Motor Control Development Toolbox

User Manual

An Embedded Target for the MPC564xL Family of Processors

By 

Version 1.3.0

Target Based Automatic Code Generation Tools
For MATLAB™/Simulink™/Stateflow™ Models working with Simulink Coder ™ and Embedded Coder®

Motor Control Development Toolbox Users Manual is for use with the Motor Control Development Toolbox, an embedded target and block set library for MATLAB/Simulink/Stateflow Modeling.

MATLAB, Simulink, Stateflow, Handle Graphics, and Real-Time Workshop are registered trademarks, and TargetBox is a trademark of The MathWorks, Inc.

Microsoft and .NET Framework are trademarks of Microsoft Corporation.

Flexera Software, Flexlm, and FlexNet Publisher are registered trademarks or trademarks of Flexera Software, Inc. and/or InstallShield Co. Inc. in the United States of America and/or other countries.

Other product or brand names are trademarks or registered trademarks of their respective holders.

1 Overview 1-16
2 Installation 2-17
2.1 Minimum Platform Requirements 2-17
2.2 Installation Steps 2-17
2.2.1 Run Setup.exe 2-17
2.2.2 Run FreeMASTER Installer 2-17
2.2.3 License Request & Installation 2-17
2.2.4 Setting the Path for Motor Control Development Toolbox 2-17
2.2.5 Setting up the Target Compilers 2-17
2.2.1 Setting up the PC Compiler for MATLAB 2-18
2.2.2 Setting up the MCU for MCD Toolbox 2-18
2.3 Example Models 2-20
2.3.1 adc_pit_block 2-20
2.3.2 can_simple_block 2-20
2.3.3 CTU_ISR 2-20
2.3.4 DigitalIO_block_demo 2-20
2.3.5 dspi_block 2-21
2.3.6 etimer_capture 2-21
2.3.7 eTimer_QD 2-21
2.3.8 flexpwm_ctu_adc_block 2-21
2.3.9 flexpwm_ctu_adc_center 2-21
2.3.10 flexpwm_ctu_etimer 2-21
2.3.11 FlexPWM_ISR 2-21
2.3.12 FlexPWM_simple_and_compl 2-21
2.3.13 FlexPWM_three_phase 2-22
2.3.14 MotorControl_0 through 5 2-22
2.3.15 ReadWrite_Demo 2-22
2.3.16 sim_acc_test 2-22
2.3.17 sim_pil_target_block 2-22
2.3.18 sim_pil_test 2-22
2.3.19 sim_sil_target_block 2-22
2.3.20 sim_sil_test 2-22
2.3.21 SineWave_block 2-22
2.3.22 tests564xl 2-23
3 Processors Supported 3-24
4 Required and Recommended Products 4-25
4.1 MATLAB Required and Recommended Products 4-25
4.2 Compiler Specific Code 4-25
5 MC Toolbox Integration into Simulink 5-26
5.1 Target Configuration Block 5-26
5.1.1 Block Name 5-26
5.1.2 Block Description 5-26
5.1.3 Block Image 5-26
5.1.4 Inputs: 5-26
5.1.5 Outputs: 5-26

5.1.6 Block Dialog and Parameters: 5-26
5.1.7 Block Dependency 5-35
5.1.8 Block Miscellaneous Details: 5-35
5.2 Block Details 5-35
5.2.1 Layout of Block GUIs 5-35
6 Peripheral Blocks 6-37
6.1 Communication Blocks 6-37
6.1.1 FlexCAN 6-37
6.1.1.1 FlexCAN Configuration Block 6-37
6.1.1.1.1 Block Name 6-37
6.1.1.1.2 Block Description 6-37
6.1.1.1.3 Block Image 6-37
6.1.1.1.4 Inputs: 6-37
6.1.1.1.5 Outputs: 6-38
6.1.1.1.6 Block Dialog and Parameters: 6-38
6.1.1.1.7 Block Dependency 6-42
6.1.1.1.8 Block Miscellaneous Details: 6-42
6.1.1.2 FlexCAN Receive Data Block 6-43
6.1.1.2.1 Block Name 6-43
6.1.1.2.2 Block Description 6-43
6.1.1.2.3 Block Image 6-43
6.1.1.2.4 Inputs: 6-43
6.1.1.2.5 Outputs: 6-43
6.1.1.2.6 Block Dialog and Parameters: 6-43
6.1.1.2.7 Block Dependency 6-44
6.1.1.2.8 Block Miscellaneous Details: 6-44
6.1.1.3 FlexCAN Receive Data Trigger Block 6-44
6.1.1.3.1 Block Name 6-44
6.1.1.3.2 Block Description 6-44
6.1.1.3.3 Block Image 6-44
6.1.1.3.4 Inputs: 6-44
6.1.1.3.5 Outputs: 6-45
6.1.1.3.6 Block Dialog and Parameters: 6-45
6.1.1.3.7 Block Dependency 6-45
6.1.1.3.8 Block Miscellaneous Details: 6-45

