



**MPLAB[®] Device Blocks
for Simulink[®]
Quick Start Guide**

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, dsPIC, FlashFlex, KEELQ, KEELQ logo, MPLAB, PIC, PICmicro, PICSTART, PIC³² logo, rfPIC, SST, SST Logo, SuperFlash and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MTP, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.


Analog-for-the-Digital Age, Application Maestro, BodyCom, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Omniscent Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICtail, REAL ICE, rLAB, Select Mode, SQI, Serial Quad I/O, Total Endurance, TSHARC, UniWinDriver, WiperLock, ZENA and Z-Scale are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

GestIC and ULPP are registered trademarks of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2013, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

ISBN: 978-1-62077-777-0

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
= ISO/TS 16949 =

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Table of Contents

Chapter 1. Introduction

1.1	MATLAB	5
1.2	Simulink	7
1.3	MPLAB Device Blocks for Simulink	7

Chapter 2. Quick Start

2.1	Set Up a Simulink Model for Code Generation	9
2.2	Sample Model Overview	10
2.3	Generate Code and Program a Device	13
2.4	Microcontroller–MATLAB Communications	16

Chapter 3. Library Content

3.1	Library Structure	26
3.2	Supported Devices	27
3.3	System Configuration Group	29
3.4	Digital IO group	33
3.5	Analog IO group	35
3.6	PWM IO group	39
3.7	QEI group	42
3.8	Pulse Input/Output group	43
3.9	BUS UART group	44
3.10	Bus SPI Group	46
3.11	Bus I ² C™ Group	50
3.12	User Functions group	54

Index	57
-------------	----

Worldwide Sales and Service	58
-----------------------------------	----

NOTES:

Chapter 1. Introduction

Microchip Technology's MPLAB[®] Device Blocks for Simulink[®] is a blockset that integrates within the MATLAB[®]/Simulink environment.

- MATLAB
- Simulink
- MPLAB Device Blocks for Simulink

1.1 MATLAB

MATLAB refers to both an environment and a programming language. The MATLAB programming language may also be called “m” due to the .m filename extension.

The MATLAB environment is mainly composed of a Command Window (Figure 1-1) and a script editor. MATLAB instructions can be typed directly into the Command Window to be evaluated within the Workspace. The Command Window also allows input of any MATLAB built-in function or calls a user script. Graphic User Interfaces (GUIs) for various tasks or analysis are available.

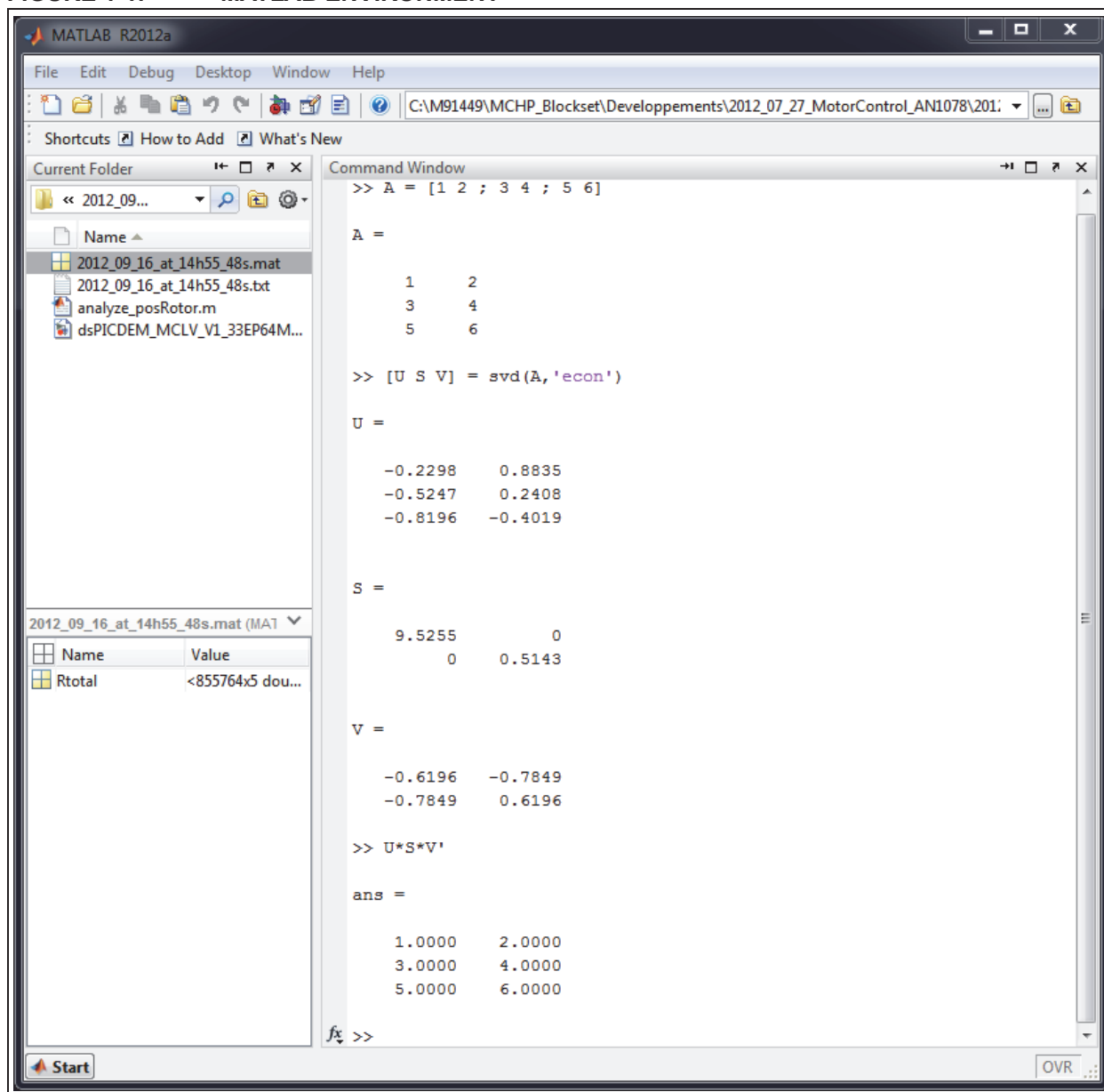
Figure 1-1 shows an example in the Command Window that sets a new 3 * 2 matrix in variable A, computes a Singular Value Decomposition (function SVD) and checks that the resulting $U * S * V'$ matrix product is equal to the original matrix A.

The MATLAB language allows the manipulation of matrices and vectors. High-level Math functions for scalar or matrix operations are provided. An example of singular decomposition is shown in Figure 1-1.

In addition to linear algebraic functions, MATLAB also provides a bundle of other functions, e.g., to plot data, read files, etc. All these capabilities simplify the development of algorithms as well as data analysis.

Toolboxes can be added to MATLAB to extend standard MATLAB functions. For further information, please go to the [MathWorks[®] web site](http://www.mathworks.com).

FIGURE 1-1: MATLAB ENVIRONMENT



1.2 SIMULINK

Simulink is a graphical tool working on top of MATLAB for the simulation of dynamic systems. Dynamic systems can be described through a continuous time domain description (using variables s or p representing the Laplace transform) or in the discrete time domain (using variable z). Using a mixture of both continuous time and discrete time domains is also possible. Simulink has a collection of differential equation solvers that can deal with most continuous time systems. A collection of built-in blocks allows the simulation of complex systems that might include non-linearity and delays.

In addition to the standard Simulink blocks, blocksets can extend Simulink functions. The MPLAB Device Blocks for Simulink is one of these additional blocksets.

1.3 MPLAB DEVICE BLOCKS FOR SIMULINK

MATLAB and Simulink are both capable of generating C code, but this discussion exclusively focuses on Simulink which is adapted for the design of embedded systems.

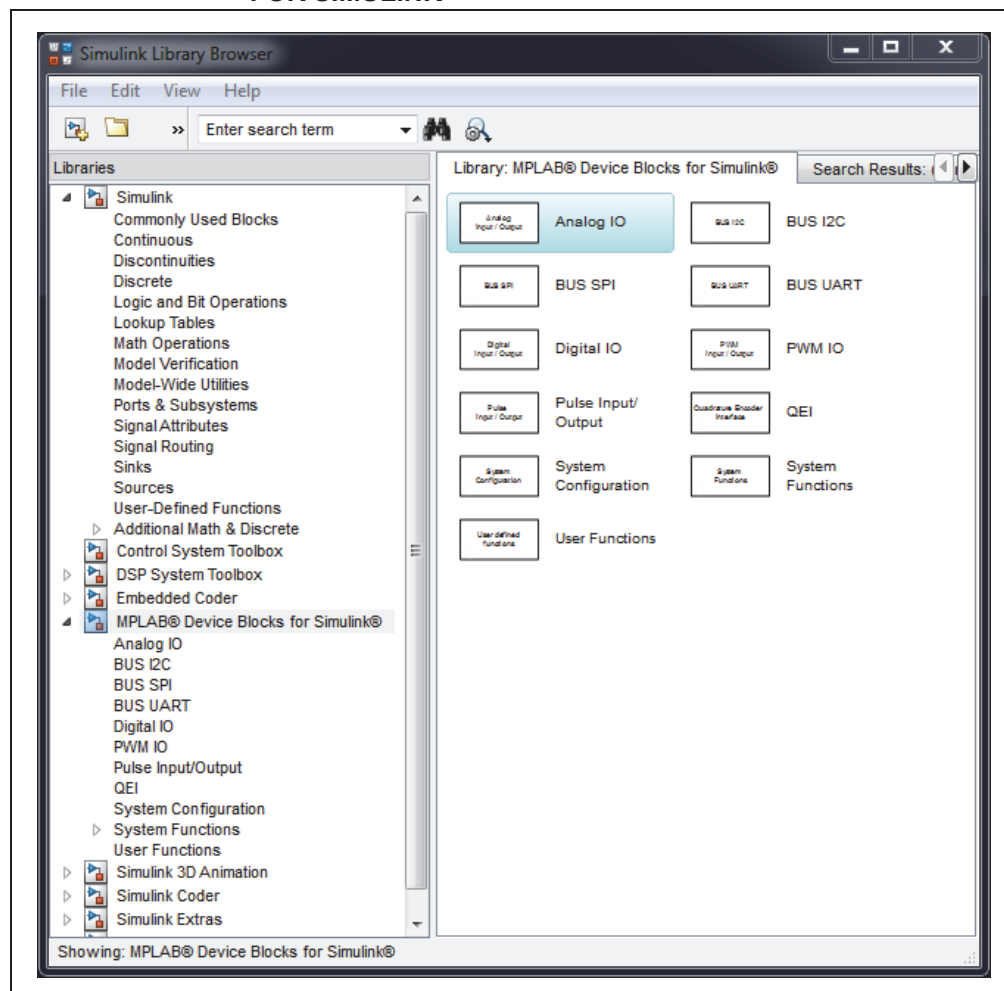
The MPLAB Device Blocks for Simulink adds blocks to the Simulink library and a few scripts to MATLAB. The added blocks allow easy configuring of a MCU (MCU) and its peripherals. The scripts added to MATLAB allow programming of the MCU and connecting to the MCU through the UART to analyze, log, or plot data in pseudo-real-time.

The MPLAB Device Blocks for Simulink will perform the following tasks during the code generation process:

- add code to set and configure the targeted MCU
- add code to configure the peripherals used
- add code to handle the scheduler
- compile the overall generated C code to get a ready-to-flash binary file (`.cof`, `.hex`, or `.elf`),
- flash the binary file on the MCU (might be activated or not)
- additionally, create an MPLAB IDE v8 and an MPLAB X IDE project

Figure 1-2 shows the Simulink library with the included MPLAB Device Blocks for Simulink. The left pane shows the list of Simulink libraries installed. These libraries contain blocks that can be dragged and dropped into a Simulink model.

FIGURE 1-2: SIMULINK LIBRARY WITH THE MPLAB DEVICE BLOCKS FOR SIMULINK



In one button push, the tool translates a discrete time model into its binary equivalent, ready to flash on a microcontroller (MCU). The simulation-to-test iterations are considerably shortened, allowing more time to concentrate on algorithmic problems and less on programming and MCU configuration techniques. The tool gives to the Simulink skilled engineer the ability to tests algorithms on a real hardware setup without requiring extensive MCU knowledge. The generated code for the MCU is expected to behave exactly as the simulation model.

Limitations: The tool is not intended to simulate the MCU's peripheral behavior, like ADC input or PWM output, but only to “interface” a discrete time model with input/output peripherals available on a MCU so as to generate code for this model and run it on the hardware.

Care must be taken of real-time constraints; while simulation does not have real-time constraints, the embedded MCU is expected to calculate its outputs within the given “Time Step” defined by the model. An output pin could be used to monitor a busy flag that reflects the CPU load, i.e., set high during computation time, set low during idle time, and the duty cycle read through a scope is the CPU load.

