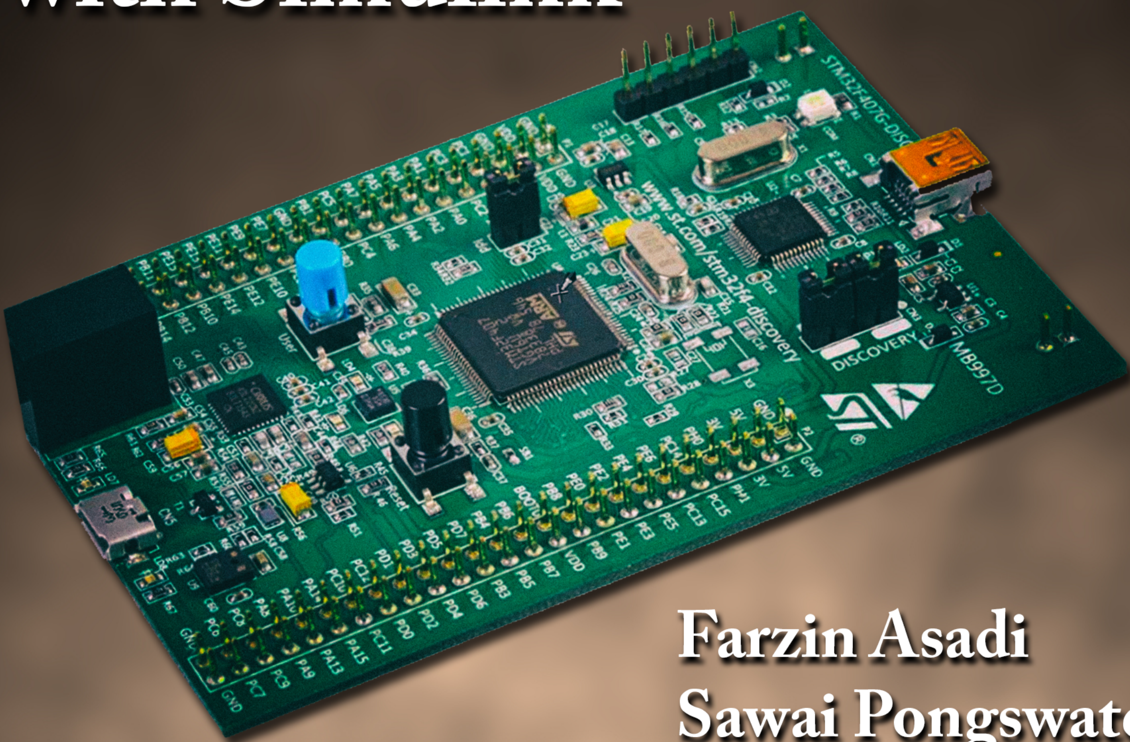
MORGAN & CLAYPOOL PUBLISHERS

Programming the ARM® Cortex®-M4-based STM32F4 Microcontrollers with Simulink®



Farzin Asadi
Sawai Pongswatd

Synthesis Lectures on Digital Circuits and Systems

Mitchell A. Thornton, *Series Editor*

Programming the ARM® Cortex®-M4-based STM32F4 Microcontrollers with Simulink®

Synthesis Lectures on Digital Circuits and Systems

Editor
Mitchell A. Thornton, ***Southern Methodist University***

The *Synthesis Lectures on Digital Circuits and Systems* series is comprised of 50- to 100-page books targeted for audience members with a wide-ranging background. The Lectures include topics that are of interest to students, professionals, and researchers in the area of design and analysis of digital circuits and systems. Each Lecture is self-contained and focuses on the background information required to understand the subject matter and practical case studies that illustrate applications. The format of a Lecture is structured such that each will be devoted to a specific topic in digital circuits and systems rather than a larger overview of several topics such as that found in a comprehensive handbook. The Lectures cover both well-established areas as well as newly developed or emerging material in digital circuits and systems design and analysis.

Programming the ARM® Cortex®-M4-based STM32F4 Microcontrollers with Simulink®
Farzin Asadi and Sawai Pongswatd
2021

Arduino III: Internet of Things
Steven F. Barrett
2021

Arduino II: Systems
Steven F. Barrett
2020

Arduino I: Getting Started
Steven F. Barrett
2020

Index Generation Functions
Tsutomu Sasao
2019

Microchip AVR® Microcontroller Primer: Programming and Interfacing, Third Edition
Steven F. Barrett and Daniel J. Pack
2019

Microcontroller Programming and Interfacing with Texas Instruments MSP430FR2433 and MSP430FR5994 – Part II, Second Edition
Steven F. Barrett and Daniel J. Pack
2019

Microcontroller Programming and Interfacing with Texas Instruments MSP430FR2433 and MSP430FR5994 – Part I, Second Edition
Steven F. Barrett and Daniel J. Pack
2019

Synthesis of Quantum Circuits vs. Synthesis of Classical Reversible Circuits
Alexis De Vos, Stijn De Baerdemacker, and Yvan Van Rentergen
2018

Boolean Differential Calculus
Bernd Steinbach and Christian Posthoff
2017

Embedded Systems Design with Texas Instruments MSP432 32-bit Processor
Dung Dang, Daniel J. Pack, and Steven F. Barrett
2016

Fundamentals of Electronics: Book 4 Oscillators and Advanced Electronics Topics
Thomas F. Schubert and Ernest M. Kim
2016

Fundamentals of Electronics: Book 3 Active Filters and Amplifier Frequency
Thomas F. Schubert and Ernest M. Kim
2016

Bad to the Bone: Crafting Electronic Systems with BeagleBone and BeagleBone Black, Second Edition
Steven F. Barrett and Jason Kridner
2015

Fundamentals of Electronics: Book 2 Amplifiers: Analysis and Design
Thomas F. Schubert and Ernest M. Kim
2015

Fundamentals of Electronics: Book 1 Electronic Devices and Circuit Applications
Thomas F. Schubert and Ernest M. Kim
2015

Applications of Zero-Suppressed Decision Diagrams
Tsutomu Sasao and Jon T. Butler
2014

Modeling Digital Switching Circuits with Linear Algebra
Mitchell A. Thornton
2014

Arduino Microcontroller Processing for Everyone! Third Edition
Steven F. Barrett
2013

Boolean Differential Equations
Bernd Steinbach and Christian Posthoff
2013

Bad to the Bone: Crafting Electronic Systems with BeagleBone and BeagleBone Black
Steven F. Barrett and Jason Kridner
2013

Introduction to Noise-Resilient Computing
S.N. Yanushkevich, S. Kasai, G. Tangim, A.H. Tran, T. Mohamed, and V.P. Shmerko
2013

Atmel AVR Microcontroller Primer: Programming and Interfacing, Second Edition
Steven F. Barrett and Daniel J. Pack
2012

Representation of Multiple-Valued Logic Functions
Radomir S. Stankovic, Jaakko T. Astola, and Claudio Moraga
2012

Arduino Microcontroller: Processing for Everyone! Second Edition
Steven F. Barrett
2012

Advanced Circuit Simulation Using Multisim Workbench
David Báez-López, Félix E. Guerrero-Castro, and Ofelia Delfina Cervantes-Villagómez
2012

Circuit Analysis with Multisim
David Báez-López and Félix E. Guerrero-Castro
2011

Microcontroller Programming and Interfacing Texas Instruments MSP430, Part I
Steven F. Barrett and Daniel J. Pack
2011

Microcontroller Programming and Interfacing Texas Instruments MSP430, Part II
Steven F. Barrett and Daniel J. Pack
2011

Pragmatic Electrical Engineering: Systems and Instruments
William Eccles
2011

Pragmatic Electrical Engineering: Fundamentals
William Eccles
2011

Introduction to Embedded Systems: Using ANSI C and the Arduino Development Environment
David J. Russell
2010

Arduino Microcontroller: Processing for Everyone! Part II
Steven F. Barrett
2010

Arduino Microcontroller Processing for Everyone! Part I
Steven F. Barrett
2010

Digital System Verification: A Combined Formal Methods and Simulation Framework
Lun Li and Mitchell A. Thornton
2010

Progress in Applications of Boolean Functions
Tsutomu Sasao and Jon T. Butler
2009

Embedded Systems Design with the Atmel AVR Microcontroller: Part II
Steven F. Barrett
2009

Embedded Systems Design with the Atmel AVR Microcontroller: Part I
Steven F. Barrett
2009

Embedded Systems Interfacing for Engineers using the Freescale HCS08 Microcontroller II: Digital and Analog Hardware Interfacing
Douglas H. Summerville
2009

Designing Asynchronous Circuits using NULL Convention Logic (NCL)
Scott C. Smith and JiaDi
2009

Embedded Systems Interfacing for Engineers using the Freescale HCS08 Microcontroller I: Assembly Language Programming
Douglas H.Summerville
2009

Developing Embedded Software using DaVinci & OMAP Technology
B.I. (Raj) Pawate
2009

Mismatch and Noise in Modern IC Processes
Andrew Marshall
2009

Asynchronous Sequential Machine Design and Analysis: A Comprehensive Development of the Design and Analysis of Clock-Independent State Machines and Systems
Richard F. Tinder
2009

An Introduction to Logic Circuit Testing
Parag K. Lala
2008

Pragmatic Power
William J. Eccles
2008

Multiple Valued Logic: Concepts and Representations
D. Michael Miller and Mitchell A. Thornton
2007

Finite State Machine Datapath Design, Optimization, and Implementation
Justin Davis and Robert Reese
2007

Atmel AVR Microcontroller Primer: Programming and Interfacing
Steven F. Barrett and Daniel J. Pack
2007

Pragmatic Logic
William J. Eccles
2007

PSpice for Filters and Transmission Lines
Paul Tobin
2007

PSpice for Digital Signal Processing
Paul Tobin
2007

PSpice for Analog Communications Engineering
Paul Tobin
2007

PSpice for Digital Communications Engineering
Paul Tobin
2007

PSpice for Circuit Theory and Electronic Devices
Paul Tobin
2007

Pragmatic Circuits: DC and Time Domain
William J. Eccles
2006

Pragmatic Circuits: Frequency Domain
William J. Eccles
2006

Pragmatic Circuits: Signals and Filters
William J. Eccles
2006

High-Speed Digital System Design
Justin Davis
2006

Introduction to Logic Synthesis using Verilog HDL
Robert B.Reese and Mitchell A.Thornton
2006

Microcontrollers Fundamentals for Engineers and Scientists
Steven F. Barrett and Daniel J. Pack
2006

Programming the ARM® Cortex®-M4-based STM32F4 Microcontrollers with Simulink®

Farzin Asadi and Sawai Pongswatd

www.morganclaypool.com

ISBN: 9781636392448 paperback
ISBN: 9781636392455 ebook
ISBN: 9781636392462 hardcover

DOI 10.2200/S01128ED1V01Y202109DCS061

A Publication in the Morgan & Claypool Publishers series
SYNTHESIS LECTURES ON DIGITAL CIRCUITS AND SYSTEMS

Lecture #61
Series Editor: Mitchell A. Thornton, *Southern Methodist University*
Series ISSN
Print 1932-3166 Electronic 1932-3174

Programming the ARM® Cortex®-M4-based STM32F4 Microcontrollers with Simulink®

Farzin Asadi
Maltepe University, Istanbul, Turkey

Sawai Pongswatd
King Mongkut's Institute of Technology, Ladkrabang, Thailand

SYNTHESIS LECTURES ON DIGITAL CIRCUITS AND SYSTEMS #61

MORGAN & CLAYPOOL PUBLISHERS

ABSTRACT

A microcontroller is a compact, integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory, and input/output (I/O) peripherals on a single chip.

When they first became available, microcontrollers solely used Assembly language. Today, the C programming language (and some other high-level languages) can be used as well. Some of advanced microcontrollers support another programming technique as well: Graphical programming. In graphical programming, the user does not write any code but draws the block diagram of the system he wants. Then a software converts the drawn block diagram into a suitable code for the target device.

Programming microcontrollers using graphical programming is quite easier than programming in C or Assembly. You can implement a complex system within hours with graphical programming while its implementation in C may take months. These features make the graphical programming an important option for engineers.

This book study the graphical programming of STM32F4 high-performance microcontrollers with the aid of Simulink® and Waijung blockset. Students of engineering (for instance, electrical, biomedical, mechatronics and robotic to name a few), engineers who work in industry, and anyone who want to learn the graphical programming of STM32F4 can benefit from this book. Prerequisite for this book is the basic knowledge of MATLABi®/Simulink®.

KEYWORDS

ARM Cortex, graphical programming, microcontroller, Simulink, STM32F4, Waijung blockset

Contents

Preface

A microcontroller is a compact, integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory, and input/output (I/O) peripherals on a single chip.

Sometimes referred to as an embedded controller or microcontroller unit (MCU), microcontrollers are found in vehicles, robots, office machines, medical devices, mobile radio transceivers, vending machines, and home appliances, among other devices. They are essentially simple miniature personal computers (PCs) designed to control small features of a larger component, without a complex front-end operating system (OS).

When they first became available, microcontrollers solely used Assembly language. Today, the C programming language (and some other high-level languages) can be used as well. Some of advanced microcontrollers support another programming technique as well: graphical programming. In graphical programming, the user does not write any code but draws the block diagram of the system he wants. Then a software converts the drawn block diagram into a suitable code for the target device.

Programming microcontrollers using graphical programming is quite easier than programming in C or Assembly. You can implement a complex system within hours with graphical programming while its implementation in C may take months. These features make the graphical programming an important option for engineers.

This book studies the graphical programming of STM32F4 high-performance microcontrollers with the aid of Simulink® and Waijung blockset. Students of engineering (for instance, electrical, biomedical, mechatronics and robotic to name a few), engineers who work in industry, and anyone who want to learn the graphical programming of STM32F4 can benefit from this book. Prerequisite for this book is the basic knowledge of MATLAB®/Simulink®.

We hope that this book will be useful to the readers, and we welcome comments on the book.

Farzin Asadi (farzinasadi@maltepe.edu.tr)
Sawai Pongswatd (sawai.po@kmitl.ac.th)
October 2021

CHAPTER 1

Basics of Simulink®

1.1 INTRODUCTION

Simulink is a software package for modeling, simulating, and analyzing dynamic systems. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two. This chapter shows the basics of simulation with Simulink®. If you are familiar with Simulink environment, you can start from Chapter 2.

1.2 EXAMPLE 1: STEP RESPONSE OF A TRANSFER FUNCTION MODEL

In this example, a transfer function is stimulated with unit step signal and its response is observed. Enter to the Simulink environment with the aid of the simulink command (Fig. 1.1).

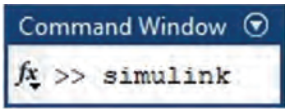


Figure 1.1

The Simulink Start Page window appeared. Click the Blank Model (Fig. 1.2). Now the Simulink environment with a blank project is ready (Fig. 1.3). Click the Library Browser button (Fig. 1.4). After clicking the Library Browser icon, the Simulink Library Browser window (Fig. 1.5) will be opened and you can add required components to the model.

Simulink Library browser contains many blocks and it is impossible to memorize each block locations. The Enter search term box is useful to find a block when you forgot its location. For instance, assume that you need a PID controller block but you don't know where it is. In this case, just type pid in the Enter search term box (Fig. 1.6) and press the Enter key of your keyboard. After pressing the Enter key, Simulink will list blocks related to the entered term in the right side of the window.

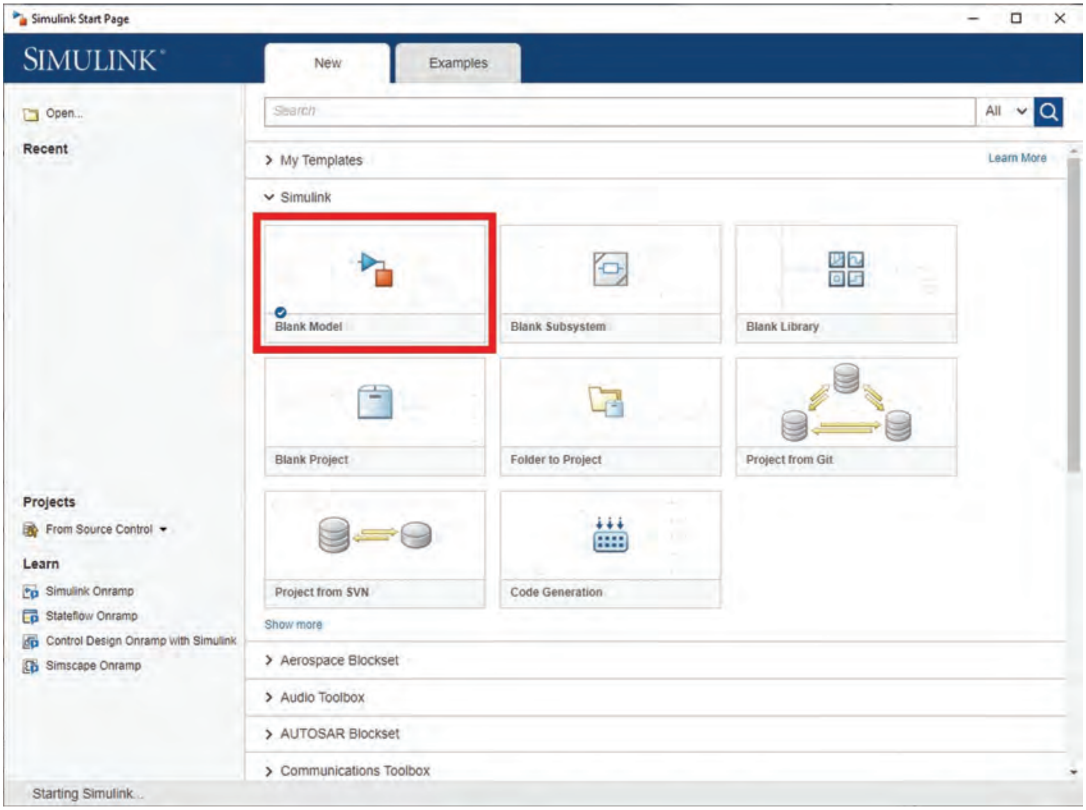

Simulink Start Page
SIMULINK
New
Examples
Open...
Recent
Search
All
My Templates
Learn More
Simulink
Blank Model
Blank Subsystem
Blank Library
Blank Project
Folder to Project
Project from Git
Project from SVN
Code Generation
Show more
Projects
From Source Control
Learn
Simulink Onramp
Stateflow Onramp
Control Design Onramp with Simulink
Simscape Onramp
Aerospace Blockset
Audio Toolbox
AUTOSAR Blockset
Communications Toolbox
Starting Simulink...

Figure 1.2

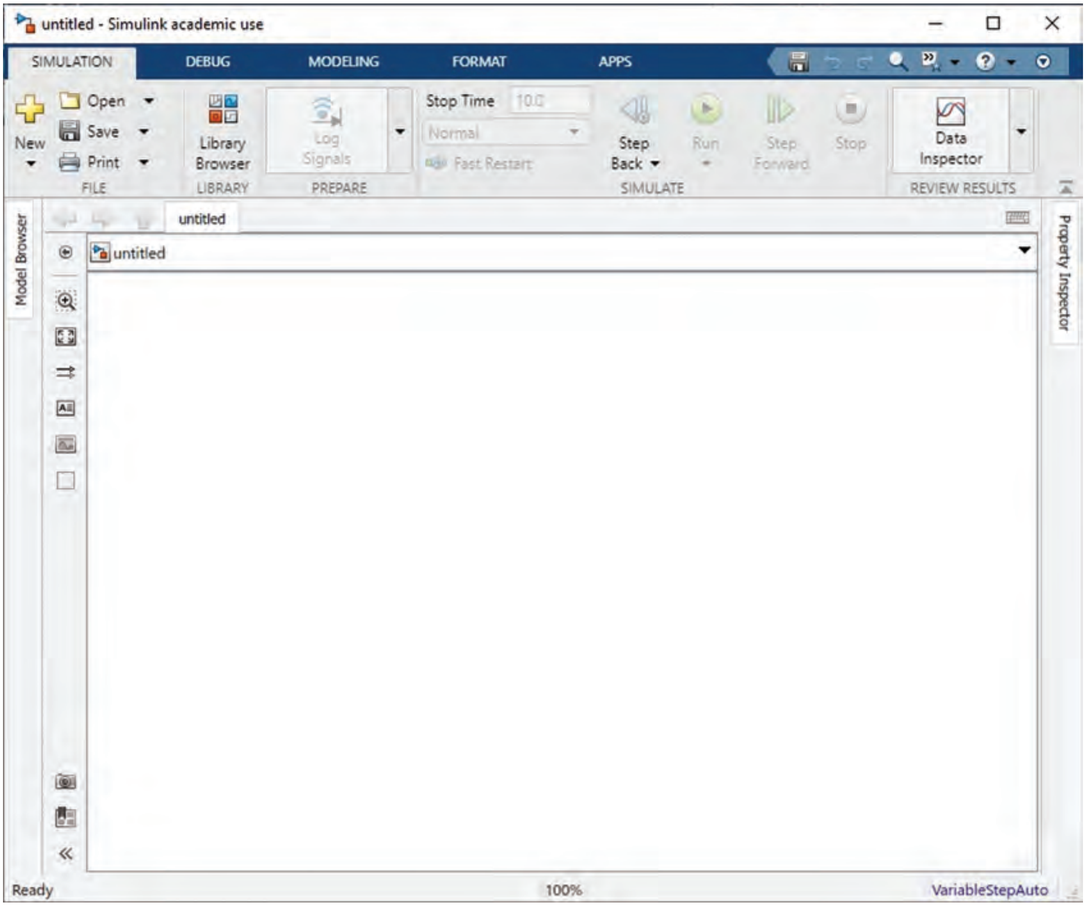


Figure 1.3

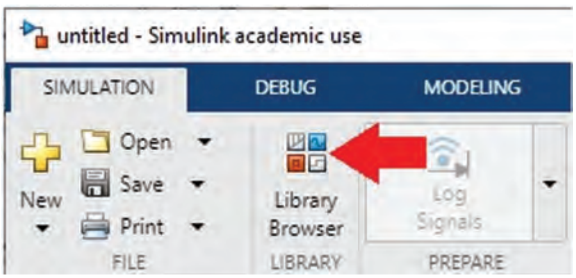


Figure 1.4

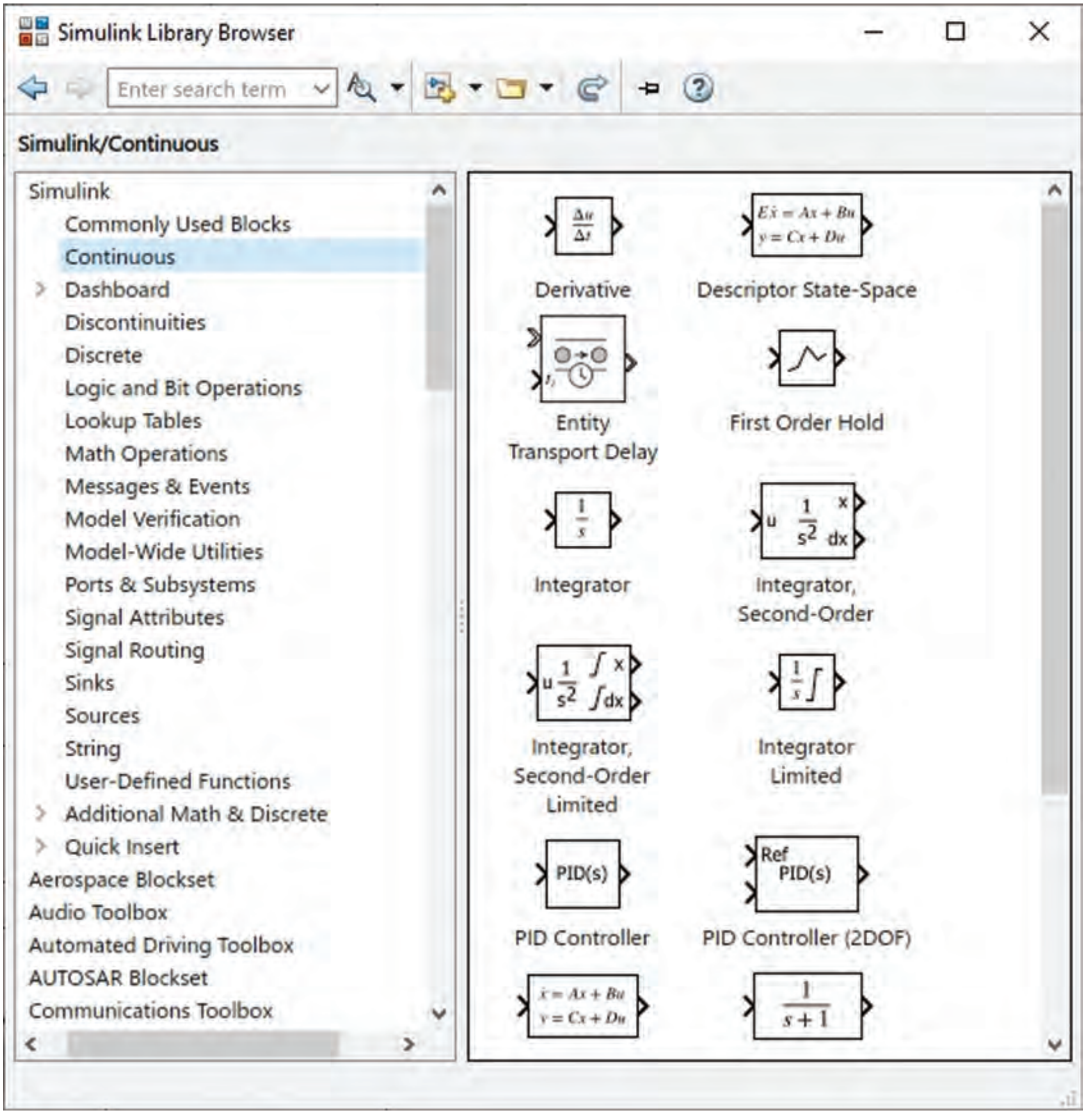

Simulink Library Browser
Enter search term
Simulink/Continuous
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Messages & Events
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
String
User-Defined Functions
Additional Math & Discrete
Quick Insert
Aerospace Blockset
Audio Toolbox
Automated Driving Toolbox
AUTOSAR Blockset
Communications Toolbox
Derivative
Descriptor State-Space
Entity Transport Delay
First Order Hold
Integrator
Integrator, Second-Order
Integrator, Second-Order Limited
Integrator Limited
PID(s)
PID Controller
Ref
PID(s)
PID Controller (2DOF)

Figure 1.5

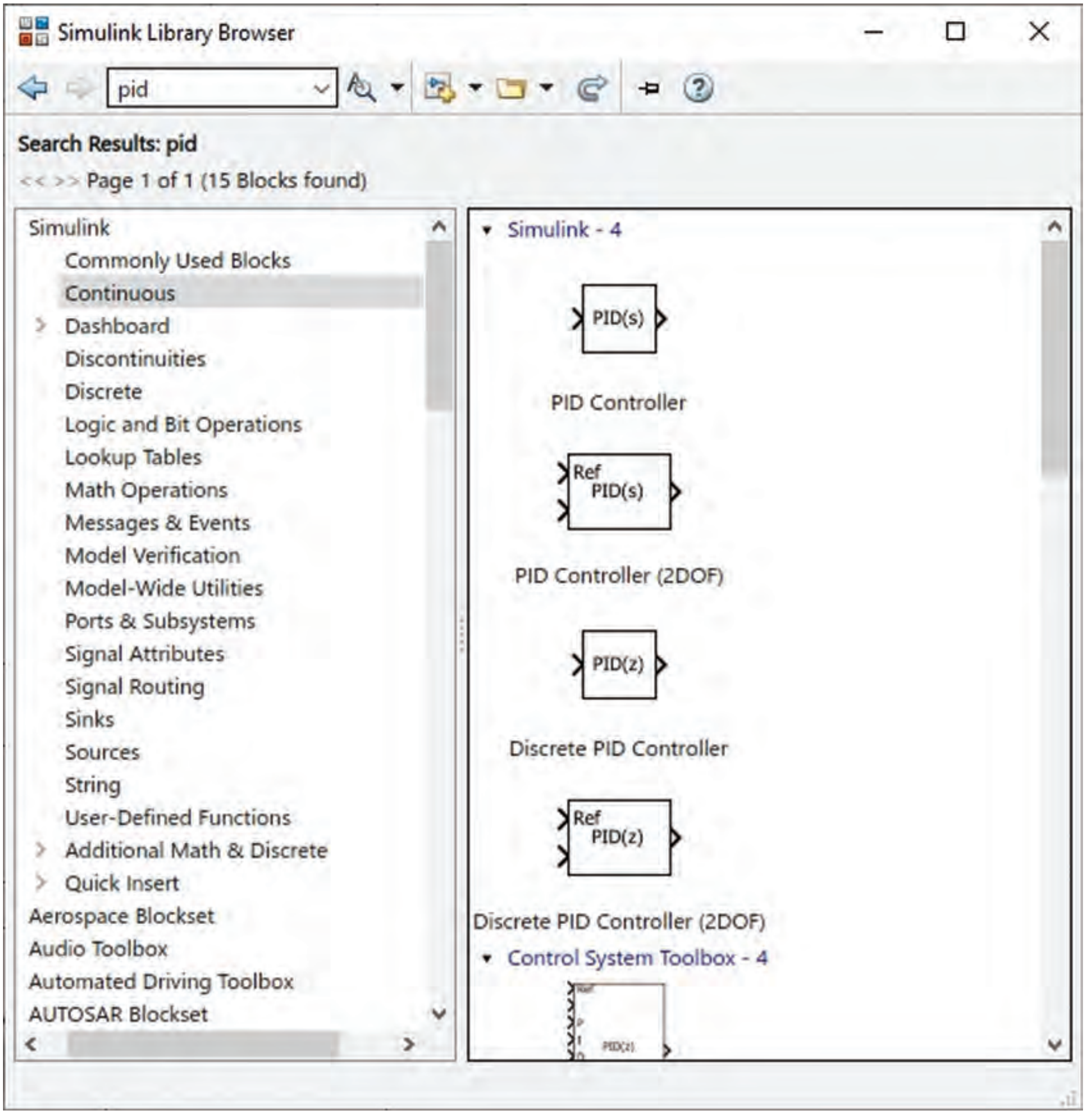

Simulink Library Browser
pid
Search Results: pid
<< >> Page 1 of 1 (15 Blocks found)
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Messages & Events
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
String
User-Defined Functions
Additional Math & Discrete
Quick Insert
Aerospace Blockset
Audio Toolbox
Automated Driving Toolbox
AUTOSAR Blockset
Simulink - 4
PID(s)
PID Controller
Ref
PID(s)
PID Controller (2DOF)
PID(z)
Discrete PID Controller
Ref
PID(z)
Discrete PID Controller (2DOF)
Control System Toolbox - 4

Figure 1.6

The Transfer Fcn block can be found in the Continuous section of Simulink Library Browser (Fig. 1.7). Click on the Transfer Fcn block to select it, then drag and drop it to the model (Fig. 1.8).

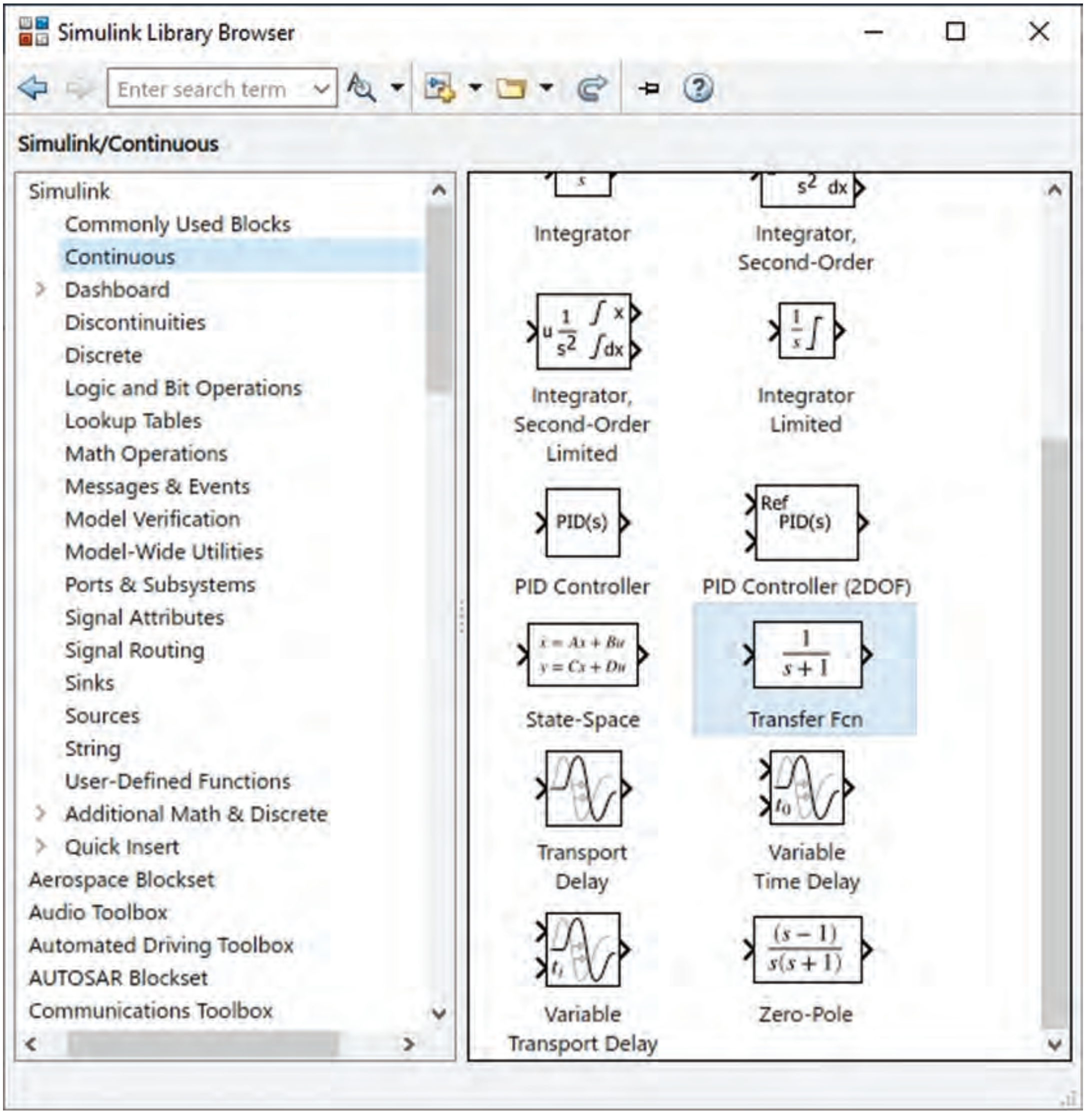


Figure 1.7

Add a Step block (Fig. 1.9) to the model (Fig. 1.10). Add a Scope block (Fig. 1.11) to the model (Fig. 1.12).

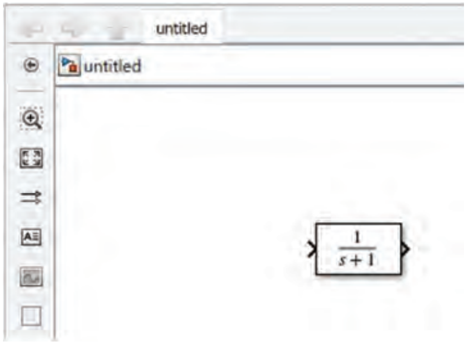


Figure 1.8

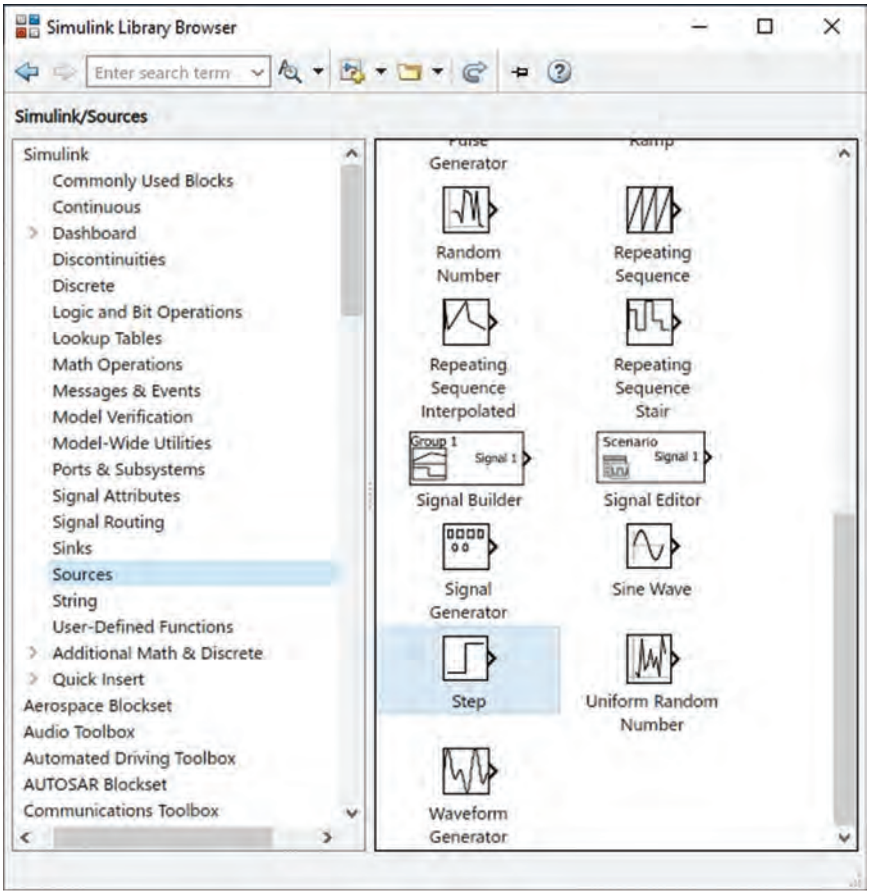


Figure 1.9

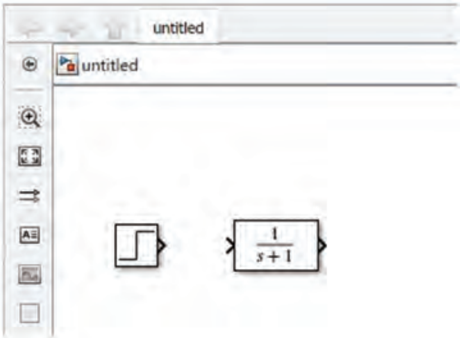


Figure 1.10

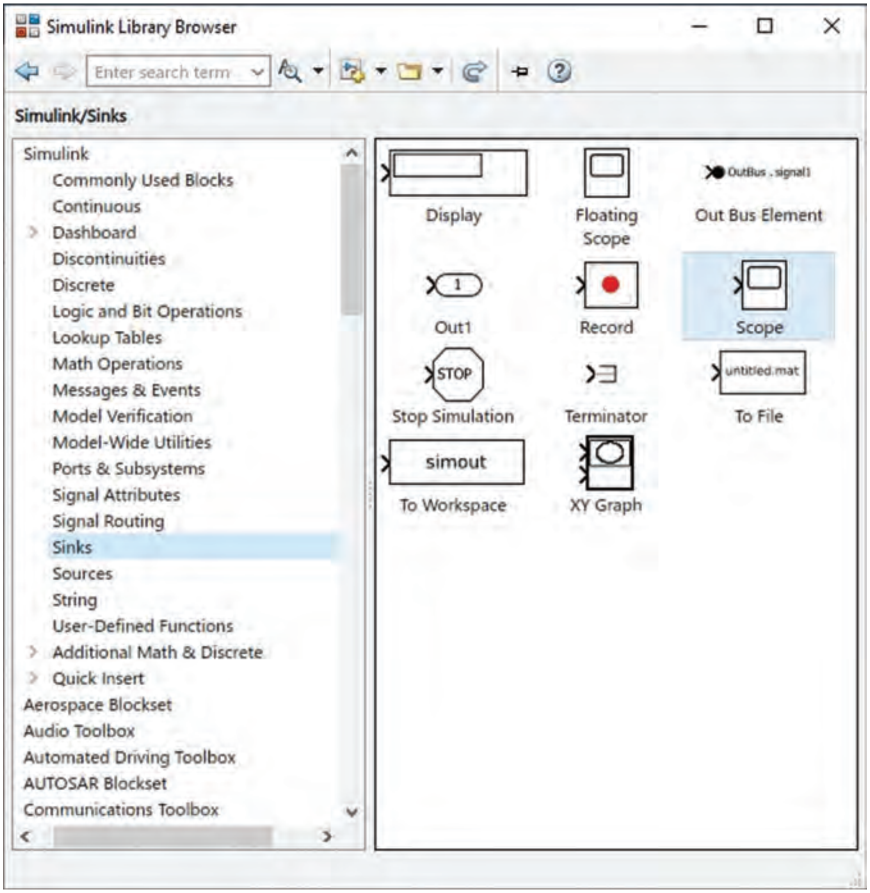


Figure 1.11

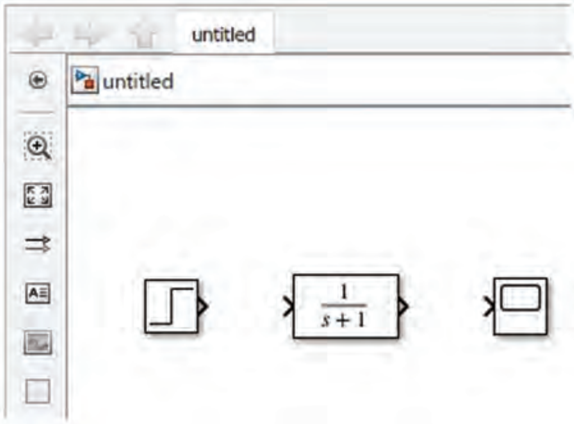


Figure 1.12

When you bring the mouse pointer close to the blocks terminals, it will be changed to crosshair and permit you to start connecting them. After seeing the crosshair, hold down the left mouse key and drag the connection toward the destination terminal and release the left mouse button on the destination terminal. Use this method to connect the blocks together.

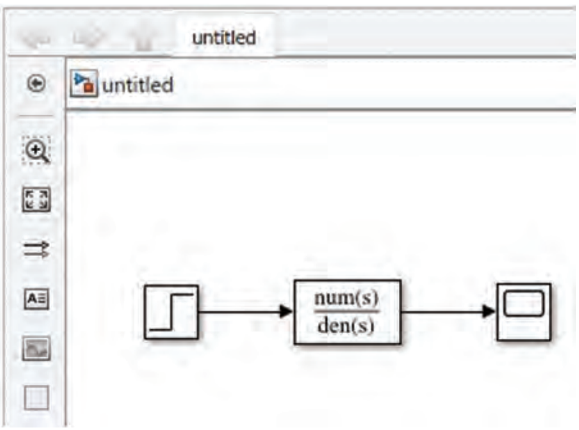


Figure 1.13

Double click the blocks and do their settings similar to what is shown in Figs. 1.14 and 1.15. Settings of Fig. 1.14 generate a pulse which jumps from 0 to 1 at $t = 0$. Settings of Fig. 1.15 simulate the $\frac{100}{s^2+8s+100}$ transfer function.

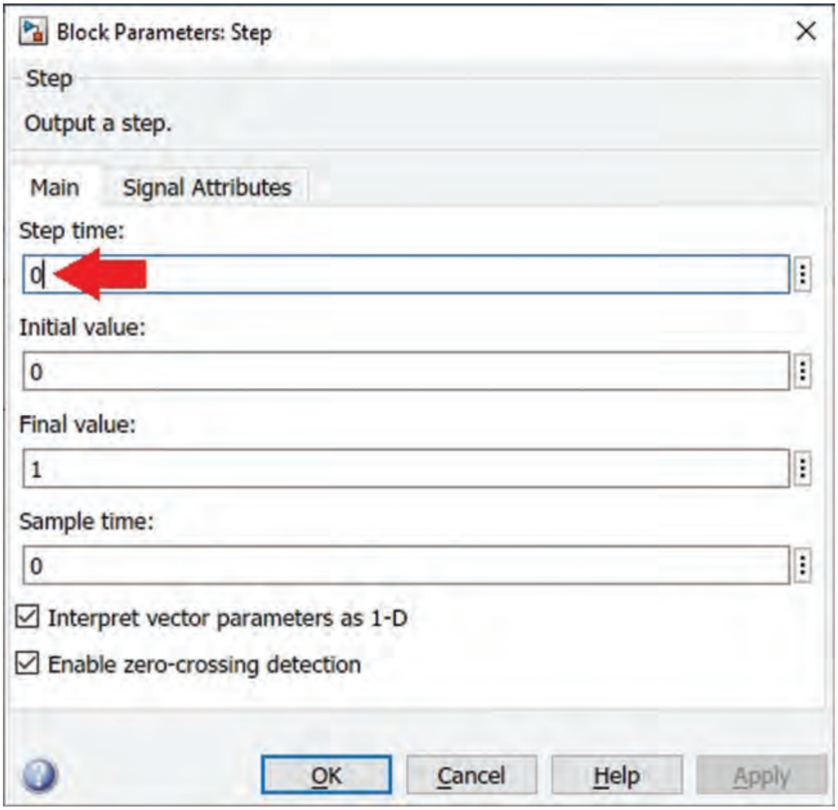

Block Parameters: Step
Step
Output a step.
Main
Signal Attributes
Step time:
0
Initial value:
0
Final value:
1
Sample time:
0
Interpret vector parameters as 1-D
Enable zero-crossing detection
OK
Cancel
Help
Apply

Figure 1.14

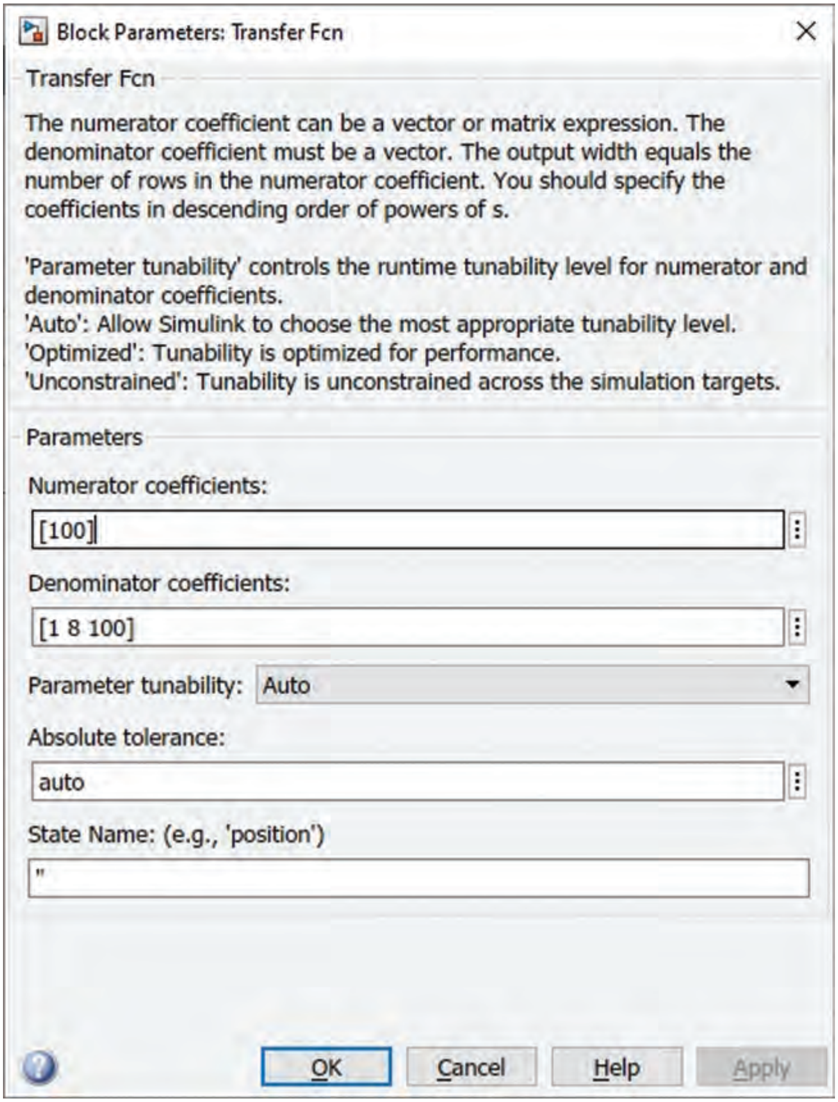

Block Parameters: Transfer Fcn
Transfer Fcn
The numerator coefficient can be a vector or matrix expression. The denominator coefficient must be a vector. The output width equals the number of rows in the numerator coefficient. You should specify the coefficients in descending order of powers of s.
'Parameter tunability' controls the runtime tunability level for numerator and denominator coefficients.
'Auto': Allow Simulink to choose the most appropriate tunability level.
'Optimized': Tunability is optimized for performance.
'Unconstrained': Tunability is unconstrained across the simulation targets.
Parameters
Numerator coefficients:
[100]
Denominator coefficients:
[1 8 100]
Parameter tunability:
Auto
Absolute tolerance:
auto
State Name: (e.g., 'position')
''
OK
Cancel
Help
Apply

Figure 1.15

Assume that you want to simulate the behavior of system for time length of 2 s. Enter 2 to the Stop Time box and click the Run button (or press the Ctrl+T) to simulate the behavior of the system (Fig. 1.16). Sometimes you need to do the simulation with a specific solver. In these cases, use the Model Settings (Fig. 1.17) to select the desired solver. After clicking the Model Solver icon (or pressing the Ctrl+E), the window shown in Fig. 1.18 appears and you can select the desired type of solver.

Figure 1.16

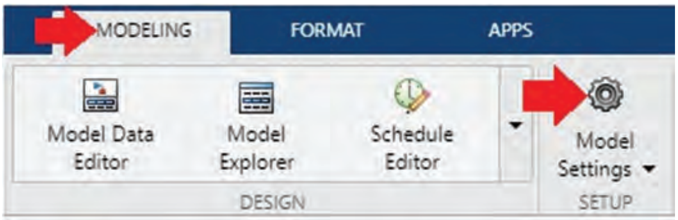


Figure 1.17

The simulation result is shown in Fig. 1.19. The simulation result can be copied into the clipboard by pressing Ctrl+C. You can paste the copied waveform in other software by pressing the Ctrl+V. This is very useful when you want to prepare a presentation or report.

You can use the Cursor Measurement (Fig. 1.20) to read the coordinate of different points of the graph. After clicking the Cursor Measurement, two vertical lines will be added to the graph (Fig. 1.21). You can move them to read the coordinate of different points of the graph.

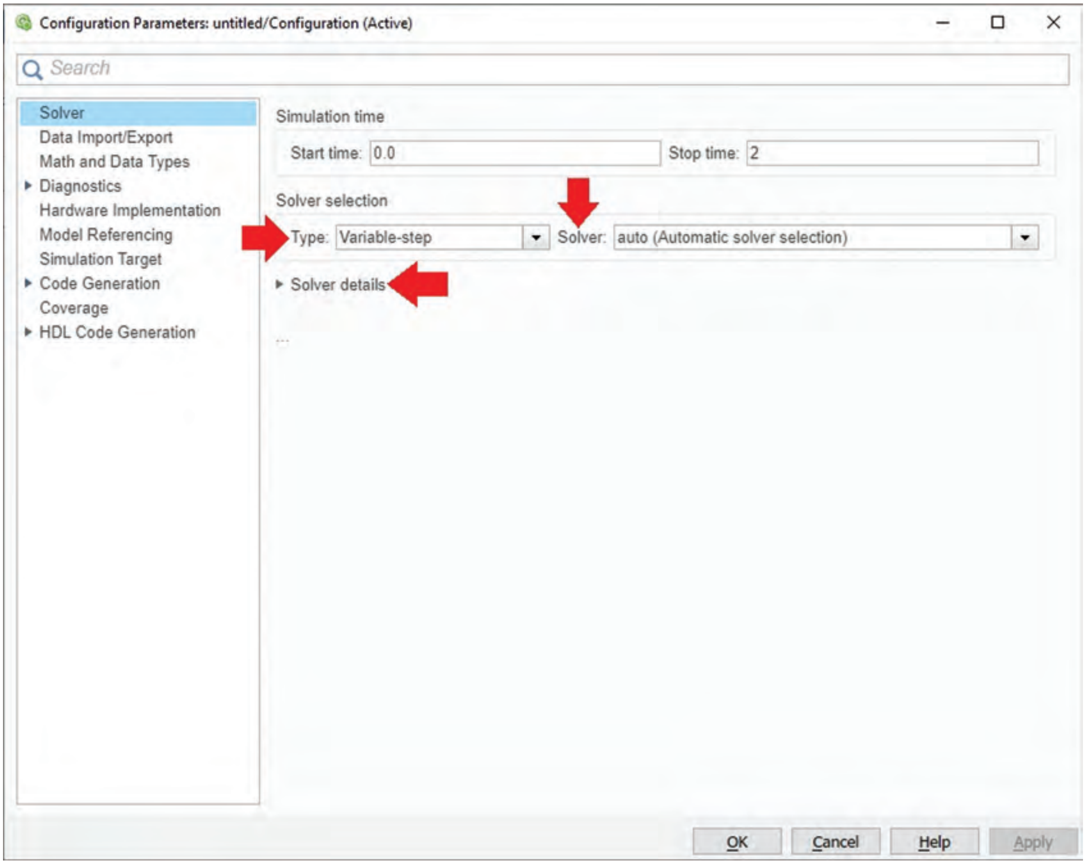

Configuration Parameters: untitled/Configuration (Active)
Search
Solver
Data Import/Export
Math and Data Types
Diagnostics
Hardware Implementation
Model Referencing
Simulation Target
Code Generation
Coverage
HDL Code Generation
Simulation time
Start time: 0.0
Stop time: 2
Solver selection
Type: Variable-step
Solver: auto (Automatic solver selection)
Solver details
OK
Cancel
Help
Apply

Figure 1.18

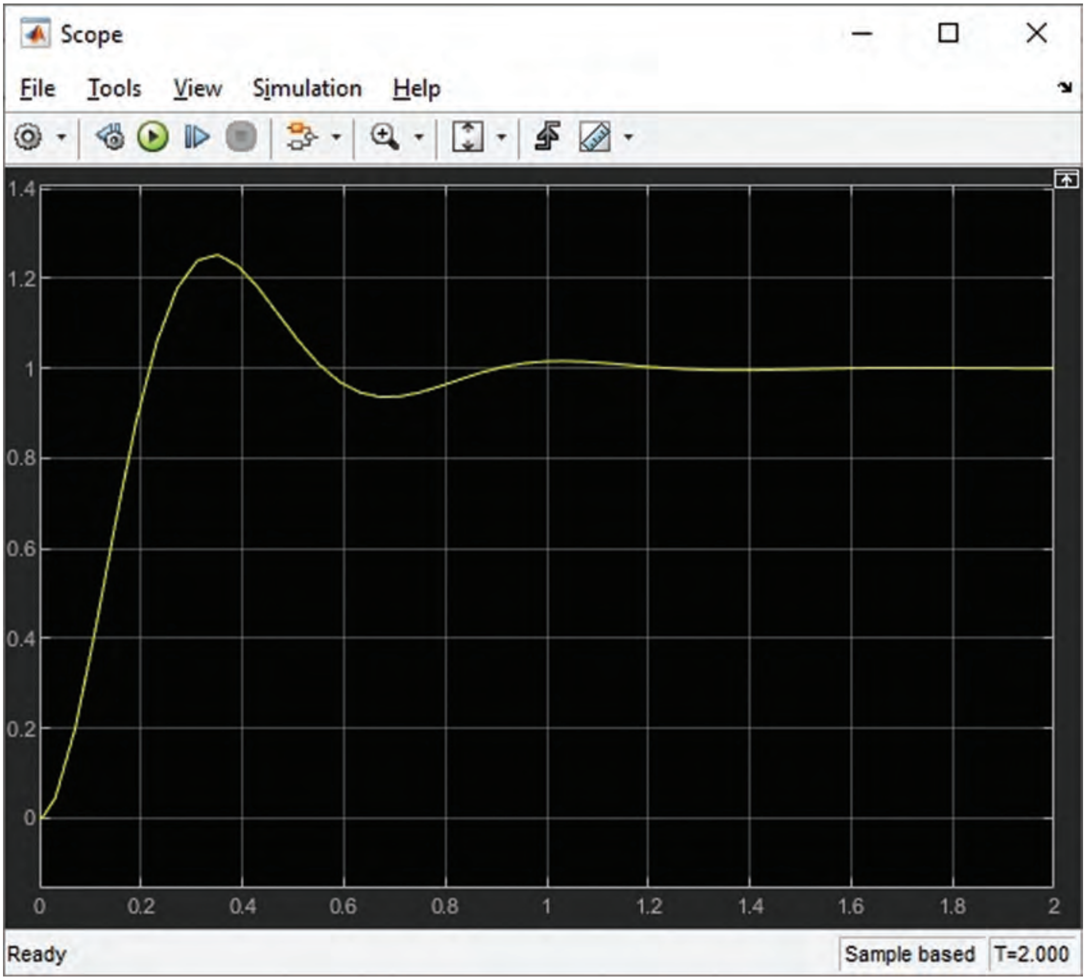

Scope
File
Tools
View
Simulation
Help
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Ready
Sample based
T=2.000

Figure 1.19

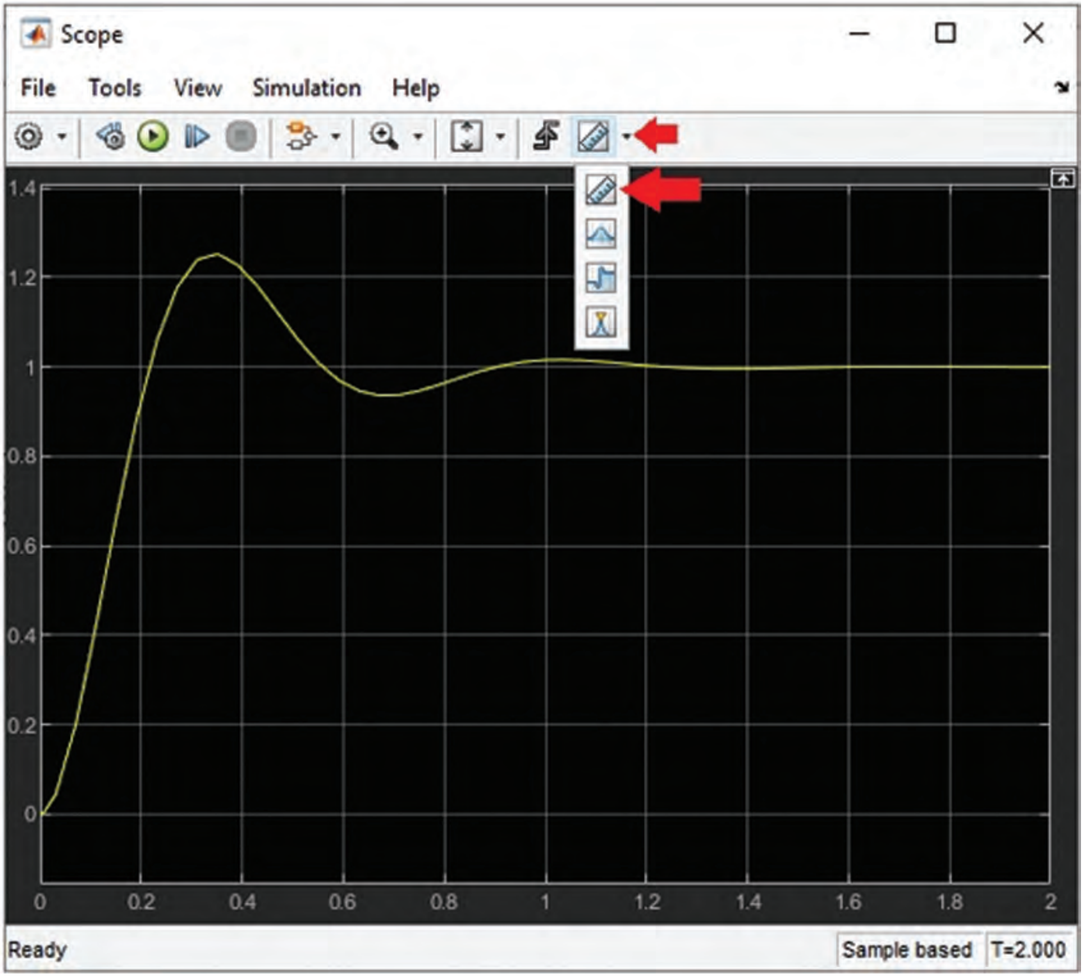

Scope
File Tools View Simulation Help
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2
Ready
Sample based
T=2.000

Figure 1.20

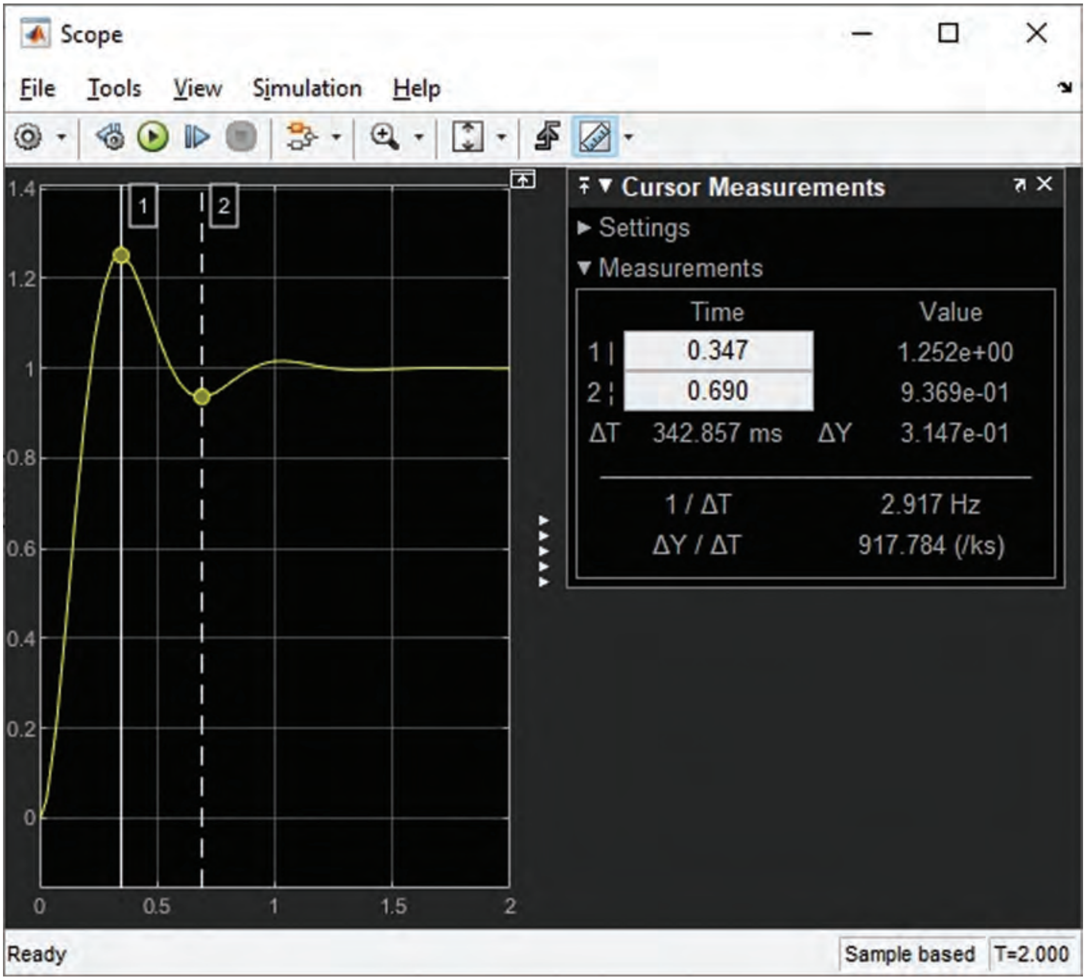


Figure 1.21

1.3 EXAMPLE 2: PID CONTROLLER DESIGN IN MATLAB ENVIRONMENT

Let's design a PID controller for the transfer function of the previous example. The command shown in Fig. 1.22 enters the transfer function to the MATLAB environment.

