TI Designs
High Performance Pulse Train Output (PTO) With PRU-ICSS for Industrial Applications

TI Designs

TI Designs provide the foundation that you need including methodology, testing, and design files to quickly evaluate and customize the system. TI Designs help *you* accelerate your time to market.

Design Resources

TIDEP0027	Tool Folder Containing Design Files
AM3359	Product Folder
TPS65910	Product Folder
Industrial SDK	Product Folder
DRV8711EVM	Product Folder
TMDSICE3359	Design Files

TI E2E™ Community

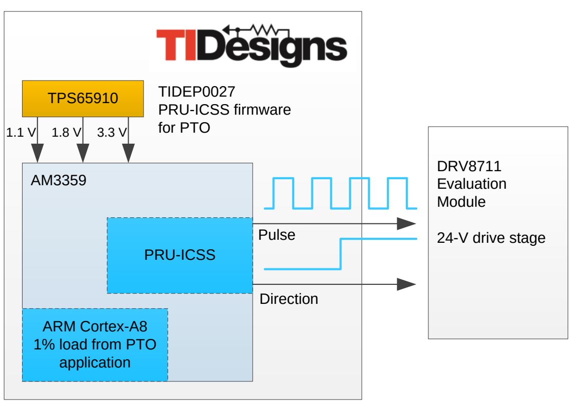
ASK Our E2E Experts
WEBENCH® Calculator Tools

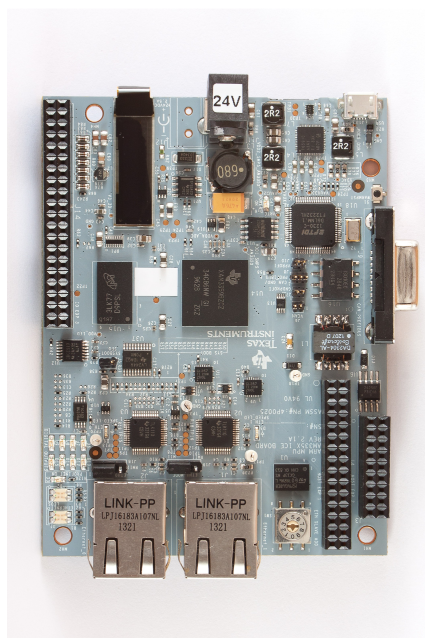
Design Features

- High Speed and Frequency With High Precision for Accurate Pulse Train Output Control and Synchronization
- Implementation Without External Application Specific Integrated Circuit (ASIC) or Field Programmable Gate Array (FPGA)
- Implementation on Programmable Real-Time Unit Subsystem and Industrial Communication Subsystem (PRU-ICSS) With Sitara™ Processor
- Contains PRU-ICSS Firmware in Source Code Adaptable to Customer Needs
- Easy Evaluation With TMDSICE3359 Industrial Communication Engine (ICE) Evaluation Module (EVM)

Featured Applications

- Factory Automation and Process Control
- Programmable Logic Controllers (PLC)
- Motor and Stepper Drives
- Remote Digital Inputs and Outputs (I/Os) With PTO Outputs





An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

1 Key System Specifications

Table 1. Key System Specifications

PARAMETER	SPECIFICATION	DETAILS
Output pins	Two output pins supporting the following modes: 1. Pulse and direction 2. Direction and pulse	PTO output pins are multiplexed by using PRUn GPO pins (PRU register R30 mapped output pins)
Number of channels	One channel per PRU	Multiple channels of PTO with multiple PRUs possible
Precision	50 ns	
Duty cycle	50%	
Max frequency	1 MHz	

2 System Description

2.1 Introduction to Pulse Train Output

PTO modules have uses in a variety of industrial automation applications, with remote I/O modules and PLCs being the most obvious.

PTO modules are used to control stepper and servo motors in industrial factories. The function of the PTO module is to precisely rotate the motor shaft to any given position by providing a specific number of impulses (also known as steps) to the motor.

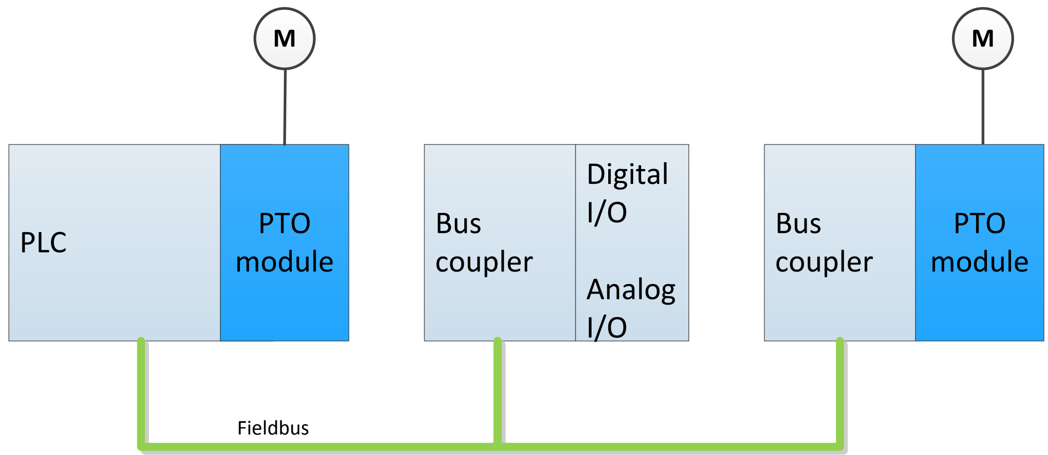


Figure 1. PLC Application With PTO Modules

The hardware building blocks of a PTO module consist of the PTO controller, PTO profile storage, PTO output generator, power stage, and fieldbus interface—see Figure 2.

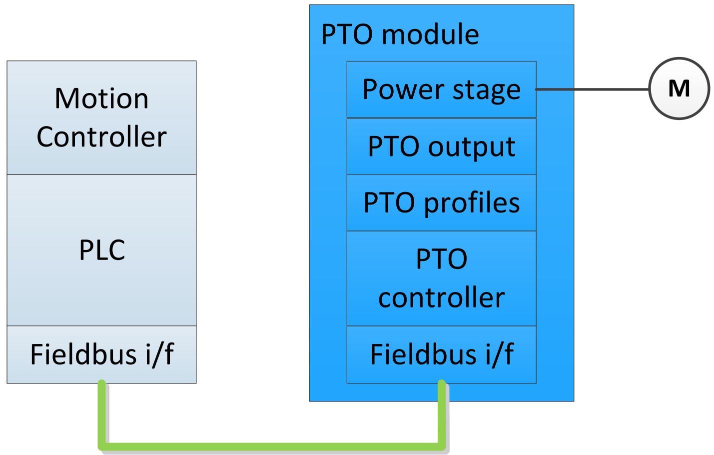


Figure 2. PTO Module Components

The PTO controller receives data through the fieldbus interface. The PTO controller receives the positions and profiles from the PLC. Based on the positions and profiles, the PTO controller generates a list of PTO profile descriptors. These descriptors are stored in shared memory within the PTO module. The PTO output generator accesses the PTO profile descriptors and creates an appropriate PTO output waveform. Finally, the power stage amplifies the PTO output to rotate the motor.

Note that in many cases the PLC operates a motion controller application, which generates complex PTO profiles. In such cases, the PTO module must only perform translation of those profiles into PTO profile descriptors.

