Texas Instruments C2000 MCU in automotive and industrial application

Classes no. 1, 2 – 20-22, 27-29 March 2023

Preparing the project

Install *Embedded Coder Support Package for TI C2000 Processors*, indicate the type of device to be programmed in the model settings.

To do this, select the *Modeling tab* and from the *Setup tab* select *Model Settings*. Then select *Hardware Implementation* and as *Hardware Board* we assign *TI Delfino F28379D LaunchPad*. *The Device Vendor* and *Device Type* fields should autocomplete (*Texas Instruments* and *C2000* respectively).

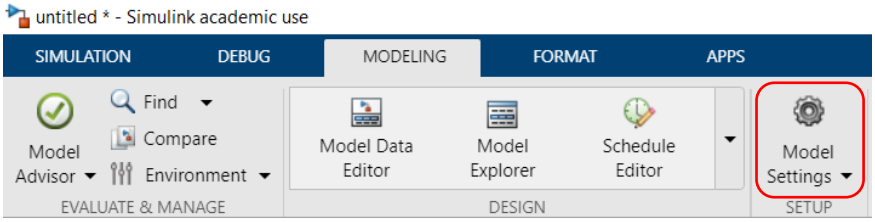


Fig. 1. Project preparation - step 1

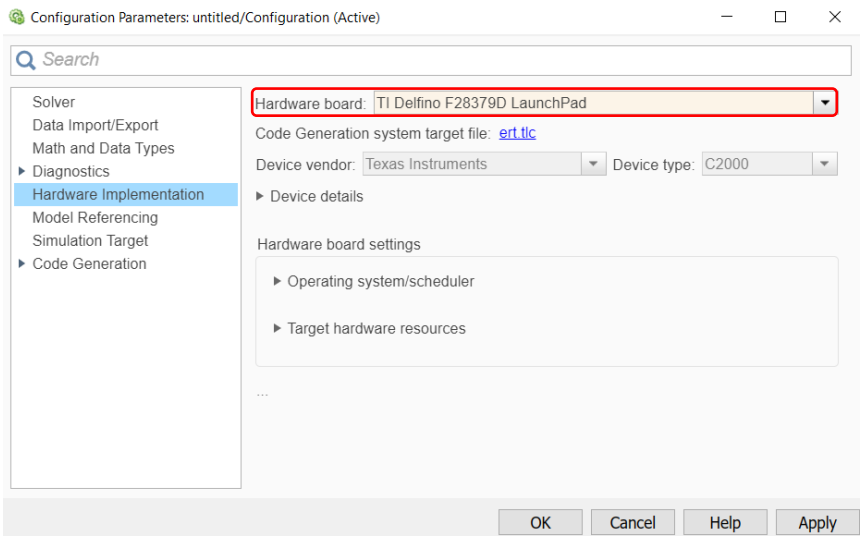


Fig. 2. Project preparation - step 2

There are wide possibilities of changing the model settings (it is recommended to familiarize yourself with the *Code Generation options*), however, the default settings will be used during the classes.

In order to upload the program to the microcontroller, you can use the *Monitor & Tune option (Hardware/Run on hardware* tab), which allows you to simultaneously observe signals in the Simulink space. If the goal is only to upload the program, you can also use the *Build, Deploy & Start option (Hardware/Deploy* tab).

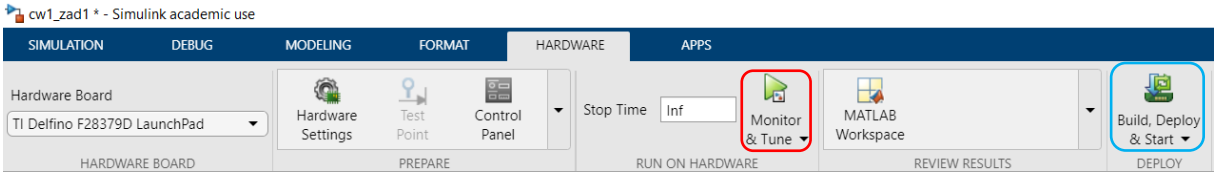


Fig. 3. Uploading the program to the microprocessor memory

Basic used blocks

■ Blocks related to the C2000 microcontroller are in the library:
Embedded Coder Support Package for Texas Instrument C2000 Processors/F2837xD

■ If you want to write part of the program in C or MATLAB programming language, you can use blocks *C Function* or *MATLAB Function* (check out these blocks in Simulink).