The pidTuner command (Fig. 1.23) helps you to tune the PID controller. After running the pidTuner command, the window shown in Fig. 1.24 appears.

Move the sliders until you obtain a good response. By default, the PID tuner does the tuning in the time domain (Fig. 1.25). You can do it in the frequency domain, as well (Fig. 1.26).

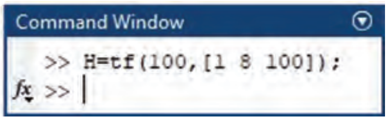


Figure 1.22: Entering the $H(s) = \frac{100}{s^2+8s+100}$ to MATLAB.

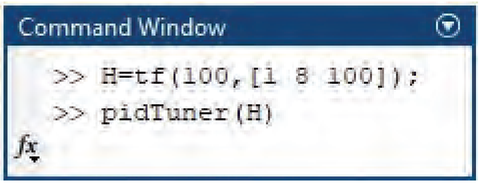


Figure 1.23

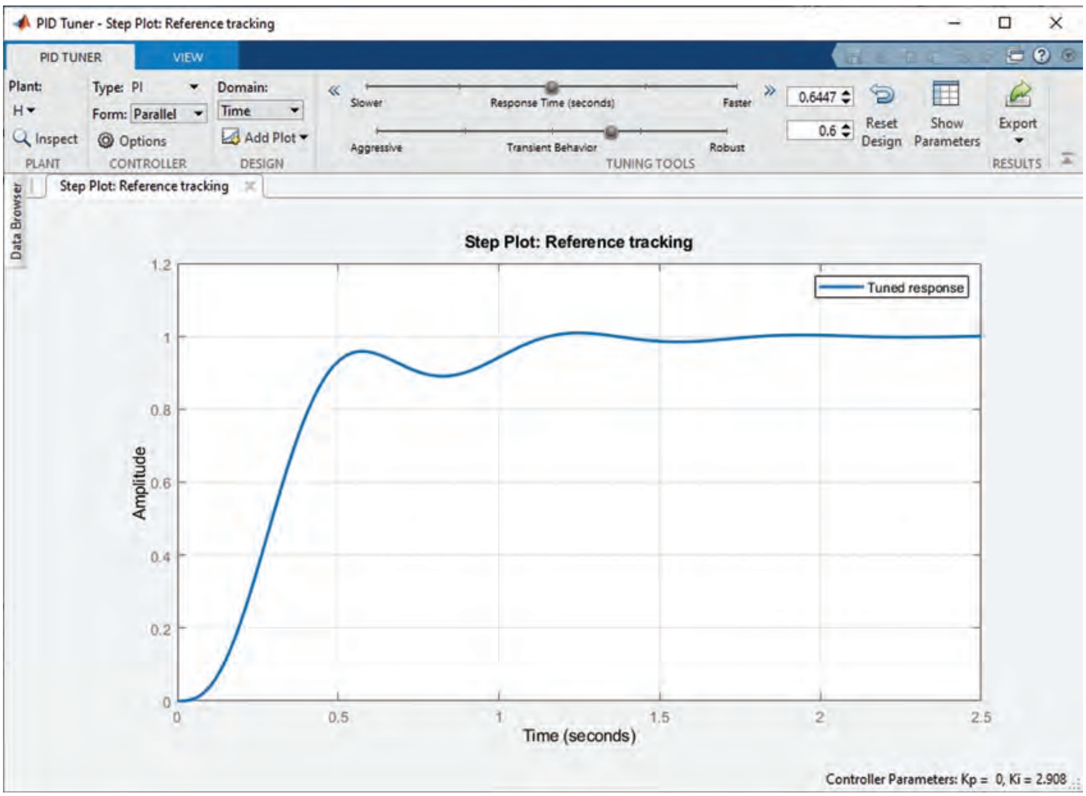


Figure 1.24

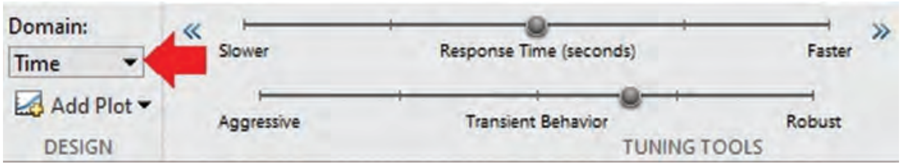


Figure 1.25

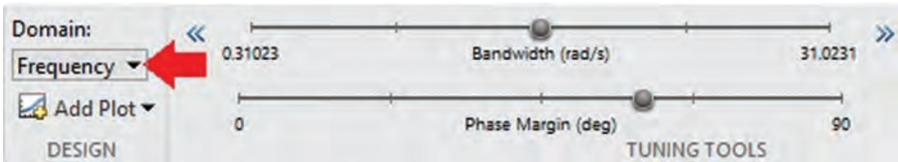


Figure 1.26

Sometimes the output signal of plant is quite good, however the control signal (which is applied to the input of plant) is too big. So, it is a good idea to activate the Controller effort window (Fig. 1.27) to see the control signal as well (Fig. 1.28). This allows you to see whether or not the control signal is in the allowed range.

After designing a suitable controller, you can export the designed controller to the MATLAB environment by clicking the Export button (Fig. 1.29). After clicking the Export button, the window shown in Fig. 1.30 appears. Enter the desired name to Export PID controller box and press the OK button.

1.4 EXAMPLE 3: FEEDBACK CONTROL SYSTEM

In this example we will simulate a feedback control system. Consider the feedback control system shown in Fig. 1.31. The plant transfer function is $\frac{100}{s^2+8s+100}$. This simulation uses the Sum (Fig. 1.32) and PID controller (Fig. 1.33) blocks. Settings of Sum and PID controller blocks are shown in Figs. 1.34 and 1.35, respectively.

Run the Simulation. The result shown in Fig. 1.36 is obtained.

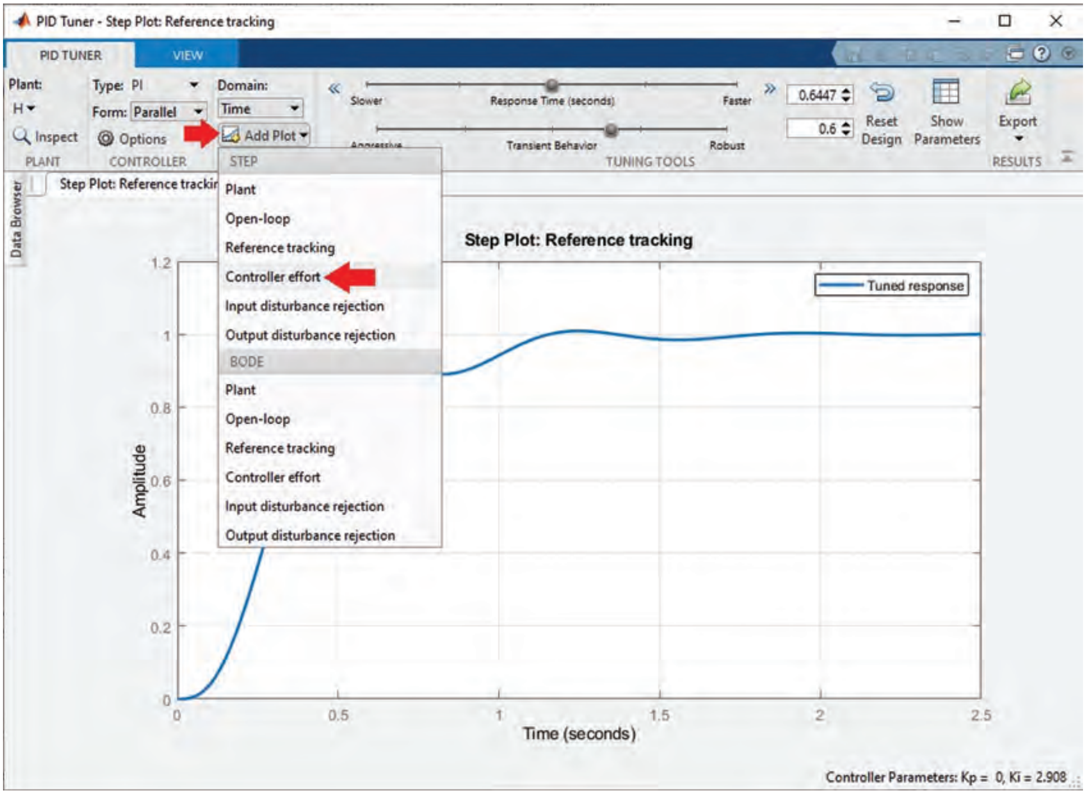

PID Tuner - Step Plot: Reference tracking
PID TUNER
VIEW
Plant:
Type: PI
Domain:
Form: Parallel
Time
Inspect
Options
Add Plot
PLANT
CONTROLLER
Slower
Response Time (seconds)
Faster
Transient Behavior
Robust
TUNING TOOLS
0.6447
0.6
Reset Design
Show Parameters
Export
RESULTS
Data Browser
Step Plot: Reference tracking
STEP
Plant
Open-loop
Reference tracking
Controller effort
Input disturbance rejection
Output disturbance rejection
BODE
Plant
Open-loop
Reference tracking
Controller effort
Input disturbance rejection
Output disturbance rejection
Step Plot: Reference tracking
Tuned response
Amplitude
Time (seconds)
Controller Parameters: Kp = 0, Ki = 2.908

Figure 1.27

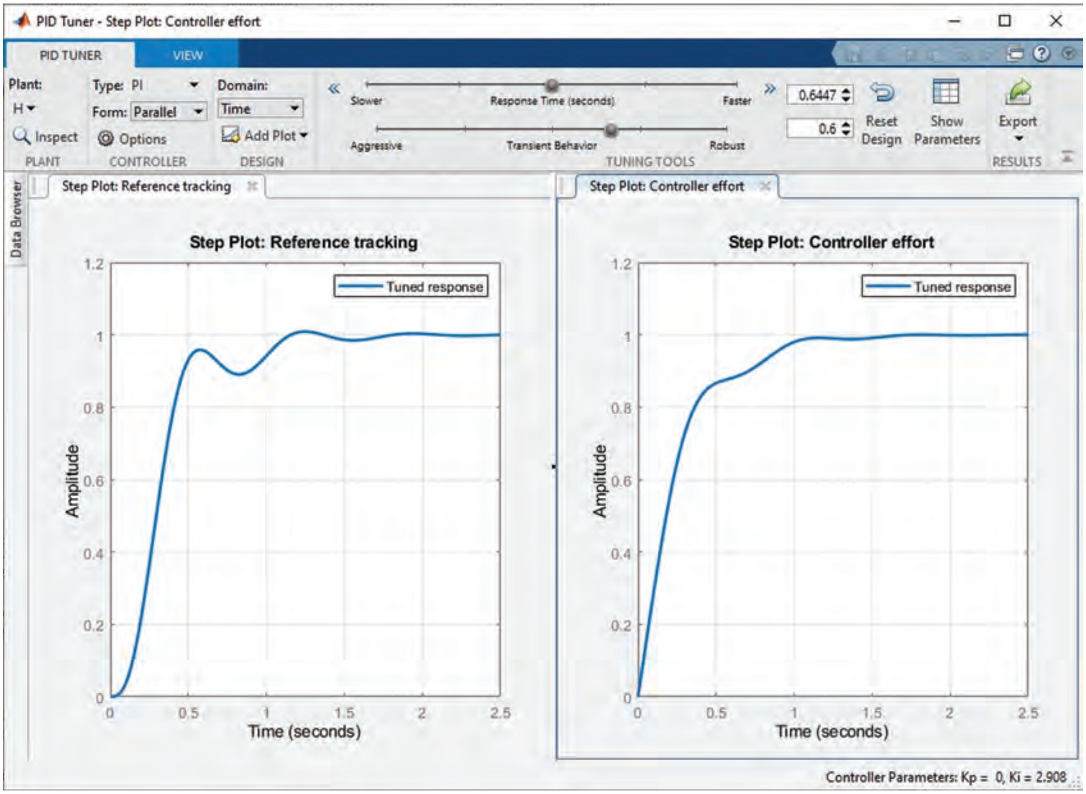

PID Tuner - Step Plot: Controller effort
PID TUNER
VIEW
Plant:
Type: PI
Domain:
Form: Parallel
Time
Inspect
Options
Add Plot
PLANT
CONTROLLER
DESIGN
Slower
Response Time (seconds)
Faster
Aggressive
Transient Behavior
Robust
TUNING TOOLS
0.6447
0.6
Reset Design
Show Parameters
Export
RESULTS
Data Browser
Step Plot: Reference tracking
Step Plot: Controller effort
Tuned response
Amplitude
Time (seconds)
Controller Parameters: Kp = 0, Ki = 2.908

Figure 1.28

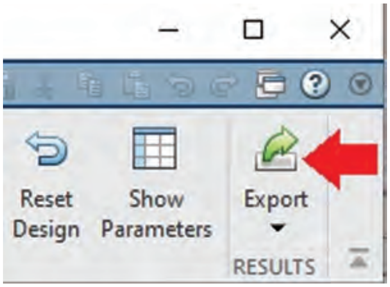


Figure 1.29

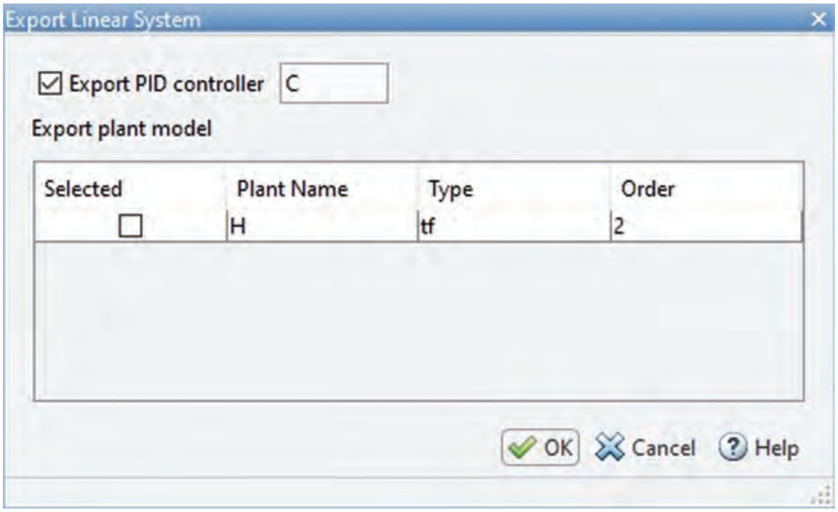


Figure 1.30

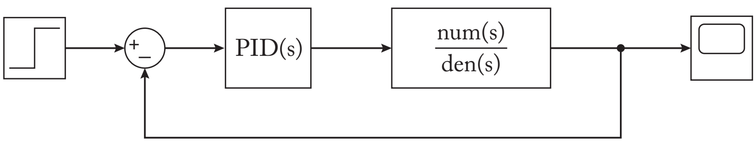


Figure 1.31

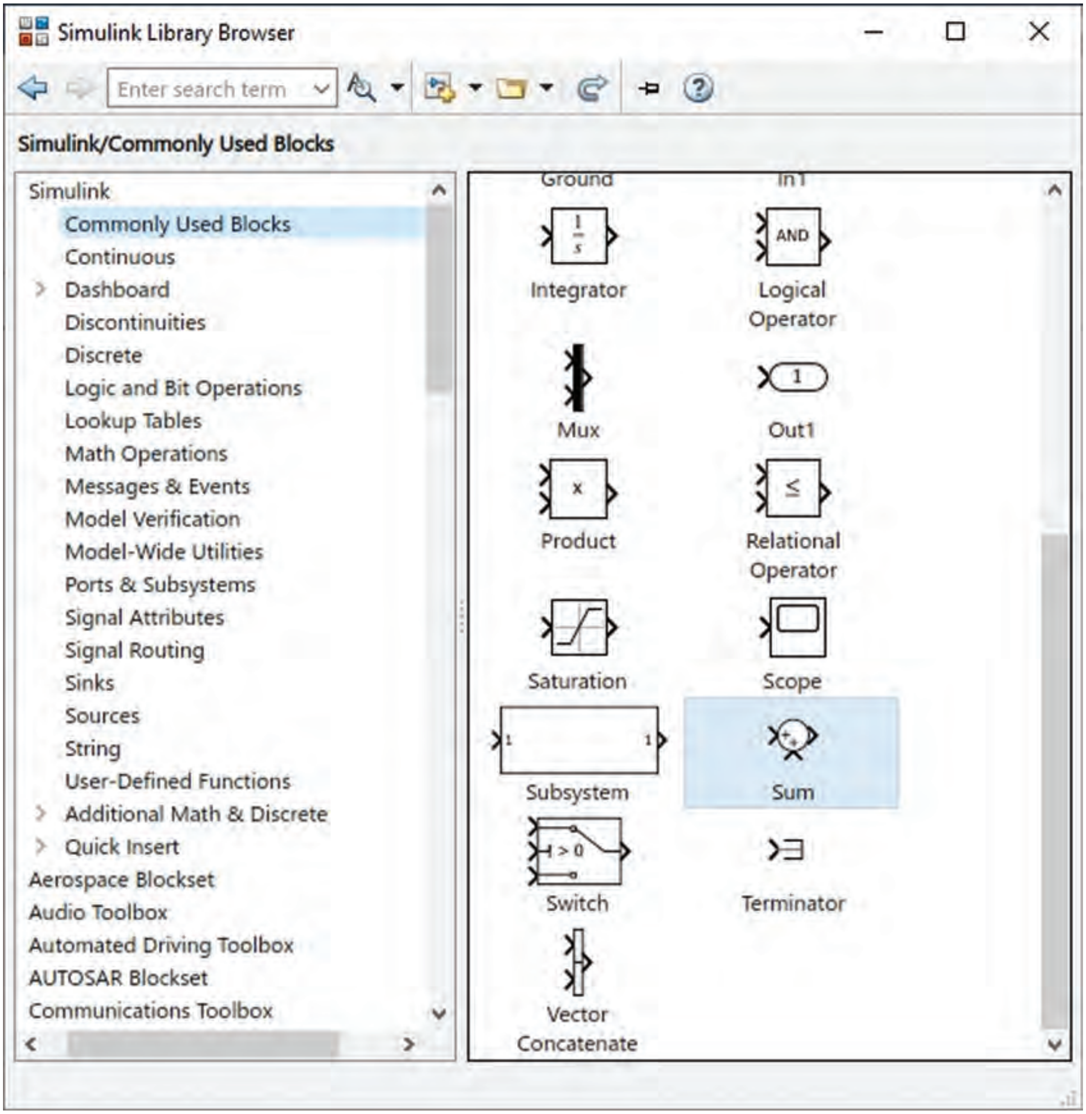

Simulink Library Browser
Enter search term
Simulink/Commonly Used Blocks
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Messages & Events
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
String
User-Defined Functions
Additional Math & Discrete
Quick Insert
Aerospace Blockset
Audio Toolbox
Automated Driving Toolbox
AUTOSAR Blockset
Communications Toolbox
Ground
In1
Integrator
AND
Logical Operator
Mux
Out1
Product
Relational Operator
Saturation
Scope
Subsystem
Sum
Switch
Terminator
Vector Concatenate

Figure 1.32

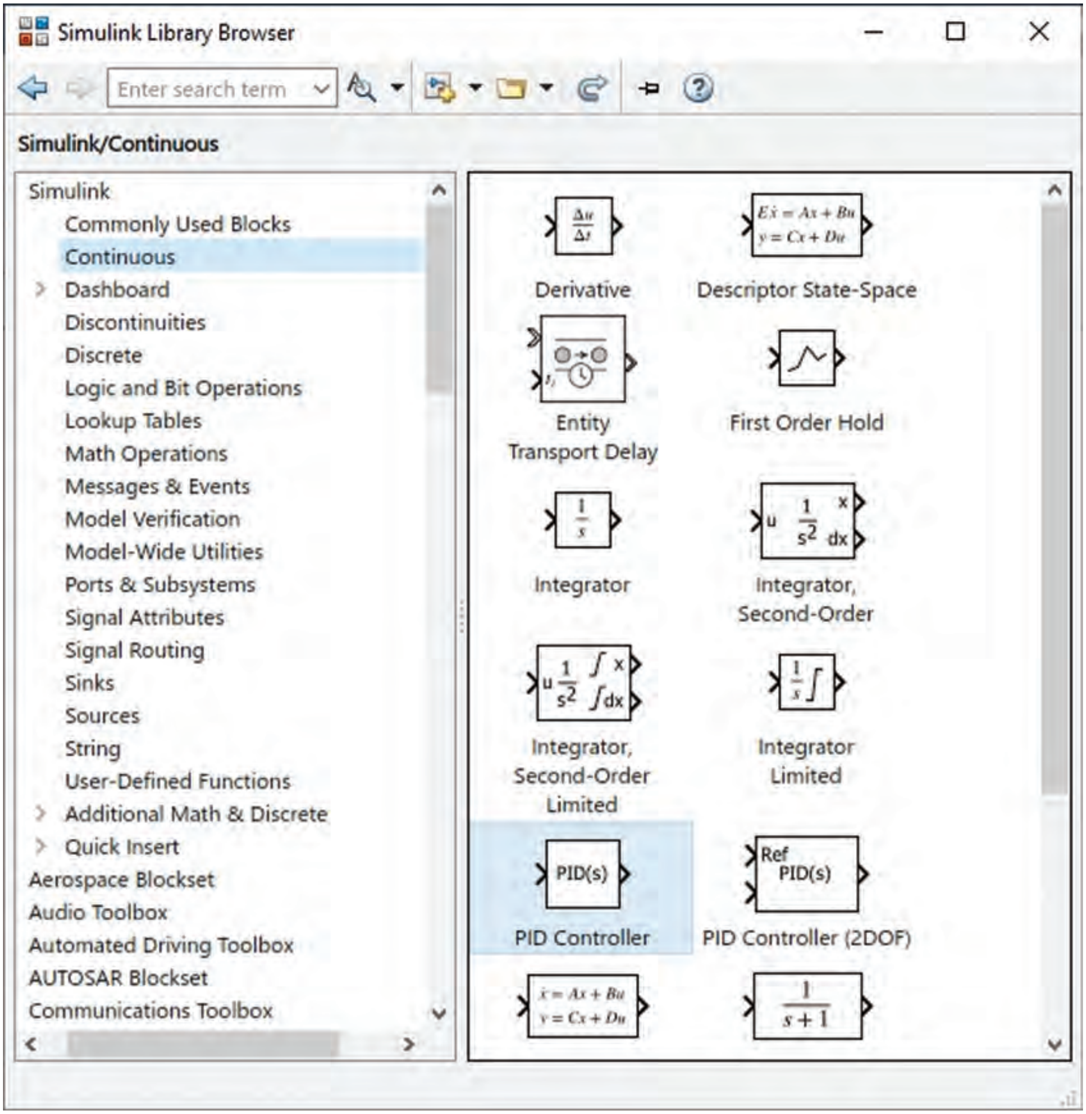

Simulink Library Browser
Enter search term
Simulink/Continuous
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Messages & Events
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
String
User-Defined Functions
Additional Math & Discrete
Quick Insert
Aerospace Blockset
Audio Toolbox
Automated Driving Toolbox
AUTOSAR Blockset
Communications Toolbox
Derivative
Descriptor State-Space
Entity Transport Delay
First Order Hold
Integrator
Integrator, Second-Order
Integrator, Second-Order Limited
Integrator Limited
PID(s)
PID Controller
Ref
PID Controller (2DOF)

Figure 1.33

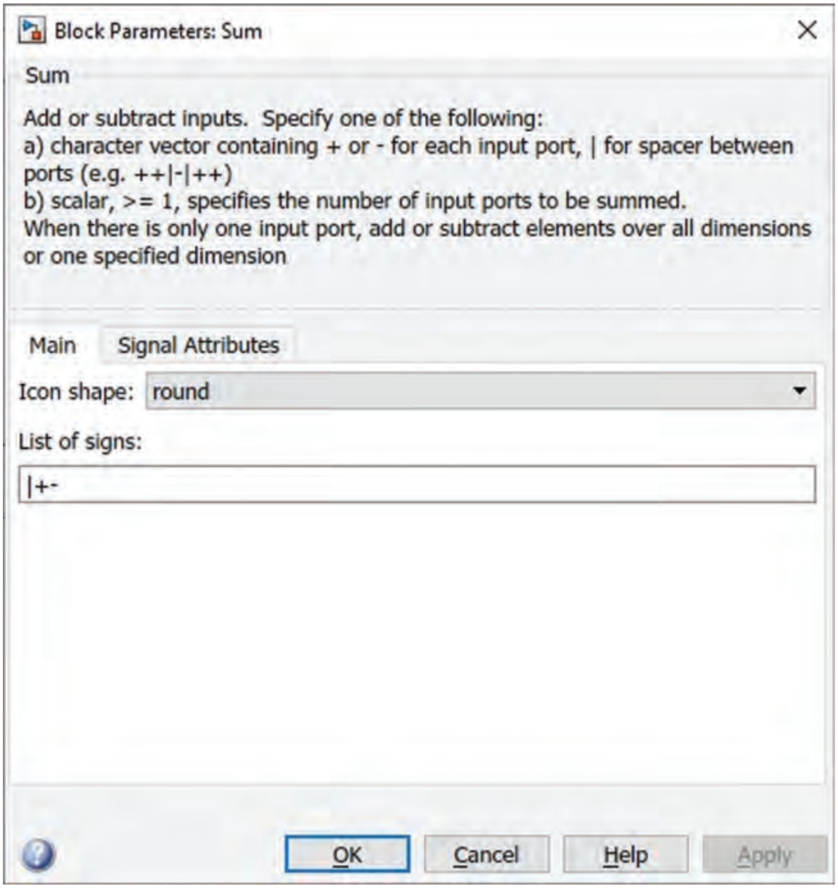

Block Parameters: Sum
Sum
Add or subtract inputs. Specify one of the following:
a) character vector containing + or - for each input port, | for spacer between ports (e.g. ++|-|++)
b) scalar, >= 1, specifies the number of input ports to be summed.
When there is only one input port, add or subtract elements over all dimensions or one specified dimension
Main
Signal Attributes
Icon shape:
round
List of signs:
|+-
OK
Cancel
Help
Apply

Figure 1.34

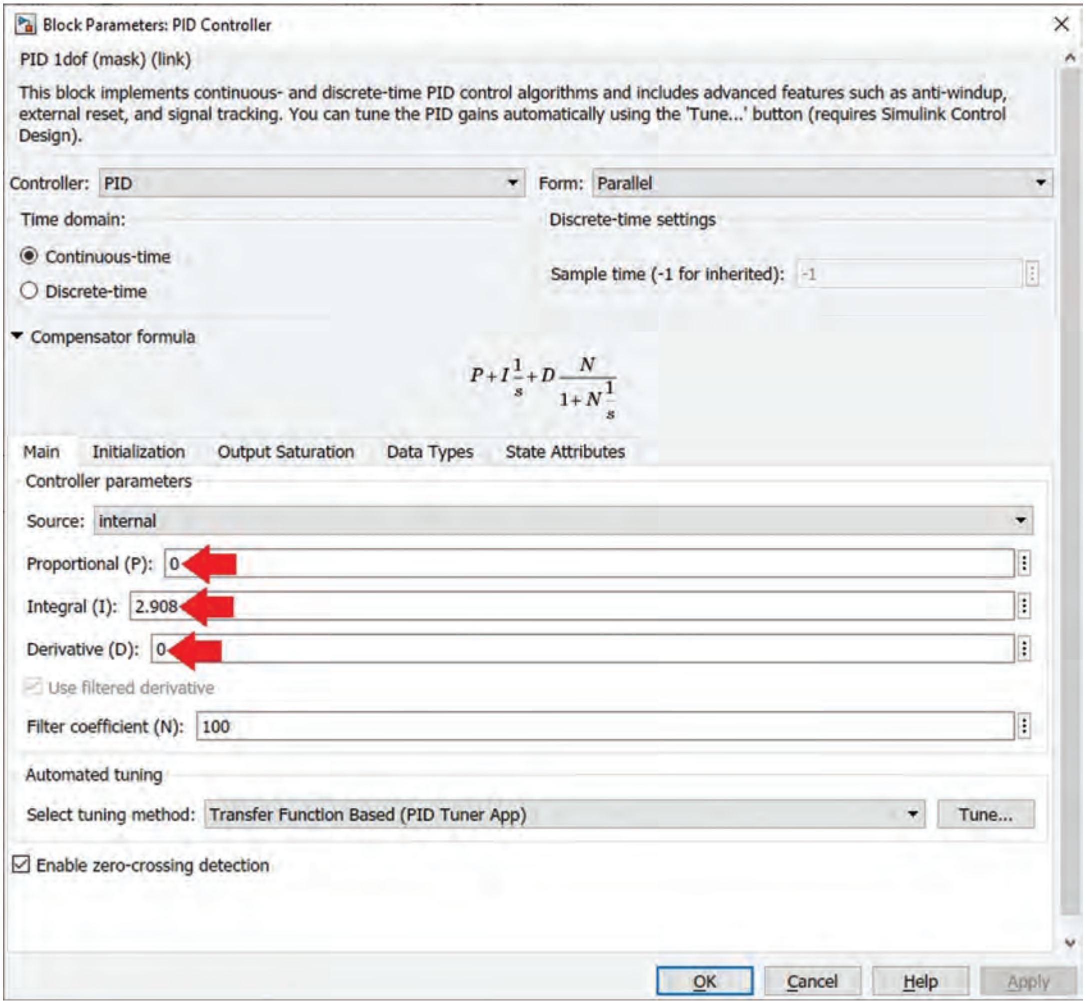

Block Parameters: PID Controller
PID 1dof (mask) (link)
This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (requires Simulink Control Design).
Controller: PID
Form: Parallel
Time domain:
Continuous-time
Discrete-time
Discrete-time settings
Sample time (-1 for inherited): -1
Compensator formula
$P+I\frac{1}{s}+D\frac{N}{1+N\frac{1}{s}}$
Main
Initialization
Output Saturation
Data Types
State Attributes
Controller parameters
Source: internal
Proportional (P): 0
Integral (I): 2.908
Derivative (D): 0
Use filtered derivative
Filter coefficient (N): 100
Automated tuning
Select tuning method: Transfer Function Based (PID Tuner App)
Tune...
Enable zero-crossing detection
OK
Cancel
Help
Apply

Figure 1.35

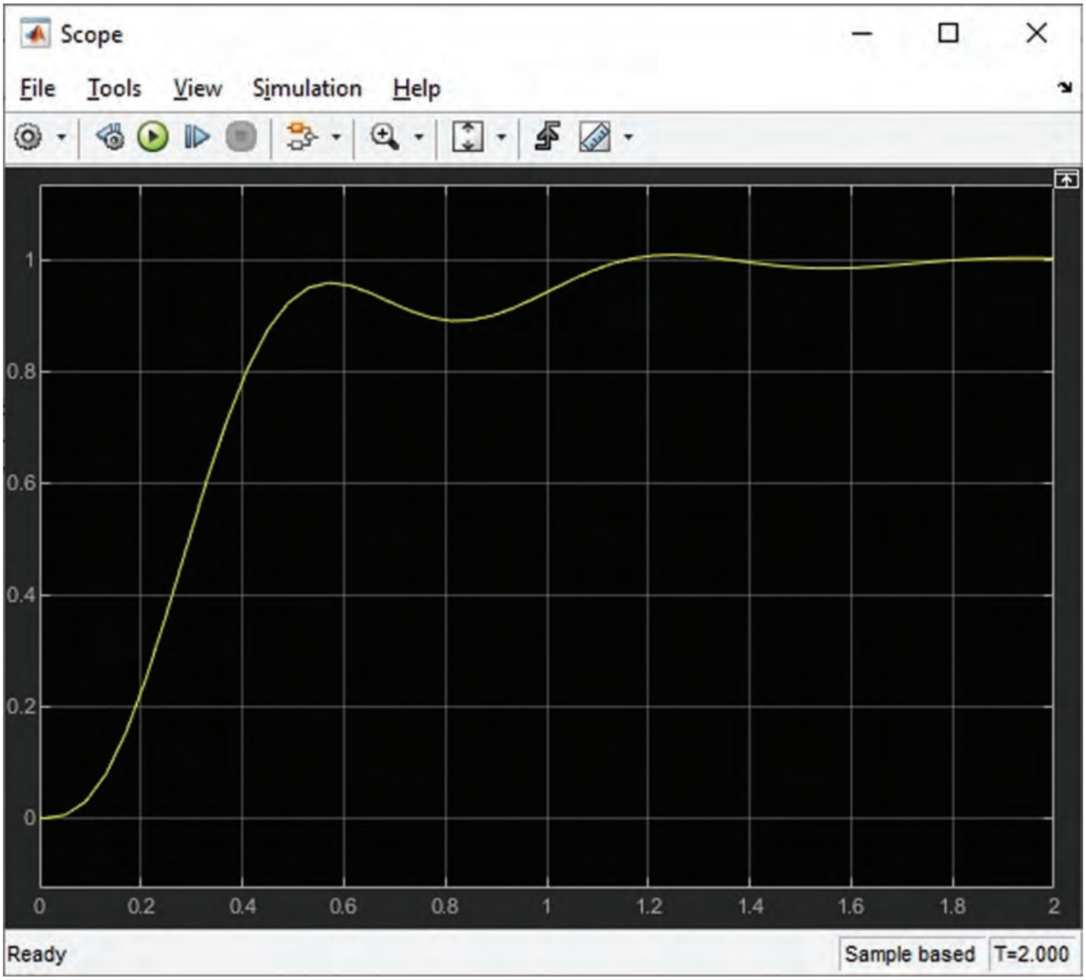


Figure 1.36

1.5 EXAMPLE 4: PID CONTROLLER DESIGN IN SIMULINK ENVIRONMENT

You can do the tuning in the Simulink environment as well. In this example, we will tune a PID controller in the Simulink environment. Consider the Simulink model shown in Fig. 1.37. The plant transfer function is $\frac{100}{s^2+8s+100}$. The PID controller block has the default parameter values.

Double click the PID controller block and click the Tune button (Fig. 1.38). After clicking the Tune button, the window shown in Fig. 1.39 appears and permits you to tune the controller.

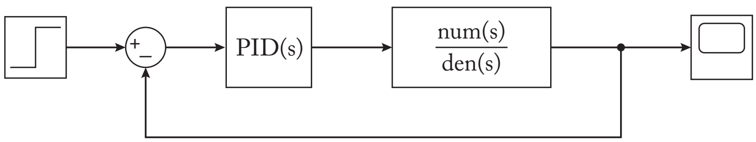


Figure 1.37

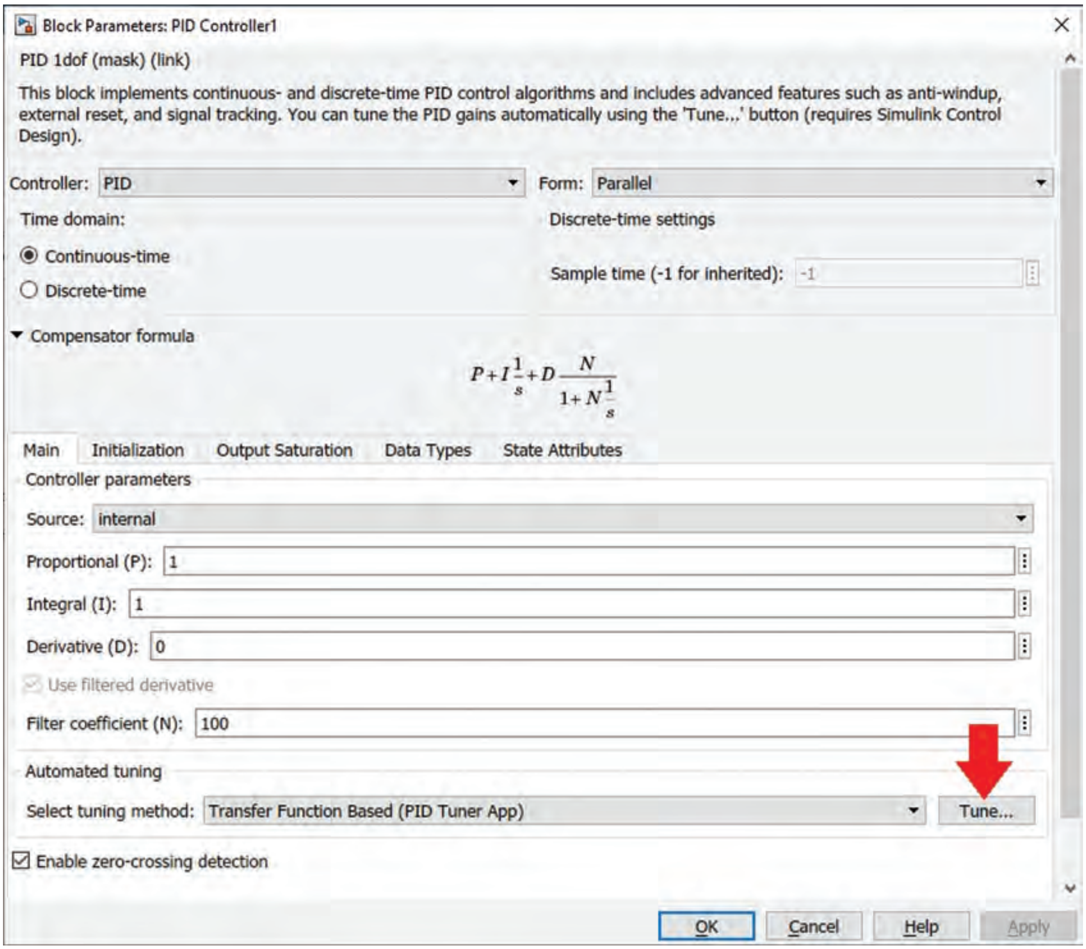


Figure 1.38

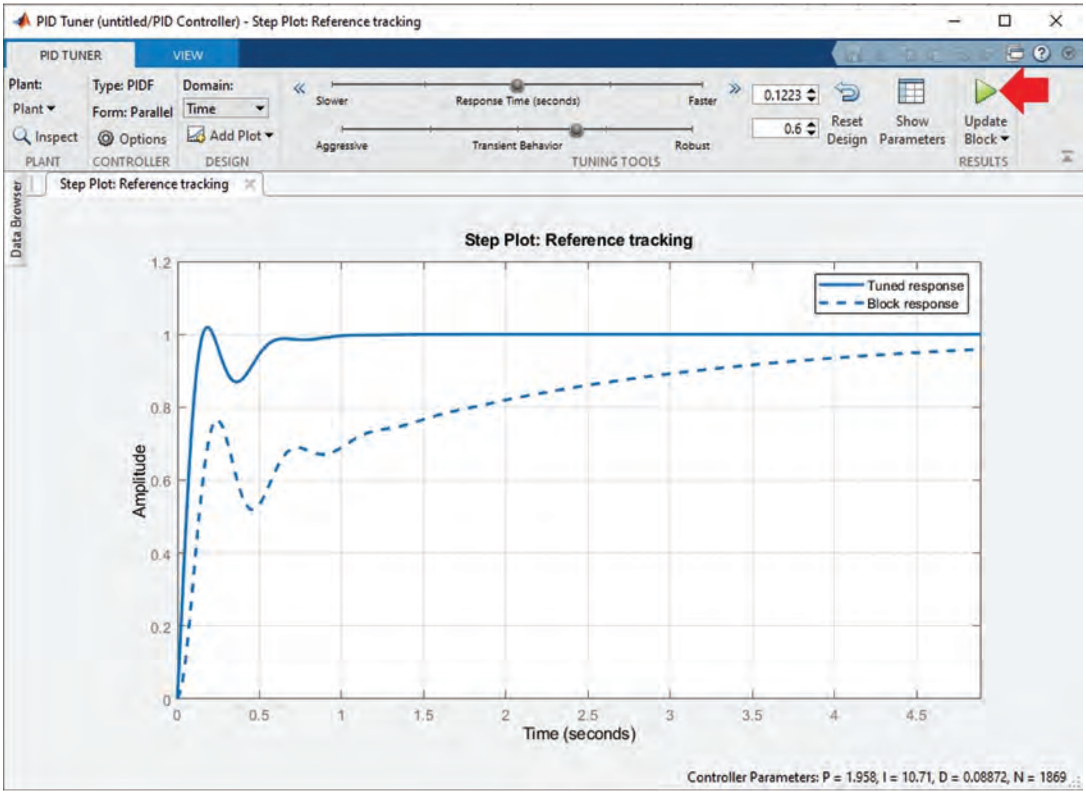


Figure 1.39

Sometimes the output signal of plant is quite good, however the control signal (which is applied to the input of plant) is too big. So, it is a good idea to activate the controller effort window (Fig. 1.40) to see the control signal as well (Fig. 1.41). This allows you to see whether or not the control signal is in the allowed range. After tuning the controller, click the Update Block button to apply the changes to the block.

1.6 EXAMPLE 5: PLOT TWO OR MORE WAVEFORMS IN ONE SCOPE BLOCK

In this example we see how to see two or more signals simultaneously. Consider the model shown in Fig. 1.42. Plant transfer function is $\frac{100}{s^2+8s+100}$. Settings of the PID controller block are shown Fig. 1.43.

Click on the connection between the scope and output of system (Fig. 1.44) and press the Delete key to remove it (Fig. 1.45).

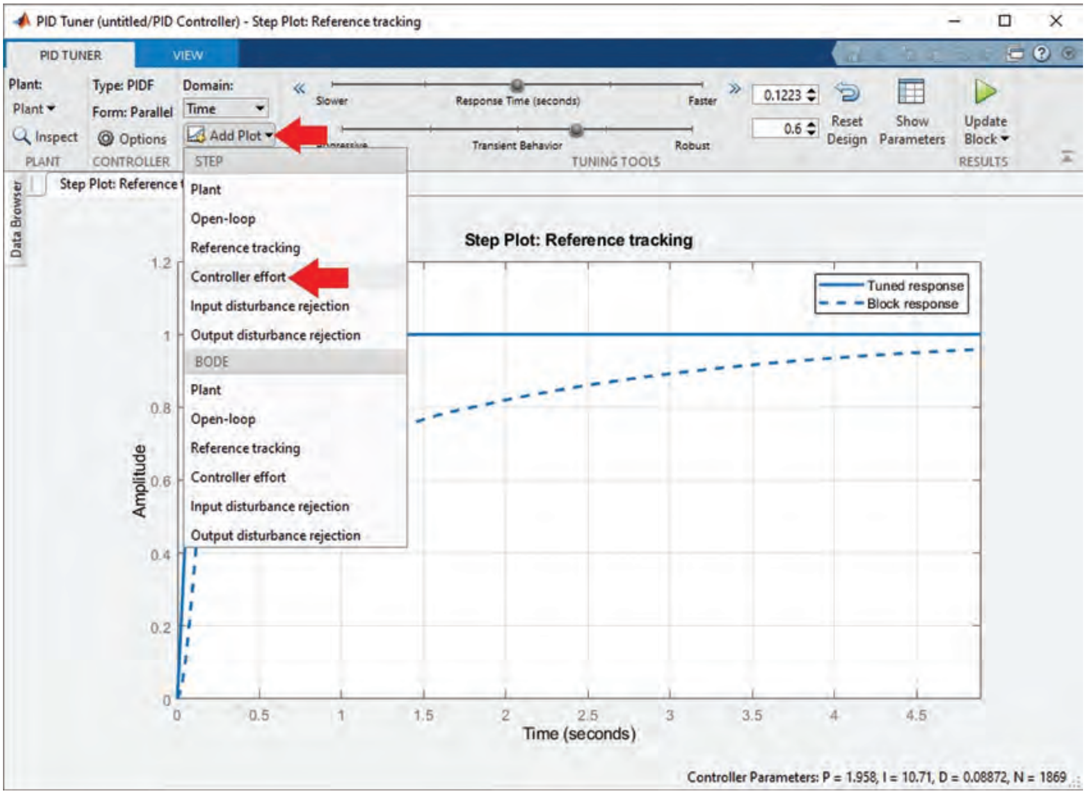

PID Tuner (untitled/PID Controller) - Step Plot: Reference tracking
PID TUNER
VIEW
Plant:
Plant
Inspect
PLANT
Type: PIDF
Form: Parallel
Options
CONTROLLER
Domain:
Time
Add Plot
Slower
Response Time (seconds)
Faster
Transient Behavior
Robust
TUNING TOOLS
0.1223
0.6
Reset Design
Show Parameters
Update Block
RESULTS
Data Browser
STEP
Plant
Open-loop
Reference tracking
Controller effort
Input disturbance rejection
Output disturbance rejection
BODE
Plant
Open-loop
Reference tracking
Controller effort
Input disturbance rejection
Output disturbance rejection
Step Plot: Reference tracking
Tuned response
Block response
Amplitude
Time (seconds)
Controller Parameters: P = 1.958, I = 10.71, D = 0.08872, N = 1869

Figure 1.40

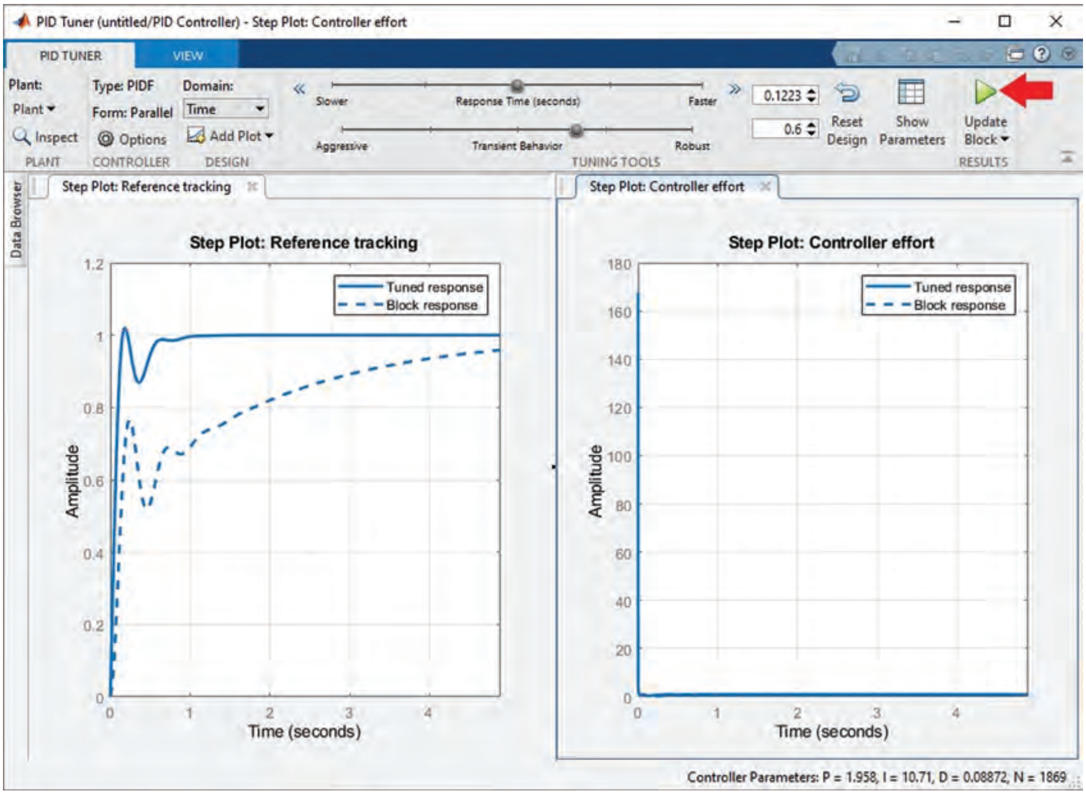


Figure 1.41

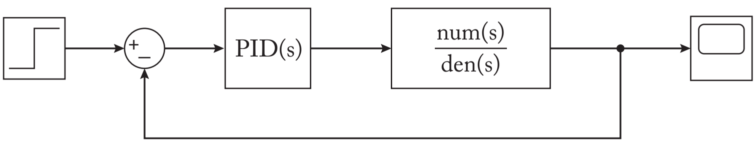


Figure 1.42

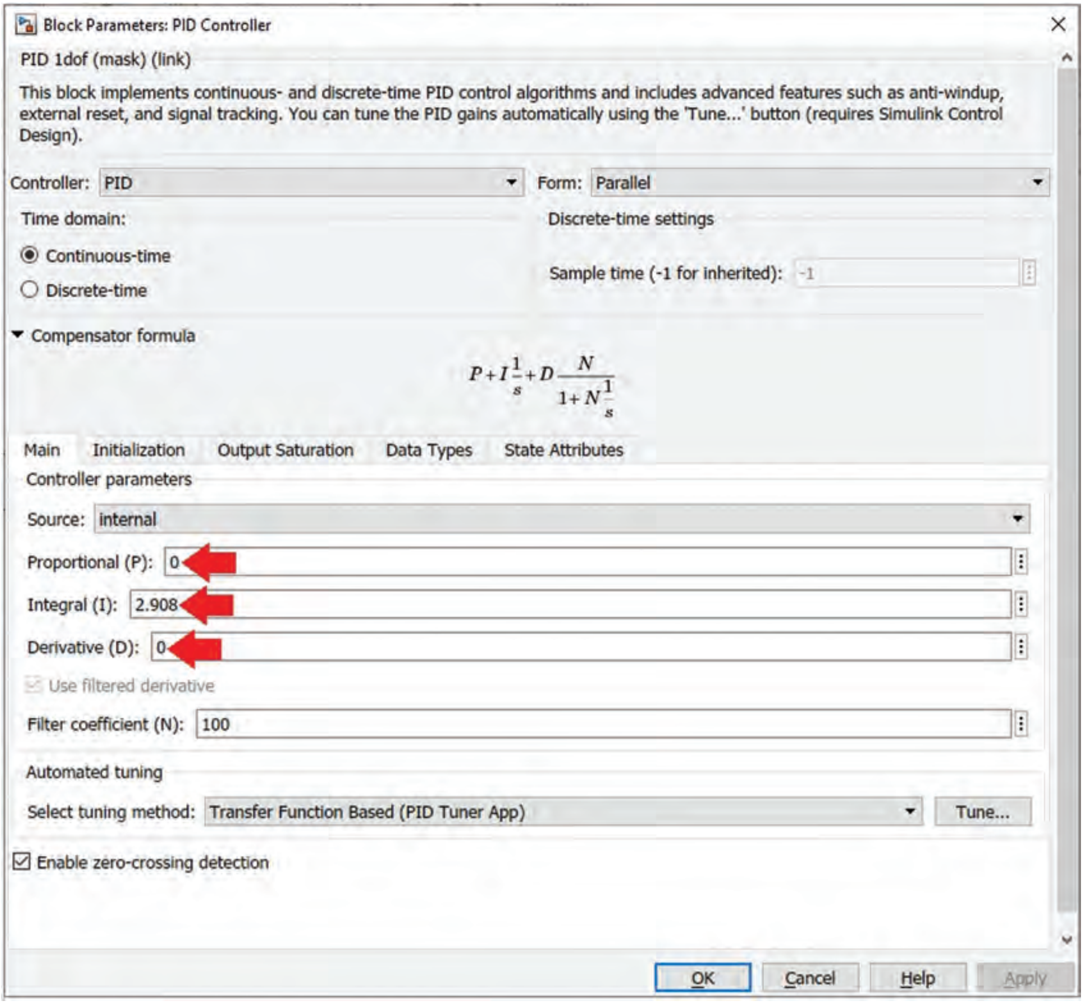


Figure 1.43

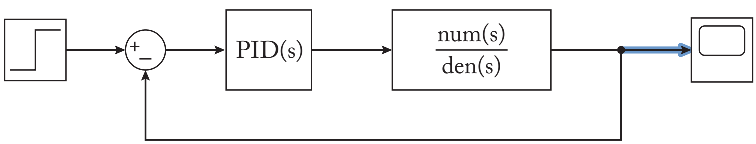


Figure 1.44

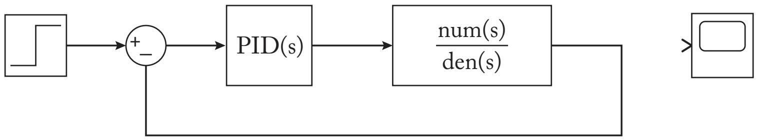


Figure 1.45

Double click on the scope block and select 2 for Number of Input Ports (Fig. 1.46). The Scope blocks changes to what is shown in Fig. 1.47. Connect the inputs of Scope block to the desired nodes of the system.

Run the simulation. The result shown in Fig. 1.49 is obtained. One of the signals has round markers on it. You can remove these round markers by clicking the Style icon (Fig. 1.50). After clicking the Style icon, the window shown in Fig. 1.51 appears. Convert the Marker box to None (Fig. 1.52). Now the waveform has no round markers on it (Fig. 1.53).

There is another way to see two or more signals simultaneously: using the multiplexer (Mux) block (Fig. 1.54). If you double click on the Mux block, the window shown in Fig. 1.55 appears and permits you to determine the desired number of inputs for the Mux block. The block diagram shown in Fig. 1.56 shows the output of system and control input simultaneously (Fig. 1.57).

1.7 EXAMPLE 6: SIMULATION OF DIFFERENTIAL EQUATIONS

In this example we want to simulate the following system:

$$\ddot{y} + 5\dot{y} - 10y = 7\sin\left(3t + \frac{\pi}{3}\right), \quad y(0) = 1, \quad \dot{y}(0) = 4 \tag{1.1}$$

Let's define two new variables and convert the given equation into the state space system.

$$\begin{cases} x_1 = y \\ x_2 = \dot{y} = \dfrac{dy}{dt} \end{cases} \tag{1.2}$$

The state space representation of the system is:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = 10x_1 - 5x_2 + 7\sin\left(3t + \dfrac{\pi}{3}\right) \end{cases}, \quad x_0 = \begin{bmatrix} 1 \\ 4 \end{bmatrix} \tag{1.3}$$

where x_0 shows the initial condition of the system. This state space representation is suitable for drawing the Simulink model. Add two Integrator blocks (Fig. 1.58) to the Simulink model (Fig. 1.59).