The number of steps that a PTO module outputs corresponds to a known rotation of the motor shaft.

A PTO profile consists of a variety of acceleration, constant speed, and decelerations segments (Figure 3). The PTO profile provides a way to describe the speed of the motor over time. Most of the motor applications do not have a position feedback included with the motor; however, the PTO profile defines a specific number of steps to rotate the motor shaft to a known position. The software counts the steps of the motor shaft and stores the information—these measures ensure that the exact position of the motor shaft is always known.

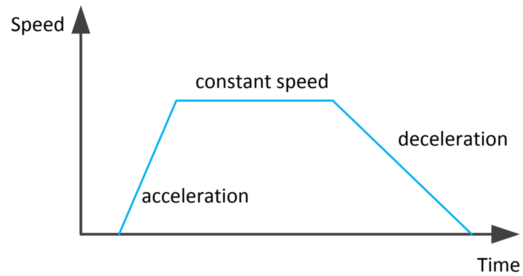


Figure 3. Basic Profile With Segments of Acceleration, Constant Speed, and Deceleration

The PTO output waveform corresponds to a square wave output with a programmable number of pulses and a fixed duty cycle. The number of pulses defines the number of steps for a stepper motor. The frequency of the output defines the speed of the stepper motor. One PTO output pulse is equal to one step of a stepper motor; therefore, the PLC or motion controller knows the exact number amount of steps that are required to precisely position the shaft of the stepper motor.

Figure 4 shows an example of the PTO pulse output for a constant speed segment. The duty cycle is constant and typically at 50%. Because this segment maintains a constant speed the PTO pulse frequency is constant and equal to the speed of the stepper motor.

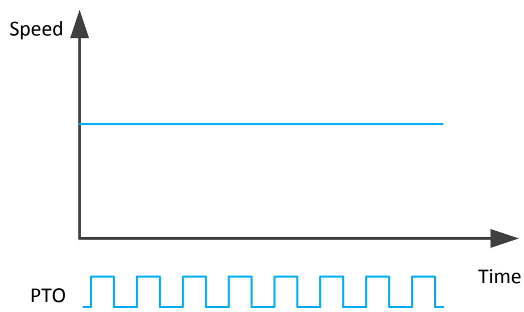


Figure 4. Constant Speed Segment

The next example in Figure 5 shows the PTO segment for acceleration. The duty cycle is still constant (50%), but the PTO output frequency increases. This increase in PTO output frequency results in an increase in the speed of the stepper motor.

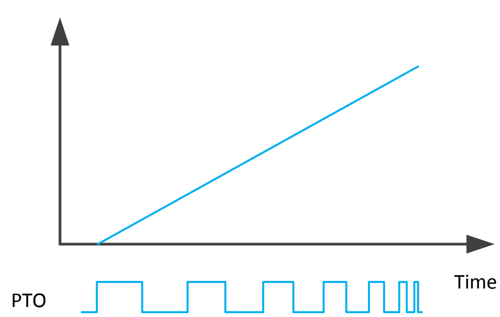


Figure 5. Acceleration Segment

Figure 6 shows the PTO segment of deceleration. Similar to the acceleration segment, the PTO output frequency decreases over time. This decrease in the PTO output frequency results in a decrease in the speed of the stepper motor.

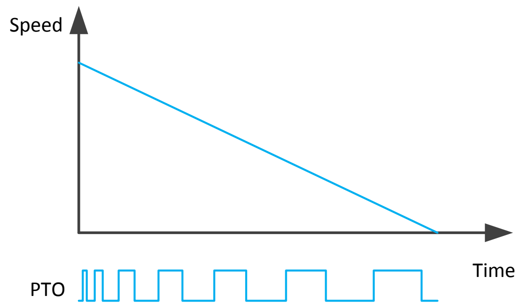


Figure 6. Deceleration Segment

Multiple acceleration, constant speed, and deceleration segments are linked into a sequence that is called a PTO profile (Figure 7). The motion controller generates this type of profile and transfers the profile into the PTO module. The PTO module finally generates the PTO output waveform based on the PTO profile.

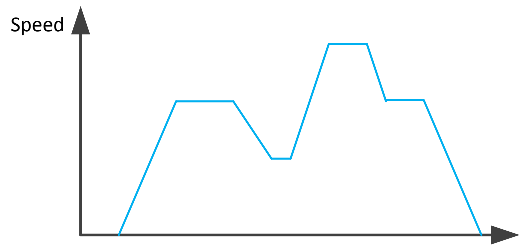


Figure 7. PTO Profile With Multiple Segments

2.2 PTO Function With Sitara Processor in the PRU-ICSS

Traditional PTO solutions use a microcontroller (MCU) timer module, which usually lack in performance in terms of frequency, speed, or precision. Higher performance solutions require an external ASIC, FPGA, or both and are usually more expensive in terms of cost and board space.

The PTO solution with the PRU-ICSS peripheral on Sitara processors provides high speed and frequency with high precision for an accurate PTO control without using the ARM processor resources to generate the PTO output waveform.

This design document describes the implementation and usage of the PTO firmware for the PRU-ICSS peripheral.

2.3 Sitara AM3359 Processor

The Sitara AM335x device family is a low-power processor with an ARM™ Cortex-A8™ RISC core and a broad range of integrated industrial peripherals (Figure 8). The ARM Cortex-A8 supports clock frequency ranges from 300 MHz for simple I/O applications to up to 1 GHz for complex control applications, which require more CPU performance.

The Sitara AM335x processor is a feature-rich device that integrates two PRU-ICSSs.

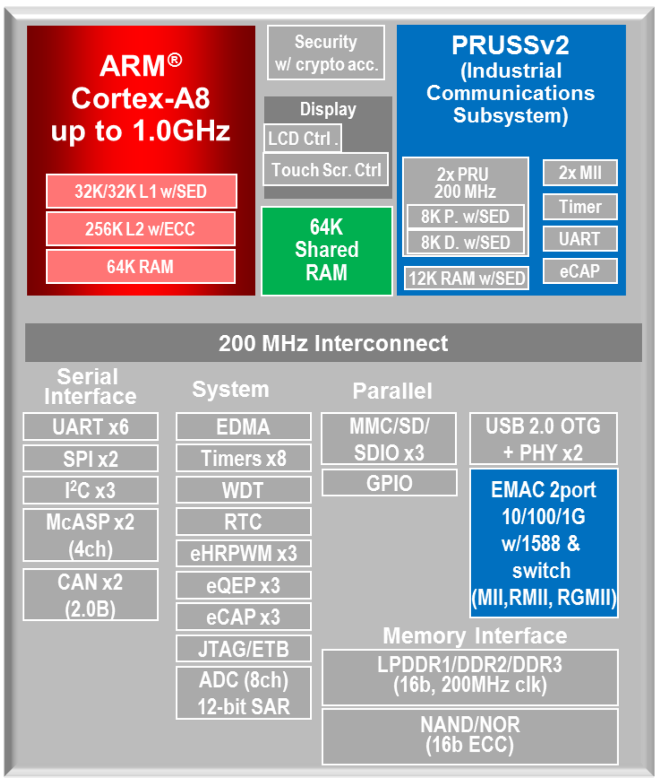


Figure 8. AM335x Family Block Diagram

3 Block Diagram

Figure 9 shows the PTO block diagram within the AM3359 processor.