6.1.1.4 FlexCAN Transmit Block 6-46
6.1.1.4.1 Block Name 6-46
6.1.1.4.2 Block Description 6-46
6.1.1.4.3 Block Image 6-46
6.1.1.4.4 Inputs: 6-46
6.1.1.4.5 Outputs: 6-46
6.1.1.4.6 Block Dialog and Parameters: 6-46
6.1.1.4.7 Block Dependency 6-47
6.1.1.4.8 Block Miscellaneous Details: 6-47
6.1.1.5 CAN ISR Block 6-47
6.1.1.5.1 Block Name 6-47
6.1.1.5.2 Block Description 6-47
6.1.1.5.3 Block Image 6-47
6.1.1.5.4 Inputs: 6-47
6.1.1.5.5 Outputs: 6-48
6.1.1.5.6 Block Dialog and Parameters: 6-48
6.1.1.5.7 Block Dependency 6-48
6.1.1.5.8 Block Miscellaneous Details: 6-48
6.1.1.6 FlexCAN ISR Enable/Disable 6-49
6.1.1.6.1 Block Name 6-49
6.1.1.6.2 Block Description 6-49
6.1.1.6.3 Block Image 6-49
6.1.1.6.4 Inputs: 6-49
6.1.1.6.5 Outputs: 6-49
6.1.1.6.6 Block Dialog and Parameters: 6-49
6.1.1.6.7 Block Dependency 6-50
6.1.1.6.8 Block Miscellaneous Details: 6-50
6.1.2 SPI Interface Blocks 6-50
6.1.2.1 DSPI Configuration Block 6-50
6.1.2.1.1 Block Name 6-50
6.1.2.1.2 Block Description 6-50
6.1.2.1.3 Block Image 6-50
6.1.2.1.4 Inputs: 6-51

6.1.2.1.5 Outputs: 6-51
6.1.2.1.6 Block Dialog and Parameters: 6-51
6.1.2.1.7 Block Dependency 6-59
6.1.2.1.8 Block Miscellaneous Details: 6-59
6.1.2.2 DSPI Receive Block 6-59
6.1.2.2.1 Block Name 6-59
6.1.2.2.2 Block Description 6-59
6.1.2.2.3 Block Image 6-60
6.1.2.2.4 Inputs: 6-60
6.1.2.2.5 Outputs: 6-60
6.1.2.2.6 Block Dialog and Parameters: 6-60
6.1.2.2.7 Block Dependency 6-60
6.1.2.2.8 Block Miscellaneous Details: 6-60
6.1.2.3 DSPI Receive Trigger Block 6-60
6.1.2.3.1 Block Name 6-60
6.1.2.3.2 Block Description 6-61
6.1.2.3.3 Block Image 6-61
6.1.2.3.4 Inputs: 6-61
6.1.2.3.5 Outputs: 6-61
6.1.2.3.6 Block Dialog and Parameters: 6-61
6.1.2.3.7 Block Dependency 6-61
6.1.2.3.8 Block Miscellaneous Details: 6-62
6.1.2.4 DSPI Transmit Block 6-62
6.1.2.4.1 Block Name 6-62
6.1.2.4.2 Block Description 6-62
6.1.2.4.3 Block Image 6-62
6.1.2.4.4 Inputs: 6-62
6.1.2.4.5 Outputs: 6-62
6.1.2.4.6 Block Dialog and Parameters: 6-62
6.1.2.4.7 Block Dependency 6-64
6.1.2.4.8 Block Miscellaneous Details: 6-64
6.1.2.5 DSPI ISR Block 6-64
6.1.2.5.1 Block Name 6-64

6.1.2.5.2 Block Description 6-65
6.1.2.5.3 Block Image 6-65
6.1.2.5.4 Inputs: 6-65
6.1.2.5.5 Outputs: 6-65
6.1.2.5.6 Block Dialog and Parameters: 6-65
6.1.2.5.7 Block Dependency 6-66
6.1.2.5.8 Block Miscellaneous Details: 6-66
6.1.3 General Purpose Blocks 6-66
6.1.3.1 General Purpose Input 6-66
6.1.3.1.1 Block Name 6-66
6.1.3.1.2 Block Description 6-66
6.1.3.1.3 Block Image 6-66
6.1.3.1.4 Inputs: 6-67
6.1.3.1.5 Outputs: 6-67
6.1.3.1.6 Parameters: 6-67
6.1.3.1.7 Block Dependency 6-67
6.1.3.1.8 Block Miscellaneous Details: 6-67
6.1.3.2 General Purpose Output 6-67
6.1.3.2.1 Block Name 6-67
6.1.3.2.2 Block Description 6-67
6.1.3.2.3 Block Image 6-67
6.1.3.2.4 Inputs: 6-68
6.1.3.2.5 Outputs: 6-68
6.1.3.2.6 Parameters: 6-68
6.1.3.2.7 Block Dependency 6-68
6.1.3.2.8 Block Miscellaneous Details: 6-68
6.1.3.3 Periodic Interrupt Timer 6-68
6.1.3.3.1 Block Name 6-68
6.1.3.3.2 Block Description 6-69
6.1.3.3.3 Block Image 6-69
6.1.3.3.4 Inputs: 6-69
6.1.3.3.5 Outputs: 6-69
6.1.3.3.6 Block Dialog and Parameters: 6-69