■ Basic mathematical blocks may be useful during classes:
- arithmetic operations: *Add* , *Subtract* , *Product* , *Divide* , *Gain* ,
- the relationship between numerical values - *Relational Operator* (greater than, equal to, etc.),
- logical operators - *Logical Operator* (AND, OR, NOT etc.),
- rounding - *Rounding Function* ,
- constant value - *Constant* ,
- rectangular pulse generator - *Discrete Pulse Generator* .

■ To make the simulation scheme clearer, you can use *Goto* and *From* blocks, which act as labels (scheme points marked with the same label are connected to each other).

■ *Convert* block can be used to ensure compatibility between data formats used in calculations.

■ The block that allows limiting the value of a given variable to a specific range is the *Saturation* block.

■ To replace the *if()* function, you can use, for example, a *Switch block* .

■ To use the value of a given variable from the previous calculation loop, you can use the *Unit Delay block* .

Digital Clock can be used to measure time .

■ Using the *Monitor & Tune option (Hardware/Run on hardware* tab) it is possible to observe signals in Simulink space. For this purpose, the *Scope block can be used to plot the time waveforms* .

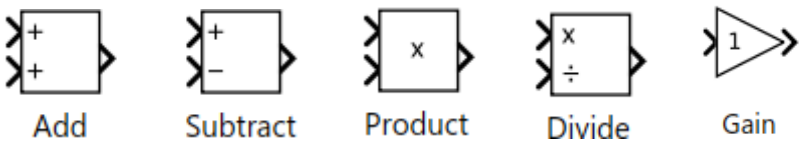


Fig. 4. Blocks of arithmetic operations

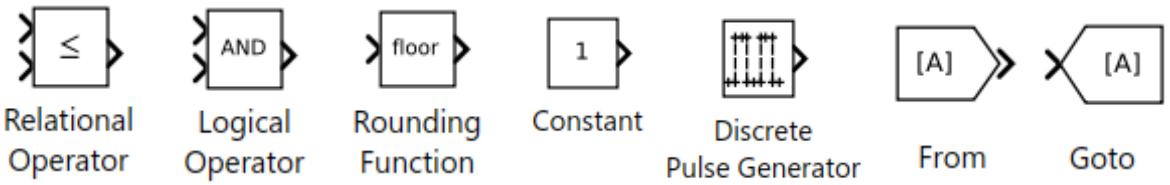


Fig. 5. Other math blocks and label blocks

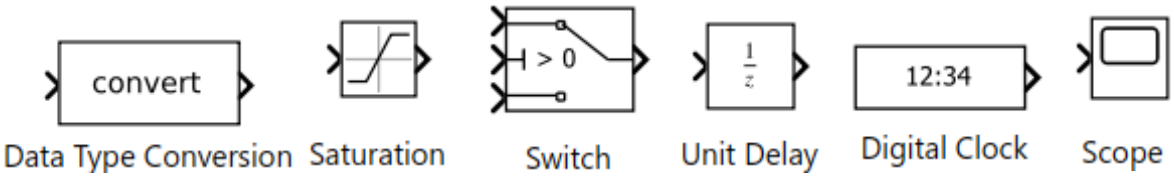


Fig. 6. Other useful blocks during classes

Triggered subsystems

After selecting a part of the layout with the left mouse button, an ellipsis symbol appears at the bottom right corner of the selected area. When you hover your mouse over this symbol, you can see e.g. options to create subsystems such as *Enabled Subsystem* and *Triggered Subsystem* .

