Application Note

Serial and Parallel Port Implementation With Programmable Real-Time Unit (PRU)

Jason Ding, Nick Saulnier, and Thomas Yang

ABSTRACT

In Medical applications, an analog or digital daughter board need to maintain high throughput point-to-point communication bandwidth with the mother board over a serial or parallel port. Depending on the application scenarios, this application may also need customized protocols.

As systems become more complex and require lower latency, choosing a processor with suitable peripherals becomes challenging. High-speed serial or parallel interfaces are mandatory for many microprocessors or microcontrollers designs, in addition to standard interfaces like, serial peripheral interface (SPI), OSPI, or QSPI. Many of TI's Sitara™ processors provide the best balance of cost and performance by providing the ability to add additional interfaces or custom protocols with TI's unique Programmable Real Time Unit (PRU) cores.

TI's PRU cores provide low latency and no jitter due to its non-pipelined CPU architecture and 1024-bit data bus. TI offers several different types of PRU subsystems: PRUSS (available on AM62x), PRU-ICSS (available on AM335x, AM437x, AM57x, AM263x), and PRU_ICSSG (available on AM243x, AM64x).

This application note discusses how the PRU_ICSSG can be used to implement serial and parallel ports.

Table of Contents

List of Figures

List of Tables

Trademarks

Sitara™ is a trademark of Texas Instruments.
All trademarks are the property of their respective owners.

1 Introduction

Many medical equipment designs need low latency communication, or custom communication protocols. TI's Sitara processors with programmable real-time unit (PRU) cores are designed to meet those needs without having to add additional field-programmable gate array (FPGA), application-specific integrated circuit (ASIC), complex programmable logic device (CPLD) to the design.

Some potential use cases include:

Clinical monitoring equipment has many types of monitoring parameters, like Electrocardiography (ECG), Electromyography (EMG), Oxyhemoglobin saturation by pulse oximetry (SpO2), temperature, and bioelectrical impedance. However, there are often real-time requirements for the refresh rate of those parameters. Those real-time needs may require non-standard interfaces or protocols between the multi-parameter module and the main board.

In imaging systems, Continuous Wave (CW) Doppler is used to measure the blood flow inside the human body. The demodulated Doppler frequency produces in-phase (I) and quadrature (Q) data as outputs. Analog-to-digital converters (ADCs) are required to support a simultaneous sampling rate over 3MHz with 14-16-bit accuracy. The ADC output is often parallel CMOS or enhanced SPI. The analog front-end readout electronics required for direct imaging to convert a charge to digital data in an X-ray flat panel detector (FPD). The digital signal processor (DSP), FPGAs, ASICs or a combination of these applies signal conditioning. These processors also manage high-speed serial or parallel communications with the external image-processing unit through a high-speed interface.

In a vitro Polymerase Chain Reaction (PCR) detection system, samples are excited by a multi-band light source. Multiple analog signals from fluorescence detection signal chains need to be captured on a single processor for further processing. When more than two ADCs need to be interfaced with processor, processors with multiple SPI interfaces have limited access to more than two ADCs simultaneously due to sharing of the same SPI DMA data path. The PRU allows for simultaneous access to multiple ADCs. For more information, see the *Flexible Interface (PRU-ICSS) Reference Design for Simultaneous, Coherent DAQ Using Multiple ADCs Design Guide*.

In these application scenarios, most pipelined CPU processors have a fundamental drawback compared to FPGA or CPLD: the pipelined processors have higher latency and higher jitter. FPGA or CPLD flexibility and low latency are well suited to these scenarios, but adding FPGAs increase complexity and system cost. TI's Sitara processors with PRU cores are designed to suit these scenarios without the need to add FPGAs, CPLDs to the design. This application note demonstrates how the PRU's ultra-fast general-purpose input/output (GPIO), board-side RAM, and shift out/in peripherals enable high-speed, customized communication over serial or parallel interfaces.

2 Implementation

Microcontroller without flexible external logic can uses software to communication with the external device via its general-purpose input/output (GPIO) to simulate protocol or to drive some outputs with a certain waveform. Chip infrastructure like a Network on Chip (NOC) exists between the CPU cores and the GPIO pins and also with pipeline. The pattern is written from the CPU's internal registers through the NOC infrastructure to reach the GPIO. The NOC can add tens to even hundreds of nanoseconds of latency and jitter to the GPIO write commands. The system might not meet the peripheral's timing requirements for the output signal duty cycle, the relationship between outputs. These and other factors mean that compared to an FPGA, a standard MCU will have increased latency and lower determinism when interacting with a GPIO.

TI's PRU core is a best-in-class custom CPU designed to have the lowest latency and highest determinism in GPIO reads & writes. The PRU has 32 general purpose register inside the core and two registers are connected to general purpose output or input directly, which means that PRU core can access to external pin at single cycle. Since there is no NOC circuitry between the registers and the pins, signal latency from the core to the processor pins is dramatically reduced. Output signal jitter due to the signal path is 100% removed. The PRU gives best in class performance without losing flexibility.

This application only focuses on PRU_ICSSG. Each PRU_ICSSG actually has six cores, three cores in each slice. So, an AM243x or AM64x with two PRU_ICSSG subsystems will have 12 cores total. For more information, see the *PRU_ICSSG* section from the *AM64x/AM243x Technical Reference Manual*.

