII1 Receive Data	HiZ
pr1_mii1_rxlink	I	MII1 Receive Link	HiZ
pr1_mii1_rxer	I	MII1 Receive Data Error	HiZ
pr1_mii1_crs	I	MII1 Carrier Sense	HiZ
pr1_mii1_col	I	MII1 Carrier Sense	HiZ
pr1_mii_mt1_clk	I	MII1 Transmit Clock	HiZ
pr1_mii1_txen	O	MII1 Transmit Enable	0
pr1_mii1_txd[0:3]	O	MII1 Transmit Data	0
pr1_mdio_mdclk	O	MDIO Clk	0
pr1_mdio_data	I/O	MDIO Data	HiZ
pr1_edio_sof	O	ECAT Digital I/O Start of Frame	0
pr1_edio_latch_in	I	ECAT Digital I/O Latch In	HiZ
pr1_edio_data_in[0:7]	I	ECAT Digital I/Os Data In	HiZ
pr1_edio_data_out[0:7]	O	ECAT Digital I/Os Data Out	0
pr1_edc_sync0_out	O	ECAT Distributed Clock Sync Out	0
pr1_edc_sync1_out	O	ECAT Distributed Clock Sync Out	0
pr1_edc_latch0_in	I	ECAT Distributed Clock Latch In	HiZ
pr1_edc_latch1_in	I	ECAT Distributed Clock Latch In	HiZ
pr1_uart0_cts_n	I	UART Clear to Send	HiZ
pr1_uart0_rts_n	O	UART Request to Send	0
pr1_uart0_rxd	I	UART Receive Data	HiZ
pr1_uart0_txd	O	UART Transmit Data	0
pr1_ecap0_ecap_capin_apwm_o	I/O	Enhanced capture (ECAP) input or Auxiliary PWM out	HiZ

Table 30-2. PRU-ICSS2 I/O Signals

Device Level Signal Name	I/O	Description	Value at Reset
pr2_pru0_gpo[20:0]	O	PRU0 Register R30 Outputs	0
pr2_pru0_gpi[20:0]	I	PRU0 Register R31 Inputs	HiZ
pr2_pru1_gpo[20:0]	O	PRU1 Register R30 Outputs	0
pr2_pru1_gpi[20:0]	I	PRU1 Register R31 Inputs	HiZ

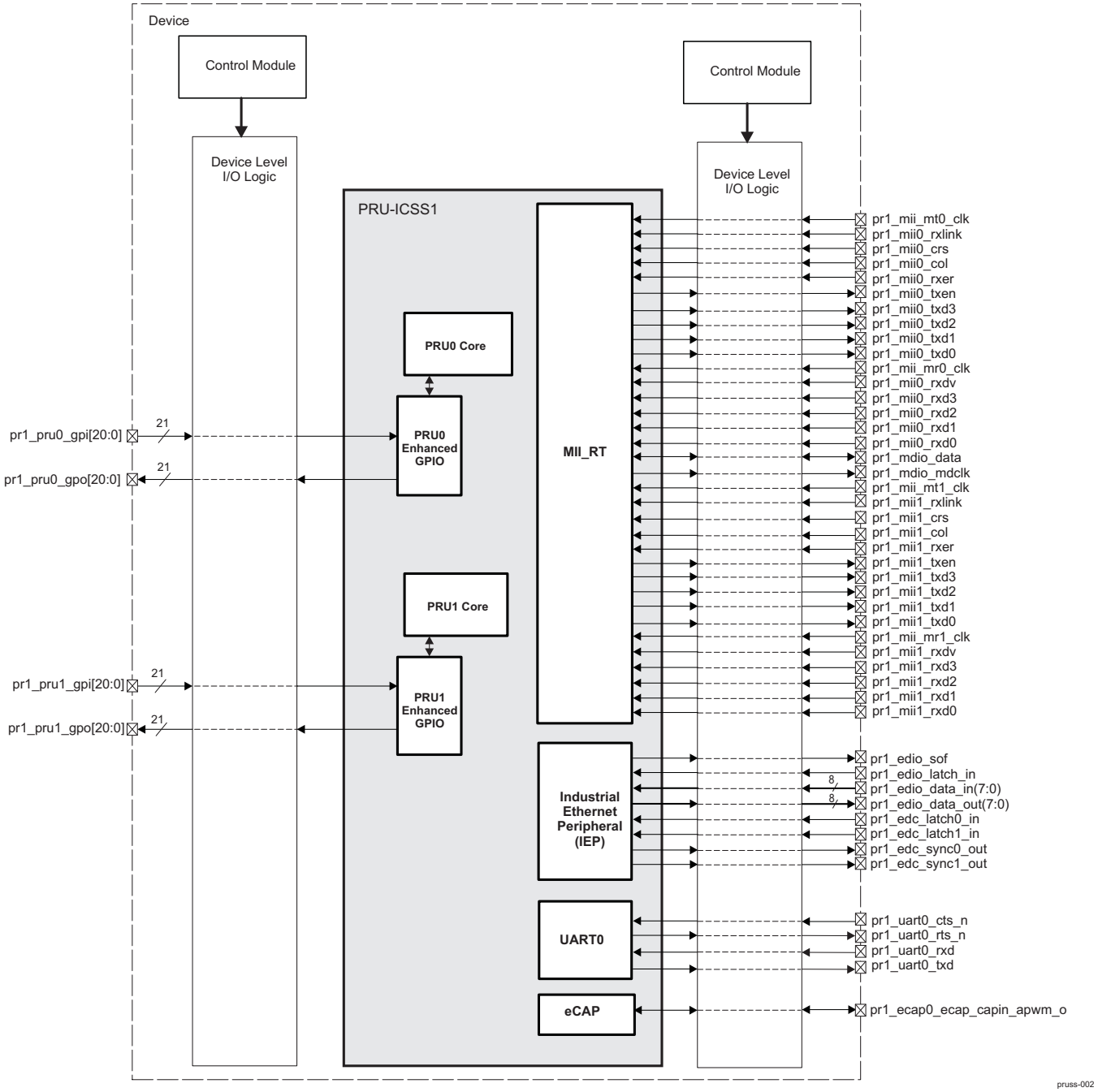
Table 30-2. PRU-ICSS2 I/O Signals (continued)

Device Level Signal Name	I/O	Description	Value at Reset
pr2_mii_mr0_clk	I	MII0 Receive Clock	HiZ
pr2_mii0_rxdv	I	MII0 Receive Data Valid	HiZ
pr2_mii0_rxd[0:3]	I	MII0 Receive Data	HiZ
pr2_mii0_rxlink	I	MII0 Receive Link	HiZ
pr2_mii0_rxer	I	MII0 Receive Data Error	HiZ
pr2_mii0_crs	I	MII0 Carrier Sense	HiZ
pr2_mii0_col	I	MII0 Carrier Sense	HiZ
pr2_mii_mt0_clk	I	MII0 Transmit Clock	HiZ
pr2_mii0_txen	O	MII0 Transmit Enable	0
pr2_mii0_txd[0:3]	O	MII0 Transmit Data	0
pr2_mii_mr1_clk	I	MII1 Receive Clock	HiZ
pr2_mii1_rxdv	I	MII1 Receive Data Valid	HiZ
pr2_mii1_rxd[0:3]	I	MII1 Receive Data	HiZ
pr2_mii1_rxlink	I	MII1 Receive Link	HiZ
pr2_mii1_rxer	I	MII1 Receive Data Error	HiZ
pr2_mii1_crs	I	MII1 Carrier Sense	HiZ
pr2_mii1_col	I	MII1 Carrier Sense	HiZ
pr2_mii_mt1_clk	I	MII1 Transmit Clock	HiZ
pr2_mii1_txen	O	MII1 Transmit Enable	0
pr2_mii1_txd[0:3]	O	MII1 Transmit Data	0
pr2_mdio_mdclk	O	MDIO Clk	0
pr2_mdio_data	I/O	MDIO Data	HiZ
pr2_edio_sof	O	ECAT Digital I/O Start of Frame	0
pr2_edio_latch_in	I	ECAT Digital I/O Latch In	HiZ
pr2_edio_data_in[0:7]	I	ECAT Digital I/Os Data In	HiZ
pr2_edio_data_out[0:7]	O	ECAT Digital I/Os Data Out	0
pr2_edc_sync0_out	O	ECAT Distributed Clock Sync Out	0
pr2_edc_sync1_out	O	ECAT Distributed Clock Sync Out	0
pr2_edc_latch0_in	I	ECAT Distributed Clock Latch In	HiZ
pr2_edc_latch1_in	I	ECAT Distributed Clock Latch In	HiZ
pr2_uart0_cts_n	I	UART Clear to Send	HiZ
pr2_uart0_rts_n	O	UART Request to Send	0
pr2_uart0_rxd	I	UART Receive Data	HiZ
pr2_uart0_txd	O	UART Transmit Data	0
pr2_ecap0_ecap_capin_apwm_o	I/O	Enhanced capture (ECAP) input or Auxiliary PWM out	HiZ

NOTE: The device I/O logic maps the PRU-ICSS signals to the different device pads by programming in the Control Module. For more information, refer to the Section 18.4.6.1.1, *Pad Configuration Registers* in the Chapter 18, *Control Module*.

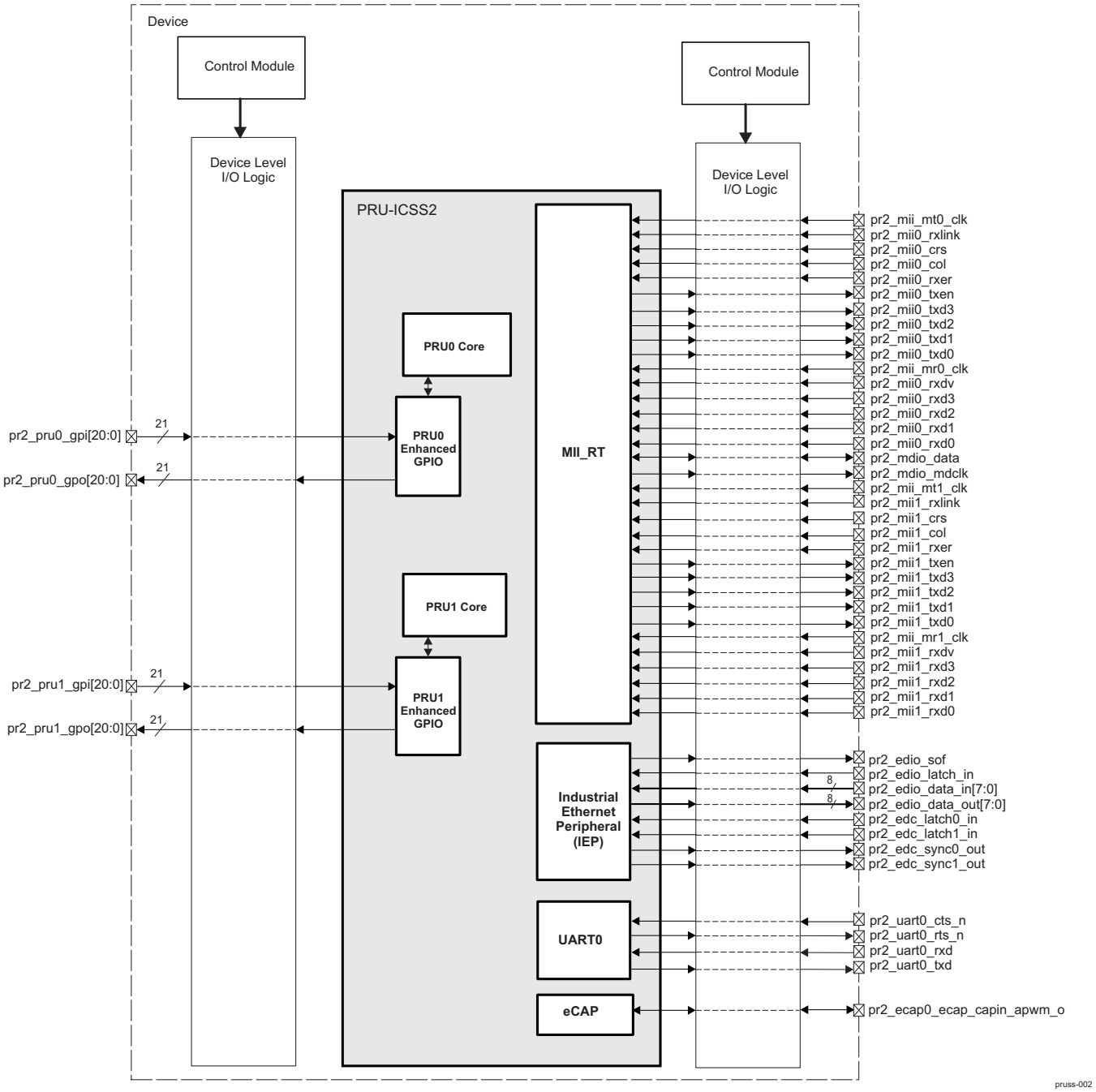
The below Figure 30-2 illustrates the PRU-ICSS1 I/O interface signals at the device boundary.

Figure 30-2. PRU-ICSS1 External Interface I/Os



The Figure 30-3 illustrates the PRU-ICSS2 I/O interface signals at the device boundary.

Figure 30-3. PRU-ICSS2 External Interface I/Os



The I/O signals are identically exported for both PRU-ICSS1 and PRU-ICSS2 subsystems.

30.1.3 PRU-ICSS Integration

The PRU-ICSS1 and PRU-ICSS2 subsystems integration in the device is shown in Figure 30-4 and Figure 30-5, respectively.

Figure 30-4. PRU-ICSS1 Integration in the Device

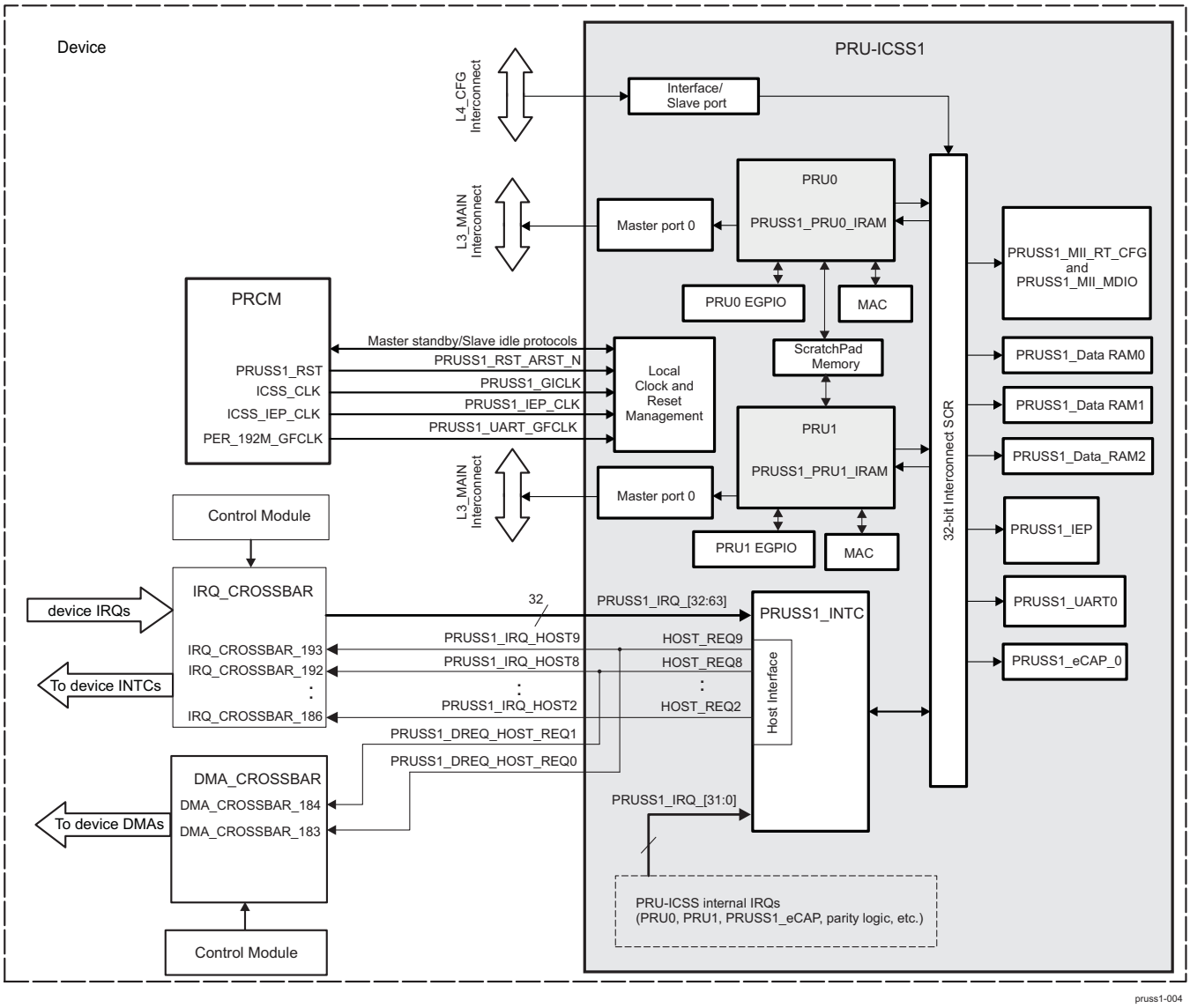
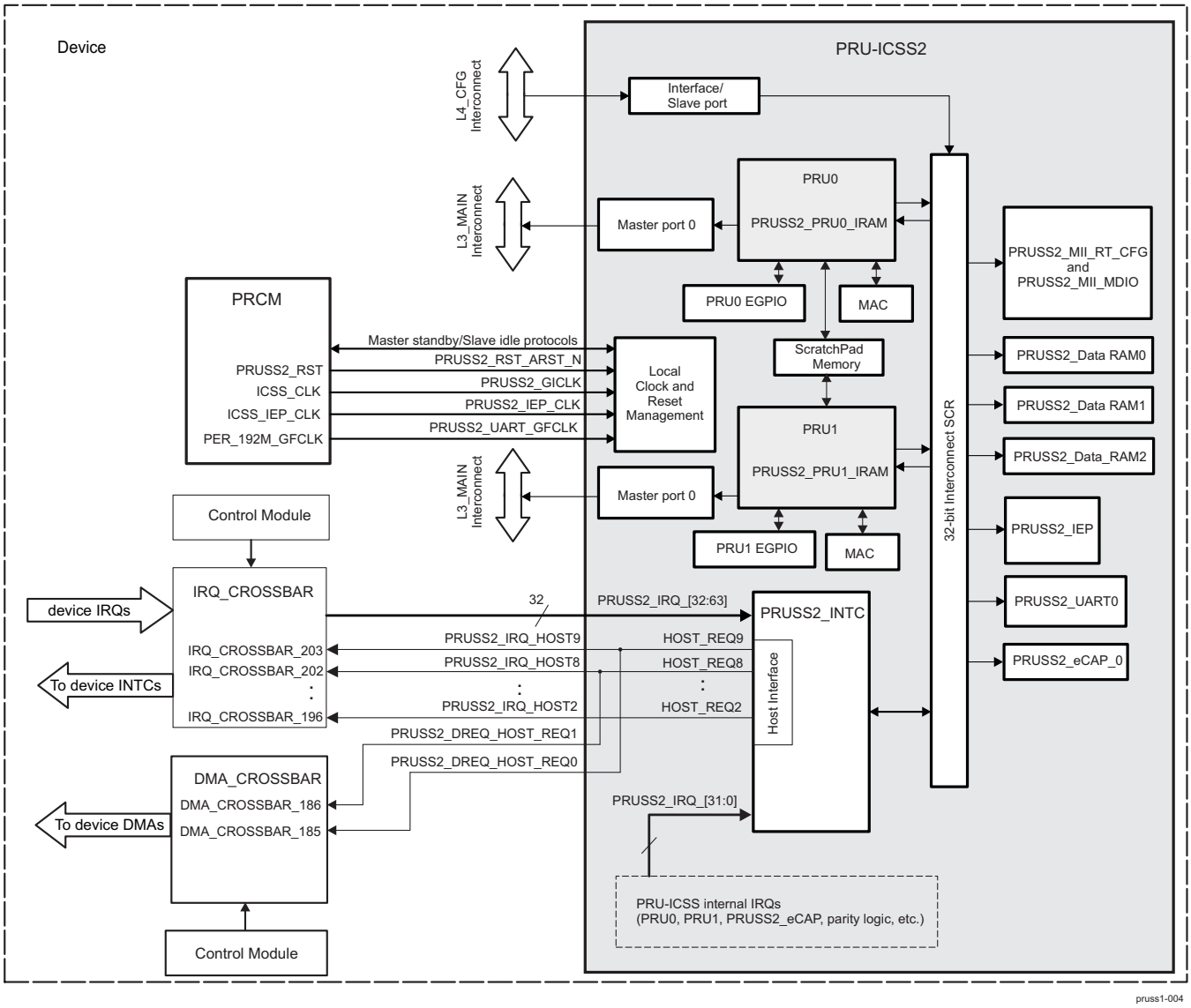


Figure 30-5. PRU-ICSS2 Integration in the Device



The PRU-ICSS1 and PRU-ICSS2 integration in the device features:

- PD_L4PER power domain instantiation
- two master ports (PRU0 and PRU1 core initiators) on the device L3_MAIN interconnect
- Non-wakeup capable - smart Standby protocol with the device PRCM
- Software assertion of a standby request "MStandby" (for master port clock disable) with local (PRU-ICSS) monitoring of the PRCM "MWait" acknowledge.
- one slave (configuration) port on the L3_MAIN interconnect for device hosts (MPU, DSP1, etc.) to access various memories and registers of PRU-ICSS
- Non-wakeup capable - smart Idle protocol with the device PRCM
- 10 output interrupt events from local interrupt controller - PRUSS_INTC:
  - 2 events to each PRU core (events 0 and 1)
  - 8 events mapped to the device IRQ_CROSSBAR which further remaps them to device interrupt controllers (events 2 through 9)
  - 2 events mapped to the device DMA_CROSSBAR, that remaps them to device DMA controllers (events 8 and 9)

- 32 external interrupts are mapped via the device IRQ_CROSSBAR to the local PRUSS_INTC
- A local software gating of clocks to several modules within PRU subsystem (local clock management protocol), as follows:
  - PRUSS_IEP
  - PRUSS_eCAP_0
  - PRUSS_UART0
  - PRUSS_INTC
  - PRUSS_PRU0
  - PRUSS_PRU1
- 3 input clocks obtained from device PRCM:
  - a PRU-ICSS top level gatable interface clock
  - a PRUSS IEP functional clock
  - a PRUSS UART0
- No memory/register retention is supported
- One hardware non-retention (level sensitive) reset

Table 30-3 through Table 30-5 summarize the integration of the module in the device.

Table 30-3. PRU-ICSS Integration Attributes

Module Instance	Attributes	
	Power Domain	Interconnect
PRU-ICSS1	PD_COREAON	L3_MAIN L4_CFG
PRU-ICSS2	PD_COREAON	L3_MAIN L4_CFG

Table 30-4. PRU-ICSS Clocks and Resets

Clocks				
Module Instance	**Destination Signal Name**	**Source Signal Name**	**Source**	**Description**
PRU-ICSS1	PRUSS1_GICLK	ICSS_CLK	PRCM	PRU-ICSS1 gated interface clock derived from DPLL_GMAC
	PRUSS1_UART_GFCLK	PER_192M_GFCLK	PRCM	PRUSS1_UART gated functional clock derived from DPLL_PER
	PRUSS1_IEP_CLK	ICSS_IEP_CLK	PRCM	PRUSS1_IEP functional clock derived from DPLL_GMAC
	PRUSS1_MII_MR0_CLK	pr1_mii_mr0_clk	pr1_mii_mr0_clk pin	MII0 RT RX functional clock
	PRUSS1_MII_MR1_CLK	pr1_mii_mr1_clk	pr1_mii_mr1_clk pin	MII1 RT RX functional clock
	PRUSS1_MII_MT0_CLK	pr1_mii_mt0_clk	pr1_mii_mt0_clk pin	MII0 RT TX functional clock
	PRUSS1_MII_MT1_CLK	pr1_mii_mt1_clk	pr1_mii_mt1_clk pin	MII1 RT TX functional clock
PRU-ICSS2	PRUSS2_GICLK	ICSS_CLK	PRCM	PRU-ICSS2 gated interface clock derived from DPLL_GMAC
	PRUSS2_UART_GFCLK	PER_192M_GFCLK	PRCM	PRUSS2_UART gated functional clock derived from DPLL_PER
	PRUSS2_IEP_CLK	ICSS_IEP_CLK	PRCM	PRUSS2_IEP functional clock derived from DPLL_GMAC
	PRUSS2_MII_MR0_CLK	pr2_mii_mr0_clk	pr2_mii_mr0_clk pin	MII0 RT RX functional clock
	PRUSS2_MII_MR1_CLK	pr2_mii_mr1_clk	pr2_mii_mr1_clk pin	MII1 RT RX functional clock
	PRUSS2_MII_MT0_CLK	pr2_mii_mt0_clk	pr2_mii_mt0_clk pin	MII0 RT TX functional clock
	PRUSS2_MII_MT1_CLK	pr2_mii_mt1_clk	pr2_mii_mt1_clk pin	MII1 RT TX functional clock
Resets				

Table 30-4. PRU-ICSS Clocks and Resets (continued)

Module Instance	Destination Signal Name	Source Signal Name	Source	Description
PRU-ICSS1	PRUSS1_RST_MAIN_ARST_N	PRUSS1_RST	PRCM	Non-retention hardware main reset to the PRU-ICSS1
PRU-ICSS2	PRUSS2_RST_MAIN_ARST_N	PRUSS2_RST	PRCM	Non-retention hardware main reset to the PRU-ICSS2

Table 30-5. PRU-ICSS Hardware Requests

Interrupt Requests				
Module Instance	**Source Signal Name**	**IRQ_CROSSBAR Input**	**Default Mapping**	**Description**
PRU-ICSS1	PRUSS1_IRQ_HOST2	IRQ_CROSSBAR_186	-	PRU-ICSS1 Host interrupt 2
	PRUSS1_IRQ_HOST3	IRQ_CROSSBAR_187	-	PRU-ICSS1 Host interrupt 3
	PRUSS1_IRQ_HOST4	IRQ_CROSSBAR_188	-	PRU-ICSS1 Host interrupt 4.
	PRUSS1_IRQ_HOST5	IRQ_CROSSBAR_189	-	PRU-ICSS1 Host interrupt 5
	PRUSS1_IRQ_HOST6	IRQ_CROSSBAR_190	-	PRU-ICSS1 Host interrupt 6
	PRUSS1_IRQ_HOST7	IRQ_CROSSBAR_191	-	PRU-ICSS1 Host interrupt 7
	PRUSS1_IRQ_HOST8	IRQ_CROSSBAR_192	-	PRU-ICSS1 Host interrupt 8
	PRUSS1_IRQ_HOST9	IRQ_CROSSBAR_193	-	PRU-ICSS1 Host interrupt 9
PRU-ICSS2	PRUSS2_IRQ_HOST2	IRQ_CROSSBAR_196	-	PRU-ICSS2 Host interrupt 2
	PRUSS2_IRQ_HOST3	IRQ_CROSSBAR_197	-	PRU-ICSS2 Host interrupt 3
	PRUSS2_IRQ_HOST4	IRQ_CROSSBAR_198	-	PRU-ICSS2 Host interrupt 4
	PRUSS2_IRQ_HOST5	IRQ_CROSSBAR_199	-	PRU-ICSS2 Host interrupt 5
	PRUSS2_IRQ_HOST6	IRQ_CROSSBAR_200	-	PRU-ICSS2 Host interrupt 6
	PRUSS2_IRQ_HOST7	IRQ_CROSSBAR_201	-	PRU-ICSS2 Host interrupt 7
	PRUSS2_IRQ_HOST8	IRQ_CROSSBAR_202	-	PRU-ICSS2 Host interrupt 8
	PRUSS2_IRQ_HOST9	IRQ_CROSSBAR_203	-	PRU-ICSS2 Host interrupt 9
DMA Requests				
Module Instance	**Source Signal Name**	**DMA_CROSSBAR Input**	**Default Mapping**	**Description**
PRU-ICSS1	PRUSS1_DREQ_HOST_REQ0	DMA_CROSSBAR_183	-	PRU-ICSS1 Host DMA request 0. Source is host interrupt 9
	PRUSS1_DREQ_HOST_REQ1	DMA_CROSSBAR_184	-	PRU-ICSS1 Host DMA request 1. Source is host interrupt 8
PRU-ICSS2	PRUSS2_DREQ_HOST_REQ0	DMA_CROSSBAR_185	-	PRU-ICSS2 Host DMA request 0. Source is host interrupt 9
	PRUSS2_DREQ_HOST_REQ1	DMA_CROSSBAR_186	-	PRU-ICSS2 Host DMA request 1. Source is host interrupt 8

NOTE: For more information about the IRQ_CROSSBAR module, see Section 18.4.6.4, *IRQ_CROSSBAR Module Functional Description*, in Chapter 18, *Control Module*.

For more information about the DMA_CROSSBAR module, see Section 18.4.6.5, *DMA_CROSSBAR Module Functional Description*, in Chapter 18, *Control Module*.

For more information about the device interrupt controllers, see Chapter 17, *Interrupt Controllers*.

For more information about the device SDMA module, see Section 16.1, *System DMA*.

For more information about the device EDMA module, see Section 16.2, *Enhanced DMA*.

30.1.4 PRU-ICSS Level Resources Functional Description

This section provides functional description of the device integrated PRU Subsystems modules.

30.1.4.1 PRU-ICSS Reset Management

An individual PRCM cold and warm hardware reset is available per PRU-ICSS1 (PRUSS1_RST) and PRU-ICSS2 (PRUSS2_RST). It is asserted by the PRCM upon cold reset and warm reset events.

For more details on the PRU-ICSS resets mapping, see also the Section 30.1.3.

NOTE: A hardware reset event (PRUSS_RST_MAIN_ARST_N input assertion) forces the PRU-ICSS to go to an IDLE and STANDBY state. For more details on PRU-ICSS clock management defined states refer to the Section 30.1.4.2.

NOTE: No global software reset is available at the PRU-ICSS top level.

30.1.4.2 PRU-ICSS Power and Clock Management

The PRU-ICSS supports 2 levels of clock gating. First level gates all clocks inside the PRU-ICSS when it is placed into IDLE and STANDBY state. The second level allows user software to enable/disable clocks in the clock gating register PRUSS_CGR to some internal modules, as follows:

- PRUSS1_IEP
- PRUSS1_eCAP_0
- PRUSS1_UART0
- PRUSS1_INTC
- PRUSS1_PRU0_Control /PRUSS1_PRU0_IRAM
- PRUSS1_PRU1_Control/PRUSS1_PRU1_IRAM
- PRUSS2_IEP
- PRUSS2_eCAP_0
- PRUSS2_UART0
- PRUSS2_INTC
- PRUSS2_PRU0_Control /PRUSS2_PRU0_IRAM
- PRUSS2_PRU1_Control/PRUSS2_PRU1_IRAM

The appropriate configuration registers block controls its local module set inside PRU-ICSS.

30.1.4.2.1 PRU-ICSS Idle and Standby States

The below Table 30-6 lists the clock management settings applicable at PRU-ICSS subsystem level (first level of local power management).

NOTE: For more details on the slave idle protocol (slave port) and master standby protocol (master port) between PRU-ICSS and device PRCM, refer to the Section 3.1.1.1.2, *Module-Level Clock Management* in the Chapter 3, *Power, Reset and Clock Management*.

Table 30-6. PRU-ICSS Idle/Standby Support

IDLE/STANDBY Mode	Comments
NO IDLE	
SMART IDLE	Default State
Wake-up capable SMART IDLE	NOT supported
FORCE IDLE	

Table 30-6. PRU-ICSS Idle/Standby Support (continued)

IDLE/STANDBY Mode	Comments
NO STANDBY	
SMART STANDBY	Default State
Wake-up capable SMART STANDBY	NOT supported
FORCE STANDBY	

NOTE: Not all of the PRU-ICSS outputs meet the IDLE state. Only the power protocol and L3_MAIN signals are Idled with all functional and interface clocks being shut-down.

A transition from an ACTIVE/Normal state to an IDLE (L3_MAIN slave) + STANDBY (L3_MAIN masters) state is performed as per the sequence:

1. The host (i.e. device MPU, DSP1, etc.) requests that the PRU firmware goes into IDLE state and waits for acknowledgement.
2. The host issues Clock Stop Request in register PRUSS_CGR to modules with gateable clocks defined at second power management level (see Section 30.1.4.2)
3. The host initiates MStandby via assertion to HIGH of the bit: PRUSS_SYSCFG [4] STANDBY_INIT (PRU-ICSS clock management configuration register).
4. The host software gets the device PRCM to issue an IDLE Request (SidleReq) towards the PRU-ICSS slave port. This is done via writing to device PRCM clock management registers dedicated to PRU-ICSS: CM_L4PER2_PRUSS1_CLKCTRLand CM_L4PER2_PRUSS2_CLKCTRL in Section 3.14, *PRCM Register Manual* of the Chapter 3,*Power Reset and Clock Management*.
5. The PRU-ICSS acknowledges IDLE Request and enters the IDLE+STANDBY state.

A transition from an IDLE + STANDBY state to an ACTIVE/ Normal state is performed as per the sequence:

1. The host (i.e. device MPU, DSP1, etc.) software gets the PRCM to de-assert IDLE Request - This is done via writing to device PRCM clock management registers dedicated to PRU-ICSS: CM_L4PER2_PRUSS1_CLKCTRLand CM_L4PER2_PRUSS2_CLKCTRL in Section 3.14, *PRCM Register Manual* of the Chapter 3,*Power Reset and Clock Management*.
2. The host CPU de-asserts the ClockStopReq to modules with gateable clocks defined at second power management level, and wait for the ClockStopAck to be asserted. This is done via PRU-ICSS host writing/reading the PRUSS_CGR.
3. The host CPU enables "NO STANDBY" via assertion of the PRUSS_SYSCFG[3:2] STANDBY_MODE to 0x1.

30.1.4.2.2 Module Clock Configurations at PRU-ICSS Top Level

IEP functional clock source selection: The clock source selection between PRUSS_IEP_CLK (default) and PRUSS_GICLK to the IEP module is done in register PRUSS_IEPCLK[0] OCP_EN in the PRUSS_CFG location. For more information on these PRU-ICSS level input clocks from PRCM, refer to the Section 30.1.3.

Enhanced GPIO clock divider settings: In certain sample/shift clock settings of the PRU0 and PRU1 EGPIOs (when enabled in serial mode) two cascaded fractional dividers are done in the PRUSS_CFG top level configuration registers PRUSS_GPCFG0 and PRUSS_GPCFG1. In addition, EGPIO clock active edge selection control can be exerted via the bit PRU0_GPI_CLK_MODE for PRU0_EGPIO and PRU1_GPI_CLK_MODE for the PRU1_EGPIO.

- For the serial PRU0's EGPOs:
  - PRUSS_GPCFG0 [24:20]PRU0_GPO_DIV1
  - PRUSS_GPCFG0 [19:15]PRU0_GPO_DIV0
- For the serial PRU0's EGPIs:
  - PRUSS_GPCFG0 [12:8]PRU0_GPI_DIV1

  - PRUSS_GPCFG0 [7:3]PRU0_GPI_DIV0
- For the serial PRU1's EGPOs:
  - PRUSS_GPCFG1 [24:20]PRU1_GPO_DIV1
  - PRUSS_GPCFG1 [19:15]PRU1_GPO_DIV0
- For the serial PRU1's EGPIs:
  - PRUSS_GPCFG1 [12:8]PRU1_GPI_DIV1
  - PRUSS_GPCFG1 [7:3]PRU1_GPI_DIV0

30.1.4.3 Other PRU-ICSS Module Functional Registers at Subsystem Level

Enhanced GPIO. The other functional mode setting for PRUs EGPIOs at PRU-ICSS top registers level are:

- PRUSS_GPCFG0/PRUSS_GPCFG1 [14] PRU1_GPO_MODE - to select between direct or serial EGPO output mode of operation.
- PRUSS_GPCFG0/PRUSS_GPCFG1 [25] PRU1_GPO_SH_SEL - to select between the EGPO shadow registers 0 and 1 used for output shifting. For more details, refer to the Section 30.1.5.2.2.3.4, *Enhanced General-Purpose Module Outputs (R30)*.
- PRUSS_GPCFG0/PRUSS_GPCFG1 [1:0] PRU1_GPI_MODE - selects the EGPI input mode of operation (selects between "direct input", "parallel capture", or "28-bit shift" or "MII_RT" modes).
- PRUSS_GPCFG0/PRUSS_GPCFG1 [13] PRU1_GPI_SB - **start bit event status for 28-bit EGPI input shift mode**. For more details, refer to the Section 30.1.5.2.2.3, *Enhanced General-Purpose Module Inputs (R31)*.

PRU 0/1 cores IRAM and DRAM parity error events: PRUSS_ISRP (raw status), PRUSS_ISP (interrupt status) and PRUSS_IESP (interrupt enable) and PRUSS_IECP (interrupt clear) registers.

PRU 0/1 cores IRAM and DRAM parity error events: PRUSS_ISRP (raw status), PRUSS_ISP (interrupt status) and PRUSS_IESP (interrupt enable) and PRUSS_IECP (interrupt clear) registers.

Enable address offset ("-0x0008_0000") feature individually per PRU0 and PRU1 master ports in the PRUSS_PMAO register in case of accessing peripherals located in the PRU-ICSS space.

PRUSS_MII_RT_CFG interrupts mapping to PRUSS_INTC is enabled in the PRUSS_MII_RT register

PRUs scratchpad (SPAD) memory priority and configuration related bits are located in the PRUSS_SPP register.

30.1.4.4 PRU-ICSS Memory Maps

The PRU-ICSS comprises various distinct addressable regions that are mapped to both a local and global memory map. The local memory maps are maps with respect to the PRU point of view. The global memory maps are maps with respect to the Host point of view, but can also be accessed by the PRU-ICSS.

30.1.4.4.1 PRU-ICSS Local Memory Map

The PRU-ICSS memory map is documented in Table 30-7 (Instruction Space) and in Table 30-8 (Data Space). Note that these two memory maps are implemented inside the PRU-ICSS and are local to the components of the PRU-ICSS.

30.1.4.4.1.1 PRU-ICSS Local Instruction Memory Map

Each PRU has a dedicated 12 KiB of Instruction Memory (PRUSS_PRU0_IRAM and PRUSS_PRU1_IRAM respectively) that must be initialized by an external to PRU-ICSS host processor before a PRU core CPU executes any instructions.

CAUTION

The PRUSS_PRU0/1_IRAM regions are ONLY accessible to PRU-ICSS masters (external hosts like MPU Cortex-A15, DSP1, etc.) when the PRU0/PRU1 is NOT running. The access is via PRU-ICSS slave port on the device L3_MAIN interconnect

Table 30-7. PRU-ICSS Local Instruction Memory Map

Start Address	PRU0	PRU1
0x0000_0000	12 KiB IRAM	12 KiB IRAM

30.1.4.4.1.2 PRU-ICSS Local Data Memory Map

The local data memory map in Table 30-8 allows each PRU core to access the PRU-ICSS addressable regions and the external host's memory map.

The PRU accesses the external Host memory map through the device L3_MAIN interconnect Interface Master port (System OCP_HP0/1) starting at address 0x0008_0000. By default, memory addresses between 0x0000_0000 – 0x0007_FFFF will correspond to the PRU-ICSS local address in Table 30-8. To access an address between 0x0000_0000–0x0007_FFFF of the external host map, the address offset of " – 0x0008_0000" feature is enabled through the PRUSS_PMAO[1] PMAO_PRU1 (for PRU1 CPU) and PRUSS_PMAO[0] PMAO_PRU0 (for PRU0 CPU) bits in the PRUSS_CFG subsystem level register space.

Table 30-8. PRU-ICSS Local Data Memory Map

Start Address	PRUSS_PRU0	PRUSS_PRU1
0x0000_0000	Data 8 KiB RAM0	Data 8 KiB RAM1
0x0000_2000	Data 8 KiB RAM1(1)	Data 8 KiB RAM0(1)
0x0000_4000	Reserved	Reserved
0x0001_0000	Data 32 KiB RAM2 (Shared RAM)	Data 32 KiB RAM2 (Shared RAM)
0x0002_0000	PRUSS_INTC	PRUSS_INTC
0x0002_2000	PRU0 Control	PRU0 Control
0x0002_2400	Reserved	Reserved
0x0002_4000	PRU1 Control	PRU1 Control
0x0002_4400	Reserved	Reserved
0x0002_6000	CFG	CFG
0x0002_8000	UART0	UART0
0x0002_A000	Reserved	Reserved
0x0002_C000	Reserved	Reserved
0x0002_E000	IEP	IEP
0x0003_0000	eCAP0	eCAP0
0x0003_2000	MII_RT_CFG	MII_RT_CFG
0x0003_2400	MII_MDIO	MII_MDIO
0x0003_4000	Reserved	Reserved
0x0003_7000	Reserved	Reserved
0x0003_8000	Reserved	Reserved
0x0003_B000	Reserved	Reserved
0x0004_0000	External PRU subsystem	External PRU subsystem
0x0008_0000	PRU-ICSS master port 0 on device L3_MAIN interconnect (OCP_HP0)(2)	PRU-ICSS master port 1 on device L3_MAIN interconnect (OCP_HP1)(2)

(1) Direct access from PRU0 to Data RAM 1 and PRU1 to Data RAM 0.
(2) For details see Section 2.8, *PRU-ICSS Memory Map.*

30.1.4.4.2 PRU-ICSS Global Memory Map

The global view of the PRU-ICSS internal memories and control ports is shown in Table 30-9. The offset addresses of each region are implemented inside the PRU-ICSS but the global device **L3_MAIN memory mapping** places the PRU-ICSS slave port in the address range shown in the external PRU-ICSS host L3_MAIN memory map.

The global memory map is with respect to the Host point of view (i.e. device MPU Cortex-A15, DSP1, etc. view of PRU-ICSS1/PRU-ICSS2 in the L3_MAIN memory space), but it can also be accessed by the PRU-ICSS1/PRU-ICSS2 itself. This is implemented via L3_MAIN redirecting PRU-ICSS master port traffic in the address range (0x0008_0000 - 0x000B_FFFF) to the PRU-ICSS slave port when PMAO_PRU0/PMAO_PRU1 = '0b1'. Note that PRU0 and PRU1 can use either the local or global addresses to access their internal memories, but using the local addresses provides access time several cycles faster than using the global addresses. This is because when accessing via the global address the access has to be routed through the L3_MAIN switch fabric outside PRU-ICSS and back in through the PRU-ICSS slave port.

Example 1: PRU1 accesses its own data RAM - Data_RAM1 in the global space:

- The PRU1 CPU sets the PRUSS_PMAO[1] PMAO_PRU1 to 1 (using PRU1 local CFG address: 0x0002_6028 of that register) and generates destination address 0x0008_2000. Thus, traffic passes through master port 1 towards PRU-ICSS1 slave port over L3_MAIN to reach Data_RAM1 (location 0x0008_2000 - 0x0008_0000 = 0x0000_2000 in the Table 30-9).

Example 2: PRU1 accesses PRU0 data RAM - Data_RAM0 in the global space:

- The PRU1 CPU sets the PRUSS_PMAO[1] PMAO_PRU1 to 1 (using PRU1 local CFG MMR address: 0x0002_6028 of that register) and generates destination address 0x0008_0000. Thus traffic passes through master port 1 towards PRU-ICSS1 slave port over L3_MAIN to reach Data_RAM0.

Example 3: DSP1 accesses the PRU0_IRAM in the global memory space to load instructions to be executed by the PRU0 upon boot time:

- Because the DSP1 is an external host to PRU-ICSS1, it has to target at first place the PRU-ICSS configuration and memory space in the L3_MAIN space. For PRU-ICSS1, slave port the base address is 0x4B20_0000.
- According to the Table 30-9, the PRU0_PRUSS1_PRU0_IRAM_TARG offset is 0x0003_4000. Hereby the physical address that DSP1 must use to store the PRU0 booting instructions to PRU0_IRAM is 0x4B23_4000.

Example 4: PRU0 accesses a non-PRU-ICSS peripheral in the global space (address offset >= 0x2000_0000):

- To access the MCASP1 config space the PRU0 keeps PRUSS_PMAO[0] PMAO_PRU0 at 0b0 and generates the MCASP1 cfg slave base address 0x4580_0000. Thus traffic passes through master port 0 and reaches MCASP1 config MMRs over L3_MAIN.

Example 5: PRUSS1_PRU1 host configures the PRU-ICSS2 module PRUSS2_MII_RT:

- Note that in case of PRUSS1_PRU0 accessing a PRU-ICSS2 peripheral, it must again disable the PMAO feature (writing at 0x0002_6028 PRUSS_PMAO[1] PMAO_PRU1 =0b0) and generate the physical address through its master port (1) adding global memory space offset of the IEP (0x0002_E000 from the Table 30-9) to the PRU-ICSS2 L3_MAIN base address (0x4B28_0000). The physical address generated from PRU-ICSS1 PRU1 therefore equals 0x4B2A E000).

Each of the PRU cores can access the rest of the device memory (including memory mapped peripheral and configuration registers) using the global memory space addresses. For details on the L3_MAIN base address of the PRU-ICSS slave configuration memory space, refer to the Chapter 2, *Memory Mapping*.

Table 30-9. PRU-ICSS Global Memory Map

Offset Address	Target
0x0000_0000	Data 8 KiB RAM0
0x0000_2000	Data 8 KiB RAM1
0x0001_0000	Data 32 KiB RAM2 (shared)
0x0002_0000	PRUSS_INTC

Table 30-9. PRU-ICSS Global Memory Map (continued)

Offset Address	Target
0x0002_2000	PRU0 Control
0x0002_2400	PRU0 Debug
0x0002_4000	PRU1 Control
0x0002_4400	PRU1 Debug
0x0002_6000	CFG
0x0002_8000	UART0
0x0002_A000	Reserved
0x0002_C000	Reserved
0x0002_E000	IEP
0x0003_0000	eCAP0
0x0003_2000	MII_RT_CFG
0x0003_2400	MII_MDIO
0x0003_4000	PRU0 12 KiB IRAM
0x0003_8000	PRU1 12 KiB IRAM
0x0004_0000	External PRU-ICSS

NOTE: The 0x0008_0000-offset-subtraction feature must be enabled only in case of PRU global accesses (0x0008_0000 - 0x000B_FFFF) to resources within the PRU subsystem. The PMAO feature must be disabled when accessing PRU-ICSS external locations.

30.1.4.5 PRUSS_CFG Register Manual

This section describes the PRU-ICSS subsystem top-level registers.

30.1.4.5.1 PRUSS_CFG Instance Summary

Table 30-10. PRUSS_CFG Instance Summary

Module Name	Base Address	Size
PRUSS1_CFG	0x4B22 6000	68 Bytes
PRUSS2_CFG	0x4B2A 6000	68 Bytes

30.1.4.5.2 PRUSS_CFG Registers

30.1.4.5.2.1 PRUSS_CFG Register Summary

Table 30-11. PRUSS_CFG Registers Mapping Summary 1

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS1_CFG Physical Address
PRUSS_REVID	R	32	0x0000 0000	0x4B22 6000
PRUSS_SYSCFG	RW	32	0x0000 0004	0x4B22 6004
PRUSS_GPCFG0	RW	32	0x0000 0008	0x4B22 6008
PRUSS_GPCFG1	RW	32	0x0000 000C	0x4B22 600C
PRUSS_CGR	RW	32	0x0000 0010	0x4B22 6010
PRUSS_ISRP	RW	32	0x0000 0014	0x4B22 6014
PRUSS_ISP	RW	32	0x0000 0018	0x4B22 6018
PRUSS_IESP	RW	32	0x0000 001C	0x4B22 601C

Table 30-11. PRUSS_CFG Registers Mapping Summary 1 (continued)

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS1_CFG Physical Address
PRUSS_IECP	RW	32	0x0000 0020	0x4B22 6020
RESERVED	RW	32	0x0000 0024	0x4B22 6024
PRUSS_PMAO	RW	32	0x0000 0028	0x4B22 6028
PRUSS_MII_RT	RW	32	0x0000 002C	0x4B22 602C
PRUSS_IEPCLK	RW	32	0x0000 0030	0x4B22 6030
PRUSS_SPP	RW	32	0x0000 0034	0x4B22 6034
PRUSS_PIN_MX	RW	32	0x0000 0040	0x4B22 6040

Table 30-12. PRUSS_CFG Registers Mapping Summary 2

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS2_CFG Physical Address
PRUSS_REVID	R	32	0x0000 0000	0x4B2A 6000
PRUSS_SYSCFG	RW	32	0x0000 0004	0x4B2A 6004
PRUSS_GPCFG0	RW	32	0x0000 0008	0x4B2A 6008
PRUSS_GPCFG1	RW	32	0x0000 000C	0x4B2A 600C
PRUSS_CGR	RW	32	0x0000 0010	0x4B2A 6010
PRUSS_ISRP	RW	32	0x0000 0014	0x4B2A 6014
PRUSS_ISP	RW	32	0x0000 0018	0x4B2A 6018
PRUSS_IESP	RW	32	0x0000 001C	0x4B2A 601C
PRUSS_IECP	RW	32	0x0000 0020	0x4B2A 6020
RESERVED	RW	32	0x0000 0024	0x4B2A 6024
PRUSS_PMAO	RW	32	0x0000 0028	0x4B2A 6028
PRUSS_MII_RT	RW	32	0x0000 002C	0x4B2A 602C
PRUSS_IEPCLK	RW	32	0x0000 0030	0x4B2A 6030
PRUSS_SPP	RW	32	0x0000 0034	0x4B2A 6034
PRUSS_PIN_MX	RW	32	0x0000 0040	0x4B2A 6040

30.1.4.5.2.2 PRUSS_CFG Register Description

Table 30-13. PRUSS_REVID

Address Offset	0x0000 0000		
Physical Address	0x4B22 6000 0x4B2A 6000	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The Revision Register contains the ID and revision information.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REVISION																															

Bits	Field Name	Description	Type	Reset
31:0	REVISION	IP Revision	R	0x-(1)

(1) TI Internal data

Table 30-14. Register Call Summary for Register PRUSS_REVID

PRU-ICSS Level Resources Functional Description
• PRUSS_CFG Register Summary: [0] [1]

Table 30-15. PRUSS_SYSCFG

Address Offset	0x0000 0004		
Physical Address	0x4B22 6004 0x4B2A 6004	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The System Configuration Register defines the power IDLE and STANDBY modes.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																										SUB_MWAIT	STANDBY_INIT	STANDBY_MODE		IDLE_MODE	

Bits	Field Name	Description	Type	Reset
31:6	RESERVED		R	0x0
5	SUB_MWAIT	Status bit for wait state. 0x0 = Ready for Transaction 0x1 = Wait until 0	R	0x0
4	STANDBY_INIT	0x1 = Initiate standby sequence. 0x0 = Enable OCP master ports.	RW	0x1
3:2	STANDBY_MODE	0x0 = Force standby mode: Initiator unconditionally in standby (standby = 1) 0x1 = No standby mode: Initiator unconditionally out of standby (standby = 0) 0x2 = Smart standby mode: Standby requested by initiator depending on internal conditions 0x3 = Reserved	RW	0x2
1:0	IDLE_MODE	0x0 = Force-idle mode 0x1 = No-idle mode 0x2 = Smart-idle mode 0x3 = Reserved	RW	0x2

Table 30-16. Register Call Summary for Register PRUSS_SYSCFG

PRU-ICSS Level Resources Functional Description
• PRU-ICSS Idle and Standby States: [0] [1]
• PRUSS_CFG Register Summary: [2] [3]

Table 30-17. PRUSS_GPCFG0

Address Offset	0x0000 0008		
Physical Address	0x4B22 6008 0x4B2A 6008	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The General Purpose Configuration 0 Register defines the GPI O configuration for PRU0.		
Type	RW		

31 30	29 28 27 26	25	24	23 22 21 20	19 18 17 16	15	14	13	12 11 10 9 8	7 6 5 4 3	2	1 0
RESERVED	PR1_PRU0_GP_MUX_SEL	PRU0_GPO_SH_SEL		PRU0_GPO_DIV1	PRU0_GPO_DIV0		PRU0_GPO_MODE	PRU0_GPI_SB	PRU0_GPI_DIV1	PRU0_GPI_DIV0	PRU0_GPI_CLK_MODE	PRU0_GPI_MODE

Bits	Field Name	Description	Type	Reset
31:30	RESERVED		R	0x0
29:26	PR1_PRU0_GP_MUX_SEL	Reserved. Keep at reset value.	R/W	0x0
25	PRU0_GPO_SH_SEL	Defines which shadow register is currently getting used for GPO shifting. 0x0 = gpo_sh0 is selected 0x1 = gpo_sh1 is selected	R	0x0
24:20	PRU0_GPO_DIV1	Divisor value (divide by PRU0_GPO_DIV1 + 1). 0x00 = div 1.0 0x01 = div 1.5 0x02 = div 2.0 .. 0x1e = div 16.0 0x1f = reserved	RW	0x0
19:15	PRU0_GPO_DIV0	Divisor value (divide by PRU0_GPO_DIV0 + 1). 0x00 = div 1.0 0x01 = div 1.5 0x02 = div 2.0 .. 0x1e = div 16.0 0x1f = reserved	RW	0x0
14	PRU0_GPO_MODE	0x0 = Direct output mode 0x1 = Serial output mode	RW	0x0
13	PRU0_GPI_SB	Start Bit event for 28-bit shift mode. PRU0_GPI_SB (pru0_r31_status[29]) is set when first capture of a 1 on pru0_r31_status[0]. Read 1: Start Bit event occurred. Read 0: Start Bit event has not occurred. Write 1: Will clear PRU0_GPI_SB and clear the whole shift register. Write 0: No Effect.	RW	0x0
12:8	PRU0_GPI_DIV1	Divisor value (divide by PRU0_GPI_DIV1 + 1). 0x00 = div 1.0 0x01 = div 1.5 0x02 = div 2.0 .. 0x1e = div 16.0 0x1f = reserved	RW	0x0
7:3	PRU0_GPI_DIV0	Divisor value (divide by PRU0_GPI_DIV0 + 1). 0x00 = div 1.0 0x01 = div 1.5 0x02 = div 2.0 .. 0x1e = div 16.0 0x1f = reserved	RW	0x0
2	PRU0_GPI_CLK_MODE	Parallel 16-bit capture mode clock edge. 0x0 = Use the positive edge of pru0_r31_status[16] 0x1 = Use the negative edge of pru0_r31_status[16]	RW	0x0
1:0	PRU0_GPI_MODE	0x0 = Direct input mode 0x1 = 16-bit parallel capture mode 0x2 = 28-bit shift mode 0x3 = MII_RT mode	RW	0x0

Table 30-18. Register Call Summary for Register PRUSS_GPCFG0

PRU-ICSS Environment

- PRU-ICSS I/O Interface:

PRU-ICSS Level Resources Functional Description

- Module Clock Configurations at PRU-ICSS Top Level: [1] [2] [3] [4] [5]
- Other PRU-ICSS Module Functional Registers at Subsystem Level: [6] [7] [8] [9]
- PRUSS_CFG Register Summary: [10] [11]

Table 30-19. PRUSS_GPCFG1

Address Offset	0x0000 000C		
Physical Address	0x4B22 600C 0x4B2A 600C	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The General Purpose Configuration 1 Register defines the GPI O configuration for PRU1.		
Type	RW		

31 30	29 28 27 26	25	24	23 22 21 20	19 18 17 16	15	14	13	12 11 10 9 8	7 6 5 4 3	2	1 0
RESERVED	PR1_PRU1_GP_MUX_SEL	PRU1_GPO_SH_SEL		PRU1_GPO_DIV1	PRU1_GPO_DIV0		PRU1_GPO_MODE	PRU1_GPI_SB	PRU1_GPI_DIV1	PRU1_GPI_DIV0	PRU1_GPI_CLK_MODE	PRU1_GPI_MODE

Bits	Field Name	Description	Type	Reset
31:30	RESERVED		R	0x0
29:26	PR1_PRU1_GP_MUX_SEL	Reserved. Keep at reset value.	R/W	0x0
25	PRU1_GPO_SH_SEL	Defines which shadow register is currently getting used for GPO shifting. 0x0 = gpo_sh0 is selected 0x1 = gpo_sh1 is selected	R	0x0
24:20	PRU1_GPO_DIV1	Divisor value (divide by PRU1_GPO_DIV1 + 1). 0x00 = div 1.0 0x01 = div 1.5 0x02 = div 2.0 .. 0x1e = div 16.0 0x1f = reserved	RW	0x0
19:15	PRU1_GPO_DIV0	Divisor value (divide by PRU1_GPO_DIV0 + 1). 0x00 = div 1.0 0x01 = div 1.5 0x02 = div 2.0 .. 0x1e = div 16.0 0x1f = reserved	RW	0x0
14	PRU1_GPO_MODE	0x0 = Direct output mode 0x1 = Serial output mode	RW	0x0
13	PRU1_GPI_SB	28-bit shift mode Start Bit event. PRU1_GPI_SB (pru1_r31_status[29]) is set when first capture of a 1 on pru1_r31_status[0]. Read 1: Start Bit event occurred. Read 0: Start Bit event has not occurred. Write 1: Will clear PRU1_GPI_SB and clear the whole shift register. Write 0: No Effect.	RW	0x0
12:8	PRU1_GPI_DIV1	Divisor value (divide by PRU1_GPI_DIV1 + 1). 0x00 = div 1.0 0x01 = div 1.5 0x02 = div 2.0 .. 0x1e = div 16.0 0x1f = reserved	RW	0x0
7:3	PRU1_GPI_DIV0	Divisor value (divide by PRU1_GPI_DIV0 + 1). 0x00 = div 1.0 0x01 = div 1.5 0x02 = div 2.0 .. 0x1e = div 16.0 0x1f = reserved	RW	0x0
2	PRU1_GPI_CLK_MODE	Parallel 16-bit capture mode clock edge. 0x0 = Use the positive edge of pru1_r31_status[16] 0x1 = Use the negative edge of pru1_r31_status[16]	RW	0x0
1:0	PRU1_GPI_MODE	0x0 = Direct input mode 0x1 = 16-bit parallel capture mode 0x2 = 28-bit shift mode 0x3 = MII_RT mode	RW	0x0

Table 30-20. Register Call Summary for Register PRUSS_GPCFG1

PRU-ICSS Environment

- PRU-ICSS I/O Interface:

PRU-ICSS Level Resources Functional Description

- Module Clock Configurations at PRU-ICSS Top Level: [2] [3] [4] [5] [6]
- Other PRU-ICSS Module Functional Registers at Subsystem Level: [7] [8] [9] [10]
- PRUSS_CFG Register Summary: [11] [12]

Table 30-21. PRUSS_CGR

Address Offset	0x0000 0010		
Physical Address	0x4B22 6010 0x4B2A 6010	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The Clock Gating Register controls the state of Clock Management of the different modules. Software should not clear {module}_CLK_EN until {module}_CLK_STOP_ACK is 0x1.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														IEP_CLK_EN	IEP_CLK_STOP_ACK	IEP_CLK_STOP_REQ	ECAP_CLK_EN	ECAP_CLK_STOP_ACK	ECAP_CLK_STOP_REQ	UART_CLK_EN	UART_CLK_STOP_ACK	UART_CLK_STOP_REQ	PRUSS_INTC_CLK_EN	PRUSS_INTC_CLK_STOP_ACK	PRUSS_INTC_CLK_STOP_REQ	PRU1_CLK_EN	PRU1_CLK_STOP_ACK	PRU1_CLK_STOP_REQ	PRU0_CLK_EN	PRU0_CLK_STOP_ACK	PRU0_CLK_STOP_REQ

Bits	Field Name	Description	Type	Reset
31:18	RESERVED		R	0x0
17	IEP_CLK_EN	IEP clock enable. 0x0 = Disable Clock 0x1 = Enable Clock	RW	0x1
16	IEP_CLK_STOP_ACK	Acknowledgement that IEP clock can be stopped. 0x0 = Not Ready to Gate Clock 0x1 = Ready to Gate Clock	R	0x0
15	IEP_CLK_STOP_REQ	IEP request to stop clock. 0x0 = do not request to stop Clock 0x1 = request to stop Clock	RW	0x0
14	ECAP_CLK_EN	ECAP clock enable. 0x0 = Disable Clock 0x1 = Enable Clock	RW	0x1
13	ECAP_CLK_STOP_ACK	Acknowledgement that ECAP clock can be stopped. 0x0 = Not Ready to Gate Clock 0x1 = Ready to Gate Clock	R	0x0
12	ECAP_CLK_STOP_REQ	ECAP request to stop clock. 0x0 = do not request to stop Clock 0x1 = request to stop Clock	RW	0x0
11	UART_CLK_EN	UART clock enable. 0x0 = Disable Clock 0x1 = Enable Clock	RW	0x1
10	UART_CLK_STOP_ACK	Acknowledgement that UART clock can be stopped. 0x0 = Not Ready to Gate Clock 0x1 = Ready to Gate Clock	R	0x0
9	UART_CLK_STOP_REQ	UART request to stop clock. 0x0 = do not request to stop Clock 0x1 = request to stop Clock	RW	0x0
8	PRUSS_INTC_CLK_EN	PRUSS_INTC clock enable. 0x0 = Disable Clock 0x1 = Enable Clock	RW	0x1
7	PRUSS_INTC_CLK_STOP_ACK	Acknowledgement that PRUSS_INTC clock can be stopped. 0x0 = Not Ready to Gate Clock 0x1 = Ready to Gate Clock	R	0x0
6	PRUSS_INTC_CLK_STOP_REQ	PRUSS_INTC request to stop clock. 0x0 = do not request to stop Clock 0x1 = request to stop Clock	RW	0x0
5	PRU1_CLK_EN	PRU1 clock enable. 0x0 = Disable Clock 0x1 = Enable Clock	RW	0x1
4	PRU1_CLK_STOP_ACK	Acknowledgement that PRU1 clock can be stopped. 0x0 = Not Ready to Gate Clock 0x1 = Ready to Gate Clock	R	0x0
3	PRU1_CLK_STOP_REQ	PRU1 request to stop clock. 0x0 = do not request to stop Clock 0x1 = request to stop Clock	RW	0x0
2	PRU0_CLK_EN	PRU0 clock enable. 0x0 = Disable Clock 0x1 = Enable Clock	RW	0x1
1	PRU0_CLK_STOP_ACK	Acknowledgement that PRU0 clock can be stopped. 0x0 = Not Ready to Gate Clock 0x1 = Ready to Gate Clock	R	0x0
0	PRU0_CLK_STOP_REQ	PRU0 request to stop clock. 0x0 = do not request to stop Clock 0x1 = request to stop Clock	RW	0x0

Table 30-22. Register Call Summary for Register PRUSS_CGR

PRU-ICSS Level Resources Functional Description

- PRU-ICSS Power and Clock Management: [0]
- PRU-ICSS Idle and Standby States: [1] [2]
- PRUSS_CFG Register Summary: [3] [4]

Table 30-23. PRUSS_ISRP

Address Offset	0x0000 0014		
Physical Address	0x4B22 6014 0x4B2A 6014	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The IRQ Status Raw Parity register is a snapshot of the IRQ raw status for the PRUSS memory parity events. The raw status is set even if the event is not enabled.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												RAM_PE_RAW				PRU1_DMEM_PE_RAW				PRU1_IMEM_PE_RAW				PRU0_DMEM_PE_RAW				PRU0_IMEM_PE_RAW			

Bits	Field Name	Description	Type	Reset
31:20	RESERVED		R	0
19:16	RAM_PE_RAW	RAM Parity Error RAW for Byte3, Byte2, Byte1, Byte0. Note RAM_PE_RAW[0] maps to Byte0. Write 0: No action. Read 0: No event pending. Read 1: Event pending. Write 1: Set event (debug).	RW	0x0
15:12	PRU1_DMEM_PE_RAW	PRU1 DMEM Parity Error RAW for Byte3, Byte2, Byte1, Byte0. Note PRU1_DMEM_PE_RAW[0] maps to Byte0. Write 0: No action. Read 0: No event pending. Read 1: Event pending. Write 1: Set event (debug).	RW	0x0
11:8	PRU1_IMEM_PE_RAW	PRU1 IMEM Parity Error RAW for Byte3, Byte2, Byte1, Byte0. Note PRU1_IMEM_PE_RAW[0] maps to Byte0. Write 0: No action. Read 0: No event pending. Read 1: Event pending. Write 1: Set event (debug).	RW	0x0
7:4	PRU0_DMEM_PE_RAW	PRU0 DMEM Parity Error RAW for Byte3, Byte2, Byte1, Byte0. Note PRU0_DMEM_PE_RAW[0] maps to Byte0. Write 0: No action. Read 0: No event pending. Read 1: Event pending. Write 1: Set event (debug) .	RW	0x0
3:0	PRU0_IMEM_PE_RAW	PRU0 IMEM Parity Error RAW for Byte3, Byte2, Byte1, Byte0. Note PRU0_IRAM_PE_RAW[0] maps to Byte0. Write 0: No action. Read 0: No event pending. Read 1: Event pending. Write 1: Set event (debug) .	RW	0x0

Table 30-24. Register Call Summary for Register PRUSS_ISRP

PRU-ICSS Level Resources Functional Description

- Other PRU-ICSS Module Functional Registers at Subsystem Level: [0] [1]
- PRUSS_CFG Register Summary: [2] [3]

Table 30-25. PRUSS_ISP

Address Offset	0x0000 0018		
Physical Address	0x4B22 6018 0x4B2A 6018	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The IRQ Status Parity Register is a snapshot of the IRQ status for the PRUSS memory parity events. The status is set only if the event is enabled. Write 1 to clear the status after the interrupt has been serviced.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												RAM_PE				PRU1_DMEM_PE				PRU1_IMEM_PE				PRU0_DMEM_PE				PRU0_IMEM_PE			

Bits	Field Name	Description	Type	Reset
31:20	RESERVED		R	0x0
19:16	RAM_PE	RAM Parity Error for Byte3, Byte2, Byte1, Byte0. Note RAM_PE[0] maps to Byte0. Write 0: No action. Read 0: No (enabled) event pending. Read 1: Event pending. Write 1: Clear event.	RW	0x0
15:12	PRU1_DMEM_PE	PRU1 DMEM Parity Error for Byte3, Byte2, Byte1, Byte0. Note PRU1_DMEM_PE[0] maps to Byte0. Write 0: No action. Read 0: No (enabled) event pending. Read 1: Event pending. Write 1: Clear event.	RW	0x0
11:8	PRU1_IMEM_PE	PRU1 IMEM Parity Error for Byte3, Byte2, Byte1, Byte0. Note PRU1_IMEM_PE[0] maps to Byte0. Write 0: No action. Read 0: No (enabled) event pending. Read 1: Event pending. Write 1: Clear event.	RW	0x0
7:4	PRU0_DMEM_PE	PRU0 DMEM Parity Error for Byte3, Byte2, Byte1, Byte0. Note PRU0_DMEM_PE[0] maps to Byte0. Write 0: No action. Read 0: No(enabled) event pending. Read 1: Event pending. Write 1: Clear event.	RW	0x0
3:0	PRU0_IMEM_PE	PRU0 IMEM Parity Error for Byte3, Byte2, Byte1, Byte0. Note PRU0_IMEM_PE[0] maps to Byte0. Write 0: No action. Read 0: No (enabled) event pending. Read 1: Event pending. Write 1: Clear event.	RW	0x0

Table 30-26. Register Call Summary for Register PRUSS_ISP

PRU-ICSS Level Resources Functional Description

- Other PRU-ICSS Module Functional Registers at Subsystem Level: [0] [1]
- PRUSS_CFG Register Summary: [2] [3]

Table 30-27. PRUSS_IESP

Address Offset	0x0000 001C		
Physical Address	0x4B22 601C 0x4B2A 601C	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The IRQ Enable Set Parity Register enables the IRQ PRUSS memory parity events.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												RAM_PE_SET				PRU1_DMEM_PE_SET				PRU1_IMEM_PE_SET				PRU0_DMEM_PE_SET				PRU0_IMEM_PE_SET			

Bits	Field Name	Description	Type	Reset
31:20	RESERVED		R	0x0
19:16	RAM_PE_SET	RAM Parity Error Set Enable for Byte3, Byte2, Byte1, Byte0. Note RAM_PE_SET[0] maps to Byte0. Write 0: No action. Read 0: Interrupt disabled (masked). Read 1: Interrupt enabled. Write 1: Enable interrupt.	RW	0x0
15:12	PRU1_DMEM_PE_SET	PRU1 DMEM Parity Error Set Enable for Byte3, Byte2, Byte1, Byte0. Note PRU1_DMEM_PE_SET[0] maps to Byte0. Write 0: No action. Read 0: Interrupt disabled (masked). Read 1: Interrupt enabled. Write 1: Enable interrupt.	RW	0x0
11:8	PRU1_IMEM_PE_SET	PRU1 IMEM Parity Error Set Enable for Byte3, Byte2, Byte1, Byte0. Note PRU1_IMEM_PE_SET[0] maps to Byte0. Write 0: No action. Read 0: Interrupt disabled (masked). Read 1: Interrupt enabled. Write 1: Enable interrupt.	RW	0x0
7:4	PRU0_DMEM_PE_SET	PRU0 DMEM Parity Error Set Enable for Byte3, Byte2, Byte1, Byte0. Note PRU0_DMEM_PE_SET[0] maps to Byte0. Write 0: No action. Read 0: Interrupt disabled (masked). Read 1: Interrupt enabled. Write 1: Enable interrupt.	RW	0x0
3:0	PRU0_IMEM_PE_SET	PRU0 IMEM Parity Error Set Enable for Byte3, Byte2, Byte1, Byte0. Note PRU0_IMEM_PE_SET[0] maps to Byte0. Write 0: No action. Read 0: Interrupt disabled (masked). Read 1: Interrupt enabled. Write 1: Enable interrupt.	RW	0x0

Table 30-28. Register Call Summary for Register PRUSS_IESP

PRU-ICSS Level Resources Functional Description

- Other PRU-ICSS Module Functional Registers at Subsystem Level: [0] [1]
- PRUSS_CFG Register Summary: [2] [3]

Table 30-29. PRUSS_IECP

Address Offset	0x0000 0020		
Physical Address	0x4B22 6020 0x4B2A 6020	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The IRQ Enable Clear Parity Register disables the IRQ PRUSS memory parity events.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PRU1_DMEM_PE_CLR				PRU1_IMEM_PE_CLR				PRU0_DMEM_PE_CLR				PRU0_IMEM_PE_CLR			

Bits	Field Name	Description	Type	Reset
31:16	RESERVED		R	0x0
15:12	PRU1_DMEM_PE_CLR	PRU1 DMEM Parity Error Clear Enable for Byte3, Byte2, Byte1, Byte0. Note PRU1_DMEM_PE_CLR[0] maps to Byte0. Write 0: No action. Read 0: Interrupt disabled (masked). Read 1: Interrupt enabled. Write 1: Disable interrupt.	RW	0x0

Bits	Field Name	Description	Type	Reset
11:8	PRU1_IMEM_PE_CLR	PRU1 IMEM Parity Error Clear Enable for Byte3, Byte2, Byte1, Byte0. Note PRU1_IMEM_PE_CLR[0] maps to Byte0. Write 0: No action. Read 0: Interrupt disabled (masked). Read 1: Interrupt enabled. Write 1: Disable interrupt.	RW	0x0
7:4	PRU0_DMEM_PE_CLR	PRU0 DMEM Parity Error Clear Enable for Byte3, Byte2, Byte1, Byte0. Note PRU0_DMEM_PE_CLR[0] maps to Byte0. Write 0: No action. Read 0: Interrupt disabled (masked). Read 1: Interrupt enabled. Write 1: Disable interrupt.	RW	0x0
3:0	PRU0_IMEM_PE_CLR	PRU0 IMEM Parity Error Clear Enable for Byte3, Byte2, Byte1, Byte0. Note PRU0_IMEM_PE_CLR[0] maps to Byte0. Write 0: No action . Read 0: Interrupt disabled (masked). Read 1: Interrupt enabled. Write 1: Disable interrupt.	RW	0x0

Table 30-30. Register Call Summary for Register PRUSS_IECP

PRU-ICSS Level Resources Functional Description

- Other PRU-ICSS Module Functional Registers at Subsystem Level: [0] [1]
- PRUSS_CFG Register Summary: [2] [3]

Table 30-31. PRUSS_PMAO

Address Offset	0x0000 0028		
Physical Address	0x4B22 6028 0x4B2A 6028	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The PRU Master OCP Address Offset Register enables for the PRU OCP Master Port Address to have an offset of minus 0x0008_0000. This enables the PRU to access External Host address space starting at 0x0000_0000.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														PMAO_PRU1	PMAO_PRU0

Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0
1	PMAO_PRU1	PRU1 OCP Master Port Address Offset Enable. 0x0 = Disable address offset. 0x1 = Enable address offset of -0x0008_0000.	RW	0x0
0	PMAO_PRU0	PRU0 OCP Master Port Address Offset Enable. 0x0 = Disable address offset. 0x1 = Enable address offset of -0x0008_0000.	RW	0x0

Table 30-32. Register Call Summary for Register PRUSS_PMAO

PRU-ICSS Level Resources Functional Description

- Other PRU-ICSS Module Functional Registers at Subsystem Level: [0]
- PRU-ICSS Local Data Memory Map: [1] [2]
- PRU-ICSS Global Memory Map: [3] [4] [5] [6]
- PRUSS_CFG Register Summary: [7] [8]

Table 30-33. PRUSS_MII_RT

Address Offset	0x0000 002C		
Physical Address	0x4B22 602C 0x4B2A 602C	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The MII_RT Event Enable Register enables MII_RT mode events to the PRUSS.PRUSS_INTC.		
Type	RW		

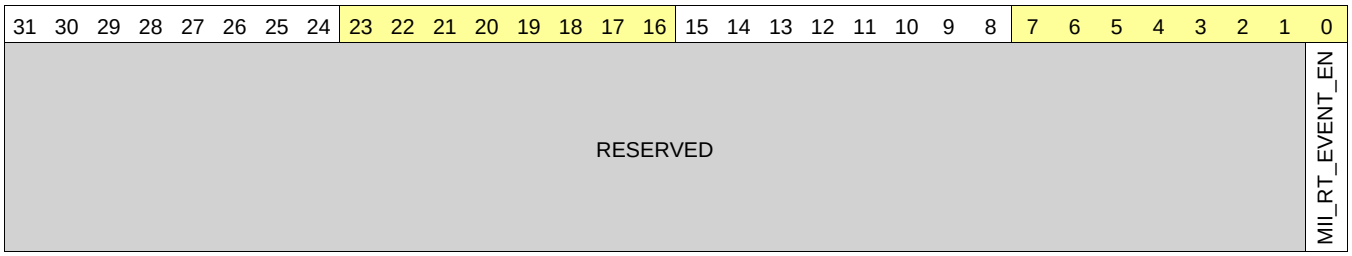


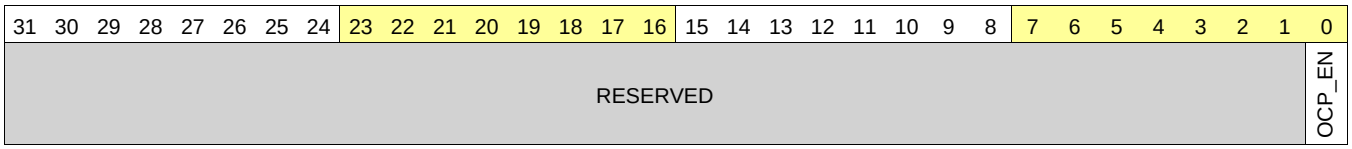
Table 30-34. Register Call Summary for Register PRUSS_MII_RT

PRU-ICSS Level Resources Functional Description
- Other PRU-ICSS Module Functional Registers at Subsystem Level: [0]
- PRUSS_CFG Register Summary: [1] [2]

PRU-ICSS Local Interrupt Controller
- PRU-ICSS Interrupt Controller System Events: [3]
- PRU-ICSS Interrupt Requests Mapping: [4] [5] [6] [7]

Table 30-35. PRUSS_IEPCLK

Address Offset	0x0000 0030		
Physical Address	0x4B22 6030 0x4B2A 6030	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The IEP Clock Source Register defines the source of the IEP clock.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:1	RESERVED		R	0x0
0	OCP_EN	IEP clock source 0x0 = IEP_CLK is the source 0x1 = ICLK is the source. While this is selected no transactions should be active. It can only be cleared by a hardware reset.	RW	0x0

Table 30-36. Register Call Summary for Register PRUSS_IEPCLK

PRU-ICSS Level Resources Functional Description
- Module Clock Configurations at PRU-ICSS Top Level: [0]
- PRUSS_CFG Register Summary: [1] [2]

PRU-ICSS Industrial Ethernet Peripheral (IEP)
- PRU-ICSS IEP Clock Generation: [3] [4]

Table 30-37. PRUSS_SPP

Address Offset	0x0000 0034		
Physical Address	0x4B22 6034 0x4B2A 6034	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The Scratch Pad Priority and Configuration Register defines the access priority assigned to the PRU cores and configures the scratch pad XFR shift functionality.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														XFR_SHIFT_EN	PRU1_PAD_HP_EN

Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0
1	XFR_SHIFT_EN	Enables XIN XOUT shift functionality. When enabled, R0[4:0] (internal to PRU) defines the 32-bit offset for XIN and XOUT operations with the scratch pad. 0x0 = Disabled. 0x1 = Enabled.	RW	0x0
0	PRU1_PAD_HP_EN	Defines which PRU wins write cycle arbitration to a common scratch pad bank. The PRU which has higher priority will always perform the write cycle with no wait states. The lower PRU will get stalled wait states until higher PRU is not performing write cycles. If the lower priority PRU writes to the same byte has the higher priority PRU, then the lower priority PRU will over write the bytes. 0x0 = PRU0 has highest priority. 0x1 = PRU1 has highest priority.	RW	0x0

Table 30-38. Register Call Summary for Register PRUSS_SPP

PRU-ICSS Level Resources Functional Description
- Other PRU-ICSS Module Functional Registers at Subsystem Level: [0]
- PRUSS_CFG Register Summary: [1] [2]

PRU-ICSS PRU Cores
- Optional XIN/XOUT Shift: [3] [4]

Table 30-39. PRUSS_PIN_MX

Address Offset	0x0000 0040		
Physical Address	0x4B22 6040 0x4B2A 6040	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The Pin Mux Select Register defines the state of the PRUSS internal pinmuxing.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						PWM3_REMAP_EN	PWM0_REMAP_EN	RESERVED							

Bits	Field Name	Description	Type	Reset
31:10	RESERVED	Reserved. Always write 0.	R	0x0
9	PWM3_REMAP_EN	UNUSED IN THIS DEVICE	RW	0x0
8	PWM0_REMAP_EN	If enabled, host intr6 of PRUSS2 controls epwm_sync_in of PWMSS1 instead of ehrpwm1_synci device pin	RW	0x0
7:0	RESERVED	Reserved	R	0x0

Table 30-40. Register Call Summary for Register PRUSS_PIN_MX

PRU-ICSS Level Resources Functional Description

- PRUSS_CFG Register Summary: [4] [5]

30.1.5 PRU-ICSS PRU Cores

This section describes the functionality of the two Programmable Real-time Unit (PRU) processors (PRU0 and PRU1) integrated in each of the device PRUSS.

30.1.5.1 PRU Cores Overview

The PRU is a processor optimized for performing embedded tasks that require manipulation of packed memory mapped data structures, handling of system events that have tight real-time constraints and interfacing with systems external to the SoC. The PRU is both very small and very efficient at handling such tasks.

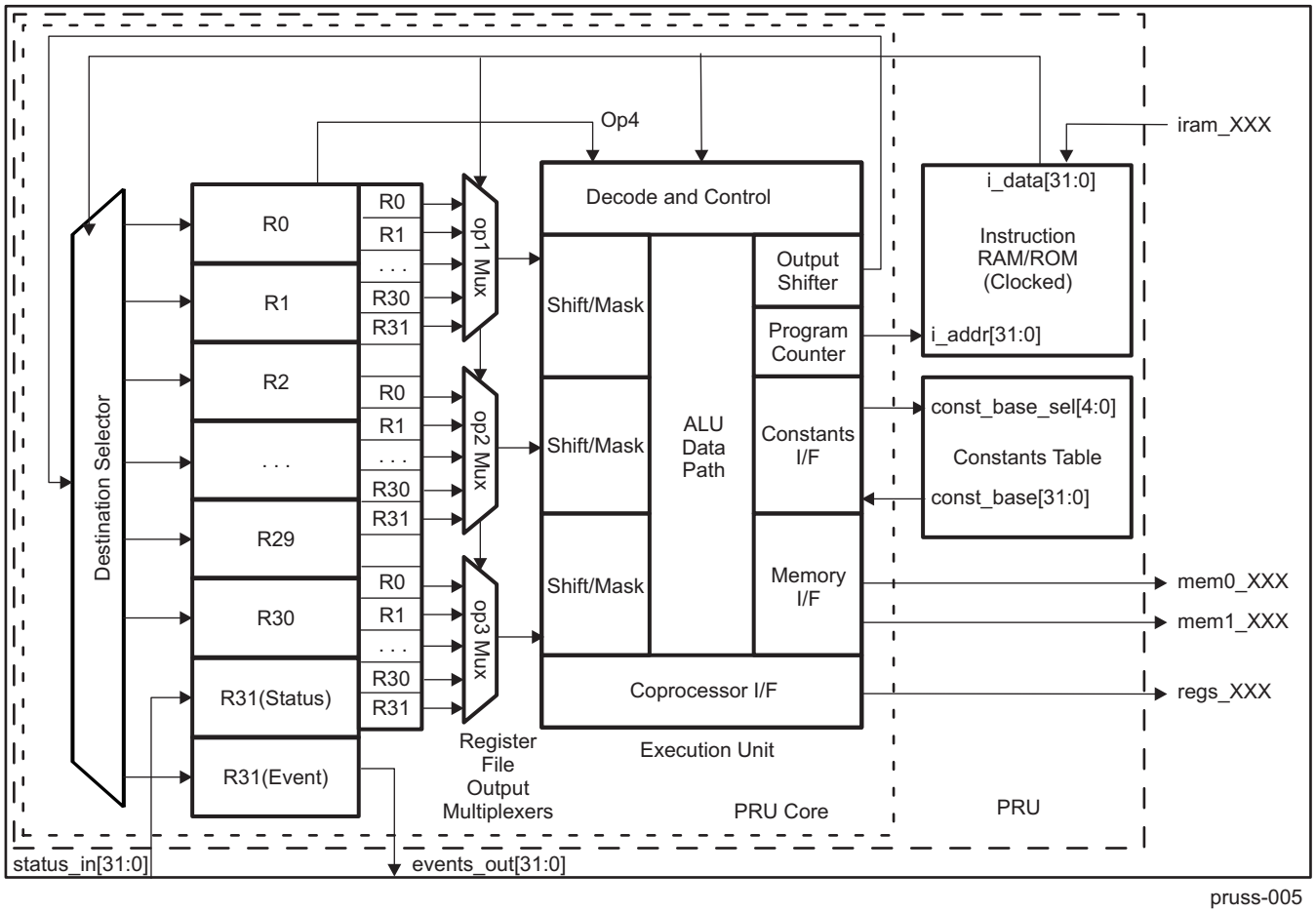
The major attributes of the PRU are in Table 30-41.

Table 30-41. PRU Features

Attribute	Value
IO Architecture	Load/Store
Data Flow Architecture	Register to Register
Core Level Bus Architecture	
Type	4-Bus Harvard (1 Instruction, 3 Data)
Instruction I/F	32-Bit
Memory I/F 0	32-Bit
Memory I/F 1	32-Bit
Execution Model	
Issue Type	Scalar
Pipelining	None (Purposefully)
Ordering	In Order
ALU Type	Unsigned Integer
Registers	
General Purpose (GP)	30 (R1 – R30)
External Status	1 (R31)
GP/Indexing	1 (R0)
Addressability in Instruction	Bit, Byte (8-bit), Half-word (16-bit), Word (32-bit), Pointer
Addressing Modes	
Load Immediate	16-bit Immediate
Load/Store – Memory	Register Base + Register Offset
	Register Base + 8-bit Immediate Offset
	Register Base with auto increment/decrement
	Constant Table Base + Register Offset
	Constant Table Base + 8-bit Immediate Offset
	Constant Table Base with auto increment/decrement
Data Path Width	32-bit
Instruction Width	32-bit
Accessibility to Internal PRU Structures	Provides 32-bit slave with three regions: • Instruction RAM • Control/Status registers • Debug access to internal registers (R0-R31) and constant table

The processor is based on a four-bus architecture which allows instructions to be fetched and executed concurrently with data transfers. In addition, an input is provided in order to allow external status information to be reflected in the internal processor status register. Figure 30-6 shows a block diagram of the processing element and the associated instruction RAM/ROM that contains the code that is to be executed.

Figure 30-6. PRU Block Diagram



30.1.5.2 PRU Cores Functional Description

This section describes the PRU cores supported functionality by describing the constant table, module interface and enhanced GPIOs.

30.1.5.2.1 PRUs Constant Table

The PRU Constants Table is a structure of hard-coded memory addresses for commonly used peripherals and memories. The constants table exists to more efficiently load/store data to these commonly accessed addresses by:

- Reduce a PRU instruction by not needing to pre-load an address into the internal register file before loading/storing data to memory address.
- Maximizing the usage of the PRU register file for embedded processing applications by moving many of the commonly used constant or deterministically calculated base addresses from the internal register file to an external table.

Table 30-42. PRU0/1 Constant Table

Entry No.	Region Pointed To	Value [31:0]
0	PRU-ICSS INTC (local)	0x0002_0000
1	Reserved	0x4804_0000
2	Reserved	0x4802_A000
3	PRU-ICSS eCAP (local)	0x0003_0000
4	PRU-ICSS CFG (local)	0x0002_6000
5	I2C3	0x4806_0000
6	Reserved	0x4803_0000
7	PRU-ICSS UART0 (local)	0x0002_8000
8	MCASP3_DAT	0x4600_0000
9	Reserved	0x4A10_0000
10	Reserved	0x4831_8000
11	Reserved	0x4802_2000
12	Reserved	0x4802_4000
13	Reserved	0x4831_0000
14	Reserved	0x481C_C000
15	Reserved	0x481D_0000
16	Reserved	0x481A_0000
17	Reserved	0x4819_C000
18	Reserved	0x4830_0000
19	Reserved	0x4830_2000
20	Reserved	0x4830_4000
21	PRU-ICSS MDIO (local)	0x0003_2400
22	Reserved	0x480C_8000
23	Reserved	0x480C_A000
24	PRU-ICSS PRU0/1 Data RAM (local)	0x0000_0n00, n = c24_blk_index[3:0]
25	PRU-ICSS PRU1/0 Data RAM (local)	0x0000_2n00, n = c25_blk_index[3:0]
26	PRU-ICSS IEP (local)	0x0002_En00, n = c26_blk_index[3:0]
27	PRU-ICSS MII_RT (local)	0x0003_2n00, n = c27_blk_index[3:0]
28	PRU-ICSS Shared RAM (local)	0x00nn_nn00, nnnn = c28_pointer[15:0]
29	OCMC_RAM2_CBUF	0x49nn_nn00, nnnn = c29_pointer[15:0]
30	OCMC_RAM	0x40nn_nn00, nnnn = c30_pointer[15:0]
31	EMIF1_SDRAM_CS0	0x80nn_nn00, nnnn = c31_pointer[15:0]

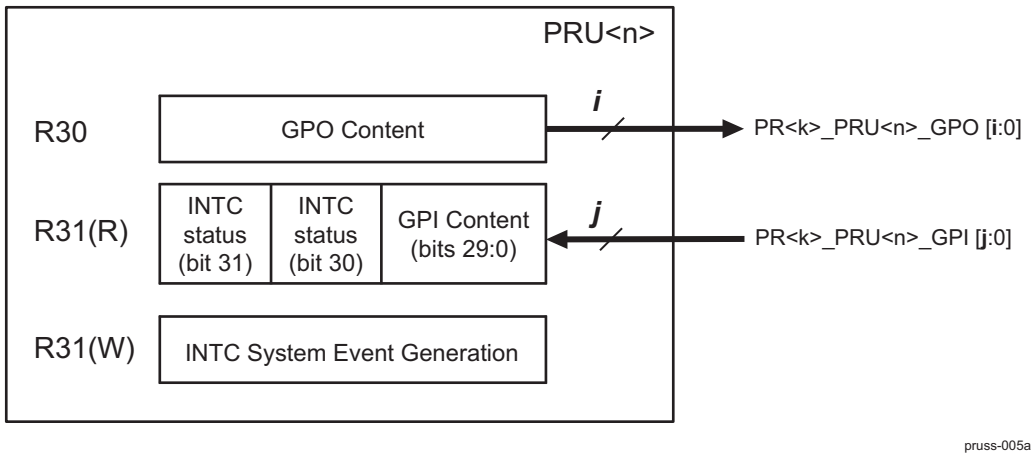
NOTE: The addresses in constants entries 24–31 are partially programmable. Their programmable bit field (for example, c24_blk_index[3:0]) is programmable through the PRU CTRL register space. As a general rule, the PRU should configure this field before using the partially programmable constant entries.

30.1.5.2.2 PRU Module Interface

The PRU module interface consists of the PRU internal registers 30 and 31 (R30 and R31). Figure 30-7 shows the PRU module interface and the functionality of R30 and R31. The register R31 serves as an interface with the dedicated PRU general purpose input (GPI) pins and PRUSS_INTC. Reading R31 returns status information from the GPI pins and PRUSS_INTC via the PRU Real Time Status Interface. Writing to R31generates PRU system events via the PRU Event Interface. The register R30 serves as an interface with the dedicated PRU general purpose output (GPO) pins.

NOTE: The below sections cover different functional modes of the PRUn cores, (where n=0,1), enhanced GPIO (EGPIO) interface. The register bits which control EGPIO functionalities are part of the (PRUSS1_CFG and PRUSS2_CFG) space. For descriptions of these EGPIO register bitfield controls, refer to the Section 30.1.4.3.

Figure 30-7. PRU Module Interface



30.1.5.2.2.1 Real-Time Status Interface Mapping (R31): Interrupt Events Input

The PRU Real Time Status Interface directly feeds information into register 31 (R31) of the PRU's internal register file. The firmware on the PRU uses the status information to make decisions during execution. The status interface is comprised of signals from different modules inside of the PRU-ICSS which require some level of interaction with the PRU. More details on the Host interrupts imported into bit 30 and 31 of register R31 of both the PRUs is provided in the Section 30.1.6, *PRU-ICSS Local Interrupt Controller*.

Table 30-43. Real-Time Status Interface Mapping (R31) Field Descriptions

Bit	Field	Description
31	pru_intr_in[1]	PRU Host Interrupt 1 from local PRUSS_INTC
30	pru_intr_in[0]	PRU Host Interrupt 0 from local PRUSS_INTC
29:0	prun_r31_status[29:0]	Status inputs from primary input via Enhanced GPI port

30.1.5.2.2.2 Event Interface Mapping (R31): PRU System Events

This PRU Event Interface directly feeds pulsed event information out of the PRU's internal ALU. These events are exported out of the PRU-ICSS and need to be connected to the system interrupt controller at the SoC level. The event interface can be used by the firmware to create software interrupts from the PRU to the Host processor.

Figure 30-8. Event Interface Mapping (R31)

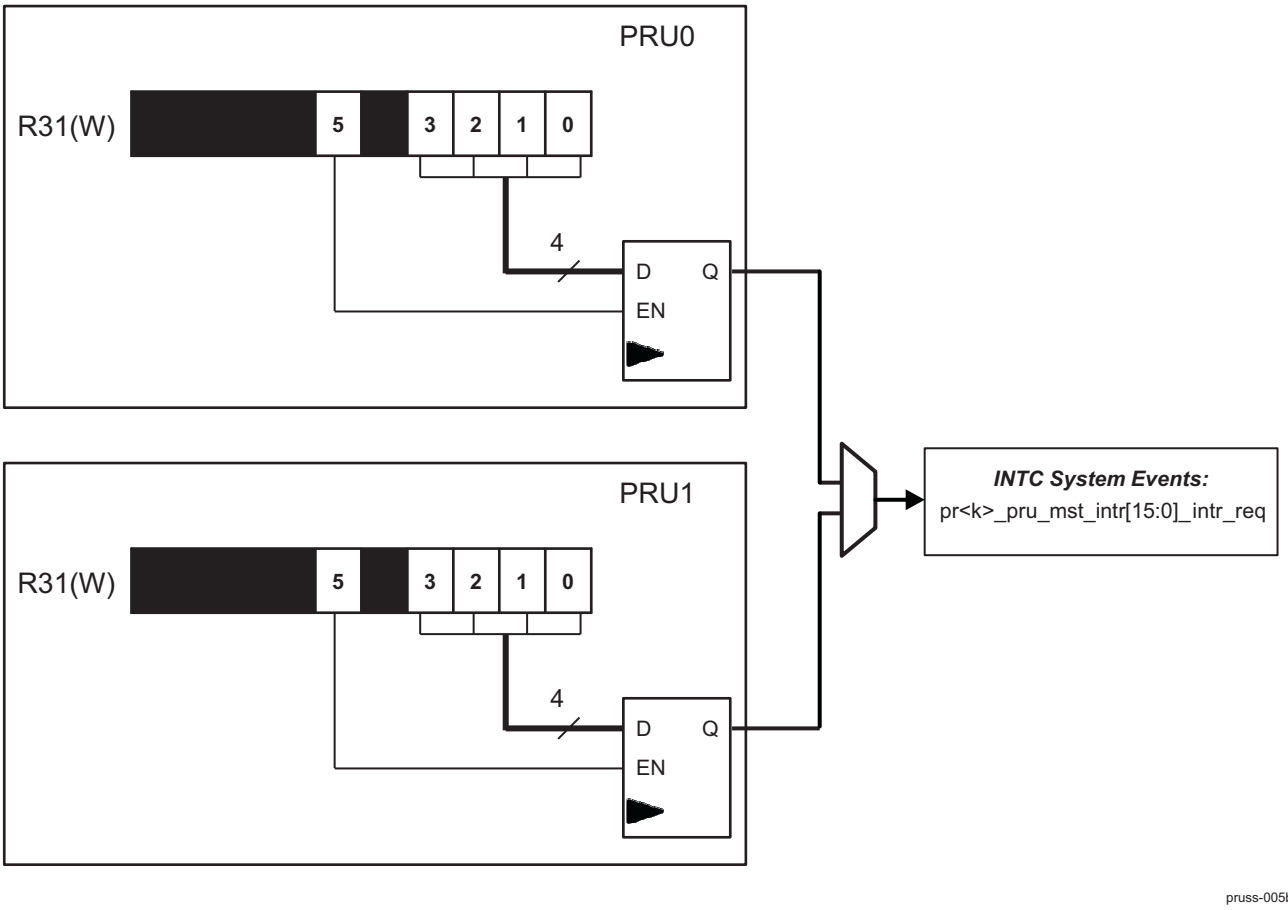


Table 30-44. Event Interface Mapping (R31) Field Descriptions

Bit	Field	Description
31:6	Reserved	
5	prun_r31_vec_valid	Valid strobe for vector output
4	Reserved	
3:0	prun_r31_vec[3:0]	Vector output

Simultaneously writing a '1' to prun_r31_vec_valid (R31 bit 5) and a channel number from 0 to 15 to prun_r31_vec[3:0] (R31 bits 3:0) creates a pulse on the output of the corresponding prk_pru_mst_intr[x]_intr_req INTC system event. For example, writing '100000' will generate a pulse on prk_pru_mst_intr[0]_intr_req, writing '100001' will generate a pulse on prk_pru_mst_intr[1]_intr_req, and so on to where writing '101111' will generate a pulse on prk_pru_mst_intr[15]_intr_req and writing '0xxxxx' will not generate any system event pulses. The output values from both PRU cores in a subsystem are ORed together.

The output channels 0-15 are connected to the PRUSS_INTC system events 16-31, respectively. This allows the PRU to assert one of the system events 16-31 by writing to its own R31 register. The system event is used to either post a completion event to one of the host CPUs (ARM) or to signal the other PRU. The host to be signaled is determined by the system interrupt to interrupt channel mapping (programmable). The 16 events are named as prk_pru_mst_intr<15:0>_intr_req. See the Section 30.1.6.4, *PRU-ICSS Interrupt Requests Mapping* , in the section, *PRU-ICSS Local Interrupt Controller*, for more details.

30.1.5.2.2.3 General-Purpose Inputs (R31): Enhanced PRU GP Module

The PRU-ICSS implements an enhanced General Purpose Input/Output (GPIO) module with SCU that supports the following general-purpose input modes: direct input, 16-bit parallel capture, 28-bit serial shift in. Register R31 serves as an interface with the general-purpose inputs. Table 30-45 describes the input modes in detail.

NOTE: Each PRU core can only be configured for one GPI mode at a time. Each mode uses the same R31 signals and internal register bits for different purposes. A summary is found in Table 30-46.

Table 30-45. PRU R31 (GPI) Modes

Mode	Function	Configuration
Direct input	GPI[20:0] feeds directly into the PRU R31	Default state
16-bit parallel capture	DATAIN[0:15] is captured by the posedge or negedge of CLOCKIN	• Enabled by CFG_GPCFGn register • CLOCKIN edge selected by CFG_GPCFGn register
28-bit shift in	DATAIN is sampled and shifted into a 28-bit shift register. Shift Counter (Cnt_16) feature uses … • Shift Counter (Cnt_16) feature is mapped to pru<n>_r31_status[28]. • SB (Start Bit detection) feature is mapped to pru<n>_r31_status[29].	• Enabled by CFG_GPCFGn register • Cnt_16 is self clearing and is connected to the PRU INTC • Start Bit (SB) is cleared by CFG_GPCFGn register

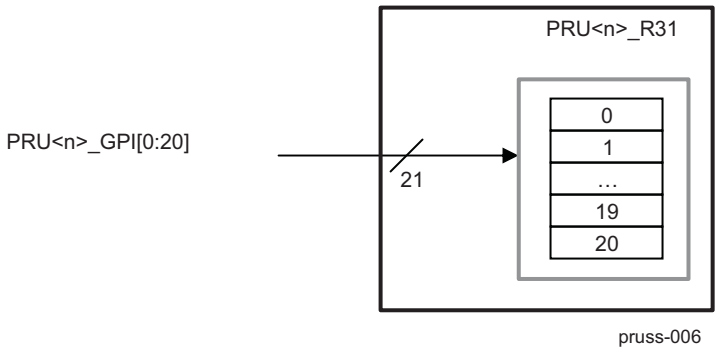
Table 30-46. PRU GPI Signals and Configurations

Pad Names at Device Level	GPI Modes		
	Direct input	Parallel Capture	28-Bit Shift in
pr<k>_pru<n>_gpi0	GPI0	DATAIN0	DATAIN
pr<k>_pru<n>_gpi1	GPI1	DATAIN1	
pr<k>_pru<n>_gpi2	GPI2	DATAIN2	
pr<k>_pru<n>_gpi3	GPI3	DATAIN3	
pr<k>_pru<n>_gpi4	GPI4	DATAIN4	
pr<k>_pru<n>_gpi5	GPI5	DATAIN5	
pr<k>_pru<n>_gpi6	GPI6	DATAIN6	
pr<k>_pru<n>_gpi7	GPI7	DATAIN7	
pr<k>_pru<n>_gpi8	GPI8	DATAIN8	
pr<k>_pru<n>_gpi9	GPI9	DATAIN9	
pr<k>_pru<n>_gpi10	GPI10	DATAIN10	
pr<k>_pru<n>_gpi11	GPI11	DATAIN11	
pr<k>_pru<n>_gpi12	GPI12	DATAIN12	
pr<k>_pru<n>_gpi13	GPI13	DATAIN13	
pr<k>_pru<n>_gpi14	GPI14	DATAIN14	
pr<k>_pru<n>_gpi15	GPI15	DATAIN15	
pr<k>_pru<n>_gpi16	GPI16	CLOCKIN	
pr<k>_pru<n>_gpi17	GPI17		
pr<k>_pru<n>_gpi18	GPI18		
pr<k>_pru<n>_gpi19	GPI19		
pr<k>_pru<n>_gpi20	GPI20		

30.1.5.2.2.3.1 PRU EGPIs Direct Input

The prun_r31_status [0:20] bits of the internal PRU register file are mapped to device-level, general purpose input pins (PRUn_GPI [0:20]). In GPI Direct Input mode, PRUn_GPI [0:20] feeds directly to prun_r31_status [0:20]. Each PRU of the PRU-ICSS has a separate mapping to device input signals - pr1_pru0_gpi[20:0] / pr2_pru0_gpi[20:0] for the PRUSS1/ PRUSS2 PRU0 core and pr1_pru1_gpi[20:0] / pr2_pru1_gpi[20:0] for the PRUSS1 / PRUSS2 PRU1 core so that there are 42 total general purpose inputs to the PRUSS1 / PRUSS2. For more details, refer also to the Section 30.1.2. See the device's system reference guide or datasheet for device specific pin mapping.Updated enhanced PRU GP module content.

Figure 30-9. PRU R31 (EGPI) Direct Input Mode Block Diagram

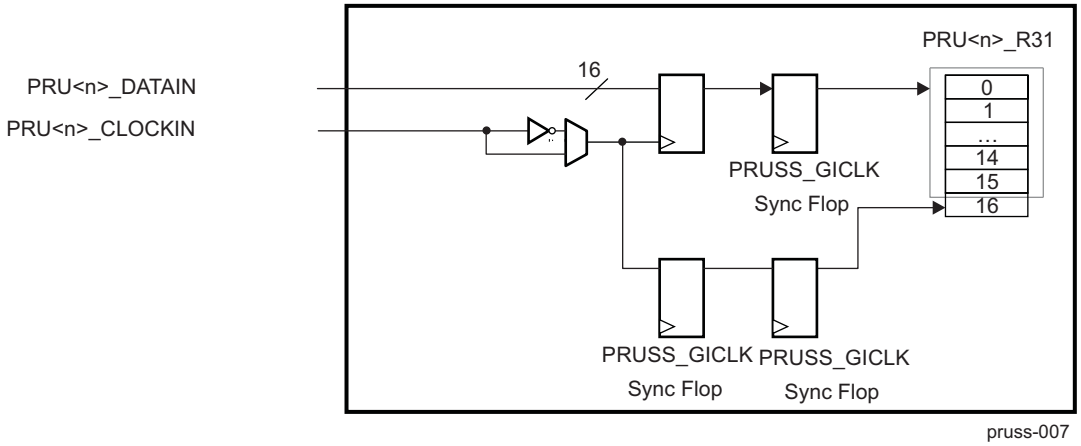


30.1.5.2.2.3.2 PRU EGPIs 16-Bit Parallel Capture

The prun_r31_status [0:15] and prun_r31_status [16] bits of the internal PRU register file mapped to device-level, general purpose input pins (PRUn_DATAIN [0:15] and PRUn_CLOCKIN, respectively). PRUn_CLOCKIN is designated for an external strobe clock, and is used to capture PRUn_DATAIN [0:15].

The PRUn_DATAIN can be captured either by the positive or the negative edge of PRUn_CLOCK, programmable through the PRU-ICSS CFG register space. If the clocking is configured through the PRUICSS CFG register to be positive, then it will equal PRU<n>_CLOCK; however, if the clocking is configured to be negative, then it will equal PRU<n>_CLOCK inverted.

Figure 30-10. PRU R31 (EGPI) 16-Bit Parallel Capture Mode Block Diagram



30.1.5.2.2.3.3 PRU EGPIs 28-Bit Shift In

In 28-bit shift in mode, the device-level, general-purpose input pin PRUn_DATAIN is sampled and shifted into a 28-bit shift register on an internal clock pulse. The register fills in LSB order (from bit 0 to 27) and then overflows into a bit bucket. The 28-bit register is mapped to prun_r31_status [0:27] and can be cleared in software through the PRU-ICSS CFG register space.

Note, the PRU will continually capture and shift the DATAIN input when the GPI mode has been set to 28-bit shift in.

The shift rate is controlled by the effective divisor of two cascaded dividers applied to the 200-MHz clock. These cascaded dividers can each be configured through the PRU-ICSS CFG register space to a value of {1, 1.5, …, 16}. Table 30-47 shows sample effective clock values and the divisor values that can be used to generate these clocks.

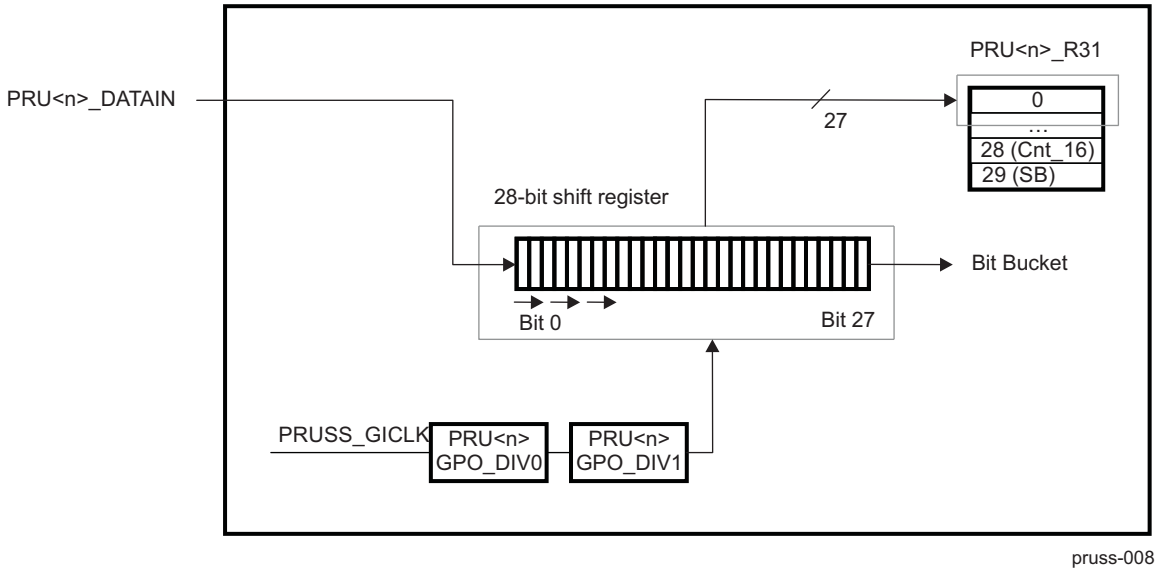
Table 30-47. PRU EGPIs Effective Clock Values

Generated clock	PRUn_GPI_DIV0	PRUn_GPI_DIV1
8-MHz	12.5 (0x17)	2 (0x02)
10-MHz	10 (0x12)	2 (0x02)
16-MHz	16 (0x1e)	1 (0x00)
20-MHz	10 (0x12)	1 (0x00)

The 28-bit shift mode also supports the following features:

- SB (Start Bit detection) is mapped to prun_r31_status[29] and is set when the first 1 is captured on PRUn_DATAIN. The SB flag in pru<n>_r31_status[29] is cleared in software through the PRU-ICSS CFG register space.
- Cnt_16 (Shift Counter) is mapped to pru<n>_r31_status[28] and is set on every 16 shift clock samples after the Start Bit has been received. CNT_16 is self clearing and is connected to the PRUSS_INTC. See the PRU-ICSS Interrupt Controller (PRUSS_INTC) section for more details.

Figure 30-11. PRU R31 (EGPI) 28-Bit Shift Mode



30.1.5.2.2.3.4 General-Purpose Outputs (R30): Enhanced PRU GP Module

The PRU-ICSS implements an enhanced General Purpose Input/Output (GPIO) module that supports two general-purpose output modes: direct output and shift out.

Table 30-48 describes these modes in detail.

NOTE: Each PRU core can only be configured for one GPO mode at a time. Each mode uses the same R30 signals and internal register bits for different purposes. A summary is found in Table 30-48.

Table 30-48. PRU R30 (EGPO) Output Mode

Mode	Function	Configuration
Direct output	pru<n>_r30[20:0] feeds directly to GPO[20:0]	Default state
Shift out	• pru<n>_r30[0] is shifted out on DATAOUT on every rising edge of pru<n>_r30[1] (CLOCKOUT). • LOAD_GPO_SH0 (Load Shadow Register 0) is mapped to pru<n>_r30[29]. • LOAD_GPO_SH1 (Load Shadow Register 1) is mapped to pru<n>_r30[30]. • ENABLE_SHIFT is mapped to pru<n>_r30[31].	Enabled by CFG_GPCFGn register

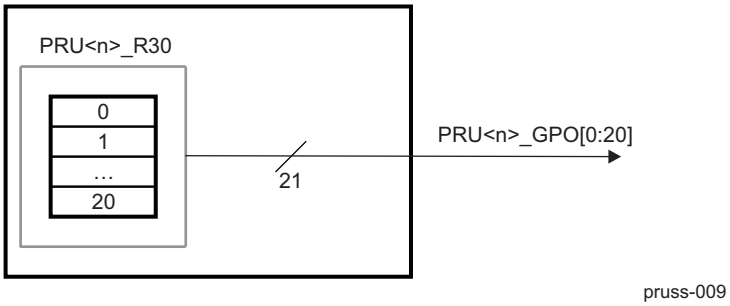
Table 30-49. GPO Mode Descriptions

Pad Names at Device Level	GPO Modes	
	Direct output	Shift out
pr<k>_pru<n>_gpo0	GPO0	DATAOUT
pr<k>_pru<n>_gpo1	GPO1	CLOCKOUT
pr<k>_pru<n>_gpo2	GPO2	
pr<k>_pru<n>_gpo3	GPO3	
pr<k>_pru<n>_gpo4	GPO4	
pr<k>_pru<n>_gpo5	GPO5	
pr<k>_pru<n>_gpo6	GPO6	
pr<k>_pru<n>_gpo7	GPO7	
pr<k>_pru<n>_gpo8	GPO8	
pr<k>_pru<n>_gpo9	GPO9	
pr<k>_pru<n>_gpo10	GPO10	
pr<k>_pru<n>_gpo11	GPO11	
pr<k>_pru<n>_gpo12	GPO12	
pr<k>_pru<n>_gpo13	GPO13	
pr<k>_pru<n>_gpo14	GPO14	
pr<k>_pru<n>_gpo15	GPO15	
pr<k>_pru<n>_gpo16	GPO16	
pr<k>_pru<n>_gpo17	GPO17	
pr<k>_pru<n>_gpo18	GPO18	
pr<k>_pru<n>_gpo19	GPO19	
pr<k>_pru<n>_gpo20	GPO20	

30.1.5.2.2.3.4.1 PRU EGPOs Direct Output

The prun_r30 [20:0] bits of the internal PRU register files are mapped to device-level, general-purpose output pins (PRUn_GPO[0:20]). In GPO Direct Output mode, prun_r30[0:20] feed directly to PRUn_GPO[0:20]. Each PRU of the PRU-ICSS has a separate mapping to pins, so that there are 42 total general-purpose outputs from the PRU-ICSS. See the device's system reference guide or datasheet for device-specific pin mapping.

Figure 30-12. PRU R30 (EGPO) Direct Output Mode Block Diagram



NOTE: R30 is not initialized after reset. To avoid unintended output signals, R30 should be initialized before pinmux configuration of PRU signals.

30.1.5.2.2.3.4.2 PRU EGPO Shift Out

In shift out mode, data is shifted out of prun_r30[0] (PRUn_DATAOUT) on every rising edge of prun_r30[1] (PRUn_CLOCK). The shift rate is controlled by the effective divisor of two cascaded dividers applied to the 200-MHz clock. These cascaded dividers can each be configured through the PRU-ICSS CFG register space to a value of {1, 1.5, …, 16}. Table 30-50 shows sample effective clock values and the divisor values that can be used to generate these clocks. Note that PRUn_CLOCKOUT is a free-running clock that starts when the PRU GPO mode is set to shift out mode.

Table 30-50. Effective Clock Values

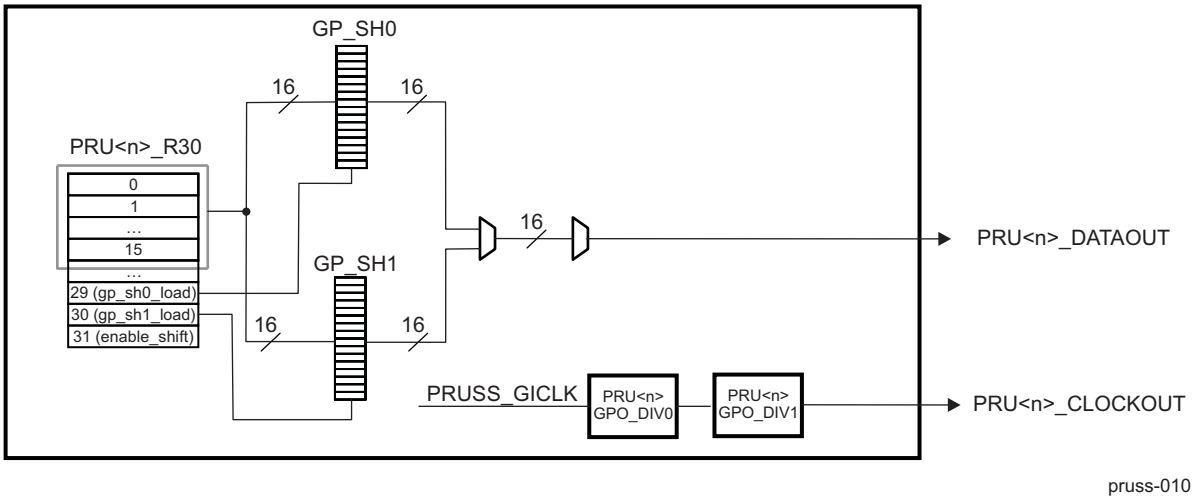
Generated Clock	PRUn_GPO_DIV0	PRUn_GPO_DIV1
8 MHz	12.5 (0x17)	2 (0x02)
10 MHz	10 (0x12)	2 (0x02)
16 MHz	16 (0x1e)	1 (0x00)
20 MHz	10 (0x12)	1 (0x00)

Shift out mode uses two 16-bit shadow registers (gpo_sh0 and gpo_sh1) to support ping-pong buffers. Each shadow register has independent load controls programmable through prun_r30[29:30] (PRUn_LOAD_GPO_SH [0:1]). While PRUn_LOAD_GPO_SH [0/1] is set, the contents of prun_r30[0:15] are loaded into gpo_sh0/1.

NOTE: If any device-level pins mapped to prun_r30[2:15] are configured for the prun_r30 [2:15] pinmux mode, then these pins will reflect the shadow register value written to prun_r30. Any pin configured for a different pinmux setting will not reflect the shadow register value written to prun_r30.

The data shift will start from the LSB of gpo_sh0 when prun_r30[31] (PRUn_ENABLE_SHIFT) is set. Note that if no new data is loaded into gpo_shnn after shift operation, the shift operation will continue looping and shifting out the pre-loaded data. When PRUn_ENABLE_SHIFT is cleared, the shift operation will finish shifting out the current shadow register, stop, and then reset.

Figure 30-13. PRU R30 (GPO) Shift Out Mode Block Diagram



Follow these steps to use the GPO shift out mode:

Step One: Initialization

1. Load 16-bits of data into gpo_sh0:
   (a) Set R30[29] = 1 (PRUn_LOAD_GPO_SH0)
   (b) Load data in R30[15:0]
   (c) Clear R30[29] to turn off load controller
2. Load 16-bits of data into gpo_sh1:
   (a) Set R30[30] = 1 (PRUn_LOAD_GPO_SH1)
   (b) Load data in R30[15:0]
   (c) Clear R30[30] to turn off load controller
3. Start shift operation:
   (a) Set R30[31] = 1 (PRUn_ENABLE_SHIFT)

Step 2: Shift Loop

1. Monitor when a shadow register has finished shifting out data and can be loaded with new data:
   (a) Poll PRUn_GPI_SH_SEL bit of the GPCFGn register
   (b) Load new 16-bits of data into gpo_sh0 if PRUn_GPI_SH_SEL = 1
   (c) Load new 16-bits of data into gpo_sh1 if PRUn_GPI_SH_SEL = 0
2. If more data to be shifted out, loop to Shift Loop
3. If no more data, exit loop

Step 3: Exit

1. End shift operation:
   (a) Clear R30[31] to turn off shift operation

NOTE: Until the shift operation is disabled, the shift loop will continue looping and shifting out the pre-loaded data if no new data has been loaded into gpo_shn.

30.1.5.3 PRU Multiplier with Optional Accumulation (MPY/MAC)

This section describes the MAC (multiplier with optional accumulation) module integrated to PRU0 and PRU1 cores of PRU-ICSS1/PRU-ICSS2.

30.1.5.3.1 PRU MACs Overview

Each of the two PRU cores (PRU0 and PRU1) has a designated unsigned multiplier with optional accumulation (MPY/MAC). The MAC supports two modes of operation: Multiply Only and Multiply and Accumulate.

The MAC is directly connected with the PRU internal registers R25-R29 and uses the broadside load/store PRU interface and XFR instructions to both control and mode of the MAC and import the multiplication results into the PRU.

30.1.5.3.1.1 PRU MAC Key Features

The MPY/MAC instantiated separately to each PRU core (PRU0 and PRU1) features:

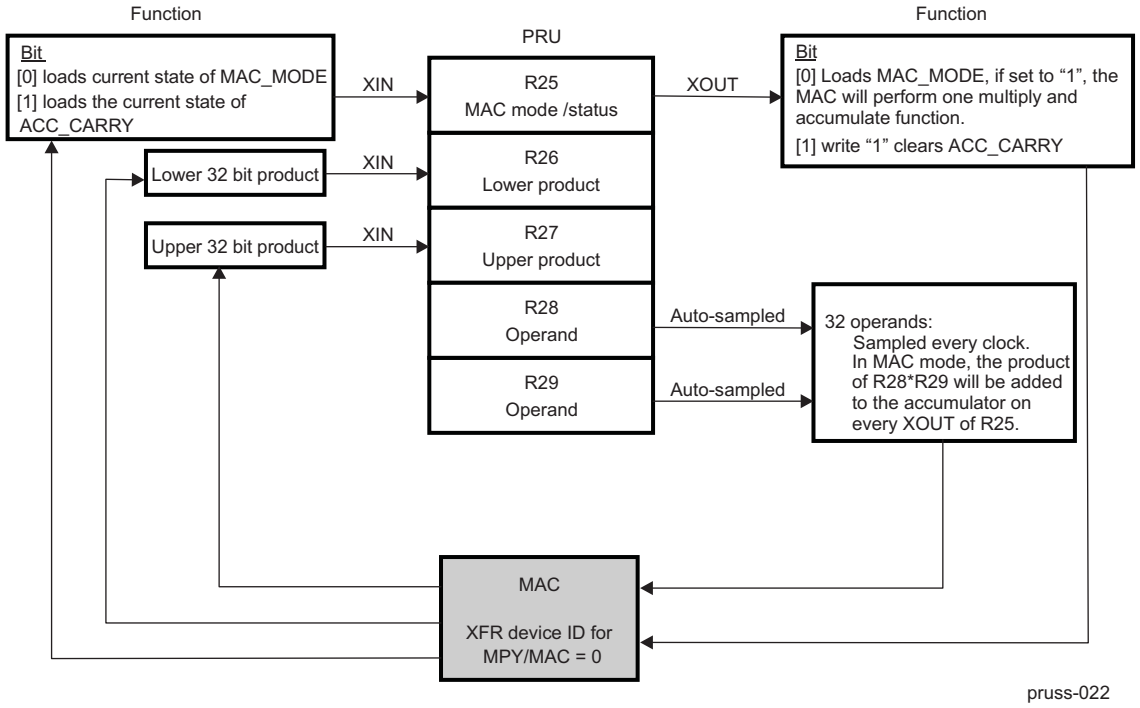
- Configurable Multiply Only and Multiply and Accumulate functionality via PRU register R25
- 32-bit operands with direct connection to PRU registers R28 and R29
- 64-bit result (with carry flag) with direct connection to PRU registers R26 and R27
- PRU broadside interface and XFR instructions (XIN, XOUT) allow for importing multiplication results and initiating accumulate function

30.1.5.3.1.2 PRU MAC Operations

30.1.5.3.1.2.1 PRU versus MAC Interface

The MAC directly connects with the PRU internal registers R25-R29 through use of the PRU broadside interface and XFR instructions. Figure 30-14 shows the functionality of each register.

Figure 30-14. Integration of the PRU and MPY/MAC



The XFR instructions (XIN and XOUT) are used to load/store register contents between the PRU core and the MAC. These instructions define the start, size, direction of the operation, and device ID. The device ID number corresponding to the MPY/MAC is shown in Table 30-51.

Table 30-51. MPY/MAC XFR ID

Device ID	Function
0	Selects MPY/MAC

The PRU register R25 is mapped to the MAC_CTRL_STATUS register (Table 30-52). The MAC's current status (MAC_MODE and ACC_CARRY states) is loaded into R25 using the XIN command on R25. The PRU sets the MAC's mode and clears the ACC_CARRY using the XOUT command on R25.

Table 30-52. MAC_CTRL_STATUS Register (R25) Field Descriptions

Bit	Field	Description
7-2	RESERVED	Reserved
1	ACC_CARRY	Write 1 to clear. 0 - 64-bit accumulator carry has not occurred 1 - 64-bit accumulator carry occurred
0	MAC_MODE	0 - Accumulation mode disabled and accumulator is cleared 1 - Accumulation mode enabled

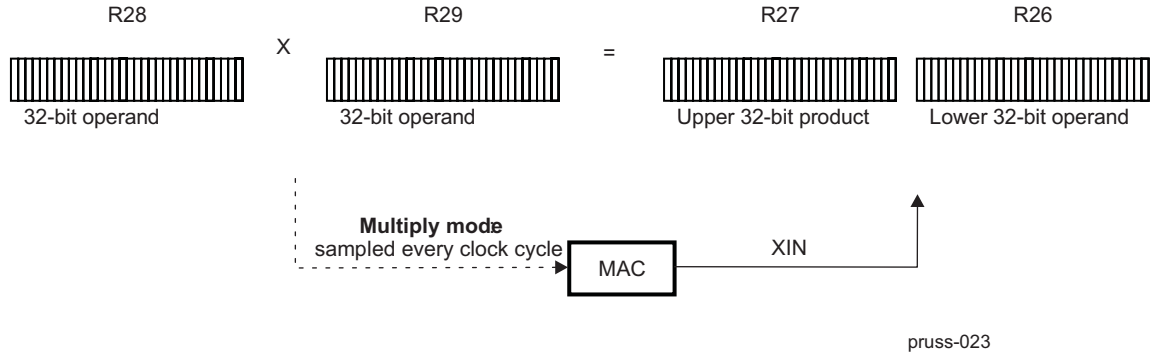
The two 32-bit operands for the multiplication are loaded into R28 and R29. These registers have a direction connection with the MAC. Therefore, XOUT is not required to load the MAC. In multiply mode, the MAC samples these registers every clock cycle. In multiply and accumulate mode, the MAC samples these registers every XOUT R25[7:0] transaction when MAC_MODE = 1.

The product from the MAC is linked to R26 (lower 32 bits) and R27 (upper 32 bits). The product is loaded into register R26 and R27 using XIN.

30.1.5.3.1.2.2 Multiply only mode(default state), MAC_MODE = 0

The Figure 30-15 summarizes the MAC operation in "Multiply-only" mode, in which the MAC multiplies the contents of R28 and R29 on every clock cycle.

Figure 30-15. MAC Multiply-only Mode- Functional Diagram



30.1.5.3.1.2.2.1 Programming PRU MAC in "Multiply-ONLY" mode

The following steps are performed by the PRU firmware for multiply-only mode:

1. Enable multiply only MAC_MODE.
   (a) Clear R25[0] for multiply only mode.
   (b) Store MAC_MODE to MAC using XOUT instruction with the following parameters:
   - Device ID = 0
   - Base register = R25
   - Size = 1
2. Load operands into R28 and R29.
3. Delay at least 1 PRU cycle before executing XIN in step 4.

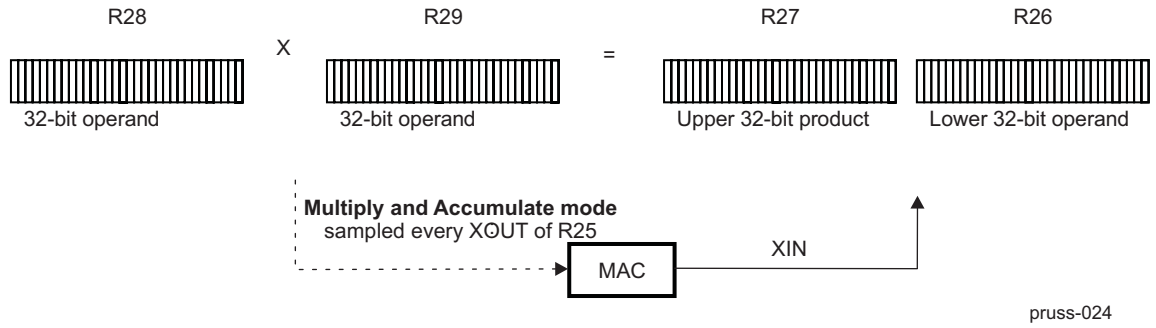
4. Load product into PRU using XIN instruction on R26, R27.

Repeat steps 2-4 for each new operand.

30.1.5.3.1.2.3 Multiply and Accumulate Mode, MAC_MODE = 1

The Figure 30-16 summarizes the MAC operation in "Multiply and Accumulate" mode. On every XOUT R25_REG[7:0] transaction, the MAC multiplies the contents of R28 and R29, adds the product to its accumulated result, and sets ACC_CARRY if an accumulation overflow occurs.

Figure 30-16. MAC Multiply and Accumulate Mode Functional Diagram



30.1.5.3.1.2.3.1 Programming PRU MAC in "Multiply and Accumulate" mode

The following steps are performed by the PRU firmware for multiply and accumulate mode:

1. Enable multiply and accumulate MAC_MODE.
   (a) Set R25[1:0] = 1 for accumulate mode.
   (b) Store MAC_MODE to MAC using XOUT instruction with the following parameters:
   - Device ID = 0
   - Base register = R25
   - Size = 1
2. Clear accumulator and carry flag.
   (a) Set R25[1:0] = 3 to clear accumulator (R25[1]=1) and preserve accumulate mode (R25[0]=1).
   (b) Store accumulator to MAC using XOUT instruction on R25.
3. Load operands into R28 and R29.
4. Multiply and accumulate, XOUT R25[1:0] = 1
   Repeat step 4 for each multiply and accumulate using same operands.
   Repeat step 3 and 4 for each multiply and accumulate for new operands.
5. Load the accumulated product into R26, R27, and the ACC_CARRY status into R25 using the XIN instruction.

NOTE: Steps one and two are required to set the accumulator mode and clear the accumulator and carry flag.

30.1.5.4 CRC16/32

The PRU0 and PRU1 cores of PRU-ICSS1/PRU-ICSS2 each have a designated CRC16/32 module.

In general, CRC adds error detection capability to communication systems. The CRC encoder appends redundant bits (or CRC bits) to the systematic data message. During reception of the data message, the received data is also encoded with the same CRC encoder. The 2 sets of CRC bits are compared together. If they match, there were no transmission errors; and if they don't match, a transmission error has been detected.

30.1.5.4.1 Features

CRC16/32 supports the following features:

- Supports CRC32:
  - $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$
- Supports CRC16:
  - $x^{16}+x^{15}+x^{2}+1$
- PRU broadside interface and XFR instructions (XIN, XOUT) allow for importing CRC results and executing accumulate function

30.1.5.4.2 PRU and CRC16/32 Interface

The CRC16/32 module directly connects with the PRU internal registers R25-R29 through use of the PRU broadside interface and XFR instructions. Table 30-53 shows the functionality of each register.

The XFR instructions (XIN and XOUT) are used to load/store register contents between the PRU core and the CRC16/32 module. These instructions define the start, size, direction of the operation, and device ID. The XFR device ID number corresponding to the CRC16/32 module is 1.

Table 30-53. CRC Register to PRU Port Mapping

CRC Register	R/W	Description	PRU Mapping
CRC_CFG	W	bit [0] CRC32_ENABLE: 0: CRC16 mode is selected. Hardware will auto-set init state of CRC_SEED to 0x0000_0000. Note CRC16 result value is only 16-bits 1: CRC32 mode is selected. Hardware will auto-set init state of CRC_SEED will be 0xffff_ffff. Always write all 4 bytes.	R25_reg
CRC_DATA_8_BFLIP	R	8-bit flip of CRC_DATA. CRC_DATA_8_BFLIP has the same byte order as CRC_DATA[31:0], but each byte has all bits flipped. CRC_DATA_32_FLIP[7:0] = CRC_DATA[0:7] CRC_DATA_32_FLIP[15:8] = CRC_DATA[8:15] CRC_DATA_32_FLIP[23:16] = CRC_DATA[16:23] CRC_DATA_32_FLIP[31:24] = CRC_DATA[24:31] For CRC16, only CRC_DATA_8_BFLIP[15:0] are valid. No auto reset on CRC_DATA_8_BFLIP read.	R27_reg
CRC_SEED	W	CRC SEED value. Hardware will auto-initialize the CRC_SEED value to 0x0000_0000 for CRC16 and 0xFFFF_FFFF for CRC32. Software only needs to initialize CRC_SEED if a different default value is required. Always write 4 bytes. Note when CRC_CFG[CRC32_ENABLE] is enabled, the hardware will switch the CRC_SEED value to 0xFFFF_FFFF. Reading the CRC_DATA register will reset the CRC value to the CRC_SEED state.	R28_reg
CRC_DATA_32_BFLIP	R	Full 32-bit flip of CRC_DATA CRC_DATA_32_BFLIP[0] = CRC_DATA[31] … CRC_DATA_32_BFLIP[31] = CRC_DATA[0] For CRC16, only CRC_DATA_32_BFLIP[31:16] are valid. No auto reset on CRC_DATA_32_BFLIP read.	R28_reg
CRC_DATA	RW	For Write, must use a fixed width throughout the session. The CRC module supports lower 8-bit, or lower 16-bit, or full 32-bit data widths. For Read, LSB or CRC_DATA[0] is first bit on the wire. Note for CRC16, only CRC_DATA[15:0] is valid. Hardware will delay CRC_DATA read operation up to 1 clock if it occurs back to back with a CRC_DATA write.	R29_reg

30.1.5.4.3 Programming Model

The following steps are performed by the PRU firmware to use the CRC module:

Step1: Configuration (optional)

1. Configure CRC type:
   For CRC32 operation, set CRC32_ENABLE using XOUT instruction with the following parameters:
   - Device ID = 1
   - Base register = R25
   - Size = 1
2. Update CRC_SEED, if required using XOUT with the following parameters:
   - Device ID = 1
   - Base register = R28
   - Size = 1 to 4

Step 2:

1. Load new CRC data into R29
2. Push CRC data to the CRC16/32 module using XOUT with the following parameters:
   - Device ID = 1
   - Base register = R29
   - Size = 1 to 4
3. Load the accumulated CRC result into the PRU using the XIN instruction with the following parameters:
   - Device ID = 1
   - Base register = R29
   - Size = 4

Repeat Step 2, numbers 1 and 2 for each new CRC data.