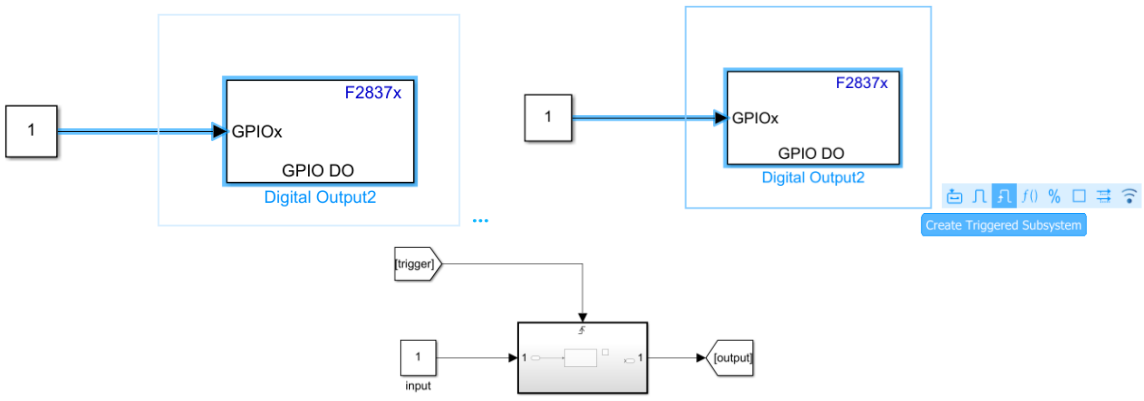


Fig. 7. Creation of an edge-triggered subsystem

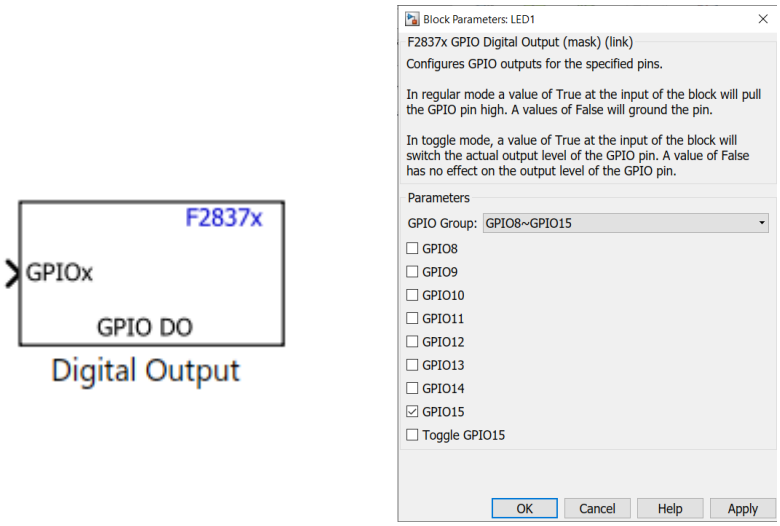
Enabled Subsystem – the program inside the subsystem is executed only when the trigger signal has a certain state (default is high state).

Triggered Subsystem - a program within a subsystem is executed once on a specified edge (rising, falling, or both).

GPIOs

The following value is given to the input of the *Digital Output block (GPIO DO) :*
- 0 (false), in order to obtain a low state of this output,
- 1 (true), in order to obtain a high state.

The GPIO pin number is selected in the block dialog.

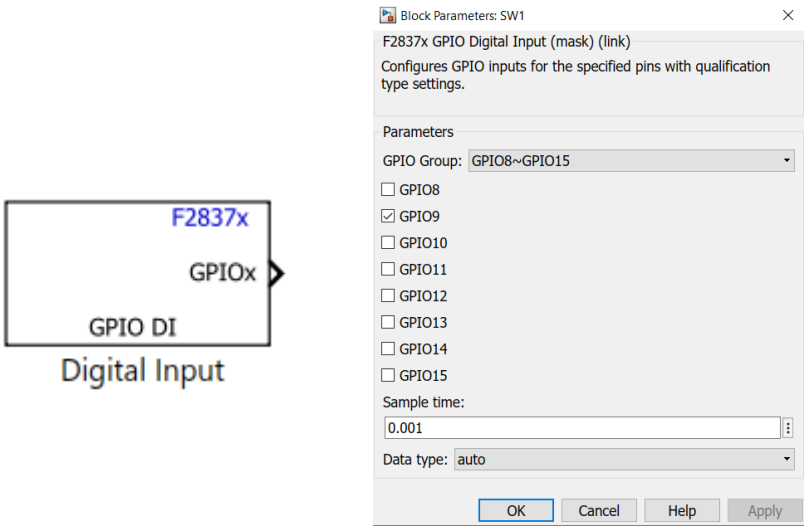


Digital Output block and its dialog box

Toggle option , which works by changing the output state to the opposite one when the value 1 is given to the input of the block.

At the output of the *Digital Input (GPIO DI) block* , the following value is obtained:
- 0 (false), if low state was read,
- 1 (true) if high state was read.