As shown in Figure 2-1, The general register R30 and R31 can be routed directly to the PRU's dedicated general purpose input (PRU GPI) and general-purpose output (PRU GPO). GPI signals and system events or interrupts to the PRU can be read by reading the R31 register, while the PRU can release system events by writing to the R31 register. Similarly, writing to the R30 register performs writes to the dedicated PRU GPO pins.

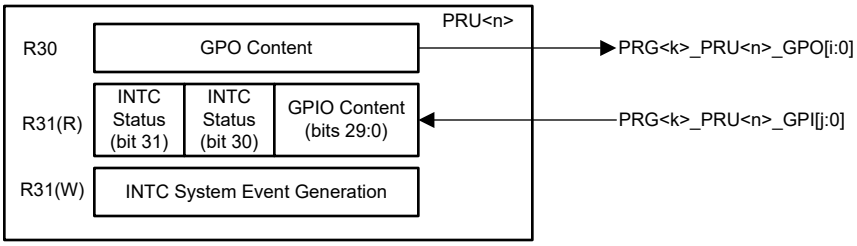

Figure 2-1. PRU GPIO Interface

The PRU implements an enhanced General-Purpose Input/output (GPIO) module with system control units that supports the following GPI and GPO modes: direct input/output, 16-bit parallel capture, 28-bit serial shift in/shift out, and MII_RT. Table 2-1 describes the input/output modes in detail.

Table 2-1. PRU R30/R31 Fast GPIO Mode

Mode	Function
Direct input/output	GPI [19:0] feeds directly into the PRU R31 and GPO [19:0] output directly through R30
16-bit parallel capture	DATAIN [0:15] is captured by the positive edge or negative edge of CLOCKIN
28-bit shift in/out	DATAIN is sampled and shifted into a 28-bit shift register. DATAOUT is shifted out through 28-bit shift register

The PRUx_R31_status [0:19] bits of the internal PRU register file are mapped to device-level, general purpose input pins (PRUx_GPI [0:19]). In GPI Direct Input mode, PRU0_GPI [0:19] feeds directly to PRUx_R31_status [0:19]. For example, instruction "ldi R10.b0, R31.b0" can shift data on the PRUx_GPI [0:7] lane into the first byte of R10 in a single PRU core cycle.

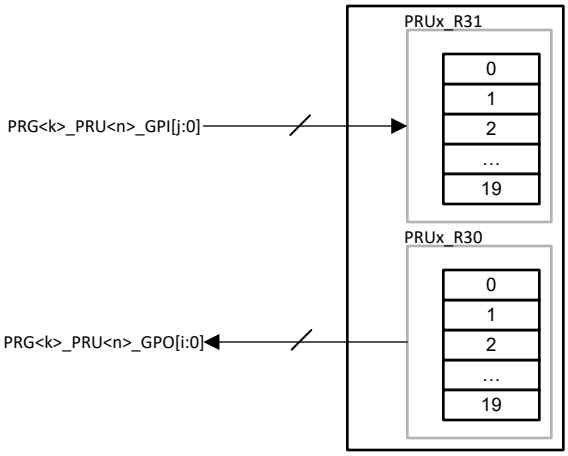

Figure 2-2. PRU Direct Input and Output Mode Block Diagram

The following sections show how to customize the serial and parallel port through GPIO direct mode. For the serial port, transmission of data is simulated through the GPO pin, for example, the GPO pin status depends on the corresponding bit of the data register.

2.1 The PRU Implement Serial Port

The macro in Figure 2-3 shows one of the ways in which serial data can be sent. The delay compensation allows for adjusting different bit rates. The transmit time for a single bit with no delay compensation should be 7 PRU clock cycles. If the PRU core is running at 333MHz, the cycle time is 21ns, or 47.6MHz.

```
m_send_packet_msb_gpo .macro  dataReg, Temp_reg,bitId, TX_PIN, DELAY_COMPEN
    ldi     Temp_reg ,  bitId ;bits to temporary registers
SEND_BIT_LOOP?:
    qbbc            skip_data_high?, dataReg, Temp_reg ;determine whether bit is 0 or 1
    set             r30,r30,TX_PIN; set PRU<n> pin
    qba             skip_data_low?;jump to delay compensation
skip_data_high?:
    clr             r30,r30,TX_PIN;clr PRU<n> pin
    NOP
skip_data_low?:
    .loop   DELAY_COMPEN
    NOP;delay compensation
    .endloop
    sub             Temp_reg, Temp_reg, 1;move pointer to next bit
    qbne            SEND_BIT_LOOP?, Temp_reg, 0xFF;jump
    .endm
```