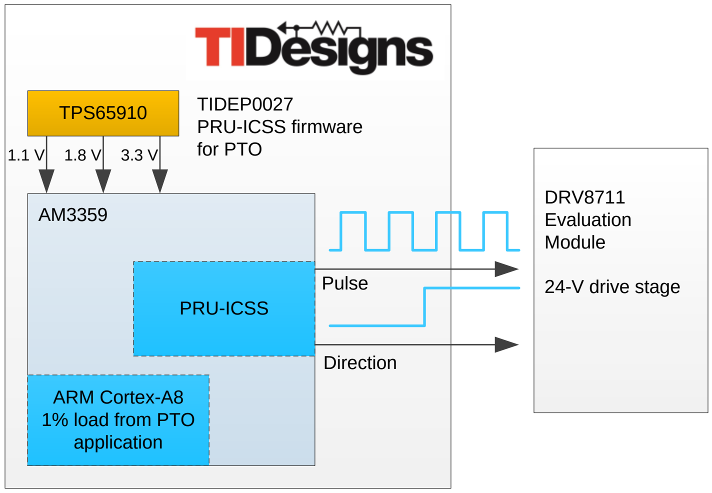


Figure 9. PTO Block Diagram

3.1 Highlighted Products

3.1.1 AM3359

Up to a 1-GHz Sitara ARM Cortex-A8 32-Bit RISC processor

- NEON™ single instruction, multiple data (SIMD) Coprocessor
- 32KB of L1 instruction and 32KB of data cache with single-error detection (parity)
- 256KB of L2 cache with error correcting code (ECC)
- 176KB of on-chip boot read-only memory (ROM)
- 64KB of dedicated random access memory (RAM)
- Emulation and debug - JTAG
- Interrupt controller (up to 128 interrupt requests)

PRU-ICSS

- Supports protocols such as EtherCAT®, Process Field Bus (PROFIBUS), PROFINET, EtherNet/IP™, and more
- Two PRUs
- 32-bit load and store RISC processor capable of running at 200 MHz
- 8KB of instruction RAM with single-error detection (parity)
- 8KB of data RAM with single-error detection (parity)
- Single-cycle 32-bit multiplier with a 64-bit accumulator
- Enhanced general-purpose input and output (GPIO) module provides shift-in and shift-out support and parallel latch on the external signal
- 12KB of shared RAM with single-error detection (parity)
- Three 120-byte register banks accessible by each PRU
- Interrupt controller module (INTC) for handling system input events
- Local interconnect bus for connecting internal and external masters to the resources inside the PRU-ICSS

- Peripherals inside the PRU-ICSS:
  - One universal asynchronous receiver and transmitter (UART) port with flow control pins—supports up to 12 Mbps
  - One enhanced capture (eCAP) module
  - Two MII Ethernet ports that support industrial Ethernet, such as EtherCAT
  - One management data input and output (MDIO) port

On-chip Memory (shared L3 RAM)

- 64KB of general-purpose on-chip memory controller (OCMC) RAM
- Accessible to all masters

External memory interfaces (EMIF)

- Mobile double data rate (mDDR or LPDDR), DDR2, DDR3, and DDR3L controller:
  - mDDR: 200-MHz clock (400-MHz data rate)
  - DDR2: 266-MHz clock (532-MHz data rate)
  - DDR3: 400-MHz clock (800-MHz data rate)
  - DDR3L: 400-MHz clock (800-MHz data rate)
  - 16-bit data bus
  - 1GB of total addressable space
  - Supports one x16 or two x8 memory device configurations
- General-purpose memory controller (GPMC):
  - Flexible 8-bit and 16-bit asynchronous memory interface with up to seven chip selects (including NAND, NOR, Muxed-NOR, and SRAM)
  - Uses Bose, Chaudhuri, and Hocquenghem (BCH) code to support 4-, 8-, or 16-bit ECC
  - Uses hamming code to support 1-bit ECC

See the AM3359 data sheet for a complete list of features.

3.1.2 TMDSICE3359 Industrial Communication Engine EVM

Hardware specifications

- AM3359 ARM Cortex-A8
- DDR3, NOR flash, and serial peripheral interface (SPI) flash
- Organic light-emitting diode (OLED) display
- TPS65910 power management
- 24-V power supply
- USB cable for the JTAG interface and serial console

Software and tools

- SYS/BIOS real-time operating system (OS)
- StarterWare™
- Code Composer Studio™ (CCS) integrated development environment (IDE)
- Application stack for industrial communication protocols
- Sample industrial applications

Connectivity

- PROFIBUS interface
- CANOpen
- EtherNet/IP
- PROFINET
- Sercos III
- I/O
- SPI
- UART
- JTAG

View the TMDSICE3350 product folder for a complete list of features and design resources.

4 System Design Theory

A PTO solution with Sitara processors consists of the following blocks:

- ARM
  - Industrial application with PTO control function on the ARM
- PRU-ICSS
  - PTO firmware in the source code
  - Descriptor list in the PRU-ICSS shared memory

The TIDEP0027 includes the PRU-ICSS firmware source code with the descriptor list definition.

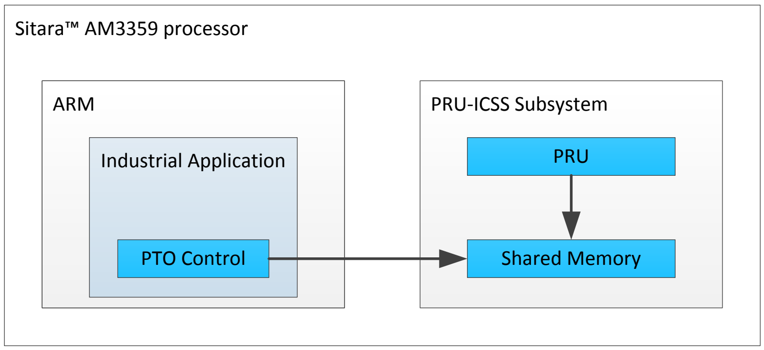


Figure 10. PTO Building Blocks

The focus of this design theory section is on the PTO firmware in the PRU-ICSS, and the PTO descriptor list in the shared memory. Additionally, this section provides an example calculation to generate the PTO descriptor list for shared memory.

The industrial application on the ARM has the control ability of the PTO within the PRU-ICSS. The application calculates the PTO descriptor list and configures the shared memory appropriately. Once configured, the ARM application loads and starts the PRU firmware for PTO function into the PRU-ICSS subsystem.

Based on the descriptor list in shared memory, the PTO firmware starts processing the profile segments. The firmware processes every descriptor until the PTO stop condition in the shared memory is set. The firmware then transitions the PRU into HALT state. As soon as the HALT state occurs, the industrial application can reprogram the descriptor list and then restart the PTO firmware.

The PRU-ICSS uses the register mapped GPO pin (pruN_R30) to generate the PTO pulse and direction output.

4.1 PTO Algorithm on PRU-ICSS

The PTO implementation algorithm in PRU-ICSS is based on *Generate stepper-motor profiles in real time* by David Austin and *Linear motor control without the math* by Pramod Ranade. The algorithm is used to convert physical quantities like speed into frequency.