NOTE: When a session starts, the PRU firmware must use the same write data width throughout the session.

30.1.5.5 PRU0 and PRU1 Scratch Pad Memory

The PRU-ICSS supports a scratch pad with three independent banks accessible by the PRU cores. The PRU cores interact with the scratch pad through broadside load/store PRU interface and XFR instructions. The scratch pad can be used as a temporary place holder for the register contents of the PRU cores. Direct connection between the PRU cores is also supported for transferring register contents directly between the cores. This section describes the Scratch Pad Memory shared between and directly accessible by the PRU0 and PRU1 cores, as well as the XFR direct method used by the PRU cores of the PRU-ICSS1 /PRU-ICSS2.

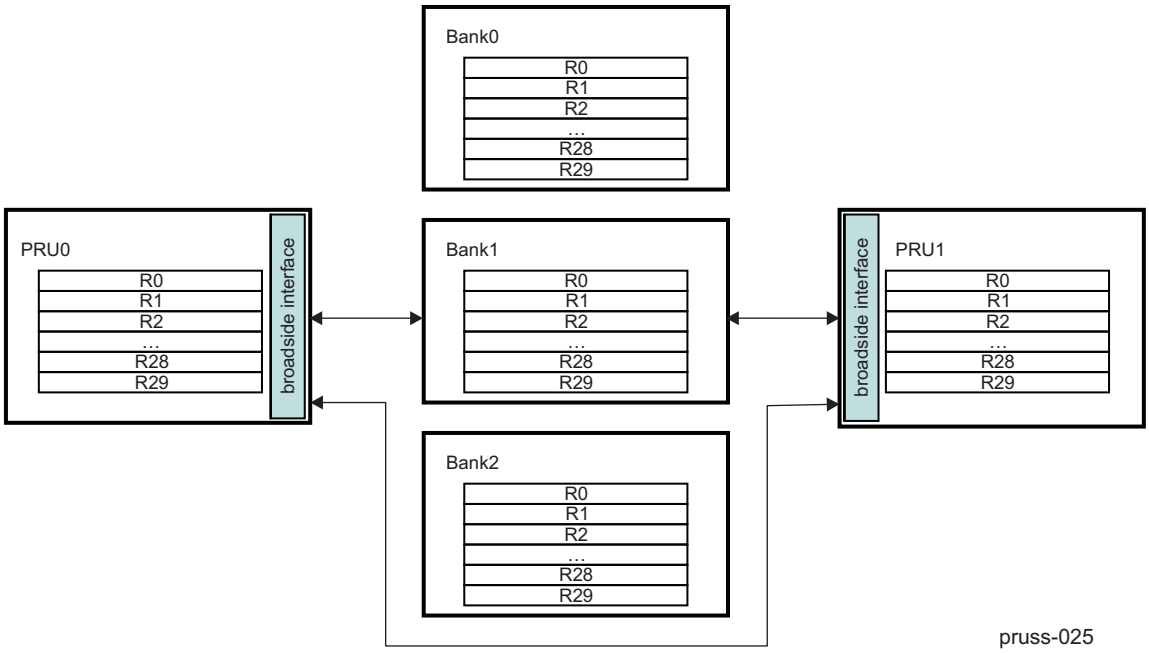
30.1.5.5.1 PRU0/1 Scratch Pad Overview

The PRU-ICSS scratch pad supports the following features:

- Three scratch pad banks of 30, 32-bit registers (R29:0)
- Flexible load/store options
  - User-defined start byte and length of the transfer
  - Length of transfer ranges from one byte of a register to the entire register content (R29 to R0)
  - Simultaneous transactions supported between PRU0 ↔ Bank<n> and PRU1 ↔ Bank<m>
  - Direct connection of PRU0 → PRU1 or PRU1 → PRU0 for all registers R29-R0
- XFR instructions operate in one clock cycle
- Optional XIN/XOUT shift functionality allows remapping of registers (R<n> → R<m>) during load/store operation

Figure 30-17 shows a simplified model of the ScratchPad integration.

Figure 30-17. ScratchPad and PRU Integration



30.1.5.5.2 PRU0 /1 Scratch Pad Operations

XFR instructions are used to load/store register contents between the PRU cores and the scratch pad banks. These instructions define the start, size, direction of the operation, and device ID. The device ID corresponds to the external source or destination (either a scratch pad bank or the other PRU core). The device ID numbers are shown in Table 30-54. Note the direct connect mode (device ID 14) can be used to synchronize the PRU cores. This mode requires the transmitting PRU core to execute XOUT and the receiving PRU core to execute XIN.

Table 30-54. Scratch Pad XFR ID

Device ID	Function
10	Selects Bank0
11	Selects Bank1
12	Selects Bank2
13	Reserved
14	Selects other PRU core (Direct connect mode)

A collision occurs when two XOUT commands simultaneously access the same asset or device ID. Table 30-55 shows the priority assigned to each operation when a collision occurs. In direct connect mode (device ID 14), any PRU transaction will be terminated if the stall is greater than 1024 cycles. This will generate the event pr<k>_xfr_timeout that is connected to INTC.

Table 30-55. Scratch Pad XFR Collision and Stall Conditions

Operation	Collision and Stall Handling
PRU <n> XOUT (→) bank[j]	If both PRU cores access the same bank simultaneously, PRU0 is given priority. PRU1 will temporarily stall until the PRU0 operation completes.
PRU <n> XOUT (→) PRU<m>	Direct connect mode requires the transmitting core (PRU<n>) to execute XOUT and the receiving core (PRU<m>) to execute XIN. If PRU<n> executes XOUT before PRU<m> executes XIN, then PRU<n> will stall until either PRU<m> executes XIN or the stall is greater than 1024 cycles.

Table 30-55. Scratch Pad XFR Collision and Stall Conditions (continued)

Operation	Collision and Stall Handling
PRU<m> XIN (←) PRU<n>	Direct connect mode requires the transmitting core (PRU<n>) to execute XOUT and the receiving core (PRU<m>) to execute XIN. If PRU<m> executes XIN before PRU<n> executes XOUT, then PRU<m> will stall until either PRU<n> executes XOUT or the stall is greater than 1024 cycles.

30.1.5.5.2.1 Optional XIN/XOUT Shift

The optional XIN/XOUT shift functionality allows register contents to be remapped or shifted within the destination's register space. For example, the contents of PRU0 R6-R8 could be remapped to Bank1 R10-12. The XIN/XOUT shift feature is not supported for direct connect mode, only for transfers between a PRU core and scratch pad bank.

The shift feature is enabled or disabled through the PRU subsystem level register PRUSS_SPP[1] XFR_SHIFT_EN bit. When enabled, R0[4:0] (internal to the PRU) defines the number of 32-bit registers in which content is shifted in the scratch pad bank. Note that scratch pad banks do not have registers R30 or R31.

The following PRU firmware examples demonstrate the shift functionality. Note these assume the XFR_SHIFT_EN bit of the PRUSS_SPP register of the PRU-ICSS CFG register space has been set.

XOUT Shift By 4 Registers

Store R4:R7 to R8:R11 in bank0:

- Load 4 into R0.b0
- XOUT using the following parameters:
  - Device ID = 10
  - Base register = R4
  - Size = 16

XOUT Shift By 9 Registers, With Wrap Around

Store R25:R29 to R4:R8 in bank1:

- Load 9 into R0.b0
- XOUT using the following parameters:
  - Device ID = 11
  - Base register = R25
  - Size = 20

XIN Shift By 10 Registers

Load R14:R16 from bank2 to R4:R6:

- Load 10 into R0.b0
- XIN using the following parameters:
  - Device ID = 12
  - Base register = R4
  - Size = 12

30.1.5.6 PRUSS_PRU_CTRL Register Manual

This section describes the PRUSS PRU0 and PRU1 cores memory mapped registers.

30.1.5.6.1 PRUSS_PRU_CTRL Instance Summary

Table 30-56. PRUSS_PRU_CTRL Instance Summary

Module Name	Base Address	Size
PRUSS1_PRU0_CTRL	0x4B22 2000	48 Bytes
PRUSS1_PRU1_CTRL	0x4B22 4000	48 Bytes
PRUSS2_PRU0_CTRL	0x4B2A 2000	48 Bytes
PRUSS2_PRU1_CTRL	0x4B2A 4000	48 Bytes

30.1.5.6.2 PRUSS_PRU_CTRL Registers

30.1.5.6.2.1 PRUSS_PRU_CTRL Register Summary

Table 30-57. PRUSS1_PRUn_CTRL Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS1_PRU0_CTRL Physical Address	PRUSS1_PRU1_CTRL Physical Address
PRU_CONTROL	RW	32	0x0000 0000	0x4B22 2000	0x4B22 4000
PRU_STATUS	R	32	0x0000 0004	0x4B22 2004	0x4B22 4004
PRU_WAKEUP_EN	RW	32	0x0000 0008	0x4B22 2008	0x4B22 4008
PRU_CYCLE	RW	32	0x0000 000C	0x4B22 200C	0x4B22 400C
PRU_STALL	RW	32	0x0000 0010	0x4B22 2010	0x4B22 4010
PRU_CTBIR0	RW	32	0x0000 0020	0x4B22 2020	0x4B22 4020
PRU_CTBIR1	RW	32	0x0000 0024	0x4B22 2024	0x4B22 4024
PRU_CTPPR0	RW	32	0x0000 0028	0x4B22 2028	0x4B22 4028
PRU_CTPPR1	RW	32	0x0000 002C	0x4B22 202C	0x4B22 402C

Table 30-58. PRUSS2_PRUn_CTRL Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS2_PRU0_CTRL Physical Address	PRUSS2_PRU1_CTRL Physical Address
PRU_CONTROL	RW	32	0x0000 0000	0x4B2A 2000	0x4B2A 4000
PRU_STATUS	R	32	0x0000 0004	0x4B2A 2004	0x4B2A 4004
PRU_WAKEUP_EN	RW	32	0x0000 0008	0x4B2A 2008	0x4B2A 4008
PRU_CYCLE	RW	32	0x0000 000C	0x4B2A 200C	0x4B2A 400C
PRU_STALL	RW	32	0x0000 0010	0x4B2A 2010	0x4B2A 4010
PRU_CTBIR0	RW	32	0x0000 0020	0x4B2A 2020	0x4B2A 4020
PRU_CTBIR1	RW	32	0x0000 0024	0x4B2A 2024	0x4B2A 4024
PRU_CTPPR0	RW	32	0x0000 0028	0x4B2A 2028	0x4B2A 4028
PRU_CTPPR1	RW	32	0x0000 002C	0x4B2A 202C	0x4B2A 402C

30.1.5.6.2.2 PRUSS_PRU_CTRL Register Description

Table 30-59. PRU_CONTROL

Address Offset	0x0000 0000		
Physical Address	0x4B22 2000 0x4B22 4000 0x4B2A 2000 0x4B2A 4000	**Instance**	PRUSS1_PRU0_CTRL PRUSS1_PRU1_CTRL PRUSS2_PRU0_CTRL PRUSS2_PRU1_CTRL
Description	CONTROL REGISTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCOUNTER_RST_VAL																RUNSTATE	BIG_ENDIAN	RESERVED					SINGLE_STEP	RESERVED				COUNTER_ENABLE	SLEEPING	ENABLE	SOFT_RST_N

Bits	Field Name	Description	Type	Reset
31:16	PCOUNTER_RST_VAL	Program Counter Reset Value: This field controls the address where the PRU will start executing code from after it is taken out of reset.	RW	0x0
15	RUNSTATE	Run State: This bit indicates whether the PRU is currently executing an instruction or is halted. 0 = PRU is halted and host has access to the instruction RAM and debug registers regions. 1 = PRU is currently running and the host is locked out of the instruction RAM and debug registers regions. This bit is used by an external debug agent to know when the PRU has actually halted when waiting for a HALT instruction to execute, a single step to finish, or any other time when the pru_enable has been cleared.	R	0x0
14	BIG_ENDIAN		R	0x0
13:9	RESERVED		R	0x0
8	SINGLE_STEP	Single Step Enable: This bit controls whether or not the PRU will only execute a single instruction when enabled. 0 = PRU will free run when enabled. 1 = PRU will execute a single instruction and then the pru_enable bit will be cleared. Note that this bit does not actually enable the PRU, it only sets the policy for how much code will be run after the PRU is enabled. The pru_enable bit must be explicitly asserted. It is legal to initialize both the single_step and pru_enable bits simultaneously. (Two independent writes are not required to cause the stated functionality.)	RW	0x0
7:4	RESERVED		R	0x0
3	COUNTER_ENABLE	PRU Cycle Counter Enable: Enables PRU cycle counters. 0 = Counters not enabled 1 = Counters enabled	RW	0x0
2	SLEEPING	PRU Sleep Indicator: This bit indicates whether or not the PRU is currently asleep. 0 = PRU is not asleep 1 = PRU is asleep If this bit is written to a 0, the PRU will be forced to power up from sleep mode.	RW	0x0
1	ENABLE	Processor Enable: This bit controls whether or not the PRU is allowed to fetch new instructions. 0 = PRU is disabled. 1 = PRU is enabled. If this bit is de-asserted while the PRU is currently running and has completed the initial cycle of a multi-cycle instruction (LBxO,SBxO,SCAN, etc.), the current instruction will be allowed to complete before the PRU pauses execution. Otherwise, the PRU will halt immediately. Because of the unpredictability timing sensitivity of the instruction execution loop, this bit is not a reliable indication of whether or not the PRU is currently running. The pru_state bit should be consulted for an absolute indication of the run state of the core. When the PRU is halted, its internal state remains coherent therefore this bit can be reasserted without issuing a software reset and the PRU will resume processing exactly where it left off in the instruction stream.	RW	0x0
0	SOFT_RST_N	Soft Reset: When this bit is cleared, the PRU will be reset. This bit is set back to 1 on the next cycle after it has been cleared.	R	0x1

Table 30-60. Register Call Summary for Register PRU_CONTROL

PRU-ICSS PRU Cores

- PRUSS_PRU_CTRL Register Summary: [0] [1]

Table 30-61. PRU_STATUS

Address Offset	0x0000 0004		
Physical Address	0x4B22 2004 0x4B22 4004 0x4B2A 2004 0x4B2A 4004	**Instance**	PRUSS1_PRU0_CTRL PRUSS1_PRU1_CTRL PRUSS2_PRU0_CTRL PRUSS2_PRU1_CTRL
Description	STATUS REGISTER		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PCOUNTER															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED		R	0x0000
15:0	PCOUNTER	Program Counter: This field is a registered (1 cycle delayed) reflection of the PRU program counter. Note that the PC is an instruction address where each instruction is a 32 bit word. This is not a byte address and to compute the byte address just multiply the PC by 4 (PC of 2 = byte address of 0x8, or PC of 8 = byte address of 0x20).	R	0x0

Table 30-62. Register Call Summary for Register PRU_STATUS

PRU-ICSS PRU Cores

- PRUSS_PRU_CTRL Register Summary: [0] [1]

Table 30-63. PRU_WAKEUP_EN

Address Offset	0x0000 0008		
Physical Address	0x4B22 2008 0x4B22 4008 0x4B2A 2008 0x4B2A 4008	**Instance**	PRUSS1_PRU0_CTRL PRUSS1_PRU1_CTRL PRUSS2_PRU0_CTRL PRUSS2_PRU1_CTRL
Description	WAKEUP ENABLE REGISTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BITWISE_ENABLES																															

Bits	Field Name	Description	Type	Reset
31:0	BITWISE_ENABLES	Wakeup Enables: This field is ANDed with the incoming R31 status inputs (whose bit positions were specified in the stmap parameter) to produce a vector which is unary ORed to produce the status_wakeup source for the core. Setting any bit in this vector will allow the corresponding status input to wake up the core when it is asserted high. The PRU should set this enable vector prior to executing a SLP (sleep) instruction to ensure that the desired sources can wake up the core.	RW	0x0

Table 30-64. Register Call Summary for Register PRU_WAKEUP_EN

PRU-ICSS PRU Cores

- PRUSS_PRU_CTRL Register Summary: [0] [1]

Table 30-65. PRU_CYCLE

Address Offset	0x0000 000C		
Physical Address	0x4B22 200C 0x4B22 400C 0x4B2A 200C 0x4B2A 400C	**Instance**	PRUSS1_PRU0_CTRL PRUSS1_PRU1_CTRL PRUSS2_PRU0_CTRL PRUSS2_PRU1_CTRL
Description	CYCLE COUNT. This register counts the number of cycles for which the PRU has been enabled.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CYCLECOUNT																															

Bits	Field Name	Description	Type	Reset
31:0	CYCLECOUNT	This value is incremented by 1 for every cycle during which the PRU is enabled and the counter is enabled (both bits ENABLE and COUNTENABLE set in the PRU control register). Counting halts while the PRU is disabled or counter is disabled, and resumes when re-eneabled. Counter clears the COUNTENABLE bit in the PRU control register when the count reaches 0xFFFFFFFF. (Count does does not wrap). The register can be read at any time. The register can be cleared when the counter or PRU is disabled. Clearing this register also clears the PRU Stall Count Register.	RW	0x0

Table 30-66. Register Call Summary for Register PRU_CYCLE

PRU-ICSS PRU Cores

- PRUSS_PRU_CTRL Register Summary: [0] [1]

Table 30-67. PRU_STALL

Address Offset	0x0000 0010		
Physical Address	0x4B22 2010 0x4B22 4010 0x4B2A 2010 0x4B2A 4010	**Instance**	PRUSS1_PRU0_CTRL PRUSS1_PRU1_CTRL PRUSS2_PRU0_CTRL PRUSS2_PRU1_CTRL
Description	STALL COUNT. This register counts the number of cycles for which the PRU has been enabled, but unable to fetch a new instruction. It is linked to the Cycle Count Register (0x0C) such that this register reflects the stall cycles measured over the same cycles as counted by the cycle count register. Thus the value of this register is always less than or equal to cycle count.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STALLCOUNT																															

Bits	Field Name	Description	Type	Reset
31:0	STALLCOUNT	This value is incremented by 1 for every cycle during which the PRU is enabled and the counter is enabled (both bits ENABLE and COUNTENABLE set in the PRU control register), and the PRU was unable to fetch a new instruction for any reason.	RW	0x0

Table 30-68. Register Call Summary for Register PRU_STALL

PRU-ICSS PRU Cores • PRUSS_PRU_CTRL Register Summary: [0] [1]

Table 30-69. PRU_CTBIR0

Address Offset	0x0000 0020		
Physical Address	0x4B22 2020 0x4B22 4020 0x4B2A 2020 0x4B2A 4020	**Instance**	PRUSS1_PRU0_CTRL PRUSS1_PRU1_CTRL PRUSS2_PRU0_CTRL PRUSS2_PRU1_CTRL
Description	CONSTANT TABLE BLOCK INDEX REGISTER 0. This register is used to set the block indices which are used to modify entries 24 and 25 in the PRU Constant Table. This register can be written by the PRU whenever it needs to change to a new base pointer for a block in the State Scratchpad RAM. This function is useful since the PRU is often processing multiple processing threads which require it to change contexts. The PRU can use this register to avoid requiring excessive amounts of code for repetitive context switching.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								C25_BLK_INDEX								RESERVED								C24_BLK_INDEX							

Bits	Field Name	Description	Type	Reset
31:24	RESERVED		R	0x00
23:16	C25_BLK_INDEX	PRU Constant Entry 25 Block Index: This field sets the value that will appear in bits 11:8 of entry 25 in the PRU Constant Table.	RW	0x0
15:8	RESERVED		R	0x00
7:0	C24_BLK_INDEX	PRU Constant Entry 24 Block Index: This field sets the value that will appear in bits 11:8 of entry 24 in the PRU Constant Table.	RW	0x0

Table 30-70. Register Call Summary for Register PRU_CTBIR0

PRU-ICSS PRU Cores • PRUSS_PRU_CTRL Register Summary: [0] [1]

Table 30-71. PRU_CTBIR1

Address Offset	0x0000 0024		
Physical Address	0x4B22 2024 0x4B22 4024 0x4B2A 2024 0x4B2A 4024	**Instance**	PRUSS1_PRU0_CTRL PRUSS1_PRU1_CTRL PRUSS2_PRU0_CTRL PRUSS2_PRU1_CTRL
Description	CONSTANT TABLE BLOCK INDEX REGISTER 1. This register is used to set the block indices which are used to modify entries 26 and 27 in the PRU Constant Table. This register can be written by the PRU whenever it needs to change to a new base pointer for a block in the State Scratchpad RAM. This function is useful since the PRU is often processing multiple processing threads which require it to change contexts. The PRU can use this register to avoid requiring excessive amounts of code for repetitive context switching.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								C27_BLK_INDEX								RESERVED								C26_BLK_INDEX							

Bits	Field Name	Description	Type	Reset
31:24	RESERVED		R	0x00
23:16	C27_BLK_INDEX	PRU Constant Entry 27 Block Index: This field sets the value that will appear in bits 11:8 of entry 27 in the PRU Constant Table.	RW	0x0
15:8	RESERVED		R	0x00
7:0	C26_BLK_INDEX	PRU Constant Entry 26 Block Index: This field sets the value that will appear in bits 11:8 of entry 26 in the PRU Constant Table.	RW	0x0

Table 30-72. Register Call Summary for Register PRU_CTBIR1

PRU-ICSS PRU Cores

- PRUSS_PRU_CTRL Register Summary: [0] [1]

Table 30-73. PRU_CTPPR0

Address Offset	0x0000 0028		
Physical Address	0x4B22 2028 0x4B22 4028 0x4B2A 2028 0x4B2A 4028	**Instance**	PRUSS1_PRU0_CTRL PRUSS1_PRU1_CTRL PRUSS2_PRU0_CTRL PRUSS2_PRU1_CTRL
Description	CONSTANT TABLE PROGRAMMABLE POINTER REGISTER 0. This register allows the PRU to set up the 256-byte page index for entries 28 and 29 in the PRU Constant Table which serve as general purpose pointers which can be configured to point to any locations inside the session router address map. This register is useful when the PRU needs to frequently access certain structures inside the session router address space whose locations are not hard coded such as tables in scratchpad memory.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C29_POINTER																C28_POINTER															

Bits	Field Name	Description	Type	Reset
31:16	C29_POINTER	PRU Constant Entry 29 Pointer: This field sets the value that will appear in bits 23:8 of entry 29 in the PRU Constant Table.	RW	0x0
15:0	C28_POINTER	PRU Constant Entry 28 Pointer: This field sets the value that will appear in bits 23:8 of entry 28 in the PRU Constant Table.	RW	0x0

Table 30-74. Register Call Summary for Register PRU_CTPPR0

PRU-ICSS PRU Cores

- PRUSS_PRU_CTRL Register Summary: [0] [1]

Table 30-75. PRU_CTPPR1

Address Offset	0x0000 002C		
Physical Address	0x4B22 202C 0x4B22 402C 0x4B2A 202C 0x4B2A 402C	**Instance**	PRUSS1_PRU0_CTRL PRUSS1_PRU1_CTRL PRUSS2_PRU0_CTRL PRUSS2_PRU1_CTRL
Description	CONSTANT TABLE PROGRAMMABLE POINTER REGISTER 1. This register functions the same as the PRU Constant Table Programmable Pointer Register 0 but allows the PRU to control entries 30 and 31 in the PRU Constant Table.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C31_POINTER																C30_POINTER															

Bits	Field Name	Description	Type	Reset
31:16	C31_POINTER	PRU Constant Entry 31 Pointer: This field sets the value that will appear in bits 23:8 of entry 31 in the PRU Constant Table.	RW	0x0
15:0	C30_POINTER	PRU Constant Entry 30 Pointer: This field sets the value that will appear in bits 23:8 of entry 30 in the PRU Constant Table.	RW	0x0

Table 30-76. Register Call Summary for Register PRU_CTPPR1

PRU-ICSS PRU Cores

- PRUSS_PRU_CTRL Register Summary: [0] [1]

30.1.5.7 PRUSS_PRU_DEBUG Register Manual

30.1.5.7.1 PRUSS_PRU_DEBUG Instance Summary

Table 30-77. PRUSS_PRU_DEBUG Instances Summary

Module Name	Base Address	Size
PRUSS1_PRU0_CTRL	0x20AA 2400	144 Bytes
PRUSS1_PRU1_CTRL	0x20AA 4400	144 Bytes
PRUSS2_PRU0_CTRL	0x20AE 2400	144 Bytes
PRUSS2_PRU1_CTRL	0x20AE 4400	144 Bytes

30.1.5.7.2 PRUSS_PRU_DEBUG Registers

30.1.5.7.2.1 PRUSS_PRU_DEBUG Register Summary

Table 30-78. PRUSS1_PRU_DEBUG Registers Mapping Summary

Acronym	Type	Register Width (Bits)	Address Offset	PRUSS1_PRU0_DEBUG Physical Address	PRUSS1_PRU1_DEBUG Physical Address
PRUSS_DBG_GPREG0	RW	32	0x0000 0000	0x20AA 2400	0x20AA 4400
PRUSS_DBG_GPREG1	RW	32	0x0000 0004	0x20AA 2404	0x20AA 4404
PRUSS_DBG_GPREG2	RW	32	0x0000 0008	0x20AA 2408	0x20AA 4408
PRUSS_DBG_GPREG3	RW	32	0x0000 000C	0x20AA 240C	0x20AA 440C
PRUSS_DBG_GPREG4	RW	32	0x0000 0010	0x20AA 2410	0x20AA 4410
PRUSS_DBG_GPREG5	RW	32	0x0000 0014	0x20AA 2414	0x20AA 4414
PRUSS_DBG_GPREG6	RW	32	0x0000 0018	0x20AA 2418	0x20AA 4418
PRUSS_DBG_GPREG7	RW	32	0x0000 001C	0x20AA 241C	0x20AA 441C
PRUSS_DBG_GPREG8	RW	32	0x0000 0020	0x20AA 2420	0x20AA 4420
PRUSS_DBG_GPREG9	RW	32	0x0000 0024	0x20AA 2424	0x20AA 4424
PRUSS_DBG_GPREG10	RW	32	0x0000 0028	0x20AA 2428	0x20AA 4428
PRUSS_DBG_GPREG11	RW	32	0x0000 002C	0x20AA 242C	0x20AA 442C
PRUSS_DBG_GPREG12	RW	32	0x0000 0030	0x20AA 2430	0x20AA 4430
PRUSS_DBG_GPREG13	RW	32	0x0000 0034	0x20AA 2434	0x20AA 4434
PRUSS_DBG_GPREG14	RW	32	0x0000 0038	0x20AA 2438	0x20AA 4438
PRUSS_DBG_GPREG15	RW	32	0x0000 003C	0x20AA 243C	0x20AA 443C

Table 30-78. PRUSS1_PRU_DEBUG Registers Mapping Summary (continued)

Acronym	Type	Register Width (Bits)	Address Offset	PRUSS1_PRU0_DEBUG Physical Address	PRUSS1_PRU1_DEBUG Physical Address
PRUSS_DBG_GPREG16	RW	32	0x0000 0040	0x20AA 2440	0x20AA 4440
PRUSS_DBG_GPREG17	RW	32	0x0000 0044	0x20AA 2444	0x20AA 4444
PRUSS_DBG_GPREG18	RW	32	0x0000 0048	0x20AA 2448	0x20AA 4448
PRUSS_DBG_GPREG19	RW	32	0x0000 004C	0x20AA 244C	0x20AA 444C
PRUSS_DBG_GPREG20	RW	32	0x0000 0050	0x20AA 2450	0x20AA 4450
PRUSS_DBG_GPREG21	RW	32	0x0000 0054	0x20AA 2454	0x20AA 4454
PRUSS_DBG_GPREG22	RW	32	0x0000 0058	0x20AA 2458	0x20AA 4458
PRUSS_DBG_GPREG23	RW	32	0x0000 005C	0x20AA 245C	0x20AA 445C
PRUSS_DBG_GPREG24	RW	32	0x0000 0060	0x20AA 2460	0x20AA 4460
PRUSS_DBG_GPREG25	RW	32	0x0000 0064	0x20AA 2464	0x20AA 4464
PRUSS_DBG_GPREG26	RW	32	0x0000 0068	0x20AA 2468	0x20AA 4468
PRUSS_DBG_GPREG27	RW	32	0x0000 006C	0x20AA 246C	0x20AA 446C
PRUSS_DBG_GPREG28	RW	32	0x0000 0070	0x20AA 2470	0x20AA 4470
PRUSS_DBG_GPREG29	RW	32	0x0000 0074	0x20AA 2474	0x20AA 4474
PRUSS_DBG_GPREG30	RW	32	0x0000 0078	0x20AA 2478	0x20AA 4478
PRUSS_DBG_GPREG31	RW	32	0x0000 007C	0x20AA 247C	0x20AA 447C
PRUSS_DBG_CT_REG0	R	32	0x0000 0080	0x20AA 2480	0x20AA 4480
PRUSS_DBG_CT_REG1	R	32	0x0000 0084	0x20AA 2484	0x20AA 4484
PRUSS_DBG_CT_REG2	R	32	0x0000 0088	0x20AA 2488	0x20AA 4488
PRUSS_DBG_CT_REG3	R	32	0x0000 008C	0x20AA 248C	0x20AA 448C
PRUSS_DBG_CT_REG4	R	32	0x0000 0090	0x20AA 2490	0x20AA 4490
PRUSS_DBG_CT_REG5	R	32	0x0000 0094	0x20AA 2494	0x20AA 4494
PRUSS_DBG_CT_REG6	R	32	0x0000 0098	0x20AA 2498	0x20AA 4498
PRUSS_DBG_CT_REG7	R	32	0x0000 009C	0x20AA 249C	0x20AA 449C
PRUSS_DBG_CT_REG8	R	32	0x0000 00A0	0x20AA 24A0	0x20AA 44A0
PRUSS_DBG_CT_REG9	R	32	0x0000 00A4	0x20AA 24A4	0x20AA 44A4
PRUSS_DBG_CT_REG10	R	32	0x0000 00A8	0x20AA 24A8	0x20AA 44A8
PRUSS_DBG_CT_REG11	R	32	0x0000 00AC	0x20AA 24AC	0x20AA 44AC
PRUSS_DBG_CT_REG12	R	32	0x0000 00B0	0x20AA 24B0	0x20AA 44B0
PRUSS_DBG_CT_REG13	R	32	0x0000 00B4	0x20AA 24B4	0x20AA 44B4
PRUSS_DBG_CT_REG14	R	32	0x0000 00B8	0x20AA 24B8	0x20AA 44B8
PRUSS_DBG_CT_REG15	R	32	0x0000 00BC	0x20AA 24BC	0x20AA 44BC
PRUSS_DBG_CT_REG16	R	32	0x0000 00C0	0x20AA 24C0	0x20AA 44C0
PRUSS_DBG_CT_REG17	R	32	0x0000 00C4	0x20AA 24C4	0x20AA 44C4
PRUSS_DBG_CT_REG18	R	32	0x0000 00C8	0x20AA 24C8	0x20AA 44C8
PRUSS_DBG_CT_REG19	R	32	0x0000 00CC	0x20AA 24CC	0x20AA 44CC
PRUSS_DBG_CT_REG20	R	32	0x0000 00D0	0x20AA 24D0	0x20AA 44D0
PRUSS_DBG_CT_REG21	R	32	0x0000 00D4	0x20AA 24D4	0x20AA 44D4
PRUSS_DBG_CT_REG22	R	32	0x0000 00D8	0x20AA 24D8	0x20AA 44D8
PRUSS_DBG_CT_REG23	R	32	0x0000 00DC	0x20AA 24DC	0x20AA 44DC
PRUSS_DBG_CT_REG24	R	32	0x0000 00E0	0x20AA 24E0	0x20AA 44E0
PRUSS_DBG_CT_REG25	R	32	0x0000 00E4	0x20AA 24E4	0x20AA 44E4
PRUSS_DBG_CT_REG26	R	32	0x0000 00E8	0x20AA 24E8	0x20AA 44E8
PRUSS_DBG_CT_REG27	R	32	0x0000 00EC	0x20AA 24EC	0x20AA 44EC
PRUSS_DBG_CT_REG28	R	32	0x0000 00F0	0x20AA 24F0	0x20AA 44F0
PRUSS_DBG_CT_REG29	R	32	0x0000 00F4	0x20AA 24F4	0x20AA 44F4

Table 30-78. PRUSS1_PRU_DEBUG Registers Mapping Summary (continued)

Acronym	Type	Register Width (Bits)	Address Offset	PRUSS1_PRU0_DEBUG Physical Address	PRUSS1_PRU1_DEBUG Physical Address
PRUSS_DBG_CT_REG30	R	32	0x0000 00F8	0x20AA 24F8	0x20AA 44F8
PRUSS_DBG_CT_REG31	R	32	0x0000 00FC	0x20AA 24FC	0x20AA 44FC

Table 30-79. PRUSS2_PRU_DEBUG Registers Mapping Summary

Acronym	Type	Register Width (Bits)	Address Offset	PRUSS2_PRU0_DEBUG Physical Address	PRUSS2_PRU1_DEBUG Physical Address
PRUSS_DBG_GPREG0	RW	32	0x0000 0000	0x20AE 2400	0x20AE 4400
PRUSS_DBG_GPREG1	RW	32	0x0000 0004	0x20AE 2404	0x20AE 4404
PRUSS_DBG_GPREG2	RW	32	0x0000 0008	0x20AE 2408	0x20AE 4408
PRUSS_DBG_GPREG3	RW	32	0x0000 000C	0x20AE 240C	0x20AE 440C
PRUSS_DBG_GPREG4	RW	32	0x0000 0010	0x20AE 2410	0x20AE 4410
PRUSS_DBG_GPREG5	RW	32	0x0000 0014	0x20AE 2414	0x20AE 4414
PRUSS_DBG_GPREG6	RW	32	0x0000 0018	0x20AE 2418	0x20AE 4418
PRUSS_DBG_GPREG7	RW	32	0x0000 001C	0x20AE 241C	0x20AE 441C
PRUSS_DBG_GPREG8	RW	32	0x0000 0020	0x20AE 2420	0x20AE 4420
PRUSS_DBG_GPREG9	RW	32	0x0000 0024	0x20AE 2424	0x20AE 4424
PRUSS_DBG_GPREG10	RW	32	0x0000 0028	0x20AE 2428	0x20AE 4428
PRUSS_DBG_GPREG11	RW	32	0x0000 002C	0x20AE 242C	0x20AE 442C
PRUSS_DBG_GPREG12	RW	32	0x0000 0030	0x20AE 2430	0x20AE 4430
PRUSS_DBG_GPREG13	RW	32	0x0000 0034	0x20AE 2434	0x20AE 4434
PRUSS_DBG_GPREG14	RW	32	0x0000 0038	0x20AE 2438	0x20AE 4438
PRUSS_DBG_GPREG15	RW	32	0x0000 003C	0x20AE 243C	0x20AE 443C
PRUSS_DBG_GPREG16	RW	32	0x0000 0040	0x20AE 2440	0x20AE 4440
PRUSS_DBG_GPREG17	RW	32	0x0000 0044	0x20AE 2444	0x20AE 4444
PRUSS_DBG_GPREG18	RW	32	0x0000 0048	0x20AE 2448	0x20AE 4448
PRUSS_DBG_GPREG19	RW	32	0x0000 004C	0x20AE 244C	0x20AE 444C
PRUSS_DBG_GPREG20	RW	32	0x0000 0050	0x20AE 2450	0x20AE 4450
PRUSS_DBG_GPREG21	RW	32	0x0000 0054	0x20AE 2454	0x20AE 4454
PRUSS_DBG_GPREG22	RW	32	0x0000 0058	0x20AE 2458	0x20AE 4458
PRUSS_DBG_GPREG23	RW	32	0x0000 005C	0x20AE 245C	0x20AE 445C
PRUSS_DBG_GPREG24	RW	32	0x0000 0060	0x20AE 2460	0x20AE 4460
PRUSS_DBG_GPREG25	RW	32	0x0000 0064	0x20AE 2464	0x20AE 4464
PRUSS_DBG_GPREG26	RW	32	0x0000 0068	0x20AE 2468	0x20AE 4468
PRUSS_DBG_GPREG27	RW	32	0x0000 006C	0x20AE 246C	0x20AE 446C
PRUSS_DBG_GPREG28	RW	32	0x0000 0070	0x20AE 2470	0x20AE 4470
PRUSS_DBG_GPREG29	RW	32	0x0000 0074	0x20AE 2474	0x20AE 4474
PRUSS_DBG_GPREG30	RW	32	0x0000 0078	0x20AE 2478	0x20AE 4478
PRUSS_DBG_GPREG31	RW	32	0x0000 007C	0x20AE 247C	0x20AE 447C
PRUSS_DBG_CT_REG0	R	32	0x0000 0080	0x20AE 2480	0x20AE 4480
PRUSS_DBG_CT_REG1	R	32	0x0000 0084	0x20AE 2484	0x20AE 4484
PRUSS_DBG_CT_REG2	R	32	0x0000 0088	0x20AE 2488	0x20AE 4488
PRUSS_DBG_CT_REG3	R	32	0x0000 008C	0x20AE 248C	0x20AE 448C
PRUSS_DBG_CT_REG4	R	32	0x0000 0090	0x20AE 2490	0x20AE 4490
PRUSS_DBG_CT_REG5	R	32	0x0000 0094	0x20AE 2494	0x20AE 4494
PRUSS_DBG_CT_REG6	R	32	0x0000 0098	0x20AE 2498	0x20AE 4498

Table 30-79. PRUSS2_PRU_DEBUG Registers Mapping Summary (continued)

Acronym	Type	Register Width (Bits)	Address Offset	PRUSS2_PRU0_DEBUG Physical Address	PRUSS2_PRU1_DEBUG Physical Address
PRUSS_DBG_CT_REG7	R	32	0x0000 009C	0x20AE 249C	0x20AE 449C
PRUSS_DBG_CT_REG8	R	32	0x0000 00A0	0x20AE 24A0	0x20AE 44A0
PRUSS_DBG_CT_REG9	R	32	0x0000 00A4	0x20AE 24A4	0x20AE 44A4
PRUSS_DBG_CT_REG10	R	32	0x0000 00A8	0x20AE 24A8	0x20AE 44A8
PRUSS_DBG_CT_REG11	R	32	0x0000 00AC	0x20AE 24AC	0x20AE 44AC
PRUSS_DBG_CT_REG12	R	32	0x0000 00B0	0x20AE 24B0	0x20AE 44B0
PRUSS_DBG_CT_REG13	R	32	0x0000 00B4	0x20AE 24B4	0x20AE 44B4
PRUSS_DBG_CT_REG14	R	32	0x0000 00B8	0x20AE 24B8	0x20AE 44B8
PRUSS_DBG_CT_REG15	R	32	0x0000 00BC	0x20AE 24BC	0x20AE 44BC
PRUSS_DBG_CT_REG16	R	32	0x0000 00C0	0x20AE 24C0	0x20AE 44C0
PRUSS_DBG_CT_REG17	R	32	0x0000 00C4	0x20AE 24C4	0x20AE 44C4
PRUSS_DBG_CT_REG18	R	32	0x0000 00C8	0x20AE 24C8	0x20AE 44C8
PRUSS_DBG_CT_REG19	R	32	0x0000 00CC	0x20AE 24CC	0x20AE 44CC
PRUSS_DBG_CT_REG20	R	32	0x0000 00D0	0x20AE 24D0	0x20AE 44D0
PRUSS_DBG_CT_REG21	R	32	0x0000 00D4	0x20AE 24D4	0x20AE 44D4
PRUSS_DBG_CT_REG22	R	32	0x0000 00D8	0x20AE 24D8	0x20AE 44D8
PRUSS_DBG_CT_REG23	R	32	0x0000 00DC	0x20AE 24DC	0x20AE 44DC
PRUSS_DBG_CT_REG24	R	32	0x0000 00E0	0x20AE 24E0	0x20AE 44E0
PRUSS_DBG_CT_REG25	R	32	0x0000 00E4	0x20AE 24E4	0x20AE 44E4
PRUSS_DBG_CT_REG26	R	32	0x0000 00E8	0x20AE 24E8	0x20AE 44E8
PRUSS_DBG_CT_REG27	R	32	0x0000 00EC	0x20AE 24EC	0x20AE 44EC
PRUSS_DBG_CT_REG28	R	32	0x0000 00F0	0x20AE 24F0	0x20AE 44F0
PRUSS_DBG_CT_REG29	R	32	0x0000 00F4	0x20AE 24F4	0x20AE 44F4
PRUSS_DBG_CT_REG30	R	32	0x0000 00F8	0x20AE 24F8	0x20AE 44F8
PRUSS_DBG_CT_REG31	R	32	0x0000 00FC	0x20AE 24FC	0x20AE 44FC

30.1.5.7.2.2 PRUSS_PRU_DEBUG Register Description

Table 30-80. PRUSS_DBG_GPREG0

Address Offset	0x0000 0000		
Physical Address	0x20AA 2400 0x20AA 4400 0x20AE 2400 0x20AE 2400	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 0. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG0																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG0	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-81. Register Call Summary for Register PRUSS_DBG_GPREG0

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-82. PRUSS_DBG_GPREG1

Address Offset	0x0000 0004		
Physical Address	0x20AA 2404 0x20AA 4404 0x20AE 2404 0x20AE 4404	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 1. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG1																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG1	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-83. Register Call Summary for Register PRUSS_DBG_GPREG1

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-84. PRUSS_DBG_GPREG2

Address Offset	0x0000 0008		
Physical Address	0x20AA 2408 0x20AA 4408 0x20AE 2408 0x20AE 4408	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 2. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG2																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG2	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-85. Register Call Summary for Register PRUSS_DBG_GPREG2

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-86. PRUSS_DBG_GPREG3

Address Offset	0x0000 000C		
Physical Address	0x20AA 240C 0x20AA 440C 0x20AE 240C 0x20AE 440C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 3. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG3																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG3	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-87. Register Call Summary for Register PRUSS_DBG_GPREG3

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-88. PRUSS_DBG_GPREG4

Address Offset	0x0000 0010		
Physical Address	0x20AA 2410 0x20AA 4410 0x20AE 2410 0x20AE 4410	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 4. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG4																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG4	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-89. Register Call Summary for Register PRUSS_DBG_GPREG4

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-90. PRUSS_DBG_GPREG5

Address Offset	0x0000 0014		
Physical Address	0x20AA 2414 0x20AA 4414 0x20AE 2414 0x20AE 4414	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 5. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG5																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG5	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-91. Register Call Summary for Register PRUSS_DBG_GPREG5

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-92. PRUSS_DBG_GPREG6

Address Offset	0x0000 0018		
Physical Address	0x20AA 2418 0x20AA 4418 0x20AE 2418 0x20AE 4418	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 6. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG6																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG6	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-93. Register Call Summary for Register PRUSS_DBG_GPREG6

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-94. PRUSS_DBG_GPREG7

Address Offset	0x0000 001C		
Physical Address	0x20AA 241C 0x20AA 441C 0x20AE 241C 0x20AE 441C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 7. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG7																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG7	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-95. Register Call Summary for Register PRUSS_DBG_GPREG7

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-96. PRUSS_DBG_GPREG8

Address Offset	0x0000 0020		
Physical Address	0x20AA 2420 0x20AA 4420 0x20AE 2420 0x20AE 4420	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 8. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG8																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG8	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-97. Register Call Summary for Register PRUSS_DBG_GPREG8

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-98. PRUSS_DBG_GPREG9

Address Offset	0x0000 0024		
Physical Address	0x20AA 2424 0x20AA 4424 0x20AE 2424 0x20AE 4424	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 9. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG9																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG9	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-99. Register Call Summary for Register PRUSS_DBG_GPREG9

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-100. PRUSS_DBG_GPREG10

Address Offset	0x0000 0028		
Physical Address	0x20AA 2428 0x20AA 4428 0x20AE 2428 0x20AE 4428	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 10. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG10																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG10	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-101. Register Call Summary for Register PRUSS_DBG_GPREG10

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-102. PRUSS_DBG_GPREG11

Address Offset	0x0000 002C		
Physical Address	0x20AA 242C 0x20AA 442C 0x20AE 242C 0x20AE 442C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 11. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG11																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG11	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-103. Register Call Summary for Register PRUSS_DBG_GPREG11

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-104. PRUSS_DBG_GPREG12

Address Offset	0x0000 0030		
Physical Address	0x20AA 2430 0x20AA 4430 0x20AE 2430 0x20AE 4430	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 12. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG12																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG12	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-105. Register Call Summary for Register PRUSS_DBG_GPREG12

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-106. PRUSS_DBG_GPREG13

Address Offset	0x0000 0034		
Physical Address	0x20AA 2434 0x20AA 4434 0x20AE 2434 0x20AE 4434	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 13. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG13																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG13	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-107. Register Call Summary for Register PRUSS_DBG_GPREG13

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-108. PRUSS_DBG_GPREG14

Address Offset	0x0000 0038		
Physical Address	0x20AA 2438 0x20AA 4438 0x20AE 2438 0x20AE 4438	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 14. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG14																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG14	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-109. Register Call Summary for Register PRUSS_DBG_GPREG14

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-110. PRUSS_DBG_GPREG15

Address Offset	0x0000 003C		
Physical Address	0x20AA 243C 0x20AA 443C 0x20AE 243C 0x20AE 443C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 15. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG15																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG15	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-111. Register Call Summary for Register PRUSS_DBG_GPREG15

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-112. PRUSS_DBG_GPREG16

Address Offset	0x0000 0040		
Physical Address	0x20AA 2440 0x20AA 4440 0x20AE 2440 0x20AE 4440	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 16. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG16																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG16	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-113. Register Call Summary for Register PRUSS_DBG_GPREG16

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-114. PRUSS_DBG_GPREG17

Address Offset	0x0000 0044		
Physical Address	0x20AA 2444 0x20AA 4444 0x20AE 2444 0x20AE 4444	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 17. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG17																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG17	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-115. Register Call Summary for Register PRUSS_DBG_GPREG17

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-116. PRUSS_DBG_GPREG18

Address Offset	0x0000 0048		
Physical Address	0x20AA 2448 0x20AA 4448 0x20AE 2448 0x20AE 4448	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 18. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG18																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG18	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-117. Register Call Summary for Register PRUSS_DBG_GPREG18

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-118. PRUSS_DBG_GPREG19

Address Offset	0x0000 004C		
Physical Address	0x20AA 244C 0x20AA 444C 0x20AE 244C 0x20AE 444C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 19. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG19																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG19	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-119. Register Call Summary for Register PRUSS_DBG_GPREG19

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-120. PRUSS_DBG_GPREG20

Address Offset	0x0000 0050		
Physical Address	0x20AA 2450 0x20AA 4450 0x20AE 2450 0x20AE 4450	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 20. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG20																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG20	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-121. Register Call Summary for Register PRUSS_DBG_GPREG20

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-122. PRUSS_DBG_GPREG21

Address Offset	0x0000 0054		
Physical Address	0x20AA 2454 0x20AA 4454 0x20AE 2454 0x20AE 4454	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 21. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG21																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG21	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-123. Register Call Summary for Register PRUSS_DBG_GPREG21

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-124. PRUSS_DBG_GPREG22

Address Offset	0x0000 0058		
Physical Address	0x20AA 2458 0x20AA 4458 0x20AE 2458 0x20AE 4458	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 22. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG22																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG22	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-125. Register Call Summary for Register PRUSS_DBG_GPREG22

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-126. PRUSS_DBG_GPREG23

Address Offset	0x0000 005C		
Physical Address	0x20AA 245C 0x20AA 445C 0x20AE 245C 0x20AE 445C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 23. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG23																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG23	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-127. Register Call Summary for Register PRUSS_DBG_GPREG23

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-128. PRUSS_DBG_GPREG24

Address Offset	0x0000 0060		
Physical Address	0x20AA 2460 0x20AA 4460 0x20AE 2460 0x20AE 4460	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 24. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG24																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG24	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-129. Register Call Summary for Register PRUSS_DBG_GPREG24

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-130. PRUSS_DBG_GPREG25

Address Offset	0x0000 0064		
Physical Address	0x20AA 2464 0x20AA 4464 0x20AE 2464 0x20AE 4464	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 25. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG25																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG25	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-131. Register Call Summary for Register PRUSS_DBG_GPREG25

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-132. PRUSS_DBG_GPREG26

Address Offset	0x0000 0068		
Physical Address	0x20AA 2468 0x20AA 4468 0x20AE 2468 0x20AE 4468	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 26. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG26																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG26	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-133. Register Call Summary for Register PRUSS_DBG_GPREG26

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-134. PRUSS_DBG_GPREG27

Address Offset	0x0000 006C		
Physical Address	0x20AA 246C 0x20AA 446C 0x20AE 246C 0x20AE 446C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 27. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG27																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG27	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-135. Register Call Summary for Register PRUSS_DBG_GPREG27

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-136. PRUSS_DBG_GPREG28

Address Offset	0x0000 0070		
Physical Address	0x20AA 2470 0x20AA 4470 0x20AE 2470 0x20AE 4470	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 28. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG28																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG28	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-137. Register Call Summary for Register PRUSS_DBG_GPREG28

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-138. PRUSS_DBG_GPREG29

Address Offset	0x0000 0074		
Physical Address	0x20AA 2474 0x20AA 4474 0x20AE 2474 0x20AE 4474	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 29. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG29																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG29	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-139. Register Call Summary for Register PRUSS_DBG_GPREG29

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-140. PRUSS_DBG_GPREG30

Address Offset	0x0000 0078		
Physical Address	0x20AA 2478 0x20AA 4478 0x20AE 2478 0x20AE 4478	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 30. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG30																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG30	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-141. Register Call Summary for Register PRUSS_DBG_GPREG30

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-142. PRUSS_DBG_GPREG31

Address Offset	0x0000 007C		
Physical Address	0x20AA 247C 0x20AA 447C 0x20AE 247C 0x20AE 447C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 31. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG31																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG31	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-143. Register Call Summary for Register PRUSS_DBG_GPREG31

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-144. PRUSS_DBG_CT_REG0

Address Offset	0x0000 0080		
Physical Address	0x20AA 2480 0x20AA 4480 0x20AE 2480 0x20AE 4480	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 0. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG0																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG0	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x02 0000

Table 30-145. Register Call Summary for Register PRUSS_DBG_CT_REG0

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-146. PRUSS_DBG_CT_REG1

Address Offset	0x0000 0084		
Physical Address	0x20AA 2484 0x20AA 4484 0x20AE 2484 0x20AE 4484	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 1. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG1																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG1	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4804 0000

Table 30-147. Register Call Summary for Register PRUSS_DBG_CT_REG1

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-148. PRUSS_DBG_CT_REG2

Address Offset	0x0000 0088		
Physical Address	0x20AA 2488 0x20AA 4488 0x20AE 2488 0x20AE 4488	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 2. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG2																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG2	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4802 A000

Table 30-149. Register Call Summary for Register PRUSS_DBG_CT_REG2

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-150. PRUSS_DBG_CT_REG3

Address Offset	0x0000 008C		
Physical Address	0x20AA 248C 0x20AA 448C 0x20AE 248C 0x20AE 448C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 3. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG3																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG3	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x03 0000

Table 30-151. Register Call Summary for Register PRUSS_DBG_CT_REG3

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-152. PRUSS_DBG_CT_REG4

Address Offset	0x0000 0090		
Physical Address	0x20AA 2490 0x20AA 4490 0x20AE 2490 0x20AE 4490	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 4. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG4																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG4	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x02 6000

Table 30-153. Register Call Summary for Register PRUSS_DBG_CT_REG4

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-154. PRUSS_DBG_CT_REG5

Address Offset	0x0000 0094		
Physical Address	0x20AA 2494 0x20AA 4494 0x20AE 2494 0x20AE 4494	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 5. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG5																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG5	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4806 0000

Table 30-155. Register Call Summary for Register PRUSS_DBG_CT_REG5

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-156. PRUSS_DBG_CT_REG6

Address Offset	0x0000 0098		
Physical Address	0x20AA 2498 0x20AA 4498 0x20AE 2498 0x20AE 4498	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 6. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG6																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG6	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4803 0000

Table 30-157. Register Call Summary for Register PRUSS_DBG_CT_REG6

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-158. PRUSS_DBG_CT_REG7

Address Offset	0x0000 009C		
Physical Address	0x20AA 249C 0x20AA 449C 0x20AE 249C 0x20AE 449C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 7. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG7																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG7	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x02 8000

Table 30-159. Register Call Summary for Register PRUSS_DBG_CT_REG7

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-160. PRUSS_DBG_CT_REG8

Address Offset	0x0000 00A0		
Physical Address	0x20AA 24A0 0x20AA 44A0 0x20AE 24A0 0x20AE 44A0	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 8. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG8																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG8	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4600 0000

Table 30-161. Register Call Summary for Register PRUSS_DBG_CT_REG8

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-162. PRUSS_DBG_CT_REG9

Address Offset	0x0000 00A4		
Physical Address	0x20AA 24A4 0x20AA 44A4 0x20AE 24A4 0x20AE 44A4	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 9. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG9																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG9	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4A10 0000

Table 30-163. Register Call Summary for Register PRUSS_DBG_CT_REG9

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-164. PRUSS_DBG_CT_REG10

Address Offset	0x0000 00A8		
Physical Address	0x20AA 24A8 0x20AA 44A8 0x20AE 24A8 0x20AE 44A8	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 10. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG10																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG10	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4831 8000

Table 30-165. Register Call Summary for Register PRUSS_DBG_CT_REG10

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-166. PRUSS_DBG_CT_REG11

Address Offset	0x0000 00AC		
Physical Address	0x20AA 24AC 0x20AA 44AC 0x20AE 24AC 0x20AE 44AC	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 11. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG11																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG11	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4802 2000

Table 30-167. Register Call Summary for Register PRUSS_DBG_CT_REG11

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-168. PRUSS_DBG_CT_REG12

Address Offset	0x0000 00B0		
Physical Address	0x20AA 24B0 0x20AA 44B0 0x20AE 24B0 0x20AE 44B0	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 12. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG12																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG12	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4802 4000

Table 30-169. Register Call Summary for Register PRUSS_DBG_CT_REG12

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-170. PRUSS_DBG_CT_REG13

Address Offset	0x0000 00B4		
Physical Address	0x20AA 24B4 0x20AA 44B4 0x20AE 24B4 0x20AE 44B4	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 13. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG13																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG13	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4831 0000

Table 30-171. Register Call Summary for Register PRUSS_DBG_CT_REG13

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-172. PRUSS_DBG_CT_REG14

Address Offset	0x0000 00B8		
Physical Address	0x20AA 24B8 0x20AA 44B8 0x20AE 24B8 0x20AE 44B8	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 14. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG14																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG14	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x481C C000

Table 30-173. Register Call Summary for Register PRUSS_DBG_CT_REG14

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-174. PRUSS_DBG_CT_REG15

Address Offset	0x0000 00BC		
Physical Address	0x20AA 24BC 0x20AA 44BC 0x20AE 24BC 0x20AE 44BC	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 15. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG15																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG15	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x481D 0000

Table 30-175. Register Call Summary for Register PRUSS_DBG_CT_REG15

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-176. PRUSS_DBG_CT_REG16

Address Offset	0x0000 00C0		
Physical Address	0x20AA 24C0 0x20AA 44C0 0x20AE 24C0 0x20AE 44C0	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 16. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG16																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG16	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x481A 0000

Table 30-177. Register Call Summary for Register PRUSS_DBG_CT_REG16

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-178. PRUSS_DBG_CT_REG17

Address Offset	0x0000 00C4		
Physical Address	0x20AA 24C4 0x20AA 44C4 0x20AE 24C4 0x20AE 44C4	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 17. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG17																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG17	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4819 C000

Table 30-179. Register Call Summary for Register PRUSS_DBG_CT_REG17

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-180. PRUSS_DBG_CT_REG18

Address Offset	0x0000 00C8		
Physical Address	0x20AA 24C8 0x20AA 44C8 0x20AE 24C8 0x20AE 44C8	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 18. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG18																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG18	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4830 0000

Table 30-181. Register Call Summary for Register PRUSS_DBG_CT_REG18

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-182. PRUSS_DBG_CT_REG19

Address Offset	0x0000 00CC		
Physical Address	0x20AA 24CC 0x20AA 44CC 0x20AE 24CC 0x20AE 44CC	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 19. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG19																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG19	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4830 2000

Table 30-183. Register Call Summary for Register PRUSS_DBG_CT_REG19

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-184. PRUSS_DBG_CT_REG20

Address Offset	0x0000 00D0		
Physical Address	0x20AA 24D0 0x20AA 44D0 0x20AE 24D0 0x20AE 44D0	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 20. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG20																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG20	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4830 4000

Table 30-185. Register Call Summary for Register PRUSS_DBG_CT_REG20

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-186. PRUSS_DBG_CT_REG21

Address Offset	0x0000 00D4		
Physical Address	0x20AA 24D4 0x20AA 44D4 0x20AE 24D4 0x20AE 44D4	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 21. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG21																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG21	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x03 2400

Table 30-187. Register Call Summary for Register PRUSS_DBG_CT_REG21

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-188. PRUSS_DBG_CT_REG22

Address Offset	0x0000 00D8		
Physical Address	0x20AA 24D8 0x20AA 44D8 0x20AE 24D8 0x20AE 44D8	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 22. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG22																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG22	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x480C 8000

Table 30-189. Register Call Summary for Register PRUSS_DBG_CT_REG22

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-190. PRUSS_DBG_CT_REG23

Address Offset	0x0000 00DC		
Physical Address	0x20AA 24DC 0x20AA 44DC 0x20AE 24DC 0x20AE 44DC	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 23. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG23																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG23	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x480C A000

Table 30-191. Register Call Summary for Register PRUSS_DBG_CT_REG23

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-192. PRUSS_DBG_CT_REG24

Address Offset	0x0000 00E0		
Physical Address	0x20AA 24E0 0x20AA 44E0 0x20AE 24E0 0x20AE 44E0	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 24. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG24																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG24	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table. This entry is partially programmable through the c24_blk_index in the PRU Control register. The reset value for this Constant Table Entry is 0x00000n00, n=c24_blk_index[3:0].	R	0x0

Table 30-193. Register Call Summary for Register PRUSS_DBG_CT_REG24

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-194. PRUSS_DBG_CT_REG25

Address Offset	0x0000 00E4		
Physical Address	0x20AA 24E4 0x20AA 44E4 0x20AE 24E4 0x20AE 44E4	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 25. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG25																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG25	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table. This entry is partially programmable through the c25_blk_index in the PRU Control register. The reset value for this Constant Table Entry is 0x00002n00, n=c25_blk_index[3:0].	R	0x0

Table 30-195. Register Call Summary for Register PRUSS_DBG_CT_REG25

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-196. PRUSS_DBG_CT_REG26

Address Offset	0x0000 00E8		
Physical Address	0x20AA 24E8 0x20AA 44E8 0x20AE 24E8 0x20AE 44E8	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 26. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG26																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG26	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table. This entry is partially programmable through the c26_blk_index in the PRU Control register. The reset value for this Constant Table Entry is 0x0002En00, n=c26_blk_index[3:0].	R	0x0

Table 30-197. Register Call Summary for Register PRUSS_DBG_CT_REG26

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-198. PRUSS_DBG_CT_REG27

Address Offset	0x0000 00EC		
Physical Address	0x20AA 24EC 0x20AA 44EC 0x20AE 24EC 0x20AE 44EC	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 27. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG27																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG27	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table. This entry is partially programmable through the c27_blk_index in the PRU Control register. The reset value for this Constant Table Entry is 0x00032n00, n=c27_blk_index[3:0].	R	0x0

Table 30-199. Register Call Summary for Register PRUSS_DBG_CT_REG27

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-200. PRUSS_DBG_CT_REG28

Address Offset	0x0000 00F0		
Physical Address	0x20AA 24F0 0x20AA 44F0 0x20AE 24F0 0x20AE 44F0	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 28. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG28																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG28	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table. This entry is partially programmable through the c28_pointer in the PRU Control register. The reset value for this Constant Table Entry is 0x00nnnn00, nnnn=c28_pointer[15:0].	R	0x0

Table 30-201. Register Call Summary for Register PRUSS_DBG_CT_REG28

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-202. PRUSS_DBG_CT_REG29

Address Offset	0x0000 00F4		
Physical Address	0x20AA 24F4 0x20AA 44F4 0x20AE 24F4 0x20AE 44F4	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 29. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG29																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG29	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table. This entry is partially programmable through the c29_pointer in the PRU Control register. The reset value for this Constant Table Entry is 0x49nnnn00, nnnn=c29_pointer[15:0].	R	0x0

Table 30-203. Register Call Summary for Register PRUSS_DBG_CT_REG29

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-204. PRUSS_DBG_CT_REG30

Address Offset	0x0000 00F8		
Physical Address	0x20AA 24F8 0x20AA 44F8 0x20AE 24F8 0x20AE 44F8	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 30. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG30																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG30	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table. This entry is partially programmable through the c30_pointer in the PRU Control register. The reset value for this Constant Table Entry is 0x40nnnn00, nnnn=c30_pointer[15:0].	R	0x0

Table 30-205. Register Call Summary for Register PRUSS_DBG_CT_REG30

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-206. PRUSS_DBG_CT_REG31

Address Offset	0x0000 00FC		
Physical Address	0x20AA 24FC 0x20AA 44FC 0x20AE 24FC 0x20AE 44FC	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 31. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG31																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG31	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table. This entry is partially programmable through the c31_pointer in the PRU Control register. The reset value for this Constant Table Entry is 0x80nnnn00, nnnn=c31_pointer[15:0].	R	0x0

Table 30-207. Register Call Summary for Register PRUSS_DBG_CT_REG31

PRUSS_PRU_DEBUG Register Manual

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

30.1.6 PRU-ICSS Local Interrupt Controller

This section describes functionality of the PRU-ICSS integrated Interrupt Controller - PRUSS_INTC.

30.1.6.1 PRU-ICSS Interrupt Controller Overview

The PRU-ICSS interrupt controller (PRUSS_INTC) maps interrupts coming from different parts of the device (mapped to PRU-ICSS1/PRU-ICSS2 via the device IRQ_CROSSBAR) to a reduced set of PRU-ICSS interrupt channels.

The PRUSS_INTC has the following features:

- Capturing up to 64 System Events (inputs)
- Supports up to 10 output interrupt channels.
- Generation of 10 Host Interrupts
  - 2 Host Interrupts for the PRUs.
  - 8 Host Interrupts exported from the PRU-ICSS for signaling the ARM interrupt controllers.
- Each system event can be enabled and disabled.
- Each host event can be enabled and disabled.
- Hardware prioritization of events.

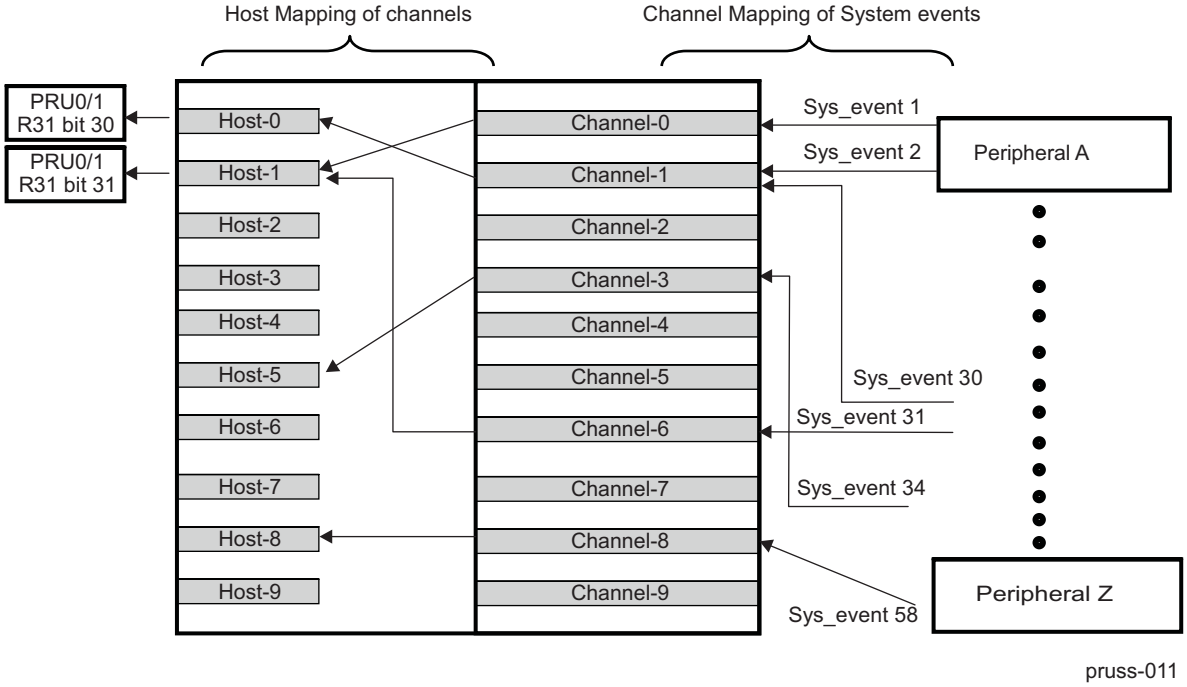
30.1.6.2 PRU-ICSS Interrupt Controller Functional Description

The PRU-ICSS incorporates an interrupt controller - PRUSS_INTC that supports up to 64 system interrupts from different peripherals (including 32 interrupts from PRU-ICSS located interrupt sources). The PRUSS_INTC maps these system events to 10 channels inside the PRUSS_INTC (see Figure 30-18). Interrupts from these 10 channels are further mapped to 10 Host Interrupts.

- Any of the 64 system interrupts can be mapped to any of the 10 channels.
- Multiple interrupts can be mapped to a single channel.
- An interrupt should not be mapped to more than one channel.
- Any of the 10 channels can be mapped to any of the 10 host interrupts. It is recommended to map channel “x” to host interrupt “x”, where x is from 0 to 9
- A channel should not be mapped to more than one host interrupt
- For channels mapping to the same host interrupt, lower number channels have higher priority.
- For interrupts on same channel, priority is determined by the hardware interrupt number. The lower the interrupt number, the higher the priority.
- Host Interrupt 0 is connected to bit 30 in register 31 (R31) of PRU0 and PRU1.
- Host Interrupt 1 is connected to bit 31 in register 31 (R31) for PRU0 and PRU1.
- Host Interrupts 2 through 9 exported from PRU-ICSS and mapped to interrupt controllers in the device.

NOTE: The Host interrupt 8 and host interrupt 9 are also exported as DMA requests to the device instantiated DMA_CROSSBAR which in turn can remap them to each line of the device integrated SDMA, EDMA, DSP1_EDMA and DSP2_EDMA controllers. For more details on PRU-ICSS DMA request outputs mapping, refer to the Section 30.1.3.

Figure 30-18. PRU-ICSS Interrupt Controller Block Diagram



30.1.6.2.1 PRU-ICSS Interrupt Controller System Events

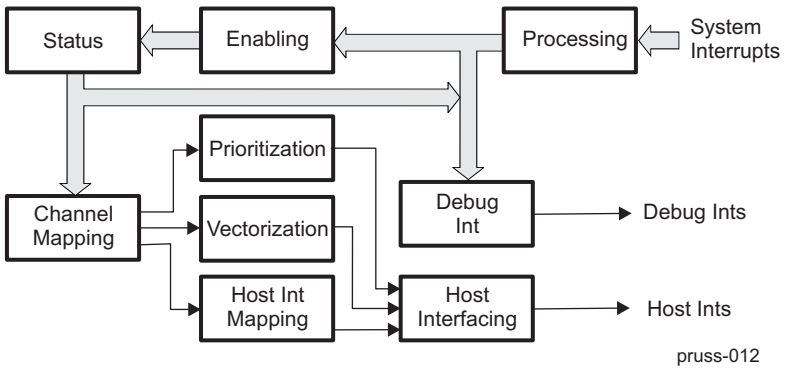
The PRU-ICSS system events - interrupt inputs. The device includes a internal mux that selects the Standard (default) or MII_RT mode system events. The mux control signal is controlled by PRUSS_MII_RT[0] MII_RT_EVENT_EN, which can be modified by software in PRU-ICSS CFG register space.

30.1.6.2.2 PRU-ICSS Interrupt Controller System Events Flow

The PRUSS_INTC module controls the system event mapping to the host interrupt interface. System events are generated by the device peripherals or PRUs. The PRUSS_INTC receives the system interrupts and maps them to internal channels. The channels are used to group interrupts together and to prioritize them. These channels are then mapped onto the host interrupts. Interrupts from the system side are active high in polarity. They are also pulse type of interrupts.

The PRUSS_INTC encompasses many functions to process the system interrupts and prepare them for the host interface. These functions are: processing, enabling, status, channel mapping, host interrupt mapping, prioritization, and host interfacing. Figure 30-19 illustrates the flow of system interrupts through the functions to the host. The following subsections describe each part of the flow.

Figure 30-19. Flow of System Interrupts to Host



30.1.6.2.2.1 PRU-ICSS Interrupt Processing

This block does following tasks:

- Synchronization of slower and asynchronous interrupts
- Conversion of polarity to active high
- Conversion of interrupt type to pulse interrupts

After the processing block, all interrupts will be active high pulses.

30.1.6.2.2.1.1 PRU-ICSS Interrupt Enabling

The next stage of PRUSS_INTC is to enable system interrupts based on programmed settings. The following sequence is to be followed to enable interrupts:

- Enable required system interrupts: System interrupts that are required to get propagated to host are to be enabled individually by writing to INDEX field in the system interrupt enable indexed set register (PRUSS_INTC_EISR). The interrupt to enable is the index value written. This sets the Enable Register bit of the given index.
- Enable required host interrupts: By writing 1 to the appropriate bit of the INDEX field in the host interrupt enable indexed set register (PRUSS_INTC_HIEISR), enable the required host interrupts. The host interrupt to enable is the index value written. This enables the host interrupt output or triggers the output again if that host interrupt is already enabled.
- Enable all host interrupts: By setting the ENABLE bit in the global enable register (PRUSS_INTC_GER) to 1, all host interrupts will be enabled. Individual host interrupts are still enabled or disabled from their individual enables and are not overridden by the global enable.

30.1.6.2.2.2 PRU-ICSS Interrupt Status Checking

The next stage is to capture which system interrupts are pending. There are two kinds of pending status: raw status and enabled status. Raw status is the pending status of the system interrupt without regards to the enable bit for the system interrupt. Enabled status is the pending status of the system interrupts with the enable bits active. When the enable bit is inactive, the enabled status will always be inactive. The enabled status of system interrupts is captured in system interrupt status enabled/clear registers (PRUSS_INTC_SECR1 and PRUSS_INTC_SECR0).

Status of system interrupt 'N' is indicated by the N-th bit of PRUSS_INTC_SECR1 and PRUSS_INTC_SECR0. Since there are 64 system interrupts, two 32-bit registers are used to capture the enabled status of interrupts. The pending status reflects whether the system interrupt occurred since the last time the status register bit was cleared. Each bit in the status register can be individually cleared.

30.1.6.2.2.3 PRU-ICSS Interrupt Channel Mapping

The PRUSS_INTC has 10 internal channels to which enabled system interrupts can be mapped. Channel 0 has highest priority and channel 9 has the lowest priority. Channels are used to group the system interrupts into a smaller number of priorities that can be given to a host interface with a very small number of interrupt inputs.

When multiple system interrupts are mapped to the same channel their interrupts are ORed together so that when either is active the output is active. The channel map registers (PRUSS_INTC_CMRi, where i=0 to 15) define the channel for each system interrupt. There is one register per 4 system interrupts; therefore, there are 16 channel map registers for a system of 64 interrupts. The channel for each system interrupt can be set using these registers.

30.1.6.2.2.3.1 PRU-ICSS Host Interrupt Mapping

The hosts can be the local PRU processors (PRU0 and PRU1) as well as device processors located outside PRU-ICSS such as MPU Cortex-A15, DSP1, IPU1, EVEs, etc. The 10 channels from the PRUSS_INTC can be mapped to any of the 10 Host interrupts. The Host map registers (PRUSS_INTC_HMR0 - PRUSS_INTC_HMR2) define the channel for each system interrupt. There is one register per 4 channels; therefore, there are 3 host map registers for 10 channels. When multiple channels are mapped to the same host interrupt, then prioritization is done to select which interrupt is in the highest-priority channel and which should be sent first to the host.

30.1.6.2.2.3.2 PRU-ICSS Interrupt Prioritization

The next stage of the PRUSS_INTC is prioritization. Since multiple interrupts can feed into a single channel and multiple channels can feed into a single host interrupt, it is to read the status of all system interrupts to determine the highest priority interrupt that is pending. The PRUSS_INTC provides hardware to perform this prioritization with a given scheme so that software does not have to do this. There are two levels of prioritizations:

- The first level of prioritization is between the active channels for a host interrupt. Channel 0 has the highest priority and channel 9 has the lowest. So the first level of prioritization picks the lowest numbered active channel.
- The second level of prioritization is between the active system interrupts for the prioritized channel. The system interrupt in position 0 has the highest priority and system interrupt 63 has the lowest priority. So the second level of prioritization picks the lowest position active system interrupt.

This is the final prioritized system interrupt for the host interrupt and is stored in the global prioritized index register (PRUSS_INTC_GPIR). The highest priority pending interrupt with respect to each host interrupts can be obtained using the host interrupt prioritized index registers (PRUSS_INTC_HIPIRj where j=0 to 9).

30.1.6.2.2.4 PRU-ICSS Interrupt Nesting

The PRUSS_INTC can also perform a nesting function in its prioritization. Nesting is a method of disabling certain interrupts (usually lower-priority interrupts) when an interrupt is taken so that only those desired interrupts can trigger to the host while it is servicing the current interrupt. The typical usage is to nest on the current interrupt and disable all interrupts of the same or lower priority (or channel). Then the host will only be interrupted from a higher priority interrupt.

The nesting is done in one of three methods:

1. Nesting for all host interrupts, based on channel priority: When an interrupt is taken, the nesting level is set to its channel priority. From then, that channel priority and all lower priority channels will be disabled from generating host interrupts and only higher priority channels are allowed. When the interrupt is completely serviced, the nesting level is returned to its original value. When there is no interrupt being serviced, there are no channels disabled due to nesting. The global nesting level register (PRUSS_INTC_GNLR) allows the checking and setting of the global nesting level across all host interrupts. The nesting level is the channel (and all of lower priority channels) that are nested out because of a current interrupt.
2. Nesting for individual host interrupts, based on channel priority: Always nest based on channel priority for each host interrupt individually. When an interrupt is taken on a host interrupt, then, the nesting level is set to its channel priority for just that host interrupt, and other host interrupts do not have their nesting affected. Then for that host interrupt, equal or lower priority channels will not interrupt the host but may on other host interrupts if programmed. When the interrupt is completely serviced the nesting level for the host interrupt is returned to its original value. The host interrupt nesting level registers (PRUSS_INTC_HINLRj where j=0 to 9) display and control the nesting level for each host interrupt. The nesting level controls which channel and lower priority channels are nested. There is one register per host interrupt.
3. Software manually performs the nesting of interrupts. When an interrupt is taken, the software will disable all the host interrupts, manually update the enables for any or all the system interrupts, and then re-enables all the host interrupts. This now allows only the system interrupts that are still enabled to trigger to the host. When the interrupt is completely serviced the software must reverse the changes to re-enable the nested out system interrupts. This method requires the most software interaction but gives the most flexibility if simple channel based nesting mechanisms are not adequate.

30.1.6.2.2.5 PRU-ICSS Interrupt Status Clearing

After servicing the interrupt (after execution of the ISR), interrupt status is to be cleared. If a system interrupt status is not cleared, then another host interrupt may not be triggered or another host interrupt may be triggered incorrectly. It is also essential to clear all system interrupts before the PRU is halted as the PRU does not power down unless all the interrupt status are cleared. For clearing the status of an interrupt, whose interrupt number is N, write a 1 to the Nth bit position in the system interrupt status enabled/clear registers (PRUSS_INTC_SECR0 and PRUSS_INTC_SECR1). System interrupt N can also be cleared by writing the value N into the system interrupt status indexed clear register (PRUSS_INTC_SICR).

30.1.6.2.3 PRU-ICSS Interrupt Disabling

At any time, if any interrupt is not to be propagated to the host, then that interrupt should be disabled. For disabling an interrupt whose interrupt number is N, write a 1 to the Nth bit in the system interrupt enable clear registers (PRUSS_INTC_ECR0 and PRUSS_INTC_ECR1). System interrupt N can also be disabled by writing the value N in the system interrupt enable indexed clear register (PRUSS_INTC_EICR).

30.1.6.3 PRU-ICSS Interrupt Controller Basic Programming Model

Follow these steps to configure the interrupt controller.

1. Set polarity and type of system event through the System Interrupt Polarity Registers (PRUSS_INTC_SIPR1 and PRUSS_INTC_SIPR0) and the System Interrupt Type Registers (PRUSS_INTC_SITR1 and PRUSS_INTC_SITR0). Polarity of all system interrupts is always high. Type of all system interrupts is always pulse.
2. Map system event to PRUSS_INTC channel through PRUSS_INTC_CMRi (i=0 to 15) channel mapping registers.
3. Map channel to host interrupt through PRUSS_INTC_HMR0/1/2 registers. Recommended channel “x” to be mapped to host interrupt “x”.
4. Clear system interrupt by writing 1 to PRUSS_INTC_SECR0/1 registers.
5. Enable host interrupt by writing index value to PRUSS_INTC_HIEISR register.
6. Enable interrupt nesting if desired.
7. Globally enable all interrupts through register PRUSS_INTC_GER[0] ENABLE_HINT_ANY bit.

30.1.6.4 PRU-ICSS Interrupt Requests Mapping

The PRU-ICSS1_INTC/PRUSS2_INTC lines 0 through 31 are mapped to events which are generated by PRU-ICSS integrated modules. Table 30-208 shows mapping of the different PRU-ICSS internally sourced IRQ events to PRUSS1_INTC/PRUSS2_INTC interrupt lines 0 through 31.

Table 30-208. PRU-ICSS1/PRU-ICSS2 Internal Interrupts

PRU-ICSS INTC IRQ input	PRU-ICSS Internal Interrupt Signal Name	Source
PRUSS1_INTC		
PRUSS1_IRQ_31	pr1_pru_mst_intr15_intr_req	pru0 or pru1
PRUSS1_IRQ_30	pr1_pru_mst_intr14_intr_req	pru0 or pru1
PRUSS1_IRQ_29	pr1_pru_mst_intr[13]_intr_req	pru0 or pru1
PRUSS1_IRQ_28	pr1_pru_mst_intr[12]_intr_req	pru0 or pru1
PRUSS1_IRQ_27	pr1_pru_mst_intr[11]_intr_req	pru0 or pru1
PRUSS1_IRQ_26	pr1_pru_mst_intr[10]_intr_req	pru0 or pru1
PRUSS1_IRQ_25	pr1_pru_mst_intr[9]_intr_req	pru0 or pru1
PRUSS1_IRQ_24	pr1_pru_mst_intr[8]_intr_req	pru0 or pru1
PRUSS1_IRQ_23	pr1_pru_mst_intr[7]_intr_req	pru0 or pru1
PRUSS1_IRQ_22	pr1_pru_mst_intr[6]_intr_req	pru0 or pru1
PRUSS1_IRQ_21	pr1_pru_mst_intr[5]_intr_req	pru0 or pru1
PRUSS1_IRQ_20	pr1_pru_mst_intr[4]_intr_req	pru0 or pru1
PRUSS1_IRQ_19	pr1_pru_mst_intr[3]_intr_req	pru0 or pru1
PRUSS1_IRQ_18	pr1_pru_mst_intr[2]_intr_req	pru0 or pru1
PRUSS1_IRQ_17	pr1_pru_mst_intr[1]_intr_req	pru0 or pru1
PRUSS1_IRQ_16	pr1_pru_mst_intr[0]_intr_req	pru0 or pru1
PRUSS1_IRQ_15	pr1_ecap_intr_req	PRUSS eCAP
PRUSS1_IRQ_8	digio_event_req	PRUSS IEP (Ethercat)
PRUSS1_IRQ_7	pr1_iep_tim_cap_cmp_pend	PRUSS IEP
PRUSS1_IRQ_6	pr1_uart_uint_intr_req	PRUSS UART
PRUSS1_IRQ_5	pr1_uart_utxevt_intr_req	PRUSS UART
PRUSS1_IRQ_4	pr1_uart_urxevt_intr_req	PRUSS UART
PRUSS1_IRQ_3	pr1_xfr_timeout	PRUSS Scratch Pad
PRUSS1_IRQ_2	pr1_pru1_r31_status_cnt16	PRU1 (Shift Capture)
PRUSS1_IRQ_1	pr1_pru0_r31_status_cnt16	PRU0 (Shift Capture)
PRUSS1_IRQ_0	pr1_parity_err_intr_pend	PRUSS Parity Logic
PRUSS2_INTC		
PRUSS2_IRQ_31	pr2_pru_mst_intr[15]_intr_req	pru0 or pru1
PRUSS2_IRQ_30	pr2_pru_mst_intr[14]_intr_req	pru0 or pru1
PRUSS2_IRQ_29	pr2_pru_mst_intr[13]_intr_req	pru0 or pru1
PRUSS2_IRQ_28	pr2_pru_mst_intr[12]_intr_req	pru0 or pru1
PRUSS2_IRQ_27	pr2_pru_mst_intr[11]_intr_req	pru0 or pru1
PRUSS2_IRQ_26	pr2_pru_mst_intr[10]_intr_req	pru0 or pru1
PRUSS2_IRQ_25	pr2_pru_mst_intr[9]_intr_req	pru0 or pru1
PRUSS2_IRQ_24	pr2_pru_mst_intr[8]_intr_req	pru0 or pru1
PRUSS2_IRQ_23	pr2_pru_mst_intr[7]_intr_req	pru0 or pru1
PRUSS2_IRQ_22	pr2_pru_mst_intr[6]_intr_req	pru0 or pru1
PRUSS2_IRQ_21	pr2_pru_mst_intr[5]_intr_req	pru0 or pru1
PRUSS2_IRQ_20	pr2_pru_mst_intr[4]_intr_req	pru0 or pru1
PRUSS2_IRQ_19	pr2_pru_mst_intr[3]_intr_req	pru0 or pru1
PRUSS2_IRQ_18	pr2_pru_mst_intr[2]_intr_req	pru0 or pru1

Table 30-208. PRU-ICSS1/PRU-ICSS2 Internal Interrupts (continued)

PRU-ICSS INTC IRQ input	PRU-ICSS Internal Interrupt Signal Name	Source
PRUSS2_IRQ_17	pr2_pru_mst_intr[1]_intr_req	pru0 or pru1
PRUSS2_IRQ_16	pr2_pru_mst_intr[0]_intr_req	pru0 or pru1
PRUSS2_IRQ_15	pr2_ecap_intr_req	PRUSS2 eCAP
PRUSS2_IRQ_8	pr2_digio_event_req	PRUSS2 IEP (Ethercat)
PRUSS2_IRQ_7	pr2_iep_tim_cap_cmp_pend	PRUSS2 IEP
PRUSS2_IRQ_6	pr2_uart_uint_intr_req	PRUSS2 UART
PRUSS2_IRQ_5	pr2_uart_utxevt_intr_req	PRUSS2 UART
PRUSS2_IRQ_4	pr2_uart_urxevt_intr_req	PRUSS2 UART
PRUSS2_IRQ_3	pr2_xfr_timeout	PRUSS2 Scratch Pad
PRUSS2_IRQ_2	pr2_pru1_r31_status_cnt16	PRUSS2.PRU1 (Shift Capture)
PRUSS2_IRQ_1	pr2_pru0_r31_status_cnt16	PRUSS2.PRU0 (Shift Capture)
PRUSS2_IRQ_0	pr2_parity_err_intr_pend	PRUSS2 Parity Logic

The IRQ input lines 32 through 63 receive interrupts which come from various device peripherals located outside PRU-ICSS1 and PRU-ICSS2. They are delivered on the PRUSS1_INTC / PRUSS2_INTC inputs (32 through 63) via the device IRQ_CROSSBAR. For more details on the device IRQ_CROSSBAR signals mapping to the PRUSS1_INTC / PRUSS2_INTC, refer to the Chapter 17, *Interrupt Controllers*. For more details on how to program mapping of the external peripheral IRQ signals to PRUSS1_IRQ_32 through PRUSS1_IRQ_63 / PRUSS2_IRQ_32 through PRUSS2_IRQ_55 inputs of the PRUSS1_INTC/PRUSS2_INTC, respectively, refer to the Section 18.4.6.4, *IRQ_CROSSBAR Module Functional Description*, in Chapter 18, *Control Module*.

Note that for the PRUSS_INTC input lines **32 through 55**, there is an additional multiplexing option programmable in the PRUSS_CFG located register bit - PRUSS_MII_RT[0] MII_RT_EVENT_EN. By default the MII_RT_EVENT_EN is set to 0b0 which selects the IRQ sources to be the PRU-ICSS dedicated device IRQ_CROSSBAR outputs ("**Standard mode**"). By setting MII_RT_EVENT_EN to 0b1, a set of PRU-ICSS MII_RT module associated events, are mapped to the same lines.

The Table 30-209 and the Table 30-210 shows PRU-ICSS1/PRU-ICSS2 MII_RT events mapping on the PRUSS1_INTC / PRUSS2_INTC inputs PRUSS1_IRQ_32 through PRUSS1_IRQ_55 / PRUSS2_IRQ_32 through PRUSS2_IRQ_55 valid for the "MII_RT" mode (with PRUSS_MII_RT[0] MII_RT_EVENT_EN register bit set to "0b1")

Table 30-209. PRU-ICSS1 MII_RT Mode Interrupts

PRU-ICSS1 IRQ [1]	Signal Name (MII_RT Mode enabled) - PRUSS_MII_RT[0] MII_RT_EVENT_EN=0b1
PRUSS1_IRQ_55	Reserved
PRUSS1_IRQ_54	PRU1_RX_EOF
PRUSS1_IRQ_53	MDIO_MII_LINK[1]
PRUSS1_IRQ_52	PORT1_TX_OVERFLOW
PRUSS1_IRQ_51	PORT1_TX_UNDERFLOW
PRUSS1_IRQ_50	PRU1_RX_OVERFLOW
PRUSS1_IRQ_49	PRU1_RX_NIBBLE_ODD
PRUSS1_IRQ_48	PRU1_RX_CRC
PRUSS1_IRQ_47	PRU1_RX_SOF
PRUSS1_IRQ_46	PRU1_RX_SFD
PRUSS1_IRQ_45	PRU1_RX_ERR32
PRUSS1_IRQ_44	PRU1_RX_ERR
PRUSS1_IRQ_43	Reserved
PRUSS1_IRQ_42	PRU0_RX_EOF

(1) Signals 63–56 and 31–0 for MII_RT Mode are the same as for Standard Mode.

Table 30-209. PRU-ICSS1 MII_RT Mode Interrupts (continued)

PRU-ICSS1 IRQ [1]	Signal Name (MII_RT Mode enabled) - PRUSS_MII_RT[0] MII_RT_EVENT_EN=0b1
PRUSS1_IRQ_41	MDIO_MII_LINK[0]
PRUSS1_IRQ_40	PORT0_TX_OVERFLOW
PRUSS1_IRQ_39	PORT0_TX_UNDERFLOW
PRUSS1_IRQ_38	PRU0_RX_OVERFLOW
PRUSS1_IRQ_37	PRU0_RX_NIBBLE_ODD
PRUSS1_IRQ_36	PRU0_RX_CRC
PRUSS1_IRQ_35	PRU0_RX_SOF
PRUSS1_IRQ_34	PRU0_RX_SFD
PRUSS1_IRQ_33	PRU0_RX_ERR32
PRUSS1_IRQ_32	PRU0_RX_ERR

Table 30-210. PRU-ICSS2 MII_RT Mode Mode Interrupts

PRU-ICSS2 IRQ [1]	Signal Name (MII_RT Mode enabled) - PRUSS_MII_RT[0] MII_RT_EVENT_EN=0b1
PRUSS2_IRQ_55	Reserved
PRUSS2_IRQ_54	PRU1_RX_EOF
PRUSS2_IRQ_53	MDIO_MII_LINK[1]
PRUSS2_IRQ_52	PORT1_TX_OVERFLOW
PRUSS2_IRQ_51	PORT1_TX_UNDERFLOW
PRUSS2_IRQ_50	PRU1_RX_OVERFLOW
PRUSS2_IRQ_49	PRU1_RX_NIBBLE_ODD
PRUSS2_IRQ_48	PRU1_RX_CRC
PRUSS2_IRQ_47	PRU1_RX_SOF
PRUSS2_IRQ_46	PRU1_RX_SFD
PRUSS2_IRQ_45	PRU1_RX_ERR32
PRUSS2_IRQ_44	PRU1_RX_ERR
PRUSS2_IRQ_43	Reserved
PRUSS2_IRQ_42	PRU0_RX_EOF
PRUSS2_IRQ_41	MDIO_MII_LINK[0]
PRUSS2_IRQ_40	PORT0_TX_OVERFLOW
PRUSS2_IRQ_39	PORT0_TX_UNDERFLOW
PRUSS2_IRQ_38	PRU0_RX_OVERFLOW
PRUSS2_IRQ_37	PRU0_RX_NIBBLE_ODD
PRUSS2_IRQ_36	PRU0_RX_CRC
PRUSS2_IRQ_35	PRU0_RX_SOF
PRUSS2_IRQ_34	PRU0_RX_SFD
PRUSS2_IRQ_33	PRU0_RX_ERR32
PRUSS2_IRQ_32	PRU0_RX_ERR

(1) Signals 63–56 and 31–0 for MII_RT Mode are the same as for Stamdard Mode.

While in the Standard mode (default), the PRU-ICSS interrupt controller PRUSS_IRQ_32 through PRUSS_IRQ_55 input lines are mapped to PRU-ICSS external events via the device IRQ_CROSSBAR, in the MII_RT mode (bit MII_RT_EVENT_EN=0b1), the same PRUSS_INTC inputs are directly mapped to PRU-ICSS internally or externally generated MII_MDIO and MII_RT RX/TX signals (i.e. not through the IRQ_CROSSBAR).

For more details on the device IRQ_CROSSBAR signals mapping to the PRUSS1_INTC/ PRUSS2_INTC , refer to the Chapter 17, *Interrupt Controllers*. For more details on the PRU-ICSS1/PRU-ICSS2 external peripheral IRQ signals programmable mapping to PRUSS1_IRQ_32 through PRUSS1_IRQ_55 / PRUSS2_IRQ_32 through PRUSS2_IRQ_55 inputs of the PRUSS1_INTC/PRUSS2_INTC, respectively, refer to the Section 18.4.6.4, *IRQ_CROSSBAR Module Functional Description*, in Chapter 18, *Control Module*.

30.1.6.5 PRU-ICSS Interrupt Controller Register Manual

This section describes the PRU-ICSS interrupt controller registers.

30.1.6.5.1 PRUSS_INTC Instance Summary

Table 30-211. PRUSS_INTC Instance Summary

Module Name	Base Address	Size
PRUSS1_INTC	0x4B22 0000	5380 Bytes
PRUSS2_INTC	0x4B2A 0000	5380 Bytes

30.1.6.5.2 PRUSS_INTC Registers

30.1.6.5.2.1 PRUSS_INTC Register Summary

Table 30-212. PRUSS1_INTC Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS1_INTC Base Address
PRUSS_INTC_REVID	R	32	0x0000 0000	0x4B22 0000
PRUSS_INTC_CR	RW	32	0x0000 0004	0x4B22 0004
PRUSS_INTC_GER	RW	32	0x0000 0010	0x4B22 0010
PRUSS_INTC_GNLR	RW	32	0x0000 001C	0x4B22 001C
PRUSS_INTC_SISR	W	32	0x0000 0020	0x4B22 0020
PRUSS_INTC_SICR	W	32	0x0000 0024	0x4B22 0024
PRUSS_INTC_EISR	W	32	0x0000 0028	0x4B22 0028
PRUSS_INTC_EICR	W	32	0x0000 002C	0x4B22 002C
PRUSS_INTC_HIEISR	RW	32	0x0000 0034	0x4B22 0034
PRUSS_INTC_HIDISR	RW	32	0x0000 0038	0x4B22 0038
PRUSS_INTC_GPIR	R	32	0x0000 0080	0x4B22 0080
PRUSS_INTC_SRSR0	RW	32	0x0000 0200	0x4B22 0200
PRUSS_INTC_SRSR1	RW	32	0x0000 0204	0x4B22 0204
PRUSS_INTC_SECR0	RW	32	0x0000 0280	0x4B22 0280
PRUSS_INTC_SECR1	RW	32	0x0000 0284	0x4B22 0284
PRUSS_INTC_ESR0	RW	32	0x0000 0300	0x4B22 0300
PRUSS_INTC_ERS1	RW	32	0x0000 0304	0x4B22 0304
PRUSS_INTC_ECR0	W	32	0x0000 0380	0x4B22 0380
PRUSS_INTC_ECR1	W	32	0x0000 0384	0x4B22 0384
PRUSS_INTC_CMRi[(1)]	RW	32	0x0000 0400 + (0x4*i)	0x4B22 0400 + (0x4*i)
PRUSS_INTC_HMR0	RW	32	0x0000 0800	0x4B22 0800
PRUSS_INTC_HMR1	RW	32	0x0000 0804	0x4B22 0804
PRUSS_INTC_HMR2	RW	32	0x0000 0808	0x4B22 0808
PRUSS_INTC_HIPIRj[(2)]	R	32	0x0000 0900 + (0x4*j)	0x4B22 0900 + (0x4*j)

(1) i=0 to 15
(2) j=0 to 9

Table 30-212. PRUSS1_INTC Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS1_INTC Base Address
PRUSS_INTC_SIPR0	RW	32	0x0000 0D00	0x4B22 0D00
PRUSS_INTC_SIPR1	RW	32	0x0000 0D04	0x4B22 0D04
PRUSS_INTC_SITR0	RW	32	0x0000 0D80	0x4B22 0D80
PRUSS_INTC_SITR1	RW	32	0x0000 0D84	0x4B22 0D84
PRUSS_INTC_HINLRj(2)	RW	32	0x0000 1100 + (0x4*j)	0x4B22 1100 + (0x4*j)
PRUSS_INTC_HIER	RW	32	0x0000 1500	0x4B22 1500

Table 30-213. PRUSS2_INTC Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS2_INTC Base Address
PRUSS_INTC_REVID	R	32	0x0000 0000	0x4B2A 0000
PRUSS_INTC_CR	RW	32	0x0000 0004	0x4B2A 0004
PRUSS_INTC_GER	RW	32	0x0000 0010	0x4B2A 0010
PRUSS_INTC_GNLR	RW	32	0x0000 001C	0x4B2A 001C
PRUSS_INTC_SISR	W	32	0x0000 0020	0x4B2A 0020
PRUSS_INTC_SICR	W	32	0x0000 0024	0x4B2A 0024
PRUSS_INTC_EISR	W	32	0x0000 0028	0x4B2A 0028
PRUSS_INTC_EICR	W	32	0x0000 002C	0x4B2A 002C
PRUSS_INTC_HIEISR	RW	32	0x0000 0034	0x4B2A 0034
PRUSS_INTC_HIDISR	RW	32	0x0000 0038	0x4B2A 0038
PRUSS_INTC_GPIR	R	32	0x0000 0080	0x4B2A 0080
PRUSS_INTC_SRSR0	RW	32	0x0000 0200	0x4B2A 0200
PRUSS_INTC_SRSR1	RW	32	0x0000 0204	0x4B2A 0204
PRUSS_INTC_SECR0	RW	32	0x0000 0280	0x4B2A 0280
PRUSS_INTC_SECR1	RW	32	0x0000 0284	0x4B2A 0284
PRUSS_INTC_ESR0	RW	32	0x0000 0300	0x4B2A 0300
PRUSS_INTC_ERS1	RW	32	0x0000 0304	0x4B2A 0304
PRUSS_INTC_ECR0	W	32	0x0000 0380	0x4B2A 0380
PRUSS_INTC_ECR1	W	32	0x0000 0384	0x4B2A 0384
PRUSS_INTC_CMRi(1)	RW	32	0x0000 0400 + (0x4*i)	0x4B2A 0400 + (0x4*i)
PRUSS_INTC_HMR0	RW	32	0x0000 0800	0x4B2A 0800
PRUSS_INTC_HMR1	RW	32	0x0000 0804	0x4B2A 0804
PRUSS_INTC_HMR2	RW	32	0x0000 0808	0x4B2A 0808
PRUSS_INTC_HIPIRj(2)	R	32	0x0000 0900 + (0x4*j)	0x4B2A 0900 + (0x4*j)
PRUSS_INTC_SIPR0	RW	32	0x0000 0D00	0x4B2A 0D00
PRUSS_INTC_SIPR1	RW	32	0x0000 0D04	0x4B2A 0D04
PRUSS_INTC_SITR0	RW	32	0x0000 0D80	0x4B2A 0D80
PRUSS_INTC_SITR1	RW	32	0x0000 0D84	0x4B2A 0D84
PRUSS_INTC_HINLRj(2)	RW	32	0x0000 1100 + (0x4*j)	0x4B2A 1100 + (0x4*j)
PRUSS_INTC_HIER	RW	32	0x0000 1500	0x4B2A 1500

(1) i=0 to 15
(2) j=0 to 9

30.1.6.5.2.2 PRUSS_INTC Register Description

Table 30-214. PRUSS_INTC_REVID

Address Offset	0x0000 0000		
Physical Address	0x4B22 0000 0x4B2A 0000	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	Revision ID Register		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REVISION																															

Bits	Field Name	Description	Type	Reset
31:0	REVISION	IP Revision	R	0x-(1)

(1) TI Internal data

Table 30-215. Register Call Summary for Register PRUSS_INTC_REVID

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

Table 30-216. PRUSS_INTC_CR

Address Offset	0x0000 0004		
Physical Address	0x4B22 0004 0x4B2A 0004	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Control Register holds global control parameters and can forces a soft reset on the module.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																											PRIORITY_HOLD_MODE	NEST_MODE		WAKEUP_MODE	RESERVED

Bits	Field Name	Description	Type	Reset
31:5	RESERVED		R	0x00000
4	PRIORITY_HOLD_MODE	Reserved	RW	0x0
3:2	NEST_MODE	The nesting mode. 0 = no nesting 1 = automatic individual nesting (per host interrupt) 2 = automatic global nesting (over all host interrupts) 3 = manual nesting	RW	0x0
1	WAKEUP_MODE	Reserved	RW	0x0
0	RESERVED		R	0

Table 30-217. Register Call Summary for Register PRUSS_INTC_CR

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

Table 30-218. PRUSS_INTC_GER

Address Offset	0x0000 0010		
Physical Address	0x4B22 0010 0x4B2A 0010	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Global Host Interrupt Enable Register enables all the host interrupts. Individual host interrupts are still enabled or disabled from their individual enables and are not overridden by the global enable.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															ENABLE_HINT_ANY

Bits	Field Name	Description	Type	Reset
31:1	RESERVED		R	0x0000 000
0	ENABLE_HINT_ANY	The current global enable value when read. Writes set the global enable.	RW	0

Table 30-219. Register Call Summary for Register PRUSS_INTC_GER

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Processing: [0]
- PRU-ICSS Interrupt Controller Basic Programming Model: [1]
- PRUSS_INTC Register Summary: [2] [3]

Table 30-220. PRUSS_INTC_GNLR

Address Offset	0x0000 001C		
Physical Address	0x4B22 001C 0x4B2A 001C	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Global Nesting Level Register allows the checking and setting of the global nesting level across all host interrupts when automatic global nesting mode is set. The nesting level is the channel (and all of lower priority) that are nested out because of a current interrupt. This register is only available when nesting is configured.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUTO_OVERRIDE	RESERVED																							GLB_NEST_LEVEL							

Bits	Field Name	Description	Type	Reset
31	AUTO_OVERRIDE	Always read as 0. Writes of 1 override the automatic nesting and set the nesting_level to the written data.	W	0x0
30:9	RESERVED		R	0x00000

Bits	Field Name	Description	Type	Reset
8:0	GLB_NEST_LEVEL	The current global nesting level (highest channel that is nested). Writes set the nesting level. In auto nesting mode this value is updated internally unless the auto_override bit is set.	RW	0x100

Table 30-221. Register Call Summary for Register PRUSS_INTC_GNLR

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Nesting: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-222. PRUSS_INTC_SISR

Address Offset	0x0000 0020		
Physical Address	0x4B22 0020 0x4B2A 0020	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Status Indexed Set Register allows setting the status of an interrupt. The interrupt to set is the index value written. This sets the Raw Status Register bit of the given index.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								STATUS_SET_INDEX							

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x0000 00
9:0	STATUS_SET_INDEX	Writes set the status of the interrupt given in the index value. Reads return 0.	W	0x00

Table 30-223. Register Call Summary for Register PRUSS_INTC_SISR

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

Table 30-224. PRUSS_INTC_SICR

Address Offset	0x0000 0024		
Physical Address	0x4B22 0024 0x4B2A 0024	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Status Indexed Clear Register allows clearing the status of an interrupt. The interrupt to clear is the index value written. This clears the Raw Status Register bit of the given index.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								STATUS_CLR_INDEX							

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x0000 00
9:0	STATUS_CLR_INDEX	Writes clear the status of the interrupt given in the index value. Reads return 0.	W	0x0

Table 30-225. Register Call Summary for Register PRUSS_INTC_SICR

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Status Clearing: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-226. PRUSS_INTC_EISR

Address Offset	0x0000 0028		
Physical Address	0x4B22 0028 0x4B2A 0028	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Enable Indexed Set Register allows enabling an interrupt. The interrupt to enable is the index value written. This sets the Enable Register bit of the given index.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								ENABLE_SET_INDEX							

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x0000 00
9:0	ENABLE_SET_INDEX	Writes set the enable of the interrupt given in the index value. Reads return 0.	W	0x0

Table 30-227. Register Call Summary for Register PRUSS_INTC_EISR

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Processing: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-228. PRUSS_INTC_EICR

Address Offset	0x0000 002C		
Physical Address	0x4B22 002C 0x4B2A 002C	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Enable Indexed Clear Register allows disabling an interrupt. The interrupt to disable is the index value written. This clears the Enable Register bit of the given index.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								ENABLE_CLR_INDEX							

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x0000 00
9:0	ENABLE_CLR_INDEX	Writes clear the enable of the interrupt given in the index value. Reads return 0.	W	0x0

Table 30-229. Register Call Summary for Register PRUSS_INTC_EICR

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Disabling: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-230. PRUSS_INTC_HIEISR

Address Offset	0x0000 0034		
Physical Address	0x4B22 0034 0x4B2A 0034	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Host Interrupt Enable Indexed Set Register allows enabling a host interrupt output. The host interrupt to enable is the index value written. This enables the host interrupt output or triggers the output again if already enabled.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						HINT_ENABLE_SET_INDEX									

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x0000 00
9:0	HINT_ENABLE_SET_INDEX	Writes set the enable of the host interrupt given in the index value. Reads return 0.	RW	0x0

Table 30-231. Register Call Summary for Register PRUSS_INTC_HIEISR

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Processing: [0]
- PRU-ICSS Interrupt Controller Basic Programming Model: [1]
- PRUSS_INTC Register Summary: [2] [3]

Table 30-232. PRUSS_INTC_HIDISR

Address Offset	0x0000 0038		
Physical Address	0x4B22 0038 0x4B2A 0038	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Host Interrupt Enable Indexed Clear Register allows disabling a host interrupt output. The host interrupt to disable is the index value written. This disables the host interrupt output.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						HINT_ENABLE_CLR_INDEX									

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x0000 00
9:0	HINT_ENABLE_CLR_INDEX	Writes clear the enable of the host interrupt given in the index value. Reads return 0.	RW	0x0

Table 30-233. Register Call Summary for Register PRUSS_INTC_HIDISR

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

Table 30-234. PRUSS_INTC_GPIR

Address Offset	0x0000 0080		
Physical Address	0x4B22 0080 0x4B2A 0080	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Global Prioritized Index Register shows the interrupt number of the highest priority interrupt pending across all the host interrupts.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GLB_NONE	RESERVED																							GLB_PRI_INTR							

Bits	Field Name	Description	Type	Reset
31	GLB_NONE	No Interrupt is pending. Can be used by host to test for a negative value to see if no interrupts are pending.	R	0x1
30:10	RESERVED		R	0x0000 00
9:0	GLB_PRI_INTR	The currently highest priority interrupt index pending across all the host interrupts.	R	0x0

Table 30-235. Register Call Summary for Register PRUSS_INTC_GPIR

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Channel Mapping: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-236. PRUSS_INTC_SRSR0

Address Offset	0x0000 0200		
Physical Address	0x4B22 0200 0x4B2A 0200	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Status Raw Set Register0 show the pending enabled status of the system interrupts 0 to 31. Software can write to the Status Set Registers to set a system interrupt without a hardware trigger. There is one bit per system interrupt.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAW_STATUS_31_0																															

Bits	Field Name	Description	Type	Reset
31:0	RAW_STATUS_31_0	System interrupt raw status and setting of the system interrupts 0 to 31. Reads return the raw status. Write a 1 in a bit position to set the status of the system interrupt. Writing a 0 has no effect.	RW	0x0

Table 30-237. Register Call Summary for Register PRUSS_INTC_SRSR0

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

Table 30-238. PRUSS_INTC_SRSR1

Address Offset	0x0000 0204		
Physical Address	0x4B22 0204 0x4B2A 0204	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Status Raw Set Register1 show the pending enabled status of the system interrupts 32 to 63. Software can write to the Status Set Registers to set a system interrupt without a hardware trigger. There is one bit per system interrupt.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAW_STATUS_63_32																															

Bits	Field Name	Description	Type	Reset
31:0	RAW_STATUS_63_32	System interrupt raw status and setting of the system interrupts 32 to 63. Reads return the raw status. Write a 1 in a bit position to set the status of the system interrupt. Writing a 0 has no effect.	RW	0x0

Table 30-239. Register Call Summary for Register PRUSS_INTC_SRSR1

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

Table 30-240. PRUSS_INTC_SECR0

Address Offset	0x0000 0280		
Physical Address	0x4B22 0280 0x4B2A 0280	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Status Enabled Clear Register0 show the pending enabled status of the system interrupts 0 to 31. Software can write to the Status Clear Registers to clear a system interrupt after it has been serviced. If a system interrupt status is not cleared then another host interrupt may not be triggered or another host interrupt may be triggered incorrectly. There is one bit per system interrupt.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENA_STATUS_31_0																															

Bits	Field Name	Description	Type	Reset
31:0	ENA_STATUS_31_0	System interrupt enabled status and clearing of the system interrupts 0 to 31. Reads return the enabled status (before enabling with the Enable Registers). Write a 1 in a bit position to clear the status of the system interrupt. Writing a 0 has no effect.	RW	0x0

Table 30-241. Register Call Summary for Register PRUSS_INTC_SECR0

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Status Checking: [0] [1]
- PRU-ICSS Interrupt Status Clearing: [2]
- PRU-ICSS Interrupt Controller Basic Programming Model: [3]
- PRUSS_INTC Register Summary: [4] [5]

Table 30-242. PRUSS_INTC_SECR1

Address Offset	0x0000 0284		
Physical Address	0x4B22 0284 0x4B2A 0284	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Status Enabled Clear Register1 show the pending enabled status of the system interrupts 32 to 63. Software can write to the Status Clear Registers to clear a system interrupt after it has been serviced. If a system interrupt status is not cleared then another host interrupt may not be triggered or another host interrupt may be triggered incorrectly. There is one bit per system interrupt.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENA_STATUS_63_32																															

Bits	Field Name	Description	Type	Reset
31:0	ENA_STATUS_63_32	System interrupt enabled status and clearing of the system interrupts 32 to 63. Reads return the enabled status (before enabling with the Enable Registers). Write a 1 in a bit position to clear the status of the system interrupt. Writing a 0 has no effect.	RW	0x0

Table 30-243. Register Call Summary for Register PRUSS_INTC_SECR1

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Status Checking: [0] [1]
- PRU-ICSS Interrupt Status Clearing: [2]
- PRUSS_INTC Register Summary: [3] [4]

Table 30-244. PRUSS_INTC_ESR0

Address Offset	0x0000 0300		
Physical Address	0x4B22 0300 0x4B2A 0300	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Enable Set Register0 enables system interrupts 0 to 31 to trigger outputs. System interrupts that are not enabled do not interrupt the host. There is a bit per system interrupt.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENABLE_SET_31_0																															

Bits	Field Name	Description	Type	Reset
31:0	ENABLE_SET_31_0	System interrupt enables system interrupts 0 to 31. Read returns the enable value (0 = disabled, 1 = enabled). Write a 1 in a bit position to set that enable. Writing a 0 has no effect.	RW	0x0

Table 30-245. Register Call Summary for Register PRUSS_INTC_ESR0

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

Table 30-246. PRUSS_INTC_ERS1

Address Offset	0x0000 0304		
Physical Address	0x4B22 0304 0x4B2A 0304	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Enable Set Register1 enables system interrupts 32 to 63 to trigger outputs. System interrupts that are not enabled do not interrupt the host. There is a bit per system interrupt.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENABLE_SET_63_32																															

Bits	Field Name	Description	Type	Reset
31:0	ENABLE_SET_63_32	System interrupt enables system interrupts 32 to 63. Read returns the enable value (0 = disabled, 1 = enabled). Write a 1 in a bit position to set that enable. Writing a 0 has no effect.	RW	0x0

Table 30-247. Register Call Summary for Register PRUSS_INTC_ERS1

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

Table 30-248. PRUSS_INTC_ECR0

Address Offset	0x0000 0380		
Physical Address	0x4B22 0380 0x4B2A 0380	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Enable Clear Register0 disables system interrupts 0 to 31 to map to channels. System interrupts that are not enabled do not interrupt the host. There is a bit per system interrupt.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENABLE_CLR_31_0																															

Bits	Field Name	Description	Type	Reset
31:0	ENABLE_CLR_31_0	System interrupt enables system interrupts 0 to 31. Write a 1 in a bit position to clear that enable. Writing a 0 has no effect.	W	0x0

Table 30-249. Register Call Summary for Register PRUSS_INTC_ECR0

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Disabling: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-250. PRUSS_INTC_ECR1

Address Offset	0x0000 0384		
Physical Address	0x4B22 0384 0x4B2A 0384	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Enable Clear Register1 disables system interrupts 32 to 63 to map to channels. System interrupts that are not enabled do not interrupt the host. There is a bit per system interrupt.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENABLE_CLR_63_32																															

Bits	Field Name	Description	Type	Reset
31:0	ENABLE_CLR_63_32	System interrupt enables system interrupts 32 to 63. Write a 1 in a bit position to clear that enable. Writing a 0 has no effect.	W	0x0

Table 30-251. Register Call Summary for Register PRUSS_INTC_ECR1

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Disabling: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-252. PRUSS_INTC_CMRi

Address Offset	0x0000 0400 + (0x4*i)	**Index**	i = 0 to 15
Physical Address	0x4B22 0400 + (0x4*i) 0x4B2A 0400 + (0x4*i)	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	There are 16 identical CMR registers (i=0 to 15). The Channel Map Register_i specify the channel for the system interrupts k to k+3, where k=4*i. There is one register per 4 system interrupts.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				CH_MAP_3				RESERVED				CH_MAP_2				RESERVED				CH_MAP_1				RESERVED				CH_MAP_0			

Bits	Field Name	Description	Type	Reset
31:28	RESERVED		R	0x0
27:24	CH_MAP_3	Sets the channel for the system interrupt (k+3). Where k=i*4	RW	0x0
23:20	RESERVED		R	0x0
19:16	CH_MAP_2	Sets the channel for the system interrupt (k+2). Where k=i*4	RW	0x0
15:12	RESERVED		R	0x0
11:8	CH_MAP_1	Sets the channel for the system interrupt (k+1). Where k=i*4	RW	0x0
7:4	RESERVED		R	0x0
3:0	CH_MAP_0	Sets the channel for the system interrupt k. Where k=i*4	RW	0x0

Table 30-253. Register Call Summary for Register PRUSS_INTC_CMRi

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Channel Mapping: [0]
- PRU-ICSS Interrupt Controller Basic Programming Model: [1]
- PRUSS_INTC Register Summary: [2] [3]

Table 30-254. PRUSS_INTC_HMR0

Address Offset	0x0000 0800		
Physical Address	0x4B22 0800 0x4B2A 0800	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Host Interrupt Map Register0 define the host interrupt for channels 0 to 3. There is one register per 4 channels. Channels with forced host interrupt mappings will have their fields read-only.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				HINT_MAP_3				RESERVED				HINT_MAP_2				RESERVED				HINT_MAP_1				RESERVED				HINT_MAP_0			

Bits	Field Name	Description	Type	Reset
31:28	RESERVED		R	0x0
27:24	HINT_MAP_3	HOST INTERRUPT MAP FOR CHANNEL 3	RW	0x0
23:20	RESERVED		R	0x0
19:16	HINT_MAP_2	HOST INTERRUPT MAP FOR CHANNEL 2	RW	0x0
15:12	RESERVED		R	0x0
11:8	HINT_MAP_1	HOST INTERRUPT MAP FOR CHANNEL 1	RW	0x0
7:4	RESERVED		R	0x0

Bits	Field Name	Description	Type	Reset
3:0	HINT_MAP_0	HOST INTERRUPT MAP FOR CHANNEL 0	RW	0x0

Table 30-255. Register Call Summary for Register PRUSS_INTC_HMR0

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Channel Mapping: [0]
- PRU-ICSS Interrupt Controller Basic Programming Model: [1]
- PRUSS_INTC Register Summary: [2] [3]

Table 30-256. PRUSS_INTC_HMR1

Address Offset	0x0000 0804		
Physical Address	0x4B22 0804 0x4B2A 0804	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Host Interrupt Map Register1 define the host interrupt for channels 4 to 7. There is one register per 4 channels. Chan_statusnels with forced host interrupt mappings will have their fields read-only.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				HINT_MAP_7				RESERVED				HINT_MAP_6				RESERVED				HINT_MAP_5				RESERVED				HINT_MAP_4			

Bits	Field Name	Description	Type	Reset
31:28	RESERVED		R	0x0
27:24	HINT_MAP_7	HOST INTERRUPT MAP FOR CHANNEL 7	RW	0x0
23:20	RESERVED		R	0x0
19:16	HINT_MAP_6	HOST INTERRUPT MAP FOR CHANNEL 6	RW	0x0
15:12	RESERVED		R	0x0
11:8	HINT_MAP_5	HOST INTERRUPT MAP FOR CHANNEL 5	RW	0x0
7:4	RESERVED		R	0x0
3:0	HINT_MAP_4	HOST INTERRUPT MAP FOR CHANNEL 4	RW	0x0

Table 30-257. Register Call Summary for Register PRUSS_INTC_HMR1

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

Table 30-258. PRUSS_INTC_HMR2

Address Offset	0x0000 0808		
Physical Address	0x4B22 0808 0x4B2A 0808	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Host Interrupt Map Register2 define the host interrupt for channels 8 to 9. There is one register per 4 channels. Channels with forced host interrupt mappings will have their fields read-only.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																				HINT_MAP_9				RESERVED				HINT_MAP_8			

Bits	Field Name	Description	Type	Reset
31:12	RESERVED		R	0x00000
11:8	HINT_MAP_9	HOST INTERRUPT MAP FOR CHANNEL 9	RW	0x0
7:4	RESERVED		R	0x0
3:0	HINT_MAP_8	HOST INTERRUPT MAP FOR CHANNEL 8	RW	0x0

Table 30-259. Register Call Summary for Register PRUSS_INTC_HMR2

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Channel Mapping: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-260. PRUSS_INTC_HIPIRj

Address Offset	0x0000 0900 + (0x4*j)	**Index**	j = 0 to 9
Physical Address	0x4B22 0900 + (0x4*j) 0x4B2A 0900 + (0x4*j)	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Host Interrupt Prioritized Index Register_j (where j=0 to 9) shows the highest priority current pending interrupt for the host interrupt j. There is one register per host interrupt.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NONE_HINT_j								RESERVED																PRI_HINT_j							

Bits	Field Name	Description	Type	Reset
31	NONE_HINT	No pending interrupt.	R	0x1
30:10	RESERVED		R	0x0000 00
9:0	PRI_HINT	HOST INT j PRIORITIZED INTERRUPT. Interrupt number of the highest priority pending interrupt for this host interrupt.	R	0x0

Table 30-261. Register Call Summary for Register PRUSS_INTC_HIPIRj

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Channel Mapping: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-262. PRUSS_INTC_SIPR0

Address Offset	0x0000 0D00		
Physical Address	0x4B22 0D00 0x4B2A 0D00	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Polarity Register0 define the polarity of the system interrupts 0 to 31. There is a polarity for each system interrupt. The polarity of all system interrupts is active high; always write 1 to the bits of this register.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLARITY_31_0																															

Bits	Field Name	Description	Type	Reset
31:0	POLARITY_31_0	Interrupt polarity of the system interrupts 0 to 31. 0 = active low. 1 = active high.	RW	0x1

Table 30-263. Register Call Summary for Register PRUSS_INTC_SIPR0

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Controller Basic Programming Model: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-264. PRUSS_INTC_SIPR1

Address Offset	0x0000 0D04		
Physical Address	0x4B22 0D04 0x4B2A 0D04	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Polarity Register1 define the polarity of the system interrupts 32 to 63. There is a polarity for each system interrupt. The polarity of all system interrupts is active high; always write 1 to the bits of this register.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLARITY_63_32																															

Bits	Field Name	Description	Type	Reset
31:0	POLARITY_63_32	Interrupt polarity of the system interrupts 32 to 63. 0 = active low. 1 = active high.	RW	0x1

Table 30-265. Register Call Summary for Register PRUSS_INTC_SIPR1

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Controller Basic Programming Model: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-266. PRUSS_INTC_SITR0

Address Offset	0x0000 0D80		
Physical Address	0x4B22 0D80 0x4B2A 0D80	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Type Register0 define the type of the system interrupts 0 to 31. There is a type for each system interrupt. The type of all system interrupts is pulse; always write 0 to the bits of this register.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE_31_0																															

Bits	Field Name	Description	Type	Reset
31:0	TYPE_31_0	Interrupt type of the system interrupts 0 to 31. 0 = level or pulse interrupt. 1 = edge interrupt (required edge detect).	RW	0x0

Table 30-267. Register Call Summary for Register PRUSS_INTC_SITR0

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Controller Basic Programming Model: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-268. PRUSS_INTC_SITR1

Address Offset	0x0000 0D84		
Physical Address	0x4B22 0D84 0x4B2A 0D84	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Type Register1 define the type of the system interrupts 32 to 63. There is a type for each system interrupt. The type of all system interrupts is pulse; always write 0 to the bits of this register.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE_63_32																															

Bits	Field Name	Description	Type	Reset
31:0	TYPE_63_32	Interrupt type of the system interrupts 32 to 63. 0 = level or pulse interrupt. 1 = edge interrupt (required edge detect).	RW	0x0

Table 30-269. Register Call Summary for Register PRUSS_INTC_SITR1

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Controller Basic Programming Model: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-270. PRUSS_INTC_HINLRj

Address Offset	0x0000 1100 + 0x4 * j	**Index**	j=0 to 9
Physical Address	0x4B22 1100 + (0x4*j) 0x4B2A 1100 + (0x4*j)	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Host Interrupt Nesting Level Register_j (where j=0 to 9) display and control the nesting level for host interrupt j. The nesting level controls which channel and lower priority channels are nested. There is one register per host interrupt.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUTO_OVERRIDE	RESERVED																							NEST_HINT_j							

Bits	Field Name	Description	Type	Reset
31	AUTO_OVERRIDE	Reads return 0. Writes of a 1 override the auto updating of the nesting_level and use the write data.	W	0x0
30:9	RESERVED		R	0x00000
8:0	NEST_HINT	Reads return the current nesting level for the host interrupt. Writes set the nesting level for the host interrupt. In auto mode the value is updated internally unless the auto_override is set and then the write data is used.	RW	0x100

Table 30-271. Register Call Summary for Register PRUSS_INTC_HINLRj

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Nesting: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-272. PRUSS_INTC_HIER

Address Offset	0x0000 1500		
Physical Address	0x4B22 1500 0x4B2A 1500	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Host Interrupt Enable Registers enable or disable individual host interrupts. These work separately from the global enables. There is one bit per host interrupt. These bits are updated when writing to the Host Interrupt Enable Index Set and Host Interrupt Enable Index Clear registers.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								ENABLE_HINT							

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x0000 00
9:0	ENABLE_HINT	The enable of the host interrupts (one per bit). 0 = disabled 1 = enabled	RW	0x0

Table 30-273. Register Call Summary for Register PRUSS_INTC_HIER

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

30.1.7 PRU-ICSS UART Module

This section describes an Universal Asynchronous Receive and Transmit (UART) module which is part of the device integrated PRU-ICSS1 and PRU-ICSS2 - PRUSS1_UART0 and PRUSS2_UART0, respectively.

30.1.7.1 PRU-ICSS UART Module Overview

30.1.7.1.1 Purpose of the PRU-ICSS integrated UART Peripheral

The PRUSS_UART0 peripheral is based on the industry standard TL16C550 asynchronous communications element, which in turn is a functional upgrade of the TL16C450. Functionally similar to the TL16C450 on power up (single character or TL16C450 mode), the PRUSS_UART0 can be placed in an alternate FIFO (TL16C550) mode. This relieves the CPU of excessive software overhead by buffering received and transmitted characters. The receiver and transmitter FIFOs store up to 16 bytes including three additional bits of error status per byte for the receiver FIFO.

The PRUSS_UART0 performs serial-to-parallel conversions on data received from a peripheral device and parallel-to-serial conversion on data received from the CPU. The CPU can read the PRUSS_UART0 status at any time. The PRUSS_UART0 includes control capability and a processor interrupt system that can be tailored to minimize software management of the communications link.

The PRUSS_UART0 includes a programmable baud generator capable of dividing the PRUSS_UART0 input clock by divisors from 1 to 65535 and producing a 16× reference clock or a 13× reference clock for the internal transmitter and receiver logic.

30.1.7.1.2 PRU-ICSS UART Key Features

30.1.7.1.2.1 PRU-ICSS UART Module Industry Standard Compliance Statement

The PRUSS_UART0 peripheral is based on the industry standard TL16C550 asynchronous communications element, which is a functional upgrade of the TL16C450. The information in this chapter assumes that user is familiar with these standards.

30.1.7.2 PRU-ICSS UART Environment

This section describes the PRUSS_UART0 module interface to the device environment

30.1.7.2.1 PRU-ICSS UART Pin Multiplexing

Extensive pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package. Pin multiplexing is controlled using a combination of hardware configuration at device reset and software programmable register settings. For more information on the PRUSS_UART0 pin multiplexing, refer to the Section 18.4.6.1.1, *Pad Configuration Registers* in the chapter, *Control Module*.

30.1.7.2.2 PRU-ICSS UART Signal Descriptions

The PRUSS_UART0 utilize a minimal number of signal connections to interface with external devices. The PRUSS_UART0 signal descriptions are included in Table 30-744.

Table 30-274. PRUSS_UART0 Signal Descriptions

Signal Name	Signal Type	Function
UART0_TXD	Output	Serial data transmit
UART0_RXD	Input	Serial data receive
$\overline{\text{UART0\_CTS}}$	Input	Clear-to-Send handshaking signal
$\overline{\text{UART0\_RTS}}$	Output	Request-to-Send handshaking signal

30.1.7.2.3 PRU-ICSS UART Data Format and Protocol Description

30.1.7.2.3.1 PRU-ICSS UART Transmission Protocol

The PRUSS_UART0 transmitter section includes a transmitter hold register (THR), memory mapped in the register PRUSS_UART_RBR_THR_REGISTERS[7:0] DATA bitfield and a transmitter shift register (TSR), which is NOT memory mapped. When the PRUSS_UART0 is in the FIFO mode, THR is a 16-byte FIFO. Transmitter section control is a function of the PRUSS_UART0 line control register PRUSS_UART_LINE_CONTROL_REGISTER. Based on the settings chosen in this register, the PRUSS_UART0 transmitter sends the following to the receiving device:

- 1 START bit
- 5, 6, 7, or 8 data bits
- 1 PARITY bit (optional)
- 1, 1.5, or 2 STOP bits

30.1.7.2.3.2 PRU-ICSS UART Reception Protocol

The PRUSS_UART0 receiver section includes a receiver shift register (RSR), that is not memory mapped, and a receiver buffer register (RBR), memory mapped as the register PRUSS_UART_RBR_THR_REGISTERS [7:0] DATA bitfield. When the PRUSS_UART0 is in the FIFO mode, RBR is a 16-byte FIFO. Receiver section control is a function of the PRUSS_UART0 line control register - PRUSS_UART_LINE_CONTROL_REGISTER. Based on the settings chosen in this register, the PRUSS_UART0 receiver accepts the following from the transmitting device:

- 1 START bit
- 5, 6, 7, or 8 data bits
- 1 PARITY bit (optional)
- 1 STOP bit (any other STOP bits transferred with the above data are not detected)

30.1.7.2.3.3 PRU-ICSS UART Data Format

The PRUSS_UART0 transmits in the following format:

1 START bit + data bits (5, 6, 7, 8) + 1 PARITY bit (optional) + STOP bit (1, 1.5, 2)

It transmits 1 START bit; 5, 6, 7, or 8 data bits, depending on the data width selection; 1 PARITY bit, if parity is selected; and 1, 1.5, or 2 STOP bits, depending on the STOP bit selection.

The PRUSS_UART0 receives in the following format:

1 START bit + data bits (5, 6, 7, 8) + 1 PARITY bit (optional) + 1 STOP bit

It receives 1 START bit; 5, 6, 7, or 8 data bits, depending on the data width selection; 1 PARITY bit, if parity is selected; and 1 STOP bit.

The protocol formats are shown in Figure 30-74.

Figure 30-20. PRU-ICSS UART Protocol Formats

Transmit/Receive for 5-bit data, parity Enable, 1 STOP bit

		D0	D1	D2	D3	D4	PARITY	STOP1

Transmit/Receive for 6-bit data, parity Enable, 1 STOP bit

		D0	D1	D2	D3	D4	D5	PARITY	STOP1

Transmit/Receive for 7-bit data, parity Enable, 1 STOP bit

		D0	D1	D2	D3	D4	D5	D6	PARITY	STOP1

Transmit/Receive for 8-bit data, parity Enable, 1 STOP bit

		D0	D1	D2	D3	D4	D5	D6	D7	PARITY	STOP1

30.1.7.2.3.3.1 Frame Formatting

Character length is specified using the PRUSS_UART_LINE_CONTROL_REGISTER[1-0] WLS bit field (see Table 30-746).

The number of stop-bits is specified using the PRUSS_UART_LINE_CONTROL_REGISTER[2] STB bit (see Table 30-746).

The parity bit is programmed using the PRUSS_UART_LINE_CONTROL_REGISTER[5-3] PEN, EPS, and SP bits (see Table 30-745).

Table 30-275. Relationship Between ST, EPS, and PEN Bits in UART_LCR

ST Bit	EPS Bit	PEN Bit	Parity Option
x	x	0	Parity disabled: No PARITY bit is transmitted or checked.
0	0	1	Odd parity selected: Odd number of logic 1s.
0	1	1	Even parity selected: Even number of logic 1s.
1	0	1	Stick parity selected with PARITY bit transmitted and checked as set.
1	1	1	Stick parity selected with PARITY bit transmitted and checked as cleared.

Table 30-276. Number of STOP Bits Generated

STB Bit	WLS Bit	Word Length Selected with WLS Bits	Number of STOP Bits Generated	Baud Clock (BCLK) Cycles
0	x	Any word length	1	16
1	0h	5 bits	1.5	24
1	1h	6 bits	2	32
1	2h	7 bits	2	32
1	3h	8 bits	2	32

30.1.7.2.4 PRU-ICSS UART Clock Generation and Control

The PRUSS_UART0 bit clock is derived from an input clock to the PRUSS_UART0. See your device-specific data manual to check the maximum data rate supported by the PRUSS_UART0.