Figure 2-3. Macro for Data Transmission

One point to consider is that after the frame data exceeds the maximum length that the PRU core general-purpose registers can hold, additional latency in extracting data from external memory or PRU inside shared memory can directly affect the communication bandwidth. The bandwidth has to be reduced in the case of the higher throughout. In contrast to the pipelined CPU, the PRU_ICSSG integrates Broadside RAM (BS RAM) that can be accessed through instructions Xin and XOUT (10.6 GB/s). This data processing accelerator allows the PRU to read or write up to 32 bytes of data from the BS RAM in a single PRU clock cycle. This is useful in situations with large data volumes, or high communication bandwidth. Data can be temporarily placed within the BS RAM, waiting for the frame data to be transferred, before moving the data out to external memory or shared memory inside the PRU. The steps in Figure 2-4 are performed by the PRU firmware to write to the BS RAM.

```
ldi r10.w0, 0xf000;BS—RAM write auto increment enable
xout 0x30 ,&r10, 2;Store RAM Address and AutoIndexEn to BS RAM using XOUT instruction
xout 0x30 ,&r2,  32;send r2-r9 data to BSRAM,If AutoIndexEn = 1h, repeat thisto write additional data
```

Figure 2-4. Programming the BS RAM

It should be noted that Auto Index enable causes each PRU or RTU_PRU core's write or read to the BS RAM causes the RAM Address to increment by 1. Each increment of the RAM address is the equivalent of 32 Bytes. The next read or write address will always be 32 Bytes later, regardless of the read or write data size. Based on the above discussion, Figure 2-5 and Figure 2-6 show the system block diagram for the PRU GPIO to implement serial communication based on direct mode and the software workflow.