The main advantage of the algorithm is the sole use of addition and subtraction operations to generate the pulses, allowing the algorithm to auto-correct the granularity errors. Multiplication and division operations are avoided by accumulating speed into distance after every pulse of raster time. Whenever the distance value exceeds or equals the expected position generated by the distance value, the expected position value subtracts from the distance value and a pulse generates on the register-mapped GPO pin.

Figure 11 shows two speed examples. The given time raster on the x-axis shows the decision points, which is the highest frequency at which the algorithm operates in the PRU firmware. The time granularity is 50 ns for the PRU firmware. At each step, the PRU firmware adds distance, which is then compared to the next step on the y-axis. If the accumulated distance is equal to or more than the compared value, the PTO pulse is generated.

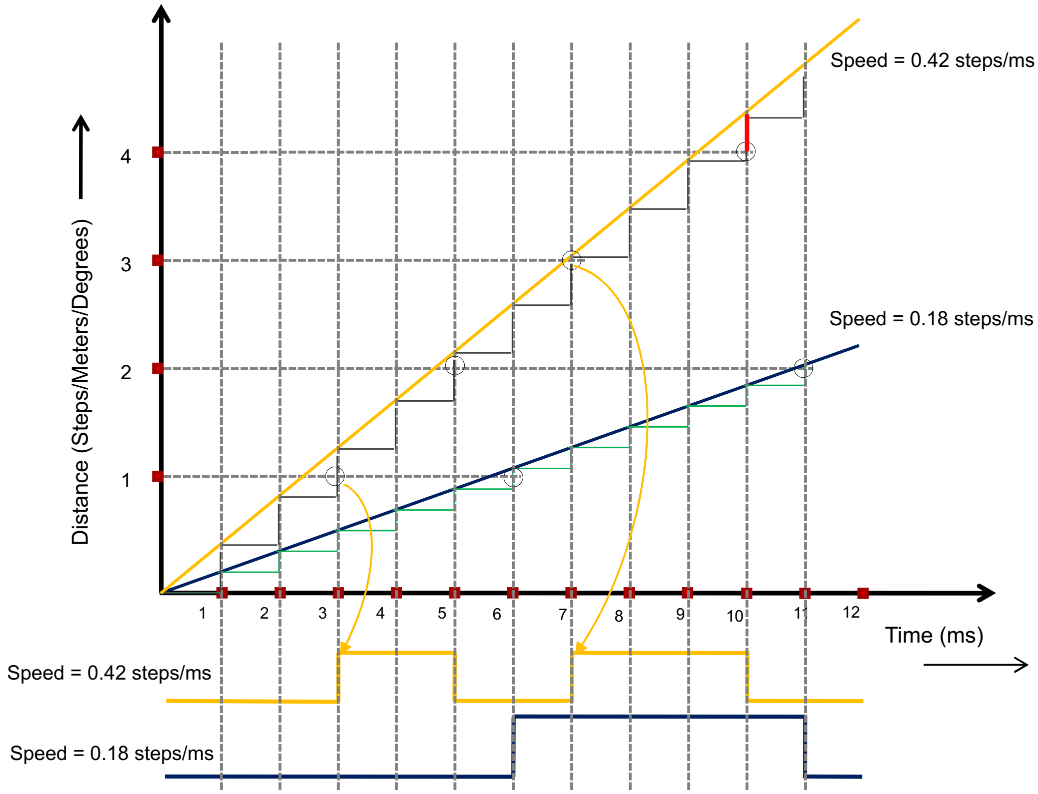


Figure 11. Two Speed Examples With the Generation of PTO Output

Figure 12 shows the flow chart of the PRU firmware for generating a pulse. After the initialization process, (load the speed value, calculate the expected position, and initialize the distance), the PRU code operates in a loop that repeats every 50ns. If the distance value is equal to or greater than the expected position, the PRU firmware generates a pulse (high-to-low or low-to-high transition).

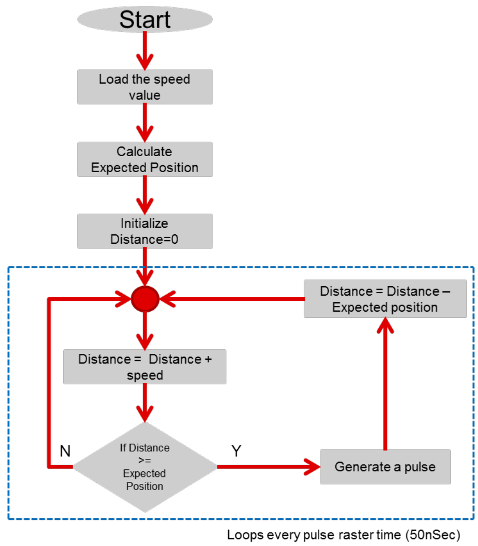


Figure 12. PRU-ICSS Firmware Control Loop for Generating PTO Pulse

4.2 PTO Descriptor List in Shared Memory

The ARM application must write the PTO descriptor list into the shared memory of the PRU-ICSS, which is located at 0x4a31.0000. The total memory size of the shared memory is 12KB. Each descriptor list entry has a size of 32 bytes, which means that the shared memory can handle up to 384 descriptors.

Table 2 shows the structure of the descriptor list in the shared memory.

Table 2. Profile Descriptor List in Shared Memory of PRU-ICSS

MEMORY OFFSET	BIT RANGE	USAGE	COMMENTS
0x00	31:16 15:0	Current Block Next Block	Specifies the current block Specifies the pointer to the next block
0x04	31 30:0	Direction bit Speed value	Specifies the direction (clockwise or counterclockwise) Specifies the speed
0x08	31:0	Acceleration value	Specifies the acceleration or deceleration; if ‘0’ then constant speed is defined
0x0C	31:0	Compare value	Specifies the step time
0x10	31:0	Ramp Time	Specifies the duration of one profile segment
0x14	31:24 23:16 15:0	Acc_type Index Acc_div	Specifies acceleration(1) or deceleration(0) Reserved Divider used to support low frequency steps
0x18	31:0	Match value	Reserved
0x1C	31:16 15:8 7:0	Status PTO Pin 1 PTO Pin 2	Specifies if profile continues or stops with this entry PTO pulse output Direction output

The last valid entry of the profile descriptor list must have the stop bit set, which halts the PRU and enables the ARM application to make updates to the descriptor list.

Note that the last descriptor can link back to any of the other profile descriptors to run a continuous loop. Also note that the descriptors can be linked in a non-linear order.

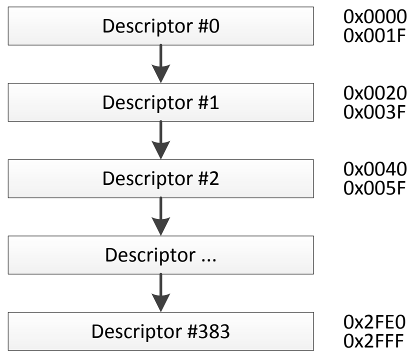


Figure 13. Descriptor List With Memory Offset of Shared Memory

4.3 Profile Example Speed Profile Calculation

This section provides an example calculation of a speed profile with a detailed description of the 32-byte descriptor information, which must be written into the shared memory location.

4.4 Motor Parameters

Typically there are several parameters known with respect to the motor used. In this example, a stepper motor is used with the following specifications.

