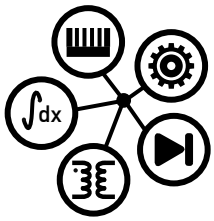

Embedded
Code Generation

DEMO MODEL

Simple SPI Model

A simple model to explore the SPI protocol with the TI C2000 TSP

Last updated in C2000 TSP 1.9.1

www.plexim.com

- Request a PLECS and PLECS Coder trial license
- Get the latest TI C2000 and RT Box Target Support Package
- Check the PLECS, RT Box and TI C2000 TSP documentation

1 Overview

This demo features a simple model using the Serial Peripheral Interface (SPI) blocks on Texas Instruments (TI) C2000 microcontrollers (MCUs) with the PLECS Coder and the TI C2000 Target Support Package.

The SPI is a high-speed synchronous serial input/output device that allows a serial bit stream of programmable length (1 to 16 bits) to be shifted into and out of the device at a configurable bit-transfer rate. The SPI is usually used for communications between the MCU controller and external peripherals, or another controller.

The model is split into six distinct subsystems called “28069”, “280049”, “28377S”, “28379D”, “28P550SJ9”, and “28P650DK9”. Each subsystem can be independently deployed to the corresponding TI C2000 LaunchPad hardware. The following sections provide a brief description of the model and instructions on how to simulate it.

2 Model

The top level schematic contains six separate subsystems, as shown in Fig. 1. The subsystem labeled “28069” is configured for the TI 28069 LaunchPad [2], the subsystem labeled “280049” is configured for the TI 280049C LaunchPad [1], the subsystem labeled “28377S” is configured for the TI 28377S LaunchPad [3], the subsystem labeled “28379D” is configured for the TI 28379D LaunchPad [4], the subsystem “28P550SJ9” is configured for the TI 28P550SJ9 LaunchPad [5], and lastly the subsystem labeled “28P650DK9” is configured for the TI 28P650DK9 LaunchPad [6].

Each subsystem is enabled for code generation, as indicated by the thick outer border of the subsystem blocks. This step is necessary to generate the model code for a subsystem via the PLECS Coder. This setting is configured by selecting the subsystem, opening **Edit + Subsystem + Execution settings...** menu, and then selecting the **Enable code generation** option.

The SPI is a controller-peripheral based interface with a single controller and one or more peripheral devices.

The interface consists of the following signals:

- **SPIPICO** Serial data output (controller out/peripheral in)
- **SPIPOCI** Serial data input (controller in/peripheral out)
- **SPICLK** Shift-clock, generated by the SPI Controller
- **/SPICS** Chip-select or peripheral-enable signal, also referred to as SPIPTE. The chip-select signal is an active-low signal that enables the POCI and PICO ports of the SPI Peripheral

The SPI Controller block provides a clock signal (SPICLK) which generates a configurable number of clock pulses during each simulation step. For both the peripheral and the controller, data is shifted out of the shift registers on one edge (rising or falling) of the SPICLK and latched into the shift register on the opposite clock edge. If the clock phase (CPHA) bit is configured to 1, data is transmitted and received a half-cycle before the SPICLK transition.

In this model, each of the subsystems includes a simple SPI model, illustrating signal exchange between one SPI Controller and one SPI Peripheral of the same MCU target, as shown in Fig. 2. To explore the signal exchange via SPI between two different MCU targets, choose any two desired targets. Then configure one of them as a SPI Controller and the other one as a SPI Peripheral.

Multiple SPI Peripherals can be supported by a single controller through chip-select (/CS) signals. For details on multiple SPI Peripherals, refer to the “Help” section of the SPI Controller block, or refer to the TI C2000 Target Support User Manual [7].

For both SPI Controller and SPI Peripheral blocks, the data to be transmitted is provided at the input **TX** and the data received is available at the output **RX**.