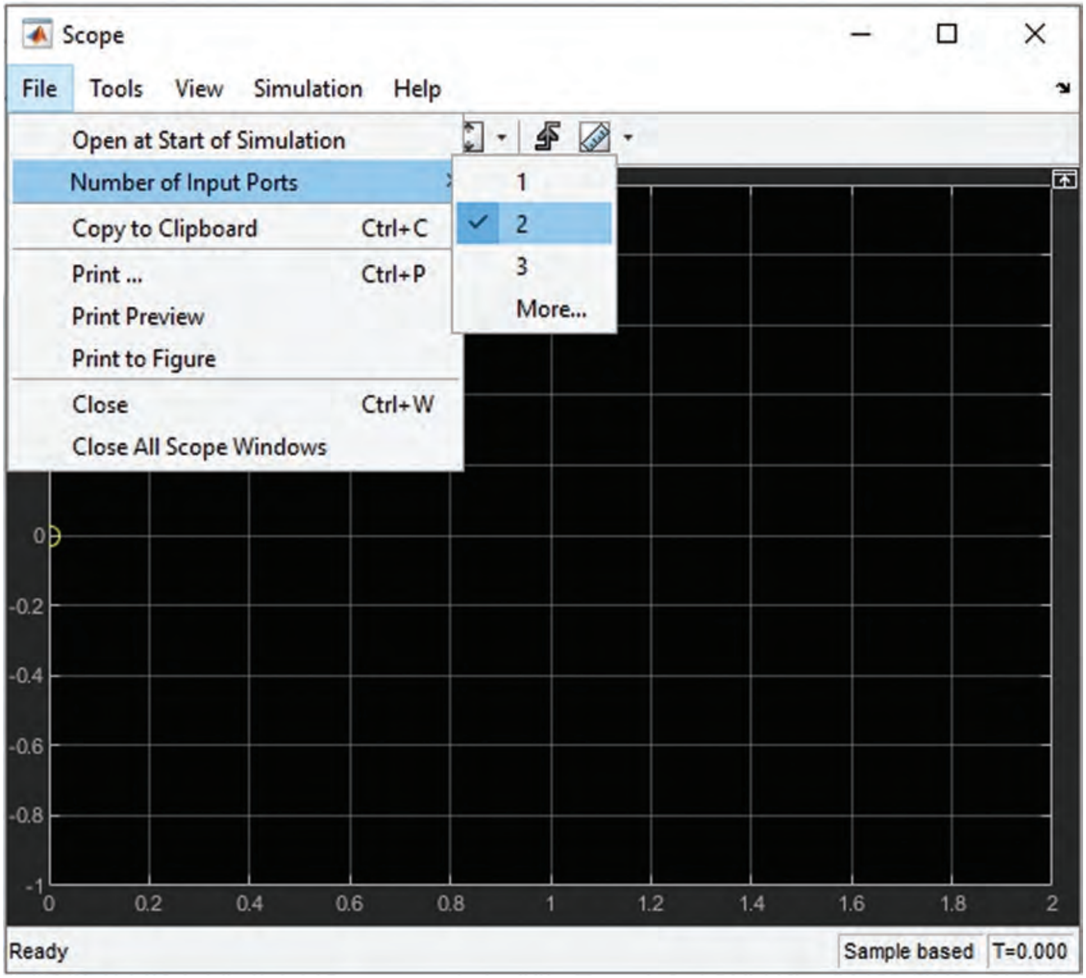


Figure 1.46

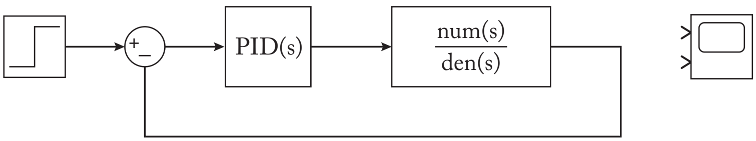


Figure 1.47

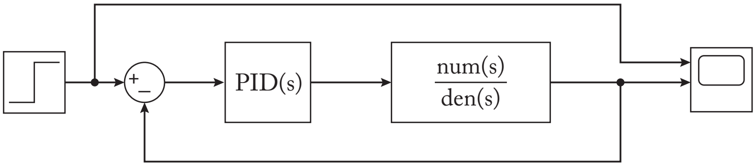


Figure 1.48

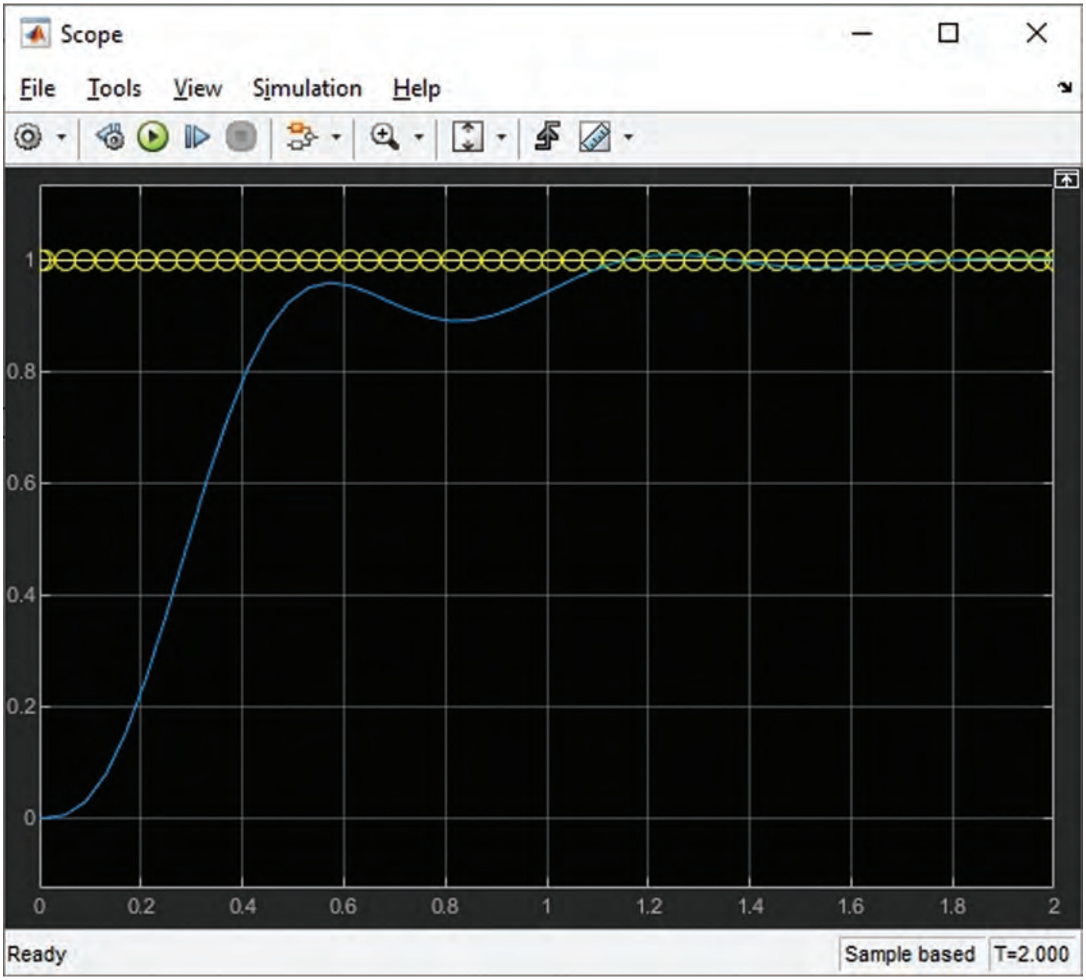


Figure 1.49

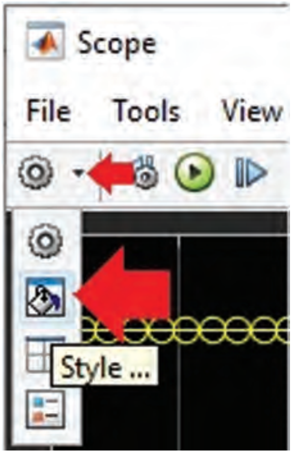


Figure 1.50

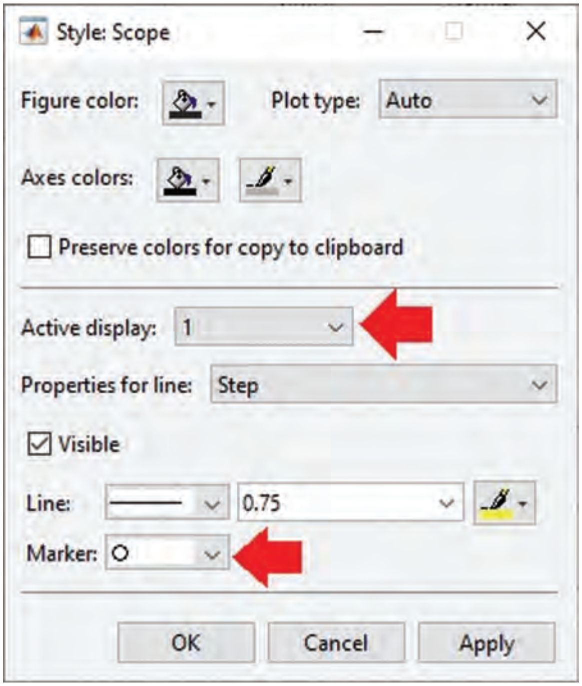


Figure 1.51

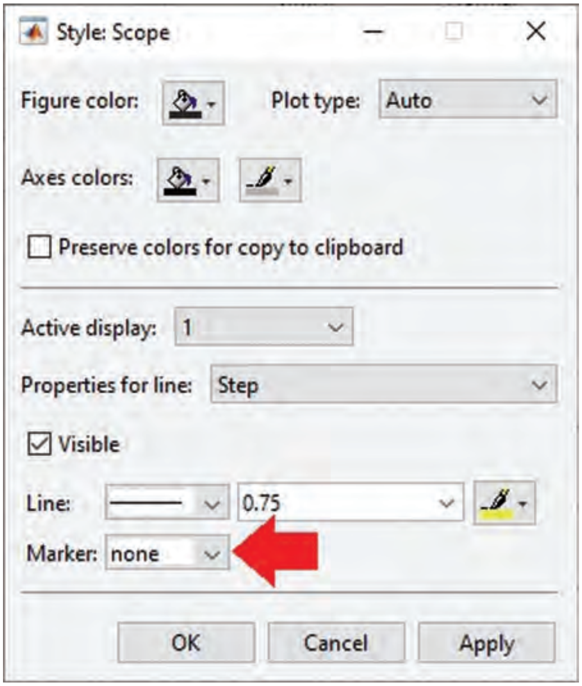

Style: Scope
Figure color:
Plot type:
Auto
Axes colors:
Preserve colors for copy to clipboard
Active display:
1
Properties for line:
Step
Visible
Line:
0.75
Marker:
none
OK
Cancel
Apply

Figure 1.52

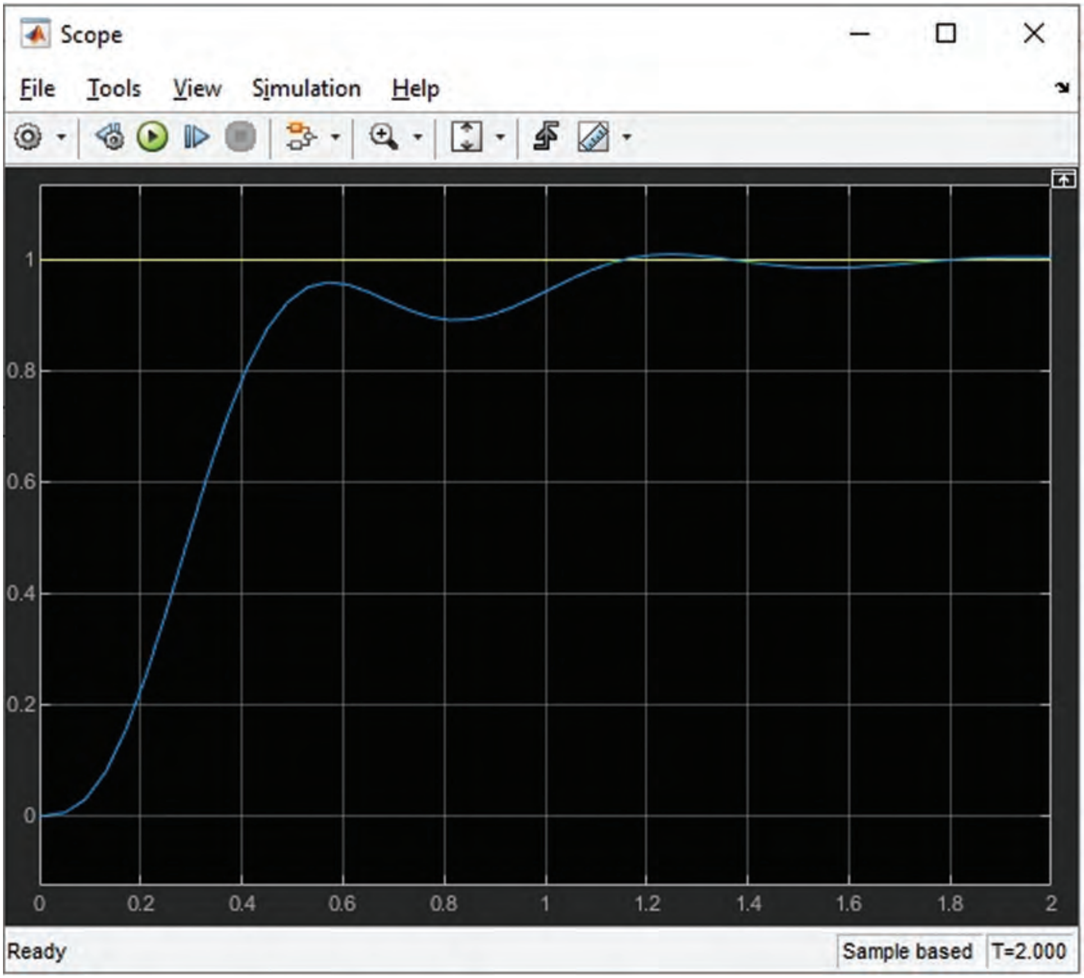

Scope
File Tools View Simulation Help
1
0.8
0.6
0.4
0.2
0
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2
Ready
Sample based
T=2.000

Figure 1.53

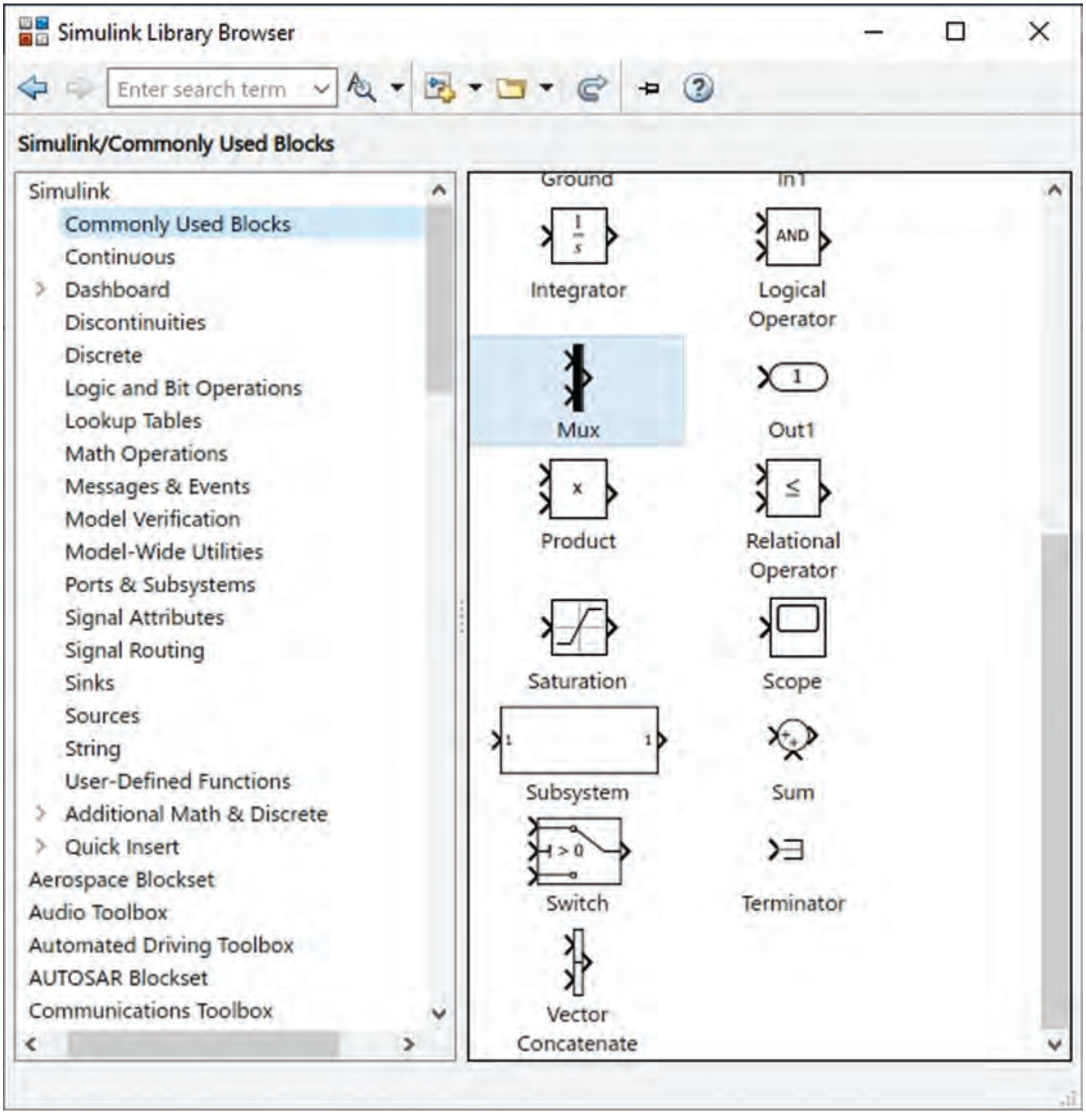

Simulink Library Browser
Enter search term
Simulink/Commonly Used Blocks
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Messages & Events
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
String
User-Defined Functions
Additional Math & Discrete
Quick Insert
Aerospace Blockset
Audio Toolbox
Automated Driving Toolbox
AUTOSAR Blockset
Communications Toolbox
Ground
In1
Integrator
AND
Logical Operator
Mux
Out1
Product
Relational Operator
Saturation
Scope
Subsystem
Sum
Switch
Terminator
Vector Concatenate

Figure 1.54

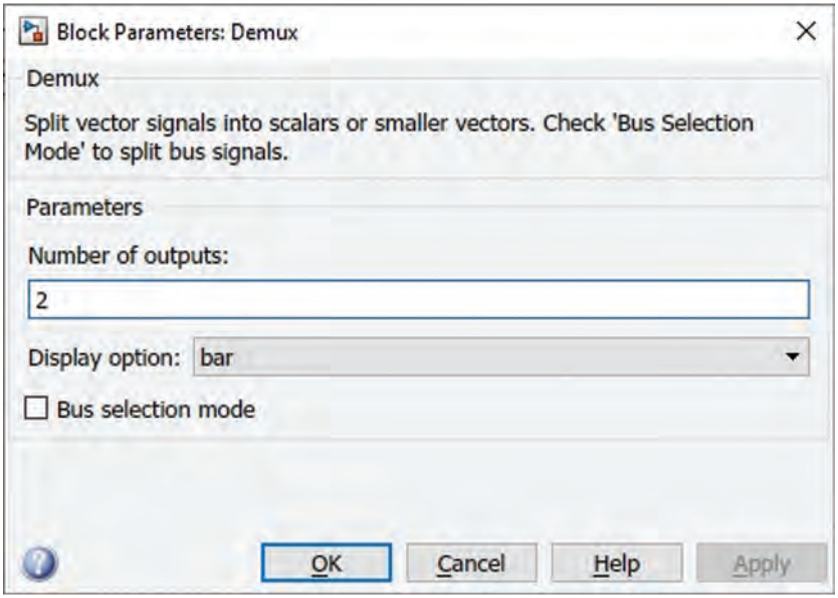


Figure 1.55

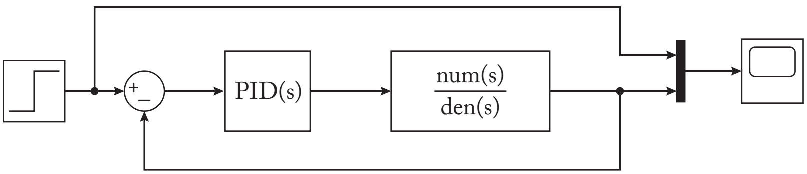


Figure 1.56

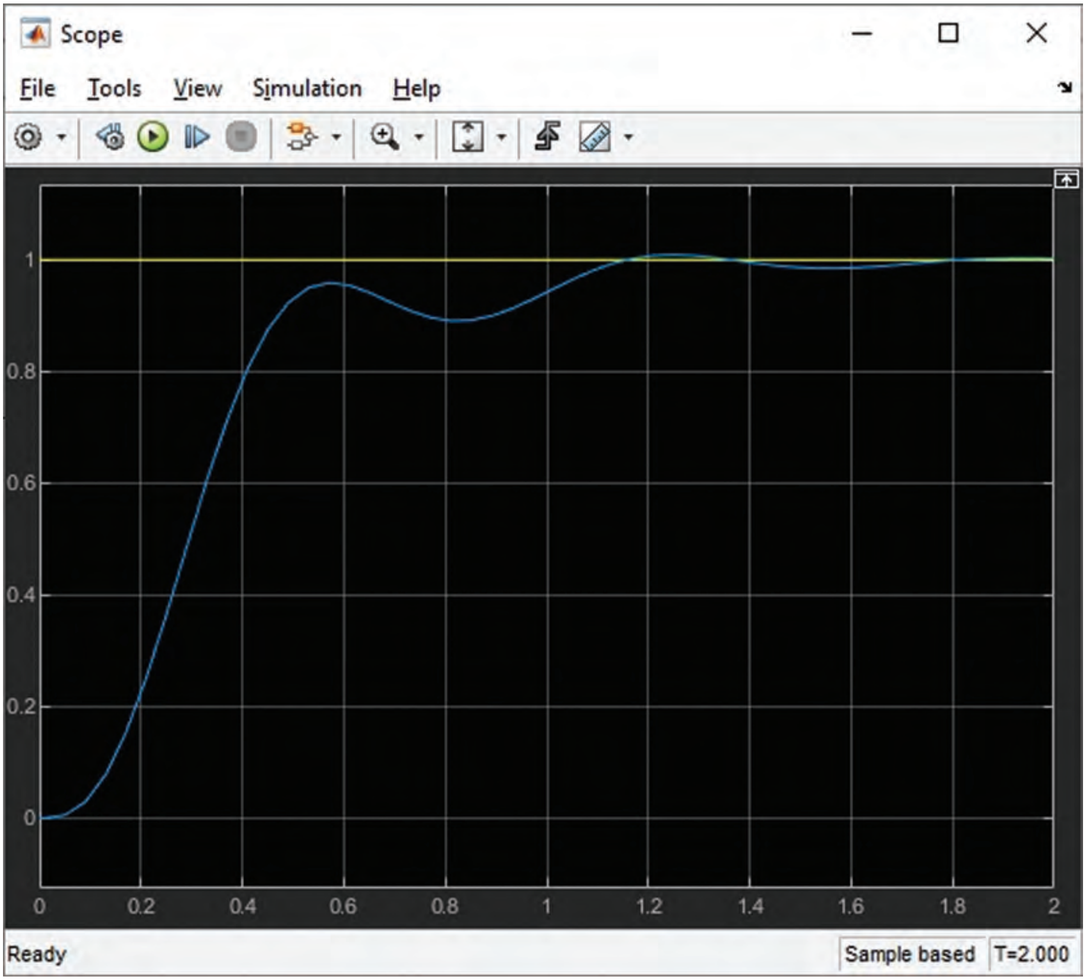

Scope
File
Tools
View
Simulation
Help
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Ready
Sample based
T=2.000

Figure 1.57

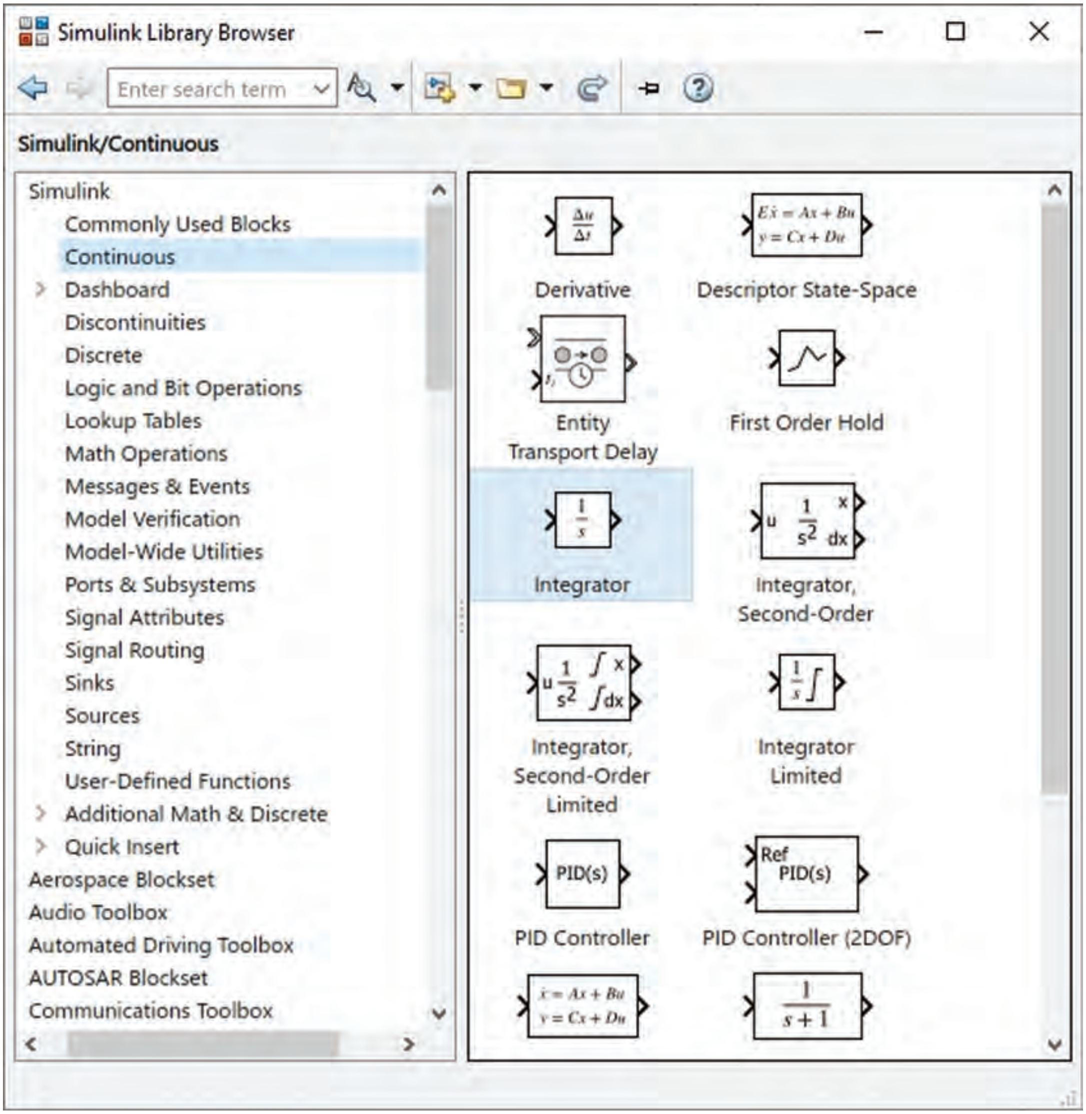


Figure 1.58

Figure 1.59

The relationship between the integrator input and outputs are showed in Fig. 1.60.

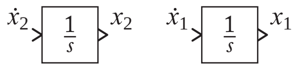


Figure 1.60

According to the obtained state space model, $\dot{x}_1 = x_2$. Implementation of this equation is shown in Fig. 1.61.

Figure 1.61

We need Gain (Fig. 1.62), Sum (Fig. 1.63), and Sine wave (Fig. 1.64) blocks to implement the $\dot{x}_2 = 10x_1 - 5x_2 + 7\sin(3t + \frac{\pi}{3})$. The implementation of this equation is shown in Fig. 1.65. Note that gain blocks are rotated by clicking on them and pressing the Ctrl+R.

Settings of blocks in Fig. 1.65 are shown in Figs. 1.66–1.71. Add two scope blocks to the Simulink model (Fig. 1.72).

We want to study the system behavior for 1 s. Enter 1 to the Stop Time box and run the simulation (Fig. 1.73). Results are shown in Figs. 1.74 and 1.75. According to the obtained result, the system is unstable.

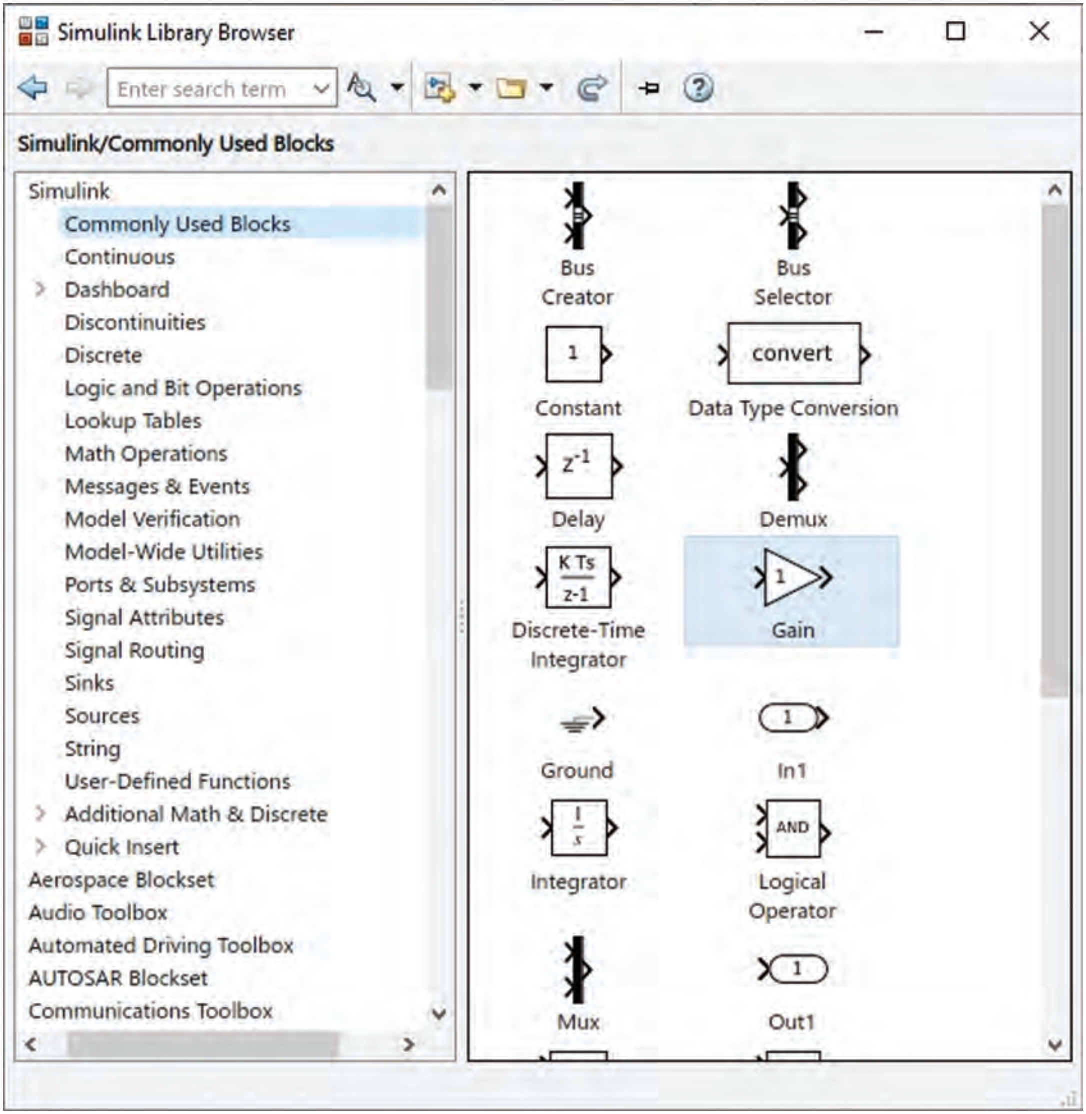

Simulink Library Browser
Enter search term
Simulink/Commonly Used Blocks
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Messages & Events
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
String
User-Defined Functions
Additional Math & Discrete
Quick Insert
Aerospace Blockset
Audio Toolbox
Automated Driving Toolbox
AUTOSAR Blockset
Communications Toolbox
Bus Creator
Bus Selector
1
Constant
convert
Data Type Conversion
Delay
Demux
K Ts
z-1
Discrete-Time Integrator
1
Gain
Ground
1
In1
Integrator
AND
Logical Operator
Mux
1
Out1

Figure 1.62

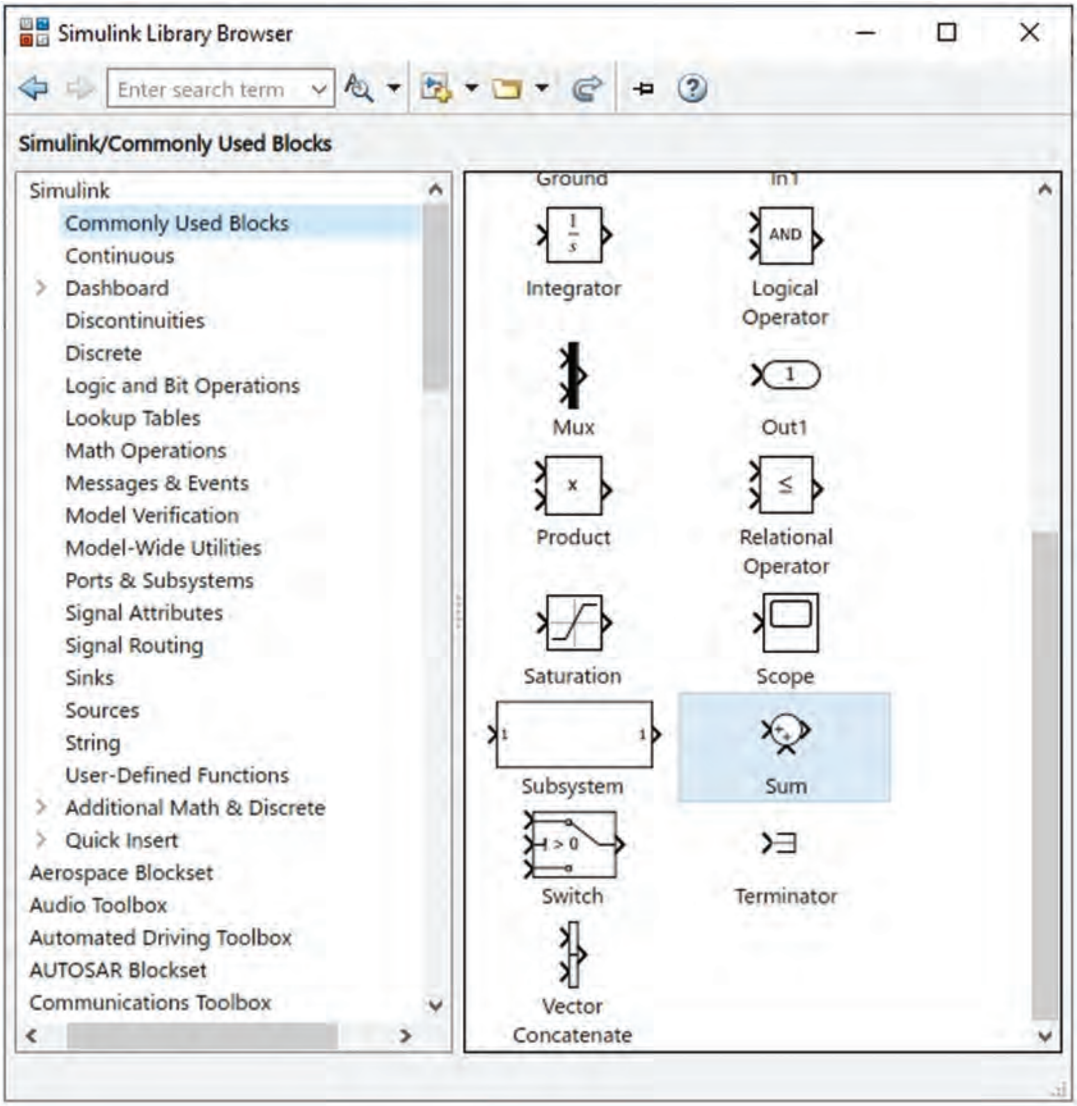

Simulink Library Browser
Enter search term
Simulink/Commonly Used Blocks
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Messages & Events
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
String
User-Defined Functions
Additional Math & Discrete
Quick Insert
Aerospace Blockset
Audio Toolbox
Automated Driving Toolbox
AUTOSAR Blockset
Communications Toolbox
Ground
In1
Integrator
AND
Logical Operator
Mux
Out1
Product
Relational Operator
Saturation
Scope
Subsystem
Sum
Switch
Terminator
Vector Concatenate

Figure 1.63

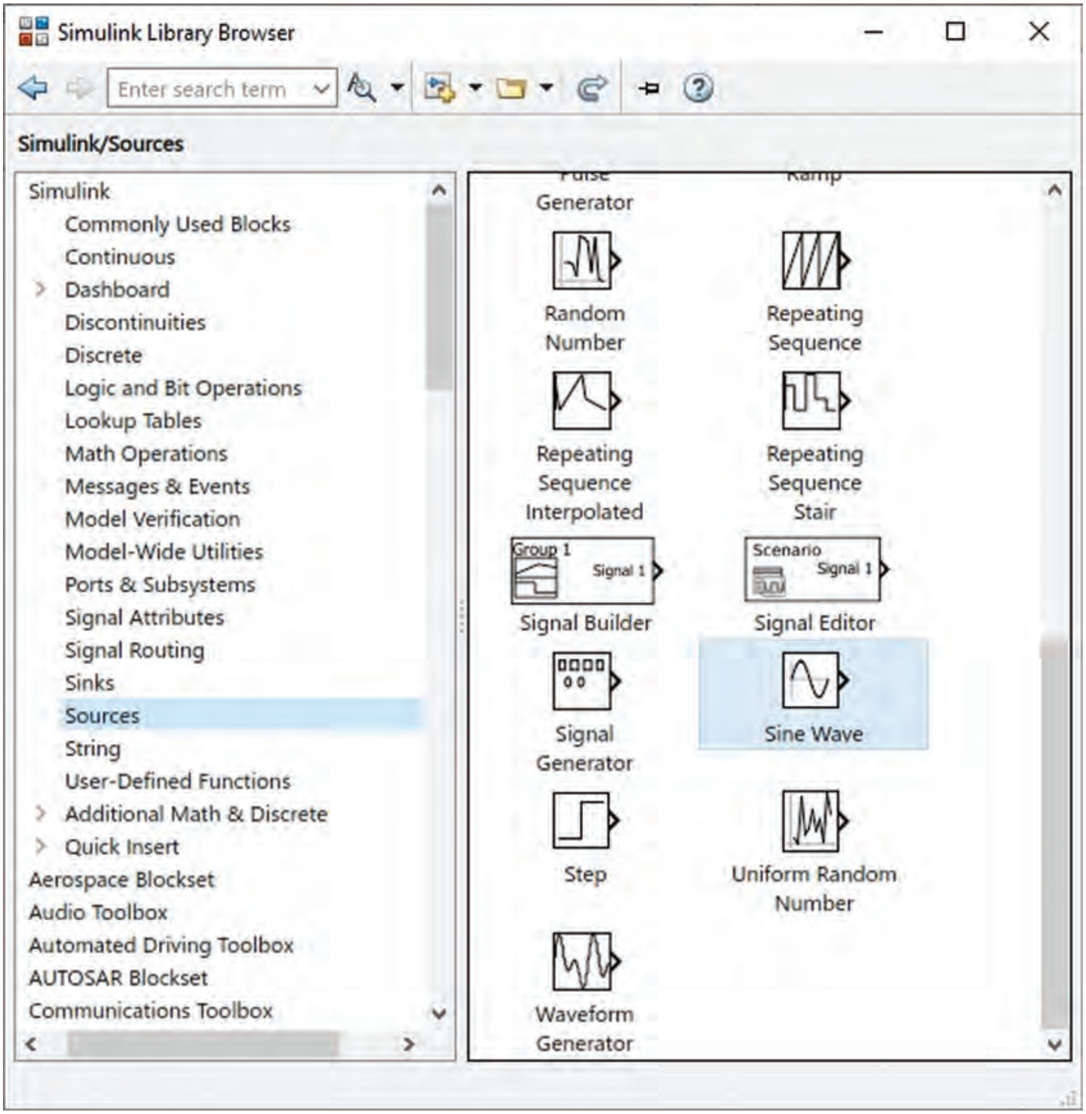

Simulink Library Browser
Enter search term
Simulink/Sources
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Messages & Events
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
String
User-Defined Functions
Additional Math & Discrete
Quick Insert
Aerospace Blockset
Audio Toolbox
Automated Driving Toolbox
AUTOSAR Blockset
Communications Toolbox
Generator
Random Number
Repeating Sequence
Repeating Sequence Interpolated
Repeating Sequence Stair
Group 1
Signal 1
Signal Builder
Scenario
Signal 1
Signal Editor
Signal Generator
Sine Wave
Step
Uniform Random Number
Waveform Generator

Figure 1.64

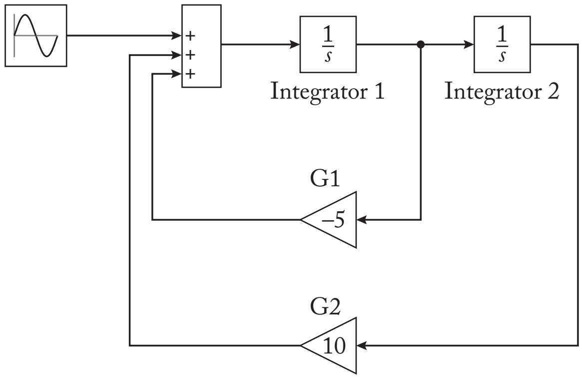


Figure 1.65

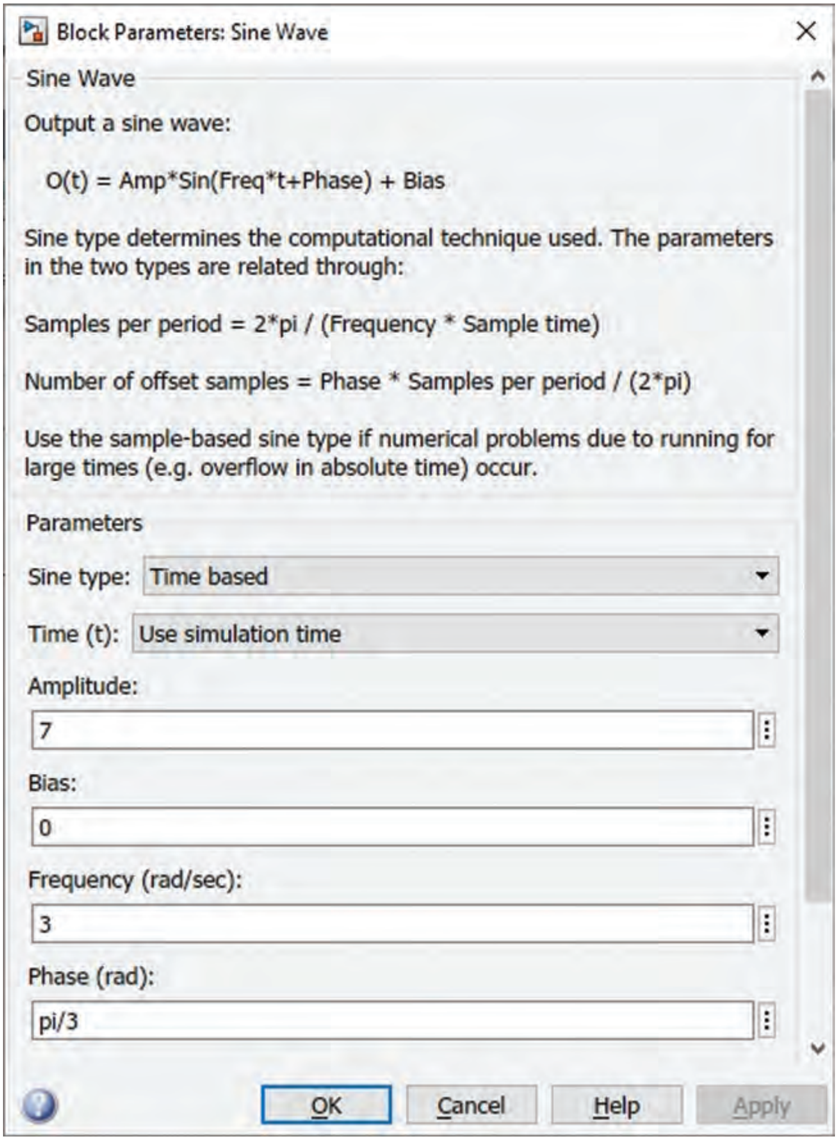

Block Parameters: Sine Wave
Sine Wave
Output a sine wave:
O(t) = Amp*Sin(Freq*t+Phase) + Bias
Sine type determines the computational technique used. The parameters in the two types are related through:
Samples per period = 2*pi / (Frequency * Sample time)
Number of offset samples = Phase * Samples per period / (2*pi)
Use the sample-based sine type if numerical problems due to running for large times (e.g. overflow in absolute time) occur.
Parameters
Sine type:
Time based
Time (t):
Use simulation time
Amplitude:
7
Bias:
0
Frequency (rad/sec):
3
Phase (rad):
pi/3
OK
Cancel
Help
Apply

Figure 1.66

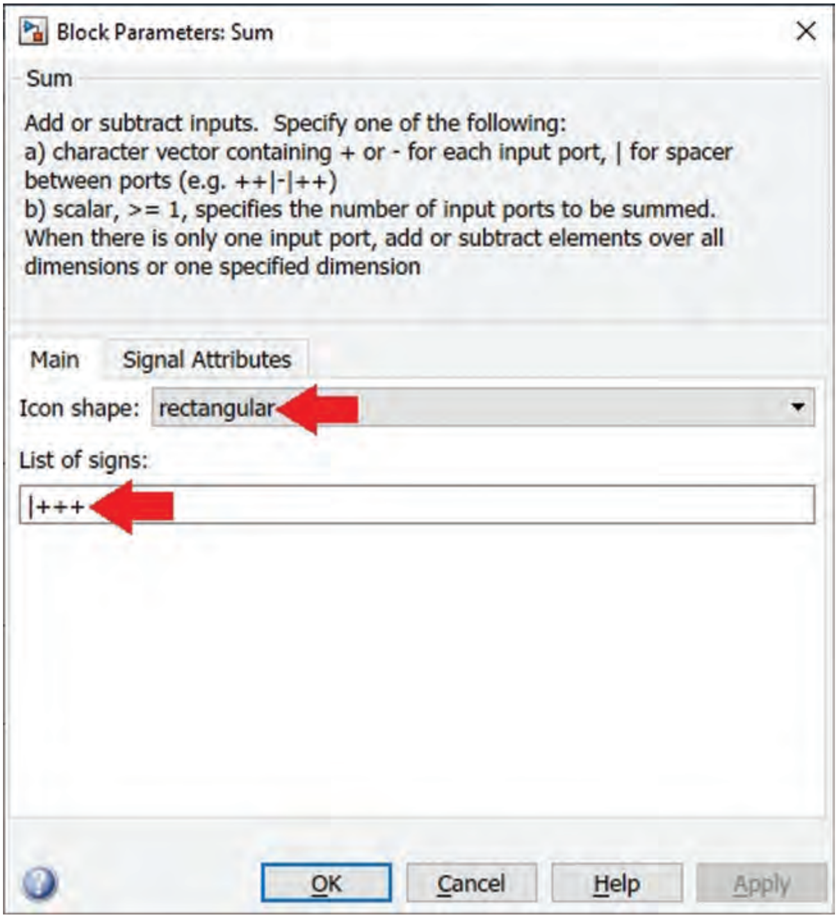

Block Parameters: Sum
Sum
Add or subtract inputs. Specify one of the following:
a) character vector containing + or - for each input port, | for spacer between ports (e.g. ++|-|++)
b) scalar, >= 1, specifies the number of input ports to be summed.
When there is only one input port, add or subtract elements over all dimensions or one specified dimension
Main
Signal Attributes
Icon shape:
rectangular
List of signs:
|+++
OK
Cancel
Help
Apply

Figure 1.67

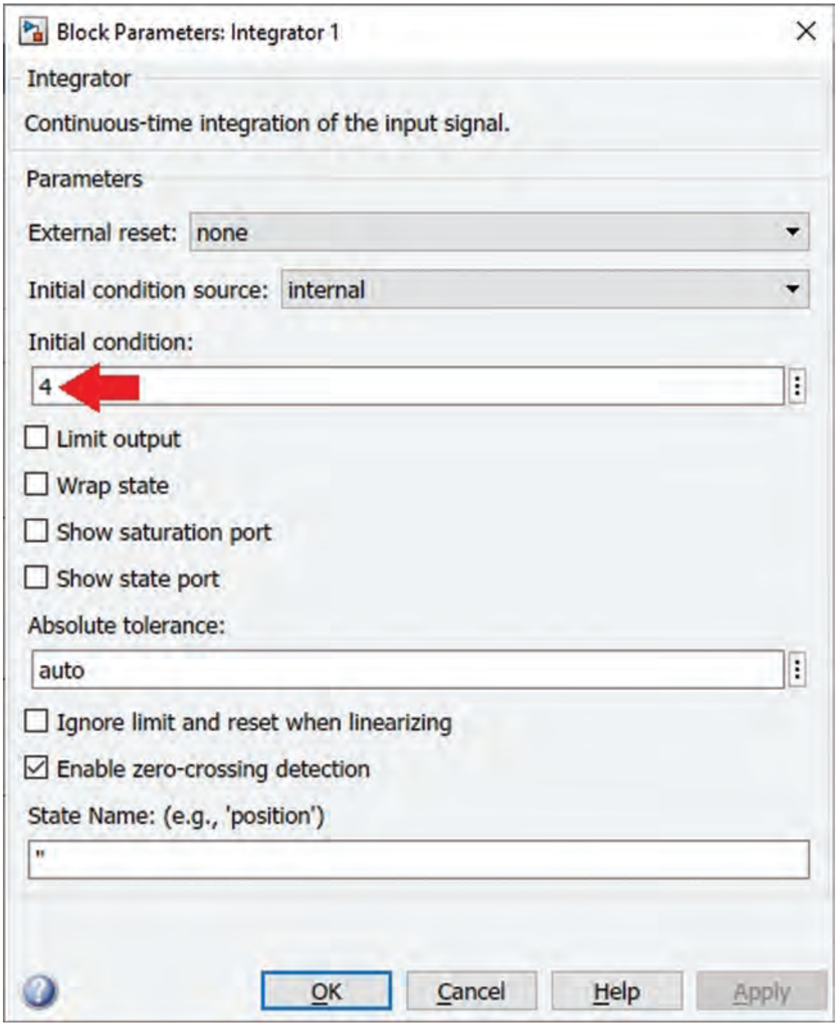

Block Parameters: Integrator 1
Integrator
Continuous-time integration of the input signal.
Parameters
External reset: none
Initial condition source: internal
Initial condition:
4
Limit output
Wrap state
Show saturation port
Show state port
Absolute tolerance:
auto
Ignore limit and reset when linearizing
Enable zero-crossing detection
State Name: (e.g., 'position')
''
OK
Cancel
Help
Apply

Figure 1.68

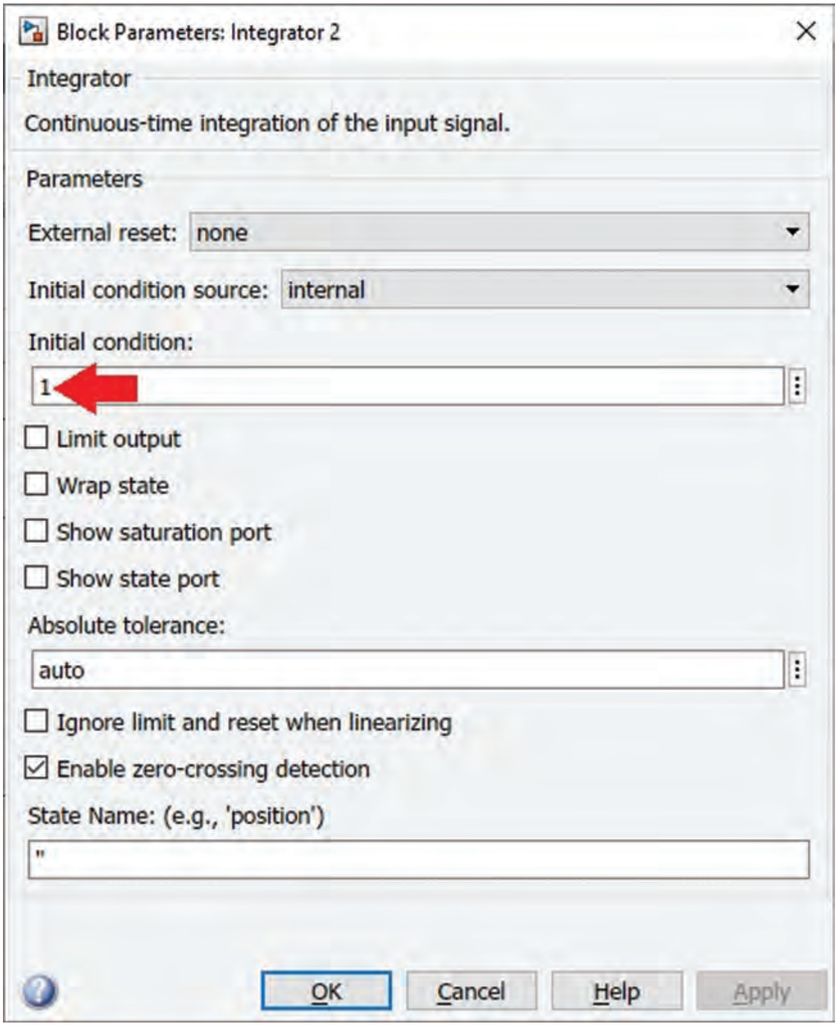

Block Parameters: Integrator 2
Integrator
Continuous-time integration of the input signal.
Parameters
External reset: none
Initial condition source: internal
Initial condition:
1
Limit output
Wrap state
Show saturation port
Show state port
Absolute tolerance:
auto
Ignore limit and reset when linearizing
Enable zero-crossing detection
State Name: (e.g., 'position')
''
OK
Cancel
Help
Apply

Figure 1.69

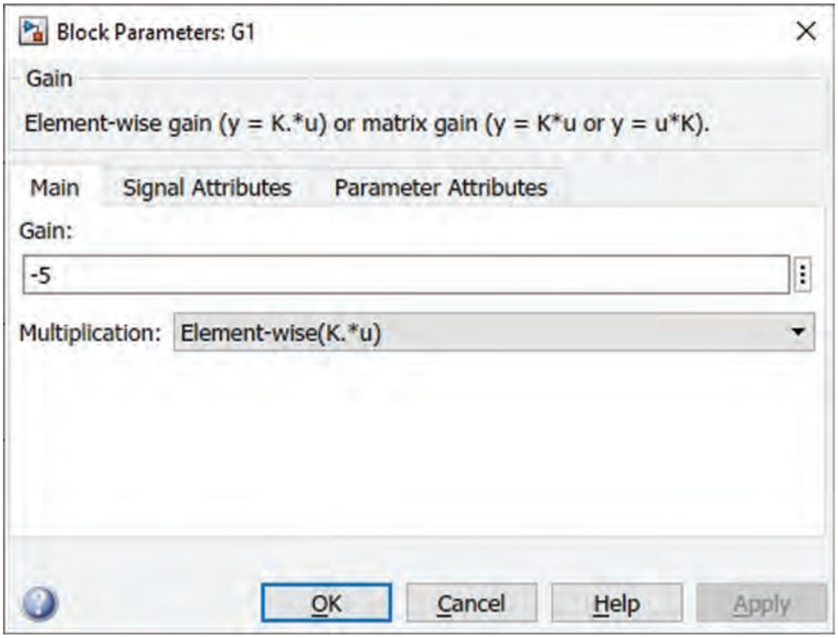


Figure 1.70

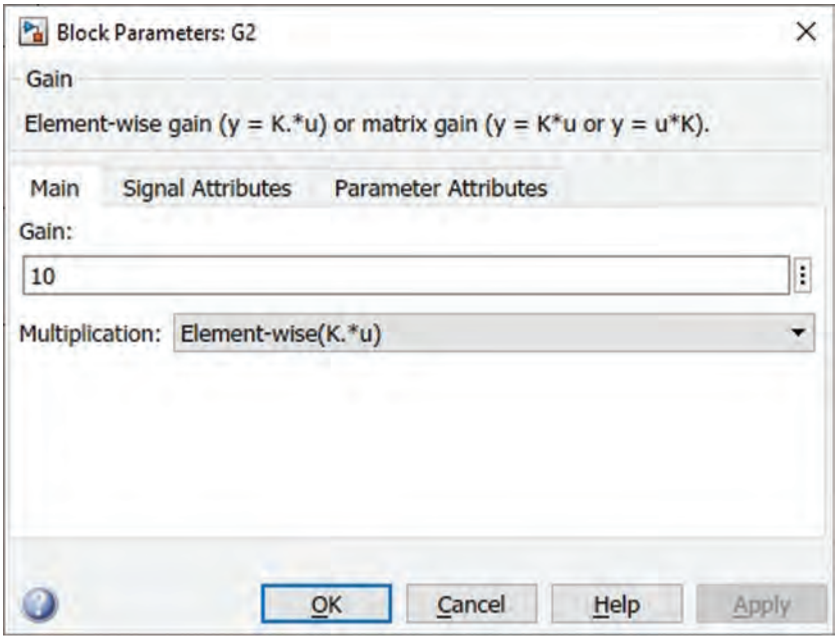


Figure 1.71

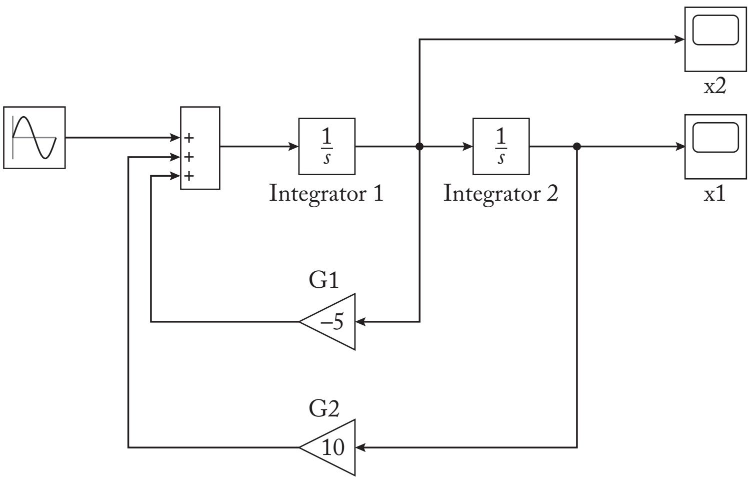


Figure 1.72

Figure 1.73

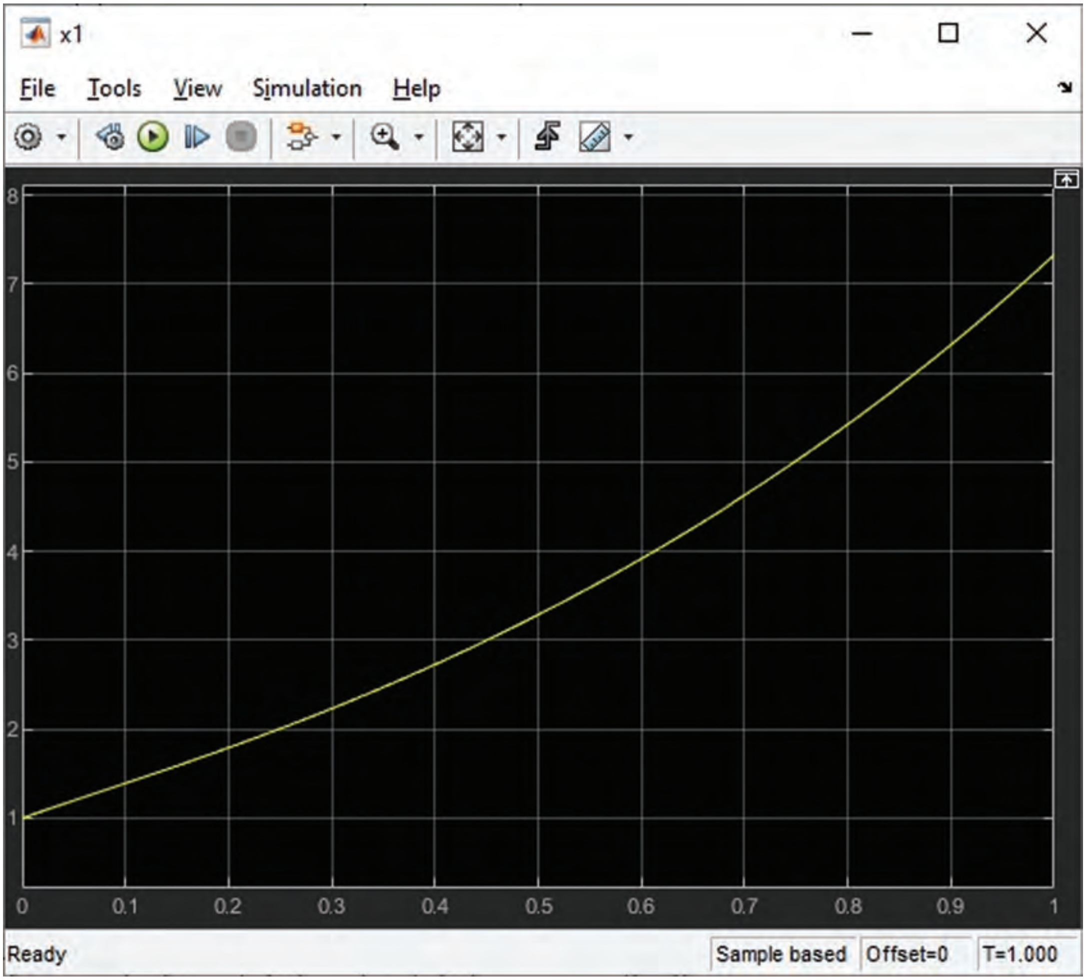

x1
File Tools View Simulation Help
8
7
6
5
4
3
2
1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Ready
Sample based
Offset=0
T=1.000

Figure 1.74

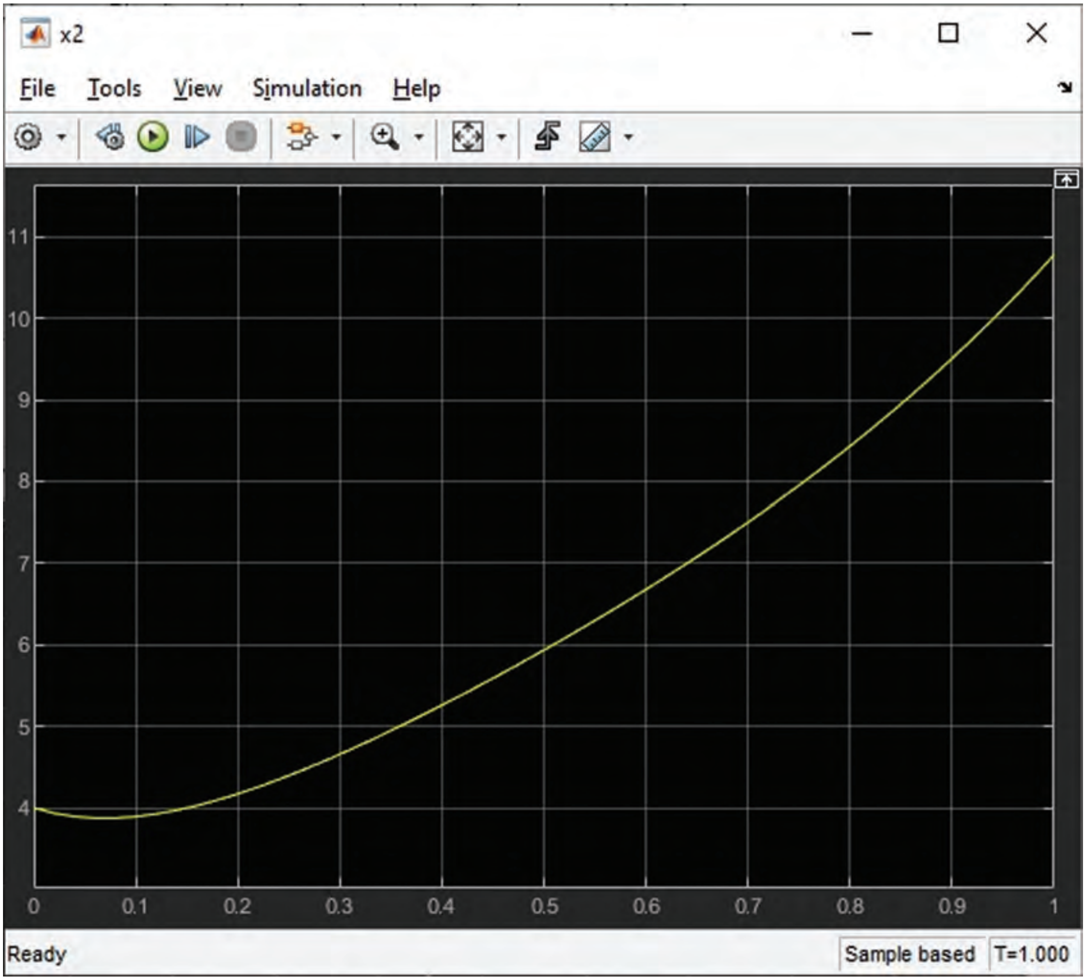

x2
File Tools View Simulation Help
11
10
9
8
7
6
5
4
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Ready
Sample based
T=1.000

Figure 1.75

CHAPTER 2

Introduction to Waijung Blockset

2.1 INTRODUCTION

Waijung blockset is a Simulink® blockset that can be used to easily and automatically generate C code from your Simulink simulation models for many kinds of microcontrollers (Targets). Installation of Waijung blockset is shown in Appendix A.

Waijung 1 Blockset has been designed specifically to support the STM32F4 family of microcontrollers (STM32F4 Target) which is high performance and DSP MCU from ST Microelectronics.

In this book we will use the STM32F407G-DISC1 board to do the experiments.

2.2 EXAMPLE 1: BLINKING THE ON-BOARD LEDS

In this example we want to blink the on-board LED. In order to do this:

1. Right click on the MATLAB icon and click the Run as administrator (Fig. 2.1).

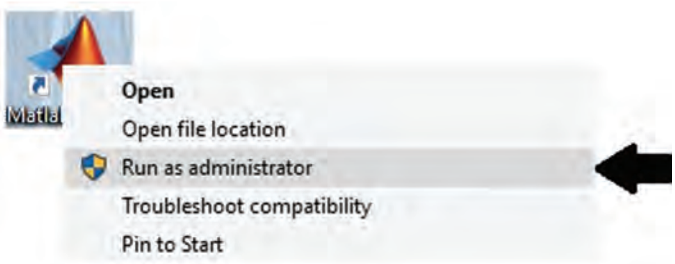


Figure 2.1

2. Make a separate folder for your STM32 projects. In this book we will use the C:\MySTM32Projects. Make a folder with name “1” inside the C:\MySTM32Projects (Fig. 2.2). Files related to the first example will be saved in this folder.

3. Enter into the Simulink environment and save a blank model with the name **blink.slx** into the **C:\MySTM32Projetcs\1** (Fig. 2.3).

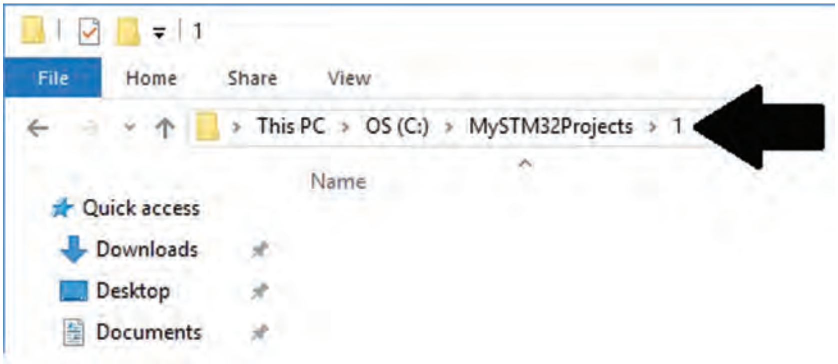


Figure 2.2

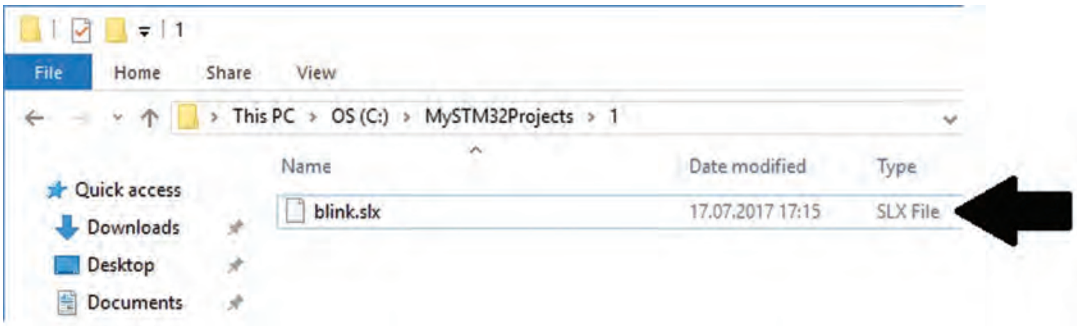


Figure 2.3

4. Add the Target setup (Fig. 2.4), Pulse Generator (Fig. 2.5), Logical Operator (Fig. 2.6), and Digital Output (Fig. 2.7) blocks to the Simulink model (Fig. 2.8).
5. Double click on the Target Setup block. This opens the window shown in Fig. 2.9. Ensure that selected model in the MCU box is the same as the model printed on the microcontroller (Fig. 2.10). There is no need to change other settings in Fig. 2.9.
6. Double click on the Logical Operator block and select the NOT for Operator box (Fig. 2.11).
7. Double click on the Pulse Generator block and do the settings similar to Fig. 2.12. These settings make a square wave with amplitude of 1 and frequency of $1/0.1 = 10$ Hz. The width of high portion of generated signal is Pulse Width (% of period) $\times$ Period (sec) $= 0.5 \times 0.1$ s $= 50$ msec. The width of low portion of generated signal is (1-Pulse Width (% of period)) $\times$ Period (sec) $= (1 - 0.5) \times 0.1$ s $= 50$ msec.
8. Connect the blocks together (Fig. 2.13) and press the Ctrl+S to save the changes.