Sample time and *Data type* are selected in the block dialog .



Digital Input block and its dialog box

The sampling times of connecting blocks must be the same.

Diodes

Light emitting diodes - LEDs (DLp1, DLp2, DLp3 and DLp4) are connected to GPIO15, GPIO14, GPIO11 and GPIO10 respectively. The LEDs are lit for high state.

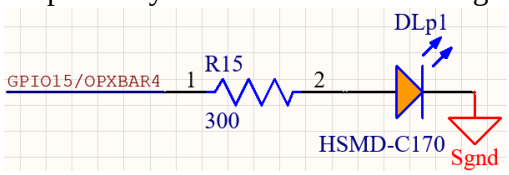


Fig. 10. Example of diode power supply

The cathode of the diode is connected to the system ground (*Sgnd*) and the anode to the appropriate GPIO pin. A current-limiting resistance is connected in series with the diode.

Buttons

The buttons (SW1 and SW2) are connected to GPIO9 and GPIO8, and the high state occurs when the button is not pressed.

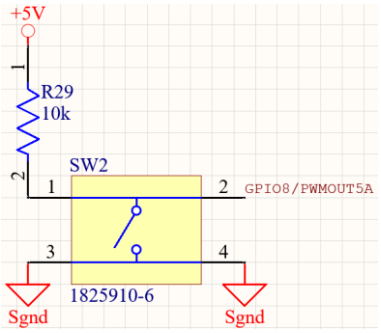


Fig. 11. Example of button power supply

Pins 1 and 2 of the button are connected to the +5V potential through a resistor of a sufficiently large value. In turn, pins 3 and 4 are shorted to the ground of the system. When the button is pressed, the pins are short-circuited, i.e. the potential of pins 1 and 2 becomes equal to the ground potential.

Task 1

Please create a program that provides:

a) the DLp1 diode is on (regardless of the state of the buttons),
b) turning off the DLp2 diode (regardless of the state of the buttons),
c) the DLp3 LED is on when the SW1 button is pressed (and the LED is off when the button is not pressed),
d) turning off the DLp4 diode when the SW2 button is pressed (and lighting this diode when the button is not pressed),

Task 2

Please create a program that provides:

a) the DLp1 diode is on when both buttons are pressed (and the diode is off when at least one of them is not pressed),
b) the DLp2 LED is on when at least one button is pressed (and the LED is off when none of them is pressed),
c) changing the state of the DLp3 diode every 1 second,
d) change of the state of the DLp4 diode when the SW1 button is pressed (pressing the button causes the change of state, which is maintained until the next button is pressed).

Pulser

The pulser has three outputs: on two of them (GPIO130 and GPIO131) rectangular signals occur when the knob is turned (shifted relative to each other, which allows determining the direction of rotation), and on the third output (GPIO66) there is a high state when the pulser knob is pressed.

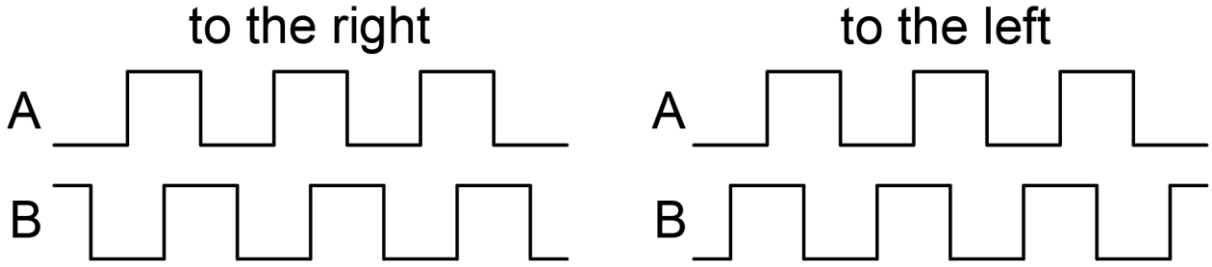


Fig. 12. Example waveforms of pulse generator signals for both directions of rotation

Task 3

Please create a program that provides:

a) increasing the value of the variable as a result of turning the pulser knob to the right and decreasing the value of this variable as a result of turning it to the left,
b) limiting the value of the variable to the accepted range,
c) changing the adjustment precision when the knob is pressed (e.g. by default, the value is incremented and decremented by 5, and when the knob is pressed by 1).