Start Speed:	400 steps per second (SPS)
Stop Speed:	200 SPS
Max Speed:	1000 SPS

4.5 Fixed Parameters

Acceleration:	2000 steps/s^2
Deceleration:	2000 steps/s^2
Acc_div:	10
Acc_raster:	1 000 000 / Acc_div
Pulse_position_raster:	10 000 000 µs
Dma_slot_per_sec:	1 000 000

4.6 Baseline Calculation

From the parameters given in Section 4.5, the user can calculate in advance the maximum number of steps the motor moves in the acceleration and deceleration ramp.

The distance calculates as:

$d = vi \times t + ½ \times a \times t^2$

where

- d = distance
- vi = initial velocity
- vf = final velocity
- a = acceleration
- t = time (1)

The equation simplifies to:

$vf^2 = vi^2 + 2 \times a \times d$

$vf = vi + a \times t$

$d = ((vi + vf) / 2) \times t$ (2)

The remaining equations derive from Equation 2.

Equation 3 calculates the acceleration steps (aSteps) and deceleration steps (dSteps):

$aSteps = ((Max\ speed)^2 – (Start\ speed)^2) / (2 \times Acceleration)$

$aSteps = ((1000)^2 – (400)^2) / (2 \times 2000)$

$aSteps = 240$ (3)

The maximum number of steps possible for the speed profile of a triangular profile is:

$tSteps = 450$ (4)

A trapezoidal profile must be followed if the total number of steps is more than 450.

Equation 5 calculates the greatest common divisor (GCD) between the acceleration and the raster. The GCD is calculated with the Euclid function.

GCD = gcd(Acceleration, Acc_raster) (5)

which results in:

gcd(2000, 100 000) = 2000

The next calculation is the base multiplier.

Base multiplier	= Acc_raster / GCD
Base multiplier	= 50

The following calculations of additional parameters are necessary to calculate the speed profiles:

rMax	= Max speed × Base multiplier
rMax	= 50 000
rStart	= Start speed × Base multiplier
rStart	= 20 000
rStop	= Stop speed × Base multiplier
rStop	= 10 000
rAccel	= Acceleration / GCD
rAccel	= 1
rCmp	= Base multiplier × Pulse_position_raster
rCmp	= 500 000 000

4.7 Triangular Profile Example

In this example, the motor must move 300 steps. A triangular profile can be chosen based on the results from Equation 4, where the number of steps is less than 450 steps.

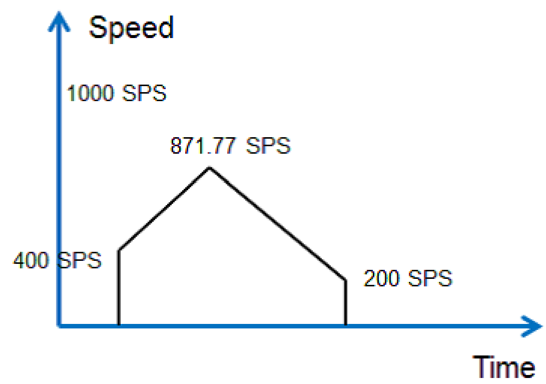


Figure 14. Triangular Profile

The first calculation is of the max speed, which can be achieved by moving 300 steps in a triangular profile:

Top speed = sqrt (((2 × Acceleration × Deceleration × Total steps) + ((Start speed^2) × Deceleration) + ((Stop speed^2) × Acceleration) / (Acceleration + Deceleration))

The equation results in:
Top speed = 871.77 SPS (6)

The next calculation is the time (in µs) required for the acceleration ramp:

Taccel = (Top speed – Start speed) / Acceleration
Taccel = 235 889 µs

Then calculate the time required for the deceleration ramp:

Tdecel = (Top Speed – Stop Speed) / Deceleration
Tdecel = 335 889 µs

4.8 Descriptor List Example

Each block entry of the 32-byte descriptor information block is described in this section for the triangular profile example.

Block entry 1: Current block and next block

The first entry of the 32-byte information is the block number. There is 12KB of shared memory, which indicates there is a total of 384 descriptor blocks. The first block number is initialized as 0 and then incremented by 32 for every new block of information written. The block pointer always points to the starting address of the next block, which is the second block in this example.

Offset 0x00: 0x0000.0020

Block entry 2: Direction bit and speed value

The second block entry consists of the speed value and direction bit, which are encoded into one 32-byte entry. Bit 31 specifies the direction of the movement. All remaining bits are used for the speed value.

The speed value is the start speed of each ramp. In the example of the triangular profile, where the first part is the acceleration ramp, the speed value is:

Speed value (1) = Start speed × Base multiplier
= 20 000
= 0x4E20

For the deceleration ramp, the speed value is:

Speed value (2) = Top speed × Base multiplier
= 43 588
= 0xAA44

For a trapezoidal profile, the speed value for the constant speed ramp and the deceleration ramp is:

Speed value = Max speed × Base multiplier
= 50 000
= 0xC350

The direction bit 31 is set as follows:

Bit 31 = '1' → Forward
Bit 31 = '0' → Backward

The first descriptor entry for the example is configured as:

Offset 0x04: 0x8000.4E20 // FORWARD, 20000

Block entry 3: Acceleration value

The next entry to be written is the acceleration value.

rAccel = Acceleration / GCD
= 1

Offset 0x08: 0x0000.0001

Block entry 4: Compare value

The next entry is the compare value.

rCmp = Base multiplier × Pulse_position_raster
= 500 000 000
= 0x1DCD6500

Offset 0x0C: 0x1DCD6500

Block entry 5: Ramp time

The next entry to be written is the ramp time. The value is in microseconds.

For the acceleration ramp, the time required to reach the top speed from the start speed is written.

Taccel = 235 889
= 0x39971

For the deceleration ramp, the time required to reach the stop speed from the top speed is written.

Tdecel = 335 889
= 0x52011

The first descriptor entry for the example is configured as:

Offset 0x10: 0x0003.9971

Block entry 6: Acc_type, Index, and Acc_div

The Acc_type, Index, and Acc_div are encoded into one entry.

The Acc_type and Index take 8 bits each and the 16 bits are provided for the Acc_div.

The Acc_div is a fixed-parameter:

Acc_div = 10
= 0x000A

The acceleration type is 1 if it is an acceleration ramp, for deceleration it is 0:

Acc_type = 1 // acceleration for the first acceleration ramp
= 0x000A

The Index is currently reserved and must be filled out with 0.

The first descriptor entry for the example is configured as:

Offset 0x14: 0x0100.000A

Block entry 7: Match value

This entry is reserved and must be filled out with 0.

Offset 0x18: 0x0000.0000

Block entry 8: Status, PTO pin 1, and PTO pin 2

Bit 31 of the block entry defines the PRU stop condition. If this bit is set, the PRU stops the execution after executing this block.

PTO pins 1 and 2 define the PRU general-purpose out pin numbers that are used for step and direction. In the TMDSICE3359 EVM example, the pin numbers are configured for R30[1] and R30[2].