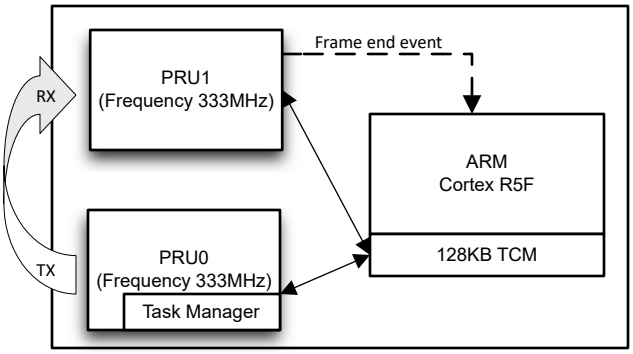

Figure 2-5. PRU Direct Mode System Block Diagram

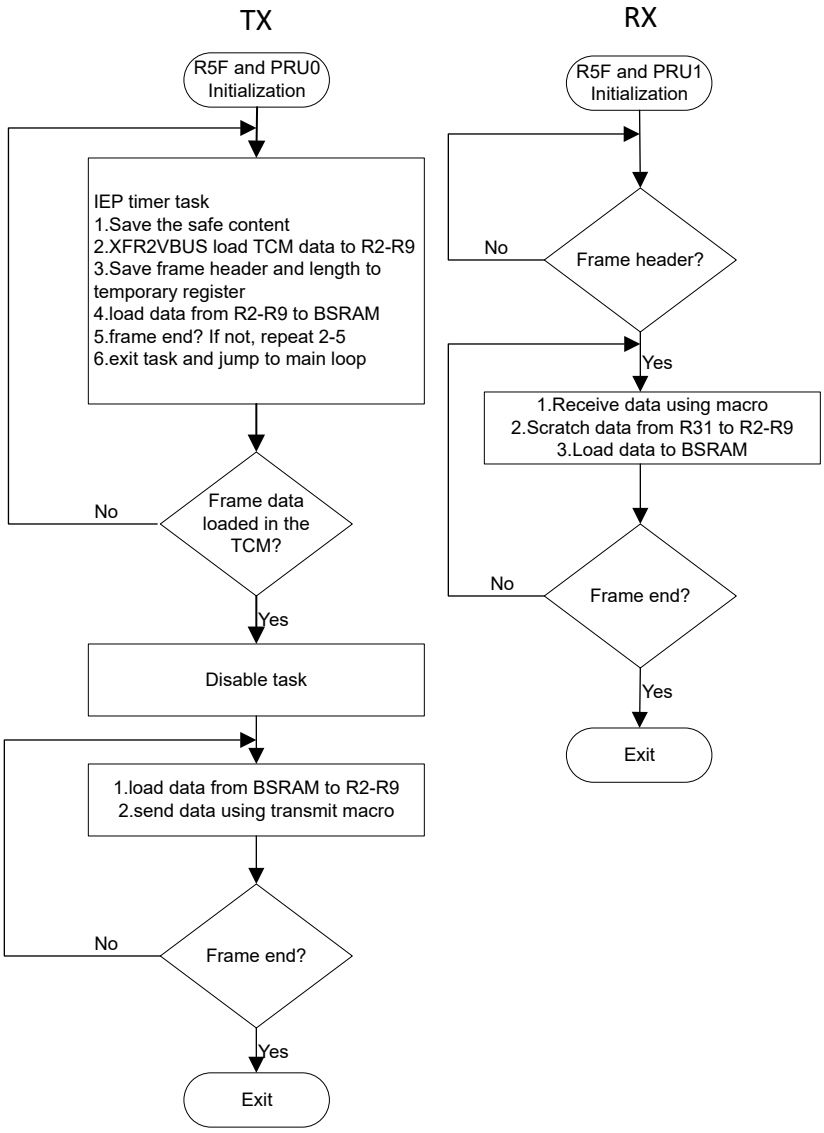

Figure 2-6. Workflow for GPIO Direct Mode

The application core initializes the PRU core before the PRU can start normally, for example, PRU initialize the GPIO interface and Industrial Ethernet Peripheral (IEP), which is hardware work required for Industrial Ethernet functions. The IEP module features an industrial Ethernet timer with 16 compare events, industrial Ethernet sync generator and latch capture, industrial Ethernet watchdog timer, and a digital I/O port (DIGIO). In this case, the data in Tightly Coupled Memory (TCM) can be periodically query through event triggered by the IEP. IEP timer trigger period is set to 10kHz (100µs) to poll the valid data in TCM in the task. It is important to protect context with the Xin, Xout instructions in the task to avoid overlap with register, then use the XFR2VBUS DMA widget to load the data at the address specified by tightly coupled memory (TCM). The XFR2VBUS is a simple hardware accelerator which is used to get the lowest read round trip latency from Memory Subsystem Multi-core (MSMC) and to decouple the latency seen by the PRU. When application core stores the frame data into the TCM, PRU will uses the header data to determine if a data transmission needs to be performed, and pre-stores the data into BSRAM before performing the data transmission to eliminate the effects of memory access latency.

Another method to implement a serial port is GPIO shift out and in mode. In shift out mode, data is shifted out of PRU_DATAOUT pin on every rising edge of PRU_CLK. The shift rate is controlled by the effective divisor of two cascaded dividers applied to the PRU core clock. To achieve a 100Mbit data rate with