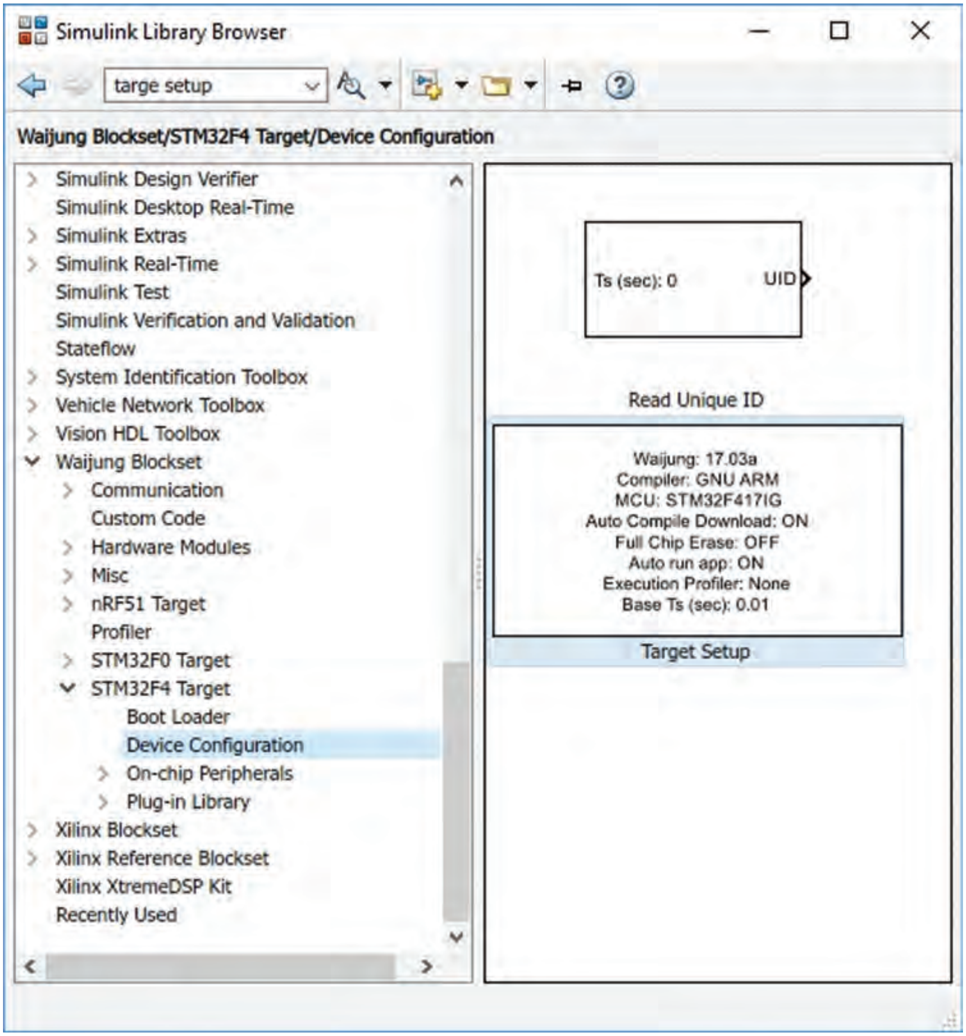

Simulink Library Browser
targe setup
Waijung Blockset/STM32F4 Target/Device Configuration
Simulink Design Verifier
Simulink Desktop Real-Time
Simulink Extras
Simulink Real-Time
Simulink Test
Simulink Verification and Validation
Stateflow
System Identification Toolbox
Vehicle Network Toolbox
Vision HDL Toolbox
Waijung Blockset
Communication
Custom Code
Hardware Modules
Misc
nRF51 Target
Profiler
STM32F0 Target
STM32F4 Target
Boot Loader
Device Configuration
On-chip Peripherals
Plug-in Library
Xilinx Blockset
Xilinx Reference Blockset
Xilinx XtremeDSP Kit
Recently Used
Ts (sec): 0
UID
Read Unique ID
Waijung: 17.03a
Compiler: GNU ARM
MCU: STM32F417IG
Auto Compile Download: ON
Full Chip Erase: OFF
Auto run app: ON
Execution Profiler: None
Base Ts (sec): 0.01
Target Setup

Figure 2.4

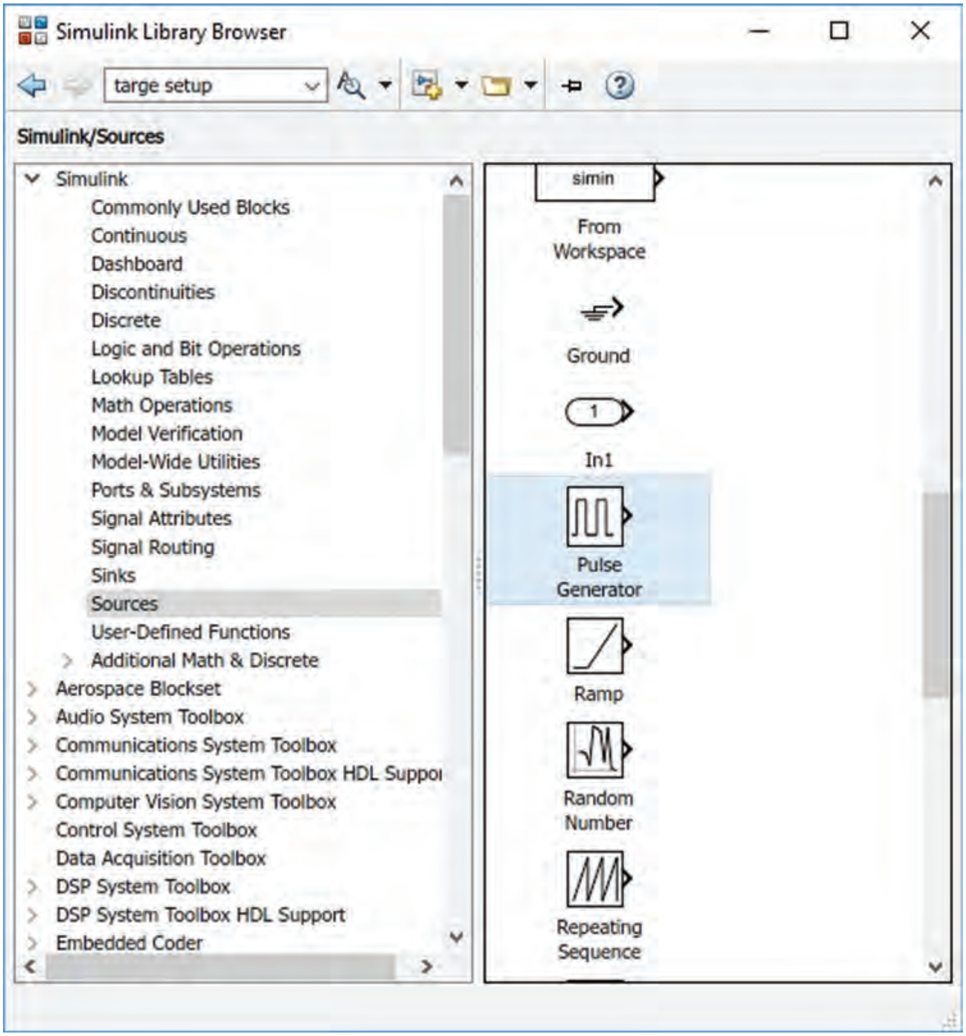

Simulink Library Browser
targe setup
Simulink/Sources
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
User-Defined Functions
Additional Math & Discrete
Aerospace Blockset
Audio System Toolbox
Communications System Toolbox
Communications System Toolbox HDL Suppo
Computer Vision System Toolbox
Control System Toolbox
Data Acquisition Toolbox
DSP System Toolbox
DSP System Toolbox HDL Support
Embedded Coder
simin
From Workspace
Ground
1
In1
Pulse Generator
Ramp
Random Number
Repeating Sequence

Figure 2.5

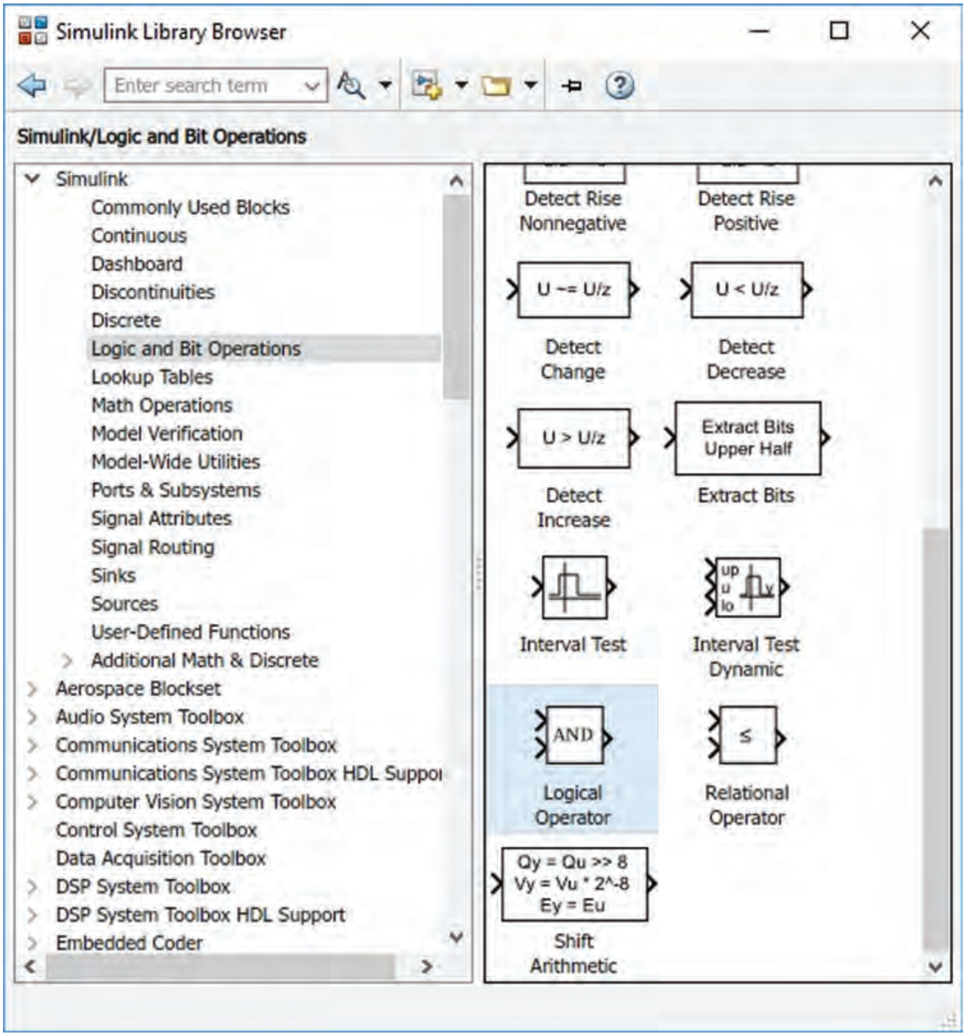

Simulink Library Browser
Enter search term
Simulink/Logic and Bit Operations
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
User-Defined Functions
Additional Math & Discrete
Aerospace Blockset
Audio System Toolbox
Communications System Toolbox
Communications System Toolbox HDL Suppo
Computer Vision System Toolbox
Control System Toolbox
Data Acquisition Toolbox
DSP System Toolbox
DSP System Toolbox HDL Support
Embedded Coder
Detect Rise Nonnegative
Detect Rise Positive
U ~= U/z
Detect Change
U < U/z
Detect Decrease
U > U/z
Detect Increase
Extract Bits Upper Half
Extract Bits
Interval Test
up
u
lo
y
Interval Test Dynamic
AND
Logical Operator
≤
Relational Operator
Qy = Qu >> 8
Vy = Vu * 2^-8
Ey = Eu
Shift Arithmetic

Figure 2.6

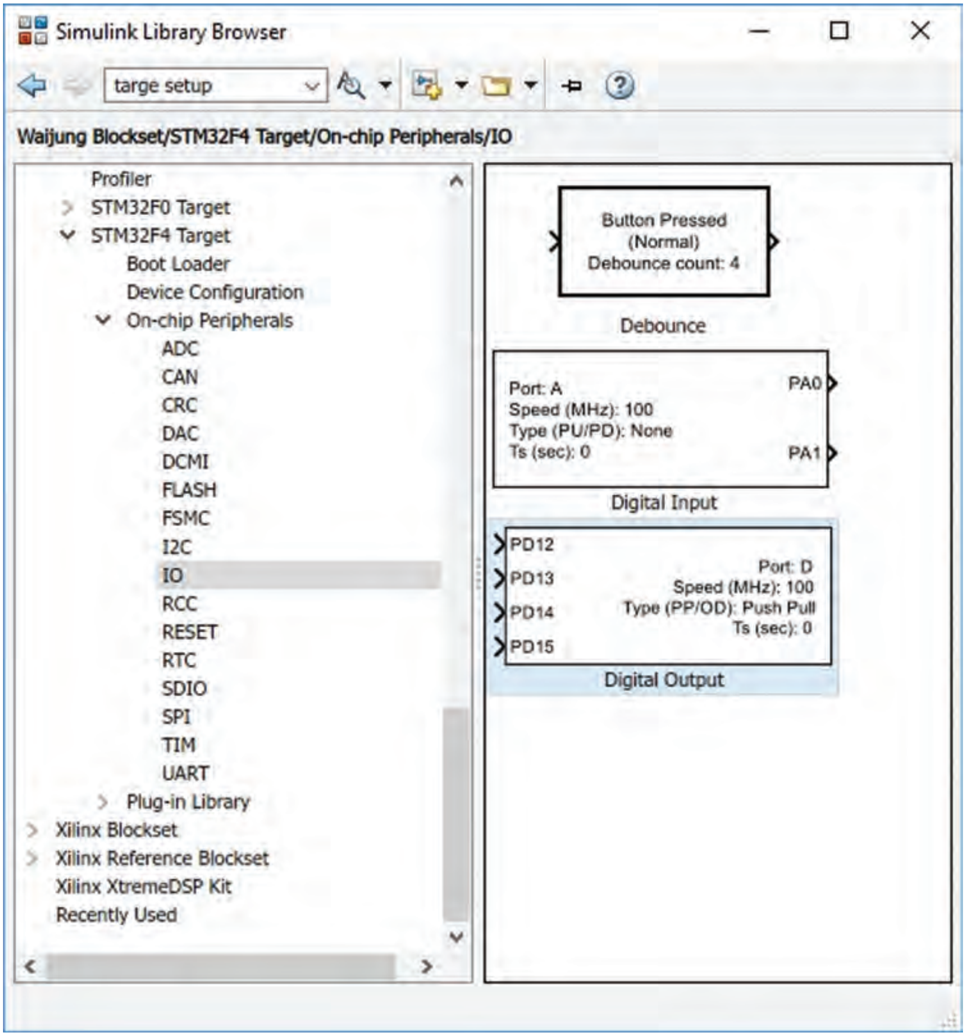

Simulink Library Browser
targe setup
Waijung Blockset/STM32F4 Target/On-chip Peripherals/IO
Profiler
STM32F0 Target
STM32F4 Target
Boot Loader
Device Configuration
On-chip Peripherals
ADC
CAN
CRC
DAC
DCMI
FLASH
FSMC
I2C
IO
RCC
RESET
RTC
SDIO
SPI
TIM
UART
Plug-in Library
Xilinx Blockset
Xilinx Reference Blockset
Xilinx XtremeDSP Kit
Recently Used
Button Pressed
(Normal)
Debounce count: 4
Debounce
Port: A
Speed (MHz): 100
Type (PU/PD): None
Ts (sec): 0
PA0
PA1
Digital Input
PD12
PD13
PD14
PD15
Port: D
Speed (MHz): 100
Type (PP/OD): Push Pull
Ts (sec): 0
Digital Output

Figure 2.7

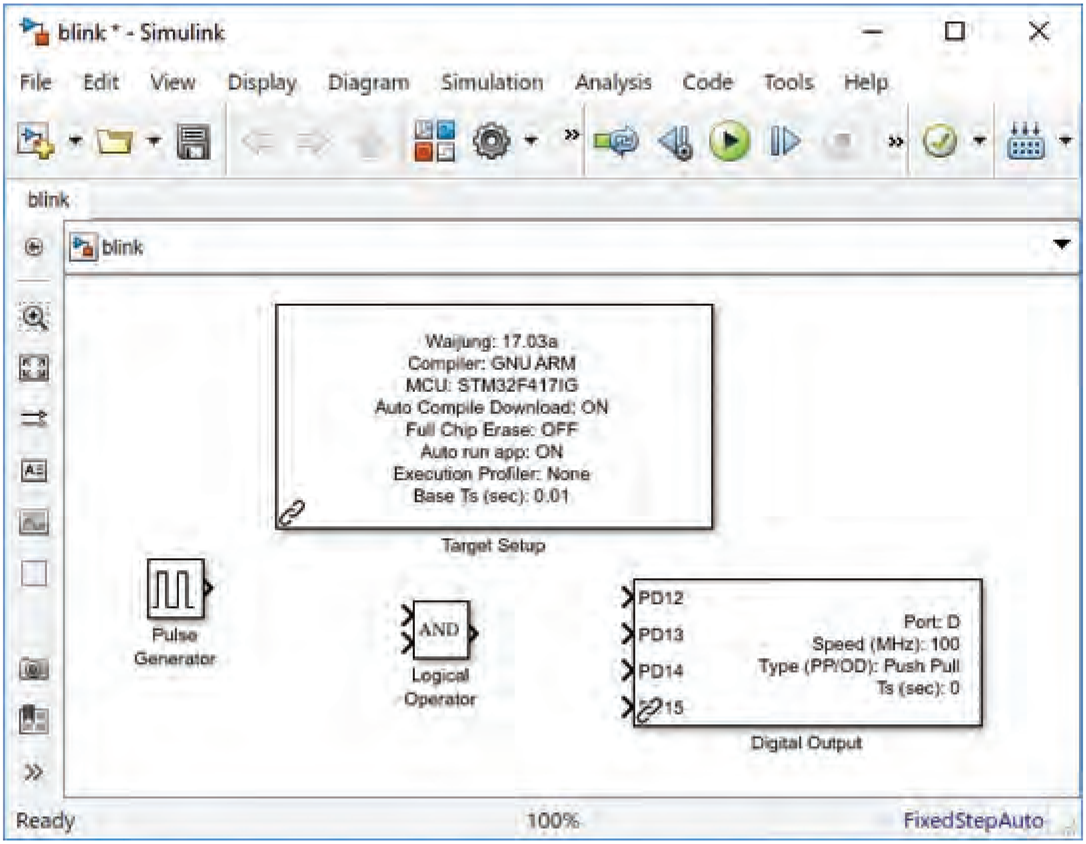

blink * - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
blink
blink
Waijung: 17.03a
Compiler: GNU ARM
MCU: STM32F417IG
Auto Compile Download: ON
Full Chip Erase: OFF
Auto run app: ON
Execution Profiler: None
Base Ts (sec): 0.01
Target Setup
Pulse Generator
AND
Logical Operator
PD12
PD13
PD14
15
Port: D
Speed (MHz): 100
Type (PP/OD): Push Pull
Ts (sec): 0
Digital Output
Ready
100%
FixedStepAuto

Figure 2.8

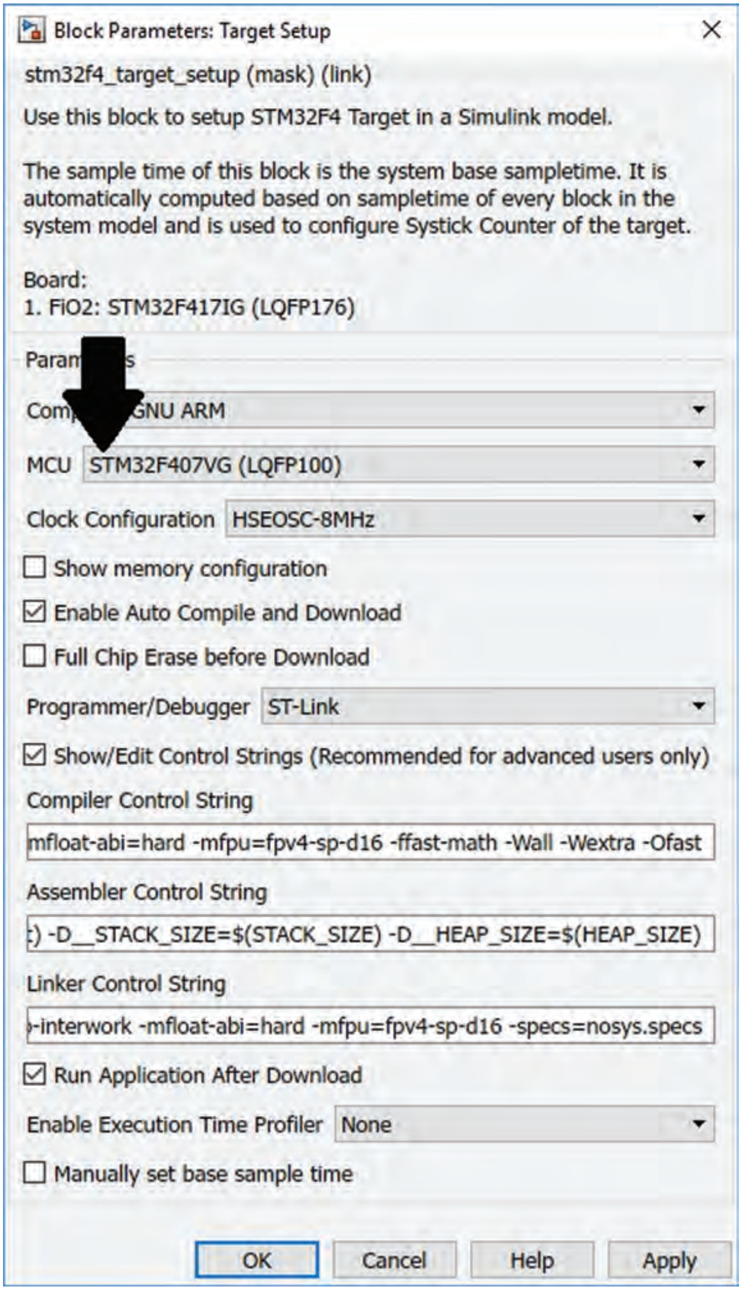

Block Parameters: Target Setup
stm32f4_target_setup (mask) (link)
Use this block to setup STM32F4 Target in a Simulink model.
The sample time of this block is the system base sampletime. It is automatically computed based on sampletime of every block in the system model and is used to configure Systick Counter of the target.
Board:
1. FiO2: STM32F417IG (LQFP176)
GNU ARM
MCU
STM32F407VG (LQFP100)
Clock Configuration
HSEOSC-8MHz
Show memory configuration
Enable Auto Compile and Download
Full Chip Erase before Download
Programmer/Debugger
ST-Link
Show/Edit Control Strings (Recommended for advanced users only)
Compiler Control String
mfloat-abi=hard -mfpu=fpv4-sp-d16 -ffast-math -Wall -Wextra -Ofast
Assembler Control String
) -D__STACK_SIZE=$(STACK_SIZE) -D__HEAP_SIZE=$(HEAP_SIZE)
Linker Control String
-interwork -mfloat-abi=hard -mfpu=fpv4-sp-d16 -specs=nosys.specs
Run Application After Download
Enable Execution Time Profiler
None
Manually set base sample time
OK
Cancel
Help
Apply

Figure 2.9

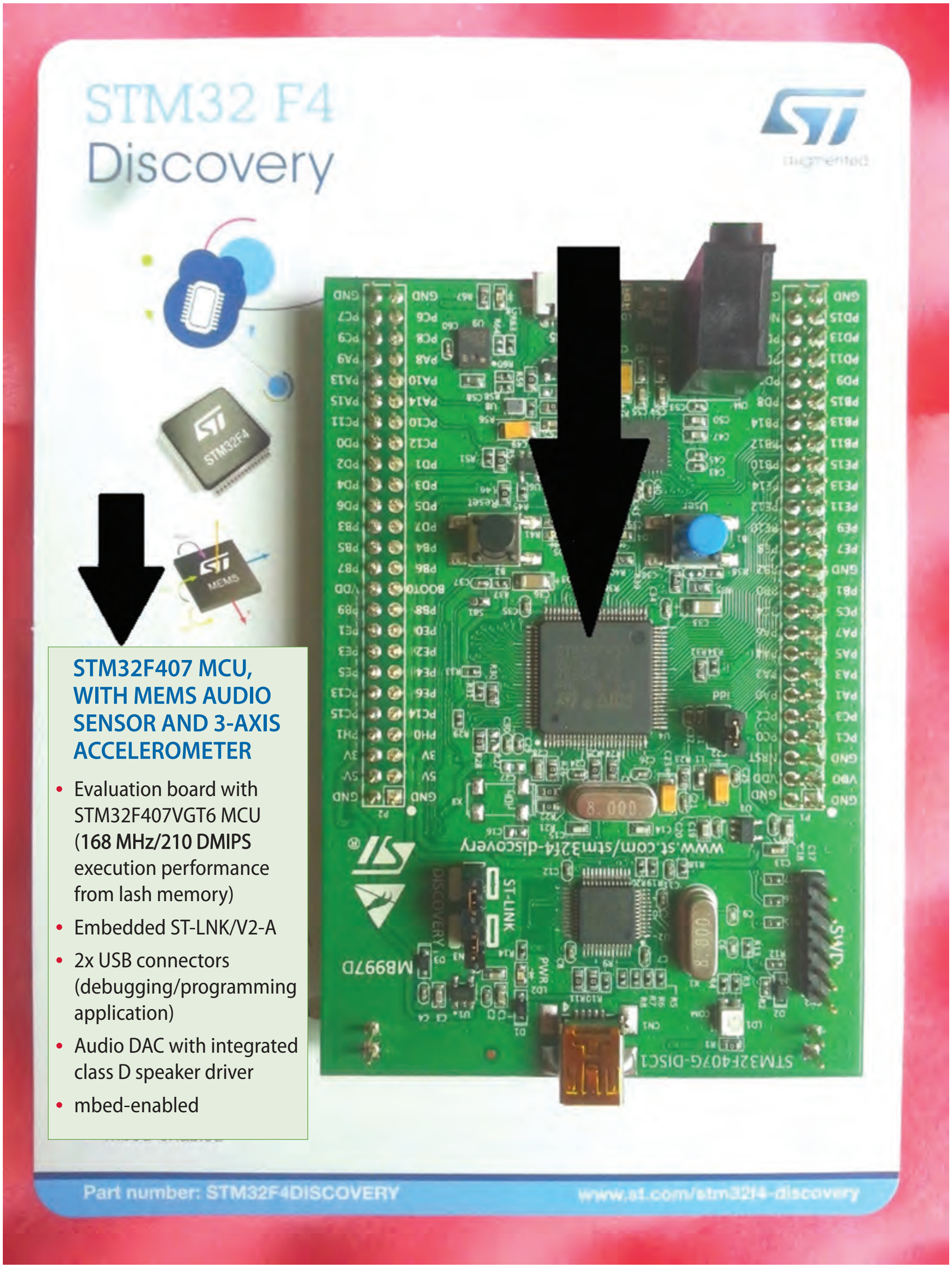

STM32 F4
Discovery
STM32F407 MCU, WITH MEMS AUDIO SENSOR AND 3-AXIS ACCELEROMETER
• Evaluation board with STM32F407VGT6 MCU (168 MHz/210 DMIPS execution performance from lash memory)
• Embedded ST-LNK/V2-A
• 2x USB connectors (debugging/programming application)
• Audio DAC with integrated class D speaker driver
• mbed-enabled
Part number: STM32F4DISCOVERY
www.st.com/stm32f4-discovery

Figure 2.10

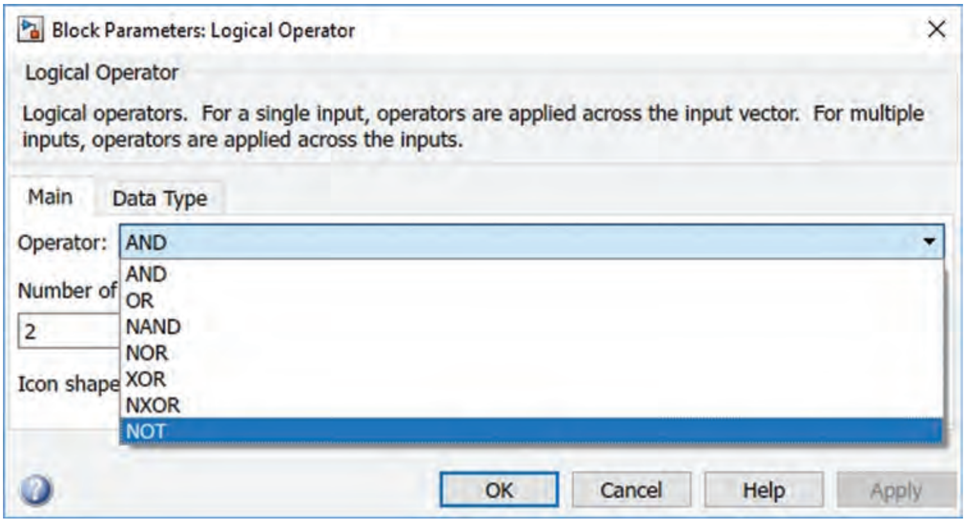

Block Parameters: Logical Operator
Logical Operator
Logical operators. For a single input, operators are applied across the input vector. For multiple inputs, operators are applied across the inputs.
Main
Data Type
Operator:
AND
AND
OR
NAND
NOR
XOR
NXOR
NOT
Number of
2
Icon shape
OK
Cancel
Help
Apply

Figure 2.11

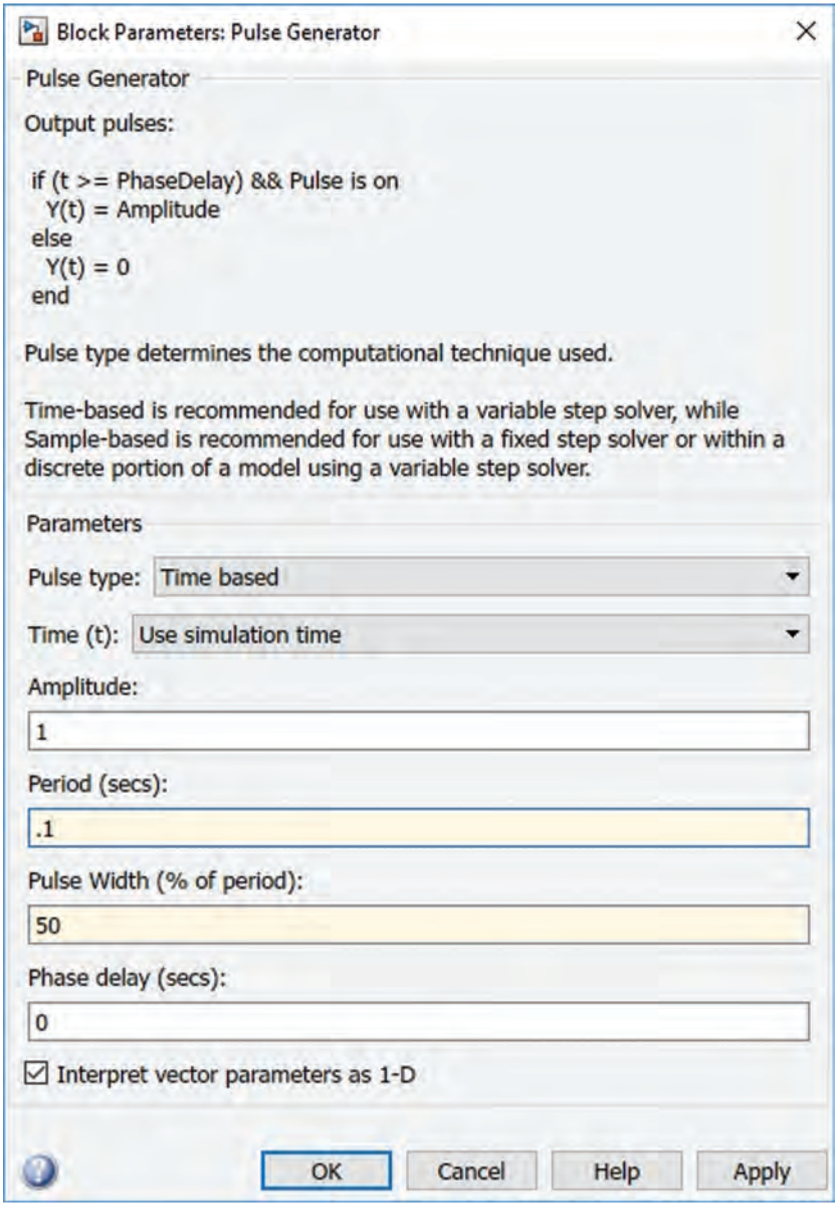

Block Parameters: Pulse Generator
Pulse Generator
Output pulses:
if (t >= PhaseDelay) && Pulse is on
Y(t) = Amplitude
else
Y(t) = 0
end
Pulse type determines the computational technique used.
Time-based is recommended for use with a variable step solver, while Sample-based is recommended for use with a fixed step solver or within a discrete portion of a model using a variable step solver.
Parameters
Pulse type:
Time based
Time (t):
Use simulation time
Amplitude:
1
Period (secs):
.1
Pulse Width (% of period):
50
Phase delay (secs):
0
Interpret vector parameters as 1-D
OK
Cancel
Help
Apply

Figure 2.12

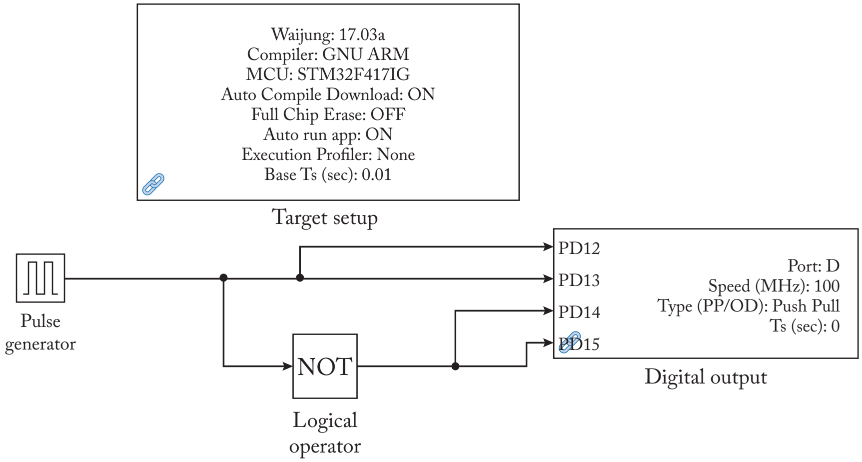


Figure 2.13

9. Use the Browse for folder icon (Fig. 2.14) to open the C:\MySTM32Projects\1 (Fig. 2.15).

Figure 2.14

Note that the Current Folder path (Fig. 2.15) must be the same as the path you saved the Simulink file (Fig. 2.3), otherwise you will receive an error (Fig. 2.16) when you want to compile your model.

10. Connect the Discovery board to the computer.

11. Click the Build Model icon (Fig. 2.17).

12. The window shown in Fig. 2.18 appears once the compile process is finished successfully. This window shows that Discovery board is programmed successfully.

    Now, the on-board LEDs must start to blink.

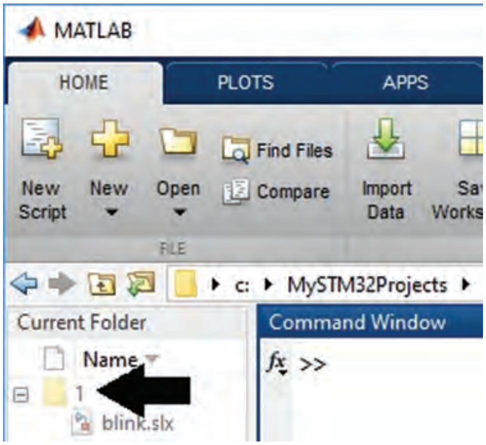


Figure 2.15

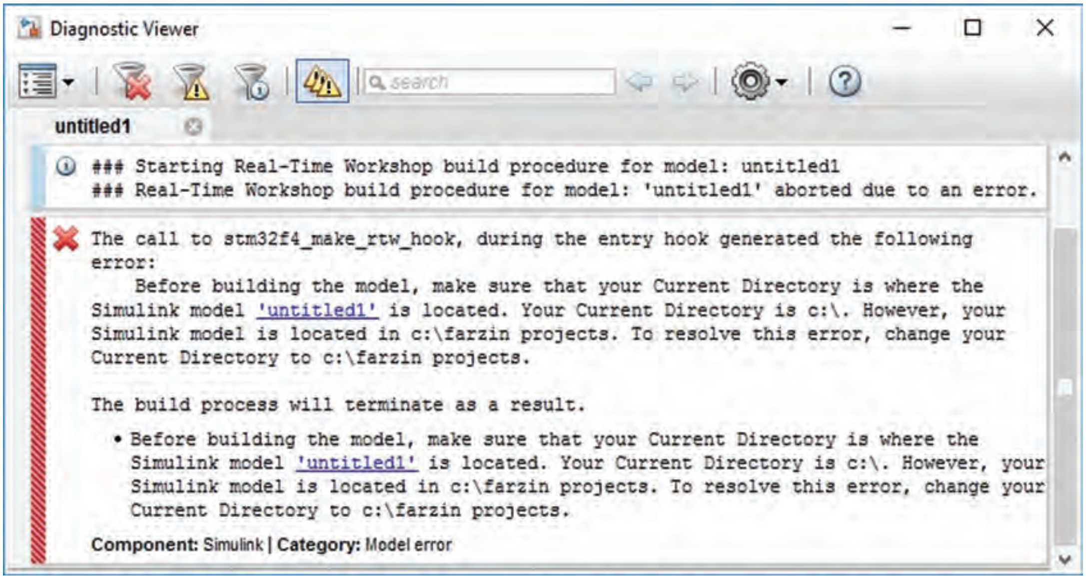


Figure 2.16

Figure 2.17

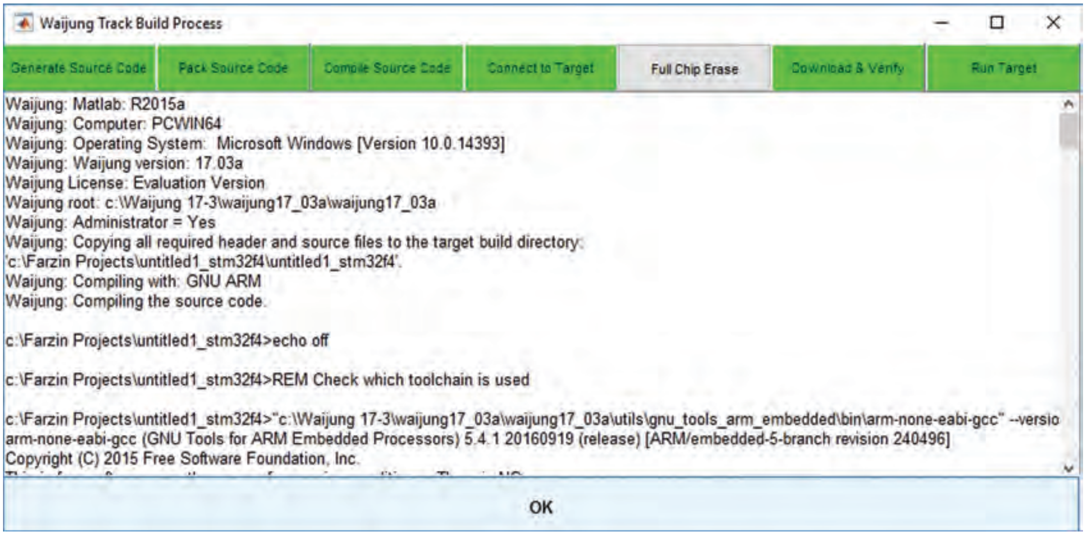


Figure 2.18

13. The Waijung added two folders to the C:\MySTM32Projects\1: slprj and blink_stm32f4 (Fig. 2.19). The generated hex file for the drawn Simulink model is in the blink_stm32f4 folder (Fig. 2.20).

2.2.1 MANUAL PROGRAMMING OF THE BOARD

The Waijung blockset automatically program the Discovery board after clicking the Build Model icon (Fig. 2.17). You can upload the generated hex file to the board manually if, for any reason, the Waijung blockset didn't upload the hex file to the board. In order to manually upload the hex file to the Discovery board.

1. Run the STM32 ST-LINK Utility (Fig. 2.21). This opens the window shown in Fig. 2.22.
2. Use the File> Open file… (Fig. 2.23) to open the hex file.
3. Connect the Discovery board to the computer.

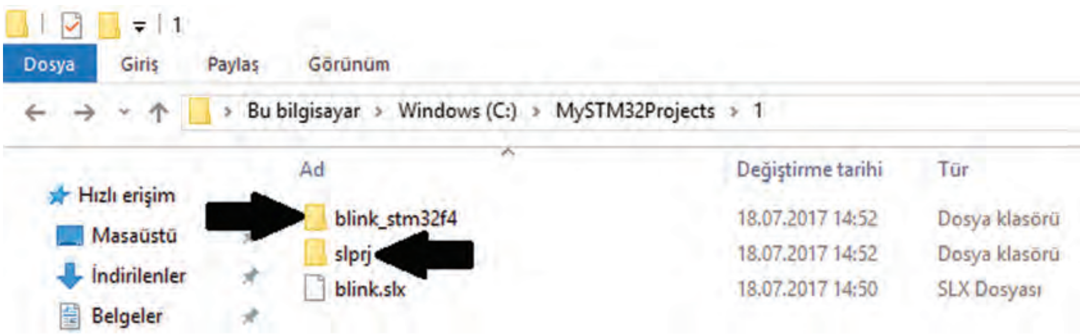


Figure 2.19

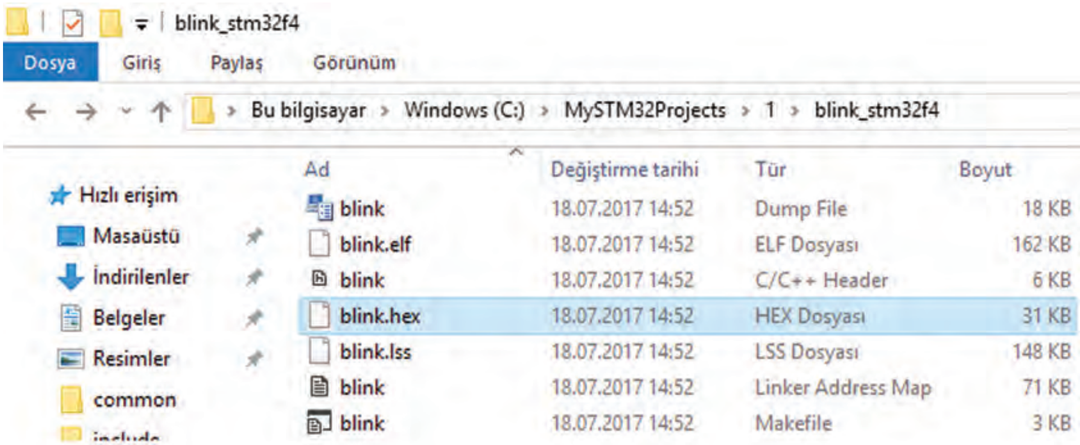


Figure 2.20

Figure 2.21

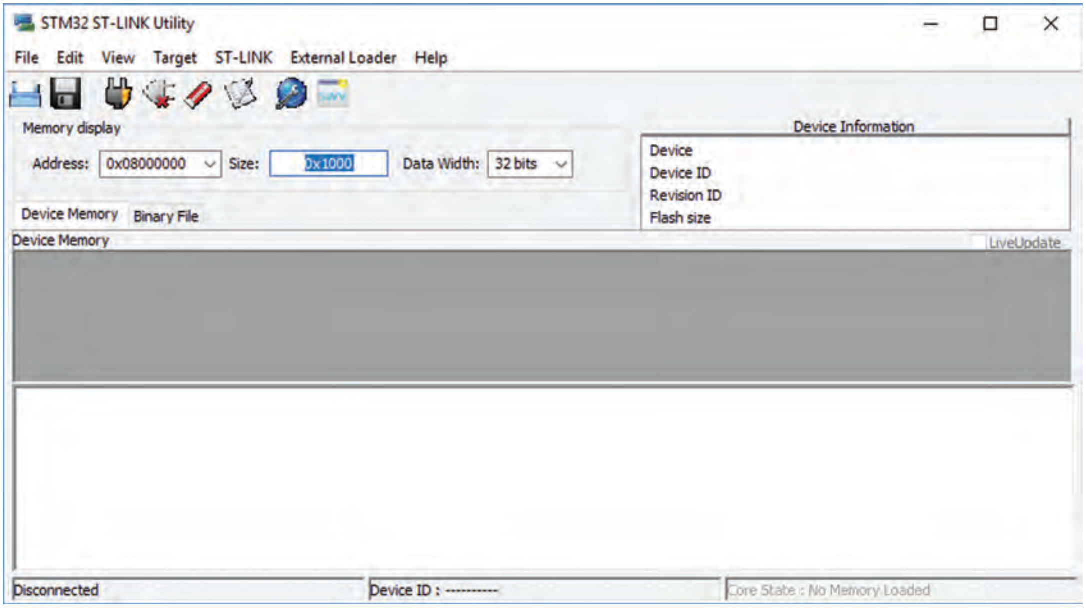


Figure 2.22

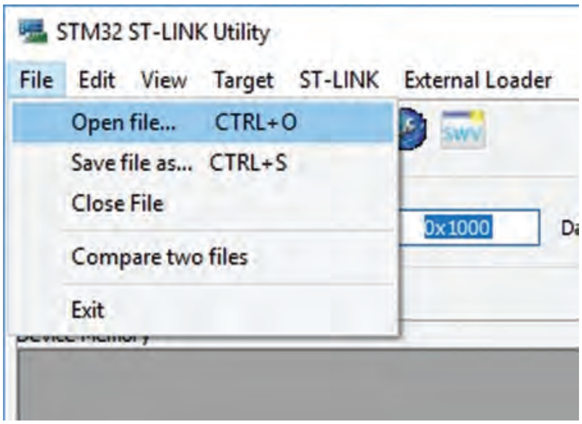


Figure 2.23

4. Click the Target> Program... or Target> Program & Verify... to program the Discovery board (Fig. 2.24).

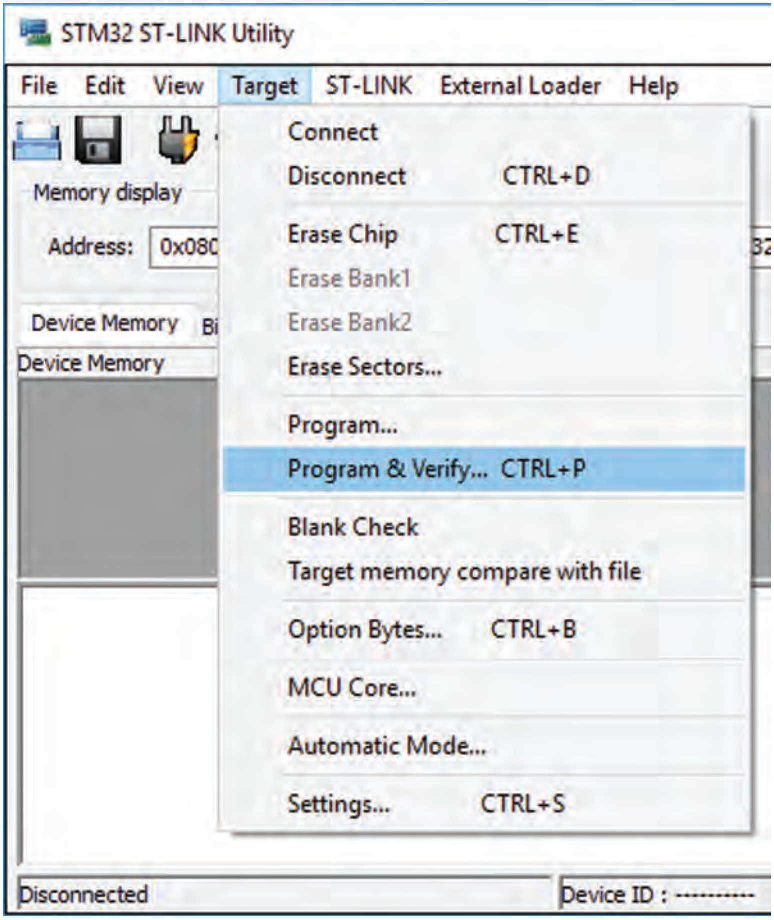


Figure 2.24

2.3 EXAMPLE 2: READING DIGITAL INPUTS

Simulink model of this example is shown in Fig. 2.25. In this example, when you press the on-board push button, the green LED which is connected to PD12 turns on. When you release the button, the orange LED which is connected to PD13 turns on. Settings of digital input and digital output blocks in Fig. 2.25 are shown in Figs. 2.26 and 2.27, respectively. In Fig. 2.26, Port A and Pin 0 are selected. So, PA0 is defined as input. In Fig. 2.27, Port D and Pins 12 and 13 are selected. So, PD12 and PD13 are defined as output.

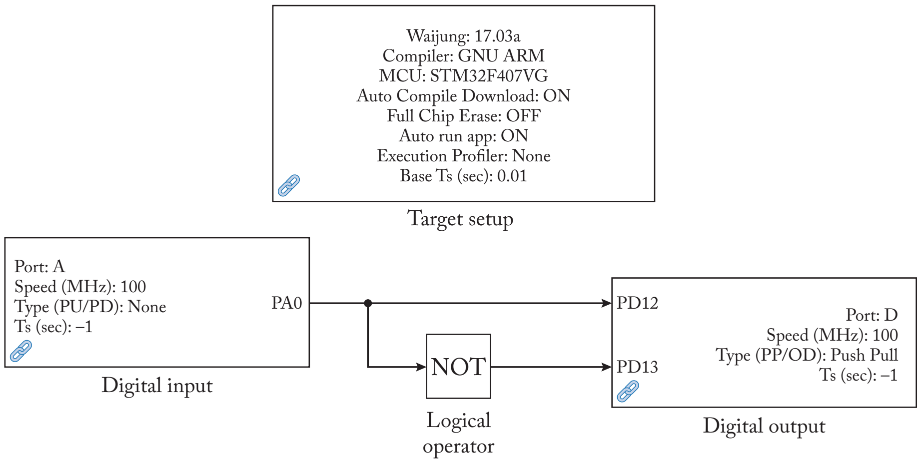

Waijung: 17.03a
Compiler: GNU ARM
MCU: STM32F407VG
Auto Compile Download: ON
Full Chip Erase: OFF
Auto run app: ON
Execution Profiler: None
Base Ts (sec): 0.01
Target setup
Port: A
Speed (MHz): 100
Type (PU/PD): None
Ts (sec): –1
PA0
Digital input
NOT
Logical operator
PD12
PD13
Port: D
Speed (MHz): 100
Type (PP/OD): Push Pull
Ts (sec): –1
Digital output

Figure 2.25

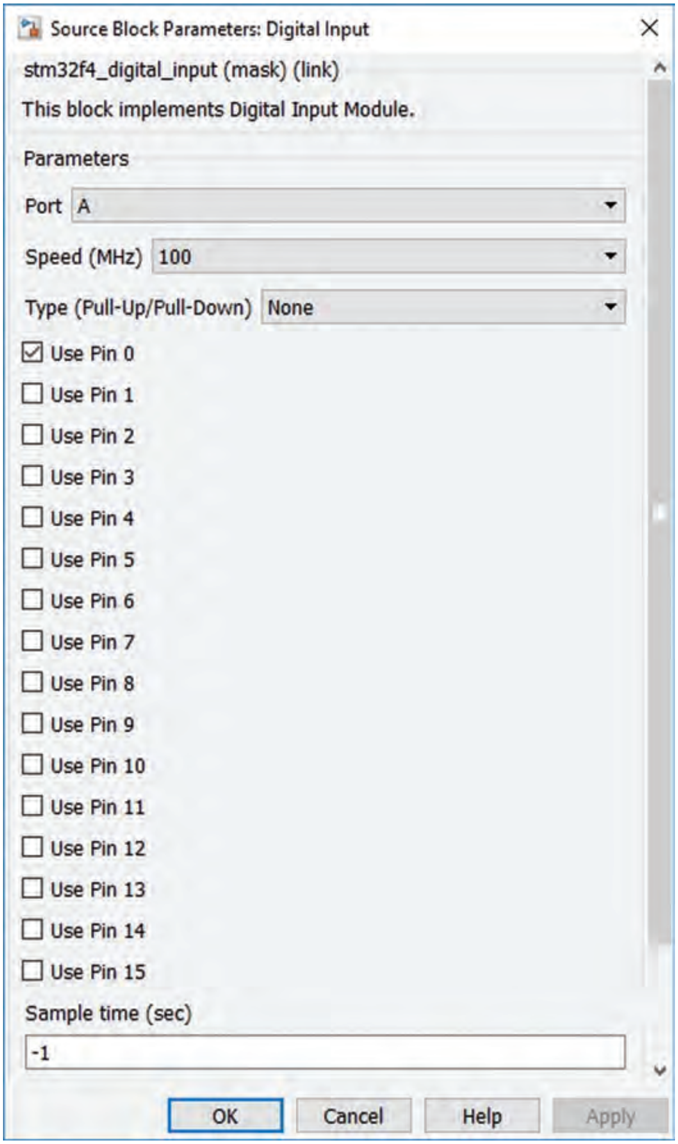

Source Block Parameters: Digital Input
stm32f4_digital_input (mask) (link)
This block implements Digital Input Module.
Parameters
Port A
Speed (MHz) 100
Type (Pull-Up/Pull-Down) None
Use Pin 0
Use Pin 1
Use Pin 2
Use Pin 3
Use Pin 4
Use Pin 5
Use Pin 6
Use Pin 7
Use Pin 8
Use Pin 9
Use Pin 10
Use Pin 11
Use Pin 12
Use Pin 13
Use Pin 14
Use Pin 15
Sample time (sec)
-1
OK
Cancel
Help
Apply

Figure 2.26

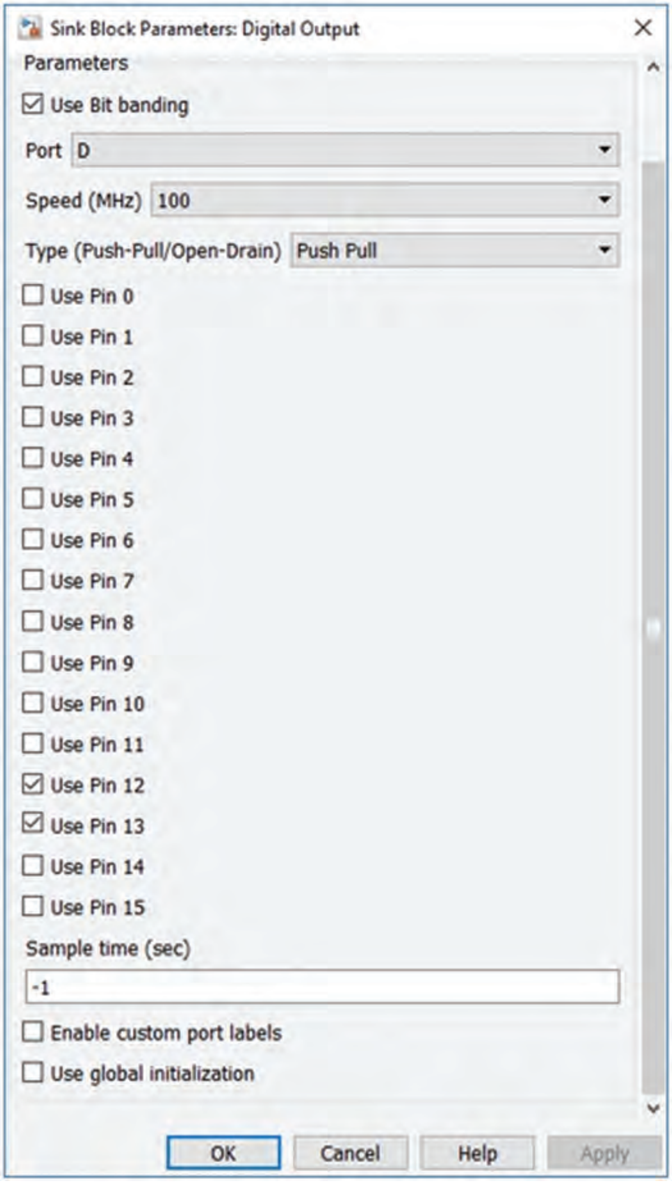

Sink Block Parameters: Digital Output
Parameters
Use Bit banding
Port D
Speed (MHz) 100
Type (Push-Pull/Open-Drain) Push Pull
Use Pin 0
Use Pin 1
Use Pin 2
Use Pin 3
Use Pin 4
Use Pin 5
Use Pin 6
Use Pin 7
Use Pin 8
Use Pin 9
Use Pin 10
Use Pin 11
Use Pin 12
Use Pin 13
Use Pin 14
Use Pin 15
Sample time (sec)
-1
Enable custom port labels
Use global initialization
OK
Cancel
Help
Apply

Figure 2.27

Figure 2.28 shows two ways to connect a button to a microcontroller pin. The capacitor in this circuit solves the bouncing problem of mechanical switches. In Fig. 2.28(A), the pin reads 0 when the button is not pressed. When the button is pressed, the pin reads 1. In Fig. 2.28(B), the pin reads 1 when the button is not pressed. When the button is pressed, the pin reads 0.

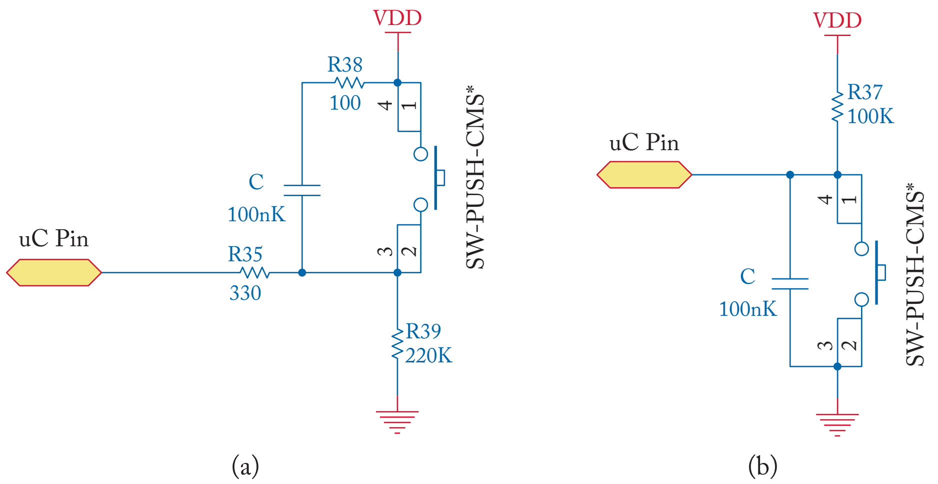


Figure 2.28

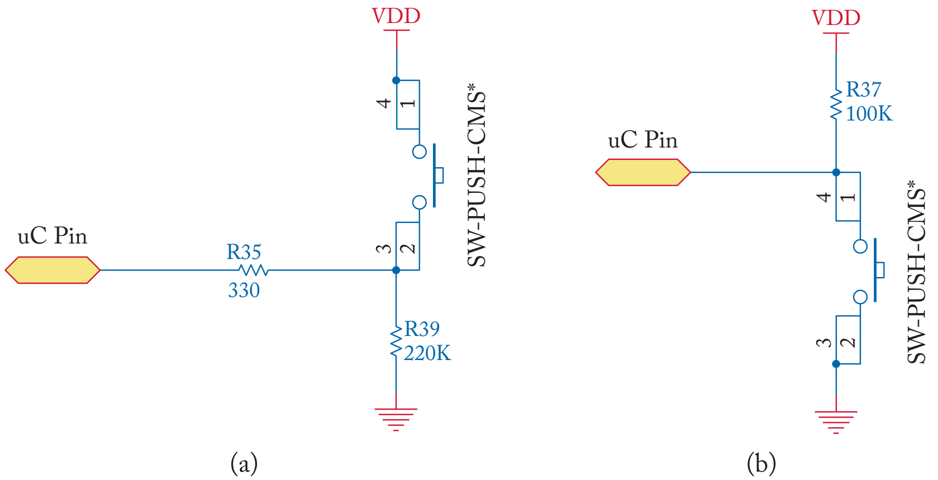


Figure 2.29

You can use the circuit shown in Fig. 2.29 as well. In this circuit, the bouncing problem is solved with the aid of the Debounce block shown in Fig. 2.30. The required Simulink model is shown in Fig. 2.31. In fact, the Debounce block is a time delay block. Presence of time delay permits the switch contacts to reach steady state before being read. The amount of time delay is set with the aid of the Prescale (Debounce count) drop-down list (Fig. 2.32). Bigger numbers in this list generate bigger delays. A number between 8–32 is suitable for most applications.

2.4 EXAMPLE 3: DETERMINING THE HIGH AND LOW VOLTAGE LEVELS FOR INPUT/OUTPUT

Simulink model of this example is shown in Fig. 2.33. The waveform shown in Fig. 2.34 is applied to Pin PA1 and its digital version is taken from output PA2. The waveform of input and output are shown in Fig. 2.35. The diagram shown in Fig. 2.36 can be drawn based on the obtained waveform. When the input is bigger than 1.7 V, the output is high (+3 V). When input is less than 1.2 V, the output is low. Between 1.2 V and 1.7 V, the output retains its value.

2.4.1 DIFFERENT TYPES OF DIGITAL OUTPUT

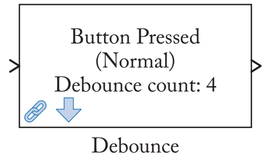
Different types of digital outputs are shown in Fig. 2.37. Figure 2.37(A) is called open drain output. When the transistor S1 is off, the Vout is +3 V. When the transistor S1 is on, the Vout is 0 V. When you want to use an open drain port, you need to add a pull-up resistor to it (Fig. 2.37(A)). Without a pull-up resistor, the open drain ports can't generate correct output since the circuit is not completed (Fig. 2.37(B)).