Chapter 2. Quick Start

Quickly setup and use MPLAB® Device Blocks for Simulink®:

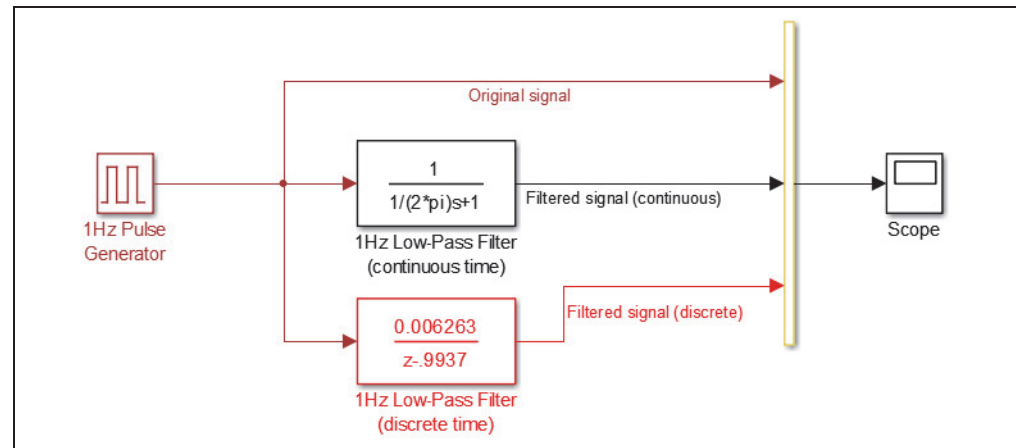
- Set Up a Simulink Model for Code Generation
- Sample Model Overview
- Generate Code and Program a Device
- Microcontroller–MATLAB Communications

2.1 SET UP A SIMULINK MODEL FOR CODE GENERATION

By default, Simulink is set with a continuous time solver. It allows simulating either a continuous time model, a discrete time model or a mixture of both continuous and discrete time model.

Figure 2-1 is a diagram showing the simulation of a continuous time (black color) and a discrete time (red color) first order 1 Hz low-pass filter. A square signal with a 1 Hz frequency is low-pass filtered. The continuous time filter (with transfer function shown in Equation 2-1) is compared against the “equivalent” discrete time filter. The z domain filter coefficient is calculated using the c2d MATLAB function from the Control System Toolbox with a targeted sampling time of 1 ms and the “zoh” transform method.

FIGURE 2-1: SIMULATION OF 1ST ORDER 1 HZ LOW PASS FILTER



EQUATION 2-1: CONTINUOUS FILTER TRANSFORM

$$\frac{1}{1 + \frac{1}{2\pi}s}$$

where s is the Laplace transform variable

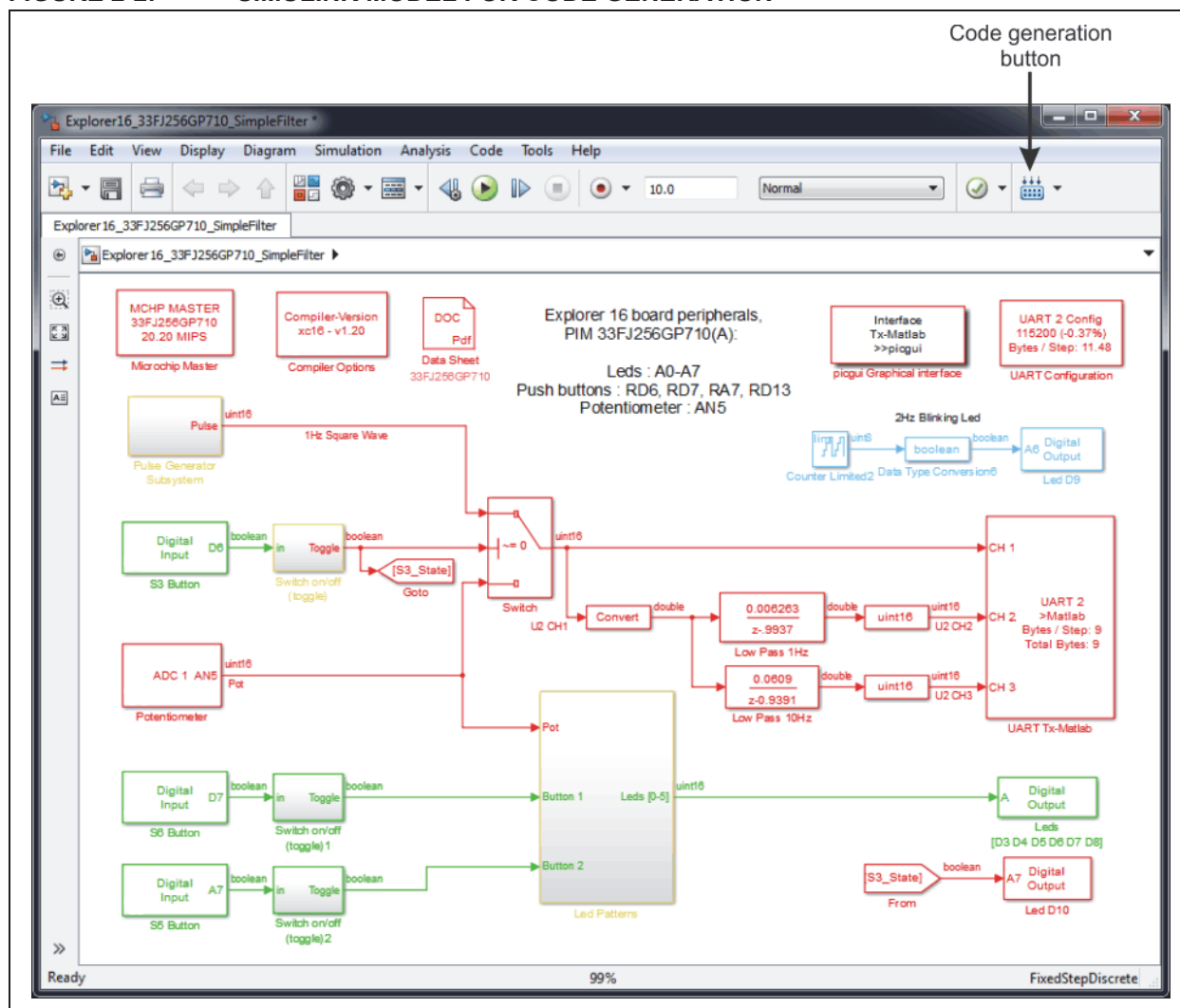
Code generation to target a PIC MCU requires setting a discrete time solver. Implementation of a differential equation solver within the embedded system is not desirable as it requires defining data length for each data type on the targeted MCU.

All these configurations are set automatically when the Microchip Master block is dropped on a model. If the model is not already set with a discrete time solver, the solver parameters are updated to a discrete solver with a time step set to 1 ms, i.e., a sampling rate of 1 kHz. This time-step time should be updated with the period (i.e., rate) of the designed algorithm.

2.2 SAMPLE MODEL OVERVIEW

Figure 2.2 shows a sample model configured for code generation that can be compiled and then flashed on the target. The model uses the default Simulink library blocks and the blocks from the MPLAB Device Blocks for Simulink, which can be recognized by the Microchip logo. Code is generated and compiled by pressing the blue button with three down arrows in the upper right menu bar.

FIGURE 2-2: SIMULINK MODEL FOR CODE GENERATION



2.2.1 Model Overview

This sample model is for an [Explorer 16 development board](#) (DM240001) with a [ds33FJ256GP710\(A\) PIM](#) (MA330011). Input to the model low-pass filter can be switched (by pressing the S3 Button) between the ADC input (Explorer 16 potentiometer value) and an internally generated wave pulse. The low-pass filter is the same discrete time filter shown in Figure 2-1. Output results are sent through the UART and can then be captured and plotted (see **Section 2.4.2 “Log and Plot Data with MATLAB: The picgui Interface”**). The model also makes the LEDs D3 through D8 blink with a pattern which depends on the potentiometer and the push buttons S5 and S6.

The sample model file can be found in the MPLAB Device Blocks for Simulink directory, `example` subdirectory, as:

`Explorer16_33FJ256GP710_SimpleFilter.mdl`.

2.2.2 Model Details

The left side of the model shows peripheral input blocks. The outputs of these blocks feed the model. The model uses two different type of input peripherals:

- The Digital Input blocks that sample the state of the push buttons S3, S5, and S6.
- The ADC block that converts analog voltage from the potentiometer that is connected to the pin AN5.

The right side of the model shows peripheral output blocks. The inputs of these blocks take results from the model. The model uses two different types of output peripherals:

- The Digital Output blocks that switch LEDs D3, D4, D5, D6, D7, D8, D9, and D10.
- The “UART Tx-MATLAB” block that sends values out through the UART using a custom protocol that allows retrieval of data within MATLAB for analysis and plotting in pseudo real-time.

The subsystem Switch (on/off toggle) contains a simple logic that toggles the subsystem output when a button is pressed.

The LED Patterns subsystem contains a logic that creates different blinking LED patterns that depend on the position of the potentiometer and the state of buttons S5 and S6.

The two Low Pass filters consist of, respectively, a 1 Hz and a 10 Hz low pass filter. These two filters are fed with either an internally-generated 1 Hz pulse or with the converted analog value from the potentiometer position. Button S3 allows switching between the internally-generated pulse and the potentiometer value. The filter outputs enter into the block “UART Tx-MATLAB”, which sends its input values through the UART using a custom protocol. It is possible to log the values and to plot them in pseudo real-time using our custom made “picgui” interface.

The sample model has blocks that execute at different rates; thus it is a “multi-rate” model. The various colors in Figure 2-2 represent the different rates, where the legend is shown in Figure 2-3, though not all colors in the legend are used in the model. The model does not contain continuous time.

FIGURE 2-3: SIMULINK MODEL REPORTED SAMPLING TIME LEGEND

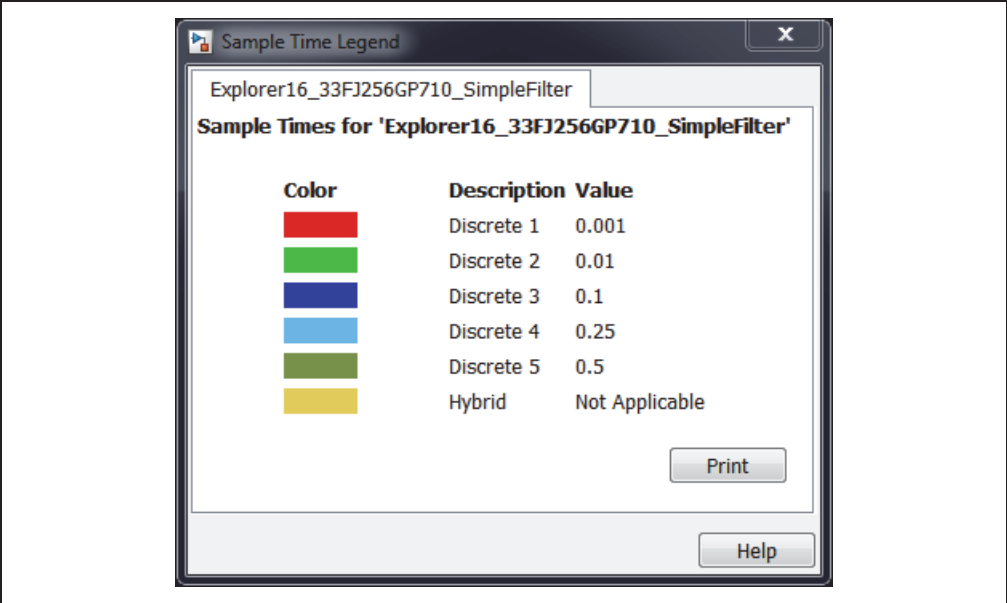


TABLE 2-1: REPORTED SAMPLING TIME LEGEND COLORS

Color	Definition
Red	Represents the faster rate, which usually corresponds to the period set within the Simulink solver configuration tab. Here the solver is set with a time step of 1 ms, which corresponds to a rate of 1 kHz.
Green	Represents blocks executed at a sub-rate of 100 Hz (i.e., 10 ms). This lower rate is used to sample signals from the push button to protect from detecting bounce.
Dark Blue	Represents the lower rate of 4 Hz (i.e. 250 ms), used to make the LED D9 blink at 2 Hz.
Light Blue	Represents the lowest rate of 2 Hz, used to create the 1 Hz pulse signal within the Pulse Generator Subsystem.

2.3 GENERATE CODE AND PROGRAM A DEVICE

Once the Simulink model is set up, code generation can begin. After this, the code generation output may be flashed/programmed on a MCU.

2.3.1 Perform Code Generation

A model that contains a Microchip Master block is automatically configured for code generation. The code generation process starts by clicking once on the top right button with the three downward arrows, or by using the shortcut <CTRL+B>.

The following tasks start sequentially:

- C code is generated from the Simulink model.
- C code is compiled into a ready-to-flash binary file (in either the .elf, .cof, or .hex format).
- Links are provided in the workspace to directly flash the compiled binary file using a compatible Microchip programmer (e.g., PICkit 3).