serial communication, a 100 MHz clock has to be used with a 250 MHz core clock divided by three. To avoid the effect of the data loading process on serial communication, the PRU shift data mode provides two shadow registers (GPO_SH0 and GPO_SH1) that can be used to support ping-pong buffers. The shadow register can be programmed independently via PRUx_R30[29:30], when PRUx_R30[29:30] is set, the data in PRUxR30[0:15]/ [15:0] is loaded into GPO_SH0 and GPO_SH1. The PRU shift-out mode can be set to Free Running mode and Fixed Clock Count Mode. For Free Running Clock Mode, the shift operation will continue until PRUx_ENABLE_SHIFT(PRUxR30[31]) is cleared. When PRUx_ENABLE_SHIFT is cleared, the shift operation will finish shifting out the current shadow register, stop, and then reset. For Fixed Clock Count mode, the number of data bits to be shifted out is defined by configuration register. Figure 2-7 shows the system architecture of the shift out mode.

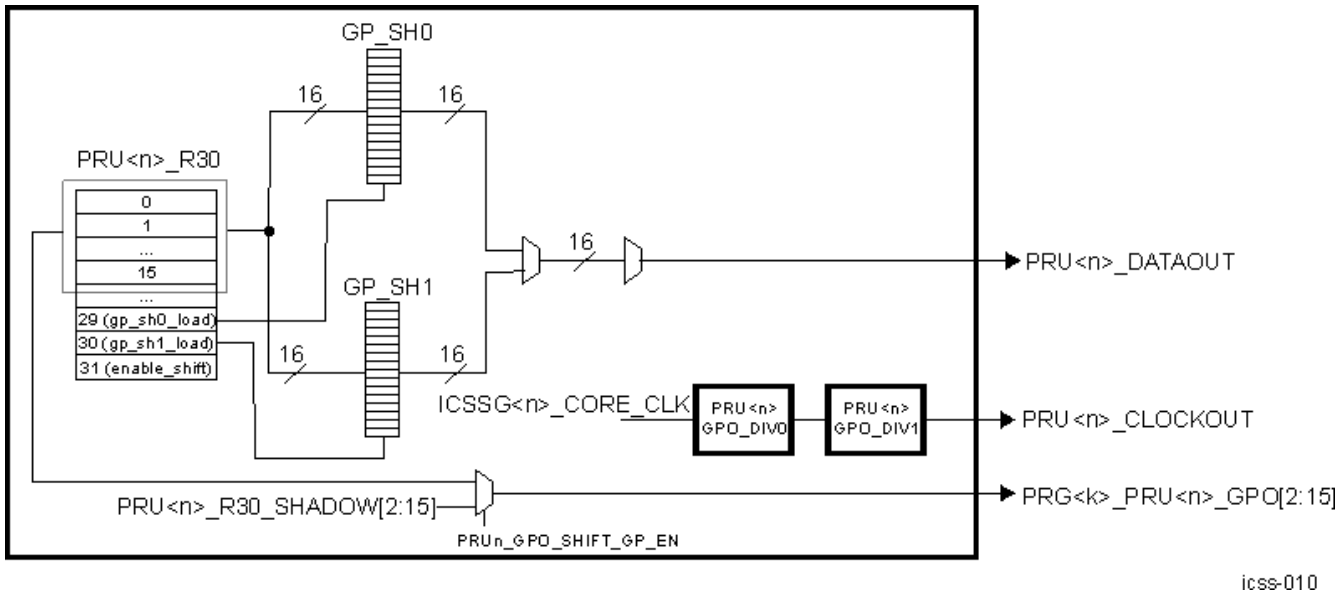

Figure 2-7. PRU R30 (GPO) Shift Out Mode Block Diagram

In 28-bit shift-in mode, general-purpose input pin PRU_DATAIN is sampled and shifted into a 28-bit shift register on an internal clock pulse. The register fills in least-significant bit (LSB) order (from bit 0 to 27) and then overflows into a bit bucket. The 28-bit register is mapped to PRU_R31[0:27], after the start bit (1/0) is detected by the shift in, Cnt_16 is set for every 16 shift clocks and serial data mapped to R31 registers can be copied to other system registers. Similarly, to shift-out mode, a 100MHz sampling clock with 250-MHz core clock is selected. Figure 2-8 shows the block diagram of the PRU GPI shift-in mode. In this application, the transmission is run on Free-Running mode. Figure 2-9 shows the programming work flow.