The first descriptor entry for the example is configured as:

Offset 0x1C: 0x0000.0201

4.9 Shared Memory Configuration

The descriptor must be written by the ARM application. The previous calculated values fill out the shared memory descriptor list for the triangular profile as follows:

First descriptor

0x4A31.0000	0x0000.0020
0x4A31.0004	0x8000.4E20
0x4A31.0008	0x0000.0001
0x4A31.000C	0x1DCD.6500
0x4A31.0010	0x0003.9971
0x4A31.0014	0x0100.000A
0x4A31.0018	0x0000.0000
0x4A31.001C	0x0000.0201

Second descriptor

0x4A31.0020	0x0020.0040
0x4A31.0024	0x8000.AA44
0x4A31.0028	0x0000.0001
0x4A31.001C	0x1DCD.6500
0x4A31.0030	0x0005.2011
0x4A31.0034	0x0000.000A
0x4A31.0038	0x0000.0000
0x4A31.003C	0x8000.0201

5 Getting Started

5.1 Overview

The following tools are required for the evaluation of the PTO firmware on the PRU-ICSS:

- Personal computer (PC) with an installation of Code Composer Studio Version 6 (CCSv6)
- PRU Compiler from the CCSv6 App Center
- TMDSICE3359 ICE EVM with a USB cable connected to the PC that has CCSv6 installed

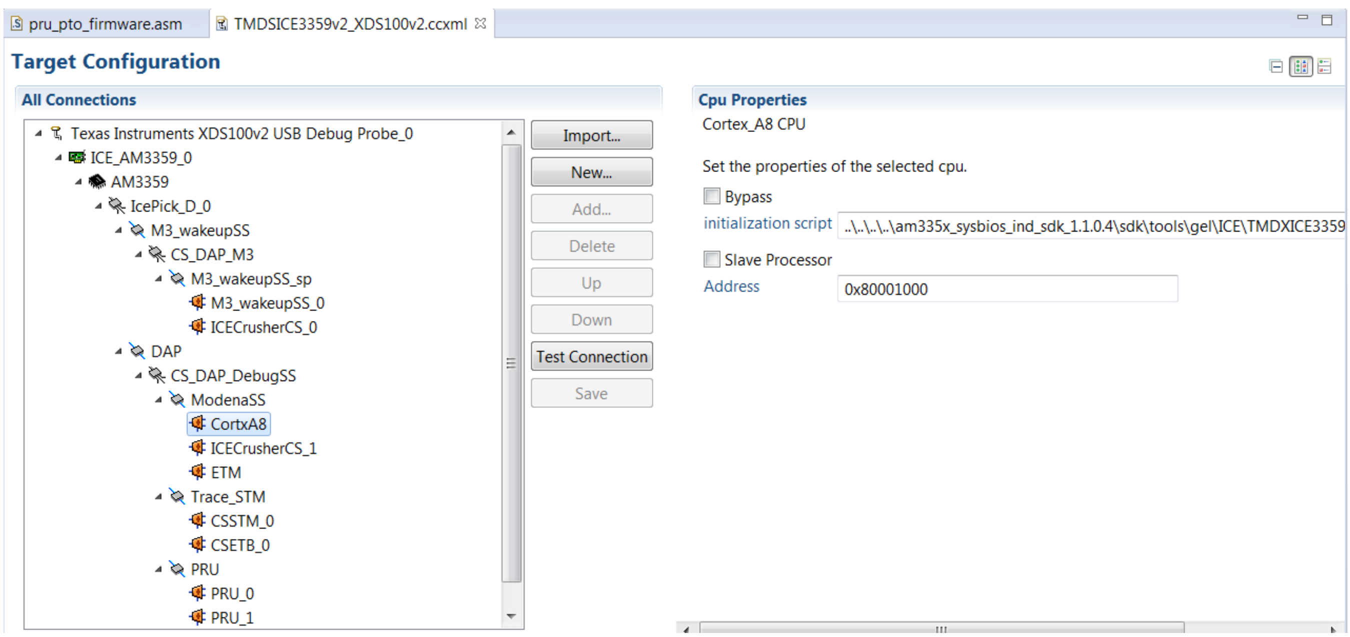
---

NOTE: The TMDSICE3359 EVM has an on-board JTAG support through the XDS100v2 JTAG emulator. For other TI development boards with Sitara processors and a PRU-ICSS, an additional JTAG debugger may be required to connect the board via the JTAG to CCSv6. If such a JTAG connection exists, the PTO example can be evaluated with those boards, as well.

5.2 PTO Firmware Load Sequence

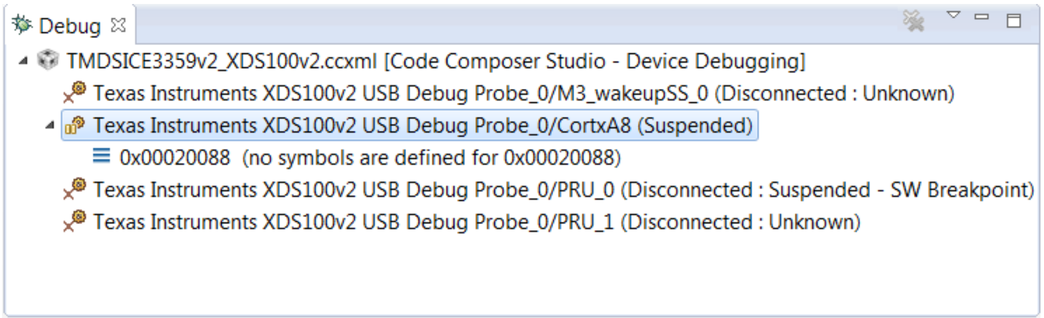
To load the PRU firmware into the PRU-ICSS, the following steps must be performed.

1. Connect the PC via USB cable to the ICE EVM. If connecting for the first time, this initiates the installation of the JTAG over the USB driver for the EVM.
2. In CCSv6, create a new *Target Configuration* file for the ICE EVM. Choose the appropriate .gel init file for the TMDSICE3359. Check using the *Test Connection* button that the *Target Configuration* file setup is satisfactory.

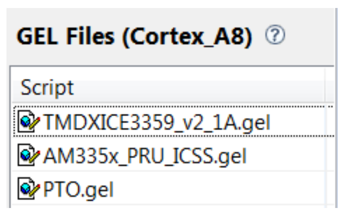


3. Launch the *Target Configuration* file to connect through the JTAG to the TMDSICE3359 board.

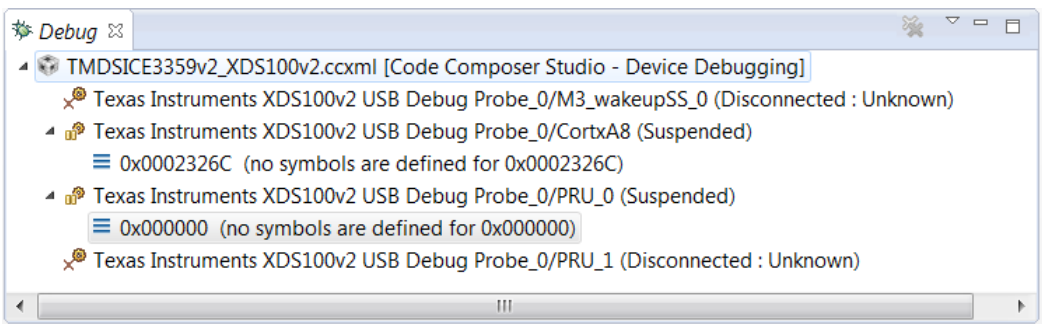
4. In the *Debug* window, connect to the Cortex A8 processor; if required, send the AM3359_ICE_Initializiation .gel script. This script initializes the AM335x on the TMDSICE3359 board.