Code generation outputs messages when compiling the sample model as shown below. For documentation purposes, a few lines have been removed and long lines are truncated, symbolized by the line ending with "...".

```
Microchip Model Check before compilation
### Starting build procedure for model: Explorer16_33FJ256GP710_SimpleFilter
### Generating code into build folder: C:\demo\Explorer16_33FJ256GP710_SimpleFilter.X\src
### Invoking Target Language Compiler on Explorer16_33FJ256GP710_SimpleFilter.rtw
### Using System Target File: C:\M91449\MCHP_Blockset\mchp\MCHP_dsPIC_stf.tlc
### Loading TLC function libraries
.....
### Initial pass through model to cache user defined code
...
### Caching model source code
.....
### Writing main
--- Multi Tasking implementation ---
.
### Writing header file Explorer16_33FJ256GP710_SimpleFilter.h
.
### Writing header file Explorer16_33FJ256GP710_SimpleFilter_types.h

***** REMOVED LINES *****
.
### TLC code generation complete.
### Creating HTML report file Explorer16_33FJ256GP710_SimpleFilter_codegen_rpt.html
### Creating project marker file: rtw_proj.tmw
.
### Processing Template Makefile: C:\M91449\MCHP_Blockset\mchp\MCHP_dsPIC.tmf
### Creating Explorer16_33FJ256GP710_SimpleFilter.mk from:
### C:\M91449\MCHP_Blockset\mchp\MCHP_dsPIC.tmf
### Building Explorer16_33FJ256GP710_SimpleFilter: .\Explorer16_33FJ256GP710_SimpleFilter

C:\demo\Explorer16_33FJ256GP710_SimpleFilter.X\src>"C:\Program Files\MATLAB\R2013b\bin\win64\...
xc16-gcc.exe -c -mcpu=33FJ256GP710 -omf=elf -O3 -funroll-loops -fomit-frame-pointer -mpa ...
xc16-gcc.exe -c -mcpu=33FJ256GP710 -omf=elf -O3 -funroll-loops -fomit-frame-pointer -mpa ...
***** REMOVED LINES *****
xc16-gcc.exe -c -mcpu=33FJ256GP710 -omf=elf -O3 -funroll-loops -fomit-frame-pointer -mpa ...
xc16-gcc.exe -mcpu=33FJ256GP710 Explorer16_33FJ256GP710_SimpleFilter_data.o
Explorer16_33FJ256GP710_SimpleFilter_main.o MCHP_ADC1_Interrupt.o c:\program files (x86)\...
Explorer16_33FJ256GP710_SimpleFilter_data.o
Explorer16_33FJ256GP710_SimpleFilter_main.o
MCHP_ADC1_Interrupt.o
MCHP_UART2_Interrupt.o
MCHP_UART2_PingPongSwitch.o
```

MPLAB® Device Blocks for Simulink®

```
rtGetInf.o
rtGetNaN.o
rt_nonfinite.o
Explorer16_33FJ256GP710_SimpleFilter.o
(C:\PROGRA~2\MICROC~1\xc16\v1.20\lib\dsPIC33F\libp33FJ256GP710-elf.a)ConfigIntTimer3.o
(C:\PROGRA~2\MICROC~1\xc16\v1.20\lib\dsPIC33F\libp33FJ256GP710-elf.a)OpenTimer3.o
***** REMOVED LINES *****
(C:\PROGRA~2\MICROC~1\xc16\v1.20\lib\libm-elf.a)fcompare.eo
(C:\PROGRA~2\MICROC~1\xc16\v1.20\lib\libm-elf.a)floatundisf.eo
```

Program Memory [Origin = 0x200, Length = 0x2a9fe]

section	address	length (PC units)	length (bytes) (dec)
.text	0x200	0x6a0	0x9f0 (2544)
.text	0x8a0	0x7f6	0xbf1 (3057)
.dinit	0x1096	0x8e	0xd5 (213)
.text	0x1124	0x44	0x66 (102)

Total program memory used (bytes): 0x171c (5916) 2%

Data Memory [Origin = 0x800, Length = 0x7800]

section	address	alignment gaps	total length (dec)
.ndata	0x800	0	0x7e (126)
.nbss	0x87e	0	0x82 (130)
.ndata	0x900	0	0x2 (2)
.nbss	0x902	0	0x2 (2)
__0434CB405284c72e	0x7f80	0	0x12 (18)
__0434CA205284c72e	0x7fc0	0	0x12 (18)

Total data memory used (bytes): 0x128 (296) <1%

Dynamic Memory Usage

region	address	maximum length (dec)
heap	0	0 (0)
stack	0x904	0x767c (30332)

Maximum dynamic memory (bytes): 0x767c (30332)

```
*** Created "executable": ..\..\Explorer16_33FJ256GP710_SimpleFilter.elf [33FJ256GP710]"
-----
* HTML report file: Explorer16_33FJ256GP710_SimpleFilter_codegen_rpt.html
-----
* MPLAB 8 project: Explorer16_33FJ256GP710_SimpleFilter.mcp
-----
* MPLAB X project: Explorer16_33FJ256GP710_SimpleFilter.X then select the project folder
-----
* FLASH compiled model: Explorer16_33FJ256GP710_SimpleFilter.elf on microcontroller 33FJ256GP710
Flash after compilation: off switch: (on/off)
-----
## Successful completion of build procedure for model: Explorer16_33FJ256GP710_SimpleFilter
>>
```

The end of the model compilation output message provides a sum-up of generated files with clickable links. Particularly, clicking on “FLASH” will program the microcontroller directly from MATLAB, provided one recognized programmer is connected. If many programmers are connected, MATLAB will automatically choose the latest programmer that successfully programs the current model. Clicking once on the “on” from the “Flash after compilation” switch option will automatically program the binary file after the Simulink model is compiled.

2.3.2 Output from Code Generation

Resulting files created within the current MATLAB folder are:

- a folder ending with `.x` that contains the generated MPLAB X IDE project with all the required sources files
- an `.elf`, `.cof`, or `.hex` binary file (dependent on the Compiler Option block settings)

<p>Note: Generated files are always over-written if they already exist. To preserve files, rename the generated MPLAB X IDE project folder to avoid any over-writing at the next Simulink model compilation.</p>

Optionally, Simulink can produce an HTML report with the generated C code, including back-and-forth links from blocks to code and from code to blocks. This option is found in the Simulink configuration panel (or <CTRL+E> shortcut). Refer to Simulink documentation for further details.

The code generation process is incremental, which means that only files that contain changes are regenerated to save time. However, this process occasionally leads to errors. If an unexpected error appears, it is usually worth deleting all created files from the current folder (i.e., the `slprj` temporary MATLAB folder, the folder ending with `.x`, and the binary files created) and restarting the code generation.

2.3.3 Program the Microcontroller from MATLAB

The end of compilation output messages provides links to easily start a script to program the microcontroller, as discussed previously in **Section 2.3.1 “Perform Code Generation”**. It is also possible to start the script manually by typing the instruction `picflash` at the MATLAB prompt. A dialog box will appear where a binary file (`.elf`, `.hex` or `.cof`) may be selected to flash. Ensure that MPLAB X IDE is not open as only one application (MPLAB X IDE or MATLAB) can use the programmer at one time.

For any problems related to programmer drivers, refer to MPLAB X IDE documentation.

2.4 MICROCONTROLLER–MATLAB COMMUNICATIONS

Communication between the MCU and MATLAB is used for data analysis, logging, and parameters update.

- Microcontroller–MATLAB UART Connexion: Physical Layer
- Log and Plot Data with MATLAB: The picgui Interface
- On-the-Fly Parameter Updates

2.4.1 Microcontroller–MATLAB UART Connexion: Physical Layer

While running a compiled Simulink model, the MCU can communicate with MATLAB through the UART interface.

If the target hardware does not have an UART connector, or if the computer running MATLAB does not have an UART connector, there are a collection of USB cables that create a virtual UART interface on the computer side, and at the other end provide the necessary TTL- or CMOS-level voltage. The cables can be connected directly to the MCU UART Tx and Rx pins. One of these products uses FTDI® chips that may be found on the [FTDI web site](#). Other cables may be found that are based on Prolific® chips on the [Prolific web site](#). Both chips provide high bandwidths (above 921600 bauds).

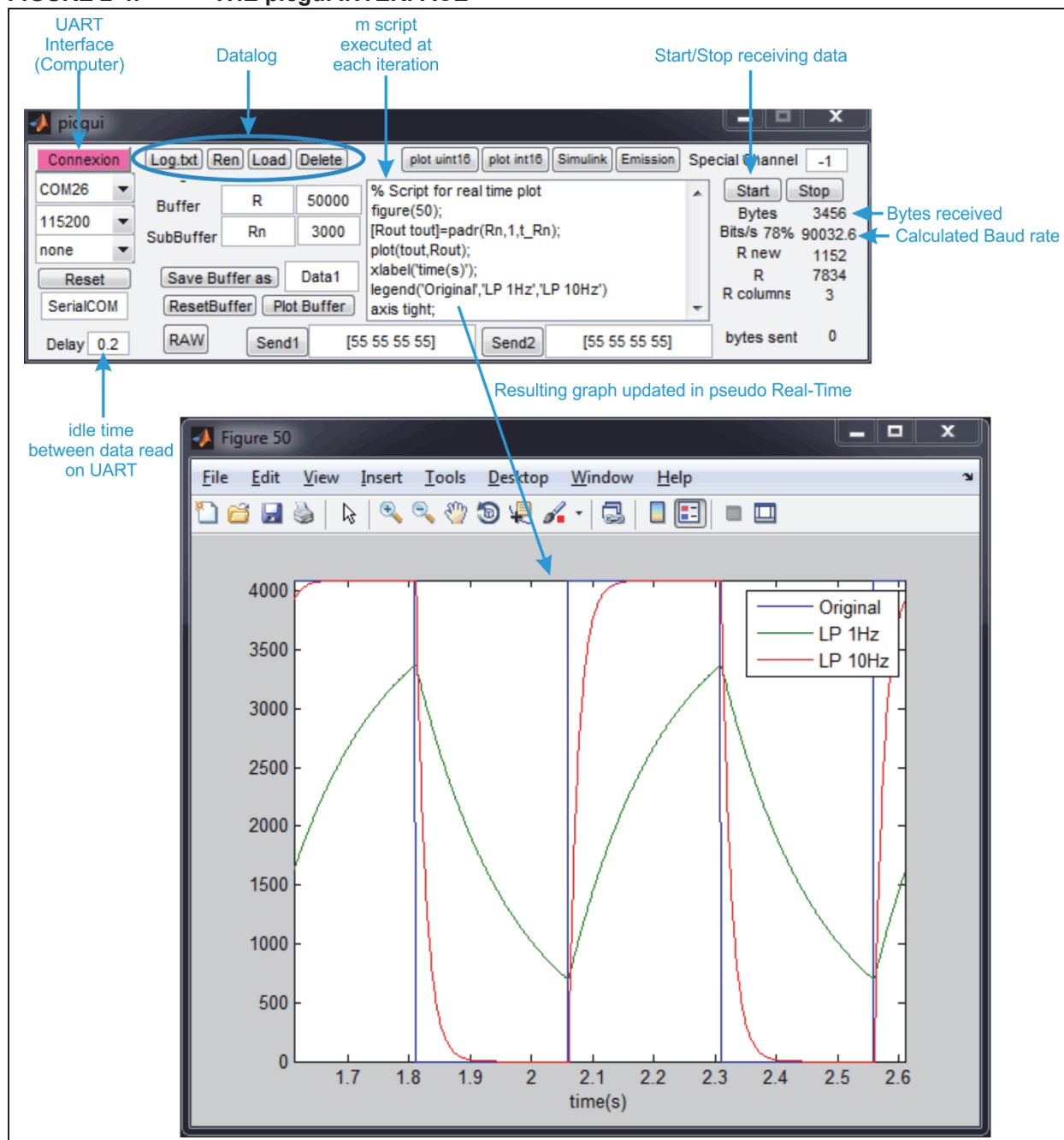
Another option is to connect a [Bluetooth® module](#) like RN41, RN42, or RN52 configured with a Serial Port Profile (SPP) to the MCU. This Bluetooth peripheral is recognized as a virtual UART on the computer side, allowing the sending and receiving of data over the air. Bandwidth, however, is not as high as with a cable.

2.4.2 Log and Plot Data with MATLAB: The picgui Interface

The picgui script, or block, opens the MATLAB interface as shown on Figure 2-4. This interface can read data from and write data to a UART port. The interface can also decode the protocol used by the Tx-MATLAB block that allows the sending of up to 16 different values, 8 or 16 bits at various rates. Received values can then be analyzed in pseudo real-time with a user-written script that can use any MATLAB functions to plot results, calculate data variances, Fourier transform, etc.

Figure 2-4 applies to the sample module. The picgui interface connects through an UART cable to the Explorer 16 board programmed by the Simulink model. The interface reads data from the UART, decodes the protocol from the block “UART Tx-MATLAB” and provide values in the form of a matrix Rn . Each Column of Rn correspond to one input of the block “UART Tx-MATLAB”. Values in Rn can then be analyzed or plotted in pseudo real-time with a customizable script. In this example, values are plotted. The blue curve corresponds to the internally-generated square wave, the green and red curves correspond respectively to the output of the 1 Hz and 10 Hz low-pass filters.

FIGURE 2-4: THE picgui INTERFACE



2.4.2.1 BAUD RATE

Select and configure the UART port used on the PC side. The baud rate must match the baud rate set in the model block “UART Configuration” (see Figure 2-2). For non-standard baud rate, selecting the “Custom” value provides an edit box where a numerical baud rate value can be directly entered. Ensure that the hardware supports the chosen baud rate.

2.4.2.2 CONNECTION

Once the UART port is selected and baud rate set, the connection should be opened by clicking on **Connexion**. Depending on your hardware, this connection might take some time. During the attempt to connect, messages are displayed in the MATLAB command windows.

A reconnection is required after a change in the UART configuration.

Clicking the **Reset** button will release the connection.

2.4.2.3 START/STOP

Once a successful connection is established (i.e., the UART port is open), receiving data from the UART is begun by clicking the **Start** button. The picgui iterates the following actions:

1. Waits the time specified under “Delay”. The default is 0.2 seconds.
2. Reads and buffers values received on the UART.
3. If a log has been started, appends new values to the `log.txt` file.
4. Decodes the incoming stream generated by the Tx-MATLAB block.
5. Concatenates new values to the R matrix. Resizes R to keep the last `xx` lines of matrix R (see buffer and sub-buffer in the picgui GUI). Copies the last `zz` lines of R into Rn (with $zz \leq xx$).
6. Executes user script.

2.4.2.4 RAW

Action 4 can be switched off by the **Raw** button. This could be useful for checking raw data or a data stream that is not encoded with the protocol created by the Tx-MATLAB block.

2.4.2.5 R AND Rn FORMAT; T_R AND T_Rn TIMESTAMP

Each R and Rn column corresponds to one input channel of block “UART Tx-MATLAB” (i.e., each column contains one variable).