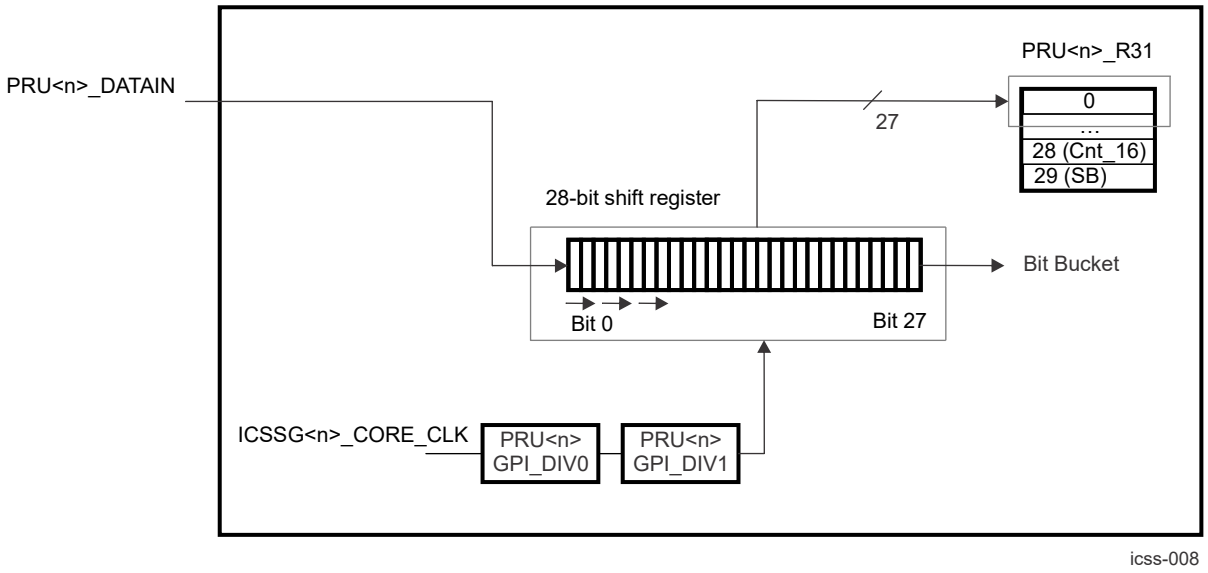

Figure 2-8. PRU R30 (GPO) Shift Out Mode Block Diagram

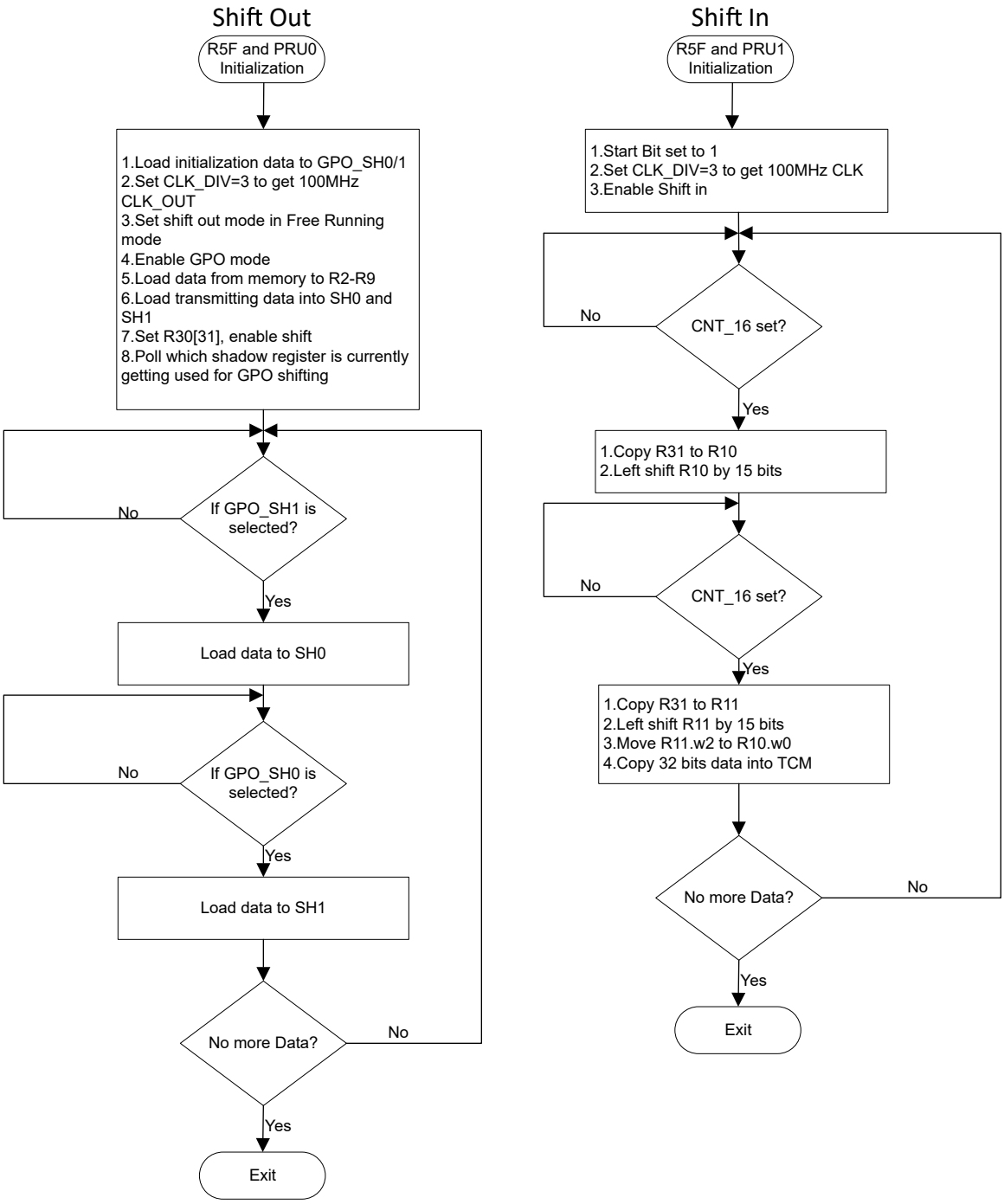

Figure 2-9. PRU Shift Out and In Programming Work Flow

It should be noted that SH0 and SH1 have a depth of 16 bits, and with divide factor of 3, the data load and process time must less than 48 PRU cycles. In shift in mode, the data input node does not know the total byte length of this communication, so it is possible to include the total data length in the start frame to define the number of cycles for the shift in mode.

For more PRUs for serial communication, see the *Intra Drive Communication Using 8b-10b Line Code with Programmable Real Time Unit*, and *FSI Bandwidth-Optimization for Multi-axis Servo Control*

2.2 The PRU Implement Parallel Port

Due to the maximum flexibility of the PRU, this makes the PRU also suitable parallel port application scenarios as well. In this section, the PRU's GPIO direct input and output mode is also applied. The PRU can implement a parallel port with either direct input/output mode, or parallel capture mode. Parallel capture mode uses an external clock to latch data on the rising or falling edge of the clock. Direct input/output mode can be used to provide a clock for the parallel port output. Set and clr instructions can be used to toggle the clock output. For example, set r30, r30, 0 will set the PRUx_GPO0 pin to high voltage level, while clr r30, r30, 0 sets the pin to a low voltage level. The PRU can also compensate for the settling time of the data as well as the hold time due to its determinism of the instructions.

A simple macro for the parallel port is shown in Figure 2-10, where the clock cycles are 5 PRU cycles.

```
m_send_packet_msb_gpo_CLK_shfit .macro  dataReg, clkPin

       lsr r30.b0,dataReg.b3,4
       set r30,r30,clkPin
       nop
       clr r30,r30,clkPin
       nop
       lsr r30.b0,dataReg.b3,0
       set r30,r30,clkPin
       nop
       clr r30,r30,clkPin
       nop
```

Figure 2-10. Macro of Parallel Port Output

3 Verification

To verify the accuracy and bandwidth of serial port data transmission, GPIO direct mode was applied for TX/RX loopback test. R5F stores 64 bytes of data to TCM at address 0x78000000. PRU0 queries TCM in periodic tasks and transmits data, clears the frame header data after transmission to avoid repeated. Meanwhile, PRU1 re-stores the received data to TCM at address 0x78000040 to verify the accuracy of the data received by comparing the transmitted and received data. oscilloscope measures the transmission bandwidth on the transmit side. The test results are shown in Figure 3-1 and Figure 3-2.