Figure 30-75 is a conceptual clock generation diagram for the PRUSS_UART0. The processor clock generator receives a signal from an external clock source and produces a PRUSS_UART0 input clock with a programmed frequency. The PRUSS_UART0 contains a programmable baud generator that takes an input clock and divides it by a divisor in the range between 1 and (2^{16} - 1) to produce a baud clock (BCLK). The frequency of BCLK is sixteen times (16×) the baud rate (each received or transmitted bit lasts 16 BCLK cycles) or thirteen times (13×) the baud rate (each received or transmitted bit lasts 13

BCLK cycles). When the PRUSS_UART0 is receiving, the bit is sampled in the 8th BCLK cycle for 16× over sampling mode and on the 6th BCLK cycle for 13× over-sampling mode. The 16× or 13× reference clock is selected by configuring the mode definition register (MDR) - PRUSS_UART_MODE_DEFINITION_REGISTER [0] OSM_SEL bit. The formula to calculate the divisor is:

$$\text{Divisor} = \frac{\text{UART input clock frequency}}{\text{Desired baud rate} \times 16} \qquad [\text{MDR.OSM\_SEL} = 0]$$

pruss-013

$$\text{Divisor} = \frac{\text{UART input clock frequency}}{\text{Desired baud rate} \times 13} \qquad [\text{MDR.OSM\_SEL} = 1]$$

pruss-014

Two 8-bit register fields:

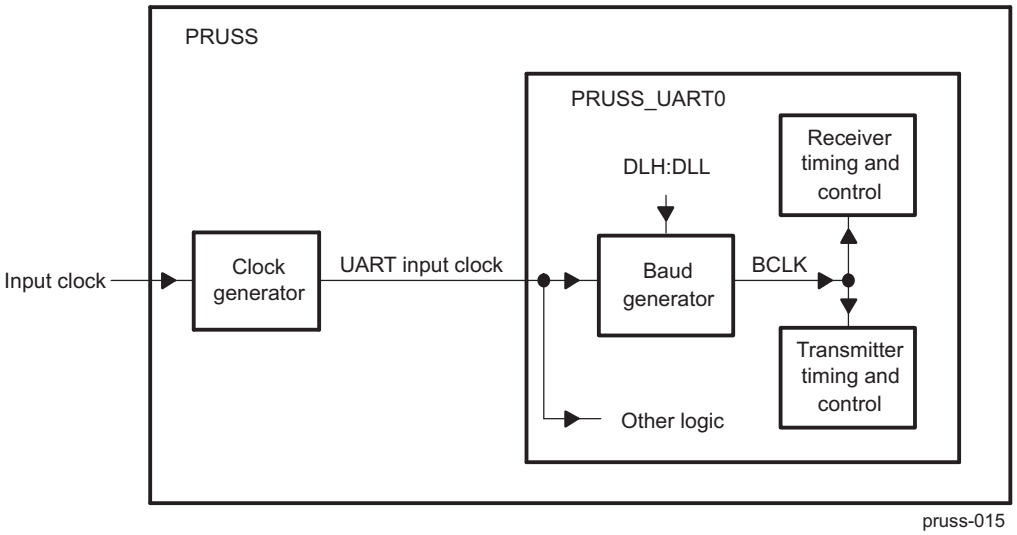
- PRUSS_UART_DIVISOR_REGISTER_MSB_[7:0] DLH
- PRUSS_UART_DIVISOR_REGISTER_LSB_[7:0] DLL,

called divisor latches, hold this 16-bit divisor. DLH holds the most significant bits of the divisor, and DLL holds the least significant bits of the divisor. For information about these register fields, see the PRUSS_UART0 register descriptions in the Section 30.2.7.4, *PRU-ICSS UART Register Manual*. These divisor latches must be loaded during initialization of the PRUSS_UART0 in order to ensure desired operation of the baud generator. Writing to the divisor latches results in two wait states being inserted during the write access while the baud generator is loaded with the new value.

Figure 30-76 summarizes the relationship between the transferred data bit, BCLK, and the PRUSS_UART0 input clock. Note that the timing relationship depicted in Figure 30-76 shows that each bit lasts for 16 BCLK cycles . This is in case of 16x over-sampling mode. For 13× over-sampling mode each bit lasts for 13 BCLK cycles .

Example baud rates and divisor values relative to a 150-MHz PRUSS_UART0 input clock and 16× over-sampling mode are shown in Table 30-747.

Figure 30-21. PRU-ICSS UART Clock Generation Diagram



pruss-015

Figure 30-22. Relationships Between PRU-ICSS UART Data Bit, BCLK, and Input Clock

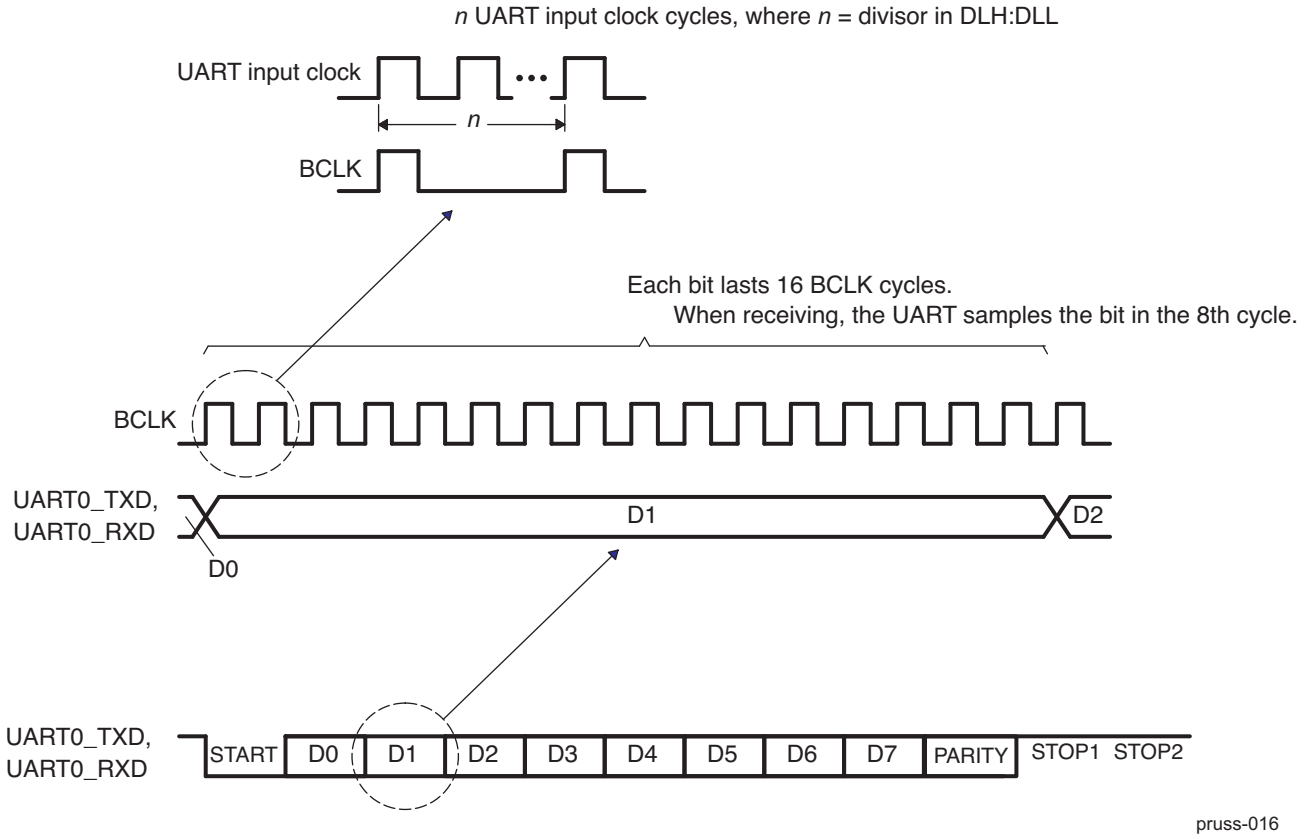


Table 30-277. Baud Rate Examples for 192-MHz PRU-ICSS UART Input Clock and 16× Over-sampling Mode

Baud Rate	Divisor Value	Actual Baud Rate	Error (%)
2400	5000	2400	0.00
4800	2500	4800	0.00
9600	1250	9600	0.00
19200	625	19200	0.00
38400	313	38338.658	-0.16
56000	214	56074.766	0.13
115200	104	115384.6	0.16
128000	94	127659.574	-0.27
3000000	4	3000000	0.00
6000000	2	6000000	0.00
12000000	1	12000000	0.00

Table 30-278. Baud Rate Examples for 192-MHz PRU-ICSS UART Input Clock and 13× Over-sampling Mode

Baud Rate	Divisor Value	Actual Baud Rate	Error (%)
2400	6154	2399.940	-0.0025
4800	3077	4799.880	-0.0025
9600	1538	9602.881	0.03

Table 30-278. Baud Rate Examples for 192-MHz PRU-ICSS UART Input Clock and 13× Over-sampling Mode (continued)

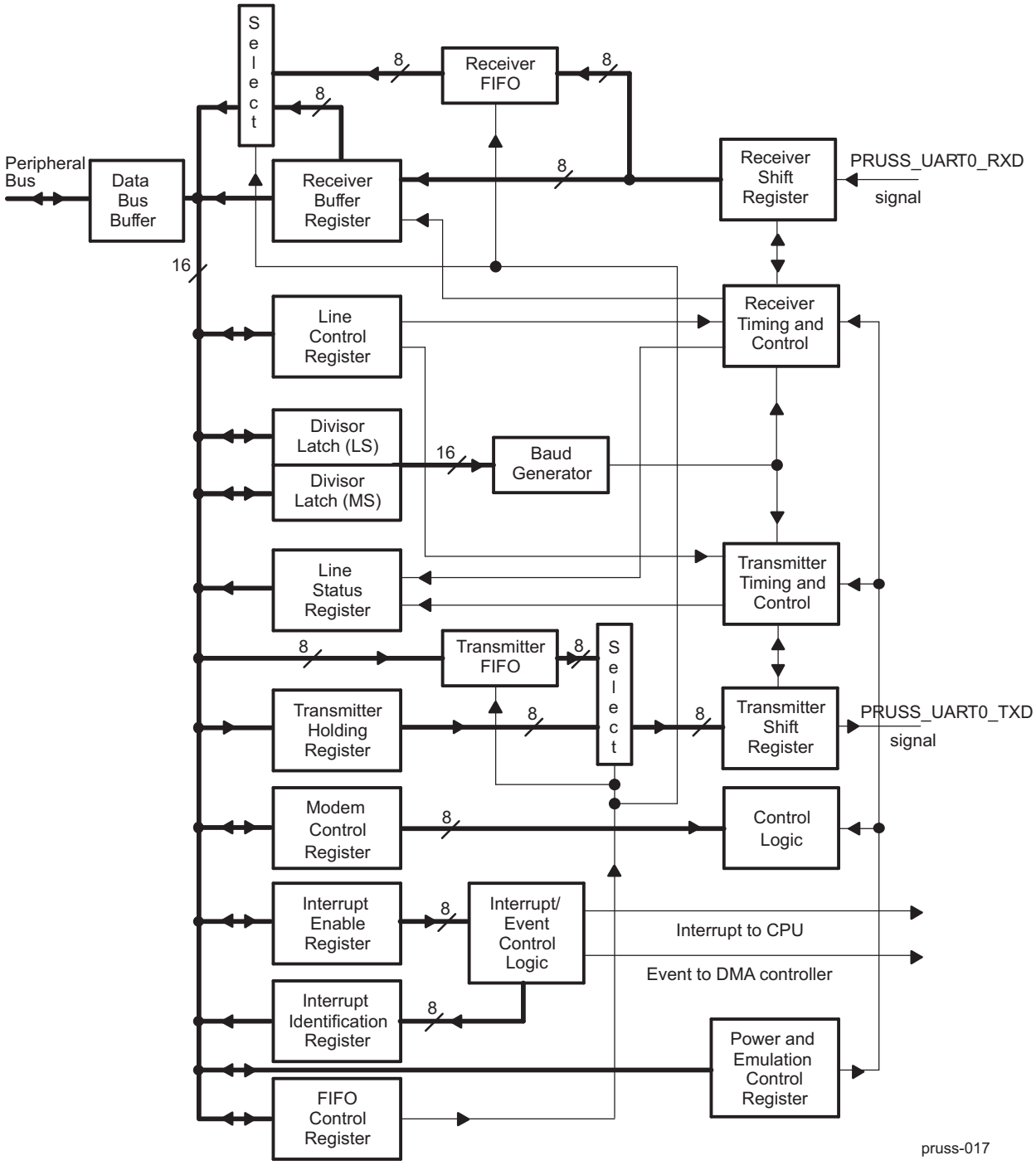
Baud Rate	Divisor Value	Actual Baud Rate	Error (%)
19200	769	19205.762	0.03
38400	385	38361.638	-0.10
56000	264	55944.056	-0.10
115200	128	115384.6	0.16
128000	115	128428.094	0.33

30.1.7.3 PRU-ICSS UART Module Functional Description

30.1.7.3.1 PRU-ICSS UART Functional Block Diagram

A functional block diagram of the PRUSS_UART0 is shown in Figure 30-77.

Figure 30-23. PRU-ICSS UART Block Diagram



NOTE: The value *n* indicates the applicable UART where there are multiple instances. For the PRU-ICSS, there is only one instance and all UART signals should reflect this (e.g., UART0_TXD instead of UARTn_TXD).

30.1.7.3.2 PRU-ICSS UART Reset Considerations

30.1.7.3.2.1 PRU-ICSS UART Software Reset Considerations

Two bits in the power and emulation management register - PRUSS_UART_POWERMANAGEMENT_AND_EMULATION_REGISTER, control resetting the parts of the PRUSS_UART0:

- The bit [14] UTRST controls resetting the transmitter only. If UTRST = 1, the transmitter is active; if UTRST = 0, the transmitter is in reset.
- The bit [13] URRST controls resetting the receiver only. If URRST = 1, the receiver is active; if URRST = 0, the receiver is in reset.

In each case, putting the receiver and/or transmitter in reset will reset the state machine of the affected portion but does not affect the PRUSS_UART0 registers.

30.1.7.3.2.2 PRU-ICSS UART Hardware Reset Considerations

When the processor RESET pin is asserted, the entire processor is reset and is held in the reset state until the RESET pin is released. As part of a device reset, the PRUSS_UART0 state machine is reset and the PRUSS_UART0 registers are forced to their default states. The default states of the registers are shown in Section 30.2.7.4.2.2.

30.1.7.3.3 PRU-ICSS UART Power Management

The PRUSS_UART0 peripheral can be placed in reduced-power modes to conserve power during periods of low activity. The power management of the PRUSS_UART0 peripheral and other PRU-ICSS peripherals is controlled by the device Power, Reset and Clock Manager (PRCM). For more details on the PRUSS_UART0 clock and power management, refer to the Section 30.2.4.2, *PRU-ICSS Power and Clock Management*.

30.1.7.3.4 PRU-ICSS UART Interrupt Support

30.1.7.3.4.1 PRU-ICSS UART Interrupt Events and Requests

The PRUSS_UART0 generates the interrupt requests described in Table 30-749. All requests are multiplexed through an arbiter to a single PRUSS_UART0 interrupt request to the CPU, as shown in Figure 30-78. Each of the interrupt requests has an enable bit in the interrupt enable register (IER) - PRUSS_UART_INTERRUPT_ENABLE_REGISTER and is recorded in INTID bitfield of PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER.

If an interrupt occurs and the corresponding enable bit is set to 1, the interrupt request is recorded in INTID bitfield and is forwarded to the CPU. If an interrupt occurs and the corresponding enable bit is cleared to 0, the interrupt request is blocked. The interrupt request is neither recorded in INTID, nor forwarded to the CPU.

30.1.7.3.4.2 PRU-ICSS UART Interrupt Multiplexing

The PRUSS_UART0 have dedicated interrupt signals to the CPU and the interrupts are not multiplexed with any other interrupt source.

Table 30-279. PRU-ICSS UART Interrupt Requests Descriptions

PRUSS_UART0 Interrupt Request	Interrupt Source	Comment
THREINT	THR-empty condition: The transmitter holding register (THR) or the transmitter FIFO is empty. All of the data has been copied from THR (i.e. PRUSS_UART_RBR_THR_REGISTERS [7:0] DATA) to the transmitter shift register (TSR).	If THREINT is enabled in PRUSS_UART_INTERRUPT_ENABLE_REGISTER, by setting the ETBEI bit, it is recorded in INTID bitfield. As an alternative to using THREINT, the CPU can poll the THRE bit in the line status register PRUSS_UART_LINE_STATUS_REGISTER.

Table 30-279. PRU-ICSS UART Interrupt Requests Descriptions (continued)

PRUSS_UART0 Interrupt Request	Interrupt Source	Comment
RDAINT	Receive data available in non-FIFO mode or trigger level reached in the FIFO mode.	If RDAINT is enabled in PRUSS_UART_INTERRUPT_ENABLE_REGISTER, by setting the ERBI bit, it is recorded in INTID bitfield. As an alternative to using RDAINT, the CPU can poll the DR bit in the line status register PRUSS_UART_LINE_STATUS_REGISTER. In the FIFO mode, this is not a functionally equivalent alternative because the DR bit does not respond to the FIFO trigger level. The DR bit only indicates the presence or absence of unread characters.
RTOINT	Receiver time-out condition (in the FIFO mode only): No characters have been removed from or input to the receiver FIFO during the last four character times (see Table 30-751), and there is at least one character in the receiver FIFO during this time.	The receiver time-out interrupt prevents the PRUSS_UART0 from waiting indefinitely, in the case when the receiver FIFO level is below the trigger level and thus does not generate a receiver data-ready interrupt. If RTOINT is enabled in PRUSS_UART_INTERRUPT_ENABLE_REGISTER, by setting the ERBI bit, it is recorded in INTID bitfield. There is no status bit to reflect the occurrence of a time-out condition.
RLSINT	Receiver line status condition: An overrun error, parity error, framing error, or break has occurred.	If RLSINT is enabled in PRUSS_UART_INTERRUPT_ENABLE_REGISTER, by setting the ELSI bit, it is recorded in INTID bitfield. As an alternative to using RLSINT, the CPU can poll the following bits in the line status register PRUSS_UART_LINE_STATUS_REGISTER: overrun error indicator (OE), parity error indicator (PE), framing error indicator (FE), and break indicator (BI).

Figure 30-24. PRU-ICSS UART Interrupt Request Enable Paths

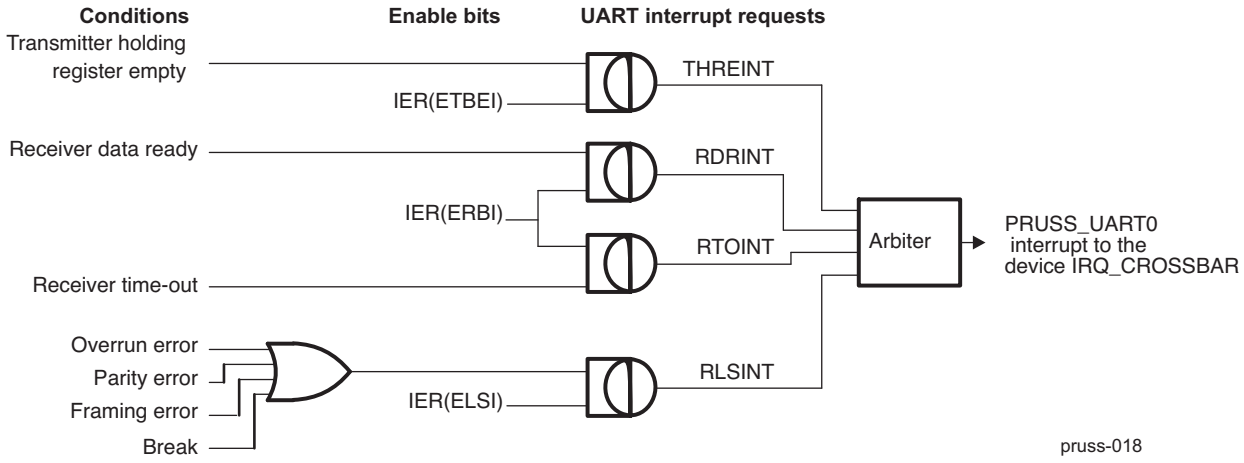


Table 30-280. Interrupt Identification and Interrupt Clearing Information

Priority Level	IIR Bits				Interrupt Type	Interrupt Source	Event That Clears Interrupt
	3	2	1	0			
None	0	0	0	1	None	None	None
1	0	1	1	0	Receiver line status	Overrun error, parity error, framing error, or break is detected.	For an overrun error, reading the PRUSS_UART_LINE_STATUS_REGISTER clears the interrupt. For a parity error, framing error, or break, the interrupt is cleared only after all the erroneous data have been read.
2	0	1	0	0	Receiver data-ready	Non-FIFO mode: Receiver data is ready.	Non-FIFO mode: The receiver buffer register (RBR) is read.
						FIFO mode: Trigger level reached. If four character times (see Table 30-751) pass with no access of the FIFO, the interrupt is asserted again.	FIFO mode: The FIFO drops below the trigger level. (1)
2	1	1	0	0	Receiver time-out	FIFO mode only: No characters have been removed from or input to the receiver FIFO during the last four character times (see Table 30-751), and there is at least one character in the receiver FIFO during this time.	One of the following events: • A character is read from the receiver FIFO (1) • A new character arrives in the receiver FIFO • The URRST bit in the power and emulation management register (PRUSS_UART_POWERMANAGEMENT_AND_EMULATION_REGISTER) is loaded with 0.
3	0	0	1	0	Transmitter holding register empty	Non-FIFO mode: Transmitter holding register (THR) is empty.	A character is written to the transmitter holding register (THR) or the interrupt identification register (IIR) is read.
						FIFO mode: Transmitter FIFO is empty.	

(1) In the FIFO mode, the receiver data-ready interrupt or receiver time-out interrupt is cleared by the CPU or by the DMA controller, whichever reads from the receiver FIFO first.

30.1.7.3.5 PRU-ICSS UART DMA Event Support

In the FIFO mode, the PRUSS_UART0 generates the following two DMA events:

- **Receive event (URXEVT):** The trigger level for the receiver FIFO (1, 4, 8, or 14 characters) is set with the FIFO control PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER [7:6] FIFOEN_RXFIFTL bitfield. Every time the trigger level is reached or a receiver time-out occurs, the PRUSS_UART0 sends a receive event to the EDMA controller. In response, the EDMA controller reads the data from the receiver FIFO by way of the receiver buffer register PRUSS_UART_RBR_THR_REGISTERS [7:0] DATA. Note that the receive event is not asserted if the data at the top of the receiver FIFO is erroneous even if the trigger level has been reached.
- **Transmit event (UTXEVT):** When the transmitter FIFO is empty (when the last byte in the transmitter FIFO has been copied to the transmitter shift register), the PRUSS_UART0 sends an UTXEVT signal to the EDMA controller. In response, the EDMA controller refills the transmitter FIFO by way of the transmitter holding register (THR) - PRUSS_UART_RBR_THR_REGISTERS [7:0] DATA. The UTXEVT signal is also sent to the DMA controller when the PRUSS_UART0 is taken out of reset using the UTRST bit in the power and emulation management register (PRUSS_UART_POWERMANAGEMENT_AND_EMULATION_REGISTER).

Activity in DMA channels can be synchronized to these events. In the non-FIFO mode, the PRUSS_UART0 generates no DMA events. Any DMA channel synchronized to either of these events must be enabled at the time the PRUSS_UART0 event is generated. Otherwise, the DMA channel will miss the event and, unless the PRUSS_UART0 generates a new event, no data transfer will occur.

30.1.7.3.6 PRU-ICSS UART Operations

30.1.7.3.6.1 PRU-ICSS UART Transmission

The PRUSS_UART0 transmitter section includes a transmitter hold register (THR) mapped in the PRUSS_UART_RBR_THR_REGISTERS [7:0] DATA bitfield and a transmitter shift register (TSR). When the PRUSS_UART0 is in the FIFO mode, THR is a 16-byte FIFO. Transmitter section control is a function of the PRUSS_UART0 line control register PRUSS_UART_LINE_CONTROL_REGISTER. Based on the settings chosen in this register, the PRUSS_UART0 transmitter sends the following to the receiving device:

- 1 START bit
- 5, 6, 7, or 8 data bits
- 1 PARITY bit (optional)
- 1, 1.5, or 2 STOP bits

THR receives data from the internal data bus, and when TSR is ready, the PRUSS_UART0 moves the data from THR to TSR. The PRUSS_UART0 serializes the data in TSR and transmits the data on the UART0_TXD pin.

In the non-FIFO mode, if THR is empty and the THR empty (THRE) interrupt is enabled in the interrupt enable register PRUSS_UART_INTERRUPT_ENABLE_REGISTER, an interrupt is generated. This interrupt is cleared when a character is loaded into THR or the interrupt identification register PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER bitfield INTID is read. In the FIFO mode, the interrupt is generated when the transmitter FIFO is empty, and it is cleared when at least one byte is loaded into the FIFO or INTID bitfield is read.

30.1.7.3.6.2 PRU-ICSS UART Reception

The PRUSS_UART0 receiver section includes a receiver shift register (RSR) and a receiver buffer register (RBR) mapped in PRUSS_UART_RBR_THR_REGISTERS [7:0] DATA bitfield. When the PRUSS_UART0 is in the FIFO mode, RBR is a 16-byte FIFO. Timing is supplied by the 16× receiver clock. Receiver section control is a function of the PRUSS_UART0 line control register PRUSS_UART_LINE_CONTROL_REGISTER. Based on the settings chosen in this register, the PRUSS_UART0 receiver accepts the following from the transmitting device:

- 1 START bit
- 5, 6, 7, or 8 data bits
- 1 PARITY bit (optional)
- 1 STOP bit (any other STOP bits transferred with the above data are not detected)

RSR receives the data bits from the UART0_RXD pin. Then RSR concatenates the data bits and moves the resulting value into RBR (or the receiver FIFO), accessible in the PRUSS_UART_RBR_THR_REGISTERS [7:0] DATA register bitfield. The PRUSS_UART0 also stores three bits of error status information next to each received character, to record a parity error, framing error, or break.

In the non-FIFO mode, when a character is placed in RBR and the receiver data-ready interrupt is enabled in the interrupt enable register - PRUSS_UART_INTERRUPT_ENABLE_REGISTER, an interrupt is generated. This interrupt is cleared when the character is read from RBR. In the FIFO mode, the interrupt is generated when the FIFO is filled to the trigger level selected in the FIFO control MSB part of the register PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER, and it is cleared when the FIFO contents drop below the trigger level.

30.1.7.3.6.3 PRU-ICSS UART FIFO Modes

The following two modes can be used for servicing the receiver and transmitter FIFOs:

- FIFO interrupt mode. The FIFO is enabled and the associated interrupts are enabled. Interrupts are sent to the CPU to indicate when specific events occur.
- FIFO poll mode. The FIFO is enabled but the associated interrupts are disabled. The CPU polls status bits to detect specific events.

Because the receiver FIFO and the transmitter FIFO are controlled separately, either one or both can be placed into the interrupt mode or the poll mode.

30.1.7.3.6.3.1 PRU-ICSS UART FIFO Interrupt Mode

When the receiver FIFO is enabled in the FIFO control register (FCR), mapped in the MSB part of the register PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER, and the receiver interrupts are enabled in the interrupt enable register PRUSS_UART_INTERRUPT_ENABLE_REGISTER, the interrupt mode is selected for the receiver FIFO. The following are important points about the receiver interrupts:

- The receiver data-ready interrupt is issued to the CPU when the FIFO has reached the trigger level that is programmed in FCR. It is cleared when the CPU or the DMA controller reads enough characters from the FIFO such that the FIFO drops below its programmed trigger level.
- The receiver line status interrupt is generated in response to an overrun error, a parity error, a framing error, or a break. This interrupt has higher priority than the receiver data-ready interrupt. For details, see Section 30.2.7.3.4.
- The data-ready (DR) bit in the line status register (LSR) - PRUSS_UART_LINE_STATUS_REGISTER, indicates the presence or absence of characters in the receiver FIFO. The DR bit is set when a character is transferred from the receiver shift register (RSR) to the empty receiver FIFO. The DR bit remains set until the FIFO is empty again.
- A receiver time-out interrupt occurs if all of the following conditions exist:
  - At least one character is in the FIFO,
  - The most recent character was received more than four continuous character times ago. A character time is the time allotted for 1 START bit, *n* data bits, 1 PARITY bit, and 1 STOP bit, where *n* depends on the word length selected with the WLS0 and WLS1 bits of the line control register PRUSS_UART_LINE_CONTROL_REGISTER. See Table 30-751.
  - The most recent read of the FIFO has occurred more than four continuous character times before.
- Character times are calculated by using the baud rate.
- When a receiver time-out interrupt has occurred, it is cleared and the time-out timer is cleared when the CPU or the EDMA controller reads one character from the receiver FIFO. The interrupt is also cleared if a new character is received in the FIFO or if the URRST bit is cleared in the power and emulation management register PRUSS_UART_POWERMANAGEMENT_AND_EMULATION_REGISTER.
- If a receiver time-out interrupt has not occurred, the time-out timer is cleared after a new character is received or after the CPU or EDMA reads the receiver FIFO.

When the transmitter FIFO is enabled in PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER [0] IPEND_FIFOEN bit and the transmitter holding register empty (THRE) interrupt is enabled in PRUSS_UART_INTERRUPT_ENABLE_REGISTER[1] ETBEI bit, the interrupt mode is selected for the transmitter FIFO. The THRE interrupt occurs when the transmitter FIFO is empty. It is cleared when the transmitter hold register (THR) PRUSS_UART_RBR_THR_REGISTERS [7:0] DATA bitfield is loaded (1 to 16 characters may be written to the transmitter FIFO while servicing this interrupt) or the interrupt identification register INTID bitfield is read in the PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER.

Table 30-281. Character Time for Word Lengths

Word Length (*n*)	Character Time	Four Character Times
5	Time for 8 bits	Time for 32 bits
6	Time for 9 bits	Time for 36 bits
7	Time for 10 bits	Time for 40 bits
8	Time for 11 bits	Time for 44 bits

30.1.7.3.6.3.2 PRU-ICSS UART FIFO Poll Mode

When the receiver FIFO is enabled in the FIFO control register (via setting the PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER [0] IPEND_FIFOEN to 0b1) and the receiver interrupts are disabled in the interrupt enable register (PRUSS_UART_INTERRUPT_ENABLE_REGISTER), the poll mode is selected for the receiver FIFO. Similarly, when the transmitter FIFO is enabled via setting the same bit (PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER [0] IPEND_FIFOEN to 0b1) and the transmitter interrupts are disabled, the transmitted FIFO is in the poll mode. In the poll mode, the CPU detects events by checking bits in the line status register - PRUSS_UART_LINE_STATUS_REGISTER:

- The PRUSS_UART_LINE_STATUS_REGISTER[7] RXFIFOE bit indicates whether there are any errors in the receiver FIFO.
- The PRUSS_UART_LINE_STATUS_REGISTER[6] TEMT bit indicates that both the transmitter holding register (THR) and the transmitter shift register (TSR) are empty.
- The PRUSS_UART_LINE_STATUS_REGISTER[5] THRE bit indicates when THR (mapped in the PRUSS_UART_RBR_THR_REGISTERS [7:0] DATA bitfield) is empty.
- The following PRUSS_UART_LINE_STATUS_REGISTER bits specify which error or errors have occurred:
  - PRUSS_UART_LINE_STATUS_REGISTER[4] BI - Break Interrupt
  - PRUSS_UART_LINE_STATUS_REGISTER[3] FE – Framing Error
  - PRUSS_UART_LINE_STATUS_REGISTER[2] PE – Parity Error
  - PRUSS_UART_LINE_STATUS_REGISTER[1] OE – Overrun Error
- The PRUSS_UART_LINE_STATUS_REGISTER[0] DR (data-ready) bit is set as long as there is at least one byte in the receiver FIFO.

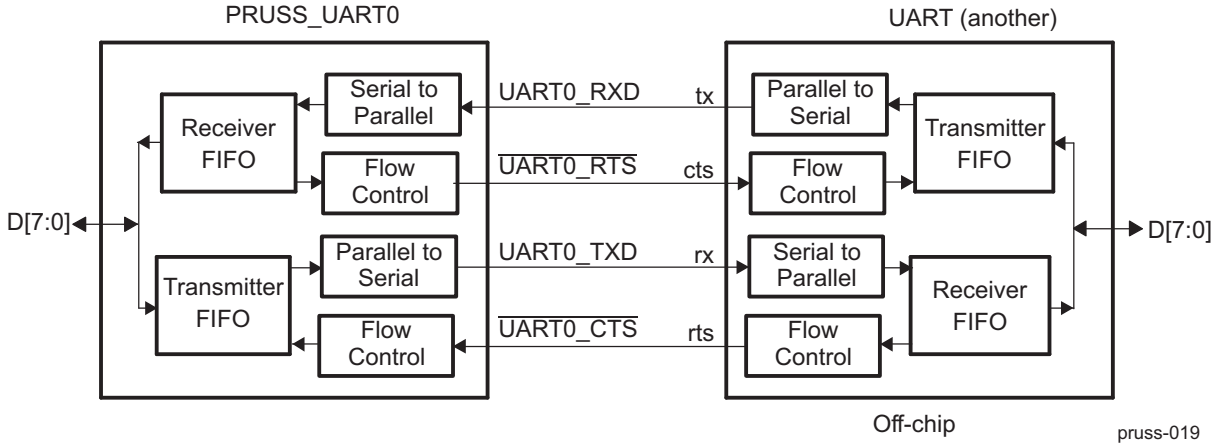
Also, in the FIFO poll mode:

- The interrupt identification register (INTID) bitfields are not affected by any events because the interrupts are disabled.
- The PRUSS_UART0 does not indicate when the receiver FIFO trigger level is reached or when a receiver time-out occurs.

30.1.7.3.6.4 PRU-ICSS UART Autoflow Control

The PRUSS_UART0 can employ autoflow control by connecting the $\overline{\text{PRUSS\_UART0\_CTS}}$ and $\overline{\text{PRUSS\_UART0\_RTS}}$ signals. The $\overline{\text{PRUSS\_UART0\_CTS}}$ input must be active before the transmitter FIFO can transmit data. The $\overline{\text{PRUSS\_UART0\_RTS}}$ becomes active when the receiver needs more data and notifies the sending device. When $\overline{\text{PRUSS\_UART0\_RTS}}$ is connected to $\overline{\text{PRUSS\_UART0\_CTS}}$, data transmission does not occur unless the receiver FIFO has space for the data. Therefore, when two UARTs are connected as shown in Figure 30-79 with autoflow enabled, overrun errors are eliminated.

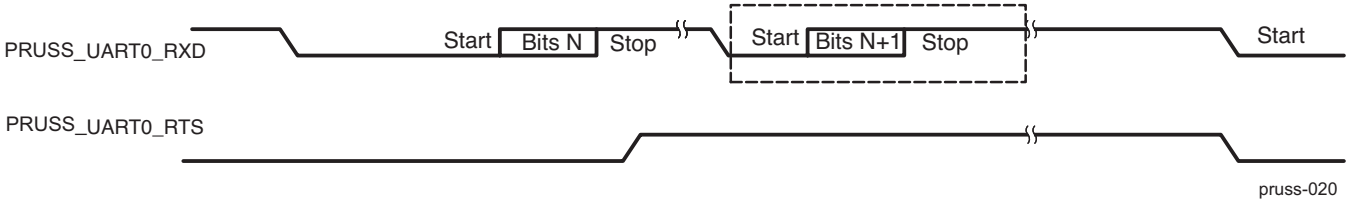
Figure 30-25. UART Interface Using Autoflow Diagram



30.1.7.3.6.4.1 PRU-ICSS UART Signal $\overline{\text{UART0\_RTS}}$ Behavior

$\overline{\text{PRUSS\_UART0\_RTS}}$ data flow control originates in the receiver block (see Figure 30-77). When the receiver FIFO level reaches a trigger level of 1, 4, 8, or 14 (see Figure 30-80), $\overline{\text{PRUSS\_UART0\_RTS}}$ is deasserted. The sending UART may send an additional byte after the trigger level is reached (assuming the sending UART has another byte to send), because it may not recognize the deassertion of $\overline{\text{PRUSS\_UART0\_RTS}}$ until after it has begun sending the additional byte. For trigger level 1, 4, and 8, $\overline{\text{PRUSS\_UART0\_RTS}}$ is automatically reasserted once the receiver FIFO is emptied. For trigger level 14, $\overline{\text{PRUSS\_UART0\_RTS}}$ is automatically reasserted once the receiver FIFO drops below the trigger level.

Figure 30-26. Autoflow Functional Timing Waveforms for $\overline{\text{PRUSS\_UART0\_RTS}}$



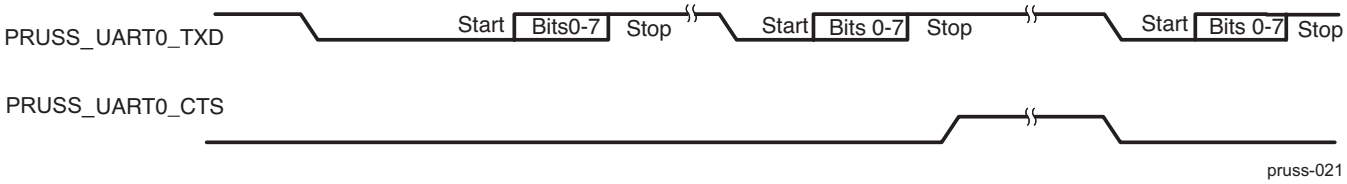
(1) N = Receiver FIFO trigger level.

(2) The two blocks in dashed lines cover the case where an additional byte is sent.

30.1.7.3.6.4.2 PRU-ICSS UART Signal $\overline{\text{PRUSS\_UART0\_CTS}}$ Behavior

The transmitter checks $\overline{\text{PRUSS\_UART0\_CTS}}$ before sending the next data byte. If $\overline{\text{PRUSS\_UART0\_CTS}}$ is active, the transmitter sends the next byte. To stop the transmitter from sending the following byte, $\overline{\text{PRUSS\_UART0\_CTS}}$ must be released before the middle of the last STOP bit that is currently being sent (see Figure 30-81). When flow control is enabled, $\overline{\text{PRUSS\_UART0\_CTS}}$ level changes do not trigger interrupts because the device automatically controls its own transmitter. Without autoflow control, the transmitter sends any data present in the transmitter FIFO and a receiver overrun error may result.

Figure 30-27. Autoflow Functional Timing Waveforms for $\overline{\text{PRUSS\_UART0\_CTS}}$



(1) When $\overline{\text{PRUSS\_UART0\_CTS}}$ is active (low), the transmitter keeps sending serial data out.

(2) When $\overline{\text{PRUSS\_UART0\_CTS}}$ goes high before the middle of the last STOP bit of the current byte, the transmitter finishes sending the current byte but it does not send the next byte.

(3) When $\overline{\text{PRUSS\_UART0\_CTS}}$ goes from high to low, the transmitter begins sending data again.

30.1.7.3.6.5 PRU-ICSS UART Loopback Control

The PRUSS_UART0 can be placed in the diagnostic mode using the LOOP bit in the modem control register - PRUSS_UART_MODEM_CONTROL_REGISTER, which internally connects the PRUSS_UART0 output back to the PRUSS_UART0's input. In this mode, the transmit and receive data paths, the transmitter and receiver interrupts, and the modem control interrupts can be verified without connecting to another UART.

30.1.7.3.7 PRU-ICSS UART Initialization

The following steps are required to initialize the PRUSS_UART0:

1. Perform the necessary device pin multiplexing setup (see your device-specific data manual).
2. Set the desired baud rate by writing the appropriate clock divisor values to the divisor latch registers PRUSS_UART_DIVISOR_REGISTER_MSB_ [7:0] DLH and PRUSS_UART_DIVISOR_REGISTER_LSB_ [7:0] DLL.
3. If the FIFOs will be used, select the desired trigger level and enable the FIFOs by writing the appropriate values to the FIFO control register. The PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER[0] IPEND_FIFOEN bit must be set first, before the other bits in this register are configured.
4. Choose the desired protocol settings by writing the appropriate values to the line control register PRUSS_UART_LINE_CONTROL_REGISTER.
5. If autoflow control is desired, write appropriate values to the modem control register PRUSS_UART_MODEM_CONTROL_REGISTER.
6. Choose the desired response to emulation suspend events by configuring the FREE bit and enable the PRUSS_UART0 by setting the UTRST and URRST bits in the power and emulation management register -PRUSS_UART_POWERMANAGEMENT_AND_EMULATION_REGISTER.

30.1.7.3.8 PRU-ICSS UART Exception Processing

30.1.7.3.8.1 PRU-ICSS UART Divisor Latch Not Programmed

Since the processor reset signal has no effect on the divisor latch, the divisor latch will have an unknown value after power up. If the divisor latch is not programmed after power up, the baud clock (BCLK) will not operate and will instead be set to a constant logic 1 state.

The divisor latch values should always be reinitialized following a processor reset.

30.1.7.3.8.2 Changing Operating Mode During Busy Serial Communication of PRU-ICSS UART

Since the serial link characteristics are based on how the control registers are programmed, the PRUSS_UART0 will expect the control registers to be static while it is busy engaging in a serial communication. Therefore, changing the control registers while the module is still busy communicating with another serial device will most likely cause an error condition and should be avoided.

30.1.7.4 PRUSS_UART Register Manual

This section describes the PRUSS_UART module registers.

30.1.7.4.1 PRUSS_UART Instance Summary

Table 30-282. PRUSS_UART Instance Summary

Module Name	Base Address	Size
PRUSS1_UART	0x4B22 8000	56 Bytes
PRUSS2_UART	0x4B2A 8000	56 Bytes

30.1.7.4.2 PRUSS_UART Registers

30.1.7.4.2.1 PRUSS_UART Register Summary

Table 30-283. PRUSS1_UART Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS1_UART Physical Address
PRUSS_UART_RBR_THR_REGISTERS	W	32	0x0000 0000	0x4B22 8000
PRUSS_UART_INTERRUPT_ENABLE_REGISTER	RW	32	0x0000 0004	0x4B22 8004
PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER	W	32	0x0000 0008	0x4B22 8008
PRUSS_UART_LINE_CONTROL_REGISTER	RW	32	0x0000 000C	0x4B22 800C
PRUSS_UART_MODEM_CONTROL_REGISTER	RW	32	0x0000 0010	0x4B22 8010
PRUSS_UART_LINE_STATUS_REGISTER	R	32	0x0000 0014	0x4B22 8014
PRUSS_UART_MODEM_STATUS_REGISTER	R	32	0x0000 0018	0x4B22 8018
PRUSS_UART_SCRATCH_REGISTER	RW	32	0x0000 001C	0x4B22 801C
PRUSS_UART_DIVISOR_REGISTER_LSB_	RW	32	0x0000 0020	0x4B22 8020
PRUSS_UART_DIVISOR_REGISTER_MSB_	RW	32	0x0000 0024	0x4B22 8024
PRUSS_UART_PERIPHERAL_ID_REGISTER	R	32	0x0000 0028	0x4B22 8028
RESERVED	R	32	0x0000 002C	0x4B22 802C
PRUSS_UART_POWERMANAGEMENT_AND_EMULATION_REGISTER	RW	32	0x0000 0030	0x4B22 8030
PRUSS_UART_MODE_DEFINITION_REGISTER	RW	32	0x0000 0034	0x4B22 8034

Table 30-284. PRUSS2_UART Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS2_UART Physical Address
PRUSS_UART_RBR_THR_REGISTERS	W	32	0x0000 0000	0x4B2A 8000
PRUSS_UART_INTERRUPT_ENABLE_REGISTER	RW	32	0x0000 0004	0x4B2A 8004
PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER	W	32	0x0000 0008	0x4B2A 8008
PRUSS_UART_LINE_CONTROL_REGISTER	RW	32	0x0000 000C	0x4B2A 800C
PRUSS_UART_MODEM_CONTROL_REGISTER	RW	32	0x0000 0010	0x4B2A 8010
PRUSS_UART_LINE_STATUS_REGISTER	R	32	0x0000 0014	0x4B2A 8014
PRUSS_UART_MODEM_STATUS_REGISTER	R	32	0x0000 0018	0x4B2A 8018
PRUSS_UART_SCRATCH_REGISTER	RW	32	0x0000 001C	0x4B2A 801C
PRUSS_UART_DIVISOR_REGISTER_LSB_	RW	32	0x0000 0020	0x4B2A 8020
PRUSS_UART_DIVISOR_REGISTER_MSB_	RW	32	0x0000 0024	0x4B2A 8024

Table 30-284. PRUSS2_UART Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS2_UART Physical Address
PRUSS_UART_PERIPHERAL_ID_REGISTER	R	32	0x0000 0028	0x4B2A 8028
RESERVED	R	32	0x0000 002C	0x4B2A 802C
PRUSS_UART_POWERMANAGEMENT_AND_EMULATION_REGISTER	RW	32	0x0000 0030	0x4B2A 8030
PRUSS_UART_MODE_DEFINITION_REGISTER	RW	32	0x0000 0034	0x4B2A 8034

30.1.7.4.2.2 PRUSS_UART Register Description

Table 30-285. PRUSS_UART_RBR_THR_REGISTERS

Address Offset	0x0000 0000		
Physical Address	0x4B22 8000 0x4B2A 8000	**Instance**	PRUSS1_UART PRUSS2_UART
Description	In the non-FIFO mode, when a character is placed in Receiver buffer register and the receiver data-ready interrupt is enabled (DR = 1 in Interrupt identification register), an interrupt is generated. This interrupt is cleared when the character is read from Receiver buffer register. In the FIFO mode, the interrupt is generated when the FIFO is filled to the trigger level selected in the FIFO control register, and it is cleared when the FIFO contents drop below the trigger level. In the non-FIFO mode, if Transmitter holding register is empty and the THR empty (THRE) interrupt is enabled (ETBEI = 1 in Interrupt enable register), an interrupt is generated. This interrupt is cleared when a character is loaded into Transmitter holding register or the Interrupt identification register is read. In the FIFO mode, the interrupt is generated when the transmitter FIFO is empty, and it is cleared when at least one byte is loaded into the FIFO or Interrupt identification register is read.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DATA							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x0
7:0	DATA	**Read:** Read Receive Buffer Register **Write:** Write Transmitter Holding Register	RW	0x0

Table 30-286. Register Call Summary for Register PRUSS_UART_RBR_THR_REGISTERS

PRU-ICSS UART Module

- PRU-ICSS UART Transmission Protocol: [0]
- PRU-ICSS UART Reception Protocol: [1]
- PRU-ICSS UART Interrupt Multiplexing: [2]
- PRU-ICSS UART DMA Event Support: [3] [4]
- PRU-ICSS UART Transmission: [5]
- PRU-ICSS UART Reception: [6] [7]
- PRU-ICSS UART FIFO Modes: [8] [9]
- PRUSS_UART Register Summary: [10] [11]

Table 30-287. PRUSS_UART_INTERRUPT_ENABLE_REGISTER

Address Offset	0x0000 0004		
Physical Address	0x4B22 8004 0x4B2A 8004	**Instance**	PRUSS1_UART PRUSS2_UART

Table 30-287. PRUSS_UART_INTERRUPT_ENABLE_REGISTER (continued)

Description	The Interrupt enable register is used to individually enable or disable each type of interrupt request that can be generated by the UART. Each interrupt request that is enabled in Interrupt enable register is forwarded to the CPU.
Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												EDSSI	ELSI	ETBEI	ERBI

Bits	Field Name	Description	Type	Reset
31:4	RESERVED	Reserved	R	0x0
3	EDSSI	Enable Modem Status Interrupt	RW	0x0
2	ELSI	Receiver line status interrupt enable. 0x0: Receiver line status interrupt is disabled. 0x1: Receiver line status interrupt is enabled.	RW	0x0
1	ETBEI	Transmitter holding register empty interrupt enable. 0x0: Transmitter holding register empty interrupt is disabled. 0x1: Transmitter holding register empty interrupt is enabled.	RW	0x0
0	ERBI	Receiver data available interrupt and character timeout indication interrupt enable. 0x0: Receiver data available interrupt and character timeout indication interrupt is disabled. 0x1: Receiver data available interrupt and character timeout indication interrupt is enabled.	RW	0x0

Table 30-288. Register Call Summary for Register PRUSS_UART_INTERRUPT_ENABLE_REGISTER

PRU-ICSS UART Module

- PRU-ICSS UART Interrupt Events and Requests: [0]
- PRU-ICSS UART Interrupt Multiplexing: [1] [2] [3] [4]
- PRU-ICSS UART Transmission: [5]
- PRU-ICSS UART Reception: [6]
- PRU-ICSS UART FIFO Modes: [7] [8] [9]
- PRUSS_UART Register Summary: [10] [11]

Table 30-289. PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER

Address Offset	0x0000 0008		
Physical Address	0x4B22 8008 0x4B2A 8008	**Instance**	PRUSS1_UART PRUSS2_UART
Description	The Interrupt identification register is a read-only register at the same address as the FIFO control register, which is a write-only register. When an interrupt is generated and enabled in the Interrupt enable register, Interrupt identification register indicates that an interrupt is pending in the IPEND bit and encodes the type of interrupt in the INTID bits. Reading Interrupt identification register clears any THR empty (THRE) interrupts that are pending. The FIFOEN bit in Interrupt identification register can be checked to determine whether the UART is in the FIFO mode or the non-FIFO mode. Use FIFO control register to enable and clear the FIFOs and to select the receiver FIFO trigger level. The FIFOEN bit in FIFO control register must be set to 1 before other FIFO control register bits are written to or the FIFO control register bits are not programmed.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								FIFOEN_RXFIFTL		RESERVED		INTID			IPEND_FIFOEN

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x0
7:6	FIFOEN_RXFIFTL	**Read:** FIFOs enabled. 0x0: Non-FIFO mode 0x1-0x2: Reserved 0x3: FIFOs are enabled. FIFOEN bit in the FIFO control register (FCR) is set to 1. **Write:** Receiver FIFO trigger level. RXFIFTL sets the trigger level for the receiver FIFO. When the trigger level is reached, a receiver data-ready interrupt is generated (if the interrupt request is enabled). Once the FIFO drops below the trigger level, the interrupt is cleared. 0x0: 1 byte 0x1: 4 bytes 0x2: 8 bytes 0x3: 14 bytes	RW	0x0
5:4	RESERVED	Reserved	R	0x0
3:1	INTID	**Read:** Interrupt type. See Table 30-750. 0x0: Reserved 0x1: Transmitter holding register empty (priority 3) 0x2: Receiver data available (priority 2) 0x3: Receiver line status (priority 1, highest) 0x4-0x5: Reserved 0x6: Character timeout indication (priority 2) 0x7: Reserved **Write:** **Bit 3: DMAMODE1:** DMA MODE1 enable if FIFOs are enabled. Always write 1 to DMAMODE1. After a hardware reset, change DMAMODE1 from 0 to 1. DMAMODE1 = 1 is a requirement for proper communication between the UART and the EDMA controller. 0x0: DMA MODE1 is disabled. 0x1: DMA MODE1 is enabled. **Bit 2: TXCLR:** Transmitter FIFO clear. Write a 1 to TXCLR to clear the bit. 0x0: No effect. 0x1: Clears transmitter FIFO and resets the transmitter FIFO counter. The shift register is not cleared. **Bit 1: RXCLR:** Receiver FIFO clear. Write a 1 to RXCLR to clear the bit. 0x0: No effect. 0x1: Clears receiver FIFO and resets the receiver FIFO counter. The shift register is not cleared.	RW	0x0

Bits	Field Name	Description	Type	Reset
0	IPEND_FIFOEN	**Read:** Interrupt pending. When any UART interrupt is generated and is enabled in IER, IPEND is forced to 0. IPEND remains 0 until all pending interrupts are cleared or until a hardware reset occurs. If no interrupts are enabled, IPEND is never forced to 0. 0x0: Interrupts pending. 0x1: No interrupts pending. **Write:** Transmitter and receiver FIFOs mode enable. FIFOEN must be set before other FCR bits are written to or the FCR bits are not programmed. Clearing this bit clears the FIFO counters. 0x0: Non-FIFO mode. The transmitter and receiver FIFOs are disabled, and the FIFO pointers are cleared. 0x1: FIFO mode. The transmitter and receiver FIFOs are enabled.	RW	0x1

Table 30-290. Register Call Summary for Register PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER

PRU-ICSS UART Module

- PRU-ICSS UART Interrupt Events and Requests: [0]
- PRU-ICSS UART Interrupt Multiplexing:
- PRU-ICSS UART DMA Event Support: [2]
- PRU-ICSS UART Transmission: [3]
- PRU-ICSS UART Reception: [4]
- PRU-ICSS UART FIFO Modes: [5] [6] [7] [8] [9]
- PRU-ICSS UART Initialization: [10]
- PRUSS_UART Register Summary: [11] [12]

Table 30-291. PRUSS_UART_LINE_CONTROL_REGISTER

Address Offset	0x0000 000C		
Physical Address	0x4B22 800C 0x4B2A 800C	**Instance**	PRUSS1_UART PRUSS2_UART
Description	The system programmer controls the format of the asynchronous data communication exchange by using Line control register. In addition, the programmer can retrieve, inspect, and modify the content of line control register; this eliminates the need for separate storage of the line characteristics in system memory.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DLAB	BC	SP	EPS	PEN	STB	WLS1	WLS0

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x0
7	DLAB	Divisor latch access bit. The divisor latch registers (DLL and DLH) can be accessed at dedicated addresses or at addresses shared by RBR, THR, and IER. Using the shared addresses requires toggling DLAB to change which registers are selected. If you use the dedicated addresses, you can keep DLAB = 0. 0x0: Allows access to the receiver buffer register (RBR), the transmitter holding register (THR), and the interrupt enable register (IER) selected. At the address shared by RBR, THR, and DLL, the CPU can read from RBR and write to THR. At the address shared by IER and DLH, the CPU can read from and write to IER. 0x1: Allows access to the divisor latches of the baud generator during a read or write operation (DLL and DLH). At the address shared by RBR, THR, and DLL, the CPU can read from and write to DLL. At the address shared by IER and DLH, the CPU can read from and write to DLH.	RW	0x0
6	BC	Break control. 0x0: Break condition is disabled. 0x1: Break condition is transmitted to the receiving UART. A break condition is a condition where the UARTn_TXD signal is forced to the spacing (cleared) state.	RW	0x0
5	SP	Stick parity. The SP bit works in conjunction with the EPS and PEN bits. The relationship between the SP, EPS, and PEN bits is summarized in Table 30-745. 0x0: Stick parity is disabled. 0x1: Stick parity is enabled. • When odd parity is selected (EPS = 0), the PARITY bit is transmitted and checked as set. • When even parity is selected (EPS = 1), the PARITY bit is transmitted and checked as cleared.	RW	0x0
4	EPS	Even parity select. Selects the parity when parity is enabled (PEN = 1). The EPS bit works in conjunction with the SP and PEN bits. The relationship between the SP, EPS, and PEN bits is summarized in Table 30-745. 0x0: Odd parity is selected (an odd number of logic 1s is transmitted or checked in the data and PARITY bits). 0x1: Even parity is selected (an even number of logic 1s is transmitted or checked in the data and PARITY bits).	RW	0x0
3	PEN	Parity enable. The PEN bit works in conjunction with the SP and EPS bits. The relationship between the SP, EPS, and PEN bits is summarized in Table 30-745. 0x0: No PARITY bit is transmitted or checked. 0x1: Parity bit is generated in transmitted data and is checked in received data between the last data word bit and the first STOP bit.	RW	0x0
2	STB	Number of STOP bits generated. STB specifies 1, 1.5, or 2 STOP bits in each transmitted character. When STB = 1, the WLS bit determines the number of STOP bits. The receiver clocks only the first STOP bit, regardless of the number of STOP bits selected. The number of STOP bits generated is summarized in Table 30-746. 0x0: 1 STOP bit is generated. 0x1: WLS bit determines the number of STOP bits: • When WLS = 0, 1.5 STOP bits are generated. • When WLS = 1h, 2h, or 3h, 2 STOP bits are generated.	RW	0x0

Bits	Field Name	Description	Type	Reset
1-0	WLS	Word length select. Number of bits in each transmitted or received serial character. When STB = 1, the WLS bit determines the number of STOP bits. 0x0: 5 bits 0x1: 6 bits 0x2: 7 bits 0x3: 8 bits	RW	0x0

Table 30-292. Register Call Summary for Register PRUSS_UART_LINE_CONTROL_REGISTER

PRU-ICSS UART Module

- PRU-ICSS UART Transmission Protocol: [0]
- PRU-ICSS UART Reception Protocol: [1]
- PRU-ICSS UART Data Format: [2] [3] [4]
- PRU-ICSS UART Transmission: [5]
- PRU-ICSS UART Reception: [6]
- PRU-ICSS UART FIFO Modes: [7]
- PRU-ICSS UART Initialization: [8]
- PRUSS_UART Register Summary: [9] [10]

Table 30-293. PRUSS_UART_MODEM_CONTROL_REGISTER

Address Offset	0x0000 0010		
Physical Address	0x4B22 8010 0x4B2A 8010	**Instance**	PRUSS1_UART PRUSS2_UART
Description	The Modem control register provides the ability to enable/disable the autoflow functions, and enable/disable the loopback function for diagnostic purposes.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																										AFE	LOOP	OUT2	OUT1	RTS	RESERVED

Bits	Field Name	Description	Type	Reset
31:6	RESERVED	Reserved	R	0x0
5	AFE	Autoflow control enable. Autoflow control allows the $\overline{\text{UARTn\_RTS}}$ and $\overline{\text{UARTn\_CTS}}$ signals to provide handshaking between UARTs during data transfer. When AFE = 1, the RTS bit determines the autoflow control enabled. Note that all UARTs do not support this feature, see your device-specific data manual for supported features. If this feature is not available, this bit is reserved in this device and should be cleared to 0. 0x0: Autoflow control is disabled. 0x1:Autoflow control is enabled: • When RTS = 0, $\overline{\text{UARTn\_CTS}}$ is only enabled. • When RTS = 1, $\overline{\text{UARTn\_RTS}}$ and $\overline{\text{UARTn\_CTS}}$ are enabled.	RW	0x0

Bits	Field Name	Description	Type	Reset
4	LOOP	Loop back mode enable. LOOP is used for the diagnostic testing using the loop back feature. 0x0: Loop back mode is disabled. 0x1: Loop back mode is enabled. When LOOP is set, the following occur: • The UARTn_TXD signal is set high. • The UARTn_RXD pin is disconnected. • The output of the transmitter shift register (TSR) is lopped back in to the receiver shift register (RSR) input.	RW	0x0
3	OUT2	OUT2 Control Bit	RW	0x0
2	OUT1	OUT1 Control Bit	RW	0x0
1	RTS	RTS control. When AFE = 1, the RTS bit determines the autoflow control enabled. Note that all UARTs do not support this feature, see your device-specific data manual for supported features. If this feature is not available, this bit is reserved in this device and should be cleared to 0. 0x0: $\overline{\text{UARTn\_RTS}}$ is disabled, $\overline{\text{UARTn\_CTS}}$ is only enabled. 0x1: $\overline{\text{UARTn\_RTS}}$ and $\overline{\text{UARTn\_CTS}}$ are enabled.	RW	0x0
0	RESERVED	Reserved	R	0

Table 30-294. Register Call Summary for Register PRUSS_UART_MODEM_CONTROL_REGISTER

PRU-ICSS UART Module

- PRU-ICSS UART Loopback Control: [0]
- PRU-ICSS UART Initialization: [1]
- PRUSS_UART Register Summary: [2] [3]

Table 30-295. PRUSS_UART_LINE_STATUS_REGISTER

Address Offset	0x0000 0014		
Physical Address	0x4B22 8014 0x4B2A 8014	**Instance**	PRUSS1_UART PRUSS2_UART
Description	The Line status register provides information to the CPU concerning the status of data transfers. Line status register is intended for read operations only; do not write to this register. Bits 1 through 4 record the error conditions that produce a receiver line status interrupt.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								RXFIFOE	TEMT	THRE	BI	FE	PE	OE	DR

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x0
7	RXFIFOE	Receiver FIFO error. **In non-FIFO mode:** 0x0: There has been no error, or RXFIFOE was cleared because the CPU read the erroneous character from the receiver buffer register (RBR). 0x1: There is a parity error, framing error, or break indicator in the receiver buffer register (RBR). **In FIFO mode:** 0x0: There has been no error, or RXFIFOE was cleared because the CPU read the erroneous character from the receiver FIFO and there are no more errors in the receiver FIFO. 0x1: At least one parity error, framing error, or break indicator in the receiver FIFO.	R	0x0
6	TEMT	Transmitter empty (TEMT) indicator. **In non-FIFO mode:** 0x0: Either the transmitter holding register (THR) or the transmitter shift register (TSR) contains a data character. 0x1: Both the transmitter holding register (THR) and the transmitter shift register (TSR) are empty. **In FIFO mode:** 0x0: Either the transmitter FIFO or the transmitter shift register (TSR) contains a data character. 0x1: Both the transmitter FIFO and the transmitter shift register (TSR) are empty.	R	0x1
5	THRE	Transmitter holding register empty (THRE) indicator. If the THRE bit is set and the corresponding interrupt enable bit is set (ETBEI = 1 in IER), an interrupt request is generated. **In non-FIFO mode:** 0x0: Transmitter holding register (THR) is not empty. THR has been loaded by the CPU. 0x1: Transmitter holding register (THR) is empty (ready to accept a new character). The content of THR has been transferred to the transmitter shift register (TSR). **In FIFO mode:** 0x0: Transmitter FIFO is not empty. At least one character has been written to the transmitter FIFO. You can write to the transmitter FIFO if it is not full. 0x1: Transmitter FIFO is empty. The last character in the FIFO has been transferred to the transmitter shift register (TSR).	R	0x1

Bits	Field Name	Description	Type	Reset
4	BI	Break indicator. The BI bit is set whenever the receive data input (UARTn_RXD) was held low for longer than a full-word transmission time. A full-word transmission time is defined as the total time to transmit the START, data, PARITY, and STOP bits. If the BI bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated. **In non-FIFO mode:** 0x0: No break has been detected, or the BI bit was cleared because the CPU read the erroneous character from the receiver buffer register (RBR). 0x1: A break has been detected with the character in the receiver buffer register (RBR). **In FIFO mode:** 0x0: No break has been detected, or the BI bit was cleared because the CPU read the erroneous character from the receiver FIFO and the next character to be read from the FIFO has no break indicator. 0x1: A break has been detected with the character at the top of the receiver FIFO.	R	0x0
3	FE	Framing error (FE) indicator. A framing error occurs when the received character does not have a valid STOP bit. In response to a framing error, the UART sets the FE bit and waits until the signal on the RX pin goes high. Once the RX signal goes high, the receiver is ready to detect a new START bit and receive new data. If the FE bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated. **In non-FIFO mode:** 0x0: No framing error has been detected, or the FE bit was cleared because the CPU read the erroneous data from the receiver buffer register (RBR). 0x1: A framing error has been detected with the character in the receiver buffer register (RBR). **In FIFO mode:** 0x0: No framing error has been detected, or the FE bit was cleared because the CPU read the erroneous data from the receiver FIFO and the next character to be read from the FIFO has no framing error. 0x1: A framing error has been detected with the character at the top of the receiver FIFO.	R	0x0
2	PE	Parity error (PE) indicator. A parity error occurs when the parity of the received character does not match the parity selected with the EPS bit in the line control register (LCR). If the PE bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated. **In non-FIFO mode:** 0x0: No parity error has been detected, or the PE bit was cleared because the CPU read the erroneous data from the receiver buffer register (RBR). 0x1: A parity error has been detected with the character in the receiver buffer register (RBR). **In FIFO mode:** 0x0: No parity error has been detected, or the PE bit was cleared because the CPU read the erroneous data from the receiver FIFO and the next character to be read from the FIFO has no parity error. 0x1: A parity error has been detected with the character at the top of the receiver FIFO.	R	0x0

Bits	Field Name	Description	Type	Reset
1	OE	Overrun error (OE) indicator. An overrun error in the non-FIFO mode is different from an overrun error in the FIFO mode. If the OE bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated. **In non-FIFO mode:** 0x0: No overrun error has been detected, or the OE bit was cleared because the CPU read the content of the line status register (LSR). 0x1: Overrun error has been detected. Before the character in the receiver buffer register (RBR) could be read, it was overwritten by the next character arriving in RBR. **In FIFO mode:** 0x0: No overrun error has been detected, or the OE bit was cleared because the CPU read the content of the line status register (LSR). 0x1: Overrun error has been detected. If data continues to fill the FIFO beyond the trigger level, an overrun error occurs only after the FIFO is full and the next character has been completely received in the shift register. An overrun error is indicated to the CPU as soon as it happens. The new character overwrites the character in the shift register, but it is not transferred to the FIFO.	R	0x0
0	DR	Data-ready (DR) indicator for the receiver. If the DR bit is set and the corresponding interrupt enable bit is set (ERBI = 1 in IER), an interrupt request is generated. **In non-FIFO mode:** 0x0: Data is not ready, or the DR bit was cleared because the character was read from the receiver buffer register (RBR). 0x1: Data is ready. A complete incoming character has been received and transferred into the receiver buffer register (RBR). **In FIFO mode:** 0x0: Data is not ready, or the DR bit was cleared because all of the characters in the receiver FIFO have been read. 0x1: Data is ready. There is at least one unread character in the receiver FIFO. If the FIFO is empty, the DR bit is set as soon as a complete incoming character has been received and transferred into the FIFO. The DR bit remains set until the FIFO is empty again.	R	0x0

Table 30-296. Register Call Summary for Register PRUSS_UART_LINE_STATUS_REGISTER

PRU-ICSS UART Module

- PRU-ICSS UART Interrupt Multiplexing: [0] [1] [2] [3]
- PRU-ICSS UART FIFO Modes: [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14]
- PRUSS_UART Register Summary: [15] [16]

Table 30-297. PRUSS_UART_MODEM_STATUS_REGISTER

Address Offset	0x0000 0018		
Physical Address	0x4B22 8018 0x4B2A 8018	**Instance**	PRUSS1_UART PRUSS2_UART
Description	The Modem status register provides information to the CPU concerning the status of modem control signals. Modem status register is intended for read operations only; do not write to this register.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CD	RI	DSR	CTS	DCD	TERI	DDSR	DCTS

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x0
7	CD	Complement of the Carrier Detect input. When the UART is in the diagnostic test mode (loopback mode MCR[4] = 1), this bit is equal to the MCR bit 3 (OUT2).	R	0x0
6	RI	Complement of the Ring Indicator input. When the UART is in the diagnostic test mode (loopback mode MCR[4] = 1), this bit is equal to the MCR bit 2 (OUT1).	R	0x0
5	DSR	Complement of the Data Set Ready input. When the UART is in the diagnostic test mode (loopback mode MCR[4] = 1), this bit is equal to the MCR bit 0 (DTR).	R	0x0
4	CTS	Complement of the Clear To Send input. When the UART is in the diagnostic test mode (loopback mode MCR[4] = 1), this bit is equal to the MCR bit 1 (RTS).	R	0x0
3	DCD	Change in DCD indicator bit. DCD indicates that the DCD input has changed state since the last time it was read by the CPU. When DCD is set and the modem status interrupt is enabled, a modem status interrupt is generated.	R	0x0
2	TERI	Trailing edge of RI (TERI) indicator bit. TERI indicates that the RI input has changed from a low to a high. When TERI is set and the modem status interrupt is enabled, a modem status interrupt is generated.	R	0x0
1	DDSR	Change in DSR indicator bit. DDSR indicates that the DSR input has changed state since the last time it was read by the CPU. When DDSR is set and the modem status interrupt is enabled, a modem status interrupt is generated.	R	0x0
0	DCTS	Change in CTS indicator bit. DCTS indicates that the CTS input has changed state since the last time it was read by the CPU. When DCTS is set (autoflow control is not enabled and the modem status interrupt is enabled), a modem status interrupt is generated. When autoflow control is enabled, no interrupt is generated.	R	0x0

Table 30-298. Register Call Summary for Register PRUSS_UART_MODEM_STATUS_REGISTER

PRU-ICSS UART Module

- PRUSS_UART Register Summary: [0] [1]

Table 30-299. PRUSS_UART_SCRATCH_REGISTER

Address Offset	0x0000 001C		
Physical Address	0x4B22 801C 0x4B2A 801C	**Instance**	PRUSS1_UART PRUSS2_UART
Description	The Scratch Pad register is intended for programmer's use as a scratch pad. It temporarily holds the programmer's data without affecting UART operation.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								SCR							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x0
7:0	SCR	These bits are intended for the programmer's use as a scratch pad in the sense that it temporarily holds the programmer's data without affecting any other UART operation.	R	0x0

Table 30-300. Register Call Summary for Register PRUSS_UART_SCRATCH_REGISTER

PRU-ICSS UART Module

- PRUSS_UART Register Summary: [0] [1]

Table 30-301. PRUSS_UART_DIVISOR_REGISTER_LSB_

Address Offset	0x0000 0020		
Physical Address	0x4B22 8020 0x4B2A 8020	**Instance**	PRUSS1_UART PRUSS2_UART
Description	Two 8-bit register fields (DLL and DLH), called divisor latches, store the 16-bit divisor for generation of the baud clock in the baud generator. DLH holds the most-significant bits of the divisor, and DLL holds the least-significant bits of the divisor. These divisor latches must be loaded during initialization of the UART in order to ensure desired operation of the baud generator. Writing to the divisor latches results in two wait states being inserted during the write access while the baud generator is loaded with the new value.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DLL							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x0
7:0	DLL	The 8 least-significant bits (LSBs) of the 16-bit divisor for generation of the baud clock in the baud rate generator.	RW	0x0

Table 30-302. Register Call Summary for Register PRUSS_UART_DIVISOR_REGISTER_LSB_

PRU-ICSS UART Module

- PRU-ICSS UART Clock Generation and Control: [0]
- PRU-ICSS UART Initialization: [1]
- PRUSS_UART Register Summary: [2] [3]

Table 30-303. PRUSS_UART_DIVISOR_REGISTER_MSB_

Address Offset	0x0000 0024		
Physical Address	0x4B22 8024 0x4B2A 8024	**Instance**	PRUSS1_UART PRUSS2_UART
Description	Two 8-bit register fields (DLL and DLH), called divisor latches, store the 16-bit divisor for generation of the baud clock in the baud generator. DLH holds the most-significant bits of the divisor, and DLL holds the least-significant bits of the divisor. These divisor latches must be loaded during initialization of the UART in order to ensure desired operation of the baud generator. Writing to the divisor latches results in two wait states being inserted during the write access while the baud generator is loaded with the new value.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DLH							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x0
7:0	DLH	The 8 most-significant bits (MSBs) of the 16-bit divisor for generation of the baud clock in the baud rate generator.	RW	0x0

Table 30-304. Register Call Summary for Register PRUSS_UART_DIVISOR_REGISTER_MSB_

PRU-ICSS UART Module

- PRU-ICSS UART Clock Generation and Control: [0]
- PRU-ICSS UART Initialization: [1]
- PRUSS_UART Register Summary: [2] [3]

Table 30-305. PRUSS_UART_PERIPHERAL_ID_REGISTER

Address Offset	0x0000 0028		
Physical Address	0x4B22 8028 0x4B2A 8028	**Instance**	PRUSS1_UART PRUSS2_UART
Description	Peripheral Identification register		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PID																															

Bits	Field Name	Description	Type	Reset
31:0	PID		R	0x44141102

Table 30-306. Register Call Summary for Register PRUSS_UART_PERIPHERAL_ID_REGISTER

PRU-ICSS UART Module

- PRUSS_UART Register Summary: [0] [1]

Table 30-307. PRUSS_UART_POWERMANAGEMENT_AND_EMULATION_REGISTER

Address Offset	0x0000 0030		
Physical Address	0x4B22 8030 0x4B2A 8030	**Instance**	PRUSS1_UART PRUSS2_UART
Description	Power and emulation management register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																RESERVED	UTRST	URRST	RESERVED											FREE	

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Reserved	R	0x0000
15	RESERVED	Reserved. This bit must always be written with a 0.	RW	0x0
14	UTRST	UART transmitter reset. Resets and enables the transmitter. 0x0: Transmitter is disabled and in reset state. 0x1: Transmitter is enabled.	RW	0x0

Bits	Field Name	Description	Type	Reset
13	URRST	UART receiver reset. Resets and enables the receiver. 0x0: Receiver is disabled and in reset state. 0x1: Receiver is enabled.	RW	0x0
12:1	RESERVED	Reserved	R	0x000
0	FREE	Free-running enable mode bit. This bit determines the emulation mode functionality of the UART. When halted, the UART can handle register read/write requests, but does not generate any transmission/reception, interrupts or events. 0x0: If a transmission is not in progress, the UART halts immediately. If a transmission is in progress, the UART halts after completion of the one-word transmission. 0x1: Free-running mode is enabled; UART continues to run normally.	RW	0x0

Table 30-308. Register Call Summary for Register PRUSS_UART_POWERMANAGEMENT_AND_EMULATION_REGISTER

PRU-ICSS UART Module

- PRU-ICSS UART Software Reset Considerations: [0]
- PRU-ICSS UART Interrupt Multiplexing: [1]
- PRU-ICSS UART DMA Event Support: [2]
- PRU-ICSS UART FIFO Modes: [3]
- PRU-ICSS UART Initialization: [4]
- PRUSS_UART Register Summary: [5] [6]

Table 30-309. PRUSS_UART_MODE_DEFINITION_REGISTER

Address Offset	0x0000 0034		
Physical Address	0x4B22 8034 0x4B2A 8034	**Instance**	PRUSS1_UART PRUSS2_UART
Description	The Mode definition register determines the over-sampling mode for the UART.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															OSM_SEL

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved	R	0x0
0	OSM_SEL	Over-Sampling Mode Select. 0x0: 16× over-sampling. 0x1: 13× over-sampling.	RW	0x0

Table 30-310. Register Call Summary for Register PRUSS_UART_MODE_DEFINITION_REGISTER

PRU-ICSS UART Module

- PRU-ICSS UART Clock Generation and Control: [0]
- PRUSS_UART Register Summary: [1] [2]

30.1.8 PRU-ICSS eCAP Module

30.1.8.1 PRU-ICSS eCAP Functional Description

A single instance of an **enhanced capture** event module is integrated in the device PRU-ICSS1 subsystem - PRUSS1_eCAP_0 and PRU-ICSS2 subsystem - PRUSS2_eCAP_0.

For more details on the PRUSS1_eCAP_0 and PRUSS2_eCAP_0 I/O signals available at device level, refer to the Section 30.1.2. For PRUSS1_eCAP_0 and PRUSS2_eCAP_0 integration details and functionalities controlled at PRU-ICSS top level (functional clock control, etc.), refer to the Section 30.1.3 and the Section 30.1.4 .

NOTE: The PRUSS1_eCAP_0 and PRUSS2_eCAP_0 "SYNCIn" hardware event synchronization input and "SYNCOut" hardware synchronization output are NOT implemented in the device PRU-ICSS1 and PRU-ICSS2, respectively. However, a software-forced synchronization via bit PRUSS_ECAP_ECCTL2[8] SWSYNC, can be used as an alternative, provided that PRUSS_ECAP_ECCTL2[5] SYNCI_EN bit is set to 0b1.

For full description of the PRUSS1_eCAP_0 and PRUSS2_eCAP_0 modules functionalities, refer to the Section 29.3, *Enhanced Capture (eCAP) Module* of the Chapter 29, *Pulse-Width Modulation Subsystem*.

30.1.8.2 PRUSS_ECAP Register Manual

This section describes the registers of the PRUSS_eCAP_0 module.

30.1.8.2.1 PRUSS_ECAP Instance Summary

Table 30-311. PRUSS_ECAP Instance Summary

Module Name	Module Base Address L3_MAIN	Size
PRUSS1_ECAP	0x4B23 0000	96 Bytes
PRUSS2_ECAP	0x4B2B 0000	96 Bytes

30.1.8.2.2 PRUSS_ECAP Registers

30.1.8.2.2.1 PRUSS_ECAP Register Summary

Table 30-312. PRUSS1_ECAP Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS1_ECAP Physical Address
PRUSS_ECAP_TSCNT	RW	32	0x0000 0000	0x4B23 0000
PRUSS_ECAP_CNTPHS	RW	32	0x0000 0004	0x4B23 0004
PRUSS_ECAP_CAP1	RW	32	0x0000 0008	0x4B23 0008
PRUSS_ECAP_CAP2	RW	32	0x0000 000C	0x4B23 000C
PRUSS_ECAP_CAP3	RW	32	0x0000 0010	0x4B23 0010
PRUSS_ECAP_CAP4	RW	32	0x0000 0014	0x4B23 0014
PRUSS_ECAP_ECCTL1	RW	16	0x0000 0028	0x4B23 0028
PRUSS_ECAP_ECCTL2	RW	16	0x0000 002A	0x4B23 002A
PRUSS_ECAP_ECEINT	RW	16	0x0000 002C	0x4B23 002C
PRUSS_ECAP_ECFLG	R	16	0x0000 002E	0x4B23 002E
PRUSS_ECAP_ECCLR	RW	16	0x0000 0030	0x4B23 0030
PRUSS_ECAP_ECFRC	RW	16	0x0000 0034	0x4B23 0034
PRUSS_ECAP_PID	R	32	0x0000 005C	0x4B23 005C

Table 30-313. PRUSS2_ECAP Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS2_ECAP Physical Address
PRUSS_ECAP_TSCNT	RW	32	0x0000 0000	0x4B2B 0000
PRUSS_ECAP_CNTPHS	RW	32	0x0000 0004	0x4B2B 0004
PRUSS_ECAP_CAP1	RW	32	0x0000 0008	0x4B2B 0008
PRUSS_ECAP_CAP2	RW	32	0x0000 000C	0x4B2B 000C
PRUSS_ECAP_CAP3	RW	32	0x0000 0010	0x4B2B 0010
PRUSS_ECAP_CAP4	RW	32	0x0000 0014	0x4B2B 0014
PRUSS_ECAP_ECCTL1	RW	16	0x0000 0028	0x4B2B 0028
PRUSS_ECAP_ECCTL2	RW	16	0x0000 002A	0x4B2B 002A
PRUSS_ECAP_ECEINT	RW	16	0x0000 002C	0x4B2B 002C
PRUSS_ECAP_ECFLG	R	16	0x0000 002E	0x4B2B 002E
PRUSS_ECAP_ECCLR	RW	16	0x0000 0030	0x4B2B 0030
PRUSS_ECAP_ECFRC	RW	16	0x0000 0034	0x4B2B 0034
PRUSS_ECAP_PID	R	32	0x0000 005C	0x4B2B 005C

30.1.8.2.2.2 PRUSS_ECAP Register Description

Table 30-314. PRUSS_ECAP_TSCNT

Address Offset	0x0000 0000		
Physical Address	0x4B23 0000 0x4B2B 0000	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	Time Stamp Counter Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSCNT																															

Bits	Field Name	Description	Type	Reset
31:0	TSCNT	Active 32 bit-counter register that is used as the capture time-base	RW	0x0

Table 30-315. Register Call Summary for Register PRUSS_ECAP_TSCNT

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [1] [2]
- PRUSS_ECAP Register Description: [3]

Table 30-316. PRUSS_ECAP_CNTPHS

Address Offset	0x0000 0004		
Physical Address	0x4B23 0004 0x4B2B 0004	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	Counter Phase Control Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTPHS																															

Bits	Field Name	Description	Type	Reset
31:0	CNTPHS	Counter phase value register that can be programmed for phase lag/lead. This register shadows TSCNT and is loaded into PRUSS_ECAP_TSCNT upon either a SYNCI event or S/W force via a control bit. Used to achieve phase control synchronization with respect to other eCAP and EPWM time-bases.	RW	0x0

Table 30-317. Register Call Summary for Register PRUSS_ECAP_CNTPHS

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [1] [2]
- PRUSS_ECAP Register Description: [3]

Table 30-318. PRUSS_ECAP_CAP1

Address Offset	0x0000 0008		
Physical Address	0x4B23 0008 0x4B2B 0008	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	Capture-1 Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAP1																															

Bits	Field Name	Description	Type	Reset
31:0	CAP1	This register can be loaded (written) by the following. (a) Time-Stamp (that is, counter value) during a capture event. (b) Software may be useful for test purposes. (c) APRD active register when used in APWM mode.	RW	0x0

Table 30-319. Register Call Summary for Register PRUSS_ECAP_CAP1

PRU-ICSS eCAP Module

- PRU-ICSS eCAP Functional Description:
- PRUSS_ECAP Register Summary: [17] [18]
- PRUSS_ECAP Register Description: [19] [20] [21] [22] [23] [24] [25]

Table 30-320. PRUSS_ECAP_CAP2

Address Offset	0x0000 000C		
Physical Address	0x4B23 000C 0x4B2B 000C	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	Capture-2 Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAP2																															

Bits	Field Name	Description	Type	Reset
31:0	CAP2	This register can be loaded (written) by the following. (a) Time- Stamp (that is, counter value) during a capture event. (b) Software may be useful for test purposes. (c) APRD active register when used in APWM mode.	RW	0x0

Table 30-321. Register Call Summary for Register PRUSS_ECAP_CAP2

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [11] [12]
- PRUSS_ECAP Register Description: [13] [14]

Table 30-322. PRUSS_ECAP_CAP3

Address Offset	0x0000 0010		
Physical Address	0x4B23 0010 0x4B2B 0010	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	Capture-3 Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAP3																															

Bits	Field Name	Description	Type	Reset
31:0	CAP3	In CMP mode, this is a time-stamp capture register. In APWM mode, this is the period shadow (APRD) register. User software updates the PWM period value through this register. In this mode, CAP3 shadows CAP1.	RW	0x0

Table 30-323. Register Call Summary for Register PRUSS_ECAP_CAP3

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [10] [11]

Table 30-324. PRUSS_ECAP_CAP4

Address Offset	0x0000 0014		
Physical Address	0x4B23 0014 0x4B2B 0014	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	Capture-4 Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAP4																															

Bits	Field Name	Description	Type	Reset
31:0	CAP4	In CMP mode, this is a time-stamp capture register. In APWM mode, this is the compare shadow (ACMP) register. User software updates the PWM compare value through this register. In this mode, CAP4 shadows CAP2.	RW	0x0

Table 30-325. Register Call Summary for Register PRUSS_ECAP_CAP4

PRU-ICSS eCAP Module

- PRU-ICSS eCAP Functional Description:
- PRUSS_ECAP Register Summary: [13] [14]
- PRUSS_ECAP Register Description: [15] [16] [17] [18] [19]

Table 30-326. PRUSS_ECAP_ECCTL1

Address Offset	0x0000 0028		
Physical Address	0x4B23 0028 0x4B2B 0028	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	ECAP Control Register1		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FREE_SOFT		EVTFLTPS					CAPLDEN	CTRRST4	CAP4POL	CTRRST3	CAP3POL	CTRRST2	CAP2POL	CTRRST1	CAP1POL

Bits	Field Name	Description	Type	Reset
15:14	FREE_SOFT	Emulation Control 0x0 = TSCNT counter stops immediately on emulation suspend. 0x1 = TSCNT counter runs until = 0. 0x2 = TSCNT counter is unaffected by emulation suspend (Run Free). 0x3 = TSCNT counter is unaffected by emulation suspend (Run Free).	RW	0x0
13:9	EVTFLTPS	Event Filter prescale select: 0x0 = Divide by 1 (i.e,. no prescale, by-pass the prescaler) 0x1 = Divide by 2 0x2 = Divide by 4 0x3 = Divide by 6 0x4 = Divide by 8 0x5 = Divide by 10 0x1E = Divide by 60 0x1F = Divide by 62	RW	0x0
8	CAPLDEN	Enable Loading of PRUSS_ECAP_CAP1 to PRUSS_ECAP_CAP4 registers on a capture event 0x0 = Disable PRUSS_ECAP_CAP1-PRUSS_ECAP_CAP4 register loads at capture event time. 0x1 = Enable PRUSS_ECAP_CAP1-PRUSS_ECAP_CAP4 register loads at capture event time.	RW	0x0
7	CTRRST4	Counter Reset on Capture Event 4 0x0 = Do not reset counter on Capture Event 4 (absolute time stamp operation) 0x1 = Reset counter after Capture Event 4 time-stamp has been captured (used in difference mode operation)	RW	0x0
6	CAP4POL	Capture Event 4 Polarity select 0x0 = Capture Event 4 triggered on a rising edge (RE) 0x1 = Capture Event 4 triggered on a falling edge (FE)	RW	0x0
5	CTRRST3	Counter Reset on Capture Event 3 0x0 = Do not reset counter on Capture Event 3 (absolute time stamp) 0x1 = Reset counter after Event 3 time-stamp has been captured (used in difference mode operation)	RW	0x0
4	CAP3POL	Capture Event 3 Polarity select 0x0 = Capture Event 3 triggered on a rising edge (RE) 0x1 = Capture Event 3 triggered on a falling edge (FE)	RW	0x0
3	CTRRST2	Counter Reset on Capture Event 2 0x0 = Do not reset counter on Capture Event 2 (absolute time stamp) 0x1 = Reset counter after Event 2 time-stamp has been captured (used in difference mode operation)	RW	0x0

Bits	Field Name	Description	Type	Reset
2	CAP2POL	Capture Event 2 Polarity select 0x0 = Capture Event 2 triggered on a rising edge (RE) 0x1 = Capture Event 2 triggered on a falling edge (FE)	RW	0x0
1	CTRRST1	Counter Reset on Capture Event 1 0x0 = Do not reset counter on Capture Event 1 (absolute time stamp) 0x1 = Reset counter after Event 1 time-stamp has been captured (used in difference mode operation)	RW	0x0
0	CAP1POL	Capture Event 1 Polarity select 0x0 = Capture Event 1 triggered on a rising edge (RE) 0x1 = Capture Event 1 triggered on a falling edge (FE)	RW	0x0

Table 30-327. Register Call Summary for Register PRUSS_ECAP_ECCTL1

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [2] [3]

Table 30-328. PRUSS_ECAP_ECCTL2

Address Offset	0x0000 002A		
Physical Address	0x4B23 002A 0x4B2B 002A	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	ECAP Control Register 2		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					APWMPOL	CAPAPWM	SWSYNC	SYNCO_SEL		SYNCI_EN	TSCNTSTP	REARMRESET	STOPVALUE		CONTONESHT

Bits	Field Name	Description	Type	Reset
15:11	RESERVED		R	0x0
10	APWMPOL	APWM output polarity select. This is applicable only in APWM operating mode 0x0 = Output is active high (Compare value defines high time) 0x1 = Output is active low (Compare value defines low time)	RW	0x0

Bits	Field Name	Description	Type	Reset
9	CAPAPWM	CAP/APWM operating mode select 0x0 = ECAP module operates in capture mode. This mode forces the following configuration. (a) Inhibits TSCNT resets via CTR = PRD event. (b) Inhibits shadow loads on PRUSS_ECAP_CAP1 and PRUSS_ECAP_CAP2 registers. (c) Permits user to enable PRUSS_ECAP_CAP1-PRUSS_ECAP_CAP4 register load. (d) ECAP input/APWM output pin operates as a capture input. 0x1 = ECAP module operates in APWM mode. This mode forces the following configuration. (a) Resets TSCNT on CTR = PRD event (period boundary). (b) Permits shadow loading on PRUSS_ECAP_CAP1 and PRUSS_ECAP_CAP2 registers. (c) Disables loading of time-stamps into PRUSS_ECAP_CAP1 - PRUSS_ECAP_CAP4 registers. (d) ECAP input/APWM ouput pin operates as a APWM output.	RW	0x0
8	SWSYNC	Software-forced Counter (TSCNT) Synchronizing. This provides a convenient software method to synchronize some or all ECAP time bases. In APWM mode, the synchronizing can also be done via the CTR = PRD event. Note: Selection CTR = PRD is meaningful only in APWM mode. However, you can choose it in CAP mode if you find doing so useful. 0x0 = Writing a zero has no effect. Reading always returns a zero 0x1 = Writing a one forces a TSCNT shadow load of current ECAP module and any ECAP modules down-stream providing the SYNCO_SEL bits are 1'b00. After writing a 1, this bit returns to a zero.	RW	0x0
7:6	SYNCO_SEL	Sync-Out Select 0x0 = Select sync-in event to be the sync-out signal (pass through) 0x1 = Select CTR = PRD event to be the sync-out signal 0x2 = Disable sync out signal 0x3 = Disable sync out signal	RW	0x0
5	SYNCI_EN	Counter (TSCNT) Sync-In select mode 0x0 = Disable sync-in option 0x1 = Enable counter (TSCNT) to be loaded from PRUSS_ECAP_CNTPHS register upon either a SYNCI signal or a S/W force event.	RW	0x0
4	TSCNTSTP	Time Stamp (TSCNT) Counter Stop (freeze) Control 0x0 = TSCNT stopped 0x1 = TSCNT free-running	RW	0x0
3	REARMRESET	One-Shot Re-Arming Control, that is, wait for stop trigger. Note: The re-arm function is valid in one shot or continuous mode. 0x0 = Has no effect (reading always returns a 0) 0x1 = Arms the one-shot sequence as follows: 1) Resets the Mod4 counter to zero. 2) Unfreezes the Mod4 counter. 3) Enables capture register loads.	RW	0x0

Bits	Field Name	Description	Type	Reset
2:1	STOPVALUE	Stop value for one-shot mode. This is the number (between 1 and 4) of captures allowed to occur before the CAP (1 through 4) registers are frozen, that is, capture sequence is stopped. Wrap value for continuous mode. This is the number (between 1 and 4) of the capture register in which the circular buffer wraps around and starts again. Notes: STOPVALUE is compared to Mod4 counter and, when equal, the following two actions occur. (1) Mod4 counter is stopped (frozen). (2) Capture register loads are inhibited. In one-shot mode, further interrupt events are blocked until re-armed. 0x0 = Stop after Capture Event 1 in one-shot mode. Wrap after Capture Event 1 in continuous mode. 0x1 = Stop after Capture Event 2 in one-shot mode. Wrap after Capture Event 2 in continuous mode. 0x2 = Stop after Capture Event 3 in one-shot mode. Wrap after Capture Event 3 in continuous mode. 0x3 = Stop after Capture Event 4 in one-shot mode. Wrap after Capture Event 4 in continuous mode.	RW	0x3
0	CONTONESHT	Continuous or one-shot mode control (applicable only in capture mode) 0x0 = Operate in continuous mode 0x1 = Operate in one-shot mode	RW	0x0

Table 30-329. Register Call Summary for Register PRUSS_ECAP_ECCTL2

PRU-ICSS eCAP Module

- PRU-ICSS eCAP Functional Description: [0] [1]
- PRUSS_ECAP Register Summary: [7] [8]

Table 30-330. PRUSS_ECAP_ECEINT

Address Offset	0x0000 002C		
Physical Address	0x4B23 002C 0x4B2B 002C	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	ECAP Interrupt Enable Register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CMPEQ	PRDEQ	CNTOVF	CEVT4	CEVT3	CEVT2	CEVT1	RESERVED

Bits	Field Name	Description	Type	Reset
15:8	RESERVED		R	0x0
7	CMPEQ	Counter Equal **Compare** Interrupt Enable. 0x0 = Disable Compare Equal as an Interrupt source. 0x1 = Enable Compare Equal as an Interrupt source.	RW	0x0
6	PRDEQ	Counter Equal **Period** Interrupt Enable. 0x0 = Disable Period Equal as an Interrupt source. 0x1 = Enable Period Equal as an Interrupt source.	RW	0x0
5	CNTOVF	Counter Overflow Interrupt Enable. 0x0 = Disable counter Overflow as an Interrupt source. 0x1 = Enable counter Overflow as an Interrupt source.	RW	0x0

Bits	Field Name	Description	Type	Reset
4	CEVT4	Capture Event 4 Interrupt Enable. 0x0 = Disable Capture Event 4 as an Interrupt source. 0x1 = Enable Capture Event 4 as an Interrupt source.	RW	0x0
3	CEVT3	Capture Event 3 Interrupt Enable. 0x0 = Disable Capture Event 3 as an Interrupt source. 0x1 = Enable Capture Event 3 as an Interrupt source.	RW	0x0
2	CEVT2	Capture Event 2 Interrupt Enable. 0x0 = Disable Capture Event 2 as an Interrupt source. 0x1 = Enable Capture Event 2 as an Interrupt source.	RW	0x0
1	CEVT1	Capture Event 1 Interrupt Enable . 0x0 = Disable Capture Event 1 as an Interrupt source. 0x1 = Enable Capture Event 1 as an Interrupt source.	RW	0x0
0	RESERVED		R	0x0

Table 30-331. Register Call Summary for Register PRUSS_ECAP_ECEINT

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [2] [3]

Table 30-332. PRUSS_ECAP_ECFLG

Address Offset	0x0000 002E		
Physical Address	0x4B23 002E 0x4B2B 002E	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	ECAP Interrupt Flag Register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CMPEQ	PRDEQ	CNTOVF	CEVT4	CEVT3	CEVT2	CEVT1	INT

Bits	Field Name	Description	Type	Reset
15:8	RESERVED		R	0x0
7	CMPEQ	Compare Equal Compare Status Flag. This flag is only active in APWM mode. 0x0 = Indicates no event occurred 0x1 = Indicates the counter (TSCNT) reached the compare register value (ACMP)	R	0x0
6	PRDEQ	Counter Equal Period Status Flag. This flag is only active in APWM mode. 0x0 = Indicates no event occurred 0x1 = Indicates the counter (TSCNT) reached the period register value (APRD) and was reset.	R	0x0
5	CNTOVF	Counter Overflow Status Flag. This flag is active in CAP and APWM mode. 0x0 = Indicates no event occurred. 0x1 = Indicates the counter (TSCNT) has made the transition from 0xFFFFFFFF to 0x00000000	R	0x0
4	CEVT4	Capture Event 4 Status Flag This flag is only active in CAP mode. 0x0 = Indicates no event occurred 0x1 = Indicates the fourth event occurred at ECAPn pin	R	0x0

Bits	Field Name	Description	Type	Reset
3	CEVT3	Capture Event 3 Status Flag. This flag is active only in CAP mode. 0x0 = Indicates no event occurred. 0x1 = Indicates the third event occurred at ECAPn pin.	R	0x0
2	CEVT2	Capture Event 2 Status Flag. This flag is only active in CAP mode. 0x0 = Indicates no event occurred. 0x1 = Indicates the second event occurred at ECAPn pin.	R	0x0
1	CEVT1	Capture Event 1 Status Flag. This flag is only active in CAP mode. 0x0 = Indicates no event occurred. 0x1 = Indicates the first event occurred at ECAPn pin.	R	0x0
0	INT	Global Interrupt Status Flag 0x0 = Indicates no interrupt generated. 0x1 = Indicates that an interrupt was generated.	R	0x0

Table 30-333. Register Call Summary for Register PRUSS_ECAP_ECFLG

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [2] [3]

Table 30-334. PRUSS_ECAP_ECCLR

Address Offset	0x0000 0030		
Physical Address	0x4B23 0030 0x4B2B 0030	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	ECAP Interrupt Clear Register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CMPEQ	PRDEQ	CNTOVF	CEVT4	CEVT3	CEVT2	CEVT1	INT

Bits	Field Name	Description	Type	Reset
15:8	RESERVED		R	0x0
7	CMPEQ	Counter Equal Compare Status Flag 0x0 = Writing a 0 has no effect. Always reads back a 0 0x1 = Writing a 1 clears the CTR=CMP flag condition	RW	0x0
6	PRDEQ	Counter Equal Period Status Flag 0x0 = Writing a 0 has no effect. Always reads back a 0 0x1 = Writing a 1 clears the CTR=PRD flag condition	RW	0x0
5	CNTOVF	Counter Overflow Status Flag 0x0 = Writing a 0 has no effect. Always reads back a 0 0x1 = Writing a 1 clears the CNTOVF flag condition	RW	0x0
4	CEVT4	Capture Event 4 Status Flag 0x0 = Writing a 0 has no effect. Always reads back a 0. 0x1 = Writing a 1 clears the CEVT3 flag condition.	RW	0x0
3	CEVT3	Capture Event 3 Status Flag 0x0 = Writing a 0 has no effect. Always reads back a 0. 0x1 = Writing a 1 clears the CEVT3 flag condition.	RW	0x0

Bits	Field Name	Description	Type	Reset
2	CEVT2	Capture Event 2 Status Flag 0x0 = Writing a 0 has no effect. Always reads back a 0. 0x1 = Writing a 1 clears the CEVT2 flag condition.	RW	0x0
1	CEVT1	Capture Event 1 Status Flag 0x0 = Writing a 0 has no effect. Always reads back a 0. 0x1 = Writing a 1 clears the CEVT1 flag condition.	RW	0x0
0	INT	Global Interrupt Clear Flag 0x0 = Writing a 0 has no effect. Always reads back a 0. 0x1 = Writing a 1 clears the INT flag and enable further interrupts to be generated if any of the event flags are set to 1.	RW	0x0

Table 30-335. Register Call Summary for Register PRUSS_ECAP_ECCLR

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [2] [3]

Table 30-336. PRUSS_ECAP_ECFRC

Address Offset	0x0000 0034		
Physical Address	0x4B23 0034 0x4B2B 0034	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	ECAP Interrupt Forcing Register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CMPEQ	PRDEQ	CNTOVF	CEVT4	CEVT3	CEVT2	CEVT1	RESERVED

Bits	Field Name	Description	Type	Reset
15:8	RESERVED		R	0x0
7	CMPEQ	Force Counter Equal Compare Interrupt 0x0 = No effect. Always reads back a 0. 0x1 = Writing a 1 sets the CTR=CMP flag bit.	RW	0x0
6	PRDEQ	Force Counter Equal Period Interrupt 0x0 = No effect. Always reads back a 0. 0x1 = Writing a 1 sets the CTR=PRD flag bit.	RW	0x0
5	CNTOVF	Force Counter Overflow 0x0 = No effect. Always reads back a 0. 0x1 = Writing a 1 to this bit sets the CNTOVF flag bit.	RW	0x0
4	CEVT4	Force Capture Event 4 0x0 = No effect. Always reads back a 0. 0x1 = Writing a 1 sets the CEVT4 flag bit	RW	0x0
3	CEVT3	Force Capture Event 3 0x0 = No effect. Always reads back a 0. 0x1 = Writing a 1 sets the CEVT3 flag bit	RW	0x0
2	CEVT2	Force Capture Event 2 0x0 = No effect. Always reads back a 0. 0x1 = Writing a 1 sets the CEVT2 flag bit.	RW	0x0

Bits	Field Name	Description	Type	Reset
1	CEVT1	Always reads back a 0. Force Capture Event 1 0x0 = No effect. 0x1 = Writing a 1 sets the CEVT1 flag bit.	RW	0x0
0	RESERVED		R	0x0

Table 30-337. Register Call Summary for Register PRUSS_ECAP_ECFRC

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [2] [3]

Table 30-338. PRUSS_ECAP_PID

Address Offset	0x0000 005C		
Physical Address	0x4B23 005C 0x4B2B 005C	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	ECAP Revision ID		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REVISION																															

Bits	Field Name	Description	Type	Reset
31:0	REVISION	IP Revision	R	0x-(1)

(1) TI Internal information

Table 30-339. Register Call Summary for Register PRUSS_ECAP_PID

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [1] [2]

30.1.9 PRU-ICSS MII RT Module

30.1.9.1 Introduction

The Real-time Media Independent Interface (MII_RT) provides a programmable I/O interface for the PRUs to access and control up to two MII ports. The MII_RT module can also be configured to push and pull data independent of the PRU cores.

NOTE: In order to guarantee the MII_RT IO timing values published in the device data manual, the PRUSS_GICLK clock must be configured for 200MHz (default value) and TX_CLK_DELAY bitfield in the PRUSS_MII_RT_TXCFG0/1 must be set to 6h (non-default value).

30.1.9.1.1 Features

The PRU-ICSS MII_RT module supports:

- Two MII ports
  - Each MII port has:
    - 32-byte RX L1 FIFO
    - 64-byte RX L2 buffer
    - 96-byte TX L1 FIFO
  - Rate decoupling on TX L1 FIFO
  - Configurable pre-amble removal on RX L1 FIFO and insertion on TX L1 FIFO
  - Configurable TX L1 FIFO trigger (10 bits with 40 ns ticks)
- MII port multiplexer per direction to support line/ring structure
  - Link detection through RX_ERR
- Cyclic redundancy check (CRC)
  - CRC32 generation on TX path
  - CRC32 checker on RX path

30.1.9.1.2 Unsupported Features

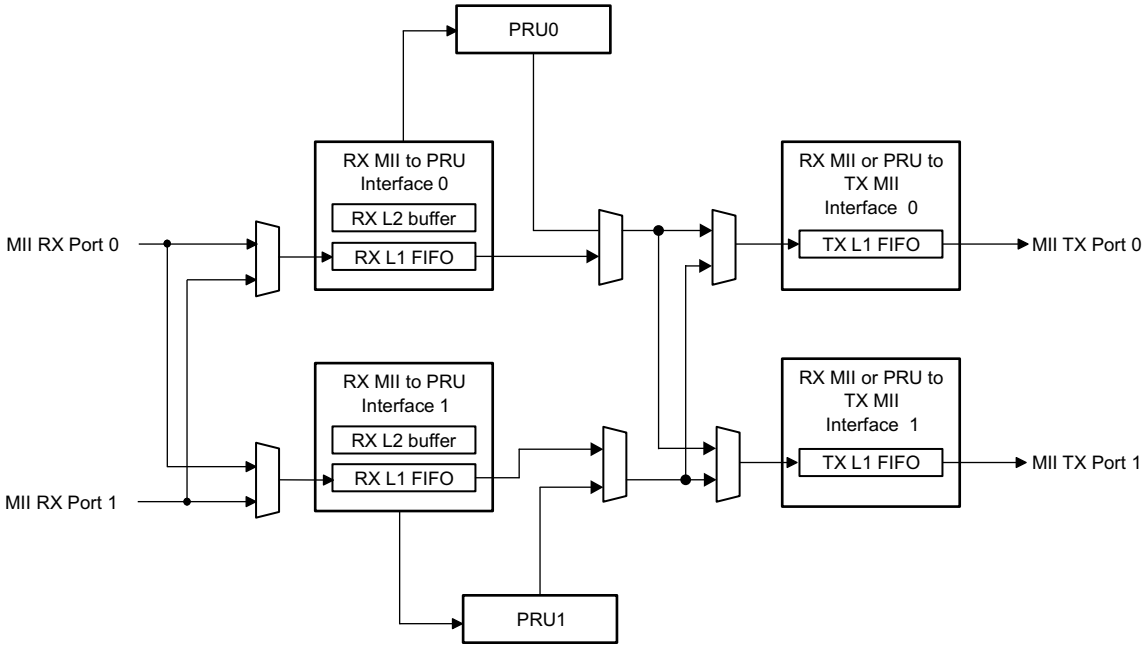
The PRU-ICSS MII_RT module does not support:

- Auto padding in TX L1 FIFO
- Dynamic TX multiplexer switching during packet handling
  - Can allow one PRU to handle both MII interfaces and a second PRU to manage the host and switch functions.

30.1.9.1.3 Block Diagram

Figure 30-28 shows the MII_RT in context of the PRU-ICSS. This diagram is a conceptual block diagram and does not necessarily reflect actual topologies.

Figure 30-28. MII_RT Block Diagram



30.1.9.2 Functional Description

30.1.9.2.1 Data Path Configuration

The MII_RT module supports three basic data path configurations. These configurations are compared in Table 30-340 and described in the following sections.

Table 30-340. Data Path Configuration Comparison

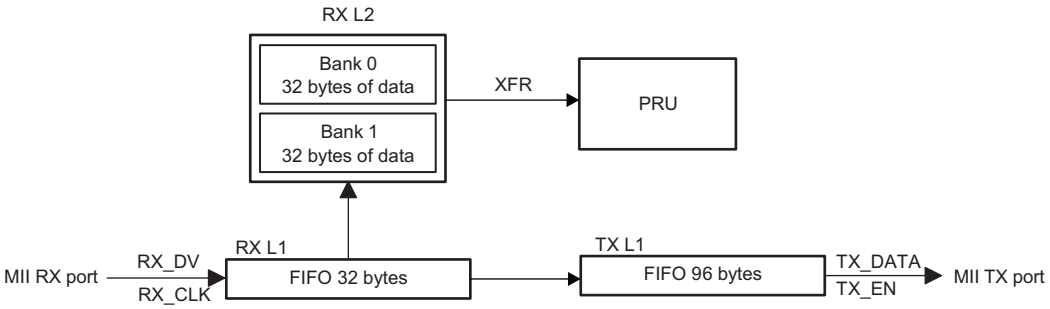
Configuration	PRU Dependency	Data Servicing	Port-to-Port Latency
Auto-forward	Snoop only	One word in flight	Low
8- or 16-bit processing with on-the-fly modifications (RX L1)	Yes	One word or byte in flight	Low
32-byte double buffer or ping-pong processing (RX L2)	Yes	Multi-words in flight	Medium (application-dependent)

30.1.9.2.1.1 Auto-forward with Optional PRU Snoop

Data is automatically forwarded from the MII RX port to the MII TX port without manipulations, as shown in Figure 30-29. This configuration does not depend on the PRU core. However, it does support an option for PRU to snoop or monitor the received data through the RX L2, shown in Figure 30-30. The PRU does not access data and status bits through R31, and it does not modify and push data.

Figure 30-29. Auto-forward

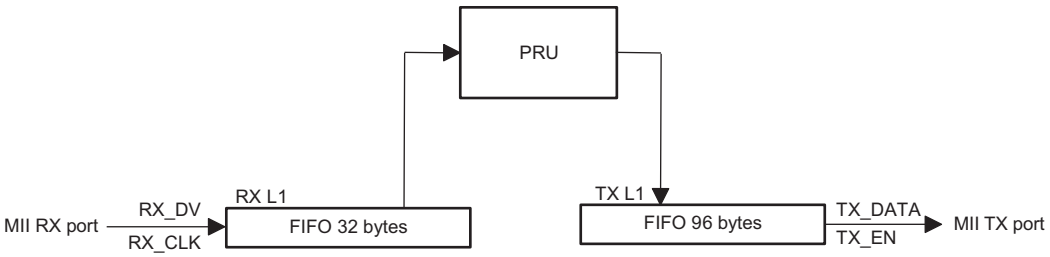
Figure 30-30. Auto-forward with PRU Snoop



30.1.9.2.1.2 8- or 16-bit Processing with On-the-Fly Modifications

This configuration services one byte or word in flight and has low latency. The PRU has the option to manipulate the received word and control popping data from the RX L1 FIFO and pushing it on the TX L1 FIFO.

Figure 30-31. 8- or 16-bit Processing with On-the-Fly Modifications

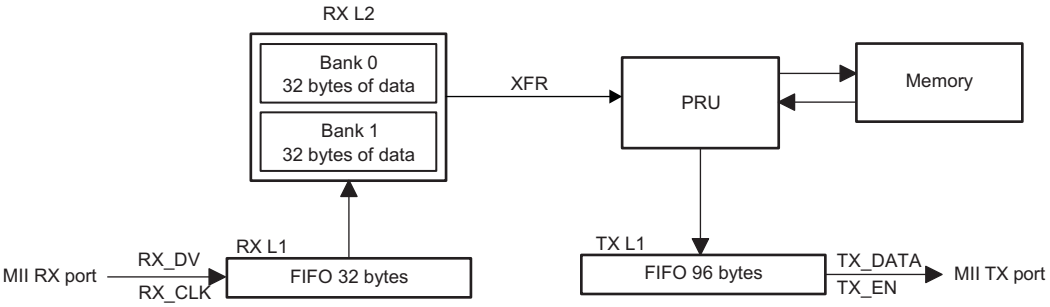


30.1.9.2.1.3 32-byte Double Buffer or Ping-Pong Processing

This configuration supports high bandwidth, high efficiency transactions. Often implementations using this mode permit relaxed servicing requirements allowing the PRU to manipulate the received data before transmitting.

Data received in this configuration is passed into the RX L2 buffer. The PRU reads multiple bytes of data from one of the RX L2 banks through the high bandwidth broadside interface and XFR instructions. The PRU can then store or manipulate data before pushing it to the TX L1 FIFO for transmission on the MII TX port.

Figure 30-32. 32-byte Double Buffer or Ping-Pong Processing



30.1.9.2.2 Definition and Terms

30.1.9.2.2.1 Data Frame Structure

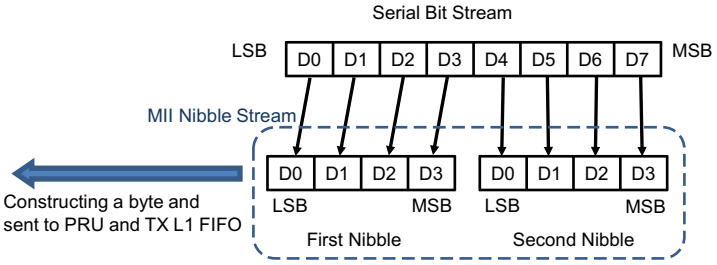
The data received and transmitted over MII conforms with the frame structure shown in Table 30-341.

Table 30-341. Frame Structure

Inter-frame	Preamble	Start of Frame Delimiter (SFD)	Data	Cyclic Redundancy Check (CRC)

The data following the SFD is formatted in a 4-bit nibble structure. Figure 30-33 illustrates the nibble order. The MSB arriving first is on the LSB side of a nibble. When receiving data, the MII_RT receive logic will wait for the next nibble to arrive before constructing a byte and delivering to the PRU.

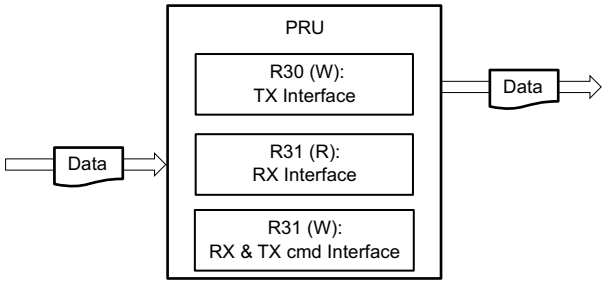
Figure 30-33. Data Nibble Structure



30.1.9.2.2.2 PRU R30 and R31

The PRU registers R30 and R31 are used to receive, transmit, and control the data for the PRU. As shown in Figure 30-34, the R31 is used to access data in the RX L1 FIFO, the R30 is used to transmit data from the PRU, and the R31 output is used the control the flow of receive and transmit. For more details about these registers, see the following sections.

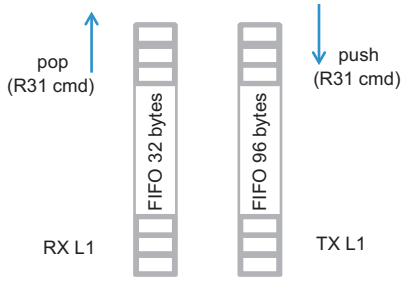
Figure 30-34. PRU R30, R31 Operations



30.1.9.2.2.3 RX and TX L1 FIFO Data Movement

To advance the next data byte seen by R31, the PRU must pop the data from the RX L1 FIFO. Likewise, the PRU can push the data from R30 to the TX L1 FIFO. These operations are illustrated in Figure 30-35.

Figure 30-35. Reading and Writing FIFO Data



30.1.9.2.2.4 CRC Computation

30.1.9.2.2.4.1 Receive CRC Computation

For the incoming data, the MII_RT calculates CRC32 and then compares against the value provided in the incoming frame. If there is a mismatch, the MII RT signals ERROR_CRC to the PRU. If a previous node or Ethernet device appended an error nibble, the CRC calculation of received packet will be wrong because the longer frame and the frame length will end at a 4-bit boundary instead of the usual 8-bit boundary. When RX_DV goes inactive on the 4-bit boundary, the interface will assert DATA_RDY and BYTE_RDY flag with the ERROR NIBBLE. The error event is also mapped into the PRU-ICSS INTC.

30.1.9.2.2.4.2 Transmit CRC Computation

For the outgoing data, the MII_RT calculates the CRC32 value and inserts it into outgoing packets. The CRC value computed on each MII transmit path is also available in memory map registers (MMRs) that can be read by the PRU and used primarily for debug and diagnostic purposes. The CRC is inserted into the outgoing packet based on the commands received through the R31 register of the PRU. The CRC will be inserted into the TX L1 FIFO, and there must be enough room to store the CRC value in the FIFO or else the FIFO will overflow. As Table 30-342 shows, the CRC programming model supports three sequences that provide more flexibility. Note “cmdR31” indicates write to the mentioned bits of the R31 command interface.

Table 30-342. TX CRC Programming Models

Option 1	Step 1: cmdR31 [TX_CRC_HIGH + TX_CRC_LOW + TX_EOF]
Option 2	Step 1: cmdR31 [TX_CRC_HIGH] Step 2: wait > 6 clocks (PRU cycles) Step 3: cmdR31 [TX_CRC_LOW + TX_EOF]
Option 3	Step 1: cmdR31 [TX_CRC_HIGH] Step 2: wait > 6 clocks (PRU cycles) Step 3: read PRUSS_MII_RT_TX_CRC0/1 Step 4: modify CRC[15:0] Step 5: cmdR31 [TX_PUSH16 + TX_EOF + TX_ERROR_NIBBLE]

30.1.9.2.3 RX MII Interface

30.1.9.2.3.1 RX MII Submodule Overview

The RX MII interface is composed of multiple submodules that process the incoming frames and pass receive data and status information into the PRU register R31. These submodules include:

- Latch received data
- Start of frame detection
- Start frame delimiter detection

- CRC calculation and error detection
- Enhanced link detection through RX error detection

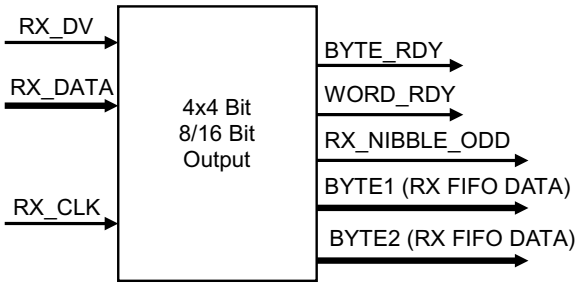
Table 30-813 includes more details about the internal signals and output of these submodules.

30.1.9.2.3.1.1 Receive Data Latch

The receive data from the MII interface is stored in the receive data FIFO which is 32 bytes. The PRU can access this data through the register R31. Depending on the configuration settings, the data can be latched on reception of one or two bytes. In each scheme, the configured number of nibbles is assembled before being copied into the PRU registers. Figure 30-36 shows the inputs and outputs of the data latch logic block.

The receiver logic in MII_RT can be programmed through the PRUSS_MII_RT_RXCFG0 and PRUSS_MII_RT_RXCFG1 registers to remove or retain the preamble + SFD from incoming frames.

Figure 30-36. RX Data Latch



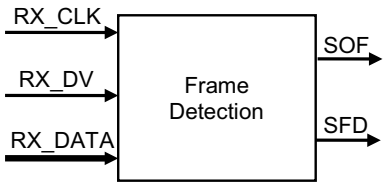
30.1.9.2.3.1.1.1 Start of Frame Detection

The start of frame detection logic tracks the frame boundaries and signals the beginning of a frame to other components of the PRU-ICSS. This logic detects two events:

- Start of Frame (SOF) event that occurs when Receive Data Valid MII signal is sampled high.
- Start of Frame Delimiter (SFD) event is seen on MII Receive Data bus.

These event triggers can be used to add timestamp to the frames. The notification for these events is available through R31 as well as through INTC which is integrated in the PRU-ICSS. Figure 30-37 shows the inputs and outputs of the start of frame detection logic block.

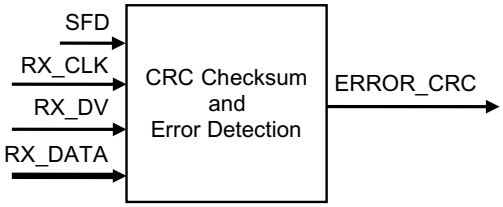
Figure 30-37. Start of Frame Detection



30.1.9.2.3.1.1.2 CRC Error Detection

For each incoming frame, the CRC is calculated by the MII_RT and compared against the CRC included in the frame. When the two values do not match, a CRC error is flagged. The ERROR_CRC indication is available in the register interface (PRU R31 Receive Interface) as well as in the FIFO interface (RX L2 Status Interface). It is also provided to the INTC which is integrated in the PRU-ICSS. Figure 30-38 shows the inputs and outputs of CRC error detection logic block.

Figure 30-38. CRC Error Detection

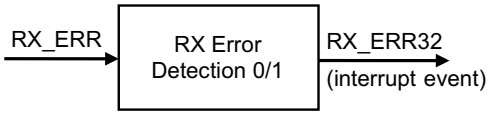


30.1.9.2.3.1.1.3 RX Error Detection and Action

The RX error detection logic tracks the receive error signaled by the physical layer and informs the PRU-ICSS INTC whenever an error is detected. Figure 30-39 shows the inputs and outputs of the RX error detection logic block. Note the following dependencies:

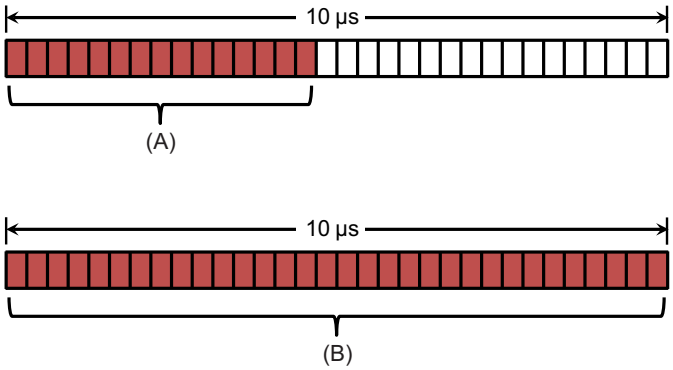
- RX_ERR signal is only sampled when RX_DV is asserted.
- All nibbles are discarded post RX_ERR event, including the nibble which had RX_ERR asserted. This state will remain until EOF occurs.
- Due to this fact, RX L1 FIFO and RX L2 FIFO will never receive any data with RX_ERR or post RX_ERR during that frame.

Figure 30-39. RX Error Detection



This submodule also keeps track of a running count of receive error events within a 10 µs error detection window, as shown in Figure 30-40. The INTC is notified when 32 or more events have occurred in a 10 µs error detection window. The error detection window is not a sliding window but a non-overlapping window with no specific initialization time with respect to incoming traffic. The timer starts its 10 µs counts immediately after de-assertion of reset to the MII_RT module.

Figure 30-40. Error Detection Window with Running Counter



A There are fewer than 32 consecutive error events in the 10 µs window. The detection module will not forward to the interrupt controller (INTC).

B There are more or equal to 32 error events in the 10 µs window. The detection module will notify the interrupt controller (INTC).

30.1.9.2.3.1.2 RX Data Path Options to PRU

There are two data path options for delivering received data to the PRU, described further in the subsequent sections:

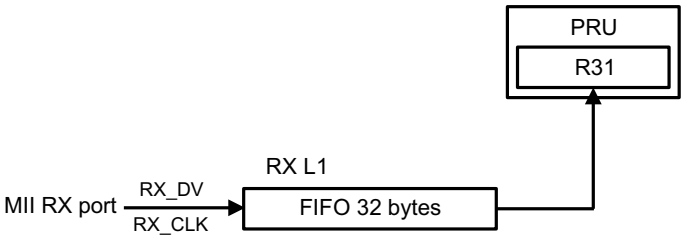
1. RX MII port → RX L1 FIFO → PRU (one word in flight)
2. RX MII port → RX L1 FIFO → RX L2 buffer → PRU (multi-word in flight)

Once the PRU has received RX data, the PRU can both manipulate received data or send data to the TX MII Interface.

30.1.9.2.3.1.2.1 RX MII Port → RX L1 FIFO → PRU

The RX L1 FIFO to PRU interface is depicted in Figure 30-41. In this mode, the data received from the MII interface is fed into the 32-byte RX L1 FIFO. The first data byte into the FIFO is automatically available in R31 of the PRU. Therefore, the PRU firmware can directly operate on this data without having to read it in a separate instruction. This allows the PRU to access receive data with low latency.

Figure 30-41. RX L1 to PRU Interface



When the new data is received, the PRU is provided with up to two bytes at a time in the R31 register, as shown in Figure 30-42. Once the PRU processes the incoming data, it instructs the MII_RT by writing to the R31 command interface bits to pop one or two bytes of data from the 32-byte RX FIFO. The pop operation causes current contents of R31 to be refreshed with new data from the incoming packet. Each time the data is popped, the status bits change to indicate so. If the pop is completed and there is no new data, the status bits immediately change to indicate no new data. Note the current R31 content, including data, will be lost after issuing the pop operation. If this information needs to be accessed later, the PRU should store the existing R31 content before popping new data.

Figure 30-42. MII RX Data to PRU R31 (R) and RX FIFO

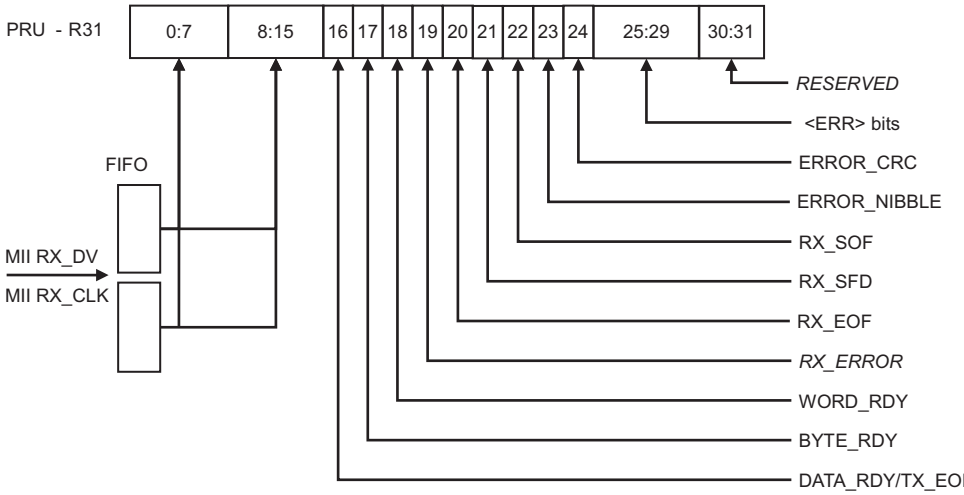


Table 30-343 describes the receive interface data and status contents provided by the R31 register. These contents are available when R31 is read. To configure this register, the PRU GPI mode should be set for MII_RT mode in the CFG register space. Note the following:

1. If the data from receive path is not read in time, it could cause an overflow event because the data is still continuously provided to the 32-byte receive FIFO. Due to the receive FIFO overflow, the data gets automatically discarded to avoid lack of space in the FIFO. At the same time, an interrupt is raised to the INTC through a system event (PRU<n>_RX_OVERFLOW). To detect an overflow condition, the PRU should poll for this system event condition and a RX RESET command through the R31 command interface is required to clear out from this condition. Note that the received Ethernet frame is corrupted and should not be used for further processing as bytes have been dropped due to the overflow condition. A FIFO reset is recommended.

2. The receive data in the R31 register is available following synchronization to the PRU clock domain. So, there is a finite delay (120 ns) when data is available from MII interface and it is accessible to the PRU.
3. The receive FIFO also has the capability to be reset through software. When reset, all contents of receive FIFO are purged and it may result in the current frame not being received as expected. When a frame is being received and the PRU resets the RX FIFO, the remaining frame is not placed into the RX FIFO. However, any new frame arriving on the receive MII port will be stored in the FIFO.

Table 30-343. PRU R31: Receive Interface Data and Status (Read Mode)

Bits	Field Name	Description
31:30	RESERVED	In case of register interface, these bits are provided to PRU by other modules in PRU-ICSS. From the MII_RT module point of view, these bits are always zero.
29	RX_MIN_FRM_CNT_ERR	RX_MIN_FRM_CNT_ERR is set to 1 when the count of total bytes of incoming frame is less than the value defined by RX_MIN_FRM_CNT. RX_MIN_FRM_CNT_ERR is cleared by RX_ERROR_CLR.
28	RX_MAX_FRM_CNT_ERR	RX_MAX_FRM_CNT_ERR is set to 1 when the count of total bytes of incoming frame is more than the value defined by RX_MAX_FRM_CNT_ERR. RX_MAX_FRM_CNT_ERR is cleared by RX_ERROR_CLR.
27	RX_EOF_ERROR	RX_EOF_ERROR is set to 1 when an RX_EOF event or RX_ERROR event occurs. RX_EOF_ERROR is cleared by RX_ EOF_CLR and/or RX_ ERROR_CLR.
26	RX_MAX_PRE_CNT_ERR	RX_MAX_PRE_CNT_ERR is set to 1 when the number of nibbles equaling 0x5 before SFD event (0xD5) is more than the value defined by PRUSS_MII_RT_RX_PCNT0/1 [RX_MAX_PCNT]. RX_MAX_PRE_CNT_ERR is cleared by RX_ERROR_CLR.
25	RX_ERR	RX_ERR is set to 1 when pr1_mii0/1_rxer is asserted while pr1_mii0/1_rxdv bit is set. RX_ERR is cleared by RX_ERROR_CLR.
24	ERROR_CRC	ERROR_CRC indicates that the frame has a CRC mismatch. This bit is valid when the RX_EOF bit is set. It should be noted that ERROR_CRC bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO. ERROR_CRC is cleared by RX_ERROR_CLR.
23	ERROR_NIBBLE	ERROR_NIBBLE indicates that the frame ended in odd nibble. It should be considered valid only when the RX_EOF bit and pr1_mii0/1_rxdv are set. Nibble counter is enabled post SFD event. It should be noted that ERROR_NIBBLE bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO. ERROR_NIBBLE is cleared by RX_ERROR_CLR.
22	RX_SOF	RX_SOF transitions from low to high when the frame data starts to arrive and pr1_mii0/1_rxdv is asserted. Note there will be a small sync delay of 0ns – 5ns. The recommended time to clear this bit via RX_SOF_CLR is at the end of frame (EOF). It should be noted that RX_SOF bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO.
21	RX_SFD	RX_SFD transitions from low to high when the SFD sequence (0xD5) post RX_SOF is observed on the receive MII data. The recommended time to clear this bit via RX_SFD_CLR is at the end of frame (EOF). It should be noted that RX_SFD bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO.
20	RX_EOF	RX_EOF indicates that the frame has ended and pr1_mii0/1_rxdv is de-asserted. It also validates the CRC match bit. Note there will be a small sync delay of 0ns – 5ns. It should be noted that RX_EOF bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO. Note also if RX_L2_EOF_SCLR_DIS is set, then this flag will remain asserted when RX_L2 is enabled until RX_EOF_CLR.
19	RX_ERROR	RX_ERROR indicates one or more of the following errors occurred: • RX_MAX/MIN_FRM_CNT_ERR • RX_MAX/MIN_PRE_CNT_ERR • RX_ERR RX_ERROR is cleared by RX_ERROR_CLR.
18	WORD_RDY	WORD_RDY indicates that all four nibbles in R31 have valid data. There is a 2 clock cycle latency from the command RX_POP16 to WORD_RDY update. Therefore, firmware needs to insure it does not read WORD_RDY until 2 clock cycles after RX_POP16.
17	BYTE_RDY	BYTE_RDY indicates that the lower two nibbles in R31 have valid data. There is a 2 clock cycle latency from the command RX_POP8 to BYTE_RDY update. Therefore, PRU firmware needs to insure it does not read BYTE_RDY until 2 clock cycles after RX_POP8.

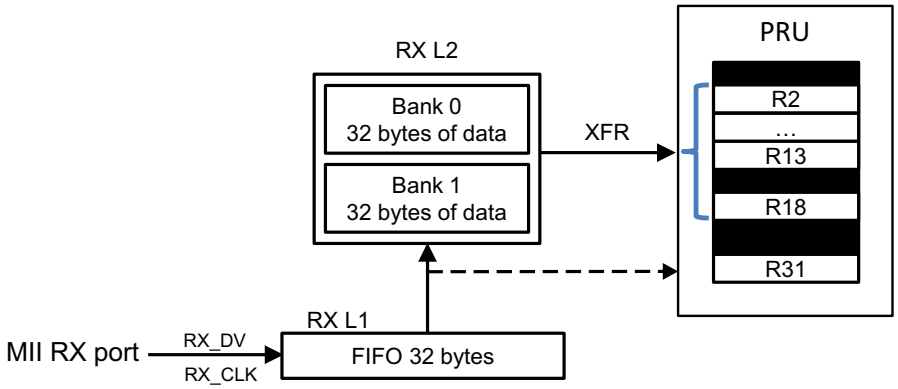
Table 30-343. PRU R31: Receive Interface Data and Status (Read Mode) (continued)

Bits	Field Name	Description
16	DATA_RDY/ TX_EOF	When RX_DATA_RDY_MODE_DIS = 0: DATA_RDY indicates there is valid data in R31 ready to be read. This bit goes to zero when the PRU does a POP8/16 and there is no new data left in the receive MII port. This bit is high if there is more receive data for PRU to read. There is a 2 clock cycle latency from the command RX_POP16/8 to WORD_RDY/BYTE_RDY update. Therefore, PRU firmware needs to insure it does not read BYTE_RDY/WORD_RDY until 2 clock cycles after RX_POP16/8. When RX_DATA_RDY_MODE_DIS = 1: TX_EOF indicates an TX EOF event (i.e. a 1 --> 0 transition on TX_EN) has occurred. After this bit has been set, a new TX frame can be loaded. This bit will clear when TX_RESET is set or when TX_FIFO is not empty.
15:8	BYTE1	Data Byte 1. This data is available such that it is safe to read by the PRU when the DATA_RDY/BYTE_RDY/WORD_RDY bits are asserted.
7:0	BYTE0	Data Byte 0. This data is available such that it is safe to read by the PRU when the DATA_RDY/BYTE_RDY/WORD_RDY bits are asserted.

30.1.9.2.3.1.2.2 RX MII Port → RX L1 FIFO → RX L2 Buffer → PRU

The RX L2 is an optional high performance buffer between the RX L1 FIFO and the PRU. Figure 30-43 illustrates the receive data path using RX L2 buffer. This data path is characterized by multi-word in flight transactions.