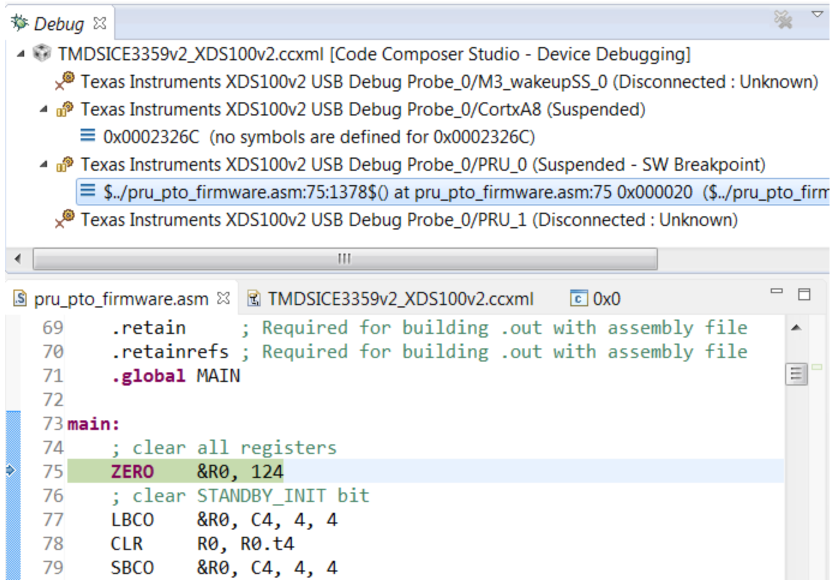
5. Load the additional .gel files to enable initialization of the PRU-ICSS and PTO functions.



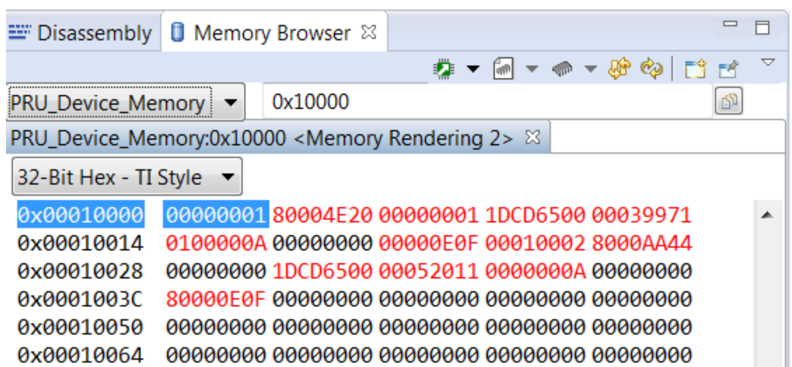
6. With the Cortex_A8 still selected in the *Debug* window, send the PRU_ICSS_init, PTO_Pinmux, and the PTO_WriteExamplePTODescriptorToSharedMemory .gel scripts.
7. Select and connect to PRU_0 in the *Debug* window.



8. Load the PRU program into *PRU_0*.



9. Revisit the shared memory location at 0x1.0000 via the *Memory Browser* window and check the PTO descriptor list. Modify as needed.



10. Run the program. The output can be observed with a scope connected to the PTO pins.

5.3 PTO .gel Script Additions

5.3.1 PRU-ICSS Init .gel Script

After the power cycle, the PRU-ICSS is in power-down mode. The PRU-ICSS must be enabled by the ARM before CCS can connect the PRU. The .gel script used to enable the PRU-ICSS is located on the BeagleBoard GitHub website: https://github.com/beagleboard/am335x_pru_package.

Please follow the instructions provided on the GitHub website to add the AM335x_PRU_ICSS.gel into the CCS project. The GitHub website offers a tutorial CCS_PRU_Debugger-training.pdf to provide further information on how to use the PRU with CCS.

5.3.2 Pinmux Configuration .gel Script

The appropriate pin-multiplexing (pinmux) must be applied to the board in order to access the PTO pins on the TMDSICE3359 board. This example uses the register mapped GPO at R30[14] and R30[15] of the PRU0.

The .gel example configures the pinmux and sets up an example descriptor list in the shared memory. TI™ recommends creating a new .gel file (for example, PTO.gel) that includes the following .gel text example.

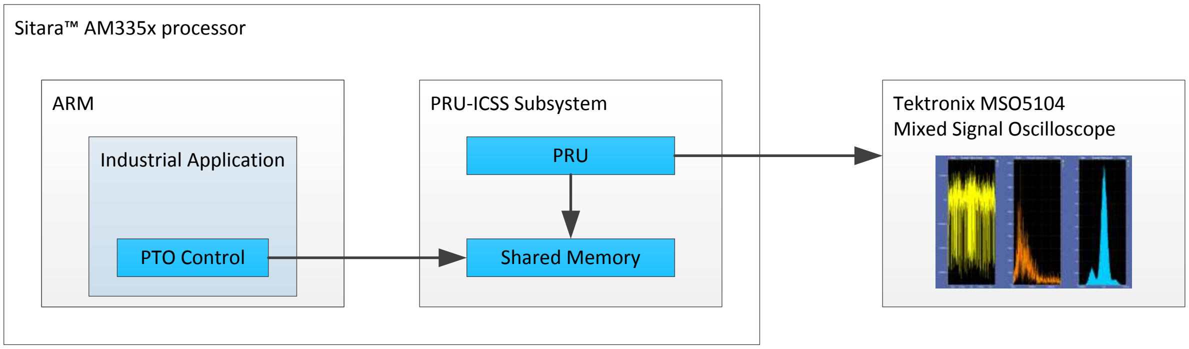
```
menuitem "PTO with PRU_ICSS"
hotmenu PTO_Pinmux()
{
        *((unsigned int*) (0x44E10000 + 0x830)) = 0xE;    // pr1_pru0_pru_r30_14 (mode 6),
pullup/pulldown disabled
        *((unsigned int*) (0x44E10000 + 0x834)) = 0xE;    // pr1_pru0_pru_r30_14 (mode 6),
pullup/pulldown disabled
}

hotmenu PTO_WriteExamplePTODescriptorToSharedMemory()
{
    // First descriptor
    *((unsigned int*) (0x4A310000 + 0x00)) = 0x00000020;
    *((unsigned int*) (0x4A310000 + 0x04)) = 0x80004E20;
    *((unsigned int*) (0x4A310000 + 0x08)) = 0x00000001;
    *((unsigned int*) (0x4A310000 + 0x0C)) = 0x1DCD6500;
    *((unsigned int*) (0x4A310000 + 0x10)) = 0x00039971;
    *((unsigned int*) (0x4A310000 + 0x14)) = 0x0100000A;
    *((unsigned int*) (0x4A310000 + 0x18)) = 0x00000000;
    *((unsigned int*) (0x4A310000 + 0x1C)) = 0x00000201;


    // Second descriptor
    *((unsigned int*) (0x4A310000 + 0x20)) = 0x00200040;
    *((unsigned int*) (0x4A310000 + 0x24)) = 0x8000AA44;
    *((unsigned int*) (0x4A310000 + 0x28)) = 0x00000001;
    *((unsigned int*) (0x4A310000 + 0x2C)) = 0x1DCD6500;
    *((unsigned int*) (0x4A310000 + 0x30)) = 0x00052011;
    *((unsigned int*) (0x4A310000 + 0x34)) = 0x0000000A;
    *((unsigned int*) (0x4A310000 + 0x38)) = 0x00000000;
    *((unsigned int*) (0x4A310000 + 0x3C)) = 0x80000201;
}
```

6 Test Setup

The PTO output pin of the TMDSICE3359 EVM is connected to the Tektronix MSO5105 Mixed Signal Oscilloscope.