Each R and Rn line contains one variable. Thus a line contains NaN and one variable value in the corresponding column. Line numbers correspond to the chronological ordering of variables.

t_R and t_Rn are time vectors estimated for each line of the R and Rn matrix, respectively. The time estimation is based on the arrival time of data and thus it is not precise. Time estimation t_R and t_Rn could be used to visualize data but should not be used for fine data analysis. Instead, a time vector should be reconstructed based on the knowledge of the embedded system.

```
>> Rn([end-8:end], :) % last 8 lines of R:
```

```
ans =
```

NaN	1578	NaN
NaN	NaN	3894
4095	NaN	NaN
NaN	1594	NaN
NaN	NaN	3906
4095	NaN	NaN
NaN	1609	NaN
NaN	NaN	3918
4095	NaN	NaN

```
>>
```

2.4.2.6 PADR SCRIPT

The `padr` script removes NaN values from R or Rn which may simplify data analysis and plotting. The function is called with two parameters: `padr(Rn, Coltrig)`. The function works in two distinct steps. In a first step, each NaN value of the input Rn or R matrix is replaced by the first numerical from the same column in the previous line. In a second step, only lines that originally contain numerical values in the column indexed by the Coltrig parameters are kept. This script could be used only if all variables are being sent with the same sampling rate. (Although it is possible to send one variable at 10 Hz and others variables at 1 kHz using several blocks with Tx-MATLAB executed at different sampling rates). The time vector could be added as a third parameter to keep track of the estimated time arrival of data.

```
% Remove NaN values and keep track of the time vector estimation
>> [Rout tout] = padr(Rn,1,t_Rn);
```

```
>> Rout([end-2:end],:)
```

```
ans =
```

4095	1578	3894
4095	1594	3906
4095	1609	3918

```
>>
```

The following is an example of how to plot data in pseudo real-time. The plot appears in the GUI shown in Figure 2-4.

```
% Script for real time plot
figure(50);
[Rout tout]=padr(Rn,1,t_Rn);
plot(tout,Rout);
xlabel('time(s)');
legend('Original','LP 1Hz','LP 10Hz')
axis tight;
```

2.4.2.7 DATALOG BUTTONS

The picgui interface can log the raw data stream received from the UART for later analysis. This is useful when long logs are required. Pressing the **Log.txt** button once will start the datalog mechanism. All data received from the UART is written within the file `log.txt` that was created in the working folder. The number below the button shows the number of bytes logged which corresponds to the size of the `log.txt` file.

Pressing on the **Ren** button once renames the file `log.txt` with a name built from the current date and time (format: `yyyy_mm_dd_at_hh_mm_ss.txt`). If logging is enabled when the **Ren** button is pressed, a new file `log.txt` is instantaneously created so as to continue the log without any loss of samples.

The **Delete** button deletes the file `log.txt`. If logging is enabled, a new file will be instantaneously created.

At the end of a log session, several files containing raw data will be produced. The **Load** button allows these files to be opened, decoded and stored into `.mat` files. Several files at a time may be selected.

It is possible to log raw values using other software or hardware. For example, [RealTerm](#) may be used to log a data stream, which may then be decoded with the picgui **Load** button.

2.4.2.8 SEND1 AND SEND2 BUTTONS

The Send1 and Send2 buttons send the vector values (in the box at their respective right sides) through the UART. Figure 2.5 shows a simple sample model where three parameters can be modified online while the model is running in the MCU.

2.4.2.9 INFORMATION DISPLAYS

Below the Start and Stop buttons is a display of information.

The first line shows the number of bytes received at each iteration.

The second line shows the average baud rate with the usage percentage of the physical layer. This value could be predicted from the model. For example, the sample model indicates that the UART Tx-MATLAB block should send 9 bytes at each time step. The UART Configuration block shows that a maximum of 11.48 bytes could be sent at each time step.

EQUATION 2-2: BYTES SENT AT EACH TIME STEP

$$\underbrace{115200 \cdot \left(1 - \frac{0.37}{100}\right)}_{\text{Real Baud Rate}} \cdot \underbrace{\frac{1}{10}}_{\text{10 bits per byte}} \cdot \underbrace{1e-3}_{\text{1ms time stamp}} = 11.48$$

Since only 9 bytes are sent at each time step, we are using $\frac{9}{11.48} = 78\%$ of the available bandwidth (i.e., for a bandwidth of 89983 bauds and picgui measured on a 0.2s period, the averaged bandwidth is 90033 bauds, where difference is due to buffer mechanisms).

2.4.3 On-the-Fly Parameter Updates

The MPLAB Device Blocks for Simulink do not provide a built-in mechanism for the online tuning of parameters. However, the model presented in Figure 2-5 demonstrates a very simple logic that allows modification of variables online using the picgui interface. This model generates a sine wave which is sent through the UART that can be plotted with the picgui interface. The frequency of the sine wave is set using the on-board potentiometer. The amplitude of the sine wave can be modified through the UART by sending an appropriate sequence.

2.4.3.1 MODEL OVERVIEW

The Simulink model `Explorer16_33FJ256GP710_OnlineTuning.mdl` is intended to run on an Explorer 16 Demo board (Figure 2-5). It generates a sine wave that is sent through the UART to plot the resulting curve in the picgui interface. The sine wave frequency depends on the on-board potentiometer position. The sine wave amplitude can be finely tuned through the UART connection and the picgui interface. The UART Rx dialog box (Figure 2-6) and the encapsulated subsystems to decode the UART input flow are involved (Figure 2-7 and Figure 2-8). The embedded logic allows the remote setting of the three variables A, B and C present in the model. The remote sequence to modify the variable A has the format: [55 MSB LSB 01] ; for variable B: [55 MSB LSB 02] ; and for variable C: [55 MSB LSB 03].

2.4.3.2 MODEL DETAIL

Three `uint16_T` memory variables A, B, and C are declared in the model. They can be tuned through the UART. The subsystem “Decode UART” (Figure 2-7) embeds the logic to detect an update sequence and to eventually update the targeted variable. The variables can be used like any other constants that are in the model. The fractional (`sfix16_En14`) output of the sine wave look-up table is multiplied by the variable A to modify the amplitude of the sine wave.

With the logic implemented, an update sequence is composed of 4 bytes. The sequence starts with the value 55, followed by the new MSB and LSB value of the variable, and then an index $\in [1\ 3]$ to select, respectively, one of the three variables ($1 \rightarrow A$, $2 \rightarrow B$, and $3 \rightarrow C$). The MSB and LSB could be calculated through the MATLAB instruction `typecast (swapbytes(uint16(2355)), 'uint8')`, where 2355 is the value being converted, and the result is [9 51]. Simpler instructions like “/” or “rem” could also be used instead. The UART Rx block is configured to read only one byte at each time step. The first block output “Nbr” counts the number of bytes read; thus it takes the value of 0 or 1. This counter is used to trigger the subsystem that analyzes the incoming data.

If a more complex logic is required, Stateflow flow could be a good candidate.

FIGURE 2-7: SIMULINK LOGIC FOR TUNABLE PARAMETERS THROUGH USING A UART CONNECTION - DECODE UART SUBSYSTEM

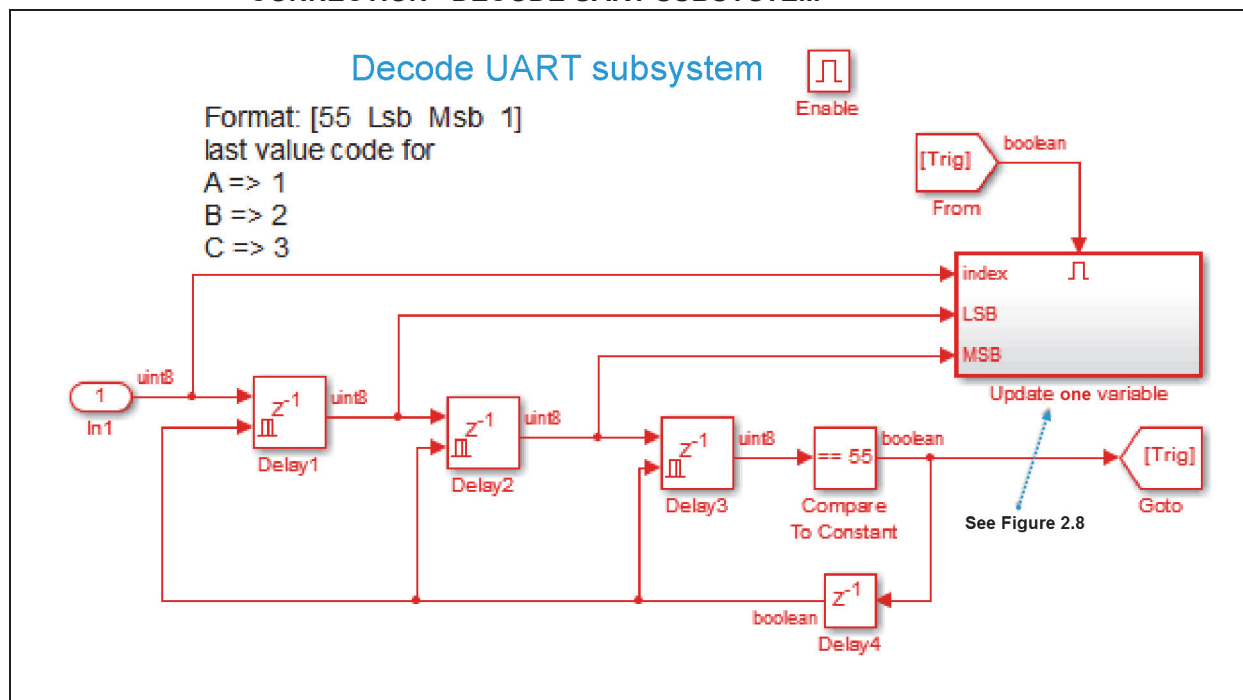
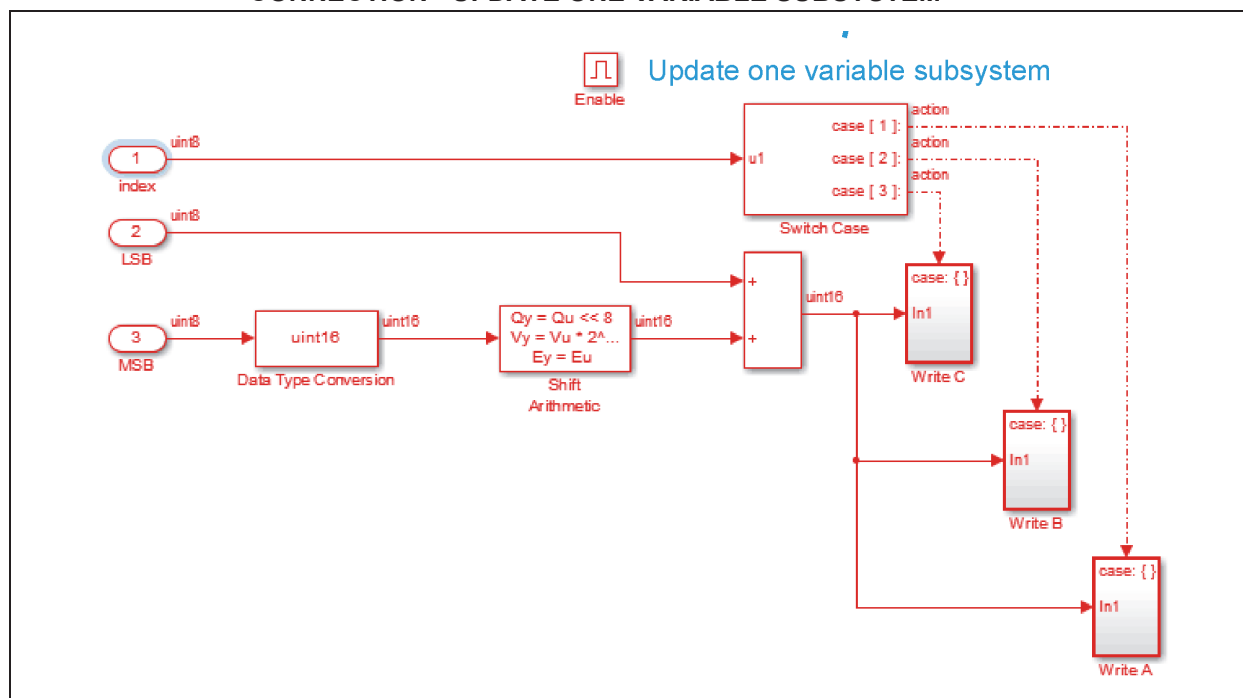


FIGURE 2-8: SIMULINK LOGIC FOR TUNABLE PARAMETERS THROUGH USING A UART CONNECTION - UPDATE ONE VARIABLE SUBSYSTEM



Chapter 3. Library Content

An overview and the various groups of the library of the MPLAB® Device Blocks for Simulink® are discussed:

Overview

- Library Structure
- Supported Devices

Groups

- System Configuration Group
- Digital IO group
- Analog IO group
- PWM IO group
- QEI group
- Pulse Input/Output group
- BUS UART group
- Bus SPI Group
- Bus I²C™ Group
- User Functions group

3.1 LIBRARY STRUCTURE

The library of the MPLAB® Device Blocks for Simulink® is structured as follow:

- System Configuration
 - Microchip Master
 - Simulink Reset Config
 - Compiler Options
 - Data Sheet
- Digital IO
 - Digital Input
 - Digital Output
- Analog IO
 - ADC
 - Comparator / Op-Amps / Volt Ref
- PWM IO
 - PWM
 - PWM High-Speed
- QEI
 - QEI
- Pulse Input/Output
 - Change Notification
 - Input Capture
 - Output Compare - HW
 - Output Compare - SW
- BUS UART
 - UART Configuration
 - UART Rx
 - UART Tx
 - UART Tx-MATLAB
 - PICGUI Graphical Interface
- BUS SPI
 - BUS SPI
- BUS I²C
 - BUS I²C MASTER
- System Functions
 - Interrupt
 - Reset
- User Functions
 - C function Call
- MATLAB Script & GUI
 - picflash to program a microcontroller from MATLAB
 - picgui a graphical user interface to plot data sent with the block UART Tx-MATLAB
 - padr script to remove NaN values from decoded matrix by picgui.