Figure 30-43. RX L2 to PRU Interface



The 64-byte RX L2 buffer is divided into two 32 byte banks, or ping/pong buffers. When the RX L2 is enabled, the incoming data from the MII RX port will transmit first to the 32 byte RX L1 FIFO. RX L1 pushes data into RX L2, starting when the first byte is ready until the final EOF marker. The RX L2 buffer does not apply any backpressure to the RX L1 FIFO. Therefore, it is the PRU firmware's responsibility to fetch the data in RX L2 before it is overwritten by the cyclic buffer. The RX L1 will remain near empty, with only one byte (nibble) stored.

Each RX L2 bank holds up to 32 bytes of data, and every four nibbles (or 16 bits) of data has a corresponding 8-bit status. The data and status information are stored in packed arrays. In each bank, R2 to R9 contains the data packed array and R10 to R13 contains the status packed array. Figure 30-44 shows the relationship of the data registers and status registers. The RX L2 status registers record status information about the received data, such as ERROR_CRC, RX_ERROR, STATUS_RDY, etc. The RX L2 status register details are described in Table 30-344. Note RX_RESET clears all Data and Status elements and resets R18.

Figure 30-44. Data and Status Register Dependency

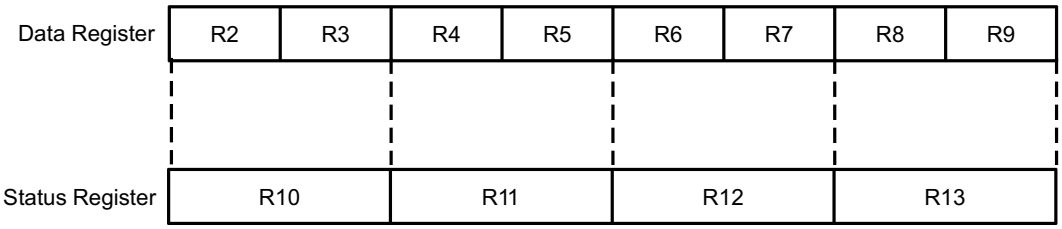


Table 30-344. RX L2 Status

Bit	Field Name	Description
7	ERROR_CRC	ERROR_CRC indicates that the frame has a CRC mismatch. This bit is valid when the RX_EOF bit is set. It should be noted that ERROR_CRC bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO. ERROR_CRC will only be set for one entry, self clear on next entry.
6	ERROR_NIBBLE	ERROR_NIBBLE indicates that the frame ended in odd nibble. It should be considered valid only when the RX_EOF bit and pr1_mii0/1_rxdv are set. Nibble counter is enabled post SFD event. It should be noted that ERROR_NIBBLE bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO. ERROR_NIBBLE will only be set for one entry, self clear on next entry.
5	RX_SOF	RX_SOF transitions from low to high when the frame data starts to arrive and pr1_mii0/1_rxdv is asserted. Note there will be a small sync delay of 0ns – 5ns. It should be noted that RX_SOF bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO. RX_SOF will only be set for one entry, self clear on next entry.
4	RX_SFD	RX_SFD transitions from low to high when the SFD sequence (0xD5) post RX_SOF is observed on the receive MII data. It should be noted that RX_SFD bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO. RX_SOF will only be set for one entry, self clear on next entry.
3	RX_EOF	RX_EOF indicates that the frame has ended and pr1_mii0/1_rxdv is de-asserted. It also validates the CRC match bit. Note there will be a small sync delay of 0ns – 5ns. It should be noted that RX_EOF bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO. If RX_L2_EOF_SCLR_DIS = 1, then RX_EOF will remain set until RX_EOF_CLR event. Otherwise, RX_ERROR is self-clearing on next entry.
2	RX_ERROR	RX_ERROR indicates one or more of the following errors occurred: • RX_MAX/MIN_FRM_CNT_ERR • RX_MAX/MIN_PRE_CNT_ERR • RX_ERR RX_ERROR is cleared by RX_ERROR_CLR.
1	STATUS_RDY	STATUS_RDY is set when RX_EOF or write pointer advanced by 2. This is a simple method for software to determine if RX_EOF event has occurred or new data is available. If RX_EOF is not set, all status bits are static.
0	RX_ERR	RX_ERR is set to 1 when pr1_mii0/1_rxer is asserted while pr1_mii0/1_rxdv bit is set. It will get set for first pr1_mii0/1_rxer event and self clear on SOF for the next FRAME.

Bank 0 and Bank 1 are used as ping/pong buffers. RX L2 supports the reading of a write pointer in R18 that allows software to determine which bank has active write transactions, as well as the specific write address within packed data arrays.

The PRU interacts with the RX L2 buffer using the high performance XFR read instructions and broadside interface. Table 30-345 shows the device XFR ID numbers for each bank.

Table 30-345. RX L2 XFR ID

Device ID	Function	Description
20	Selects RX L2 Bank0	R2:R9 Data packed array R10:R13 Status packed array
21	Selects RX L2 Bank1	R2:R9 Data packed array R10:R13 Status packed array
20/21	Byte pointer of current write	R18[5:0] Pointer indicating location of current write in data packed array. 0 = Bank0.R2.Byte0 (default and reset value) 1 = Bank0.R2.Byte1 2 = Bank0.R2.Byte2 3 = Bank0.R2.Byte3 4 = Bank0.R3.Byte0 … 63=Bank1.R9.Byte3

XFR read transactions are passive and have no effect on any status or other states in RX L2. The firmware can also read R18 to determine which Bank has active write transactions and the location of the transaction. With this information, the firmware can read multiple times the stable preserved data. Note when RX L1 data is written to RX L2, the next status byte gets cleared at the same time the current status byte gets updated. The rest of the status buffer is persistent. When software is accessing any register of the ping/ pong buffer, software needs to issue an XFER read transaction to fetch the latest/current state of the ping/pong buffer. The PRU registers will not reflect the current snapshot of L2 unless an XFER is issued by software.

30.1.9.2.4 TX MII Interface

Data to be transmitted is loaded into the TX L1 FIFO. The transmit FIFO (TX L1) stores up to 96 bytes of transmit data. From the FIFO, the data is sent to the MII TX port of the PHY by the MII_RT transmit logic.

The transmit FIFO also has the capability to be reset through software (TX_RESET). When reset, all contents of transmit FIFO are purged and this may result in a frame not getting transmitted as expected, if the transmission is already ongoing. Any new data written in the transmit FIFO results in a new frame being composed and transmitted. An overflow event will require a TX_RESET to recover from this condition.

There are four dependencies that must be true for TX_EN to assert.

1. TX L1 FIFO not empty
2. Interpacket gap (IPG) timer expiration
3. RX_DV to TX_EN timer expiration
4. TX_EN compare timer expiration

The transmit interface also provides an underflow error signal in case there was no data loaded when TX_EN triggered. The transmit underflow signal is mapped to the INTC in PRU-ICSS. The PRU firmware must track the FIFO fill level, such as by a timer or the PRU cycle count register (PRU_ICSS_CTRL_CYCLE).The current FIFO fill level cannot be accessed by PRU firmware. The firmware can issue an R31 command via R31 bit 29 (TX_EOF) to indicate that the last byte has been written into the TX FIFO.

30.1.9.2.4.1 TX Data Path Options to TX L1 FIFO

There are two data path options for delivering data to the TX L1 FIFO and transmit port, described further in the subsequent sections:

1. PRU → TX L1 FIFO → TX MII port
2. RX L1 FIFO → TX L1 FIFO → TX MII port

30.1.9.2.4.1.1 PRU → TX L1 FIFO → TX MII Port

The PRU can be used to feed data into the TX L1 FIFO using the R30 and R31 registers, shown in Figure 30-45. The PRU has the option two write up to two or four bytes of R30 and then pushes the data into the TX L1 FIFO by writing to the R31 command interface.

Figure 30-45. PRU to TX L1 Interface

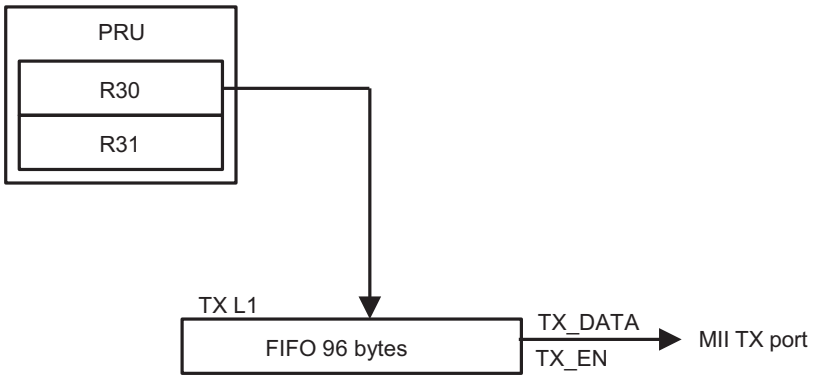


Figure 30-46 shows the R30 transmit interface. The lower 16 bits of the R30 (or FIFO transmit word) contain transmit data nibbles. When PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 0, then the upper 16 bits contain mask information. Alternatively, when PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 1, then the upper 16 bits contain transmit data nibbles. The operation to be performed on the transmit interface is controlled by PRU writes to the R31 command interface. Table 30-346 describes the supported configurations for 8, 16, and 32 bit TX push operations.

Figure 30-46. PRU to TX MII Interface

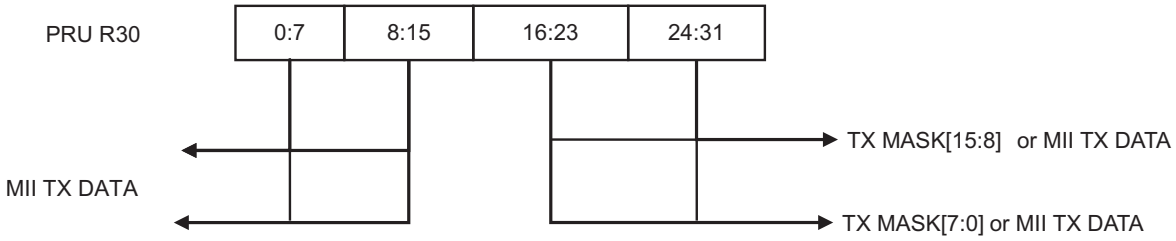


Table 30-346. TX Push

R31[25] TX_PUSH16/32	R31[24] TX_PUSH8/32	Supported R30 bits	TX_32_MODE_EN	TX_BYTE_SWAP	TX Push Action
0	1	X	0	X	8 bits of TXDATA (R30[7:0]) pushed post TX mask
1	0	X	0	X	16 bits of TXDATA (R30[15:0]) pushed post TX mask
1	1	X	0	X	Illegal
X	X	0x000000FF	1	0	8 bits of TXDATA (R30[7:0]) pushed
X	X	0x0000FFFF	1	0	16 bits of TXDATA (R30[15:0]) pushed
X	X	0xFFFFFFFF	1	X	32 bits of TXDATA (R30[31:0]) pushed
X	X	All other - reserved	1	X	Reserved

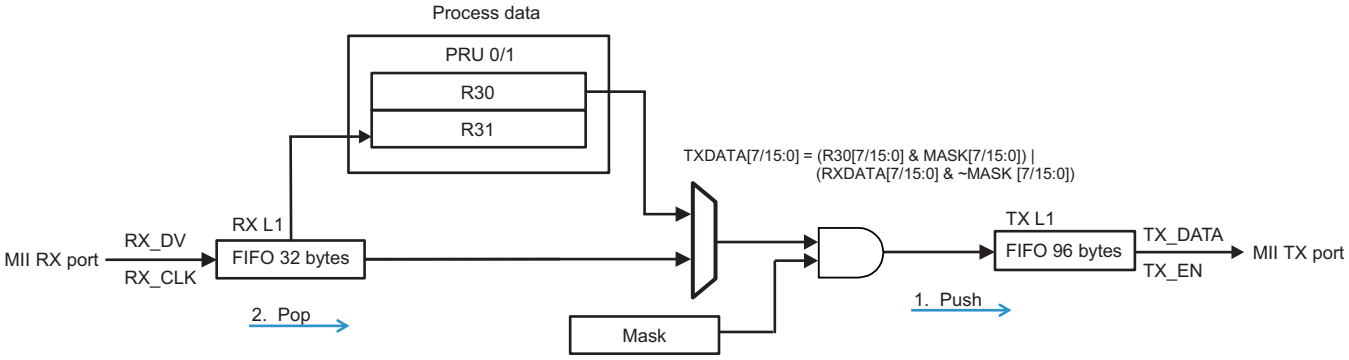
Using PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 0 and the TX mask, the PRU can send a mix of R30 and RX L1 FIFO data to the TX L1 FIFO. Note the TX mask is only available when the PRU is fed one word or byte at a time by the RX L1 FIFO. It is not applicable when the RX L2 buffer is enabled. To disable TX mask, set TXMASK to 0xFFFF.

As shown in Figure 30-47, the PRU drives the MII transmit interface through its R30 register. The contents of R30 and RX data from the receive interface are taken and fed into a 96 byte transmit FIFO.

If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 0, then before transmission, a mask is applied to the data portion of the R30 register. By using the mask, the PRU firmware can control whether received data from the RX L1 FIFO is sent to transmit, R30 data is sent to transmit, or a mix of the two is sent. The Boolean equation that is used by MII_RT to compose TX data is:

TXDATA[7/15:0] = (R30[7/15:0] & MASK[7/15:0]) | (RXDATA[7/15:0] & ~MASK [7/15:0])

Figure 30-47. TX Mask Mode (PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 0)



30.1.9.2.4.1.2 RX L1 FIFO → TX L1 FIFO → TX MII Port

When PRUSS_MII_RT_TXCFG0/1[TX_AUTO_SEQUENCE] is set, the data frame is passed from the RX to TX FIFOs without the any interaction of the PRU. This mode of operations is shown in Figure 30-48. The RX L1 will push into TX L1 as long as it is enabled and not full.

There is no PRU dependency in this mode and no option for the PRU to perform any operation to the TX L1 FIFO. RX_RESET clears all data and status elements.

Figure 30-48. RX L1 to TX L1 Interface

30.1.9.2.5 PRU R31 Command Interface

The PRU uses writes to R31[31:16] to control the reception and transmission of packets in register mode. Table 30-347 lists the available commands. Each bit in the table is a single clock pulse output from the PRU. When more than one action is to be performed in the same instant, the PRU firmware must set those command bits in one instruction.

Table 30-347. PRU R31: Command Interface (Write Mode)

Bit	Command	Description
31	TX_CRC_ERR	TX_CRC_ERR command when set will add 0xa5 byte to the TX L1 FIFO if the current FCS is valid. This bit can only be set with the TX_EOF command and optionally with the TX_ERROR_NIBBLE command. It cannot get set with any other commands, and the PRU firmware must wait > 2 clocks from the last command. Note for proper operations auto-forward preamble must be enabled.
30	TX_RESET	TX_RESET command is used to reset the transmit FIFO and clear all its contents. This is required to recover from a TX FIFO overrun.

Table 30-347. PRU R31: Command Interface (Write Mode) (continued)

Bit	Command	Description
29	TX_EOF	TX_EOF command is used to indicate that the data loaded is considered last for the current frame
28	TX_ERROR_NIBBLE	TX_ERROR_NIBBLE command is used to insert an error nibble. This makes the frame invalid. Also, it will add 0x0 after the 32-bit CRC.
27	TX_CRC_HIGH	TX_CRC_HIGH command ends the CRC calculations and pushes CRC[31:16] to append to the outgoing frame in the TX L1 FIFO. Note PRUSS_MII_RT_TX_CRC0/1 will become valid after 6 clock cycles.
26	TX_CRC_LOW	TX_CRC_LOW command pushes CRC[15:0] to append to the outgoing frame in the TX L1 FIFO.
25	TX_PUSH16	TX_PUSH16 command pushes R30[15:0] when PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 0. See Table x, TX Push for more details. Note there are no restrictions on concurrent PUSH/POP nor R30 requirements to maintain data. Back to back PUSH is supported.
24	TX_PUSH8	TX_PUSH8 command pushes R30[7:0] when PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 0. See Table x, TX Push for more details. Note there are no restrictions on concurrent PUSH/POP nor R30 requirements to maintain data. Back to back PUSH is supported.
23	RX_ ERROR_CLR	RX_ERROR_CLR command is used to clear RX_ ERROR indicator bit by writing 1.
22	RX_EOF_CLR	RX_EOF_CLR command is used to clear RX_EOF status indicator bit by writing 1.
21	RX_SFD_CLR	RX_SFD_CLR command is used to clear RX_SFD indicator bit by writing 1.
20	RX_SOF_CLR	RX_SOF_CLR command is used to clear RX_SOF indicator bit by writing 1.
19	Reserved	Reserved
18	RX_RESET	RX_RESET is used to reset the receive FIFO and clear all contents. This is required to recover from a RX FIFO overrun, if software does not want to undrain. The typical use case is assertion after RX_EOF. If asserted during an active frame, the following actions will occur: 1. Terminate the current frame 2. Block/terminate all new data 3. Flush/clear all FIFO elements 4. Cause RX state machine into an idle state 5. Cause EOF event 6. Cause minimum frame error, if you abort before minimum size reached
17	RX_POP16	RX_POP16 command advances the receive traffic by two bytes. This is only required when you are using R31 to read the data. After R31[15:0] is ready to read by PRU, it will set 1 to WORD_RDY, and the next new data will be allowed to advance. RX_POP16 to WORD_RDY update has 2 clock cycles latency. Firmware needs to insure it does not read WORD_RDY/BYTE_RDY until 2 clock cycles after RX_POP16.
16	RX_POP8	RX_POP8 command advances the receive traffic by one bytes. This is only required when you are using R31 to read the data. After R31[7:0] is ready to read by PRU, it will set 1 to BYTE_RDY, and the next new data will be allowed to advance. RX_POP8 to BYTE_RDY update has 2 clock cycles latency. Firmware needs to insure it does not read WORD_RDY/BYTE_RDY until 2 clock cycles after RX_POP8.

30.1.9.2.6 Other Configuration Options

30.1.9.2.6.1 Nibble and Byte Order

The PRU core is little endian. To support big endian, the MII_RT supports optional nibble swapping on both the RX and TX side.

On the receive side, the order of the two data bytes in RX R31 and the RX L2 buffer are configurable through the RX_BYTE_SWAP bit in the PRUSS_MII_RT_RXCFG0/1 registers, as shown in Table 30-348. Note the Nibble0 is the first nibble received.

Table 30-348. RX Nibble and Byte Order

Configuration	Order
PRUSS_MII_RT_RXCFG0/1[RX_BYTE_SWAP] = 0 (default)	R31[15:8] / RXL2[15:8] = Byte1{Nibble3,Nibble2} R31[7:0] / RXL2[7:0] = Byte0{Nibble1,Nibble0}
PRUSS_MII_RT_RXCFG0/1[RX_BYTE_SWAP] = 1	R31[15:8] / RXL2[15:8] = Byte0{Nibble1,Nibble0} R31[7:0] / RXL2[7:0] = Byte1{Nibble3,Nibble2}

On the transmit side, the order of the two data bytes and mask bytes in TX R30 are configurable through the TX_BYTE_SWAP bit in the PRUSS_MII_RT_TXCFG0/1 registers, as shown in Table 30-349. Note the Nibble0 is the first nibble received.

Table 30-349. TX Nibble and Byte Order

Configuration	Order
PRUSS_MII_RT_TXCFG0/1[TX_BYTE_SWAP] = 0 (default)	If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 0, R30[15:8] = Byte1{Nibble3,Nibble2} R30[7:0] = Byte0{Nibble1,Nibble0} R30[31:24] = TX_MASK[15:8] R30[23:16] = TX_MASK[7:0] If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 1, R30[31:24] = Byte3{Nibble7,Nibble6} R30[23:16] = Byte2{Nibble5,Nibble4} R30[15:8] = Byte1{Nibble3,Nibble2} R30[7:0] = Byte0{Nibble1,Nibble0}
PRUSS_MII_RT_TXCFG0/1[TX_BYTE_SWAP] = 1	If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 0, R30[15:8] = Byte0{Nibble1,Nibble0} R30[7:0] = Byte1{Nibble3,Nibble2} R30[31:24] = TX_MASK[7:0] R30[23:16] = TX_MASK[15:8] If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 1, Only 32bit push is supported. R30[31:24] = Byte0{Nibble1,Nibble0} R30[23:16] = Byte1{Nibble3,Nibble2} R30[15:8] = Byte2{Nibble5,Nibble4} R30[7:0] = Byte3{Nibble7,Nibble6}

30.1.9.2.6.2 Preamble Source

The MII_RT module has the option to preserve and forward a received preamble in the TX data stream, use a preamble provided by the PRU, or auto-generate a preamble. These configurations are highlighted in Table 30-350.

Table 30-350. Preamble Configuration Options

RX_CUT_PREAMBLE	Determines whether RX preamble is passed to the RX L1/L2 FIFO
RX_AUTO_FWD_PRE	Determines whether RX preamble is automatically passed to TX L1 FIFO

Table 30-350. Preamble Configuration Options (continued)

TX_AUTO_PREAMBLE	TX interface logic auto-generates and appends preamble to TX data stream with the first push of data into the TX L1 FIFO. Note that enabling this option does fill the TX FIFO with the preamble length, hence software has to consider this to not overrun the TX FIFO.

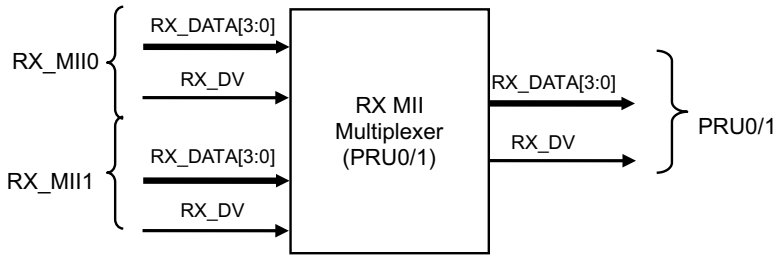
30.1.9.2.6.3 PRU and MII Port Multiplexer

The MII_RT module supports configurable PRU core to MII TXn / RXn port mapping. By default, PRU0 is mapped to TX1 and RX0 and PRU1 is mapped to TX0 and RX1. However, the system supports the flexibility to map any PRU core to any TX and RX port. Note the mapping options are destination fixed. For example, the input to PRU0 can be either RX_MII0 or RX_MII1. Similarly, the input to TX_MII0 can be either PRU0 or PRU1.

30.1.9.2.6.3.1 Receive Multiplexer

A multiplexer is provided to allow selecting either of the two MII interfaces for the receive data that is sent to PRU. Figure 30-49 shows the symbol of receive multiplexer of PRU.

Figure 30-49. MII Receive Multiplexer

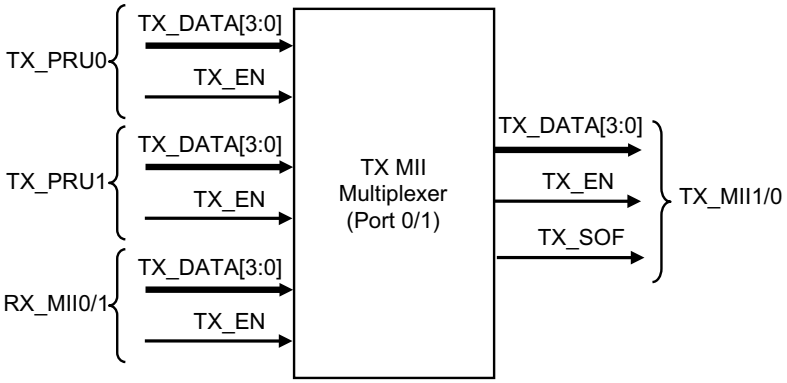


There are two receive multiplexer instances to enable selection of RX MII path for each PRU. The select lines of the RX multiplexers are driven from the PRU-ICSS programmable registers (PRUSS_MII_RT_RXCFG0/1).

30.1.9.2.6.3.2 Transmit Multiplexer

On the MII transmit ports, there is a multiplexer for each MII transmit port that enables selection of either the transmit data from the PRUs or from the RX MII interface of the other MII interface. Figure 30-104 shows the symbol of transmit multiplexer of PRU.

Figure 30-50. MII Transmit Multiplexer

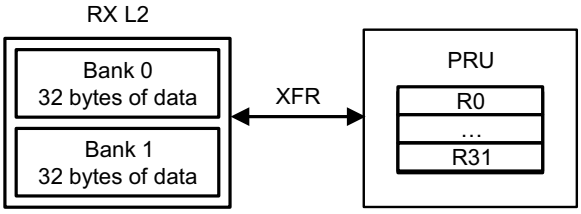


The transmit multiplexers enable the PRU-ICSS to either operate in a bypass mode where the PRU is not involved in processing MII traffic or use of one of the PRU cores for transmitting data into the MII interface. There are two instances of the TX MII multiplexer and the select lines for each TX multiplexer are provided by the PRU-ICSS. The select lines are common between register and FIFO interface. It is expected that the select lines will not change during the course of a frame so that can avoid data exchange error.

30.1.9.2.6.4 RX L2 Scratch Pad

When the RX L2 is disabled (PRUSS_MII_RT_RXCFG0/1[RX_L2_EN] = 0), the RX L2 banks can be used as a generic scratch pad. In scratch pad mode, RX L2 Bank0 and RX L2 Bank1 operate like simple write/read memory mapped registers (MMRs). All XFR size and start operations are supported. RX_RESET has no effect in this mode. This mode is shown in Figure 30-51.

Figure 30-51. Scratch Pad Mode



30.1.9.3 PRU-ICSS MII RT Module Register Manual

This section describes the PRU-ICSS MII_RT module configuration registers.

30.1.9.3.1 PRUSS_MII_RT Instance Summary

Table 30-351. PRUSS_MII_RT Instance Summary

Module Name	Module Base Address L3_MAIN	Size
PRUSS1_MII_RT	0x4B23 2000	88 Bytes
PRUSS2_MII_RT	0x4B2B 2000	88 Bytes

30.1.9.3.2 PRUSS_MII_RT Registers

30.1.9.3.2.1 PRUSS_MII_RT Register Summary

Table 30-352. PRUSS1_MII_RT Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
PRUSS_MII_RT_RXCFG0	RW	32	0x0000 0000	0x4B23 2000
PRUSS_MII_RT_RXCFG1	RW	32	0x0000 0004	0x4B23 2004
PRUSS_MII_RT_TXCFG0	RW	32	0x0000 0010	0x4B23 2010
PRUSS_MII_RT_TXCFG1	RW	32	0x0000 0014	0x4B23 2014
PRUSS_MII_RT_TX_CRC0	R	32	0x0000 0020	0x4B23 2020
PRUSS_MII_RT_TX_CRC1	R	32	0x0000 0024	0x4B23 2024
PRUSS_MII_RT_TX_IPG0	RW	32	0x0000 0030	0x4B23 2030
PRUSS_MII_RT_TX_IPG1	RW	32	0x0000 0034	0x4B23 2034
PRUSS_MII_RT_PRS0	R	32	0x0000 0038	0x4B23 2038
PRUSS_MII_RT_PRS1	R	32	0x0000 003C	0x4B23 203C
PRUSS_MII_RT_RX_FRMS0	RW	32	0x0000 0040	0x4B23 2040
PRUSS_MII_RT_RX_FRMS1	RW	32	0x0000 0044	0x4B23 2044
PRUSS_MII_RT_RX_PCNT0	RW	32	0x0000 0048	0x4B23 2048
PRUSS_MII_RT_RX_PCNT1	RW	32	0x0000 004C	0x4B23 204C
PRUSS_MII_RT_RX_ERR0	RW	32	0x0000 0050	0x4B23 2050
PRUSS_MII_RT_RX_ERR1	RW	32	0x0000 0054	0x4B23 2054
PRUSS_MII_RT_RXFLV0	R	32	0x0000 0060	0x4B23 2060
PRUSS_MII_RT_RXFLV1	R	32	0x0000 0064	0x4B23 2064
PRUSS_MII_RT_TXFLV0	R	32	0x0000 0068	0x4B23 2068
PRUSS_MII_RT_TXFLV1	R	32	0x0000 006C	0x4B23 206C

Table 30-353. PRUSS2_MII_RT Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
PRUSS_MII_RT_RXCFG0	RW	32	0x0000 0000	0x4B2B 2000
PRUSS_MII_RT_RXCFG1	RW	32	0x0000 0004	0x4B2B 2004
PRUSS_MII_RT_TXCFG0	RW	32	0x0000 0010	0x4B2B 2010
PRUSS_MII_RT_TXCFG1	RW	32	0x0000 0014	0x4B2B 2014
PRUSS_MII_RT_TX_CRC0	R	32	0x0000 0020	0x4B2B 2020
PRUSS_MII_RT_TX_CRC1	R	32	0x0000 0024	0x4B2B 2024
PRUSS_MII_RT_TX_IPG0	RW	32	0x0000 0030	0x4B2B 2030
PRUSS_MII_RT_TX_IPG1	RW	32	0x0000 0034	0x4B2B 2034
PRUSS_MII_RT_PRS0	R	32	0x0000 0038	0x4B2B 2038
PRUSS_MII_RT_PRS1	R	32	0x0000 003C	0x4B2B 203C
PRUSS_MII_RT_RX_FRMS0	RW	32	0x0000 0040	0x4B2B 2040
PRUSS_MII_RT_RX_FRMS1	RW	32	0x0000 0044	0x4B2B 2044
PRUSS_MII_RT_RX_PCNT0	RW	32	0x0000 0048	0x4B2B 2048
PRUSS_MII_RT_RX_PCNT1	RW	32	0x0000 004C	0x4B2B 204C
PRUSS_MII_RT_RX_ERR0	RW	32	0x0000 0050	0x4B2B 2050
PRUSS_MII_RT_RX_ERR1	RW	32	0x0000 0054	0x4B2B 2054
PRUSS_MII_RT_RXFLV0	R	32	0x0000 0060	0x4B2B 2060
PRUSS_MII_RT_RXFLV1	R	32	0x0000 0064	0x4B2B 2064
PRUSS_MII_RT_TXFLV0	R	32	0x0000 0068	0x4B2B 2068
PRUSS_MII_RT_TXFLV1	R	32	0x0000 006C	0x4B2B 206C

30.1.9.3.2.2 PRUSS_MII_RT Register Description

Table 30-354. PRUSS_MII_RT_RXCFG0

Address Offset	0x0000 0000		
Physical Address	0x4B23 2000 0x4B2B 2000	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII RXCFG 0 REGISTER This register contains the PRU0 RXCFG configuration variables (PRUSS_MII_RT_RXCFG0) for the RX path. PRUSS_MII_RT_RXCFG0 is attached to PRU0. PRUSS_MII_RT_RXCFG0 controls which RX port is attached to PRU0.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						RX_L2_EOF_SCLR_DIS	RX_ERR_RAW	RX_SFD_RAW	RX_AUTO_FWD_PRE	RX_BYTE_SWAP	RX_L2_EN	RX_MUX_SEL	RX_CUT_PREAMBLE	RX_DATA_RDY_MODE_DIS	RX_ENABLE

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x000000
9	RX_L2_EOF_SCLR_DIS	0x0: RX_EOF flag in R31 and RXL2 is self cleared by hardware when RXL2 is enabled 0x1: RX_EOF flag in R31 and RXL2 is not self cleared by hardware when RXL2 is enabled. To clear this flag, RX_EOF_CLR must be set.	RW	0x0
8	RX_ERR_RAW	0x0: Error Raw Mode Disabled. RX_ERR is qualified with RX_DV, meaning RX_DV = 1 before RX_ERR action/event is generated. 0x1: Error Raw Mode Enabled. RX_ERR is not qualified with RX_DV, meaning RX_ERR action/event is generated even if RX_DV = 0.	RW	0x0
7	RX_SFD_RAW	0x0: SFD Raw Mode Disabled. RX_SFD requires a pattern of 5D. 0x1: SFD Raw Mode Enable. The first byte of any pattern after RX_DV assertion will trigger RX_SFD event. The first nibble of the frame (RX_DV = 1) will be in the RX FIFO.	RW	0x0
6	RX_AUTO_FWD_PRE	Enables auto-forward of received preamble. When enabled, this will forward the preamble nibbles including the SFD to the TX L1 FIFO that is attached to the PRU. First data byte seen by PRU R31 and/or RX L2 is destination address (DA). **Note: Odd number of preamble nibbles is supported in this mode. For example, 0x55D Note that new RX should only occur after the current TX completes** 0x0: Disable 0x1: Enable, it must disable RX_CUT_PREAMBLE and TX_AUTO_PREAMBLE	RW	0x0

Bits	Field Name	Description	Type	Reset
5	RX_BYTE_SWAP	Defines the order of Byte0/1 placement for RX R31 and RX L2. **Note: that if TX_AUTO_SEQUENCE enabled, this bit cannot get enable since TX_BYTE_SWAP on swaps the PRU output. This bit must be selected/updated when the port is disabled or there is no traffic.** 0x0: R31 [15:8]/RXL2 [15:8] = Byte1{Nibble3,Nibble2} R31[7:0]/RXL2 [7:0] = Byte0{Nibble1,Nibble0} 0x1: R31 [15:8]/RXL2 [15:8] = Byte0{Nibble1,Nibble0} R31[7:0]/RXL2 [7:0] = Byte1{Nibble3,Nibble2} Nibble0 is the first nibble received.	RW	0x0
4	RX_L2_EN	Enables RX L2 buffer. 0x0: Disable (RX L2 can function as generic scratch pad) 0x1: Enable	RW	0x0
3	RX_MUX_SEL	Selects receive data source. Typically, the setting for this will not be identical for the two MII receive configuration registers. 0x0: MII RX Data from Port 0 (default for PRUSS_MII_RT_RXCFG0) 0x1: MII RX Data from Port 1 (default for PRUSS_MII_RT_RXCFG1)	RW	0x0
2	RX_CUT_PREAMBLE	Removes received preamble. 0x0: All data from Ethernet PHY are passed on to PRU register. This assumes Ethernet PHY which does not shorten the preamble. 0x1: MII interface suppresses preamble and sync frame delimiter. First data byte seen by PRU register is destination address.	RW	0x0
1	RX_DATA_RDY_MODE_DIS	0x0: R31, Bit 16 is configured for DATA_RDY mode. 0x1: R31, Bit 16 is configured for TX_EOF mode.	RW	0x0
0	RX_ENABLE	Enables the receive traffic currently selected by RX_MUX_SELECT. 0x0: Disable 0x1: Enable	RW	0x0

Table 30-355. Register Call Summary for Register PRUSS_MII_RT_RXCFG0

PRU-ICSS MII RT Module

- RX MII Submodule Overview: [12]
- Nibble and Byte Order: [13] [14] [15]
- PRU and MII Port Multiplexer: [16]
- RX L2 Scratch Pad: [17]
- PRUSS_MII_RT Register Summary: [18] [19]
- PRUSS_MII_RT Register Description: [20] [21] [22] [23] [24]

Table 30-356. PRUSS_MII_RT_RXCFG1

Address Offset	0x0000 0004		
Physical Address	0x4B23 2004 0x4B2B 2004	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII RXCFG 1 REGISTER This register contains the PRU1 RXCFG configuration variables (PRUSS_MII_RT_RXCFG1) for the RX path. PRUSS_MII_RT_RXCFG1 is attached to PRU1. PRUSS_MII_RT_RXCFG1 controls which RX port is attached to PRU1		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						RX_L2_EOF_SCLR_DIS	RX_ERR_RAW	RX_SFD_RAW	RX_AUTO_FWD_PRE	RX_BYTE_SWAP	RX_L2_EN	RX_MUX_SEL	RX_CUT_PREAMBLE	RX_DATA_RDY_MODE_DIS	RX_ENABLE

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x000000
9	RX_L2_EOF_SCLR_DIS	0x0: RX_EOF flag in R31 and RXL2 is self cleared by hardware when RXL2 is enabled 0x1: RX_EOF flag in R31 and RXL2 is not self cleared by hardware when RXL2 is enabled. To clear this flag, RX_EOF_CLR must be set.	RW	0x0
8	RX_ERR_RAW	0x0: Error Raw Mode Disabled. RX_ERR is qualified with RX_DV, meaning RX_DV = 1 before RX_ERR action/event is generated. 0x1: Error Raw Mode Enabled. RX_ERR is not qualified with RX_DV, meaning RX_ERR action/event is generated even if RX_DV = 0.	RW	0x0
7	RX_SFD_RAW	0x0: SFD Raw Mode Disabled. RX_SFD requires a pattern of 5D. 0x1: SFD Raw Mode Enable. The first byte of any pattern after RX_DV assertion will trigger RX_SFD event. The first nibble of the frame (RX_DV = 1) will be in the RX FIFO.	RW	0x0
6	RX_AUTO_FWD_PRE	Enables auto-forward of received preamble. When enabled, this will forward the preamble nibbles including the SFD to the TX L1 FIFO that is attached to the PRU. First data byte seen by PRU R31 and/or RX L2 is destination address (DA). **Note: Odd number of preamble nibbles is supported in this mode. For example, 0x55D Note that new RX should only occur after the current TX completes** 0x0: Disable 0x1: Enable, it must disable RX_CUT_PREAMBLE and TX_AUTO_PREAMBLE	RW	0x0
5	RX_BYTE_SWAP	Defines the order of Byte0/1 placement for RX R31 and RX L2. **Note: that if TX_AUTO_SEQUENCE enabled, this bit cannot get enable since TX_BYTE_SWAP on swaps the PRU output. This bit must be selected/updated when the port is disabled or there is no traffic.** 0x0: R31 [15:8]/RXL2 [15:8] = Byte1{Nibble3,Nibble2} R31[7:0]/RXL2 [7:0] = Byte0{Nibble1,Nibble0} 0x1: R31 [15:8]/RXL2 [15:8] = Byte0{Nibble1,Nibble0} R31[7:0]/RXL2 [7:0] = Byte1{Nibble3,Nibble2} Nibble0 is the first nibble received.	RW	0x0
4	RX_L2_EN	Enables RX L2 buffer. 0x0: Disable (RX L2 can function as generic scratch pad) 0x1: Enable	RW	0x0
3	RX_MUX_SEL	Selects receive data source. Typically, the setting for this will not be identical for the two MII receive configuration registers. 0x0: MII RX Data from Port 0 (default for PRUSS_MII_RT_RXCFG0) 0x1: MII RX Data from Port 1 (default for PRUSS_MII_RT_RXCFG1)	RW	0x1

Bits	Field Name	Description	Type	Reset
2	RX_CUT_PREAMBLE	Removes received preamble. 0x0: All data from Ethernet PHY are passed on to PRU register. This assumes Ethernet PHY which does not shorten the preamble. 0x1: MII interface suppresses preamble and sync frame delimiter. First data byte seen by PRU register is destination address.	RW	0x0
1	RX_DATA_RDY_MODE_DIS	0x0: R31, Bit 16 is configured for DATA_RDY mode. 0x1: R31, Bit 16 is configured for TX_EOF mode.	RW	0x0
0	RX_ENABLE	Enables the receive traffic currently selected by RX_MUX_SELECT. 0x0: Disable 0x1: Enable	RW	0x0

Table 30-357. Register Call Summary for Register PRUSS_MII_RT_RXCFG1

PRU-ICSS MII RT Module

- RX MII Submodule Overview: [2]
- PRUSS_MII_RT Register Summary: [3] [4]
- PRUSS_MII_RT Register Description: [5] [6] [7] [8] [9]

Table 30-358. PRUSS_MII_RT_TXCFG0

Address Offset	0x0000 0010		
Physical Address	0x4B23 2010 0x4B2B 2010	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII TXCFG 0 REGISTER This register contains the configuration variables for the transmit path on the MII interface port 0. PRUSS_MII_RT_TXCFG0 is attached to Port TX0. PRUSS_MII_RT_TXCFG0 controls which PRU is selected for TX0		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		TX_CLK_DELAY		RESERVED				TX_START_DELAY								RESERVED				TX_32_MODE_EN	RESERVED	TX_AUTO_SEQUENCE	TX_MUX_SEL	RESERVED				TX_BYTE_SWAP	TX_EN_MODE	TX_AUTO_PREAMBLE	TX_ENABLE

Bits	Field Name	Description	Type	Reset
31	RESERVED		R	0
30:28	TX_CLK_DELAY	In order to guarantee the MII_RT IO timing values published in the device data manual, the PRUSS_GICLK clock must be configured for 200MHz and TX_CLK_DELAY must be set to 6h.	RW	0x0
27:26	RESERVED		R	0x0

Bits	Field Name	Description	Type	Reset
25:16	TX_START_DELAY	Defines the minimum time interval (delay) between receiving the RXDV for the current frame and the start of the transmit interface sending data to the MII interface. Delay value is in units of MII_RT clock cycles, which uses the PRUSS_GICLK (default is 200MHz, or 5ns). Default TX_START_DELAY value is 320ns, which is optimized for minimum latency at 16 bit processing. Counter is started with RX_DV signal going active. Transmit interface stops sending data when no more data is written into transmit interface by PRU along with TX_EOF marker bit set. If the TX FIFO has data when the delay expires, then TX will start sending data. But if the TX FIFO is empty, it will not start until the TX FIFO is not empty. It is possible to overflow the TX FIFO with the max delay setting when auto-forwarding is enabled since the time delay is larger than the amount of data it needs to store. As long as TX L1 FIFO overflows, software will need to issue a TX_RESET to reset the TX FIFO. The total delay is 96-byte times (size of TX FIFO), but you need to allow delays for synchronization. Do to this fact, the maximum delay should be 80ns less when auto forwarding is enabled. Therefore, 0x3F0 is the maximum in this configuration.	RW	0x40
15:12	RESERVED		R	0x0
11	TX_32_MODE_EN	0x0: Disable 32-bit Data Push mode. 0x1: Enable 32-bit, 16-bit, and 8-bit Data Push mode with TX_MASK disabled. In this mode, the internal PRU R30 byte write strobes are used and not the R31 CMD TX_PUSH mode. Any update to R30 will trigger an TX PUSH. See Table 30-346 .	RW	0x0
10	RESERVED		R	0x0
9	TX_AUTO_SEQUENCE	Enables transmit auto-sequence. Note the transmit data source is determined by TX_MUX_SEL setting. 0x0: Disable 0x1: Enable, transmit state machine based on events on receiver path that is connected to the respective transmitter. Also, the masking logic is disabled and only the MII data is used.	RW	0x0
8	TX_MUX_SEL	Selects transmit data source. The default/reset setting for TX Port 0 is 1. This setting permits MII TX Port 0 to receive data from PRU1 and the MII TX Port 1 which is connected to PRU0 by default. 0x0: Data from PRU0 (default for PRUSS_MII_RT_TXCFG1) 0x1: Data from PRU1 (default for PRUSS_MII_RT_TXCFG0)	RW	0x1
7:4	RESERVED		R	0x0

Bits	Field Name	Description	Type	Reset
3	TX_BYTE_SWAP	Defines the order of Byte0/1 placement for TX R30. This bit must be selected/updated when the port is disabled or there is no traffic. **0x0: If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 0,** R30[15:8] = Byte1{Nibble3,Nibble2} R30[7:0] = Byte0{Nibble1,Nibble0} R30[31:24] = TX_MASK[15:8] R30[23:16] = TX_MASK[7:0] **If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 1,** R30[31:24] = Byte3{Nibble7,Nibble6} R30[23:16] = Byte2{Nibble5,Nibble4} R30[15:8] = Byte1{Nibble3,Nibble2} R30[7:0] = Byte0{Nibble1,Nibble0} **0x1: If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 0,** R30[15:8] = Byte0{Nibble1,Nibble0} R30[7:0] = Byte1{Nibble3,Nibble2} R30[31:24] = TX_MASK[7:0] R30[23:16] = TX_MASK[15:8] **If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 1,** (ONLY SUPPORT 32bit push) R30[31:24] = Byte0{Nibble1,Nibble0} R30[23:16] = Byte1{Nibble3,Nibble2} R30[15:8] = Byte2{Nibble5,Nibble4} R30[7:0] = Byte3{Nibble7,Nibble6 **Note Nibble0 is the first nibble received.**	RW	0x0
2	TX_EN_MODE	Enables transmit self clear on TX_EOF event. **Note that iep.cmp[3] must be set before transmission will start for TX0, and iep_cmp[4] for TX1. This is a new dependency, in addition to TX L1 FIFO not empty and TX_START_DELAY expiration, to start transmission.** 0x0: Disable 0x1: Enable, TX_ENABLE will be clear for a TX_EOF event by itself.	RW	0x0
1	TX_AUTO_PREAMBLE	Transmit data auto-preamble. 0x0: PRU will provide full preamble 0x1: TX FIFO will insert pre-amble automatically **Note: the TX FIFO does not get preloaded with the preamble until the first write occurs. This can cause the latency to be larger the min latency.**	RW	0x0
0	TX_ENABLE	Enables transmit traffic on TX PORT. If TX_EN_MODE is set, then TX_ENABLE will self clear during a TX_EOF event. Note Software can use this to pre-fill the TX FIFO and then start the TX frame during non-ECS operations. 0x0: TX PORT is disabled/stopped immediately 0x1: TX PORT is enabled and the frame will start once the IPG counter expired and TX Start Delay counter has expired	RW	0x0

Table 30-359. Register Call Summary for Register PRUSS_MII_RT_TXCFG0

PRU-ICSS MII RT Module

- Introduction: [0]
- TX Data Path Options to TX L1 FIFO: [1] [2] [3] [4] [5] [6]
- PRU R31 Command Interface: [7] [8]
- Nibble and Byte Order: [9] [10] [11] [12] [13] [14] [15]
- PRUSS_MII_RT Register Summary: [16] [17]
- PRUSS_MII_RT Register Description: [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29]

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRU-ICSS Industrial Ethernet Timer Features:

Table 30-360. PRUSS_MII_RT_TXCFG1

Address Offset	0x0000 0014		
Physical Address	0x4B23 2014 0x4B2B 2014	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII TXCFG 1 REGISTER This register contains the configuration variables for the transmit path on the MII interface port 1. PRUSS_MII_RT_TXCFG1 is attached to Port TX1. PRUSS_MII_RT_TXCFG1 controls which PRU is selected for TX1		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				TX_CLK_DELAY				RESERVED				TX_START_DELAY				RESERVED				TX_32_MODE_EN	RESERVED	TX_AUTO_SEQUENCE	TX_MUX_SEL	RESERVED				TX_BYTE_SWAP	TX_EN_MODE	TX_AUTO_PREAMBLE	TX_ENABLE

Bits	Field Name	Description	Type	Reset
31	RESERVED		R	0
30:28	TX_CLK_DELAY	In order to guarantee the MII_RT IO timing values published in the device data manual, the PRUSS_GICLK clock must be configured for 200MHz and TX_CLK_DELAY must be set to 6h.	RW	0x0
27:26	RESERVED		R	0x0
25:16	TX_START_DELAY	Defines the minimum time interval (delay) between receiving the RXDV for the current frame and the start of the transmit interface sending data to the MII interface. Delay value is in units of MII_RT clock cycles, which uses the PRUSS_GICLK (default is 200MHz, or 5ns). Default TX_START_DELAY value is 320ns, which is optimized for minimum latency at 16 bit processing. Counter is started with RX_DV signal going active. Transmit interface stops sending data when no more data is written into transmit interface by PRU along with TX_EOF marker bit set. If the TX FIFO has data when the delay expires, then TX will start sending data. But if the TX FIFO is empty, it will not start until the TX FIFO is not empty. It is possible to overflow the TX FIFO with the max delay setting when auto-forwarding is enabled since the time delay is larger than the amount of data it needs to store. As long as TX L1 FIFO overflows, software will need to issue a TX_RESET to reset the TX FIFO. The total delay is 96-byte times (size of TX FIFO), but you need to allow delays for synchronization. Do to this fact, the maximum delay should be 80ns less when auto forwarding is enabled. Therefore, 0x3F0 is the maximum in this configuration.	RW	0x40
15:12	RESERVED		R	0x0
11	TX_32_MODE_EN	0x0: Disable 32-bit Data Push mode. 0x1: Enable 32-bit, 16-bit, and 8-bit Data Push mode with TX_MASK disabled. In this mode, the internal PRU R30 byte write strobes are used and not the R31 CMD TX_PUSH mode. Any update to R30 will trigger an TX PUSH. See Table 30-346 .	RW	0x0
10	RESERVED		RW	0x0

Bits	Field Name	Description	Type	Reset
9	TX_AUTO_SEQUENCE	Enables transmit auto-sequence. Note the transmit data source is determined by TX_MUX_SEL setting. 0x0: Disable 0x1: Enable, transmit state machine based on events on receiver path that is connected to the respective transmitter. Also, the masking logic is disabled and only the MII data is used.	RW	0x0
8	TX_MUX_SEL	Selects transmit data source. The default/reset setting for TX Port 0 is 1. This setting permits MII TX Port 0 to receive data from PRU1 and the MII TX Port 1 which is connected to PRU0 by default. 0x0: Data from PRU0 (default for PRUSS_MII_RT_TXCFG1) 0x1: Data from PRU1 (default for PRUSS_MII_RT_TXCFG0)	RW	0x0
7:4	RESERVED		R	0x0
3	TX_BYTE_SWAP	Defines the order of Byte0/1 placement for TX R30. This bit must be selected/updated when the port is disabled or there is no traffic. 0x0: If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 0, R30[15:8] = Byte1{Nibble3,Nibble2} R30[7:0] = Byte0{Nibble1,Nibble0} R30[31:24] = TX_MASK[15:8] R30[23:16] = TX_MASK[7:0] If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 1, R30[31:24] = Byte3{Nibble7,Nibble6} R30[23:16] = Byte2{Nibble5,Nibble4} R30[15:8] = Byte1{Nibble3,Nibble2} R30[7:0] = Byte0{Nibble1,Nibble0} 0x1: If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 0, R30[15:8] = Byte0{Nibble1,Nibble0} R30[7:0] = Byte1{Nibble3,Nibble2} R30[31:24] = TX_MASK[7:0] R30[23:16] = TX_MASK[15:8] If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 1, (ONLY SUPPORT 32bit push) R30[31:24] = Byte0{Nibble1,Nibble0} R30[23:16] = Byte1{Nibble3,Nibble2} R30[15:8] = Byte2{Nibble5,Nibble4} R30[7:0] = Byte3{Nibble7,Nibble6 **Note Nibble0 is the first nibble received.**	RW	0x0
2	TX_EN_MODE	Enables transmit self clear on TX_EOF event. **Note that iep.cmp[3] must be set before transmission will start for TX0, and iep_cmp[4] for TX1. This is a new dependency, in addition to TX L1 FIFO not empty and TX_START_DELAY expiration, to start transmission.** 0x0: Disable 0x1: Enable, TX_ENABLE will be clear for a TX_EOF event by itself.	RW	0x0
1	TX_AUTO_PREAMBLE	Transmit data auto-preamble. 0x0: PRU will provide full preamble 0x1: TX FIFO will insert pre-amble automatically **Note: the TX FIFO does not get preloaded with the preamble until the first write occurs. This can cause the latency to be larger the min latency.**	RW	0x0
0	TX_ENABLE	Enables transmit traffic on TX PORT. If TX_EN_MODE is set, then TX_ENABLE will self clear during a TX_EOF event. Note Software can use this to pre-fill the TX FIFO and then start the TX frame during non-ECS operations. 0x0: TX PORT is disabled/stopped immediately 0x1: TX PORT is enabled and the frame will start once the IPG counter expired and TX Start Delay counter has expired	RW	0x0

Table 30-361. Register Call Summary for Register PRUSS_MII_RT_TXCFG1

PRU-ICSS MII RT Module • PRUSS_MII_RT Register Summary: [2] [3] • PRUSS_MII_RT Register Description: [4] [5] [6] [7]
PRU-ICSS Industrial Ethernet Peripheral (IEP) • PRU-ICSS Industrial Ethernet Timer Features:

Table 30-362. PRUSS_MII_RT_TX_CRC0

Address Offset	0x0000 0020		
Physical Address	0x4B23 2020 0x4B2B 2020	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII TXCRC 0 REGISTER It contains CRC32 which PRU0 reads		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX_CRC																															

Bits	Field Name	Description	Type	Reset
31:0	TX_CRC	FCS (CRC32) data can be read by PRU for diagnostics. It is only valid after 6 clocks after a TX_CRC_HIGH command is given.	R	0x0

Table 30-363. Register Call Summary for Register PRUSS_MII_RT_TX_CRC0

PRU-ICSS MII RT Module • CRC Computation: [2] • PRU R31 Command Interface: [3] • PRUSS_MII_RT Register Summary: [4] [5]

Table 30-364. PRUSS_MII_RT_TX_CRC1

Address Offset	0x0000 0024		
Physical Address	0x4B23 2024 0x4B2B 2024	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII TXCRC 1 REGISTER It contains CRC32 which PRU1 reads		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX_CRC																															

Bits	Field Name	Description	Type	Reset
31:0	TX_CRC	FCS (CRC32) data can be read by PRU for diagnostics. It is only valid after 6 clocks after a TX_CRC_HIGH command is given.	R	0x0

Table 30-365. Register Call Summary for Register PRUSS_MII_RT_TX_CRC1

PRU-ICSS MII RT Module • PRUSS_MII_RT Register Summary: [2] [3]

Table 30-366. PRUSS_MII_RT_TX_IPG0

Address Offset	0x0000 0030		
Physical Address	0x4B23 2030 0x4B2B 2030	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII TXIPG 0 REGISTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						TX_IPG									

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x0
9:0	TX_IPG	Defines the minimum of transmit Inter Packet Gap (IPG) which is the number of PRUSS_GICLK cycles between the de-assertion of TX_EN and the assertion of TX_EN. The start of the TX will get delayed when the incoming packet IPG is less than defined minimum value. In general, software should program in increments of 8, 40ns to insure the extra delays takes effect.	RW	0x28

Table 30-367. Register Call Summary for Register PRUSS_MII_RT_TX_IPG0

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]

Table 30-368. PRUSS_MII_RT_TX_IPG1

Address Offset	0x0000 0034		
Physical Address	0x4B23 2034 0x4B2B 2034	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII TXIPG 1 REGISTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						TX_IPG									

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x0
9:0	TX_IPG	Defines the minimum of transmit Inter Packet Gap (IPG) which is the number of PRUSS_GICLK cycles between the de-assertion of TX_EN and the assertion of TX_EN. The start of the TX will get delayed when the incoming packet IPG is less than defined minimum value. In general, software should program in increments of 8, 40ns to insure the extra delays takes effect.	RW	0x28

Table 30-369. Register Call Summary for Register PRUSS_MII_RT_TX_IPG1

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]

Table 30-370. PRUSS_MII_RT_PRS0

Address Offset	0x0000 0038		
Physical Address	0x4B23 2038 0x4B2B 2038	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII PORT STATUS 0 REGISTER		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														MII_CRS	MII_COL

Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0
1	MII_CRS	Read the current state of pr1_mii0_crs	R	0x0
0	MII_COL	Read the current state of pr1_mii0_col	R	0x0

Table 30-371. Register Call Summary for Register PRUSS_MII_RT_PRS0

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]

Table 30-372. PRUSS_MII_RT_PRS1

Address Offset	0x0000 003C		
Physical Address	0x4B23 203C 0x4B2B 203C	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII PORT STATUS 1 REGISTER		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														MII_CRS	MII_COL

Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0
1	MII_CRS	Read the current state of pr1_mii1_crs	R	0x0
0	MII_COL	Read the current state of pr1_mii1_col	R	0x0

Table 30-373. Register Call Summary for Register PRUSS_MII_RT_PRS1

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]

Table 30-374. PRUSS_MII_RT_RX_FRMS0

Address Offset	0x0000 0040		
Physical Address	0x4B23 2040 0x4B2B 2040	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII RXFRMS 0 REGISTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX_MAX_FRM																RX_MIN_FRM															

Bits	Field Name	Description	Type	Reset
31:16	RX_MAX_FRM	Defines the maximum received frame count. If the total byte count of received frame is more than defined value, RX_MAX_FRM_ERR will get set. 0x0 = 1 byte after SFD and including CRC N= N+1 bytes after SFD and including CRC Note if the incoming frame is truncated at the marker, RX_CRC and RX_NIBBLE_ODD will not get asserted.	RW	0x5F1
15:0	RX_MIN_FRM	Defines the minimum received frame count. If the total byte count of received frame is less than defined value, RX_MIN_FRM_ERR will get set. 0x0 = 1 byte after SFD and including CRC N=N+1 bytes after SFD and including CRC	RW	0x3F

Table 30-375. Register Call Summary for Register PRUSS_MII_RT_RX_FRMS0

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]

Table 30-376. PRUSS_MII_RT_RX_FRMS1

Address Offset	0x0000 0044		
Physical Address	0x4B23 2044 0x4B2B 2044	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII RXFRMS 1 REGISTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX_MAX_FRM																RX_MIN_FRM															

Bits	Field Name	Description	Type	Reset
31:16	RX_MAX_FRM	Defines the maximum received frame count. If the total byte count of the received frame is more than defined value, RX_MAX_FRM_ERR will get set. 0x0 = 1 byte after SFD and including CRC N= N+1 bytes after SFD and including CRC Note if the incoming frame is truncated at the marker, RX_CRC and RX_NIBBLE_ODD will not get asserted.	RW	0x5F1
15:0	RX_MIN_FRM	Defines the minimum received frame count. If the total byte count of received frame is less than defined value, RX_MIN_FRM_ERR will get set. 0x0 = 1 byte after SFD and including CRC N=N+1 bytes after SFD and including CRC	RW	0x3F

Table 30-377. Register Call Summary for Register PRUSS_MII_RT_RX_FRMS1

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]

Table 30-378. PRUSS_MII_RT_RX_PCNT0

Address Offset	0x0000 0048		
Physical Address	0x4B23 2048 0x4B2B 2048	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII RXPCNT 0 REGISTER		

Table 30-378. PRUSS_MII_RT_RX_PCNT0 (continued)

Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								RX_MAX_PCNT				RX_MIN_PCNT			

Bits	Field Name	Description	Type	Reset
31:8	RESERVED		R	0x000000
7:4	RX_MAX_PCNT	Defines the maximum number of nibbles until the start of frame delimiter (SFD) event occurred (i.e. matches 0xD5). RX_MAX_PRE_COUNT_ERR will be set if the preamble counts more than the value of RX_MAX_PCNT. If the SFD does not occur within 16 nibbles, the error will assert and the incoming frame will be truncated. 0x0: Disabled 0x1: Reserved 0x2: 4th nibble needs to have built 0xD5 0xe: 16th nibble needs to have built 0xD5 **Note the 16th nibble is transmitted.** **Note for firmware enabling preamble error detection, it is recommended to keep RX_MAX_PCNT disabled (0x0). Otherwise, hardware can truncate a valid frame with too long of a preamble.**	RW	0xE
3:0	RX_MIN_PCNT	Defines the minimum number of nibbles until the start of frame delimiter (SFD) event occurred, which is matched the value 0xD5. RX_MIN_PRE_COUNT_ERR will be set if the preamble counts less than the value of RX_MIN_PCNT. 0x0 Disabled 0x1 1 0x5 before 0xD5 0x2 2 0x5 before 0xD5 N min of N 0x5 before 0xD5 **Note it does not need to be "0x5"**	RW	0x1

Table 30-379. Register Call Summary for Register PRUSS_MII_RT_RX_PCNT0

PRU-ICSS MII RT Module

- RX MII Submodule Overview: [2]
- PRUSS_MII_RT Register Summary: [3] [4]

Table 30-380. PRUSS_MII_RT_RX_PCNT1

Address Offset	0x0000 004C		
Physical Address	0x4B23 204C 0x4B2B 204C	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII RXPCNT 1 REGISTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								RX_MAX_PCNT				RX_MIN_PCNT			

Bits	Field Name	Description	Type	Reset
31:8	RESERVED		R	0x000000
7:4	RX_MAX_PCNT	Defines the maximum number of nibbles until the start of frame delimiter (SFD) event occurred (i.e. matches 0xD5). RX_MAX_PRE_COUNT_ERR will be set if the preamble counts more than the value of RX_MAX_PCNT. If the SFD does not occur within 16 nibbles, the error will assert and the incoming frame will be truncated. 0x0: Disabled 0x1: Reserved 0x2: 4th nibble needs to have built 0xD5 0xe: 16th nibble needs to have built 0xD5 **Note the 16th nibble is transmitted** Note for firmware enabling preamble error detection, it is recommended to keep RX_MAX_PCNT disabled (0x0). Otherwise, hardware can truncate a valid frame with too long of a preamble.	RW	0xE
3:0	RX_MIN_PCNT	Defines the minimum number of nibbles until the start of frame delimiter (SFD) event occurred, which is matched the value 0xD5. RX_MIN_PRE_COUNT_ERR will be set if the preamble counts less than the value of RX_MIN_PCNT. 0x0 Disabled 0x1: 1 0x5 before 0xD5 0x2: 2 0x5 before 0xD5 N: N 0x5 before 0xD5 **Note it does not need to be "0x5"**	RW	0x1

Table 30-381. Register Call Summary for Register PRUSS_MII_RT_RX_PCNT1

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]

Table 30-382. PRUSS_MII_RT_RX_ERR0

Address Offset	0x0000 0050		
Physical Address	0x4B23 2050 0x4B2B 2050	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII RXERR 0 REGISTER		
Type	RWr1Clr		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												RX_MAX_FRM_ERR	RX_MIN_FRM_ERR	RX_MAX_PCNT_ERR	RX_MIN_PCNT_ERR

Bits	Field Name	Description	Type	Reset
31:4	RESERVED		R	0x00000
3	RX_MAX_FRM_ERR	Error status of received frame is more than the value of RX_MAX_FRM. 0x0: No error occurred 0x1: Error occurred Write 1 to Clear	RWr1Clr	0x0
2	RX_MIN_FRM_ERR	Error status of received frame is less than the value of RX_MIN_FRM. 0x0: No error occurred 0x1: Error occurred Write 1 to Clear	RWr1Clr	0x0
1	RX_MAX_PCNT_ERR	Error status of received preamble nibble is more than the value of RX_MAX_PCNT. 0x0: No error occurred 0x1: Error occurred Write 1 to Clear	RWr1Clr	0x0
0	RX_MIN_PCNT_ERR	Error status of received preamble nibble is less than the value of RX_MIN_PCNT. 0x0: No error occurred 0x1: Error occurred Write 1 to Clear	RWr1Clr	0x0

Table 30-383. Register Call Summary for Register PRUSS_MII_RT_RX_ERR0

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]

Table 30-384. PRUSS_MII_RT_RX_ERR1

Address Offset	0x0000 0054		
Physical Address	0x4B23 2054 0x4B2B 2054	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII RXERR 1 REGISTER		
Type	RWr1Clr		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												RX_MAX_FRM_ERR	RX_MIN_FRM_ERR	RX_MAX_PCNT_ERR	RX_MIN_PCNT_ERR

Bits	Field Name	Description	Type	Reset
31:4	RESERVED		R	0x00000
3	RX_MAX_FRM_ERR	Error status of received frame is more than the value of RX_MAX_FRM_CNT. 0x0: No error occurred 0x1: Error occurred Write 1 to Clear	RW	0x0
2	RX_MIN_FRM_ERR	Error status of received frame is less than the value of RX_MIN_FRM_CNT. 0x0: No error occurred 0x1: Error occurred Write 1 to Clear	RW	0x0

Bits	Field Name	Description	Type	Reset
1	RX_MAX_PCNT_ERR	Error status of received preamble nibble is more than the value of RX_MAX_PCNT. 0x0: No error occurred 0x1: Error occurred Write 1 to Clear	RW	0x0
0	RX_MIN_PCNT_ERR	Error status of received preamble nibble is less than the value of RX_MIN_PCNT. 0x0: No error occurred 0x1: Error occurred Write 1 to Clear	RW	0x0

Table 30-385. Register Call Summary for Register PRUSS_MII_RT_RX_ERR1

PRU-ICSS MII RT Module
- PRUSS_MII_RT Register Summary: [2] [3]

Table 30-386. PRUSS_MII_RT_RXFLV0

Address Offset	0x0000 0060		
Physical Address	0x4B23 2060 0x4B2B 2060	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII PRUSS_MII_RT_RXFLV0 REGISTER This register defines the number of valid bytes in the RX FIFO MII interface port 0. PRUSS_MII_RT_RXFLV0 is attached to Port RX0. PRUSS_MII_RT_RXFLV0 controls which PRU is selected for RX0		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								RX_FIFO_LEVEL							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED		R	0
7:0	RX_FIFO_LEVEL	Define the number of valid bytes in the RX FIFO 0 = empty 1 = 1 Byte/ 2 Nibbles 2 = 2 Byte/ 4 Nibble ... 32 = 32 Bytes/ 64 Nibbles	R	0x0

Table 30-387. Register Call Summary for Register PRUSS_MII_RT_RXFLV0

PRU-ICSS MII RT Module
- PRUSS_MII_RT Register Summary: [2] [3]
- PRUSS_MII_RT Register Description: [4] [5] [6]

Table 30-388. PRUSS_MII_RT_RXFLV1

Address Offset	0x0000 0064		
Physical Address	0x4B23 2064 0x4B2B 2064	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII PRUSS_MII_RT_RXFLV1 REGISTER This register defines the number of valid bytes in the RX FIFO MII interface port 1. PRUSS_MII_RT_RXFLV1 is attached to Port RX1. PRUSS_MII_RT_RXFLV1 controls which PRU is selected for RX1		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								RX_FIFO_LEVEL							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED		R	0
7:0	RX_FIFO_LEVEL	Define the number of valid bytes in the RX FIFO 0 = empty 1 = 1 Byte/ 2 Nibbles 2 = 2 Byte/ 4 Nibble ... 32 = 32 Bytes/ 64 Nibbles	R	0x0

Table 30-389. Register Call Summary for Register PRUSS_MII_RT_RXFLV1

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]
- PRUSS_MII_RT Register Description: [4] [5] [6]

Table 30-390. PRUSS_MII_RT_TXFLV0

Address Offset	0x0000 0068		
Physical Address	0x4B23 2068 0x4B2B 2068	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII PRUSS_MII_RT_TXFLV0 REGISTER This register defines the number of valid bytes in the TX FIFO MII interface port 0. PRUSS_MII_RT_TXFLV0 is attached to Port TX0. PRUSS_MII_RT_TXFLV0 controls which PRU is selected for TX0.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								TX_FIFO_LEVEL							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED		R	0
7:0	TX_FIFO_LEVEL	Define the number of valid bytes in the TX FIFO 0 = empty 1 = 1 Nibbles 2 = 1 Byte/ 2 Nibble ... 128 = 64 Bytes/ 128 Nibbles ... 192 = 96 Bytes/ 192 Nibbles	R	0x0

Table 30-391. Register Call Summary for Register PRUSS_MII_RT_TXFLV0

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]
- PRUSS_MII_RT Register Description: [4] [5] [6]

Table 30-392. PRUSS_MII_RT_TXFLV1

Address Offset	0x0000 006C		
Physical Address	0x4B23 206C 0x4B2B 206C	**Instance**	PRUSS1_MII_RT PRUSS2_MII_RT
Description	MII PRUSS_MII_RT_TXFLV1 REGISTER This register defines the number of valid bytes in the TX FIFO MII interface port 1. PRUSS_MII_RT_TXFLV1 is attached to Port TX1. PRUSS_MII_RT_TXFLV1 controls which PRU is selected for TX1.		

Table 30-392. PRUSS_MII_RT_TXFLV1 (continued)

Type	R

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								TX_FIFO_LEVEL							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED		R	0
7:0	TX_FIFO_LEVEL	Define the number of valid bytes in the TX FIFO 0 = empty 1 = 1 Nibbles 2 = 1 Byte/ 2 Nibble ... 128 = 64 Bytes/ 128 Nibbles ... 192 = 96 Bytes/ 192 Nibbles	R	0x0

Table 30-393. Register Call Summary for Register PRUSS_MII_RT_TXFLV1

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]
- PRUSS_MII_RT Register Description: [4] [5] [6]

30.1.10 PRU-ICSS MII MDIO Module

This section describes the PRU-ICSS1 and PRU-ICSS2 integrated **MII management interface module - MII_MDIO** module (PRUSS1_MII_MDIO / PRUSS2_MII_MDIO, respectively).

30.1.10.1 PRU-ICSS MII MDIO Overview

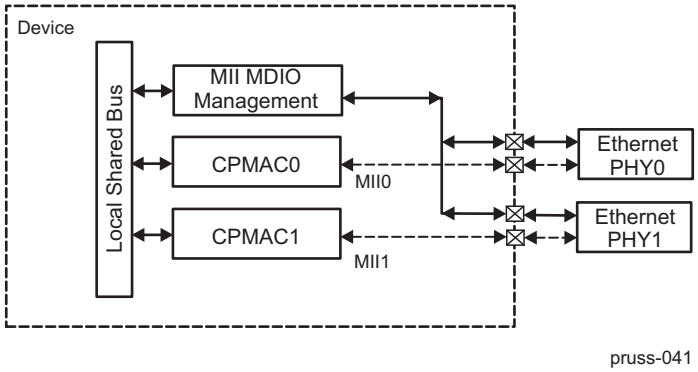
The following features are supported:

- Supports up to 32 PHY addresses.
- Two user access registers to control and monitor up to two PHYs simultaneously.
- Slave interface for configuration and control (MII RT MDIO CFG)

The PRU-ICSS MII MDIO management I/F module implements the **802.3 serial management interface** to interrogate and control two Ethernet PHYs simultaneously using a shared two-wire bus. Figure 1 shows a device with two MACs, each connected to an Ethernet PHY, being managed by the MII interface module using a shared bus.

The Figure 30-52 gives an overview of the MII MDIO management interface.

Figure 30-52. Device PRU-ICSS MII MDIO Management Interface Overview



30.1.10.2 PRU-ICSS MII MDIO Functional Description

The MII Management interface incorporates:

- **MDIO Registers** - Host interaction with this module is facilitated through the registers in this block.
- **Control and Schedule** - The control and register logic in the MII Management Interface module contain the state machine and scheduling logic which control the wire side operation.
- **MDIO Interface** - The MDIO interface block provides the serial interface to the MDIO interface.

The MDIO logic is fully synchronous to the PRU-ICSS local shared bus clock.

30.1.10.2.1 MII MDIO Management Interface Frame Formats

The below Table 30-394 shows the read and write format of the 32-bit MII Management interface frames, respectively.

Table 30-394. MII MDIO Frame Formats

Pre-amble	Start Delimiter	Operation Code	PHY Address	Register Address	Turnaround	Data
MDIO Read Frame Format						
0xFFFFFFFF	01	10	AAAAA	RRRRR	Z0	DDDD.DDDD.DDDD.DDDD
MDIO Write Frame Format						
0xFFFFFFFF	01	00	AAAAA	RRRRR	10	DDDD.DDDD.DDDD.DDDD

The default or idle state of the two wire serial interface is a logic one. All tri-state drivers should be disabled and the PHY's pull-up resistor will pull the **MDIO** line to a logic one. Prior to initiating any other transaction, the station management entity shall send a preamble sequence of 32 contiguous logic one bits on the **MDIO** line with 32 corresponding cycles on **MDCLK** to provide the PHY with a pattern that it can use to establish synchronization. A PHY shall observe a sequence of 32 contiguous logic one bits on **MDIO** with 32 corresponding **MDCLK** cycles before it responds to any other transaction.

Preamble
The start of a frame is indicated by a preamble, which consists of a sequence of 32 contiguous bits all of which are a "1". This sequence provides the Ethernet PHY a pattern to use to establish synchronization.

Start Delimiter
The preamble is followed by the start delimiter which is indicated by a "01" pattern. The pattern assures transitions from the default logic one state to zero and back to one.

Operation Code
The operation code for a read is "10", while the operation code for a write is a "00".

Ethernet PHY Address
The PHY address is 5 bits allowing 32 unique values. The first bit transmitted is the MSB of the PHY address.

Register Address
The Register address is 5 bits allowing 32 registers to be addressed within each PHY. Refer to the 10/100 PHY address map for addresses of individual registers.

Turnaround
An idle bit time during which no device actively drives the MDIO signal shall be inserted between the register address field and the data field of a read frame in order to avoid contention. During a read frame, the PHY shall drive a zero bit onto MDIO for the first bit time following the idle bit and preceding the Data field. During a write frame, this field shall consist of a one bit followed by a zero bit.

Data
The Data field is 16 bits. The first bit transmitted and received is the MSB of the data word.

The Table 30-395 shows the PRU-ICSS1 / PRU-ICSS2 MII MDIO signals and their availability at the device boundary.

Table 30-395. PRU-ICSS MII MDIO Control and Interface Signals

MDIO Control Signals			
Pin Name	**Type**	**Available as device I/O**	**Function**
MDIO_LINKINT[1:0]	O	N.A.	Serial interface link change interrupt. Indicates a change in the state of the PHY link.
MDIO_USERINT[1:0]	O	N.A.	Serial interface user command event complete interrupt.
MDIO Interface Signals			
Pin Name	**Type**	**Available as device I/O**	**Function**
MDIO_I	I	device bidi **pr1_mdio_data** and pr2_mdio_mdclk pin in input mode	Serial data input
MDIO_O	O	device bidi **pr1_mdio_data** pr2_mdio_mdclk pin in output mode	Serial data output
MDIO_OE_N	O	N.A.	Serial data output enable. Asserted "0" when data output is valid
MDCLK_O	O	device output - **pr1_mdio_mdclk** pr2_mdio_mdclk	Serial clock output
MLINK_I[1:0]	I	N.A.	Optional link status inputs from PHY. Each input is connected to a single PHY. Unused inputs are tied '0'.

30.1.10.2.2 PRU-ICSS MII MDIO Interractions

The MII Management I/F will remain idle until enabled by setting the **enable** bit in the **MDIO Control** register. The MII Management I/F will then continuously poll the link status from within the Generic Status Register of all possible 32 PHY addresses in turn recording the results in the **MDIOLink** register. The link status of two of the 32 possible PHY addresses can also be determined using the **MLINK** pin inputs. The linksel bit in the **MDIOUserPhySel** register determines the status input that is used. A change in the link status of the two PHYs being monitored will set the appropriate bit in the **MDIOLinkIntRaw** register and the **MDIOLinkIntMasked** register, if enabled by the **linkint_enable** bit in the **MDIOUserPhySel** register.

The **MDIOAlive** register is updated by the MII Management I/F module if the PHY acknowledged the read of the generic status register. In addition, any PHY register read transactions initiated by the host also cause the **MDIOAlive** register to be updated.

At any time, the host can define a transaction for the MII Management interface module to undertake using the **data**, **Ethernet PHY address**, **register address**, and **write** fields in a **MDIOUserAccess** register. When the host sets the go bit in this register, the MII Management interface module will begin the transaction without any further intervention from the host. Upon completion, the MII Management interface will clear the go bit and set the **userintraw** bit in the **MDIOUserIntRaw** register corresponding to the **MDIOUserAccess** register being used. The corresponding bit in the **MDIOUserIntMasked** register may also be set depending on the mask setting in the **MDIOUserIntMaskSet** and **MDIOUserIntMaskClr** registers. A round-robin arbitration scheme is used to schedule transactions which may queued by the host in different **MDIOUserAccess** registers. The host should check the status of the go bit in the **MDIOUserAccess** register before initiating a new transaction to ensure that the previous transaction has completed. The host can use the ack bit in the **MDIOUserAccess** register to determine the status of a read transaction.

It is necessary for software to use the MII Management interface module to setup the autonegotiation parameters of each PHY attached to a MAC port, retrieve the negotiation results, and setup the **MACControl** register in the corresponding MAC.

30.1.10.2.3 PRU-ICSS MII MDIO Interrupts

The MII Management interface state machine will assert the **MDIO_LINKINT** signals if there is a change in the link state of the Ethernet PHY corresponding to the address in the **phyadr_mon** field of the **MDIOUserPhySel** register and the corresponding **linkint_enable** bit is set. The **MDIO_LINKINT** event is also captured in the **MDIOLinkIntMasked** register. **MDIO_LINKINT[0]** and **MDIO_LINKINT[1]** correspond to the **MDIOUserPhySel0** and **MDIOUserPhySel1** registers, respectively.

When the "GO" bit in the **MDIOUserAccess** registers transitions from ‘1’ to ‘0’, indicating the completion of a user access, and the corresponding **userintmaskset** bit in the

MDIOUserIntMaskSet register is set, the **MDIO_USERINT** signal is asserted ‘1’. The **MDIO_USERINT** event is also captured in the **MDIOUserIntMasked** register. **MDIO_USERINT[0]** and **MDIO_USERINT[1]** correspond to the **MDIOUserAccess0** and **MDIOUserAccess1** registers, respectively.

30.1.10.3 PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface

To facilitate transmission and reception of serial management frames, the host has to perform the following operations:

- Configure the **preamble** and **clkdiv** fields in the **MDIOControl** register.
- Enable the VBUS MII management module by setting the **enable** bit in the MDIOControl register. If Byte access is being used, the **enable** bit should be written last.
- The **MDIOAlive** register can be read after a delay to determine which Ethernet PHYs responded.
- Setup the appropriate PHY addresses in the **MDIOUserPhySel** registers.
- Setup the appropriate **linkint_enable** bit in the **MDIOUserPhySel** register.
- Setup the appropriate **linksel** fields in the **MDIOUserPhySel** register.
- Setup the appropriate **userintmaskset** bits in the **MDIOUserIntMaskSet** register.
- **To write to**an Ethernet PHY register the host should first check to ensure that the go bit in a **MDIOUserAcess** register is cleared. The **GO**, **write**, **regadr**, **phyadr** and data fields in that

MDIOUserAccess register can then be updated to be appropriate value. If byte access is being used, the go bit should be written last. The write operation to the PHY will be scheduled and completed by the module. Completion of the write operation can be determined by examining the go bit in the **MDIOUserAccess** register. It also results in a transition on the appropriate **MDIO_INT** signal and the corresponding bit in the **MDIOUserIntMasked** register based on the setting of the **MDIOUserIntMaskSet** register.

- **To read from** an Ethernet PHY register the host should first check to ensure that the "GO" bit in a **MDIOUserAccess** register bit is cleared. The **GO**, **regadr**, and **phyadr** fields in that **MDIOUserAccess** register can then be updated to the appropriate value The read data value will be available in the data field of the MDIOUserAccess register after the module completes the read operation on the serial bus. The completion of the read operation can be determined by examining the "GO" and "ACK" bits in the **MDIOUserAccess** register. It also results in a transition on the appropriate **MDIO_INT** signal and the corresponding bit in the **MDIOUserIntMasked** register based on the setting of the **MDIOUserIntMaskSet** register.
- The module de-asserts the **MDIO_USERINT** signal when the host writes to the appropriate **"userintmasked"** bit in the **MDIOUserIntMasked** register or the **userintraw** bit in the **MDIOUserIntRaw** register.
- The host can poll the **MDIOLink** register periodically or use the **MDIO_LINKINT** signals to determine the state of the serial interface to a particular Ethernet PHY.
- The module de-asserts the **MDIO_LINKINT** when the host writes to the appropriate linkintraw bit in the **MDIOLinkIntRaw** register or the **linkintmasked** bit in the **MDIOLinkIntMasked** register.

Table 30-396. Summary of the PRU-ICSS MII MDIO Functional Registers

Address Offset	Register Mnemonic	Register Name	Register Purpose
0x04	**MDIOControl**	PRUSS_MII_MDIO_CONTROL	Module control register
0x08	**MDIOAlive**	PRUSS_MII_MDIO_ALIVE	Ethernet PHY acknowledge status register
0x0c	**MDIOLink**	PRUSS_MII_MDIO_LINK	Ethernet PHY link status register
0x10	**MDIOLinkIntRaw**	PRUSS_MII_MDIO_LINKINTRAW	Link status change interrupt register (raw value)
0x14	**MDIOLinkIntMasked**	PRUSS_MII_MDIO_LINKINTMASKED	Link status change interrupt register (masked value)
0x18-0x1c	**Reserved**	-	Reserved
0x20	**MDIOUserIntRaw**	PRUSS_MII_MDIO_USERINTRAW	User command complete interrupt register (raw value)
0x24	**MDIOUserIntMasked**	PRUSS_MII_MDIO_USERINTMASKED	User command complete interrupt register (masked value)
0x28	**MDIOUserIntMaskSet**	PRUSS_MII_MDIO_USERINTMASKSET	User interrupt mask set register
0x2c	**MDIOUserIntMaskClr**	PRUSS_MII_MDIO_USERINTMASKCLR	User interrupt mask clear register
0x30 – 0x7c	**Reserved**	-	Reserved
0x80	**MDIOUserAccess0**	PRUSS_MII_MDIO_USERACCESS0	User access register 0
0x84	**MDIOUserPhySel0**	PRUSS_MII_MDIO_USERPHYSEL0	User PHY select register 0
0x88	**MDIOUserAccess1**	PRUSS_MII_MDIO_USERACCESS1	User access register 1
0x8c	**MDIOUserPhySel1**	PRUSS_MII_MDIO_USERPHYSEL1	User PHY select register 1

Table 30-396. Summary of the PRU-ICSS MII MDIO Functional Registers (continued)

Address Offset	Register Mnemonic	Register Name	Register Purpose
0x90 – 0xff	**Reserved**	-	Reserved

30.1.10.4 PRU-ICSS MII MDIO Module Register Manual

30.1.10.4.1 PRUSS_MII_MDIO Instance Summary

Table 30-397. PRUSS_MII_MDIO Instance Summary

Module Name	Base Address	Size
PRUSS1_MII_MDIO	0x4B23 2400	144 Bytes
PRUSS2_MII_MDIO	0x4B2B 2400	144 Bytes

30.1.10.4.2 PRUSS_MII_MDIO Registers

30.1.10.4.2.1 PRUSS_MII_MDIO Register Summary

Table 30-398. PRUSS1_MII_MDIO Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
PRUSS_MII_MDIO_VER	R	32	0x0000 0000	0x4B23 2400
PRUSS_MII_MDIO_CONTROL	RW	32	0x0000 0004	0x4B23 2404
PRUSS_MII_MDIO_ALIVE	RW	32	0x0000 0008	0x4B23 2408
PRUSS_MII_MDIO_LINK	R	32	0x0000 000C	0x4B23 240C
PRUSS_MII_MDIO_LINKINTRAW	RW	32	0x0000 0010	0x4B23 2410
PRUSS_MII_MDIO_LINKINTMASKED	RW	32	0x0000 0014	0x4B23 2414
PRUSS_MII_MDIO_USERINTRAW	RW	32	0x0000 0020	0x4B23 2420
PRUSS_MII_MDIO_USERINTMASKED	RW	32	0x0000 0024	0x4B23 2424
PRUSS_MII_MDIO_USERINTMASKSET	RW	32	0x0000 0028	0x4B23 2428
PRUSS_MII_MDIO_USERINTMASKCLR	RW	32	0x0000 002C	0x4B23 242C
PRUSS_MII_MDIO_USERACCESS0	RW	32	0x0000 0080	0x4B23 2480
PRUSS_MII_MDIO_USERPHYSEL0	RW	32	0x0000 0084	0x4B23 2484
PRUSS_MII_MDIO_USERACCESS1	RW	32	0x0000 0088	0x4B23 2488
PRUSS_MII_MDIO_USERPHYSEL1	RW	32	0x0000 008C	0x4B23 248C

Table 30-399. PRUSS2_MII_MDIO Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
PRUSS_MII_MDIO_VER	R	32	0x0000 0000	0x4B2B 2400
PRUSS_MII_MDIO_CONTROL	RW	32	0x0000 0004	0x4B2B 2404
PRUSS_MII_MDIO_ALIVE	RW	32	0x0000 0008	0x4B2B 2408
PRUSS_MII_MDIO_LINK	R	32	0x0000 000C	0x4B2B 240C
PRUSS_MII_MDIO_LINKINTRAW	RW	32	0x0000 0010	0x4B2B 2410
PRUSS_MII_MDIO_LINKINTMASKED	RW	32	0x0000 0014	0x4B2B 2414
PRUSS_MII_MDIO_USERINTRAW	RW	32	0x0000 0020	0x4B2B 2420
PRUSS_MII_MDIO_USERINTMASKED	RW	32	0x0000 0024	0x4B2B 2424
PRUSS_MII_MDIO_USERINTMASKSET	RW	32	0x0000 0028	0x4B2B 2428

Table 30-399. PRUSS2_MII_MDIO Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
PRUSS_MII_MDIO_USERINTMASKCLR	RW	32	0x0000 002C	0x4B2B 242C
PRUSS_MII_MDIO_USERACCESS0	RW	32	0x0000 0080	0x4B2B 2480
PRUSS_MII_MDIO_USERPHYSEL0	RW	32	0x0000 0084	0x4B2B 2484
PRUSS_MII_MDIO_USERACCESS1	RW	32	0x0000 0088	0x4B2B 2488
PRUSS_MII_MDIO_USERPHYSEL1	RW	32	0x0000 008C	0x4B2B 248C

30.1.10.4.2.2 PRUSS_MII_MDIO Register Description

Table 30-400. PRUSS_MII_MDIO_VER

Address Offset	0x0000 0000		
Physical Address	0x4B23 2400 0x4B2B 2400	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	MDIO MODULE VERSION REGISTER		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REVISION																															

Bits	Field Name	Description	Type	Reset
31:0	REVISION	IP Revision.	R	0x-(1)

(1) TI Internal Data

Table 30-401. Register Call Summary for Register PRUSS_MII_MDIO_VER

PRU-ICSS MII MDIO Module

- PRUSS_MII_MDIO Register Summary: [0] [1]

Table 30-402. PRUSS_MII_MDIO_CONTROL

Address Offset	0x0000 0004		
Physical Address	0x4B23 2404 0x4B2B 2404	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	MDIO MODULE CONTROL REGISTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDLE	ENABLE	RESERVED	HIGHEST_USER_CHANNEL					RESERVED				PREAMBLE	FAULT	FAULT_DETECT_ENABLE	INT_TEST_ENABLE	RESERVED	CLKDIV																

Bits	Field Name	Description	Type	Reset
31	IDLE	MDIO state machine IDLE. Set to 1 when the state machine is in the idle state.	R	0x1
30	ENABLE	Enable control. Writing a 1 to this bit enables the MDIO state machine, writing a 0 disables it. If the MDIO state machine is active at the time it is disabled, it will complete the current operation before halting and setting the idle bit. If using byte access, the enable bit has to be the last bit written in this register.	RW	0x0
29	RESERVED		R	0
28:24	HIGHEST_USER_CHANNEL	Highest user channel. This field specifies the highest useraccess channel that is available in the module and is currently set to 1. This implies that **MDIOUserAccess1** is the highest available user access channel.	R	0x1
23:21	RESERVED		R	0x0
20	PREAMBLE	Preamble disable. Writing a 1 to this bit disables this device from sending MDIO frame preambles.	RW	0x0
19	FAULT	Fault indicator. This bit is set to 1 if the MDIO pins fail to read back what the device is driving onto them. This indicates a physical layer fault and the module state machine is reset. Writing a 1 to it clears this bit.	RW	0x0
18	FAULT_DETECT_ENABLE	Fault detect enable. This bit has to be set to 1 to enable the physical layer fault detection.	RW	0x0
17	INT_TEST_ENABLE	Interrupt test enable. This bit can be set to 1 to enable the host to set the userint and linkint bits for test purposes.	RW	0x0
16	RESERVED		R	0
15:0	CLKDIV	Clock Divider. This field specifies the division ratio between CLK and the frequency of MDCLK. MDCLK is disabled when clkdiv is set to 0. MDCLK frequency = clk frequency/(clkdiv+1).	RW	0xff

Table 30-403. Register Call Summary for Register PRUSS_MII_MDIO_CONTROL

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-404. PRUSS_MII_MDIO_ALIVE

Address Offset	0x0000 0008		
Physical Address	0x4B23 2408 0x4B2B 2408	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	PHY ACKNOWLEDGE STATUS REGISTER		
Type	RWr1Clr		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALIVE																															

Bits	Field Name	Description	Type	Reset
31:0	ALIVE	MDIO Alive bitfield. Each of the 32 bits of this register is set if the most recent access to the PHY with address corresponding to the register bit number was acknowledged by the PHY, the bit is reset if the PHY fails to acknowledge the access. Both the user and polling accesses to a PHY will cause the corresponding alive bit to be updated. The alive bits are only meant to be used to give an indication of the presence or not of a PHY with the corresponding address. Writing a 1 to any bit will clear it, writing a 0 has no effect.	RWr1Clr	0x0

Table 30-405. Register Call Summary for Register PRUSS_MII_MDIO_ALIVE

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-406. PRUSS_MII_MDIO_LINK

Address Offset	0x0000 000C		
Physical Address	0x4B23 240C 0x4B2B 240C	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	PHY LINK STATUS REGISTER		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINK																															

Bits	Field Name	Description	Type	Reset
31:0	LINK	MDIO Link state. This register is updated after a read of the Generic Status Register of a PHY. The bit is set if the PHY with the corresponding address has link and the PHY acknowledges the read transaction. The bit is reset if the PHY indicates it does not have link or fails to acknowledge the read transaction. Writes to the register have no effect. In addition, the status of the two PHYs specified in the **MDIOUserPhySel** registers can be determined using the **MLINK** input pins. This is determined by the **linksel** bit in the **MDIOUserPhySel** register.	R	0x0

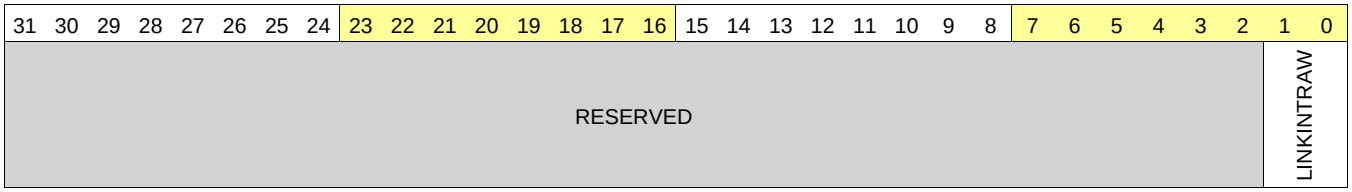
Table 30-407. Register Call Summary for Register PRUSS_MII_MDIO_LINK

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-408. PRUSS_MII_MDIO_LINKINTRAW

Address Offset	0x0000 0010		
Physical Address	0x4B23 2410 0x4B2B 2410	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	LINK STATUS CHANGE INTERRUPT REGISTER (RAW VALUE)		
Type	RWr1Clr		



Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0000 000
1:0	LINKINTRAW	MDIO link change event, raw value. When asserted '1', a bit indicates that there was an **MDIO link** change event (i.e. change in the MDIOLink register) corresponding to the PHY address in the **MDIOUserPhySel** register. **linkintraw[0] and linkintraw[1]** correspond to **MDIOUserPhySel0 and MDIOUserPhysel1**, respectively. Writing a '1' will clear the event and writing 0 has no effect. **If the int_test bit in the MDIOControl register is set, the host may set the linkintraw bits to a '1'.** This mode may be used for test purposes.	RWr1Clr	0x0

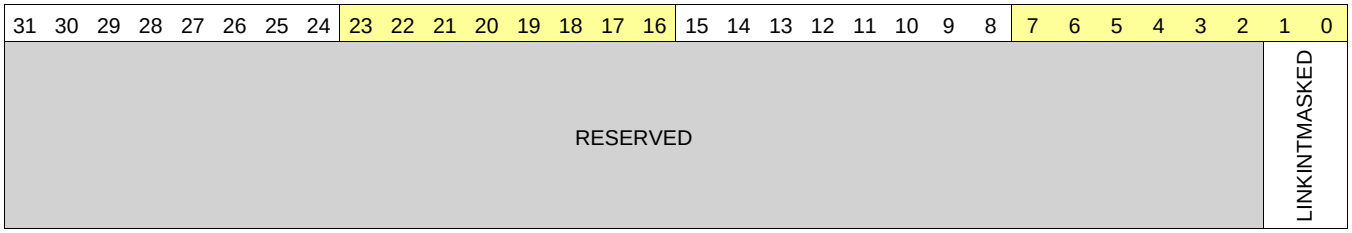
Table 30-409. Register Call Summary for Register PRUSS_MII_MDIO_LINKINTRAW

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-410. PRUSS_MII_MDIO_LINKINTMASKED

Address Offset	0x0000 0014		
Physical Address	0x4B23 2414 0x4B2B 2414	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	LINK STATUS CHANGE INTERRUPT REGISTER (MASKED VALUE)		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0000 000
1:0	LINKINTMASKED	MDIO link change interrupt, masked value. When asserted '1', a bit indicates that there was an MDIO link change event (i.e. change in the MDIOLink register) corresponding to the PHY address in the MDIOUserPhySel register and the corresponding linkint_enable bit was set.. linkintmasked[0] and linkintmasked[1] correspond to MDIOUserPhySel0 and MDIOUserPhysel1, respectively. Writing a '1' will clear the interrupt and writing 0 has no effect. If the int_test bit in the MDIOControl register is set, the host may set the linkint bits to a '1'. This mode may be used for test purposes.	RW	0x0

Table 30-411. Register Call Summary for Register PRUSS_MII_MDIO_LINKINTMASKED

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-412. PRUSS_MII_MDIO_USERINTRAW

Address Offset	0x0000 0020		
Physical Address	0x4B23 2420 0x4B2B 2420	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	USER COMMAND COMPLETE INTERRUPT REGISTER (RAW VALUE)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														USERINTRAW	

Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0000 000
1:0	USERINTRAW	Raw value of MDIO user command complete event for MDIOUserAccess1 through MDIOUserAccess0, respectively. When asserted ‘1’, a bit indicates that the previously scheduled PHY read or write command using that particular MDIOUserAccess register has completed. Writing a ‘1’ will clear the event and writing ‘0’ has no effect. . If the int_test bit in the MDIOControl register is set, the host may set the userintraw bits to a ‘1’. This mode may be used for test purposes.	RW	0x0

Table 30-413. Register Call Summary for Register PRUSS_MII_MDIO_USERINTRAW

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-414. PRUSS_MII_MDIO_USERINTMASKED

Address Offset	0x0000 0024		
Physical Address	0x4B23 2424 0x4B2B 2424	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	USER COMMAND COMPLETE INTERRUPT REGISTER (MASKED VALUE)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														USERINTMASKED	

Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0000 000
1:0	USERINTMASKED	Masked value of MDIO user command complete interrupt for MDIOUserAccess1 through MDIOUserAccess0, respectively. When asserted '1', a bit indicates that the previously scheduled PHY read or write command using that particular MDIOUserAccess register has completed and the corresponding userintmaskset bit is set to '1'. . Writing a '1' will clear the interrupt and writing '0' has no effect. . If the int_test bit in the MDIOControl register is set, the host may set the userintmasked bits to a '1'. This mode may be used for test purposes.	RW	0x0

Table 30-415. Register Call Summary for Register PRUSS_MII_MDIO_USERINTMASKED

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-416. PRUSS_MII_MDIO_USERINTMASKSET

Address Offset	0x0000 0028		
Physical Address	0x4B23 2428 0x4B2B 2428	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	USER INTERRUPT MASK SET REGISTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														USERINTMASKEDSET	

Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0000 000
1:0	USERINTMASKEDSET	MDIO user interrupt mask set for userintmasked[1:0], respectively. Writing a bit to '1' will enable MDIO user command complete interrupts for that particular MDIOUserAccess register. MDIO user interrupt for a particular MDIOUserAccess register is disabled if the corresponding bit is '0'. Writing a '0' to this register has no effect.	RW	0x0

Table 30-417. Register Call Summary for Register PRUSS_MII_MDIO_USERINTMASKSET

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-418. PRUSS_MII_MDIO_USERINTMASKCLR

Address Offset	0x0000 002C		
Physical Address	0x4B23 242C 0x4B2B 242C	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO

Table 30-418. PRUSS_MII_MDIO_USERINTMASKCLR (continued)

Description	USER INTERRUPT MASK CLEAR REGISTER
Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														USERINTMASKEDCLR	

Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0000 000
1:0	USERINTMASKEDCLR	MDIO user command complete interrupt mask clear for userintmasked[1:0], respectively. Writing a bit to '1' will disable further user command complete interrupts for that particular MDIOUserAccess register. Writing a '0' to this register has no effect.	RW	0x0

Table 30-419. Register Call Summary for Register PRUSS_MII_MDIO_USERINTMASKCLR

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-420. PRUSS_MII_MDIO_USERACCESS0

Address Offset	0x0000 0080		
Physical Address	0x4B23 2480 0x4B2B 2480	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	USER ACCESS REGISTER0		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GO	WRITE	ACK	RESERVED					REGADR								PHYADR								DATA							

Bits	Field Name	Description	Type	Reset
31	GO	Go. Writing a 1 to this bit causes the MDIO state machine to perform an MDIO access when it is convenient for it to do so, this is not an instantaneous process. Writing a 0 to this bit has no effect. This bit is write able only if the MDIO state machine is enabled. This bit will self clear when the requested access has been completed. Any writes to the MDIOUserAccess0 register are blocked when the go bit is '1'. If byte access is being used, the go bit should be written last.	RW	0x0
30	WRITE	Write enable. Setting this bit to a 1 causes the MDIO transaction to be a register write, otherwise it is a register read.	RW	0x0

Bits	Field Name	Description	Type	Reset
29	ACK	Acknowledge. This bit is set if the PHY acknowledged the read transaction.	RW	0x0
28:26	RESERVED		R	0x0
25:21	REGADR	Register address. This field specifies the PHY register to be accessed for this transaction.	RW	0x0
20:16	PHYADR	PHY address. This field specifies the PHY to be accessed for this transaction.	RW	0x0
15:0	DATA	User data. The data value read from or to be written to the specified PHY register.	RW	0x0

Table 30-421. Register Call Summary for Register PRUSS_MII_MDIO_USERACCESS0

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-422. PRUSS_MII_MDIO_USERPHYSEL0

Address Offset	0x0000 0084		
Physical Address	0x4B23 2484 0x4B2B 2484	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	USER PHY SELECT REGISTER0		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								LINKSEL	LINKINT_ENABLE	RESERVED	PHYADR_MON				

Bits	Field Name	Description	Type	Reset
31:8	RESERVED		R	0x0000 00
7	LINKSEL	Link status determination select. Set to ‘1’ to determine link status using the MLINK pin. Default value is ‘0’ which implies that the link status is determined by the MDIO state machine.	RW	0x0
6	LINKINT_ENABLE	Link change interrupt enable. Set to ‘1’ to enable link change status interrupts for PHY address specified in phyadr_mon. Link change interrupts are disabled if this bit is set to ‘0’.	RW	0x0
5	RESERVED		R	0
4:0	PHYADR_MON	PHY address whose link status is to be monitored.	RW	0x0

Table 30-423. Register Call Summary for Register PRUSS_MII_MDIO_USERPHYSEL0

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-424. PRUSS_MII_MDIO_USERACCESS1

Address Offset	0x0000 0088		
Physical Address	0x4B23 2488 0x4B2B 2488	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	USER ACCESS REGISTER1		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GO	WRITE	ACK	RESERVED					REGADR								PHYADR								DATA							

Bits	Field Name	Description	Type	Reset
31	GO	Go. Writing a 1 to this bit causes the MDIO state machine to perform an MDIO access when it is convenient for it to do so, this is not an instantaneous process. Writing a 0 to this bit has no effect. This bit is write able only if the MDIO state machine is enabled. This bit will self clear when the requested access has been completed. Any writes to the MDIOUserAccess0 register are blocked when the go bit is '1'. If byte access is being used, the go bit should be written last.	RW	0x0
30	WRITE	Write enable. Setting this bit to a 1 causes the MDIO transaction to be a register write, otherwise it is a register read.	RW	0x0
29	ACK	Acknowledge. This bit is set if the PHY acknowledged the read transaction.	RW	0x0
28:26	RESERVED		R	0x0
25:21	REGADR	Register address. This field specifies the PHY register to be accessed for this transaction.	RW	0x0
20:16	PHYADR	PHY address. This field specifies the PHY to be accessed for this transaction.	RW	0x0
15:0	DATA	User data. The data value read from or to be written to the specified PHY register.	RW	0x0

Table 30-425. Register Call Summary for Register PRUSS_MII_MDIO_USERACCESS1

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-426. PRUSS_MII_MDIO_USERPHYSEL1

Address Offset	0x0000 008C		
Physical Address	0x4B23 248C 0x4B2B 248C	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	USER PHY SELECT REGISTER1		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								LINKSEL	LINKINT_ENABLE	RESERVED	PHYADR_MON				

Bits	Field Name	Description	Type	Reset
31:8	RESERVED		R	0x0000 000
7	LINKSEL	Link status determination select. Set to ‘1’ to determine link status using the MLINK pin. Default value is ‘0’ which implies that the link status is determined by the MDIO state machine.	RW	0x0
6	LINKINT_ENABLE	Link change interrupt enable. Set to ‘1’ to enable link change status interrupts for PHY address specified in phyadr_mon. Link change interrupts are disabled if this bit is set to ‘0’.	RW	0x0
5	RESERVED		R	0
4:0	PHYADR_MON	PHY address whose link status is to be monitored.	RW	0x0

Table 30-427. Register Call Summary for Register PRUSS_MII_MDIO_USERPHYSEL1

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

30.1.11 PRU-ICSS Industrial Ethernet Peripheral (IEP)

This section describes the Industrial Ethernet Peripheral (IEP) module which is part of the PRU-ICSS.

30.1.11.1 PRU-ICSS IEP Overview

The Industrial Ethernet Peripheral (IEP) performs hardware work required for industrial ethernet functions. The IEP module features an industrial ethernet timer with 16 compare events and a digital I/O port (DIGIO).

30.1.11.2 PRU-ICSS IEP Functional Description

This section provides the functional description of the IEP components.

30.1.11.2.1 PRU-ICSS IEP Clock Generation

The IEP has a selectable module input clock (ICSS_IEP_CLK input, see also Section 30.1.3). The clock source is selected by the state of the IEPCLK.OCP_EN bit within the PRU-ICSS CFG register space.

Two clock sources are supported for the IEP input clock:

- PRUSS_IEP_CLK: The default functional clock for IEP derived from PRCM. Runs at 200 MHz.
- PRUSS_GICLK: The PRUSS_CFG gateable interface clock derived from PRCM.

Switching from PRUSS_IEP_CLK to PRUSS_GICLK is done by writing 1 to the PRUSS_IEPCLK.OCP_EN bit. This is a one time configuration step before enabling the IEP function. Switching back from PRUSS_GICLK to PRUSS_IEP_CLK is only supported through a hardware reset of the PRU-ICSS.

CAUTION

When software enables the clock (at PRU-ICSS level) to the IEP module clock input via setting bit PRUSS_IEPCLK[0] OCP_EN to 0b1 in the PRUSS_CFG space, there must be NO in-flight transactions to the IEP block.

CAUTION

ONLY switching from PRUSS_IEP_CLK (the IEP specific functional clock source) to the PRUSS_GICLK (top level interface clock) source is supported in software by device integrated PRU-ICSS. Switching back from PRUSS_GICLK to PRUSS_IEP_CLK is ONLY supported via assertion of a hardware reset to the PRU-ICSS.

30.1.11.2.2 PRU-ICSS Industrial Ethernet Timer

The industrial ethernet timer is a simple 64-bit timer. This timer is intended for use by industrial ethernet functions but can also be leveraged as a generic timer in other applications.

30.1.11.2.2.1 PRU-ICSS Industrial Ethernet Timer Features

The industrial ethernet timer supports the following features:

- One master 64-bit count-up counter with an overflow status bit.
  - Runs on PRUSS_IEP_CLK or PRUSS_GICLK.
  - Write 1 to clear status.
  - Supports a programmable increment value from 1 to 16 (default 5).
  - An optional compensation method allows the increment value to apply compensation increment value from 1 to 16 count up to 2^{24} PRUSS_IEP_CLK/PRUSS_GICLK events with additional slow compensation mode

- 16× 64-bit compare registers: PRUSS_IEP_COMPARE0j/PRUSS_IEP_COMPARE1j (where j=0 to 15) and PRUSS_IEP_COMPARE_STATUS.
  - 16 status bits, write 1 to clear
  - 16 individual event outputs
  - One global event output for interrupt generation triggered by any compare event
- 32 outputs, one high-level and one high-pulse for each compare hit event
- PRUSS_IEP_COMPARE_CFG[0] CMP0_RST_CNT_EN, if enabled, will reset the master counter
- pwm0_sync_out/pwm3_sync_out, if enabled, will reset the master counter
- master counter reset-state is programmable

30.1.11.2.2.2 PRU-ICSS Industrial Ethernet Timer Basic Programming Sequence

Follow these basic steps to configure the IEP Timer.

Compare function:

1. Initialize timer to known state (default values)
   - Disable counter (PRUSS_IEP_GLOBAL_CFG[0] CNT_ENABLE = 0)
   - Reset Count Register (PRUSS_IEP_LOW_COUNTER, PRUSS_IEP_HIGH_COUNTER) by writing 0xFFFFFFFF to clear
   - Clear overflow status register (PRUSS_IEP_STATUS[0] CNT_OVF = 1)
   - Clear compare status (PRUSS_IEP_COMPARE_STATUS) by writing 0xFFFFFFFF to clear
2. Set compare values PRUSS_IEP_COMPARE0j,PRUSS_IEP_COMPARE1j
3. Enable compare events (PRUSS_IEP_COMPARE_CFG[8:1] CMP_EN).
4. Set increment value (PRUSS_IEP_GLOBAL_CFG[7:4] DEFAULT_INC).
5. Set compensation value (PRUSS_IEP_COMPENSATION[23:0] COMPEN_CNT)
6. Enable counter (PRUSS_IEP_GLOBAL_CFG[0] CNT_ENABLE = 1)

30.1.11.2.3 PRU-ICSS Industrial Ethernet Digital IOs

The IEP Digital I/O (DIGIO) block provides dedicated I/Os intended for industrial ethernet protocols, but they can also be used as generic I/Os in other applications.

30.1.11.2.3.1 Features

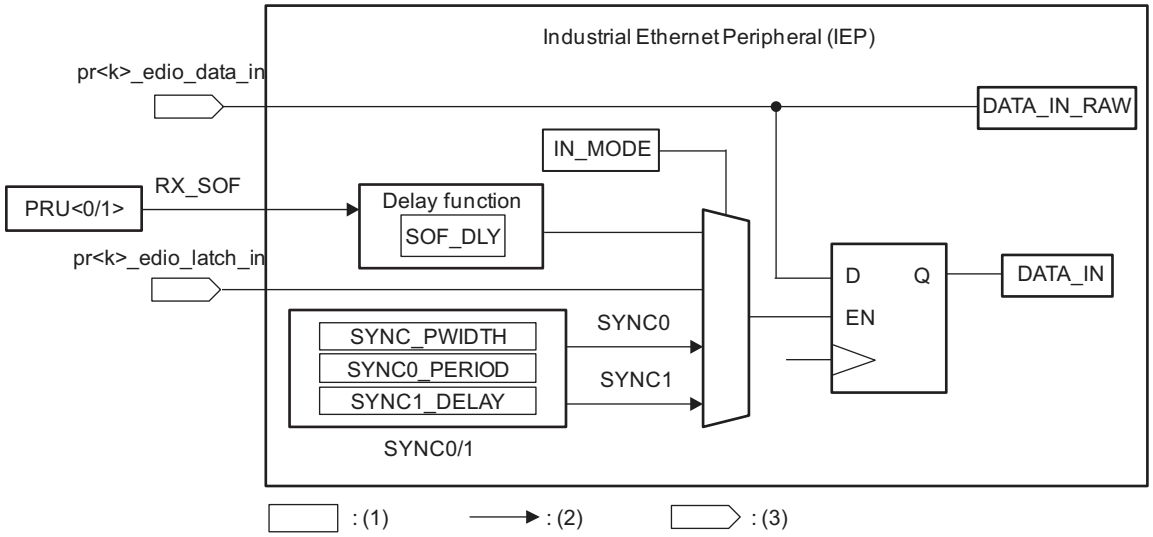
The industrial ethernet digital I/O supports the following features:

- Digital data output
  - 8 channels (pr1_edio_data_out[7:0])
  - Software controls enable signal driving output data output
- Digital data out enable (optional tri-state control)
- Digital data input
  - 8 channels (pr1_edio_data_in[7:0])
  - PRUSS_IEP_DIGIO_DATA_IN_RAW supports direct sampling of pr1_edio_data_in
  - External latch event signal (pr1_edio_latch_in) triggers a pulse on which pr1_edio_data_in is sampled

30.1.11.2.3.2 DIGIO Block Diagrams

Figure 30-53 shows the signals and registers for capturing the DIGIO data in. Note that IN_MODE in the PRUSS_IEP_DIGIO_CTRL register must be set to 1 for data to be latched on the external pr1_edio_latch_in signal.

Figure 30-53. IEP DIGIO Data In



(1) Register

(2) Internal signal wire

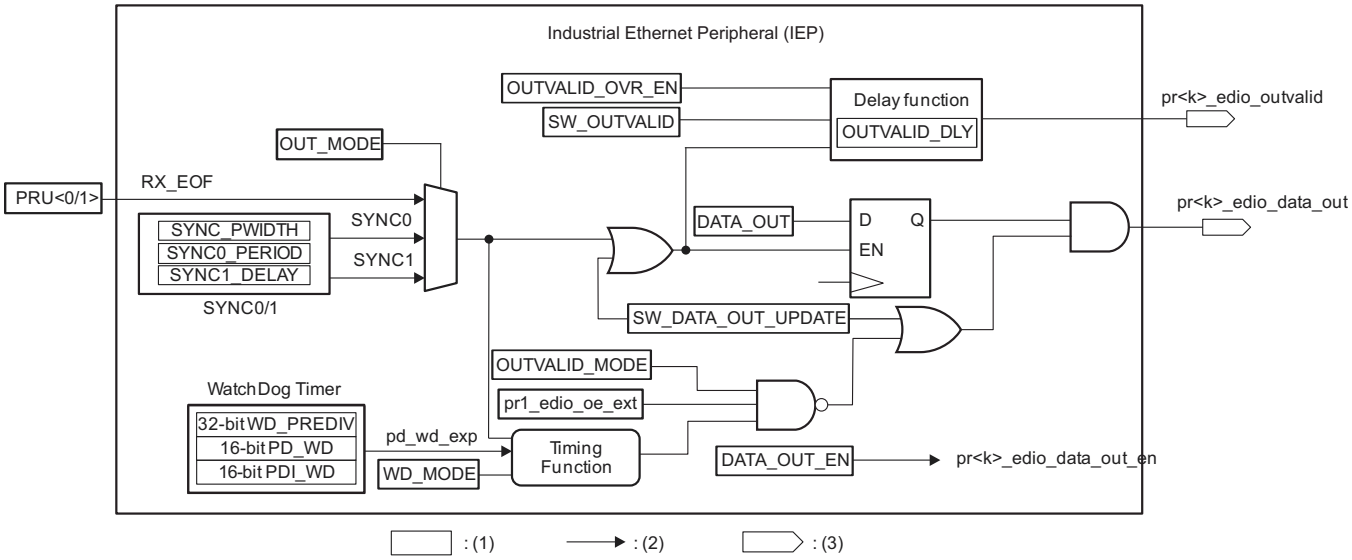
(3) External pin input/output

(1) Register

(2) External pin input/output

Figure 30-54 shows the signals and registers for driving the DIGIO data out.

Figure 30-54. IEP DIGIO Data Out



(1) Register

(2) Internal signal wire

(3) External pin input/output

(1) Register

(2) External pin input/output

30.1.11.2.3.3 Basic Programming Model

Follow these steps to configure and read the DIGIO Data Input.

1. Read PRUSS_IEP_DIGIO_DATA_IN_RAW for raw input data

or

1. Enable sampling of pr1_edio_data_in[7:0] by setting PRUSS_IEP_DIGIO_CTRL[5:4] IN_MODE = 0x1
2. Read PRUSS_IEP_DIGIO_DATA_IN for data sampled upon pr1_edio_latch_in posedge

Follow these steps to configure and write to the DIGIO Data Output.

1. Pre-configure DIGIO by setting PRUSS_IEP_DIGIO_EXP[1] OUTVALID_OVR_EN and PRUSS_IEP_DIGIO_EXP[0] SW_DATA_OUT_UPDATE
2. Write to PRUSS_IEP_DIGIO_DATA_OUT to configure output data
3. To Hi-Z output, set corresponding PRUSS_IEP_DIGIO_DATA_OUT_EN bits to 1 (clear to 0 to drive value stored in PRUSS_IEP_DIGIO_DATA_OUT)

30.1.11.3 PRUSS_IEP Register Manual

This section describes the registers of the PRUSS_IEP module.

30.1.11.3.1 PRUSS_IEP Instance Summary

Table 30-428. PRUSS_IEP Instance Summary

Module Name	Base Address	Size
PRUSS1_IEP	0x4B22 E000	796 Bytes
PRUSS2_IEP	0x4B2A E000	796 Bytes

30.1.11.3.2 PRUSS_IEP Registers

30.1.11.3.2.1 PRUSS_IEP Register Summary

Table 30-429. PRUSS_IEP Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS1_IEP Physical Address	PRUSS2_IEP Physical Address
PRUSS_IEP_GLOBAL_CFG	RW	32	0x0000 0000	0x4B22 E000	0x4B2A E000
PRUSS_IEP_STATUS	RW	32	0x0000 0004	0x4B22 E004	0x4B2A E004
PRUSS_IEP_COMPENSATION	RW	32	0x0000 0008	0x4B22 E008	0x4B2A E008
PRUSS_IEP_SLOW_COMPENSATION	RW	32	0x0000 000C	0x4B22 E00C	0x4B2A E00C
PRUSS_IEP_LOW_COUNTER	RW	32	0x0000 0010	0x4B22 E010	0x4B2A E010
PRUSS_IEP_HIGH_COUNTER	RW	32	0x0000 0014	0x4B22 E014	0x4B2A E014
PRUSS_IEP_COMPARE_CFG	RW	32	0x0000 0070	0x4B22 E070	0x4B2A E070
PRUSS_IEP_COMPARE_STATUS	RW	32	0x0000 0074	0x4B22 E074	0x4B2A E074
PRUSS_IEP_COMPARE0j(1)	RW	32	0x0000 0078 + (0x8 * j)	0x4B22 E078 + (0x8 * j)	0x4B2A E078 + (0x8 * j)
PRUSS_IEP_COMPARE1j(1)	RW	32	0x0000 007C + (0x8 * j)	0x4B22 E07C + (0x8 * j)	0x4B2A E07C + (0x8 * j)
PRUSS_IEP_COMPARE0k(2)	RW	32	0x0000 00C0 + (0x8 *k)	0x4B22 E0C0 + (0x8 *k)	0x4B2A E0C0 + (0x8 *k)
PRUSS_IEP_COMPARE1k(2)	RW	32	0x0000 00C4 + (0x8 *k)	0x4B22 E0C4 + (0x8 *k)	0x4B2A E0C4 + (0x8 *k)
PRUSS_IEP_LOW_COUNTER_RESET_VALUE	RW	32	0x0000 0100	0x4B22 E100	0x4B2A E100
PRUSS_IEP_HIGH_COUNTER_RESET_VALUE	RW	32	0x0000 0104	0x4B22 E104	0x4B2A E104
PRUSS_IEP_PWM	RW	32	0x0000 0108	0x4B22 E108	0x4B2A E108

(1) j=0 to 7
(2) k=8 to 15

Table 30-429. PRUSS_IEP Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS1_IEP Physical Address	PRUSS2_IEP Physical Address
PRUSS_IEP_DIGIO_CTRL	RW	32	0x0000 0300	0x4B22 E300	0x4B2A E300
RESERVED	R	32	0x0000 0304	0x4B22 E304	0x4B2A E304
PRUSS_IEP_DIGIO_DATA_IN	R	32	0x0000 0308	0x4B22 E308	0x4B2A E308
PRUSS_IEP_DIGIO_DATA_IN_RAW	R	32	0x0000 030C	0x4B22 E30C	0x4B2A E30C
PRUSS_IEP_DIGIO_DATA_OUT	RW	32	0x0000 0310	0x4B22 E310	0x4B2A E310
PRUSS_IEP_DIGIO_DATA_OUT_EN	RW	32	0x0000 0314	0x4B22 E314	0x4B2A E314
PRUSS_IEP_DIGIO_EXP	RW	32	0x0000 0318	0x4B22 E318	0x4B2A E318

30.1.11.3.2.2 PRUSS_IEP Register Description

Table 30-430. PRUSS_IEP_GLOBAL_CFG

Address Offset	0x0000 0000		
Physical Address	0x4B22 E000 0x4B2A E000	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	GLOBAL CFG		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CMP_INC																DEFAULT_INC				RESERVED			CNT_ENABLE

Bits	Field Name	Description	Type	Reset
31:20	RESERVED		R	0x0
19:8	CMP_INC	Defines the increment value when compensation is active	RW	0x5
7:4	DEFAULT_INC	Defines the default increment value	RW	0x5
3:1	RESERVED		R	0
0	CNT_ENABLE	Counter enable 0: Disables the counter. The counter maintains the current count. 1: Enables the counter.	RW	0x0

Table 30-431. Register Call Summary for Register PRUSS_IEP_GLOBAL_CFG

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRU-ICSS Industrial Ethernet Timer Basic Programming Sequence: [0] [1] [2]
- PRUSS_IEP Register Summary: [5]

Table 30-432. PRUSS_IEP_STATUS

Address Offset	0x0000 0004		
Physical Address	0x4B22 E004 0x4B2A E004	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	STATUS		
Type	RWr1Clr		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															CNT_OVF

Bits	Field Name	Description	Type	Reset
31:1	RESERVED		R	0x0
0	CNT_OVF	Counter overflow status. 0: No overflow 1: Overflow occurred	RWr1Clr	0x0

Table 30-433. Register Call Summary for Register PRUSS_IEP_STATUS

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRU-ICSS Industrial Ethernet Timer Basic Programming Sequence: [0]
- PRUSS_IEP Register Summary: [1]

Table 30-434. PRUSS_IEP_COMPENSATION

Address Offset	0x0000 0008		
Physical Address	0x4B22 E008 0x4B2A E008	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	COMPENSATION		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								COMPEN_CNT																							

Bits	Field Name	Description	Type	Reset
31:24	RESERVED		R	0
23:0	COMPEN_CNT	Compensation counter. Read returns the current COMPEN_CNT value. 0: Compensation is disabled and counter will increment by DEFAULT_INC. n: Compensation is enabled until COMPEN_CNT decrements to 0. The COMPEN_CNT value decrements on every iep_clk cycle. When COMPEN_CNT is greater than 0, then count value increments by CMP_INC. NOTE: SLOW_COMPEN_CNT MUST be set to zero IF COMPEN_CNT is not zero.	RW	0x0

Table 30-435. Register Call Summary for Register PRUSS_IEP_COMPENSATION

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRU-ICSS Industrial Ethernet Timer Basic Programming Sequence: [0]
- PRUSS_IEP Register Summary: [1]

Table 30-436. PRUSS_IEP_SLOW_COMPENSATION

Address Offset	0x0000 000C		
Physical Address	0x4B22 E00C 0x4B2A E00C	**Instance**	PRUSS1_IEP PRUSS1_IEPPRUSS2_IEP
Description	SLOW COMPENSATION		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SLOW_COMPEN_CNT																															

Bits	Field Name	Description	Type	Reset
31:0	SLOW_COMPEN_CNT	Slow compensation counter. Write: 0x0: Slow compensation is disabled and counter will increment by DEFAULT_INC. 0xn: Compensation is enabled for 1 count for every SLOW_COMPEN_CNT cycle, this is free running and continuous until software clears the MMR For example, SLOW_COMPEN_CNT = 16, every 16 clock cycles the compensation value is used for 1 count. Note COMPEN_CNT MUST be set to zero IF SLOW_COMPEN_CNT is not zero. Read: Software can read the number of cycles left until the compensation event. For example, software writes SLOW_COMPEN_CNT = 0x100 and reads SLOW_COMPEN_CNT = 0x7. This means in 6 more IEP_CLK cycles before the counter reaches 0x1 for the compensation event. If software writes SLOW_COMPEN_CNT = 0x8000 before compensation event, then the counter will reset to 0x8000.	RW	0x0

Table 30-437. Register Call Summary for Register PRUSS_IEP_SLOW_COMPENSATION

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRUSS_IEP Register Summary: [0]

Table 30-438. PRUSS_IEP_LOW_COUNTER

Address Offset	0x0000 0010		
Physical Address	0x4B22 E010 0x4B2A E010	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	64 bit count value low		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT																															

Bits	Field Name	Description	Type	Reset
31:0	COUNT	64-bit count value (lower 32-bits). Increments by (DEFAULT_INC or CMP_INC) on every positive edge of PRUSS_IEP_CLK (200MHz).	RW	0x0

Table 30-439. Register Call Summary for Register PRUSS_IEP_LOW_COUNTER

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRU-ICSS Industrial Ethernet Timer Basic Programming Sequence: [0]
- PRUSS_IEP Register Summary: [1]

Table 30-440. PRUSS_IEP_HIGH_COUNTER

Address Offset	0x0000 0014		
Physical Address	0x4B22 E014 0x4B2A E014	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	64 bit count value high		

Table 30-440. PRUSS_IEP_HIGH_COUNTER (continued)

Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT																															

Bits	Field Name	Description	Type	Reset
31:0	COUNT	64-bit count value (upper 32-bits). Increments by (DEFAULT_INC or CMP_INC) on every positive edge of PRUSS_IEP_CLK (200MHz).	RW	0x0

Table 30-441. Register Call Summary for Register PRUSS_IEP_HIGH_COUNTER

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRU-ICSS Industrial Ethernet Timer Basic Programming Sequence: [0]
- PRUSS_IEP Register Summary: [1]

Table 30-442. PRUSS_IEP_COMPARE_CFG

Address Offset	0x0000 0070		
Physical Address	0x4B22 E070 0x4B2A E070	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	COMPARE CFG		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CMP_EN															CMP0_RST_CNT_EN

Bits	Field Name	Description	Type	Reset
31:17	RESERVED		R	0x00000
16:1	CMP_EN	Enable bits for each of the compare registers CMP_EN =0 : Disables CMPj/k Event CMP_EN=1: Enables CMPj/k Event CMP_EN[0] (bit 1 of register) maps to CMP0 event	RW	0x0
0	CMP0_RST_CNT_EN	Enable the reset of the counter 0: Disable 1: Enable the reset of the counter if a CMP0 event occurs	RW	0x0

Table 30-443. Register Call Summary for Register PRUSS_IEP_COMPARE_CFG

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRU-ICSS Industrial Ethernet Timer Features: [0]
- PRU-ICSS Industrial Ethernet Timer Basic Programming Sequence: [1]
- PRUSS_IEP Register Summary: [2]

Table 30-444. PRUSS_IEP_COMPARE_STATUS

Address Offset	0x0000 0074		
Physical Address	0x4B22 E074 0x4B2A E074	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	COMPARE STATUS		
Type	RWr1Clr		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CMP_HIT															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED		R	0x0000 00
15:0	CMP_HIT	Status bit for each of the compare registers "Match" indicates the current counter is greater than or equal to the compare value. Note it is the firmware's responsibility to handle the IEP overflow. CMP_HIT<n> = 0: No match has occured CMP_HIT<n> = 1: A match occured. The associated hardware event signal will assert and remain high until the status is cleared.	RWr1Clr	0x0

Table 30-445. Register Call Summary for Register PRUSS_IEP_COMPARE_STATUS

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRU-ICSS Industrial Ethernet Timer Features: [0]
- PRU-ICSS Industrial Ethernet Timer Basic Programming Sequence: [1]
- PRUSS_IEP Register Summary: [2]

Table 30-446. PRUSS_IEP_COMPARE0j

Address Offset	0x0000 0078 + (0x8 * j)	**Index**	j = 0 to 7
Physical Address	0x4B22 E078 + (0x8 * j) 0x4B2A E078 + (0x8 * j)	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	COMPARE(j) low		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP																															

Bits	Field Name	Description	Type	Reset
31:0	CMP	Compare j value >= comparator	RW	0x0

Table 30-447. Register Call Summary for Register PRUSS_IEP_COMPARE0j

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRU-ICSS Industrial Ethernet Timer Features: [0]
- PRU-ICSS Industrial Ethernet Timer Basic Programming Sequence: [1]
- PRUSS_IEP Register Summary: [2]

Table 30-448. PRUSS_IEP_COMPARE1j

Address Offset	0x0000 007C + (0x8 * j)	**Index**	j = 0 to 7
Physical Address	0x4B22 E07C + (0x8 * j) 0x4B2A E07C + (0x8 * j)	**Instance**	PRUSS1_IEP PRUSS2_IEP

Table 30-448. PRUSS_IEP_COMPARE1j (continued)

Description	COMPARE(j) high
Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP																															

Bits	Field Name	Description	Type	Reset
31:0	CMP	Compare j value >= comparator	RW	0x0

Table 30-449. Register Call Summary for Register PRUSS_IEP_COMPARE1j

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRU-ICSS Industrial Ethernet Timer Features: [0]
- PRU-ICSS Industrial Ethernet Timer Basic Programming Sequence: [1]
- PRUSS_IEP Register Summary: [2]

Table 30-450. PRUSS_IEP_COMPARE0k

Address Offset	0x0000 0078 + (0x8 * k)	**Index**	k= 8 to 15
Physical Address	0x4B22 E0C0 + (0x8 *k) 0x4B2A E0C0 + (0x8 *k)	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	COMPARE(k) low		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP																															

Bits	Field Name	Description	Type	Reset
31:0	CMP	Compare k value >= comparator	RW	0x0

Table 30-451. Register Call Summary for Register PRUSS_IEP_COMPARE0k

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRUSS_IEP Register Summary: [0]

Table 30-452. PRUSS_IEP_COMPARE1k

Address Offset	0x0000 007C + (0x8 * k)	**Index**	k= 8 to 15
Physical Address	0x4B22 E0C4 + (0x8 *k) 0x4B2A E0C4 + (0x8 *k)	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	COMPARE(k) high		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP																															

Bits	Field Name	Description	Type	Reset
31:0	CMP	Compare k value >= comparator	RW	0x0

Table 30-453. Register Call Summary for Register PRUSS_IEP_COMPARE1k

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRUSS_IEP Register Summary: [0]

Table 30-454. PRUSS_IEP_LOW_COUNTER_RESET_VALUE

Address Offset	0x0000 0100		
Physical Address	0x4B22 E100 0x4B2A E100	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	LOW_COUNTER_RESET_VALUE		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET_VAL																															

Bits	Field Name	Description	Type	Reset
31:0	RESET_VAL	Reset value (lower 32-bits). This register enables SW to define the reset state of the Master Counter, which can be reset by the following events (if enabled): CMP0 event; PWM0_SYNC_OUT event; PWM3_SYNC_OUT event. The RESET_VAL should be in increments of the DEFAULT_INC (default state is 5). For example, 0x0000_000A.	RW	0x0

Table 30-455. Register Call Summary for Register PRUSS_IEP_LOW_COUNTER_RESET_VALUE

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRUSS_IEP Register Summary: [0]

Table 30-456. PRUSS_IEP_HIGH_COUNTER_RESET_VALUE

Address Offset	0x0000 0104		
Physical Address	0x4B22 E104 0x4B2A E104	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	HIGH_COUNTER_RESET_VALUE		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET_VAL																															

Bits	Field Name	Description	Type	Reset
31:0	RESET_VAL	Reset value (upper 32-bits). This register enables SW to define the reset state of the Master Counter, which can be reset by the following events (if enabled): CMP0 event; PWM0_SYNC_OUT event; PWM3_SYNC_OUT event. The RESET_VAL should be in increments of the DEFAULT_INC (default state is 5). For example, 0x0000_000A.	RW	0x0

Table 30-457. Register Call Summary for Register PRUSS_IEP_HIGH_COUNTER_RESET_VALUE

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRUSS_IEP Register Summary: [0]

Table 30-458. PRUSS_IEP_PWM

Address Offset	0x0000 0108		
Physical Address	0x4B22 E108 0x4B2A E108	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	PWM Sync Out		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												PWM3_HIT	PWM3_RST_CNT_EN	PWM0_HIT	PWM0_RST_CNT_EN

Bits	Field Name	Description	Type	Reset
31:4	RESERVED		R	0
3	PWM3_HIT	The raw status bit of pwm3_sync_out event. 0x0: No pwm3_sync_out event 0x1: pwm3_sync_out event occurred Write 1 to Clear.	RW1Clr	0
2	PWM3_RST_CNT_EN	Enable the reset of the counter by a pwm3_sync_out event. 0x0: Disable 0x1: Enable the reset of the counter if a pwm3_sync_out event occurs Notes: input must be high for > 10ns to ensure reset event occurs , if less it might not occur. The rising edge event causes the reset after delay of 5 to 15ns. To reset again, it must go low for > 100ns. If the low pulse is less, a 2nd reset event may not occur.	RW	0x0
1	PWM0_HIT	The raw status bit of pwm0_sync_out event. 0x0: No pwm0_sync_out event 0x1: pwm0_sync_out event occurred Write 1 to Clear.	RW1Clr	0x0
0	PWM0_RST_CNT_EN	Enable the reset of the counter by a pwm0_sync_out event 0x0: Disable 0x1: Enable the reset of the counter if a pwm0_sync_out event occurs Notes: input must be high for > 10ns to ensure reset event occurs , if less it might not occur. The rising edge event causes the reset after delay of 5 to 15ns. To reset again, it must go low for > 100ns. If the low pulse is less, a 2nd reset event may not occur.	RW	0x0

Table 30-459. Register Call Summary for Register PRUSS_IEP_PWM

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRUSS_IEP Register Summary: [0]

Table 30-460. PRUSS_IEP_DIGIO_CTRL

Address Offset	0x0000 0300		
Physical Address	0x4B22 E300 0x4B2A E300	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	DIGIO		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								OUT_MODE		IN_MODE		WD_MODE	BIDI_MODE	OUTVALID_MODE	OUTVALID_POL

Bits	Field Name	Description	Type	Reset
31:8	RESERVED		R	0x0
7:6	OUT_MODE	Reserved	RW	0x0
5:4	IN_MODE	Defines event that triggers data in to be sampled 0b00: Reserved 0b01: Rising edge of external pr<k>-edio_latch_in signal 0b10: Reserved 0b11: Reserved	RW	0x0
3	WD_MODE	Reserved	RW	0x0
2	BIDI_MODE	Reserved	R	0x1
1	OUTVALID_MODE	Reserved	RW	0x0
0	OUTVALID_POL	Reserved	R	0x0

Table 30-461. Register Call Summary for Register PRUSS_IEP_DIGIO_CTRL

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- DIGIO Block Diagrams: [0]
- Basic Programming Model: [1]
- PRUSS_IEP Register Summary: [5]
- PRUSS_IEP Register Description: [6]

Table 30-462. PRUSS_IEP_DIGIO_DATA_IN

Address Offset	0x0000 0308		
Physical Address	0x4B22 E308 0x4B22 E308 0x4B2A E308 0x4B2A E308	**Instance**	PRUSS1_IEP PRUSS1_IEP PRUSS2_IEP PRUSS2_IEP
Description	DIGIO		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_IN																															

Bits	Field Name	Description	Type	Reset
31:0	DATA_IN	Data input. Sample time of digital inputs is controlled externally by using the pr<k>_edio_latch_in signal. Must enable by setting PRUSS_IEP_DIGIO_CTRL[5:4] IN_MODE. Only [7:0] are exported to device pins in this device.	R	0x-

Table 30-463. Register Call Summary for Register PRUSS_IEP_DIGIO_DATA_IN

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- Basic Programming Model: [0]
- PRUSS_IEP Register Summary: [3]

Table 30-464. PRUSS_IEP_DIGIO_DATA_IN_RAW

Address Offset	0x0000 030C		
Physical Address	0x4B22 E30C 0x4B22 E30C 0x4B2A E30C 0x4B2A E30C	**Instance**	PRUSS1_IEP PRUSS1_IEP PRUSS2_IEP PRUSS2_IEP
Description	DIGIO		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_IN_RAW																															

Bits	Field Name	Description	Type	Reset
31:0	DATA_IN_RAW	Raw Data Input. Direct sample of EDIO_DATA_IN[31:0]. Only [7:0] are exported to device pins in this device.	R	0x-

Table 30-465. Register Call Summary for Register PRUSS_IEP_DIGIO_DATA_IN_RAW

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- Features: [0]
- Basic Programming Model: [1]
- PRUSS_IEP Register Summary: [4]

Table 30-466. PRUSS_IEP_DIGIO_DATA_OUT

Address Offset	0x0000 0310		
Physical Address	0x4B22 E310 0x4B22 E310 0x4B2A E310 0x4B2A E310	**Instance**	PRUSS1_IEP PRUSS1_IEP PRUSS2_IEP PRUSS2_IEP
Description	DIGIO		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_OUT																															

Bits	Field Name	Description	Type	Reset
31:0	DATA_OUT	Data Output	RW	0x0

Table 30-467. Register Call Summary for Register PRUSS_IEP_DIGIO_DATA_OUT

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- Basic Programming Model: [0] [1]
- PRUSS_IEP Register Summary: [4]

Table 30-468. PRUSS_IEP_DIGIO_DATA_OUT_EN

Address Offset	0x0000 0314		
Physical Address	0x4B22 E314 0x4B22 E314 0x4B2A E314 0x4B2A E314	**Instance**	PRUSS1_IEP PRUSS1_IEP PRUSS2_IEP PRUSS2_IEP
Description	DIGIO		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_OUT_EN																															

Bits	Field Name	Description	Type	Reset
31:0	DATA_OUT_EN	Enables tri-state control for pr<k>_edio_data_out[7:0].	RW	0x0

Table 30-469. Register Call Summary for Register PRUSS_IEP_DIGIO_DATA_OUT_EN

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- Basic Programming Model: [0]
- PRUSS_IEP Register Summary: [3]

Table 30-470. PRUSS_IEP_DIGIO_EXP

Address Offset	0x0000 0318		
Physical Address	0x4B22 E318 0x4B22 E318 0x4B2A E318 0x4B2A E318	**Instance**	PRUSS1_IEP PRUSS1_IEP PRUSS2_IEP PRUSS2_IEP
Description	DIGIO		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																		EOF_SEL	SOF_SEL	SOF_DLY				OUTVALID_DLY				RESERVED	SW_OUTVALID	OUTVALID_OVR_EN	SW_DATA_OUT_UPDATE

Bits	Field Name	Description	Type	Reset
31:14	RESERVED		R	0x0
13	EOF_SEL	Reserved	RW	0x0
12	SOF_SEL	Reserved	RW	0x0
11:8	SOF_DLY	Reserved	RW	0x0
7:4	OUTVALID_DLY	Reserved	RW	0x2
3	RESERVED		R	0
2	SW_OUTVALID	Reserved	RW	0x0
1	OUTVALID_OVR_EN	Enable software to control value of pr<k>_edio_data_out [7:0] 0: Disable 1: Enable	RW	0x0
0	SW_DATA_OUT_UPDATE	Enable PRUSS_DIGIO_DATA_OUT to be driven out on pr<k>_edio_data_out. Only valid if OUTVALID_OVR_EN = 1. 0: Disable 1: Enable	RW	0x0

Table 30-471. Register Call Summary for Register PRUSS_IEP_DIGIO_EXP

PRU-ICSS Industrial Ethernet Peripheral (IEP) • Features: • Basic Programming Model: [1] [2] • PRUSS_IEP Register Summary: [5]

30.2 Programmable Real-Time Unit Subsystem and Industrial Communication Subsystem AM572x SR1.1

This section describes the Programmable Real-Time Unit and Industrial Communication Subsystems (PRU-ICSS) in the SR1.1 of this device.