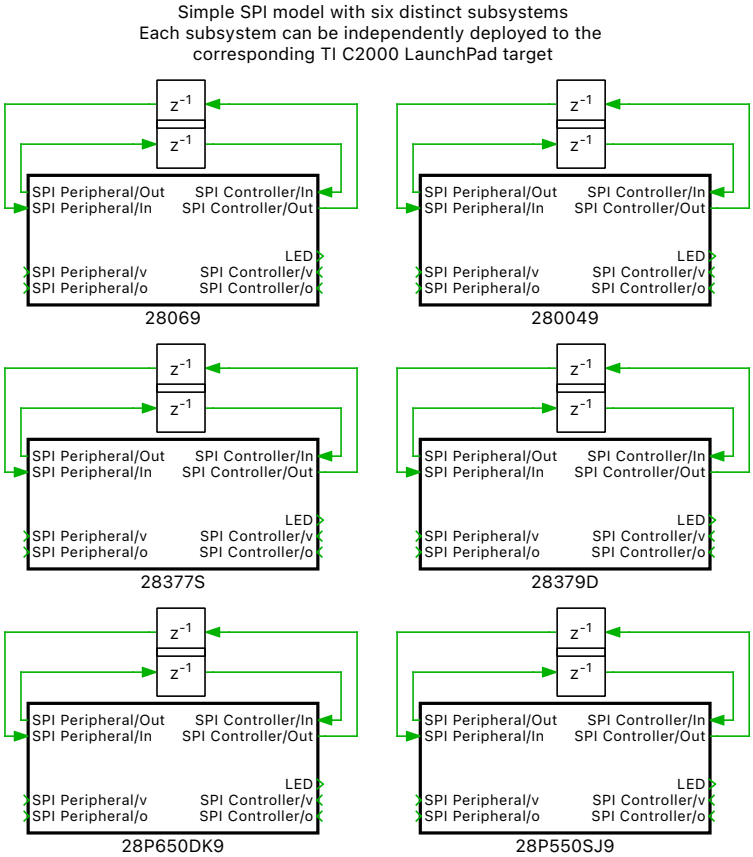

Figure 1: Top level schematic of the model with six subsystems

Considerations for SPI Controller

An output value of 1 at the **v** port of the SPI Controller indicates that valid data is sent to all the peripherals. If the SPI Controller does not have enough time to complete the transmission before the block is executed again, the output **o** turns 1 to indicate an overrun error. In the model, as shown in Fig. 2, a counter logic is included at the **v** and **o** ports. This doesn't have much meaning offline.

If an overrun error is being signaled at the **o** port of the SPI Controller, it is possible that the task with which the SPI Controller is associated executes too fast. In this case, either reduce the SPI Controller execution task rate or increase the SPI clock rate.

For example, if SPICLK is set as 180000 Hz, and is expected to transmit a packet of 4 words at 8 bits per word, then the time it would take to transmit one packet is

$$\frac{1}{180000} * 4 * 8 = 1.78 * 10^{-4} \text{ seconds}$$

In this case, the execution step size of the SPI Controller must be set to values greater than 0.178 milliseconds.

Considerations for SPI Peripheral

An output value of 1 at the **v** port indicates a valid data exchange with the SPI Controller. If the SPI **RX** port of the SPI Peripheral receives new data before the previous data has been read, the existing data will be overwritten and lost. If this occurs, the output **o** turns 1 to indicate an overrun error. In the model, as shown in Fig. 2, a counter logic is included at the **v** and **o** ports. This doesn't have much meaning offline.

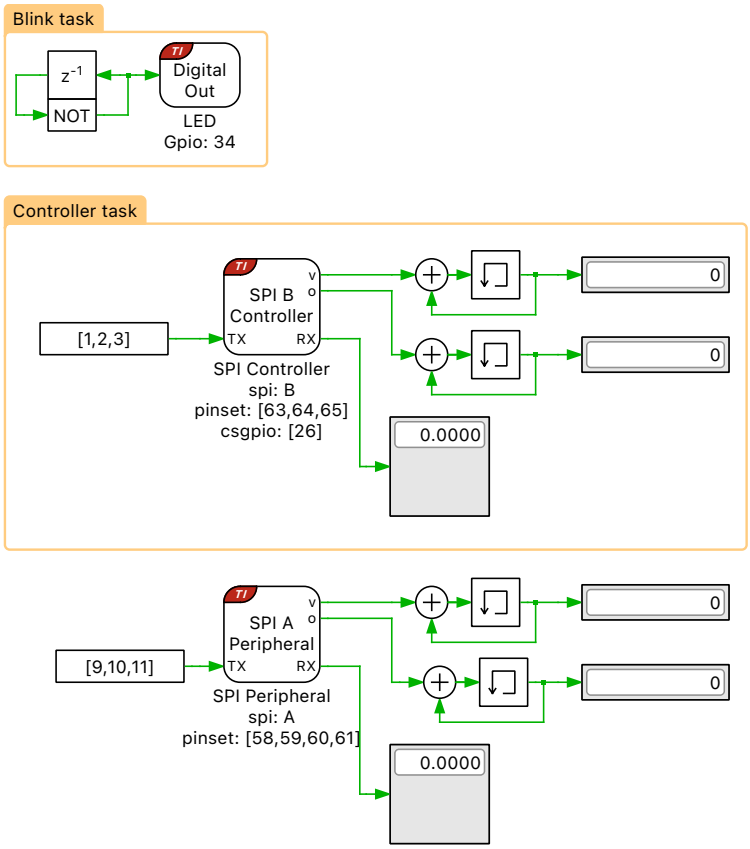

Figure 2: Schematic of the subsystem labeled 28379D

There are two considerations to note when overrun errors occur:

- The controller is not allowed to start transmitting before the peripheral is up and running. If the peripheral is booting up while the controller is transmitting, then it may receive an incomplete first message, from which it will not be able to recover.
- In order to avoid overruns, the SPI Peripheral block must be executed faster than the rate at which the SPI Controller is sending data.