3.2 SUPPORTED DEVICES

Below are tables of supported microcontrollers (MCUs).

TABLE 3-1: dsPIC30F DEVICES

dsPIC30F2010	dsPIC30F5015
dsPIC30F2011	dsPIC30F5016
dsPIC30F2012	dsPIC30F6010
dsPIC30F3010	dsPIC30F6010A
dsPIC30F3011	dsPIC30F6011
dsPIC30F3012	dsPIC30F6011A
dsPIC30F3013	dsPIC30F6012
dsPIC30F3014	dsPIC30F6012A
dsPIC30F4011	dsPIC30F6013
dsPIC30F4012	dsPIC30F6013A
dsPIC30F4013	dsPIC30F6014
dsPIC30F5011	dsPIC30F6014A
dsPIC30F5013	dsPIC30F6015

TABLE 3-2: dsPIC33EP DEVICES

dsPIC33EP128GP502	dsPIC33EP256MU204	dsPIC33EP512MC504
dsPIC33EP128GP504	dsPIC33EP256MU206	dsPIC33EP512MC506
dsPIC33EP128GP506	dsPIC33EP256MU502	dsPIC33EP512MC806
dsPIC33EP128MC202	dsPIC33EP256MU504	dsPIC33EP512MU810
dsPIC33EP128MC204	dsPIC33EP256MU506	dsPIC33EP512MU814
dsPIC33EP128MC206	dsPIC33EP256MU806	dsPIC33EP64GP502
dsPIC33EP128MC502	dsPIC33EP256MU810	dsPIC33EP64GP503
dsPIC33EP128MC504	dsPIC33EP256MU814	dsPIC33EP64GP504
dsPIC33EP128MC506	dsPIC33EP32GP502	dsPIC33EP64GP506
dsPIC33EP128MU202	dsPIC33EP32GP503	dsPIC33EP64MC202
dsPIC33EP128MU204	dsPIC33EP32GP504	dsPIC33EP64MC203
dsPIC33EP128MU206	dsPIC33EP32MC202	dsPIC33EP64MC204
dsPIC33EP128MU502	dsPIC33EP32MC203	dsPIC33EP64MC206
dsPIC33EP128MU504	dsPIC33EP32MC204	dsPIC33EP64MC502
dsPIC33EP128MU506	dsPIC33EP32MC502	dsPIC33EP64MC503
dsPIC33EP256GP502	dsPIC33EP32MC503	dsPIC33EP64MC504
dsPIC33EP256GP504	dsPIC33EP32MC504	dsPIC33EP64MC506
dsPIC33EP256GP506	dsPIC33EP512GP502	dsPIC33EP64MU202
dsPIC33EP256MC202	dsPIC33EP512GP504	dsPIC33EP64MU204
dsPIC33EP256MC204	dsPIC33EP512GP506	dsPIC33EP64MU206
dsPIC33EP256MC206	dsPIC33EP512GP806	dsPIC33EP64MU502
dsPIC33EP256MC502	dsPIC33EP512MC202	dsPIC33EP64MU504
dsPIC33EP256MC504	dsPIC33EP512MC204	dsPIC33EP64MU506
dsPIC33EP256MC506	dsPIC33EP512MC206	
dsPIC33EP256MU202	dsPIC33EP512MC502	

TABLE 3-3: dsPIC33FJ DEVICES

dsPIC33FJ128GP202	dsPIC33FJ128MC802	dsPIC33FJ64GP206
dsPIC33FJ128GP204	dsPIC33FJ128MC804	dsPIC33FJ64GP206A
dsPIC33FJ128GP206	dsPIC33FJ12GP201	dsPIC33FJ64GP306
dsPIC33FJ128GP206A	dsPIC33FJ12GP202	dsPIC33FJ64GP306A
dsPIC33FJ128GP306	dsPIC33FJ12MC201	dsPIC33FJ64GP310
dsPIC33FJ128GP306A	dsPIC33FJ12MC202	dsPIC33FJ64GP310A
dsPIC33FJ128GP310	dsPIC33FJ16GP304	dsPIC33FJ64GP706
dsPIC33FJ128GP310A	dsPIC33FJ16MC304	dsPIC33FJ64GP706A
dsPIC33FJ128GP706	dsPIC33FJ256GP506	dsPIC33FJ64GP708
dsPIC33FJ128GP706A	dsPIC33FJ256GP506A	dsPIC33FJ64GP708A
dsPIC33FJ128GP708	dsPIC33FJ256GP510	dsPIC33FJ64GP710
dsPIC33FJ128GP708A	dsPIC33FJ256GP510A	dsPIC33FJ64GP710A
dsPIC33FJ128GP710	dsPIC33FJ256GP710	dsPIC33FJ64GP802
dsPIC33FJ128GP710A	dsPIC33FJ256GP710A	dsPIC33FJ64GP804
dsPIC33FJ128GP802	dsPIC33FJ256MC510	dsPIC33FJ64MC202
dsPIC33FJ128GP804	dsPIC33FJ256MC510A	dsPIC33FJ64MC204
dsPIC33FJ128MC202	dsPIC33FJ256MC710	dsPIC33FJ64MC506
dsPIC33FJ128MC204	dsPIC33FJ256MC710A	dsPIC33FJ64MC506A
dsPIC33FJ128MC506	dsPIC33FJ32GP202	dsPIC33FJ64MC508
dsPIC33FJ128MC506A	dsPIC33FJ32GP204	dsPIC33FJ64MC508A
dsPIC33FJ128MC510	dsPIC33FJ32GP302	dsPIC33FJ64MC510
dsPIC33FJ128MC510A	dsPIC33FJ32GP304	dsPIC33FJ64MC510A
dsPIC33FJ128MC706	dsPIC33FJ32MC202	dsPIC33FJ64MC706
dsPIC33FJ128MC706A	dsPIC33FJ32MC204	dsPIC33FJ64MC706A
dsPIC33FJ128MC708	dsPIC33FJ32MC302	dsPIC33FJ64MC710
dsPIC33FJ128MC708A	dsPIC33FJ32MC304	dsPIC33FJ64MC710A
dsPIC33FJ128MC710	dsPIC33FJ64GP202	dsPIC33FJ64MC802
dsPIC33FJ128MC710A	dsPIC33FJ64GP204	dsPIC33FJ64MC804

3.3 SYSTEM CONFIGURATION GROUP

The system configuration group contains blocks to configure Simulink for code generation and blocks to set system parameters related to the targeted MCU. The available blocks are:

- Microchip Master Block
- Simulink Reset Config Block
- Compiler Options Block
- Data Sheet Block

3.3.1 Microchip Master Block

The Microchip Master block is required when the model is used for code generation based on the Microchip blockset. This block checks and corrects Simulink parameters to comply with code generation constraints. This block defines the essential parameters related to the targeted MCU. To set up these parameters, display the block parameter dialog box and select a tab:

- The Main Tab
- The Oscillator Tab
- The Memory and Debug Tabs

3.3.1.1 THE MAIN TAB

Use this tab to set up general options for MCU operation.

TABLE 3-4: MAIN TAB OPTIONS

Option	Description
Target	Select the targeted MCU from the list. Note: Selecting another MCU requires closing and reopening the block to get updated options parameters.
Time-Step synchronization	View the triggering source for the time step. Timer 1 is usually used, but it can be replaced by an ADC end-of-conversion trigger or by a combination of Timer 1 + ADC, depending on peripherals configuration.
Busy Flag Port	Set one pin as a digital output. This pin will be set to 1 during the execution of the step and will be set to 0 during the Wait state (idle time). Monitoring this pin with an oscilloscope allows retrieval of information on the real-time execution. The duty cycle represents the CPU load.
Power Save Mode	This check box turns on the MCU's low-power mode during the idle time (i.e., between time-steps).
<i>Other Power Up Options</i>	Depends on the MCU used (e.g., Brown-out Reset, Power-on Reset, or use of Master Clear Reset pin.)

3.3.1.2 THE OSCILLATOR TAB

Use this tab to set up the MCU oscillator.

TABLE 3-5: OSCILLATOR TAB OPTIONS

Option	Description
Oscillator Source	Depending on your selected device, some or all of these oscillator options may be available for selection: <ul style="list-style-type: none"> • Internal Fast RC (FRC) • Internal Low Power RC (LPRC) • Crystal or Ceramic Resonator • External Clock
Oscillator Frequency (Hz)	Enter an oscillator frequency in Hertz. For an external oscillator, the possible range for the selected option is shown between brackets. Choosing an oscillator outside this range might not work. For internal oscillators FRC or LPRC, the default frequency is always shown. The selected value must be within the range of the default value $\pm 15\%$; otherwise a warning is printed and the value is forced to the default value. Modifying the default value allows correction for any mismatch of the internal oscillator frequency. Note: This change does not modify the MCU running frequency. It is different from fine-tuning the frequency of the internal oscillator available on some chips.
Achieved Instructions per Second (MIPS)	Provides the number of instructions executed per second. This value depends on the oscillator source frequency, the PLL configuration and the number of cycles per instruction for the selected MCU.
Activate clock multiplier / divider (PLL)	This check box will appear for MCUs that support PLL for the selected oscillator. Activating this option will show in "Targeted Instructions per Second".
Targeted Instructions per Second	Set the number of instructions per second (IPS, not MIPS) for the MCU. The reachable range is shown between brackets for an MCU with a PLL that can be fine-tuned, or a discrete value will be shown between braces for an older MCU with coarse PLL options. The IPS set should be inside the given range. The block will configure the MCU using the closest reachable frequency. The frequency achieved is updated in the "Achieved Instruction per Seconds" text box. The blockset will tolerate a PLL solution that could result in an IPS number 5% above the upper bound. The resulting IPS should be checked to insure it is within the acceptable range.
<i>Other Oscillator Options</i>	Depending on the MCU used, other options may be available (e.g., Clock Switching and Monitor, or Dual speed startup.)

3.3.1.3 THE MEMORY AND DEBUG TABS

Set memory protection and debugging pins options. Available options depend on the selected device.

3.3.2 Simulink Reset Config Block

The Simulink Reset Config block does not have a GUI. The Simulink configuration is checked when trying to open this block. Configuration messages are displayed at the MATLAB Command prompt. This block might be used, for example, when working on a different computer, as it will allow for path correction when switching between systems.

3.3.3 Compiler Options Block

The Compiler Option block provides for the selection of a compiler and some compilation options. Both the PICC30 and the MPLAB XC16 C compilers are supported. The block lists all the compilers installed on the system but no compatibility check is performed. The blockset is tested with the most recent compiler version; so it is recommended that the most recent compiler available on the Microchip web site be used.

To set up options, display the block parameter dialog box and select a tab:

- The Compiler Tab
- The Optimization Tab
- The Output Tab

3.3.3.1 THE COMPILER TAB

Use this tab to select and set up the compiler.

TABLE 3-6: COMPILER TAB OPTIONS

Option	Description
Select Compiler	Select the Microchip compiler within the list of installed compilers.
Always use the latest compiler	Select whether or not to use the latest compiler. View the Release Notes that come with the compiler to determine what compiler features you require. Checked: The most recent compiler will be used (based on the compiler version) and the compiler list will be removed. Unchecked: A list of all compilers found on the system, ordered in decreasing version number, is shown.
Use 64-bit double	Select whether or not to use 64-bit floating point as <code>double</code> . Checked: Use 64-bit floating point as <code>double</code> , which is more precise. Unchecked: (Default) The compiler will treat <code>double</code> (64-bit floating point) as <code>single</code> (32-bit floating point), which makes the code execute faster.

3.3.3.2 THE OPTIMIZATION TAB

Select the Math Library to be used, as well as some optimization options. These options may change depending on the compiler selected.

3.3.3.3 THE OUTPUT TAB

Select the binary output format from ELF, COFF or Hex. Also, there is an option to produce the final assembly listing file.

3.3.4 Data Sheet Block

If you **do not** have an Internet connection, this block will not be able to perform downloading and version checking of data sheets.

If you **do** have an Internet connection, downloading of the data sheet might take some time depending on your Internet connection, firewall configuration, or size of the data sheet.

Opening the Data Sheet block for the first time will:

1. Download the data sheet (PDF) for the selected MCU (within the Master Block) from the Microchip web site to the MATLAB `temp` folder
2. Open the data sheet

Each time after, opening the Data Sheet block will:

1. Check the data sheet status:
 - a) If there is a data sheet in the `temp` folder, and the block last checked for an update on the Microchip web site within the last 3 weeks, the existing data sheet is opened. If the check was more than 3 weeks ago, then a check for an update is performed and any updated data sheet is downloaded.
 - b) If there is no data sheet in the `temp` folder, one will be downloaded from the Microchip web site
2. Open the data sheet

3.4 DIGITAL IO GROUP

The Digital Input and Digital Output blocks configure MCU pins as digital input/output and allow the writing and reading of these pins.

- Digital Input Block
- Digital Output Block

3.4.1 Digital Input Block

The Digital Input block configures pins as digital input that allows reading of the pin state. The block outputs correspond to the selected digital input. One block can configure only one port (i.e., PORTA or PORTB). For the selected port, many pins can be selected as digital inputs. Port numbers are set within the Refs field as a vector [0 1 4 6].

When more than one pin is configured within the block, the option “Exact simultaneous sample” appears. When this option is checked, the generated code will read the whole port once and then extract pin values, allowing sampling of all pins simultaneously. When unchecked, pin values are read in sequence.