30.2.1 PRU-ICSS Overview

The Programmable Real-Time Unit Subsystem and Industrial Communication Subsystem (PRU-ICSS) consists of:

- Two 32-bit load/store RISC CPU cores - Programmable Real-Time Units (PRU0 and PRU1)
- Data RAMs per PRU core
- Instruction RAMs per PRU core
- Shared RAM
- Peripheral modules
- Interrupt controller (PRUSS_INTC)

The programmable nature of the PRU cores, along with their access to pins, events and all device 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 device.

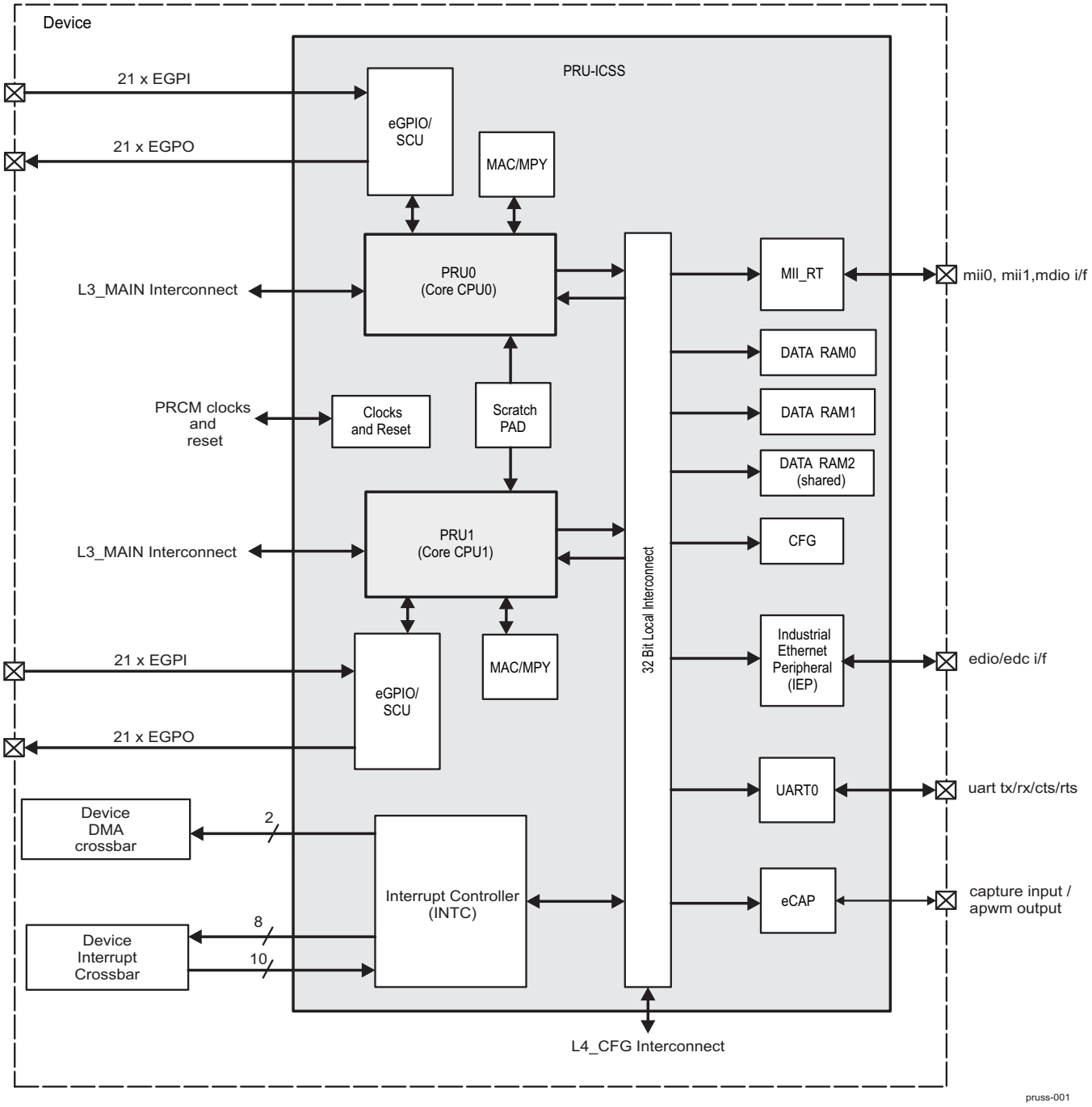
The device has integrated two identical PRU subsystems (PRU-ICSS1 and PRU-ICSS2). The PRUs have access to all resources on the device through a master port on the L3_MAIN interconnect, and vice versa, the external host processors can access the PRU-ICSS resources through a L3_MAIN slave port.

The PRU-ICSS L2 interconnect, provides access to the various internal and external masters to the resources inside the PRU-ICSS. A subsystem local Interrupt Controller - PRUSS_INTC handles system input events and posts events back to the device-level host CPUs.

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. This interaction between processors is determined by the nature of the firmware loaded into the PRU's instruction memory.

Figure 30-55 shows an overview of the PRU subsystem.

Figure 30-55. PRU-ICSS Overview



30.2.1.1 PRU-ICSS Key Features

The PRU subsystem includes the following main features:

- Two PRU CPUs
  - 21 Enhanced General-Purpose Inputs (EGPI) and 21 Enhanced General-Purpose Outputs (EGPO)
  - Asynchronous capture
  - Multiplier with optional accumulation (MAC)
  - 12-KiB program RAM per PRU CPU (signified IRAM0 for PRU0 and IRAM1 for PRU1)

  - 8-KiB data RAM per PRU CPU (signified RAM0 for PRU0 and RAM1 for PRU1)
  - Two high-performance master (initiator) ports on the L3_MAIN interconnect - one per PRU
- 32-KiB general purpose memory RAM (signified RAM2) shared between PRU0 and PRU1
- One Scratch-Pad (SPAD) memory
  - 3 Banks of 30 × 32-bit registers
- Broadside direct connect between PRU cores within subsystem. Optional address translation for PRU transaction to External Host
- 16 software events generated by two PRUs
- One Ethernet MII_RT module (PRUSS_MII_RT_CFG) with two MII ports and configurable connections to PRUs
- MDIO Port (PRUSS_MII_MDIO) to control extenal Ethernet PHY
- Industrial Ethernet Peripheral (IEP) to manage/generate Industrial Ethernet functions
- 16550-compatible UART with a dedicated 192-MHz clock to support 12-Mbps PROFIBUS
- Industrial Ethernet timer with 7/9 capture and 8 compare events
- Enhanced Capture Module (ECAP)
- Interrupt Controller (PRUSS_INTC)
  - Up to 64 input events supported
  - Interrupt mapping to 10 interrupt channels via an interrupt crossbar
  - 10 Host interrupts (2 to PRU0 and PRU1, 8 outputs to device level)
  - Each system event can be enabled and disabled
  - Each host event can be enabled and disabled
  - Hardware prioritization of events
  - Two level-sensitive DMA requests generated by the local PRUSS INTC to the device DMA Crossbar
- One Slave (target) port for memory mapped register and internal memories access through device L3_MAIN
- Two (master and slave) 32-bit ports for low-latency interface between PRU-ICSS subsystems
- Flexible power management support
- Integrated 32-bit interconnect
- Parity control supported by all memories

NOTE: Use of IEP Sync and WatchDog features are only supported via the TI Industrial software development kit (SDK).

PRU-ICSS unsupported features:

- Only 8 bits are supported of the 32-bit ECAT Digital Data Input
- Only 8 bits are supported of the 32-bit ECAT Digital Data Output
- UART Modem interface is not supported

30.2.2 PRU-ICSS Environment

This section specifies the PRU-ICSS subsystem (top) interface signals to the device environment components.

30.2.2.1 PRU-ICSS I/O Interface

The PRU-ICSS1 and PRU-ICSS2 external interface signals are described in Table 30-472 and Table 30-473, respectively. The PRU-ICSS has a large number of available I/O signals. Most of these are multiplexed with other functional signals at the device level.

Table 30-472. PRU-ICSS1 I/O Signals

Device Level Signal Name	I/O	Description	Value at Reset
pr1_pru0_gpo[20:0]	O	PRU0 Register R30 Outputs	0
pr1_pru0_gpi[20:0]	I	PRU0 Register R31 Inputs	HiZ
pr1_pru1_gpo[20:0]	O	PRU1 Register R30 Outputs	0
pr1_pru1_gpi[20:0]	I	PRU1 Register R31 Inputs	HiZ
pr1_mii_mr0_clk	I	MII0 Receive Clock	HiZ
pr1_mii0_rxdv	I	MII0 Receive Data Valid	HiZ
pr1_mii0_rxd[0:3]	I	MII0 Receive Data	HiZ
pr1_mii0_rxlink	I	MII0 Receive Link	HiZ
pr1_mii0_rxer	I	MII0 Receive Data Error	HiZ
pr1_mii0_crs	I	MII0 Carrier Sense	HiZ
pr1_mii0_col	I	MII0 Carrier Sense	HiZ
pr1_mii_mt0_clk	I	MII0 Transmit Clock	HiZ
pr1_mii0_txen	O	MII0 Transmit Enable	0
pr1_mii0_txd[0:3]	O	MII0 Transmit Data	0
pr1_mii_mr1_clk	I	MII1 Receive Clock	HiZ
pr1_mii1_rxdv	I	MII1 Receive Data Valid	HiZ
pr1_mii1_rxd[0:3]	I	MII1 Receive Data	HiZ
pr1_mii1_rxlink	I	MII1 Receive Link	HiZ
pr1_mii1_rxer	I	MII1 Receive Data Error	HiZ
pr1_mii1_crs	I	MII1 Carrier Sense	HiZ
pr1_mii1_col	I	MII1 Carrier Sense	HiZ
pr1_mii_mt1_clk	I	MII1 Transmit Clock	HiZ
pr1_mii1_txen	O	MII1 Transmit Enable	0
pr1_mii1_txd[0:3]	O	MII1 Transmit Data	0
pr1_mdio_mdclk	O	MDIO Clk	0
pr1_mdio_data	I/O	MDIO Data	HiZ
pr1_edio_sof	O	ECAT Digital I/O Start of Frame	0
pr1_edio_latch_in	I	ECAT Digital I/O Latch In	HiZ
pr1_edio_data_in[0:7]	I	ECAT Digital I/Os Data In	HiZ
pr1_edio_data_out[0:7]	O	ECAT Digital I/Os Data Out	0
pr1_edc_sync0_out	O	ECAT Distributed Clock Sync Out	0
pr1_edc_sync1_out	O	ECAT Distributed Clock Sync Out	0
pr1_edc_latch0_in	I	ECAT Distributed Clock Latch In	HiZ
pr1_edc_latch1_in	I	ECAT Distributed Clock Latch In	HiZ
pr1_uart0_cts_n	I	UART Clear to Send	HiZ
pr1_uart0_rts_n	O	UART Request to Send	0
pr1_uart0_rxd	I	UART Receive Data	HiZ
pr1_uart0_txd	O	UART Transmit Data	0
pr1_ecap0_ecap_capin_apwm_o	I/O	Enhanced capture (ECAP) input or Auxiliary PWM out	HiZ

Table 30-473. PRU-ICSS2 I/O Signals

Device Level Signal Name	I/O	Description	Value at Reset
pr2_pru0_gpo[20:0]	O	PRU0 Register R30 Outputs	0
pr2_pru0_gpi[20:0]	I	PRU0 Register R31 Inputs	HiZ
pr2_pru1_gpo[20:0]	O	PRU1 Register R30 Outputs	0
pr2_pru1_gpi[20:0]	I	PRU1 Register R31 Inputs	HiZ

Table 30-473. PRU-ICSS2 I/O Signals (continued)

Device Level Signal Name	I/O	Description	Value at Reset
pr2_mii_mr0_clk	I	MII0 Receive Clock	HiZ
pr2_mii0_rxdv	I	MII0 Receive Data Valid	HiZ
pr2_mii0_rxd[0:3]	I	MII0 Receive Data	HiZ
pr2_mii0_rxlink	I	MII0 Receive Link	HiZ
pr2_mii0_rxer	I	MII0 Receive Data Error	HiZ
pr2_mii0_crs	I	MII0 Carrier Sense	HiZ
pr2_mii0_col	I	MII0 Carrier Sense	HiZ
pr2_mii_mt0_clk	I	MII0 Transmit Clock	HiZ
pr2_mii0_txen	O	MII0 Transmit Enable	0
pr2_mii0_txd[0:3]	O	MII0 Transmit Data	0
pr2_mii_mr1_clk	I	MII1 Receive Clock	HiZ
pr2_mii1_rxdv	I	MII1 Receive Data Valid	HiZ
pr2_mii1_rxd[0:3]	I	MII1 Receive Data	HiZ
pr2_mii1_rxlink	I	MII1 Receive Link	HiZ
pr2_mii1_rxer	I	MII1 Receive Data Error	HiZ
pr2_mii1_crs	I	MII1 Carrier Sense	HiZ
pr2_mii1_col	I	MII1 Carrier Sense	HiZ
pr2_mii_mt1_clk	I	MII1 Transmit Clock	HiZ
pr2_mii1_txen	O	MII1 Transmit Enable	0
pr2_mii1_txd[0:3]	O	MII1 Transmit Data	0
pr2_mdio_mdclk	O	MDIO Clk	0
pr2_mdio_data	I/O	MDIO Data	HiZ
pr2_edio_sof	O	ECAT Digital I/O Start of Frame	0
pr2_edio_latch_in	I	ECAT Digital I/O Latch In	HiZ
pr2_edio_data_in[0:7]	I	ECAT Digital I/Os Data In	HiZ
pr2_edio_data_out[0:7]	O	ECAT Digital I/Os Data Out	0
pr2_edc_sync0_out	O	ECAT Distributed Clock Sync Out	0
pr2_edc_sync1_out	O	ECAT Distributed Clock Sync Out	0
pr2_edc_latch0_in	I	ECAT Distributed Clock Latch In	HiZ
pr2_edc_latch1_in	I	ECAT Distributed Clock Latch In	HiZ
pr2_uart0_cts_n	I	UART Clear to Send	HiZ
pr2_uart0_rts_n	O	UART Request to Send	0
pr2_uart0_rxd	I	UART Receive Data	HiZ
pr2_uart0_txd	O	UART Transmit Data	0
pr2_ecap0_ecap_capin_apwm_o	I/O	Enhanced capture (ECAP) input or Auxiliary PWM out	HiZ

NOTE: The device I/O logic maps the PRU-ICSS signals to the different device pads by programming in the Control Module. For more information, refer to the Section 18.4.6.1.1, *Pad Configuration Registers* in the Chapter 18, *Control Module*.

Figure 30-56 illustrates the PRU-ICSS1 I/O interface signals at the device boundary.

Figure 30-56. PRU-ICSS1 External Interface I/Os

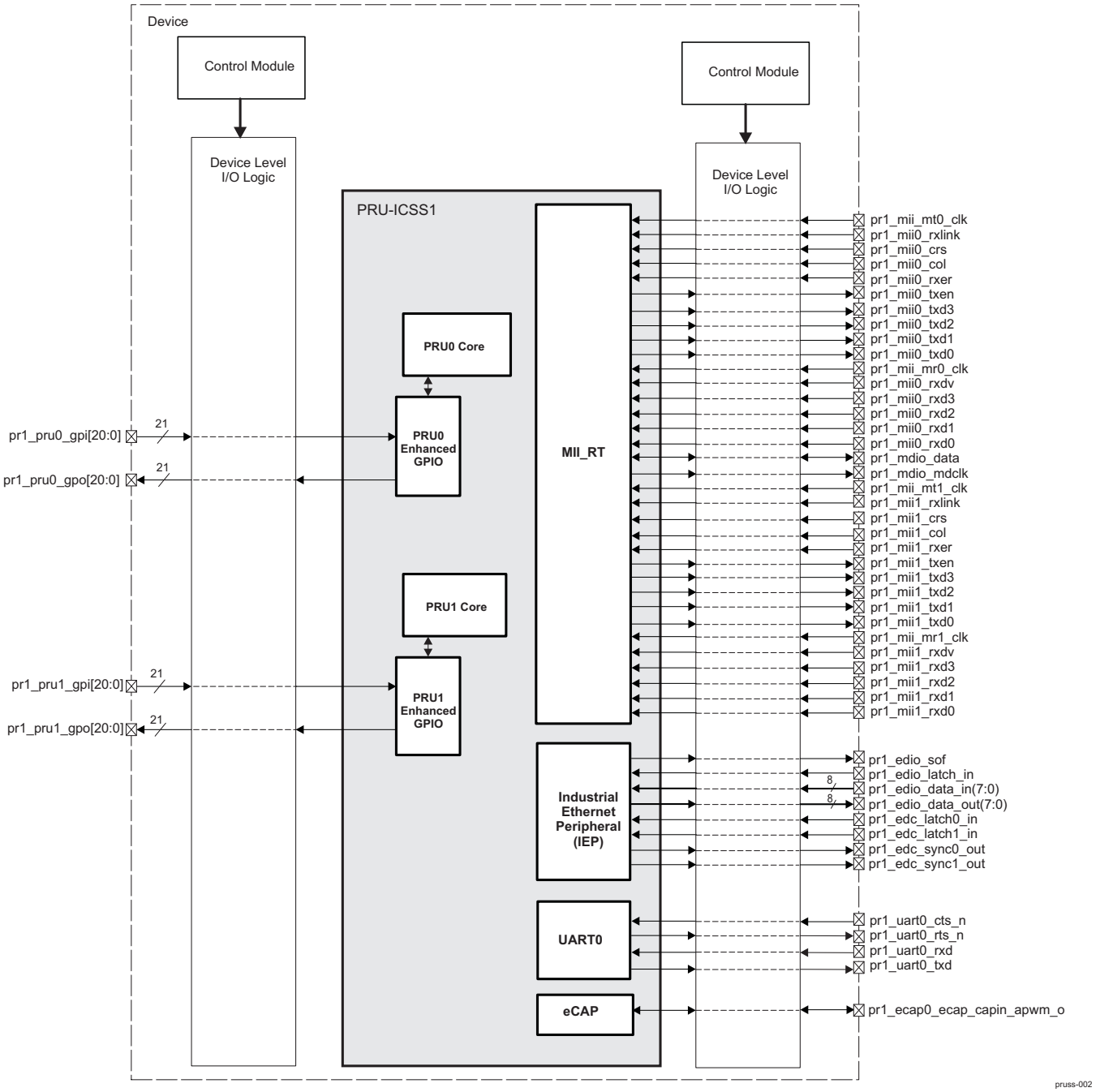
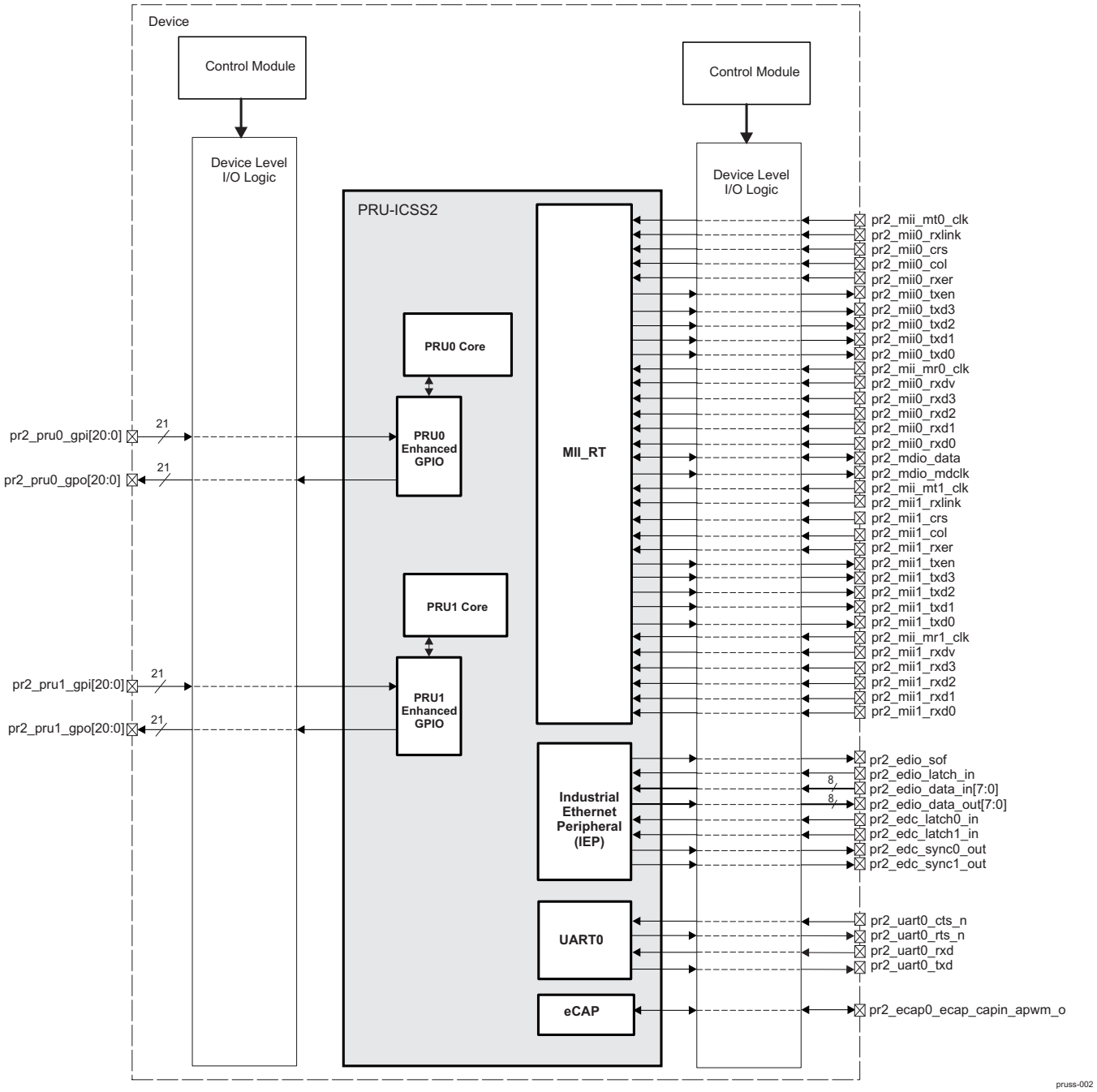


Figure 30-57 illustrates the PRU-ICSS2 I/O interface signals at the device boundary.

Figure 30-57. PRU-ICSS2 External Interface I/Os



The I/O signals are identically exported for both PRU-ICSS1 and PRU-ICSS2 subsystems.

30.2.3 PRU-ICSS Integration

The PRU-ICSS1 and PRU-ICSS2 subsystems integration in the device is shown in Figure 30-58 and Figure 30-59, respectively.

Figure 30-58. PRU-ICSS1 Integration in the Device

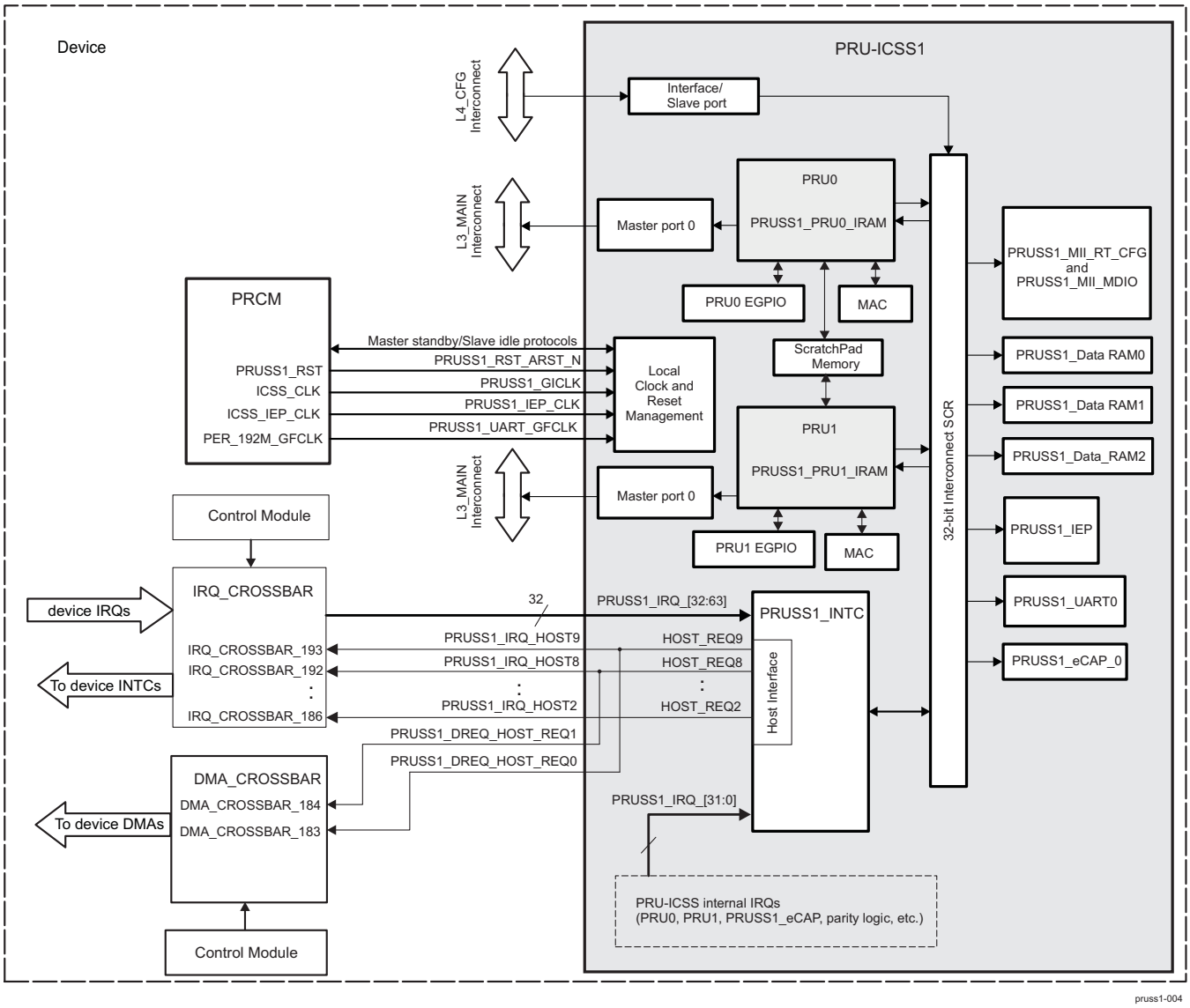
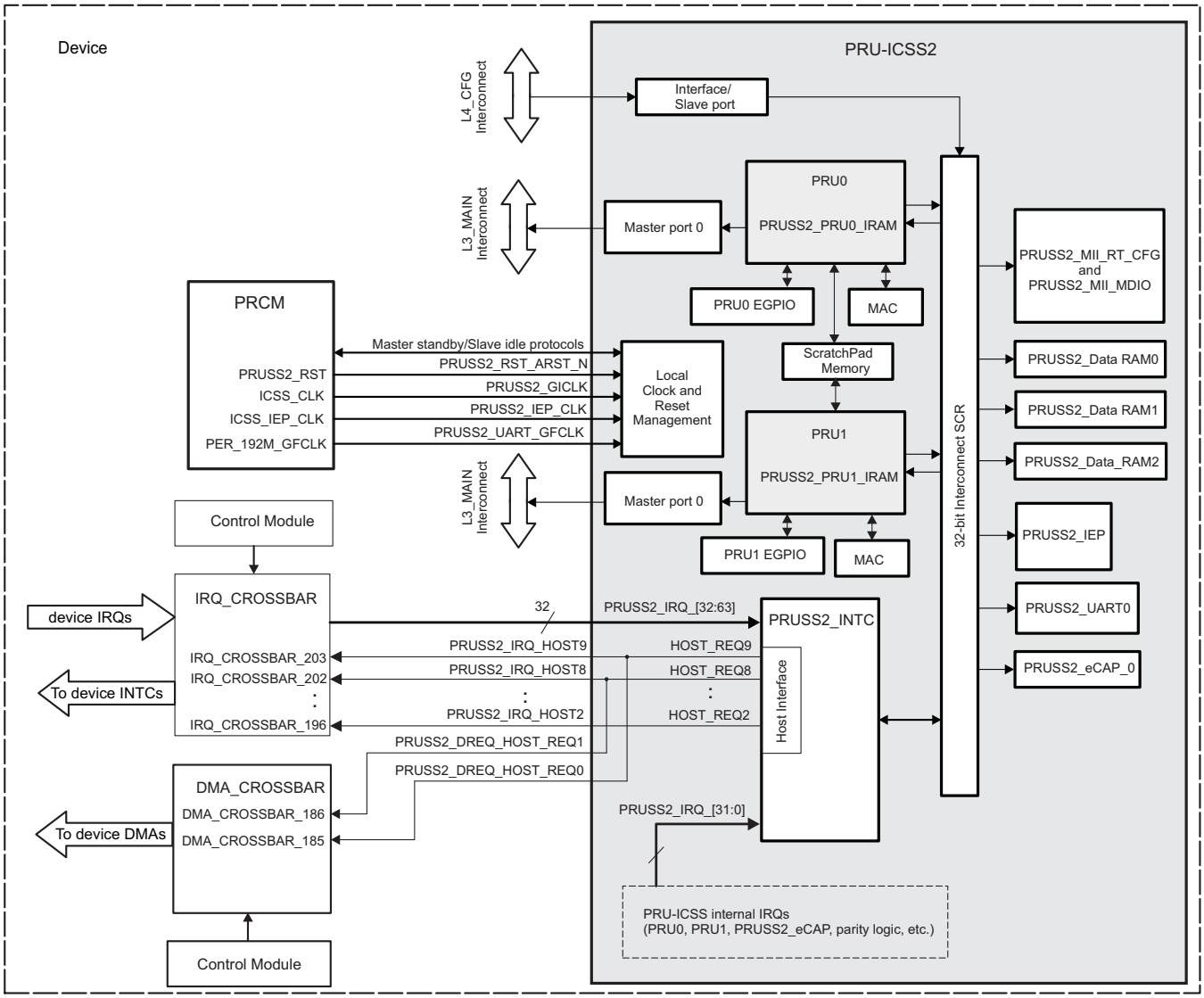


Figure 30-59. PRU-ICSS2 Integration in the Device



The PRU-ICSS1 and PRU-ICSS2 integration in the device features:

- PD_L4PER power domain instantiation
- two master ports (PRU0 and PRU1 core initiators) on the device L3_MAIN interconnect
- Non-wakeup capable - smart Standby protocol with the device PRCM
- Software assertion of a standby request "MStandby" (for master port clock disable) with local (PRU-ICSS) monitoring of the PRCM "MWait" acknowledge.
- one slave (configuration) port on the L3_MAIN interconnect for device hosts (MPU, DSP1, etc.) to access various memories and registers of PRU-ICSS
- Non-wakeup capable - smart Idle protocol with the device PRCM
- 10 output interrupt events from local interrupt controller - PRUSS_INTC:
  - 2 events to each PRU core (events 0 and 1)
  - 8 events mapped to the device IRQ_CROSSBAR which further remaps them to device interrupt controllers (events 2 through 9)
  - 2 events mapped to the device DMA_CROSSBAR, that remaps them to device DMA controllers (events 8 and 9)

- 32 external interrupts are mapped via the device IRQ_CROSSBAR to the local PRUSS_INTC
- A local software gating of clocks to several modules within PRU subsystem (local clock management protocol), as follows:
  - PRUSS_IEP
  - PRUSS_eCAP_0
  - PRUSS_UART0
  - PRUSS_INTC
  - PRUSS_PRU0
  - PRUSS_PRU1
- 3 input clocks obtained from device PRCM:
  - a PRU-ICSS top level gatable interface clock
  - a PRUSS IEP functional clock
  - a PRUSS UART0
- No memory/register retention is supported
- One hardware non-retention (level sensitive) reset

Table 30-474 through Table 30-476 summarize the integration of the module in the device.

Table 30-474. PRU-ICSS Integration Attributes

Module Instance	Attributes	
	Power Domain	Interconnect
PRU-ICSS1	PD_COREAON	L3_MAIN L4_CFG
PRU-ICSS2	PD_COREAON	L3_MAIN L4_CFG

Table 30-475. PRU-ICSS Clocks and Resets

Clocks				
Module Instance	**Destination Signal Name**	**Source Signal Name**	**Source**	**Description**
PRU-ICSS1	PRUSS1_GICLK	ICSS_CLK	PRCM	PRU-ICSS1 gated interface clock derived from DPLL_GMAC
	PRUSS1_UART_GFCLK	PER_192M_GFCLK	PRCM	PRUSS1_UART gated functional clock derived from DPLL_PER
	PRUSS1_IEP_CLK	ICSS_IEP_CLK	PRCM	PRUSS1_IEP functional clock derived from DPLL_GMAC
	PRUSS1_MII_MR0_CLK	pr1_mii_mr0_clk	pr1_mii_mr0_clk pin	MII0 RT RX functional clock
	PRUSS1_MII_MR1_CLK	pr1_mii_mr1_clk	pr1_mii_mr1_clk pin	MII1 RT RX functional clock
	PRUSS1_MII_MT0_CLK	pr1_mii_mt0_clk	pr1_mii_mt0_clk pin	MII0 RT TX functional clock
	PRUSS1_MII_MT1_CLK	pr1_mii_mt1_clk	pr1_mii_mt1_clk pin	MII1 RT TX functional clock
PRU-ICSS2	PRUSS2_GICLK	ICSS_CLK	PRCM	PRU-ICSS2 gated interface clock derived from DPLL_GMAC
	PRUSS2_UART_GFCLK	PER_192M_GFCLK	PRCM	PRUSS2_UART gated functional clock derived from DPLL_PER
	PRUSS2_IEP_CLK	ICSS_IEP_CLK	PRCM	PRUSS2_IEP functional clock derived from DPLL_GMAC
	PRUSS2_MII_MR0_CLK	pr2_mii_mr0_clk	pr2_mii_mr0_clk pin	MII0 RT RX functional clock
	PRUSS2_MII_MR1_CLK	pr2_mii_mr1_clk	pr2_mii_mr1_clk pin	MII1 RT RX functional clock
	PRUSS2_MII_MT0_CLK	pr2_mii_mt0_clk	pr2_mii_mt0_clk pin	MII0 RT TX functional clock
	PRUSS2_MII_MT1_CLK	pr2_mii_mt1_clk	pr2_mii_mt1_clk pin	MII1 RT TX functional clock
Resets				

Table 30-475. PRU-ICSS Clocks and Resets (continued)

Module Instance	Destination Signal Name	Source Signal Name	Source	Description
PRU-ICSS1	PRUSS1_RST_MAIN_ARST_N	PRUSS1_RST	PRCM	Non-retention hardware main reset to the PRU-ICSS1
PRU-ICSS2	PRUSS2_RST_MAIN_ARST_N	PRUSS2_RST	PRCM	Non-retention hardware main reset to the PRU-ICSS2

Table 30-476. PRU-ICSS Hardware Requests

Interrupt Requests				
Module Instance	**Source Signal Name**	**IRQ_CROSSBAR Input**	**Default Mapping**	**Description**
PRU-ICSS1	PRUSS1_IRQ_HOST2	IRQ_CROSSBAR_186	-	PRU-ICSS1 Host interrupt 2
	PRUSS1_IRQ_HOST3	IRQ_CROSSBAR_187	-	PRU-ICSS1 Host interrupt 3
	PRUSS1_IRQ_HOST4	IRQ_CROSSBAR_188	-	PRU-ICSS1 Host interrupt 4.
	PRUSS1_IRQ_HOST5	IRQ_CROSSBAR_189	-	PRU-ICSS1 Host interrupt 5
	PRUSS1_IRQ_HOST6	IRQ_CROSSBAR_190	-	PRU-ICSS1 Host interrupt 6
	PRUSS1_IRQ_HOST7	IRQ_CROSSBAR_191	-	PRU-ICSS1 Host interrupt 7
	PRUSS1_IRQ_HOST8	IRQ_CROSSBAR_192	-	PRU-ICSS1 Host interrupt 8
	PRUSS1_IRQ_HOST9	IRQ_CROSSBAR_193	-	PRU-ICSS1 Host interrupt 9
PRU-ICSS2	PRUSS2_IRQ_HOST2	IRQ_CROSSBAR_196	-	PRU-ICSS2 Host interrupt 2
	PRUSS2_IRQ_HOST3	IRQ_CROSSBAR_197	-	PRU-ICSS2 Host interrupt 3
	PRUSS2_IRQ_HOST4	IRQ_CROSSBAR_198	-	PRU-ICSS2 Host interrupt 4
	PRUSS2_IRQ_HOST5	IRQ_CROSSBAR_199	-	PRU-ICSS2 Host interrupt 5
	PRUSS2_IRQ_HOST6	IRQ_CROSSBAR_200	-	PRU-ICSS2 Host interrupt 6
	PRUSS2_IRQ_HOST7	IRQ_CROSSBAR_201	-	PRU-ICSS2 Host interrupt 7
	PRUSS2_IRQ_HOST8	IRQ_CROSSBAR_202	-	PRU-ICSS2 Host interrupt 8
	PRUSS2_IRQ_HOST9	IRQ_CROSSBAR_203	-	PRU-ICSS2 Host interrupt 9
DMA Requests				
Module Instance	**Source Signal Name**	**DMA_CROSSBAR Input**	**Default Mapping**	**Description**
PRU-ICSS1	PRUSS1_DREQ_HOST_REQ0	DMA_CROSSBAR_183	-	PRU-ICSS1 Host DMA request 0. Source is host interrupt 9
	PRUSS1_DREQ_HOST_REQ1	DMA_CROSSBAR_184	-	PRU-ICSS1 Host DMA request 1. Source is host interrupt 8
PRU-ICSS2	PRUSS2_DREQ_HOST_REQ0	DMA_CROSSBAR_185	-	PRU-ICSS2 Host DMA request 0. Source is host interrupt 9
	PRUSS2_DREQ_HOST_REQ1	DMA_CROSSBAR_186	-	PRU-ICSS2 Host DMA request 1. Source is host interrupt 8

NOTE: For more information about the IRQ_CROSSBAR module, see Section 18.4.6.4, *IRQ_CROSSBAR Module Functional Description*, in Chapter 18, *Control Module*.

For more information about the DMA_CROSSBAR module, see Section 18.4.6.5, *DMA_CROSSBAR Module Functional Description*, in Chapter 18, *Control Module*.

For more information about the device interrupt controllers, see Chapter 17, *Interrupt Controllers*.

For more information about the device SDMA module, see Section 16.1, *System DMA*.

For more information about the device EDMA module, see Section 16.2, *Enhanced DMA*.

30.2.4 PRU-ICSS Level Resources Functional Description

This section provides functional description of the device integrated PRU Subsystems modules.

30.2.4.1 PRU-ICSS Reset Management

An individual PRCM cold and warm hardware reset is available per PRU-ICSS1 (PRUSS1_RST) and PRU-ICSS2 (PRUSS2_RST). It is asserted by the PRCM upon cold reset and warm reset events.

For more details on the PRU-ICSS resets mapping, see also the Section 30.2.3.

NOTE: A hardware reset event (PRUSS_RST_MAIN_ARST_N input assertion) forces the PRU-ICSS to go to an IDLE and STANDBY state. For more details on PRU-ICSS clock management defined states refer to the Section 30.2.4.2.

NOTE: No global software reset is available at the PRU-ICSS top level.

30.2.4.2 PRU-ICSS Power and Clock Management

The PRU-ICSS supports 2 levels of clock gating. First level gates all clocks inside the PRU-ICSS when it is placed into IDLE and STANDBY state. The second level allows user software to enable/disable clocks in the clock gating register PRUSS_CGR to some internal modules, as follows:

- PRUSS1_IEP
- PRUSS1_eCAP_0
- PRUSS1_UART0
- PRUSS1_INTC
- PRUSS1_PRU0_Control /PRUSS1_PRU0_IRAM
- PRUSS1_PRU1_Control/PRUSS1_PRU1_IRAM
- PRUSS2_IEP
- PRUSS2_eCAP_0
- PRUSS2_UART0
- PRUSS2_INTC
- PRUSS2_PRU0_Control /PRUSS2_PRU0_IRAM
- PRUSS2_PRU1_Control/PRUSS2_PRU1_IRAM

The appropriate configuration registers block controls its local module set inside PRU-ICSS.

30.2.4.2.1 PRU-ICSS Idle and Standby States

The below Table 30-477 lists the clock management settings applicable at PRU-ICSS subsystem level (first level of local power management).

NOTE: For more details on the slave idle protocol (slave port) and master standby protocol (master port) between PRU-ICSS and device PRCM, refer to the Section 3.1.1.1.2, *Module-Level Clock Management* in the Chapter 3, *Power, Reset and Clock Management*.

Table 30-477. PRU-ICSS Idle/Standby Support

IDLE/STANDBY Mode	Comments
NO IDLE	
SMART IDLE	Default State
Wake-up capable SMART IDLE	NOT supported
FORCE IDLE	

Table 30-477. PRU-ICSS Idle/Standby Support (continued)

IDLE/STANDBY Mode	Comments
NO STANDBY	
SMART STANDBY	Default State
Wake-up capable SMART STANDBY	NOT supported
FORCE STANDBY	

NOTE: Not all of the PRU-ICSS outputs meet the IDLE state. Only the power protocol and L3_MAIN signals are Idled with all functional and interface clocks being shut-down.

A transition from an ACTIVE/Normal state to an IDLE (L3_MAIN slave) + STANDBY (L3_MAIN masters) state is performed as per the sequence:

1. The host (i.e. device MPU, DSP1, etc.) requests that the PRU firmware goes into IDLE state and waits for acknowledgement.
2. The host issues Clock Stop Request in register PRUSS_CGR to modules with gateable clocks defined at second power management level (see Section 30.2.4.2)
3. The host initiates MStandby via assertion to HIGH of the bit: PRUSS_SYSCFG [4] STANDBY_INIT (PRU-ICSS clock management configuration register).
4. The host software gets the device PRCM to issue an IDLE Request (SidleReq) towards the PRU-ICSS slave port. This is done via writing to device PRCM clock management registers dedicated to PRU-ICSS: CM_L4PER2_PRUSS1_CLKCTRL and CM_L4PER2_PRUSS2_CLKCTRL in Section 3.14, *PRCM Register Manual* of the Chapter 3,*Power Reset and Clock Management*.
5. The PRU-ICSS acknowledges IDLE Request and enters the IDLE+STANDBY state.

A transition from an IDLE + STANDBY state to an ACTIVE/ Normal state is performed as per the sequence:

1. The host (i.e. device MPU, DSP1, etc.) software gets the PRCM to de-assert IDLE Request - This is done via writing to device PRCM clock management registers dedicated to PRU-ICSS: CM_L4PER2_PRUSS1_CLKCTRLand CM_L4PER2_PRUSS2_CLKCTRL in Section 3.14, *PRCM Register Manual* of the Chapter 3,*Power Reset and Clock Management*.
2. The host CPU de-asserts the ClockStopReq to modules with gateable clocks defined at second power management level, and wait for the ClockStopAck to be asserted. This is done via PRU-ICSS host writing/reading the PRUSS_CGR.
3. The host CPU enables "NO STANDBY" via assertion of the PRUSS_SYSCFG[3:2] STANDBY_MODE to 0x1.

30.2.4.2.2 Module Clock Configurations at PRU-ICSS Top Level

IEP functional clock source selection: The clock source selection between PRUSS_IEP_CLK (default) and PRUSS_GICLK to the IEP module is done in register PRUSS_IEPCLK[0] OCP_EN in the PRUSS_CFG location. For more information on these PRU-ICSS level input clocks from PRCM, refer to the Section 30.2.3.

Enhanced GPIO clock divider settings: In certain sample/shift clock settings of the PRU0 and PRU1 EGPIOs (when enabled in serial mode) two cascaded fractional dividers are done in the PRUSS_CFG top level configuration registers PRUSS_GPCFG0 and PRUSS_GPCFG1. In addition, EGPIO clock active edge selection control can be exerted via the bit PRU0_GPI_CLK_MODE for PRU0_EGPIO and PRU1_GPI_CLK_MODE for the PRU1_EGPIO.

- For the serial PRU0's EGPOs:
  - PRUSS_GPCFG0 [24:20]PRU0_GPO_DIV1
  - PRUSS_GPCFG0 [19:15]PRU0_GPO_DIV0
- For the serial PRU0's EGPIs:

  - PRUSS_GPCFG0 [12:8]PRU0_GPI_DIV1
  - PRUSS_GPCFG0 [7:3]PRU0_GPI_DIV0
- For the serial PRU1's EGPOs:
  - PRUSS_GPCFG1 [24:20]PRU1_GPO_DIV1
  - PRUSS_GPCFG1 [19:15]PRU1_GPO_DIV0
- For the serial PRU1's EGPIs:
  - PRUSS_GPCFG1 [12:8]PRU1_GPI_DIV1
  - PRUSS_GPCFG1 [7:3]PRU1_GPI_DIV0

30.2.4.3 Other PRU-ICSS Module Functional Registers at Subsystem Level

Enhanced GPIO. The other functional mode setting for PRUs EGPIOs at PRU-ICSS top registers level are:

- PRUSS_GPCFG0/PRUSS_GPCFG1 [14] PRU1_GPO_MODE - to select between direct or serial EGPO output mode of operation.
- PRUSS_GPCFG0/PRUSS_GPCFG1 [25] PRU1_GPO_SH_SEL - to select between the EGPO shadow registers 0 and 1 used for output shifting. For more details, refer to the Section 30.2.5.2.2.3.4, *Enhanced General-Purpose Module Outputs (R30)*.
- PRUSS_GPCFG0/PRUSS_GPCFG1 [1:0] PRU1_GPI_MODE - selects the EGPI input mode of operation (selects between direct input, parallel capture, 28-bit shift or MII_RT modes).
- PRUSS_GPCFG0/PRUSS_GPCFG1 [13] PRU1_GPI_SB - **start bit event status for 28-bit EGPI input shift mode**. For more details, refer to the Section 30.2.5.2.2.3, *Enhanced General-Purpose Module Inputs (R31)*.

PRU 0/1 cores IRAM and DRAM parity error events: PRUSS_ISRP (raw status), PRUSS_ISP (interrupt status) and PRUSS_IESP (interrupt enable) and PRUSS_IECP (interrupt clear) registers.

PRU 0/1 cores IRAM and DRAM parity error events: PRUSS_ISRP (raw status), PRUSS_ISP (interrupt status) and PRUSS_IESP (interrupt enable) and PRUSS_IECP (interrupt clear) registers.

Enable address offset ("-0x0008_0000") feature individually per PRU0 and PRU1 master ports in the PRUSS_PMAO register in case of accessing peripherals located in the PRU-ICSS space.

PRUSS_MII_RT_CFG interrupts mapping to PRUSS_INTC is enabled in the PRUSS_MII_RT register

PRUs scratchpad (SPAD) memory priority and configuration related bits are located in the PRUSS_SPP register.

30.2.4.4 PRU-ICSS Memory Maps

The PRU-ICSS comprises various distinct addressable regions that are mapped to both a local and global memory map. The local memory maps are maps with respect to the PRU point of view. The global memory maps are maps with respect to the Host point of view, but can also be accessed by the PRU-ICSS.

30.2.4.4.1 PRU-ICSS Local Memory Map

The PRU-ICSS memory map is documented in Table 30-478 (Instruction Space) and in Table 30-479 (Data Space). Note that these two memory maps are implemented inside the PRU-ICSS and are local to the components of the PRU-ICSS.

30.2.4.4.1.1 PRU-ICSS Local Instruction Memory Map

Each PRU has a dedicated 12 KiB of Instruction Memory (PRUSS_PRU0_IRAM and PRUSS_PRU1_IRAM respectively) that must be initialized by an external to PRU-ICSS host processor before a PRU core CPU executes any instructions.

CAUTION

The PRUSS_PRU0/1_IRAM regions are ONLY accessible to PRU-ICSS masters (external hosts like MPU Cortex-A15, DSP1, etc.) when the PRU0/PRU1 is NOT running. The access is via PRU-ICSS slave port on the device L3_MAIN interconnect

Table 30-478. PRU-ICSS Local Instruction Memory Map

Start Address	PRU0	PRU1
0x0000_0000	12 KiB IRAM	12 KiB IRAM

30.2.4.4.1.2 PRU-ICSS Local Data Memory Map

The local data memory map in Table 30-479 allows each PRU core to access the PRU-ICSS addressable regions and the external host's memory map.

The PRU accesses the external Host memory map through the device L3_MAIN interconnect Interface Master port (System OCP_HP0/1) starting at address 0x0008_0000. By default, memory addresses between 0x0000_0000 – 0x0007_FFFF will correspond to the PRU-ICSS local address in Table 30-479. To access an address between 0x0000_0000–0x0007_FFFF of the external host map, the address offset of " – 0x0008_0000" feature is enabled through the PRUSS_PMAO[1] PMAO_PRU1 (for PRU1 CPU) and PRUSS_PMAO[0] PMAO_PRU0 (for PRU0 CPU) bits in the PRUSS_CFG subsystem level register space.

Table 30-479. PRU-ICSS Local Data Memory Map

Start Address	PRUSS_PRU0	PRUSS_PRU1
0x0000_0000	Data 8 KiB RAM0	Data 8 KiB RAM1
0x0000_2000	Data 8 KiB RAM1(1)	Data 8 KiB RAM0(1)
0x0000_4000	Reserved	Reserved
0x0001_0000	Data 32 KiB RAM2 (Shared RAM)	Data 32 KiB RAM2 (Shared RAM)
0x0002_0000	PRUSS_INTC	PRUSS_INTC
0x0002_2000	PRU0 Control	PRU0 Control
0x0002_2400	Reserved	Reserved
0x0002_4000	PRU1 Control	PRU1 Control
0x0002_4400	Reserved	Reserved
0x0002_6000	CFG	CFG
0x0002_8000	UART0	UART0
0x0002_A000	Reserved	Reserved
0x0002_C000	Reserved	Reserved
0x0002_E000	IEP	IEP
0x0003_0000	eCAP0	eCAP0
0x0003_2000	MII_RT_CFG	MII_RT_CFG
0x0003_2400	MII_MDIO	MII_MDIO
0x0003_4000	Reserved	Reserved
0x0003_7000	Reserved	Reserved
0x0003_8000	Reserved	Reserved
0x0003_B000	Reserved	Reserved
0x0004_0000	External PRU subsystem	External PRU subsystem
0x0008_0000	PRU-ICSS master port 0 on device L3_MAIN interconnect (OCP_HP0)(2)	PRU-ICSS master port 1 on device L3_MAIN interconnect (OCP_HP1)(2)

(1) Direct access from PRU0 to Data RAM 1 and PRU1 to Data RAM 0.
(2) For details see Section 2.8, *PRU-ICSS Memory Map.*

30.2.4.4.2 PRU-ICSS Global Memory Map

The global view of the PRU-ICSS internal memories and control ports is shown in Table 30-480. The offset addresses of each region are implemented inside the PRU-ICSS but the global device **L3_MAIN memory mapping** places the PRU-ICSS slave port in the address range shown in the external PRU-ICSS host L3_MAIN memory map.

The global memory map is with respect to the Host point of view (i.e. device MPU Cortex-A15, DSP1, etc. view of PRU-ICSS1/PRU-ICSS2 in the L3_MAIN memory space), but it can also be accessed by the PRU-ICSS1/PRU-ICSS2 itself. This is implemented via L3_MAIN redirecting PRU-ICSS master port traffic in the address range (0x0008_0000 - 0x000B_FFFF) to the PRU-ICSS slave port when PMAO_PRU0/PMAO_PRU1 = '0b1'. Note that PRU0 and PRU1 can use either the local or global addresses to access their internal memories, but using the local addresses provides access time several cycles faster than using the global addresses. This is because when accessing via the global address the access has to be routed through the L3_MAIN switch fabric outside PRU-ICSS and back in through the PRU-ICSS slave port.

Example 1: PRU1 accesses its own data RAM - Data_RAM1 in the global space:

- The PRU1 CPU sets the PRUSS_PMAO[1] PMAO_PRU1 to 1 (using PRU1 local CFG address: 0x0002_6028 of that register) and generates destination address 0x0008_2000. Thus, traffic passes through master port 1 towards PRU-ICSS1 slave port over L3_MAIN to reach Data_RAM1 (location 0x0008_2000 - 0x0008_0000 = 0x0000_2000 in the Table 30-480).

Example 2: PRU1 accesses PRU0 data RAM - Data_RAM0 in the global space:

- The PRU1 CPU sets the PRUSS_PMAO[1] PMAO_PRU1 to 1 (using PRU1 local CFG MMR address: 0x0002_6028 of that register) and generates destination address 0x0008_0000. Thus traffic passes through master port 1 towards PRU-ICSS1 slave port over L3_MAIN to reach Data_RAM0.

Example 3: DSP1 accesses the PRU0_IRAM in the global memory space to load instructions to be executed by the PRU0 upon boot time:

- Because the DSP1 is an external host to PRU-ICSS1, it has to target at first place the PRU-ICSS configuration and memory space in the L3_MAIN space. For PRU-ICSS1, slave port the base address is 0x4B20_0000.
- According to the Table 30-480, the PRU0_PRUSS1_PRU0_IRAM_TARG offset is 0x0003_4000. Hereby the physical address that DSP1 must use to store the PRU0 booting instructions to PRU0_IRAM is 0x4B23_4000.

Example 4: PRU0 accesses a non-PRU-ICSS peripheral in the global space (address offset >= 0x2000_0000):

- To access the McASP1 config space the PRU0 keeps PRUSS_PMAO[0] PMAO_PRU0 at 0b0 and generates the McASP1 cfg slave base address 0x4580_0000. Thus traffic passes through master port 0 and reaches McASP1 config MMRs over L3_MAIN.

Example 5: PRUSS1_PRU1 host configures the PRU-ICSS2 module PRUSS2_MII_RT:

- Note that in case of PRUSS1_PRU0 accessing a PRU-ICSS2 peripheral, it must again disable the PMAO feature (writing at 0x0002_6028 PRUSS_PMAO[1] PMAO_PRU1 =0b0) and generate the physical address through its master port (1) adding global memory space offset of the IEP (0x0002_E000 from the Table 30-480) to the PRU-ICSS2 L3_MAIN base address (0x4B28_0000). The physical address generated from PRU-ICSS1 PRU1 therefore equals 0x4B2A E000).

Each of the PRU cores can access the rest of the device memory (including memory mapped peripheral and configuration registers) using the global memory space addresses. For details on the L3_MAIN base address of the PRU-ICSS slave configuration memory space, refer to the Chapter 2, *Memory Mapping*.

Table 30-480. PRU-ICSS Global Memory Map

Offset Address	Target
0x0000_0000	Data 8 KiB RAM0
0x0000_2000	Data 8 KiB RAM1
0x0001_0000	Data 32 KiB RAM2 (shared)
0x0002_0000	PRUSS_INTC

Table 30-480. PRU-ICSS Global Memory Map (continued)

Offset Address	Target
0x0002_2000	PRU0 Control
0x0002_2400	PRU0 Debug
0x0002_4000	PRU1 Control
0x0002_4400	PRU1 Debug
0x0002_6000	CFG
0x0002_8000	UART0
0x0002_A000	Reserved
0x0002_C000	Reserved
0x0002_E000	IEP
0x0003_0000	eCAP0
0x0003_2000	MII_RT_CFG
0x0003_2400	MII_MDIO
0x0003_4000	PRU0 12 KiB IRAM
0x0003_8000	PRU1 12 KiB IRAM
0x0004_0000	External PRU-ICSS

NOTE: The 0x0008_0000-offset-subtraction feature must be enabled only in case of PRU global accesses (0x0008_0000 - 0x000B_FFFF) to resources within the PRU subsystem. The PMAO feature must be disabled when accessing PRU-ICSS external locations.

30.2.4.5 PRUSS_CFG Register Manual

This section describes the PRU-ICSS subsystem top-level registers.

30.2.4.5.1 PRUSS_CFG Instance Summary

Table 30-481. PRUSS_CFG Instance Summary

Module Name	Base Address	Size
PRUSS1_CFG	0x4B22 6000	68 Bytes
PRUSS2_CFG	0x4B2A 6000	68 Bytes

30.2.4.5.2 PRUSS_CFG Registers

30.2.4.5.2.1 PRUSS_CFG Register Summary

Table 30-482. PRUSS_CFG Registers Mapping Summary 1

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS1_CFG Physical Address
PRUSS_REVID	R	32	0x0000 0000	0x4B22 6000
PRUSS_SYSCFG	RW	32	0x0000 0004	0x4B22 6004
PRUSS_GPCFG0	RW	32	0x0000 0008	0x4B22 6008
PRUSS_GPCFG1	RW	32	0x0000 000C	0x4B22 600C
PRUSS_CGR	RW	32	0x0000 0010	0x4B22 6010
PRUSS_ISRP	RW	32	0x0000 0014	0x4B22 6014
PRUSS_ISP	RW	32	0x0000 0018	0x4B22 6018

Table 30-482. PRUSS_CFG Registers Mapping Summary 1 (continued)

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS1_CFG Physical Address
PRUSS_IESP	RW	32	0x0000 001C	0x4B22 601C
PRUSS_IECP	RW	32	0x0000 0020	0x4B22 6020
RESERVED	RW	32	0x0000 0024	0x4B22 6024
PRUSS_PMAO	RW	32	0x0000 0028	0x4B22 6028
PRUSS_MII_RT	RW	32	0x0000 002C	0x4B22 602C
PRUSS_IEPCLK	RW	32	0x0000 0030	0x4B22 6030
PRUSS_SPP	RW	32	0x0000 0034	0x4B22 6034
PRUSS_PIN_MX	RW	32	0x0000 0040	0x4B22 6040

Table 30-483. PRUSS_CFG Registers Mapping Summary 2

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS2_CFG Physical Address
PRUSS_REVID	R	32	0x0000 0000	0x4B2A 6000
PRUSS_SYSCFG	RW	32	0x0000 0004	0x4B2A 6004
PRUSS_GPCFG0	RW	32	0x0000 0008	0x4B2A 6008
PRUSS_GPCFG1	RW	32	0x0000 000C	0x4B2A 600C
PRUSS_CGR	RW	32	0x0000 0010	0x4B2A 6010
PRUSS_ISRP	RW	32	0x0000 0014	0x4B2A 6014
PRUSS_ISP	RW	32	0x0000 0018	0x4B2A 6018
PRUSS_IESP	RW	32	0x0000 001C	0x4B2A 601C
PRUSS_IECP	RW	32	0x0000 0020	0x4B2A 6020
RESERVED	RW	32	0x0000 0024	0x4B2A 6024
PRUSS_PMAO	RW	32	0x0000 0028	0x4B2A 6028
PRUSS_MII_RT	RW	32	0x0000 002C	0x4B2A 602C
PRUSS_IEPCLK	RW	32	0x0000 0030	0x4B2A 6030
PRUSS_SPP	RW	32	0x0000 0034	0x4B2A 6034
PRUSS_PIN_MX	RW	32	0x0000 0040	0x4B2A 6040

30.2.4.5.2.2 PRUSS_CFG Register Description

Table 30-484. PRUSS_REVID

Address Offset	0x0000 0000		
Physical Address	0x4B22 6000 0x4B2A 6000	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The Revision Register contains the ID and revision information.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REVISION																															

Bits	Field Name	Description	Type	Reset
31:0	REVISION	IP Revision	R	0x-(1)

(1) TI Internal data

Table 30-485. Register Call Summary for Register PRUSS_REVID

PRU-ICSS Level Resources Functional Description
• PRUSS_CFG Register Summary: [0] [1]

Table 30-486. PRUSS_SYSCFG

Address Offset	0x0000 0004		
Physical Address	0x4B22 6004 0x4B2A 6004	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The System Configuration Register defines the power IDLE and STANDBY modes.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																										SUB_MWAIT	STANDBY_INIT	STANDBY_MODE		IDLE_MODE	

Bits	Field Name	Description	Type	Reset
31:6	RESERVED		R	0x0
5	SUB_MWAIT	Status bit for wait state. 0x0 = Ready for Transaction 0x1 = Wait until 0	R	0x0
4	STANDBY_INIT	0x1 = Initiate standby sequence. 0x0 = Enable OCP master ports.	RW	0x1
3:2	STANDBY_MODE	0x0 = Force standby mode: Initiator unconditionally in standby (standby = 1) 0x1 = No standby mode: Initiator unconditionally out of standby (standby = 0) 0x2 = Smart standby mode: Standby requested by initiator depending on internal conditions 0x3 = Reserved	RW	0x2
1:0	IDLE_MODE	0x0 = Force-idle mode 0x1 = No-idle mode 0x2 = Smart-idle mode 0x3 = Reserved	RW	0x2

Table 30-487. Register Call Summary for Register PRUSS_SYSCFG

PRU-ICSS Level Resources Functional Description
• PRU-ICSS Idle and Standby States: [0] [1]
• PRUSS_CFG Register Summary: [2] [3]

Table 30-488. PRUSS_GPCFG0

Address Offset	0x0000 0008		
Physical Address	0x4B22 6008 0x4B2A 6008	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The General Purpose Configuration 0 Register defines the GPIO configuration for PRU0.		
Type	RW		

31 30	29 28 27 26	25	24	23 22 21 20	19 18 17 16	15	14	13	12 11 10 9 8	7 6 5 4 3	2	1 0
RESERVED	PR1_PRU0_GP_MUX_SEL	PRU0_GPO_SH_SEL		PRU0_GPO_DIV1	PRU0_GPO_DIV0		PRU0_GPO_MODE	PRU0_GPI_SB	PRU0_GPI_DIV1	PRU0_GPI_DIV0	PRU0_GPI_CLK_MODE	PRU0_GPI_MODE

Bits	Field Name	Description	Type	Reset
31:30	RESERVED		R	0x0
29:26	PR1_PRU0_GP_MUX_SEL	Reserved. Keep at reset value.	R/W	0x0
25	PRU0_GPO_SH_SEL	Defines which shadow register is currently getting used for GPO shifting. 0x0 = gpo_sh0 is selected 0x1 = gpo_sh1 is selected	R	0x0
24:20	PRU0_GPO_DIV1	Divisor value (divide by PRU0_GPO_DIV1 + 1). 0x00 = div 1.0 0x01 = div 1.5 0x02 = div 2.0 .. 0x1e = div 16.0 0x1f = reserved	RW	0x0
19:15	PRU0_GPO_DIV0	Divisor value (divide by PRU0_GPO_DIV0 + 1). 0x00 = div 1.0 0x01 = div 1.5 0x02 = div 2.0 .. 0x1e = div 16.0 0x1f = reserved	RW	0x0
14	PRU0_GPO_MODE	0x0 = Direct output mode 0x1 = Serial output mode	RW	0x0
13	PRU0_GPI_SB	Start Bit event for 28-bit shift mode. PRU0_GPI_SB (pru0_r31_status[29]) is set when first capture of a 1 on pru0_r31_status[0]. Read 1: Start Bit event occurred. Read 0: Start Bit event has not occurred. Write 1: Will clear PRU0_GPI_SB and clear the whole shift register. Write 0: No Effect.	RW	0x0
12:8	PRU0_GPI_DIV1	Divisor value (divide by PRU0_GPI_DIV1 + 1). 0x00 = div 1.0 0x01 = div 1.5 0x02 = div 2.0 .. 0x1e = div 16.0 0x1f = reserved	RW	0x0
7:3	PRU0_GPI_DIV0	Divisor value (divide by PRU0_GPI_DIV0 + 1). 0x00 = div 1.0 0x01 = div 1.5 0x02 = div 2.0 .. 0x1e = div 16.0 0x1f = reserved	RW	0x0
2	PRU0_GPI_CLK_MODE	Parallel 16-bit capture mode clock edge. 0x0 = Use the positive edge of pru0_r31_status[16] 0x1 = Use the negative edge of pru0_r31_status[16]	RW	0x0
1:0	PRU0_GPI_MODE	0x0 = Direct input mode 0x1 = 16-bit parallel capture mode 0x2 = 28-bit shift mode 0x3 = MII_RT mode	RW	0x0

Table 30-489. Register Call Summary for Register PRUSS_GPCFG0

PRU-ICSS Environment
- PRU-ICSS I/O Interface:

PRU-ICSS Level Resources Functional Description
- Module Clock Configurations at PRU-ICSS Top Level: [1] [2] [3] [4] [5]
- Other PRU-ICSS Module Functional Registers at Subsystem Level: [6] [7] [8] [9]
- PRUSS_CFG Register Summary: [10] [11]

PRU-ICSS PRU Cores
- General-Purpose Inputs (R31): Enhanced PRU GP Module: [14]

Table 30-490. PRUSS_GPCFG1

Address Offset	0x0000 000C		
Physical Address	0x4B22 600C 0x4B2A 600C	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The General Purpose Configuration 1 Register defines the GPI O configuration for PRU1.		

Table 30-490. PRUSS_GPCFG1 (continued)

Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		PR1_PRU1_GP_MUX_SEL				PRU1_GPO_SH_SEL		PRU1_GPO_DIV1				PRU1_GPO_DIV0				PRU1_GPO_MODE		PRU1_GPI_SB		PRU1_GPI_DIV1				PRU1_GPI_DIV0					PRU1_GPI_CLK_MODE		PRU1_GPI_MODE	

Bits	Field Name	Description	Type	Reset
31:30	RESERVED		R	0x0
29:26	PR1_PRU1_GP_MUX_SEL	Reserved. Keep at reset value.	R/W	0x0
25	PRU1_GPO_SH_SEL	Defines which shadow register is currently getting used for GPO shifting. 0x0 = gpo_sh0 is selected 0x1 = gpo_sh1 is selected	R	0x0
24:20	PRU1_GPO_DIV1	Divisor value (divide by PRU1_GPO_DIV1 + 1). 0x00 = div 1.0 0x01 = div 1.5 0x02 = div 2.0 .. 0x1e = div 16.0 0x1f = reserved	RW	0x0
19:15	PRU1_GPO_DIV0	Divisor value (divide by PRU1_GPO_DIV0 + 1). 0x00 = div 1.0 0x01 = div 1.5 0x02 = div 2.0 .. 0x1e = div 16.0 0x1f = reserved	RW	0x0
14	PRU1_GPO_MODE	0x0 = Direct output mode 0x1 = Serial output mode	RW	0x0
13	PRU1_GPI_SB	28-bit shift mode Start Bit event. PRU1_GPI_SB (pru1_r31_status[29]) is set when first capture of a 1 on pru1_r31_status[0]. Read 1: Start Bit event occurred. Read 0: Start Bit event has not occurred. Write 1: Will clear PRU1_GPI_SB and clear the whole shift register. Write 0: No Effect.	RW	0x0
12:8	PRU1_GPI_DIV1	Divisor value (divide by PRU1_GPI_DIV1 + 1). 0x00 = div 1.0 0x01 = div 1.5 0x02 = div 2.0 .. 0x1e = div 16.0 0x1f = reserved	RW	0x0
7:3	PRU1_GPI_DIV0	Divisor value (divide by PRU1_GPI_DIV0 + 1). 0x00 = div 1.0 0x01 = div 1.5 0x02 = div 2.0 .. 0x1e = div 16.0 0x1f = reserved	RW	0x0
2	PRU1_GPI_CLK_MODE	Parallel 16-bit capture mode clock edge. 0x0 = Use the positive edge of pru1_r31_status[16] 0x1 = Use the negative edge of pru1_r31_status[16]	RW	0x0
1:0	PRU1_GPI_MODE	0x0 = Direct input mode 0x1 = 16-bit parallel capture mode 0x2 = 28-bit shift mode 0x3 = MII_RT mode	RW	0x0

Table 30-491. Register Call Summary for Register PRUSS_GPCFG1

PRU-ICSS Environment

- PRU-ICSS I/O Interface:

PRU-ICSS Level Resources Functional Description

- Module Clock Configurations at PRU-ICSS Top Level: [2] [3] [4] [5] [6]
- Other PRU-ICSS Module Functional Registers at Subsystem Level: [7] [8] [9] [10]
- PRUSS_CFG Register Summary: [11] [12]

Table 30-492. PRUSS_CGR

Address Offset	0x0000 0010		
Physical Address	0x4B22 6010 0x4B2A 6010	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The Clock Gating Register controls the state of Clock Management of the different modules. Software should not clear {module}_CLK_EN until {module}_CLK_STOP_ACK is 0x1.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														IEP_CLK_EN	IEP_CLK_STOP_ACK	IEP_CLK_STOP_REQ	ECAP_CLK_EN	ECAP_CLK_STOP_ACK	ECAP_CLK_STOP_REQ	UART_CLK_EN	UART_CLK_STOP_ACK	UART_CLK_STOP_REQ	PRUSS_INTC_CLK_EN	PRUSS_INTC_CLK_STOP_ACK	PRUSS_INTC_CLK_STOP_REQ	PRU1_CLK_EN	PRU1_CLK_STOP_ACK	PRU1_CLK_STOP_REQ	PRU0_CLK_EN	PRU0_CLK_STOP_ACK	PRU0_CLK_STOP_REQ

Bits	Field Name	Description	Type	Reset
31:18	RESERVED		R	0x0
17	IEP_CLK_EN	IEP clock enable. 0x0 = Disable Clock 0x1 = Enable Clock	RW	0x1
16	IEP_CLK_STOP_ACK	Acknowledgement that IEP clock can be stopped. 0x0 = Not Ready to Gate Clock 0x1 = Ready to Gate Clock	R	0x0
15	IEP_CLK_STOP_REQ	IEP request to stop clock. 0x0 = do not request to stop Clock 0x1 = request to stop Clock	RW	0x0
14	ECAP_CLK_EN	ECAP clock enable. 0x0 = Disable Clock 0x1 = Enable Clock	RW	0x1
13	ECAP_CLK_STOP_ACK	Acknowledgement that ECAP clock can be stopped. 0x0 = Not Ready to Gate Clock 0x1 = Ready to Gate Clock	R	0x0
12	ECAP_CLK_STOP_REQ	ECAP request to stop clock. 0x0 = do not request to stop Clock 0x1 = request to stop Clock	RW	0x0
11	UART_CLK_EN	UART clock enable. 0x0 = Disable Clock 0x1 = Enable Clock	RW	0x1
10	UART_CLK_STOP_ACK	Acknowledgement that UART clock can be stopped. 0x0 = Not Ready to Gate Clock 0x1 = Ready to Gate Clock	R	0x0
9	UART_CLK_STOP_REQ	UART request to stop clock. 0x0 = do not request to stop Clock 0x1 = request to stop Clock	RW	0x0
8	PRUSS_INTC_CLK_EN	PRUSS_INTC clock enable. 0x0 = Disable Clock 0x1 = Enable Clock	RW	0x1
7	PRUSS_INTC_CLK_STOP_ACK	Acknowledgement that PRUSS_INTC clock can be stopped. 0x0 = Not Ready to Gate Clock 0x1 = Ready to Gate Clock	R	0x0
6	PRUSS_INTC_CLK_STOP_REQ	PRUSS_INTC request to stop clock. 0x0 = do not request to stop Clock 0x1 = request to stop Clock	RW	0x0
5	PRU1_CLK_EN	PRU1 clock enable. 0x0 = Disable Clock 0x1 = Enable Clock	RW	0x1
4	PRU1_CLK_STOP_ACK	Acknowledgement that PRU1 clock can be stopped. 0x0 = Not Ready to Gate Clock 0x1 = Ready to Gate Clock	R	0x0
3	PRU1_CLK_STOP_REQ	PRU1 request to stop clock. 0x0 = do not request to stop Clock 0x1 = request to stop Clock	RW	0x0
2	PRU0_CLK_EN	PRU0 clock enable. 0x0 = Disable Clock 0x1 = Enable Clock	RW	0x1

Bits	Field Name	Description	Type	Reset
1	PRU0_CLK_STOP_ACK	Acknowledgement that PRU0 clock can be stopped. 0x0 = Not Ready to Gate Clock 0x1 = Ready to Gate Clock	R	0x0
0	PRU0_CLK_STOP_REQ	PRU0 request to stop clock. 0x0 = do not request to stop Clock 0x1 = request to stop Clock	RW	0x0

Table 30-493. Register Call Summary for Register PRUSS_CGR

PRU-ICSS Level Resources Functional Description

- PRU-ICSS Power and Clock Management: [0]
- PRU-ICSS Idle and Standby States: [1] [2]
- PRUSS_CFG Register Summary: [3] [4]

Table 30-494. PRUSS_ISRP

Address Offset	0x0000 0014		
Physical Address	0x4B22 6014 0x4B2A 6014	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The IRQ Status Raw Parity register is a snapshot of the IRQ raw status for the PRUSS memory parity events. The raw status is set even if the event is not enabled.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												RAM_PE_RAW				PRU1_DMEM_PE_RAW				PRU1_IMEM_PE_RAW				PRU0_DMEM_PE_RAW				PRU0_IMEM_PE_RAW			

Bits	Field Name	Description	Type	Reset
31:20	RESERVED		R	0
19:16	RAM_PE_RAW	RAM Parity Error RAW for Byte3, Byte2, Byte1, Byte0. Note RAM_PE_RAW[0] maps to Byte0. Write 0: No action. Read 0: No event pending. Read 1: Event pending. Write 1: Set event (debug).	RW	0x0
15:12	PRU1_DMEM_PE_RAW	PRU1 DMEM Parity Error RAW for Byte3, Byte2, Byte1, Byte0. Note PRU1_DMEM_PE_RAW[0] maps to Byte0. Write 0: No action. Read 0: No event pending. Read 1: Event pending. Write 1: Set event (debug).	RW	0x0
11:8	PRU1_IMEM_PE_RAW	PRU1 IMEM Parity Error RAW for Byte3, Byte2, Byte1, Byte0. Note PRU1_IMEM_PE_RAW[0] maps to Byte0. Write 0: No action. Read 0: No event pending. Read 1: Event pending. Write 1: Set event (debug).	RW	0x0
7:4	PRU0_DMEM_PE_RAW	PRU0 DMEM Parity Error RAW for Byte3, Byte2, Byte1, Byte0. Note PRU0_DMEM_PE_RAW[0] maps to Byte0. Write 0: No action. Read 0: No event pending. Read 1: Event pending. Write 1: Set event (debug) .	RW	0x0
3:0	PRU0_IMEM_PE_RAW	PRU0 IMEM Parity Error RAW for Byte3, Byte2, Byte1, Byte0. Note PRU0_IRAM_PE_RAW[0] maps to Byte0. Write 0: No action. Read 0: No event pending. Read 1: Event pending. Write 1: Set event (debug) .	RW	0x0

Table 30-495. Register Call Summary for Register PRUSS_ISRP

PRU-ICSS Level Resources Functional Description

- Other PRU-ICSS Module Functional Registers at Subsystem Level: [0] [1]
- PRUSS_CFG Register Summary: [2] [3]

Table 30-496. PRUSS_ISP

Address Offset	0x0000 0018		
Physical Address	0x4B22 6018 0x4B2A 6018	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The IRQ Status Parity Register is a snapshot of the IRQ status for the PRUSS memory parity events. The status is set only if the event is enabled. Write 1 to clear the status after the interrupt has been serviced.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												RAM_PE				PRU1_DMEM_PE				PRU1_IMEM_PE				PRU0_DMEM_PE				PRU0_IMEM_PE			

Bits	Field Name	Description	Type	Reset
31:20	RESERVED		R	0x0
19:16	RAM_PE	RAM Parity Error for Byte3, Byte2, Byte1, Byte0. Note RAM_PE[0] maps to Byte0. Write 0: No action. Read 0: No (enabled) event pending. Read 1: Event pending. Write 1: Clear event.	RW	0x0
15:12	PRU1_DMEM_PE	PRU1 DMEM Parity Error for Byte3, Byte2, Byte1, Byte0. Note PRU1_DMEM_PE[0] maps to Byte0. Write 0: No action. Read 0: No (enabled) event pending. Read 1: Event pending. Write 1: Clear event.	RW	0x0
11:8	PRU1_IMEM_PE	PRU1 IMEM Parity Error for Byte3, Byte2, Byte1, Byte0. Note PRU1_IMEM_PE[0] maps to Byte0. Write 0: No action. Read 0: No (enabled) event pending. Read 1: Event pending. Write 1: Clear event.	RW	0x0
7:4	PRU0_DMEM_PE	PRU0 DMEM Parity Error for Byte3, Byte2, Byte1, Byte0. Note PRU0_DMEM_PE[0] maps to Byte0. Write 0: No action. Read 0: No(enabled) event pending. Read 1: Event pending. Write 1: Clear event.	RW	0x0
3:0	PRU0_IMEM_PE	PRU0 IMEM Parity Error for Byte3, Byte2, Byte1, Byte0. Note PRU0_IMEM_PE[0] maps to Byte0. Write 0: No action. Read 0: No (enabled) event pending. Read 1: Event pending. Write 1: Clear event.	RW	0x0

Table 30-497. Register Call Summary for Register PRUSS_ISP

PRU-ICSS Level Resources Functional Description

- Other PRU-ICSS Module Functional Registers at Subsystem Level: [0] [1]
- PRUSS_CFG Register Summary: [2] [3]

Table 30-498. PRUSS_IESP

Address Offset	0x0000 001C		
Physical Address	0x4B22 601C 0x4B2A 601C	**Instance**	PRUSS1_CFG PRUSS2_CFG

Table 30-498. PRUSS_IESP (continued)

Description	The IRQ Enable Set Parity Register enables the IRQ PRUSS memory parity events.
Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												RAM_PE_SET				PRU1_DMEM_PE_SET				PRU1_IMEM_PE_SET				PRU0_DMEM_PE_SET				PRU0_IMEM_PE_SET			

Bits	Field Name	Description	Type	Reset
31:20	RESERVED		R	0x0
19:16	RAM_PE_SET	RAM Parity Error Set Enable for Byte3, Byte2, Byte1, Byte0. Note RAM_PE_SET[0] maps to Byte0. Write 0: No action. Read 0: Interrupt disabled (masked). Read 1: Interrupt enabled. Write 1: Enable interrupt.	RW	0x0
15:12	PRU1_DMEM_PE_SET	PRU1 DMEM Parity Error Set Enable for Byte3, Byte2, Byte1, Byte0. Note PRU1_DMEM_PE_SET[0] maps to Byte0. Write 0: No action. Read 0: Interrupt disabled (masked). Read 1: Interrupt enabled. Write 1: Enable interrupt.	RW	0x0
11:8	PRU1_IMEM_PE_SET	PRU1 IMEM Parity Error Set Enable for Byte3, Byte2, Byte1, Byte0. Note PRU1_IMEM_PE_SET[0] maps to Byte0. Write 0: No action. Read 0: Interrupt disabled (masked). Read 1: Interrupt enabled. Write 1: Enable interrupt.	RW	0x0
7:4	PRU0_DMEM_PE_SET	PRU0 DMEM Parity Error Set Enable for Byte3, Byte2, Byte1, Byte0. Note PRU0_DMEM_PE_SET[0] maps to Byte0. Write 0: No action. Read 0: Interrupt disabled (masked). Read 1: Interrupt enabled. Write 1: Enable interrupt.	RW	0x0
3:0	PRU0_IMEM_PE_SET	PRU0 IMEM Parity Error Set Enable for Byte3, Byte2, Byte1, Byte0. Note PRU0_IMEM_PE_SET[0] maps to Byte0. Write 0: No action. Read 0: Interrupt disabled (masked). Read 1: Interrupt enabled. Write 1: Enable interrupt.	RW	0x0

Table 30-499. Register Call Summary for Register PRUSS_IESP

PRU-ICSS Level Resources Functional Description

- Other PRU-ICSS Module Functional Registers at Subsystem Level: [0] [1]
- PRUSS_CFG Register Summary: [2] [3]

Table 30-500. PRUSS_IECP

Address Offset	0x0000 0020		
Physical Address	0x4B22 6020 0x4B2A 6020	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The IRQ Enable Clear Parity Register disables the IRQ PRUSS memory parity events.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PRU1_DMEM_PE_CLR				PRU1_IMEM_PE_CLR				PRU0_DMEM_PE_CLR				PRU0_IMEM_PE_CLR			

Bits	Field Name	Description	Type	Reset
31:16	RESERVED		R	0x0
15:12	PRU1_DMEM_PE_CLR	PRU1 DMEM Parity Error Clear Enable for Byte3, Byte2, Byte1, Byte0. Note PRU1_DMEM_PE_CLR[0] maps to Byte0. Write 0: No action. Read 0: Interrupt disabled (masked). Read 1: Interrupt enabled. Write 1: Disable interrupt.	RW	0x0
11:8	PRU1_IMEM_PE_CLR	PRU1 IMEM Parity Error Clear Enable for Byte3, Byte2, Byte1, Byte0. Note PRU1_IMEM_PE_CLR[0] maps to Byte0. Write 0: No action. Read 0: Interrupt disabled (masked). Read 1: Interrupt enabled. Write 1: Disable interrupt.	RW	0x0
7:4	PRU0_DMEM_PE_CLR	PRU0 DMEM Parity Error Clear Enable for Byte3, Byte2, Byte1, Byte0. Note PRU0_DMEM_PE_CLR[0] maps to Byte0. Write 0: No action. Read 0: Interrupt disabled (masked). Read 1: Interrupt enabled. Write 1: Disable interrupt.	RW	0x0
3:0	PRU0_IMEM_PE_CLR	PRU0 IMEM Parity Error Clear Enable for Byte3, Byte2, Byte1, Byte0. Note PRU0_IMEM_PE_CLR[0] maps to Byte0. Write 0: No action . Read 0: Interrupt disabled (masked). Read 1: Interrupt enabled. Write 1: Disable interrupt.	RW	0x0

Table 30-501. Register Call Summary for Register PRUSS_IECP

PRU-ICSS Level Resources Functional Description

- Other PRU-ICSS Module Functional Registers at Subsystem Level: [0] [1]
- PRUSS_CFG Register Summary: [2] [3]

Table 30-502. PRUSS_PMAO

Address Offset	0x0000 0028		
Physical Address	0x4B22 6028 0x4B2A 6028	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The PRU Master OCP Address Offset Register enables for the PRU OCP Master Port Address to have an offset of minus 0x0008_0000. This enables the PRU to access External Host address space starting at 0x0000_0000.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														PMAO_PRU1	PMAO_PRU0

Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0
1	PMAO_PRU1	PRU1 OCP Master Port Address Offset Enable. 0x0 = Disable address offset. 0x1 = Enable address offset of - 0x0008_0000.	RW	0x0
0	PMAO_PRU0	PRU0 OCP Master Port Address Offset Enable. 0x0 = Disable address offset. 0x1 = Enable address offset of - 0x0008_0000.	RW	0x0

Table 30-503. Register Call Summary for Register PRUSS_PMAO

PRU-ICSS Level Resources Functional Description
- Other PRU-ICSS Module Functional Registers at Subsystem Level: [0]
- PRU-ICSS Local Data Memory Map: [1] [2]
- PRU-ICSS Global Memory Map: [3] [4] [5] [6]
- PRUSS_CFG Register Summary: [7] [8]

Table 30-504. PRUSS_MII_RT

Address Offset	0x0000 002C		
Physical Address	0x4B22 602C 0x4B2A 602C	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The MII_RT Event Enable Register enables MII_RT mode events to the PRUSS.PRUSS_INTC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															MII_RT_EVENT_EN

Table 30-505. Register Call Summary for Register PRUSS_MII_RT

PRU-ICSS Level Resources Functional Description
- Other PRU-ICSS Module Functional Registers at Subsystem Level: [0]
- PRUSS_CFG Register Summary: [1] [2]

PRU-ICSS Local Interrupt Controller
- PRU-ICSS Interrupt Controller System Events: [3]
- PRU-ICSS Interrupt Requests Mapping: [4] [5] [6] [7]

Table 30-506. PRUSS_IEPCLK

Address Offset	0x0000 0030		
Physical Address	0x4B22 6030 0x4B2A 6030	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The IEP Clock Source Register defines the source of the IEP clock.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															OCP_EN

Bits	Field Name	Description	Type	Reset
31:1	RESERVED		R	0x0
0	OCP_EN	IEP clock source 0x0 = IEP_CLK is the source 0x1 = ICLK is the source. While this is selected no transactions should be active. It can only be cleared by a hardware reset.	RW	0x0

Table 30-507. Register Call Summary for Register PRUSS_IEPCLK

PRU-ICSS Level Resources Functional Description • Module Clock Configurations at PRU-ICSS Top Level: [0] • PRUSS_CFG Register Summary: [1] [2]
PRU-ICSS Industrial Ethernet Peripheral (IEP) • PRU-ICSS IEP Clock Generation: [3] [4]

Table 30-508. PRUSS_SPP

Address Offset	0x0000 0034		
Physical Address	0x4B22 6034 0x4B2A 6034	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The Scratch Pad Priority and Configuration Register defines the access priority assigned to the PRU cores and configures the scratch pad XFR shift functionality.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														XFR_SHIFT_EN	PRU1_PAD_HP_EN

Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0
1	XFR_SHIFT_EN	Enables XIN XOUT shift functionality. When enabled, R0[4:0] (internal to PRU) defines the 32-bit offset for XIN and XOUT operations with the scratch pad. 0x0 = Disabled. 0x1 = Enabled.	RW	0x0
0	PRU1_PAD_HP_EN	Defines which PRU wins write cycle arbitration to a common scratch pad bank. The PRU which has higher priority will always perform the write cycle with no wait states. The lower PRU will get stalled wait states until higher PRU is not performing write cycles. If the lower priority PRU writes to the same byte has the higher priority PRU, then the lower priority PRU will over write the bytes. 0x0 = PRU0 has highest priority. 0x1 = PRU1 has highest priority.	RW	0x0

Table 30-509. Register Call Summary for Register PRUSS_SPP

PRU-ICSS Level Resources Functional Description • Other PRU-ICSS Module Functional Registers at Subsystem Level: [0] • PRUSS_CFG Register Summary: [1] [2]
PRU-ICSS PRU Cores • Optional XIN/XOUT Shift: [3] [4]

Table 30-510. PRUSS_PIN_MX

Address Offset	0x0000 0040		
Physical Address	0x4B22 6040 0x4B2A 6040	**Instance**	PRUSS1_CFG PRUSS2_CFG
Description	The Pin Mux Select Register defines the state of the PRUSS internal pinmuxing.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						PWM3_REMAP_EN	PWM0_REMAP_EN	RESERVED							

Bits	Field Name	Description	Type	Reset
31:10	RESERVED	Reserved. Always write 0.	R	0x0
9	PWM3_REMAP_EN	UNUSED IN THIS DEVICE	RW	0x0
8	PWM0_REMAP_EN	If enabled, host intr6 of PRUSS2 controls epwm_sync_in of PWMSS1 instead of ehrpwm1_synci device pin	RW	0x0
7:0	RESERVED	Reserved	R	0x0

Table 30-511. Register Call Summary for Register PRUSS_PIN_MX

PRU-ICSS Environment
PRU-ICSS Level Resources Functional Description • Other PRU-ICSS Module Functional Registers at Subsystem Level: • PRUSS_CFG Register Summary: [4] [5]
PRU-ICSS PRU Cores • General-Purpose Inputs (R31): Enhanced PRU GP Module:

30.2.5 PRU-ICSS PRU Cores

This section describes the functionality of the two Programmable Real-time Unit (PRU) processors (PRU0 and PRU1) integrated in each of the device PRUSS.

30.2.5.1 PRU Cores Overview

The PRU is a processor optimized for performing embedded tasks that require manipulation of packed memory mapped data structures, handling of system events that have tight real-time constraints and interfacing with systems external to the SoC. The PRU is both very small and very efficient at handling such tasks.

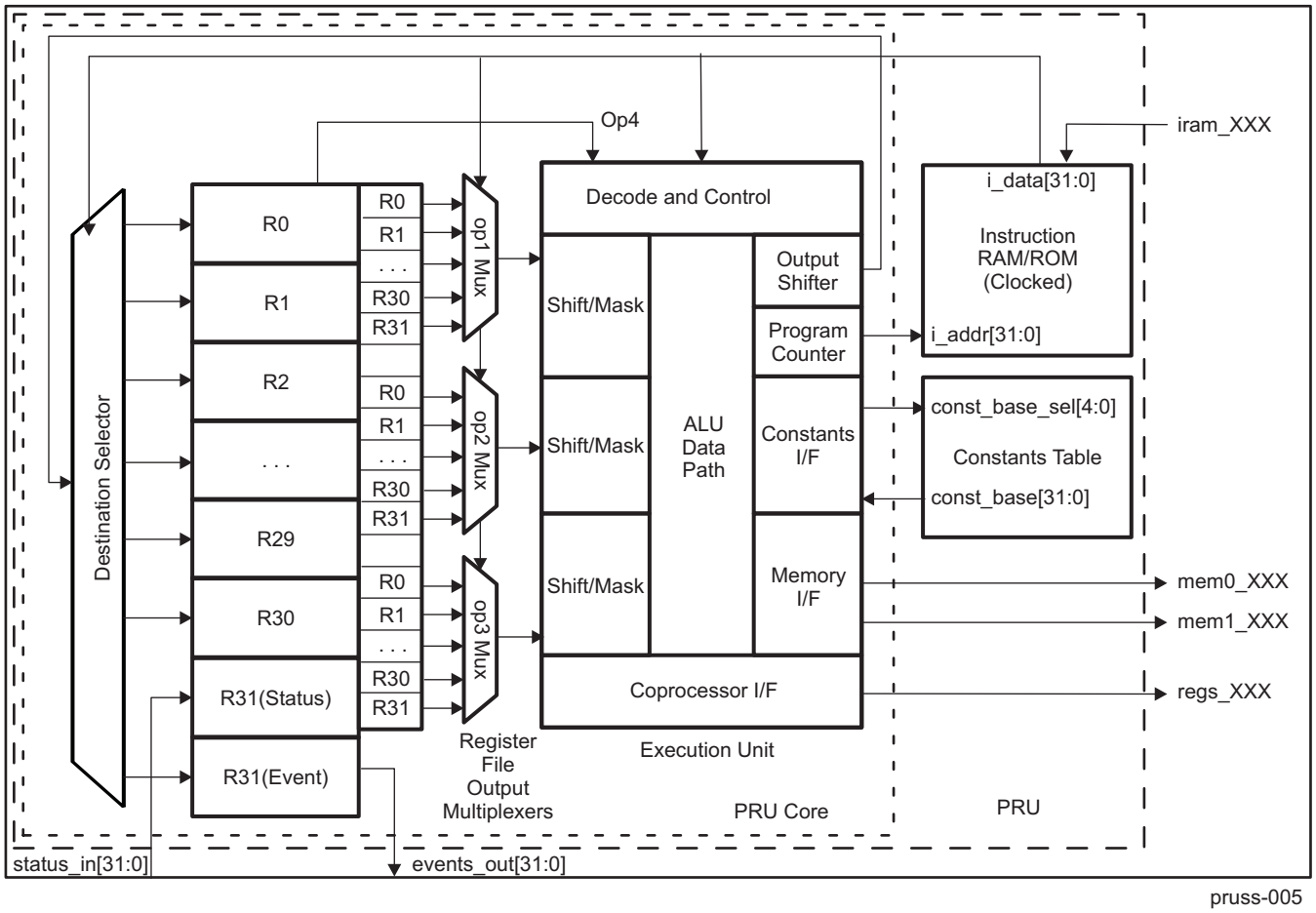
The major attributes of the PRU are in Table 30-512.

Table 30-512. PRU Features

Attribute	Value
IO Architecture	Load/Store
Data Flow Architecture	Register to Register
Core Level Bus Architecture	
Type	4-Bus Harvard (1 Instruction, 3 Data)
Instruction I/F	32-Bit
Memory I/F 0	32-Bit
Memory I/F 1	32-Bit
Execution Model	
Issue Type	Scalar
Pipelining	None (Purposefully)
Ordering	In Order
ALU Type	Unsigned Integer
Registers	
General Purpose (GP)	30 (R1 – R30)
External Status	1 (R31)
GP/Indexing	1 (R0)
Addressability in Instruction	Bit, Byte (8-bit), Half-word (16-bit), Word (32-bit), Pointer
Addressing Modes	
Load Immediate	16-bit Immediate
Load/Store – Memory	Register Base + Register Offset
	Register Base + 8-bit Immediate Offset
	Register Base with auto increment/decrement
	Constant Table Base + Register Offset
	Constant Table Base + 8-bit Immediate Offset
	Constant Table Base with auto increment/decrement
Data Path Width	32-bit
Instruction Width	32-bit
Accessibility to Internal PRU Structures	Provides 32-bit slave with three regions: • Instruction RAM • Control/Status registers • Debug access to internal registers (R0-R31) and constant table

The processor is based on a four-bus architecture which allows instructions to be fetched and executed concurrently with data transfers. In addition, an input is provided in order to allow external status information to be reflected in the internal processor status register. Figure 30-60 shows a block diagram of the processing element and the associated instruction RAM/ROM that contains the code that is to be executed.

Figure 30-60. PRU Block Diagram



pruss-005

30.2.5.2 PRU Cores Functional Description

This section describes the PRU cores supported functionality by describing the constant table, module interface and enhanced GPIOs.

30.2.5.2.1 PRUs Constant Table

The PRU Constants Table is a structure of hard-coded memory addresses for commonly used peripherals and memories. The constants table exists to more efficiently load/store data to these commonly accessed addresses by:

- Reduce a PRU instruction by not needing to pre-load an address into the internal register file before loading/storing data to memory address.
- Maximizing the usage of the PRU register file for embedded processing applications by moving many of the commonly used constant or deterministically calculated base addresses from the internal register file to an external table.

Table 30-513. PRU0/1 Constant Table

Entry No.	Region Pointed To	Value [31:0]
0	PRU-ICSS INTC (local)	0x0002_0000
1	Reserved	0x4804_0000
2	Reserved	0x4802_A000
3	PRU-ICSS eCAP (local)	0x0003_0000
4	PRU-ICSS CFG (local)	0x0002_6000
5	I2C3	0x4806_0000
6	Reserved	0x4803_0000
7	PRU-ICSS UART0 (local)	0x0002_8000
8	MCASP3_DAT	0x4600_0000
9	Reserved	0x4A10_0000
10	Reserved	0x4831_8000
11	Reserved	0x4802_2000
12	Reserved	0x4802_4000
13	Reserved	0x4831_0000
14	Reserved	0x481C_C000
15	Reserved	0x481D_0000
16	Reserved	0x481A_0000
17	Reserved	0x4819_C000
18	Reserved	0x4830_0000
19	Reserved	0x4830_2000
20	Reserved	0x4830_4000
21	PRU-ICSS MDIO (local)	0x0003_2400
22	Reserved	0x480C_8000
23	Reserved	0x480C_A000
24	PRU-ICSS PRU0/1 Data RAM (local)	0x0000_0n00, n = c24_blk_index[3:0]
25	PRU-ICSS PRU1/0 Data RAM (local)	0x0000_2n00, n = c25_blk_index[3:0]
26	PRU-ICSS IEP (local)	0x0002_En00, n = c26_blk_index[3:0]
27	PRU-ICSS MII_RT (local)	0x0003_2n00, n = c27_blk_index[3:0]
28	PRU-ICSS Shared RAM (local)	0x00nn_nn00, nnnn = c28_pointer[15:0]
29	OCMC_RAM2_CBUF	0x49nn_nn00, nnnn = c29_pointer[15:0]
30	OCMC_RAM	0x40nn_nn00, nnnn = c30_pointer[15:0]
31	EMIF1_SDRAM_CS0	0x80nn_nn00, nnnn = c31_pointer[15:0]

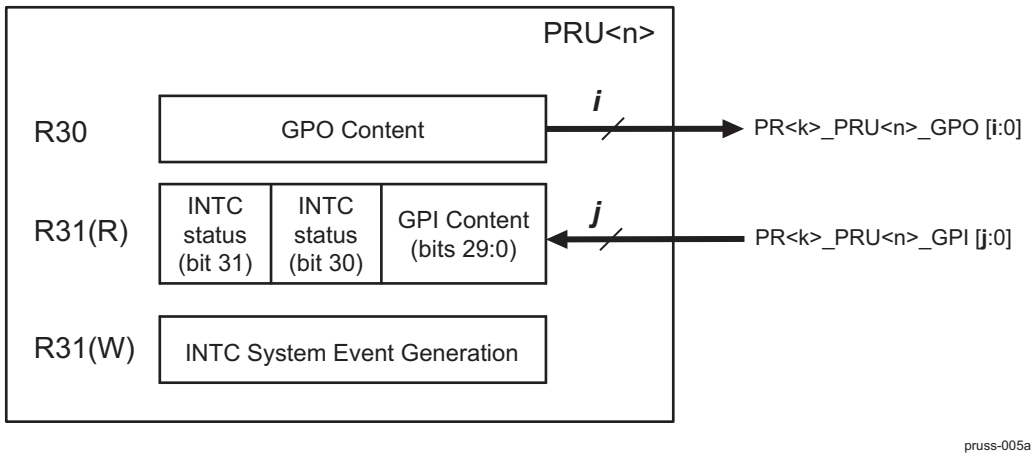
NOTE: The addresses in constants entries 24–31 are partially programmable. Their programmable bit field (for example, c24_blk_index[3:0]) is programmable through the PRU CTRL register space. As a general rule, the PRU should configure this field before using the partially programmable constant entries.

30.2.5.2.2 PRU Module Interface

The PRU module interface consists of the PRU internal registers 30 and 31 (R30 and R31). Figure 30-61 shows the PRU module interface and the functionality of R30 and R31. The register R31 serves as an interface with the dedicated PRU general purpose input (GPI) pins and PRUSS_INTC. Reading R31 returns status information from the GPI pins and PRUSS_INTC via the PRU Real Time Status Interface. Writing to R31generates PRU system events via the PRU Event Interface. The register R30 serves as an interface with the dedicated PRU general purpose output (GPO) pins.

NOTE: The below sections cover different functional modes of the PRUn cores, (where n=0,1), enhanced GPIO (EGPIO) interface. The register bits which control EGPIO functionalities are part of the (PRUSS1_CFG and PRUSS2_CFG) space. For descriptions of these EGPIO register bitfield controls, refer to the Section 30.2.4.3.

Figure 30-61. PRU Module Interface



30.2.5.2.2.1 Real-Time Status Interface Mapping (R31): Interrupt Events Input

The PRU Real Time Status Interface directly feeds information into register 31 (R31) of the PRU's internal register file. The firmware on the PRU uses the status information to make decisions during execution. The status interface is comprised of signals from different modules inside of the PRU-ICSS which require some level of interaction with the PRU. More details on the Host interrupts imported into bit 30 and 31 of register R31 of both the PRUs is provided in the Section 30.2.6, *PRU-ICSS Local Interrupt Controller*.

Table 30-514. Real-Time Status Interface Mapping (R31) Field Descriptions

Bit	Field	Description
31	pru_intr_in[1]	PRU Host Interrupt 1 from local PRUSS_INTC
30	pru_intr_in[0]	PRU Host Interrupt 0 from local PRUSS_INTC
29:0	prun_r31_status[29:0]	Status inputs from primary input via Enhanced GPI port

30.2.5.2.2.2 Event Interface Mapping (R31): PRU System Events

This PRU Event Interface directly feeds pulsed event information out of the PRU's internal ALU. These events are exported out of the PRU-ICSS and need to be connected to the system interrupt controller at the SoC level. The event interface can be used by the firmware to create software interrupts from the PRU to the Host processor.

Figure 30-62. Event Interface Mapping (R31)

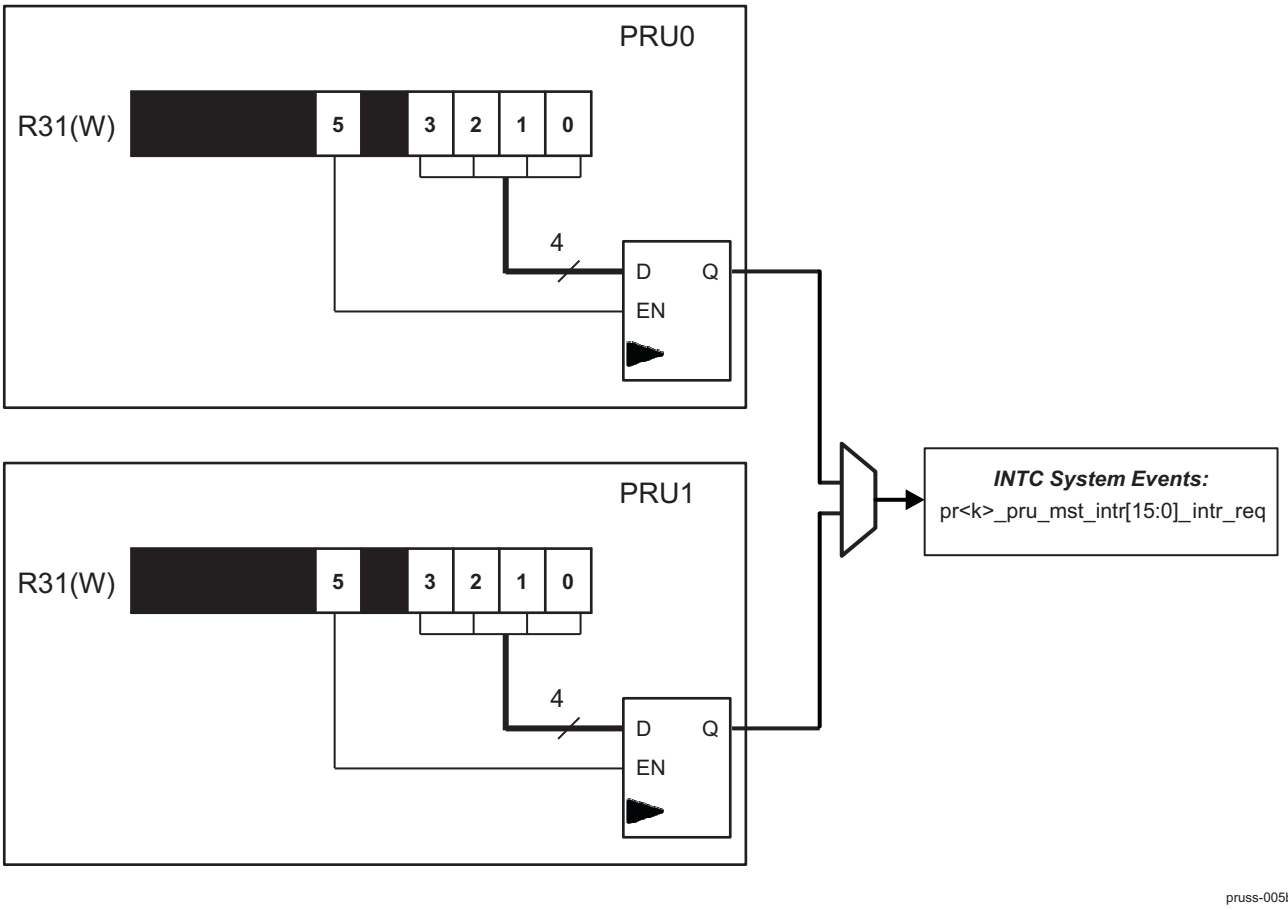


Table 30-515. Event Interface Mapping (R31) Field Descriptions

Bit	Field	Description
31:6	Reserved	
5	prun_r31_vec_valid	Valid strobe for vector output
4	Reserved	
3:0	prun_r31_vec[3:0]	Vector output

Simultaneously writing a '1' to prun_r31_vec_valid (R31 bit 5) and a channel number from 0 to 15 to prun_r31_vec[3:0] (R31 bits 3:0) creates a pulse on the output of the corresponding prk_pru_mst_intr[x]_intr_req INTC system event. For example, writing '100000' will generate a pulse on prk_pru_mst_intr[0]_intr_req, writing '100001' will generate a pulse on prk_pru_mst_intr[1]_intr_req, and so on to where writing '101111' will generate a pulse on prk_pru_mst_intr[15]_intr_req and writing '0xxxxx' will not generate any system event pulses. The output values from both PRU cores in a subsystem are ORed together.

The output channels 0-15 are connected to the PRUSS_INTC system events 16-31, respectively. This allows the PRU to assert one of the system events 16-31 by writing to its own R31 register. The system event is used to either post a completion event to one of the host CPUs (ARM) or to signal the other PRU. The host to be signaled is determined by the system interrupt to interrupt channel mapping (programmable). The 16 events are named as prk_pru_mst_intr<15:0>_intr_req. See the Section 30.2.6.4, *PRU-ICSS Interrupt Requests Mapping* , in the section, *PRU-ICSS Local Interrupt Controller*, for more details.

30.2.5.2.2.3 General-Purpose Inputs (R31): Enhanced PRU GP Module

The PRU-ICSS implements an enhanced General Purpose Input/Output (GPIO) module that supports the following general-purpose input modes: direct input, 16-bit parallel capture, 28-bit serial shift in. Register R31 serves as an interface with the general-purpose inputs. Table 30-516 describes the input modes in detail.

NOTE: Each PRU core can only be configured for one GPI mode at a time. Each mode uses the same R31 signals and internal register bits for different purposes. A summary is found in Table 30-517.

Table 30-516. PRU R31 (GPI) Modes

Mode	Function	Configuration
Direct input	GPI[20:0] feeds directly into the PRU R31	Default state
16-bit parallel capture	DATAIN[0:15] is captured by the posedge or negedge of CLOCKIN	• Enabled by CFG_GPCFGn register • CLOCKIN edge selected by CFG_GPCFGn register
28-bit shift in	DATAIN is sampled and shifted into a 28-bit shift register. Shift Counter (Cnt_16) feature uses … • Shift Counter (Cnt_16) feature is mapped to pru<n>_r31_status[28]. • SB (Start Bit detection) feature is mapped to pru<n>_r31_status[29].	• Enabled by CFG_GPCFGn register • Cnt_16 is self clearing and is connected to the PRU INTC • Start Bit (SB) is cleared by CFG_GPCFGn register

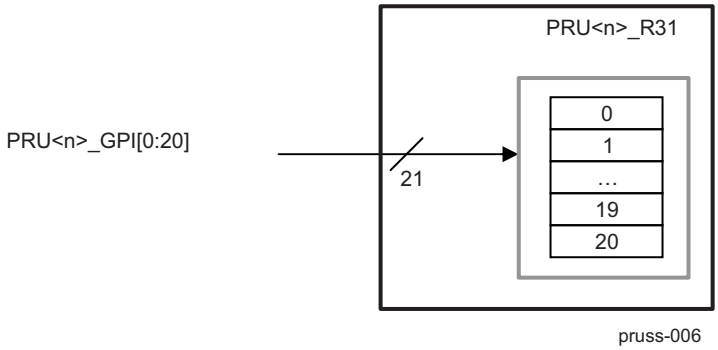
Table 30-517. PRU GPI Signals and Configurations

Pad Names at Device Level	GPI Modes		
	Direct input	Parallel Capture	28-Bit Shift in
pr<k>_pru<n>_gpi0	GPI0	DATAIN0	DATAIN
pr<k>_pru<n>_gpi1	GPI1	DATAIN1	
pr<k>_pru<n>_gpi2	GPI2	DATAIN2	
pr<k>_pru<n>_gpi3	GPI3	DATAIN3	
pr<k>_pru<n>_gpi4	GPI4	DATAIN4	
pr<k>_pru<n>_gpi5	GPI5	DATAIN5	
pr<k>_pru<n>_gpi6	GPI6	DATAIN6	
pr<k>_pru<n>_gpi7	GPI7	DATAIN7	
pr<k>_pru<n>_gpi8	GPI8	DATAIN8	
pr<k>_pru<n>_gpi9	GPI9	DATAIN9	
pr<k>_pru<n>_gpi10	GPI10	DATAIN10	
pr<k>_pru<n>_gpi11	GPI11	DATAIN11	
pr<k>_pru<n>_gpi12	GPI12	DATAIN12	
pr<k>_pru<n>_gpi13	GPI13	DATAIN13	
pr<k>_pru<n>_gpi14	GPI14	DATAIN14	
pr<k>_pru<n>_gpi15	GPI15	DATAIN15	
pr<k>_pru<n>_gpi16	GPI16	CLOCKIN	
pr<k>_pru<n>_gpi17	GPI17		
pr<k>_pru<n>_gpi18	GPI18		
pr<k>_pru<n>_gpi19	GPI19		
pr<k>_pru<n>_gpi20	GPI20		

30.2.5.2.2.3.1 PRU EGPIs Direct Input

The prun_r31_status [0:20] bits of the internal PRU register file are mapped to device-level, general purpose input pins (PRUn_GPI [0:20]). In GPI Direct Input mode, PRUn_GPI [0:20] feeds directly to prun_r31_status [0:20]. Each PRU of the PRU-ICSS has a separate mapping to device input signals - pr1_pru0_gpi[20:0] / pr2_pru0_gpi[20:0] for the PRUSS1/ PRUSS2 PRU0 core and pr1_pru1_gpi[20:0] / pr2_pru1_gpi[20:0] for the PRUSS1 / PRUSS2 PRU1 core so that there are 42 total general purpose inputs to the PRUSS1 / PRUSS2. For more details, refer also to the Section 30.2.2. See the device's system reference guide or datasheet for device specific pin mapping.

Figure 30-63. PRU R31 (EGPI) Direct Input Mode Block Diagram

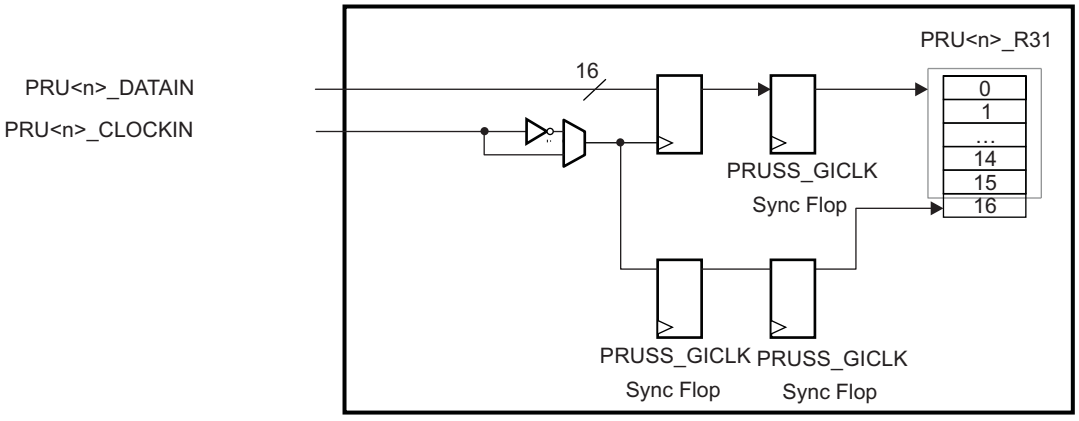


30.2.5.2.2.3.2 PRU EGPIs 16-Bit Parallel Capture

The prun_r31_status [0:15] and prun_r31_status [16] bits of the internal PRU register file mapped to device-level, general purpose input pins (PRUn_DATAIN [0:15] and PRUn_CLOCKIN, respectively). PRUn_CLOCKIN is designated for an external strobe clock, and is used to capture PRUn_DATAIN [0:15].

The PRUn_DATAIN can be captured either by the positive or the negative edge of PRUn_CLOCK, programmable through the PRU-ICSS CFG register space. If the clocking is configured through the PRUICSS CFG register to be positive, then it will equal PRU<n>_CLOCK; however, if the clocking is configured to be negative, then it will equal PRU<n>_CLOCK inverted.

Figure 30-64. PRU R31 (EGPI) 16-Bit Parallel Capture Mode Block Diagram



30.2.5.2.2.3.3 PRU EGPIs 28-Bit Shift In

In 28-bit shift in mode, the device-level, general-purpose input pin PRUn_DATAIN is sampled and shifted into a 28-bit shift register on an internal clock pulse. The register fills in LSB order (from bit 0 to 27) and then overflows into a bit bucket. The 28-bit register is mapped to prun_r31_status [0:27] and can be cleared in software through the PRU-ICSS CFG register space.

Note, the PRU will continually capture and shift the DATAIN input when the GPI mode has been set to 28-bit shift in.

The shift rate is controlled by the effective divisor of two cascaded dividers applied to the 200-MHz clock. These cascaded dividers can each be configured through the PRU-ICSS CFG register space to a value of {1, 1.5, …, 16}. Table 30-518 shows sample effective clock values and the divisor values that can be used to generate these clocks.

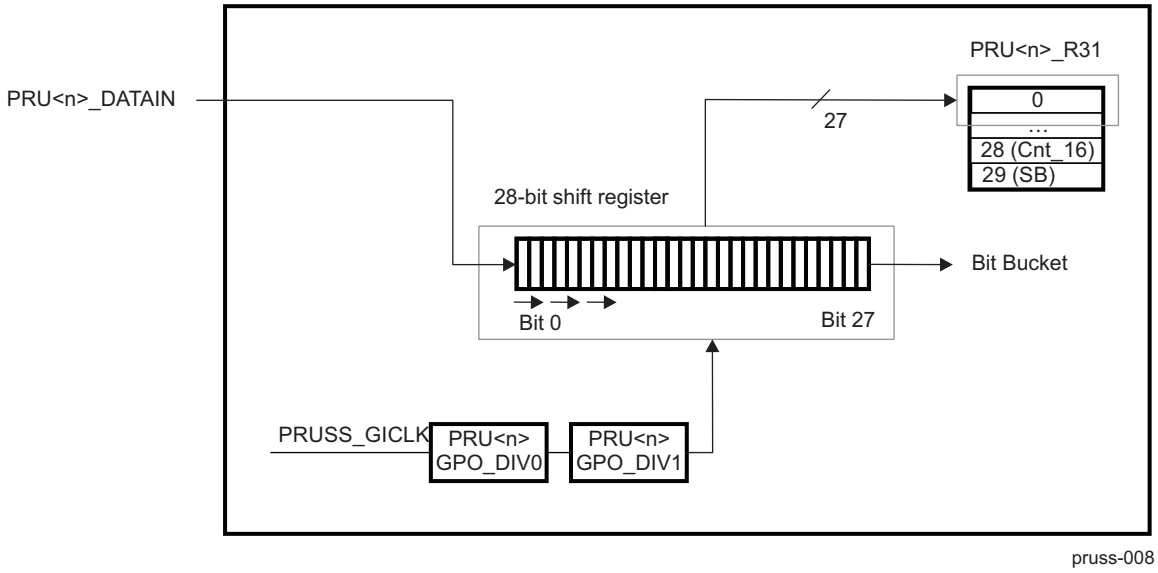
Table 30-518. PRU EGPIs Effective Clock Values

Generated clock	PRUn_GPI_DIV0	PRUn_GPI_DIV1
8-MHz	12.5 (0x17)	2 (0x02)
10-MHz	10 (0x12)	2 (0x02)
16-MHz	16 (0x1e)	1 (0x00)
20-MHz	10 (0x12)	1 (0x00)

The 28-bit shift mode also supports the following features:

- SB (Start Bit detection) is mapped to prun_r31_status[29] and is set when the first 1 is captured on PRUn_DATAIN. The SB flag in pru<n>_r31_status[29] is cleared in software through the PRU-ICSS CFG register space.
- Cnt_16 (Shift Counter) is mapped to pru<n>_r31_status[28] and is set on every 16 shift clock samples after the Start Bit has been received. CNT_16 is self clearing and is connected to the PRUSS_INTC. See the PRU-ICSS Interrupt Controller (PRUSS_INTC) section for more details.

Figure 30-65. PRU R31 (EGPI) 28-Bit Shift Mode



30.2.5.2.2.3.4 General-Purpose Outputs (R30): Enhanced PRU GP Module

The PRU-ICSS implements an enhanced General Purpose Input/Output (GPIO) module that supports two general-purpose output modes: direct output and shift out.

Table 30-519 describes these modes in detail.

NOTE: Each PRU core can only be configured for one GPO mode at a time. Each mode uses the same R30 signals and internal register bits for different purposes. A summary is found in Table 30-519.

Table 30-519. PRU R30 (EGPO) Output Mode

Mode	Function	Configuration
Direct output	pru<n>_r30[20:0] feeds directly to GPO[20:0]	Default state
Shift out	• pru<n>_r30[0] is shifted out on DATAOUT on every rising edge of pru<n>_r30[1] (CLOCKOUT). • LOAD_GPO_SH0 (Load Shadow Register 0) is mapped to pru<n>_r30[29]. • LOAD_GPO_SH1 (Load Shadow Register 1) is mapped to pru<n>_r30[30]. • ENABLE_SHIFT is mapped to pru<n>_r30[31].	Enabled by CFG_GPCFGn register

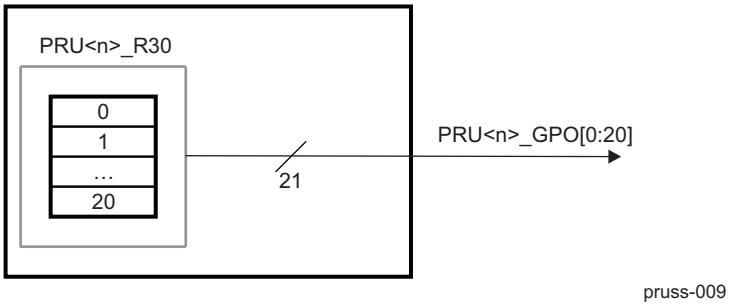
Table 30-520. GPO Mode Descriptions

Pad Names at Device Level	GPO Modes	
	Direct output	Shift out
pr<k>_pru<n>_gpo0	GPO0	DATAOUT
pr<k>_pru<n>_gpo1	GPO1	CLOCKOUT
pr<k>_pru<n>_gpo2	GPO2	
pr<k>_pru<n>_gpo3	GPO3	
pr<k>_pru<n>_gpo4	GPO4	
pr<k>_pru<n>_gpo5	GPO5	
pr<k>_pru<n>_gpo6	GPO6	
pr<k>_pru<n>_gpo7	GPO7	
pr<k>_pru<n>_gpo8	GPO8	
pr<k>_pru<n>_gpo9	GPO9	
pr<k>_pru<n>_gpo10	GPO10	
pr<k>_pru<n>_gpo11	GPO11	
pr<k>_pru<n>_gpo12	GPO12	
pr<k>_pru<n>_gpo13	GPO13	
pr<k>_pru<n>_gpo14	GPO14	
pr<k>_pru<n>_gpo15	GPO15	
pr<k>_pru<n>_gpo16	GPO16	
pr<k>_pru<n>_gpo17	GPO17	
pr<k>_pru<n>_gpo18	GPO18	
pr<k>_pru<n>_gpo19	GPO19	
pr<k>_pru<n>_gpo20	GPO20	

30.2.5.2.2.3.4.1 PRU EGPOs Direct Output

The prun_r30 [20:0] bits of the internal PRU register files are mapped to device-level, general-purpose output pins (PRUn_GPO[0:20]). In GPO Direct Output mode, prun_r30[0:20] feed directly to PRUn_GPO[0:20]. Each PRU of the PRU-ICSS has a separate mapping to pins, so that there are 42 total general-purpose outputs from the PRU-ICSS. See Section 30.2.2, *PRU-ICSS Environment*, and device Data Manual for device-specific pin mapping.

Figure 30-66. PRU R30 (EGPO) Direct Output Mode Block Diagram



NOTE: R30 is not initialized after reset. To avoid unintended output signals, R30 should be initialized before pinmux configuration of PRU signals.

30.2.5.2.2.3.4.2 PRU EGPO Shift Out

In shift out mode, data is shifted out of prun_r30[0] (PRUn_DATAOUT) on every rising edge of prun_r30[1] (PRUn_CLOCK). The shift rate is controlled by the effective divisor of two cascaded dividers applied to the 200-MHz clock. These cascaded dividers can each be configured through the PRU-ICSS CFG register space to a value of {1, 1.5, …, 16}. Table 30-521 shows sample effective clock values and the divisor values that can be used to generate these clocks. Note that PRUn_CLOCKOUT is a free-running clock that starts when the PRU GPO mode is set to shift out mode.

Table 30-521. Effective Clock Values

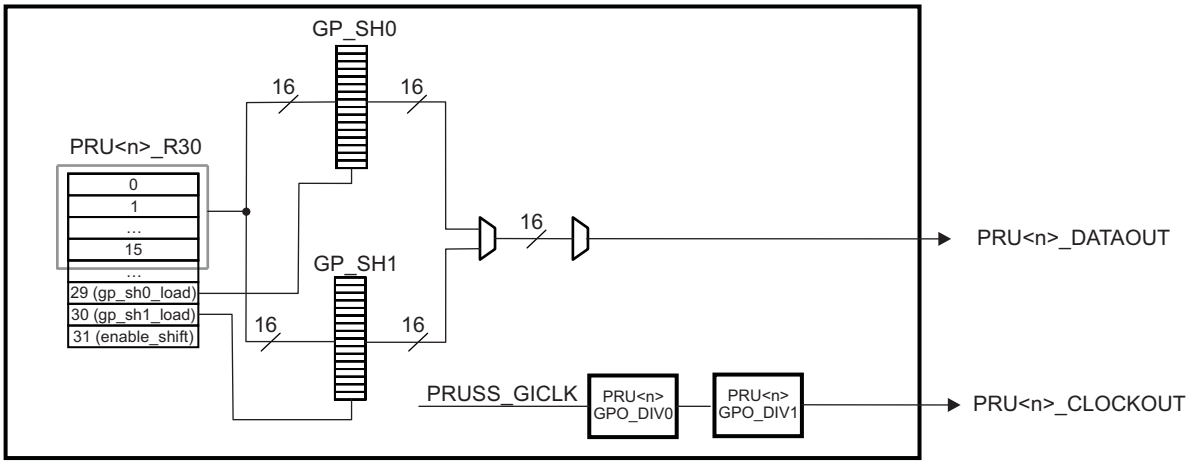
Generated Clock	PRUn_GPO_DIV0	PRUn_GPO_DIV1
8 MHz	12.5 (0x17)	2 (0x02)
10 MHz	10 (0x12)	2 (0x02)
16 MHz	16 (0x1e)	1 (0x00)
20 MHz	10 (0x12)	1 (0x00)

Shift out mode uses two 16-bit shadow registers (gpo_sh0 and gpo_sh1) to support ping-pong buffers. Each shadow register has independent load controls programmable through prun_r30[29:30] (PRUn_LOAD_GPO_SH [0:1]). While PRUn_LOAD_GPO_SH [0/1] is set, the contents of prun_r30[0:15] are loaded into gpo_sh0/1.

NOTE: If any device-level pins mapped to prun_r30[2:15] are configured for the prun_r30 [2:15] pinmux mode, then these pins will reflect the shadow register value written to prun_r30. Any pin configured for a different pinmux setting will not reflect the shadow register value written to prun_r30.

The data shift will start from the LSB of gpo_sh0 when prun_r30[31] (PRUn_ENABLE_SHIFT) is set. Note that if no new data is loaded into gpo_shnn after shift operation, the shift operation will continue looping and shifting out the pre-loaded data. When PRUn_ENABLE_SHIFT is cleared, the shift operation will finish shifting out the current shadow register, stop, and then reset.

Figure 30-67. PRU R30 (GPO) Shift Out Mode Block Diagram



Follow these steps to use the GPO shift out mode:

Step One: Initialization

1. Load 16-bits of data into gpo_sh0:
   (a) Set R30[29] = 1 (PRUn_LOAD_GPO_SH0)
   (b) Load data in R30[15:0]
   (c) Clear R30[29] to turn off load controller
2. Load 16-bits of data into gpo_sh1:
   (a) Set R30[30] = 1 (PRUn_LOAD_GPO_SH1)
   (b) Load data in R30[15:0]
   (c) Clear R30[30] to turn off load controller
3. Start shift operation:
   (a) Set R30[31] = 1 (PRUn_ENABLE_SHIFT)

Step 2: Shift Loop

1. Monitor when a shadow register has finished shifting out data and can be loaded with new data:
   (a) Poll PRUn_GPI_SH_SEL bit of the PRUSS_GPCFG0/1 register
   (b) Load new 16-bits of data into gpo_sh0 if PRUn_GPI_SH_SEL = 1
   (c) Load new 16-bits of data into gpo_sh1 if PRUn_GPI_SH_SEL = 0
2. If more data to be shifted out, loop to Shift Loop
3. If no more data, exit loop

Step 3: Exit

1. End shift operation:
   (a) Clear R30[31] to turn off shift operation

NOTE: Until the shift operation is disabled, the shift loop will continue looping and shifting out the pre-loaded data if no new data has been loaded into gpo_shn.

30.2.5.3 PRU Multiplier with Optional Accumulation (MPY/MAC)

This section describes the MAC (multiplier with optional accumulation) module integrated to PRU0 and PRU1 cores of PRU-ICSS1/PRU-ICSS2.