7 Test Data

7.1 Jitter Measurement

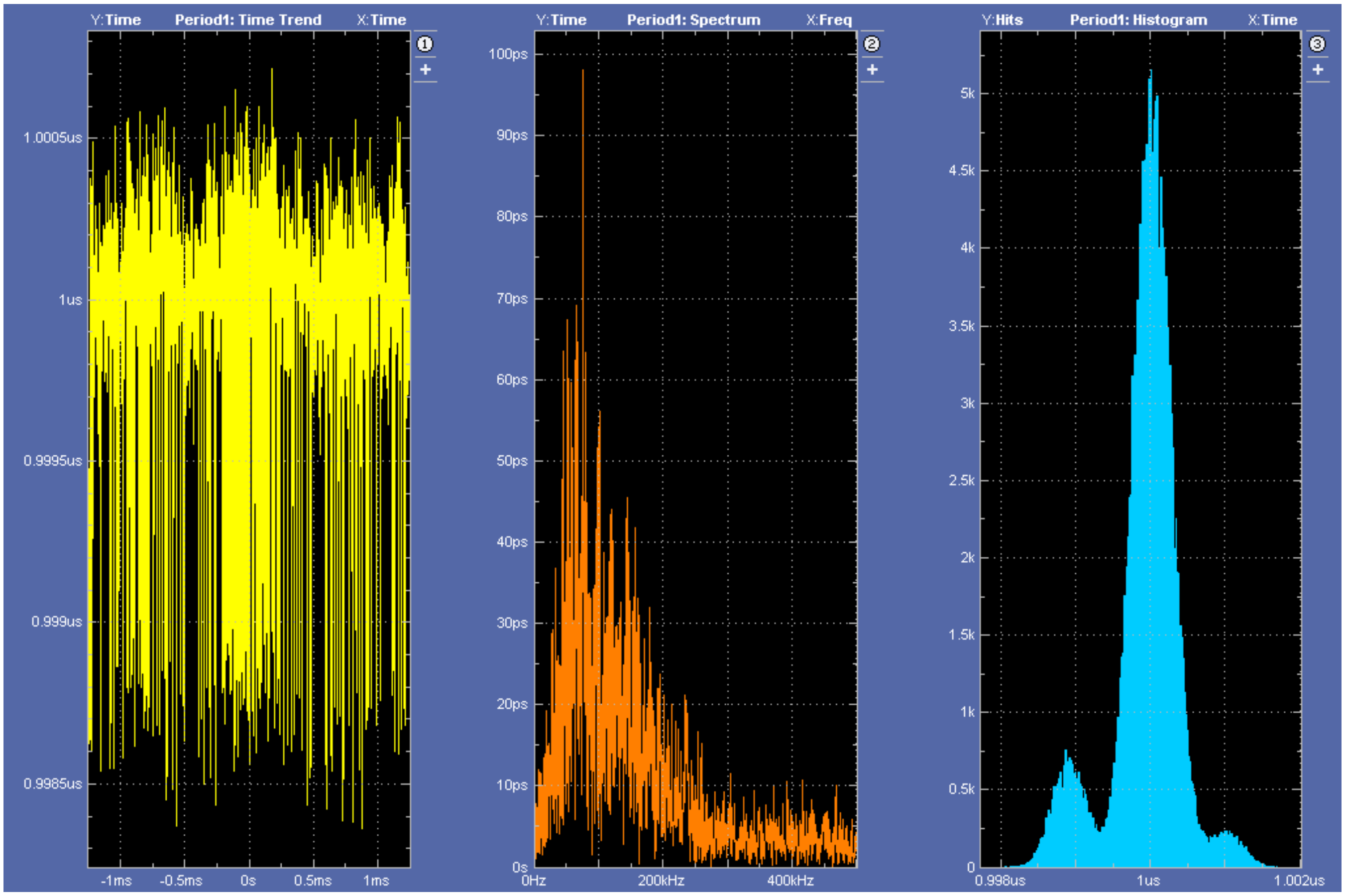
The test is performed with the integrated histogram test feature of the Tektronix MSO5104 Mixed Signal Oscilloscope.

The jitter measurement test is performed with the following parameters:

- PTO frequency: 1 MHz
- Compare value: 500 000 000
- Speed value: 50 000 000
- Population: 222 500

The histogram chart (right window) in Figure 15 shows that the maximum expected jitter of 50 ns at the maximum PTO frequency is not crossed. Most hits are at 1 µs and the deviation is never more than 50 ns.

Figure 15. Oscilloscope Image



8 Design Files

8.1 Schematics

To download the schematics, see the design files at TIDEP0027.

8.2 BOM

To download the bill of materials (BOM), see the design files at TIDEP0027.

8.3 PCB Layout

To download the layer plots, see the design files at TIDEP0027.

9 Software Files

To download the software files for the TIDEP0027 reference design, visit the TI website at http://www.ti.com/tool/tidep0027.

10 References

1. Texas Instruments, *AM3359 Industrial Communications Engine,*TMDSICE3359 EVM, (TMDSICE3359).
2. EE Times-India, *Generate stepper-motor speed profiles in real time*, David Austin, http://fab.cba.mit.edu/classes/MIT/961.09/projects/i0/Stepper_Motor_Speed_Profile.pdf.
3. GitHub, BeagleBone PRU Package on Github, https://github.com/beagleboard/am335x_pru_package.
4. Code Composer Studio Download page, http://processors.wiki.ti.com/index.php/Download_CCS.

11 Terminology

ASIC— Application Specific Integrated Circuit

CCS— Code Composer Studio

GPO— General Purpose Output

ICSS— Industrial Communication Subsystem

FPGA— Field Programmable Gate Array

PLC— Programmable Logic Controller

PRU— Programmable Real-time Unit

PTO— Pulse Train Output

12 About the Author

THOMAS MAUER is a System Applications Engineer in the Factory Automation and Control Team at Texas Instruments Freising, where he is responsible for developing reference design solutions for the industrial segment. Thomas brings to this role his extensive experience in industrial communications like Industrial Ethernet and fieldbuses and industrial applications. Thomas earned his Electrical Engineering degree (Dipl. Ing. (FH)) at the University of Applied Sciences in Wiesbaden, Germany.

GANESH MOHAN NELLIPARAMBIL is a Master Intern (Control and Embedded Instrumentation at ESIGELEC in Rouen, France and in Embedded Systems and Instrumentation at Manipal University in Manipal, India) in the Factory Automation and Control Team at Texas Instruments Freising, where he is responsible for developing reference design solutions for the industrial segment. Ganesh earned his Bachelor’s degree in Instrumentation and Control Engineering from Anna University in Chennai, India.

INGOLF FRANK is a Systems Engineer in the Texas Instruments Factory Automation and Control team, focusing on programmable logic controller I/O modules. Ingolf works across multiple product families and technologies to leverage the best solutions possible for system level application design. Ingolf earned his Electrical Engineering degree (Dipl. Ing. (FH)) in the field of information technology at the University of Applied Sciences Bielefeld, Germany in 1991.

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