When 16 bits of one port are being read (Refs field set to [0 : 15]), the option “set output as one uint16” appears. When this option is checked, the 16 boolean port outputs will be replaced with one uint16 port output reflecting the port’s value. The option “Exact simultaneous sample” is hidden.

The block execution ordering choice allows the addition of one extra block input, block output, or both input and output. This additional input and/or output does not convey any information and will not generate any code. However, it forces the execution order of two Simulink blocks (for example, two Digital Input blocks) that would otherwise have no constraints on their execution order.

The Sample Time option defines the sampling time for this block. For discrete blocks, the notation is $[T_s \ T_0]$ where T_s is the sampling period and T_0 is the initial time offset. The value -1 stands for inherited. When a scalar is given instead of a vector, the value T_0 is inherited.

- -1 stands for inherited
- 0.001 stands for 1 ms sample time
- [0.005 0.002] stands for 5 ms sample time with a time shift of 2 ms
(The Simulink base sample rate defined for the model should be a common divisor of 0.005 and 0.002)

For more on sample time, see the Simulink documentation: "How to Specify the Sample Time".

3.4.2 Digital Output Block

The Digital Output block configures pins as digital output that allows the writing of pin states. The block inputs correspond to the selected digital output. One block can configure only one port (i.e., PORTA or PORTB). For the selected port, many pins can be configured as digital output. Port numbers are set within the Refs field as a vector [0 1 4 6].

When more than one pin is configured within the block, the option “Exact simultaneous update” appears. When this option is checked, the generated code will update all pins simultaneously; otherwise pins are written in sequence.

When the 16 bits of one port are being written (Refs field set to [0 : 15]), the option “set input as one uint16” appears. When this option is checked, the 16 Boolean port inputs will be replaced with one uint16 port input that will be written in the respective port’s register. The option “Exact simultaneous update” is hidden.

The option “Read previous value written” (1/z delay) adds one block output that is a copy of the block input with one delay. This output might be used to toggle an LED without adding a delay (z^{-1}) block.

3.5 ANALOG IO GROUP

The Analog IO Group block configures MCU analog pins. The following blocks are in the Analog IO Group:

- ADC Block
- Comparators / Op-Amps / Voltage Reference Block

3.5.1 ADC Block

The ADC block configures the Analog-to-Digital Converter. The block does not start one sample and convert that on one selected pin, but configures a complete sequence of samples and converts those during the time step. This allows precise control of the sample timing as well as a better integration into the Simulink time step. Only one instance of the ADC block should be present within the Simulink model.

The Mode parameter allows the selection of the 10-bit or the 12-bit ADC for a MCU that has both options. In 10-bit mode, the ADC provides faster sampling and conversions. Thanks to its four internal capacitors, sampling up to four input channels in parallel is possible.

The Output Format allows setting of the conversion format for the ADC as:

- 0000 dddd dddd dddd ==> Unsigned integer
- 0000 sddd dddd dddd ==> Signed integer
- dddd dddd dddd 0000 ==> Unsigned fractional
- sddd dddd dddd 0000 ==> Signed fractional

The bit field representation shown corresponds to a 12-bit conversion and can be adapted for a 10-bit conversion.

Whatever conversion format is selected, the ADC block outputs representation for Simulink are always set to uint 16. A Data Type Conversion block (Simulink standard block) might be required. In that conversion block, the "Stored Integer (SI)" option might be checked instead of the "Real World Value (RWV)", to change the representation of the data without modifying its bit-field. Changing the representation of the data using advanced fixed-point representation like "slope and bias" allows scaling of the data directly to its real physical value without any computation having been done at that point.

In both 10- or 12-bit mode, a one or two sequence conversion can be set, respectively noted as A and B. The sequences A, or A and B, can be repeated several times. The Sample and Hold mechanism of sequence A allows multiplexing of channel 0 (capacitor 0) input to any of the analog inputs. This allows the sampling of a sequence of analog input pins set in the CH0 Sequential Sampling box. Sequence A for capacity 1, 2, or 3 or sequence B allows the choosing of analog input pins but this choice is limited and is kept fixed (no sequential sampling).

The sequence A, or A and B, is repeated corresponding to the number of analog inputs set in the "CH0 Sequential Sampling" times the "Repeat conversion sequence xx times".

Depending on the ADC mode and the selected MCU, options may be added that allow differential sampling between two pins.

The next option, "Sample and Conversions", sets the triggering mechanism for the ADC conversion. Select from:

1. are done as fast as possible at end of previous time step
2. are done as fast as possible, triggered by a user-defined source
3. are stretched to fit exactly previous time step
4. are continuously sampling as fast as possible

The first choice is done as fast as possible at the end of the previous time step. The conversion sequences are started just before the end of the previous time step, so as to be complete for the next time step.

The second choice is done as fast as possible when triggered by a user-defined source that starts the conversion sequence. This trigger might come from the PWM peripheral which allows the synchronization of sampling within the PWM period. The end of the sequence conversion triggers the next time step.

The third choice is stretched to fit exactly the previous time step. Stretching slows down the sampling and conversion sequence so that the complete sampling and conversion sequence time is equal to one time step. The end of the sequence conversion triggers the next time step. This choice allows even spacing of samples.

The fourth choice is continuously sampling as fast as possible to continuously perform the sampling and conversion sequence as fast as possible. The ADC block output will contain the latest updated converted value. This choice minimizes the delay between the conversion and the use of the conversion result at the expense of a high conversion rate and a possible non-chronological output result.

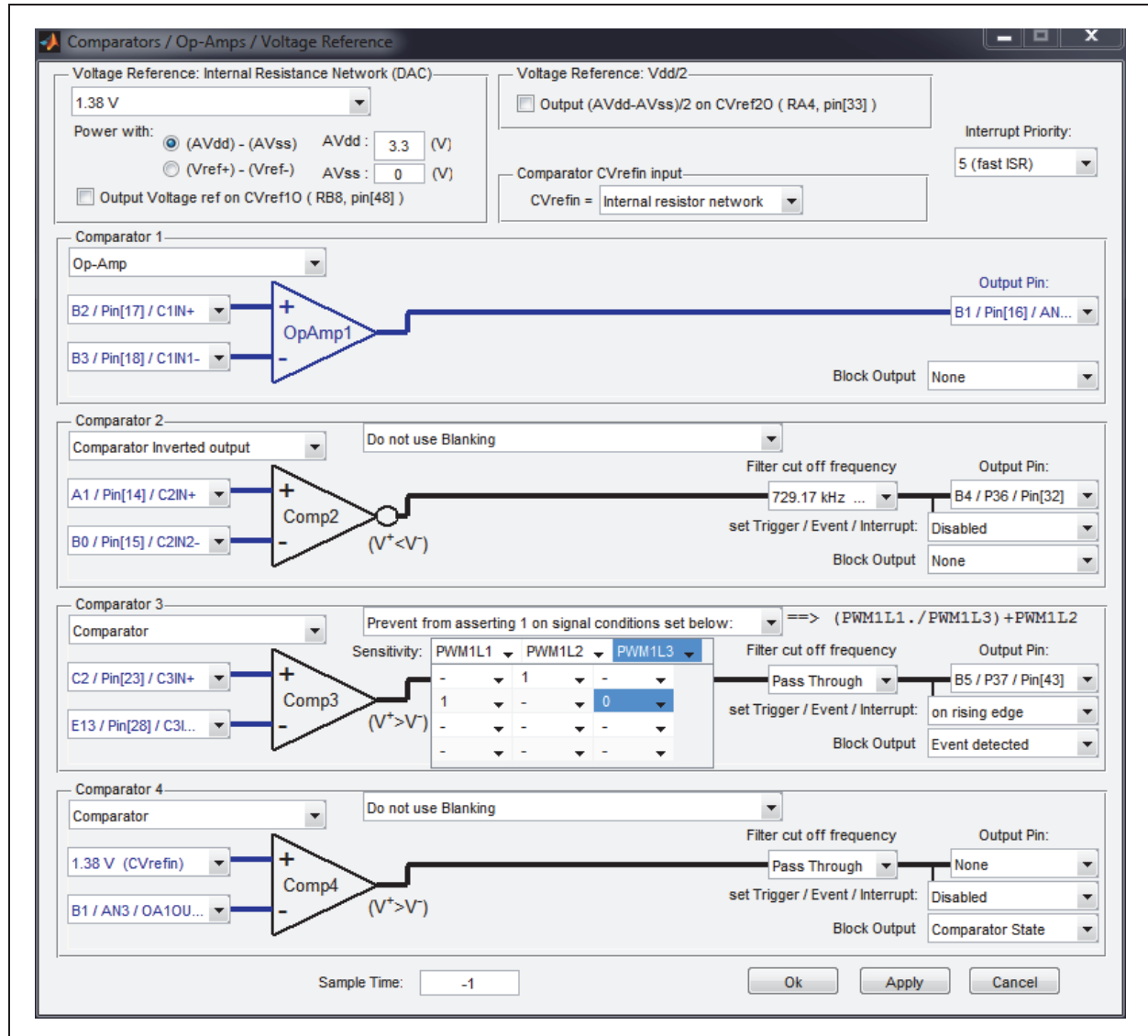
The "DMA" option allows setting the use of either interrupt or DMA when available. DMA should be the preferred method, when available.

The "No sampling rate" option means that the ADC block output is always evaluated at the same frequency as the model time step. It is still possible to use these outputs to do calculations at a lower rate.

3.5.2 Comparators / Op-Amps / Voltage Reference Block

The Comparators / Op-Amps / Voltage Reference Block configures both the voltage reference and the comparators/op-amps. The GUI is shown on Figure 3-1 for the dsPIC33EP64MC504 MCU. Voltage reference can be configured in the upper part of the window. Up to 4 comparators/op-amps can be configured in the lower part of the window.

FIGURE 3-1: COMPARATORS / OP-AMPS / VOLTAGE REFERENCE BLOCK INTERFACE



3.5.2.1 COMPARATORS

Any of the comparators 1 through 4 can be configured as a comparator, an inverted comparator, or an op-amp, provided the device peripherals support these functions. As a comparator or an inverted comparator, the input pin can be selected or the internal reference voltage input (CV_{refin}) can be used.

The blanking function protects the comparator when switching (either from 0 to 1 or from 1 to 0). When blanking is used, a table allows signal state combinations to be set that will prevent switching. Each line represents a condition that prevents switching. Signal value “–” means any value. The equivalent logic equation is shown provided it is reachable; otherwise “no solution” is displayed.

The filter “triple sampling point” is disabled for the filter cut-off frequency set to “Pass Through”; otherwise the cut-off frequency represents the frequency below which no transitions are dropped.

An output pin can be set.

The Set Trigger / Event / Interrupt popup sets an event on either a rising edge, a falling edge or on change (after blanking and filtering). This trigger can then be used in other blocks like PWM-HS. Also, a user interrupt could be used to execute a Simulink subsystem (i.e. asynchronously).

The block output could be used to react to the current comparator output state.

3.5.2.2 OP AMPS

The Op-Amp configuration is the simplest. The two input pins and the output pin of each Op-Amp can be selected.

3.5.2.3 VOLTAGE REFERENCES

Voltage reference 1 allows the creation of a reference voltage through an internal resistance network. This reference voltage can be used as one positive input of the comparator peripherals, or can be output on one MCU CV_{ref1O} pin. The fixed voltages proposed are computed based on the voltage power information given. By default, AV_{dd} = 3.3V and AV_{ss} = 0V.

The internal fractional part defined through the internal resistance could be a block input, making the output voltage reference pin a simple 4-bit digital-to-analog converter.

Voltage reference 2 outputs a fixed voltage on the CV_{ref2O} MCU output pin which is typically V_{dd}/2.

3.6 PWM IO GROUP

Pulse width modulation (PWM) features may be set up in these blocks.

- PWM Block
- PWM High-Speed Block

3.6.1 PWM Block

The PWM block configures the PWM peripheral. Only one instance of this block should be present in the model. Five tabs allow setting the peripheral properties.

- The General Tab
- The Dead Time Tab
- The Fault Tab
- The Event Trigger Tab
- The Override Tab

3.6.1.1 THE GENERAL TAB

This tab may be used to enable PWM outputs and to set the maximum reachable period and a few others parameters.

TABLE 3-7: PWM GENERAL TAB OPTIONS

Option	Description
Distinct Enable of PWM High and PWM Low channels	Activate PWM High and PWM Low Channels independently.
Output Channels (High and Low): [x x x]	Activate the respective PWM output. The [x x x] values show the PWM channels available on the MCU. When the “Distinct Enable of PWM High and PWM Low Channels” option is activated, this option is replaced by the two options: “Output Channels (Low)” and “Output Channel (High)”.
Input Period	The Input Period is the initial period given in seconds for all the activated PWM signals. This timing is also the maximum period for the PWM signal if the period is set as a block input. This time corresponds to an internal uint16 value shown in the Info box. This value is evaluated in the MATLAB workspace as PWMxmax and should be used to scale any period or duty cycle time connected to the block input. Add the PWM period as a block input and thus modulate the PWM period. This uint16 block input must be scaled using the PWMxmax value evaluated within the MATLAB workspace.
Center Aligned Mode	Center all PWM signals. A side effect of this option is the modification of the resolution of the PWM period and duty cycle.
Immediate Update	Update the PWM output before the next cycle starts.

3.6.1.2 THE DEAD TIME TAB

This tab may be used to set the Dead Time. Depending on the selected MCU, one or two Dead Times (respectively called A and B) can be set.

3.6.1.3 THE FAULT TAB

This tab can be used to set the reaction of the PWM output to a Fault input (on the PWM Fault input A and B when available).