30.2.5.3.1 PRU MACs Overview

Each of the two PRU cores (PRU0 and PRU1) has a designated unsigned multiplier with optional accumulation (MPY/MAC). The MAC supports two modes of operation: Multiply Only and Multiply and Accumulate.

The MAC is directly connected with the PRU internal registers R25-R29 and uses the broadside load/store PRU interface and XFR instructions to both control and mode of the MAC and import the multiplication results into the PRU.

30.2.5.3.1.1 PRU MAC Key Features

The MPY/MAC instantiated separately to each PRU core (PRU0 and PRU1) features:

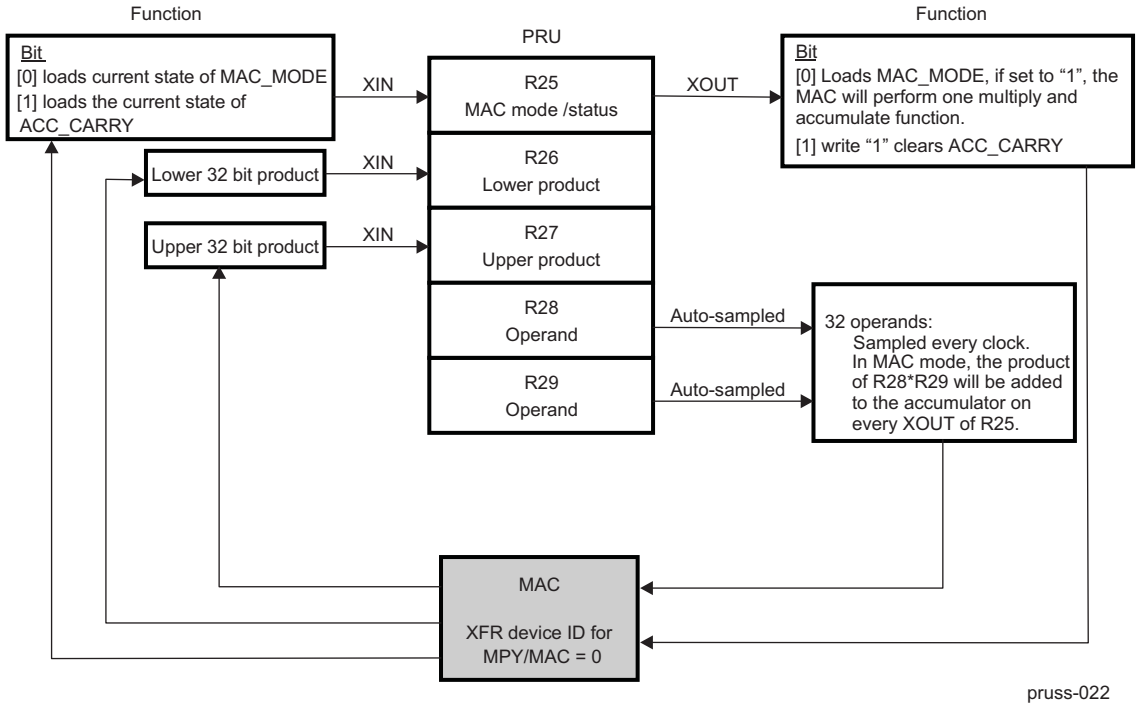
- Configurable Multiply Only and Multiply and Accumulate functionality via PRU register R25
- 32-bit operands with direct connection to PRU registers R28 and R29
- 64-bit result (with carry flag) with direct connection to PRU registers R26 and R27
- PRU broadside interface and XFR instructions (XIN, XOUT) allow for importing multiplication results and initiating accumulate function

30.2.5.3.1.2 PRU MAC Operations

30.2.5.3.1.2.1 PRU versus MAC Interface

The MAC directly connects with the PRU internal registers R25-R29 through use of the PRU broadside interface and XFR instructions. Figure 30-68 shows the functionality of each register.

Figure 30-68. Integration of the PRU and MPY/MAC



The XFR instructions (XIN and XOUT) are used to load/store register contents between the PRU core and the MAC. These instructions define the start, size, direction of the operation, and device ID. The device ID number corresponding to the MPY/MAC is shown in Table 30-522.

Table 30-522. MPY/MAC XFR ID

Device ID	Function
0	Selects MPY/MAC

The PRU register R25 is mapped to the MAC_CTRL_STATUS register (Table 30-523). The MAC's current status (MAC_MODE and ACC_CARRY states) is loaded into R25 using the XIN command on R25. The PRU sets the MAC's mode and clears the ACC_CARRY using the XOUT command on R25.

Table 30-523. MAC_CTRL_STATUS Register (R25) Field Descriptions

Bit	Field	Description
7-2	RESERVED	Reserved
1	ACC_CARRY	Write 1 to clear. 0 - 64-bit accumulator carry has not occurred 1 - 64-bit accumulator carry occurred
0	MAC_MODE	0 - Accumulation mode disabled and accumulator is cleared 1 - Accumulation mode enabled

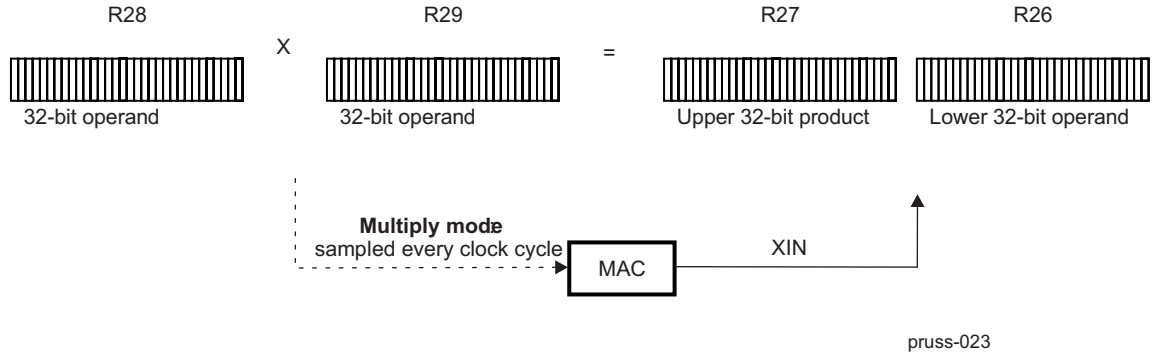
The two 32-bit operands for the multiplication are loaded into R28 and R29. These registers have a direction connection with the MAC. Therefore, XOUT is not required to load the MAC. In multiply mode, the MAC samples these registers every clock cycle. In multiply and accumulate mode, the MAC samples these registers every XOUT R25[7:0] transaction when MAC_MODE = 1.

The product from the MAC is linked to R26 (lower 32 bits) and R27 (upper 32 bits). The product is loaded into register R26 and R27 using XIN.

30.2.5.3.1.2.2 Multiply only mode(default state), MAC_MODE = 0

The Figure 30-69 summarizes the MAC operation in "Multiply-only" mode, in which the MAC multiplies the contents of R28 and R29 on every clock cycle.

Figure 30-69. MAC Multiply-only Mode- Functional Diagram



30.2.5.3.1.2.2.1 Programming PRU MAC in "Multiply-ONLY" mode

The following steps are performed by the PRU firmware for multiply-only mode:

1. Enable multiply only MAC_MODE.
   (a) Clear R25[0] for multiply only mode.
   (b) Store MAC_MODE to MAC using XOUT instruction with the following parameters:
   - Device ID = 0
   - Base register = R25
   - Size = 1
2. Load operands into R28 and R29.
3. Delay at least 1 PRU cycle before executing XIN in step 4.

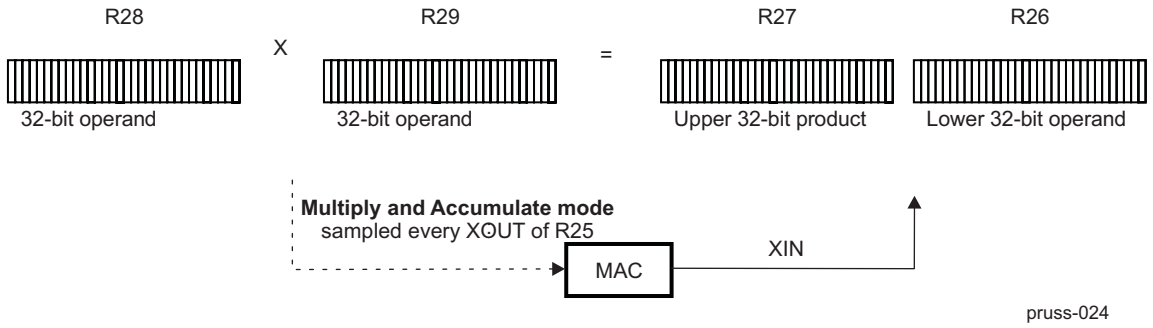
4. Load product into PRU using XIN instruction on R26, R27.

Repeat steps 2-4 for each new operand.

30.2.5.3.1.2.3 Multiply and Accumulate Mode, MAC_MODE = 1

The Figure 30-70 summarizes the MAC operation in "Multiply and Accumulate" mode. On every XOUT R25_REG[7:0] transaction, the MAC multiplies the contents of R28 and R29, adds the product to its accumulated result, and sets ACC_CARRY if an accumulation overflow occurs.

Figure 30-70. MAC Multiply and Accumulate Mode Functional Diagram



30.2.5.3.1.2.3.1 Programming PRU MAC in "Multiply and Accumulate" mode

The following steps are performed by the PRU firmware for multiply and accumulate mode:

1. Enable multiply and accumulate MAC_MODE.
   (a) Set R25[1:0] = 1 for accumulate mode.
   (b) Store MAC_MODE to MAC using XOUT instruction with the following parameters:
   - Device ID = 0
   - Base register = R25
   - Size = 1
2. Clear accumulator and carry flag.
   (a) Set R25[1:0] = 3 to clear accumulator (R25[1]=1) and preserve accumulate mode (R25[0]=1).
   (b) Store accumulator to MAC using XOUT instruction on R25.
3. Load operands into R28 and R29.
4. Multiply and accumulate, XOUT R25[1:0] = 1
   Repeat step 4 for each multiply and accumulate using same operands.
   Repeat step 3 and 4 for each multiply and accumulate for new operands.
5. Load the accumulated product into R26, R27, and the ACC_CARRY status into R25 using the XIN instruction.

NOTE: Steps one and two are required to set the accumulator mode and clear the accumulator and carry flag.

30.2.5.4 PRU0 and PRU1 Scratch Pad Memory

The PRU-ICSS supports a scratch pad with three independent banks accessible by the PRU cores. The PRU cores interact with the scratch pad through broadside load/store PRU interface and XFR instructions. The scratch pad can be used as a temporary place holder for the register contents of the PRU cores. Direct connection between the PRU cores is also supported for transferring register contents directly between the cores. This section describes the Scratch Pad Memory shared between and directly accessible by the PRU0 and PRU1 cores, as well as the XFR direct method used by the PRU cores of the PRU-ICSS1 /PRU-ICSS2.

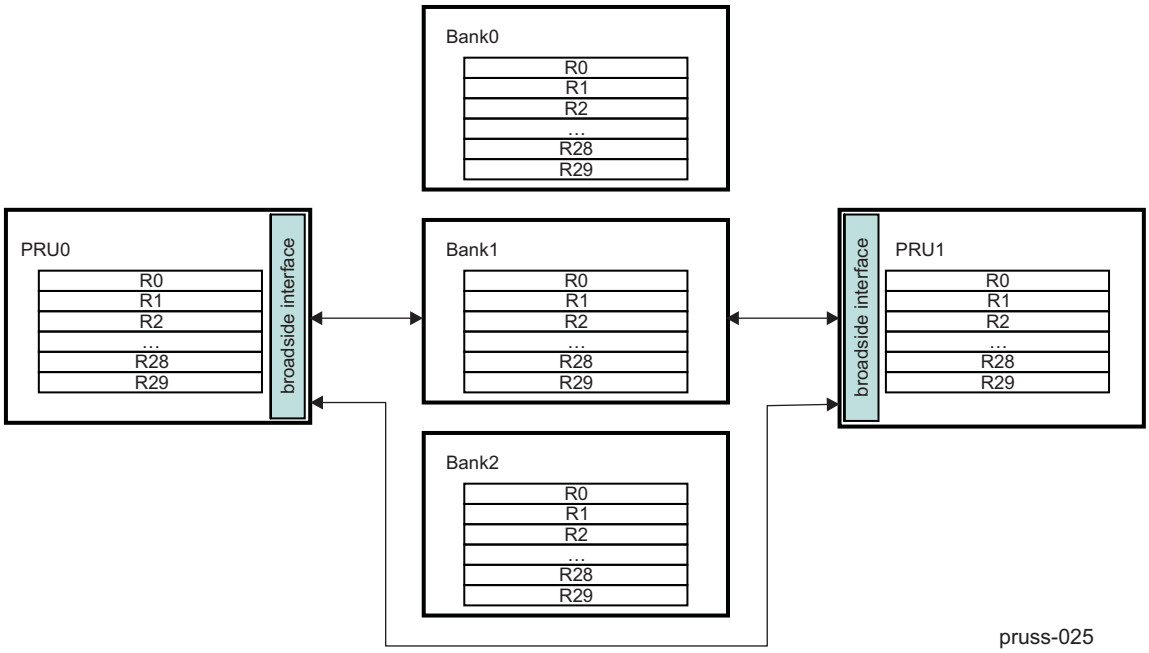
30.2.5.4.1 PRU0/1 Scratch Pad Overview

The PRU-ICSS scratch pad supports the following features:

- Three scratch pad banks of 30, 32-bit registers (R29:0)
- Flexible load/store options
  - User-defined start byte and length of the transfer
  - Length of transfer ranges from one byte of a register to the entire register content (R29 to R0)
  - Simultaneous transactions supported between PRU0 ↔ Bank<n> and PRU1 ↔ Bank<m>
  - Direct connection of PRU0 → PRU1 or PRU1 → PRU0 for all registers R29-R0
- XFR instructions operate in one clock cycle
- Optional XIN/XOUT shift functionality allows remapping of registers (R<n> → R<m>) during load/store operation

Figure 30-71 shows a simplified model of the ScratchPad integration.

Figure 30-71. ScratchPad and PRU Integration



30.2.5.4.2 PRU0 /1 Scratch Pad Operations

XFR instructions are used to load/store register contents between the PRU cores and the scratch pad banks. These instructions define the start, size, direction of the operation, and device ID. The device ID corresponds to the external source or destination (either a scratch pad bank or the other PRU core). The device ID numbers are shown in Table 30-524. Note the direct connect mode (device ID 14) can be used to synchronize the PRU cores. This mode requires the transmitting PRU core to execute XOUT and the receiving PRU core to execute XIN.

Table 30-524. Scratch Pad XFR ID

Device ID	Function
10	Selects Bank0
11	Selects Bank1
12	Selects Bank2
13	Reserved
14	Selects other PRU core (Direct connect mode)

A collision occurs when two XOUT commands simultaneously access the same asset or device ID. Table 30-525 shows the priority assigned to each operation when a collision occurs. In direct connect mode (device ID 14), any PRU transaction will be terminated if the stall is greater than 1024 cycles. This will generate the event pr<k>_xfr_timeout that is connected to INTC.

Table 30-525. Scratch Pad XFR Collision and Stall Conditions

Operation	Collision and Stall Handling
PRU <n> XOUT (→) bank[j]	If both PRU cores access the same bank simultaneously, PRU0 is given priority. PRU1 will temporarily stall until the PRU0 operation completes.
PRU <n> XOUT (→) PRU<m>	Direct connect mode requires the transmitting core (PRU<n>) to execute XOUT and the receiving core (PRU<m>) to execute XIN. If PRU<n> executes XOUT before PRU<m> executes XIN, then PRU<n> will stall until either PRU<m> executes XIN or the stall is greater than 1024 cycles.
PRU<m> XIN (←) PRU<n>	Direct connect mode requires the transmitting core (PRU<n>) to execute XOUT and the receiving core (PRU<m>) to execute XIN. If PRU<m> executes XIN before PRU<n> executes XOUT, then PRU<m> will stall until either PRU<n> executes XOUT or the stall is greater than 1024 cycles.

30.2.5.4.2.1 Optional XIN/XOUT Shift

The optional XIN/XOUT shift functionality allows register contents to be remapped or shifted within the destination's register space. For example, the contents of PRU0 R6-R8 could be remapped to Bank1 R10-12. The XIN/XOUT shift feature is not supported for direct connect mode, only for transfers between a PRU core and scratch pad bank.

The shift feature is enabled or disabled through the PRU subsystem level register PRUSS_SPP[1] XFR_SHIFT_EN bit. When enabled, R0[4:0] (internal to the PRU) defines the number of 32-bit registers in which content is shifted in the scratch pad bank. Note that scratch pad banks do not have registers R30 or R31.

The following PRU firmware examples demonstrate the shift functionality. Note these assume the XFR_SHIFT_EN bit of the PRUSS_SPP register of the PRU-ICSS CFG register space has been set.

XOUT Shift By 4 Registers

Store R4:R7 to R8:R11 in bank0:

- Load 4 into R0.b0
- XOUT using the following parameters:
  - Device ID = 10
  - Base register = R4
  - Size = 16

XOUT Shift By 9 Registers, With Wrap Around

Store R25:R29 to R4:R8 in bank1:

- Load 9 into R0.b0
- XOUT using the following parameters:
  - Device ID = 11
  - Base register = R25
  - Size = 20

XIN Shift By 10 Registers

Load R14:R16 from bank2 to R4:R6:

- Load 10 into R0.b0
- XIN using the following parameters:
  - Device ID = 12
  - Base register = R4

– Size = 12

30.2.5.5 PRUSS_PRU_CTRL Register Manual

This section describes the PRUSS PRU0 and PRU1 cores memory mapped registers.

30.2.5.5.1 PRUSS_PRU_CTRL Instance Summary

Table 30-526. PRUSS_PRU_CTRL Instance Summary

Module Name	Base Address	Size
PRUSS1_PRU0_CTRL	0x4B22 2000	48 Bytes
PRUSS1_PRU1_CTRL	0x4B22 4000	48 Bytes
PRUSS2_PRU0_CTRL	0x4B2A 2000	48 Bytes
PRUSS2_PRU1_CTRL	0x4B2A 4000	48 Bytes

30.2.5.5.2 PRUSS_PRU_CTRL Registers

30.2.5.5.2.1 PRUSS_PRU_CTRL Register Summary

Table 30-527. PRUSS1_PRUn_CTRL Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS1_PRU0_CTRL Physical Address	PRUSS1_PRU1_CTRL Physical Address
PRU_CONTROL	RW	32	0x0000 0000	0x4B22 2000	0x4B22 4000
PRU_STATUS	R	32	0x0000 0004	0x4B22 2004	0x4B22 4004
PRU_WAKEUP_EN	RW	32	0x0000 0008	0x4B22 2008	0x4B22 4008
PRU_CYCLE	RW	32	0x0000 000C	0x4B22 200C	0x4B22 400C
PRU_STALL	RW	32	0x0000 0010	0x4B22 2010	0x4B22 4010
PRU_CTBIR0	RW	32	0x0000 0020	0x4B22 2020	0x4B22 4020
PRU_CTBIR1	RW	32	0x0000 0024	0x4B22 2024	0x4B22 4024
PRU_CTPPR0	RW	32	0x0000 0028	0x4B22 2028	0x4B22 4028
PRU_CTPPR1	RW	32	0x0000 002C	0x4B22 202C	0x4B22 402C

Table 30-528. PRUSS2_PRUn_CTRL Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS2_PRU0_CTRL Physical Address	PRUSS2_PRU1_CTRL Physical Address
PRU_CONTROL	RW	32	0x0000 0000	0x4B2A 2000	0x4B2A 4000
PRU_STATUS	R	32	0x0000 0004	0x4B2A 2004	0x4B2A 4004
PRU_WAKEUP_EN	RW	32	0x0000 0008	0x4B2A 2008	0x4B2A 4008
PRU_CYCLE	RW	32	0x0000 000C	0x4B2A 200C	0x4B2A 400C
PRU_STALL	RW	32	0x0000 0010	0x4B2A 2010	0x4B2A 4010
PRU_CTBIR0	RW	32	0x0000 0020	0x4B2A 2020	0x4B2A 4020
PRU_CTBIR1	RW	32	0x0000 0024	0x4B2A 2024	0x4B2A 4024
PRU_CTPPR0	RW	32	0x0000 0028	0x4B2A 2028	0x4B2A 4028
PRU_CTPPR1	RW	32	0x0000 002C	0x4B2A 202C	0x4B2A 402C

30.2.5.5.2.2 PRUSS_PRU_CTRL Register Description

Table 30-529. PRU_CONTROL

Address Offset	0x0000 0000		
Physical Address	0x4B22 2000 0x4B22 4000 0x4B2A 2000 0x4B2A 4000	**Instance**	PRUSS1_PRU0_CTRL PRUSS1_PRU1_CTRL PRUSS2_PRU0_CTRL PRUSS2_PRU1_CTRL
Description	CONTROL REGISTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCOUNTER_RST_VAL																RUNSTATE	BIG_ENDIAN	RESERVED					SINGLE_STEP	RESERVED				COUNTER_ENABLE	SLEEPING	ENABLE	SOFT_RST_N

Bits	Field Name	Description	Type	Reset
31:16	PCOUNTER_RST_VAL	Program Counter Reset Value: This field controls the address where the PRU will start executing code from after it is taken out of reset.	RW	0x0
15	RUNSTATE	Run State: This bit indicates whether the PRU is currently executing an instruction or is halted. 0 = PRU is halted and host has access to the instruction RAM and debug registers regions. 1 = PRU is currently running and the host is locked out of the instruction RAM and debug registers regions. This bit is used by an external debug agent to know when the PRU has actually halted when waiting for a HALT instruction to execute, a single step to finish, or any other time when the pru_enable has been cleared.	R	0x0
14	BIG_ENDIAN		R	0x0
13:9	RESERVED		R	0x0
8	SINGLE_STEP	Single Step Enable: This bit controls whether or not the PRU will only execute a single instruction when enabled. 0 = PRU will free run when enabled. 1 = PRU will execute a single instruction and then the pru_enable bit will be cleared. Note that this bit does not actually enable the PRU, it only sets the policy for how much code will be run after the PRU is enabled. The pru_enable bit must be explicitly asserted. It is legal to initialize both the single_step and pru_enable bits simultaneously. (Two independent writes are not required to cause the stated functionality.)	RW	0x0
7:4	RESERVED		R	0x0
3	COUNTER_ENABLE	PRU Cycle Counter Enable: Enables PRU cycle counters. 0 = Counters not enabled 1 = Counters enabled	RW	0x0
2	SLEEPING	PRU Sleep Indicator: This bit indicates whether or not the PRU is currently asleep. 0 = PRU is not asleep 1 = PRU is asleep If this bit is written to a 0, the PRU will be forced to power up from sleep mode.	RW	0x0

Bits	Field Name	Description	Type	Reset
1	ENABLE	Processor Enable: This bit controls whether or not the PRU is allowed to fetch new instructions. 0 = PRU is disabled. 1 = PRU is enabled. If this bit is de-asserted while the PRU is currently running and has completed the initial cycle of a multi-cycle instruction (LBxO,SBxO,SCAN, etc.), the current instruction will be allowed to complete before the PRU pauses execution. Otherwise, the PRU will halt immediately. Because of the unpredictability timing sensitivity of the instruction execution loop, this bit is not a reliable indication of whether or not the PRU is currently running. The pru_state bit should be consulted for an absolute indication of the run state of the core. When the PRU is halted, its internal state remains coherent therefore this bit can be reasserted without issuing a software reset and the PRU will resume processing exactly where it left off in the instruction stream.	RW	0x0
0	SOFT_RST_N	Soft Reset: When this bit is cleared, the PRU will be reset. This bit is set back to 1 on the next cycle after it has been cleared.	R	0x1

Table 30-530. Register Call Summary for Register PRU_CONTROL

PRU-ICSS PRU Cores

- PRUSS_PRU_CTRL Register Summary: [0] [1]

Table 30-531. PRU_STATUS

Address Offset	0x0000 0004		
Physical Address	0x4B22 2004 0x4B22 4004 0x4B2A 2004 0x4B2A 4004	**Instance**	PRUSS1_PRU0_CTRL PRUSS1_PRU1_CTRL PRUSS2_PRU0_CTRL PRUSS2_PRU1_CTRL
Description	STATUS REGISTER		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PCOUNTER															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED		R	0x0000
15:0	PCOUNTER	Program Counter: This field is a registered (1 cycle delayed) reflection of the PRU program counter. Note that the PC is an instruction address where each instruction is a 32 bit word. This is not a byte address and to compute the byte address just multiply the PC by 4 (PC of 2 = byte address of 0x8, or PC of 8 = byte address of 0x20).	R	0x0

Table 30-532. Register Call Summary for Register PRU_STATUS

PRU-ICSS PRU Cores

- PRUSS_PRU_CTRL Register Summary: [0] [1]

Table 30-533. PRU_WAKEUP_EN

Address Offset	0x0000 0008		
Physical Address	0x4B22 2008 0x4B22 4008 0x4B2A 2008 0x4B2A 4008	**Instance**	PRUSS1_PRU0_CTRL PRUSS1_PRU1_CTRL PRUSS2_PRU0_CTRL PRUSS2_PRU1_CTRL
Description	WAKEUP ENABLE REGISTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BITWISE_ENABLES																															

Bits	Field Name	Description	Type	Reset
31:0	BITWISE_ENABLES	Wakeup Enables: This field is ANDed with the incoming R31 status inputs (whose bit positions were specified in the stmap parameter) to produce a vector which is unary ORed to produce the status_wakeup source for the core. Setting any bit in this vector will allow the corresponding status input to wake up the core when it is asserted high. The PRU should set this enable vector prior to executing a SLP (sleep) instruction to ensure that the desired sources can wake up the core.	RW	0x0

Table 30-534. Register Call Summary for Register PRU_WAKEUP_EN

PRU-ICSS PRU Cores

- PRUSS_PRU_CTRL Register Summary: [0] [1]

Table 30-535. PRU_CYCLE

Address Offset	0x0000 000C		
Physical Address	0x4B22 200C 0x4B22 400C 0x4B2A 200C 0x4B2A 400C	**Instance**	PRUSS1_PRU0_CTRL PRUSS1_PRU1_CTRL PRUSS2_PRU0_CTRL PRUSS2_PRU1_CTRL
Description	CYCLE COUNT. This register counts the number of cycles for which the PRU has been enabled.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CYCLECOUNT																															

Bits	Field Name	Description	Type	Reset
31:0	CYCLECOUNT	This value is incremented by 1 for every cycle during which the PRU is enabled and the counter is enabled (both bits ENABLE and COUNTENABLE set in the PRU control register). Counting halts while the PRU is disabled or counter is disabled, and resumes when re-eneabled. Counter clears the COUNTENABLE bit in the PRU control register when the count reaches 0xFFFFFFFF. (Count does does not wrap). The register can be read at any time. The register can be cleared when the counter or PRU is disabled. Clearing this register also clears the PRU Stall Count Register.	RW	0x0

Table 30-536. Register Call Summary for Register PRU_CYCLE

PRU-ICSS PRU Cores

- PRUSS_PRU_CTRL Register Summary: [0] [1]

Table 30-537. PRU_STALL

Address Offset	0x0000 0010		
Physical Address	0x4B22 2010 0x4B22 4010 0x4B2A 2010 0x4B2A 4010	**Instance**	PRUSS1_PRU0_CTRL PRUSS1_PRU1_CTRL PRUSS2_PRU0_CTRL PRUSS2_PRU1_CTRL
Description	STALL COUNT. This register counts the number of cycles for which the PRU has been enabled, but unable to fetch a new instruction. It is linked to the Cycle Count Register (0x0C) such that this register reflects the stall cycles measured over the same cycles as counted by the cycle count register. Thus the value of this register is always less than or equal to cycle count.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STALLCOUNT																															

Bits	Field Name	Description	Type	Reset
31:0	STALLCOUNT	This value is incremented by 1 for every cycle during which the PRU is enabled and the counter is enabled (both bits ENABLE and COUNTENABLE set in the PRU control register), and the PRU was unable to fetch a new instruction for any reason.	RW	0x0

Table 30-538. Register Call Summary for Register PRU_STALL

PRU-ICSS PRU Cores

- PRUSS_PRU_CTRL Register Summary: [0] [1]

Table 30-539. PRU_CTBIR0

Address Offset	0x0000 0020		
Physical Address	0x4B22 2020 0x4B22 4020 0x4B2A 2020 0x4B2A 4020	**Instance**	PRUSS1_PRU0_CTRL PRUSS1_PRU1_CTRL PRUSS2_PRU0_CTRL PRUSS2_PRU1_CTRL
Description	CONSTANT TABLE BLOCK INDEX REGISTER 0. This register is used to set the block indices which are used to modify entries 24 and 25 in the PRU Constant Table. This register can be written by the PRU whenever it needs to change to a new base pointer for a block in the State Scratchpad RAM. This function is useful since the PRU is often processing multiple processing threads which require it to change contexts. The PRU can use this register to avoid requiring excessive amounts of code for repetitive context switching.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								C25_BLK_INDEX								RESERVED								C24_BLK_INDEX							

Bits	Field Name	Description	Type	Reset
31:24	RESERVED		R	0x00
23:16	C25_BLK_INDEX	PRU Constant Entry 25 Block Index: This field sets the value that will appear in bits 11:8 of entry 25 in the PRU Constant Table.	RW	0x0
15:8	RESERVED		R	0x00
7:0	C24_BLK_INDEX	PRU Constant Entry 24 Block Index: This field sets the value that will appear in bits 11:8 of entry 24 in the PRU Constant Table.	RW	0x0

Table 30-540. Register Call Summary for Register PRU_CTBIR0

PRU-ICSS PRU Cores

- PRUSS_PRU_CTRL Register Summary: [0] [1]

Table 30-541. PRU_CTBIR1

Address Offset	0x0000 0024		
Physical Address	0x4B22 2024 0x4B22 4024 0x4B2A 2024 0x4B2A 4024	**Instance**	PRUSS1_PRU0_CTRL PRUSS1_PRU1_CTRL PRUSS2_PRU0_CTRL PRUSS2_PRU1_CTRL
Description	CONSTANT TABLE BLOCK INDEX REGISTER 1. This register is used to set the block indices which are used to modify entries 26 and 27 in the PRU Constant Table. This register can be written by the PRU whenever it needs to change to a new base pointer for a block in the State Scratchpad RAM. This function is useful since the PRU is often processing multiple processing threads which require it to change contexts. The PRU can use this register to avoid requiring excessive amounts of code for repetitive context switching.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								C27_BLK_INDEX								RESERVED								C26_BLK_INDEX							

Bits	Field Name	Description	Type	Reset
31:24	RESERVED		R	0x00
23:16	C27_BLK_INDEX	PRU Constant Entry 27 Block Index: This field sets the value that will appear in bits 11:8 of entry 27 in the PRU Constant Table.	RW	0x0
15:8	RESERVED		R	0x00
7:0	C26_BLK_INDEX	PRU Constant Entry 26 Block Index: This field sets the value that will appear in bits 11:8 of entry 26 in the PRU Constant Table.	RW	0x0

Table 30-542. Register Call Summary for Register PRU_CTBIR1

PRU-ICSS PRU Cores

- PRUSS_PRU_CTRL Register Summary: [0] [1]

Table 30-543. PRU_CTPPR0

Address Offset	0x0000 0028		
Physical Address	0x4B22 2028 0x4B22 4028 0x4B2A 2028 0x4B2A 4028	**Instance**	PRUSS1_PRU0_CTRL PRUSS1_PRU1_CTRL PRUSS2_PRU0_CTRL PRUSS2_PRU1_CTRL
Description	CONSTANT TABLE PROGRAMMABLE POINTER REGISTER 0. This register allows the PRU to set up the 256-byte page index for entries 28 and 29 in the PRU Constant Table which serve as general purpose pointers which can be configured to point to any locations inside the session router address map. This register is useful when the PRU needs to frequently access certain structures inside the session router address space whose locations are not hard coded such as tables in scratchpad memory.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C29_POINTER																C28_POINTER															

Bits	Field Name	Description	Type	Reset
31:16	C29_POINTER	PRU Constant Entry 29 Pointer: This field sets the value that will appear in bits 23:8 of entry 29 in the PRU Constant Table.	RW	0x0
15:0	C28_POINTER	PRU Constant Entry 28 Pointer: This field sets the value that will appear in bits 23:8 of entry 28 in the PRU Constant Table.	RW	0x0

Table 30-544. Register Call Summary for Register PRU_CTPPR0

PRU-ICSS PRU Cores
- PRUSS_PRU_CTRL Register Summary: [0] [1]

Table 30-545. PRU_CTPPR1

Address Offset	0x0000 002C		
Physical Address	0x4B22 202C 0x4B22 402C 0x4B2A 202C 0x4B2A 402C	**Instance**	PRUSS1_PRU0_CTRL PRUSS1_PRU1_CTRL PRUSS2_PRU0_CTRL PRUSS2_PRU1_CTRL
Description	CONSTANT TABLE PROGRAMMABLE POINTER REGISTER 1. This register functions the same as the PRU Constant Table Programmable Pointer Register 0 but allows the PRU to control entries 30 and 31 in the PRU Constant Table.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C31_POINTER																C30_POINTER															

Bits	Field Name	Description	Type	Reset
31:16	C31_POINTER	PRU Constant Entry 31 Pointer: This field sets the value that will appear in bits 23:8 of entry 31 in the PRU Constant Table.	RW	0x0
15:0	C30_POINTER	PRU Constant Entry 30 Pointer: This field sets the value that will appear in bits 23:8 of entry 30 in the PRU Constant Table.	RW	0x0

Table 30-546. Register Call Summary for Register PRU_CTPPR1

PRU-ICSS PRU Cores
- PRUSS_PRU_CTRL Register Summary: [0] [1]

30.2.5.6 PRUSS_PRU_DEBUG Register Manual

30.2.5.6.1 PRUSS_PRU_DEBUG Instance Summary

Table 30-547. PRUSS_PRU_DEBUG Instances Summary

Module Name	Base Address	Size
PRUSS1_PRU0_CTRL	0x20AA 2400	144 Bytes
PRUSS1_PRU1_CTRL	0x20AA 4400	144 Bytes
PRUSS2_PRU0_CTRL	0x20AE 2400	144 Bytes
PRUSS2_PRU1_CTRL	0x20AE 4400	144 Bytes

30.2.5.6.2 PRUSS_PRU_DEBUG Registers

30.2.5.6.2.1 PRUSS_PRU_DEBUG Register Summary

Table 30-548. PRUSS1_PRU_DEBUG Registers Mapping Summary

Acronym	Type	Register Width (Bits)	Address Offset	PRUSS1_PRU0_DEBUG Physical Address	PRUSS1_PRU1_DEBUG Physical Address
PRUSS_DBG_GPREG0	RW	32	0x0000 0000	0x20AA 2400	0x20AA 4400
PRUSS_DBG_GPREG1	RW	32	0x0000 0004	0x20AA 2404	0x20AA 4404
PRUSS_DBG_GPREG2	RW	32	0x0000 0008	0x20AA 2408	0x20AA 4408
PRUSS_DBG_GPREG3	RW	32	0x0000 000C	0x20AA 240C	0x20AA 440C
PRUSS_DBG_GPREG4	RW	32	0x0000 0010	0x20AA 2410	0x20AA 4410
PRUSS_DBG_GPREG5	RW	32	0x0000 0014	0x20AA 2414	0x20AA 4414
PRUSS_DBG_GPREG6	RW	32	0x0000 0018	0x20AA 2418	0x20AA 4418
PRUSS_DBG_GPREG7	RW	32	0x0000 001C	0x20AA 241C	0x20AA 441C
PRUSS_DBG_GPREG8	RW	32	0x0000 0020	0x20AA 2420	0x20AA 4420
PRUSS_DBG_GPREG9	RW	32	0x0000 0024	0x20AA 2424	0x20AA 4424
PRUSS_DBG_GPREG10	RW	32	0x0000 0028	0x20AA 2428	0x20AA 4428
PRUSS_DBG_GPREG11	RW	32	0x0000 002C	0x20AA 242C	0x20AA 442C
PRUSS_DBG_GPREG12	RW	32	0x0000 0030	0x20AA 2430	0x20AA 4430
PRUSS_DBG_GPREG13	RW	32	0x0000 0034	0x20AA 2434	0x20AA 4434
PRUSS_DBG_GPREG14	RW	32	0x0000 0038	0x20AA 2438	0x20AA 4438
PRUSS_DBG_GPREG15	RW	32	0x0000 003C	0x20AA 243C	0x20AA 443C
PRUSS_DBG_GPREG16	RW	32	0x0000 0040	0x20AA 2440	0x20AA 4440
PRUSS_DBG_GPREG17	RW	32	0x0000 0044	0x20AA 2444	0x20AA 4444
PRUSS_DBG_GPREG18	RW	32	0x0000 0048	0x20AA 2448	0x20AA 4448
PRUSS_DBG_GPREG19	RW	32	0x0000 004C	0x20AA 244C	0x20AA 444C
PRUSS_DBG_GPREG20	RW	32	0x0000 0050	0x20AA 2450	0x20AA 4450
PRUSS_DBG_GPREG21	RW	32	0x0000 0054	0x20AA 2454	0x20AA 4454
PRUSS_DBG_GPREG22	RW	32	0x0000 0058	0x20AA 2458	0x20AA 4458
PRUSS_DBG_GPREG23	RW	32	0x0000 005C	0x20AA 245C	0x20AA 445C
PRUSS_DBG_GPREG24	RW	32	0x0000 0060	0x20AA 2460	0x20AA 4460
PRUSS_DBG_GPREG25	RW	32	0x0000 0064	0x20AA 2464	0x20AA 4464
PRUSS_DBG_GPREG26	RW	32	0x0000 0068	0x20AA 2468	0x20AA 4468
PRUSS_DBG_GPREG27	RW	32	0x0000 006C	0x20AA 246C	0x20AA 446C
PRUSS_DBG_GPREG28	RW	32	0x0000 0070	0x20AA 2470	0x20AA 4470
PRUSS_DBG_GPREG29	RW	32	0x0000 0074	0x20AA 2474	0x20AA 4474
PRUSS_DBG_GPREG30	RW	32	0x0000 0078	0x20AA 2478	0x20AA 4478
PRUSS_DBG_GPREG31	RW	32	0x0000 007C	0x20AA 247C	0x20AA 447C
PRUSS_DBG_CT_REG0	R	32	0x0000 0080	0x20AA 2480	0x20AA 4480
PRUSS_DBG_CT_REG1	R	32	0x0000 0084	0x20AA 2484	0x20AA 4484
PRUSS_DBG_CT_REG2	R	32	0x0000 0088	0x20AA 2488	0x20AA 4488
PRUSS_DBG_CT_REG3	R	32	0x0000 008C	0x20AA 248C	0x20AA 448C
PRUSS_DBG_CT_REG4	R	32	0x0000 0090	0x20AA 2490	0x20AA 4490
PRUSS_DBG_CT_REG5	R	32	0x0000 0094	0x20AA 2494	0x20AA 4494
PRUSS_DBG_CT_REG6	R	32	0x0000 0098	0x20AA 2498	0x20AA 4498
PRUSS_DBG_CT_REG7	R	32	0x0000 009C	0x20AA 249C	0x20AA 449C
PRUSS_DBG_CT_REG8	R	32	0x0000 00A0	0x20AA 24A0	0x20AA 44A0
PRUSS_DBG_CT_REG9	R	32	0x0000 00A4	0x20AA 24A4	0x20AA 44A4

Table 30-548. PRUSS1_PRU_DEBUG Registers Mapping Summary (continued)

Acronym	Type	Register Width (Bits)	Address Offset	PRUSS1_PRU0_DEBUG Physical Address	PRUSS1_PRU1_DEBUG Physical Address
PRUSS_DBG_CT_REG10	R	32	0x0000 00A8	0x20AA 24A8	0x20AA 44A8
PRUSS_DBG_CT_REG11	R	32	0x0000 00AC	0x20AA 24AC	0x20AA 44AC
PRUSS_DBG_CT_REG12	R	32	0x0000 00B0	0x20AA 24B0	0x20AA 44B0
PRUSS_DBG_CT_REG13	R	32	0x0000 00B4	0x20AA 24B4	0x20AA 44B4
PRUSS_DBG_CT_REG14	R	32	0x0000 00B8	0x20AA 24B8	0x20AA 44B8
PRUSS_DBG_CT_REG15	R	32	0x0000 00BC	0x20AA 24BC	0x20AA 44BC
PRUSS_DBG_CT_REG16	R	32	0x0000 00C0	0x20AA 24C0	0x20AA 44C0
PRUSS_DBG_CT_REG17	R	32	0x0000 00C4	0x20AA 24C4	0x20AA 44C4
PRUSS_DBG_CT_REG18	R	32	0x0000 00C8	0x20AA 24C8	0x20AA 44C8
PRUSS_DBG_CT_REG19	R	32	0x0000 00CC	0x20AA 24CC	0x20AA 44CC
PRUSS_DBG_CT_REG20	R	32	0x0000 00D0	0x20AA 24D0	0x20AA 44D0
PRUSS_DBG_CT_REG21	R	32	0x0000 00D4	0x20AA 24D4	0x20AA 44D4
PRUSS_DBG_CT_REG22	R	32	0x0000 00D8	0x20AA 24D8	0x20AA 44D8
PRUSS_DBG_CT_REG23	R	32	0x0000 00DC	0x20AA 24DC	0x20AA 44DC
PRUSS_DBG_CT_REG24	R	32	0x0000 00E0	0x20AA 24E0	0x20AA 44E0
PRUSS_DBG_CT_REG25	R	32	0x0000 00E4	0x20AA 24E4	0x20AA 44E4
PRUSS_DBG_CT_REG26	R	32	0x0000 00E8	0x20AA 24E8	0x20AA 44E8
PRUSS_DBG_CT_REG27	R	32	0x0000 00EC	0x20AA 24EC	0x20AA 44EC
PRUSS_DBG_CT_REG28	R	32	0x0000 00F0	0x20AA 24F0	0x20AA 44F0
PRUSS_DBG_CT_REG29	R	32	0x0000 00F4	0x20AA 24F4	0x20AA 44F4
PRUSS_DBG_CT_REG30	R	32	0x0000 00F8	0x20AA 24F8	0x20AA 44F8
PRUSS_DBG_CT_REG31	R	32	0x0000 00FC	0x20AA 24FC	0x20AA 44FC

Table 30-549. PRUSS2_PRU_DEBUG Registers Mapping Summary

Acronym	Type	Register Width (Bits)	Address Offset	PRUSS2_PRU0_DEBUG Physical Address	PRUSS2_PRU1_DEBUG Physical Address
PRUSS_DBG_GPREG0	RW	32	0x0000 0000	0x20AE 2400	0x20AE 4400
PRUSS_DBG_GPREG1	RW	32	0x0000 0004	0x20AE 2404	0x20AE 4404
PRUSS_DBG_GPREG2	RW	32	0x0000 0008	0x20AE 2408	0x20AE 4408
PRUSS_DBG_GPREG3	RW	32	0x0000 000C	0x20AE 240C	0x20AE 440C
PRUSS_DBG_GPREG4	RW	32	0x0000 0010	0x20AE 2410	0x20AE 4410
PRUSS_DBG_GPREG5	RW	32	0x0000 0014	0x20AE 2414	0x20AE 4414
PRUSS_DBG_GPREG6	RW	32	0x0000 0018	0x20AE 2418	0x20AE 4418
PRUSS_DBG_GPREG7	RW	32	0x0000 001C	0x20AE 241C	0x20AE 441C
PRUSS_DBG_GPREG8	RW	32	0x0000 0020	0x20AE 2420	0x20AE 4420
PRUSS_DBG_GPREG9	RW	32	0x0000 0024	0x20AE 2424	0x20AE 4424
PRUSS_DBG_GPREG10	RW	32	0x0000 0028	0x20AE 2428	0x20AE 4428
PRUSS_DBG_GPREG11	RW	32	0x0000 002C	0x20AE 242C	0x20AE 442C
PRUSS_DBG_GPREG12	RW	32	0x0000 0030	0x20AE 2430	0x20AE 4430
PRUSS_DBG_GPREG13	RW	32	0x0000 0034	0x20AE 2434	0x20AE 4434
PRUSS_DBG_GPREG14	RW	32	0x0000 0038	0x20AE 2438	0x20AE 4438
PRUSS_DBG_GPREG15	RW	32	0x0000 003C	0x20AE 243C	0x20AE 443C
PRUSS_DBG_GPREG16	RW	32	0x0000 0040	0x20AE 2440	0x20AE 4440
PRUSS_DBG_GPREG17	RW	32	0x0000 0044	0x20AE 2444	0x20AE 4444
PRUSS_DBG_GPREG18	RW	32	0x0000 0048	0x20AE 2448	0x20AE 4448

Table 30-549. PRUSS2_PRU_DEBUG Registers Mapping Summary (continued)

Acronym	Type	Register Width (Bits)	Address Offset	PRUSS2_PRU0_DEBUG Physical Address	PRUSS2_PRU1_DEBUG Physical Address
PRUSS_DBG_GPREG19	RW	32	0x0000 004C	0x20AE 244C	0x20AE 444C
PRUSS_DBG_GPREG20	RW	32	0x0000 0050	0x20AE 2450	0x20AE 4450
PRUSS_DBG_GPREG21	RW	32	0x0000 0054	0x20AE 2454	0x20AE 4454
PRUSS_DBG_GPREG22	RW	32	0x0000 0058	0x20AE 2458	0x20AE 4458
PRUSS_DBG_GPREG23	RW	32	0x0000 005C	0x20AE 245C	0x20AE 445C
PRUSS_DBG_GPREG24	RW	32	0x0000 0060	0x20AE 2460	0x20AE 4460
PRUSS_DBG_GPREG25	RW	32	0x0000 0064	0x20AE 2464	0x20AE 4464
PRUSS_DBG_GPREG26	RW	32	0x0000 0068	0x20AE 2468	0x20AE 4468
PRUSS_DBG_GPREG27	RW	32	0x0000 006C	0x20AE 246C	0x20AE 446C
PRUSS_DBG_GPREG28	RW	32	0x0000 0070	0x20AE 2470	0x20AE 4470
PRUSS_DBG_GPREG29	RW	32	0x0000 0074	0x20AE 2474	0x20AE 4474
PRUSS_DBG_GPREG30	RW	32	0x0000 0078	0x20AE 2478	0x20AE 4478
PRUSS_DBG_GPREG31	RW	32	0x0000 007C	0x20AE 247C	0x20AE 447C
PRUSS_DBG_CT_REG0	R	32	0x0000 0080	0x20AE 2480	0x20AE 4480
PRUSS_DBG_CT_REG1	R	32	0x0000 0084	0x20AE 2484	0x20AE 4484
PRUSS_DBG_CT_REG2	R	32	0x0000 0088	0x20AE 2488	0x20AE 4488
PRUSS_DBG_CT_REG3	R	32	0x0000 008C	0x20AE 248C	0x20AE 448C
PRUSS_DBG_CT_REG4	R	32	0x0000 0090	0x20AE 2490	0x20AE 4490
PRUSS_DBG_CT_REG5	R	32	0x0000 0094	0x20AE 2494	0x20AE 4494
PRUSS_DBG_CT_REG6	R	32	0x0000 0098	0x20AE 2498	0x20AE 4498
PRUSS_DBG_CT_REG7	R	32	0x0000 009C	0x20AE 249C	0x20AE 449C
PRUSS_DBG_CT_REG8	R	32	0x0000 00A0	0x20AE 24A0	0x20AE 44A0
PRUSS_DBG_CT_REG9	R	32	0x0000 00A4	0x20AE 24A4	0x20AE 44A4
PRUSS_DBG_CT_REG10	R	32	0x0000 00A8	0x20AE 24A8	0x20AE 44A8
PRUSS_DBG_CT_REG11	R	32	0x0000 00AC	0x20AE 24AC	0x20AE 44AC
PRUSS_DBG_CT_REG12	R	32	0x0000 00B0	0x20AE 24B0	0x20AE 44B0
PRUSS_DBG_CT_REG13	R	32	0x0000 00B4	0x20AE 24B4	0x20AE 44B4
PRUSS_DBG_CT_REG14	R	32	0x0000 00B8	0x20AE 24B8	0x20AE 44B8
PRUSS_DBG_CT_REG15	R	32	0x0000 00BC	0x20AE 24BC	0x20AE 44BC
PRUSS_DBG_CT_REG16	R	32	0x0000 00C0	0x20AE 24C0	0x20AE 44C0
PRUSS_DBG_CT_REG17	R	32	0x0000 00C4	0x20AE 24C4	0x20AE 44C4
PRUSS_DBG_CT_REG18	R	32	0x0000 00C8	0x20AE 24C8	0x20AE 44C8
PRUSS_DBG_CT_REG19	R	32	0x0000 00CC	0x20AE 24CC	0x20AE 44CC
PRUSS_DBG_CT_REG20	R	32	0x0000 00D0	0x20AE 24D0	0x20AE 44D0
PRUSS_DBG_CT_REG21	R	32	0x0000 00D4	0x20AE 24D4	0x20AE 44D4
PRUSS_DBG_CT_REG22	R	32	0x0000 00D8	0x20AE 24D8	0x20AE 44D8
PRUSS_DBG_CT_REG23	R	32	0x0000 00DC	0x20AE 24DC	0x20AE 44DC
PRUSS_DBG_CT_REG24	R	32	0x0000 00E0	0x20AE 24E0	0x20AE 44E0
PRUSS_DBG_CT_REG25	R	32	0x0000 00E4	0x20AE 24E4	0x20AE 44E4
PRUSS_DBG_CT_REG26	R	32	0x0000 00E8	0x20AE 24E8	0x20AE 44E8
PRUSS_DBG_CT_REG27	R	32	0x0000 00EC	0x20AE 24EC	0x20AE 44EC
PRUSS_DBG_CT_REG28	R	32	0x0000 00F0	0x20AE 24F0	0x20AE 44F0
PRUSS_DBG_CT_REG29	R	32	0x0000 00F4	0x20AE 24F4	0x20AE 44F4
PRUSS_DBG_CT_REG30	R	32	0x0000 00F8	0x20AE 24F8	0x20AE 44F8
PRUSS_DBG_CT_REG31	R	32	0x0000 00FC	0x20AE 24FC	0x20AE 44FC

30.2.5.6.2.2 PRUSS_PRU_DEBUG Register Description

Table 30-550. PRUSS_DBG_GPREG0

Address Offset	0x0000 0000		
Physical Address	0x20AA 2400 0x20AA 4400 0x20AE 2400 0x20AE 2400	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 0. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG0																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG0	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-551. Register Call Summary for Register PRUSS_DBG_GPREG0

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-552. PRUSS_DBG_GPREG1

Address Offset	0x0000 0004		
Physical Address	0x20AA 2404 0x20AA 4404 0x20AE 2404 0x20AE 4404	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 1. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG1																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG1	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-553. Register Call Summary for Register PRUSS_DBG_GPREG1

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-554. PRUSS_DBG_GPREG2

Address Offset	0x0000 0008		
Physical Address	0x20AA 2408 0x20AA 4408 0x20AE 2408 0x20AE 4408	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 2. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG2																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG2	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-555. Register Call Summary for Register PRUSS_DBG_GPREG2

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-556. PRUSS_DBG_GPREG3

Address Offset	0x0000 000C		
Physical Address	0x20AA 240C 0x20AA 440C 0x20AE 240C 0x20AE 440C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 3. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG3																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG3	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-557. Register Call Summary for Register PRUSS_DBG_GPREG3

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-558. PRUSS_DBG_GPREG4

Address Offset	0x0000 0010		
Physical Address	0x20AA 2410 0x20AA 4410 0x20AE 2410 0x20AE 4410	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 4. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG4																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG4	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-559. Register Call Summary for Register PRUSS_DBG_GPREG4

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-560. PRUSS_DBG_GPREG5

Address Offset	0x0000 0014		
Physical Address	0x20AA 2414 0x20AA 4414 0x20AE 2414 0x20AE 4414	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 5. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG5																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG5	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-561. Register Call Summary for Register PRUSS_DBG_GPREG5

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-562. PRUSS_DBG_GPREG6

Address Offset	0x0000 0018		
Physical Address	0x20AA 2418 0x20AA 4418 0x20AE 2418 0x20AE 4418	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 6. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG6																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG6	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-563. Register Call Summary for Register PRUSS_DBG_GPREG6

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-564. PRUSS_DBG_GPREG7

Address Offset	0x0000 001C		
Physical Address	0x20AA 241C 0x20AA 441C 0x20AE 241C 0x20AE 441C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 7. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG7																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG7	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-565. Register Call Summary for Register PRUSS_DBG_GPREG7

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-566. PRUSS_DBG_GPREG8

Address Offset	0x0000 0020		
Physical Address	0x20AA 2420 0x20AA 4420 0x20AE 2420 0x20AE 4420	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 8. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG8																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG8	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-567. Register Call Summary for Register PRUSS_DBG_GPREG8

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-568. PRUSS_DBG_GPREG9

Address Offset	0x0000 0024		
Physical Address	0x20AA 2424 0x20AA 4424 0x20AE 2424 0x20AE 4424	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 9. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG9																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG9	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-569. Register Call Summary for Register PRUSS_DBG_GPREG9

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-570. PRUSS_DBG_GPREG10

Address Offset	0x0000 0028		
Physical Address	0x20AA 2428 0x20AA 4428 0x20AE 2428 0x20AE 4428	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 10. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG10																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG10	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-571. Register Call Summary for Register PRUSS_DBG_GPREG10

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-572. PRUSS_DBG_GPREG11

Address Offset	0x0000 002C		
Physical Address	0x20AA 242C 0x20AA 442C 0x20AE 242C 0x20AE 442C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 11. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG11																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG11	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-573. Register Call Summary for Register PRUSS_DBG_GPREG11

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-574. PRUSS_DBG_GPREG12

Address Offset	0x0000 0030		
Physical Address	0x20AA 2430 0x20AA 4430 0x20AE 2430 0x20AE 4430	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 12. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG12																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG12	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-575. Register Call Summary for Register PRUSS_DBG_GPREG12

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-576. PRUSS_DBG_GPREG13

Address Offset	0x0000 0034		
Physical Address	0x20AA 2434 0x20AA 4434 0x20AE 2434 0x20AE 4434	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 13. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG13																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG13	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-577. Register Call Summary for Register PRUSS_DBG_GPREG13

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-578. PRUSS_DBG_GPREG14

Address Offset	0x0000 0038		
Physical Address	0x20AA 2438 0x20AA 4438 0x20AE 2438 0x20AE 4438	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 14. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG14																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG14	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-579. Register Call Summary for Register PRUSS_DBG_GPREG14

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-580. PRUSS_DBG_GPREG15

Address Offset	0x0000 003C		
Physical Address	0x20AA 243C 0x20AA 443C 0x20AE 243C 0x20AE 443C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 15. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG15																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG15	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-581. Register Call Summary for Register PRUSS_DBG_GPREG15

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-582. PRUSS_DBG_GPREG16

Address Offset	0x0000 0040		
Physical Address	0x20AA 2440 0x20AA 4440 0x20AE 2440 0x20AE 4440	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 16. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG16																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG16	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-583. Register Call Summary for Register PRUSS_DBG_GPREG16

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-584. PRUSS_DBG_GPREG17

Address Offset	0x0000 0044		
Physical Address	0x20AA 2444 0x20AA 4444 0x20AE 2444 0x20AE 4444	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 17. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG17																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG17	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-585. Register Call Summary for Register PRUSS_DBG_GPREG17

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-586. PRUSS_DBG_GPREG18

Address Offset	0x0000 0048		
Physical Address	0x20AA 2448 0x20AA 4448 0x20AE 2448 0x20AE 4448	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 18. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG18																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG18	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-587. Register Call Summary for Register PRUSS_DBG_GPREG18

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-588. PRUSS_DBG_GPREG19

Address Offset	0x0000 004C		
Physical Address	0x20AA 244C 0x20AA 444C 0x20AE 244C 0x20AE 444C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 19. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG19																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG19	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-589. Register Call Summary for Register PRUSS_DBG_GPREG19

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-590. PRUSS_DBG_GPREG20

Address Offset	0x0000 0050		
Physical Address	0x20AA 2450 0x20AA 4450 0x20AE 2450 0x20AE 4450	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 20. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG20																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG20	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-591. Register Call Summary for Register PRUSS_DBG_GPREG20

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-592. PRUSS_DBG_GPREG21

Address Offset	0x0000 0054		
Physical Address	0x20AA 2454 0x20AA 4454 0x20AE 2454 0x20AE 4454	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 21. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG21																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG21	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-593. Register Call Summary for Register PRUSS_DBG_GPREG21

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-594. PRUSS_DBG_GPREG22

Address Offset	0x0000 0058		
Physical Address	0x20AA 2458 0x20AA 4458 0x20AE 2458 0x20AE 4458	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 22. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG22																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG22	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-595. Register Call Summary for Register PRUSS_DBG_GPREG22

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-596. PRUSS_DBG_GPREG23

Address Offset	0x0000 005C		
Physical Address	0x20AA 245C 0x20AA 445C 0x20AE 245C 0x20AE 445C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 23. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG23																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG23	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-597. Register Call Summary for Register PRUSS_DBG_GPREG23

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-598. PRUSS_DBG_GPREG24

Address Offset	0x0000 0060		
Physical Address	0x20AA 2460 0x20AA 4460 0x20AE 2460 0x20AE 4460	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 24. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG24																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG24	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-599. Register Call Summary for Register PRUSS_DBG_GPREG24

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-600. PRUSS_DBG_GPREG25

Address Offset	0x0000 0064		
Physical Address	0x20AA 2464 0x20AA 4464 0x20AE 2464 0x20AE 4464	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 25. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG25																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG25	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-601. Register Call Summary for Register PRUSS_DBG_GPREG25

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-602. PRUSS_DBG_GPREG26

Address Offset	0x0000 0068		
Physical Address	0x20AA 2468 0x20AA 4468 0x20AE 2468 0x20AE 4468	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 26. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG26																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG26	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-603. Register Call Summary for Register PRUSS_DBG_GPREG26

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-604. PRUSS_DBG_GPREG27

Address Offset	0x0000 006C		
Physical Address	0x20AA 246C 0x20AA 446C 0x20AE 246C 0x20AE 446C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 27. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG27																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG27	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-605. Register Call Summary for Register PRUSS_DBG_GPREG27

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-606. PRUSS_DBG_GPREG28

Address Offset	0x0000 0070		
Physical Address	0x20AA 2470 0x20AA 4470 0x20AE 2470 0x20AE 4470	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 28. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG28																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG28	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-607. Register Call Summary for Register PRUSS_DBG_GPREG28

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-608. PRUSS_DBG_GPREG29

Address Offset	0x0000 0074		
Physical Address	0x20AA 2474 0x20AA 4474 0x20AE 2474 0x20AE 4474	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 29. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG29																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG29	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-609. Register Call Summary for Register PRUSS_DBG_GPREG29

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-610. PRUSS_DBG_GPREG30

Address Offset	0x0000 0078		
Physical Address	0x20AA 2478 0x20AA 4478 0x20AE 2478 0x20AE 4478	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 30. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG30																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG30	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-611. Register Call Summary for Register PRUSS_DBG_GPREG30

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-612. PRUSS_DBG_GPREG31

Address Offset	0x0000 007C		
Physical Address	0x20AA 247C 0x20AA 447C 0x20AE 247C 0x20AE 447C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU GENERAL PURPOSE REGISTER 31. This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP_REG31																															

Bits	Field Name	Description	Type	Reset
31:0	GP_REG31	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile	RW	0x0

Table 30-613. Register Call Summary for Register PRUSS_DBG_GPREG31

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-614. PRUSS_DBG_CT_REG0

Address Offset	0x0000 0080		
Physical Address	0x20AA 2480 0x20AA 4480 0x20AE 2480 0x20AE 4480	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 0. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG0																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG0	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x02 0000

Table 30-615. Register Call Summary for Register PRUSS_DBG_CT_REG0

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-616. PRUSS_DBG_CT_REG1

Address Offset	0x0000 0084		
Physical Address	0x20AA 2484 0x20AA 4484 0x20AE 2484 0x20AE 4484	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 1. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG1																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG1	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4804 0000

Table 30-617. Register Call Summary for Register PRUSS_DBG_CT_REG1

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-618. PRUSS_DBG_CT_REG2

Address Offset	0x0000 0088		
Physical Address	0x20AA 2488 0x20AA 4488 0x20AE 2488 0x20AE 4488	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 2. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG2																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG2	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4802 A000

Table 30-619. Register Call Summary for Register PRUSS_DBG_CT_REG2

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-620. PRUSS_DBG_CT_REG3

Address Offset	0x0000 008C		
Physical Address	0x20AA 248C 0x20AA 448C 0x20AE 248C 0x20AE 448C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 3. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG3																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG3	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x03 0000

Table 30-621. Register Call Summary for Register PRUSS_DBG_CT_REG3

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-622. PRUSS_DBG_CT_REG4

Address Offset	0x0000 0090		
Physical Address	0x20AA 2490 0x20AA 4490 0x20AE 2490 0x20AE 4490	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 4. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG4																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG4	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x02 6000

Table 30-623. Register Call Summary for Register PRUSS_DBG_CT_REG4

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-624. PRUSS_DBG_CT_REG5

Address Offset	0x0000 0094		
Physical Address	0x20AA 2494 0x20AA 4494 0x20AE 2494 0x20AE 4494	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 5. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG5																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG5	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4806 0000

Table 30-625. Register Call Summary for Register PRUSS_DBG_CT_REG5

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-626. PRUSS_DBG_CT_REG6

Address Offset	0x0000 0098		
Physical Address	0x20AA 2498 0x20AA 4498 0x20AE 2498 0x20AE 4498	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 6. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG6																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG6	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4803 0000

Table 30-627. Register Call Summary for Register PRUSS_DBG_CT_REG6

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-628. PRUSS_DBG_CT_REG7

Address Offset	0x0000 009C		
Physical Address	0x20AA 249C 0x20AA 449C 0x20AE 249C 0x20AE 449C	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 7. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG7																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG7	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x02 8000

Table 30-629. Register Call Summary for Register PRUSS_DBG_CT_REG7

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-630. PRUSS_DBG_CT_REG8

Address Offset	0x0000 00A0		
Physical Address	0x20AA 24A0 0x20AA 44A0 0x20AE 24A0 0x20AE 44A0	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 8. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG8																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG8	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4600 0000

Table 30-631. Register Call Summary for Register PRUSS_DBG_CT_REG8

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-632. PRUSS_DBG_CT_REG9

Address Offset	0x0000 00A4		
Physical Address	0x20AA 24A4 0x20AA 44A4 0x20AE 24A4 0x20AE 44A4	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 9. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG9																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG9	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4A10 0000

Table 30-633. Register Call Summary for Register PRUSS_DBG_CT_REG9

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-634. PRUSS_DBG_CT_REG10

Address Offset	0x0000 00A8		
Physical Address	0x20AA 24A8 0x20AA 44A8 0x20AE 24A8 0x20AE 44A8	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 10. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG10																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG10	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4831 8000

Table 30-635. Register Call Summary for Register PRUSS_DBG_CT_REG10

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-636. PRUSS_DBG_CT_REG11

Address Offset	0x0000 00AC		
Physical Address	0x20AA 24AC 0x20AA 44AC 0x20AE 24AC 0x20AE 44AC	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 11. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG11																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG11	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4802 2000

Table 30-637. Register Call Summary for Register PRUSS_DBG_CT_REG11

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-638. PRUSS_DBG_CT_REG12

Address Offset	0x0000 00B0		
Physical Address	0x20AA 24B0 0x20AA 44B0 0x20AE 24B0 0x20AE 44B0	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 12. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG12																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG12	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4802 4000

Table 30-639. Register Call Summary for Register PRUSS_DBG_CT_REG12

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-640. PRUSS_DBG_CT_REG13

Address Offset	0x0000 00B4		
Physical Address	0x20AA 24B4 0x20AA 44B4 0x20AE 24B4 0x20AE 44B4	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 13. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG13																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG13	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4831 0000

Table 30-641. Register Call Summary for Register PRUSS_DBG_CT_REG13

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-642. PRUSS_DBG_CT_REG14

Address Offset	0x0000 00B8		
Physical Address	0x20AA 24B8 0x20AA 44B8 0x20AE 24B8 0x20AE 44B8	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 14. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG14																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG14	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x481C C000

Table 30-643. Register Call Summary for Register PRUSS_DBG_CT_REG14

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-644. PRUSS_DBG_CT_REG15

Address Offset	0x0000 00BC		
Physical Address	0x20AA 24BC 0x20AA 44BC 0x20AE 24BC 0x20AE 44BC	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 15. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG15																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG15	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x481D 0000

Table 30-645. Register Call Summary for Register PRUSS_DBG_CT_REG15

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-646. PRUSS_DBG_CT_REG16

Address Offset	0x0000 00C0		
Physical Address	0x20AA 24C0 0x20AA 44C0 0x20AE 24C0 0x20AE 44C0	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 16. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG16																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG16	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x481A 0000

Table 30-647. Register Call Summary for Register PRUSS_DBG_CT_REG16

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-648. PRUSS_DBG_CT_REG17

Address Offset	0x0000 00C4		
Physical Address	0x20AA 24C4 0x20AA 44C4 0x20AE 24C4 0x20AE 44C4	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 17. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG17																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG17	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4819 C000

Table 30-649. Register Call Summary for Register PRUSS_DBG_CT_REG17

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-650. PRUSS_DBG_CT_REG18

Address Offset	0x0000 00C8		
Physical Address	0x20AA 24C8 0x20AA 44C8 0x20AE 24C8 0x20AE 44C8	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 18. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG18																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG18	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4830 0000

Table 30-651. Register Call Summary for Register PRUSS_DBG_CT_REG18

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-652. PRUSS_DBG_CT_REG19

Address Offset	0x0000 00CC		
Physical Address	0x20AA 24CC 0x20AA 44CC 0x20AE 24CC 0x20AE 44CC	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 19. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG19																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG19	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4830 2000

Table 30-653. Register Call Summary for Register PRUSS_DBG_CT_REG19

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-654. PRUSS_DBG_CT_REG20

Address Offset	0x0000 00D0		
Physical Address	0x20AA 24D0 0x20AA 44D0 0x20AE 24D0 0x20AE 44D0	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 20. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG20																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG20	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x4830 4000

Table 30-655. Register Call Summary for Register PRUSS_DBG_CT_REG20

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-656. PRUSS_DBG_CT_REG21

Address Offset	0x0000 00D4		
Physical Address	0x20AA 24D4 0x20AA 44D4 0x20AE 24D4 0x20AE 44D4	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 21. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG21																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG21	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x03 2400

Table 30-657. Register Call Summary for Register PRUSS_DBG_CT_REG21

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-658. PRUSS_DBG_CT_REG22

Address Offset	0x0000 00D8		
Physical Address	0x20AA 24D8 0x20AA 44D8 0x20AE 24D8 0x20AE 44D8	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 22. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG22																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG22	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x480C 8000

Table 30-659. Register Call Summary for Register PRUSS_DBG_CT_REG22

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-660. PRUSS_DBG_CT_REG23

Address Offset	0x0000 00DC		
Physical Address	0x20AA 24DC 0x20AA 44DC 0x20AE 24DC 0x20AE 44DC	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 23. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG23																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG23	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.	R	0x480C A000

Table 30-661. Register Call Summary for Register PRUSS_DBG_CT_REG23

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-662. PRUSS_DBG_CT_REG24

Address Offset	0x0000 00E0		
Physical Address	0x20AA 24E0 0x20AA 44E0 0x20AE 24E0 0x20AE 44E0	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 24. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG24																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG24	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table. This entry is partially programmable through the c24_blk_index in the PRU Control register. The reset value for this Constant Table Entry is 0x00000n00, n=c24_blk_index[3:0].	R	0x0

Table 30-663. Register Call Summary for Register PRUSS_DBG_CT_REG24

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-664. PRUSS_DBG_CT_REG25

Address Offset	0x0000 00E4		
Physical Address	0x20AA 24E4 0x20AA 44E4 0x20AE 24E4 0x20AE 44E4	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 25. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG25																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG25	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table. This entry is partially programmable through the c25_blk_index in the PRU Control register. The reset value for this Constant Table Entry is 0x00002n00, n=c25_blk_index[3:0].	R	0x0

Table 30-665. Register Call Summary for Register PRUSS_DBG_CT_REG25

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-666. PRUSS_DBG_CT_REG26

Address Offset	0x0000 00E8		
Physical Address	0x20AA 24E8 0x20AA 44E8 0x20AE 24E8 0x20AE 44E8	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 26. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG26																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG26	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table. This entry is partially programmable through the c26_blk_index in the PRU Control register. The reset value for this Constant Table Entry is 0x0002En00, n=c26_blk_index[3:0].	R	0x0

Table 30-667. Register Call Summary for Register PRUSS_DBG_CT_REG26

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-668. PRUSS_DBG_CT_REG27

Address Offset	0x0000 00EC		
Physical Address	0x20AA 24EC 0x20AA 44EC 0x20AE 24EC 0x20AE 44EC	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 27. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG27																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG27	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table. This entry is partially programmable through the c27_blk_index in the PRU Control register. The reset value for this Constant Table Entry is 0x00032n00, n=c27_blk_index[3:0].	R	0x0

Table 30-669. Register Call Summary for Register PRUSS_DBG_CT_REG27

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-670. PRUSS_DBG_CT_REG28

Address Offset	0x0000 00F0		
Physical Address	0x20AA 24F0 0x20AA 44F0 0x20AE 24F0 0x20AE 44F0	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 28. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG28																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG28	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table. This entry is partially programmable through the c28_pointer in the PRU Control register. The reset value for this Constant Table Entry is 0x00nnnn00, nnnn=c28_pointer[15:0].	R	0x0

Table 30-671. Register Call Summary for Register PRUSS_DBG_CT_REG28

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-672. PRUSS_DBG_CT_REG29

Address Offset	0x0000 00F4		
Physical Address	0x20AA 24F4 0x20AA 44F4 0x20AE 24F4 0x20AE 44F4	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 29. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG29																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG29	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table. This entry is partially programmable through the c29_pointer in the PRU Control register. The reset value for this Constant Table Entry is 0x49nnnn00, nnnn=c29_pointer[15:0].	R	0x0

Table 30-673. Register Call Summary for Register PRUSS_DBG_CT_REG29

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-674. PRUSS_DBG_CT_REG30

Address Offset	0x0000 00F8		
Physical Address	0x20AA 24F8 0x20AA 44F8 0x20AE 24F8 0x20AE 44F8	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 30. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG30																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG30	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table. This entry is partially programmable through the c30_pointer in the PRU Control register. The reset value for this Constant Table Entry is 0x40nnnn00, nnnn=c30_pointer[15:0].	R	0x0

Table 30-675. Register Call Summary for Register PRUSS_DBG_CT_REG30

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

Table 30-676. PRUSS_DBG_CT_REG31

Address Offset	0x0000 00FC		
Physical Address	0x20AA 24FC 0x20AA 44FC 0x20AE 24FC 0x20AE 44FC	**Instance**	PRUSS1_PRU0_DEBUG PRUSS1_PRU1_DEBUG PRUSS2_PRU0_DEBUG PRUSS2_PRU0_DEBUG
Description	DEBUG PRU CONSTANTS TABLE ENTRY 31. This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT_REG31																															

Bits	Field Name	Description	Type	Reset
31:0	CT_REG31	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table. This entry is partially programmable through the c31_pointer in the PRU Control register. The reset value for this Constant Table Entry is 0x80nnnn00, nnnn=c31_pointer[15:0].	R	0x0

Table 30-677. Register Call Summary for Register PRUSS_DBG_CT_REG31

PRU-ICSS PRU Cores

- PRUSS_PRU_DEBUG Register Summary: [0] [1]

30.2.6 PRU-ICSS Local Interrupt Controller

This section describes functionality of the PRU-ICSS integrated Interrupt Controller - PRUSS_INTC.

30.2.6.1 PRU-ICSS Interrupt Controller Overview

The PRU-ICSS interrupt controller (PRUSS_INTC) maps interrupts coming from different parts of the device (mapped to PRU-ICSS1/PRU-ICSS2 via the device IRQ_CROSSBAR) to a reduced set of PRU-ICSS interrupt channels.

The PRUSS_INTC has the following features:

- Capturing up to 64 System Events (inputs)
- Supports up to 10 output interrupt channels.
- Generation of 10 Host Interrupts
  - 2 Host Interrupts for the PRUs.
  - 8 Host Interrupts exported from the PRU-ICSS for signaling the ARM interrupt controllers.
- Each system event can be enabled and disabled.
- Each host event can be enabled and disabled.
- Hardware prioritization of events.

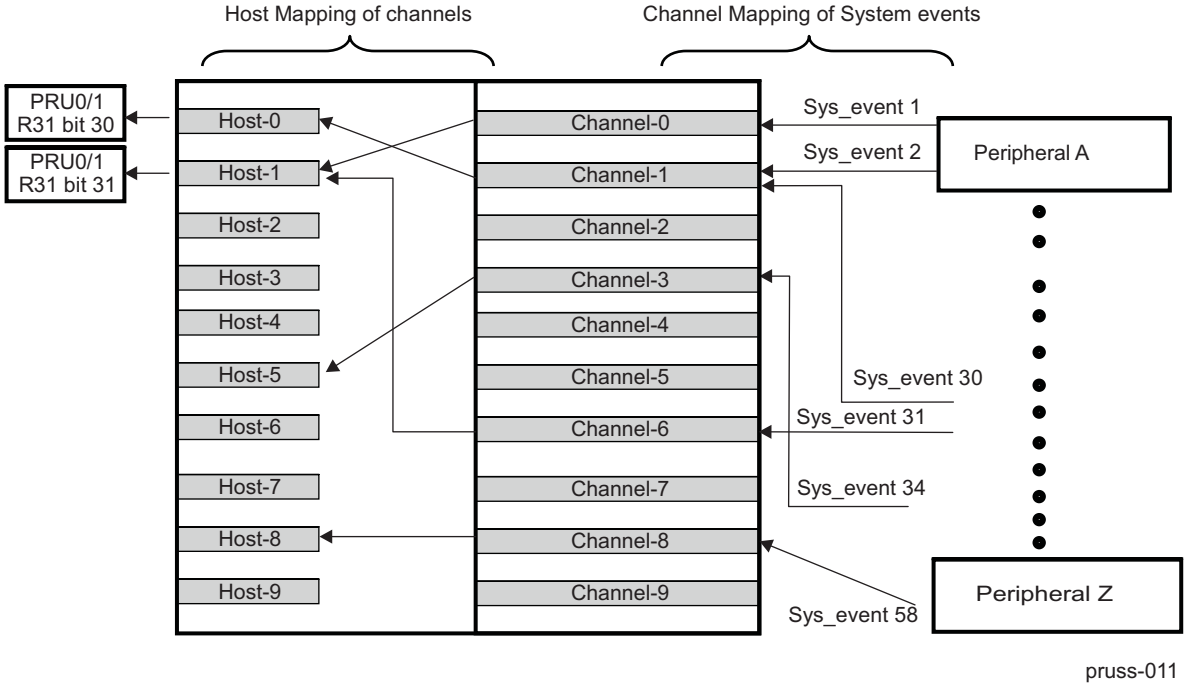
30.2.6.2 PRU-ICSS Interrupt Controller Functional Description

The PRU-ICSS incorporates an interrupt controller - PRUSS_INTC that supports up to 64 system interrupts from different peripherals (including 32 interrupts from PRU-ICSS located interrupt sources). The PRUSS_INTC maps these system events to 10 channels inside the PRUSS_INTC (see Figure 30-72). Interrupts from these 10 channels are further mapped to 10 Host Interrupts.

- Any of the 64 system interrupts can be mapped to any of the 10 channels.
- Multiple interrupts can be mapped to a single channel.
- An interrupt should not be mapped to more than one channel.
- Any of the 10 channels can be mapped to any of the 10 host interrupts. It is recommended to map channel “x” to host interrupt “x”, where x is from 0 to 9
- A channel should not be mapped to more than one host interrupt
- For channels mapping to the same host interrupt, lower number channels have higher priority.
- For interrupts on same channel, priority is determined by the hardware interrupt number. The lower the interrupt number, the higher the priority.
- Host Interrupt 0 is connected to bit 30 in register 31 (R31) of PRU0 and PRU1.
- Host Interrupt 1 is connected to bit 31 in register 31 (R31) for PRU0 and PRU1.
- Host Interrupts 2 through 9 exported from PRU-ICSS and mapped to interrupt controllers in the device.

NOTE: The Host interrupt 8 and host interrupt 9 are also exported as DMA requests to the device instantiated DMA_CROSSBAR which in turn can remap them to each line of the device integrated SDMA, EDMA, DSP1_EDMA and DSP2_EDMA controllers. For more details on PRU-ICSS DMA request outputs mapping, refer to the Section 30.2.3.

Figure 30-72. PRU-ICSS Interrupt Controller Block Diagram



30.2.6.2.1 PRU-ICSS Interrupt Controller System Events

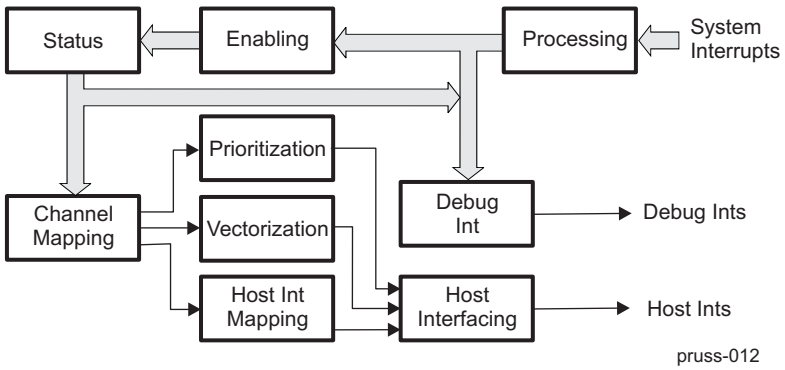
The PRU-ICSS system events - interrupt inputs. The device includes a internal mux that selects the Standard (default) or MII_RT mode system events. The mux control signal is controlled by PRUSS_MII_RT[0] MII_RT_EVENT_EN, which can be modified by software in PRU-ICSS CFG register space.

30.2.6.2.2 PRU-ICSS Interrupt Controller System Events Flow

The PRUSS_INTC module controls the system event mapping to the host interrupt interface. System events are generated by the device peripherals or PRUs. The PRUSS_INTC receives the system interrupts and maps them to internal channels. The channels are used to group interrupts together and to prioritize them. These channels are then mapped onto the host interrupts. Interrupts from the system side are active high in polarity. They are also pulse type of interrupts.

The PRUSS_INTC encompasses many functions to process the system interrupts and prepare them for the host interface. These functions are: processing, enabling, status, channel mapping, host interrupt mapping, prioritization, and host interfacing. Figure 30-73 illustrates the flow of system interrupts through the functions to the host. The following subsections describe each part of the flow.

Figure 30-73. Flow of System Interrupts to Host



30.2.6.2.2.1 PRU-ICSS Interrupt Processing

This block does following tasks:

- Synchronization of slower and asynchronous interrupts
- Conversion of polarity to active high
- Conversion of interrupt type to pulse interrupts

After the processing block, all interrupts will be active high pulses.

30.2.6.2.2.1.1 PRU-ICSS Interrupt Enabling

The next stage of PRUSS_INTC is to enable system interrupts based on programmed settings. The following sequence is to be followed to enable interrupts:

- Enable required system interrupts: System interrupts that are required to get propagated to host are to be enabled individually by writing to INDEX field in the system interrupt enable indexed set register (PRUSS_INTC_EISR). The interrupt to enable is the index value written. This sets the Enable Register bit of the given index.
- Enable required host interrupts: By writing 1 to the appropriate bit of the INDEX field in the host interrupt enable indexed set register (PRUSS_INTC_HIEISR), enable the required host interrupts. The host interrupt to enable is the index value written. This enables the host interrupt output or triggers the output again if that host interrupt is already enabled.
- Enable all host interrupts: By setting the ENABLE bit in the global enable register (PRUSS_INTC_GER) to 1, all host interrupts will be enabled. Individual host interrupts are still enabled or disabled from their individual enables and are not overridden by the global enable.

30.2.6.2.2.2 PRU-ICSS Interrupt Status Checking

The next stage is to capture which system interrupts are pending. There are two kinds of pending status: raw status and enabled status. Raw status is the pending status of the system interrupt without regards to the enable bit for the system interrupt. Enabled status is the pending status of the system interrupts with the enable bits active. When the enable bit is inactive, the enabled status will always be inactive. The enabled status of system interrupts is captured in system interrupt status enabled/clear registers (PRUSS_INTC_SECR1 and PRUSS_INTC_SECR0).

Status of system interrupt 'N' is indicated by the N-th bit of PRUSS_INTC_SECR1 and PRUSS_INTC_SECR0. Since there are 64 system interrupts, two 32-bit registers are used to capture the enabled status of interrupts. The pending status reflects whether the system interrupt occurred since the last time the status register bit was cleared. Each bit in the status register can be individually cleared.

30.2.6.2.2.3 PRU-ICSS Interrupt Channel Mapping

The PRUSS_INTC has 10 internal channels to which enabled system interrupts can be mapped. Channel 0 has highest priority and channel 9 has the lowest priority. Channels are used to group the system interrupts into a smaller number of priorities that can be given to a host interface with a very small number of interrupt inputs.

When multiple system interrupts are mapped to the same channel their interrupts are ORed together so that when either is active the output is active. The channel map registers (PRUSS_INTC_CMRi, where i=0 to 15) define the channel for each system interrupt. There is one register per 4 system interrupts; therefore, there are 16 channel map registers for a system of 64 interrupts. The channel for each system interrupt can be set using these registers.

30.2.6.2.2.3.1 PRU-ICSS Host Interrupt Mapping

The hosts can be the local PRU processors (PRU0 and PRU1) as well as device processors located outside PRU-ICSS such as MPU Cortex-A15, DSP1, IPU1, EVEs, etc. The 10 channels from the PRUSS_INTC can be mapped to any of the 10 Host interrupts. The Host map registers (PRUSS_INTC_HMR0 - PRUSS_INTC_HMR2) define the channel for each system interrupt. There is one register per 4 channels; therefore, there are 3 host map registers for 10 channels. When multiple channels are mapped to the same host interrupt, then prioritization is done to select which interrupt is in the highest-priority channel and which should be sent first to the host.

30.2.6.2.2.3.2 PRU-ICSS Interrupt Prioritization

The next stage of the PRUSS_INTC is prioritization. Since multiple interrupts can feed into a single channel and multiple channels can feed into a single host interrupt, it is to read the status of all system interrupts to determine the highest priority interrupt that is pending. The PRUSS_INTC provides hardware to perform this prioritization with a given scheme so that software does not have to do this. There are two levels of prioritizations:

- The first level of prioritization is between the active channels for a host interrupt. Channel 0 has the highest priority and channel 9 has the lowest. So the first level of prioritization picks the lowest numbered active channel.
- The second level of prioritization is between the active system interrupts for the prioritized channel. The system interrupt in position 0 has the highest priority and system interrupt 63 has the lowest priority. So the second level of prioritization picks the lowest position active system interrupt.

This is the final prioritized system interrupt for the host interrupt and is stored in the global prioritized index register (PRUSS_INTC_GPIR). The highest priority pending interrupt with respect to each host interrupts can be obtained using the host interrupt prioritized index registers (PRUSS_INTC_HIPIRj where j=0 to 9).

30.2.6.2.2.4 PRU-ICSS Interrupt Nesting

The PRUSS_INTC can also perform a nesting function in its prioritization. Nesting is a method of disabling certain interrupts (usually lower-priority interrupts) when an interrupt is taken so that only those desired interrupts can trigger to the host while it is servicing the current interrupt. The typical usage is to nest on the current interrupt and disable all interrupts of the same or lower priority (or channel). Then the host will only be interrupted from a higher priority interrupt.

The nesting is done in one of three methods:

1. Nesting for all host interrupts, based on channel priority: When an interrupt is taken, the nesting level is set to its channel priority. From then, that channel priority and all lower priority channels will be disabled from generating host interrupts and only higher priority channels are allowed. When the interrupt is completely serviced, the nesting level is returned to its original value. When there is no interrupt being serviced, there are no channels disabled due to nesting. The global nesting level register (PRUSS_INTC_GNLR) allows the checking and setting of the global nesting level across all host interrupts. The nesting level is the channel (and all of lower priority channels) that are nested out because of a current interrupt.
2. Nesting for individual host interrupts, based on channel priority: Always nest based on channel priority for each host interrupt individually. When an interrupt is taken on a host interrupt, then, the nesting level is set to its channel priority for just that host interrupt, and other host interrupts do not have their nesting affected. Then for that host interrupt, equal or lower priority channels will not interrupt the host but may on other host interrupts if programmed. When the interrupt is completely serviced the nesting level for the host interrupt is returned to its original value. The host interrupt nesting level registers (PRUSS_INTC_HINLRj where j=0 to 9) display and control the nesting level for each host interrupt. The nesting level controls which channel and lower priority channels are nested. There is one register per host interrupt.
3. Software manually performs the nesting of interrupts. When an interrupt is taken, the software will disable all the host interrupts, manually update the enables for any or all the system interrupts, and then re-enables all the host interrupts. This now allows only the system interrupts that are still enabled to trigger to the host. When the interrupt is completely serviced the software must reverse the changes to re-enable the nested out system interrupts. This method requires the most software interaction but gives the most flexibility if simple channel based nesting mechanisms are not adequate.

30.2.6.2.2.5 PRU-ICSS Interrupt Status Clearing

After servicing the interrupt (after execution of the ISR), interrupt status is to be cleared. If a system interrupt status is not cleared, then another host interrupt may not be triggered or another host interrupt may be triggered incorrectly. It is also essential to clear all system interrupts before the PRU is halted as the PRU does not power down unless all the interrupt status are cleared. For clearing the status of an interrupt, whose interrupt number is N, write a 1 to the Nth bit position in the system interrupt status enabled/clear registers (PRUSS_INTC_SECR0 and PRUSS_INTC_SECR1). System interrupt N can also be cleared by writing the value N into the system interrupt status indexed clear register (PRUSS_INTC_SICR).

30.2.6.2.3 PRU-ICSS Interrupt Disabling

At any time, if any interrupt is not to be propagated to the host, then that interrupt should be disabled. For disabling an interrupt whose interrupt number is N, write a 1 to the Nth bit in the system interrupt enable clear registers (PRUSS_INTC_ECR0 and PRUSS_INTC_ECR1). System interrupt N can also be disabled by writing the value N in the system interrupt enable indexed clear register (PRUSS_INTC_EICR).

30.2.6.3 PRU-ICSS Interrupt Controller Basic Programming Model

Follow these steps to configure the interrupt controller.

1. Set polarity and type of system event through the System Interrupt Polarity Registers (PRUSS_INTC_SIPR1 and PRUSS_INTC_SIPR0) and the System Interrupt Type Registers (PRUSS_INTC_SITR1 and PRUSS_INTC_SITR0). Polarity of all system interrupts is always high. Type of all system interrupts is always pulse.
2. Map system event to PRUSS_INTC channel through PRUSS_INTC_CMRi (i=0 to 15) channel mapping registers.
3. Map channel to host interrupt through PRUSS_INTC_HMR0/1/2 registers. Recommended channel “x” to be mapped to host interrupt “x”.
4. Clear system interrupt by writing 1 to PRUSS_INTC_SECR0/1 registers.
5. Enable host interrupt by writing index value to PRUSS_INTC_HIEISR register.
6. Enable interrupt nesting if desired.
7. Globally enable all interrupts through register PRUSS_INTC_GER[0] ENABLE_HINT_ANY bit.

30.2.6.4 PRU-ICSS Interrupt Requests Mapping

The PRU-ICSS1_INTC/PRUSS2_INTC lines 0 through 31 are mapped to events which are generated by PRU-ICSS integrated modules. Table 30-678 shows mapping of the different PRU-ICSS internally sourced IRQ events to PRUSS1_INTC/PRUSS2_INTC interrupt lines 0 through 31.

Table 30-678. PRU-ICSS1/PRU-ICSS2 Internal Interrupts

PRU-ICSS INTC IRQ input	PRU-ICSS Internal Interrupt Signal Name	Source
PRUSS1_INTC		
PRUSS1_IRQ_31	pr1_pru_mst_intr15_intr_req	pru0 or pru1
PRUSS1_IRQ_30	pr1_pru_mst_intr14_intr_req	pru0 or pru1
PRUSS1_IRQ_29	pr1_pru_mst_intr[13]_intr_req	pru0 or pru1
PRUSS1_IRQ_28	pr1_pru_mst_intr[12]_intr_req	pru0 or pru1
PRUSS1_IRQ_27	pr1_pru_mst_intr[11]_intr_req	pru0 or pru1
PRUSS1_IRQ_26	pr1_pru_mst_intr[10]_intr_req	pru0 or pru1
PRUSS1_IRQ_25	pr1_pru_mst_intr[9]_intr_req	pru0 or pru1
PRUSS1_IRQ_24	pr1_pru_mst_intr[8]_intr_req	pru0 or pru1
PRUSS1_IRQ_23	pr1_pru_mst_intr[7]_intr_req	pru0 or pru1
PRUSS1_IRQ_22	pr1_pru_mst_intr[6]_intr_req	pru0 or pru1
PRUSS1_IRQ_21	pr1_pru_mst_intr[5]_intr_req	pru0 or pru1
PRUSS1_IRQ_20	pr1_pru_mst_intr[4]_intr_req	pru0 or pru1
PRUSS1_IRQ_19	pr1_pru_mst_intr[3]_intr_req	pru0 or pru1
PRUSS1_IRQ_18	pr1_pru_mst_intr[2]_intr_req	pru0 or pru1
PRUSS1_IRQ_17	pr1_pru_mst_intr[1]_intr_req	pru0 or pru1
PRUSS1_IRQ_16	pr1_pru_mst_intr[0]_intr_req	pru0 or pru1
PRUSS1_IRQ_15	pr1_ecap_intr_req	PRUSS1 eCAP
PRUSS1_IRQ_8	digio_event_req	PRUSS1 IEP (Ethercat)
PRUSS1_IRQ_7	pr1_iep_tim_cap_cmp_pend	PRUSS1 IEP
PRUSS1_IRQ_6	pr1_uart_uint_intr_req	PRUSS1 UART
PRUSS1_IRQ_5	pr1_uart_utxevt_intr_req	PRUSS1 UART
PRUSS1_IRQ_4	pr1_uart_urxevt_intr_req	PRUSS1 UART
PRUSS1_IRQ_3	pr1_xfr_timeout	PRUSS1 Scratch Pad
PRUSS1_IRQ_2	pr1_pru1_r31_status_cnt16	PRUSS1.PRU1 (Shift Capture)
PRUSS1_IRQ_1	pr1_pru0_r31_status_cnt16	PRUSS1.PRU0 (Shift Capture)
PRUSS1_IRQ_0	pr1_parity_err_intr_pend	PRUSS1 Parity Logic
PRUSS2_INTC		
PRUSS2_IRQ_31	pr2_pru_mst_intr[15]_intr_req	pru0 or pru1
PRUSS2_IRQ_30	pr2_pru_mst_intr[14]_intr_req	pru0 or pru1
PRUSS2_IRQ_29	pr2_pru_mst_intr[13]_intr_req	pru0 or pru1
PRUSS2_IRQ_28	pr2_pru_mst_intr[12]_intr_req	pru0 or pru1
PRUSS2_IRQ_27	pr2_pru_mst_intr[11]_intr_req	pru0 or pru1
PRUSS2_IRQ_26	pr2_pru_mst_intr[10]_intr_req	pru0 or pru1
PRUSS2_IRQ_25	pr2_pru_mst_intr[9]_intr_req	pru0 or pru1
PRUSS2_IRQ_24	pr2_pru_mst_intr[8]_intr_req	pru0 or pru1
PRUSS2_IRQ_23	pr2_pru_mst_intr[7]_intr_req	pru0 or pru1
PRUSS2_IRQ_22	pr2_pru_mst_intr[6]_intr_req	pru0 or pru1
PRUSS2_IRQ_21	pr2_pru_mst_intr[5]_intr_req	pru0 or pru1
PRUSS2_IRQ_20	pr2_pru_mst_intr[4]_intr_req	pru0 or pru1
PRUSS2_IRQ_19	pr2_pru_mst_intr[3]_intr_req	pru0 or pru1
PRUSS2_IRQ_18	pr2_pru_mst_intr[2]_intr_req	pru0 or pru1

Table 30-678. PRU-ICSS1/PRU-ICSS2 Internal Interrupts (continued)

PRU-ICSS INTC IRQ input	PRU-ICSS Internal Interrupt Signal Name	Source
PRUSS2_IRQ_17	pr2_pru_mst_intr[1]_intr_req	pru0 or pru1
PRUSS2_IRQ_16	pr2_pru_mst_intr[0]_intr_req	pru0 or pru1
PRUSS2_IRQ_15	pr2_ecap_intr_req	PRUSS2 eCAP
PRUSS2_IRQ_8	pr2_digio_event_req	PRUSS2 IEP (Ethercat)
PRUSS2_IRQ_7	pr2_iep_tim_cap_cmp_pend	PRUSS2 IEP
PRUSS2_IRQ_6	pr2_uart_uint_intr_req	PRUSS2 UART
PRUSS2_IRQ_5	pr2_uart_utxevt_intr_req	PRUSS2 UART
PRUSS2_IRQ_4	pr2_uart_urxevt_intr_req	PRUSS2 UART
PRUSS2_IRQ_3	pr2_xfr_timeout	PRUSS2 Scratch Pad
PRUSS2_IRQ_2	pr2_pru1_r31_status_cnt16	PRUSS2.PRU1 (Shift Capture)
PRUSS2_IRQ_1	pr2_pru0_r31_status_cnt16	PRUSS2.PRU0 (Shift Capture)
PRUSS2_IRQ_0	pr2_parity_err_intr_pend	PRUSS2 Parity Logic

The IRQ input lines 32 through 63 receive interrupts which come from various device peripherals located outside PRU-ICSS1 and PRU-ICSS2. They are delivered on the PRUSS1_INTC / PRUSS2_INTC inputs (32 through 63) via the device IRQ_CROSSBAR. For more details on the device IRQ_CROSSBAR signals mapping to the PRUSS1_INTC / PRUSS2_INTC, refer to the Chapter 17, *Interrupt Controllers*. For more details on how to program mapping of the external peripheral IRQ signals to PRUSS1_IRQ_32 through PRUSS1_IRQ_63 / PRUSS2_IRQ_32 through PRUSS2_IRQ_55 inputs of the PRUSS1_INTC/PRUSS2_INTC, respectively, refer to the Section 18.4.6.4, *IRQ_CROSSBAR Module Functional Description*, in Chapter 18, *Control Module*.

Note that for the PRUSS_INTC input lines **32 through 55**, there is an additional multiplexing option programmable in the PRUSS_CFG located register bit - PRUSS_MII_RT[0] MII_RT_EVENT_EN. By default the MII_RT_EVENT_EN is set to 0b0 which selects the IRQ sources to be the PRU-ICSS dedicated device IRQ_CROSSBAR outputs ("**Standard**"). By setting MII_RT_EVENT_EN to 0b1, a set of PRU-ICSS MII_RT module associated events, are mapped to the same lines.

The Table 30-679 and the Table 30-680 shows PRU-ICSS1/PRU-ICSS2 MII_RT events mapping on the PRUSS1_INTC / PRUSS2_INTC inputs PRUSS1_IRQ_32 through PRUSS1_IRQ_55 / PRUSS2_IRQ_32 through PRUSS2_IRQ_55 valid for the "MII_RT" mode (with PRUSS_MII_RT[0] MII_RT_EVENT_EN register bit set to "0b1")

Table 30-679. PRU-ICSS1 MII_RT Mode Interrupts

PRU-ICSS1 IRQ [1]	Signal Name (MII_RT Mode enabled) - PRUSS_MII_RT[0] MII_RT_EVENT_EN=0b1
PRUSS1_IRQ_55	Reserved
PRUSS1_IRQ_54	PRU1_RX_EOF
PRUSS1_IRQ_53	MDIO_MII_LINK[1]
PRUSS1_IRQ_52	PORT1_TX_OVERFLOW
PRUSS1_IRQ_51	PORT1_TX_UNDERFLOW
PRUSS1_IRQ_50	PRU1_RX_OVERFLOW
PRUSS1_IRQ_49	PRU1_RX_NIBBLE_ODD
PRUSS1_IRQ_48	PRU1_RX_CRC
PRUSS1_IRQ_47	PRU1_RX_SOF
PRUSS1_IRQ_46	PRU1_RX_SFD
PRUSS1_IRQ_45	PRU1_RX_ERR32
PRUSS1_IRQ_44	PRU1_RX_ERR
PRUSS1_IRQ_43	Reserved
PRUSS1_IRQ_42	PRU0_RX_EOF

(1) Signals 63–56 and 31–0 for MII_RT Mode are the same as for Standard Mode.

Table 30-679. PRU-ICSS1 MII_RT Mode Interrupts (continued)

PRU-ICSS1 IRQ [1]	Signal Name (MII_RT Mode enabled) - PRUSS_MII_RT[0] MII_RT_EVENT_EN=0b1
PRUSS1_IRQ_41	MDIO_MII_LINK[0]
PRUSS1_IRQ_40	PORT0_TX_OVERFLOW
PRUSS1_IRQ_39	PORT0_TX_UNDERFLOW
PRUSS1_IRQ_38	PRU0_RX_OVERFLOW
PRUSS1_IRQ_37	PRU0_RX_NIBBLE_ODD
PRUSS1_IRQ_36	PRU0_RX_CRC
PRUSS1_IRQ_35	PRU0_RX_SOF
PRUSS1_IRQ_34	PRU0_RX_SFD
PRUSS1_IRQ_33	PRU0_RX_ERR32
PRUSS1_IRQ_32	PRU0_RX_ERR

Table 30-680. PRU-ICSS2 MII_RT Mode Interrupts

PRU-ICSS2 IRQ [1]	Signal Name (MII_RT Mode enabled) - PRUSS_MII_RT[0] MII_RT_EVENT_EN=0b1
PRUSS2_IRQ_55	Reserved
PRUSS2_IRQ_54	PRU1_RX_EOF
PRUSS2_IRQ_53	MDIO_MII_LINK[1]
PRUSS2_IRQ_52	PORT1_TX_OVERFLOW
PRUSS2_IRQ_51	PORT1_TX_UNDERFLOW
PRUSS2_IRQ_50	PRU1_RX_OVERFLOW
PRUSS2_IRQ_49	PRU1_RX_NIBBLE_ODD
PRUSS2_IRQ_48	PRU1_RX_CRC
PRUSS2_IRQ_47	PRU1_RX_SOF
PRUSS2_IRQ_46	PRU1_RX_SFD
PRUSS2_IRQ_45	PRU1_RX_ERR32
PRUSS2_IRQ_44	PRU1_RX_ERR
PRUSS2_IRQ_43	Reserved
PRUSS2_IRQ_42	PRU0_RX_EOF
PRUSS2_IRQ_41	MDIO_MII_LINK[0]
PRUSS2_IRQ_40	PORT0_TX_OVERFLOW
PRUSS2_IRQ_39	PORT0_TX_UNDERFLOW
PRUSS2_IRQ_38	PRU0_RX_OVERFLOW
PRUSS2_IRQ_37	PRU0_RX_NIBBLE_ODD
PRUSS2_IRQ_36	PRU0_RX_CRC
PRUSS2_IRQ_35	PRU0_RX_SOF
PRUSS2_IRQ_34	PRU0_RX_SFD
PRUSS2_IRQ_33	PRU0_RX_ERR32
PRUSS2_IRQ_32	PRU0_RX_ERR

(1) Signals 63–56 and 31–0 for MII_RT Mode are the same as for Standard Mode.

While in the Standard mode (default), the PRU-ICSS interrupt controller PRUSS_IRQ_32 through PRUSS_IRQ_55 input lines are mapped to PRU-ICSS external events via the device IRQ_CROSSBAR, in the MII_RT mode (bit MII_RT_EVENT_EN=0b1), the same PRUSS_INTC inputs are directly mapped to PRU-ICSS internally or externally generated MII_MDIO and MII_RT RX/TX signals (i.e. not through the IRQ_CROSSBAR).

For more details on the device IRQ_CROSSBAR signals mapping to the PRUSS1_INTC/ PRUSS2_INTC , refer to the Chapter 17, *Interrupt Controllers*. For more details on the PRU-ICSS1/PRU-ICSS2 external peripheral IRQ signals programmable mapping to PRUSS1_IRQ_32 through PRUSS1_IRQ_55 / PRUSS2_IRQ_32 through PRUSS2_IRQ_55 inputs of the PRUSS1_INTC/PRUSS2_INTC, respectively, refer to the Section 18.4.6.4, *IRQ_CROSSBAR Module Functional Description*, in Chapter 18, *Control Module*.

30.2.6.5 PRU-ICSS Interrupt Controller Register Manual

This section describes the PRU-ICSS interrupt controller registers.

30.2.6.5.1 PRUSS_INTC Instance Summary

Table 30-681. PRUSS_INTC Instance Summary

Module Name	Base Address	Size
PRUSS1_INTC	0x4B22 0000	5380 Bytes
PRUSS2_INTC	0x4B2A 0000	5380 Bytes

30.2.6.5.2 PRUSS_INTC Registers

30.2.6.5.2.1 PRUSS_INTC Register Summary

Table 30-682. PRUSS1_INTC Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS1_INTC Base Address
PRUSS_INTC_REVID	R	32	0x0000 0000	0x4B22 0000
PRUSS_INTC_CR	RW	32	0x0000 0004	0x4B22 0004
PRUSS_INTC_GER	RW	32	0x0000 0010	0x4B22 0010
PRUSS_INTC_GNLR	RW	32	0x0000 001C	0x4B22 001C
PRUSS_INTC_SISR	W	32	0x0000 0020	0x4B22 0020
PRUSS_INTC_SICR	W	32	0x0000 0024	0x4B22 0024
PRUSS_INTC_EISR	W	32	0x0000 0028	0x4B22 0028
PRUSS_INTC_EICR	W	32	0x0000 002C	0x4B22 002C
PRUSS_INTC_HIEISR	RW	32	0x0000 0034	0x4B22 0034
PRUSS_INTC_HIDISR	RW	32	0x0000 0038	0x4B22 0038
PRUSS_INTC_GPIR	R	32	0x0000 0080	0x4B22 0080
PRUSS_INTC_SRSR0	RW	32	0x0000 0200	0x4B22 0200
PRUSS_INTC_SRSR1	RW	32	0x0000 0204	0x4B22 0204
PRUSS_INTC_SECR0	RW	32	0x0000 0280	0x4B22 0280
PRUSS_INTC_SECR1	RW	32	0x0000 0284	0x4B22 0284
PRUSS_INTC_ESR0	RW	32	0x0000 0300	0x4B22 0300
PRUSS_INTC_ERS1	RW	32	0x0000 0304	0x4B22 0304
PRUSS_INTC_ECR0	W	32	0x0000 0380	0x4B22 0380
PRUSS_INTC_ECR1	W	32	0x0000 0384	0x4B22 0384

Table 30-682. PRUSS1_INTC Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS1_INTC Base Address
PRUSS_INTC_CMRi[1]	RW	32	0x0000 0400 + (0x4*i)	0x4B22 0400 + (0x4*i)
PRUSS_INTC_HMR0	RW	32	0x0000 0800	0x4B22 0800
PRUSS_INTC_HMR1	RW	32	0x0000 0804	0x4B22 0804
PRUSS_INTC_HMR2	RW	32	0x0000 0808	0x4B22 0808
PRUSS_INTC_HIPIRj[2]	R	32	0x0000 0900 + (0x4*j)	0x4B22 0900 + (0x4*j)
PRUSS_INTC_SIPR0	RW	32	0x0000 0D00	0x4B22 0D00
PRUSS_INTC_SIPR1	RW	32	0x0000 0D04	0x4B22 0D04
PRUSS_INTC_SITR0	RW	32	0x0000 0D80	0x4B22 0D80
PRUSS_INTC_SITR1	RW	32	0x0000 0D84	0x4B22 0D84
PRUSS_INTC_HINLRj[2]	RW	32	0x0000 1100 + (0x4*j)	0x4B22 1100 + (0x4*j)
PRUSS_INTC_HIER	RW	32	0x0000 1500	0x4B22 1500

(1) i=0 to 15
(2) j=0 to 9

Table 30-683. PRUSS2_INTC Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS2_INTC Base Address
PRUSS_INTC_REVID	R	32	0x0000 0000	0x4B2A 0000
PRUSS_INTC_CR	RW	32	0x0000 0004	0x4B2A 0004
PRUSS_INTC_GER	RW	32	0x0000 0010	0x4B2A 0010
PRUSS_INTC_GNLR	RW	32	0x0000 001C	0x4B2A 001C
PRUSS_INTC_SISR	W	32	0x0000 0020	0x4B2A 0020
PRUSS_INTC_SICR	W	32	0x0000 0024	0x4B2A 0024
PRUSS_INTC_EISR	W	32	0x0000 0028	0x4B2A 0028
PRUSS_INTC_EICR	W	32	0x0000 002C	0x4B2A 002C
PRUSS_INTC_HIEISR	RW	32	0x0000 0034	0x4B2A 0034
PRUSS_INTC_HIDISR	RW	32	0x0000 0038	0x4B2A 0038
PRUSS_INTC_GPIR	R	32	0x0000 0080	0x4B2A 0080
PRUSS_INTC_SRSR0	RW	32	0x0000 0200	0x4B2A 0200
PRUSS_INTC_SRSR1	RW	32	0x0000 0204	0x4B2A 0204
PRUSS_INTC_SECR0	RW	32	0x0000 0280	0x4B2A 0280
PRUSS_INTC_SECR1	RW	32	0x0000 0284	0x4B2A 0284
PRUSS_INTC_ESR0	RW	32	0x0000 0300	0x4B2A 0300
PRUSS_INTC_ERS1	RW	32	0x0000 0304	0x4B2A 0304
PRUSS_INTC_ECR0	W	32	0x0000 0380	0x4B2A 0380
PRUSS_INTC_ECR1	W	32	0x0000 0384	0x4B2A 0384
PRUSS_INTC_CMRi[1]	RW	32	0x0000 0400 + (0x4*i)	0x4B2A 0400 + (0x4*i)
PRUSS_INTC_HMR0	RW	32	0x0000 0800	0x4B2A 0800
PRUSS_INTC_HMR1	RW	32	0x0000 0804	0x4B2A 0804
PRUSS_INTC_HMR2	RW	32	0x0000 0808	0x4B2A 0808
PRUSS_INTC_HIPIRj[2]	R	32	0x0000 0900 + (0x4*j)	0x4B2A 0900 + (0x4*j)
PRUSS_INTC_SIPR0	RW	32	0x0000 0D00	0x4B2A 0D00
PRUSS_INTC_SIPR1	RW	32	0x0000 0D04	0x4B2A 0D04
PRUSS_INTC_SITR0	RW	32	0x0000 0D80	0x4B2A 0D80
PRUSS_INTC_SITR1	RW	32	0x0000 0D84	0x4B2A 0D84

(1) i=0 to 15
(2) j=0 to 9

Table 30-683. PRUSS2_INTC Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS2_INTC Base Address
PRUSS_INTC_HINLRj[2]	RW	32	0x0000 1100 + (0x4*j)	0x4B2A 1100 + (0x4*j)
PRUSS_INTC_HIER	RW	32	0x0000 1500	0x4B2A 1500

30.2.6.5.2.2 PRUSS_INTC Register Description

Table 30-684. PRUSS_INTC_REVID

Address Offset	0x0000 0000		
Physical Address	0x4B22 0000 0x4B2A 0000	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	Revision ID Register		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REVISION																															

Bits	Field Name	Description	Type	Reset
31:0	REVISION	IP Revision	R	0x-[1]

(1) TI Internal data

Table 30-685. Register Call Summary for Register PRUSS_INTC_REVID

PRU-ICSS Local Interrupt Controller
- PRUSS_INTC Register Summary: [0] [1]

Table 30-686. PRUSS_INTC_CR

Address Offset	0x0000 0004		
Physical Address	0x4B22 0004 0x4B2A 0004	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Control Register holds global control parameters and can forces a soft reset on the module.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																											PRIORITY_HOLD_MODE	NEST_MODE		WAKEUP_MODE	RESERVED

Bits	Field Name	Description	Type	Reset
31:5	RESERVED		R	0x00000
4	PRIORITY_HOLD_MODE	Reserved	RW	0x0
3:2	NEST_MODE	The nesting mode. 0 = no nesting 1 = automatic individual nesting (per host interrupt) 2 = automatic global nesting (over all host interrupts) 3 = manual nesting	RW	0x0

Bits	Field Name	Description	Type	Reset
1	WAKEUP_MODE	Reserved	RW	0x0
0	RESERVED		R	0

Table 30-687. Register Call Summary for Register PRUSS_INTC_CR

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

Table 30-688. PRUSS_INTC_GER

Address Offset	0x0000 0010		
Physical Address	0x4B22 0010 0x4B2A 0010	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Global Host Interrupt Enable Register enables all the host interrupts. Individual host interrupts are still enabled or disabled from their individual enables and are not overridden by the global enable.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															ENABLE_HINT_ANY

Bits	Field Name	Description	Type	Reset
31:1	RESERVED		R	0x0000 000
0	ENABLE_HINT_ANY	The current global enable value when read. Writes set the global enable.	RW	0

Table 30-689. Register Call Summary for Register PRUSS_INTC_GER

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Processing: [0]
- PRU-ICSS Interrupt Controller Basic Programming Model: [1]
- PRUSS_INTC Register Summary: [2] [3]

Table 30-690. PRUSS_INTC_GNLR

Address Offset	0x0000 001C		
Physical Address	0x4B22 001C 0x4B2A 001C	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Global Nesting Level Register allows the checking and setting of the global nesting level across all host interrupts when automatic global nesting mode is set. The nesting level is the channel (and all of lower priority) that are nested out because of a current interrupt. This register is only available when nesting is configured.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUTO_OVERRIDE	RESERVED																							GLB_NEST_LEVEL							

Bits	Field Name	Description	Type	Reset
31	AUTO_OVERRIDE	Always read as 0. Writes of 1 override the automatic nesting and set the nesting_level to the written data.	W	0x0
30:9	RESERVED		R	0x00000
8:0	GLB_NEST_LEVEL	The current global nesting level (highest channel that is nested). Writes set the nesting level. In auto nesting mode this value is updated internally unless the auto_override bit is set.	RW	0x100

Table 30-691. Register Call Summary for Register PRUSS_INTC_GNLR

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Nesting: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-692. PRUSS_INTC_SISR

Address Offset	0x0000 0020		
Physical Address	0x4B22 0020 0x4B2A 0020	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Status Indexed Set Register allows setting the status of an interrupt. The interrupt to set is the index value written. This sets the Raw Status Register bit of the given index.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								STATUS_SET_INDEX							

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x0000 00
9:0	STATUS_SET_INDEX	Writes set the status of the interrupt given in the index value. Reads return 0.	W	0x00

Table 30-693. Register Call Summary for Register PRUSS_INTC_SISR

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

Table 30-694. PRUSS_INTC_SICR

Address Offset	0x0000 0024		
Physical Address	0x4B22 0024 0x4B2A 0024	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Status Indexed Clear Register allows clearing the status of an interrupt. The interrupt to clear is the index value written. This clears the Raw Status Register bit of the given index.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								STATUS_CLR_INDEX							

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x0000 00
9:0	STATUS_CLR_INDEX	Writes clear the status of the interrupt given in the index value. Reads return 0.	W	0x0

Table 30-695. Register Call Summary for Register PRUSS_INTC_SICR

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Status Clearing: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-696. PRUSS_INTC_EISR

Address Offset	0x0000 0028		
Physical Address	0x4B22 0028 0x4B2A 0028	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Enable Indexed Set Register allows enabling an interrupt. The interrupt to enable is the index value written. This sets the Enable Register bit of the given index.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								ENABLE_SET_INDEX							

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x0000 00
9:0	ENABLE_SET_INDEX	Writes set the enable of the interrupt given in the index value. Reads return 0.	W	0x0

Table 30-697. Register Call Summary for Register PRUSS_INTC_EISR

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Processing: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-698. PRUSS_INTC_EICR

Address Offset	0x0000 002C		
Physical Address	0x4B22 002C 0x4B2A 002C	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Enable Indexed Clear Register allows disabling an interrupt. The interrupt to disable is the index value written. This clears the Enable Register bit of the given index.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								ENABLE_CLR_INDEX							

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x0000 00
9:0	ENABLE_CLR_INDEX	Writes clear the enable of the interrupt given in the index value. Reads return 0.	W	0x0

Table 30-699. Register Call Summary for Register PRUSS_INTC_EICR

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Disabling: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-700. PRUSS_INTC_HIEISR

Address Offset	0x0000 0034		
Physical Address	0x4B22 0034 0x4B2A 0034	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Host Interrupt Enable Indexed Set Register allows enabling a host interrupt output. The host interrupt to enable is the index value written. This enables the host interrupt output or triggers the output again if already enabled.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						HINT_ENABLE_SET_INDEX									

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x0000 00
9:0	HINT_ENABLE_SET_INDEX	Writes set the enable of the host interrupt given in the index value. Reads return 0.	RW	0x0

Table 30-701. Register Call Summary for Register PRUSS_INTC_HIEISR

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Processing: [0]
- PRU-ICSS Interrupt Controller Basic Programming Model: [1]
- PRUSS_INTC Register Summary: [2] [3]

Table 30-702. PRUSS_INTC_HIDISR

Address Offset	0x0000 0038		
Physical Address	0x4B22 0038 0x4B2A 0038	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Host Interrupt Enable Indexed Clear Register allows disabling a host interrupt output. The host interrupt to disable is the index value written. This disables the host interrupt output.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						HINT_ENABLE_CLR_INDEX									

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x0000 00
9:0	HINT_ENABLE_CLR_INDEX	Writes clear the enable of the host interrupt given in the index value. Reads return 0.	RW	0x0

Table 30-703. Register Call Summary for Register PRUSS_INTC_HIDISR

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

Table 30-704. PRUSS_INTC_GPIR

Address Offset	0x0000 0080		
Physical Address	0x4B22 0080 0x4B2A 0080	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Global Prioritized Index Register shows the interrupt number of the highest priority interrupt pending across all the host interrupts.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GLB_NONE	RESERVED																							GLB_PRI_INTR							

Bits	Field Name	Description	Type	Reset
31	GLB_NONE	No Interrupt is pending. Can be used by host to test for a negative value to see if no interrupts are pending.	R	0x1
30:10	RESERVED		R	0x0000 00
9:0	GLB_PRI_INTR	The currently highest priority interrupt index pending across all the host interrupts.	R	0x0

Table 30-705. Register Call Summary for Register PRUSS_INTC_GPIR

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Channel Mapping: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-706. PRUSS_INTC_SRSR0

Address Offset	0x0000 0200		
Physical Address	0x4B22 0200 0x4B2A 0200	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Status Raw Set Register0 show the pending enabled status of the system interrupts 0 to 31. Software can write to the Status Set Registers to set a system interrupt without a hardware trigger. There is one bit per system interrupt.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAW_STATUS_31_0																															

Bits	Field Name	Description	Type	Reset
31:0	RAW_STATUS_31_0	System interrupt raw status and setting of the system interrupts 0 to 31. Reads return the raw status. Write a 1 in a bit position to set the status of the system interrupt. Writing a 0 has no effect.	RW	0x0

Table 30-707. Register Call Summary for Register PRUSS_INTC_SRSR0

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

Table 30-708. PRUSS_INTC_SRSR1

Address Offset	0x0000 0204		
Physical Address	0x4B22 0204 0x4B2A 0204	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Status Raw Set Register1 show the pending enabled status of the system interrupts 32 to 63. Software can write to the Status Set Registers to set a system interrupt without a hardware trigger. There is one bit per system interrupt.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAW_STATUS_63_32																															

Bits	Field Name	Description	Type	Reset
31:0	RAW_STATUS_63_32	System interrupt raw status and setting of the system interrupts 32 to 63. Reads return the raw status. Write a 1 in a bit position to set the status of the system interrupt. Writing a 0 has no effect.	RW	0x0

Table 30-709. Register Call Summary for Register PRUSS_INTC_SRSR1

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

Table 30-710. PRUSS_INTC_SECR0

Address Offset	0x0000 0280		
Physical Address	0x4B22 0280 0x4B2A 0280	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Status Enabled Clear Register0 show the pending enabled status of the system interrupts 0 to 31. Software can write to the Status Clear Registers to clear a system interrupt after it has been serviced. If a system interrupt status is not cleared then another host interrupt may not be triggered or another host interrupt may be triggered incorrectly. There is one bit per system interrupt.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENA_STATUS_31_0																															

Bits	Field Name	Description	Type	Reset
31:0	ENA_STATUS_31_0	System interrupt enabled status and clearing of the system interrupts 0 to 31. Reads return the enabled status (before enabling with the Enable Registers). Write a 1 in a bit position to clear the status of the system interrupt. Writing a 0 has no effect.	RW	0x0

Table 30-711. Register Call Summary for Register PRUSS_INTC_SECR0

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Status Checking: [0] [1]
- PRU-ICSS Interrupt Status Clearing: [2]
- PRU-ICSS Interrupt Controller Basic Programming Model: [3]
- PRUSS_INTC Register Summary: [4] [5]

Table 30-712. PRUSS_INTC_SECR1

Address Offset	0x0000 0284		
Physical Address	0x4B22 0284 0x4B2A 0284	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Status Enabled Clear Register1 show the pending enabled status of the system interrupts 32 to 63. Software can write to the Status Clear Registers to clear a system interrupt after it has been serviced. If a system interrupt status is not cleared then another host interrupt may not be triggered or another host interrupt may be triggered incorrectly. There is one bit per system interrupt.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENA_STATUS_63_32																															

Bits	Field Name	Description	Type	Reset
31:0	ENA_STATUS_63_32	System interrupt enabled status and clearing of the system interrupts 32 to 63. Reads return the enabled status (before enabling with the Enable Registers). Write a 1 in a bit position to clear the status of the system interrupt. Writing a 0 has no effect.	RW	0x0

Table 30-713. Register Call Summary for Register PRUSS_INTC_SECR1

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Status Checking: [0] [1]
- PRU-ICSS Interrupt Status Clearing: [2]
- PRUSS_INTC Register Summary: [3] [4]

Table 30-714. PRUSS_INTC_ESR0

Address Offset	0x0000 0300		
Physical Address	0x4B22 0300 0x4B2A 0300	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Enable Set Register0 enables system interrupts 0 to 31 to trigger outputs. System interrupts that are not enabled do not interrupt the host. There is a bit per system interrupt.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENABLE_SET_31_0																															

Bits	Field Name	Description	Type	Reset
31:0	ENABLE_SET_31_0	System interrupt enables system interrupts 0 to 31. Read returns the enable value (0 = disabled, 1 = enabled). Write a 1 in a bit position to set that enable. Writing a 0 has no effect.	RW	0x0

Table 30-715. Register Call Summary for Register PRUSS_INTC_ESR0

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

Table 30-716. PRUSS_INTC_ERS1

Address Offset	0x0000 0304		
Physical Address	0x4B22 0304 0x4B2A 0304	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Enable Set Register1 enables system interrupts 32 to 63 to trigger outputs. System interrupts that are not enabled do not interrupt the host. There is a bit per system interrupt.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENABLE_SET_63_32																															

Bits	Field Name	Description	Type	Reset
31:0	ENABLE_SET_63_32	System interrupt enables system interrupts 32 to 63. Read returns the enable value (0 = disabled, 1 = enabled). Write a 1 in a bit position to set that enable. Writing a 0 has no effect.	RW	0x0

Table 30-717. Register Call Summary for Register PRUSS_INTC_ERS1

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

Table 30-718. PRUSS_INTC_ECR0

Address Offset	0x0000 0380		
Physical Address	0x4B22 0380 0x4B2A 0380	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Enable Clear Register0 disables system interrupts 0 to 31 to map to channels. System interrupts that are not enabled do not interrupt the host. There is a bit per system interrupt.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENABLE_CLR_31_0																															

Bits	Field Name	Description	Type	Reset
31:0	ENABLE_CLR_31_0	System interrupt enables system interrupts 0 to 31. Write a 1 in a bit position to clear that enable. Writing a 0 has no effect.	W	0x0

Table 30-719. Register Call Summary for Register PRUSS_INTC_ECR0

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Disabling: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-720. PRUSS_INTC_ECR1

Address Offset	0x0000 0384		
Physical Address	0x4B22 0384 0x4B2A 0384	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Enable Clear Register1 disables system interrupts 32 to 63 to map to channels. System interrupts that are not enabled do not interrupt the host. There is a bit per system interrupt.		

Table 30-720. PRUSS_INTC_ECR1 (continued)

Type	W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENABLE_CLR_63_32																															

Bits	Field Name	Description	Type	Reset
31:0	ENABLE_CLR_63_32	System interrupt enables system interrupts 32 to 63. Write a 1 in a bit position to clear that enable. Writing a 0 has no effect.	W	0x0

Table 30-721. Register Call Summary for Register PRUSS_INTC_ECR1

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Disabling: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-722. PRUSS_INTC_CMRi

Address Offset	0x0000 0400 + (0x4*i)	**Index**	i = 0 to 15
Physical Address	0x4B22 0400 + (0x4*i) 0x4B2A 0400 + (0x4*i)	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	There are 16 identical CMR registers (i=0 to 15). The Channel Map Register_i specify the channel for the system interrupts k to k+3, where k=4*i. There is one register per 4 system interrupts.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				CH_MAP_3				RESERVED				CH_MAP_2				RESERVED				CH_MAP_1				RESERVED				CH_MAP_0			

Bits	Field Name	Description	Type	Reset
31:28	RESERVED		R	0x0
27:24	CH_MAP_3	Sets the channel for the system interrupt (k+3). Where k=i*4	RW	0x0
23:20	RESERVED		R	0x0
19:16	CH_MAP_2	Sets the channel for the system interrupt (k+2). Where k=i*4	RW	0x0
15:12	RESERVED		R	0x0
11:8	CH_MAP_1	Sets the channel for the system interrupt (k+1). Where k=i*4	RW	0x0
7:4	RESERVED		R	0x0
3:0	CH_MAP_0	Sets the channel for the system interrupt k. Where k=i*4	RW	0x0

Table 30-723. Register Call Summary for Register PRUSS_INTC_CMRi

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Channel Mapping: [0]
- PRU-ICSS Interrupt Controller Basic Programming Model: [1]
- PRUSS_INTC Register Summary: [2] [3]

Table 30-724. PRUSS_INTC_HMR0

Address Offset	0x0000 0800		
Physical Address	0x4B22 0800 0x4B2A 0800	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Host Interrupt Map Register0 define the host interrupt for channels 0 to 3. There is one register per 4 channels. Channels with forced host interrupt mappings will have their fields read-only.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				HINT_MAP_3				RESERVED				HINT_MAP_2				RESERVED				HINT_MAP_1				RESERVED				HINT_MAP_0			

Bits	Field Name	Description	Type	Reset
31:28	RESERVED		R	0x0
27:24	HINT_MAP_3	HOST INTERRUPT MAP FOR CHANNEL 3	RW	0x0
23:20	RESERVED		R	0x0
19:16	HINT_MAP_2	HOST INTERRUPT MAP FOR CHANNEL 2	RW	0x0
15:12	RESERVED		R	0x0
11:8	HINT_MAP_1	HOST INTERRUPT MAP FOR CHANNEL 1	RW	0x0
7:4	RESERVED		R	0x0
3:0	HINT_MAP_0	HOST INTERRUPT MAP FOR CHANNEL 0	RW	0x0

Table 30-725. Register Call Summary for Register PRUSS_INTC_HMR0

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Channel Mapping: [0]
- PRU-ICSS Interrupt Controller Basic Programming Model: [1]
- PRUSS_INTC Register Summary: [2] [3]

Table 30-726. PRUSS_INTC_HMR1

Address Offset	0x0000 0804		
Physical Address	0x4B22 0804 0x4B2A 0804	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Host Interrupt Map Register1 define the host interrupt for channels 4 to 7. There is one register per 4 channels. Chan_statusnels with forced host interrupt mappings will have their fields read-only.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				HINT_MAP_7				RESERVED				HINT_MAP_6				RESERVED				HINT_MAP_5				RESERVED				HINT_MAP_4			

Bits	Field Name	Description	Type	Reset
31:28	RESERVED		R	0x0
27:24	HINT_MAP_7	HOST INTERRUPT MAP FOR CHANNEL 7	RW	0x0
23:20	RESERVED		R	0x0
19:16	HINT_MAP_6	HOST INTERRUPT MAP FOR CHANNEL 6	RW	0x0
15:12	RESERVED		R	0x0
11:8	HINT_MAP_5	HOST INTERRUPT MAP FOR CHANNEL 5	RW	0x0
7:4	RESERVED		R	0x0
3:0	HINT_MAP_4	HOST INTERRUPT MAP FOR CHANNEL 4	RW	0x0

Table 30-727. Register Call Summary for Register PRUSS_INTC_HMR1

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

Table 30-728. PRUSS_INTC_HMR2

Address Offset	0x0000 0808		
Physical Address	0x4B22 0808 0x4B2A 0808	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Host Interrupt Map Register2 define the host interrupt for channels 8 to 9. There is one register per 4 channels. Channels with forced host interrupt mappings will have their fields read-only.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																				HINT_MAP_9				RESERVED				HINT_MAP_8			

Bits	Field Name	Description	Type	Reset
31:12	RESERVED		R	0x00000
11:8	HINT_MAP_9	HOST INTERRUPT MAP FOR CHANNEL 9	RW	0x0
7:4	RESERVED		R	0x0
3:0	HINT_MAP_8	HOST INTERRUPT MAP FOR CHANNEL 8	RW	0x0

Table 30-729. Register Call Summary for Register PRUSS_INTC_HMR2

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Channel Mapping: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-730. PRUSS_INTC_HIPIRj

Address Offset	0x0000 0900 + (0x4*j)	**Index**	j = 0 to 9
Physical Address	0x4B22 0900 + (0x4*j) 0x4B2A 0900 + (0x4*j)	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Host Interrupt Prioritized Index Register_j (where j=0 to 9) shows the highest priority current pending interrupt for the host interrupt j. There is one register per host interrupt.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NONE_HINT_j	RESERVED																							PRI_HINT_j							

Bits	Field Name	Description	Type	Reset
31	NONE_HINT	No pending interrupt.	R	0x1
30:10	RESERVED		R	0x0000 00
9:0	PRI_HINT	HOST INT j PRIORITIZED INTERRUPT. Interrupt number of the highest priority pending interrupt for this host interrupt.	R	0x0

Table 30-731. Register Call Summary for Register PRUSS_INTC_HIPIRj

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Channel Mapping: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-732. PRUSS_INTC_SIPR0

Address Offset	0x0000 0D00		
Physical Address	0x4B22 0D00 0x4B2A 0D00	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Polarity Register0 define the polarity of the system interrupts 0 to 31. There is a polarity for each system interrupt. The polarity of all system interrupts is active high; always write 1 to the bits of this register.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLARITY_31_0																															

Bits	Field Name	Description	Type	Reset
31:0	POLARITY_31_0	Interrupt polarity of the system interrupts 0 to 31. 0 = active low. 1 = active high.	RW	0x1

Table 30-733. Register Call Summary for Register PRUSS_INTC_SIPR0

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Controller Basic Programming Model: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-734. PRUSS_INTC_SIPR1

Address Offset	0x0000 0D04		
Physical Address	0x4B22 0D04 0x4B2A 0D04	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Polarity Register1 define the polarity of the system interrupts 32 to 63. There is a polarity for each system interrupt. The polarity of all system interrupts is active high; always write 1 to the bits of this register.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLARITY_63_32																															

Bits	Field Name	Description	Type	Reset
31:0	POLARITY_63_32	Interrupt polarity of the system interrupts 32 to 63. 0 = active low. 1 = active high.	RW	0x1

Table 30-735. Register Call Summary for Register PRUSS_INTC_SIPR1

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Controller Basic Programming Model: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-736. PRUSS_INTC_SITR0

Address Offset	0x0000 0D80		
Physical Address	0x4B22 0D80 0x4B2A 0D80	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Type Register0 define the type of the system interrupts 0 to 31. There is a type for each system interrupt. The type of all system interrupts is pulse; always write 0 to the bits of this register.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE_31_0																															

Bits	Field Name	Description	Type	Reset
31:0	TYPE_31_0	Interrupt type of the system interrupts 0 to 31. 0 = level or pulse interrupt. 1 = edge interrupt (required edge detect).	RW	0x0

Table 30-737. Register Call Summary for Register PRUSS_INTC_SITR0

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Controller Basic Programming Model: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-738. PRUSS_INTC_SITR1

Address Offset	0x0000 0D84		
Physical Address	0x4B22 0D84 0x4B2A 0D84	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The System Interrupt Type Register1 define the type of the system interrupts 32 to 63. There is a type for each system interrupt. The type of all system interrupts is pulse; always write 0 to the bits of this register.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE_63_32																															

Bits	Field Name	Description	Type	Reset
31:0	TYPE_63_32	Interrupt type of the system interrupts 32 to 63. 0 = level or pulse interrupt. 1 = edge interrupt (required edge detect).	RW	0x0

Table 30-739. Register Call Summary for Register PRUSS_INTC_SITR1

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Controller Basic Programming Model: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-740. PRUSS_INTC_HINLRj

Address Offset	0x0000 1100 + 0x4 * j	**Index**	j=0 to 9
Physical Address	0x4B22 1100 + (0x4*j) 0x4B2A 1100 + (0x4*j)	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Host Interrupt Nesting Level Register_j (where j=0 to 9) display and control the nesting level for host interrupt j. The nesting level controls which channel and lower priority channels are nested. There is one register per host interrupt.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUTO_OVERRIDE	RESERVED																							NEST_HINT_j							

Bits	Field Name	Description	Type	Reset
31	AUTO_OVERRIDE	Reads return 0. Writes of a 1 override the auto updating of the nesting_level and use the write data.	W	0x0
30:9	RESERVED		R	0x00000
8:0	NEST_HINT	Reads return the current nesting level for the host interrupt. Writes set the nesting level for the host interrupt. In auto mode the value is updated internally unless the auto_override is set and then the write data is used.	RW	0x100

Table 30-741. Register Call Summary for Register PRUSS_INTC_HINLRj

PRU-ICSS Local Interrupt Controller

- PRU-ICSS Interrupt Nesting: [0]
- PRUSS_INTC Register Summary: [1] [2]

Table 30-742. PRUSS_INTC_HIER

Address Offset	0x0000 1500		
Physical Address	0x4B22 1500 0x4B2A 1500	**Instance**	PRUSS1_INTC PRUSS2_INTC
Description	The Host Interrupt Enable Registers enable or disable individual host interrupts. These work separately from the global enables. There is one bit per host interrupt. These bits are updated when writing to the Host Interrupt Enable Index Set and Host Interrupt Enable Index Clear registers.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								ENABLE_HINT							

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x0000 00
9:0	ENABLE_HINT	The enable of the host interrupts (one per bit). 0 = disabled 1 = enabled	RW	0x0

Table 30-743. Register Call Summary for Register PRUSS_INTC_HIER

PRU-ICSS Local Interrupt Controller

- PRUSS_INTC Register Summary: [0] [1]

30.2.7 PRU-ICSS UART Module

This section describes an Universal Asynchronous Receive and Transmit (UART) module which is part of the device integrated PRU-ICSS1 and PRU-ICSS2 - PRUSS1_UART0 and PRUSS2_UART0, respectively.