Figure 2.37(B) is called push-pull. Generally, this type of output is preferred since it can sink and source more current. Note that S1 and S2 are never on simultaneously. When S2 is on, S1 is off. In this case, Vout is +3 V. When S1 is on, S2 is off. In this case, Vout is 0 V.

2.4.2 DATA TYPE CONVERSION BLOCK

Data Type Conversion block (Fig. 2.39) permits you to ensure that what reaches the block is what it should be. For instance, consider the Simulink model shown in Fig. 2.40. In this case, the digital input block generates logical 0 and 1. The generated logical 0 and 1 can't directly be entered into the gain block because the gain block expects a number, not a logical value. So, in this case, we need to put a Data Conversion block between these two blocks.

If you double click the Data Type Conversion block, the window shown in Fig. 2.41 appears. There is no need to change these settings. Simulink automatically determines what was entered and what should get out once you click the Build model icon (Fig. 2.17).


Button Pressed
(Normal)
Debounce count: 4
Debounce

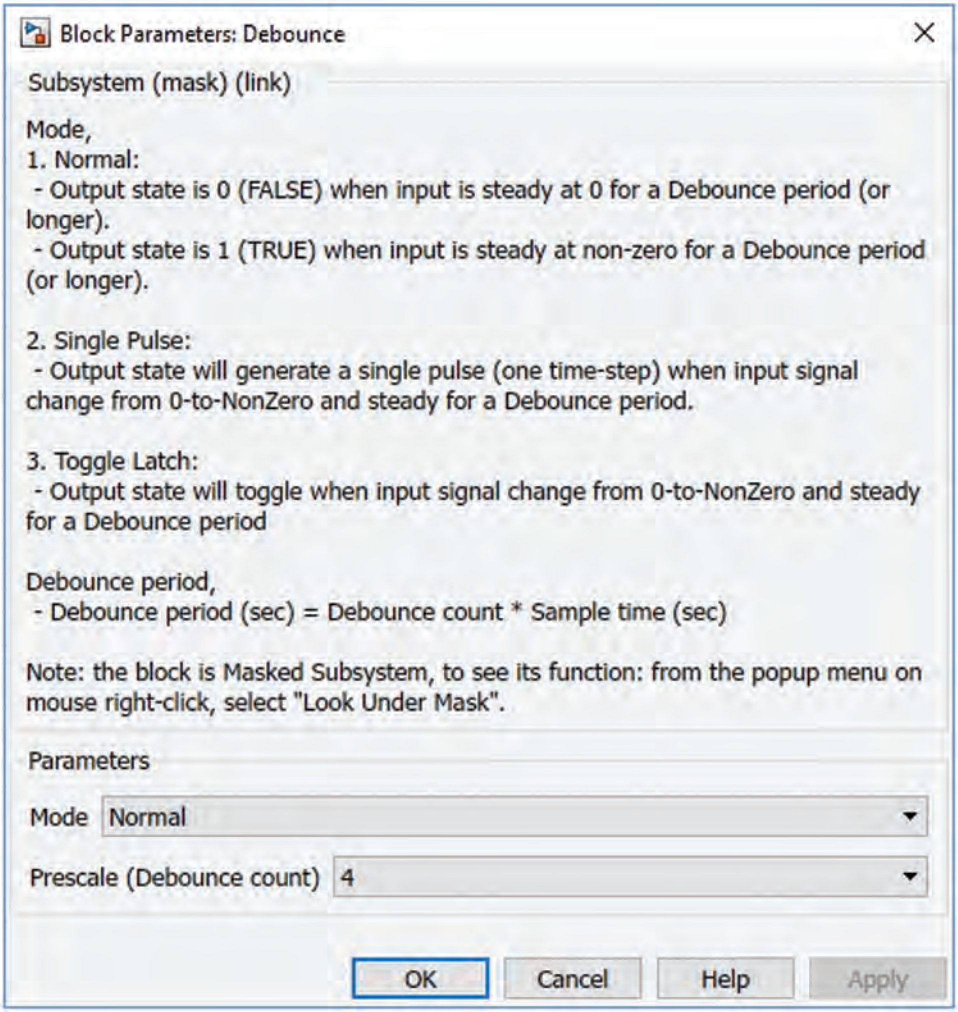

Block Parameters: Debounce
Subsystem (mask) (link)
Mode,
1. Normal:
- Output state is 0 (FALSE) when input is steady at 0 for a Debounce period (or longer).
- Output state is 1 (TRUE) when input is steady at non-zero for a Debounce period (or longer).
2. Single Pulse:
- Output state will generate a single pulse (one time-step) when input signal change from 0-to-NonZero and steady for a Debounce period.
3. Toggle Latch:
- Output state will toggle when input signal change from 0-to-NonZero and steady for a Debounce period
Debounce period,
- Debounce period (sec) = Debounce count * Sample time (sec)
Note: the block is Masked Subsystem, to see its function: from the popup menu on mouse right-click, select "Look Under Mask".
Parameters
Mode
Normal
Prescale (Debounce count)
4
OK
Cancel
Help
Apply

Figure 2.30

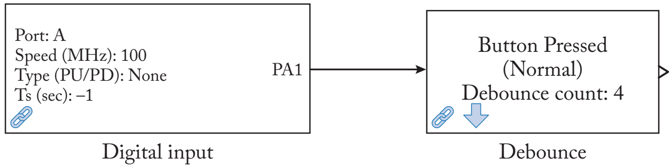


Figure 2.31

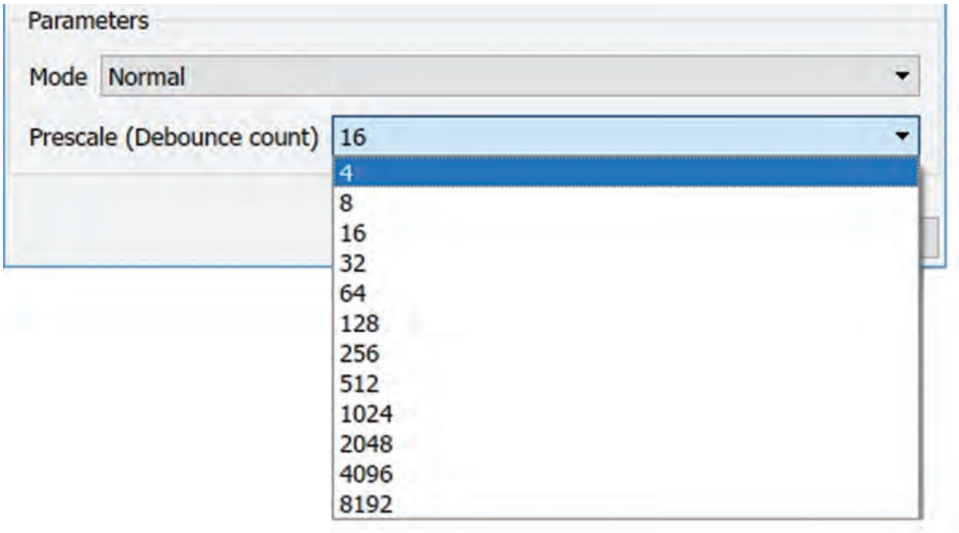


Figure 2.32

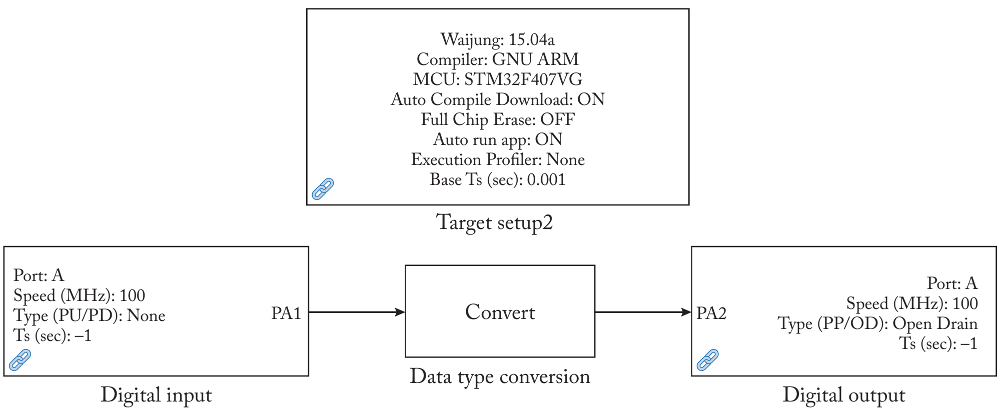


Figure 2.33

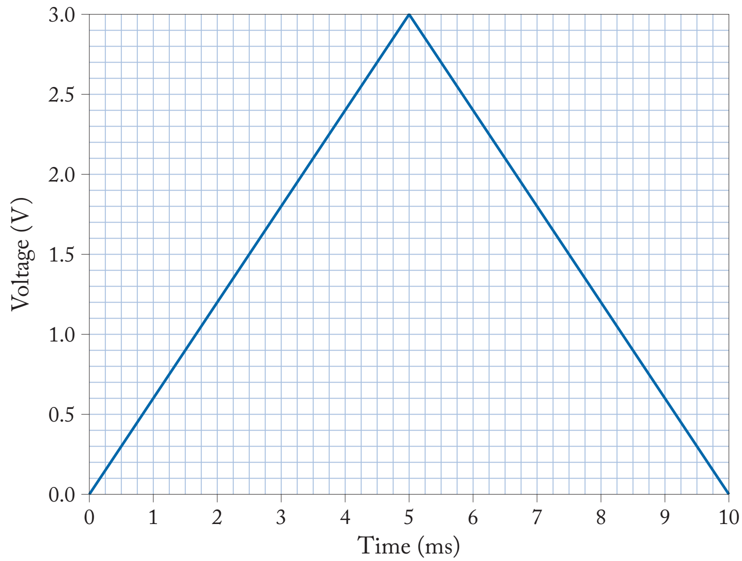


Figure 2.34

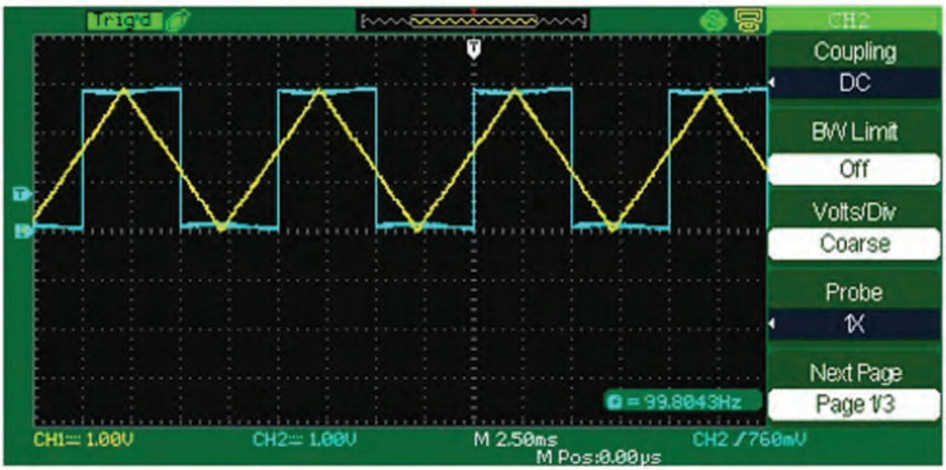


Figure 2.35

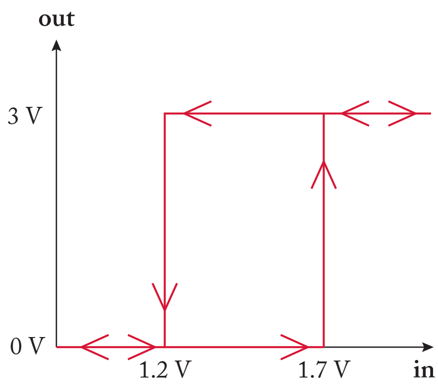


Figure 2.36

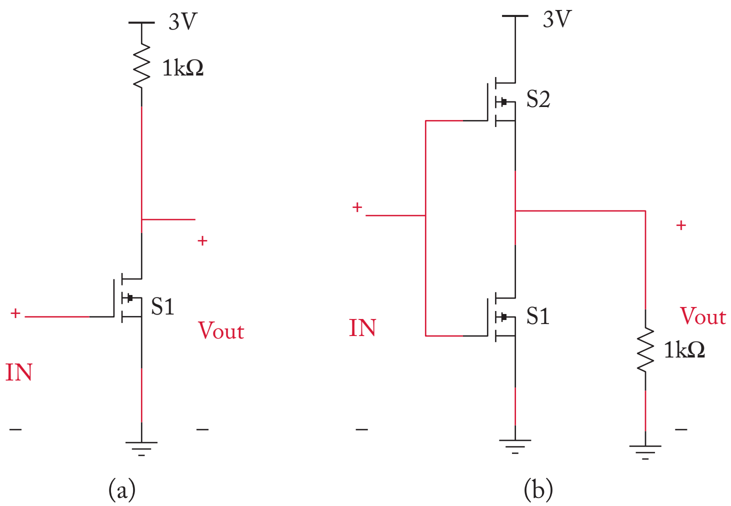


Figure 2.37

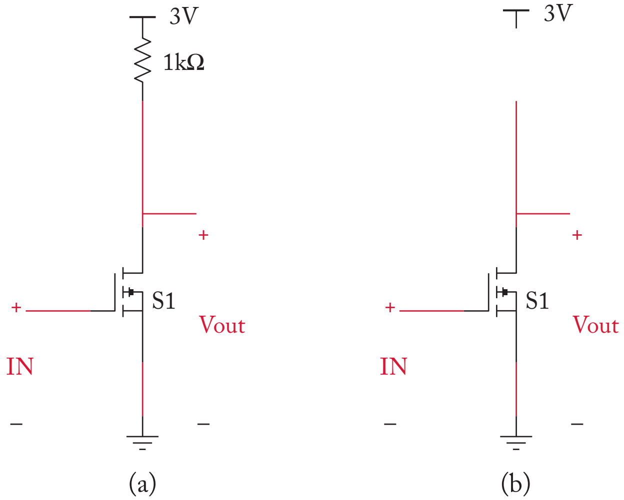


Figure 2.38

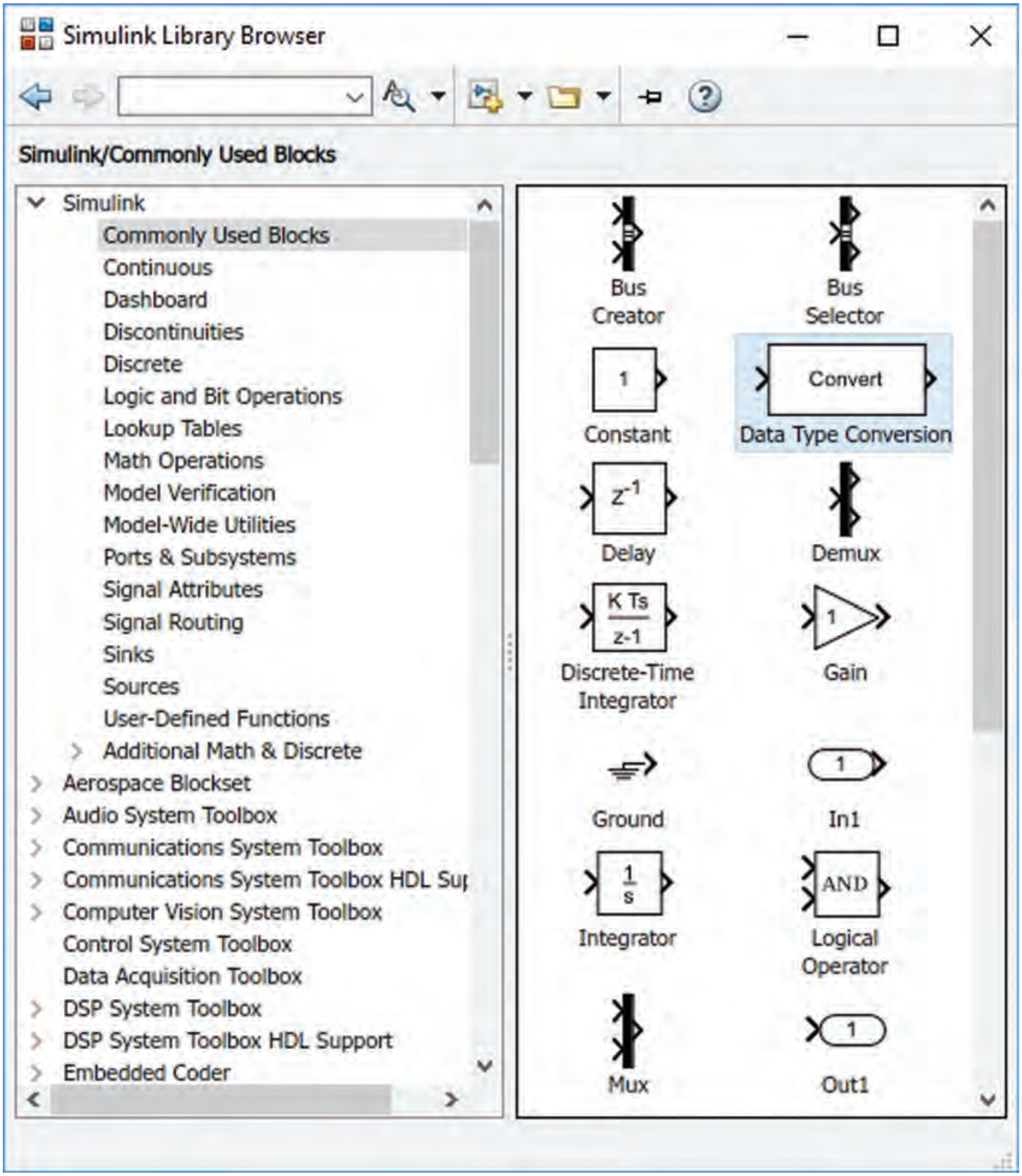

Simulink Library Browser
Simulink/Commonly Used Blocks
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
User-Defined Functions
Additional Math & Discrete
Aerospace Blockset
Audio System Toolbox
Communications System Toolbox
Communications System Toolbox HDL Sup
Computer Vision System Toolbox
Control System Toolbox
Data Acquisition Toolbox
DSP System Toolbox
DSP System Toolbox HDL Support
Embedded Coder
Bus Creator
Bus Selector
1
Constant
Convert
Data Type Conversion
z-1
Delay
Demux
K Ts
z-1
Discrete-Time Integrator
1
Gain
Ground
1
In1
1
s
Integrator
AND
Logical Operator
Mux
1
Out1

Figure 2.39

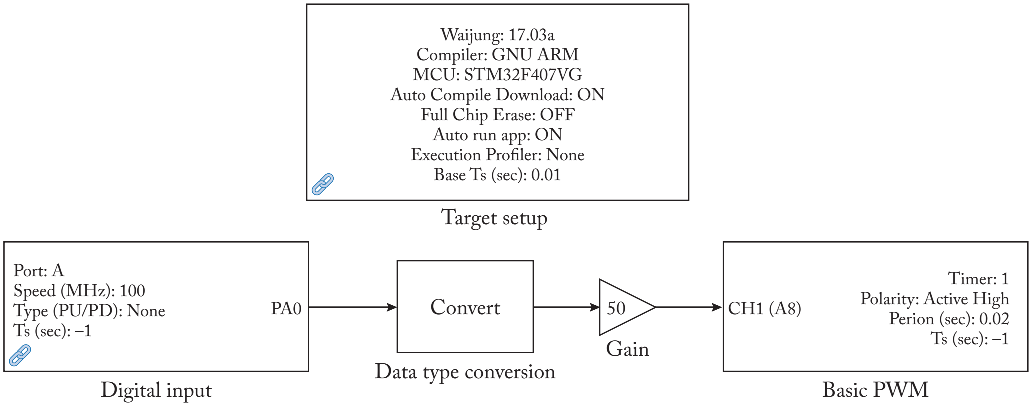


Figure 2.40: The PWM block is studied in Section 3.3.

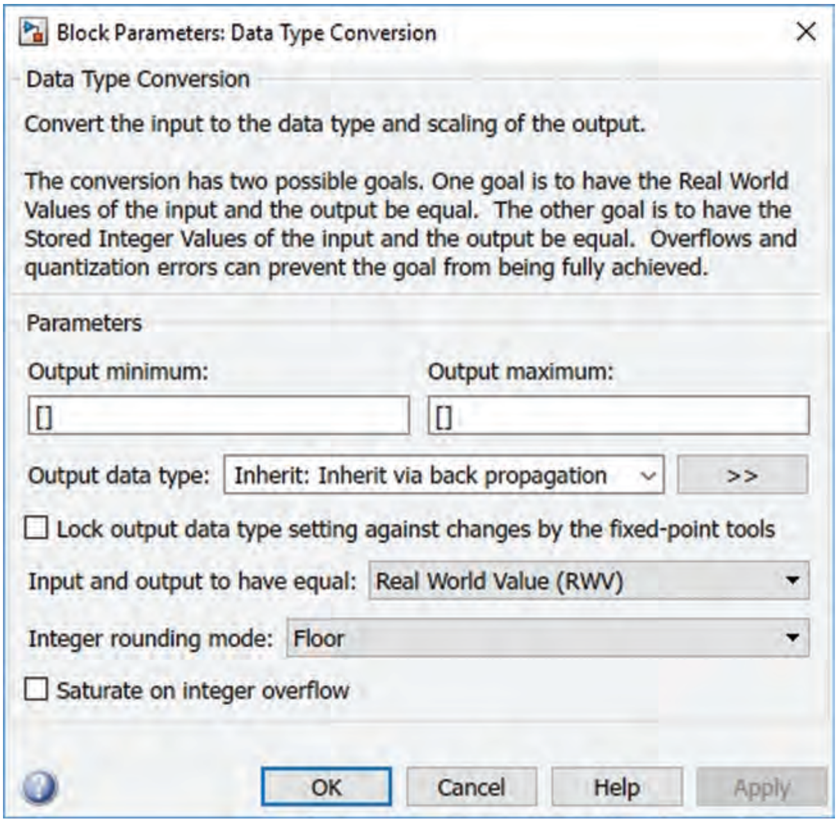


Figure 2.41

2.5 EXAMPLE 4: COMPARISON OF OPEN DRAIN AND PUSH-PULL OUTPUTS

The Simulink model of this example is shown in Fig. 2.42. Note that the Data conversion blocks convert the numeric values into logical values. (Non-zero numeric values are converted into high and zero numeric value is converted into low.) The hardware connection of this example is shown in Fig. 2.43. LEDs and 560 Ω resistors are available on the discovery board. You only need to add the 1 kΩ pull-up resistor between the PD13 and VDD line.

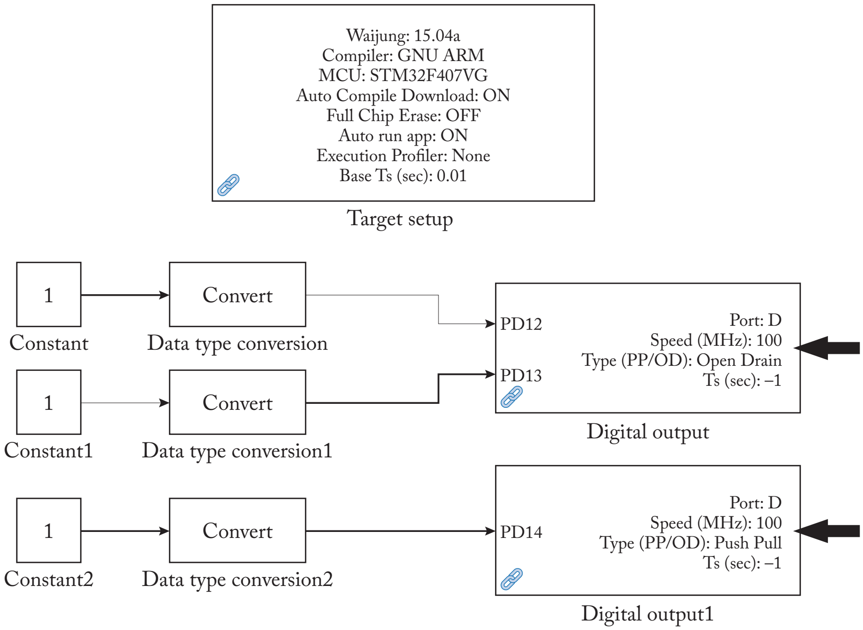
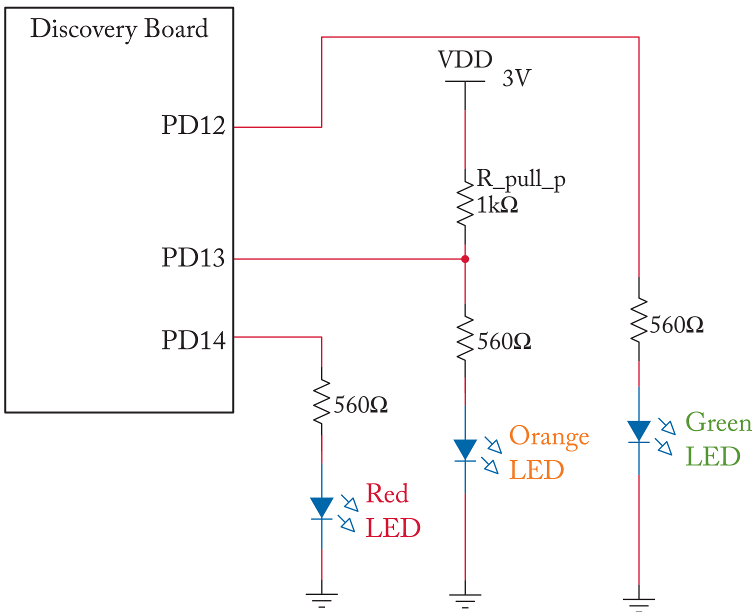


Figure 2.42

Double click the "Digital Output" block and select the Open Drain for Type (Push-Pull/Open-Drain) drop-down box (Fig. 2.44). Double click the "Digital Output 1" block and ensure that Push-Pull is selected for Type (Push-Pull/Open-Drain) drop-down box.

After uploading the code into the board you will see that red LED connected to PD14 and orange LED connected to Pin PD13 are turned on, however, the green LED connected to PD12 is off.

The reason is easily understandable with the aid of Fig. 2.45. Note that absence of pull-up resistor doesn't permit the circuit to be completed. The PD14 is configured as push-pull and needs to pull-up resistor.



Discovery Board
PD12
PD13
PD14
VDD
3V
R_pull_p
1kΩ
560Ω
560Ω
560Ω
Red
LED
Orange
LED
Green
LED

Figure 2.43

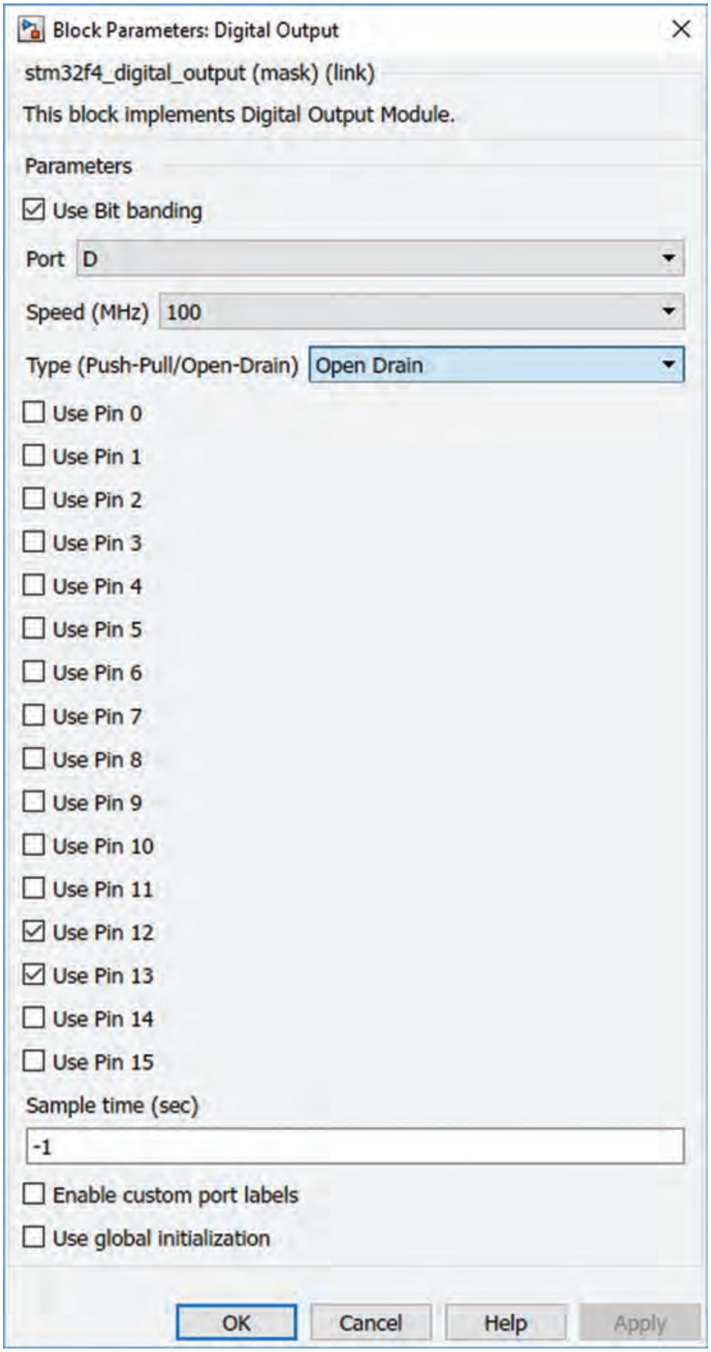

Block Parameters: Digital Output
stm32f4_digital_output (mask) (link)
This block implements Digital Output Module.
Parameters
Use Bit banding
Port
D
Speed (MHz)
100
Type (Push-Pull/Open-Drain)
Open Drain
Use Pin 0
Use Pin 1
Use Pin 2
Use Pin 3
Use Pin 4
Use Pin 5
Use Pin 6
Use Pin 7
Use Pin 8
Use Pin 9
Use Pin 10
Use Pin 11
Use Pin 12
Use Pin 13
Use Pin 14
Use Pin 15
Sample time (sec)
-1
Enable custom port labels
Use global initialization
OK
Cancel
Help
Apply

Figure 2.44

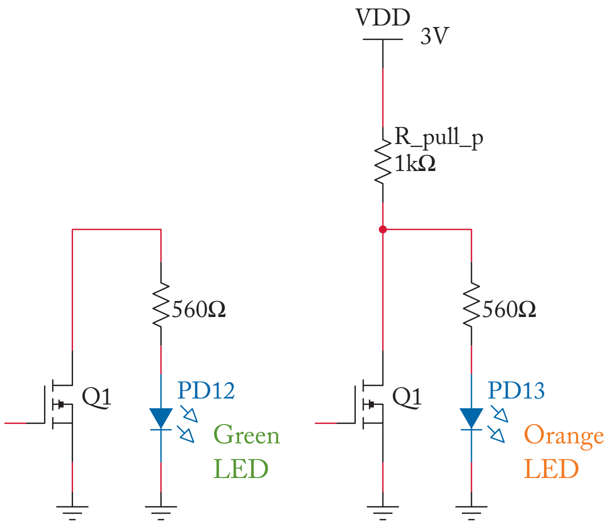


Figure 2.45

2.6 EXAMPLE 5: SEQUENTIALLY TURNING THE ON-BOARD LEDS ON AND OFF

A schematic of this example is shown in Fig. 2.46. In this example, the on-board LEDs are turned on one-by-one sequentially.

Settings of the Counter Limited block (Fig. 2.47) are shown in Fig. 2.48. These settings generate the 0, 1, 2, 3, 0, 1, 2, 3,… sequence. Duration of each value is 1 sec since the Sample time box is filled with 1 sec.

The Compare To Constant block (Fig. 2.49) compares the output of Counter Limited block with a constant value entered to the Constant value box (Fig. 2.50). Select the "==" from the Operator drop-down list (Fig. 2.50) since we want to turn on the LEDs when the output of Counter Limited block equals a specific value.

Upload the model to the board. You will see that LEDs turn on one-by-one and each LED is on for period of 1 sec.

2.7 EXAMPLE 6: BINARY COUNTING

Simulink model of this example is shown in Fig. 2.51. In this example we want to use the on-board LEDs to count in binary.

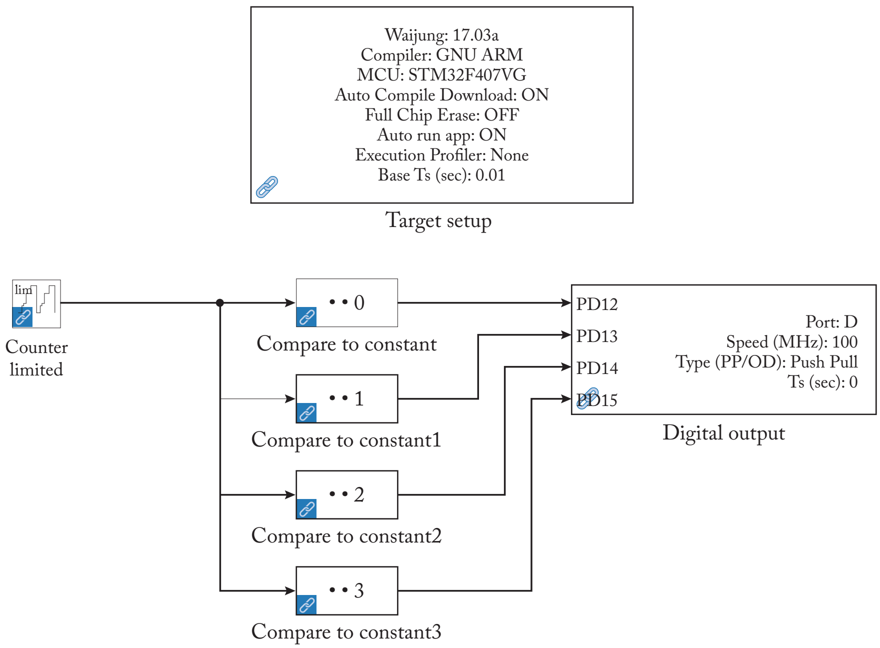

Waijung: 17.03a
Compiler: GNU ARM
MCU: STM32F407VG
Auto Compile Download: ON
Full Chip Erase: OFF
Auto run app: ON
Execution Profiler: None
Base Ts (sec): 0.01
Target setup
limi
Counter limited
• • 0
Compare to constant
• • 1
Compare to constant1
• • 2
Compare to constant2
• • 3
Compare to constant3
PD12
PD13
PD14
PD15
Port: D
Speed (MHz): 100
Type (PP/OD): Push Pull
Ts (sec): 0
Digital output

Figure 2.46

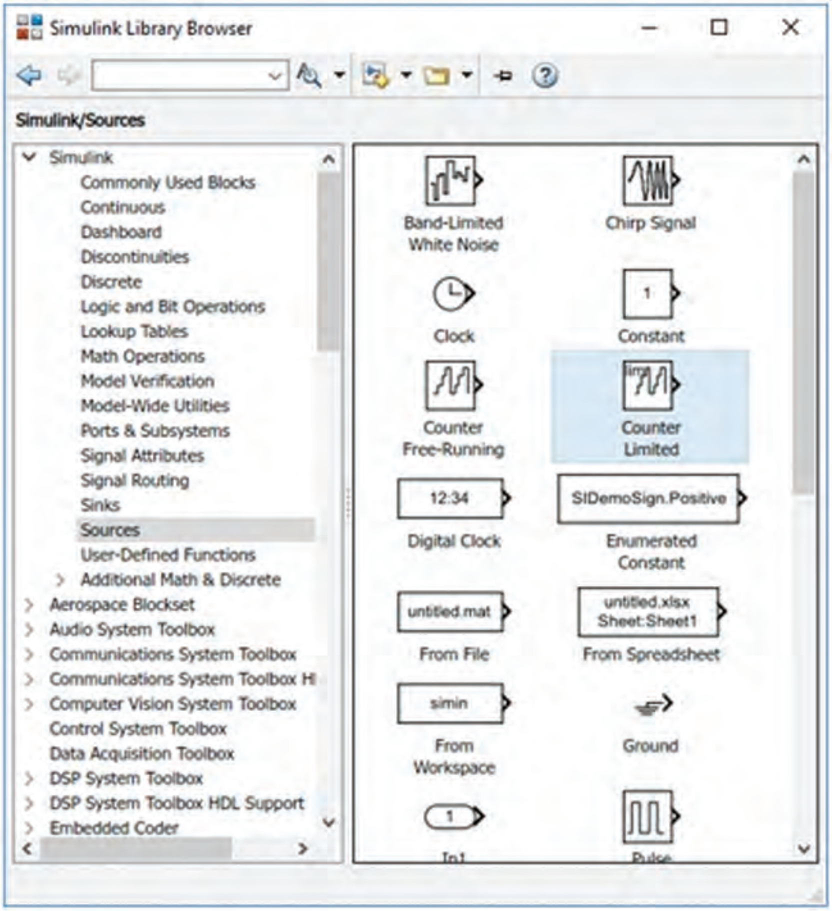

Simulink Library Browser
Simulink/Sources
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
User-Defined Functions
Additional Math & Discrete
Aerospace Blockset
Audio System Toolbox
Communications System Toolbox
Communications System Toolbox HI
Computer Vision System Toolbox
Control System Toolbox
Data Acquisition Toolbox
DSP System Toolbox
DSP System Toolbox HDL Support
Embedded Coder
Band-Limited White Noise
Chirp Signal
Clock
1
Constant
Counter Free-Running
Counter Limited
12:34
Digital Clock
SIDemoSign.Positive
Enumerated Constant
untitled.mat
From File
untitled.xlsx Sheet:Sheet1
From Spreadsheet
simin
From Workspace
Ground
1
In1
Pulse

Figure 2.47

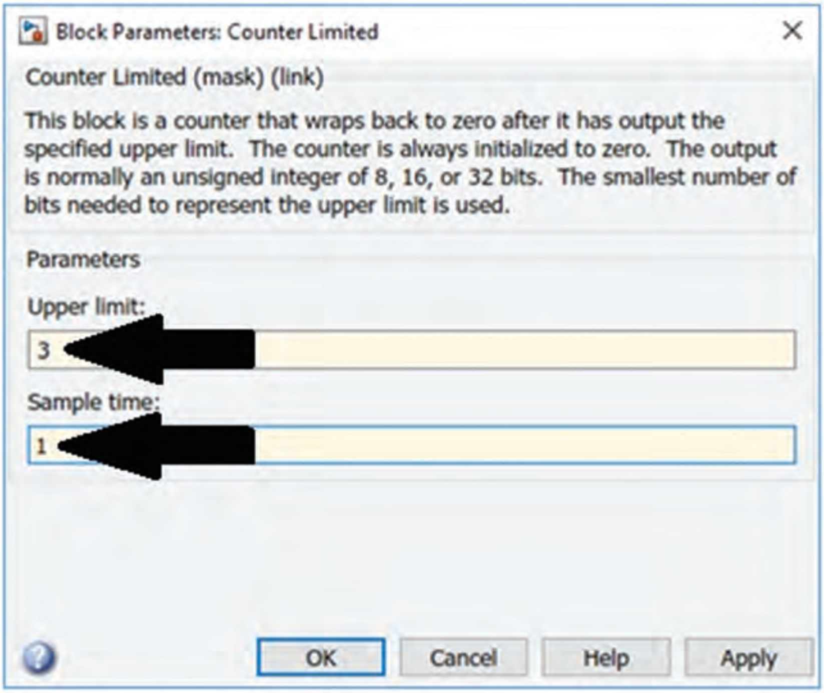

Block Parameters: Counter Limited
Counter Limited (mask) (link)
This block is a counter that wraps back to zero after it has output the specified upper limit. The counter is always initialized to zero. The output is normally an unsigned integer of 8, 16, or 32 bits. The smallest number of bits needed to represent the upper limit is used.
Parameters
Upper limit:
3
Sample time:
1
OK
Cancel
Help
Apply

Figure 2.48

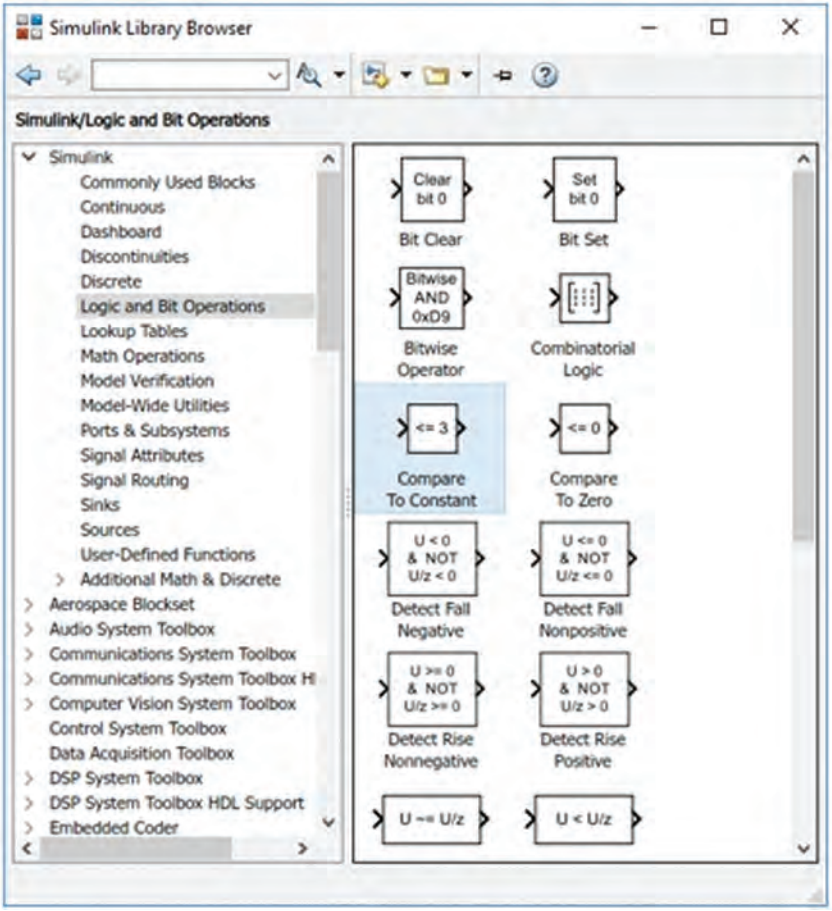

Simulink Library Browser
Simulink/Logic and Bit Operations
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
User-Defined Functions
Additional Math & Discrete
Aerospace Blockset
Audio System Toolbox
Communications System Toolbox
Communications System Toolbox HI
Computer Vision System Toolbox
Control System Toolbox
Data Acquisition Toolbox
DSP System Toolbox
DSP System Toolbox HDL Support
Embedded Coder
Clear bit 0
Bit Clear
Set bit 0
Bit Set
Bitwise AND 0xD9
Bitwise Operator
Combinatorial Logic
<= 3
Compare To Constant
<= 0
Compare To Zero
U < 0 & NOT U/z < 0
Detect Fall Negative
U <= 0 & NOT U/z <= 0
Detect Fall Nonpositive
U >= 0 & NOT U/z >= 0
Detect Rise Nonnegative
U > 0 & NOT U/z > 0
Detect Rise Positive
U ~= U/z
U < U/z

Figure 2.49

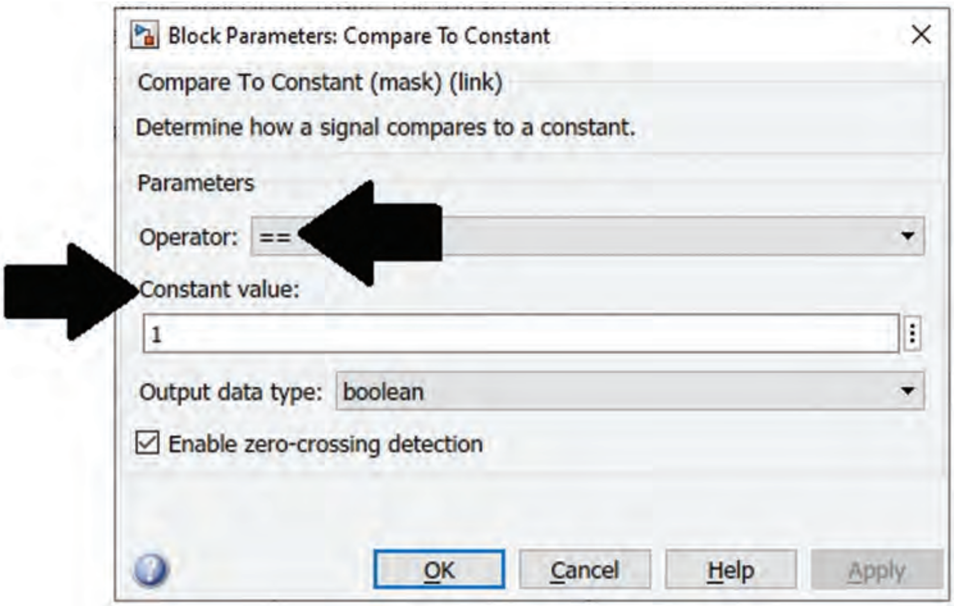


Figure 2.50

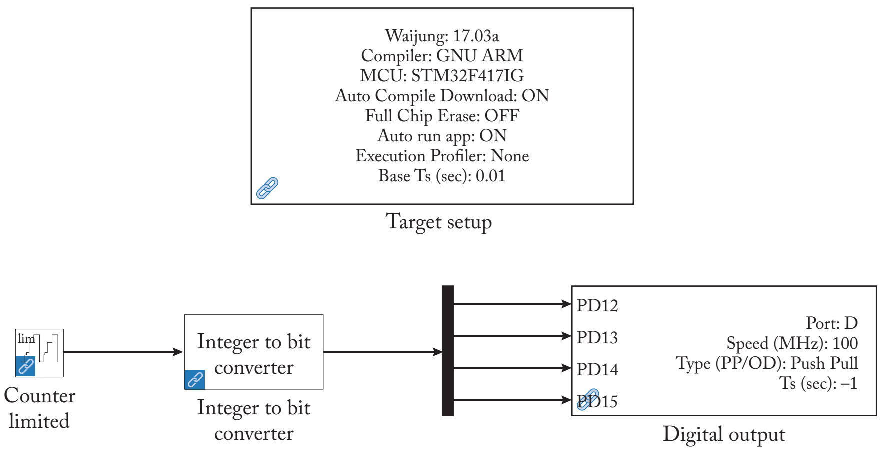


Figure 2.51

In this example we used an Integer to Bit Converter block (Fig. 2.52) to convert the output of the Counter Limited block into a binary number. Each bit of the binary number is shown on one of the on-board LEDs. Settings of the Integer to Bit Converter and Counter limited blocks are shown in Figs. 2.53 and 2.54, respectively.

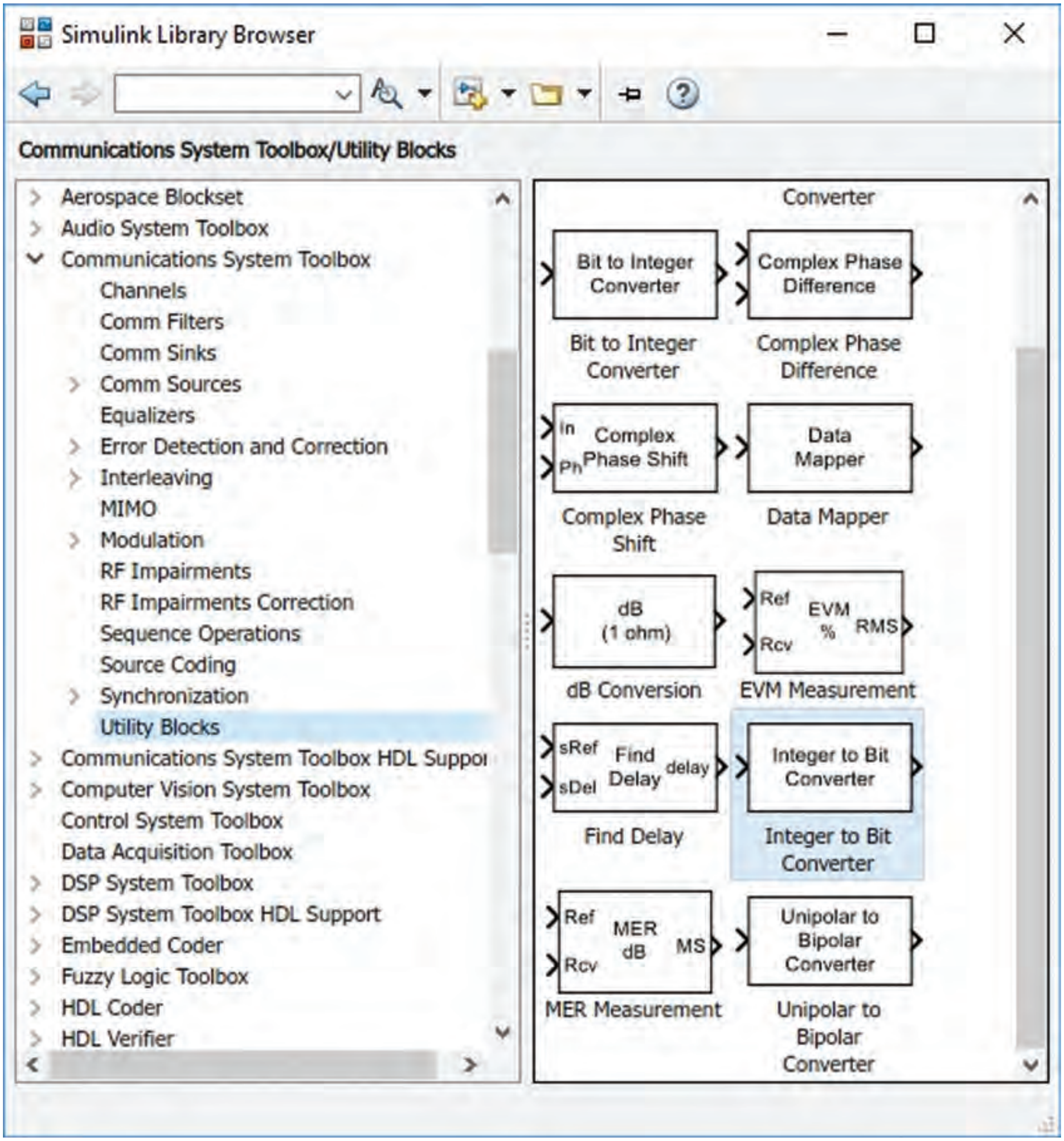


Figure 2.52

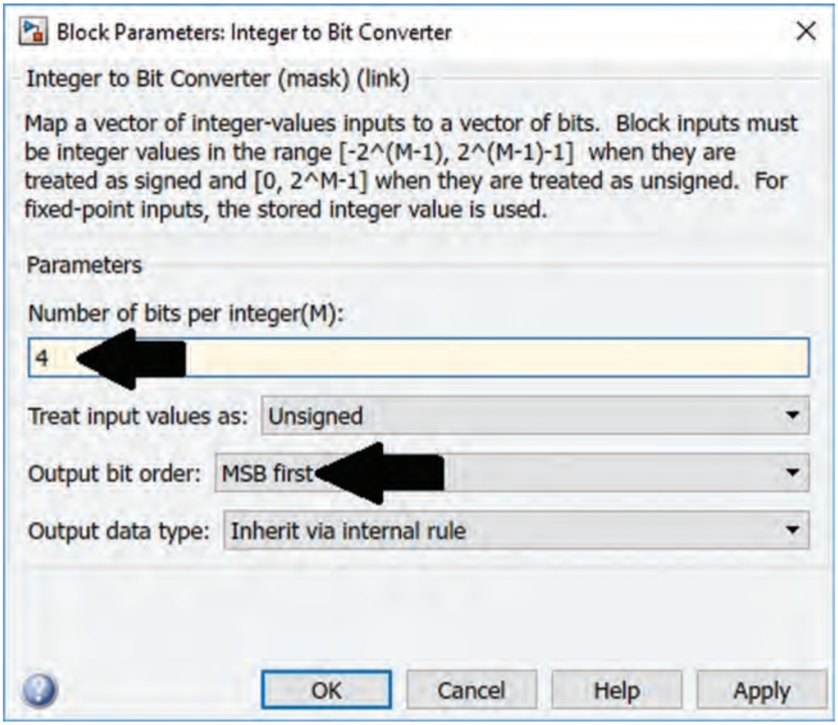


Figure 2.53

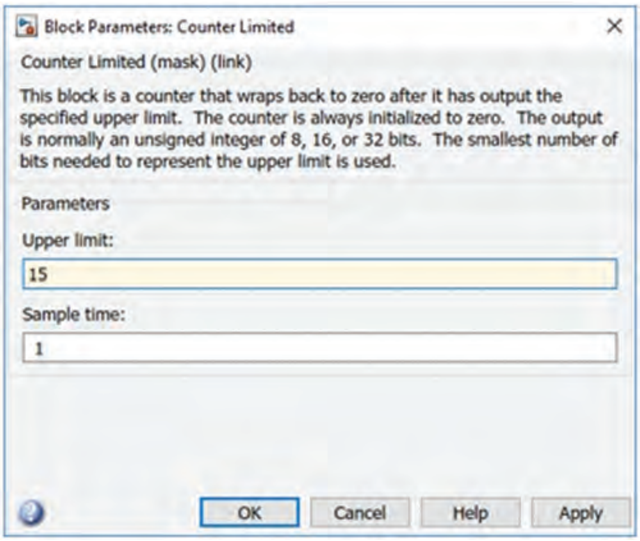


Figure 2.54

The Integer to Bit Converter block (Fig. 2.52) takes a decimal integer and converts it into a binary number. The obtained binary number can be converted into a decimal number again with the aid of Bit to Integer Converter block (Fig. 2.55).

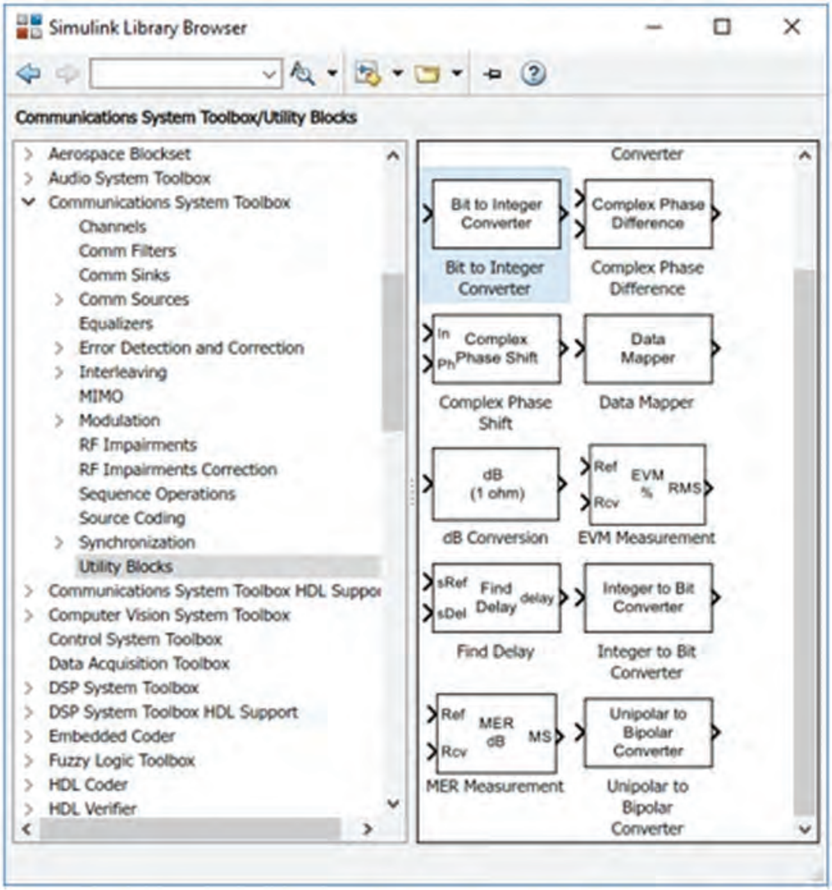


Figure 2.55

You can do the decimal to binary conversion with the aid of the MATLAB Function block (Fig. 2.56) as well. The binary equivalent of decimal numbers from 1 to 15 are shown in Table 2.1.

Let's use the MATLAB Function block to do the decimal to binary conversion. Draw the Simulink model shown in Fig. 2.57. Double click on the MATLAB Function block. This opens the window shown in Fig. 2.58. Enter the code of Table 2.2 to the MATLAB Function block (Fig. 2.59).

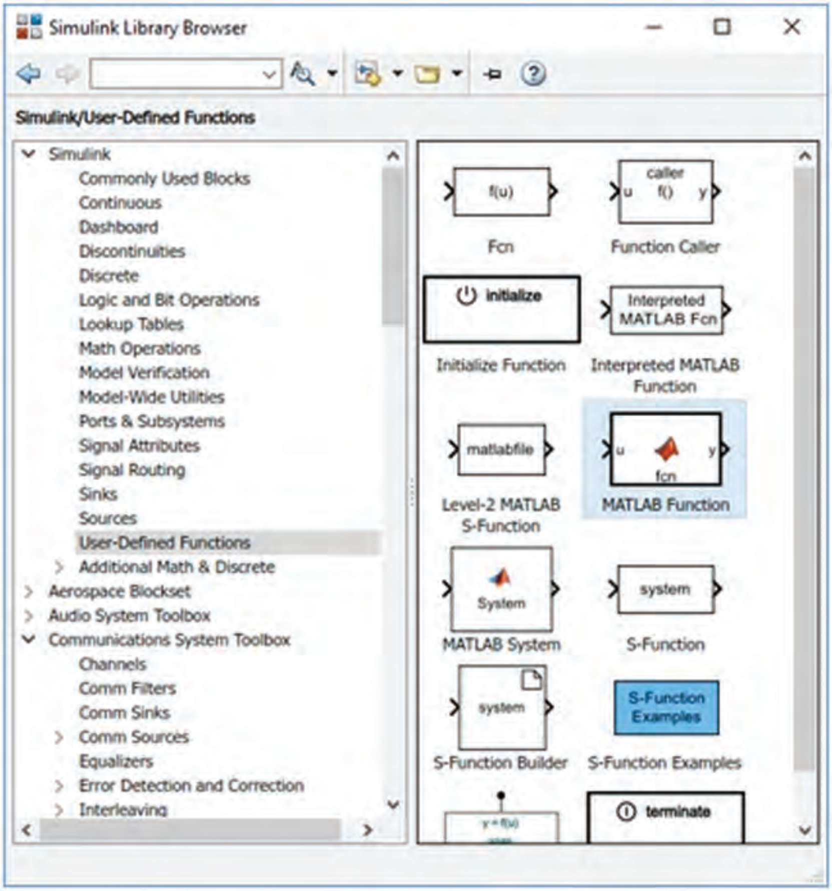

Simulink Library Browser
Simulink/User-Defined Functions
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
User-Defined Functions
Additional Math & Discrete
Aerospace Blockset
Audio System Toolbox
Communications System Toolbox
Channels
Comm Filters
Comm Sinks
Comm Sources
Equalizers
Error Detection and Correction
Interleaving
f(u)
Fcn
caller
f()
Function Caller
initialize
Initialize Function
Interpreted
MATLAB Fcn
Interpreted MATLAB Function
matlabfile
Level-2 MATLAB S-Function
fcn
MATLAB Function
System
MATLAB System
system
S-Function
S-Function Builder
S-Function Examples
terminate

Figure 2.56

Table 2.1: Decimal to binary conversion

Input	b3	b2	b1	b0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

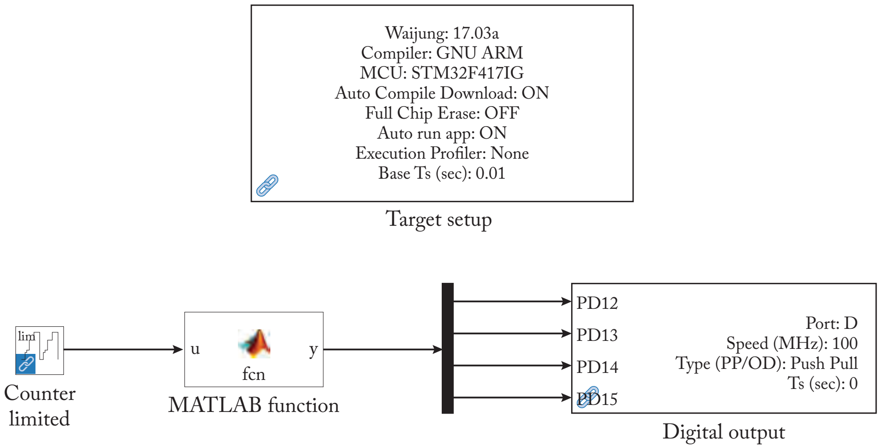


Figure 2.57

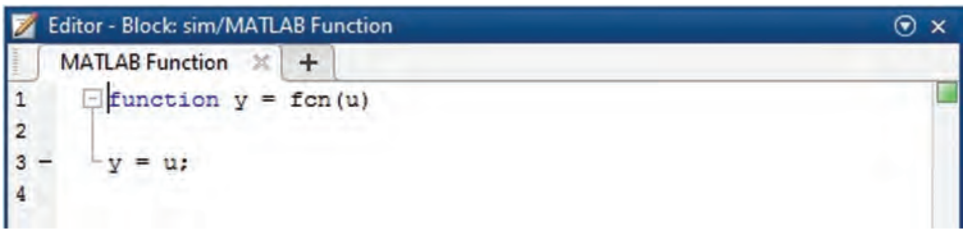


Figure 2.58

Table 2.2: Code for MATLAB Function block

```
function y = fcn(u)
out=[0 0 0 0];
switch u
    case 0
        out=[0 0 0 0];
    case 1
        out=[0 0 0 1];
    case 2
        out=[0 0 1 0];
    case 3
        out=[0 0 1 1];
    case 4
        out=[0 1 0 0];
    case 5
        out=[0 1 0 1];
    case 6
        out=[0 1 1 0];
    case 7
        out=[0 1 1 1];
    case 8
        out=[1 0 0 0];
    case 9
        out=[1 0 0 1];
    case 10
        out=[1 0 1 0];
    case 11
        out=[1 0 1 1];
    case 12
        out=[1 1 0 0];
    case 13
        out=[1 1 0 1];
    case 14
        out=[1 1 1 0];
    case 15
        out=[1 1 1 1];
end
y=out
```

Editor - Block: sim552/MATLAB Function

MATLAB Function

```
function y = fcn(u)

out=[0 0 0 0];

switch u
    case 0
        out=[0 0 0 0];
    case 1
        out=[0 0 0 1];
    case 2
        out=[0 0 1 0];
    case 3
        out=[0 0 1 1];
    case 4
        out=[0 1 0 0];
    case 5
        out=[0 1 0 1];
    case 6
        out=[0 1 1 0];
    case 7
        out=[0 1 1 1];
    case 8
        out=[1 0 0 0];
    case 9
        out=[1 0 0 1];
    case 10
        out=[1 0 1 0];
    case 11
        out=[1 0 1 1];
    case 12
        out=[1 1 0 0];
    case 13
```

Figure 2.59

Compile and upload the Simulink model into the board. Output is the same as the Simulink model shown in Fig. 2.51.