TABLE 3-8: PWM FAULT TAB OPTIONS

Option	Description
Fault A Behaviour	Select how the re-enabling of the PWM signal takes place. Two options are available: <ul style="list-style-type: none">• Re-Enable PWM when Fault Pin is deactivated• Use Re-Enable block input after Fault is detected
Use Re-Enable block input after Fault is detected	Add a block input and provide a Boolean signal that re-enables the PWM output.
Fault A Detection block output	Add a block output which is set to Boolean 1 when a Fault is detected.
Pins State	Set the PWM pin state during a fault. "No Change" means that the channel does not react to a fault.

3.6.1.4 THE EVENT TRIGGER TAB

This tab may be used to set a trigger that can be used by other peripherals, like ADC to trigger a conversion. The trigger time given in seconds provides the time from the PWM period start for the trigger event. This trigger can be a block input when the option "Special Event Trigger is a block input" is checked. The added input block time must be scaled using the PWMxmax variable.

3.6.1.5 THE OVERRIDE TAB

This tab may be used to add the OVDCON block input to override PWM signals.

3.6.2 PWM High-Speed Block

The PWM High-Speed block configures the PWM High Speed peripheral. Only one instance of this block should be present in the model. Four tabs allow setting the peripheral properties.

- The General Tab
- The Block Input and Initialization Tab
- The Fault Tab
- The Dead Time Tab

3.6.2.1 THE GENERAL TAB

This tab may be used to enable PWM High-Speed outputs and set the maximum reachable period and few others parameters.

TABLE 3-9: PWM HIGH SPEED GENERAL TAB OPTIONS

Option	Description
PWM Mode	Set the PWM mode for the Low and High side as: <ul style="list-style-type: none"> • Complementary • Redundant • Push-Pull.
Independent Period	Set an independent period for each channel.
Independent Duty Cycle	Set an independent duty cycle for each channel.
Enable Channels	Independently activate each channel low or high.
Max Period(s)	Set the maximum period for each channel or for all channels (depending on the Independent Period options). The Max Period is set in seconds. For each Max Period, one corresponding variable PWM[1..9]max is evaluated within the MATLAB workspace which correspond to the block input value coding for the specified max Period. Block input values should be scaled using these variables.

3.6.2.2 THE BLOCK INPUT AND INITIALIZATION TAB

This tab can be used to set the initial characteristics for PWM signals as the period and the duty cycle. These values must be lower than the Max Period set in the main tab. PWM signal characteristics that are to be changed can be selected and these characteristics will appear as a block input. Block inputs like “Period” or “Duty Cycle” must be scaled using the corresponding channel PWM[1..9]max variable. A trigger can also be set to trigger other peripherals, like an ADC.

3.6.2.3 THE FAULT TAB

This tab can be used to configure the Fault behavior. See **Section 3.6.1.3 “The Fault Tab”**.

3.6.2.4 THE DEAD TIME TAB

This tab can be used to configure the Dead Time configuration. See **Section 3.6.1.2 “The Dead Time Tab”**.

3.7 QEI GROUP

The QEI group contains only the QEI block.

3.7.1 QEI Block

This block configures the QEI peripheral which can be used as a Quadrature encoder or as a pulse counter. To set up QEI parameters, display the block parameter dialog box and select a tab:

- The Main Tab
- The Position Tab
- The Speed Tab
- The Filter Tab

3.7.1.1 THE MAIN TAB

This tab may be used to select the mode in which the QEI will be used. Depending on the selected MCU, options are available to:

- set the pins for the QEA and QEB pins
- invert the polarity of the QEA or QEB pins
- reverse the counting direction
- divide the resolution by a factor 2

3.7.1.2 THE POSITION TAB

This tab can be used to set the output for the position and the mechanism to reset that position. The option "Reset block input Re-Initialize Position" adds a block input which provides for the reinitialization of the position by software.

3.7.1.3 THE SPEED TAB

This tab may be used to set the speed output. This tab will be empty for MCUs whose QEI peripheral does not provide hardware speed measurement. For more advanced QEI peripheral, two-speed measurement can be used.

The first method of speed measurement is a (hardware) differential of the position counter. The second method of speed measurement is a period measurement. Both methods can be used at the same time (resulting as two blocks outputs). When using the period measurement, the maximum time between two pulses must be provided. The peripheral internal timer will be configured to be able to measure that period.

The period block output must be scaled using the QEIxmax variable evaluated in the MATLAB workspace. The value of QEIxmax corresponds to the Max Period set.

3.7.1.4 THE FILTER TAB

This tab may be used to set up filtering on the QEI inputs pins based on a triple sampling method. The minimum cut-off frequency to avoid losing any count depends on the QEI configuration and is given in the GUI.

3.8 PULSE INPUT/OUTPUT GROUP

The Pulse Input/Output group contains blocks used to configure Input Capture and Output Compare and to set up change notification.

- Input Capture Block
- Change Notification Block
- Output Compare (HW) and (SW) Block

3.8.1 Input Capture Block

The Input Capture block allows measuring of the period, the up time, or the down time of an incoming signal. The block GUI can be used to select the Input capture channels used. The type of measurement for each channel is independent and is provided through a vector with the same length as the number of channels used. For each channel, the measurement could be:

- 0: no Measurements (for change detect only)
- 1: up
- 2: down
- 3: up & down
- 4: period on rising edge
- 8: period on falling edge
- 5: up & period on rising edge
- 10: down & period on falling edge

The Max Measurement Time is also set using a vector with the same length as the number of channels used. A value of 0.001 means that the maximum measurement is 1 ms. Any event time (either up, down, or period) longer than 1 ms might be misread.

The block output is the uint16 raw value measured by the peripheral and must be scaled. Variables are evaluated in the MATLAB workspace that could be used within the Simulink block for scaling purposes. The value ICxmax, where x is the Input Capture channel, is equal to the maximum measurement time previously given in the dialog box and allows precise scaling.

EXAMPLE 3-1: IC1MAX CALCULATION

The maximum period for IC1max is set to 0.001 and IC1max = 27359 (variable available in the MATLAB workspace). The scaled measurement of the block output (named IC1out) is:

$$\frac{0.001 \cdot IC1_{out}}{IC1_{max}} = \frac{0.001}{27359} \cdot IC1_{out}$$

Care should be taken when using data scaling. Either use fractional variables or floating point variables to perform this operation. Otherwise, the result will be 0 no matter what value is read.

Block outputs are updated only when new events occur. If no new event occurs, the last value measured is held at the block output.

The “change detect on” block output will count the number of changes that occur between block read operations. Combining the block output measurement with this additional output will help to determine if the measurement is “fresh” or if the signal input is not changing.

3.8.2 Change Notification Block

The Change Notification block works with logic similar to the Input Capture block. Measurements are less precise than with the Input Capture peripheral.

3.8.3 Output Compare (HW) and (SW) Block

The Output Compare (HW) and (SW) works with logic similar to the Input Capture block but allows the generation of pulses, even a single pulse or repeating pulses within a given period.

The Output Compare (HW) is more precise than the Output Compare (SW) but it requires the use of more timers. If there are not enough timer resources, the blockset will cause an error. Consider using the Output Compare (SW) block. When the period is modified, the Output Compare (HW) might create glitches on the generated pulse. However the Output Compare (SW) is not as glitch prone.

The Output Compare (SW) is more flexible regarding the measurement possibilities as well.

3.9 BUS UART GROUP

The BUS UART group contains blocks to configure and use the UART peripheral. One high-level protocol is designed and allows quick sending and visualizing of data within MATLAB.

To set up UART parameters, display the block parameter dialog box and select a tab:

- UART Configuration Block
- UART Tx Block
- UART Tx-MATLAB Block
- PICGUI Block

3.9.1 UART Configuration Block

The UART Configuration block configures the selected UART peripheral.

3.9.1.1 THE MAIN TAB

This tab contains configuration parameters like Baud Rate and the Rx and Tx pins, provided these can be remapped on the selected MCU.

The Current Baud value between [] is the exact baud rate reached with the current configuration. The baud list values show the % of error for each standard baud rate listed. The last choice, "Custom", allows defining a non-standard baud rate.

3.9.1.2 THE TX TAB

This tab contains transmitter implementation parameters. The implementation parameters provide four choice:

- None (4-byte internal buffer only)
- Circular Buffer
- DMA Ping-Pong Mode
- DMA Single Buffer (Possible data loss)

The DMA options will not be shown for a MCU that does not support DMA. For "DMA Ping-Pong Mode", two buffers are initialized and filled-in in sequence. The first buffer is sent when it is completely full. For "DMA Single Buffer", only one buffer is used. While this buffer is being transmitted, any new values that should be sent are lost.

3.9.2 UART Tx Block

The UART Tx block sends one byte or a vector of uint8 byte through the UART. This byte might be buffered depending on the configuration of the UART.

The UART Ref selected must be configured using a UART Configuration block. Multiple instances of this block can be used.

3.9.3 UART Tx-MATLAB Block

The UART Tx-MATLAB block allows sending variables to MATLAB through the UART. The protocol used allows sending up to 16 distinct channels, each channel being composed of data of type uint8, int8, uint16 or int16. Each channel can be sent using a different sampling rate. Data is typically sent continuously.

The MATLAB script (used in the PICGUI block) decodes the data that supports the hot plug feature. The overall system allows receiving data from within MATLAB and analyzing or plotting these data in pseudo real-time using MATLAB functions. The system is also useful to log data for performing simulation based on real logged data.

3.9.4 PICGUI Block

The PICGUI block opens the picgui interface. See Figure 2-4 for further details.

3.10 BUS SPI GROUP

The BUS SPI group contains only BUS SPI blocks.

3.10.1 The BUS SPI Block

The BUS SPI block configures and defines an SPI sequence to read and/or write data through the SPI peripheral. Several SPI BUS blocks could be used within the same model to define different sequences and could be executed conditionally, with different rates, or at startup only (Sample time set with “inf”). SPI generated code uses interrupts and, thus, the block execution is fast. This code only adds to the internal buffer values to be sent during the time step, and reads values received from the previous block execution (output values have a z^{-1} delay).

The Block User Interface contains two distinct configurations, namely “Static Peripheral Configuration” and “SPI BUS Parameters”, with actions defined afterward.

- Static Peripheral Configuration
- SPI BUS Parameters
- Action List

Figure 3-2 shows the SPI Block Interface configured to read raw values from accelerometers, a temperature sensor, and rate gyros from the sensor [InvenSense MPU6000](#), using its SPI interface. The test hardware is the [Asrovtch AUAV V3](#), where a dsPIC33EP512MU810 MCU is connected to the MPU6000 SPI bus through pins RE3 (SDI), RE2 (SDO), RG15 (SCK), and RE4 (Chip Select).

The MPU6000 sensor is configured once at start-up with another SPI block (not shown) with a sampling time of “inf”. The sensor sequentially writes the values [0x6A80 0x6B01 0x6A81], separated by a chip select and 60 ms wait delay for correct MPU6000 initialization. The block sequence shown here sends the MPU6000 memory address to start reading from:

$$\underbrace{0x3A}_{\text{address}} + \underbrace{0x80}_{\text{read}} = 0xBA$$

See MPU6000 documentation for further details.

The sensor reacts by sending one byte (with the value 0x00, which will be lost) followed by the requested seven 16-bit values, which are read. This block has three outputs, namely: A_{xyz} (size=3), T° (size=1), and G_{xyz} (size=3).

The outputs will be updated with values refreshed at a 1 kHz sample time. This example is for the MPU6000 sensor with an SPI interface, but each SPI sensor defines its own protocol.

FIGURE 3-2: SPI BLOCK INTERFACE

MCHP_BUS_SPI_GUI

SPI Driver

Static Peripheral Configuration

SPI Ref: ☐ use DMA

Mode: ☒ Master ☐ Slave

Interrupt Priority: SPI Rx/Tx

Pinout:

SDI:

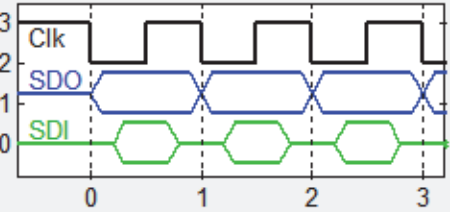
SDO:

SCK:

SPI BUS Parameters -- (can be update in the SPI sequence)

☒ Clock (SCK) Enabled bitrate: (+8.57%)

☒ SDO Enabled



Action	Port	Value	Comments
Set Pin to 0/1	E4 / P	1	/CS

Initial-Configuration

```

Set Port E4 / Pin[100] to 0    % /CS
Write only (x1)  Write:{[0xBA00]}    % read one byte
Read only (x3)   Read:{Axyz}    % Accelerometers
Read only (x1)   Read:{T°}    % Temperature
Read only (x3)   Read:{Gxyz}    % rate gyros
Set Port E4 / Pin[100] to 1    % /CS
  
```

Sample Time

Up Down Add X

Ok Apply Cancel

3.10.1.1 STATIC PERIPHERAL CONFIGURATION

These configurations cannot be changed during runtime.

TABLE 3-10: STATIC PERIPHERAL CONFIGURATION OPTIONS

Option	Description
SPI Ref	Specify the SPI reference.
use DMA	Check to implement SPI using DMA. Uncheck in implement SPI using interrupts. DMA should be preferred when more than 8 bytes are received/sent for an enhanced SPI peripheral, or more than 1 byte is received/sent for a standard SPI peripheral.
Mode: Master or Slave	Set the SPI peripheral as either Master or Slave.
Interrupt Priority: SPI Rx/Tx	Set the interrupt priority for SPI.
Pinout: SDI, SDO, SCK	Set the SPI SDI, SCK, and SS pins, unless the pins are not remappable. Then the choice is reduced to a list of 1 pin.