Additional considerations

- Always make sure to provide a common power supply to all the MCUs communicating via SPI.
- From the RT Box LaunchPad Interface User Manual [8], it can be observed that a few SPI signals are connected to digital signals, which would cause interference in SPI communication. Therefore to use SPI, the launchpad board must be disconnected or isolated from the RT Box LaunchPad Interface board.
- Do not use the RT Box to measure the “CLK” signal. This will result in overrun errors.

Multi-tasking code

The generated code runs at a base sample time on the MCU or the **Discretization step size**. In this model, the discretization step size of each of the subsystems is set to 100 μs. In order to avoid overruns, the SPI Peripheral block must be executed faster than the rate at which the SPI Controller is sending data. Therefore, multi-tasking code is used to execute the SPI Controller block at 200 μs.

Multi-tasking code generation is configured in the **Scheduling** tab of the **Coder + Coder options...** dialog. By changing the **Tasking mode** to `multi-tasking` and the **Task configuration** to `specify`, the

sample time for each task can be configured. The base sample time is always equal to the **Discretization step size**. The **Sample time** setting for lower priority tasks must be an integer multiple of the base sample time. Up to 15 slower lower priority tasks that execute at different rates can be specified, preserving processor time for the fastest, highest priority task in the application. For further information, refer to the "Code Generation" section in the PLECS User Manual [9].

3 Simulation

Each subsystem can be directly converted into target specific code for the corresponding TI LaunchPad hardware.

Note Before proceeding, ensure the DIP switch position and jumper configuration on the LaunchPad device are correctly configured. Guidance for each LaunchPad device is provided in the “Tips for Programming C2000 Development Kits” section of the TI C2000 Target Support User Manual [7].

Connect the hardware

Next, connect the pin numbers listed below using jumper wires for the desired MCU. To explore the signal exchange via SPI between two different MCU targets, choose any two desired targets. Then configure one of them as a SPI Controller and the other one as a SPI Peripheral.

28069	Controller	Peripheral
PICO	J6-55 (GPIO 24)	J2-15 (GPIO 16)
POCI	J6-54 (GPIO 25)	J2-14 (GPIO 17)
CLK	J5-47 (GPIO 14)	J1-7 (GPIO 18)
CS	J6-53 (GPIO 52)	J2-19 (GPIO 19)

280049	Controller	Peripheral
PICO	J6-55 (GPIO 24)	J2-15 (GPIO 16)
POCI	J6-54 (GPIO 31)	J2-14 (GPIO 17)
CLK	J5-47 (GPIO 22)	J1-7 (GPIO 56)
CS	J6-59 (GPIO 27)	J2-19 (GPIO 57)

28377S	Controller	Peripheral
PICO	J6-55 (GPIO 63)	J2-15 (GPIO 58)
POCI	J6-54 (GPIO 64)	J2-14 (GPIO 59)
CLK	J5-47 (GPIO 65)	J1-7 (GPIO 60)
CS	J6-53 (GPIO 99)	J1-8 (GPIO 61)

28379D	Controller	Peripheral
PICO	J6-55 (GPIO 63)	J2-15 (GPIO 58)
POCI	J6-54 (GPIO 64)	J2-14 (GPIO 59)
CLK	J5-47 (GPIO 65)	J1-7 (GPIO 60)
CS	J6-53 (GPIO 26)	J2-19 (GPIO 61)

28P650DK9	Controller	Peripheral
PICO	J6-55 (GPIO 91)	J2-15 (GPIO 16)
POCI	J6-54 (GPIO 92)	J2-14 (GPIO 17)
CLK	J8-74 (GPIO 32)	J1-7 (GPIO 18)
CS	J6-59 (GPIO 94)	J2-19 (GPIO 57)

28P550SJ9	Controller	Peripheral
PICO	J1-8 (GPIO 24)	J2-15 (GPIO 8)
POCI	J6-58 (GPIO 25)	J4-36 (GPIO 10)
CLK	J6-52 (GPIO 26)	J1-7 (GPIO 9)
CS	J6-59 (GPIO 27)	J4-35 (GPIO 11)

Flash the MCU

Follow the instructions below to upload the subsystems to a TI MCU.