2.8 EXAMPLE 7: CHANGING THE STATE OF OUTPUT WITH A BUTTON

Simulink model of this example is shown in Fig. 2.60. In this example, the state of the on-board green LED connected to Pin PD12 is changed by pressing the button connected to PA0. When the LED is on, pressing the button cause it to turn off. When the LED is off, pressing the button causes it to turn on. This model uses the Triggered Subsystem (Fig. 2.61) and Memory (Fig. 2.62) blocks. Settings of Memory block are shown in Fig. 2.63.

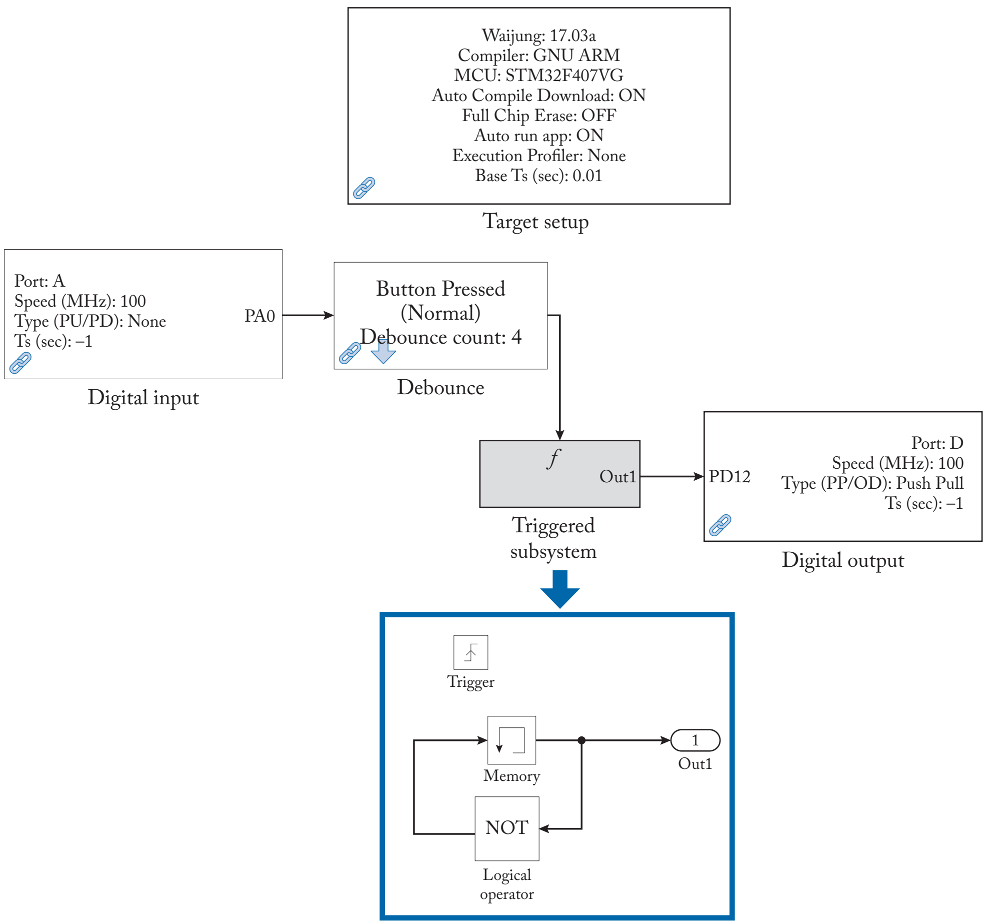


Figure 2.60

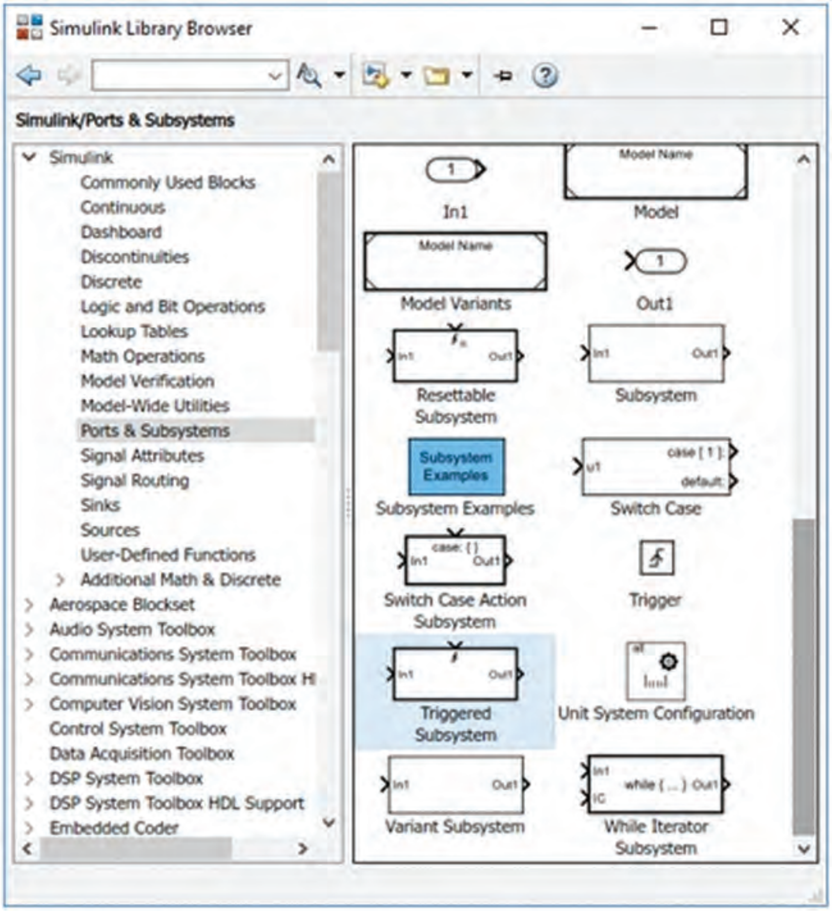

Simulink Library Browser
Simulink/Ports & Subsystems
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
User-Defined Functions
Additional Math & Discrete
Aerospace Blockset
Audio System Toolbox
Communications System Toolbox
Communications System Toolbox HI
Computer Vision System Toolbox
Control System Toolbox
Data Acquisition Toolbox
DSP System Toolbox
DSP System Toolbox HDL Support
Embedded Coder
In1
Model
Model Variants
Out1
Resettable Subsystem
Subsystem
Subsystem Examples
Switch Case
Switch Case Action Subsystem
Trigger
Triggered Subsystem
Unit System Configuration
Variant Subsystem
While Iterator Subsystem

Figure 2.61

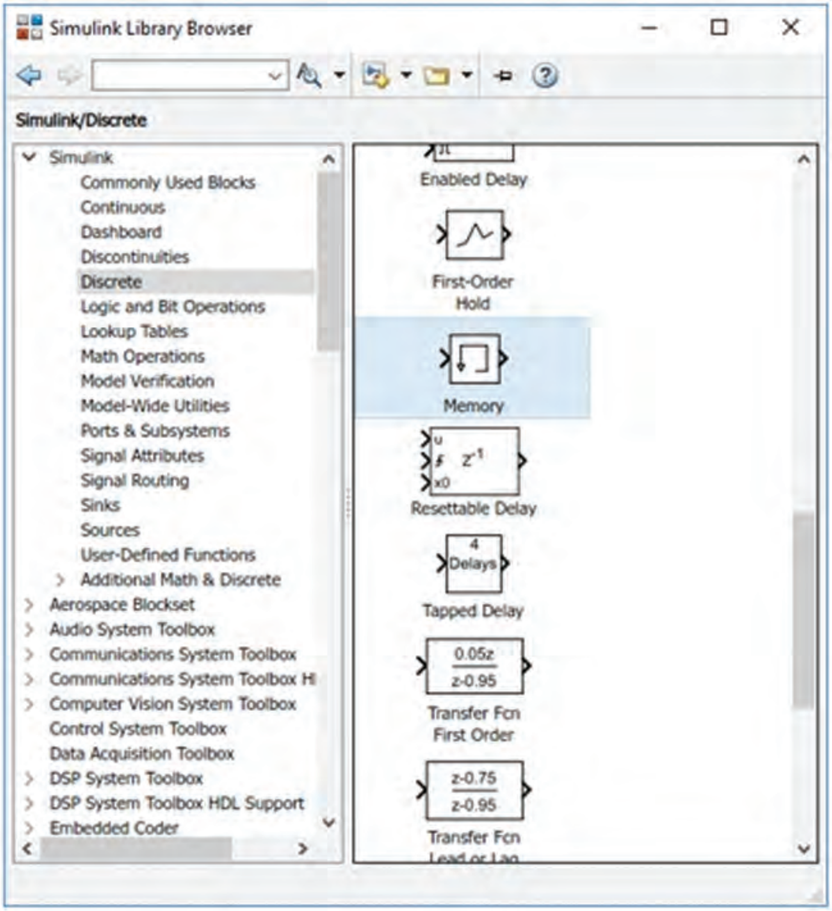

Simulink Library Browser
Simulink/Discrete
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
User-Defined Functions
Additional Math & Discrete
Aerospace Blockset
Audio System Toolbox
Communications System Toolbox
Communications System Toolbox HI
Computer Vision System Toolbox
Control System Toolbox
Data Acquisition Toolbox
DSP System Toolbox
DSP System Toolbox HDL Support
Embedded Coder
Enabled Delay
First-Order Hold
Memory
Resettable Delay
Tapped Delay
Transfer Fcn First Order
Transfer Fcn

Figure 2.62

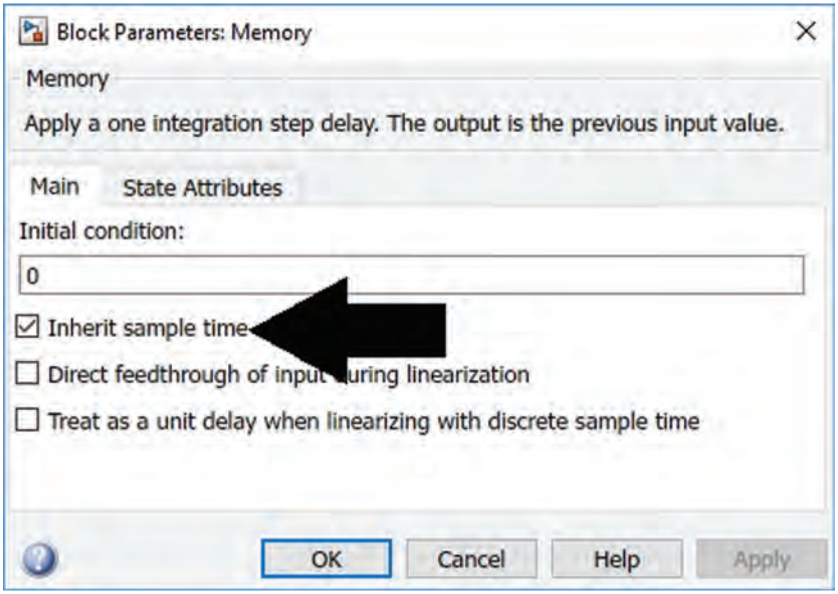


Figure 2.63

The NOT gate in Fig. 2.60 is a Logical Operator block (Fig. 2.64). In order to convert the block into a NOT, double click on the block and select the NOT for Operator drop down list (Fig. 2.65).

2.9 EXAMPLE 8: COUNTING THE NUMBER OF TIMES A SWITCH IS PRESSED

Simulink model of this example is shown in Fig. 2.66. In this example, the on-board LEDs turn on if the user presses the on-board switch more than or equal to three times. When the user presses the on-board LED, the value stored in the memory block increases by one. Settings of the Memory block are shown in Fig. 2.67.

A compare-to-constant block (Fig. 2.49) is used to see whether the value inside the memory block is bigger than three. If output of the block is logical 1, then all the on-board LEDs turn on.

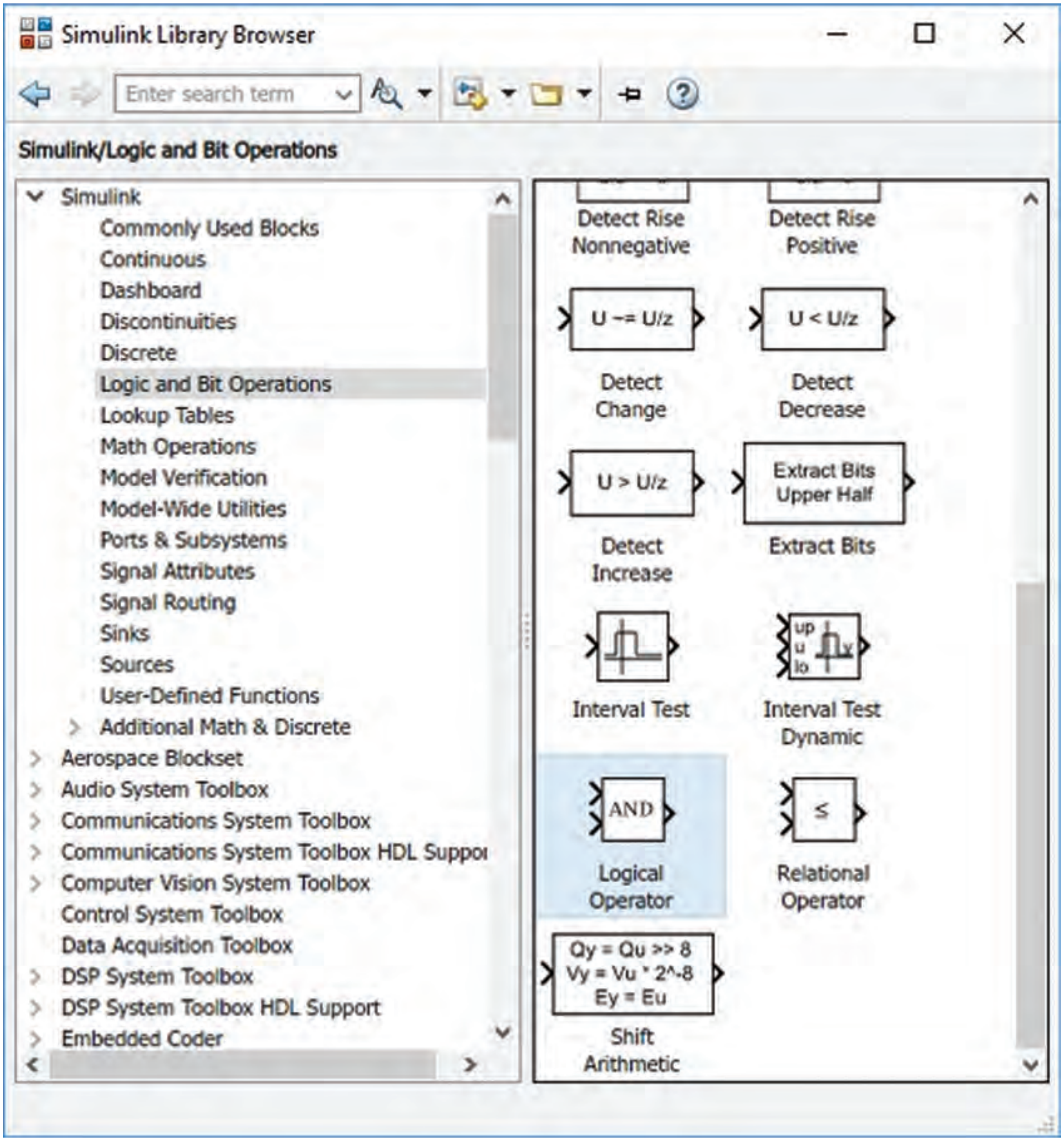

Simulink Library Browser
Enter search term
Simulink/Logic and Bit Operations
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
User-Defined Functions
Additional Math & Discrete
Aerospace Blockset
Audio System Toolbox
Communications System Toolbox
Communications System Toolbox HDL Suppor
Computer Vision System Toolbox
Control System Toolbox
Data Acquisition Toolbox
DSP System Toolbox
DSP System Toolbox HDL Support
Embedded Coder
Detect Rise Nonnegative
Detect Rise Positive
U ~= U/z
Detect Change
U < U/z
Detect Decrease
U > U/z
Detect Increase
Extract Bits Upper Half
Extract Bits
Interval Test
up
u
lo
y
Interval Test Dynamic
AND
Logical Operator
≤
Relational Operator
Qy = Qu >> 8
Vy = Vu * 2^-8
Ey = Eu
Shift Arithmetic

Figure 2.64

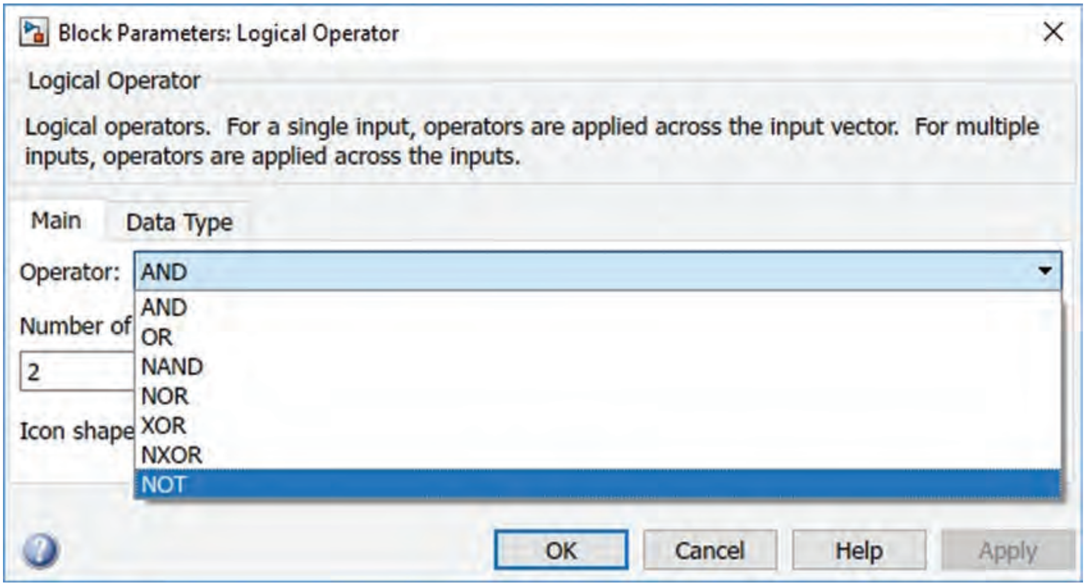

Block Parameters: Logical Operator
Logical Operator
Logical operators. For a single input, operators are applied across the input vector. For multiple inputs, operators are applied across the inputs.
Main
Data Type
Operator:
AND
Number of
2
Icon shape
AND
OR
NAND
NOR
XOR
NXOR
NOT
OK
Cancel
Help
Apply

Figure 2.65

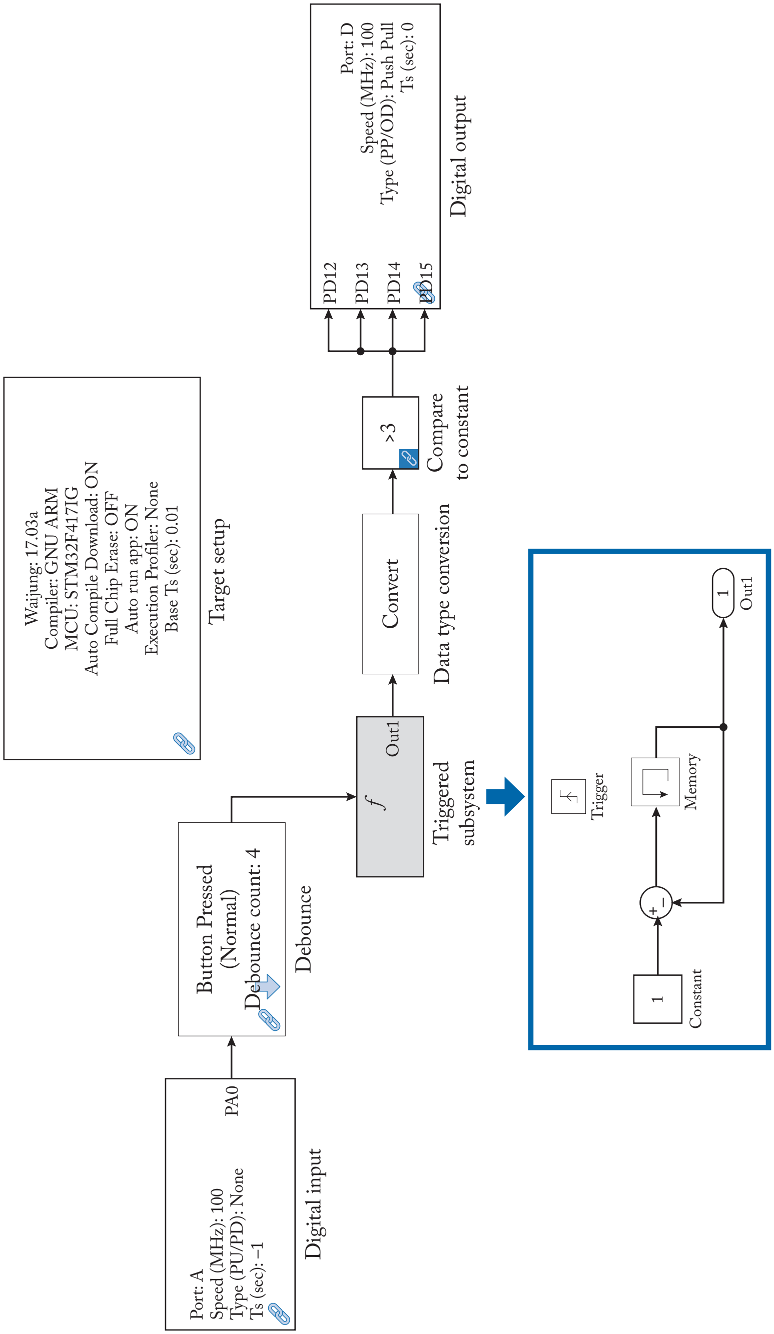

Port: D
Speed (MHz): 100
Type (PP/OD): Push Pull
Ts (sec): 0
PD12
PD13
PD14
PD15
Digital output
Waijung: 17.03a
Compiler: GNU ARM
MCU: STM32F417IG
Auto Compile Download: ON
Full Chip Erase: OFF
Auto run app: ON
Execution Profiler: None
Base Ts (sec): 0.01
Target setup
>3
Compare
to constant
Convert
Data type conversion
Out1
Triggered
subsystem
Button Pressed
(Normal)
Debounce count: 4
Debounce
Port: A
Speed (MHz): 100
Type (PU/PD): None
Ts (sec): -1
PA0
Digital input
Trigger
Memory
Constant
1
Out1

Figure 2.66

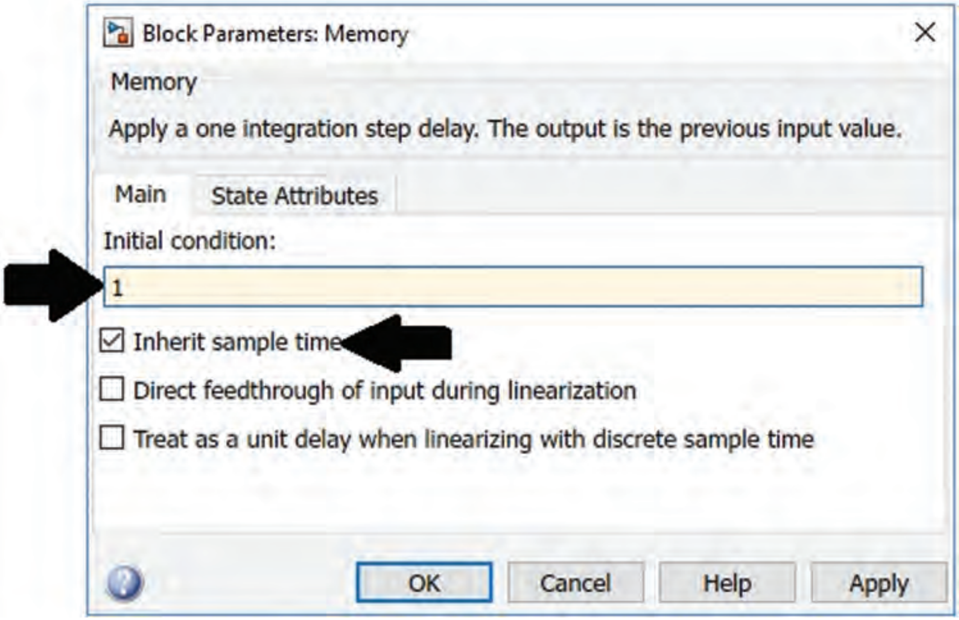


Figure 2.67

2.10 EXAMPLE 9: IMPLEMENTATION OF TRUTH TABLE

In this example we want to implement the truth table shown in Table 2.3.

Table 2.3: Truth table of Example 9

Input 1 (PA 0)	Input 2 (PA 1)	Input 3 (PA 2)	Output 1 (PD 12)	Output 2 (PD 13)
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Draw the Simulink model shown in Fig. 2.68. This model used a Combinational Logic block (Fig. 2.69) to implement the given truth table. Settings of the truth table are shown in Fig. 2.70. Note that only the output rows of the given truth table are entered into the Truth table box in Fig. 2.70.

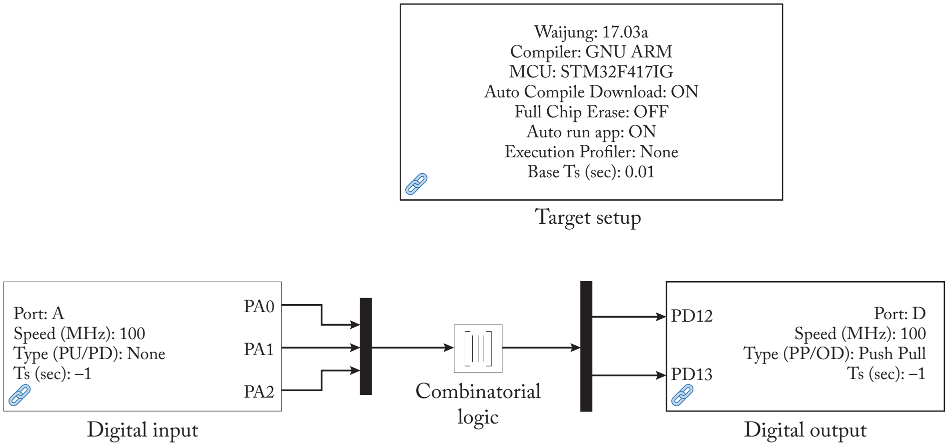


Figure 2.68

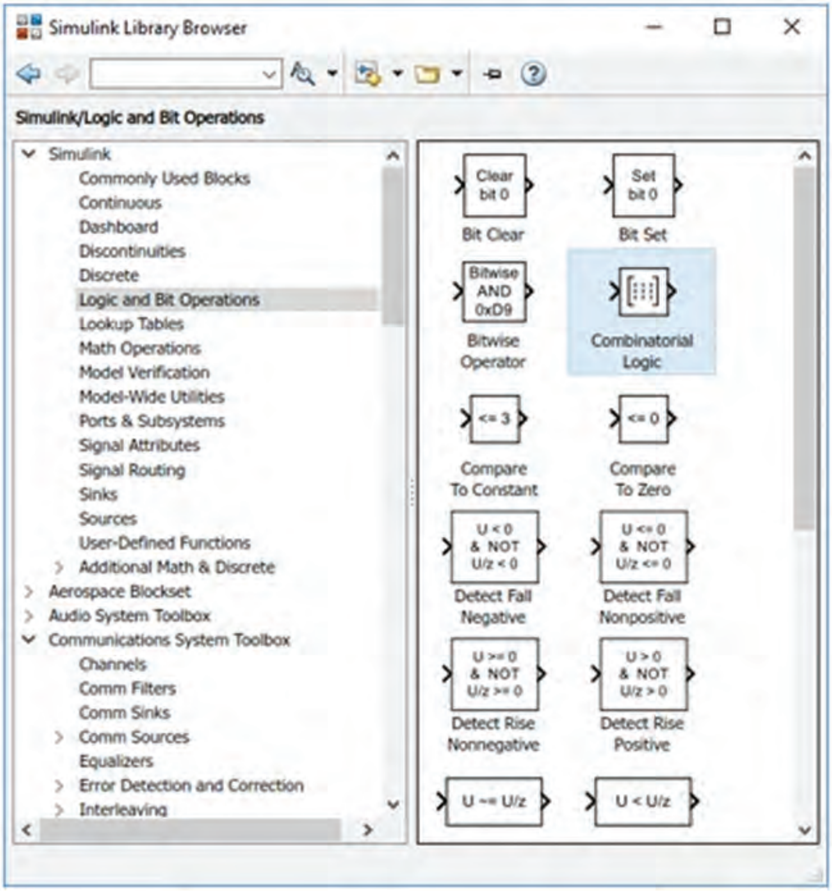

Simulink Library Browser
Simulink/Logic and Bit Operations
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
User-Defined Functions
Additional Math & Discrete
Aerospace Blockset
Audio System Toolbox
Communications System Toolbox
Channels
Comm Filters
Comm Sinks
Comm Sources
Equalizers
Error Detection and Correction
Interleaving
Clear bit 0
Bit Clear
Set bit 0
Bit Set
Bitwise AND 0xD9
Bitwise Operator
Combinatorial Logic
<= 3
Compare To Constant
<= 0
Compare To Zero
U < 0 & NOT U/z < 0
Detect Fall Negative
U <= 0 & NOT U/z <= 0
Detect Fall Nonpositive
U >= 0 & NOT U/z >= 0
Detect Rise Nonnegative
U > 0 & NOT U/z > 0
Detect Rise Positive
U ~= U/z
U < U/z

Figure 2.69

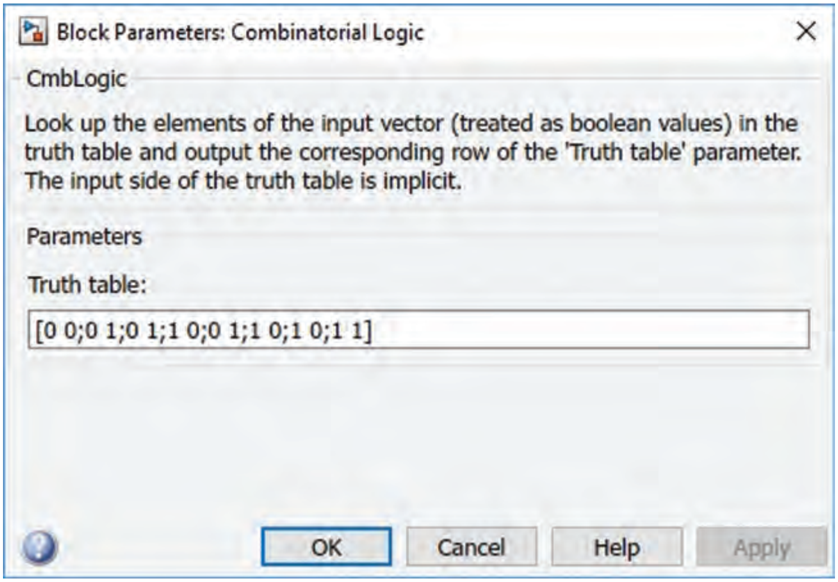

Block Parameters: Combinatorial Logic
CmbLogic
Look up the elements of the input vector (treated as boolean values) in the truth table and output the corresponding row of the 'Truth table' parameter. The input side of the truth table is implicit.
Parameters
Truth table:
[0 0;0 1;0 1;1 0;0 1;1 0;1 0;1 1]
OK
Cancel
Help
Apply

Figure 2.70

CHAPTER 3

Pulse Width Modulation (PWM)

3.1 INTRODUCTION

Pulse Width Modulation (PWM) is a method of controlling the average power delivered to the load. This technique has many applications (DC motor speed/position control, switch mode power supply to name a few). This chapter shows how to generate a PWM signal with STM32F407G-DISC1 board.

3.2 EXAMPLE 1: GENERATION OF PULSE WIDTH MODULATION (PWM) SIGNAL WITH THE BASIC PWM BLOCK

The basic PWM block (Fig. 3.1) can be used for generation of PWM signal. Input of the block is the required duty cycle. Input of the block can change from 0 up to 100. For instance, when the input is 75 the duty cycle of output signal of the block is 75%. PWM Period (seconds) box determine the frequency of output signal of the block. For instance, in Fig. 3.1, PWM Period (seconds) is filled with 0.02 sec. So, the output frequency of the block is $1/0.02 = 50$ Hz. Sample time (sec) box determines the sampling time of the input duty cycle signal. For instance, in Fig. 3.1 sample time (sec) box equals to 0.01 s. This means that the block reads input (duty cycle) signal at $t = 0, 0.01, 0.02, 0.03, \ldots$ and other values of input signal are ignored.

Draw the Simulink model shown in Fig. 3.2. Settings of Basic PWM block are shown in Fig. 3.3.

Upload the model into the board and use an oscilloscope to see the voltage of Pin A8. Waveform of Pin A8 is shown in Fig. 3.4. Note that the frequency of the obtained waveform is 50 Hz and its duty cycle is 75% as expected.

3.3 EXAMPLE 2: TWO-CHANNEL PWM WITH BASIC PWM BLOCK

Simulink model of this example is shown in Fig. 3.5. In this example, we want to generate PWM signal on two different pins. Settings of the Basic PWM block are shown in Fig. 3.6.

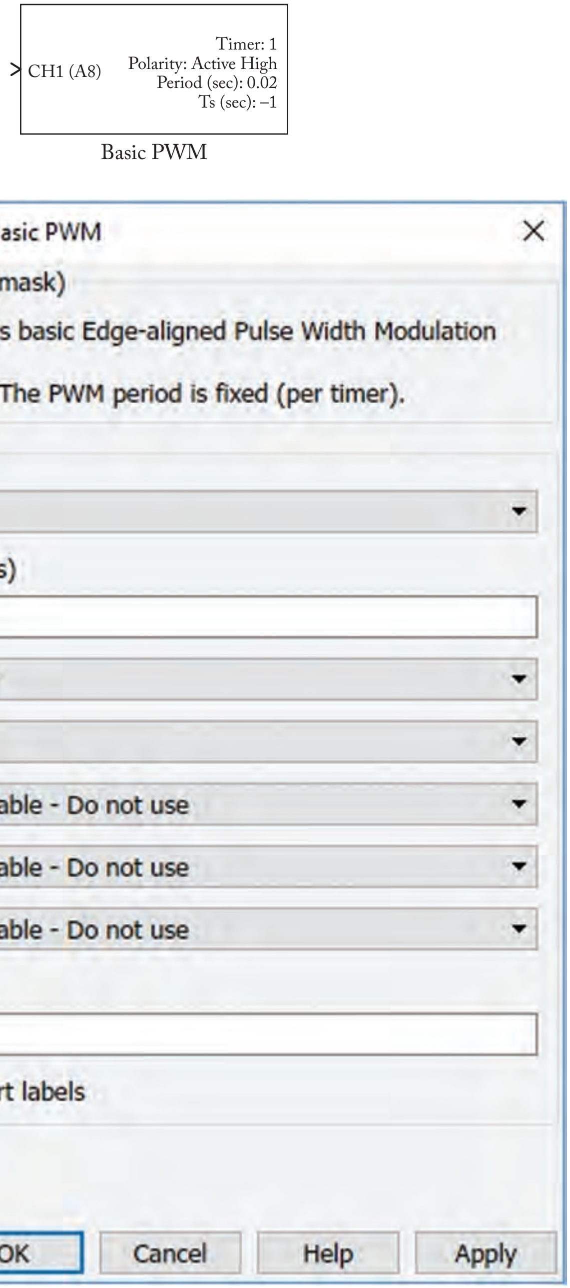

CH1 (A8)
Timer: 1
Polarity: Active High
Period (sec): 0.02
Ts (sec): –1
Basic PWM
Block Parameters: Basic PWM
stm32f4_basicpwm (mask)
This block implements basic Edge-aligned Pulse Width Modulation (PWM) generation.
All timers are 16-bit. The PWM period is fixed (per timer).
Parameters
Timer 1
PWM Period (seconds)
0.02
Polarity Active High
Channel 1 A8
Channel 2 Not available - Do not use
Channel 3 Not available - Do not use
Channel 4 Not available - Do not use
Sample time (sec)
0.01
Enable custom port labels
OK
Cancel
Help
Apply

Figure 3.1

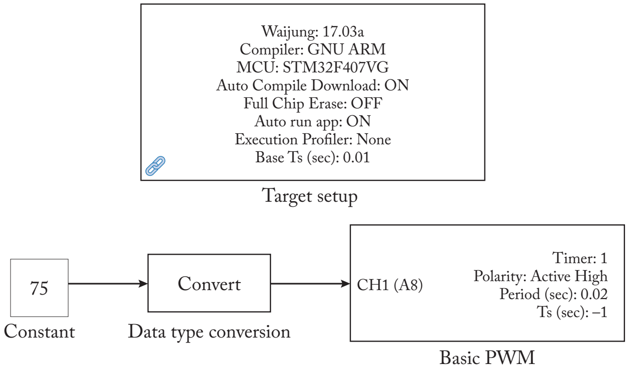


Figure 3.2

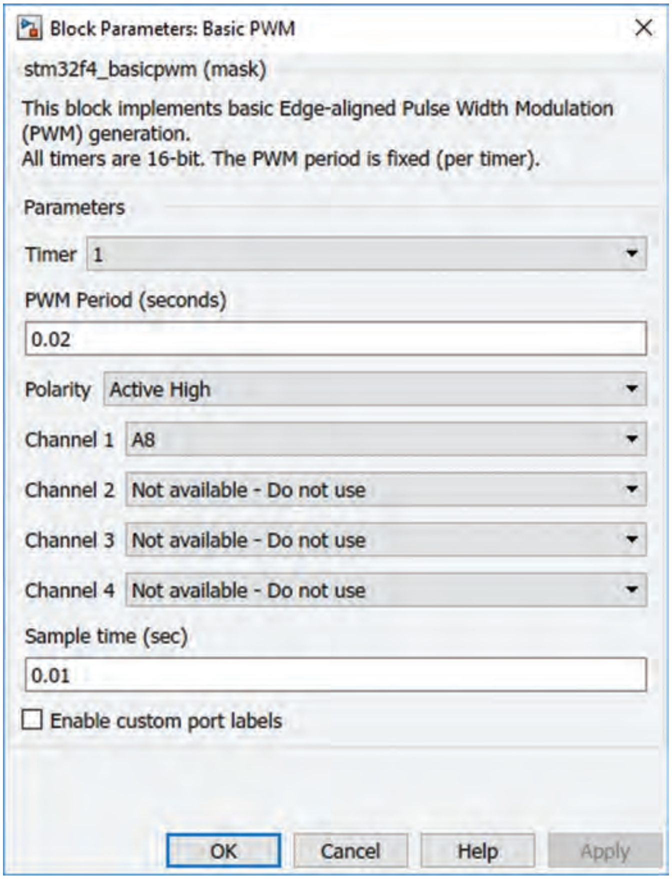


Figure 3.3

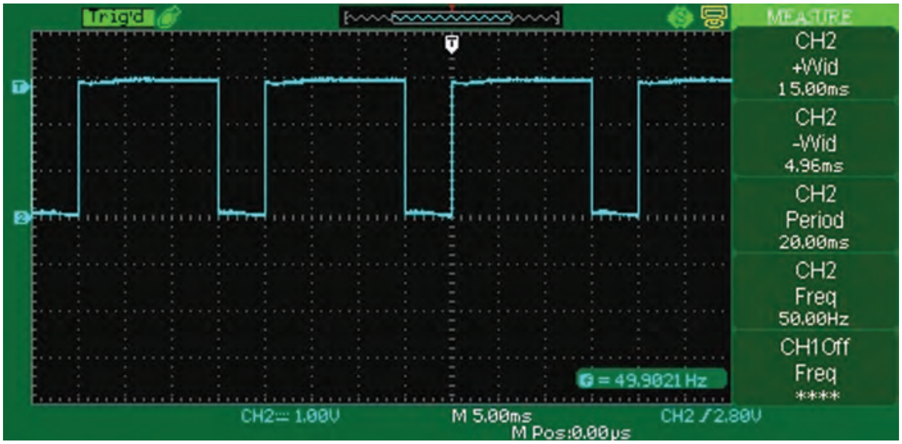


Figure 3.4

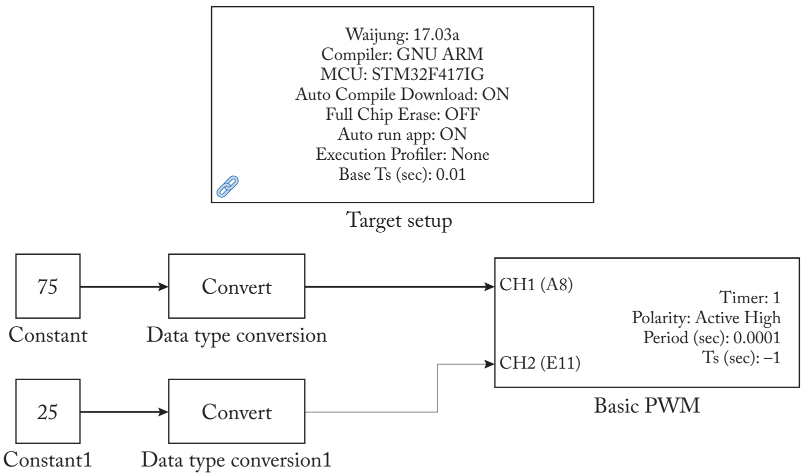


Figure 3.5

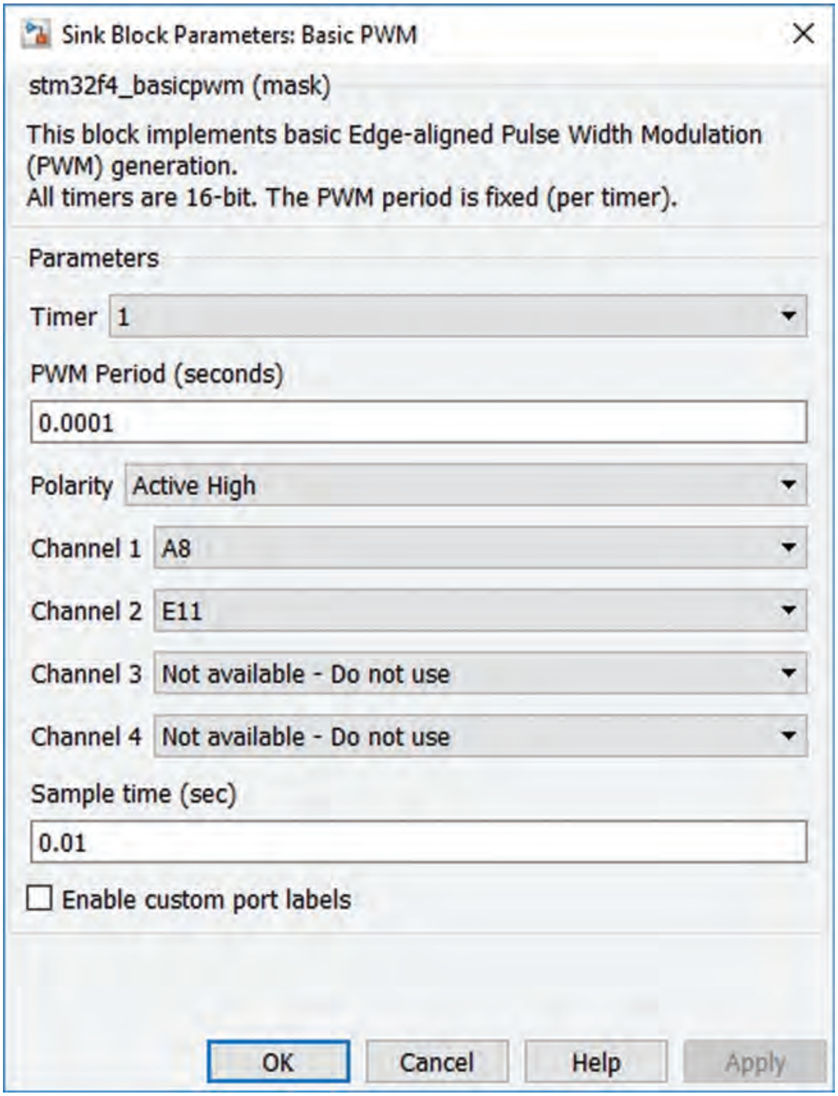

Sink Block Parameters: Basic PWM
stm32f4_basicpwm (mask)
This block implements basic Edge-aligned Pulse Width Modulation (PWM) generation.
All timers are 16-bit. The PWM period is fixed (per timer).
Parameters
Timer 1
PWM Period (seconds)
0.0001
Polarity Active High
Channel 1 A8
Channel 2 E11
Channel 3 Not available - Do not use
Channel 4 Not available - Do not use
Sample time (sec)
0.01
Enable custom port labels
OK
Cancel
Help
Apply

Figure 3.6

Upload the code to the board and use an oscilloscope to observe the signals on Pins A8 and E11. According to Fig. 3.7, frequency of both signal is 10 kHz. Note that duty cycle of signals in Fig. 3.7 are 25% and 75%.

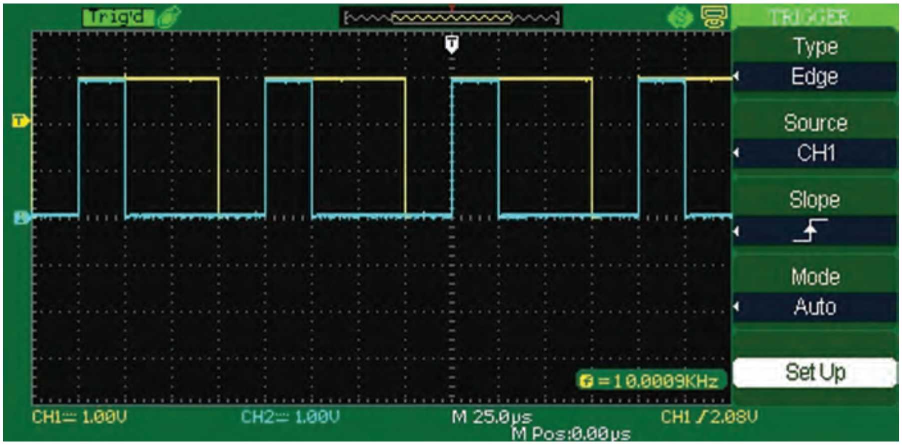


Figure 3.7

3.4 EXAMPLE 3: GENERATING A PWM SIGNAL WITH VARIABLE DUTY CYCLE

Simulink model of this example is shown in Fig. 3.8. In this example, the duty cycle is a variable signal and changes with time. Settings of Counter Limited block are shown in Fig. 3.9.

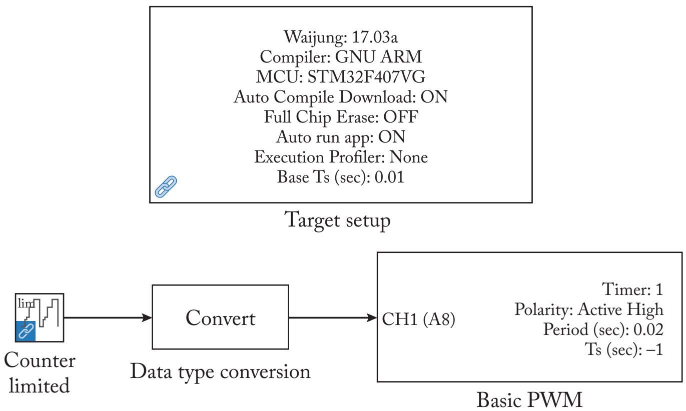


Figure 3.8

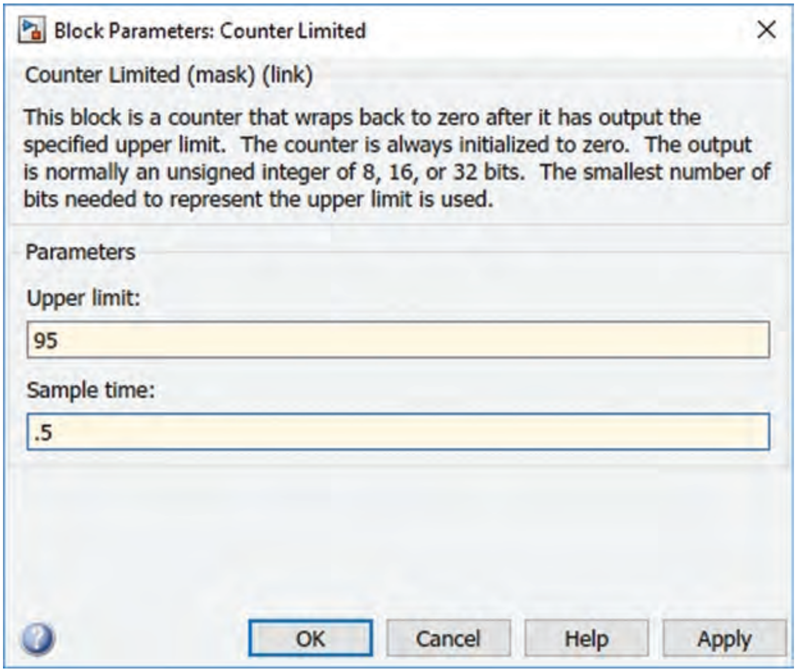


Figure 3.9

Connect an oscilloscope to Pin A8 and observe the increase in the duty cycle of a generated signal.

3.5 EXAMPLE 4: MEASUREMENT OF FREQUENCY, WIDTH +, AND DUTY CYCLE WITH PWM CAPTURE BLOCK

Simulink model of this example is shown in Fig. 3.10. In this example we want to measure the frequency, width of high portion, and duty cycle (= width of high portion of signal divided by the period of the signal) of an input signal applied to Pin B6. Input signal is a pulse signal. Aforementioned quantities can be measured with the aid of PWM Capture block.

Settings of blocks used in Fig. 3.10 are shown in Figs. 3.11, 3.12, and 3.13. Serial communication is studied in Chapter 5. Use the Docklight® program to receive and see the data that comes from the Discovery board. The Docklight can be downloaded from https://docklight.de/downloads/.

Upload the code to the board and use a signal generator to produce the input pulse. Connect the ground of signal generator block to the ground of Discovery board and connect the

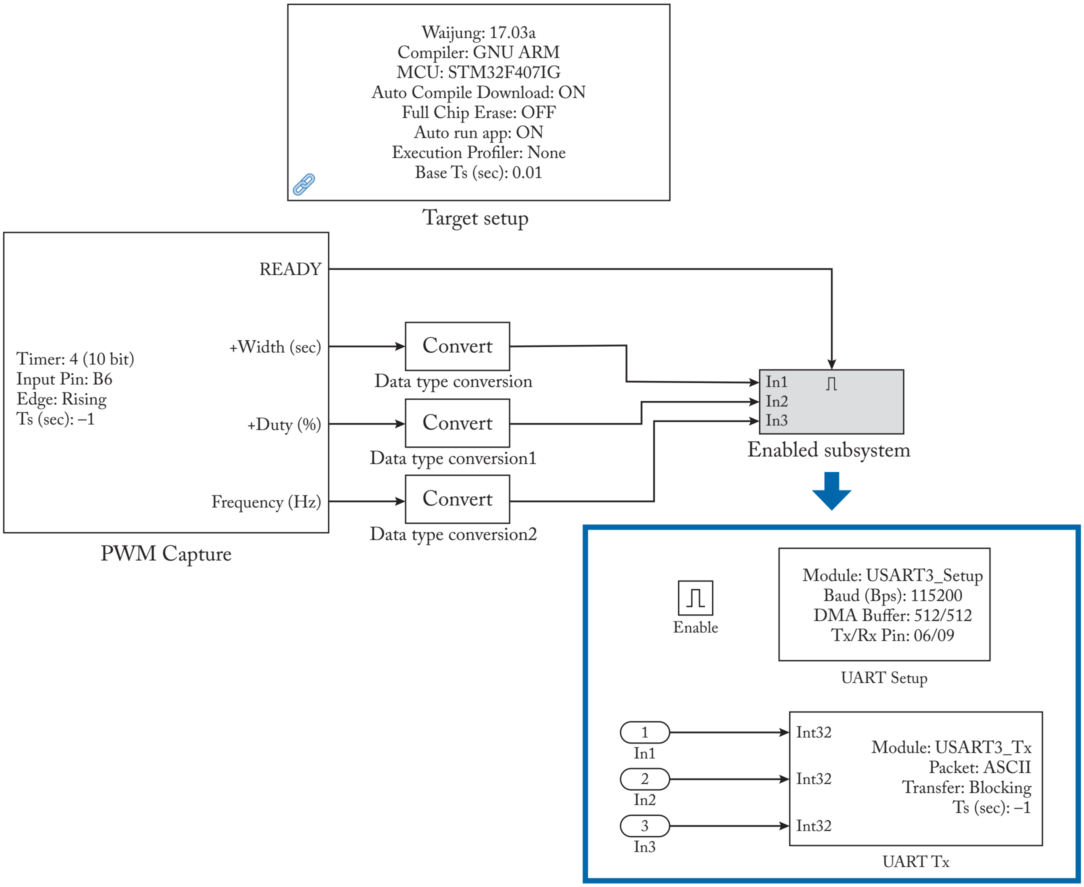


Figure 3.10

other wire to Pin PB6 of Discovery board. Figures 3.14–3.16 show the outputs for different input pulses.

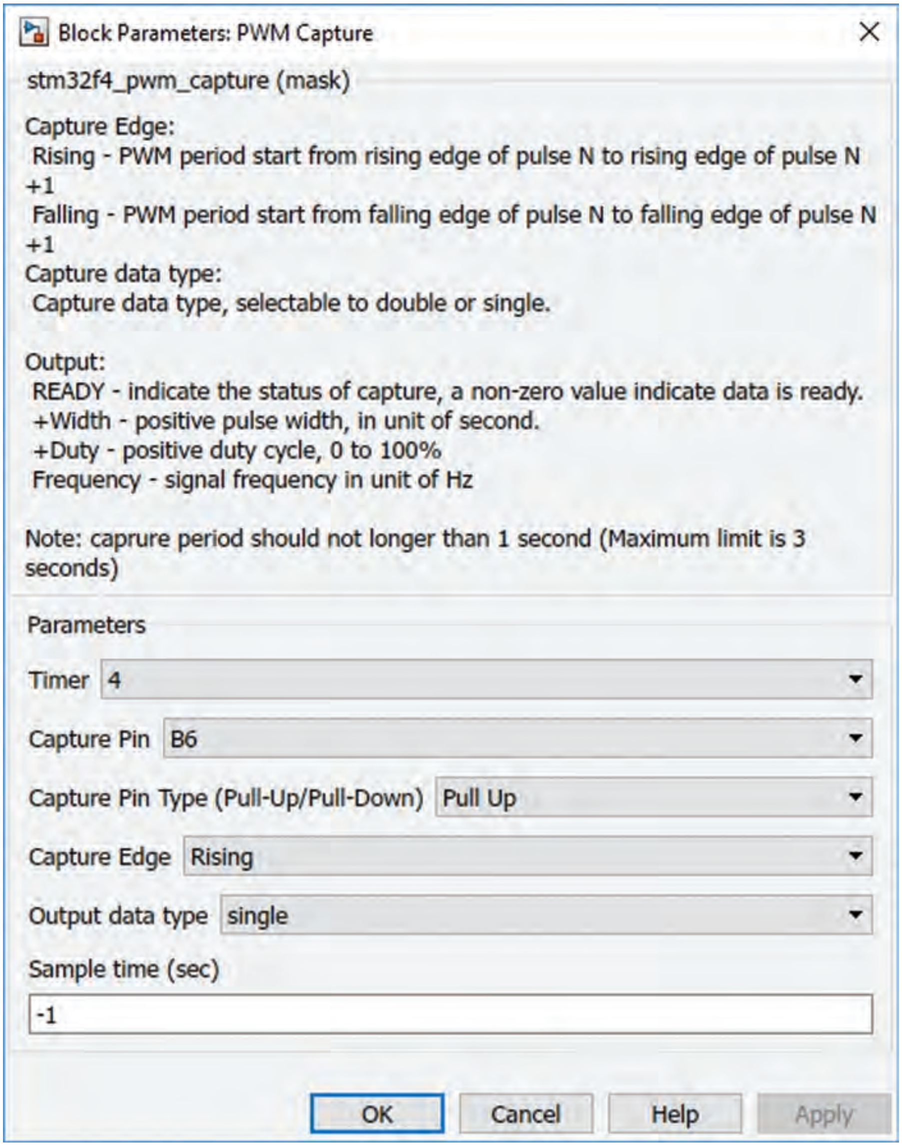

Block Parameters: PWM Capture
stm32f4_pwm_capture (mask)
Capture Edge:
Rising - PWM period start from rising edge of pulse N to rising edge of pulse N +1
Falling - PWM period start from falling edge of pulse N to falling edge of pulse N +1
Capture data type:
Capture data type, selectable to double or single.
Output:
READY - indicate the status of capture, a non-zero value indicate data is ready.
+Width - positive pulse width, in unit of second.
+Duty - positive duty cycle, 0 to 100%
Frequency - signal frequency in unit of Hz
Note: caprure period should not longer than 1 second (Maximum limit is 3 seconds)
Parameters
Timer
4
Capture Pin
B6
Capture Pin Type (Pull-Up/Pull-Down)
Pull Up
Capture Edge
Rising
Output data type
single
Sample time (sec)
-1
OK
Cancel
Help
Apply

Figure 3.11

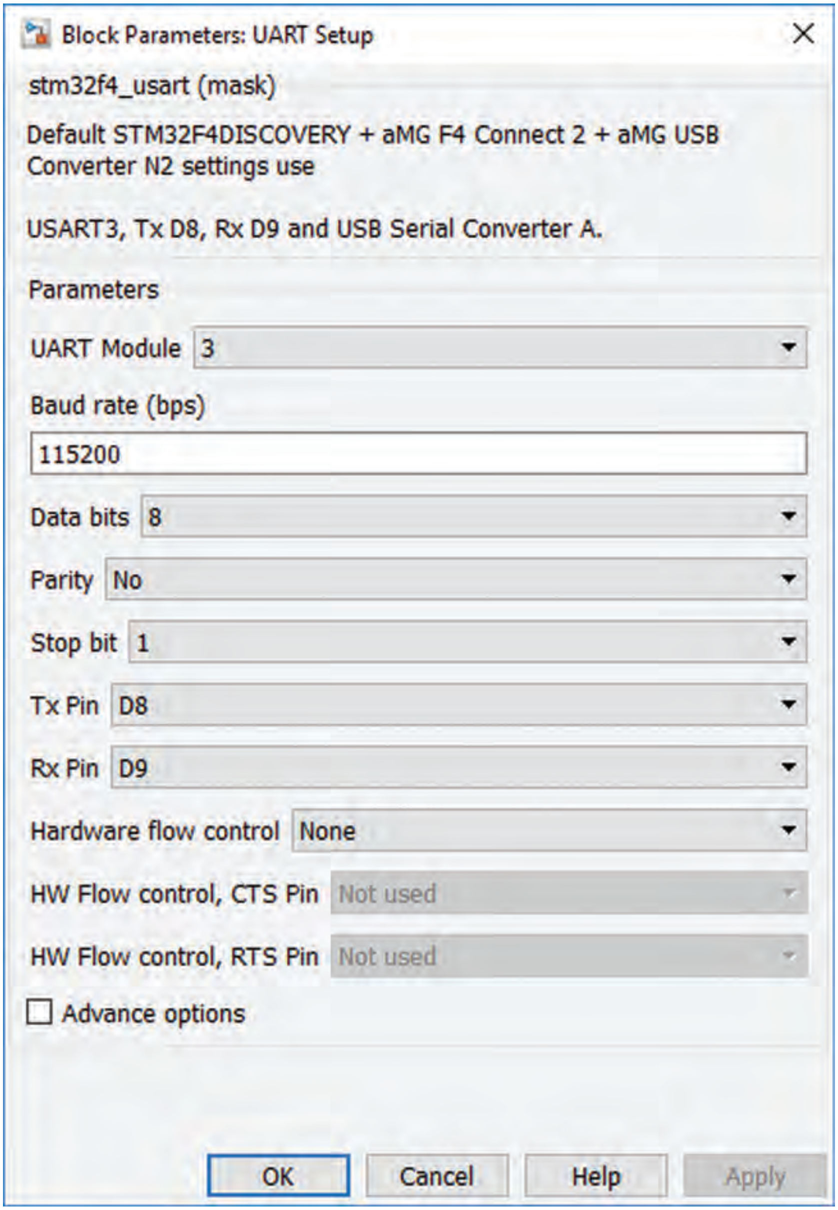

Block Parameters: UART Setup
stm32f4_usart (mask)
Default STM32F4DISCOVERY + aMG F4 Connect 2 + aMG USB Converter N2 settings use
USART3, Tx D8, Rx D9 and USB Serial Converter A.
Parameters
UART Module
3
Baud rate (bps)
115200
Data bits
8
Parity
No
Stop bit
1
Tx Pin
D8
Rx Pin
D9
Hardware flow control
None
HW Flow control, CTS Pin
Not used
HW Flow control, RTS Pin
Not used
Advance options
OK
Cancel
Help
Apply

Figure 3.12

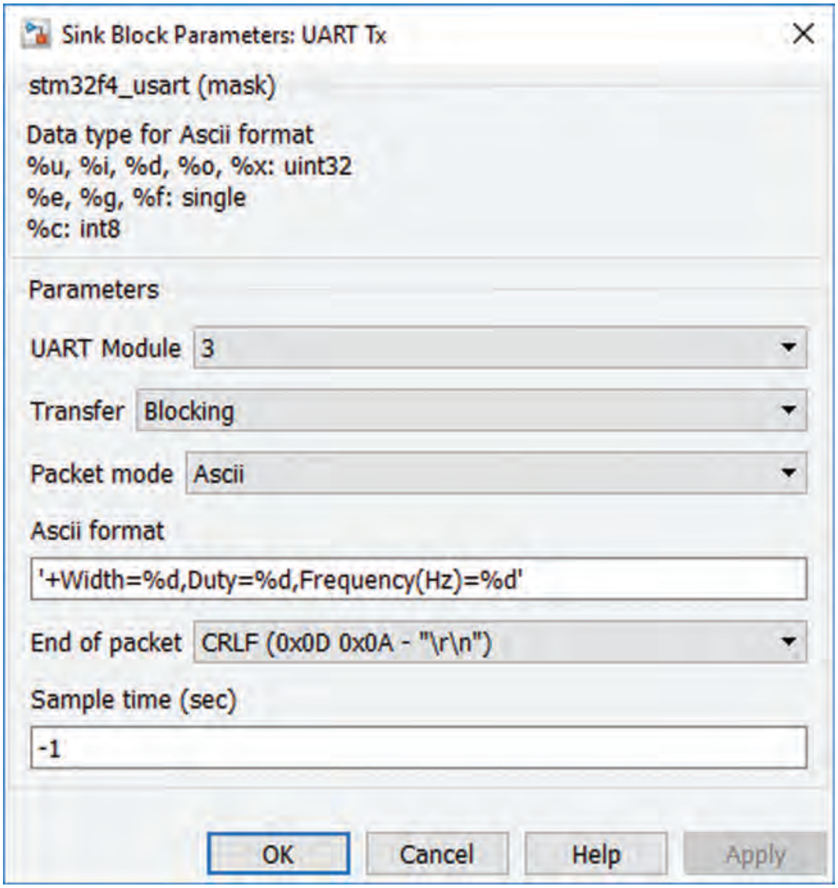

Sink Block Parameters: UART Tx
stm32f4_usart (mask)
Data type for Ascii format
%u, %i, %d, %o, %x: uint32
%e, %g, %f: single
%c: int8
Parameters
UART Module
3
Transfer
Blocking
Packet mode
Ascii
Ascii format
'+Width=%d,Duty=%d,Frequency(Hz)=%d'
End of packet
CRLF (0x0D 0x0A - "\r\n")
Sample time (sec)
-1
OK
Cancel
Help
Apply

Figure 3.13

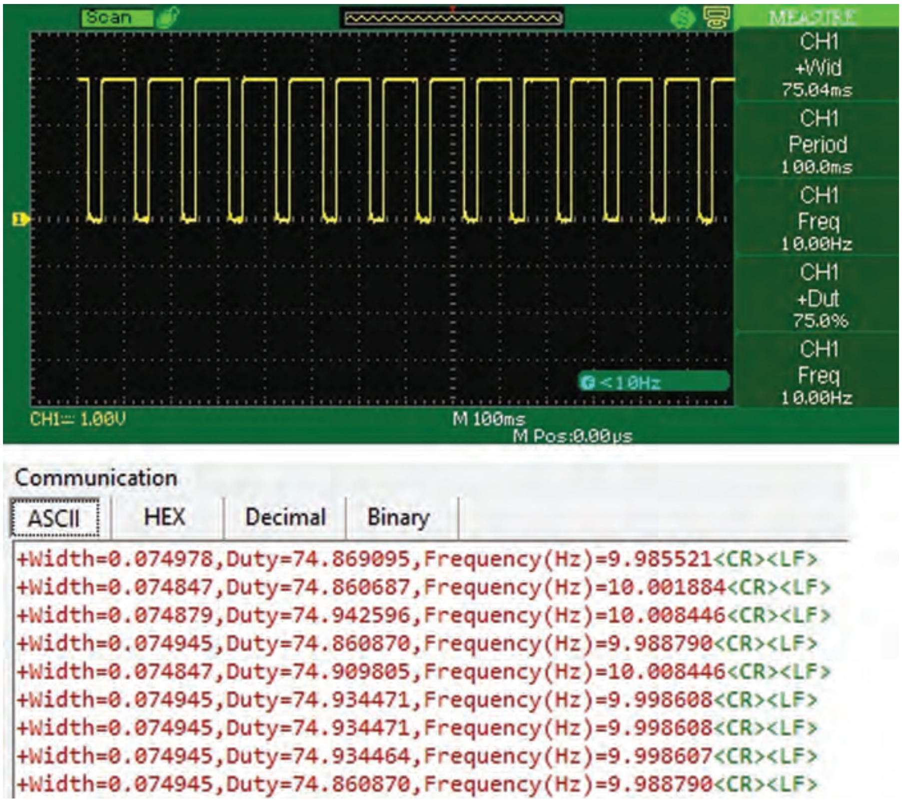


Figure 3.14: Low frequency (10 Hz).