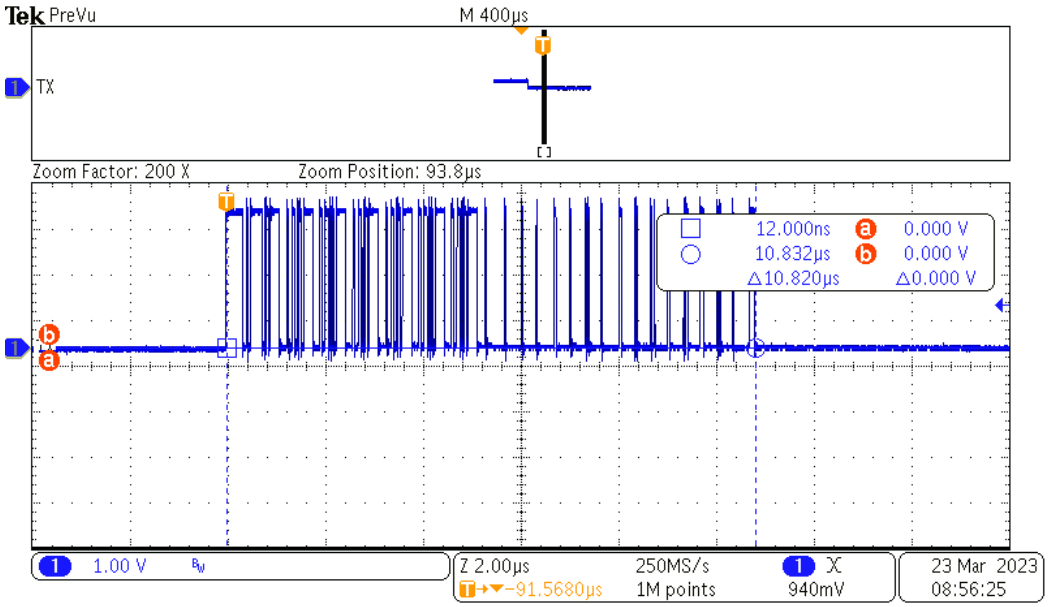

Figure 3-1. 64 Bytes Transmit Time

As shown in Figure 3-1, bandwidth= length of data/time=512 bit/10.82μs=47.32Mbps.

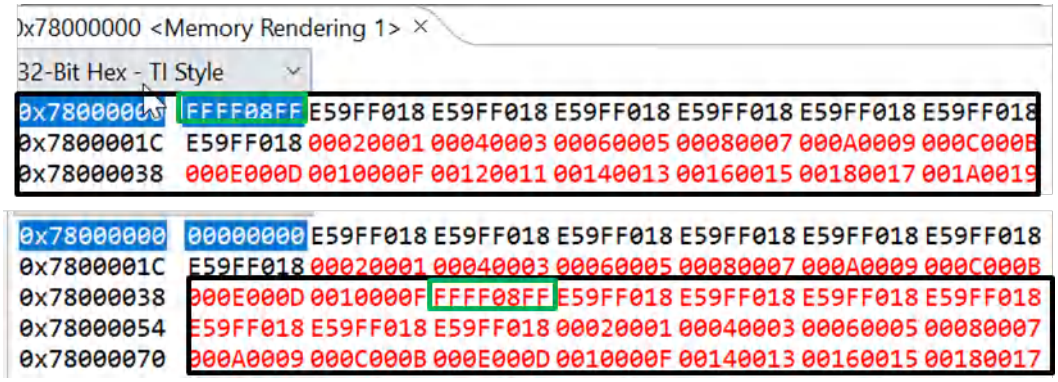

Figure 3-2. Transmit and Receive Data Comparison

PRU_GPIO shift mode is similar to direct mode, 64bytes of TX/RX loopback test was performed, PRU0 in shift out mode, PRU1 in shift in mode, and transmit data is stored at 0x78000114 on R5F TCM. After data reception is complete, stored at 0x78000180.

Figure 3-4 shows that the PRU1 core received the correct data without errors.