30.2.7.1 PRU-ICSS UART Module Overview

30.2.7.1.1 Purpose of the PRU-ICSS integrated UART Peripheral

The PRUSS_UART0 peripheral is based on the industry standard TL16C550 asynchronous communications element, which in turn is a functional upgrade of the TL16C450. Functionally similar to the TL16C450 on power up (single character or TL16C450 mode), the PRUSS_UART0 can be placed in an alternate FIFO (TL16C550) mode. This relieves the CPU of excessive software overhead by buffering received and transmitted characters. The receiver and transmitter FIFOs store up to 16 bytes including three additional bits of error status per byte for the receiver FIFO.

The PRUSS_UART0 performs serial-to-parallel conversions on data received from a peripheral device and parallel-to-serial conversion on data received from the CPU. The CPU can read the PRUSS_UART0 status at any time. The PRUSS_UART0 includes control capability and a processor interrupt system that can be tailored to minimize software management of the communications link.

The PRUSS_UART0 includes a programmable baud generator capable of dividing the PRUSS_UART0 input clock by divisors from 1 to 65535 and producing a 16× reference clock or a 13× reference clock for the internal transmitter and receiver logic.

30.2.7.1.2 PRU-ICSS UART Key Features

30.2.7.1.2.1 PRU-ICSS UART Module Industry Standard Compliance Statement

The PRUSS_UART0 peripheral is based on the industry standard TL16C550 asynchronous communications element, which is a functional upgrade of the TL16C450. The information in this chapter assumes that user is familiar with these standards.

30.2.7.2 PRU-ICSS UART Environment

This section describes the PRUSS_UART0 module interface to the device environment

30.2.7.2.1 PRU-ICSS UART Pin Multiplexing

Extensive pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package. Pin multiplexing is controlled using a combination of hardware configuration at device reset and software programmable register settings. For more information on the PRUSS_UART0 pin multiplexing, refer to the Section 18.4.6.1.1, *Pad Configuration Registers* in the chapter, *Control Module*.

30.2.7.2.2 PRU-ICSS UART Signal Descriptions

The PRUSS_UART0 utilize a minimal number of signal connections to interface with external devices. The PRUSS_UART0 signal descriptions are included in Table 30-744.

Table 30-744. PRUSS_UART0 Signal Descriptions

Signal Name	Signal Type	Function
UART0_TXD	Output	Serial data transmit
UART0_RXD	Input	Serial data receive
$\overline{\text{UART0\_CTS}}$	Input	Clear-to-Send handshaking signal
$\overline{\text{UART0\_RTS}}$	Output	Request-to-Send handshaking signal

30.2.7.2.3 PRU-ICSS UART Data Format and Protocol Description

30.2.7.2.3.1 PRU-ICSS UART Transmission Protocol

The PRUSS_UART0 transmitter section includes a transmitter hold register (THR), memory mapped in the register PRUSS_UART_RBR_THR_REGISTERS[7:0] DATA bitfield and a transmitter shift register (TSR), which is NOT memory mapped. When the PRUSS_UART0 is in the FIFO mode, THR is a 16-byte FIFO. Transmitter section control is a function of the PRUSS_UART0 line control register PRUSS_UART_LINE_CONTROL_REGISTER. Based on the settings chosen in this register, the PRUSS_UART0 transmitter sends the following to the receiving device:

- 1 START bit
- 5, 6, 7, or 8 data bits
- 1 PARITY bit (optional)
- 1, 1.5, or 2 STOP bits

30.2.7.2.3.2 PRU-ICSS UART Reception Protocol

The PRUSS_UART0 receiver section includes a receiver shift register (RSR), that is not memory mapped, and a receiver buffer register (RBR), memory mapped as the register PRUSS_UART_RBR_THR_REGISTERS [7:0] DATA bitfield. When the PRUSS_UART0 is in the FIFO mode, RBR is a 16-byte FIFO. Receiver section control is a function of the PRUSS_UART0 line control register - PRUSS_UART_LINE_CONTROL_REGISTER. Based on the settings chosen in this register, the PRUSS_UART0 receiver accepts the following from the transmitting device:

- 1 START bit
- 5, 6, 7, or 8 data bits
- 1 PARITY bit (optional)
- 1 STOP bit (any other STOP bits transferred with the above data are not detected)

30.2.7.2.3.3 PRU-ICSS UART Data Format

The PRUSS_UART0 transmits in the following format:

1 START bit + data bits (5, 6, 7, 8) + 1 PARITY bit (optional) + STOP bit (1, 1.5, 2)

It transmits 1 START bit; 5, 6, 7, or 8 data bits, depending on the data width selection; 1 PARITY bit, if parity is selected; and 1, 1.5, or 2 STOP bits, depending on the STOP bit selection.

The PRUSS_UART0 receives in the following format:

1 START bit + data bits (5, 6, 7, 8) + 1 PARITY bit (optional) + 1 STOP bit

It receives 1 START bit; 5, 6, 7, or 8 data bits, depending on the data width selection; 1 PARITY bit, if parity is selected; and 1 STOP bit.

The protocol formats are shown in Figure 30-74.

Figure 30-74. PRU-ICSS UART Protocol Formats

Transmit/Receive for 5-bit data, parity Enable, 1 STOP bit

		D0	D1	D2	D3	D4	PARITY	STOP1

Transmit/Receive for 6-bit data, parity Enable, 1 STOP bit

		D0	D1	D2	D3	D4	D5	PARITY	STOP1

Transmit/Receive for 7-bit data, parity Enable, 1 STOP bit

		D0	D1	D2	D3	D4	D5	D6	PARITY	STOP1

Transmit/Receive for 8-bit data, parity Enable, 1 STOP bit

		D0	D1	D2	D3	D4	D5	D6	D7	PARITY	STOP1

30.2.7.2.3.3.1 Frame Formatting

Character length is specified using the PRUSS_UART_LINE_CONTROL_REGISTER[1-0] WLS bit field (see Table 30-746).

The number of stop-bits is specified using the PRUSS_UART_LINE_CONTROL_REGISTER[2] STB bit (see Table 30-746).

The parity bit is programmed using the PRUSS_UART_LINE_CONTROL_REGISTER[5-3] PEN, EPS, and SP bits (see Table 30-745).

Table 30-745. Relationship Between ST, EPS, and PEN Bits in UART_LCR

ST Bit	EPS Bit	PEN Bit	Parity Option
x	x	0	Parity disabled: No PARITY bit is transmitted or checked.
0	0	1	Odd parity selected: Odd number of logic 1s.
0	1	1	Even parity selected: Even number of logic 1s.
1	0	1	Stick parity selected with PARITY bit transmitted and checked as set.
1	1	1	Stick parity selected with PARITY bit transmitted and checked as cleared.

Table 30-746. Number of STOP Bits Generated

STB Bit	WLS Bit	Word Length Selected with WLS Bits	Number of STOP Bits Generated	Baud Clock (BCLK) Cycles
0	x	Any word length	1	16
1	0h	5 bits	1.5	24
1	1h	6 bits	2	32
1	2h	7 bits	2	32
1	3h	8 bits	2	32

30.2.7.2.4 PRU-ICSS UART Clock Generation and Control

The PRUSS_UART0 bit clock is derived from an input clock to the PRUSS_UART0. See your device-specific data manual to check the maximum data rate supported by the PRUSS_UART0.

Figure 30-75 is a conceptual clock generation diagram for the PRUSS_UART0. The processor clock generator receives a signal from an external clock source and produces a PRUSS_UART0 input clock with a programmed frequency. The PRUSS_UART0 contains a programmable baud generator that takes an input clock and divides it by a divisor in the range between 1 and (2^{16} - 1) to produce a baud clock (BCLK). The frequency of BCLK is sixteen times (16×) the baud rate (each received or transmitted bit lasts 16 BCLK cycles) or thirteen times (13×) the baud rate (each received or transmitted bit lasts 13

BCLK cycles). When the PRUSS_UART0 is receiving, the bit is sampled in the 8th BCLK cycle for 16× over sampling mode and on the 6th BCLK cycle for 13× over-sampling mode. The 16× or 13× reference clock is selected by configuring the mode definition register (MDR) - PRUSS_UART_MODE_DEFINITION_REGISTER [0] OSM_SEL bit. The formula to calculate the divisor is:

$$\text{Divisor} = \frac{\text{UART input clock frequency}}{\text{Desired baud rate} \times 16} \quad [\text{MDR.OSM\_SEL} = 0]$$

pruss-013

$$\text{Divisor} = \frac{\text{UART input clock frequency}}{\text{Desired baud rate} \times 13} \quad [\text{MDR.OSM\_SEL} = 1]$$

pruss-014

Two 8-bit register fields:

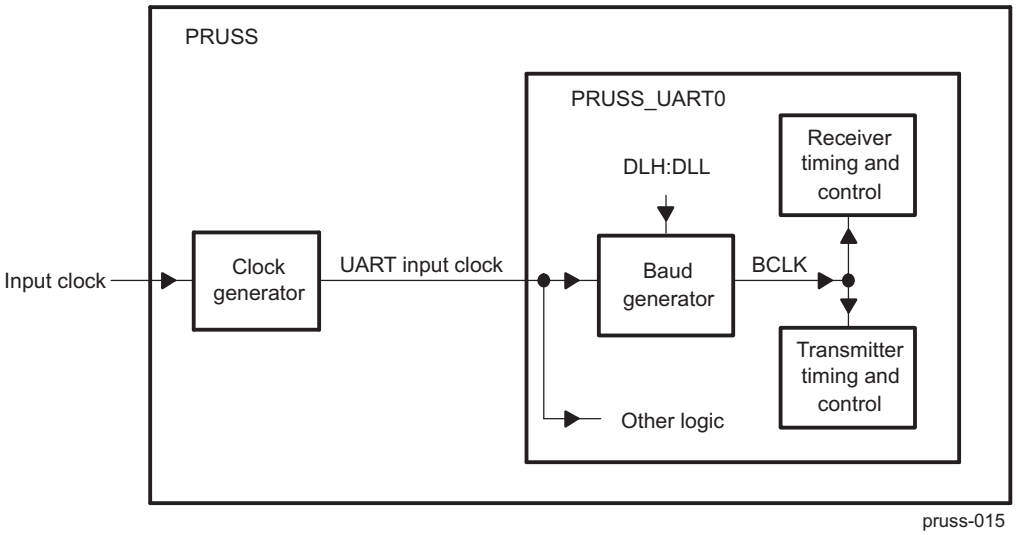
- PRUSS_UART_DIVISOR_REGISTER_MSB_[7:0] DLH
- PRUSS_UART_DIVISOR_REGISTER_LSB_[7:0] DLL,

called divisor latches, hold this 16-bit divisor. DLH holds the most significant bits of the divisor, and DLL holds the least significant bits of the divisor. For information about these register fields, see the PRUSS_UART0 register descriptions in the Section 30.2.7.4, *PRU-ICSS UART Register Manual*. These divisor latches must be loaded during initialization of the PRUSS_UART0 in order to ensure desired operation of the baud generator. Writing to the divisor latches results in two wait states being inserted during the write access while the baud generator is loaded with the new value.

Figure 30-76 summarizes the relationship between the transferred data bit, BCLK, and the PRUSS_UART0 input clock. Note that the timing relationship depicted in Figure 30-76 shows that each bit lasts for 16 BCLK cycles . This is in case of 16x over-sampling mode. For 13× over-sampling mode each bit lasts for 13 BCLK cycles .

Example baud rates and divisor values relative to a 150-MHz PRUSS_UART0 input clock and 16× over-sampling mode are shown in Table 30-747.

Figure 30-75. PRU-ICSS UART Clock Generation Diagram



pruss-015

Figure 30-76. Relationships Between PRU-ICSS UART Data Bit, BCLK, and Input Clock

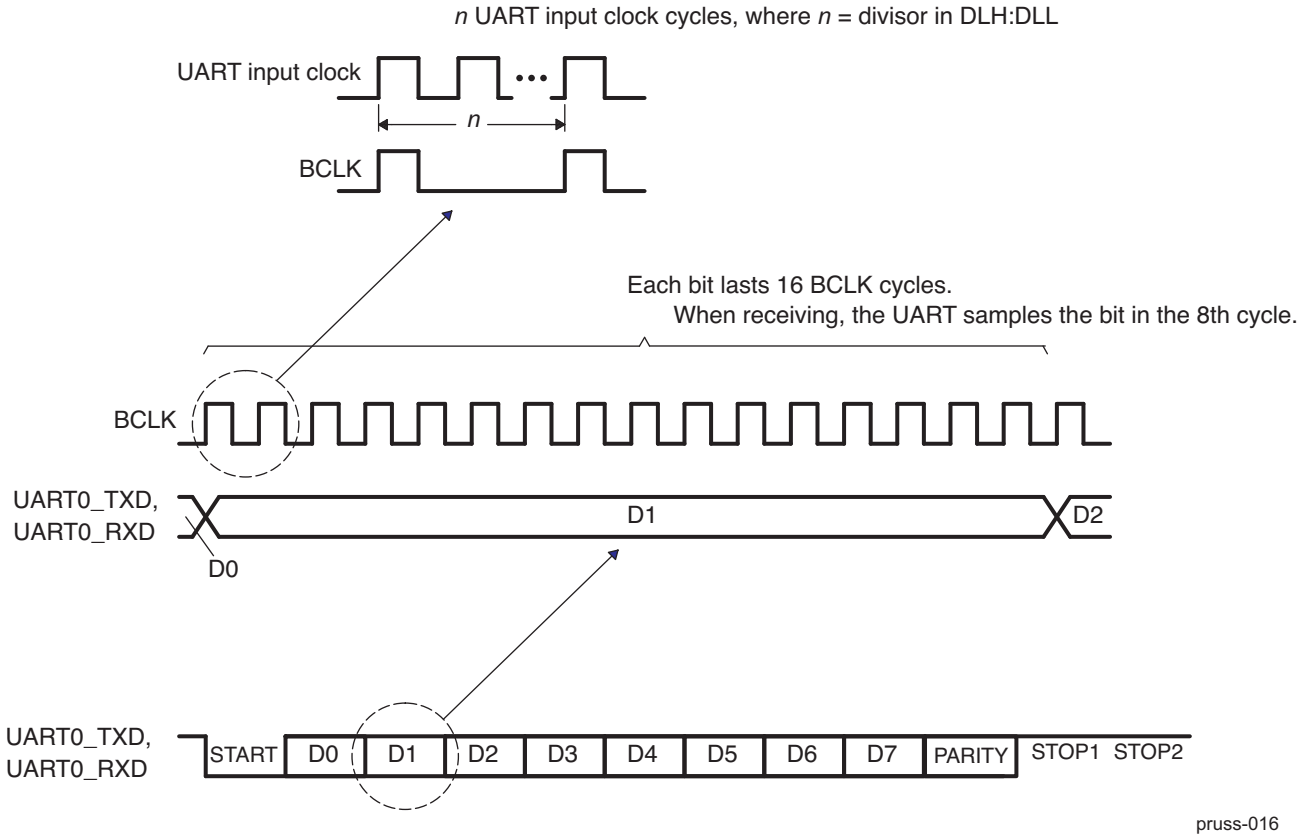


Table 30-747. Baud Rate Examples for 192-MHz PRU-ICSS UART Input Clock and 16× Over-sampling Mode

Baud Rate	Divisor Value	Actual Baud Rate	Error (%)
2400	5000	2400	0.00
4800	2500	4800	0.00
9600	1250	9600	0.00
19200	625	19200	0.00
38400	313	38338.658	-0.16
56000	214	56074.766	0.13
115200	104	115384.6	0.16
128000	94	127659.574	-0.27
3000000	4	3000000	0.00
6000000	2	6000000	0.00
12000000	1	12000000	0.00

Table 30-748. Baud Rate Examples for 192-MHz PRU-ICSS UART Input Clock and 13× Over-sampling Mode

Baud Rate	Divisor Value	Actual Baud Rate	Error (%)
2400	6154	2399.940	-0.0025
4800	3077	4799.880	-0.0025
9600	1538	9602.881	0.03

Table 30-748. Baud Rate Examples for 192-MHz PRU-ICSS UART Input Clock and 13× Over-sampling Mode (continued)

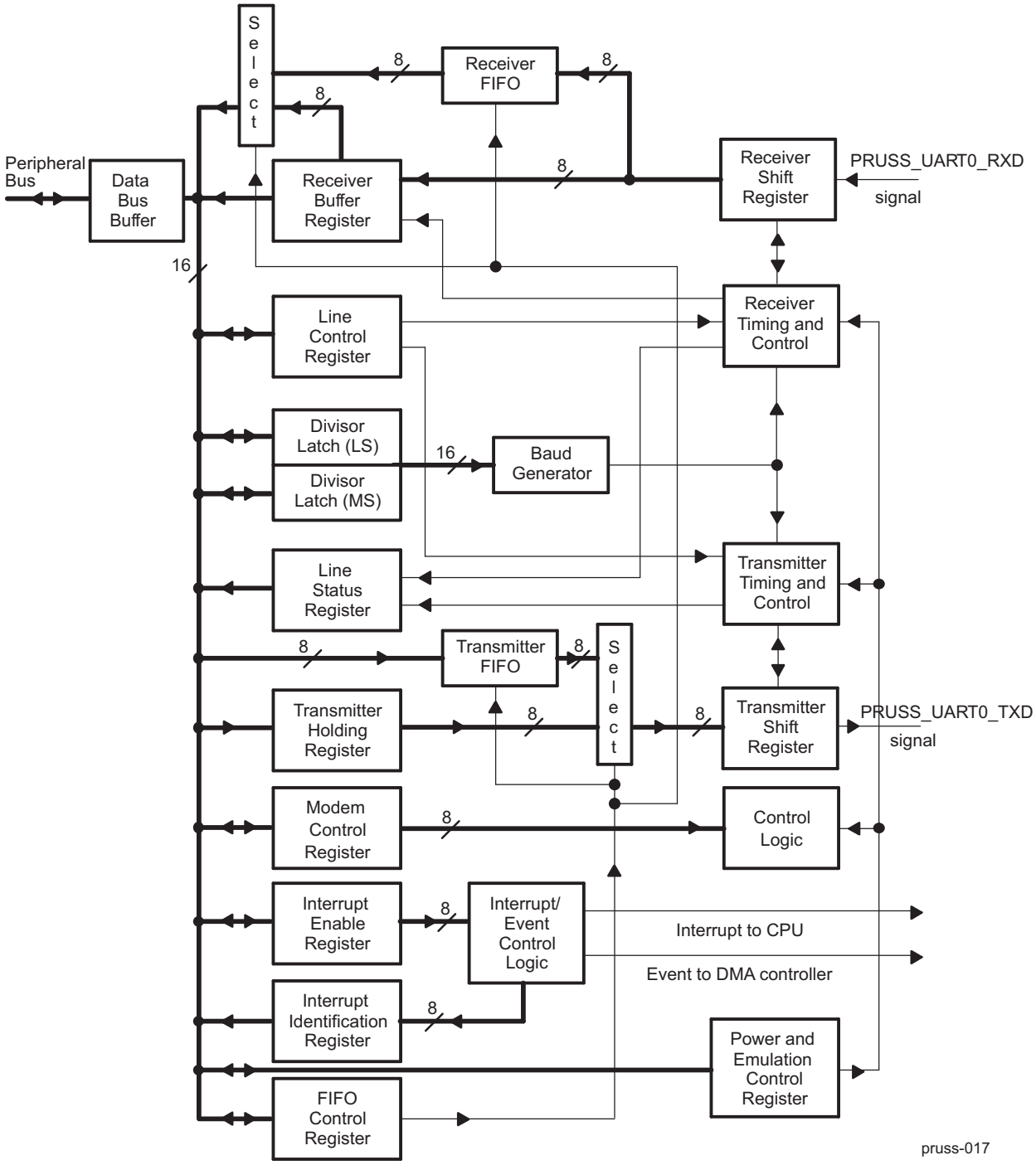
Baud Rate	Divisor Value	Actual Baud Rate	Error (%)
19200	769	19205.762	0.03
38400	385	38361.638	-0.10
56000	264	55944.056	-0.10
115200	128	115384.6	0.16
128000	115	128428.094	0.33

30.2.7.3 PRU-ICSS UART Module Functional Description

30.2.7.3.1 PRU-ICSS UART Functional Block Diagram

A functional block diagram of the PRUSS_UART0 is shown in Figure 30-77.

Figure 30-77. PRU-ICSS UART Block Diagram



NOTE: The value *n* indicates the applicable UART where there are multiple instances. For the PRU-ICSS, there is only one instance and all UART signals should reflect this (e.g., UART0_TXD instead of UARTn_TXD).

30.2.7.3.2 PRU-ICSS UART Reset Considerations

30.2.7.3.2.1 PRU-ICSS UART Software Reset Considerations

Two bits in the power and emulation management register - PRUSS_UART_POWERMANAGEMENT_AND_EMULATION_REGISTER, control resetting the parts of the PRUSS_UART0:

- The bit [14] UTRST controls resetting the transmitter only. If UTRST = 1, the transmitter is active; if UTRST = 0, the transmitter is in reset.
- The bit [13] URRST controls resetting the receiver only. If URRST = 1, the receiver is active; if URRST = 0, the receiver is in reset.

In each case, putting the receiver and/or transmitter in reset will reset the state machine of the affected portion but does not affect the PRUSS_UART0 registers.

30.2.7.3.2.2 PRU-ICSS UART Hardware Reset Considerations

When the processor RESET pin is asserted, the entire processor is reset and is held in the reset state until the RESET pin is released. As part of a device reset, the PRUSS_UART0 state machine is reset and the PRUSS_UART0 registers are forced to their default states. The default states of the registers are shown in Section 30.2.7.4.2.2.

30.2.7.3.3 PRU-ICSS UART Power Management

The PRUSS_UART0 peripheral can be placed in reduced-power modes to conserve power during periods of low activity. The power management of the PRUSS_UART0 peripheral and other PRU-ICSS peripherals is controlled by the device Power, Reset and Clock Manager (PRCM). For more details on the PRUSS_UART0 clock and power management, refer to the Section 30.2.4.2, *PRU-ICSS Power and Clock Management*.

30.2.7.3.4 PRU-ICSS UART Interrupt Support

30.2.7.3.4.1 PRU-ICSS UART Interrupt Events and Requests

The PRUSS_UART0 generates the interrupt requests described in Table 30-749. All requests are multiplexed through an arbiter to a single PRUSS_UART0 interrupt request to the CPU, as shown in Figure 30-78. Each of the interrupt requests has an enable bit in the interrupt enable register (IER) - PRUSS_UART_INTERRUPT_ENABLE_REGISTER and is recorded in INTID bitfield of PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER.

If an interrupt occurs and the corresponding enable bit is set to 1, the interrupt request is recorded in INTID bitfield and is forwarded to the CPU. If an interrupt occurs and the corresponding enable bit is cleared to 0, the interrupt request is blocked. The interrupt request is neither recorded in INTID, nor forwarded to the CPU.

30.2.7.3.4.2 PRU-ICSS UART Interrupt Multiplexing

The PRUSS_UART0 have dedicated interrupt signals to the CPU and the interrupts are not multiplexed with any other interrupt source.

Table 30-749. PRU-ICSS UART Interrupt Requests Descriptions

PRUSS_UART0 Interrupt Request	Interrupt Source	Comment
THREINT	THR-empty condition: The transmitter holding register (THR) or the transmitter FIFO is empty. All of the data has been copied from THR (i.e. PRUSS_UART_RBR_THR_REGISTERS [7:0] DATA) to the transmitter shift register (TSR).	If THREINT is enabled in PRUSS_UART_INTERRUPT_ENABLE_REGISTER, by setting the ETBEI bit, it is recorded in INTID bitfield. As an alternative to using THREINT, the CPU can poll the THRE bit in the line status register PRUSS_UART_LINE_STATUS_REGISTER.

Table 30-749. PRU-ICSS UART Interrupt Requests Descriptions (continued)

PRUSS_UART0 Interrupt Request	Interrupt Source	Comment
RDAINT	Receive data available in non-FIFO mode or trigger level reached in the FIFO mode.	If RDAINT is enabled in PRUSS_UART_INTERRUPT_ENABLE_REGISTER, by setting the ERBI bit, it is recorded in INTID bitfield. As an alternative to using RDAINT, the CPU can poll the DR bit in the line status register PRUSS_UART_LINE_STATUS_REGISTER. In the FIFO mode, this is not a functionally equivalent alternative because the DR bit does not respond to the FIFO trigger level. The DR bit only indicates the presence or absence of unread characters.
RTOINT	Receiver time-out condition (in the FIFO mode only): No characters have been removed from or input to the receiver FIFO during the last four character times (see Table 30-751), and there is at least one character in the receiver FIFO during this time.	The receiver time-out interrupt prevents the PRUSS_UART0 from waiting indefinitely, in the case when the receiver FIFO level is below the trigger level and thus does not generate a receiver data-ready interrupt. If RTOINT is enabled in PRUSS_UART_INTERRUPT_ENABLE_REGISTER, by setting the ERBI bit, it is recorded in INTID bitfield. There is no status bit to reflect the occurrence of a time-out condition.
RLSINT	Receiver line status condition: An overrun error, parity error, framing error, or break has occurred.	If RLSINT is enabled in PRUSS_UART_INTERRUPT_ENABLE_REGISTER, by setting the ELSI bit, it is recorded in INTID bitfield. As an alternative to using RLSINT, the CPU can poll the following bits in the line status register PRUSS_UART_LINE_STATUS_REGISTER: overrun error indicator (OE), parity error indicator (PE), framing error indicator (FE), and break indicator (BI).

Figure 30-78. PRU-ICSS UART Interrupt Request Enable Paths

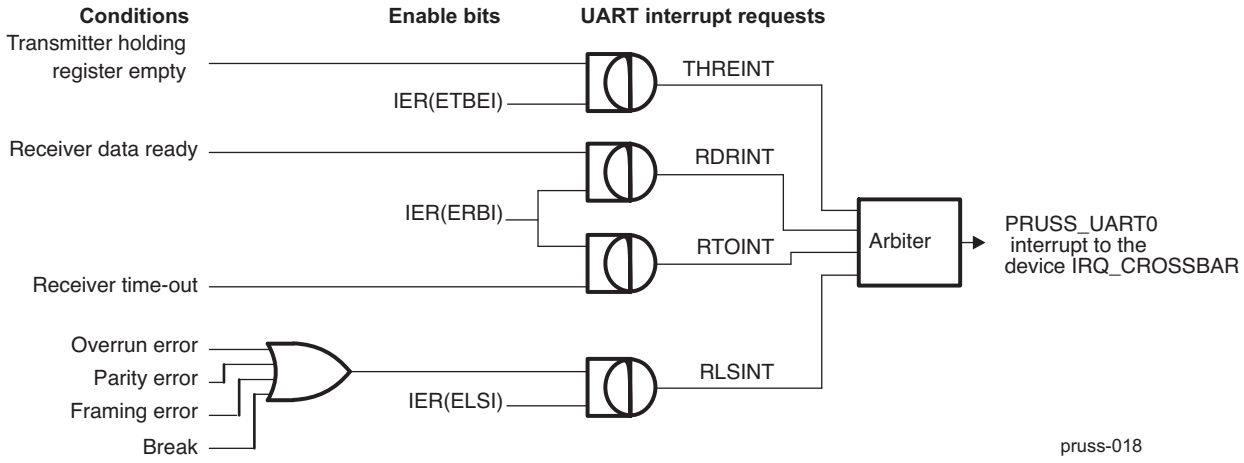


Table 30-750. Interrupt Identification and Interrupt Clearing Information

Priority Level	IIR Bits				Interrupt Type	Interrupt Source	Event That Clears Interrupt
	3	2	1	0			
None	0	0	0	1	None	None	None
1	0	1	1	0	Receiver line status	Overrun error, parity error, framing error, or break is detected.	For an overrun error, reading the PRUSS_UART_LINE_STATUS_REGISTER clears the interrupt. For a parity error, framing error, or break, the interrupt is cleared only after all the erroneous data have been read.
2	0	1	0	0	Receiver data-ready	Non-FIFO mode: Receiver data is ready.	Non-FIFO mode: The receiver buffer register (RBR) is read.
						FIFO mode: Trigger level reached. If four character times (see Table 30-751) pass with no access of the FIFO, the interrupt is asserted again.	FIFO mode: The FIFO drops below the trigger level. (1)
2	1	1	0	0	Receiver time-out	FIFO mode only: No characters have been removed from or input to the receiver FIFO during the last four character times (see Table 30-751), and there is at least one character in the receiver FIFO during this time.	One of the following events: • A character is read from the receiver FIFO (1) • A new character arrives in the receiver FIFO • The URRST bit in the power and emulation management register (PRUSS_UART_POWERMANAGEMENT_AND_EMULATION_REGISTER) is loaded with 0.
3	0	0	1	0	Transmitter holding register empty	Non-FIFO mode: Transmitter holding register (THR) is empty.	A character is written to the transmitter holding register (THR) or the interrupt identification register (IIR) is read.
						FIFO mode: Transmitter FIFO is empty.	

(1) In the FIFO mode, the receiver data-ready interrupt or receiver time-out interrupt is cleared by the CPU or by the DMA controller, whichever reads from the receiver FIFO first.

30.2.7.3.5 PRU-ICSS UART DMA Event Support

In the FIFO mode, the PRUSS_UART0 generates the following two DMA events:

- **Receive event (URXEVT):** The trigger level for the receiver FIFO (1, 4, 8, or 14 characters) is set with the FIFO control PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER [7:6] FIFOEN_RXFIFTL bitfield. Every time the trigger level is reached or a receiver time-out occurs, the PRUSS_UART0 sends a receive event to the EDMA controller. In response, the EDMA controller reads the data from the receiver FIFO by way of the receiver buffer register PRUSS_UART_RBR_THR_REGISTERS [7:0] DATA. Note that the receive event is not asserted if the data at the top of the receiver FIFO is erroneous even if the trigger level has been reached.
- **Transmit event (UTXEVT):** When the transmitter FIFO is empty (when the last byte in the transmitter FIFO has been copied to the transmitter shift register), the PRUSS_UART0 sends an UTXEVT signal to the EDMA controller. In response, the EDMA controller refills the transmitter FIFO by way of the transmitter holding register (THR) - PRUSS_UART_RBR_THR_REGISTERS [7:0] DATA. The UTXEVT signal is also sent to the DMA controller when the PRUSS_UART0 is taken out of reset using the UTRST bit in the power and emulation management register (PRUSS_UART_POWERMANAGEMENT_AND_EMULATION_REGISTER).

Activity in DMA channels can be synchronized to these events. In the non-FIFO mode, the PRUSS_UART0 generates no DMA events. Any DMA channel synchronized to either of these events must be enabled at the time the PRUSS_UART0 event is generated. Otherwise, the DMA channel will miss the event and, unless the PRUSS_UART0 generates a new event, no data transfer will occur.

30.2.7.3.6 PRU-ICSS UART Operations

30.2.7.3.6.1 PRU-ICSS UART Transmission

The PRUSS_UART0 transmitter section includes a transmitter hold register (THR) mapped in the PRUSS_UART_RBR_THR_REGISTERS [7:0] DATA bitfield and a transmitter shift register (TSR). When the PRUSS_UART0 is in the FIFO mode, THR is a 16-byte FIFO. Transmitter section control is a function of the PRUSS_UART0 line control register PRUSS_UART_LINE_CONTROL_REGISTER. Based on the settings chosen in this register, the PRUSS_UART0 transmitter sends the following to the receiving device:

- 1 START bit
- 5, 6, 7, or 8 data bits
- 1 PARITY bit (optional)
- 1, 1.5, or 2 STOP bits

THR receives data from the internal data bus, and when TSR is ready, the PRUSS_UART0 moves the data from THR to TSR. The PRUSS_UART0 serializes the data in TSR and transmits the data on the UART0_TXD pin.

In the non-FIFO mode, if THR is empty and the THR empty (THRE) interrupt is enabled in the interrupt enable register PRUSS_UART_INTERRUPT_ENABLE_REGISTER, an interrupt is generated. This interrupt is cleared when a character is loaded into THR or the interrupt identification register PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER bitfield INTID is read. In the FIFO mode, the interrupt is generated when the transmitter FIFO is empty, and it is cleared when at least one byte is loaded into the FIFO or INTID bitfield is read.

30.2.7.3.6.2 PRU-ICSS UART Reception

The PRUSS_UART0 receiver section includes a receiver shift register (RSR) and a receiver buffer register (RBR) mapped in PRUSS_UART_RBR_THR_REGISTERS [7:0] DATA bitfield. When the PRUSS_UART0 is in the FIFO mode, RBR is a 16-byte FIFO. Timing is supplied by the 16× receiver clock. Receiver section control is a function of the PRUSS_UART0 line control register PRUSS_UART_LINE_CONTROL_REGISTER. Based on the settings chosen in this register, the PRUSS_UART0 receiver accepts the following from the transmitting device:

- 1 START bit
- 5, 6, 7, or 8 data bits
- 1 PARITY bit (optional)
- 1 STOP bit (any other STOP bits transferred with the above data are not detected)

RSR receives the data bits from the UART0_RXD pin. Then RSR concatenates the data bits and moves the resulting value into RBR (or the receiver FIFO), accessible in the PRUSS_UART_RBR_THR_REGISTERS [7:0] DATA register bitfield. The PRUSS_UART0 also stores three bits of error status information next to each received character, to record a parity error, framing error, or break.

In the non-FIFO mode, when a character is placed in RBR and the receiver data-ready interrupt is enabled in the interrupt enable register - PRUSS_UART_INTERRUPT_ENABLE_REGISTER, an interrupt is generated. This interrupt is cleared when the character is read from RBR. In the FIFO mode, the interrupt is generated when the FIFO is filled to the trigger level selected in the FIFO control MSB part of the register PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER, and it is cleared when the FIFO contents drop below the trigger level.

30.2.7.3.6.3 PRU-ICSS UART FIFO Modes

The following two modes can be used for servicing the receiver and transmitter FIFOs:

- FIFO interrupt mode. The FIFO is enabled and the associated interrupts are enabled. Interrupts are sent to the CPU to indicate when specific events occur.
- FIFO poll mode. The FIFO is enabled but the associated interrupts are disabled. The CPU polls status bits to detect specific events.

Because the receiver FIFO and the transmitter FIFO are controlled separately, either one or both can be placed into the interrupt mode or the poll mode.

30.2.7.3.6.3.1 PRU-ICSS UART FIFO Interrupt Mode

When the receiver FIFO is enabled in the FIFO control register (FCR), mapped in the MSB part of the register PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER, and the receiver interrupts are enabled in the interrupt enable register PRUSS_UART_INTERRUPT_ENABLE_REGISTER, the interrupt mode is selected for the receiver FIFO. The following are important points about the receiver interrupts:

- The receiver data-ready interrupt is issued to the CPU when the FIFO has reached the trigger level that is programmed in FCR. It is cleared when the CPU or the DMA controller reads enough characters from the FIFO such that the FIFO drops below its programmed trigger level.
- The receiver line status interrupt is generated in response to an overrun error, a parity error, a framing error, or a break. This interrupt has higher priority than the receiver data-ready interrupt. For details, see Section 30.2.7.3.4.
- The data-ready (DR) bit in the line status register (LSR) - PRUSS_UART_LINE_STATUS_REGISTER, indicates the presence or absence of characters in the receiver FIFO. The DR bit is set when a character is transferred from the receiver shift register (RSR) to the empty receiver FIFO. The DR bit remains set until the FIFO is empty again.
- A receiver time-out interrupt occurs if all of the following conditions exist:
  - At least one character is in the FIFO,
  - The most recent character was received more than four continuous character times ago. A character time is the time allotted for 1 START bit, *n* data bits, 1 PARITY bit, and 1 STOP bit, where *n* depends on the word length selected with the WLS0 and WLS1 bits of the line control register PRUSS_UART_LINE_CONTROL_REGISTER. See Table 30-751.
  - The most recent read of the FIFO has occurred more than four continuous character times before.
- Character times are calculated by using the baud rate.
- When a receiver time-out interrupt has occurred, it is cleared and the time-out timer is cleared when the CPU or the EDMA controller reads one character from the receiver FIFO. The interrupt is also cleared if a new character is received in the FIFO or if the URRST bit is cleared in the power and emulation management register PRUSS_UART_POWERMANAGEMENT_AND_EMULATION_REGISTER.
- If a receiver time-out interrupt has not occurred, the time-out timer is cleared after a new character is received or after the CPU or EDMA reads the receiver FIFO.

When the transmitter FIFO is enabled in PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER [0] IPEND_FIFOEN bit and the transmitter holding register empty (THRE) interrupt is enabled in PRUSS_UART_INTERRUPT_ENABLE_REGISTER[1] ETBEI bit, the interrupt mode is selected for the transmitter FIFO. The THRE interrupt occurs when the transmitter FIFO is empty. It is cleared when the transmitter hold register (THR) PRUSS_UART_RBR_THR_REGISTERS [7:0] DATA bitfield is loaded (1 to 16 characters may be written to the transmitter FIFO while servicing this interrupt) or the interrupt identification register INTID bitfield is read in the PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER.

Table 30-751. Character Time for Word Lengths

Word Length (*n*)	Character Time	Four Character Times
5	Time for 8 bits	Time for 32 bits
6	Time for 9 bits	Time for 36 bits
7	Time for 10 bits	Time for 40 bits
8	Time for 11 bits	Time for 44 bits

30.2.7.3.6.3.2 PRU-ICSS UART FIFO Poll Mode

When the receiver FIFO is enabled in the FIFO control register (via setting the PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER [0] IPEND_FIFOEN to 0b1) and the receiver interrupts are disabled in the interrupt enable register (PRUSS_UART_INTERRUPT_ENABLE_REGISTER), the poll mode is selected for the receiver FIFO. Similarly, when the transmitter FIFO is enabled via setting the same bit (PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER [0] IPEND_FIFOEN to 0b1) and the transmitter interrupts are disabled, the transmitted FIFO is in the poll mode. In the poll mode, the CPU detects events by checking bits in the line status register - PRUSS_UART_LINE_STATUS_REGISTER:

- The PRUSS_UART_LINE_STATUS_REGISTER[7] RXFIFOE bit indicates whether there are any errors in the receiver FIFO.
- The PRUSS_UART_LINE_STATUS_REGISTER[6] TEMT bit indicates that both the transmitter holding register (THR) and the transmitter shift register (TSR) are empty.
- The PRUSS_UART_LINE_STATUS_REGISTER[5] THRE bit indicates when THR (mapped in the PRUSS_UART_RBR_THR_REGISTERS [7:0] DATA bitfield) is empty.
- The following PRUSS_UART_LINE_STATUS_REGISTER bits specify which error or errors have occurred:
  - PRUSS_UART_LINE_STATUS_REGISTER[4] BI - Break Interrupt
  - PRUSS_UART_LINE_STATUS_REGISTER[3] FE – Framing Error
  - PRUSS_UART_LINE_STATUS_REGISTER[2] PE – Parity Error
  - PRUSS_UART_LINE_STATUS_REGISTER[1] OE – Overrun Error
- The PRUSS_UART_LINE_STATUS_REGISTER[0] DR (data-ready) bit is set as long as there is at least one byte in the receiver FIFO.

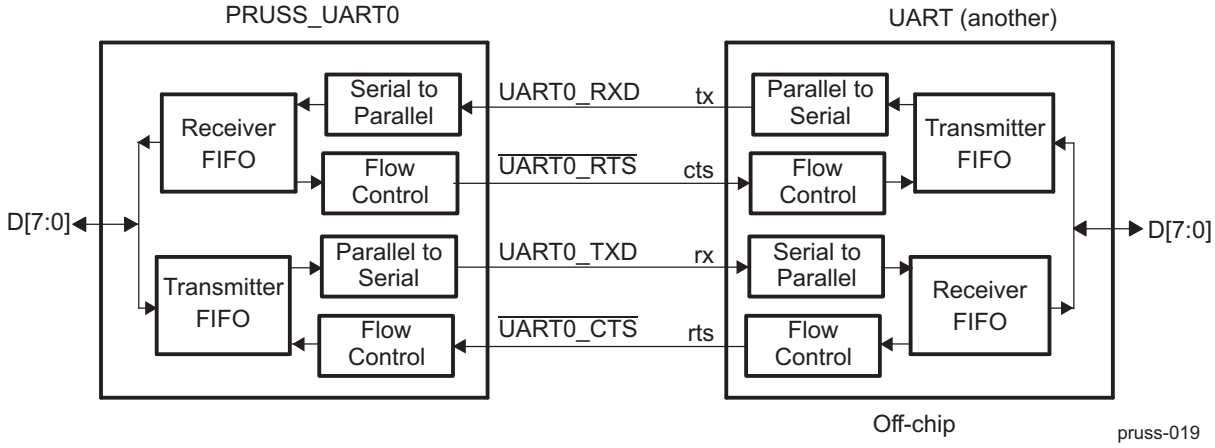
Also, in the FIFO poll mode:

- The interrupt identification register (INTID) bitfields are not affected by any events because the interrupts are disabled.
- The PRUSS_UART0 does not indicate when the receiver FIFO trigger level is reached or when a receiver time-out occurs.

30.2.7.3.6.4 PRU-ICSS UART Autoflow Control

The PRUSS_UART0 can employ autoflow control by connecting the $\overline{\text{PRUSS\_UART0\_CTS}}$ and $\overline{\text{PRUSS\_UART0\_RTS}}$ signals. The $\overline{\text{PRUSS\_UART0\_CTS}}$ input must be active before the transmitter FIFO can transmit data. The $\overline{\text{PRUSS\_UART0\_RTS}}$ becomes active when the receiver needs more data and notifies the sending device. When $\overline{\text{PRUSS\_UART0\_RTS}}$ is connected to $\overline{\text{PRUSS\_UART0\_CTS}}$, data transmission does not occur unless the receiver FIFO has space for the data. Therefore, when two UARTs are connected as shown in Figure 30-79 with autoflow enabled, overrun errors are eliminated.

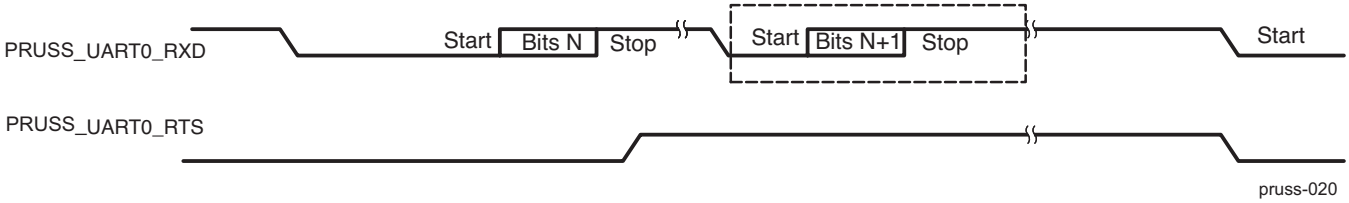
Figure 30-79. UART Interface Using Autoflow Diagram



30.2.7.3.6.4.1 PRU-ICSS UART Signal UART0_RTS Behavior

PRUSS_UART0_RTS data flow control originates in the receiver block (see Figure 30-77). When the receiver FIFO level reaches a trigger level of 1, 4, 8, or 14 (see Figure 30-80), PRUSS_UART0_RTS is deasserted. The sending UART may send an additional byte after the trigger level is reached (assuming the sending UART has another byte to send), because it may not recognize the deassertion of PRUSS_UART0_RTS until after it has begun sending the additional byte. For trigger level 1, 4, and 8, PRUSS_UART0_RTS is automatically reasserted once the receiver FIFO is emptied. For trigger level 14, PRUSS_UART0_RTS is automatically reasserted once the receiver FIFO drops below the trigger level.

Figure 30-80. Autoflow Functional Timing Waveforms for PRUSS_UART0_RTS



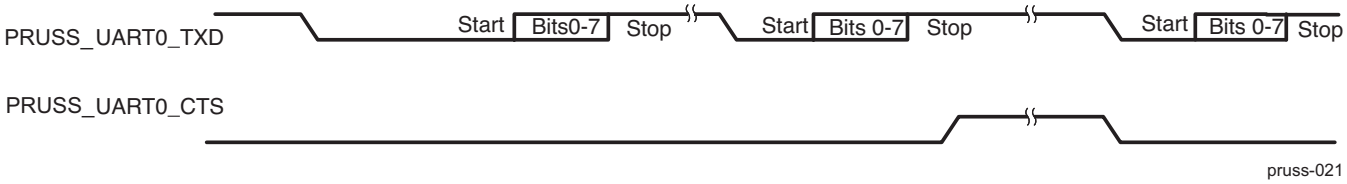
(1) N = Receiver FIFO trigger level.

(2) The two blocks in dashed lines cover the case where an additional byte is sent.

30.2.7.3.6.4.2 PRU-ICSS UART Signal PRUSS_UART0_CTS Behavior

The transmitter checks PRUSS_UART0_CTS before sending the next data byte. If PRUSS_UART0_CTS is active, the transmitter sends the next byte. To stop the transmitter from sending the following byte, PRUSS_UART0_CTS must be released before the middle of the last STOP bit that is currently being sent (see Figure 30-81). When flow control is enabled, PRUSS_UART0_CTS level changes do not trigger interrupts because the device automatically controls its own transmitter. Without autoflow control, the transmitter sends any data present in the transmitter FIFO and a receiver overrun error may result.

Figure 30-81. Autoflow Functional Timing Waveforms for PRUSS_UART0_CTS



(1) When PRUSS_UART0_CTS is active (low), the transmitter keeps sending serial data out.

(2) When PRUSS_UART0_CTS goes high before the middle of the last STOP bit of the current byte, the transmitter finishes sending the current byte but it does not send the next byte.

(3) When PRUSS_UART0_CTS goes from high to low, the transmitter begins sending data again.

30.2.7.3.6.5 PRU-ICSS UART Loopback Control

The PRUSS_UART0 can be placed in the diagnostic mode using the LOOP bit in the modem control register - PRUSS_UART_MODEM_CONTROL_REGISTER, which internally connects the PRUSS_UART0 output back to the PRUSS_UART0's input. In this mode, the transmit and receive data paths, the transmitter and receiver interrupts, and the modem control interrupts can be verified without connecting to another UART.

30.2.7.3.7 PRU-ICSS UART Initialization

The following steps are required to initialize the PRUSS_UART0:

1. Perform the necessary device pin multiplexing setup (see your device-specific data manual).
2. Set the desired baud rate by writing the appropriate clock divisor values to the divisor latch registers PRUSS_UART_DIVISOR_REGISTER_MSB_ [7:0] DLH and PRUSS_UART_DIVISOR_REGISTER_LSB_ [7:0] DLL.
3. If the FIFOs will be used, select the desired trigger level and enable the FIFOs by writing the appropriate values to the FIFO control register. The PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER[0] IPEND_FIFOEN bit must be set first, before the other bits in this register are configured.
4. Choose the desired protocol settings by writing the appropriate values to the line control register PRUSS_UART_LINE_CONTROL_REGISTER.
5. If autoflow control is desired, write appropriate values to the modem control register PRUSS_UART_MODEM_CONTROL_REGISTER.
6. Choose the desired response to emulation suspend events by configuring the FREE bit and enable the PRUSS_UART0 by setting the UTRST and URRST bits in the power and emulation management register -PRUSS_UART_POWERMANAGEMENT_AND_EMULATION_REGISTER.

30.2.7.3.8 PRU-ICSS UART Exception Processing

30.2.7.3.8.1 PRU-ICSS UART Divisor Latch Not Programmed

Since the processor reset signal has no effect on the divisor latch, the divisor latch will have an unknown value after power up. If the divisor latch is not programmed after power up, the baud clock (BCLK) will not operate and will instead be set to a constant logic 1 state.

The divisor latch values should always be reinitialized following a processor reset.

30.2.7.3.8.2 Changing Operating Mode During Busy Serial Communication of PRU-ICSS UART

Since the serial link characteristics are based on how the control registers are programmed, the PRUSS_UART0 will expect the control registers to be static while it is busy engaging in a serial communication. Therefore, changing the control registers while the module is still busy communicating with another serial device will most likely cause an error condition and should be avoided.

30.2.7.4 PRUSS_UART Register Manual

This section describes the PRUSS_UART module registers.

30.2.7.4.1 PRUSS_UART Instance Summary

Table 30-752. PRUSS_UART Instance Summary

Module Name	Base Address	Size
PRUSS1_UART	0x4B22 8000	56 Bytes
PRUSS2_UART	0x4B2A 8000	56 Bytes

30.2.7.4.2 PRUSS_UART Registers

30.2.7.4.2.1 PRUSS_UART Register Summary

Table 30-753. PRUSS1_UART Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS1_UART Physical Address
PRUSS_UART_RBR_THR_REGISTERS	W	32	0x0000 0000	0x4B22 8000
PRUSS_UART_INTERRUPT_ENABLE_REGISTER	RW	32	0x0000 0004	0x4B22 8004
PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER	W	32	0x0000 0008	0x4B22 8008
PRUSS_UART_LINE_CONTROL_REGISTER	RW	32	0x0000 000C	0x4B22 800C
PRUSS_UART_MODEM_CONTROL_REGISTER	RW	32	0x0000 0010	0x4B22 8010
PRUSS_UART_LINE_STATUS_REGISTER	R	32	0x0000 0014	0x4B22 8014
PRUSS_UART_MODEM_STATUS_REGISTER	R	32	0x0000 0018	0x4B22 8018
PRUSS_UART_SCRATCH_REGISTER	RW	32	0x0000 001C	0x4B22 801C
PRUSS_UART_DIVISOR_REGISTER_LSB_	RW	32	0x0000 0020	0x4B22 8020
PRUSS_UART_DIVISOR_REGISTER_MSB_	RW	32	0x0000 0024	0x4B22 8024
PRUSS_UART_PERIPHERAL_ID_REGISTER	R	32	0x0000 0028	0x4B22 8028
RESERVED	R	32	0x0000 002C	0x4B22 802C
PRUSS_UART_POWERMANAGEMENT_AND_EMULATION_REGISTER	RW	32	0x0000 0030	0x4B22 8030
PRUSS_UART_MODE_DEFINITION_REGISTER	RW	32	0x0000 0034	0x4B22 8034

Table 30-754. PRUSS2_UART Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS2_UART Physical Address
PRUSS_UART_RBR_THR_REGISTERS	W	32	0x0000 0000	0x4B2A 8000
PRUSS_UART_INTERRUPT_ENABLE_REGISTER	RW	32	0x0000 0004	0x4B2A 8004
PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER	W	32	0x0000 0008	0x4B2A 8008
PRUSS_UART_LINE_CONTROL_REGISTER	RW	32	0x0000 000C	0x4B2A 800C
PRUSS_UART_MODEM_CONTROL_REGISTER	RW	32	0x0000 0010	0x4B2A 8010
PRUSS_UART_LINE_STATUS_REGISTER	R	32	0x0000 0014	0x4B2A 8014
PRUSS_UART_MODEM_STATUS_REGISTER	R	32	0x0000 0018	0x4B2A 8018
PRUSS_UART_SCRATCH_REGISTER	RW	32	0x0000 001C	0x4B2A 801C
PRUSS_UART_DIVISOR_REGISTER_LSB_	RW	32	0x0000 0020	0x4B2A 8020
PRUSS_UART_DIVISOR_REGISTER_MSB_	RW	32	0x0000 0024	0x4B2A 8024

Table 30-754. PRUSS2_UART Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS2_UART Physical Address
PRUSS_UART_PERIPHERAL_ID_REGISTER	R	32	0x0000 0028	0x4B2A 8028
RESERVED	R	32	0x0000 002C	0x4B2A 802C
PRUSS_UART_POWERMANAGEMENT_AND_EMULATION_REGISTER	RW	32	0x0000 0030	0x4B2A 8030
PRUSS_UART_MODE_DEFINITION_REGISTER	RW	32	0x0000 0034	0x4B2A 8034

30.2.7.4.2.2 PRUSS_UART Register Description

Table 30-755. PRUSS_UART_RBR_THR_REGISTERS

Address Offset	0x0000 0000		
Physical Address	0x4B22 8000 0x4B2A 8000	**Instance**	PRUSS1_UART PRUSS2_UART
Description	In the non-FIFO mode, when a character is placed in Receiver buffer register and the receiver data-ready interrupt is enabled (DR = 1 in Interrupt identification register), an interrupt is generated. This interrupt is cleared when the character is read from Receiver buffer register. In the FIFO mode, the interrupt is generated when the FIFO is filled to the trigger level selected in the FIFO control register, and it is cleared when the FIFO contents drop below the trigger level. In the non-FIFO mode, if Transmitter holding register is empty and the THR empty (THRE) interrupt is enabled (ETBEI = 1 in Interrupt enable register), an interrupt is generated. This interrupt is cleared when a character is loaded into Transmitter holding register or the Interrupt identification register is read. In the FIFO mode, the interrupt is generated when the transmitter FIFO is empty, and it is cleared when at least one byte is loaded into the FIFO or Interrupt identification register is read.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DATA							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x0
7:0	DATA	**Read:** Read Receive Buffer Register **Write:** Write Transmitter Holding Register	RW	0x0

Table 30-756. Register Call Summary for Register PRUSS_UART_RBR_THR_REGISTERS

PRU-ICSS UART Module

- PRU-ICSS UART Transmission Protocol: [0]
- PRU-ICSS UART Reception Protocol: [1]
- PRU-ICSS UART Interrupt Multiplexing: [2]
- PRU-ICSS UART DMA Event Support: [3] [4]
- PRU-ICSS UART Transmission: [5]
- PRU-ICSS UART Reception: [6] [7]
- PRU-ICSS UART FIFO Modes: [8] [9]
- PRUSS_UART Register Summary: [10] [11]

Table 30-757. PRUSS_UART_INTERRUPT_ENABLE_REGISTER

Address Offset	0x0000 0004		
Physical Address	0x4B22 8004 0x4B2A 8004	**Instance**	PRUSS1_UART PRUSS2_UART

Table 30-757. PRUSS_UART_INTERRUPT_ENABLE_REGISTER (continued)

Description	The Interrupt enable register is used to individually enable or disable each type of interrupt request that can be generated by the UART. Each interrupt request that is enabled in Interrupt enable register is forwarded to the CPU.
Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												EDSSI	ELSI	ETBEI	ERBI

Bits	Field Name	Description	Type	Reset
31:4	RESERVED	Reserved	R	0x0
3	EDSSI	Enable Modem Status Interrupt	RW	0x0
2	ELSI	Receiver line status interrupt enable. 0x0: Receiver line status interrupt is disabled. 0x1: Receiver line status interrupt is enabled.	RW	0x0
1	ETBEI	Transmitter holding register empty interrupt enable. 0x0: Transmitter holding register empty interrupt is disabled. 0x1: Transmitter holding register empty interrupt is enabled.	RW	0x0
0	ERBI	Receiver data available interrupt and character timeout indication interrupt enable. 0x0: Receiver data available interrupt and character timeout indication interrupt is disabled. 0x1: Receiver data available interrupt and character timeout indication interrupt is enabled.	RW	0x0

Table 30-758. Register Call Summary for Register PRUSS_UART_INTERRUPT_ENABLE_REGISTER

PRU-ICSS UART Module

- PRU-ICSS UART Interrupt Events and Requests: [0]
- PRU-ICSS UART Interrupt Multiplexing: [1] [2] [3] [4]
- PRU-ICSS UART Transmission: [5]
- PRU-ICSS UART Reception: [6]
- PRU-ICSS UART FIFO Modes: [7] [8] [9]
- PRUSS_UART Register Summary: [10] [11]

Table 30-759. PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER

Address Offset	0x0000 0008		
Physical Address	0x4B22 8008 0x4B2A 8008	**Instance**	PRUSS1_UART PRUSS2_UART
Description	The Interrupt identification register is a read-only register at the same address as the FIFO control register, which is a write-only register. When an interrupt is generated and enabled in the Interrupt enable register, Interrupt identification register indicates that an interrupt is pending in the IPEND bit and encodes the type of interrupt in the INTID bits. Reading Interrupt identification register clears any THR empty (THRE) interrupts that are pending. The FIFOEN bit in Interrupt identification register can be checked to determine whether the UART is in the FIFO mode or the non-FIFO mode. Use FIFO control register to enable and clear the FIFOs and to select the receiver FIFO trigger level. The FIFOEN bit in FIFO control register must be set to 1 before other FIFO control register bits are written to or the FIFO control register bits are not programmed.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								FIFOEN_RXFIFTL		RESERVED		INTID			IPEND_FIFOEN

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x0
7:6	FIFOEN_RXFIFTL	**Read:** FIFOs enabled. 0x0: Non-FIFO mode 0x1-0x2: Reserved 0x3: FIFOs are enabled. FIFOEN bit in the FIFO control register (FCR) is set to 1. **Write:** Receiver FIFO trigger level. RXFIFTL sets the trigger level for the receiver FIFO. When the trigger level is reached, a receiver data-ready interrupt is generated (if the interrupt request is enabled). Once the FIFO drops below the trigger level, the interrupt is cleared. 0x0: 1 byte 0x1: 4 bytes 0x2: 8 bytes 0x3: 14 bytes	RW	0x0
5:4	RESERVED	Reserved	R	0x0
3:1	INTID	**Read:** Interrupt type. See Table 30-750. 0x0: Reserved 0x1: Transmitter holding register empty (priority 3) 0x2: Receiver data available (priority 2) 0x3: Receiver line status (priority 1, highest) 0x4-0x5: Reserved 0x6: Character timeout indication (priority 2) 0x7: Reserved **Write:** **Bit 3: DMAMODE1:** DMA MODE1 enable if FIFOs are enabled. Always write 1 to DMAMODE1. After a hardware reset, change DMAMODE1 from 0 to 1. DMAMODE1 = 1 is a requirement for proper communication between the UART and the EDMA controller. 0x0: DMA MODE1 is disabled. 0x1: DMA MODE1 is enabled. **Bit 2: TXCLR:** Transmitter FIFO clear. Write a 1 to TXCLR to clear the bit. 0x0: No effect. 0x1: Clears transmitter FIFO and resets the transmitter FIFO counter. The shift register is not cleared. **Bit 1: RXCLR:** Receiver FIFO clear. Write a 1 to RXCLR to clear the bit. 0x0: No effect. 0x1: Clears receiver FIFO and resets the receiver FIFO counter. The shift register is not cleared.	RW	0x0

Bits	Field Name	Description	Type	Reset
0	IPEND_FIFOEN	**Read:** Interrupt pending. When any UART interrupt is generated and is enabled in IER, IPEND is forced to 0. IPEND remains 0 until all pending interrupts are cleared or until a hardware reset occurs. If no interrupts are enabled, IPEND is never forced to 0. 0x0: Interrupts pending. 0x1: No interrupts pending. **Write:** Transmitter and receiver FIFOs mode enable. FIFOEN must be set before other FCR bits are written to or the FCR bits are not programmed. Clearing this bit clears the FIFO counters. 0x0: Non-FIFO mode. The transmitter and receiver FIFOs are disabled, and the FIFO pointers are cleared. 0x1: FIFO mode. The transmitter and receiver FIFOs are enabled.	RW	0x1

Table 30-760. Register Call Summary for Register PRUSS_UART_INTERRUPT_IDENTIFICATION_REGISTER_FIFO_CONTROL_REGISTER

PRU-ICSS UART Module

- PRU-ICSS UART Interrupt Events and Requests: [0]
- PRU-ICSS UART Interrupt Multiplexing:
- PRU-ICSS UART DMA Event Support: [2]
- PRU-ICSS UART Transmission: [3]
- PRU-ICSS UART Reception: [4]
- PRU-ICSS UART FIFO Modes: [5] [6] [7] [8] [9]
- PRU-ICSS UART Initialization: [10]
- PRUSS_UART Register Summary: [11] [12]

Table 30-761. PRUSS_UART_LINE_CONTROL_REGISTER

Address Offset	0x0000 000C		
Physical Address	0x4B22 800C 0x4B2A 800C	**Instance**	PRUSS1_UART PRUSS2_UART
Description	The system programmer controls the format of the asynchronous data communication exchange by using Line control register. In addition, the programmer can retrieve, inspect, and modify the content of line control register; this eliminates the need for separate storage of the line characteristics in system memory.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DLAB	BC	SP	EPS	PEN	STB	WLS1	WLS0

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x0
7	DLAB	Divisor latch access bit. The divisor latch registers (DLL and DLH) can be accessed at dedicated addresses or at addresses shared by RBR, THR, and IER. Using the shared addresses requires toggling DLAB to change which registers are selected. If you use the dedicated addresses, you can keep DLAB = 0. 0x0: Allows access to the receiver buffer register (RBR), the transmitter holding register (THR), and the interrupt enable register (IER) selected. At the address shared by RBR, THR, and DLL, the CPU can read from RBR and write to THR. At the address shared by IER and DLH, the CPU can read from and write to IER. 0x1: Allows access to the divisor latches of the baud generator during a read or write operation (DLL and DLH). At the address shared by RBR, THR, and DLL, the CPU can read from and write to DLL. At the address shared by IER and DLH, the CPU can read from and write to DLH.	RW	0x0
6	BC	Break control. 0x0: Break condition is disabled. 0x1: Break condition is transmitted to the receiving UART. A break condition is a condition where the UARTn_TXD signal is forced to the spacing (cleared) state.	RW	0x0
5	SP	Stick parity. The SP bit works in conjunction with the EPS and PEN bits. The relationship between the SP, EPS, and PEN bits is summarized in Table 30-745. 0x0: Stick parity is disabled. 0x1: Stick parity is enabled. • When odd parity is selected (EPS = 0), the PARITY bit is transmitted and checked as set. • When even parity is selected (EPS = 1), the PARITY bit is transmitted and checked as cleared.	RW	0x0
4	EPS	Even parity select. Selects the parity when parity is enabled (PEN = 1). The EPS bit works in conjunction with the SP and PEN bits. The relationship between the SP, EPS, and PEN bits is summarized in Table 30-745. 0x0: Odd parity is selected (an odd number of logic 1s is transmitted or checked in the data and PARITY bits). 0x1: Even parity is selected (an even number of logic 1s is transmitted or checked in the data and PARITY bits).	RW	0x0
3	PEN	Parity enable. The PEN bit works in conjunction with the SP and EPS bits. The relationship between the SP, EPS, and PEN bits is summarized in Table 30-745. 0x0: No PARITY bit is transmitted or checked. 0x1: Parity bit is generated in transmitted data and is checked in received data between the last data word bit and the first STOP bit.	RW	0x0
2	STB	Number of STOP bits generated. STB specifies 1, 1.5, or 2 STOP bits in each transmitted character. When STB = 1, the WLS bit determines the number of STOP bits. The receiver clocks only the first STOP bit, regardless of the number of STOP bits selected. The number of STOP bits generated is summarized in Table 30-746. 0x0: 1 STOP bit is generated. 0x1: WLS bit determines the number of STOP bits: • When WLS = 0, 1.5 STOP bits are generated. • When WLS = 1h, 2h, or 3h, 2 STOP bits are generated.	RW	0x0

Bits	Field Name	Description	Type	Reset
1-0	WLS	Word length select. Number of bits in each transmitted or received serial character. When STB = 1, the WLS bit determines the number of STOP bits. 0x0: 5 bits 0x1: 6 bits 0x2: 7 bits 0x3: 8 bits	RW	0x0

Table 30-762. Register Call Summary for Register PRUSS_UART_LINE_CONTROL_REGISTER

PRU-ICSS UART Module

- PRU-ICSS UART Transmission Protocol: [0]
- PRU-ICSS UART Reception Protocol: [1]
- PRU-ICSS UART Data Format: [2] [3] [4]
- PRU-ICSS UART Transmission: [5]
- PRU-ICSS UART Reception: [6]
- PRU-ICSS UART FIFO Modes: [7]
- PRU-ICSS UART Initialization: [8]
- PRUSS_UART Register Summary: [9] [10]

Table 30-763. PRUSS_UART_MODEM_CONTROL_REGISTER

Address Offset	0x0000 0010		
Physical Address	0x4B22 8010 0x4B2A 8010	**Instance**	PRUSS1_UART PRUSS2_UART
Description	The Modem control register provides the ability to enable/disable the autoflow functions, and enable/disable the loopback function for diagnostic purposes.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																										AFE	LOOP	OUT2	OUT1	RTS	RESERVED

Bits	Field Name	Description	Type	Reset
31:6	RESERVED	Reserved	R	0x0
5	AFE	Autoflow control enable. Autoflow control allows the $\overline{\text{UARTn\_RTS}}$ and $\overline{\text{UARTn\_CTS}}$ signals to provide handshaking between UARTs during data transfer. When AFE = 1, the RTS bit determines the autoflow control enabled. Note that all UARTs do not support this feature, see your device-specific data manual for supported features. If this feature is not available, this bit is reserved in this device and should be cleared to 0. 0x0: Autoflow control is disabled. 0x1:Autoflow control is enabled: • When RTS = 0, $\overline{\text{UARTn\_CTS}}$ is only enabled. • When RTS = 1, $\overline{\text{UARTn\_RTS}}$ and $\overline{\text{UARTn\_CTS}}$ are enabled.	RW	0x0

Bits	Field Name	Description	Type	Reset
4	LOOP	Loop back mode enable. LOOP is used for the diagnostic testing using the loop back feature. 0x0: Loop back mode is disabled. 0x1: Loop back mode is enabled. When LOOP is set, the following occur: • The UARTn_TXD signal is set high. • The UARTn_RXD pin is disconnected. • The output of the transmitter shift register (TSR) is lopped back in to the receiver shift register (RSR) input.	RW	0x0
3	OUT2	OUT2 Control Bit	RW	0x0
2	OUT1	OUT1 Control Bit	RW	0x0
1	RTS	RTS control. When AFE = 1, the RTS bit determines the autoflow control enabled. Note that all UARTs do not support this feature, see your device-specific data manual for supported features. If this feature is not available, this bit is reserved in this device and should be cleared to 0. 0x0: $\overline{\text{UARTn\_RTS}}$ is disabled, $\overline{\text{UARTn\_CTS}}$ is only enabled. 0x1: $\overline{\text{UARTn\_RTS}}$ and $\overline{\text{UARTn\_CTS}}$ are enabled.	RW	0x0
0	RESERVED	Reserved	R	0

Table 30-764. Register Call Summary for Register PRUSS_UART_MODEM_CONTROL_REGISTER

PRU-ICSS UART Module

- PRU-ICSS UART Loopback Control: [0]
- PRU-ICSS UART Initialization: [1]
- PRUSS_UART Register Summary: [2] [3]

Table 30-765. PRUSS_UART_LINE_STATUS_REGISTER

Address Offset	0x0000 0014		
Physical Address	0x4B22 8014 0x4B2A 8014	**Instance**	PRUSS1_UART PRUSS2_UART
Description	The Line status register provides information to the CPU concerning the status of data transfers. Line status register is intended for read operations only; do not write to this register. Bits 1 through 4 record the error conditions that produce a receiver line status interrupt.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								RXFIFOE	TEMT	THRE	BI	FE	PE	OE	DR

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x0
7	RXFIFOE	Receiver FIFO error. **In non-FIFO mode:** 0x0: There has been no error, or RXFIFOE was cleared because the CPU read the erroneous character from the receiver buffer register (RBR). 0x1: There is a parity error, framing error, or break indicator in the receiver buffer register (RBR). **In FIFO mode:** 0x0: There has been no error, or RXFIFOE was cleared because the CPU read the erroneous character from the receiver FIFO and there are no more errors in the receiver FIFO. 0x1: At least one parity error, framing error, or break indicator in the receiver FIFO.	R	0x0
6	TEMT	Transmitter empty (TEMT) indicator. **In non-FIFO mode:** 0x0: Either the transmitter holding register (THR) or the transmitter shift register (TSR) contains a data character. 0x1: Both the transmitter holding register (THR) and the transmitter shift register (TSR) are empty. **In FIFO mode:** 0x0: Either the transmitter FIFO or the transmitter shift register (TSR) contains a data character. 0x1: Both the transmitter FIFO and the transmitter shift register (TSR) are empty.	R	0x1
5	THRE	Transmitter holding register empty (THRE) indicator. If the THRE bit is set and the corresponding interrupt enable bit is set (ETBEI = 1 in IER), an interrupt request is generated. **In non-FIFO mode:** 0x0: Transmitter holding register (THR) is not empty. THR has been loaded by the CPU. 0x1: Transmitter holding register (THR) is empty (ready to accept a new character). The content of THR has been transferred to the transmitter shift register (TSR). **In FIFO mode:** 0x0: Transmitter FIFO is not empty. At least one character has been written to the transmitter FIFO. You can write to the transmitter FIFO if it is not full. 0x1: Transmitter FIFO is empty. The last character in the FIFO has been transferred to the transmitter shift register (TSR).	R	0x1

Bits	Field Name	Description	Type	Reset
4	BI	Break indicator. The BI bit is set whenever the receive data input (UARTn_RXD) was held low for longer than a full-word transmission time. A full-word transmission time is defined as the total time to transmit the START, data, PARITY, and STOP bits. If the BI bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated. **In non-FIFO mode:** 0x0: No break has been detected, or the BI bit was cleared because the CPU read the erroneous character from the receiver buffer register (RBR). 0x1: A break has been detected with the character in the receiver buffer register (RBR). **In FIFO mode:** 0x0: No break has been detected, or the BI bit was cleared because the CPU read the erroneous character from the receiver FIFO and the next character to be read from the FIFO has no break indicator. 0x1: A break has been detected with the character at the top of the receiver FIFO.	R	0x0
3	FE	Framing error (FE) indicator. A framing error occurs when the received character does not have a valid STOP bit. In response to a framing error, the UART sets the FE bit and waits until the signal on the RX pin goes high. Once the RX signal goes high, the receiver is ready to detect a new START bit and receive new data. If the FE bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated. **In non-FIFO mode:** 0x0: No framing error has been detected, or the FE bit was cleared because the CPU read the erroneous data from the receiver buffer register (RBR). 0x1: A framing error has been detected with the character in the receiver buffer register (RBR). **In FIFO mode:** 0x0: No framing error has been detected, or the FE bit was cleared because the CPU read the erroneous data from the receiver FIFO and the next character to be read from the FIFO has no framing error. 0x1: A framing error has been detected with the character at the top of the receiver FIFO.	R	0x0
2	PE	Parity error (PE) indicator. A parity error occurs when the parity of the received character does not match the parity selected with the EPS bit in the line control register (LCR). If the PE bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated. **In non-FIFO mode:** 0x0: No parity error has been detected, or the PE bit was cleared because the CPU read the erroneous data from the receiver buffer register (RBR). 0x1: A parity error has been detected with the character in the receiver buffer register (RBR). **In FIFO mode:** 0x0: No parity error has been detected, or the PE bit was cleared because the CPU read the erroneous data from the receiver FIFO and the next character to be read from the FIFO has no parity error. 0x1: A parity error has been detected with the character at the top of the receiver FIFO.	R	0x0

Bits	Field Name	Description	Type	Reset
1	OE	Overrun error (OE) indicator. An overrun error in the non-FIFO mode is different from an overrun error in the FIFO mode. If the OE bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated. **In non-FIFO mode:** 0x0: No overrun error has been detected, or the OE bit was cleared because the CPU read the content of the line status register (LSR). 0x1: Overrun error has been detected. Before the character in the receiver buffer register (RBR) could be read, it was overwritten by the next character arriving in RBR. **In FIFO mode:** 0x0: No overrun error has been detected, or the OE bit was cleared because the CPU read the content of the line status register (LSR). 0x1: Overrun error has been detected. If data continues to fill the FIFO beyond the trigger level, an overrun error occurs only after the FIFO is full and the next character has been completely received in the shift register. An overrun error is indicated to the CPU as soon as it happens. The new character overwrites the character in the shift register, but it is not transferred to the FIFO.	R	0x0
0	DR	Data-ready (DR) indicator for the receiver. If the DR bit is set and the corresponding interrupt enable bit is set (ERBI = 1 in IER), an interrupt request is generated. **In non-FIFO mode:** 0x0: Data is not ready, or the DR bit was cleared because the character was read from the receiver buffer register (RBR). 0x1: Data is ready. A complete incoming character has been received and transferred into the receiver buffer register (RBR). **In FIFO mode:** 0x0: Data is not ready, or the DR bit was cleared because all of the characters in the receiver FIFO have been read. 0x1: Data is ready. There is at least one unread character in the receiver FIFO. If the FIFO is empty, the DR bit is set as soon as a complete incoming character has been received and transferred into the FIFO. The DR bit remains set until the FIFO is empty again.	R	0x0

Table 30-766. Register Call Summary for Register PRUSS_UART_LINE_STATUS_REGISTER

PRU-ICSS UART Module

- PRU-ICSS UART Interrupt Multiplexing: [0] [1] [2] [3]
- PRU-ICSS UART FIFO Modes: [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14]
- PRUSS_UART Register Summary: [15] [16]

Table 30-767. PRUSS_UART_MODEM_STATUS_REGISTER

Address Offset	0x0000 0018		
Physical Address	0x4B22 8018 0x4B2A 8018	**Instance**	PRUSS1_UART PRUSS2_UART
Description	The Modem status register provides information to the CPU concerning the status of modem control signals. Modem status register is intended for read operations only; do not write to this register.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CD	RI	DSR	CTS	DCD	TERI	DDSR	DCTS

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x0
7	CD	Complement of the Carrier Detect input. When the UART is in the diagnostic test mode (loopback mode MCR[4] = 1), this bit is equal to the MCR bit 3 (OUT2).	R	0x0
6	RI	Complement of the Ring Indicator input. When the UART is in the diagnostic test mode (loopback mode MCR[4] = 1), this bit is equal to the MCR bit 2 (OUT1).	R	0x0
5	DSR	Complement of the Data Set Ready input. When the UART is in the diagnostic test mode (loopback mode MCR[4] = 1), this bit is equal to the MCR bit 0 (DTR).	R	0x0
4	CTS	Complement of the Clear To Send input. When the UART is in the diagnostic test mode (loopback mode MCR[4] = 1), this bit is equal to the MCR bit 1 (RTS).	R	0x0
3	DCD	Change in DCD indicator bit. DCD indicates that the DCD input has changed state since the last time it was read by the CPU. When DCD is set and the modem status interrupt is enabled, a modem status interrupt is generated.	R	0x0
2	TERI	Trailing edge of RI (TERI) indicator bit. TERI indicates that the RI input has changed from a low to a high. When TERI is set and the modem status interrupt is enabled, a modem status interrupt is generated.	R	0x0
1	DDSR	Change in DSR indicator bit. DDSR indicates that the DSR input has changed state since the last time it was read by the CPU. When DDSR is set and the modem status interrupt is enabled, a modem status interrupt is generated.	R	0x0
0	DCTS	Change in CTS indicator bit. DCTS indicates that the CTS input has changed state since the last time it was read by the CPU. When DCTS is set (autoflow control is not enabled and the modem status interrupt is enabled), a modem status interrupt is generated. When autoflow control is enabled, no interrupt is generated.	R	0x0

Table 30-768. Register Call Summary for Register PRUSS_UART_MODEM_STATUS_REGISTER

PRU-ICSS UART Module

- PRUSS_UART Register Summary: [0] [1]

Table 30-769. PRUSS_UART_SCRATCH_REGISTER

Address Offset	0x0000 001C		
Physical Address	0x4B22 801C 0x4B2A 801C	**Instance**	PRUSS1_UART PRUSS2_UART
Description	The Scratch Pad register is intended for programmer's use as a scratch pad. It temporarily holds the programmer's data without affecting UART operation.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								SCR							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x0
7:0	SCR	These bits are intended for the programmer's use as a scratch pad in the sense that it temporarily holds the programmer's data without affecting any other UART operation.	R	0x0

Table 30-770. Register Call Summary for Register PRUSS_UART_SCRATCH_REGISTER

PRU-ICSS UART Module

- PRUSS_UART Register Summary: [0] [1]

Table 30-771. PRUSS_UART_DIVISOR_REGISTER_LSB_

Address Offset	0x0000 0020		
Physical Address	0x4B22 8020 0x4B2A 8020	**Instance**	PRUSS1_UART PRUSS2_UART
Description	Two 8-bit register fields (DLL and DLH), called divisor latches, store the 16-bit divisor for generation of the baud clock in the baud generator. DLH holds the most-significant bits of the divisor, and DLL holds the least-significant bits of the divisor. These divisor latches must be loaded during initialization of the UART in order to ensure desired operation of the baud generator. Writing to the divisor latches results in two wait states being inserted during the write access while the baud generator is loaded with the new value.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DLL							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x0
7:0	DLL	The 8 least-significant bits (LSBs) of the 16-bit divisor for generation of the baud clock in the baud rate generator.	RW	0x0

Table 30-772. Register Call Summary for Register PRUSS_UART_DIVISOR_REGISTER_LSB_

PRU-ICSS UART Module

- PRU-ICSS UART Clock Generation and Control: [0]
- PRU-ICSS UART Initialization: [1]
- PRUSS_UART Register Summary: [2] [3]

Table 30-773. PRUSS_UART_DIVISOR_REGISTER_MSB_

Address Offset	0x0000 0024		
Physical Address	0x4B22 8024 0x4B2A 8024	**Instance**	PRUSS1_UART PRUSS2_UART
Description	Two 8-bit register fields (DLL and DLH), called divisor latches, store the 16-bit divisor for generation of the baud clock in the baud generator. DLH holds the most-significant bits of the divisor, and DLL holds the least-significant bits of the divisor. These divisor latches must be loaded during initialization of the UART in order to ensure desired operation of the baud generator. Writing to the divisor latches results in two wait states being inserted during the write access while the baud generator is loaded with the new value.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DLH							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x0
7:0	DLH	The 8 most-significant bits (MSBs) of the 16-bit divisor for generation of the baud clock in the baud rate generator.	RW	0x0

Table 30-774. Register Call Summary for Register PRUSS_UART_DIVISOR_REGISTER_MSB_

PRU-ICSS UART Module
- PRU-ICSS UART Clock Generation and Control: [0]
- PRU-ICSS UART Initialization: [1]
- PRUSS_UART Register Summary: [2] [3]

Table 30-775. PRUSS_UART_PERIPHERAL_ID_REGISTER

Address Offset	0x0000 0028		
Physical Address	0x4B22 8028 0x4B2A 8028	**Instance**	PRUSS1_UART PRUSS2_UART
Description	Peripheral Identification register		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PID																															

Bits	Field Name	Description	Type	Reset
31:0	PID		R	0x44141102

Table 30-776. Register Call Summary for Register PRUSS_UART_PERIPHERAL_ID_REGISTER

PRU-ICSS UART Module
- PRUSS_UART Register Summary: [0] [1]

Table 30-777. PRUSS_UART_POWERMANAGEMENT_AND_EMULATION_REGISTER

Address Offset	0x0000 0030		
Physical Address	0x4B22 8030 0x4B2A 8030	**Instance**	PRUSS1_UART PRUSS2_UART
Description	Power and emulation management register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																RESERVED	UTRST	URRST	RESERVED											FREE	

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Reserved	R	0x0000
15	RESERVED	Reserved. This bit must always be written with a 0.	RW	0x0
14	UTRST	UART transmitter reset. Resets and enables the transmitter. 0x0: Transmitter is disabled and in reset state. 0x1: Transmitter is enabled.	RW	0x0

Bits	Field Name	Description	Type	Reset
13	URRST	UART receiver reset. Resets and enables the receiver. 0x0: Receiver is disabled and in reset state. 0x1: Receiver is enabled.	RW	0x0
12:1	RESERVED	Reserved	R	0x000
0	FREE	Free-running enable mode bit. This bit determines the emulation mode functionality of the UART. When halted, the UART can handle register read/write requests, but does not generate any transmission/reception, interrupts or events. 0x0: If a transmission is not in progress, the UART halts immediately. If a transmission is in progress, the UART halts after completion of the one-word transmission. 0x1: Free-running mode is enabled; UART continues to run normally.	RW	0x0

Table 30-778. Register Call Summary for Register PRUSS_UART_POWERMANAGEMENT_AND_EMULATION_REGISTER

PRU-ICSS UART Module

- PRU-ICSS UART Software Reset Considerations: [0]
- PRU-ICSS UART Interrupt Multiplexing: [1]
- PRU-ICSS UART DMA Event Support: [2]
- PRU-ICSS UART FIFO Modes: [3]
- PRU-ICSS UART Initialization: [4]
- PRUSS_UART Register Summary: [5] [6]

Table 30-779. PRUSS_UART_MODE_DEFINITION_REGISTER

Address Offset	0x0000 0034		
Physical Address	0x4B22 8034 0x4B2A 8034	**Instance**	PRUSS1_UART PRUSS2_UART
Description	The Mode definition register determines the over-sampling mode for the UART.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															OSM_SEL

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved	R	0x0
0	OSM_SEL	Over-Sampling Mode Select. 0x0: 16× over-sampling. 0x1: 13× over-sampling.	RW	0x0

Table 30-780. Register Call Summary for Register PRUSS_UART_MODE_DEFINITION_REGISTER

PRU-ICSS UART Module

- PRU-ICSS UART Clock Generation and Control: [0]
- PRUSS_UART Register Summary: [1] [2]

30.2.8 PRU-ICSS eCAP Module

30.2.8.1 PRU-ICSS eCAP Functional Description

A single instance of an **enhanced capture** event module is integrated in the device PRU-ICSS1 subsystem - PRUSS1_eCAP_0 and PRU-ICSS2 subsystem - PRUSS2_eCAP_0.

For more details on the PRUSS1_eCAP_0 and PRUSS2_eCAP_0 I/O signals available at device level, refer to the Section 30.2.2. For PRUSS1_eCAP_0 and PRUSS2_eCAP_0 integration details and functionalities controlled at PRU-ICSS top level (functional clock control, etc.), refer to the Section 30.2.3 and the Section 30.2.4 .

NOTE: The PRUSS1_eCAP_0 and PRUSS2_eCAP_0 "SYNCIn" hardware event synchronization input and "SYNCOut" hardware synchronization output are NOT implemented in the device PRU-ICSS1 and PRU-ICSS2, respectively. However, a software-forced synchronization via bit PRUSS_ECAP_ECCTL2[8] SWSYNC, can be used as an alternative, provided that PRUSS_ECAP_ECCTL2[5] SYNCI_EN bit is set to 0b1.

For full description of the PRUSS1_eCAP_0 and PRUSS2_eCAP_0 modules functionalities, refer to the Section 29.3, *Enhanced Capture (eCAP) Module* of the Chapter 29, *Pulse-Width Modulation Subsystem*.

30.2.8.2 PRUSS_ECAP Register Manual

This section describes the registers of the PRUSS_eCAP_0 module.

30.2.8.2.1 PRUSS_ECAP Instance Summary

Table 30-781. PRUSS_ECAP Instance Summary

Module Name	Module Base Address L3_MAIN	Size
PRUSS1_ECAP	0x4B23 0000	96 Bytes
PRUSS2_ECAP	0x4B2B 0000	96 Bytes

30.2.8.2.2 PRUSS_ECAP Registers

30.2.8.2.2.1 PRUSS_ECAP Register Summary

Table 30-782. PRUSS1_ECAP Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS1_ECAP Physical Address
PRUSS_ECAP_TSCNT	RW	32	0x0000 0000	0x4B23 0000
PRUSS_ECAP_CNTPHS	RW	32	0x0000 0004	0x4B23 0004
PRUSS_ECAP_CAP1	RW	32	0x0000 0008	0x4B23 0008
PRUSS_ECAP_CAP2	RW	32	0x0000 000C	0x4B23 000C
PRUSS_ECAP_CAP3	RW	32	0x0000 0010	0x4B23 0010
PRUSS_ECAP_CAP4	RW	32	0x0000 0014	0x4B23 0014
PRUSS_ECAP_ECCTL1	RW	16	0x0000 0028	0x4B23 0028
PRUSS_ECAP_ECCTL2	RW	16	0x0000 002A	0x4B23 002A
PRUSS_ECAP_ECEINT	RW	16	0x0000 002C	0x4B23 002C
PRUSS_ECAP_ECFLG	R	16	0x0000 002E	0x4B23 002E
PRUSS_ECAP_ECCLR	RW	16	0x0000 0030	0x4B23 0030
PRUSS_ECAP_ECFRC	RW	16	0x0000 0034	0x4B23 0034
PRUSS_ECAP_PID	R	32	0x0000 005C	0x4B23 005C

Table 30-783. PRUSS2_ECAP Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS2_ECAP Physical Address
PRUSS_ECAP_TSCNT	RW	32	0x0000 0000	0x4B2B 0000
PRUSS_ECAP_CNTPHS	RW	32	0x0000 0004	0x4B2B 0004
PRUSS_ECAP_CAP1	RW	32	0x0000 0008	0x4B2B 0008
PRUSS_ECAP_CAP2	RW	32	0x0000 000C	0x4B2B 000C
PRUSS_ECAP_CAP3	RW	32	0x0000 0010	0x4B2B 0010
PRUSS_ECAP_CAP4	RW	32	0x0000 0014	0x4B2B 0014
PRUSS_ECAP_ECCTL1	RW	16	0x0000 0028	0x4B2B 0028
PRUSS_ECAP_ECCTL2	RW	16	0x0000 002A	0x4B2B 002A
PRUSS_ECAP_ECEINT	RW	16	0x0000 002C	0x4B2B 002C
PRUSS_ECAP_ECFLG	R	16	0x0000 002E	0x4B2B 002E
PRUSS_ECAP_ECCLR	RW	16	0x0000 0030	0x4B2B 0030
PRUSS_ECAP_ECFRC	RW	16	0x0000 0034	0x4B2B 0034
PRUSS_ECAP_PID	R	32	0x0000 005C	0x4B2B 005C

30.2.8.2.2.2 PRUSS_ECAP Register Description

Table 30-784. PRUSS_ECAP_TSCNT

Address Offset	0x0000 0000		
Physical Address	0x4B23 0000 0x4B2B 0000	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	Time Stamp Counter Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSCNT																															

Bits	Field Name	Description	Type	Reset
31:0	TSCNT	Active 32 bit-counter register that is used as the capture time-base	RW	0x0

Table 30-785. Register Call Summary for Register PRUSS_ECAP_TSCNT

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [1] [2]
- PRUSS_ECAP Register Description: [3]

Table 30-786. PRUSS_ECAP_CNTPHS

Address Offset	0x0000 0004		
Physical Address	0x4B23 0004 0x4B2B 0004	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	Counter Phase Control Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTPHS																															

Bits	Field Name	Description	Type	Reset
31:0	CNTPHS	Counter phase value register that can be programmed for phase lag/lead. This register shadows TSCNT and is loaded into PRUSS_ECAP_TSCNT upon either a SYNCI event or S/W force via a control bit. Used to achieve phase control synchronization with respect to other eCAP and EPWM time-bases.	RW	0x0

Table 30-787. Register Call Summary for Register PRUSS_ECAP_CNTPHS

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [1] [2]
- PRUSS_ECAP Register Description: [3]

Table 30-788. PRUSS_ECAP_CAP1

Address Offset	0x0000 0008		
Physical Address	0x4B23 0008 0x4B2B 0008	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	Capture-1 Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAP1																															

Bits	Field Name	Description	Type	Reset
31:0	CAP1	This register can be loaded (written) by the following. (a) Time-Stamp (that is, counter value) during a capture event. (b) Software may be useful for test purposes. (c) APRD active register when used in APWM mode.	RW	0x0

Table 30-789. Register Call Summary for Register PRUSS_ECAP_CAP1

PRU-ICSS eCAP Module

- PRU-ICSS eCAP Functional Description:
- PRUSS_ECAP Register Summary: [17] [18]
- PRUSS_ECAP Register Description: [19] [20] [21] [22] [23] [24] [25]

Table 30-790. PRUSS_ECAP_CAP2

Address Offset	0x0000 000C		
Physical Address	0x4B23 000C 0x4B2B 000C	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	Capture-2 Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAP2																															

Bits	Field Name	Description	Type	Reset
31:0	CAP2	This register can be loaded (written) by the following. (a) Time- Stamp (that is, counter value) during a capture event. (b) Software may be useful for test purposes. (c) APRD active register when used in APWM mode.	RW	0x0

Table 30-791. Register Call Summary for Register PRUSS_ECAP_CAP2

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [11] [12]
- PRUSS_ECAP Register Description: [13] [14]

Table 30-792. PRUSS_ECAP_CAP3

Address Offset	0x0000 0010		
Physical Address	0x4B23 0010 0x4B2B 0010	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	Capture-3 Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAP3																															

Bits	Field Name	Description	Type	Reset
31:0	CAP3	In CMP mode, this is a time-stamp capture register. In APWM mode, this is the period shadow (APRD) register. User software updates the PWM period value through this register. In this mode, CAP3 shadows CAP1.	RW	0x0

Table 30-793. Register Call Summary for Register PRUSS_ECAP_CAP3

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [10] [11]

Table 30-794. PRUSS_ECAP_CAP4

Address Offset	0x0000 0014		
Physical Address	0x4B23 0014 0x4B2B 0014	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	Capture-4 Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAP4																															

Bits	Field Name	Description	Type	Reset
31:0	CAP4	In CMP mode, this is a time-stamp capture register. In APWM mode, this is the compare shadow (ACMP) register. User software updates the PWM compare value through this register. In this mode, CAP4 shadows CAP2.	RW	0x0

Table 30-795. Register Call Summary for Register PRUSS_ECAP_CAP4

PRU-ICSS eCAP Module

- PRU-ICSS eCAP Functional Description:
- PRUSS_ECAP Register Summary: [13] [14]
- PRUSS_ECAP Register Description: [15] [16] [17] [18] [19]

Table 30-796. PRUSS_ECAP_ECCTL1

Address Offset	0x0000 0028		
Physical Address	0x4B23 0028 0x4B2B 0028	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	ECAP Control Register1		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FREE_SOFT		EVTFLTPS					CAPLDEN	CTRRST4	CAP4POL	CTRRST3	CAP3POL	CTRRST2	CAP2POL	CTRRST1	CAP1POL

Bits	Field Name	Description	Type	Reset
15:14	FREE_SOFT	Emulation Control 0x0 = TSCNT counter stops immediately on emulation suspend. 0x1 = TSCNT counter runs until = 0. 0x2 = TSCNT counter is unaffected by emulation suspend (Run Free). 0x3 = TSCNT counter is unaffected by emulation suspend (Run Free).	RW	0x0
13:9	EVTFLTPS	Event Filter prescale select: 0x0 = Divide by 1 (i.e,. no prescale, by-pass the prescaler) 0x1 = Divide by 2 0x2 = Divide by 4 0x3 = Divide by 6 0x4 = Divide by 8 0x5 = Divide by 10 0x1E = Divide by 60 0x1F = Divide by 62	RW	0x0
8	CAPLDEN	Enable Loading of PRUSS_ECAP_CAP1 to PRUSS_ECAP_CAP4 registers on a capture event 0x0 = Disable PRUSS_ECAP_CAP1-PRUSS_ECAP_CAP4 register loads at capture event time. 0x1 = Enable PRUSS_ECAP_CAP1-PRUSS_ECAP_CAP4 register loads at capture event time.	RW	0x0
7	CTRRST4	Counter Reset on Capture Event 4 0x0 = Do not reset counter on Capture Event 4 (absolute time stamp operation) 0x1 = Reset counter after Capture Event 4 time-stamp has been captured (used in difference mode operation)	RW	0x0
6	CAP4POL	Capture Event 4 Polarity select 0x0 = Capture Event 4 triggered on a rising edge (RE) 0x1 = Capture Event 4 triggered on a falling edge (FE)	RW	0x0
5	CTRRST3	Counter Reset on Capture Event 3 0x0 = Do not reset counter on Capture Event 3 (absolute time stamp) 0x1 = Reset counter after Event 3 time-stamp has been captured (used in difference mode operation)	RW	0x0
4	CAP3POL	Capture Event 3 Polarity select 0x0 = Capture Event 3 triggered on a rising edge (RE) 0x1 = Capture Event 3 triggered on a falling edge (FE)	RW	0x0
3	CTRRST2	Counter Reset on Capture Event 2 0x0 = Do not reset counter on Capture Event 2 (absolute time stamp) 0x1 = Reset counter after Event 2 time-stamp has been captured (used in difference mode operation)	RW	0x0

Bits	Field Name	Description	Type	Reset
2	CAP2POL	Capture Event 2 Polarity select 0x0 = Capture Event 2 triggered on a rising edge (RE) 0x1 = Capture Event 2 triggered on a falling edge (FE)	RW	0x0
1	CTRRST1	Counter Reset on Capture Event 1 0x0 = Do not reset counter on Capture Event 1 (absolute time stamp) 0x1 = Reset counter after Event 1 time-stamp has been captured (used in difference mode operation)	RW	0x0
0	CAP1POL	Capture Event 1 Polarity select 0x0 = Capture Event 1 triggered on a rising edge (RE) 0x1 = Capture Event 1 triggered on a falling edge (FE)	RW	0x0

Table 30-797. Register Call Summary for Register PRUSS_ECAP_ECCTL1

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [2] [3]

Table 30-798. PRUSS_ECAP_ECCTL2

Address Offset	0x0000 002A		
Physical Address	0x4B23 002A 0x4B2B 002A	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	ECAP Control Register 2		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					APWMPOL	CAPAPWM	SWSYNC	SYNCO_SEL		SYNCI_EN	TSCNTSTP	REARMRESET	STOPVALUE		CONTONESHT

Bits	Field Name	Description	Type	Reset
15:11	RESERVED		R	0x0
10	APWMPOL	APWM output polarity select. This is applicable only in APWM operating mode 0x0 = Output is active high (Compare value defines high time) 0x1 = Output is active low (Compare value defines low time)	RW	0x0

Bits	Field Name	Description	Type	Reset
9	CAPAPWM	CAP/APWM operating mode select 0x0 = ECAP module operates in capture mode. This mode forces the following configuration. (a) Inhibits TSCNT resets via CTR = PRD event. (b) Inhibits shadow loads on PRUSS_ECAP_CAP1 and PRUSS_ECAP_CAP2 registers. (c) Permits user to enable PRUSS_ECAP_CAP1-PRUSS_ECAP_CAP4 register load. (d) ECAP input/APWM output pin operates as a capture input. 0x1 = ECAP module operates in APWM mode. This mode forces the following configuration. (a) Resets TSCNT on CTR = PRD event (period boundary). (b) Permits shadow loading on PRUSS_ECAP_CAP1 and PRUSS_ECAP_CAP2 registers. (c) Disables loading of time-stamps into PRUSS_ECAP_CAP1 - PRUSS_ECAP_CAP4 registers. (d) ECAP input/APWM ouput pin operates as a APWM output.	RW	0x0
8	SWSYNC	Software-forced Counter (TSCNT) Synchronizing. This provides a convenient software method to synchronize some or all ECAP time bases. In APWM mode, the synchronizing can also be done via the CTR = PRD event. Note: Selection CTR = PRD is meaningful only in APWM mode. However, you can choose it in CAP mode if you find doing so useful. 0x0 = Writing a zero has no effect. Reading always returns a zero 0x1 = Writing a one forces a TSCNT shadow load of current ECAP module and any ECAP modules down-stream providing the SYNCO_SEL bits are 1'b00. After writing a 1, this bit returns to a zero.	RW	0x0
7:6	SYNCO_SEL	Sync-Out Select 0x0 = Select sync-in event to be the sync-out signal (pass through) 0x1 = Select CTR = PRD event to be the sync-out signal 0x2 = Disable sync out signal 0x3 = Disable sync out signal	RW	0x0
5	SYNCI_EN	Counter (TSCNT) Sync-In select mode 0x0 = Disable sync-in option 0x1 = Enable counter (TSCNT) to be loaded from PRUSS_ECAP_CNTPHS register upon either a SYNCI signal or a S/W force event.	RW	0x0
4	TSCNTSTP	Time Stamp (TSCNT) Counter Stop (freeze) Control 0x0 = TSCNT stopped 0x1 = TSCNT free-running	RW	0x0
3	REARMRESET	One-Shot Re-Arming Control, that is, wait for stop trigger. Note: The re-arm function is valid in one shot or continuous mode. 0x0 = Has no effect (reading always returns a 0) 0x1 = Arms the one-shot sequence as follows: 1) Resets the Mod4 counter to zero. 2) Unfreezes the Mod4 counter. 3) Enables capture register loads.	RW	0x0

Bits	Field Name	Description	Type	Reset
2:1	STOPVALUE	Stop value for one-shot mode. This is the number (between 1 and 4) of captures allowed to occur before the CAP (1 through 4) registers are frozen, that is, capture sequence is stopped. Wrap value for continuous mode. This is the number (between 1 and 4) of the capture register in which the circular buffer wraps around and starts again. Notes: STOPVALUE is compared to Mod4 counter and, when equal, the following two actions occur. (1) Mod4 counter is stopped (frozen). (2) Capture register loads are inhibited. In one-shot mode, further interrupt events are blocked until re-armed. 0x0 = Stop after Capture Event 1 in one-shot mode. Wrap after Capture Event 1 in continuous mode. 0x1 = Stop after Capture Event 2 in one-shot mode. Wrap after Capture Event 2 in continuous mode. 0x2 = Stop after Capture Event 3 in one-shot mode. Wrap after Capture Event 3 in continuous mode. 0x3 = Stop after Capture Event 4 in one-shot mode. Wrap after Capture Event 4 in continuous mode.	RW	0x3
0	CONTONESHT	Continuous or one-shot mode control (applicable only in capture mode) 0x0 = Operate in continuous mode 0x1 = Operate in one-shot mode	RW	0x0

Table 30-799. Register Call Summary for Register PRUSS_ECAP_ECCTL2

PRU-ICSS eCAP Module

- PRU-ICSS eCAP Functional Description: [0] [1]
- PRUSS_ECAP Register Summary: [7] [8]

Table 30-800. PRUSS_ECAP_ECEINT

Address Offset	0x0000 002C		
Physical Address	0x4B23 002C 0x4B2B 002C	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	ECAP Interrupt Enable Register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CMPEQ	PRDEQ	CNTOVF	CEVT4	CEVT3	CEVT2	CEVT1	RESERVED

Bits	Field Name	Description	Type	Reset
15:8	RESERVED		R	0x0
7	CMPEQ	Counter Equal **Compare** Interrupt Enable. 0x0 = Disable Compare Equal as an Interrupt source. 0x1 = Enable Compare Equal as an Interrupt source.	RW	0x0
6	PRDEQ	Counter Equal **Period** Interrupt Enable. 0x0 = Disable Period Equal as an Interrupt source. 0x1 = Enable Period Equal as an Interrupt source.	RW	0x0
5	CNTOVF	Counter Overflow Interrupt Enable. 0x0 = Disable counter Overflow as an Interrupt source. 0x1 = Enable counter Overflow as an Interrupt source.	RW	0x0

Bits	Field Name	Description	Type	Reset
4	CEVT4	Capture Event 4 Interrupt Enable. 0x0 = Disable Capture Event 4 as an Interrupt source. 0x1 = Enable Capture Event 4 as an Interrupt source.	RW	0x0
3	CEVT3	Capture Event 3 Interrupt Enable. 0x0 = Disable Capture Event 3 as an Interrupt source. 0x1 = Enable Capture Event 3 as an Interrupt source.	RW	0x0
2	CEVT2	Capture Event 2 Interrupt Enable. 0x0 = Disable Capture Event 2 as an Interrupt source. 0x1 = Enable Capture Event 2 as an Interrupt source.	RW	0x0
1	CEVT1	Capture Event 1 Interrupt Enable . 0x0 = Disable Capture Event 1 as an Interrupt source. 0x1 = Enable Capture Event 1 as an Interrupt source.	RW	0x0
0	RESERVED		R	0x0

Table 30-801. Register Call Summary for Register PRUSS_ECAP_ECEINT

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [2] [3]

Table 30-802. PRUSS_ECAP_ECFLG

Address Offset	0x0000 002E		
Physical Address	0x4B23 002E 0x4B2B 002E	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	ECAP Interrupt Flag Register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CMPEQ	PRDEQ	CNTOVF	CEVT4	CEVT3	CEVT2	CEVT1	INT

Bits	Field Name	Description	Type	Reset
15:8	RESERVED		R	0x0
7	CMPEQ	Compare Equal Compare Status Flag. This flag is only active in APWM mode. 0x0 = Indicates no event occurred 0x1 = Indicates the counter (TSCNT) reached the compare register value (ACMP)	R	0x0
6	PRDEQ	Counter Equal Period Status Flag. This flag is only active in APWM mode. 0x0 = Indicates no event occurred 0x1 = Indicates the counter (TSCNT) reached the period register value (APRD) and was reset.	R	0x0
5	CNTOVF	Counter Overflow Status Flag. This flag is active in CAP and APWM mode. 0x0 = Indicates no event occurred. 0x1 = Indicates the counter (TSCNT) has made the transition from 0xFFFFFFFF to 0x00000000	R	0x0
4	CEVT4	Capture Event 4 Status Flag This flag is only active in CAP mode. 0x0 = Indicates no event occurred 0x1 = Indicates the fourth event occurred at ECAPn pin	R	0x0

Bits	Field Name	Description	Type	Reset
3	CEVT3	Capture Event 3 Status Flag. This flag is active only in CAP mode. 0x0 = Indicates no event occurred. 0x1 = Indicates the third event occurred at ECAPn pin.	R	0x0
2	CEVT2	Capture Event 2 Status Flag. This flag is only active in CAP mode. 0x0 = Indicates no event occurred. 0x1 = Indicates the second event occurred at ECAPn pin.	R	0x0
1	CEVT1	Capture Event 1 Status Flag. This flag is only active in CAP mode. 0x0 = Indicates no event occurred. 0x1 = Indicates the first event occurred at ECAPn pin.	R	0x0
0	INT	Global Interrupt Status Flag 0x0 = Indicates no interrupt generated. 0x1 = Indicates that an interrupt was generated.	R	0x0

Table 30-803. Register Call Summary for Register PRUSS_ECAP_ECFLG

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [2] [3]

Table 30-804. PRUSS_ECAP_ECCLR

Address Offset	0x0000 0030		
Physical Address	0x4B23 0030 0x4B2B 0030	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	ECAP Interrupt Clear Register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CMPEQ	PRDEQ	CNTOVF	CEVT4	CEVT3	CEVT2	CEVT1	INT

Bits	Field Name	Description	Type	Reset
15:8	RESERVED		R	0x0
7	CMPEQ	Counter Equal Compare Status Flag 0x0 = Writing a 0 has no effect. Always reads back a 0 0x1 = Writing a 1 clears the CTR=CMP flag condition	RW	0x0
6	PRDEQ	Counter Equal Period Status Flag 0x0 = Writing a 0 has no effect. Always reads back a 0 0x1 = Writing a 1 clears the CTR=PRD flag condition	RW	0x0
5	CNTOVF	Counter Overflow Status Flag 0x0 = Writing a 0 has no effect. Always reads back a 0 0x1 = Writing a 1 clears the CNTOVF flag condition	RW	0x0
4	CEVT4	Capture Event 4 Status Flag 0x0 = Writing a 0 has no effect. Always reads back a 0. 0x1 = Writing a 1 clears the CEVT3 flag condition.	RW	0x0
3	CEVT3	Capture Event 3 Status Flag 0x0 = Writing a 0 has no effect. Always reads back a 0. 0x1 = Writing a 1 clears the CEVT3 flag condition.	RW	0x0

Bits	Field Name	Description	Type	Reset
2	CEVT2	Capture Event 2 Status Flag 0x0 = Writing a 0 has no effect. Always reads back a 0. 0x1 = Writing a 1 clears the CEVT2 flag condition.	RW	0x0
1	CEVT1	Capture Event 1 Status Flag 0x0 = Writing a 0 has no effect. Always reads back a 0. 0x1 = Writing a 1 clears the CEVT1 flag condition.	RW	0x0
0	INT	Global Interrupt Clear Flag 0x0 = Writing a 0 has no effect. Always reads back a 0. 0x1 = Writing a 1 clears the INT flag and enable further interrupts to be generated if any of the event flags are set to 1.	RW	0x0

Table 30-805. Register Call Summary for Register PRUSS_ECAP_ECCLR

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [2] [3]

Table 30-806. PRUSS_ECAP_ECFRC

Address Offset	0x0000 0034		
Physical Address	0x4B23 0034 0x4B2B 0034	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	ECAP Interrupt Forcing Register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CMPEQ	PRDEQ	CNTOVF	CEVT4	CEVT3	CEVT2	CEVT1	RESERVED

Bits	Field Name	Description	Type	Reset
15:8	RESERVED		R	0x0
7	CMPEQ	Force Counter Equal Compare Interrupt 0x0 = No effect. Always reads back a 0. 0x1 = Writing a 1 sets the CTR=CMP flag bit.	RW	0x0
6	PRDEQ	Force Counter Equal Period Interrupt 0x0 = No effect. Always reads back a 0. 0x1 = Writing a 1 sets the CTR=PRD flag bit.	RW	0x0
5	CNTOVF	Force Counter Overflow 0x0 = No effect. Always reads back a 0. 0x1 = Writing a 1 to this bit sets the CNTOVF flag bit.	RW	0x0
4	CEVT4	Force Capture Event 4 0x0 = No effect. Always reads back a 0. 0x1 = Writing a 1 sets the CEVT4 flag bit	RW	0x0
3	CEVT3	Force Capture Event 3 0x0 = No effect. Always reads back a 0. 0x1 = Writing a 1 sets the CEVT3 flag bit	RW	0x0
2	CEVT2	Force Capture Event 2 0x0 = No effect. Always reads back a 0. 0x1 = Writing a 1 sets the CEVT2 flag bit.	RW	0x0

Bits	Field Name	Description	Type	Reset
1	CEVT1	Always reads back a 0. Force Capture Event 1 0x0 = No effect. 0x1 = Writing a 1 sets the CEVT1 flag bit.	RW	0x0
0	RESERVED		R	0x0

Table 30-807. Register Call Summary for Register PRUSS_ECAP_ECFRC

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [2] [3]

Table 30-808. PRUSS_ECAP_PID

Address Offset	0x0000 005C		
Physical Address	0x4B23 005C 0x4B2B 005C	**Instance**	PRUSS1_ECAP PRUSS2_ECAP
Description	ECAP Revision ID		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REVISION																															

Bits	Field Name	Description	Type	Reset
31:0	REVISION	IP Revision	R	0x-(1)

(1) TI Internal information

Table 30-809. Register Call Summary for Register PRUSS_ECAP_PID

PRU-ICSS eCAP Module

- PRUSS_ECAP Register Summary: [1] [2]

30.2.9 PRU-ICSS MII RT Module

30.2.9.1 Introduction

The Real-time Media Independent Interface (MII_RT) provides a programmable I/O interface for the PRUs to access and control up to two MII ports. The MII_RT module can also be configured to push and pull data independent of the PRU cores.

NOTE: In order to guarantee the MII_RT IO timing values published in the device data manual, the PRUSS_GICLK clock must be configured for 200MHz (default value) and PRUSS_MII_RT_TXCFG0/1 must be set to 6h (non-default value).

30.2.9.1.1 Features

The PRU-ICSS MII_RT module supports:

- Two MII ports
  - Each MII port has:
    - 32-byte RX L1 FIFO
    - 64-byte RX L2 buffer
    - 64-byte TX L1 FIFO
  - Rate decoupling on TX L1 FIFO
  - Configurable pre-amble removal on RX L1 FIFO and insertion on TX L1 FIFO
  - Configurable TX L1 FIFO trigger (10 bits with 40 ns ticks)
- MII port multiplexer per direction to support line/ring structure
  - Link detection through RX_ERR
- Cyclic redundancy check (CRC)
  - CRC32 generation on TX path
  - CRC32 checker on RX path

30.2.9.1.2 Unsupported Features

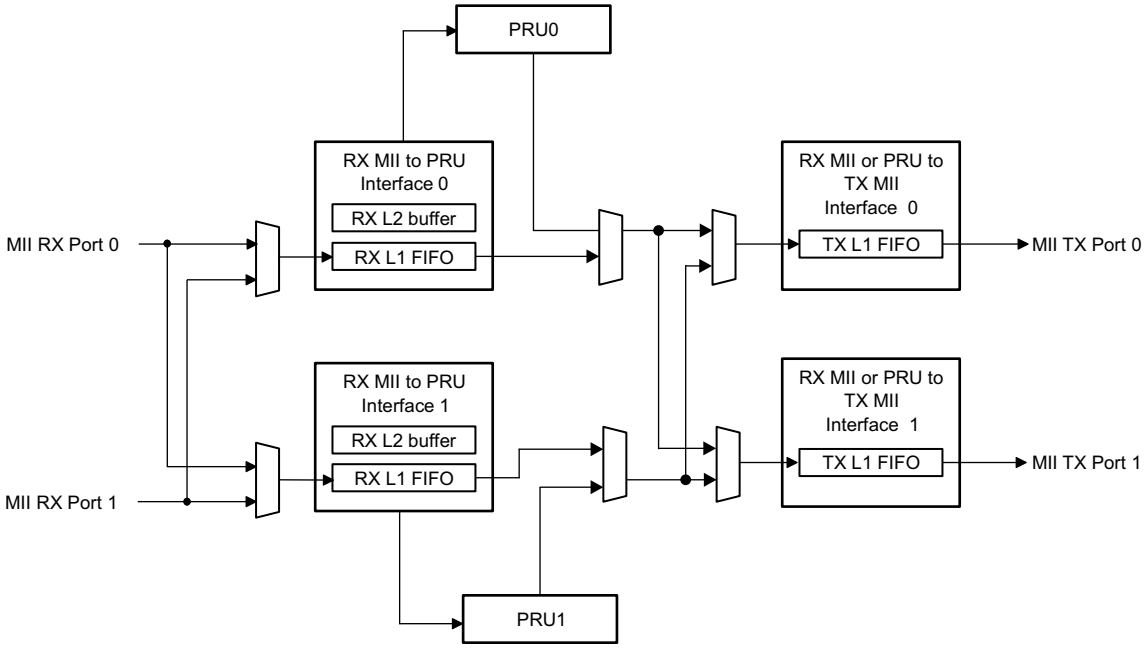
The PRU-ICSS MII_RT module does not support:

- Auto padding in TX L1 FIFO
- Dynamic TX multiplexer switching during packet handling
  - Can allow one PRU to handle both MII interfaces and a second PRU to manage the host and switch functions.

30.2.9.1.3 Block Diagram

Figure 30-82 shows the MII_RT in context of the PRU-ICSS. This diagram is a conceptual block diagram and does not necessarily reflect actual topologies.

Figure 30-82. MII_RT Block Diagram



30.2.9.2 Functional Description

30.2.9.2.1 Data Path Configuration

The MII_RT module supports three basic data path configurations. These configurations are compared in Table 30-810 and described in the following sections.

Table 30-810. Data Path Configuration Comparison

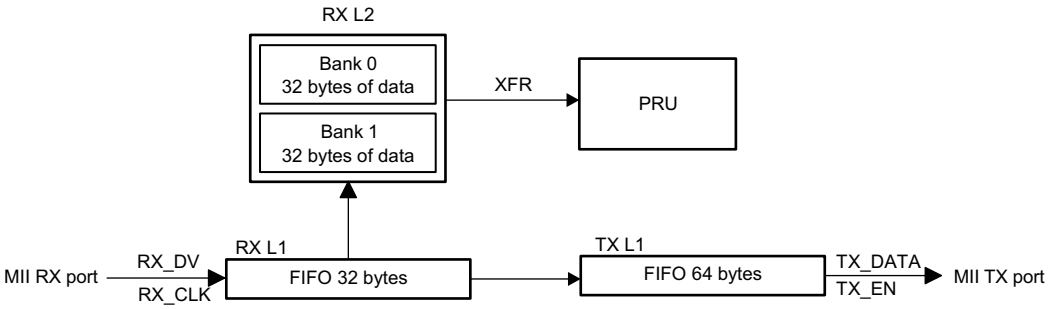
Configuration	PRU Dependency	Data Servicing	Port-to-Port Latency
Auto-forward	Snoop only	One word in flight	Low
8- or 16-bit processing with on-the-fly modifications (RX L1)	Yes	One word or byte in flight	Low
32-byte double buffer or ping-pong processing (RX L2)	Yes	Multi-words in flight	Medium (application-dependent)

30.2.9.2.1.1 Auto-forward with Optional PRU Snoop

Data is automatically forwarded from the MII RX port to the MII TX port without manipulations, as shown in Figure 30-83. This configuration does not depend on the PRU core. However, it does support an option for PRU to snoop or monitor the received data through the RX L2, shown in Figure 30-84. The PRU does not access data and status bits through R31, and it does not modify and push data.

Figure 30-83. Auto-forward

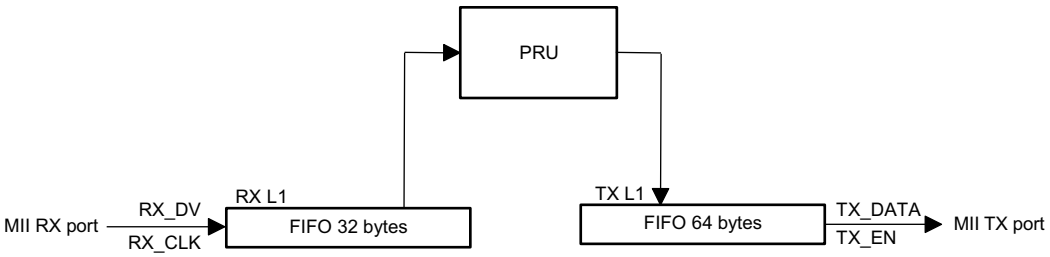
Figure 30-84. Auto-forward with PRU Snoop



30.2.9.2.1.2 8- or 16-bit Processing with On-the-Fly Modifications

This configuration services one byte or word in flight and has low latency. The PRU has the option to manipulate the received word and control popping data from the RX L1 FIFO and pushing it on the TX L1 FIFO.

Figure 30-85. 8- or 16-bit Processing with On-the-Fly Modifications

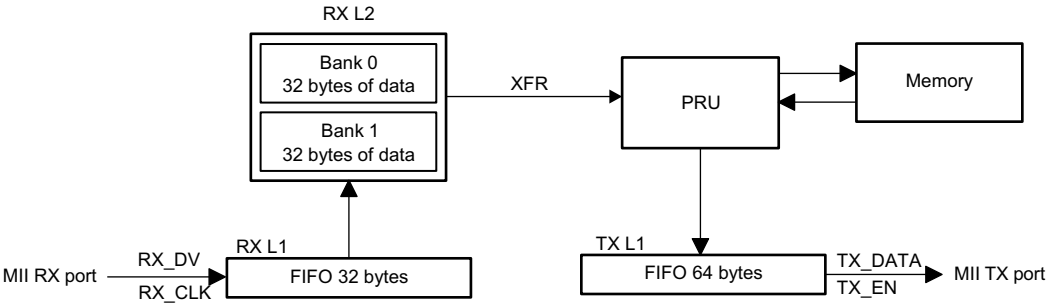


30.2.9.2.1.3 32-byte Double Buffer or Ping-Pong Processing

This configuration supports high bandwidth, high efficiency transactions. Often implementations using this mode permit relaxed servicing requirements allowing the PRU to manipulate the received data before transmitting.

Data received in this configuration is passed into the RX L2 buffer. The PRU reads multiple bytes of data from one of the RX L2 banks through the high bandwidth broadside interface and XFR instructions. The PRU can then store or manipulate data before pushing it to the TX L1 FIFO for transmission on the MII TX port.

Figure 30-86. 32-byte Double Buffer or Ping-Pong Processing



30.2.9.2.2 Definition and Terms

30.2.9.2.2.1 Data Frame Structure

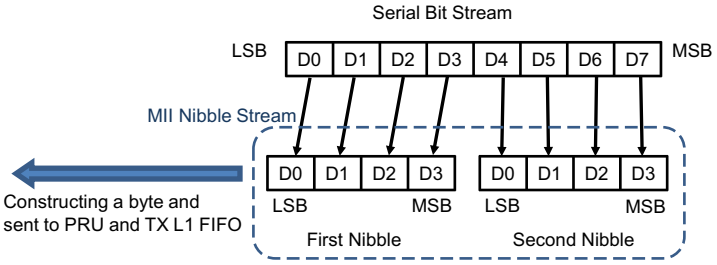
The data received and transmitted over MII conforms with the frame structure shown in Table 30-811.

Table 30-811. Frame Structure

Inter-frame	Preamble	Start of Frame Delimiter (SFD)	Data	Cyclic Redundancy Check (CRC)

The data following the SFD is formatted in a 4-bit nibble structure. Figure 30-87 illustrates the nibble order. The MSB arriving first is on the LSB side of a nibble. When receiving data, the MII_RT receive logic will wait for the next nibble to arrive before constructing a byte and delivering to the PRU.

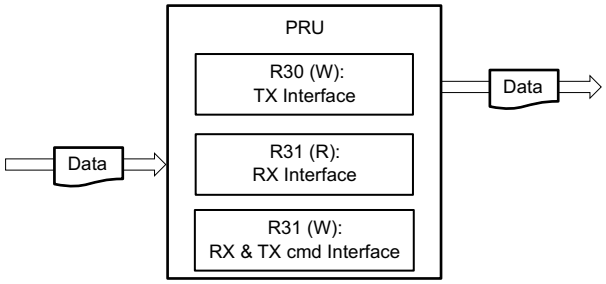
Figure 30-87. Data Nibble Structure



30.2.9.2.2.2 PRU R30 and R31

The PRU registers R30 and R31 are used to receive, transmit, and control the data for the PRU. As shown in Figure 30-88, the R31 is used to access data in the RX L1 FIFO, the R30 is used to transmit data from the PRU, and the R31 output is used the control the flow of receive and transmit. For more details about these registers, see the following sections.

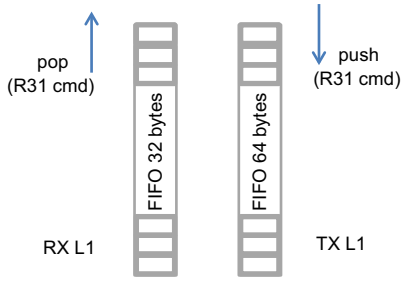
Figure 30-88. PRU R30, R31 Operations



30.2.9.2.2.3 RX and TX L1 FIFO Data Movement

To advance the next data byte seen by R31, the PRU must pop the data from the RX L1 FIFO. Likewise, the PRU can push the data from R30 to the TX L1 FIFO. These operations are illustrated in Figure 30-89.

Figure 30-89. Reading and Writing FIFO Data



30.2.9.2.2.4 CRC Computation

30.2.9.2.2.4.1 Receive CRC Computation

For the incoming data, the MII_RT calculates CRC32 and then compares against the value provided in the incoming frame. If there is a mismatch, the MII RT signals ERROR_CRC to the PRU. If a previous node or Ethernet device appended an error nibble, the CRC calculation of received packet will be wrong because the longer frame and the frame length will end at a 4-bit boundary instead of the usual 8-bit boundary. When RX_DV goes inactive on the 4-bit boundary, the interface will assert DATA_RDY and BYTE_RDY flag with the ERROR NIBBLE. The error event is also mapped into the PRU-ICSS INTC.

30.2.9.2.2.4.2 Transmit CRC Computation

For the outgoing data, the MII_RT calculates the CRC32 value and inserts it into outgoing packets. The CRC value computed on each MII transmit path is also available in memory map registers (MMRs) that can be read by the PRU and used primarily for debug and diagnostic purposes. The CRC is inserted into the outgoing packet based on the commands received through the R31 register of the PRU. The CRC will be inserted into the TX L1 FIFO, and there must be enough room to store the CRC value in the FIFO or else the FIFO will overflow. As Table 30-812 shows, the CRC programming model supports three sequences that provide more flexibility. Note “cmdR31” indicates write to the mentioned bits of the R31 command interface.

Table 30-812. TX CRC Programming Models

Option 1	Step 1: cmdR31 [TX_CRC_HIGH + TX_CRC_LOW + TX_EOF]
Option 2	Step 1: cmdR31 [TX_CRC_HIGH] Step 2: wait > 6 clocks (PRU cycles) Step 3: cmdR31 [TX_CRC_LOW + TX_EOF]
Option 3	Step 1: cmdR31 [TX_CRC_HIGH] Step 2: wait > 6 clocks (PRU cycles) Step 3: read PRUSS_MII_RT_TX_CRC0/1 Step 4: modify CRC[15:0] Step 5: cmdR31 [TX_PUSH16 + TX_EOF + TX_ERROR_NIBBLE]

30.2.9.2.3 RX MII Interface

30.2.9.2.3.1 RX MII Submodule Overview

The RX MII interface is composed of multiple submodules that process the incoming frames and pass receive data and status information into the PRU register R31. These submodules include:

- Latch received data
- Start of frame detection
- Start frame delimiter detection

- CRC calculation and error detection
- Enhanced link detection through RX error detection

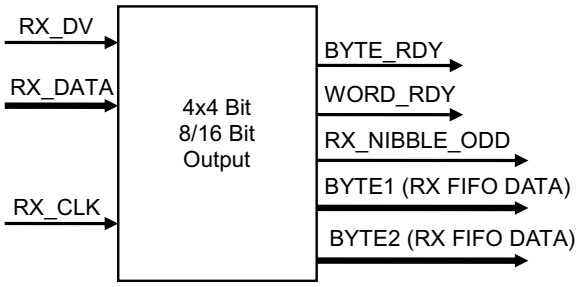
Table 30-813 includes more details about the internal signals and output of these submodules.

30.2.9.2.3.1.1 Receive Data Latch

The receive data from the MII interface is stored in the receive data FIFO which is 32 bytes. The PRU can access this data through the register R31. Depending on the configuration settings, the data can be latched on reception of one or two bytes. In each scheme, the configured number of nibbles is assembled before being copied into the PRU registers. Figure 30-90 shows the inputs and outputs of the data latch logic block.

The receiver logic in MII_RT can be programmed through the PRUSS_MII_RT_RXCFG0 and PRUSS_MII_RT_RXCFG1 registers to remove or retain the preamble + SFD from incoming frames.

Figure 30-90. RX Data Latch



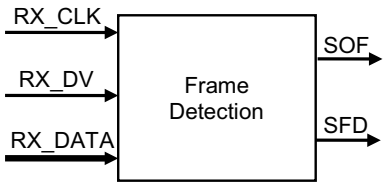
30.2.9.2.3.1.1.1 Start of Frame Detection

The start of frame detection logic tracks the frame boundaries and signals the beginning of a frame to other components of the PRU-ICSS. This logic detects two events:

- Start of Frame (SOF) event that occurs when Receive Data Valid MII signal is sampled high.
- Start of Frame Delimiter (SFD) event is seen on MII Receive Data bus.

These event triggers can be used to add timestamp to the frames. The notification for these events is available through R31 as well as through INTC which is integrated in the PRU-ICSS. Figure 30-91 shows the inputs and outputs of the start of frame detection logic block.

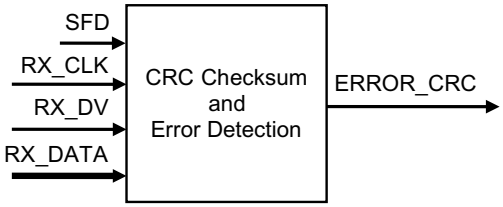
Figure 30-91. Start of Frame Detection



30.2.9.2.3.1.1.2 CRC Error Detection

For each incoming frame, the CRC is calculated by the MII_RT and compared against the CRC included in the frame. When the two values do not match, a CRC error is flagged. The ERROR_CRC indication is available in the register interface (PRU R31 Receive Interface) as well as in the FIFO interface (RX L2 Status Interface). It is also provided to the INTC which is integrated in the PRU-ICSS. Figure 30-92 shows the inputs and outputs of CRC error detection logic block.

Figure 30-92. CRC Error Detection

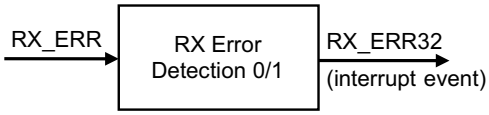


30.2.9.2.3.1.1.3 RX Error Detection and Action

The RX error detection logic tracks the receive error signaled by the physical layer and informs the PRU-ICSS INTC whenever an error is detected. Figure 30-93 shows the inputs and outputs of the RX error detection logic block. Note the following dependencies:

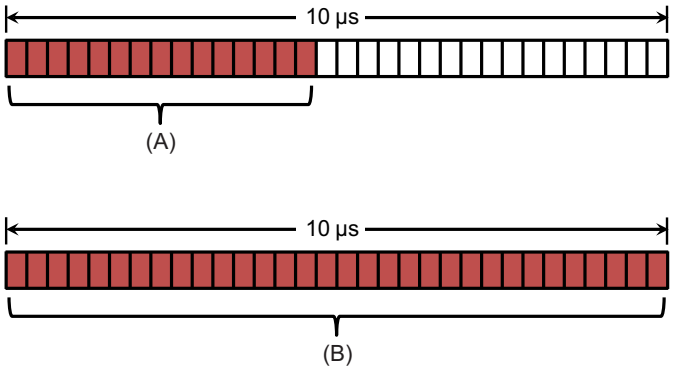
- RX_ERR signal is only sampled when RX_DV is asserted.
- All nibbles are discarded post RX_ERR event, including the nibble which had RX_ERR asserted. This state will remain until EOF occurs.
- Due to this fact, RX L1 FIFO and RX L2 FIFO will never receive any data with RX_ERR or post RX_ERR during that frame.

Figure 30-93. RX Error Detection



This submodule also keeps track of a running count of receive error events within a 10 µs error detection window, as shown in Figure 30-94. The INTC is notified when 32 or more events have occurred in a 10 µs error detection window. The error detection window is not a sliding window but a non-overlapping window with no specific initialization time with respect to incoming traffic. The timer starts its 10 µs counts immediately after de-assertion of reset to the MII_RT module.

Figure 30-94. Error Detection Window with Running Counter



A There are fewer than 32 consecutive error events in the 10 µs window. The detection module will not forward to the interrupt controller (INTC).

B There are more or equal to 32 error events in the 10 µs window. The detection module will notify the interrupt controller (INTC).

30.2.9.2.3.1.2 RX Data Path Options to PRU

There are two data path options for delivering received data to the PRU, described further in the subsequent sections:

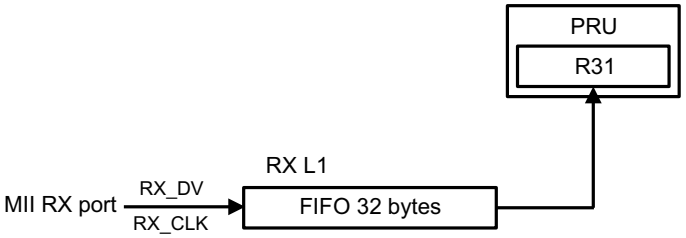
1. RX MII port → RX L1 FIFO → PRU (one word in flight)
2. RX MII port → RX L1 FIFO → RX L2 buffer → PRU (multi-word in flight)

Once the PRU has received RX data, the PRU can both manipulate received data or send data to the TX MII Interface.

30.2.9.2.3.1.2.1 RX MII Port → RX L1 FIFO → PRU

The RX L1 FIFO to PRU interface is depicted in Figure 30-95. In this mode, the data received from the MII interface is fed into the 32-byte RX L1 FIFO. The first data byte into the FIFO is automatically available in R31 of the PRU. Therefore, the PRU firmware can directly operate on this data without having to read it in a separate instruction. This allows the PRU to access receive data with low latency.

Figure 30-95. RX L1 to PRU Interface



When the new data is received, the PRU is provided with up to two bytes at a time in the R31 register, as shown in Figure 30-96. Once the PRU processes the incoming data, it instructs the MII_RT by writing to the R31 command interface bits to pop one or two bytes of data from the 32-byte RX FIFO. The pop operation causes current contents of R31 to be refreshed with new data from the incoming packet. Each time the data is popped, the status bits change to indicate so. If the pop is completed and there is no new data, the status bits immediately change to indicate no new data. Note the current R31 content, including data, will be lost after issuing the pop operation. If this information needs to be accessed later, the PRU should store the existing R31 content before popping new data.