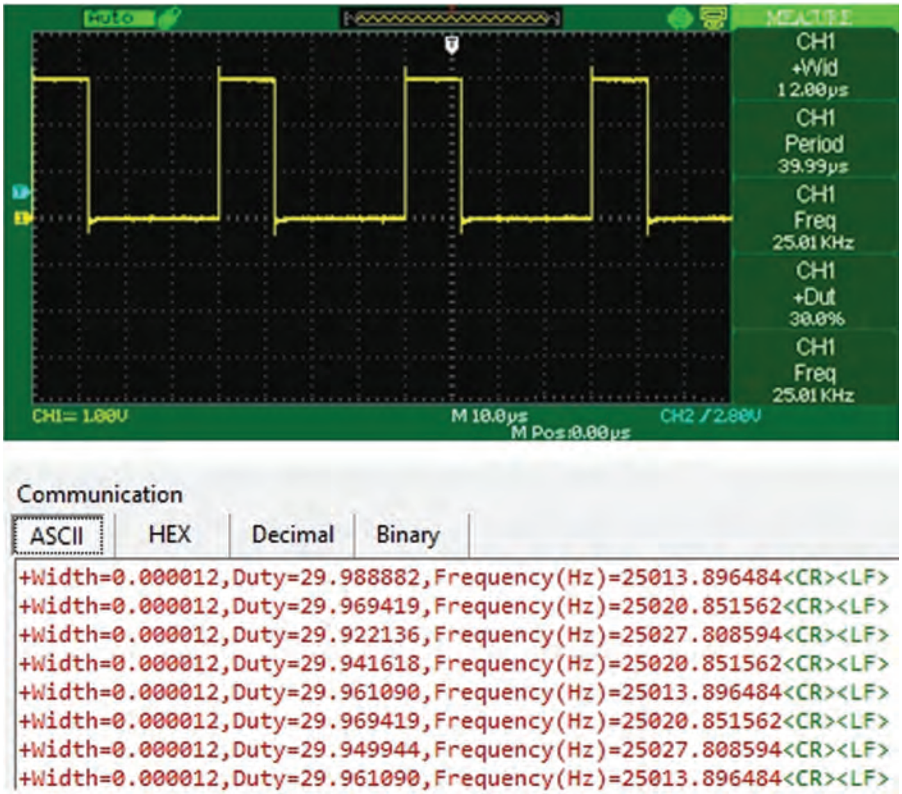


Figure 3.15: High frequency 1 (25 kHz).

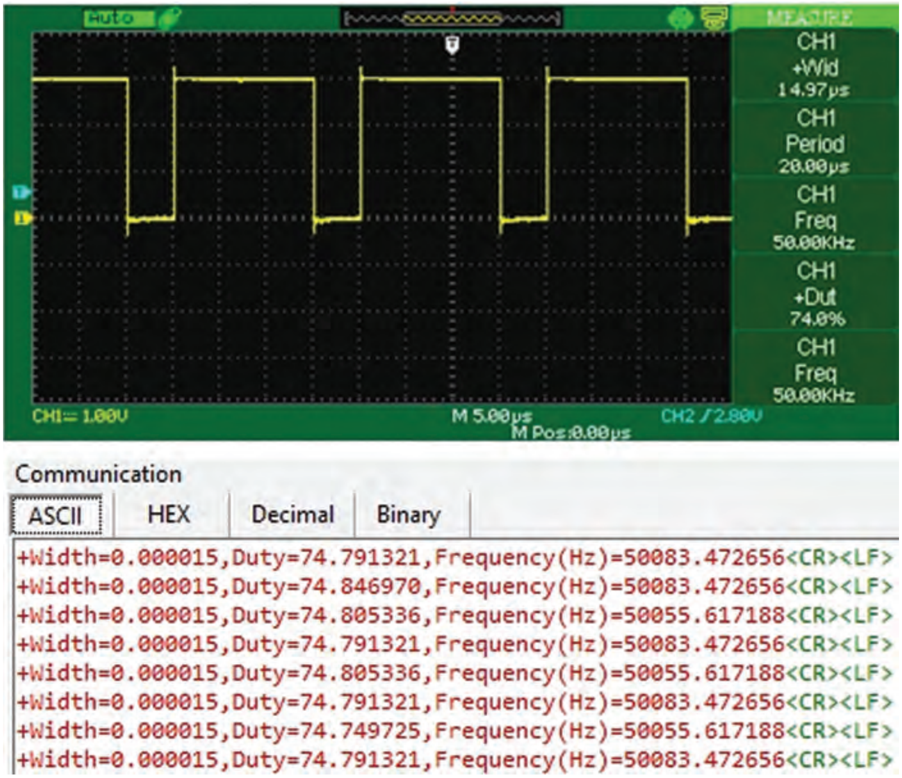


Figure 3.16: High frequency 2 (50 kHz).

3.6 EXAMPLE 5: CONTROLLING A DC MOTOR

The schematic of this example is shown in Fig. 3.17. In this example, when you press the on-board switch connected to Pin PA0, a PWM signal with duty cycle of 50% appears on the Pin A8. When PA0 is not pressed, no signal is available on the Pin PA8. If you connect the Pin PA8 to "PWM IN" of a DC motor driver, then the motor starts to rotate when PA0 is pressed (Fig. 3.18).

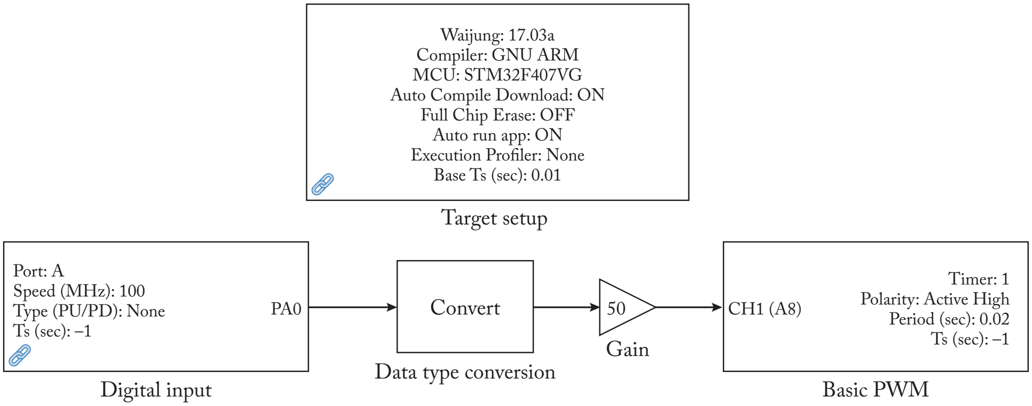


Figure 3.17

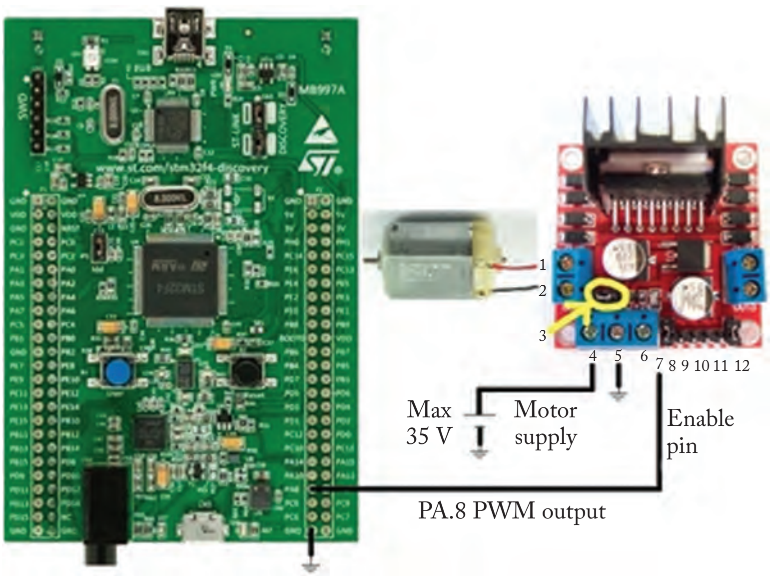


Figure 3.18

CHAPTER 4

Analog to Digital Conversion and Timer

4.1 INTRODUCTION

The Analog to Digital Converter (ADC) block permits you to generate an analog signal. The Timer block is used to execute some actions periodically. This chapter focuses on these two blocks.

4.2 EXAMPLE 1: REGULAR ADC BLOCK

The simulink model of this example is shown in Fig. 4.1. In this example we will use a potentiometer block to control the on-board LEDs. The potentiometer block is connected to the port PA5 (Fig. 4.2). In Fig. 4.1, the regular ADC block reads the voltage of pin A5 and generates a value between 0 and 4095. 0 shows 0 V and 4095 shows 3 V. So, if we multiply the output of the block by 3/4095, we will obtain the input analog voltage value. Note that the voltage entering the Discovery board must be between 0 and +3 V. Larger and negative voltages may damage the board.

According to the diagram shown in Fig. 4.1, if the input voltage applied to pin PA5 is greater than,

1. $1000 \times \frac{3}{4095} = 0.733$ V, then the on-board green LED connected to PD 12 turns on.
2. $2000 \times \frac{3}{4095} = 1.465$ V, then the on-board orange LED connected to PD 13 turns on.
3. $3000 \times \frac{3}{4095} = 2.198$ V, then the on-board red LED connected to PD 14 turns on.
4. $4000 \times \frac{3}{4095} = 2.930$ V, then the on-board blue LED connected to PD 15 turns on.

Settings of the ADC block in Fig. 4.1 are shown in Fig. 4.3.

Upload the Simulink model to the board. Rotate the potentiometer and measure the voltages that turn on each LED. Compare them with values given above. As another example, upload the Simulink model shown in Fig. 4.4 and measure the voltages that turns on the LEDs.

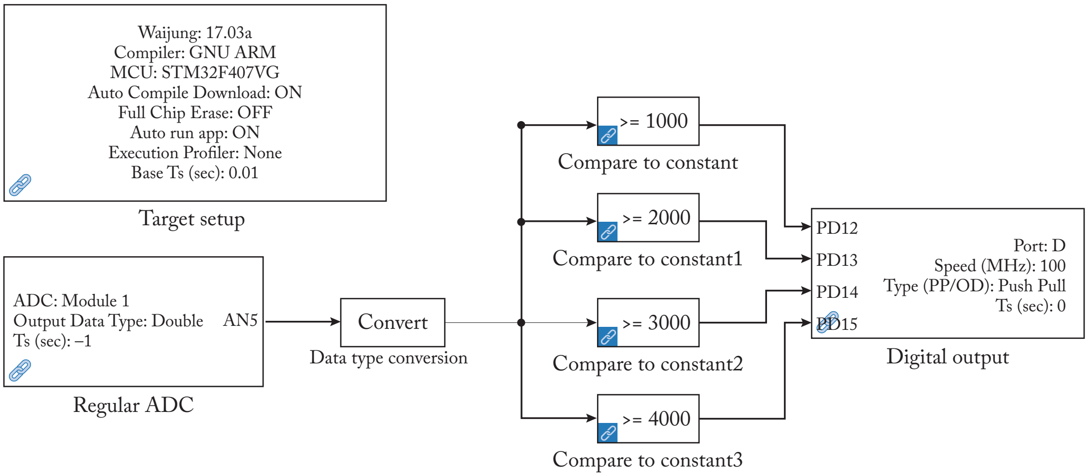


Figure 4.1

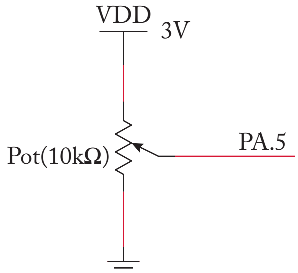


Figure 4.2

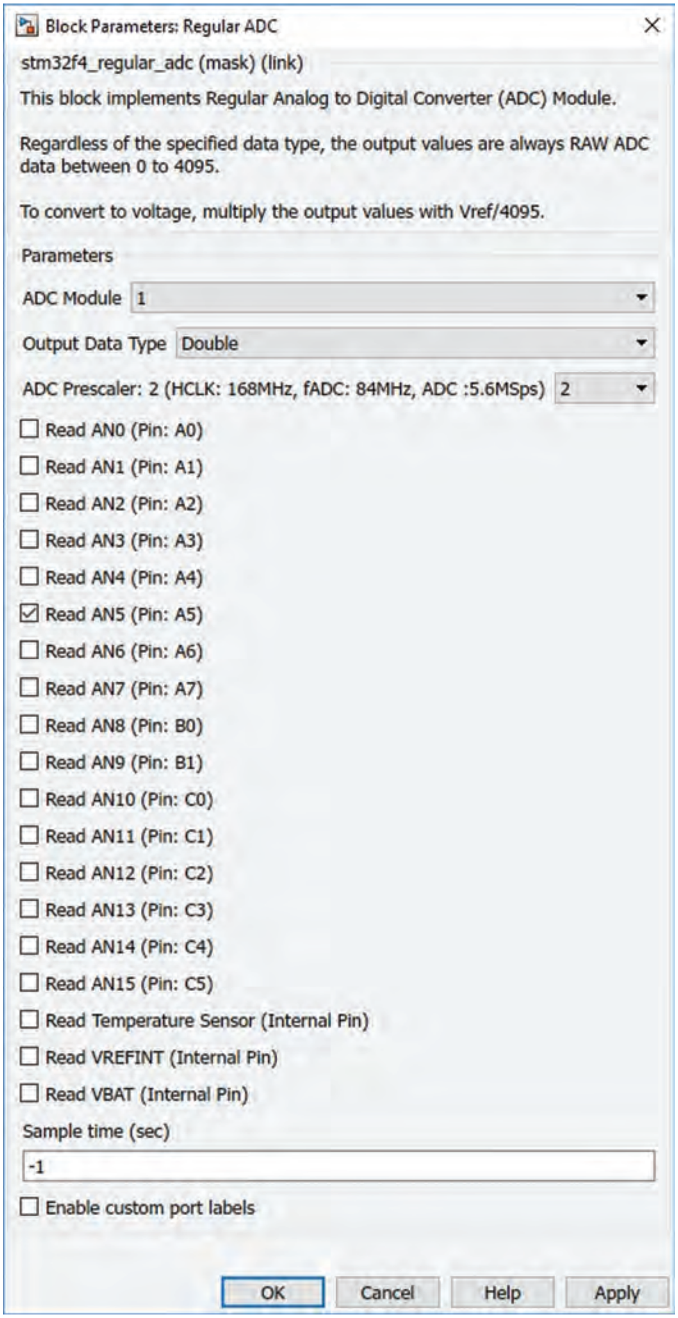

Block Parameters: Regular ADC
stm32f4_regular_adc (mask) (link)
This block implements Regular Analog to Digital Converter (ADC) Module.
Regardless of the specified data type, the output values are always RAW ADC data between 0 to 4095.
To convert to voltage, multiply the output values with Vref/4095.
Parameters
ADC Module 1
Output Data Type Double
ADC Prescaler: 2 (HCLK: 168MHz, fADC: 84MHz, ADC :5.6MSps) 2
Read AN0 (Pin: A0)
Read AN1 (Pin: A1)
Read AN2 (Pin: A2)
Read AN3 (Pin: A3)
Read AN4 (Pin: A4)
Read AN5 (Pin: A5)
Read AN6 (Pin: A6)
Read AN7 (Pin: A7)
Read AN8 (Pin: B0)
Read AN9 (Pin: B1)
Read AN10 (Pin: C0)
Read AN11 (Pin: C1)
Read AN12 (Pin: C2)
Read AN13 (Pin: C3)
Read AN14 (Pin: C4)
Read AN15 (Pin: C5)
Read Temperature Sensor (Internal Pin)
Read VREFINT (Internal Pin)
Read VBAT (Internal Pin)
Sample time (sec)
-1
Enable custom port labels
OK
Cancel
Help
Apply

Figure 4.3

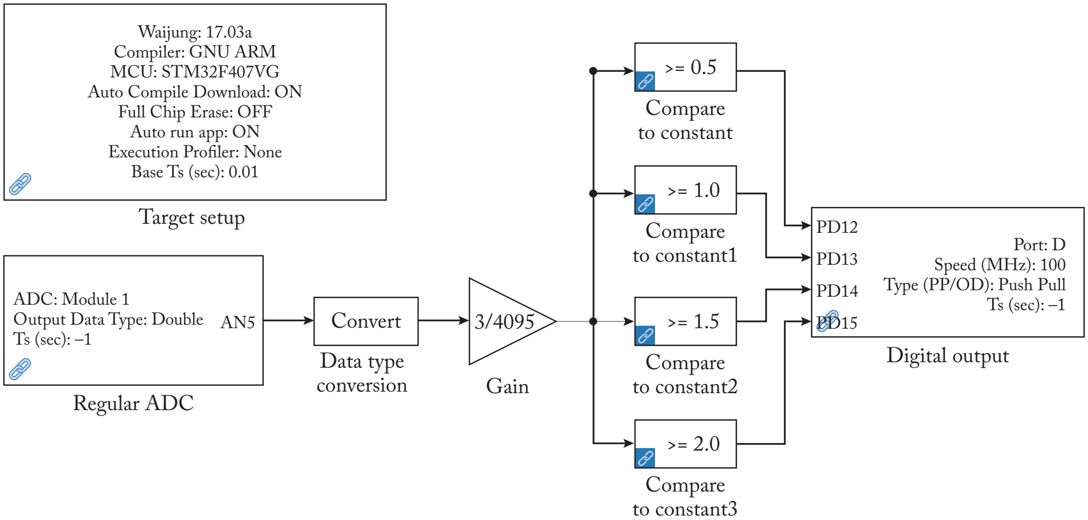
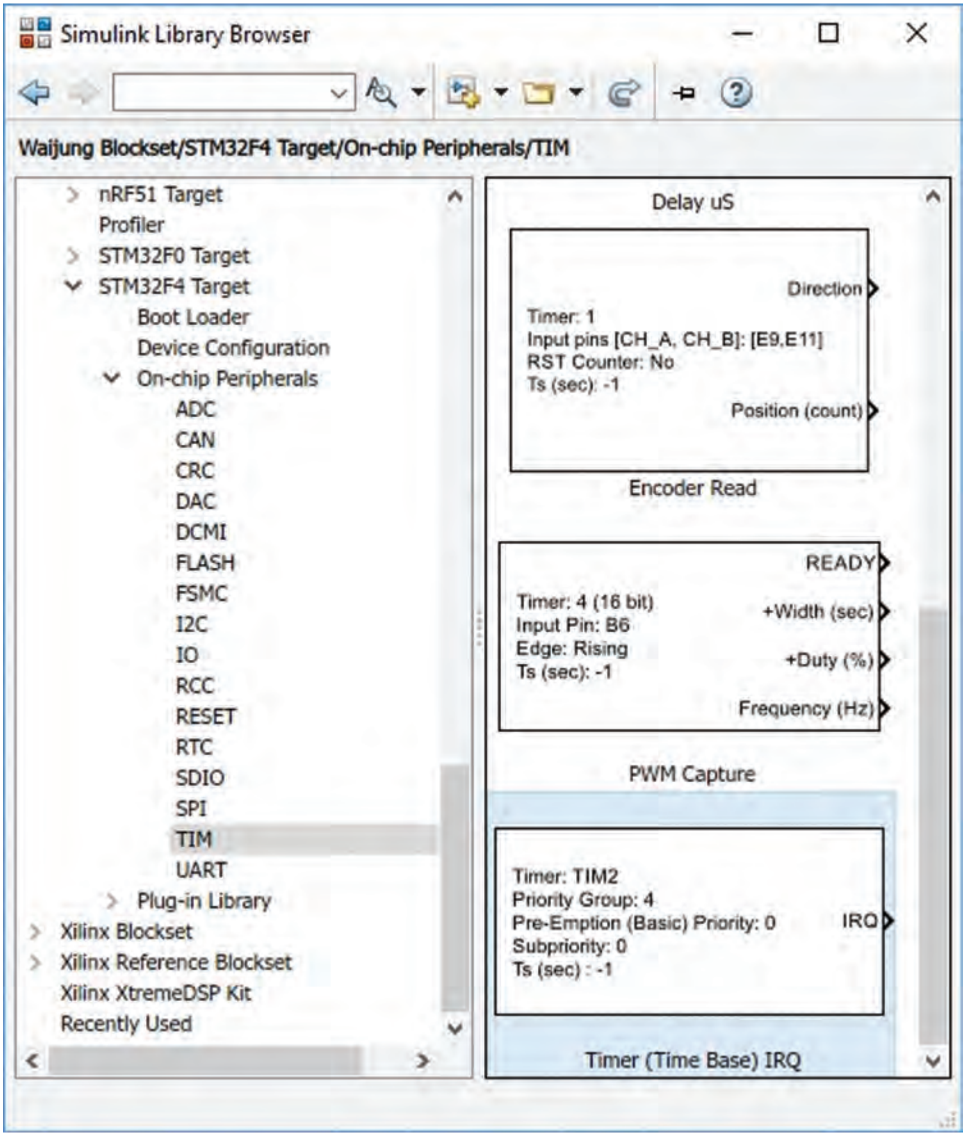


Figure 4.4

4.3 EXAMPLE 2: TIMER BLOCK

Timer block (Fig. 4.5) is used to execute some actions periodically. If you double click the Timer block, the window shown in Fig. 4.6 appears. The amount of time which actions are repeated must be entered into the Sample time (sec). [No greater than 47.7204 sec] box. The actions that must be done are determined with the aid of Function-Call Subsystem block (Fig. 4.7).

Let's study a simple example. The block diagram shown in Fig. 4.8 change the status of the on-board green LED connected to pin PD12 each 1 sec. So, the LED is off for 1 sec and it is on for 1 sec and this process repeats. Settings of the Timer and Memory blocks are shown in Figs. 4.9 and 4.10, respectively.



Simulink Library Browser
Waijung Blockset/STM32F4 Target/On-chip Peripherals/TIM
nRF51 Target
Profiler
STM32F0 Target
STM32F4 Target
Boot Loader
Device Configuration
On-chip Peripherals
ADC
CAN
CRC
DAC
DCMI
FLASH
FSMC
I2C
IO
RCC
RESET
RTC
SDIO
SPI
TIM
UART
Plug-in Library
Xilinx Blockset
Xilinx Reference Blockset
Xilinx XtremeDSP Kit
Recently Used
Delay uS
Direction
Timer: 1
Input pins [CH_A, CH_B]: [E9,E11]
RST Counter: No
Ts (sec): -1
Position (count)
Encoder Read
READY
Timer: 4 (16 bit)
Input Pin: B6
Edge: Rising
Ts (sec): -1
+Width (sec)
+Duty (%)
Frequency (Hz)
PWM Capture
Timer: TIM2
Priority Group: 4
Pre-Emption (Basic) Priority: 0
Subpriority: 0
Ts (sec) : -1
IRQ
Timer (Time Base) IRQ

Figure 4.5

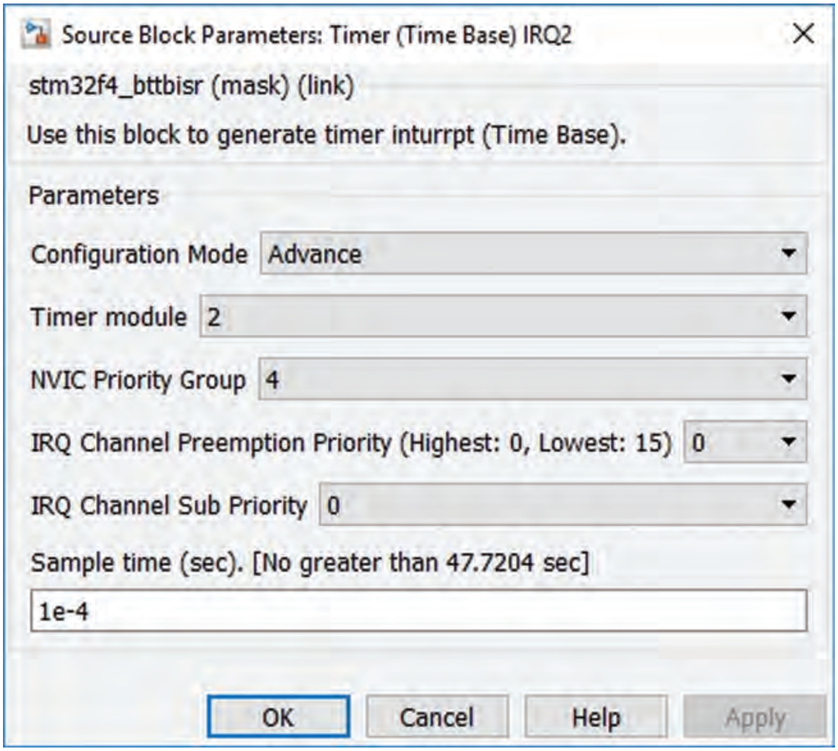
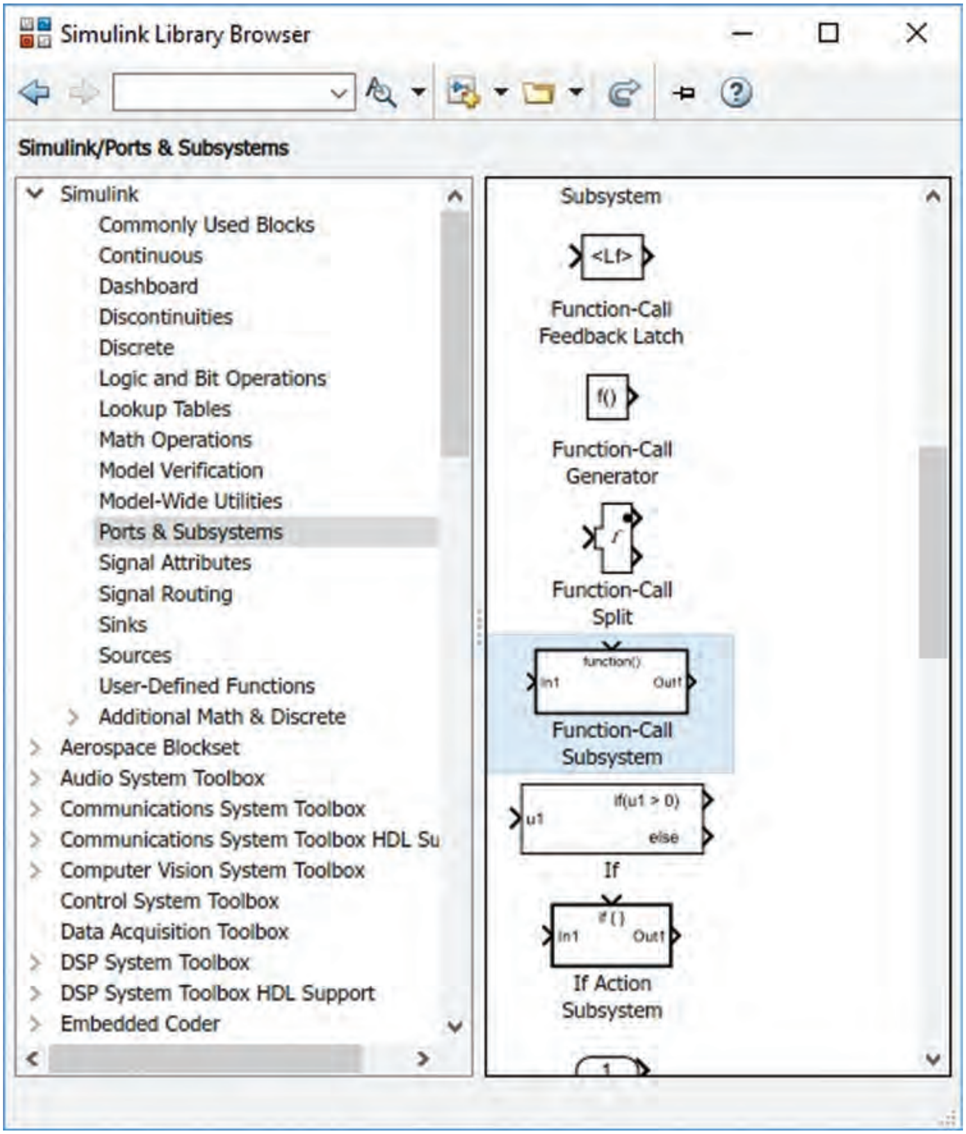

Source Block Parameters: Timer (Time Base) IRQ2
stm32f4_bttbisr (mask) (link)
Use this block to generate timer inturrpt (Time Base).
Parameters
Configuration Mode
Advance
Timer module
2
NVIC Priority Group
4
IRQ Channel Preemption Priority (Highest: 0, Lowest: 15)
0
IRQ Channel Sub Priority
0
Sample time (sec). [No greater than 47.7204 sec]
1e-4
OK
Cancel
Help
Apply

Figure 4.6



Simulink Library Browser
Simulink/Ports & Subsystems
Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
User-Defined Functions
Additional Math & Discrete
Aerospace Blockset
Audio System Toolbox
Communications System Toolbox
Communications System Toolbox HDL Su
Computer Vision System Toolbox
Control System Toolbox
Data Acquisition Toolbox
DSP System Toolbox
DSP System Toolbox HDL Support
Embedded Coder
Subsystem
<Lf>
Function-Call Feedback Latch
f()
Function-Call Generator
Function-Call Split
function()
In1
Out1
Function-Call Subsystem
u1
if(u1 > 0)
else
If
if { }
In1
Out1
If Action Subsystem
1

Figure 4.7

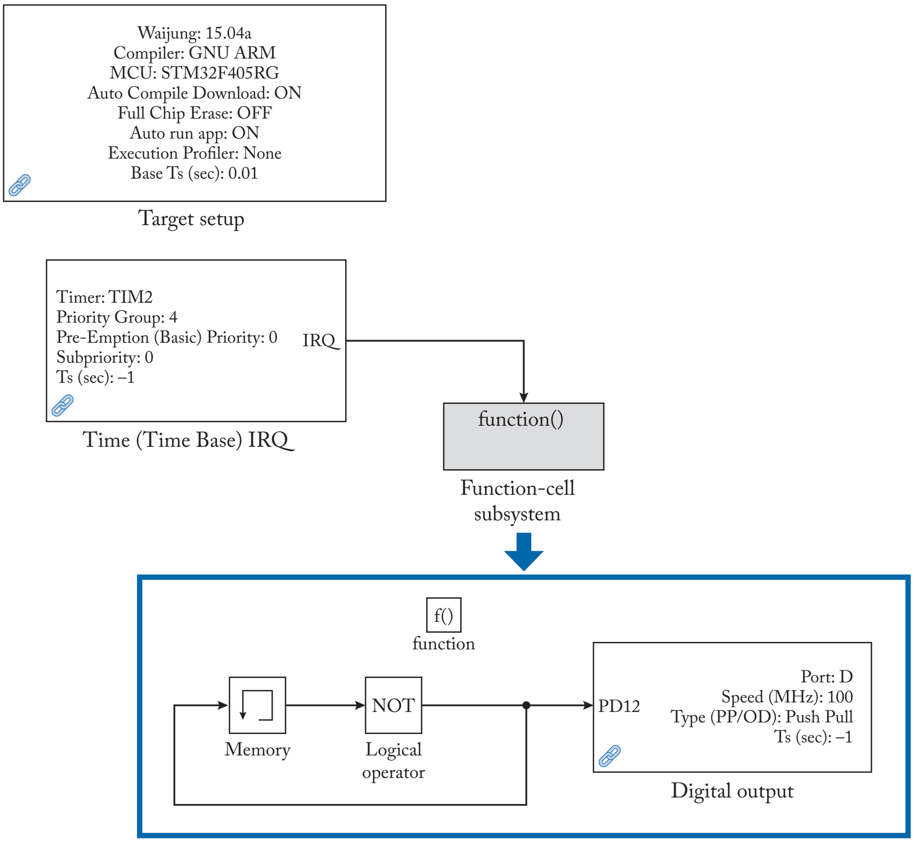

Waijung: 15.04a
Compiler: GNU ARM
MCU: STM32F405RG
Auto Compile Download: ON
Full Chip Erase: OFF
Auto run app: ON
Execution Profiler: None
Base Ts (sec): 0.01
Target setup
Timer: TIM2
Priority Group: 4
Pre-Emption (Basic) Priority: 0
Subpriority: 0
Ts (sec): –1
IRQ
Time (Time Base) IRQ
function()
Function-cell
subsystem
f()
function
Memory
NOT
Logical
operator
PD12
Port: D
Speed (MHz): 100
Type (PP/OD): Push Pull
Ts (sec): –1
Digital output

Figure 4.8

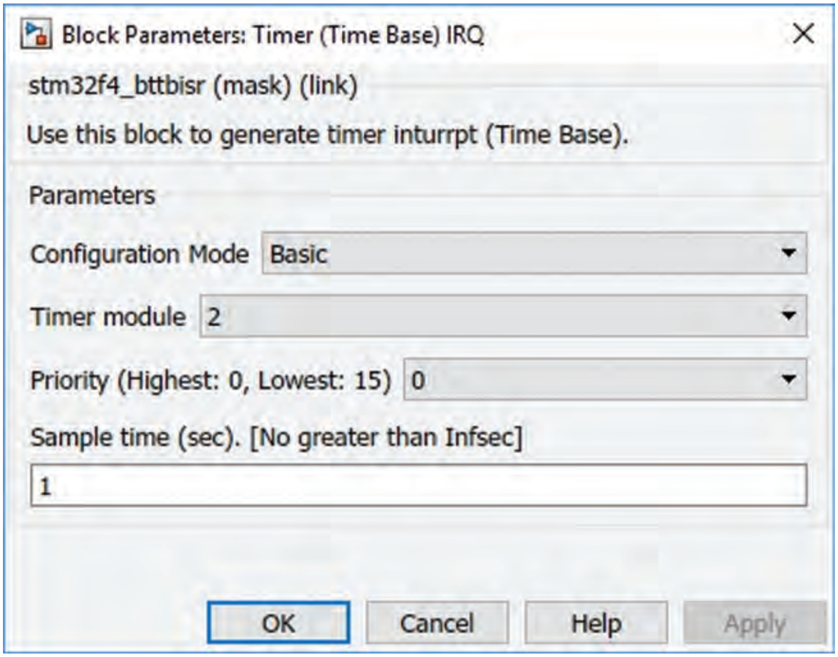


Figure 4.9

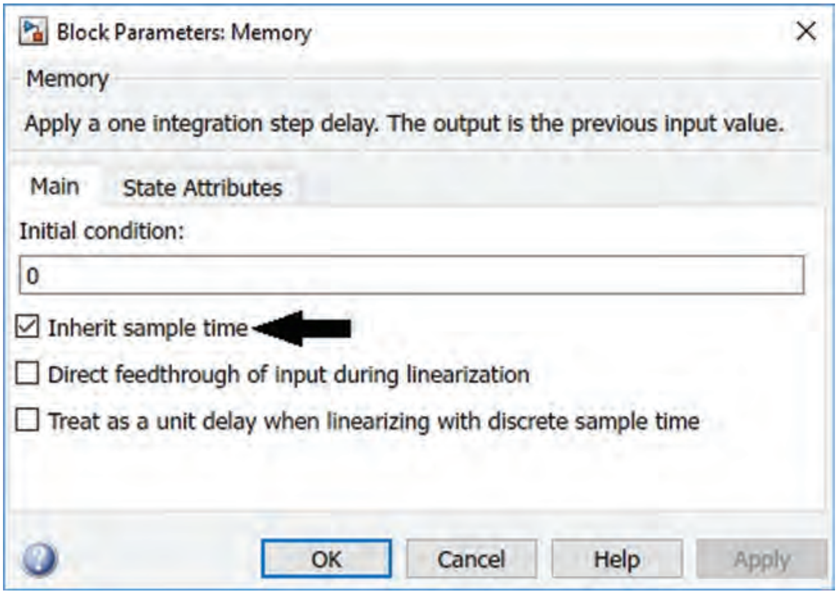


Figure 4.10

4.4 EXAMPLE 3: GENERATION OF ANALOG WAVEFORMS

You can generate an analog signal with the aid of Regular DAC block (Fig. 4.11).

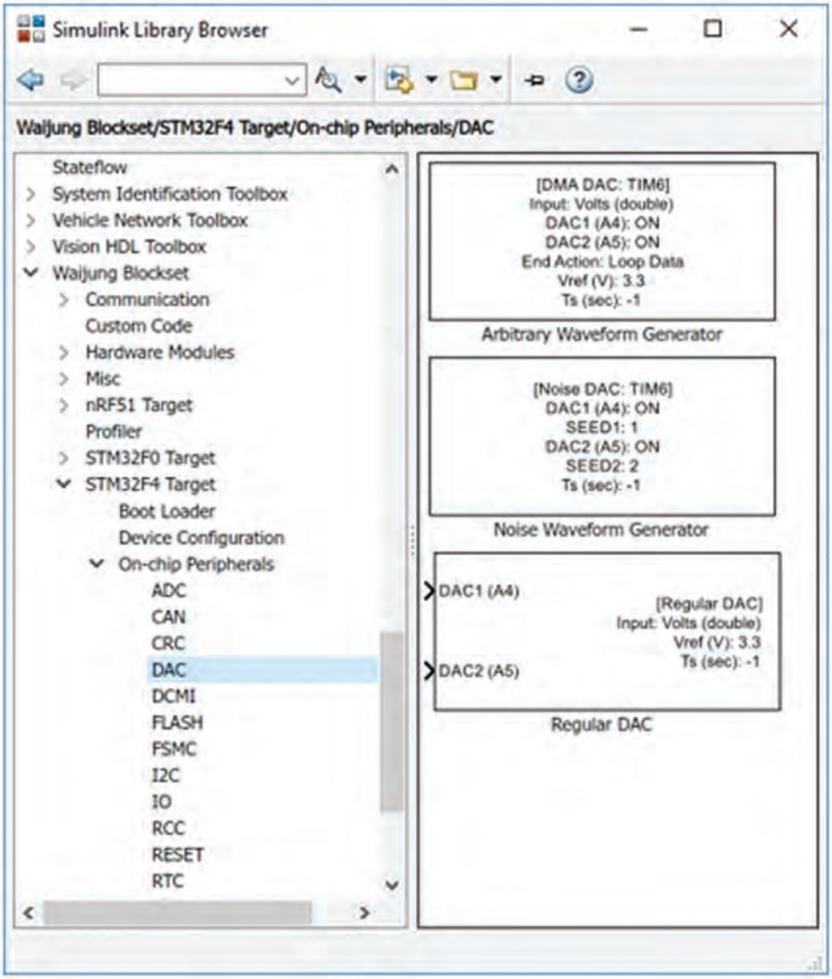


Figure 4.11

If you double click on the Regular DAC block, the window shown in Fig. 4.12 appears on the screen. The Input Type has two types of options: Volts and Raw × Bits.

When you select Volts (double) or Volts (single), the output of block equals to the value which enters to the block. In other words, $V_{out} = Input$.

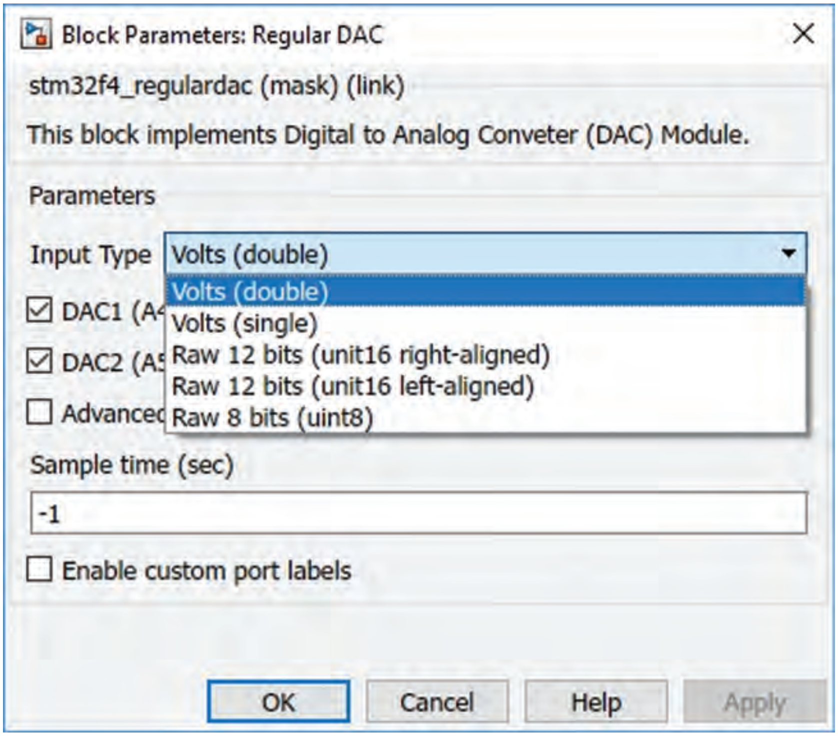


Figure 4.12

When you select Raw × bits, the output of block equals to $V_{out} = \frac{V_{Ref}}{4095} \times Input$. Input Vref box (Fig. 4.13) determines the value of V_{Ref}.

Let's study an example. The Simulink model in Fig. 4.14 generates the $v_{out}(t) = 0.7 + \frac{1}{2} \times \sin(2\pi \times 1000 \times t) + \frac{1}{4} \times \sin(2\pi \times 3000 \times t)$. Graph of this function is shown in Fig. 4.15. According to Fig. 4.15, the function is not negative and its values are less than 3 V. So, it can be generated by the Discovery board. Settings of the blocks used in Fig. 4.14 are shown in Figs. 4.16–4.18.

Upload the Simulink model to the board and use an oscilloscope to see the voltage of pin PA4. Result is shown in Fig. 4.19. Enter 1/200/1000 to the Sample time boxes in Figs. 4.17 and 4.18. This time the waveform shown in Fig. 4.20 is obtained.

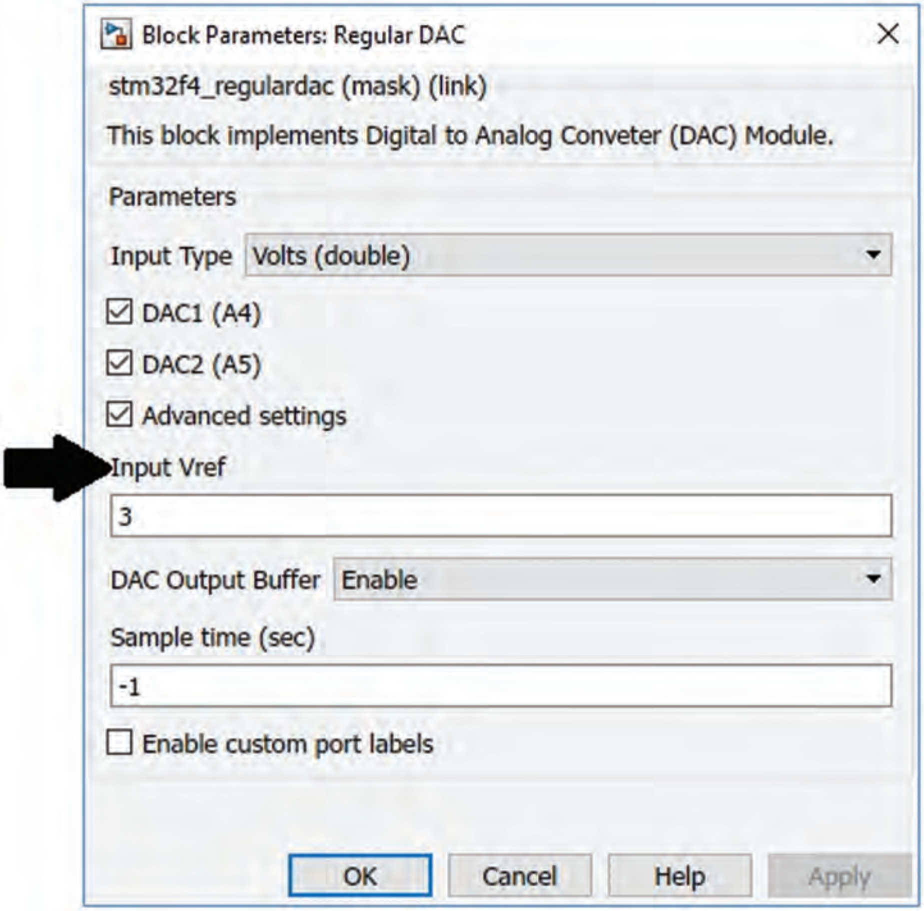

Block Parameters: Regular DAC
stm32f4_regulardac (mask) (link)
This block implements Digital to Analog Conveter (DAC) Module.
Parameters
Input Type
Volts (double)
DAC1 (A4)
DAC2 (A5)
Advanced settings
Input Vref
3
DAC Output Buffer
Enable
Sample time (sec)
-1
Enable custom port labels
OK
Cancel
Help
Apply

Figure 4.13

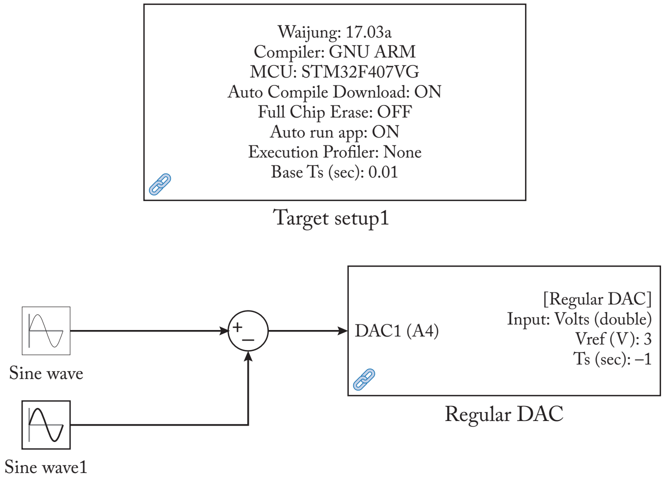


Figure 4.14

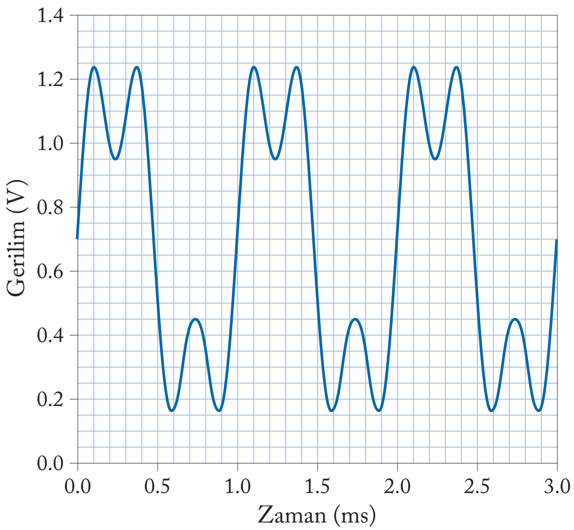


Figure 4.15

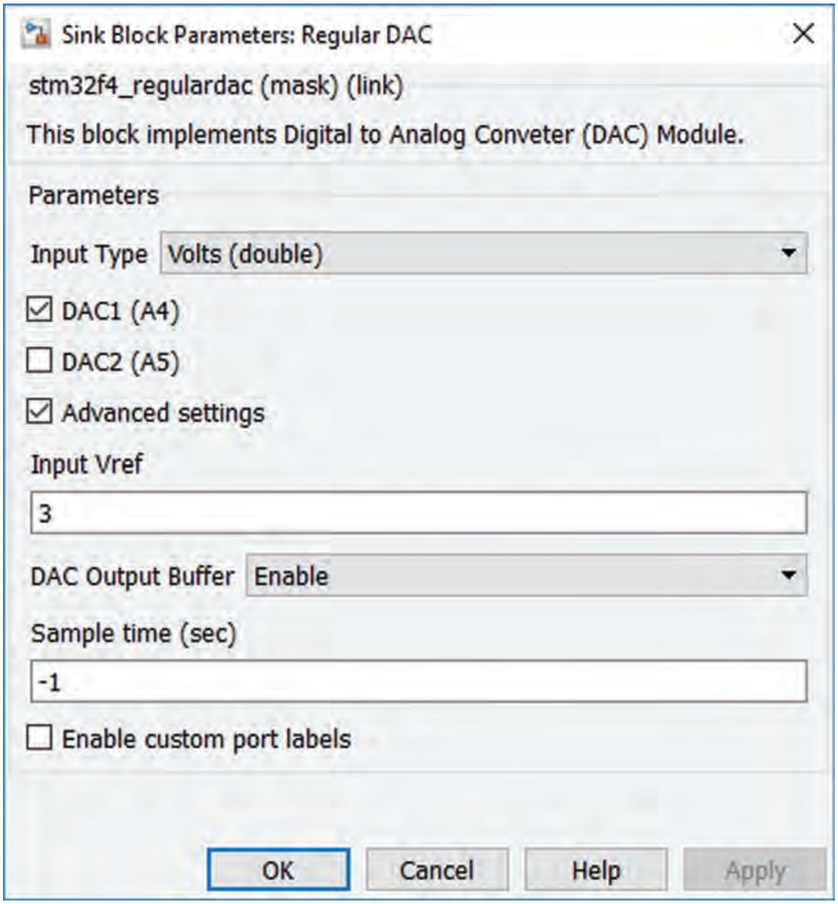

Sink Block Parameters: Regular DAC
stm32f4_regulardac (mask) (link)
This block implements Digital to Analog Conveter (DAC) Module.
Parameters
Input Type
Volts (double)
DAC1 (A4)
DAC2 (A5)
Advanced settings
Input Vref
3
DAC Output Buffer
Enable
Sample time (sec)
-1
Enable custom port labels
OK
Cancel
Help
Apply

Figure 4.16

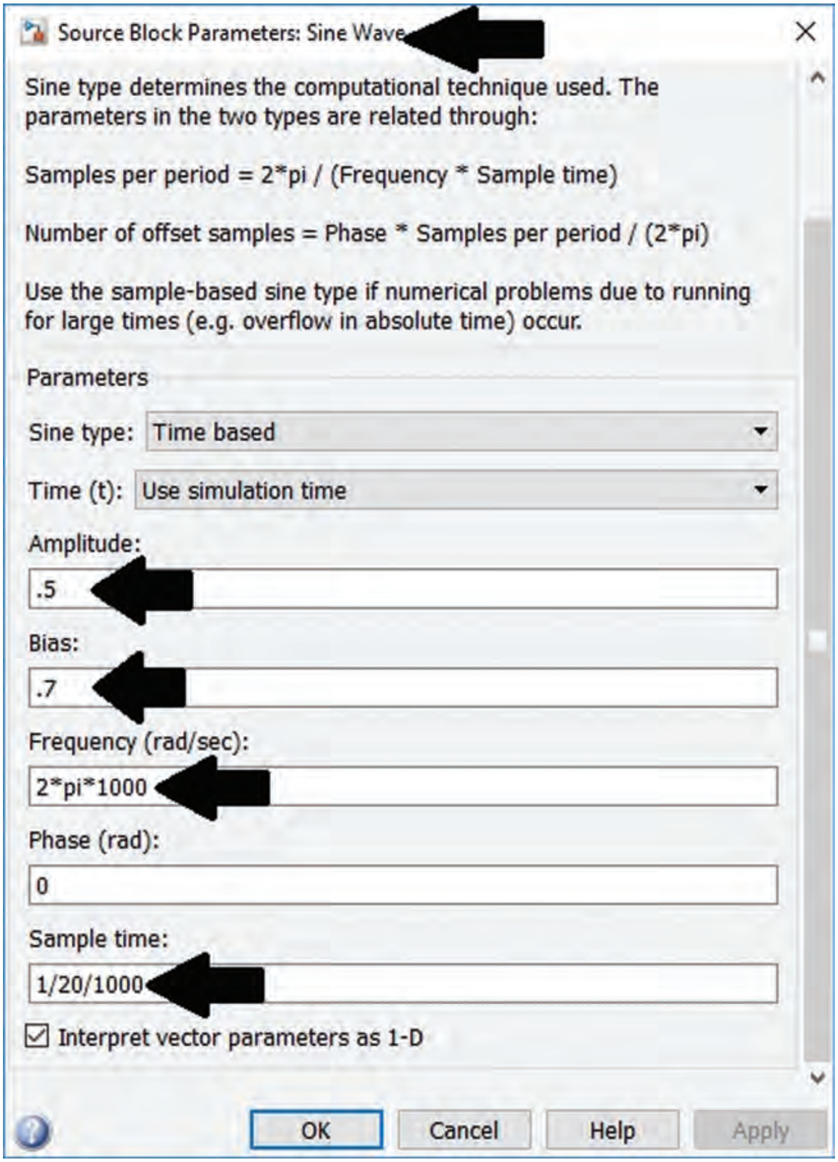

Source Block Parameters: Sine Wave
Sine type determines the computational technique used. The parameters in the two types are related through:
Samples per period = 2*pi / (Frequency * Sample time)
Number of offset samples = Phase * Samples per period / (2*pi)
Use the sample-based sine type if numerical problems due to running for large times (e.g. overflow in absolute time) occur.
Parameters
Sine type: Time based
Time (t): Use simulation time
Amplitude:
.5
Bias:
.7
Frequency (rad/sec):
2*pi*1000
Phase (rad):
0
Sample time:
1/20/1000
Interpret vector parameters as 1-D
OK
Cancel
Help
Apply

Figure 4.17

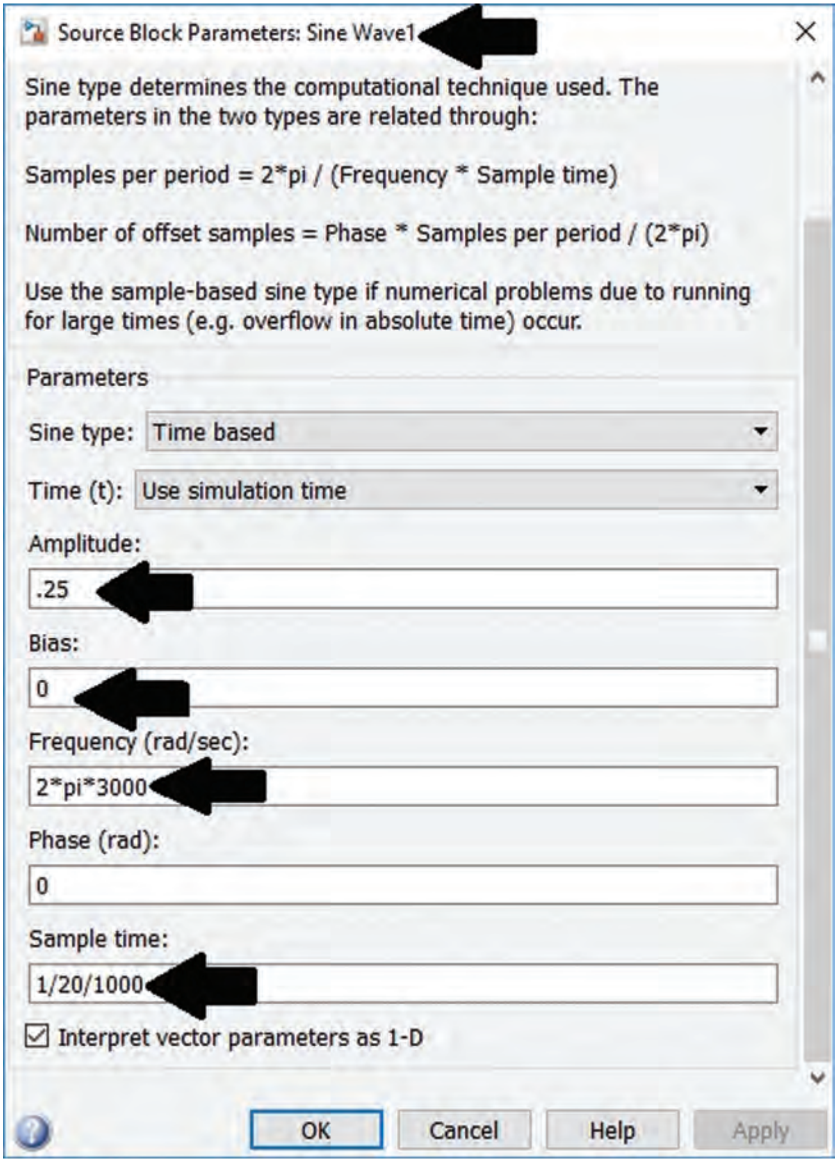

Source Block Parameters: Sine Wave1
Sine type determines the computational technique used. The parameters in the two types are related through:
Samples per period = 2*pi / (Frequency * Sample time)
Number of offset samples = Phase * Samples per period / (2*pi)
Use the sample-based sine type if numerical problems due to running for large times (e.g. overflow in absolute time) occur.
Parameters
Sine type: Time based
Time (t): Use simulation time
Amplitude:
.25
Bias:
0
Frequency (rad/sec):
2*pi*3000
Phase (rad):
0
Sample time:
1/20/1000
Interpret vector parameters as 1-D
OK
Cancel
Help
Apply

Figure 4.18

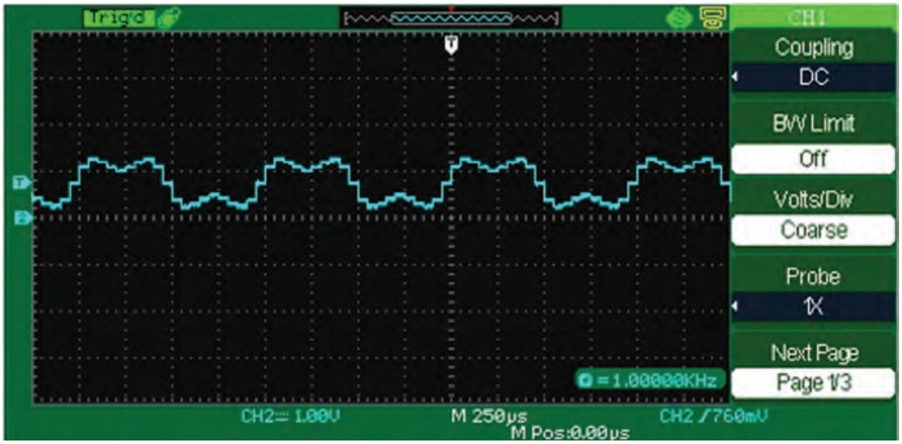


Figure 4.19: Waveform for Sample time = 1/20/1000.

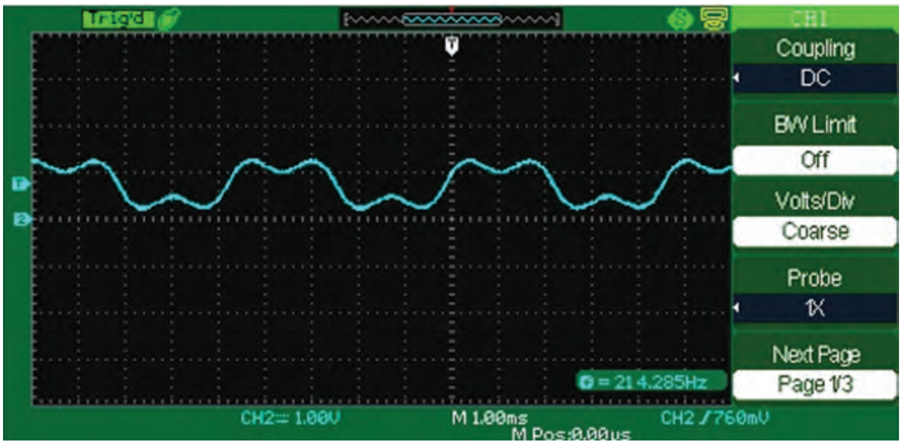


Figure 4.20: Waveform for Sample time = 1/200/1000.