- Connect the desired MCU to the host computer through a USB cable.
- From the **System** tab of the **Coder + Coder options...** window, select the MCU of interest.
- Next, from the **Target** tab, select the appropriate target from the dropdown menu. Then under the **General** sub-tab, select the desired **Build type**.
- Then, to `Build and program` the MCU directly from PLECS, choose either `Run from Flash` or `Run from RAM` as the **Build configuration**, then select `LaunchPad` as the **Board** type, and click **Build**.

Note If programmed correctly, the LED on the LaunchPad board should blink.

For advanced users who are familiar with Code Composer Studio (CCS), there is an option to `Generate code into CCS project`. Included with the TI C2000 Target Support package is a folder titled `projects`. Within the folder there are ZIP archives containing pre-built CCS projects for each MCU. Import the zip archive folder that corresponds to the desired target into CCS. You will notice a new project created in your CCS workspace. Enter the location of the `${workspace_loc}/dev_28xx/cg/` folder from the CCS project into the **CCS project directory** field and click **Build**. Then, proceed to build and debug the project as a normal CCS project. Refer to "Quick Start" section of the TI C2000 Target Support User Manual [7] for detailed step-by-step instructions.

External Mode

Once the generated code is running on the C2000 target, the user can enter the External Mode to update displays in the PLECS application with real-time values and change certain simulation parame-

ters. The steps below outline how to connect to the target device, with additional debugging details provided in the "Start the External Mode" section of the user manual [7].

- First, from the **System** menu on the left hand side of the **Coder + Coder options...** window, select the desired MCU.
- Then, from the **External Mode** tab, select the **Target device** by clicking icon next to the **Target device** field.
- Next, click **Connect** and then **Activate autotriggering** to observe the results in the subsystem display.

The exchanged SPI signals on a 28379D target are shown in Fig. 2. To learn about the **v** and **o** ports, and tips to handle overrun errors, refer to Section 2.

Parameter Inlining

To configure parameters as tunable, open the **Coder + Coder options...** menu and navigate to the **Parameter Inlining** tab. When a component from the schematic is dragged and dropped into the **Exceptions** list, tunable parameters associated with that component will be tunable during runtime. Note this behavior depends on the **Default behavior** setting, as the **Exceptions** list specifies components which have opposite behavior of the default setting.

In this case, the "TX" inputs to the SPI block can be adjusted on the fly when the model executes on the embedded target device, when connected to External Mode. Changes in the parameters will be reflected in the Scope traces and Display once they take effect.

4 Conclusion

This model explores the SPI protocol with the TI C2000 TSP using a simple model.

References

[1] TI C2000 Piccolo MCU F280049C LaunchPad Development Kit
URL: http://www.ti.com/tool/LAUNCHXL-F280049C.

[2] TI C2000 Piccolo MCU F28069M LaunchPad Development Kit,
URL: http://www.ti.com/tool/LAUNCHXL-F28069M.

[3] TI C2000 Delfino MCUs F28377S LaunchPad Development Kit
URL: https://www.ti.com/lit/pdf/sprui25

[4] TI C2000 Delfino MCUs F28379D LaunchPad Development Kit,
URL: http://www.ti.com/tool/LAUNCHXL-F28379D.

[5] TI C2000 F28P550SJ9 LaunchPad development kit,
URL: https://www.ti.com/tool/LAUNCHXL-F28P55X.

[6] TI C2000 F28P650DK9 LaunchPad development kit,
URL: https://www.ti.com/tool/LAUNCHXL-F28P65X.

[7] PLECS TI C2000 Target Support User Manual,
URL: https://www.plexim.com/sites/default/files/c2000manual.pdf.

[8] RT Box LaunchPad Interface User Manual,
URL: https://plexim.com/sites/default/files/launchpadinterfacemanual.pdf.

[9] PLECS User Manual,
URL: https://www.plexim.com/sites/default/files/plecsmanual.pdf.

Revision History:

C2000 TSP 1.3.1	First release
C2000 TSP 1.4.5	Updated the web links
C2000 TSP 1.5.2	Updated SPI mode settings
C2000 TSP 1.9.1	Updated for F28P55x chip support and new block names

How to Contact Plexim:

☎	+41 44 533 51 00	Phone
	+41 44 533 51 01	Fax
✉	Plexim GmbH Technoparkstrasse 1 8005 Zurich Switzerland	Mail
@	`info@plexim.com`	Email
	`http://www.plexim.com`	Web

Embedded Code Generation Demo Model