3.10.1.2 SPI BUS PARAMETERS

These parameters could be updated during runtime, allowing the addressing of several cases like:

- Configuration of an external sensor using a low-speed bit-rate to that of one using a high-speed bit-rate.
- Addressing several external chips using different SPI configurations (8 and 16 bits, SPI mode, etc).

Parameter updates are done within the Action list at the bottom of the user interface.

TABLE 3-11: SPI BUS PARAMETERS OPTIONS

Option	Description
Clock (SCK) Enabled	Check to enable the SPI clock on the SCK pin.
SDO Enabled	Check to enable the SPI output on the SDO pin.
Drop-down box 1	Set the number of bits per transfer (either 8 or 16 bits)
Drop-down box 2	Set the SPI mode
Drop-down box 3	Set SDI sampling
bit rate	Set the bit rate
<i>Signal Chart</i>	View the SPI signals

3.10.1.3 ACTION LIST

The Action List defines the SPI sequence sequentially. Available actions are:

- Initial-Configuration
- Update-Configuration
- Read & Write
- Read only
- Write only
- Set Pin to 0/1
- Delay
- Interrupt

The **Initial-Configuration action** is always the first action and cannot be removed. It represents the configuration set at model startup and is common to all SPI blocks with identical SPI Ref. The Update-Configuration action can be added anywhere in the sequence and allows updating the parameters within a sequence. Only parameters set under “SPI Bus Parameters” can be updated.

Changes made to items under “SPI Bus Parameters” are “attached” to the first “Configuration” action (either “Initial-Configuration” or “Update-Configuration”) found in the Action list above the current action selected. If BUS parameters are updated within a sequence, it is recommended that the parameters be switched to their initial values before the end of the sequence. Otherwise, another SPI block might start its sequence with the last parameters set and, thus, the expected initial-configuration parameters might be incorrect.

A **graph** showing signals Clk, SDI and SDO represents the SPI mode selected and the optional SDI sampling shift.

The **Read and/or Write action** reads and sends values through the SPI peripheral. Write sequences are non-blocking as they execute through interrupts in the background. Values could be a scalar or vector, constant (defined within the Action sequence) or a block input. Block outputs are values read in the background and started at the previous block execution, so the output value might have a maximum delay of z^{-1} . Block output values at first execution are not initialized.

The **Set Pin to 0/1 action** is typically used in SPI Master Mode to assert a chip select signal. Any pin could be used allowing the addressing of different slaves. Such a pin could also be used for debugging purpose.

The Delay action adds a delay in the execution of the sequence. For a slow slave, a tiny delay might be required after selecting the chip with a Chip Select signal. The delay value should be as small as possible as the delay will block the SPI interrupt for the selected amount of time and might have consequences for other pending interrupts.

The **Interrupt action** defines a user interrupt that could be used through the “User Interrupt block” to execute a Simulink subsystem asynchronously. This might be particularly useful when the SPI is used in slave mode, to reply quickly to a received request, or in SPI Master mode, to wait for sensor initialization. If the SPI block generating the interrupt is within the executed subsystem (i.e., in the interrupt), the interrupt must be placed at the very end of the sequence; otherwise the SPI block will not re-enable itself. This user interrupt could be forced at startup, allowing the design of a simple and responsive system (asynchronous).

3.11 BUS I²C™ GROUP

The Bus I²C group contains only Bus I²C Master blocks.

3.11.1 The BUS I²C Master Block

The “BUS I²C Master” block configures and defines an I²C sequence to read and/or write data through the I²C peripheral. Several “BUS I²C Master” blocks could be used within the same model to define different sequences that could execute conditionally, with different rates, or at start-up only (with Sample time set to “inf”). I²C generated code uses interrupts and, thus, the block execution is fast. The code only adds values to an internal buffer to be sent during the time step, and reads values received from the previous block execution (so output values have a z^{-1} delay).

The Block User Interface contains two distinct configurations, namely “Static Peripheral Configuration” and “I²C BUS Parameters”, with actions defined after.

- Static Peripheral Configuration
- I²C BUS Parameters
- Action List
- I²C Bus Timeout

Figure 3-3 shows the I²C Block Interface configured to read raw values from accelerometers, a temperature sensor, and rate gyros from the [InvenSense MPU6000](#) sensor using its I²C interface. The test hardware is the sensor board [Drotek IMU 10DOF V2](#) connected to an external dsPIC33EP12MC202 MCU through I²C pin RB8 (SCL) and RB9 (SDA).

An I²C sequence is sent at a ½ Hz (low) rate with another I²C block (not shown) that configures the sensor. Repeated initialization sequences support sensor “hot plug” during tests.

The block sequence shown here is for reading the 14 bytes of raw values that are sent sequentially by the three accelerometers, the temperature sensor and the three rate gyros in that order. The 7-bit I²C address for the MPU6000 (0x69) is written on the bus (Write Address) to start each read or write sequence. The first sequence writes one time (x1) the value 0x3B, which is the MPU memory address to be read. Then a read sequence is repeated 14 times (x14) to repetitively read consecutive memory values from the MPU6000. The block has one output vector containing 14 bytes of raw values consisting of three 16-bit accelerometers values, one 16-bit temperature value, and three 16-bit rate gyro values, updated at 1 kHz. Each 16 bit value must then be reconstructed using Simulink blocks. This example is for the MPU6000 sensor with I²C interface, but each I²C sensor defines its own protocol.

FIGURE 3-3: I²C BLOCK INTERFACE

The screenshot shows the **MCHP_BUS_I2C_MASTER_GUI** window. It contains the following sections:

- Static Peripheral Configuration:**
 - I2C Ref: 1
 - Interrupt Priority: 5 ...
 - Pinout:
 - SCL: B8 / Pin[17]
 - SDA: B9 / Pin[18]
- I2C BUS Parameters -- (can be update in the I2C sequence):**
 - ☒ Enable Slew rate bitrate: (-0.06%) 400000 399.77 KHz
 - ☒ Enable SMBus thresholds
- Action / Comments:**
 - Stop
- Initial-Configuration:**

```
---
Start
Write Address:{[0x69]}      % MPU 6000 addr with AD0 = 1
Write Data (x1)    Write:{[0x3B]}      % Addr Accel Xout
Stop
Start
Write Address:{[0x69]}      % MPU 6000 addr with AD0 = 1
Read Data (x14)    Read:{Datas}
Stop
```
- Sample Time:** [0.001]
- Buttons:** Up, Down, Add, X, Ok, Apply, Cancel

3.11.1.1 STATIC PERIPHERAL CONFIGURATION

These configurations cannot be changed during runtime.

TABLE 3-12: STATIC PERIPHERAL CONFIGURATION OPTIONS

Option	Description
I ² C Ref	Specify the I ² C reference
Interrupt Priority	Set the interrupt priority for I ² C
Pinout: SCL, SDA	Set the SCL and SDA pins, unless the pins are not remappable, and then the choice is reduced to a list of 1 pin

3.11.1.2 I²C BUS PARAMETERS

These parameters can be updated during runtime, allowing you to address cases like the configuration of an external sensor using a low-speed bit-rate to that of one using a high-speed bit-rate.

Parameter updates are done within the Action list at the bottom of the user GUI.

TABLE 3-13: I²C BUS PARAMETERS OPTIONS

Option	Description
Enable slew rate	Check to enable I ² C slew rate
Enable SMBus thresholds	Check to enable the SMBus thresholds
bit rate	Set the bit rate

3.11.1.3 ACTION LIST

The Action List define the I²C sequence sequentially. Available actions are:

- Initial-Configuration
- Update-Configuration
- Start
- Restart
- Stop
- Write Address
- Write Data
- Read Data
- Set Pin to 0/1 - Toggle
- Delay
- Interrupt

The **Initial-Configuration action** is always the first action and cannot be removed. It represents configurations set at the model startup and is common to all I²C blocks with identical I²C Ref. The Update-Configuration action can be added anywhere in the sequence and allows updating the parameters within a sequence. Only parameters set under “I²C BUS Parameters” can be updated.

Changes to the “I²C Bus Parameters” are attached to the first Configuration action (either “Initial-Configuration” or “Update-Configuration”) found in the Action list above the current action selected. If BUS parameters are updated within a sequence, it is recommended to switch the parameters to their initial values before the end of the sequence. Otherwise, another I²C block might start its sequence with last parameters set and thus the expected initial-configuration parameters could be incorrect.

The **Start, Re-Start and Stop action** sets one of these events on the I²C Bus.

The **Read and/or Write action** reads and sends values through the I²C peripheral. Write sequences are non-blocking as they execute through interrupts in the background. Values could be a scalar or a vector, a constant (defined within the Action sequence) or a block input. Block outputs are values read in the background and started at the previous block execution (Thus the output value might have a maximum delay of z^{-1} ; block output values at first execution are not initialized.).

The **Set Pin to 0/1 - Toggle action** is not required for data transfer (I²C bus does not require a Chip Select signal like the SPI bus) but could be useful for debugging purposes.

The **Delay action** adds a delay in the execution of the sequence. For a slow slave, a tiny delay might be required after selecting the chip with a Chip Select signal. Delay values should be as small as possible as the delay will block the I²C interrupt for the selected amount of time and might have collateral consequences for others pending interrupts.

The **Interrupt action** defines a user interrupt that could be used through the "User Interrupt block" to execute a Simulink subsystem asynchronously. If the I²C block generating the interrupt is within the executed subsystem (i.e. in the interrupt), the interrupt must be placed at the very end of the sequence; otherwise the SPI block will not re-enable itself. This user interrupt could be forced to execute once at startup, allowing the design of a simple and responsive system (asynchronous).

3.11.1.4 I²C BUS TIMEOUT

Even in Master mode, the I²C peripheral waits for a response from the addressed component. If no response is received, the I²C peripheral will be stuck waiting for a reply. As the I²C Bus is handled through interrupts as a background process, such a blocking situation will not stop the model from the execution of other tasks. Still, one glitch on the I²C lines jeopardize all I²C peripherals attached to the same bus. To avoid such a situation, the I²C bus is reset if, during three consecutive execution of one block, its previous sequence is still not finished. This "timeout" mechanism allows the reset of the I²C bus if a problem appears.

A similar problem also appears when designing the I²C sequence to address a specific chip. Any mistake in the design might make the MCU I²C peripheral wait for an I²C event from the external chip that never happens. In such a case, the "timeout" mechanism will reset the I²C bus and try again. Note that using the "Set Pin to 0/1 - Toggle" action might be very useful to debug in the design process.

3.12 USER FUNCTIONS GROUP

The User Function group contains only C Function blocks.

3.12.1 The C Function Block

The "C Function Call" block allows you to include your own C files within the Simulink project. These C files will only be used for code generation (i.e., not for simulation) and this allows you to use MCU-specific registers or instructions.

Your C file should be declared in the Simulink Configuration Parameters panel <CTRL+E> in the Code Generation menu, Custom Code submenu, and in the lower part: Include list of additional "Source files".

When the C Function block is opened, it will open additional files that have been declared and list all the functions available in the first pop-up menu. If the function is not properly detected, it is still possible to write the function directly into the Function Declaration box.

The Block Input and Block Output allow you to define which function parameters are input or output.

The return value from the function is always defined as a block output. Function parameters could also be block outputs which would only make sense if they are pointers (thus the function could modify their content).

Array size is defined using the `MyVar[xx]` syntax (instead of `MyVar`) in the block input/output parameters, with `xx` representing the number of elements in the array. Block input/output sizes reflect the size of the array.

Structure parameters will not be recognized, except if they have been declared in Simulink as a BUS. The declared BUS must match with the structure definition (same name and same content).

Figure 3-4 shows a simple function "MyMult" implementing multiplication. The two block inputs are the two function parameters (they are not pointers and cannot be a block output). The function output parameters are block outputs.

FIGURE 3-4: C FUNCTION CALL BLOCK INTERFACE

Function Block Parameters: C Function Call

C Function Call (mask) (link)

Function Pins Configuration

Functions found: unsigned int MyMult(unsigned int A; unsigned int B) ▼

Function Declaration:

unsigned int MyMult(unsigned int A, unsigned int B)

execute function: when block is updated (default) ▼

☐ Add the function declaration in header file

Block Input (coma separated function parameters)

A , B

Block Output (coma separated function parameters)

Block execution ordering: None ▼

Sample Time

-1

OK Cancel Help Apply

NOTES:

Index

B		Simulink Model Example.....	10
Baud Rate	18	Simulink Model Set Up	9
Bluetooth Module	16	Start/Stop	18
Buttons		T	
Delete Log	21	Timestamps.....	19
Load Raw Data	21	Tune Parameters	22
Log Data	21	U	
Rename Log	21	UART	16
Send1, Send2	21	UART Connection	18
Bytes Sent at Each Time Step	21		
C			
Code Generation Output Messages	13		
Colors	12		
Communications between MATLAB and MCU	16		
D			
Delete Button	21		
F			
FTDI chips.....	16		
G			
Generate Code	13		
L			
Library Content	25		
Limitations	8		
Load Button.....	21		
Log Data	16		
Log.txt Button	21		
M			
MATLAB Introduction	5		
MPLAB Device Blocks for Simulink Introduction.....	7		
O			
On the Fly Parameter Updates	22		
P			
padr Script.....	20		
Plot Data	16		
Program a Device	13		
Prolific Chips	16		
R			
Raw Data	18		
Ren Button	21		
S			
Sample Time Legend	12		
Send1 Button	21		
Send2 Button	21		
Simulink Introduction.....	7		

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
<http://www.microchip.com/support>
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110

Canada - Toronto
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hangzhou
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880-3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Dusseldorf
Tel: 49-2129-3766400

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Pforzheim
Tel: 49-7231-424750

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Venice
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Poland - Warsaw
Tel: 48-22-3325737

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820