Figure 30-96. MII RX Data to PRU R31 (R) and RX FIFO

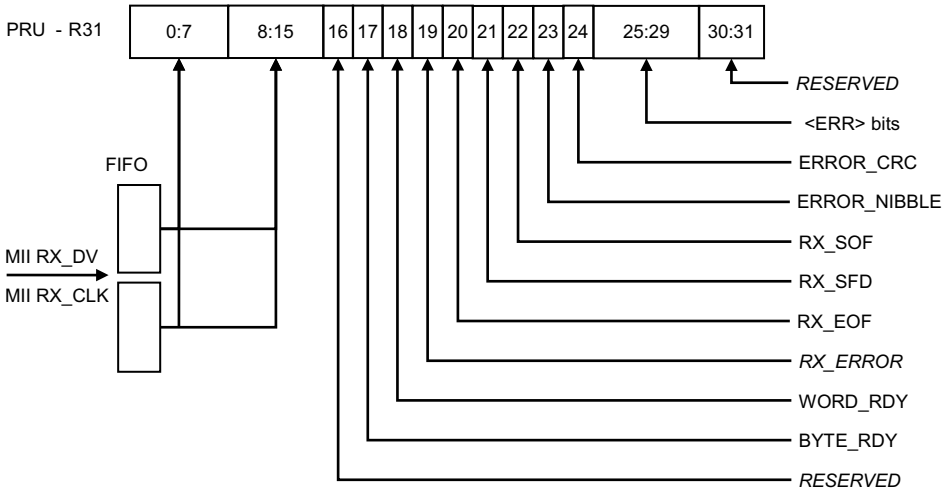


Table 30-813 describes the receive interface data and status contents provided by the R31 register. These contents are available when R31 is read. To configure this register, the PRU GPI mode should be set for MII_RT mode in the CFG register space. Note the following:

1. If the data from receive path is not read in time, it could cause an overflow event because the data is still continuously provided to the 32-byte receive FIFO. Due to the receive FIFO overflow, the data gets automatically discarded to avoid lack of space in the FIFO. At the same time, an interrupt is raised to the INTC through a system event (PRU<n>_RX_OVERFLOW). To detect an overflow condition, the PRU should poll for this system event condition and a RX RESET command through the R31 command interface is required to clear out from this condition. Note that the received Ethernet frame is corrupted and should not be used for further processing as bytes have been dropped due to the overflow condition. A FIFO reset is recommended.

2. The receive data in the R31 register is available following synchronization to the PRU clock domain. So, there is a finite delay (120 ns) when data is available from MII interface and it is accessible to the PRU.
3. The receive FIFO also has the capability to be reset through software. When reset, all contents of receive FIFO are purged and it may result in the current frame not being received as expected. When a frame is being received and the PRU resets the RX FIFO, the remaining frame is not placed into the RX FIFO. However, any new frame arriving on the receive MII port will be stored in the FIFO.

Table 30-813. PRU R31: Receive Interface Data and Status (Read Mode)

Bits	Field Name	Description
31:30	RESERVED	In case of register interface, these bits are provided to PRU by other modules in PRU-ICSS. From the MII_RT module point of view, these bits are always zero.
29	RX_MIN_FRM_CNT_ERR	RX_MIN_FRM_CNT_ERR is set to 1 when the count of total bytes of incoming frame is less than the value defined by RX_MIN_FRM_CNT. RX_MIN_FRM_CNT_ERR is cleared by RX_ERROR_CLR.
28	RX_MAX_FRM_CNT_ERR	RX_MAX_FRM_CNT_ERR is set to 1 when the count of total bytes of incoming frame is more than the value defined by RX_MAX_FRM_CNT_ERR. RX_MAX_FRM_CNT_ERR is cleared by RX_ERROR_CLR.
27	RX_EOF_ERROR	RX_EOF_ERROR is set to 1 when an RX_EOF event or RX_ERROR event occurs. RX_EOF_ERROR is cleared by RX_ EOF_CLR and/or RX_ ERROR_CLR.
26	RX_MAX_PRE_CNT_ERR	RX_MAX_PRE_CNT_ERR is set to 1 when the number of nibbles equaling 0x5 before SFD event (0xD5) is more than the value defined by PRUSS_MII_RT_RX_PCNT0/1 [RX_MAX_PCNT]. RX_MAX_PRE_CNT_ERR is cleared by RX_ERROR_CLR.
25	RX_ERR	RX_ERR is set to 1 when pr1_mii0/1_rxer is asserted while pr1_mii0/1_rxdv bit is set. RX_ERR is cleared by RX_ERROR_CLR.
24	ERROR_CRC	ERROR_CRC indicates that the frame has a CRC mismatch. This bit is valid when the RX_EOF bit is set. It should be noted that ERROR_CRC bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO. ERROR_CRC is cleared by RX_ERROR_CLR.
23	ERROR_NIBBLE	ERROR_NIBBLE indicates that the frame ended in odd nibble. It should be considered valid only when the RX_EOF bit and pr1_mii0/1_rxdv are set. Nibble counter is enabled post SFD event. It should be noted that ERROR_NIBBLE bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO. ERROR_NIBBLE is cleared by RX_ERROR_CLR.
22	RX_SOF	RX_SOF transitions from low to high when the frame data starts to arrive and pr1_mii0/1_rxdv is asserted. Note there will be a small sync delay of 0ns – 5ns. The PRU must write one to this bit through the R31 command interface to clear it. The recommended time to clear this bit is at the end of frame (EOF). It should be noted that RX_SOF bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO.
21	RX_SFD	RX_SFD transitions from low to high when the SFD sequence (0xD5) post RX_SOF is observed on the receive MII data. The PRU must write one to this bit through the R31 command interface to clear it. The recommended time to clear this bit is at the end of frame (EOF). It should be noted that RX_SFD bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO.
20	RX_EOF	RX_EOF indicates that the frame has ended and pr1_mii0/1_rxdv is de-asserted. It also validates the CRC match bit. Note there will be a small sync delay of 0ns – 5ns. The PRU must write one to clear this bit in the R31 command interface at the end of the frame. It should be noted that RX_EOF bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO.
19	RX_ERROR	RX_ERROR indicates one or more of the following errors occurred: • RX_MAX/MIN_FRM_CNT_ERR • RX_MAX/MIN_PRE_CNT_ERR • RX_ERR RX_ERROR is cleared by RX_ERROR_CLR.
18	WORD_RDY	WORD_RDY indicates that all four nibbles in R31 have valid data. There is a 2 clock cycle latency from the command RX_POP16 to WORD_RDY update. Therefore, firmware needs to insure it does not read WORD_RDY until 2 clock cycles after RX_POP16.
17	BYTE_RDY	BYTE_RDY indicates that the lower two nibbles in R31 have valid data. There is a 2 clock cycle latency from the command RX_POP8 to BYTE_RDY update. Therefore, PRU firmware needs to insure it does not read BYTE_RDY until 2 clock cycles after RX_POP8.
16	RESERVED	

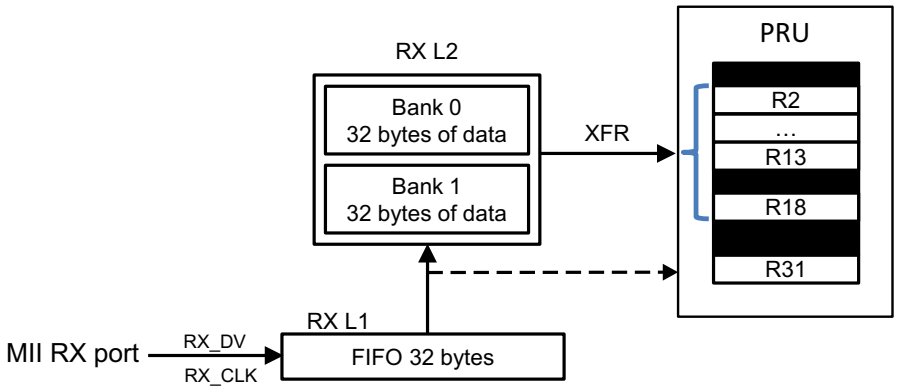
Table 30-813. PRU R31: Receive Interface Data and Status (Read Mode) (continued)

Bits	Field Name	Description
15:8	BYTE1	Data Byte 1. This data is available such that it is safe to read by the PRU when the DATA_RDY/BYTE_RDY/WORD_RDY bits are asserted.
7:0	BYTE0	Data Byte 0. This data is available such that it is safe to read by the PRU when the DATA_RDY/BYTE_RDY/WORD_RDY bits are asserted.

30.2.9.2.3.1.2.2 RX MII Port → RX L1 FIFO → RX L2 Buffer → PRU

The RX L2 is an optional high performance buffer between the RX L1 FIFO and the PRU. Figure 30-97 illustrates the receive data path using RX L2 buffer. This data path is characterized by multi-word in flight transactions.

Figure 30-97. RX L2 to PRU Interface



The 64-byte RX L2 buffer is divided into two 32 byte banks, or ping/pong buffers. When the RX L2 is enabled, the incoming data from the MII RX port will transmit first to the 32 byte RX L1 FIFO. RX L1 pushes data into RX L2, starting when the first byte is ready until the final EOF marker. The RX L2 buffer does not apply any backpressure to the RX L1 FIFO. Therefore, it is the PRU firmware's responsibility to fetch the data in RX L2 before it is overwritten by the cyclic buffer. The RX L1 will remain near empty, with only one byte (nibble) stored.

Each RX L2 bank holds up to 32 bytes of data, and every four nibbles (or 16 bits) of data has a corresponding 8-bit status. The data and status information are stored in packed arrays. In each bank, R2 to R9 contains the data packed array and R10 to R13 contains the status packed array. Figure 30-98 shows the relationship of the data registers and status registers. The RX L2 status registers record status information about the received data, such as ERROR_CRC, RX_ERROR, STATUS_RDY, etc. The RX L2 status register details are described in Table 30-814. Note RX_RESET clears all Data and Status elements and resets R18.

Figure 30-98. Data and Status Register Dependency

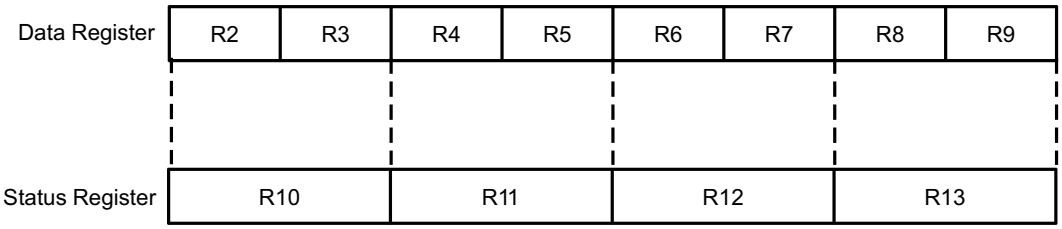


Table 30-814. RX L2 Status

Bit	Field Name	Description
7	ERROR_CRC	ERROR_CRC indicates that the frame has a CRC mismatch. This bit is valid when the RX_EOF bit is set. It should be noted that ERROR_CRC bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO. ERROR_CRC will only be set for one entry, self clear on next entry.
6	ERROR_NIBBLE	ERROR_NIBBLE indicates that the frame ended in odd nibble. It should be considered valid only when the RX_EOF bit and pr1_mii0/1_rxdv are set. Nibble counter is enabled post SFD event. It should be noted that ERROR_NIBBLE bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO. ERROR_NIBBLE will only be set for one entry, self clear on next entry.
5	RX_SOF	RX_SOF transitions from low to high when the frame data starts to arrive and pr1_mii0/1_rxdv is asserted. Note there will be a small sync delay of 0ns – 5ns. The recommended time to clear this bit is at the end of frame (EOF). It should be noted that RX_SOF bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO. RX_SOF will only be set for one entry, self clear on next entry.
4	RX_SFD	RX_SFD transitions from low to high when the SFD sequence (0xD5) post RX_SOF is observed on the receive MII data. The recommended time to clear this bit is at the end of frame (EOF). It should be noted that RX_SFD bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO. RX_SOF will only be set for one entry, self clear on next entry.
3	RX_EOF	RX_EOF indicates that the frame has ended and pr1_mii0/1_rxdv is de-asserted. It also validates the CRC match bit. Note there will be a small sync delay of 0ns – 5ns. It should be noted that RX_EOF bit is ready in early status, which means it is calculated before data is available in RXL1 FIFO. RX_EOF will only be set for one entry, self clear on next entry.
2	RX_ERROR	RX_ERROR indicates one or more of the following errors occurred: • RX_MAX/MIN_FRM_CNT_ERR • RX_MAX/MIN_PRE_CNT_ERR • RX_ERR RX_ERROR is cleared by RX_ERROR_CLR.
1	STATUS_RDY	STATUS_RDY is set when RX_EOF or write pointer advanced by 2. This is a simple method for software to determine if RX_EOF event has occurred or new data is available. If RX_EOF is not set, all status bits are static.
0	RX_ERR	RX_ERR is set to 1 when pr1_mii0/1_rxer is asserted while pr1_mii0/1_rxdv bit is set. It will get set for first pr1_mii0/1_rxer event and self clear on SOF for the next FRAME.

Bank 0 and Bank 1 are used as ping/pong buffers. RX L2 supports the reading of a write pointer in R18 that allows software to determine which bank has active write transactions, as well as the specific write address within packed data arrays.

The PRU interacts with the RX L2 buffer using the high performance XFR read instructions and broadside interface. Table 30-815 shows the device XFR ID numbers for each bank.

Table 30-815. RX L2 XFR ID

Device ID	Function	Description
20	Selects RX L2 Bank0	R2:R9 Data packed array R10:R13 Status packed array
21	Selects RX L2 Bank1	R2:R9 Data packed array R10:R13 Status packed array

Table 30-815. RX L2 XFR ID (continued)

Device ID	Function	Description
20/21	Byte pointer of current write	R18[5:0] Pointer indicating location of current write in data packed array. 0 = Bank0.R2.Byte0 (default and reset value) 1 = Bank0.R2.Byte1 2 = Bank0.R2.Byte2 3 = Bank0.R2.Byte3 4 = Bank0.R3.Byte0 … 63=Bank1.R9.Byte3

XFR read transactions are passive and have no effect on any status or other states in RX L2. The firmware can also read R18 to determine which Bank has active write transactions and the location of the transaction. With this information, the firmware can read multiple times the stable preserved data. Note when RX L1 data is written to RX L2, the next status byte gets cleared at the same time the current status byte gets updated. The rest of the status buffer is persistent. When software is accessing any register of the ping/ pong buffer, software needs to issue an XFER read transaction to fetch the latest/current state of the ping/pong buffer. The PRU registers will not reflect the current snapshot of L2 unless an XFER is issued by software.

30.2.9.2.4 TX MII Interface

Data to be transmitted is loaded into the TX L1 FIFO. The transmit FIFO (TX L1) stores up to 64 bytes of transmit data. From the FIFO, the data is sent to the MII TX port of the PHY by the MII_RT transmit logic.

The transmit FIFO also has the capability to be reset through software (TX_RESET). When reset, all contents of transmit FIFO are purged and this may result in a frame not getting transmitted as expected, if the transmission is already ongoing. Any new data written in the transmit FIFO results in a new frame being composed and transmitted. An overflow event will require a TX_RESET to recover from this condition.

There are four dependencies that must be true for TX_EN to assert.

1. TX L1 FIFO not empty
2. Interpacket gap (IPG) timer expiration
3. RX_DV to TX_EN timer expiration
4. TX_EN compare timer expiration

The transmit interface also provides an underflow error signal in case there was no data loaded when TX_EN triggered. The transmit underflow signal is mapped to the INTC in PRU-ICSS. The PRU firmware must track the FIFO fill level, such as by a timer or the PRU cycle count register (PRU_ICSS_CTRL_CYCLE).The current FIFO fill level cannot be accessed by PRU firmware. The firmware can issue an R31 command via R31 bit 29 (TX_EOF) to indicate that the last byte has been written into the TX FIFO.

30.2.9.2.4.1 TX Data Path Options to TX L1 FIFO

There are two data path options for delivering data to the TX L1 FIFO and transmit port, described further in the subsequent sections:

1. PRU → TX L1 FIFO → TX MII port
2. RX L1 FIFO → TX L1 FIFO → TX MII port

30.2.9.2.4.1.1 PRU → TX L1 FIFO → TX MII Port

The PRU can be used to feed data into the TX L1 FIFO using the R30 and R31 registers, shown in Figure 30-99. The PRU writes up to two bytes of data into R30 and then pushes the data into the TX L1 FIFO by writing to the R31 command interface.

Figure 30-99. PRU to TX L1 Interface

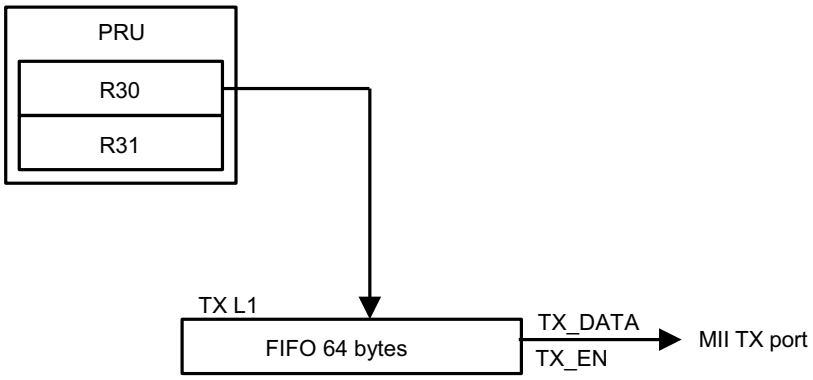


Figure 30-100 shows the R30 transmit interface. The lower 16 bits of the R30 (or FIFO transmit word) contain transmit data nibbles. The upper 16 bits contain mask information. The operation to be performed on the transmit interface is controlled by PRU writes to the R31 command interface. Table 30-816 describes the TXMASK and TXDATA bit fields of the R30 transmit interface.

Figure 30-100. PRU to TX MII Interface

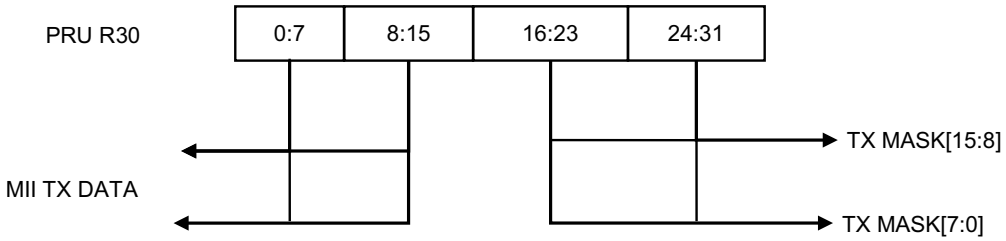


Table 30-816. PRU R30: Transmit Interface

Bits	Field Name	Description
31:16	TXMASK	The TXMASK is used to determine which of RX L1 FIFO received data and R30 data is sent to transmit FIFO. It must to be applied to TXDATA and RXDATA before it is transmitted. To disable TXMASK and transmit only TXDATA via R30, set to 0xFFFF. Note software should not pop the RXDATA from the RX L1 FIFO before pushing the TXDATA. This will cause new data to propagate before the push. Otherwise, software can pop and push on the same command for bytes only or delay the pop after the push for words or bytes.
15:0	TXDATA	Data provided by the PRU to be sent to transmit path after applying the mask. When 16 bits are to be transmitted, all bits of this and TXMASK field are used. When 8 bits are to be transmitted, the bits [7:0] of this and TXMASK field are used.

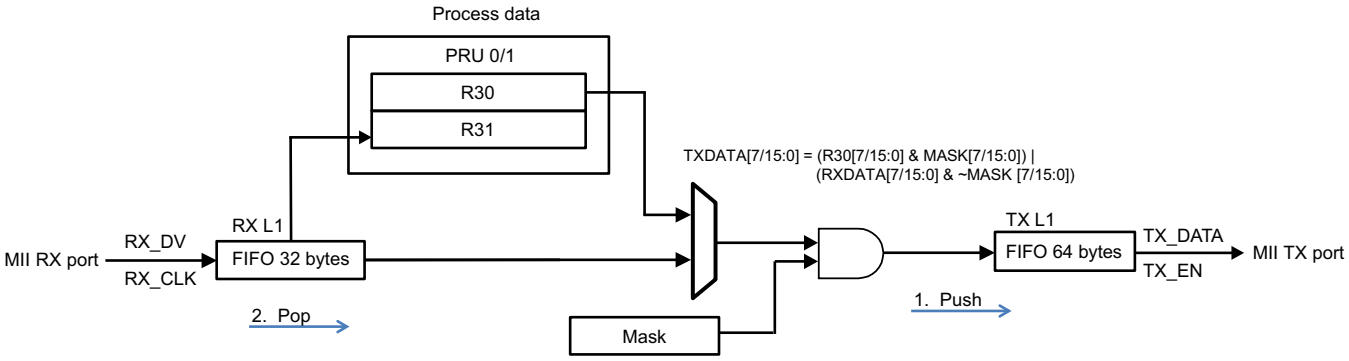
Using the TX mask, the PRU can send a mix of R30 and RX L1 FIFO data to the TX L1 FIFO. Note the TX mask is only available when the PRU is fed one word or byte at a time by the RX L1 FIFO. It is not applicable when the RX L2 buffer is enabled. To disable TX mask, set TXMASK to 0xFFFF.

As shown in Figure 30-101, the PRU drives the MII transmit interface through its R30 register. The contents of R30 and RX data from the receive interface are taken and fed into a 64 byte transmit FIFO.

Before transmission, a mask is applied to the data portion of the R30 register. By using the mask, the PRU firmware can control whether received data from the RX L1 FIFO is sent to transmit, R30 data is sent to transmit, or a mix of the two is sent. The Boolean equation that is used by MII_RT to compose TX data is:

TXDATA[7/15:0] = (R30[7/15:0] & MASK[7/15:0]) | (RXDATA[7/15:0] & ~MASK [7/15:0])

Figure 30-101. TX Mask Mode

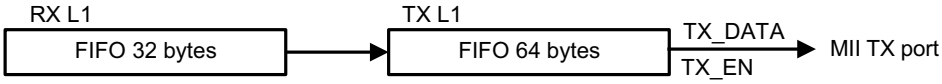


30.2.9.2.4.1.2 RX L1 FIFO → TX L1 FIFO → TX MII Port

When PRUSS_MII_RT_TXCFG0/1[TX_AUTO_SEQUENCE] is set, the data frame is passed from the RX to TX FIFOs without the any interaction of the PRU. This mode of operations is shown in Figure 30-102. The RX L1 will push into TX L1 as long as it is enabled and not full.

There is no PRU dependency in this mode and no option for the PRU to perform any operation to the TX L1 FIFO. RX_RESET clears all data and status elements.

Figure 30-102. RX L1 to TX L1 Interface



30.2.9.2.5 PRU R31 Command Interface

The PRU uses writes to R31[31:16] to control the reception and transmission of packets in register mode. Table 30-817 lists the available commands. Each bit in the table is a single clock pulse output from the PRU. When more than one action is to be performed in the same instant, the PRU firmware must set those command bits in one instruction.

Table 30-817. PRU R31: Command Interface (Write Mode)

Bit	Command	Description
31	TX_CRC_ERR	TX_CRC_ERR command when set will add 0xa5 byte to the TX L1 FIFO if the current FCS is valid. This bit can only be set with the TX_EOF command and optionally with the TX_ERROR_NIBBLE command. It cannot get set with any other commands, and the PRU firmware must wait > 2 clocks from the last command. Note for proper operations auto-forward preamble must be enabled.
30	TX_RESET	TX_RESET command is used to reset the transmit FIFO and clear all its contents. This is required to recover from a TX FIFO overrun.
29	TX_EOF	TX_EOF command is used to indicate that the data loaded is considered last for the current frame
28	TX_ERROR_NIBBLE	TX_ERROR_NIBBLE command is used to insert an error nibble. This makes the frame invalid. Also, it will add 0x0 after the 32-bit CRC.
27	TX_CRC_HIGH	TX_CRC_HIGH command ends the CRC calculations and pushes CRC[31:16] to append to the outgoing frame in the TX L1 FIFO. Note PRUSS_MII_RT_TX_CRC0/1 will become valid after 6 clock cycles.
26	TX_CRC_LOW	TX_CRC_LOW command pushes CRC[15:0] to append to the outgoing frame in the TX L1 FIFO.

Table 30-817. PRU R31: Command Interface (Write Mode) (continued)

Bit	Command	Description
25	TX_PUSH16	TX_PUSH16 command applies mask to two bytes from receive path and transmit. Note TX_PUSH16 needs to occur before TX_POP16 if data is not fully masked. TX CRC requires the data to be valid for 2 clock cycles.
24	TX_PUSH8	TX_PUSH8 command applies mask to one byte from receive path and transmit. Note TX_PUSH8 needs to occur before TX_POP8 if data is not fully masked.
23	RX_ ERROR_CLR	RX_ERROR_CLR command is used to clear RX_ ERROR indicator bit by writing 1.
22	RX_EOF_CLR	RX_EOF_CLR command is used to clear RX_EOF status indicator bit by writing 1.
21	RX_SFD_CLR	RX_SFD_CLR command is used to clear RX_SFD indicator bit by writing 1.
20	RX_SOF_CLR	RX_SOF_CLR command is used to clear RX_SOF indicator bit by writing 1.
19	Reserved	Reserved
18	RX_RESET	RX_RESET is used to reset the receive FIFO and clear all contents. This is required to recover from a RX FIFO overrun, if software does not want to undrain. The typical use case is assertion after RX_EOF. If asserted during an active frame, the following actions will occur: 1. Terminate the current frame 2. Block/terminate all new data 3. Flush/clear all FIFO elements 4. Cause RX state machine into an idle state 5. Cause EOF event 6. Cause minimum frame error, if you abort before minimum size reached
17	RX_POP16	RX_POP16 command advances the receive traffic by two bytes. This is only required when you are using R31 to read the data. After R31[15:0] is ready to read by PRU, it will set 1 to WORD_RDY, and the next new data will be allowed to advance. RX_POP16 to WORD_RDY update has 2 clock cycles latency. Firmware needs to insure it does not read WORD_RDY/BYTE_RDY until 2 clock cycles after RX_POP16.
16	RX_POP8	RX_POP8 command advances the receive traffic by one bytes. This is only required when you are using R31 to read the data. After R31[7:0] is ready to read by PRU, it will set 1 to BYTE_RDY, and the next new data will be allowed to advance. RX_POP8 to BYTE_RDY update has 2 clock cycles latency. Firmware needs to insure it does not read WORD_RDY/BYTE_RDY until 2 clock cycles after RX_POP8.

30.2.9.2.6 Other Configuration Options

30.2.9.2.6.1 Nibble and Byte Order

The PRU core is little endian. To support big endian, the MII_RT supports optional nibble swapping on both the RX and TX side.

On the receive side, the order of the two data bytes in RX R31 and the RX L2 buffer are configurable through the RX_BYTE_SWAP bit in the PRUSS_MII_RT_RXCFG0/1 registers, as shown in Table 30-818. Note the Nibble0 is the first nibble received.

Table 30-818. RX Nibble and Byte Order

Configuration	Order
PRUSS_MII_RT_RXCFG0/1 [RX_BYTE_SWAP] = 0 (default)	R31[15:8] / RXL2[15:8] = Byte1{Nibble3,Nibble2} R31[7:0] / RXL2[7:0] = Byte0{Nibble1,Nibble0}

Table 30-818. RX Nibble and Byte Order (continued)

Configuration	Order
PRUSS_MII_RT_RXCFG0/1 [RX_BYTE_SWAP] = 1	R31[15:8] / RXL2[15:8] = Byte0{Nibble1,Nibble0} R31[7:0] / RXL2[7:0] = Byte1{Nibble3,Nibble2}

On the transmit side, the order of the two data bytes and mask bytes in TX R30 are configurable through the TX_BYTE_SWAP bit in the PRUSS_MII_RT_TXCFG0/1 registers, as shown in Table 30-819. Note the Nibble0 is the first nibble received.

Table 30-819. TX Nibble and Byte Order

Configuration	Order
PRUSS_MII_RT_TXCFG0/1 [TX_BYTE_SWAP] = 0 (default)	R30[15:8] = Byte1{Nibble3,Nibble2} R30[7:0] = Byte0{Nibble1,Nibble0} R30[31:24] = TX_MASK[15:8] R30[23:16] = TX_MASK[7:0]
PRUSS_MII_RT_TXCFG0/1 [TX_BYTE_SWAP] = 1	R30[15:8] = Byte0{Nibble1,Nibble0} R30[7:0] = Byte1{Nibble3,Nibble2} R30[31:24] = TX_MASK[7:0] R30[23:16] = TX_MASK[15:8]

30.2.9.2.6.2 Preamble Source

The MII_RT module has the option to preserve and forward a received preamble in the TX data stream, use a preamble provided by the PRU, or auto-generate a preamble. These configurations are highlighted in Table 30-820.

Table 30-820. Preamble Configuration Options

RX_CUT_PREAMBLE	Determines whether RX preamble is passed to the RX L1/L2 FIFO
RX_AUTO_FWD_PRE	Determines whether RX preamble is automatically passed to TX L1 FIFO
TX_AUTO_PREAMBLE	TX interface logic auto-generates and appends preamble to TX data stream with the first push of data into the TX L1 FIFO. Note that enabling this option does fill the TX FIFO with the preamble length, hence software has to consider this to not overrun the TX FIFO.

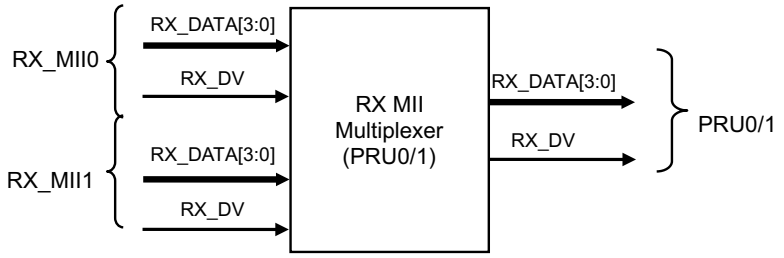
30.2.9.2.6.3 PRU and MII Port Multiplexer

The MII_RT module supports configurable PRU core to MII TXn / RXn port mapping. By default, PRU0 is mapped to TX1 and RX0 and PRU1 is mapped to TX0 and RX1. However, the system supports the flexibility to map any PRU core to any TX and RX port. Note the mapping options are destination fixed. For example, the input to PRU0 can be either RX_MII0 or RX_MII1. Similarly, the input to TX_MII0 can be either PRU0 or PRU1.

30.2.9.2.6.3.1 Receive Multiplexer

A multiplexer is provided to allow selecting either of the two MII interfaces for the receive data that is sent to PRU. Figure 30-103 shows the symbol of receive multiplexer of PRU.

Figure 30-103. MII Receive Multiplexer

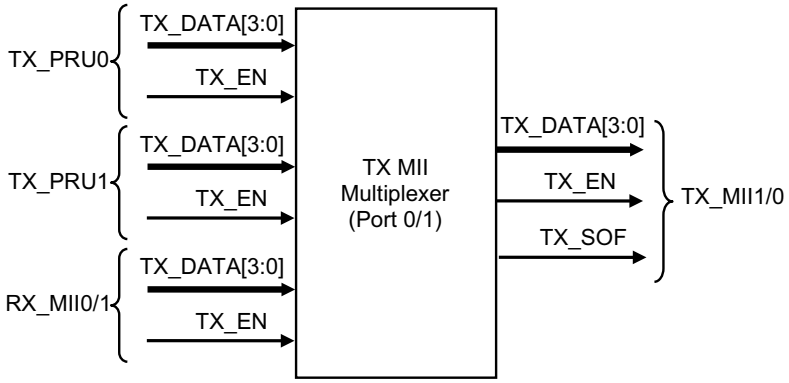


There are two receive multiplexer instances to enable selection of RX MII path for each PRU. The select lines of the RX multiplexers are driven from the PRU-ICSS programmable registers (PRUSS_MII_RT_RXCFG0/1).

30.2.9.2.6.3.2 Transmit Multiplexer

On the MII transmit ports, there is a multiplexer for each MII transmit port that enables selection of either the transmit data from the PRUs or from the RX MII interface of the other MII interface. Figure 30-104 shows the symbol of transmit multiplexer of PRU.

Figure 30-104. MII Transmit Multiplexer

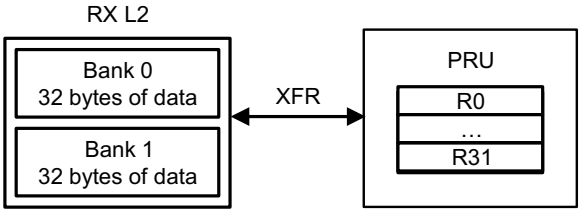


The transmit multiplexers enable the PRU-ICSS to either operate in a bypass mode where the PRU is not involved in processing MII traffic or use of one of the PRU cores for transmitting data into the MII interface. There are two instances of the TX MII multiplexer and the select lines for each TX multiplexer are provided by the PRU-ICSS. The select lines are common between register and FIFO interface. It is expected that the select lines will not change during the course of a frame so that can avoid data exchange error.

30.2.9.2.6.4 RX L2 Scratch Pad

When the RX L2 is disabled (PRUSS_MII_RT_RXCFG0/1 [RX_L2_EN] = 0), the RX L2 banks can be used as a generic scratch pad. In scratch pad mode, RX L2 Bank0 and RX L2 Bank1 operate like simple write/read memory mapped registers (MMRs). All XFR size and start operations are supported. RX_RESET has no effect in this mode. This mode is shown in Figure 30-105.

Figure 30-105. Scratch Pad Mode



30.2.9.3 PRU-ICSS MII RT Module Register Manual

This section describes the PRU-ICSS MII_RT module configuration registers.

30.2.9.3.1 PRUSS_MII_RT Instance Summary

Table 30-821. PRUSS_MII_RT Instance Summary

Module Name	Module Base Address L3_MAIN	Size
PRUSS1_MII_RT	0x4B23 2000	88 Bytes
PRUSS2_MII_RT	0x4B2B 2000	88 Bytes

30.2.9.3.2 PRUSS_MII_RT Registers

30.2.9.3.2.1 PRUSS_MII_RT Register Summary

Table 30-822. PRUSS1_MII_RT Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
PRUSS_MII_RT_RXCFG0	RW	32	0x0000 0000	0x4B23 2000
PRUSS_MII_RT_RXCFG1	RW	32	0x0000 0004	0x4B23 2004
PRUSS_MII_RT_TXCFG0	RW	32	0x0000 0010	0x4B23 2010
PRUSS_MII_RT_TXCFG1	RW	32	0x0000 0014	0x4B23 2014
PRUSS_MII_RT_TX_CRC0	R	32	0x0000 0020	0x4B23 2020
PRUSS_MII_RT_TX_CRC1	R	32	0x0000 0024	0x4B23 2024
PRUSS_MII_RT_TX_IPG0	RW	32	0x0000 0030	0x4B23 2030
PRUSS_MII_RT_TX_IPG1	RW	32	0x0000 0034	0x4B23 2034
PRUSS_MII_RT_PRS0	R	32	0x0000 0038	0x4B23 2038
PRUSS_MII_RT_PRS1	R	32	0x0000 003C	0x4B23 203C
PRUSS_MII_RT_RX_FRMS0	RW	32	0x0000 0040	0x4B23 2040
PRUSS_MII_RT_RX_FRMS1	RW	32	0x0000 0044	0x4B23 2044
PRUSS_MII_RT_RX_PCNT0	RW	32	0x0000 0048	0x4B23 2048
PRUSS_MII_RT_RX_PCNT1	RW	32	0x0000 004C	0x4B23 204C
PRUSS_MII_RT_RX_ERR0	RW	32	0x0000 0050	0x4B23 2050
PRUSS_MII_RT_RX_ERR1	RW	32	0x0000 0054	0x4B23 2054

Table 30-823. PRUSS2_MII_RT Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
PRUSS_MII_RT_RXCFG0	RW	32	0x0000 0000	0x4B2B 2000

Table 30-823. PRUSS2_MII_RT Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
PRUSS_MII_RT_RXCFG1	RW	32	0x0000 0004	0x4B2B 2004
PRUSS_MII_RT_TXCFG0	RW	32	0x0000 0010	0x4B2B 2010
PRUSS_MII_RT_TXCFG1	RW	32	0x0000 0014	0x4B2B 2014
PRUSS_MII_RT_TX_CRC0	R	32	0x0000 0020	0x4B2B 2020
PRUSS_MII_RT_TX_CRC1	R	32	0x0000 0024	0x4B2B 2024
PRUSS_MII_RT_TX_IPG0	RW	32	0x0000 0030	0x4B2B 2030
PRUSS_MII_RT_TX_IPG1	RW	32	0x0000 0034	0x4B2B 2034
PRUSS_MII_RT_PRS0	R	32	0x0000 0038	0x4B2B 2038
PRUSS_MII_RT_PRS1	R	32	0x0000 003C	0x4B2B 203C
PRUSS_MII_RT_RX_FRMS0	RW	32	0x0000 0040	0x4B2B 2040
PRUSS_MII_RT_RX_FRMS1	RW	32	0x0000 0044	0x4B2B 2044
PRUSS_MII_RT_RX_PCNT0	RW	32	0x0000 0048	0x4B2B 2048
PRUSS_MII_RT_RX_PCNT1	RW	32	0x0000 004C	0x4B2B 204C
PRUSS_MII_RT_RX_ERR0	RW	32	0x0000 0050	0x4B2B 2050
PRUSS_MII_RT_RX_ERR1	RW	32	0x0000 0054	0x4B2B 2054

30.2.9.3.2.2 PRUSS_MII_RT Register Description

Table 30-824. PRUSS_MII_RT_RXCFG0

Address Offset	0x0000 0000		
Physical Address	0x4B23 2000 0x4B23 2000 0x4B2B 2000 0x4B2B 2000	**Instance**	PRUSS1_MII_RT PRUSS1_MII_RT PRUSS2_MII_RT PRUSS2_MII_RT
Description	MII RXCFG 0 REGISTER This register contains the PRU0 RXCFG configuration variables (PRUSS_MII_RT_RXCFG0) for the RX path. PRUSS_MII_RT_RXCFG0 is attached to PRU0. PRUSS_MII_RT_RXCFG0 controls which RX port is attached to PRU0.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						RESERVED	RESERVED	RESERVED	RX_AUTO_FWD_PRE	RX_BYTE_SWAP	RX_L2_EN	RX_MUX_SEL	RX_CUT_PREAMBLE	RESERVED	RX_ENABLE

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x000000
9	RESERVED		R	0x0
8	RESERVED		R	0x0
7	RESERVED		R	0x0

Bits	Field Name	Description	Type	Reset
6	RX_AUTO_FWD_PRE	Enables auto-forward of received preamble. When enabled, this will forward the preamble nibbles including the SFD to the TX L1 FIFO that is attached to the PRU. First data byte seen by PRU R31 and/or RX L2 is destination address (DA). **Note: Odd number of preamble nibbles is supported in this mode. For example, 0x55D Note that new RX should only occur after the current TX completes** 0x0: Disable 0x1:Enable, it must disable RX_CUT_PREAMBLE and TX_AUTO_PREAMBLE.	RW	0x0
5	RX_BYTE_SWAP	Defines the order of Byte0/1 placement for RX R31 and RX L2. **Note: that if TX_AUTO_SEQUENCE enabled, this bit cannot get enable since TX_BYTE_SWAP on swaps the PRU output. This bit must be selected/updated when the port is disabled or there is no traffic.** 0x0: R31 [15:8]/RXL2 [15:8] = Byte1{Nibble3,Nibble2} R31[7:0]/RXL2 [7:0] = Byte0{Nibble1,Nibble0} 0x1: R31 [15:8]/RXL2 [15:8] = Byte0{Nibble1,Nibble0} R31[7:0]/RXL2 [7:0] = Byte1{Nibble3,Nibble2} Nibble0 is the first nibble received. Must be selected /updated when the port is disabled or no traffic It only effects R31 and RX L2 order	RW	0x0
4	RX_L2_EN	Enables RX L2 buffer. 0x0: Disable (RX L2 can function as generic scratch pad) 0x1: Enable	RW	0x0
3	RX_MUX_SEL	Selects receive data source. Typically, the setting for this will not be identical for the two MII receive configuration registers. 0x0: MII RX Data from Port 0 (default for PRUSS_MII_RT_RXCFG0) 0x1: MII RX Data from Port 1 (default for PRUSS_MII_RT_RXCFG1)	RW	0x0
2	RX_CUT_PREAMBLE	Removes received preamble. 0x0: All data from Ethernet PHY are passed on to PRU register. This assumes Ethernet PHY which does not shorten the preamble. 0x1: MII interface suppresses preamble and sync frame delimiter. First data byte seen by PRU register is DA	RW	0x0
1	RESERVED		R	0x0
0	RX_ENABLE	Enables the receive traffic currently selected by RX_MUX_SELECT. 0x0 Disable 0x1 Enable	RW	0x0

Table 30-825. Register Call Summary for Register PRUSS_MII_RT_RXCFG0

PRU-ICSS MII RT Module

- RX MII Submodule Overview: [12]
- Nibble and Byte Order: [13] [14] [15]
- PRU and MII Port Multiplexer: [16]
- RX L2 Scratch Pad: [17]
- PRUSS_MII_RT Register Summary: [18] [19]
- PRUSS_MII_RT Register Description: [20] [21] [22] [23] [24]

Table 30-826. PRUSS_MII_RT_RXCFG1

Address Offset	0x0000 0004		
Physical Address	0x4B23 2004	**Instance**	PRUSS1_MII_RT
	0x4B23 2004		PRUSS1_MII_RT
	0x4B2B 2004		PRUSS2_MII_RT
	0x4B2B 2004		PRUSS2_MII_RT

Table 30-826. PRUSS_MII_RT_RXCFG1 (continued)

Description	This register contains the PRU1 RXCFG configuration variables (PRUSS_MII_RT_RXCFG1) for the RX path. PRUSS_MII_RT_RXCFG1 is attached to PRU1. PRUSS_MII_RT_RXCFG1 controls which RX port is attached to PRU1
Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						RESERVED	RESERVED	RESERVED	RX_AUTO_FWD_PRE	RX_BYTE_SWAP	RX_L2_EN	RX_MUX_SEL	RX_CUT_PREAMBLE	RESERVED	RX_ENABLE

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x000000
9	RESERVED		R	0x0
8	RESERVED		R	0x0
7	RESERVED		R	0x0
6	RX_AUTO_FWD_PRE	Enables auto-forward of received preamble. When enabled, this will forward the preamble nibbles including the SFD to the TX L1 FIFO that is attached to the PRU. First data byte seen by PRU R31 and/or RX L2 is destination address (DA). **Note: Odd number of preamble nibbles is supported in this mode. For example, 0x55D Note that new RX should only occur after the current TX completes** 0x0: Disable 0x1: Enable, it must disable RX_CUT_PREAMBLE and TX_AUTO_PREAMBLE	R	0x0
5	RX_BYTE_SWAP	Defines the order of Byte0/1 placement for RX R31 and RX L2. **Note: If TX_AUTO_SEQUENCE is enabled, this bit cannot get enabled since TX_BYTE_SWAP on swaps the PRU output. This bit must be selected/updated when the port is disabled or there is no traffic.** 0x0: R31 [15:8]/RXL2 [15:8] = Byte1{Nibble3,Nibble2} R31[7:0]/RXL2 [7:0] = Byte0{Nibble1,Nibble0} 0x1: R31 [15:8]/RXL2 [15:8] = Byte0{Nibble1,Nibble0} R31[7:0]/RXL2 [7:0] = Byte1{Nibble3,Nibble2} Nibble0 is the first nibble received.	RW	0x0
4	RX_L2_EN	Enables RX L2 buffer. 0x0: Disable (RX L2 can function as generic scratch pad) 0x1: Enable	RW	0x0
3	RX_MUX_SEL	Selects receive data source. Typically, the setting for this will not be identical for the two MII receive configuration registers. 0x0: MII RX Data from Port 0 (default for PRUSS_MII_RT_RXCFG0) 0x1: MII RX Data from Port 1 (default for PRUSS_MII_RT_RXCFG1)	RW	0x1
2	RX_CUT_PREAMBLE	Removes received preamble. 0x0: All data from Ethernet PHY are passed on to PRU register. This assumes Ethernet PHY which does not shorten the preamble. 0x1: MII interface suppresses preamble and sync frame delimiter. First data byte seen by PRU register is destination address.	RW	0x0
1	RESERVED		R	0x0

Bits	Field Name	Description	Type	Reset
0	RX_ENABLE	Enables the receive traffic currently selected by RX_MUX_SELECT. 0x0: Disable 0x1: Enable	RW	0x0

Table 30-827. Register Call Summary for Register PRUSS_MII_RT_RXCFG1

PRU-ICSS MII RT Module

- RX MII Submodule Overview: [2]
- PRUSS_MII_RT Register Summary: [3] [4]
- PRUSS_MII_RT Register Description: [5] [6] [7] [8] [9]

Table 30-828. PRUSS_MII_RT_TXCFG0

Address Offset	0x0000 0010		
Physical Address	0x4B23 2010 0x4B23 2010 0x4B2B 2010 0x4B2B 2010	**Instance**	PRUSS1_MII_RT PRUSS1_MII_RT PRUSS2_MII_RT PRUSS2_MII_RT
Description	This register contains the configuration variables for the transmit path on the MII interface port 0. PRUSS_MII_RT_TXCFG0 is attached to Port TX0. PRUSS_MII_RT_TXCFG0 controls which PRU is selected for TX0		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				TX_CLK_DELAY				RESERVED								TX_START_DELAY								RESERVED							
RESERVED																RESERVED				RESERVED	RESERVED	TX_AUTO_SEQUENCE	TX_MUX_SEL	RESERVED				TX_BYTE_SWAP	TX_EN_MODE	TX_AUTO_PREAMBLE	TX_ENABLE

Bits	Field Name	Description	Type	Reset
31	RESERVED		R	0
30:28	TX_CLK_DELAY	In order to guarantee the MII_RT IO timing values published in the device data manual, the PRUSS_GICLK clock must be configured for 200MHz and TX_CLK_DELAY must be set to 6h.	RW	0x0
27:26	RESERVED		R	0x0

Bits	Field Name	Description	Type	Reset
25:16	TX_START_DELAY	Defines the minimum time interval (delay) between receiving the RXDV for the current frame and the start of the transmit interface sending data to the MII interface. Delay value is in units of MII_RT clock cycles, which uses the PRUSS_GICLK (default is 200MHz, or 5ns). Default TX_START_DELAY value is 320ns, which is optimized for minimum latency at 16 bit processing. Counter is started with RX_DV signal going active. Transmit interface stops sending data when no more data is written into transmit interface by PRU along with TX_EOF marker bit set. If the TX FIFO has data when the delay expires, then TX will start sending data. But if the TX FIFO is empty, it will not start until the TX FIFO is not empty. It is possible to overflow the TX FIFO with the max delay setting when auto-forwarding is enabled since the time delay is larger than the amount of data it needs to store. As long as TX L1 FIFO overflows, software will need to issue a TX_RESET to reset the TX FIFO. The total delay is 64-byte times (size of TX FIFO), but you need to allow delays for synchronization. Do to this fact, the maximum delay should be 80ns less when auto forwarding is enabled. Therefore, 0x3F0 is the maximum in this configuration.	RW	0x40
15:12	RESERVED		R	0x0
11	RESERVED		R	0x0
10	RESERVED		R	0x0
9	TX_AUTO_SEQUENCE	Enables transmit auto-sequence. Note the transmit data source is determined by TX_MUX_SEL setting. 0x0: Disable 0x1: Enable, transmit state machine based on events on receiver path that is connected to the respective transmitter. Also, the masking logic is disabled and only the MII data is used.	RW	0x0
8	TX_MUX_SEL	Selects transmit data source. The default/reset setting for TX Port 0 is 1. This setting permits MII TX Port 0 to receive data from PRU1 and the MII TX Port 1 which is connected to PRU0 by default. 0x0: Data from PRU0 (default for PRUSS_MII_RT_TXCFG1) 0x1: Data from PRU1 (default for PRUSS_MII_RT_TXCFG0)	RW	0x1
7:4	RESERVED		R	0x0

Bits	Field Name	Description	Type	Reset
3	TX_BYTE_SWAP	Defines the order of Byte0/1 placement for TX R30. This bit must be selected/updated when the port is disabled or there is no traffic. 0x0: If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 0, R30[15:8] = Byte1{Nibble3,Nibble2} R30[7:0] = Byte0{Nibble1,Nibble0} R30[31:24] = TX_MASK[15:8] R30[23:16] = TX_MASK[7:0] If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 1, R30[31:24] = Byte3{Nibble7,Nibble6} R30[23:16] = Byte2{Nibble5,Nibble4} R30[15:8] = Byte1{Nibble3,Nibble2} R30[7:0] = Byte0{Nibble1,Nibble0} 0x1: If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 0, R30[15:8] = Byte0{Nibble1,Nibble0} R30[7:0] = Byte1{Nibble3,Nibble2} R30[31:24] = TX_MASK[7:0] R30[23:16] = TX_MASK[15:8] If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 1, (ONLY SUPPORT 32bit push) R30[31:24] = Byte0{Nibble1,Nibble0} R30[23:16] = Byte1{Nibble3,Nibble2} R30[15:8] = Byte2{Nibble5,Nibble4} R30[7:0] = Byte3{Nibble7,Nibble6 **Note Nibble0 is the first nibble received.**	RW	0x0
2	TX_EN_MODE	Enables transmit self clear on TX_EOF event. **Note that iep.cmp[3] must be set before transmission will start for TX0, and iep_cmp[4] for TX1. This is a new dependency, in addition to TX L1 FIFO not empty and TX_START_DELAY expiration, to start transmission.** 0x0: Disable 0x1: Enable, TX_ENABLE will be clear for a TX_EOF event by itself.	RW	0x0
1	TX_AUTO_PREAMBLE	Transmit data auto-preamble. 0x0: PRU will provide full preamble 0x1: TX FIFO will insert pre-amble automatically **Note: the TX FIFO does not get preloaded with the preamble until the first write occurs. This can cause the latency to be larger the min latency.**	RW	0x0
0	TX_ENABLE	Enables transmit traffic on TX PORT. If TX_EN_MODE is set, then TX_ENABLE will self clear during a TX_EOF event. Note Software can use this to pre-fill the TX FIFO and then start the TX frame during non-ECS operations. 0x0: TX PORT is disabled/stopped immediately 0x1: TX PORT is enabled and the frame will start once the IPG counter expired and TX Start Delay counter has expired	RW	0x0

Table 30-829. Register Call Summary for Register PRUSS_MII_RT_TXCFG0

PRU-ICSS MII RT Module

- Introduction: [7]
- TX Data Path Options to TX L1 FIFO: [12]
- PRU R31 Command Interface:
- Nibble and Byte Order: [15] [16] [19]
- PRUSS_MII_RT Register Summary: [22] [23]
- PRUSS_MII_RT Register Description: [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35]

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRU-ICSS Industrial Ethernet Timer Features:

Table 30-830. PRUSS_MII_RT_TXCFG1

Address Offset	0x0000 0014		
Physical Address	0x4B23 2014 0x4B23 2014 0x4B2B 2014 0x4B2B 2014	**Instance**	PRUSS1_MII_RT PRUSS1_MII_RT PRUSS2_MII_RT PRUSS2_MII_RT
Description	MII TXCFG 1 REGISTER This register contains the configuration variables for the transmit path on the MII interface port 1. PRUSS_MII_RT_TXCFG1 is attached to Port TX1. PRUSS_MII_RT_TXCFG1 controls which PRU is selected for TX1		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		TX_CLK_DELAY		RESERVED				TX_START_DELAY								RESERVED				RESERVED	RESERVED	TX_AUTO_SEQUENCE	TX_MUX_SEL	RESERVED				TX_BYTE_SWAP	TX_EN_MODE	TX_AUTO_PREAMBLE	TX_ENABLE

Bits	Field Name	Description	Type	Reset
31	RESERVED		R	0
30:28	TX_CLK_DELAY	In order to guarantee the MII_RT IO timing values published in the device data manual, the PRUSS_GICLK clock must be configured for 200MHz and TX_CLK_DELAY must be set to 6h.	RW	0x0
27:26	RESERVED		R	0x0
25:16	TX_START_DELAY	Defines the minimum time interval (delay) between receiving the RXDV for the current frame and the start of the transmit interface sending data to the MII interface. Delay value is in units of MII_RT clock cycles, which uses the PRUSS_GICLK (default is 200MHz, or 5ns). Default TX_START_DELAY value is 320ns, which is optimized for minimum latency at 16 bit processing. Counter is started with RX_DV signal going active. Transmit interface stops sending data when no more data is written into transmit interface by PRU along with TX_EOF marker bit set. If the TX FIFO has data when the delay expires, then TX will start sending data. But if the TX FIFO is empty, it will not start until the TX FIFO is not empty. It is possible to overflow the TX FIFO with the max delay setting when auto-forwarding is enabled since the time delay is larger than the amount of data it needs to store. As long as TX L1 FIFO overflows, software will need to issue a TX_RESET to reset the TX FIFO. The total delay is 64-byte times (size of TX FIFO), but you need to allow delays for synchronization. Do to this fact, the maximum delay should be 80ns less when auto forwarding is enabled. Therefore, 0x3F0 is the maximum in this configuration.	RW	0x40
15:12	RESERVED		R	0x0
11	RESERVED		R	0x0
10	RESERVED		R	0x0

Bits	Field Name	Description	Type	Reset
9	TX_AUTO_SEQUENCE	Enables transmit auto-sequence. Note the transmit data source is determined by TX_MUX_SEL setting. 0x0: Disable 0x1: Enable, transmit state machine based on events on receiver path that is connected to the respective transmitter.TX data from PRU1 is selected Also, the masking logic is disabled and only the MII data is used.	RW	0x0
8	TX_MUX_SEL	Selects transmit data source. The default/reset setting for TX Port 0 is 1. This setting permits MII TX Port 0 to receive data from PRU1 and the MII TX Port 1 which is connected to PRU0 by default. 0x0: Data from PRU0 (default for PRUSS_MII_RT_TXCFG1) 0x1: Data from PRU1 (default for PRUSS_MII_RT_TXCFG0)	RW	0x0
7:4	RESERVED		R	0x0
3	TX_BYTE_SWAP	Defines the order of Byte0/1 placement for TX R30. This bit must be selected/updated when the port is disabled or there is no traffic. 0x0: If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 0, R30[15:8] = Byte1{Nibble3,Nibble2} R30[7:0] = Byte0{Nibble1,Nibble0} R30[31:24] = TX_MASK[15:8] R30[23:16] = TX_MASK[7:0] If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 1, R30[31:24] = Byte3{Nibble7,Nibble6} R30[23:16] = Byte2{Nibble5,Nibble4} R30[15:8] = Byte1{Nibble3,Nibble2} R30[7:0] = Byte0{Nibble1,Nibble0} 0x1: If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 0, R30[15:8] = Byte0{Nibble1,Nibble0} R30[7:0] = Byte1{Nibble3,Nibble2} R30[31:24] = TX_MASK[7:0] R30[23:16] = TX_MASK[15:8] If PRUSS_MII_RT_TXCFG0/1 [TX_32_MODE_EN] = 1, (ONLY SUPPORT 32bit push) R30[31:24] = Byte0{Nibble1,Nibble0} R30[23:16] = Byte1{Nibble3,Nibble2} R30[15:8] = Byte2{Nibble5,Nibble4} R30[7:0] = Byte3{Nibble7,Nibble6 **Note Nibble0 is the first nibble received.**	RW	0x0
2	TX_EN_MODE	Enables transmit self clear on TX_EOF event. **Note that iep.cmp[3] must be set before transmission will start for TX0, and iep_cmp[4] for TX1. This is a new dependency, in addition to TX L1 FIFO not empty and TX_START_DELAY expiration, to start transmission.** 0x0: Disable 0x1: Enable, TX_ENABLE will be clear for a TX_EOF event by itself.	RW	0x0
1	TX_AUTO_PREAMBLE	Transmit data auto-preamble. 0x0: PRU will provide full preamble 0x1: TX FIFO will insert pre-amble automatically **Note: the TX FIFO does not get preloaded with the preamble until the first write occurs. This can cause the latency to be larger the min latency.**	RW	0x0
0	TX_ENABLE	Enables transmit traffic on TX PORT. If TX_EN_MODE is set, then TX_ENABLE will self clear during a TX_EOF event. Note Software can use this to pre-fill the TX FIFO and then start the TX frame during non-ECS operations. 0x0: TX PORT is disabled/stopped immediately 0x1: TX PORT is enabled and the frame will start once the IPG counter expired and TX Start Delay counter has expired	RW	0x0

Table 30-831. Register Call Summary for Register PRUSS_MII_RT_TXCFG1

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]
- PRUSS_MII_RT Register Description: [4] [5] [6] [7]

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRU-ICSS Industrial Ethernet Timer Features:

Table 30-832. PRUSS_MII_RT_TX_CRC0

Address Offset	0x0000 0020		
Physical Address	0x4B23 2020 0x4B23 2020 0x4B2B 2020 0x4B2B 2020	**Instance**	PRUSS1_MII_RT PRUSS1_MII_RT PRUSS2_MII_RT PRUSS2_MII_RT
Description	MII TXCRC 0 REGISTER It contains CRC32 which PRU0 reads		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX_CRC																															

Bits	Field Name	Description	Type	Reset
31:0	TX_CRC	FCS (CRC32) data can be read by PRU for diagnostics. It is only valid after 6 clocks after a TX_CRC_HIGH command is given.	R	0x0

Table 30-833. Register Call Summary for Register PRUSS_MII_RT_TX_CRC0

PRU-ICSS MII RT Module

- CRC Computation: [2]
- PRU R31 Command Interface: [3]
- PRUSS_MII_RT Register Summary: [4] [5]

Table 30-834. PRUSS_MII_RT_TX_CRC1

Address Offset	0x0000 0024		
Physical Address	0x4B23 2024 0x4B23 2024 0x4B2B 2024 0x4B2B 2024	**Instance**	PRUSS1_MII_RT PRUSS1_MII_RT PRUSS2_MII_RT PRUSS2_MII_RT
Description	MII TXCRC 1 REGISTER It contains CRC32 which PRU1 reads		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX_CRC																															

Bits	Field Name	Description	Type	Reset
31:0	TX_CRC	FCS (CRC32) data can be read by PRU for diagnostics. It is only valid after 6 clocks after a TX_CRC_HIGH command is given.	R	0x0

Table 30-835. Register Call Summary for Register PRUSS_MII_RT_TX_CRC1

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]

Table 30-836. PRUSS_MII_RT_TX_IPG0

Address Offset	0x0000 0030		
Physical Address	0x4B23 2030 0x4B23 2030 0x4B2B 2030 0x4B2B 2030	**Instance**	PRUSS1_MII_RT PRUSS1_MII_RT PRUSS2_MII_RT PRUSS2_MII_RT
Description	MII TXIPG 0 REGISTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								TX_IPG							

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x0
9:0	TX_IPG	Defines the minimum of transmit Inter Packet Gap (IPG) which is the number of PRUSS_GICLK cycles between the de-assertion of TX_EN and the assertion of TX_EN. The start of the TX will get delayed when the incoming packet IPG is less than defined minimum value. In general, software should program in increments of 8, 40ns to insure the extra delays takes effect.	RW	0x28

Table 30-837. Register Call Summary for Register PRUSS_MII_RT_TX_IPG0

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]

Table 30-838. PRUSS_MII_RT_TX_IPG1

Address Offset	0x0000 0034		
Physical Address	0x4B23 2034 0x4B23 2034 0x4B2B 2034 0x4B2B 2034	**Instance**	PRUSS1_MII_RT PRUSS1_MII_RT PRUSS2_MII_RT PRUSS2_MII_RT
Description	MII TXIPG 1 REGISTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								TX_IPG							

Bits	Field Name	Description	Type	Reset
31:10	RESERVED		R	0x0
9:0	TX_IPG	Defines the minimum of transmit Inter Packet Gap (IPG) which is the number of PRUSS_GICLK cycles between the de-assertion of TX_EN and the assertion of TX_EN. The start of the TX will get delayed when the incoming packet IPG is less than defined minimum value. In general, software should program in increments of 8, 40ns to insure the extra delays takes effect.	RW	0x28

Table 30-839. Register Call Summary for Register PRUSS_MII_RT_TX_IPG1

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]

Table 30-840. PRUSS_MII_RT_PRS0

Address Offset	0x0000 0038		
Physical Address	0x4B23 2038 0x4B23 2038 0x4B2B 2038 0x4B2B 2038	**Instance**	PRUSS1_MII_RT PRUSS1_MII_RT PRUSS2_MII_RT PRUSS2_MII_RT
Description	MII PORT STATUS 0 REGISTER		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														MII_CRS	MII_COL

Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0
1	MII_CRS	Read the current state of pr1_mii0_crs	R	0x0
0	MII_COL	Read the current state of pr1_mii0_col	R	0x0

Table 30-841. Register Call Summary for Register PRUSS_MII_RT_PRS0

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]

Table 30-842. PRUSS_MII_RT_PRS1

Address Offset	0x0000 003C		
Physical Address	0x4B23 203C 0x4B23 203C 0x4B2B 203C 0x4B2B 203C	**Instance**	PRUSS1_MII_RT PRUSS1_MII_RT PRUSS2_MII_RT PRUSS2_MII_RT
Description	MII PORT STATUS 1 REGISTER		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														MII_CRS	MII_COL

Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0
1	MII_CRS	Read the current state of pr1_mii1_crs	R	0x0
0	MII_COL	Read the current state of pr1_mii1_col	R	0x0

Table 30-843. Register Call Summary for Register PRUSS_MII_RT_PRS1

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]

Table 30-844. PRUSS_MII_RT_RX_FRMS0

Address Offset	0x0000 0040		
Physical Address	0x4B23 2040 0x4B23 2040 0x4B2B 2040 0x4B2B 2040	**Instance**	PRUSS1_MII_RT PRUSS1_MII_RT PRUSS2_MII_RT PRUSS2_MII_RT
Description	MII RXFRMS 0 REGISTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX_MAX_FRM																RX_MIN_FRM															

Bits	Field Name	Description	Type	Reset
31:16	RX_MAX_FRM	Defines the maximum received frame count. If the total byte count of received frame is more than defined value, RX_MAX_FRM_ERR will get set. 0x0 = 1 byte after SFD and including CRC N= N+1 bytes after SFD and including CRC Note if the incoming frame is truncated at the marker, RX_CRC and RX_NIBBLE_ODD will not get asserted.	RW	0x5F1
15:0	RX_MIN_FRM	Defines the minimum received frame count. If the total byte count of received frame is less than defined value, RX_MIN_FRM_ERR will get set. 0x0 = 1 byte after SFD and including CRC N=N+1 bytes after SFD and including CRC	RW	0x3F

Table 30-845. Register Call Summary for Register PRUSS_MII_RT_RX_FRMS0

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]

Table 30-846. PRUSS_MII_RT_RX_FRMS1

Address Offset	0x0000 0044		
Physical Address	0x4B23 2044 0x4B23 2044 0x4B2B 2044 0x4B2B 2044	**Instance**	PRUSS1_MII_RT PRUSS1_MII_RT PRUSS2_MII_RT PRUSS2_MII_RT
Description	MII RXFRMS 1 REGISTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX_MAX_FRM																RX_MIN_FRM															

Bits	Field Name	Description	Type	Reset
31:16	RX_MAX_FRM	Defines the maximum received frame count. If the total byte count of the received frame is more than defined value, RX_MAX_FRM_ERR will get set. 0x0 = 1 byte after SFD and including CRC N= N+1 bytes after SFD and including CRC Note if the incoming frame is truncated at the marker, RX_CRC and RX_NIBBLE_ODD will not get asserted.	RW	0x5F1
15:0	RX_MIN_FRM	Defines the minimum received frame count. If the total byte count of received frame is less than defined value, RX_MIN_FRM_ERR will get set. 0x0 = 1 byte after SFD and including CRC N=N+1 bytes after SFD and including CRC	RW	0x3F

Table 30-847. Register Call Summary for Register PRUSS_MII_RT_RX_FRMS1

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]

Table 30-848. PRUSS_MII_RT_RX_PCNT0

Address Offset	0x0000 0048		
Physical Address	0x4B23 2048 0x4B23 2048 0x4B2B 2048 0x4B2B 2048	**Instance**	PRUSS1_MII_RT PRUSS1_MII_RT PRUSS2_MII_RT PRUSS2_MII_RT
Description	MII RXPCNT 0 REGISTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								RX_MAX_PCNT				RX_MIN_PCNT			

Bits	Field Name	Description	Type	Reset
31:8	RESERVED		R	0x000000
7:4	RX_MAX_PCNT	Defines the maximum number of nibbles until the start of frame delimiter (SFD) event occurred (i.e. matches 0xD5). RX_MAX_PRE_COUNT_ERR will be set if the preamble counts more than the value of RX_MAX_PCNT. If the SFD does not occur within 16 nibbles, the error will assert and the incoming frame will be truncated. 0x0: Disabled 0x1: Reserved 0x2: 4th nibble needs to have built 0xD5 0xe: 16th nibble needs to have built 0xD5 **Note the 16th nibble is transmitted.** **Note for firmware enabling preamble error detection, it is recommended to keep RX_MAX_PCNT disabled (0x0). Otherwise, hardware can truncate a valid frame with too long of a preamble.**	RW	0xE
3:0	RX_MIN_PCNT	Defines the minimum number of nibbles until the start of frame delimiter (SFD) event occurred, which is matched the value 0xD5. RX_MIN_PRE_COUNT_ERR will be set if the preamble counts less than the value of RX_MIN_PCNT. 0x0 Disabled 0x1 1 0x5 before 0xD5 0x2 2 0x5 before 0xD5 N min of N 0x5 before 0xD5 **Note it does not need to be "0x5"**	RW	0x1

Table 30-849. Register Call Summary for Register PRUSS_MII_RT_RX_PCNT0

PRU-ICSS MII RT Module

- RX MII Submodule Overview: [2]
- PRUSS_MII_RT Register Summary: [3] [4]

Table 30-850. PRUSS_MII_RT_RX_PCNT1

Address Offset	0x0000 004C		
Physical Address	0x4B23 204C 0x4B23 204C 0x4B2B 204C 0x4B2B 204C	**Instance**	PRUSS1_MII_RT PRUSS1_MII_RT PRUSS2_MII_RT PRUSS2_MII_RT
Description	MII RXPCNT 1 REGISTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								RX_MAX_PCNT				RX_MIN_PCNT			

Bits	Field Name	Description	Type	Reset
31:8	RESERVED		R	0x000000
7:4	RX_MAX_PCNT	Defines the maximum number of nibbles until the start of frame delimiter (SFD) event occurred (i.e. matches 0xD5). RX_MAX_PRE_COUNT_ERR will be set if the preamble counts more than the value of RX_MAX_PCNT. If the SFD does not occur within 16 nibbles, the error will assert and the incoming frame will be truncated. 0x0: Disabled 0x1: Reserved 0x2: 4th nibble needs to have built 0xD5 0xe: 16th nibble needs to have built 0xD5 **Note the 16th nibble is transmitted** **Note for firmware enabling preamble error detection, it is recommended to keep RX_MAX_PCNT disabled (0x0). Otherwise, hardware can truncate a valid frame with too long of a preamble.**	RW	0xE
3:0	RX_MIN_PCNT	Defines the minimum number of nibbles until the start of frame delimiter (SFD) event occurred, which is matched the value 0xD5. RX_MIN_PRE_COUNT_ERR will be set if the preamble counts less than the value of RX_MIN_PCNT. 0x0 Disabled 0x1: 1 0x5 before 0xD5 0x2: 2 0x5 before 0xD5 N: N 0x5 before 0xD5 **Note it does not need to be "0x5"**	RW	0x1

Table 30-851. Register Call Summary for Register PRUSS_MII_RT_RX_PCNT1

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]

Table 30-852. PRUSS_MII_RT_RX_ERR0

Address Offset	0x0000 0050		
Physical Address	0x4B23 2050 0x4B23 2050 0x4B2B 2050 0x4B2B 2050	**Instance**	PRUSS1_MII_RT PRUSS1_MII_RT PRUSS2_MII_RT PRUSS2_MII_RT
Description	MII RXERR 0 REGISTER		
Type	RWr1Clr		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												RX_MAX_FRM_ERR	RX_MIN_FRM_ERR	RX_MAX_PCNT_ERR	RX_MIN_PCNT_ERR

Bits	Field Name	Description	Type	Reset
31:4	RESERVED		R	0x00000
3	RX_MAX_FRM_ERR	Error status of received frame is more than the value of RX_MAX_FRM. 0x0: No error occurred 0x1: Error occurred Write 1 to Clear	RWr1Clr	0x0
2	RX_MIN_FRM_ERR	Error status of received frame is less than the value of RX_MIN_FRM_CNT. 0x0: No error occurred 0x1: Error occurred Write 1 to Clear	RWr1Clr	0x0
1	RX_MAX_PCNT_ERR	Error status of received preamble nibble is more than the value of RX_MAX_PCNT. 0x0: No error occurred 0x1: Error occurred Write 1 to Clear	RWr1Clr	0x0
0	RX_MIN_PCNT_ERR	Error status of received preamble nibble is less than the value of RX_MIN_PCNT. 0x0: No error occurred 0x1: Error occurred Write 1 to Clear	RWr1Clr	0x0

Table 30-853. Register Call Summary for Register PRUSS_MII_RT_RX_ERR0

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]

Table 30-854. PRUSS_MII_RT_RX_ERR1

Address Offset	0x0000 0054		
Physical Address	0x4B23 2054 0x4B23 2054 0x4B2B 2054 0x4B2B 2054	**Instance**	PRUSS1_MII_RT PRUSS1_MII_RT PRUSS2_MII_RT PRUSS2_MII_RT
Description	MII RXERR 1 REGISTER		
Type	RWr1Clr		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												RX_MAX_FRM_ERR	RX_MIN_FRM_ERR	RX_MAX_PCNT_ERR	RX_MIN_PCNT_ERR

Bits	Field Name	Description	Type	Reset
31:4	RESERVED		R	0x00000
3	RX_MAX_FRM_ERR	Error status of received frame is more than the value of RX_MAX_FRM_CNT. 0x0: No error occurred 0x1: Error occurred Write 1 to Clear	RW	0x0
2	RX_MIN_FRM_ERR	Error status of received frame is less than the value of RX_MIN_FRM. 0x0: No error occurred 0x1: Error occurred Write 1 to Clear	RW	0x0
1	RX_MAX_PCNT_ERR	Error status of received preamble nibble is more than the value of RX_MAX_PCNT. 0x0: No error occurred 0x1: Error occurred Write 1 to Clear	RW	0x0
0	RX_MIN_PCNT_ERR	Error status of received preamble nibble is less than the value of RX_MIN_PCNT. 0x0: No error occurred 0x1: Error occurred Write 1 to Clear	RW	0x0

Table 30-855. Register Call Summary for Register PRUSS_MII_RT_RX_ERR1

PRU-ICSS MII RT Module

- PRUSS_MII_RT Register Summary: [2] [3]

30.2.10 PRU-ICSS MII MDIO Module

This section describes the PRU-ICSS1 and PRU-ICSS2 integrated **MII management interface module - MII_MDIO** module (PRUSS1_MII_MDIO / PRUSS2_MII_MDIO, respectively).

30.2.10.1 PRU-ICSS MII MDIO Overview

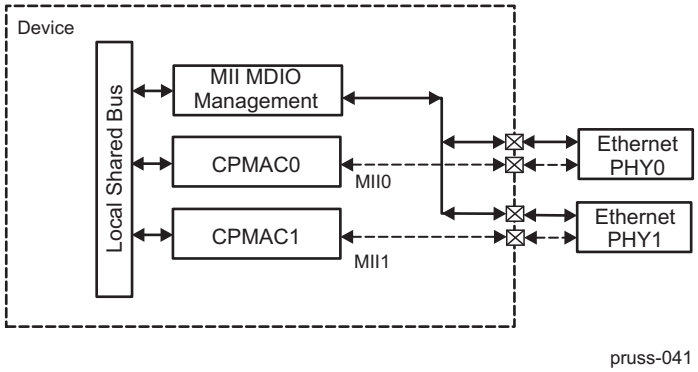
The following features are supported:

- Supports up to 32 PHY addresses.
- Two user access registers to control and monitor up to two PHYs simultaneously.
- Slave interface for configuration and control (MII RT MDIO CFG)

The PRU-ICSS MII MDIO management I/F module implements the **802.3 serial management interface** to interrogate and control two Ethernet PHYs simultaneously using a shared two-wire bus. Figure 1 shows a device with two MACs, each connected to an Ethernet PHY, being managed by the MII interface module using a shared bus.

The Figure 30-106 gives an overview of the MII MDIO management interface.

Figure 30-106. Device PRU-ICSS MII MDIO Management Interface Overview



30.2.10.2 PRU-ICSS MII MDIO Functional Description

The MII Management interface incorporates:

- **MDIO Registers** - Host interaction with this module is facilitated through the registers in this block.
- **Control and Schedule** - The control and register logic in the MII Management Interface module contain the state machine and scheduling logic which control the wire side operation.
- **MDIO Interface** - The MDIO interface block provides the serial interface to the MDIO interface.

The MDIO logic is fully synchronous to the PRU-ICSS local shared bus clock.

30.2.10.2.1 MII MDIO Management Interface Frame Formats

The below Table 30-856 shows the read and write format of the 32-bit MII Management interface frames, respectively.

Table 30-856. MII MDIO Frame Formats

Pre-amble	Start Delimiter	Operation Code	PHY Address	Register Address	Turnaround	Data
MDIO Read Frame Format						
0xFFFFFFFF	01	10	AAAAA	RRRRR	Z0	DDDD.DDDD.DDDD.DDDD
MDIO Write Frame Format						
0xFFFFFFFF	01	00	AAAAA	RRRRR	10	DDDD.DDDD.DDDD.DDDD

The default or idle state of the two wire serial interface is a logic one. All tri-state drivers should be disabled and the PHY's pull-up resistor will pull the **MDIO** line to a logic one. Prior to initiating any other transaction, the station management entity shall send a preamble sequence of 32 contiguous logic one bits on the **MDIO** line with 32 corresponding cycles on **MDCLK** to provide the PHY with a pattern that it can use to establish synchronization. A PHY shall observe a sequence of 32 contiguous logic one bits on **MDIO** with 32 corresponding **MDCLK** cycles before it responds to any other transaction.

Preamble
The start of a frame is indicated by a preamble, which consists of a sequence of 32 contiguous bits all of which are a "1". This sequence provides the Ethernet PHY a pattern to use to establish synchronization.

Start Delimiter
The preamble is followed by the start delimiter which is indicated by a "01" pattern. The pattern assures transitions from the default logic one state to zero and back to one.

Operation Code
The operation code for a read is "10", while the operation code for a write is a "00".

Ethernet PHY Address
The PHY address is 5 bits allowing 32 unique values. The first bit transmitted is the MSB of the PHY address.

Register Address
The Register address is 5 bits allowing 32 registers to be addressed within each PHY. Refer to the 10/100 PHY address map for addresses of individual registers.

Turnaround
An idle bit time during which no device actively drives the MDIO signal shall be inserted between the register address field and the data field of a read frame in order to avoid contention. During a read frame, the PHY shall drive a zero bit onto MDIO for the first bit time following the idle bit and preceding the Data field. During a write frame, this field shall consist of a one bit followed by a zero bit.

Data
The Data field is 16 bits. The first bit transmitted and received is the MSB of the data word.

The Table 30-857 shows the PRU-ICSS1 / PRU-ICSS2 MII MDIO signals and their availability at the device boundary.

Table 30-857. PRU-ICSS MII MDIO Control and Interface Signals

MDIO Control Signals			
Pin Name	**Type**	**Available as device I/O**	**Function**
MDIO_LINKINT[1:0]	O	N.A.	Serial interface link change interrupt. Indicates a change in the state of the PHY link.
MDIO_USERINT[1:0]	O	N.A.	Serial interface user command event complete interrupt.
MDIO Interface Signals			
Pin Name	**Type**	**Available as device I/O**	**Function**
MDIO_I	I	device bidi **pr1_mdio_data** and pr2_mdio_mdclk pin in input mode	Serial data input
MDIO_O	O	device bidi **pr1_mdio_data** pr2_mdio_mdclk pin in output mode	Serial data output
MDIO_OE_N	O	N.A.	Serial data output enable. Asserted "0" when data output is valid
MDCLK_O	O	device output - **pr1_mdio_mdclk** pr2_mdio_mdclk	Serial clock output
MLINK_I[1:0]	I	N.A.	Optional link status inputs from PHY. Each input is connected to a single PHY. Unused inputs are tied '0'.

30.2.10.2.2 PRU-ICSS MII MDIO Interractions

The MII Management I/F will remain idle until enabled by setting the **enable** bit in the **MDIO Control** register. The MII Management I/F will then continuously poll the link status from within the Generic Status Register of all possible 32 PHY addresses in turn recording the results in the **MDIOLink** register. The link status of two of the 32 possible PHY addresses can also be determined using the **MLINK** pin inputs. The linksel bit in the **MDIOUserPhySel** register determines the status input that is used. A change in the link status of the two PHYs being monitored will set the appropriate bit in the **MDIOLinkIntRaw** register and the **MDIOLinkIntMasked** register, if enabled by the **linkint_enable** bit in the **MDIOUserPhySel** register.

The **MDIOAlive** register is updated by the MII Management I/F module if the PHY acknowledged the read of the generic status register. In addition, any PHY register read transactions initiated by the host also cause the **MDIOAlive** register to be updated.

At any time, the host can define a transaction for the MII Management interface module to undertake using the **data**, **Ethernet PHY address**, **register address**, and **write** fields in a **MDIOUserAccess** register. When the host sets the go bit in this register, the MII Management interface module will begin the transaction without any further intervention from the host. Upon completion, the MII Management interface will clear the go bit and set the **userintraw** bit in the **MDIOUserIntRaw** register corresponding to the **MDIOUserAccess** register being used. The corresponding bit in the **MDIOUserIntMasked** register may also be set depending on the mask setting in the **MDIOUserIntMaskSet** and **MDIOUserIntMaskClr** registers. A round-robin arbitration scheme is used to schedule transactions which may queued by the host in different **MDIOUserAccess** registers. The host should check the status of the go bit in the **MDIOUserAccess** register before initiating a new transaction to ensure that the previous transaction has completed. The host can use the ack bit in the **MDIOUserAccess** register to determine the status of a read transaction.

It is necessary for software to use the MII Management interface module to setup the autonegotiation parameters of each PHY attached to a MAC port, retrieve the negotiation results, and setup the **MACControl** register in the corresponding MAC.

30.2.10.2.3 PRU-ICSS MII MDIO Interrupts

The MII Management interface state machine will assert the **MDIO_LINKINT** signals if there is a change in the link state of the Ethernet PHY corresponding to the address in the **phyadr_mon** field of the **MDIOUserPhySel** register and the corresponding **linkint_enable** bit is set. The **MDIO_LINKINT** event is also captured in the **MDIOLinkIntMasked** register. **MDIO_LINKINT[0]** and **MDIO_LINKINT[1]** correspond to the **MDIOUserPhySel0** and **MDIOUserPhySel1** registers, respectively.

When the "GO" bit in the **MDIOUserAccess** registers transitions from ‘1’ to ‘0’, indicating the completion of a user access, and the corresponding **userintmaskset** bit in the

MDIOUserIntMaskSet register is set, the **MDIO_USERINT** signal is asserted ‘1’. The **MDIO_USERINT** event is also captured in the **MDIOUserIntMasked** register. **MDIO_USERINT[0]** and **MDIO_USERINT[1]** correspond to the **MDIOUserAccess0** and **MDIOUserAccess1** registers, respectively.

30.2.10.3 PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface

To facilitate transmission and reception of serial management frames, the host has to perform the following operations:

- Configure the **preamble** and **clkdiv** fields in the **MDIOControl** register.
- Enable the VBUS MII management module by setting the **enable** bit in the MDIOControl register. If Byte access is being used, the **enable** bit should be written last.
- The **MDIOAlive** register can be read after a delay to determine which Ethernet PHYs responded.
- Setup the appropriate PHY addresses in the **MDIOUserPhySel** registers.
- Setup the appropriate **linkint_enable** bit in the **MDIOUserPhySel** register.
- Setup the appropriate **linksel** fields in the **MDIOUserPhySel** register.
- Setup the appropriate **userintmaskset** bits in the **MDIOUserIntMaskSet** register.
- **To write to**an Ethernet PHY register the host should first check to ensure that the go bit in a **MDIOUserAcess** register is cleared. The **GO**, **write**, **regadr**, **phyadr** and data fields in that

MDIOUserAccess register can then be updated to be appropriate value. If byte access is being used, the go bit should be written last. The write operation to the PHY will be scheduled and completed by the module. Completion of the write operation can be determined by examining the go bit in the **MDIOUserAccess** register. It also results in a transition on the appropriate **MDIO_INT** signal and the corresponding bit in the **MDIOUserIntMasked** register based on the setting of the **MDIOUserIntMaskSet** register.

- **To read from** an Ethernet PHY register the host should first check to ensure that the "GO" bit in a **MDIOUserAccess** register bit is cleared. The **GO**, **regadr**, and **phyadr** fields in that **MDIOUserAccess** register can then be updated to the appropriate value The read data value will be available in the data field of the MDIOUserAccess register after the module completes the read operation on the serial bus. The completion of the read operation can be determined by examining the "GO" and "ACK" bits in the **MDIOUserAccess** register. It also results in a transition on the appropriate **MDIO_INT** signal and the corresponding bit in the **MDIOUserIntMasked** register based on the setting of the **MDIOUserIntMaskSet** register.
- The module de-asserts the **MDIO_USERINT** signal when the host writes to the appropriate **"userintmasked"** bit in the **MDIOUserIntMasked** register or the **userintraw** bit in the **MDIOUserIntRaw** register.
- The host can poll the **MDIOLink** register periodically or use the **MDIO_LINKINT** signals to determine the state of the serial interface to a particular Ethernet PHY.
- The module de-asserts the **MDIO_LINKINT** when the host writes to the appropriate linkintraw bit in the **MDIOLinkIntRaw** register or the **linkintmasked** bit in the **MDIOLinkIntMasked** register.

Table 30-858. Summary of the PRU-ICSS MII MDIO Functional Registers

Address Offset	Register Mnemonic	Register Name	Register Purpose
0x04	**MDIOControl**	PRUSS_MII_MDIO_CONTROL	Module control register
0x08	**MDIOAlive**	PRUSS_MII_MDIO_ALIVE	Ethernet PHY acknowledge status register
0x0c	**MDIOLink**	PRUSS_MII_MDIO_LINK	Ethernet PHY link status register
0x10	**MDIOLinkIntRaw**	PRUSS_MII_MDIO_LINKINTRAW	Link status change interrupt register (raw value)
0x14	**MDIOLinkIntMasked**	PRUSS_MII_MDIO_LINKINTMASKED	Link status change interrupt register (masked value)
0x18-0x1c	**Reserved**	-	Reserved
0x20	**MDIOUserIntRaw**	PRUSS_MII_MDIO_USERINTRAW	User command complete interrupt register (raw value)
0x24	**MDIOUserIntMasked**	PRUSS_MII_MDIO_USERINTMASKED	User command complete interrupt register (masked value)
0x28	**MDIOUserIntMaskSet**	PRUSS_MII_MDIO_USERINTMASKSET	User interrupt mask set register
0x2c	**MDIOUserIntMaskClr**	PRUSS_MII_MDIO_USERINTMASKCLR	User interrupt mask clear register
0x30 – 0x7c	**Reserved**	-	Reserved
0x80	**MDIOUserAccess0**	PRUSS_MII_MDIO_USERACCESS0	User access register 0
0x84	**MDIOUserPhySel0**	PRUSS_MII_MDIO_USERPHYSEL0	User PHY select register 0
0x88	**MDIOUserAccess1**	PRUSS_MII_MDIO_USERACCESS1	User access register 1
0x8c	**MDIOUserPhySel1**	PRUSS_MII_MDIO_USERPHYSEL1	User PHY select register 1

Table 30-858. Summary of the PRU-ICSS MII MDIO Functional Registers (continued)

Address Offset	Register Mnemonic	Register Name	Register Purpose
0x90 – 0xff	**Reserved**	-	Reserved

30.2.10.4 PRU-ICSS MII MDIO Module Register Manual

30.2.10.4.1 PRUSS_MII_MDIO Instance Summary

Table 30-859. PRUSS_MII_MDIO Instance Summary

Module Name	Base Address	Size
PRUSS1_MII_MDIO	0x4B23 2400	144 Bytes
PRUSS2_MII_MDIO	0x4B2B 2400	144 Bytes

30.2.10.4.2 PRUSS_MII_MDIO Registers

30.2.10.4.2.1 PRUSS_MII_MDIO Register Summary

Table 30-860. PRUSS1_MII_MDIO Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
PRUSS_MII_MDIO_VER	R	32	0x0000 0000	0x4B23 2400
PRUSS_MII_MDIO_CONTROL	RW	32	0x0000 0004	0x4B23 2404
PRUSS_MII_MDIO_ALIVE	RW	32	0x0000 0008	0x4B23 2408
PRUSS_MII_MDIO_LINK	R	32	0x0000 000C	0x4B23 240C
PRUSS_MII_MDIO_LINKINTRAW	RW	32	0x0000 0010	0x4B23 2410
PRUSS_MII_MDIO_LINKINTMASKED	RW	32	0x0000 0014	0x4B23 2414
PRUSS_MII_MDIO_USERINTRAW	RW	32	0x0000 0020	0x4B23 2420
PRUSS_MII_MDIO_USERINTMASKED	RW	32	0x0000 0024	0x4B23 2424
PRUSS_MII_MDIO_USERINTMASKSET	RW	32	0x0000 0028	0x4B23 2428
PRUSS_MII_MDIO_USERINTMASKCLR	RW	32	0x0000 002C	0x4B23 242C
PRUSS_MII_MDIO_USERACCESS0	RW	32	0x0000 0080	0x4B23 2480
PRUSS_MII_MDIO_USERPHYSEL0	RW	32	0x0000 0084	0x4B23 2484
PRUSS_MII_MDIO_USERACCESS1	RW	32	0x0000 0088	0x4B23 2488
PRUSS_MII_MDIO_USERPHYSEL1	RW	32	0x0000 008C	0x4B23 248C

Table 30-861. PRUSS2_MII_MDIO Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
PRUSS_MII_MDIO_VER	R	32	0x0000 0000	0x4B2B 2400
PRUSS_MII_MDIO_CONTROL	RW	32	0x0000 0004	0x4B2B 2404
PRUSS_MII_MDIO_ALIVE	RW	32	0x0000 0008	0x4B2B 2408
PRUSS_MII_MDIO_LINK	R	32	0x0000 000C	0x4B2B 240C
PRUSS_MII_MDIO_LINKINTRAW	RW	32	0x0000 0010	0x4B2B 2410
PRUSS_MII_MDIO_LINKINTMASKED	RW	32	0x0000 0014	0x4B2B 2414
PRUSS_MII_MDIO_USERINTRAW	RW	32	0x0000 0020	0x4B2B 2420
PRUSS_MII_MDIO_USERINTMASKED	RW	32	0x0000 0024	0x4B2B 2424
PRUSS_MII_MDIO_USERINTMASKSET	RW	32	0x0000 0028	0x4B2B 2428

Table 30-861. PRUSS2_MII_MDIO Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
PRUSS_MII_MDIO_USERINTMASKCLR	RW	32	0x0000 002C	0x4B2B 242C
PRUSS_MII_MDIO_USERACCESS0	RW	32	0x0000 0080	0x4B2B 2480
PRUSS_MII_MDIO_USERPHYSEL0	RW	32	0x0000 0084	0x4B2B 2484
PRUSS_MII_MDIO_USERACCESS1	RW	32	0x0000 0088	0x4B2B 2488
PRUSS_MII_MDIO_USERPHYSEL1	RW	32	0x0000 008C	0x4B2B 248C

30.2.10.4.2.2 PRUSS_MII_MDIO Register Description

Table 30-862. PRUSS_MII_MDIO_VER

Address Offset	0x0000 0000		
Physical Address	0x4B23 2400 0x4B2B 2400	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	MDIO MODULE VERSION REGISTER		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REVISION																															

Bits	Field Name	Description	Type	Reset
31:0	REVISION	IP Revision.	R	0x-[(1)]

(1) TI Internal Data

Table 30-863. Register Call Summary for Register PRUSS_MII_MDIO_VER

PRU-ICSS MII MDIO Module

- PRUSS_MII_MDIO Register Summary: [0] [1]

Table 30-864. PRUSS_MII_MDIO_CONTROL

Address Offset	0x0000 0004		
Physical Address	0x4B23 2404 0x4B2B 2404	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	MDIO MODULE CONTROL REGISTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDLE	ENABLE	RESERVED	HIGHEST_USER_CHANNEL					RESERVED			PREAMBLE	FAULT	FAULT_DETECT_ENABLE	INT_TEST_ENABLE	RESERVED	CLKDIV																

Bits	Field Name	Description	Type	Reset
31	IDLE	MDIO state machine IDLE. Set to 1 when the state machine is in the idle state.	R	0x1
30	ENABLE	Enable control. Writing a 1 to this bit enables the MDIO state machine, writing a 0 disables it. If the MDIO state machine is active at the time it is disabled, it will complete the current operation before halting and setting the idle bit. If using byte access, the enable bit has to be the last bit written in this register.	RW	0x0
29	RESERVED		R	0
28:24	HIGHEST_USER_CHANNEL	Highest user channel. This field specifies the highest useraccess channel that is available in the module and is currently set to 1. This implies that **MDIOUserAccess1** is the highest available user access channel.	R	0x1
23:21	RESERVED		R	0x0
20	PREAMBLE	Preamble disable. Writing a 1 to this bit disables this device from sending MDIO frame preambles.	RW	0x0
19	FAULT	Fault indicator. This bit is set to 1 if the MDIO pins fail to read back what the device is driving onto them. This indicates a physical layer fault and the module state machine is reset. Writing a 1 to it clears this bit.	RW	0x0
18	FAULT_DETECT_ENABLE	Fault detect enable. This bit has to be set to 1 to enable the physical layer fault detection.	RW	0x0
17	INT_TEST_ENABLE	Interrupt test enable. This bit can be set to 1 to enable the host to set the userint and linkint bits for test purposes.	RW	0x0
16	RESERVED		R	0
15:0	CLKDIV	Clock Divider. This field specifies the division ratio between CLK and the frequency of MDCLK. MDCLK is disabled when clkdiv is set to 0. MDCLK frequency = clk frequency/(clkdiv+1).	RW	0xff

Table 30-865. Register Call Summary for Register PRUSS_MII_MDIO_CONTROL

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-866. PRUSS_MII_MDIO_ALIVE

Address Offset	0x0000 0008		
Physical Address	0x4B23 2408 0x4B2B 2408	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	PHY ACKNOWLEDGE STATUS REGISTER		
Type	RWr1Clr		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALIVE																															

Bits	Field Name	Description	Type	Reset
31:0	ALIVE	MDIO Alive bitfield. Each of the 32 bits of this register is set if the most recent access to the PHY with address corresponding to the register bit number was acknowledged by the PHY, the bit is reset if the PHY fails to acknowledge the access. Both the user and polling accesses to a PHY will cause the corresponding alive bit to be updated. The alive bits are only meant to be used to give an indication of the presence or not of a PHY with the corresponding address. Writing a 1 to any bit will clear it, writing a 0 has no effect.	RWr1Clr	0x0

Table 30-867. Register Call Summary for Register PRUSS_MII_MDIO_ALIVE

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-868. PRUSS_MII_MDIO_LINK

Address Offset	0x0000 000C		
Physical Address	0x4B23 240C 0x4B2B 240C	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	PHY LINK STATUS REGISTER		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINK																															

Bits	Field Name	Description	Type	Reset
31:0	LINK	MDIO Link state. This register is updated after a read of the Generic Status Register of a PHY. The bit is set if the PHY with the corresponding address has link and the PHY acknowledges the read transaction. The bit is reset if the PHY indicates it does not have link or fails to acknowledge the read transaction. Writes to the register have no effect. In addition, the status of the two PHYs specified in the **MDIOUserPhySel** registers can be determined using the **MLINK** input pins. This is determined by the **linksel** bit in the **MDIOUserPhySel** register.	R	0x0

Table 30-869. Register Call Summary for Register PRUSS_MII_MDIO_LINK

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-870. PRUSS_MII_MDIO_LINKINTRAW

Address Offset	0x0000 0010		
Physical Address	0x4B23 2410 0x4B2B 2410	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	LINK STATUS CHANGE INTERRUPT REGISTER (RAW VALUE)		
Type	RWr1Clr		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														LINKINTRAW	

Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0000 000
1:0	LINKINTRAW	MDIO link change event, raw value. When asserted '1', a bit indicates that there was an **MDIO link** change event (i.e. change in the MDIOLink register) corresponding to the PHY address in the **MDIOUserPhySel** register. **linkintraw[0] and linkintraw[1]** correspond to **MDIOUserPhySel0 and MDIOUserPhysel1**, respectively. Writing a '1' will clear the event and writing 0 has no effect. **If the int_test bit in the MDIOControl register is set, the host may set the linkintraw bits to a '1'.** This mode may be used for test purposes.	RWr1Clr	0x0

Table 30-871. Register Call Summary for Register PRUSS_MII_MDIO_LINKINTRAW

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-872. PRUSS_MII_MDIO_LINKINTMASKED

Address Offset	0x0000 0014		
Physical Address	0x4B23 2414 0x4B2B 2414	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	LINK STATUS CHANGE INTERRUPT REGISTER (MASKED VALUE)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														LINKINTMASKED	

Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0000 000
1:0	LINKINTMASKED	MDIO link change interrupt, masked value. When asserted '1', a bit indicates that there was an MDIO link change event (i.e. change in the MDIOLink register) corresponding to the PHY address in the MDIOUserPhySel register and the corresponding linkint_enable bit was set.. linkintmasked[0] and linkintmasked[1] correspond to MDIOUserPhySel0 and MDIOUserPhysel1, respectively. Writing a '1' will clear the interrupt and writing 0 has no effect. If the int_test bit in the MDIOControl register is set, the host may set the linkint bits to a '1'. This mode may be used for test purposes.	RW	0x0

Table 30-873. Register Call Summary for Register PRUSS_MII_MDIO_LINKINTMASKED

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-874. PRUSS_MII_MDIO_USERINTRAW

Address Offset	0x0000 0020		
Physical Address	0x4B23 2420 0x4B2B 2420	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	USER COMMAND COMPLETE INTERRUPT REGISTER (RAW VALUE)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														USERINTRAW	

Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0000 000
1:0	USERINTRAW	Raw value of MDIO user command complete event for MDIOUserAccess1 through MDIOUserAccess0, respectively. When asserted ‘1’, a bit indicates that the previously scheduled PHY read or write command using that particular MDIOUserAccess register has completed. Writing a ‘1’ will clear the event and writing ‘0’ has no effect. . If the int_test bit in the MDIOControl register is set, the host may set the userintraw bits to a ‘1’. This mode may be used for test purposes.	RW	0x0

Table 30-875. Register Call Summary for Register PRUSS_MII_MDIO_USERINTRAW

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-876. PRUSS_MII_MDIO_USERINTMASKED

Address Offset	0x0000 0024		
Physical Address	0x4B23 2424 0x4B2B 2424	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	USER COMMAND COMPLETE INTERRUPT REGISTER (MASKED VALUE)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														USERINTMASKED	

Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0000 000
1:0	USERINTMASKED	Masked value of MDIO user command complete interrupt for MDIOUserAccess1 through MDIOUserAccess0, respectively. When asserted '1', a bit indicates that the previously scheduled PHY read or write command using that particular MDIOUserAccess register has completed and the corresponding userintmaskset bit is set to '1'. . Writing a '1' will clear the interrupt and writing '0' has no effect. . If the int_test bit in the MDIOControl register is set, the host may set the userintmasked bits to a '1'. This mode may be used for test purposes.	RW	0x0

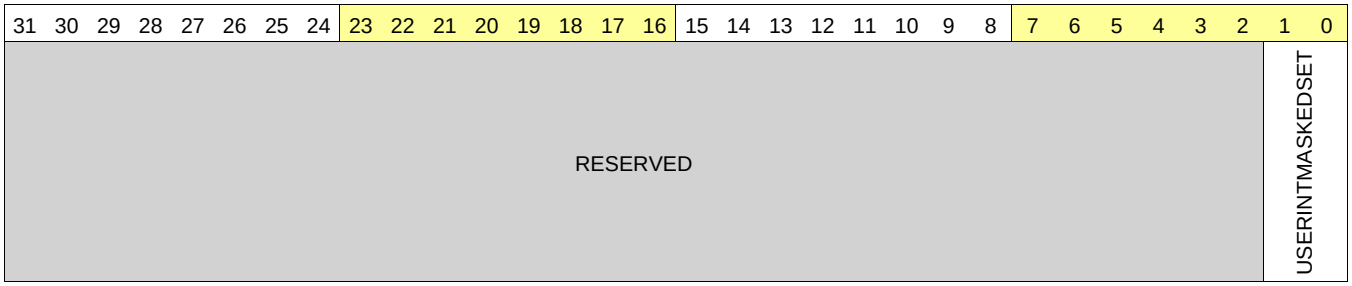
Table 30-877. Register Call Summary for Register PRUSS_MII_MDIO_USERINTMASKED

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-878. PRUSS_MII_MDIO_USERINTMASKSET

Address Offset	0x0000 0028		
Physical Address	0x4B23 2428 0x4B2B 2428	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	USER INTERRUPT MASK SET REGISTER		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0000 000
1:0	USERINTMASKEDSET	MDIO user interrupt mask set for userintmasked[1:0], respectively. Writing a bit to '1' will enable MDIO user command complete interrupts for that particular MDIOUserAccess register. MDIO user interrupt for a particular MDIOUserAccess register is disabled if the corresponding bit is '0'. Writing a '0' to this register has no effect.	RW	0x0

Table 30-879. Register Call Summary for Register PRUSS_MII_MDIO_USERINTMASKSET

PRU-ICSS MII MDIO Module

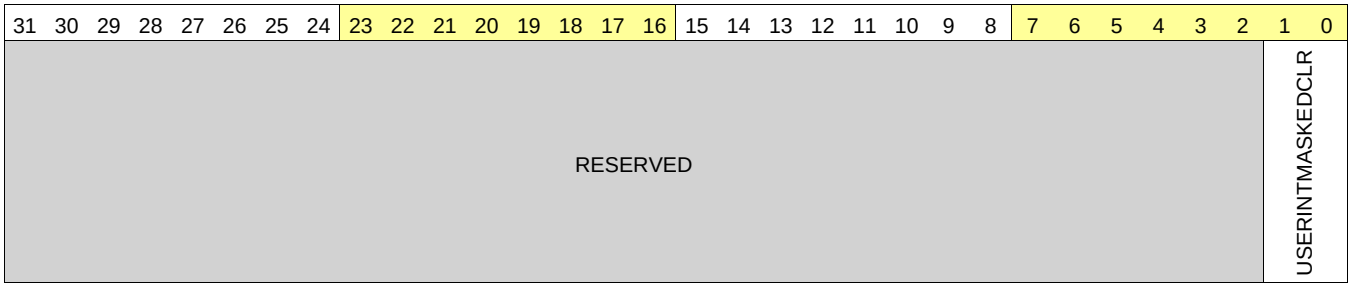
- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-880. PRUSS_MII_MDIO_USERINTMASKCLR

Address Offset	0x0000 002C		
Physical Address	0x4B23 242C 0x4B2B 242C	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO

Table 30-880. PRUSS_MII_MDIO_USERINTMASKCLR (continued)

Description	USER INTERRUPT MASK CLEAR REGISTER
Type	RW



Bits	Field Name	Description	Type	Reset
31:2	RESERVED		R	0x0000 000
1:0	USERINTMASKEDCLR	MDIO user command complete interrupt mask clear for userintmasked[1:0], respectively. Writing a bit to '1' will disable further user command complete interrupts for that particular MDIOUserAccess register. Writing a '0' to this register has no effect.	RW	0x0

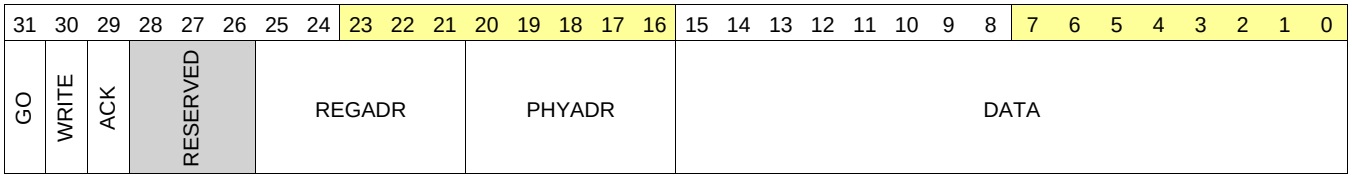
Table 30-881. Register Call Summary for Register PRUSS_MII_MDIO_USERINTMASKCLR

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-882. PRUSS_MII_MDIO_USERACCESS0

Address Offset	0x0000 0080		
Physical Address	0x4B23 2480 0x4B2B 2480	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	USER ACCESS REGISTER0		
Type	RW		



Bits	Field Name	Description	Type	Reset
31	GO	Go. Writing a 1 to this bit causes the MDIO state machine to perform an MDIO access when it is convenient for it to do so, this is not an instantaneous process. Writing a 0 to this bit has no effect. This bit is write able only if the MDIO state machine is enabled. This bit will self clear when the requested access has been completed. Any writes to the MDIOUserAccess0 register are blocked when the go bit is '1'. If byte access is being used, the go bit should be written last.	RW	0x0
30	WRITE	Write enable. Setting this bit to a 1 causes the MDIO transaction to be a register write, otherwise it is a register read.	RW	0x0

Bits	Field Name	Description	Type	Reset
29	ACK	Acknowledge. This bit is set if the PHY acknowledged the read transaction.	RW	0x0
28:26	RESERVED		R	0x0
25:21	REGADR	Register address. This field specifies the PHY register to be accessed for this transaction.	RW	0x0
20:16	PHYADR	PHY address. This field specifies the PHY to be accessed for this transaction.	RW	0x0
15:0	DATA	User data. The data value read from or to be written to the specified PHY register.	RW	0x0

Table 30-883. Register Call Summary for Register PRUSS_MII_MDIO_USERACCESS0

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-884. PRUSS_MII_MDIO_USERPHYSEL0

Address Offset	0x0000 0084		
Physical Address	0x4B23 2484 0x4B2B 2484	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	USER PHY SELECT REGISTER0		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								LINKSEL	LINKINT_ENABLE	RESERVED	PHYADR_MON				

Bits	Field Name	Description	Type	Reset
31:8	RESERVED		R	0x0000 00
7	LINKSEL	Link status determination select. Set to ‘1’ to determine link status using the MLINK pin. Default value is ‘0’ which implies that the link status is determined by the MDIO state machine.	RW	0x0
6	LINKINT_ENABLE	Link change interrupt enable. Set to ‘1’ to enable link change status interrupts for PHY address specified in phyadr_mon. Link change interrupts are disabled if this bit is set to ‘0’.	RW	0x0
5	RESERVED		R	0
4:0	PHYADR_MON	PHY address whose link status is to be monitored.	RW	0x0

Table 30-885. Register Call Summary for Register PRUSS_MII_MDIO_USERPHYSEL0

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-886. PRUSS_MII_MDIO_USERACCESS1

Address Offset	0x0000 0088		
Physical Address	0x4B23 2488 0x4B2B 2488	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	USER ACCESS REGISTER1		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GO	WRITE	ACK	RESERVED					REGADR					PHYADR			DATA															

Bits	Field Name	Description	Type	Reset
31	GO	Go. Writing a 1 to this bit causes the MDIO state machine to perform an MDIO access when it is convenient for it to do so, this is not an instantaneous process. Writing a 0 to this bit has no effect. This bit is write able only if the MDIO state machine is enabled. This bit will self clear when the requested access has been completed. Any writes to the MDIOUserAccess0 register are blocked when the go bit is '1'. If byte access is being used, the go bit should be written last.	RW	0x0
30	WRITE	Write enable. Setting this bit to a 1 causes the MDIO transaction to be a register write, otherwise it is a register read.	RW	0x0
29	ACK	Acknowledge. This bit is set if the PHY acknowledged the read transaction.	RW	0x0
28:26	RESERVED		R	0x0
25:21	REGADR	Register address. This field specifies the PHY register to be accessed for this transaction.	RW	0x0
20:16	PHYADR	PHY address. This field specifies the PHY to be accessed for this transaction.	RW	0x0
15:0	DATA	User data. The data value read from or to be written to the specified PHY register.	RW	0x0

Table 30-887. Register Call Summary for Register PRUSS_MII_MDIO_USERACCESS1

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

Table 30-888. PRUSS_MII_MDIO_USERPHYSEL1

Address Offset	0x0000 008C		
Physical Address	0x4B23 248C 0x4B2B 248C	**Instance**	PRUSS1_MII_MDIO PRUSS2_MII_MDIO
Description	USER PHY SELECT REGISTER1		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								LINKSEL	LINKINT_ENABLE	RESERVED	PHYADR_MON				

Bits	Field Name	Description	Type	Reset
31:8	RESERVED		R	0x0000 000
7	LINKSEL	Link status determination select. Set to ‘1’ to determine link status using the MLINK pin. Default value is ‘0’ which implies that the link status is determined by the MDIO state machine.	RW	0x0
6	LINKINT_ENABLE	Link change interrupt enable. Set to ‘1’ to enable link change status interrupts for PHY address specified in phyadr_mon. Link change interrupts are disabled if this bit is set to ‘0’.	RW	0x0
5	RESERVED		R	0
4:0	PHYADR_MON	PHY address whose link status is to be monitored.	RW	0x0

Table 30-889. Register Call Summary for Register PRUSS_MII_MDIO_USERPHYSEL1

PRU-ICSS MII MDIO Module

- PRU-ICSS MII MDIO Receive/Transmit Frame Host Software Interface: [0]
- PRUSS_MII_MDIO Register Summary: [1] [2]

30.2.11 PRU-ICSS Industrial Ethernet Peripheral (IEP)

This section describes the Industrial Ethernet Peripheral (IEP) module which is part of the PRU-ICSS.

30.2.11.1 PRU-ICSS IEP Overview

The Industrial Ethernet Peripheral (IEP) performs hardware work required for industrial ethernet functions. The IEP module features an industrial ethernet timer with 8 compare events and a digital I/O port (DIGIO).

30.2.11.2 PRU-ICSS IEP Functional Description

This section provides the functional description of the IEP components.

30.2.11.2.1 PRU-ICSS IEP Clock Generation

The IEP has a selectable module input clock (ICSS_IEP_CLK input, see also Section 30.2.3). The clock source is selected by the state of the IEPCLK.OCP_EN bit within the PRU-ICSS CFG register space.

Two clock sources are supported for the IEP input clock:

- PRUSS_IEP_CLK: The default functional clock for IEP derived from PRCM. Runs at 200 MHz.
- PRUSS_GICLK: The PRUSS_CFG gateable interface clock derived from PRCM.

Switching from PRUSS_IEP_CLK to PRUSS_GICLK is done by writing 1 to the PRUSS_IEPCLK.OCP_EN bit. This is a one time configuration step before enabling the IEP function. Switching back from PRUSS_GICLK to PRUSS_IEP_CLK is only supported through a hardware reset of the PRU-ICSS.

CAUTION

When software enables the clock (at PRU-ICSS level) to the IEP module clock input via setting bit PRUSS_IEPCLK[0] OCP_EN to 0b1 in the PRUSS_CFG space, there must be NO in-flight transactions to the IEP block.

CAUTION

ONLY switching from PRUSS_IEP_CLK (the IEP specific functional clock source) to the PRUSS_GICLK (top level interface clock) source is supported in software by device integrated PRU-ICSS. Switching back from PRUSS_GICLK to PRUSS_IEP_CLK is ONLY supported via assertion of a hardware reset to the PRU-ICSS.

30.2.11.2.2 PRU-ICSS Industrial Ethernet Timer

The industrial ethernet timer is a simple 32-bit timer. This timer is intended for use by industrial ethernet functions but can also be leveraged as a generic timer in other applications.

30.2.11.2.2.1 PRU-ICSS Industrial Ethernet Timer Features

The industrial ethernet timer supports the following features:

- One master 32-bit count-up counter with an overflow status bit.
  - Runs on PRUSS_IEP_CLK or PRUSS_GICLK.
  - Write 1 to clear status.
  - Supports a programmable increment value from 1 to 16 (default 5).
  - An optional compensation method allows the increment value to apply compensation increment value from 1 to 16 count up to 2^{24} PRUSS_IEP_CLK/PRUSS_GICLK events with additional slow compensation mode
- 8× 32-bit compare registers: PRUSS_IEP_CMPj (where j=0 to 7) and PRUSS_IEP_CMP_STATUS.

  - 8 status bits, write 1 to clear
  - 8 individual event outputs
  - One global event output for interrupt generation triggered by any compare event
- 16 outputs, one high-level and one high-pulse for each compare hit event
- PRUSS_IEP_CMP_CFG[0] CMP0_RST_CNT_EN, if enabled, will reset the master counter
- master counter reset-state is programmable

30.2.11.2.2.2 PRU-ICSS Industrial Ethernet Timer Basic Programming Sequence

Follow these basic steps to configure the IEP Timer.

Compare function:

1. Initialize timer to known state (default values)
   - Disable counter (PRUSS_IEP_GLOBAL_CFG[0] CNT_ENABLE = 0)
   - Reset Count Register (PRUSS_IEP_COUNT) by writing 0xFFFFFFFF to clear
   - Clear overflow status register (PRUSS_IEP_GLOBAL_STATUS[0] CNT_OVF = 1)
   - Clear compare status (PRUSS_IEP_CMP_STATUS) by writing 0xFFFFFFFF to clear
2. Set compare values PRUSS_IEP_CMPj
3. Enable compare events (PRUSS_IEP_CMP_CFG[8:1] CMP_EN).
4. Set increment value (PRUSS_IEP_GLOBAL_CFG[7:4] DEFAULT_INC).
5. Set compensation value (PRUSS_IEP_COMPEN[23:0] COMPEN_CNT)
6. Enable counter (PRUSS_IEP_GLOBAL_CFG[0] CNT_ENABLE = 1)

30.2.11.2.3 PRU-ICSS Industrial Ethernet Digital IOs

The IEP Digital I/O (DIGIO) block provides dedicated I/Os intended for industrial ethernet protocols, but they can also be used as generic I/Os in other applications.

30.2.11.2.3.1 Features

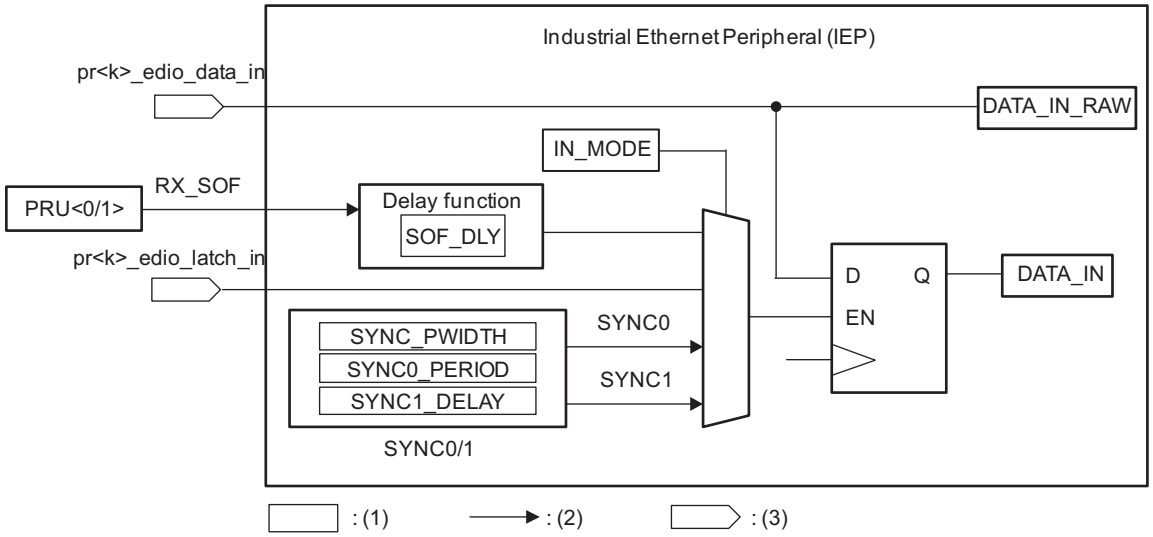
The industrial ethernet digital I/O supports the following features:

- Digital data output
  - 8 channels (pr1_edio_data_out[7:0])
  - Software controls enable signal driving output data output
- Digital data out enable (optional tri-state control)
- Digital data input
  - 8 channels (pr1_edio_data_in[7:0])
  - PRUSS_IEP_DIGIO_DATA_IN_RAW supports direct sampling of pr1_edio_data_in
  - External latch event signal (pr1_edio_latch_in) triggers a pulse on which pr1_edio_data_in is sampled

30.2.11.2.3.2 DIGIO Block Diagrams

Figure 30-107 shows the signals and registers for capturing the DIGIO data in. Note that IN_MODE in the PRUSS_IEP_DIGIO_CTRL register must be set to 1 for data to be latched on the external pr1_edio_latch_in signal.

Figure 30-107. IEP DIGIO Data In



(1) Register

(2) Internal signal wire

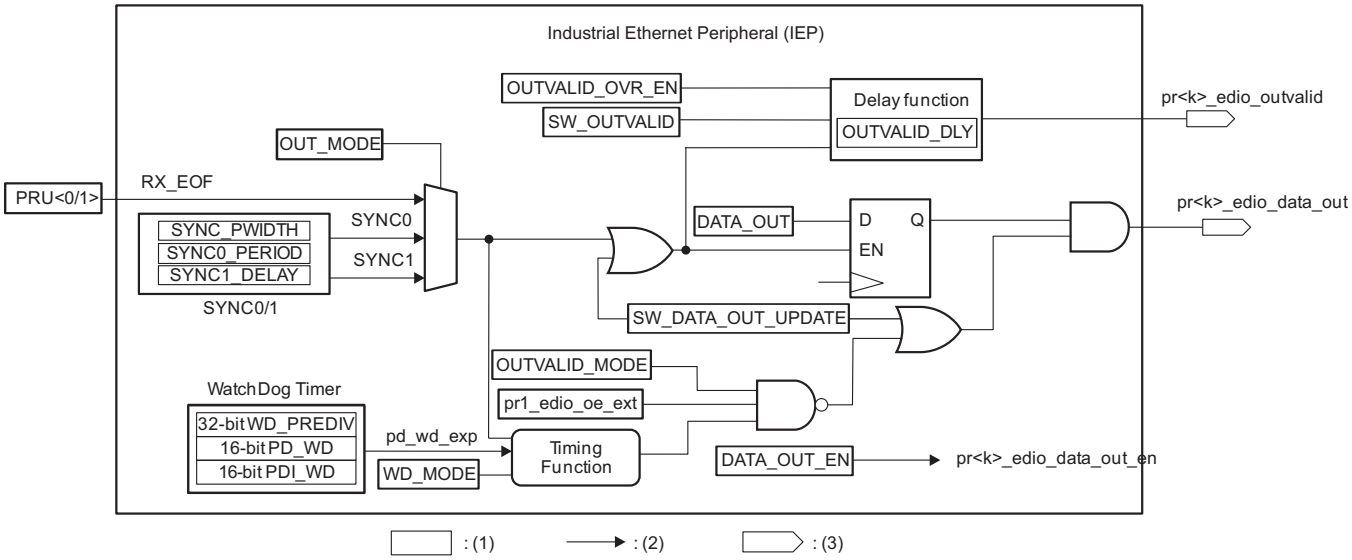
(3) External pin input/output

(1) Register

(2) External pin input/output

Figure 30-108 shows the signals and registers for driving the DIGIO data out.

Figure 30-108. IEP DIGIO Data Out



(1) Register

(2) Internal signal wire

(3) External pin input/output

(1) Register

(2) External pin input/output

30.2.11.2.3.3 Basic Programming Model

Follow these steps to configure and read the DIGIO Data Input.

1. Read PRUSS_IEP_DIGIO_DATA_IN_RAW for raw input data

or

1. Enable sampling of pr1_edio_data_in[7:0] by setting PRUSS_IEP_DIGIO_CTRL[5:4] IN_MODE = 0x1
2. Read PRUSS_IEP_DIGIO_DATA_IN for data sampled upon pr1_edio_latch_in posedge

Follow these steps to configure and write to the DIGIO Data Output.

1. Pre-configure DIGIO by setting PRUSS_IEP_DIGIO_EXP[1] OUTVALID_OVR_EN and PRUSS_IEP_DIGIO_EXP[0] SW_DATA_OUT_UPDATE
2. Write to PRUSS_IEP_DIGIO_DATA_OUT to configure output data
3. To Hi-Z output, set corresponding PRUSS_IEP_DIGIO_DATA_OUT_EN bits to 1 (clear to 0 to drive value stored in PRUSS_IEP_DIGIO_DATA_OUT)

30.2.11.3 PRUSS_IEP Register Manual

This section describes the registers of the PRUSS_IEP module.

30.2.11.3.1 PRUSS_IEP Instance Summary

Table 30-890. PRUSS_IEP Instance Summary

Module Name	Base Address	Size
PRUSS1_IEP	0x4B22 E000	796 Bytes
PRUSS2_IEP	0x4B2A E000	796 Bytes

30.2.11.3.2 PRUSS_IEP Registers

30.2.11.3.2.1 PRUSS_IEP Register Summary

Table 30-891. PRUSS1_IEP Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS1_IEP Physical Address
PRUSS_IEP_GLOBAL_CFG	RW	32	0x0000 0000	0x4B22 E000
PRUSS_IEP_GLOBAL_STATUS	RW	32	0x0000 0004	0x4B22 E004
PRUSS_IEP_COMPEN	RW	32	0x0000 0008	0x4B22 E008
PRUSS_IEP_COUNT	RW	32	0x0000 000C	0x4B22 E00C
PRUSS_IEP_CMP_CFG	RW	32	0x0000 0040	0x4B22 E040
PRUSS_IEP_CMP_STATUS	RW	32	0x0000 0044	0x4B22 E044
PRUSS_IEP_CMPj[(1)]	RW	32	0x0000 0048 + (0x4*j)	0x4B22 E048 + (0x4*j)
PRUSS_IEP_DIGIO_CTRL	RW	32	0x0000 0300	0x4B22 E300
PRUSS_IEP_DIGIO_STATUSRESERVED	R	32	0x0000 0304	0x4B22 E304
PRUSS_IEP_DIGIO_DATA_IN	R	32	0x0000 0308	0x4B22 E308
PRUSS_IEP_DIGIO_DATA_IN_RAW	R	32	0x0000 030C	0x4B22 E30C
PRUSS_IEP_DIGIO_DATA_OUT	RW	32	0x0000 0310	0x4B22 E310
PRUSS_IEP_DIGIO_DATA_OUT_EN	RW	32	0x0000 0314	0x4B22 E314
PRUSS_IEP_DIGIO_EXP	RW	32	0x0000 0318	0x4B22 E318

(1) j=0 to 7

Table 30-892. PRUSS2_IEP Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS2_IEP Physical Address
PRUSS_IEP_GLOBAL_CFG	RW	32	0x0000 0000	0x4B2A E000

Table 30-892. PRUSS2_IEP Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	PRUSS2_IEP Physical Address
PRUSS_IEP_GLOBAL_STATUS	RW	32	0x0000 0004	0x4B2A E004
PRUSS_IEP_COMPEN	RW	32	0x0000 0008	0x4B2A E008
PRUSS_IEP_COUNT	RW	32	0x0000 000C	0x4B2A E00C
PRUSS_IEP_CMP_CFG	RW	32	0x0000 0040	0x4B2A E040
PRUSS_IEP_CMP_STATUS	RW	32	0x0000 0044	0x4B2A E044
PRUSS_IEP_CMPj[(1)]	RW	32	0x0000 0048 + (0x4*j)	0x4B2A E048 + (0x4*j)
PRUSS_IEP_DIGIO_CTRL	RW	32	0x0000 0300	0x4B2A E300
PRUSS_IEP_DIGIO_STATUSRESERVED	R	32	0x0000 0304	0x4B2A E304
PRUSS_IEP_DIGIO_DATA_IN	R	32	0x0000 0308	0x4B2A E308
PRUSS_IEP_DIGIO_DATA_IN_RAW	R	32	0x0000 030C	0x4B2A E30C
PRUSS_IEP_DIGIO_DATA_OUT	RW	32	0x0000 0310	0x4B2A E310
PRUSS_IEP_DIGIO_DATA_OUT_EN	RW	32	0x0000 0314	0x4B2A E314
PRUSS_IEP_DIGIO_EXP	RW	32	0x0000 0318	0x4B2A E318

(1) j=0 to 7

30.2.11.3.2.2 PRUSS_IEP Register Description

Table 30-893. PRUSS_IEP_GLOBAL_CFG

Address Offset	0x0000 0000		
Physical Address	0x4B22 E000 0x4B2A E000	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	GLOBAL CFG		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CMP_INC																DEFAULT_INC				RESERVED			CNT_ENABLE

Bits	Field Name	Description	Type	Reset
31:20	RESERVED		R	0x0
19:8	CMP_INC	Defines the increment value when compensation is active	RW	0x5
7:4	DEFAULT_INC	Defines the default increment value	RW	0x5
3:1	RESERVED		R	0
0	CNT_ENABLE	Counter enable 0: Disables the counter. The counter maintains the current count. 1: Enables the counter.	RW	0x0

Table 30-894. PRUSS_IEP_GLOBAL_STATUS

Address Offset	0x0000 0004		
Physical Address	0x4B22 E004 0x4B2A E004	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	STATUS		
Type	RWr1Clr		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															CNT_OVF

Bits	Field Name	Description	Type	Reset
31:1	RESERVED		R	0x0
0	CNT_OVF	Counter overflow status. 0: No overflow 1: Overflow occurred	RWr1Clr	0x0

Table 30-895. Register Call Summary for Register PRUSS_IEP_GLOBAL_STATUS

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRU-ICSS Industrial Ethernet Timer Basic Programming Sequence: [0]
- PRUSS_IEP Register Summary: [1] [2]

Table 30-896. PRUSS_IEP_COMPEN

Address Offset	0x0000 0008		
Physical Address	0x4B22 E008 0x4B2A E008	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	COMPENSATION		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								COMPEN_CNT																							

Bits	Field Name	Description	Type	Reset
31:24	RESERVED		R	0
23:0	COMPEN_CNT	Compensation counter. Read returns the current COMPEN_CNT value. 0: Compensation is disabled and counter will increment by DEFAULT_INC. n: Compensation is enabled until COMPEN_CNT decrements to 0. The COMPEN_CNT value decrements on every iep_clk cycle. When COMPEN_CNT is greater than 0, then count value increments by CMP_INC.	RW	0x0

Table 30-897. Register Call Summary for Register PRUSS_IEP_COMPEN

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRU-ICSS Industrial Ethernet Timer Basic Programming Sequence: [0]
- PRUSS_IEP Register Summary: [1] [2]

Table 30-898. PRUSS_IEP_COUNT

Address Offset	0x0000 000C		
Physical Address	0x4B22 E00C 0x4B2A E00C	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	COUNTER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT																															

Bits	Field Name	Description	Type	Reset
31:0	COUNT	32-bit count value. Increments by (DEFAULT_INC or CMP_INC) on every positive edge of PRUSS_IEP_CLK (200MHz) or PRUSS_GICLK.	RW	0x0

Table 30-899. Register Call Summary for Register PRUSS_IEP_COUNT

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRU-ICSS Industrial Ethernet Timer Basic Programming Sequence: [0]
- PRUSS_IEP Register Summary: [1] [2]

Table 30-900. PRUSS_IEP_CMP_CFG

Address Offset	0x0000 0040		
Physical Address	0x4B22 E040 0x4B2A E040	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	COMPARE CFG		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CMP_EN							CMP0_RST_CNT_EN

Bits	Field Name	Description	Type	Reset
31:9	RESERVED		R	0x00000
8:1	CMP_EN	Enable bit for each of the compare registers cmp_en<n> =0: Disables CMP<n> Event cmp_en<n> =1: Enables CMP<n> Event cmp_en[0] maps to CMP0	RW	0x0
0	CMP0_RST_CNT_EN	Enable the reset of the counter 0: Disable 1: Enable the reset of the counter if a cmp0 event occurs	RW	0x0

Table 30-901. Register Call Summary for Register PRUSS_IEP_CMP_CFG

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRU-ICSS Industrial Ethernet Timer Features: [0]
- PRU-ICSS Industrial Ethernet Timer Basic Programming Sequence: [1]
- PRUSS_IEP Register Summary: [2] [3]

Table 30-902. PRUSS_IEP_CMP_STATUS

Address Offset	0x0000 0044		
Physical Address	0x4B22 E044 0x4B2A E044	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	COMPARE STATUS		
Type	RWr1Clr		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CMP_HIT							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED		R	0x0000 00
7:0	CMP_HIT	Status bit for each of the compare registers. "Match" indicates the current counter is greater than or equal to the compare value. Note it is the firmware's responsibility to handle the IEP overflow. cmp_hit<n> = 0: No match has occured cmp_hit<n> = 1: A match occured. The associated hardware event signal will assert and remain high until the status is cleared.	RWr1Clr	0x0

Table 30-903. Register Call Summary for Register PRUSS_IEP_CMP_STATUS

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRU-ICSS Industrial Ethernet Timer Features: [0]
- PRU-ICSS Industrial Ethernet Timer Basic Programming Sequence: [1]
- PRUSS_IEP Register Summary: [2] [3]

Table 30-904. PRUSS_IEP_CMPj

Address Offset	0x0000 0048 + (0x4*j)	**Index**	j = 0 to 7
Physical Address	0x4B22 E048 + (0x4*j) 0x4B2A E048 + (0x4*j)	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	COMPARE(j), where j=0 to 7		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP																															

Bits	Field Name	Description	Type	Reset
31:0	CMP	Compare j value >= comparator	RW	0x0

Table 30-905. Register Call Summary for Register PRUSS_IEP_CMPj

PRU-ICSS Industrial Ethernet Peripheral (IEP)

- PRU-ICSS Industrial Ethernet Timer Features: [0]
- PRU-ICSS Industrial Ethernet Timer Basic Programming Sequence: [1]
- PRUSS_IEP Register Summary: [2] [3]

Table 30-906. PRUSS_IEP_DIGIO_CTRL

Address Offset	0x0000 0300		
Physical Address	0x4B22 E300 0x4B2A E300	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	DIGIO		
Type	RW		

Bits	Field Name	Description	Type	Reset
31:8	RESERVED		R	0x0
7:6	OUT_MODE	Reserved	RW	0x0
5:4	IN_MODE	Defines event that triggers data in to be sampled 0b00: Reserved 0b01: Rising edge of external pr<k>_edio_latch_in signal 0b10: Reserved 0b11: Reserved	RW	0x0
3	WD_MODE	Reserved	RW	0x0
2	BIDI_MODE	Reserved	R	0x1
1	OUTVALID_MODE	Reserved	RW	0x0
0	OUTVALID_POL	Reserved	R	0x0

Table 30-907. PRUSS_IEP_DIGIO_DATA_IN

Address Offset	0x0000 0308		
Physical Address	0x4B22 E308 0x4B2A E308	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	DIGIO		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_IN																															

Bits	Field Name	Description	Type	Reset
31:0	DATA_IN	Data input. Sample time of digital inputs is controlled externally by using the pr<k>_edio_latch_in signal. Must enable by setting PRUSS_IEP_DIGIO_CTRL[5:4] IN_MODE. Only [7:0] are exported to device pins in this device.	R	0x0

Table 30-908. PRUSS_IEP_DIGIO_DATA_IN_RAW

Address Offset	0x0000 030C		
Physical Address	0x4B22 E30C 0x4B2A E30C	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	DIGIO		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_IN_RAW																															

Bits	Field Name	Description	Type	Reset
31:0	DATA_IN_RAW	Raw Data Input. Direct sample of EDIO_DATA_IN[31:0]. Only [7:0] are exported to device pins in this device.	R	0x0

Table 30-909. PRUSS_IEP_DIGIO_DATA_OUT

Address Offset	0x0000 0310		
Physical Address	0x4B22 E310 0x4B2A E310	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	DIGIO		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_OUT																															

Bits	Field Name	Description	Type	Reset
31:0	DATA_OUT	Data Output. Only [7:0] are exported to device pins in this device.	RW	0x0

Table 30-910. PRUSS_IEP_DIGIO_DATA_OUT_EN

Address Offset	0x0000 0314		
Physical Address	0x4B22 E314 0x4B2A E314	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	DIGIO		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_OUT_EN																															

Bits	Field Name	Description	Type	Reset
31:0	DATA_OUT_EN	Enables tri-state control for pr<k>_edio_data_out[7:0].	RW	0x0

Table 30-911. PRUSS_IEP_DIGIO_EXP

Address Offset	0x0000 0318		
Physical Address	0x4B22 E318 0x4B2A E318	**Instance**	PRUSS1_IEP PRUSS2_IEP
Description	DIGIO		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																		EOF_SEL	SOF_SEL	SOF_DLY				OUTVALID_DLY				RESERVED	SW_OUTVALID	OUTVALID_OVR_EN	SW_DATA_OUT_UPDATE

Bits	Field Name	Description	Type	Reset
31:14	RESERVED		R	0x0
13	EOF_SEL	Reserved	RW	0x0
12	SOF_SEL	Reserved	RW	0x0
11:8	SOF_DLY	Reserved	RW	0x0
7:4	OUTVALID_DLY	Reserved	RW	0x2
3	RESERVED		R	0
2	SW_OUTVALID	Reserved	RW	0x0
1	OUTVALID_OVR_EN	Enable software to control value of pr<k>_edio_data_out [7:0]. 0: Disable 1: Enable	RW	0x0

Bits	Field Name	Description	Type	Reset
0	SW_DATA_OUT_UPDATE	Enable PRUSS_DIGIO_DATA_OUT to be driven out on pr<k>_edio_data_out. Only valid if OUTVALID_OVR_EN = 1. 0: Disable 1: Enable	RW	0x0