CHAPTER 5

Serial Communication

5.1 INTRODUCTION

Serial communication permits the microcontroller to speak with the real world, i.e., receive/send data from/to outside. This chapter studies the blocks related to serial communication.

5.2 EXAMPLE 1: SERIAL COMMUNICATION (I)

The UART Tx block (Fig. 5.1) permits the Discovery board to send out the data to outside.

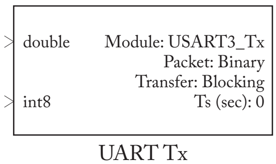


Figure 5.1

The UART Rx block (Fig. 5.2) permits the Discovery board to receive the data from outside.

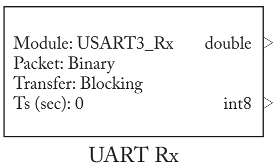


Figure 5.2

When you want to use serial communications, your model must contain a UART Setup block. Settings of UART block are shown in Fig. 5.3. The Baud rate (bps) box determines the speed of transfer.

Let's study an example. The Simulink model shown in Fig. 5.4 sends out the data generated by the Counter Limited block. Settings of used blocks are shown in Figs. 5.5–5.7.

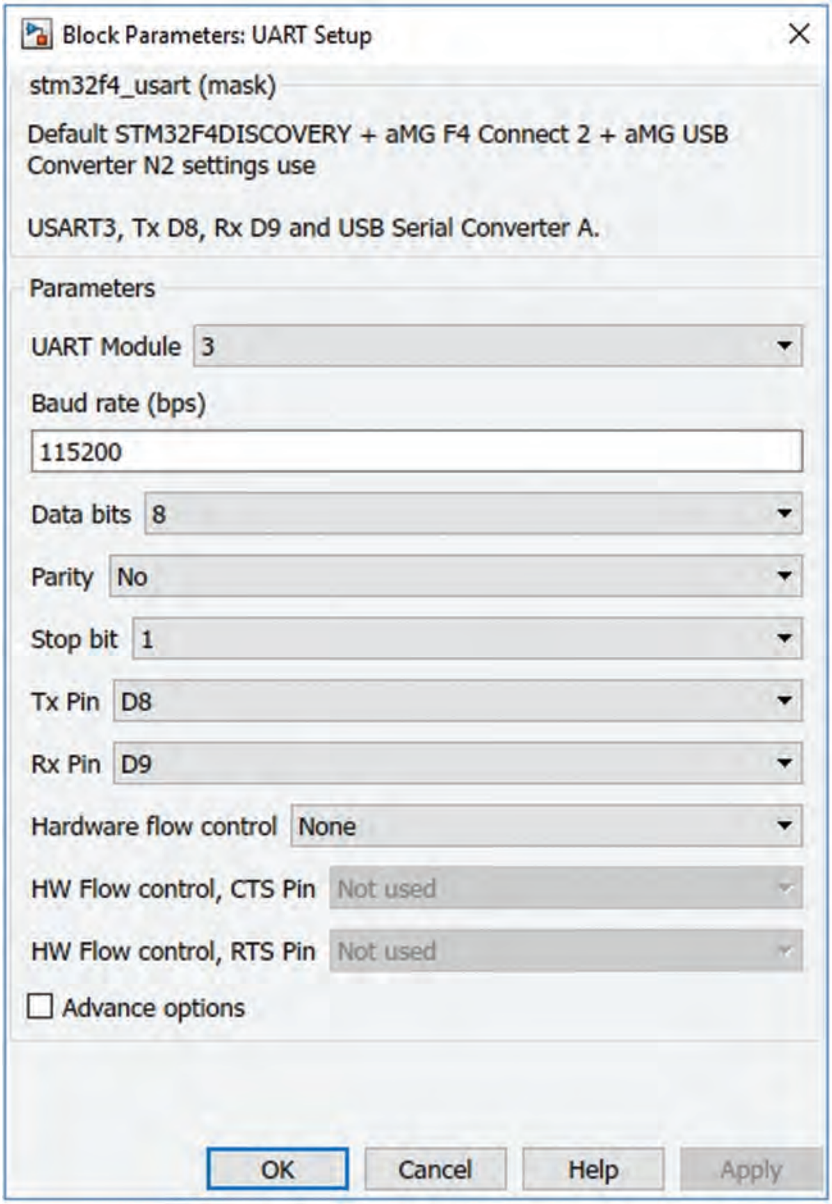

Block Parameters: UART Setup
stm32f4_usart (mask)
Default STM32F4DISCOVERY + aMG F4 Connect 2 + aMG USB Converter N2 settings use
USART3, Tx D8, Rx D9 and USB Serial Converter A.
Parameters
UART Module
3
Baud rate (bps)
115200
Data bits
8
Parity
No
Stop bit
1
Tx Pin
D8
Rx Pin
D9
Hardware flow control
None
HW Flow control, CTS Pin
Not used
HW Flow control, RTS Pin
Not used
Advance options
OK
Cancel
Help
Apply

Figure 5.3

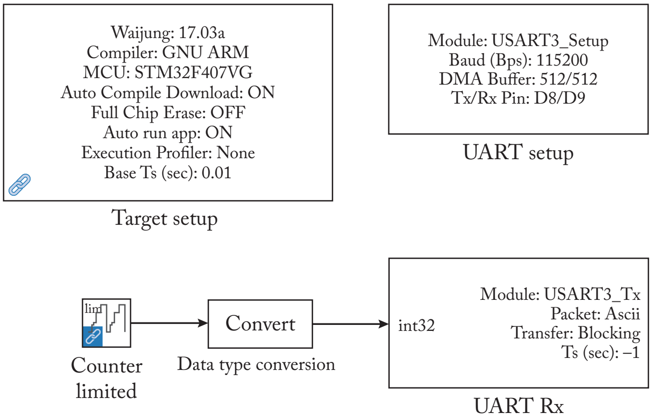


Figure 5.4

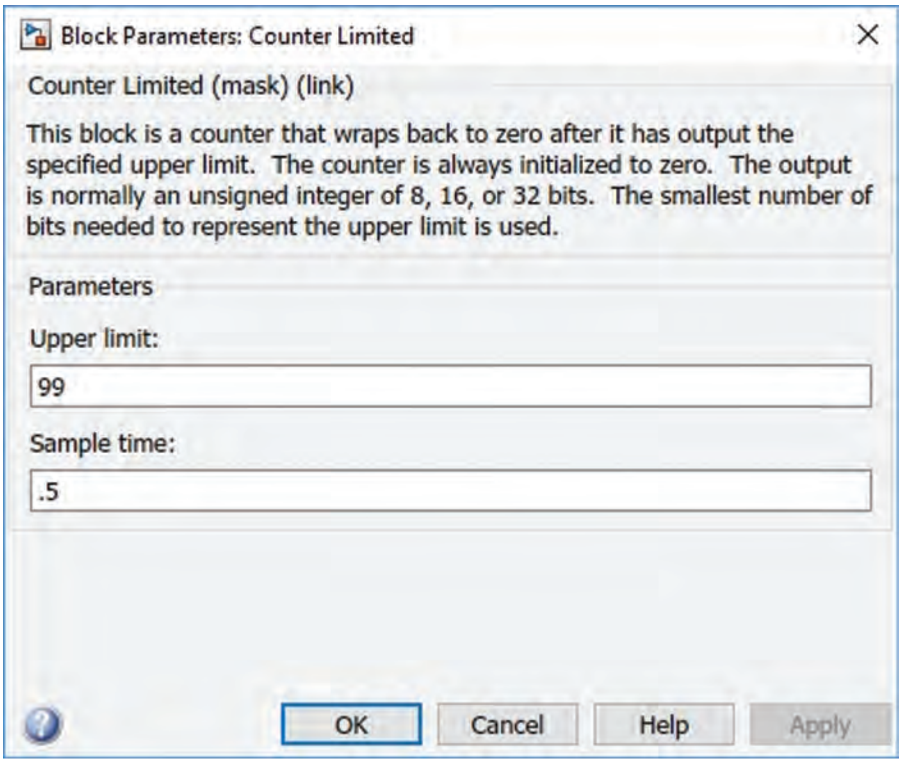


Figure 5.5

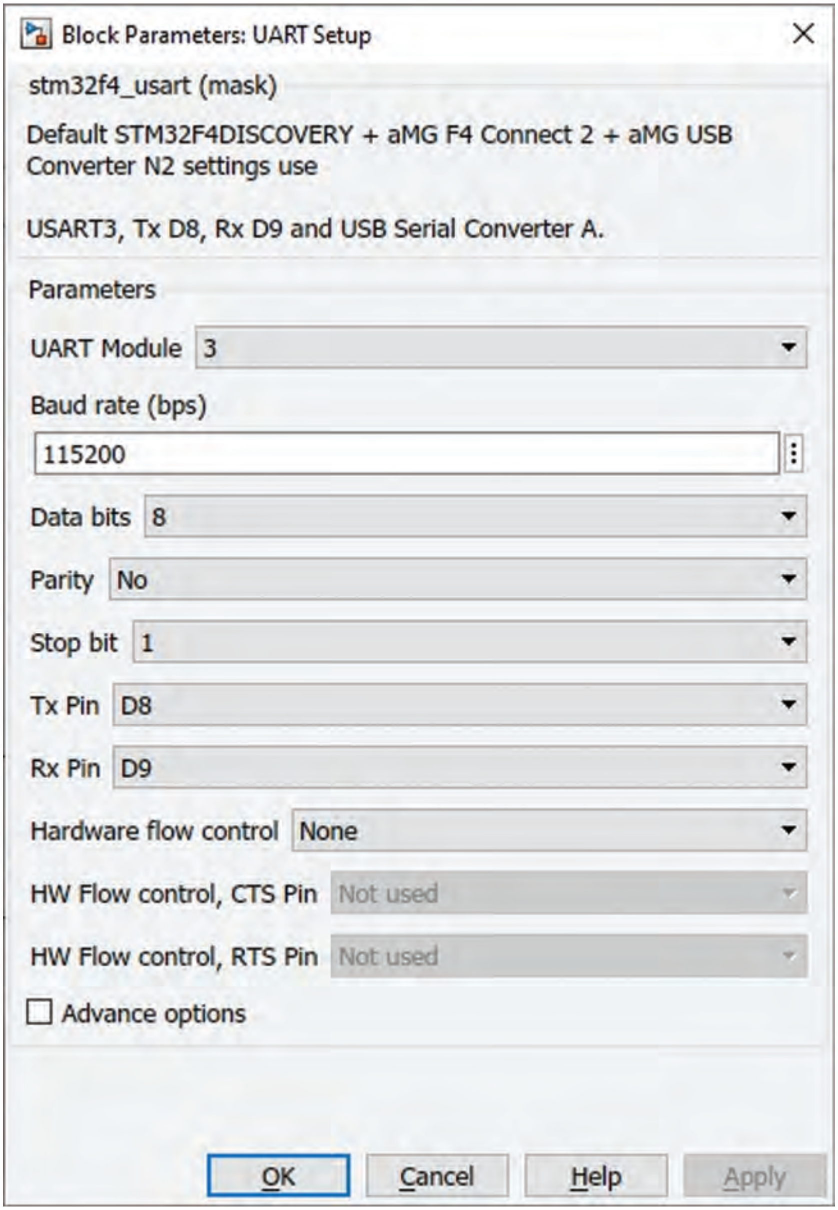

Block Parameters: UART Setup
stm32f4_usart (mask)
Default STM32F4DISCOVERY + aMG F4 Connect 2 + aMG USB Converter N2 settings use
USART3, Tx D8, Rx D9 and USB Serial Converter A.
Parameters
UART Module
3
Baud rate (bps)
115200
Data bits
8
Parity
No
Stop bit
1
Tx Pin
D8
Rx Pin
D9
Hardware flow control
None
HW Flow control, CTS Pin
Not used
HW Flow control, RTS Pin
Not used
Advance options
OK
Cancel
Help
Apply

Figure 5.6

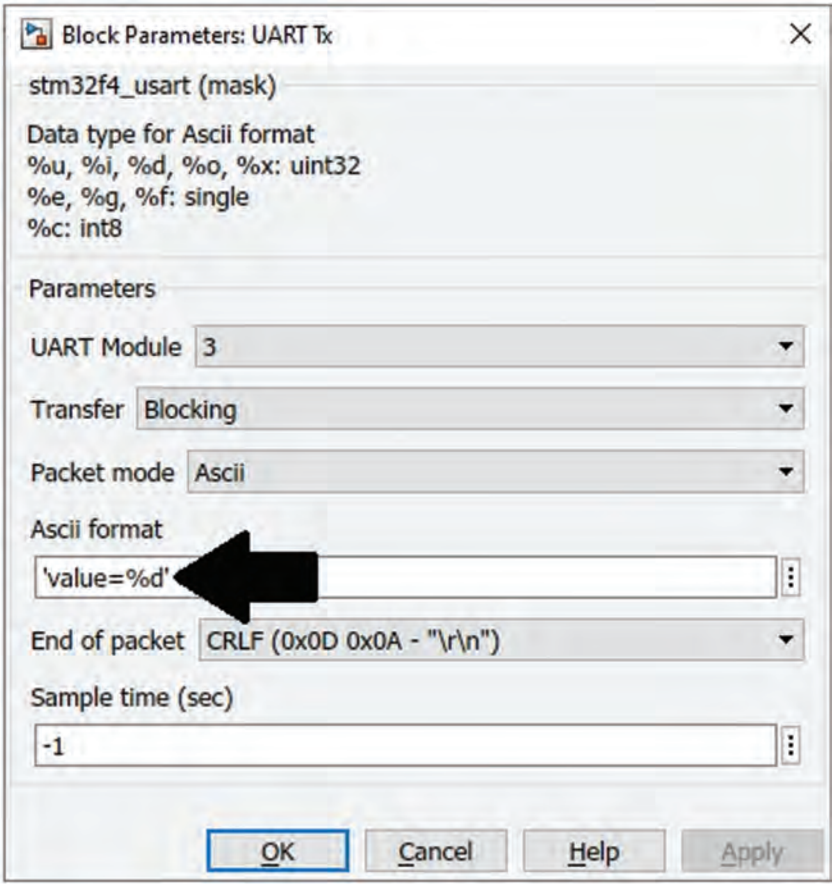
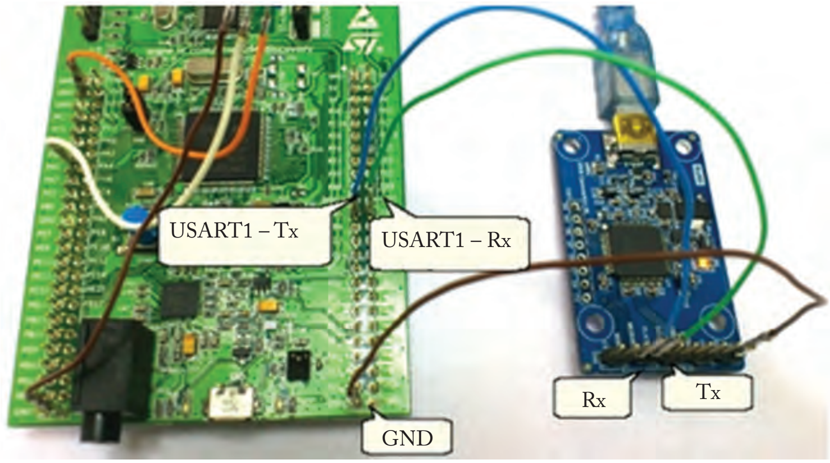


Figure 5.7

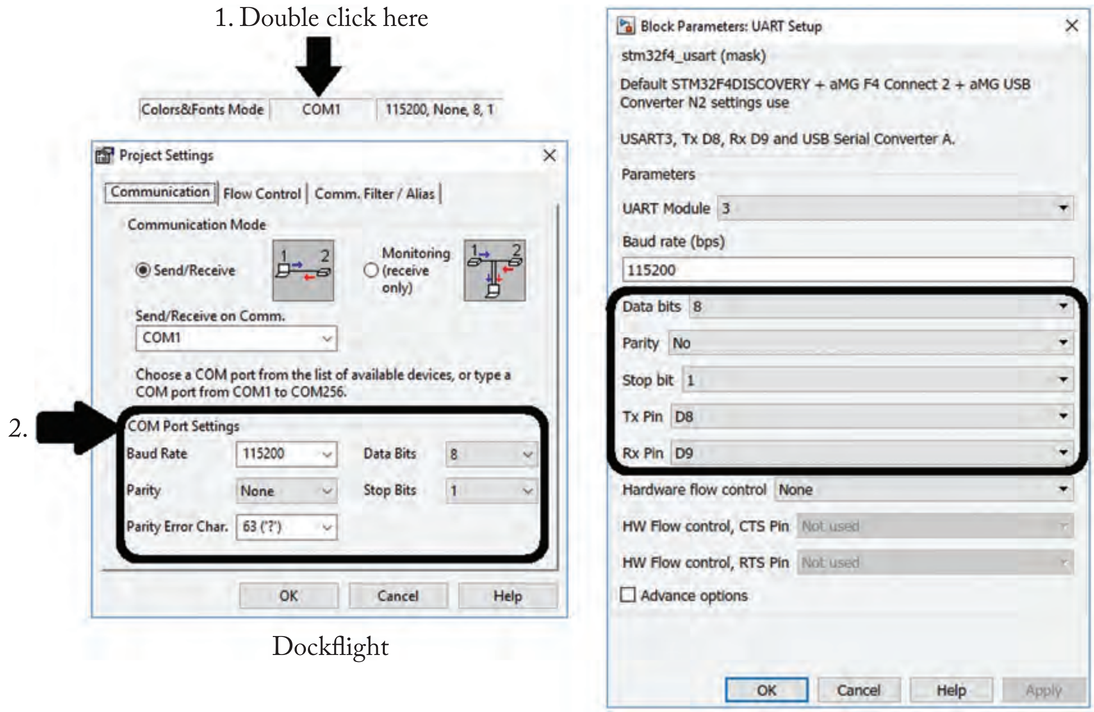
You can use a computer to read the data send by the board. If you want to use a computer to read the data, then you need a USB-Serial converter. Connections are shown in Fig. 5.8. Tx pin of the Discovery board (pin PD8) is connected to the Rx pin of the USB-Serial converter. The ground pin of the Discovery board must be connected to the ground pin of the USB-Serial converter.

Ensure that Docklight settings are the same as the UART Setup block (Fig. 5.9).



STM32f4	USB-Serial
Tx(D8)	Rx
GND	GND

Figure 5.8



Dockflight

UART setup

Figure 5.9

Upload the model to the board. The Docklight starts to receive the data from the micro-controller (Fig. 5.10).

5.3 EXAMPLE 2: SERIAL COMMUNICATION (II)

The Simulink model of this example is shown in Fig. 5.11. In this example we want to receive a serial data from outside. So, the Discovery board is a receiver in this example. This example turns on the on-board LED's if the received value is less than or equal to 3. Settings of UART Rx block is shown in Fig. 5.12.

You can use a computer to send data to the board. If you want to use a computer to send the data, then you need a USB-Serial converter. Connections are shown in Fig. 5.13. Rx pin of the Discovery board (pin PD9) is connected to the Tx pin of the USB-Serial converter. The ground pin of the Discovery board must be connected to the ground pin of USB-Serial converter. Upload the model to the Discovery board. Use the Docklight to send a number to the Discovery board. If the number is less than or equal to 3, then all of the on-board LEDs turn on.

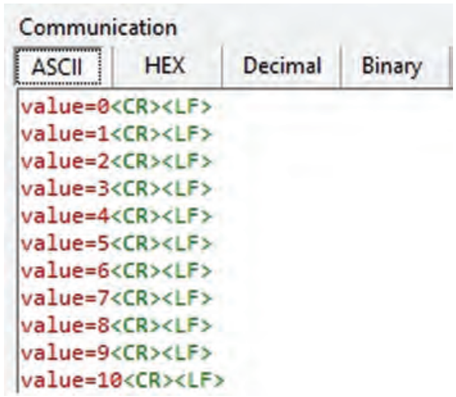


Figure 5.10

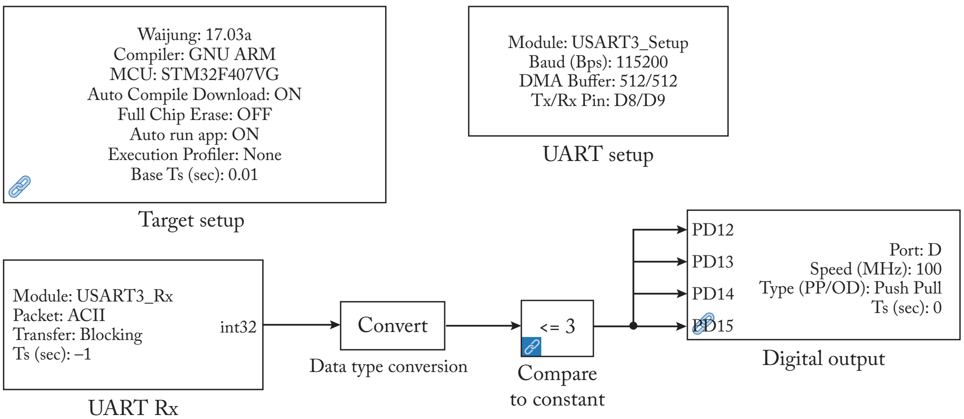


Figure 5.11

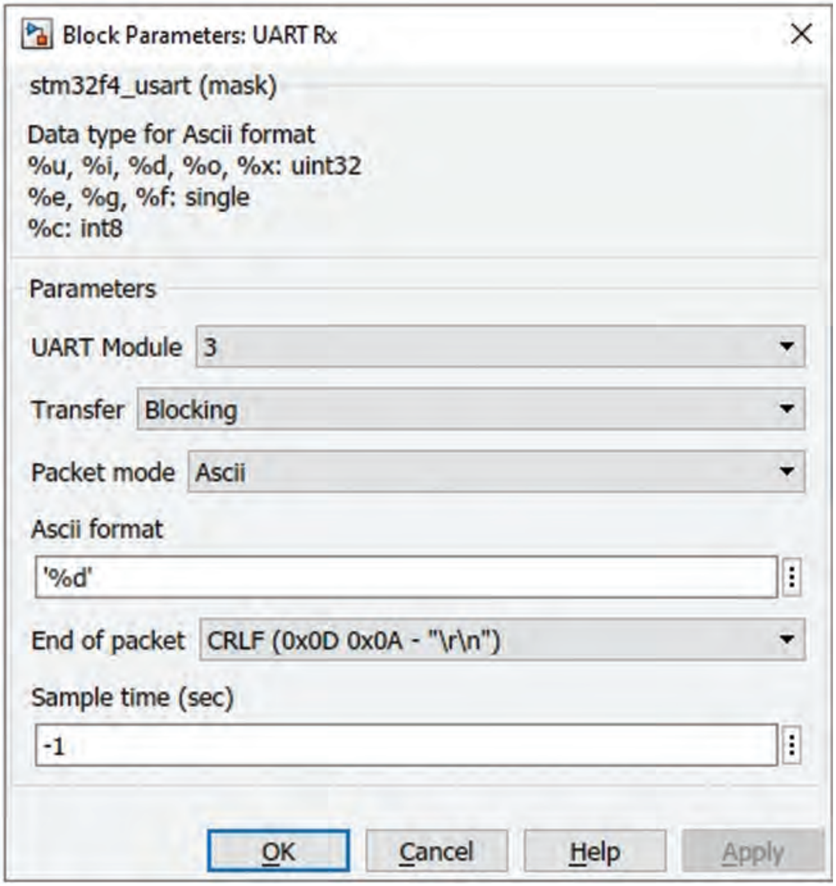
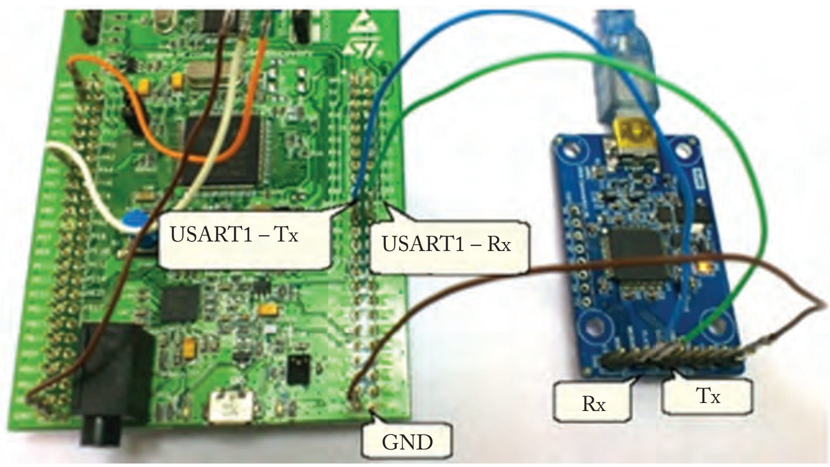

Block Parameters: UART Rx
stm32f4_usart (mask)
Data type for Ascii format
%u, %i, %d, %o, %x: uint32
%e, %g, %f: single
%c: int8
Parameters
UART Module
3
Transfer
Blocking
Packet mode
Ascii
Ascii format
'%d'
End of packet
CRLF (0x0D 0x0A - "\r\n")
Sample time (sec)
-1
OK
Cancel
Help
Apply

Figure 5.12



STM32f4	USB-Serial
Rx(D9)	Tx
GND	GND

Figure 5.13

APPENDIX A

Installation of the Waijung Block Set

The "Waijung 1" block set is used in this book. Waijung 1 blockset only works with MATLAB between R2009a and R2018b. Go to https://www.aimagin.com/en/waijung-1-stm32-target.html in order to install the block set (Fig. A.1).

Figure A.1

Scroll down the page and click the Create an Account in order to make an account (Fig. A.2). After making the account sign into your account and download the `waijung17_03a.7z`. The download instructions can be found in https://www.aimagin.com/en/download as well.

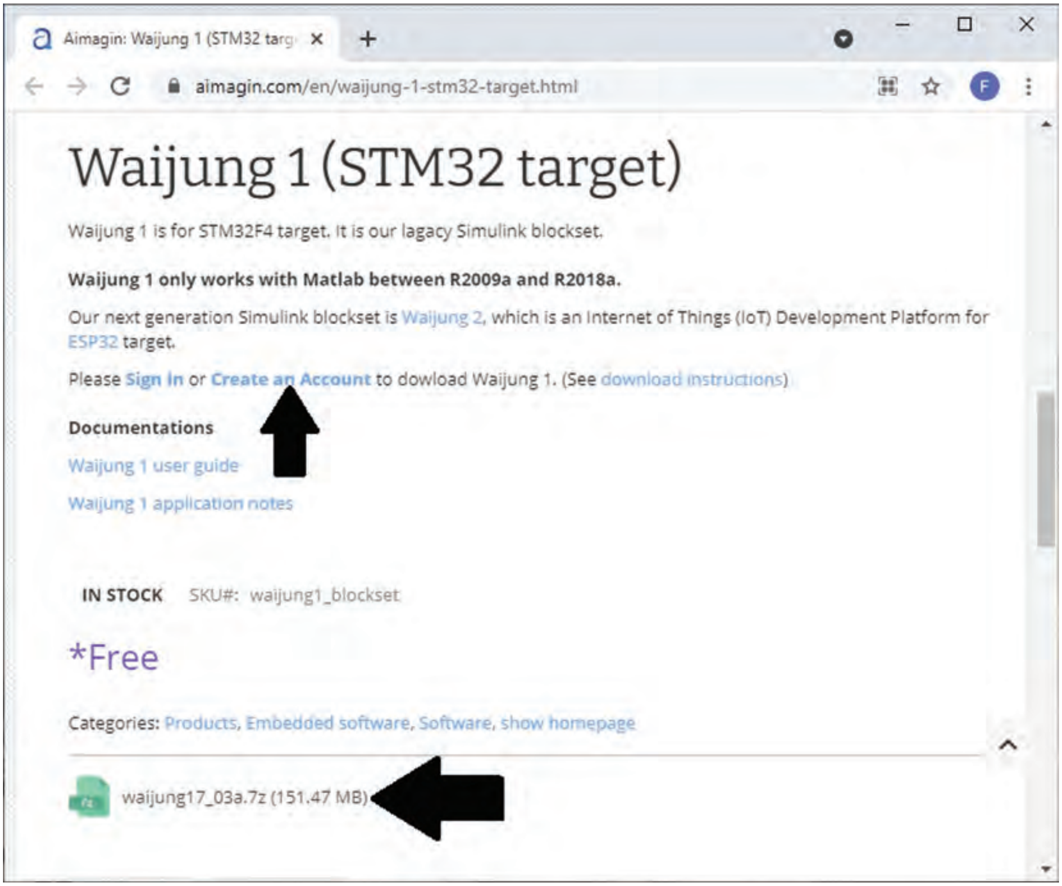


Figure A.2

Right click on the downloaded file and click the Extract Here. This extracts the files in the `Waijung17_03a` folder (Fig. A.3). Open the MATLAB and click the "Browse for folder" icon (Fig. A.4). This opens the Select a new folder dialog box (Fig. A.5). Go to the path that you extracted the downloaded file and open the `Waijung17_03a` folder (Fig. A.6).

Type the `install_waijung` in the MATLAB command window and press the Enter key (Fig. A.7). This installs the block set. After installation, open the Simulink and click the Library Browser (Fig. A.8). Note that Waijung blockset is added to Simulink (Fig. A.9).

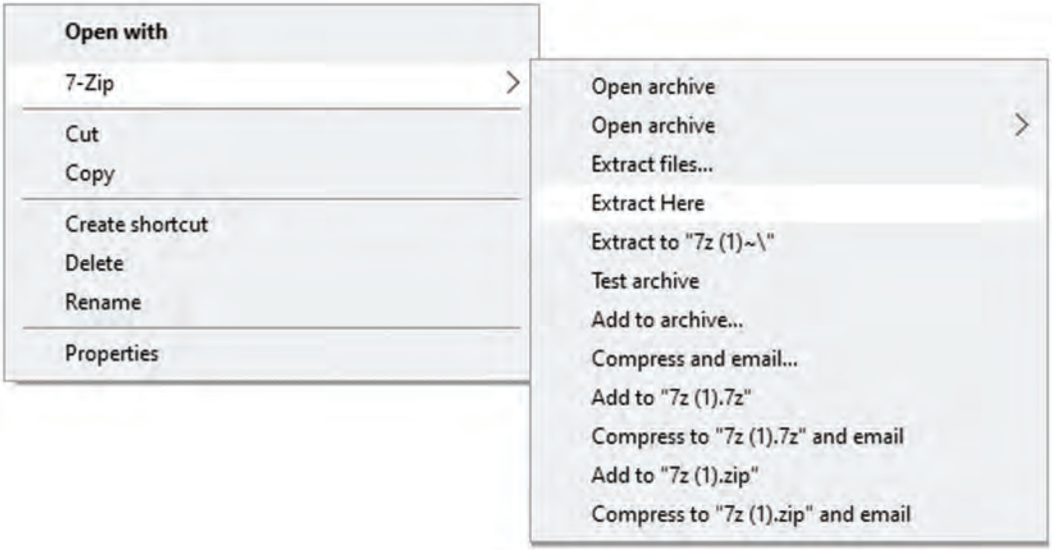


Figure A.3

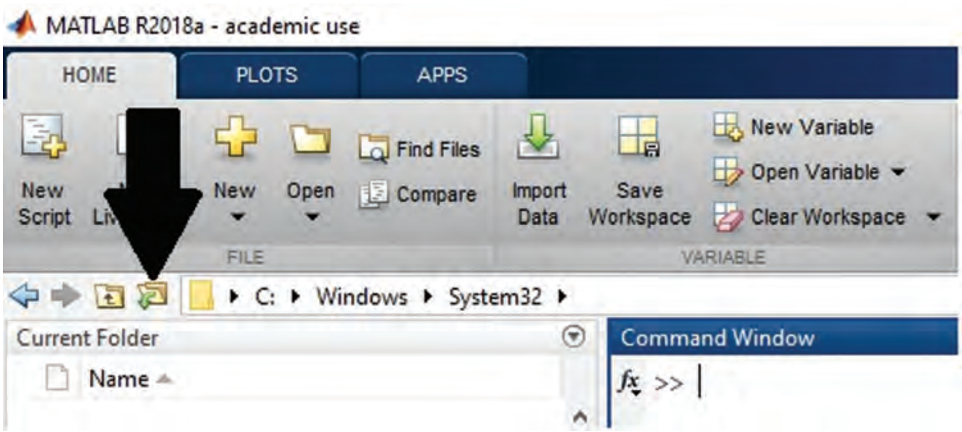


Figure A.4

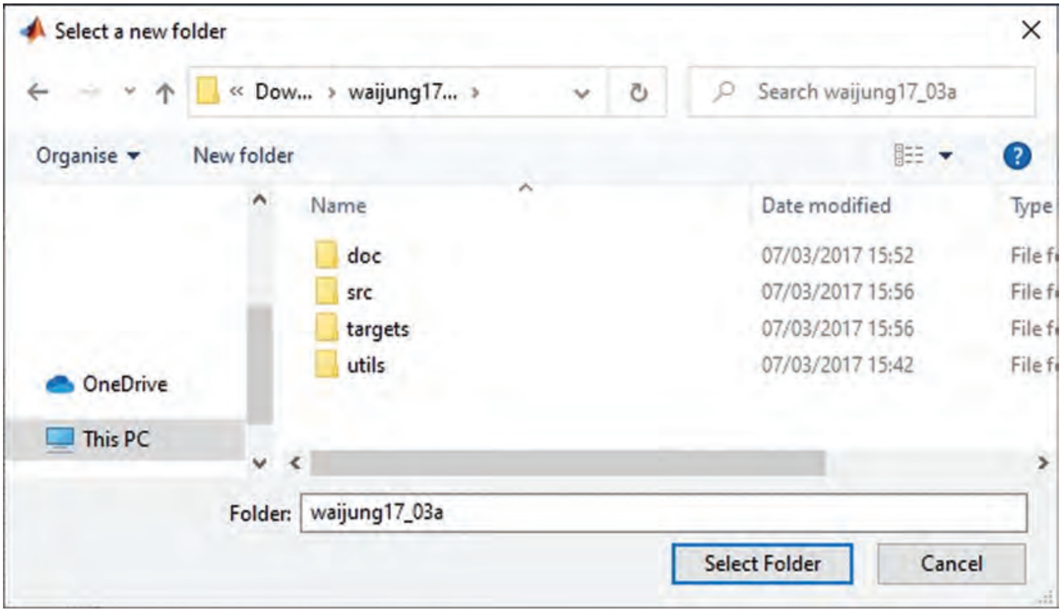


Figure A.5

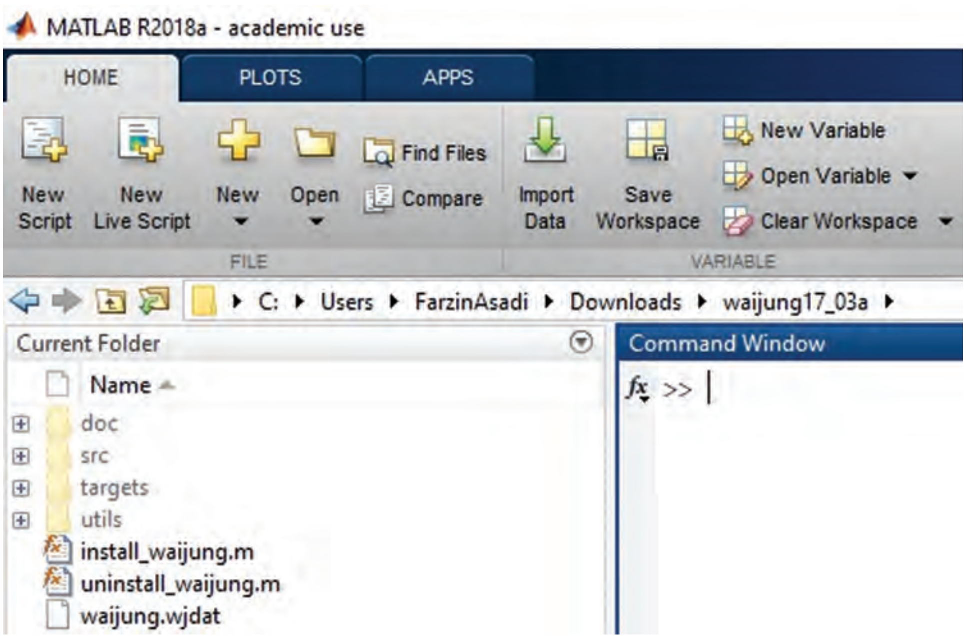


Figure A.6

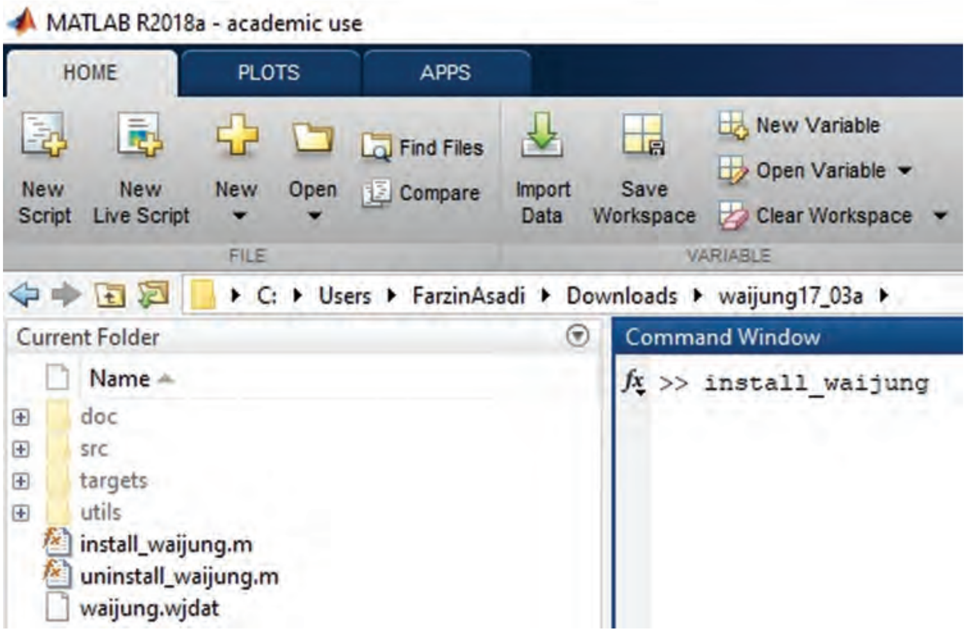


Figure A.7

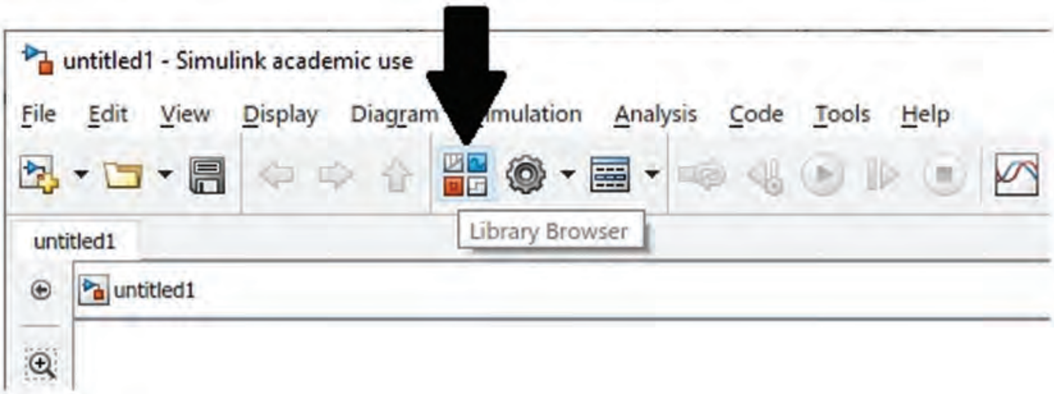


Figure A.8

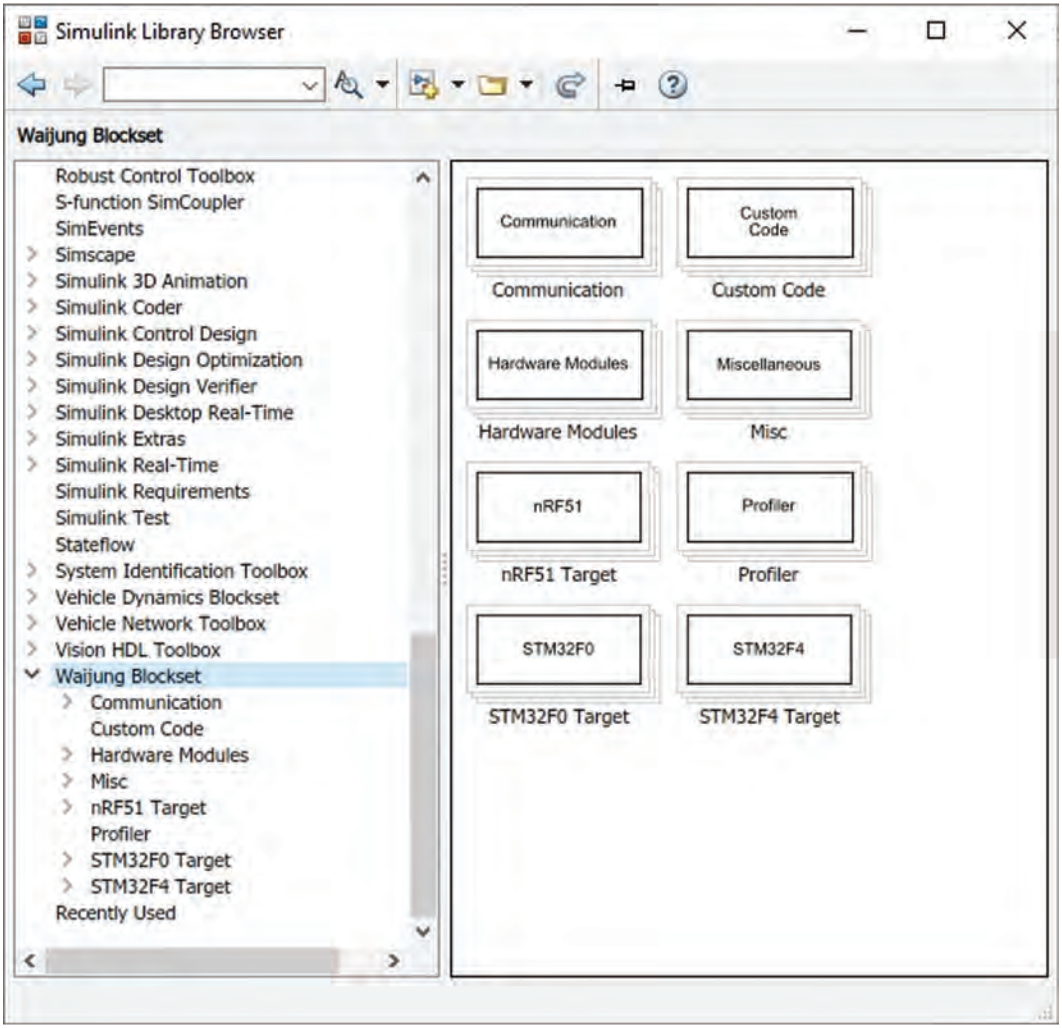

Simulink Library Browser
Waijung Blockset
Robust Control Toolbox
S-function SimCoupler
SimEvents
Simscape
Simulink 3D Animation
Simulink Coder
Simulink Control Design
Simulink Design Optimization
Simulink Design Verifier
Simulink Desktop Real-Time
Simulink Extras
Simulink Real-Time
Simulink Requirements
Simulink Test
Stateflow
System Identification Toolbox
Vehicle Dynamics Blockset
Vehicle Network Toolbox
Vision HDL Toolbox
Waijung Blockset
Communication
Custom Code
Hardware Modules
Misc
nRF51 Target
Profiler
STM32F0 Target
STM32F4 Target
Recently Used
Communication
Communication
Custom Code
Custom Code
Hardware Modules
Hardware Modules
Miscellaneous
Misc
nRF51
nRF51 Target
Profiler
Profiler
STM32F0
STM32F0 Target
STM32F4
STM32F4 Target

Figure A.9

The blocks that are used in this book can be found in the STM32F4 Target section (Fig. A.10).

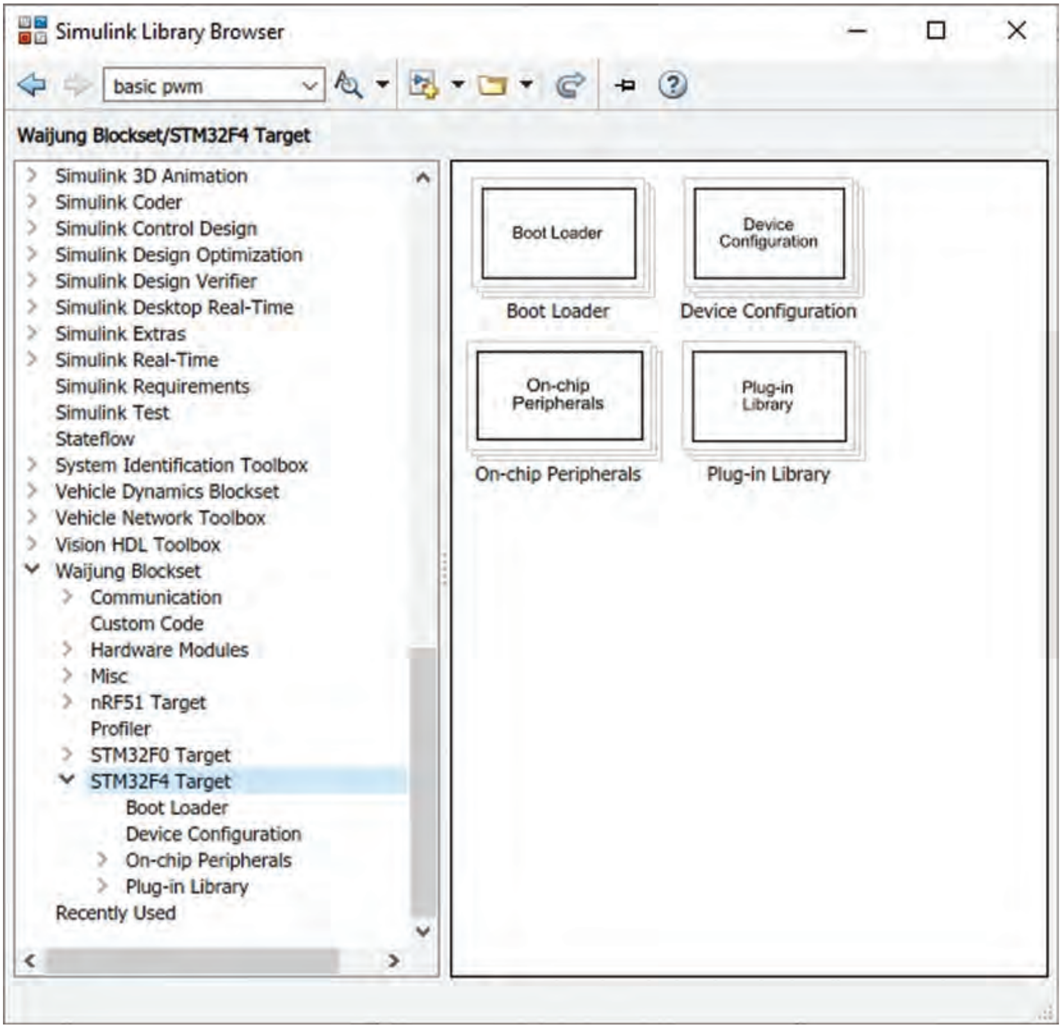


Figure A.10

The Waijung blockset has a demo folder. The demo folder contains many inspiring Simulink models. Click the Open icon (Fig. A.11) to open the Demo folder.

Go to `Waijung17_03a` folder and open the "targets" folder (Fig. A.12). Open the `stm32f4_target` folder (Fig. A.13). Open the `stm32f4` folder (Fig. A.14). Open the "demo" folder (Fig. A.15). Now you have access to the sample Simulink models (Fig. A.16).

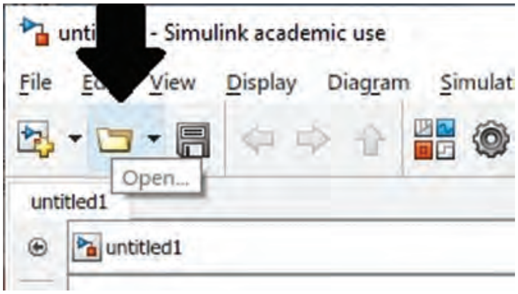


Figure A.11

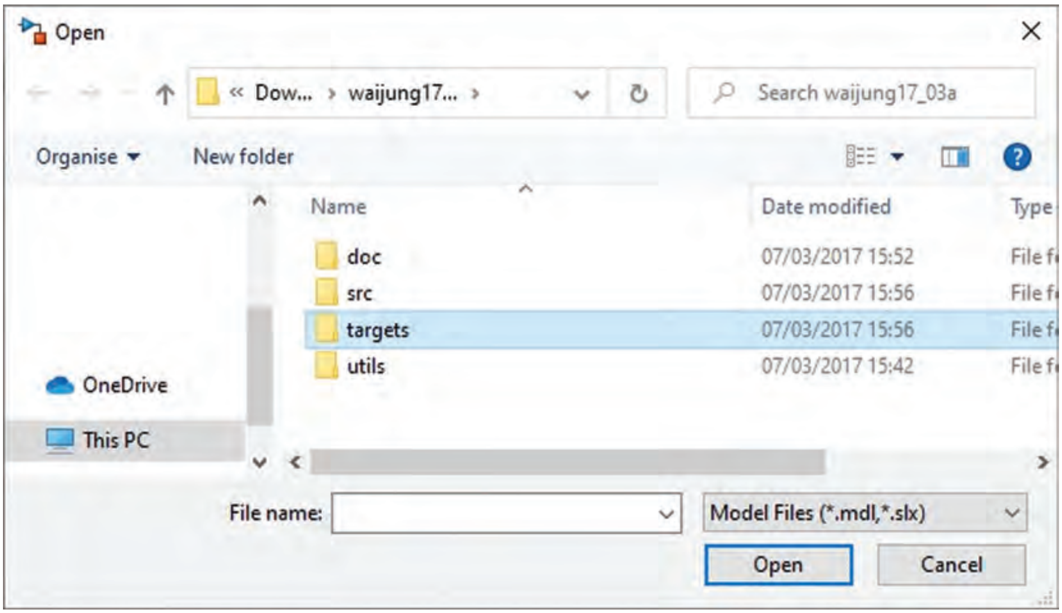


Figure A.12

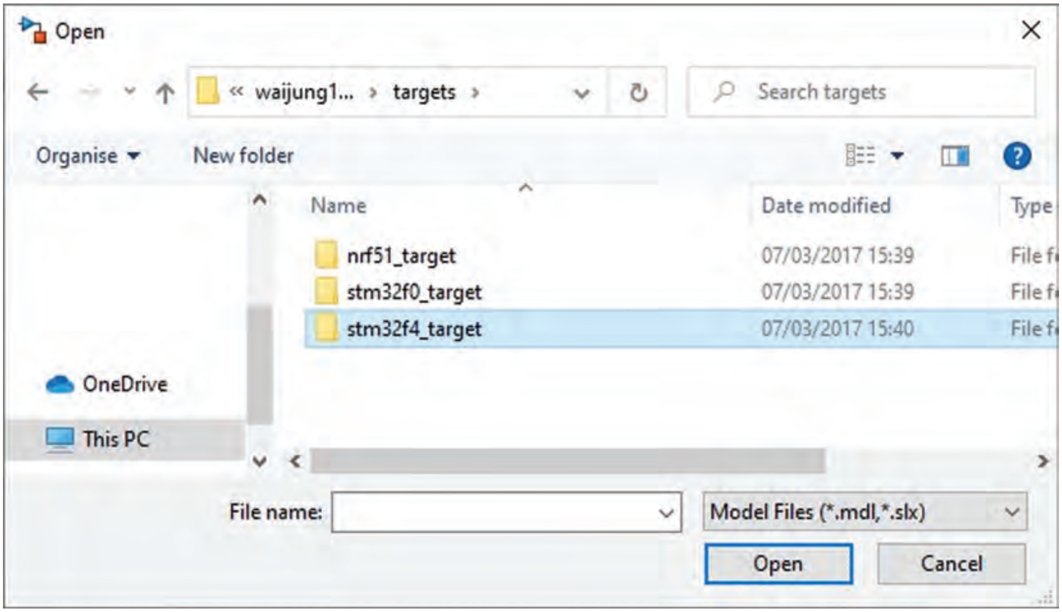


Figure A.13

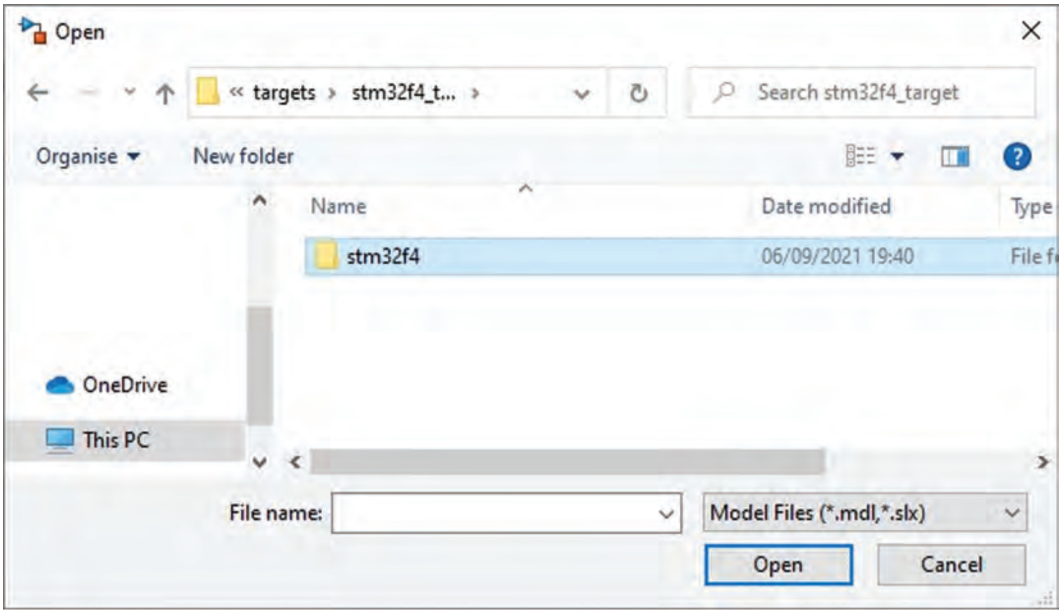


Figure A.14

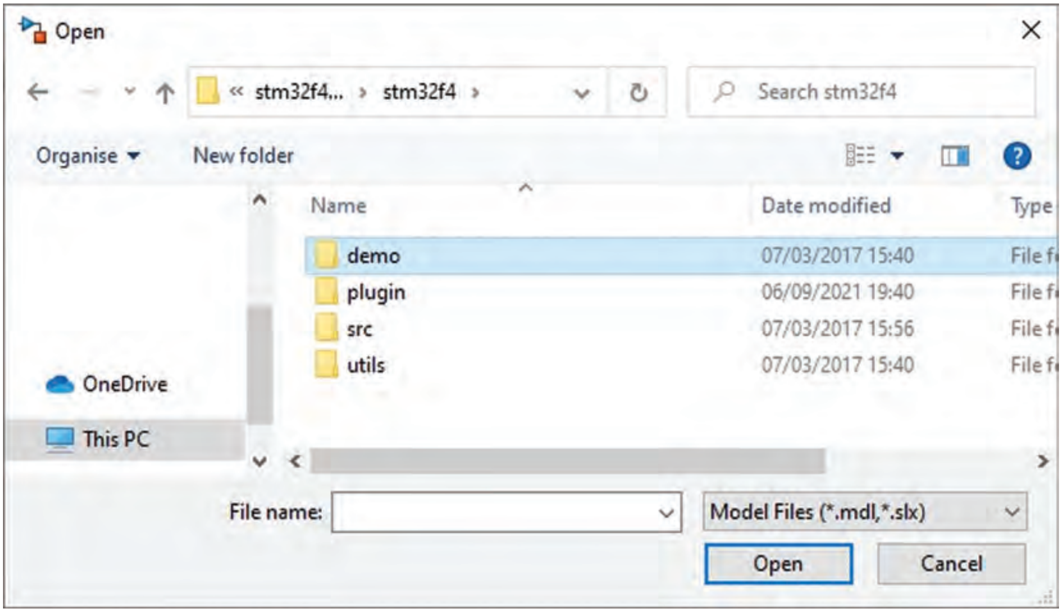


Figure A.15

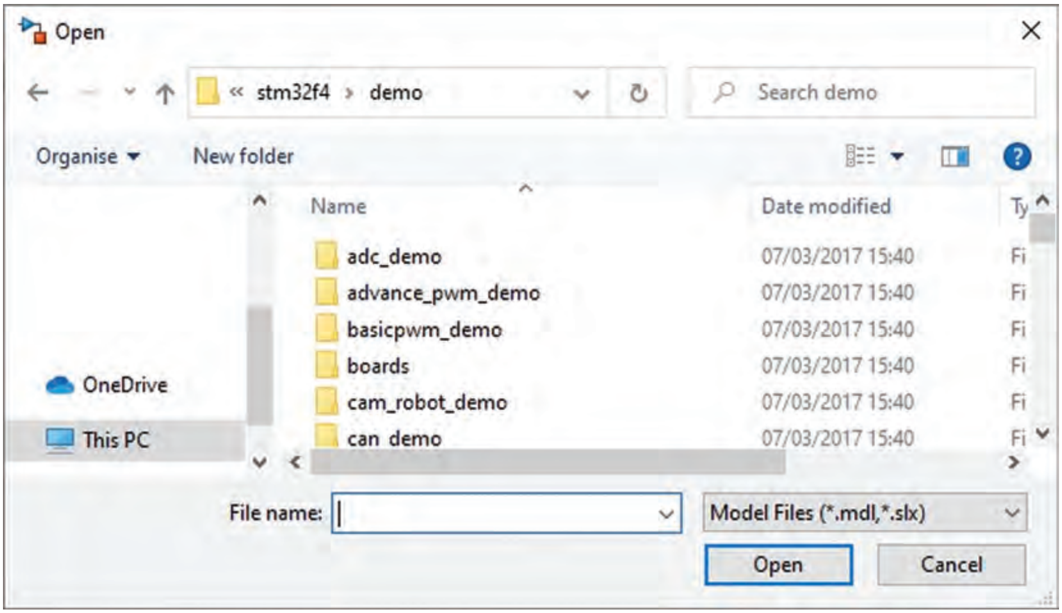


Figure A.16

Another source to increase your knowledge is https://waijung1.aimagin.com/ (Fig. A.17). This page contains many sample projects and other useful material.

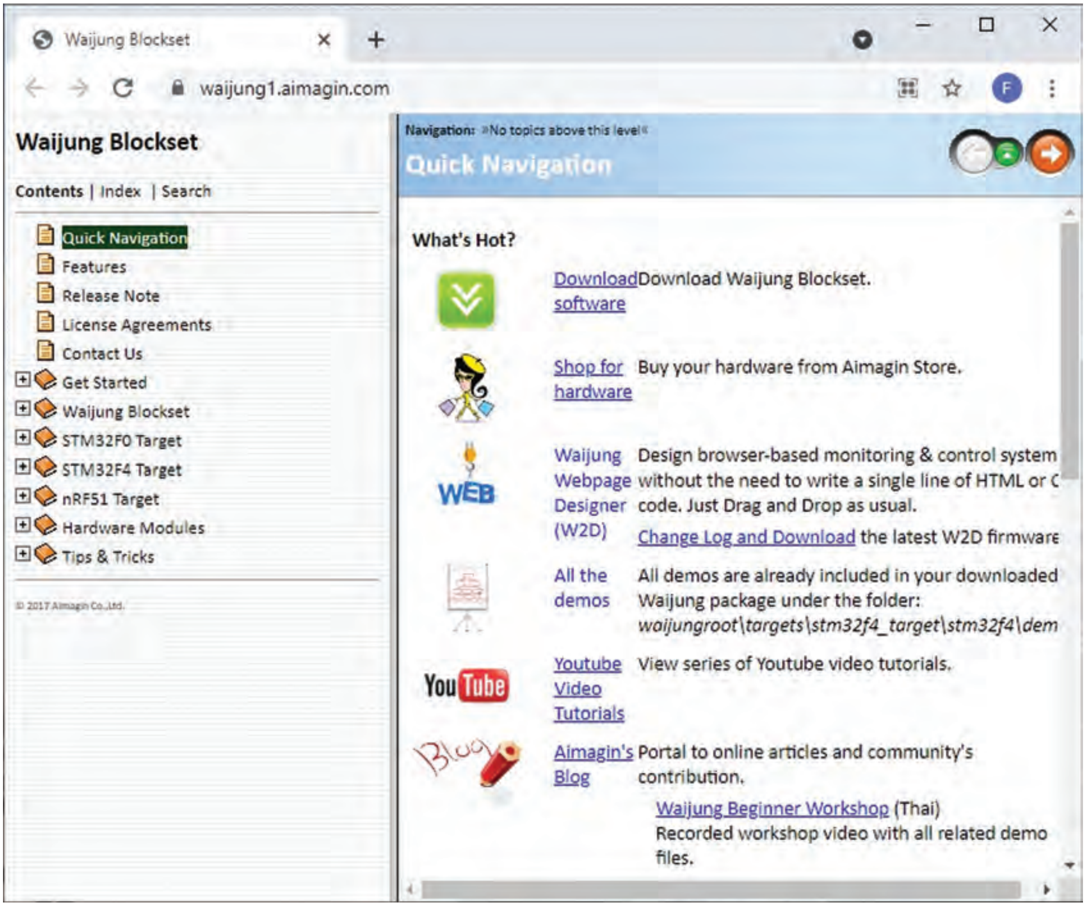


Figure A.17

Authors' Biographies

FARZIN ASADI

Farzin Asadi received his B.Sc. in Electronics Engineering, M.Sc. in Control Engineering, and Ph.D. in Mechatronics Engineering. Currently, he is with the Department of Electrical and Electronics Engineering at Maltepe University, Istanbul, Turkey.

Dr. Asadi has published more than 40 international papers and 15 books. He is on the editorial board of 7 scientific journals as well. His research interests include switching converters, control theory, robust control of power electronics converters, and robotics.

SAWAI PONGSWATD

Sawai Pongswatd received his B.Sc. in Instrumentation Engineering, M.Sc. in Electrical Engineering, and Ph.D. in Electrical Engineering. Currently, he is with the Department of Instrumentation and Control Engineering, King Mongkut's Institute of Technology Ladkrabang (KMITL), Bangkok, Thailand.

Dr. Pongswatd is a chairman of technical committee of Thai Industrial Standards Institute and instructor of Fieldbus Certified Training Program (FCTP). His research interest includes power electronics, energy conversion, and industrial applications.