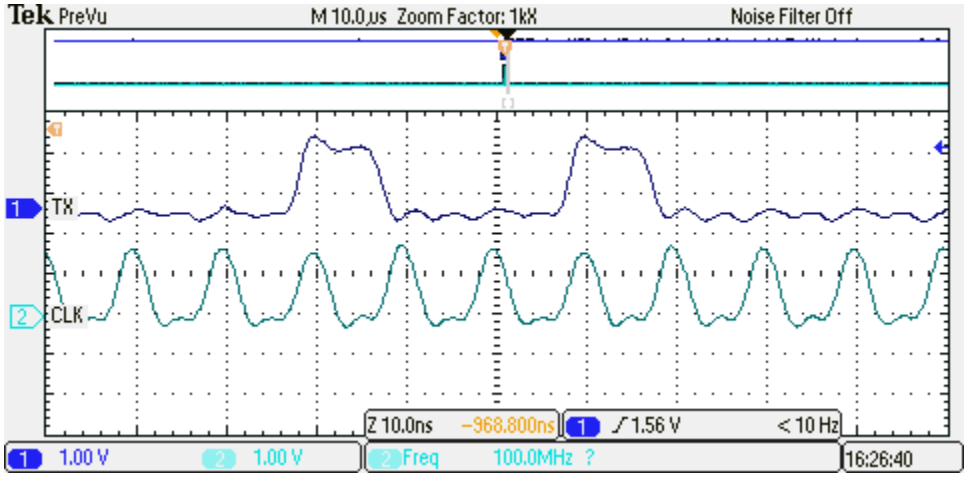

Figure 3-3. Shift Mode

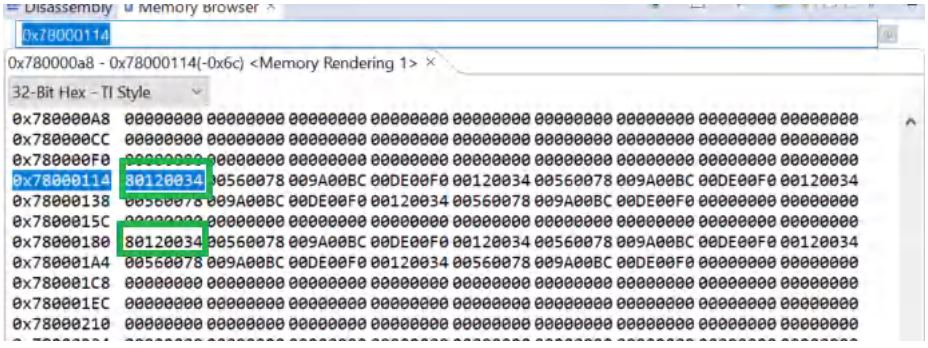

Figure 3-4. Transmit/Receive Data Comparison

The goal was to use GPIO shift out mode to achieve a 100Mbit data rate with serial communication.Figure 3-3 shows that CLK_OUT clock period is 10MHz and there is no error bit between transmit and receive data, which proves the validity of the serial communication. And to improve the reliability of serial communication, the CRC hardware accelerator is integrated inside the PRU_ICSSG to add CRC checksum after each communication frame.

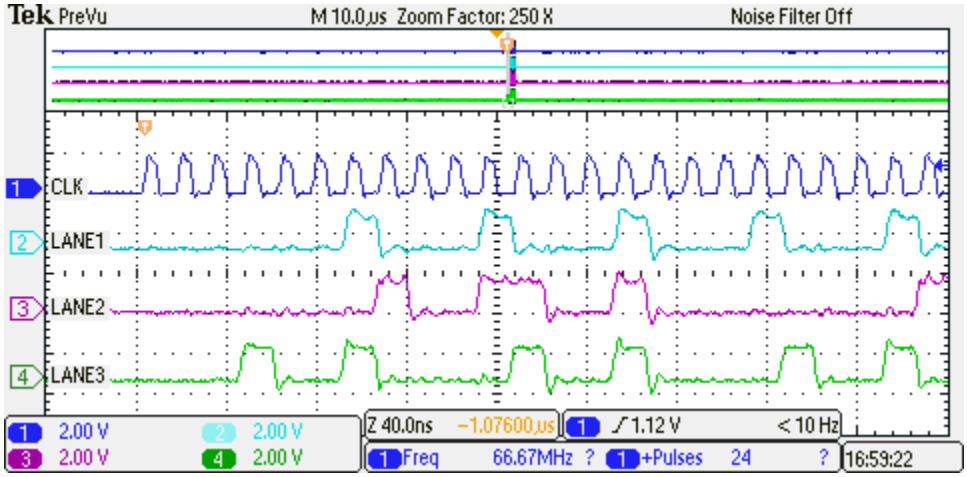

Figure 3-5. 4-Lane Parallel Data Transmission

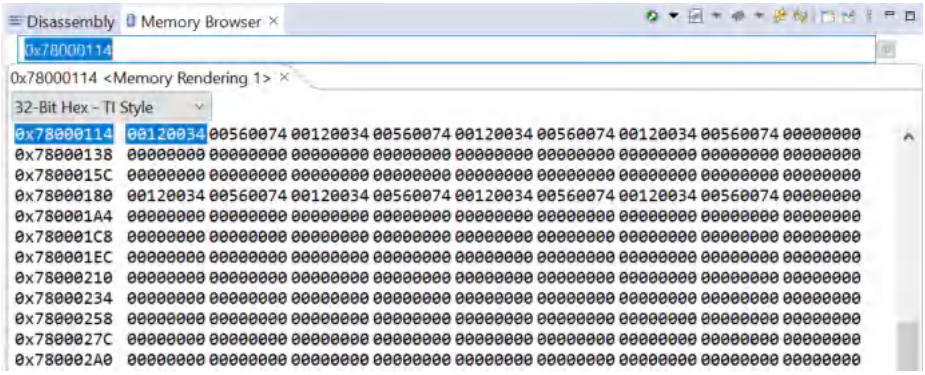

Figure 3-6. Transmit/Receive Data Comparison

To shows the parallel port performance, TX/RX loopback experiment was performed, transmit data is stored at 0x78000114 on R5F TCM. After data reception is complete, stored at 0x78000180. In this case, the clock cycle is equal to 5 PRU cycle, and bandwidth = length of data/time = 4 bits/cycle / (3ns/clock×5clocks/cycle)= 266.4Mbps. As more PRU GPI/GPO signals are used, the data throughput goes up. For example, a bandwidth of 1Gbps could be achieved with 1 clock signal and 15 data lines: bandwidth = length of data/time = 15 bits/ cycle / (3ns/clock×5clocks/cycle) = 1Gbps.

4 Summary

This application provides methods and examples for implementing serial and parallel port using PRU subsystem, which provides a highly flexible, high-bandwidth ways of communication for point to point data exchange in system.

5 References

- Texas Instruments: *AM243x Sitara™ Microcontrollers Data Sheet*
- Texas Instruments: *AM64x / AM243x Processors Silicon Technical Reference Manual*
- Texas Instruments: *PRU Assembly Instruction User's Guide*
- Texas Instruments: *Intra Drive Communication Using 8b-10b Line Code with Programmable Real Time Unit*
- Texas Instruments: *FSI Bandwidth-Optimization for Multi-axis Servo Control*

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265