6.1.3.3.7 Block Dependency 6-70
6.1.3.3.8 Block Miscellaneous Details: 6-70
6.2 Motor Control Blocks 6-70
6.2.1 ADC 6-70
6.2.1.1 ADC Channel Block 6-70
6.2.1.1.1 Block Name 6-70
6.2.1.1.2 Block Description 6-70
6.2.1.1.3 Block Image 6-71
6.2.1.1.4 Inputs: 6-71
6.2.1.1.5 Outputs: 6-71
6.2.1.1.6 Block Dialog and Parameters: 6-71
6.2.1.1.7 Block Dependency 6-72
6.2.1.1.8 Block Miscellaneous Details: 6-72
6.2.1.2 ADC Configuration Block 6-72
6.2.1.2.1 Block Name 6-72
6.2.1.2.2 Block Description 6-72
6.2.1.2.3 Block Image 6-72
6.2.1.2.4 Inputs: 6-73
6.2.1.2.5 Outputs: 6-73
6.2.1.2.6 Block Dialog and Parameters: 6-73
6.2.1.2.7 Block Dependency 6-75
6.2.1.2.8 Block Miscellaneous Details: 6-75
6.2.1.3 ADC Interrupt 6-75
6.2.1.3.1 Block Name 6-75
6.2.1.3.2 Block Description 6-75
6.2.1.3.3 Block Image 6-75
6.2.1.3.4 Inputs: 6-75
6.2.1.3.5 Outputs: 6-76
6.2.1.3.6 Block Dialog and Parameters: 6-76
6.2.1.3.7 Block Dependency 6-77
6.2.1.3.8 Block Miscellaneous Details: 6-77
6.2.1.4 ADC Trigger 6-78
6.2.1.4.1 Block Name 6-78

6.2.1.4.2 Block Description 6-78
6.2.1.4.3 Block Image 6-78
6.2.1.4.4 Inputs: 6-78
6.2.1.4.5 Outputs: 6-78
6.2.1.4.6 Block Dialog and Parameters: 6-78
6.2.1.4.7 Block Dependency 6-79
6.2.1.4.8 Block Miscellaneous Details: 6-79
6.2.1.5 ADC Watchdog Threshold 6-79
6.2.1.5.1 Block Name 6-79
6.2.1.5.2 Block Description 6-79
6.2.1.5.3 Block Image 6-79
6.2.1.5.4 Inputs: 6-80
6.2.1.5.5 Outputs: 6-80
6.2.1.5.6 Block Dialog and Parameters: 6-80
6.2.1.5.7 Block Dependency 6-81
6.2.1.5.8 Block Miscellaneous Details: 6-81
6.2.2 CTU 6-81
6.2.2.1 CTU Configuration 6-82
6.2.2.1.1 Block Name 6-82
6.2.2.1.2 Block Description 6-82
6.2.2.1.3 Block Image 6-82
6.2.2.1.4 Inputs: 6-82
6.2.2.1.5 Outputs: 6-82
6.2.2.1.6 Block Dialog and Parameters: 6-82
6.2.2.1.7 Block Dependency 6-88
6.2.2.1.8 Block Miscellaneous Details: 6-88
6.2.2.2 CTU ISR 6-89
6.2.2.2.1 Block Name 6-89
6.2.2.2.2 Block Description 6-89
6.2.2.2.3 Block Image 6-89
6.2.2.2.4 Inputs: 6-90
6.2.2.2.5 Outputs: 6-90
6.2.2.2.6 Block Dialog and Parameters: 6-90

6.2.2.2.7 Block Dependency 6-90
6.2.2.2.8 Block Miscellaneous Details: 6-91
6.2.2.3 CTU ISR Enable 6-91
6.2.2.3.1 Block Name 6-91
6.2.2.3.2 Block Description 6-91
6.2.2.3.3 Block Image 6-91
6.2.2.3.4 Inputs: 6-91
6.2.2.3.5 Outputs: 6-91
6.2.2.3.6 Block Dialog and Parameters: 6-91
6.2.2.3.7 Block Dependency 6-92
6.2.2.3.8 Block Miscellaneous Details: 6-92
6.2.3 PWM 6-92
6.2.3.1 Complementary PWM Output 6-92
6.2.3.1.1 Block Name 6-92
6.2.3.1.2 Block Description 6-93
6.2.3.1.3 Block Image 6-93
6.2.3.1.4 Inputs: 6-93
6.2.3.1.5 Outputs: 6-93
6.2.3.1.6 Block Dialog and Parameters: 6-93
6.2.3.1.7 Block Dependency 6-97
6.2.3.1.8 Block Miscellaneous Details: 6-97
6.2.3.2 FlexPWM ISR 6-97
6.2.3.2.1 Block Name 6-97
6.2.3.2.2 Block Description 6-97
6.2.3.2.3 Block Image 6-97
6.2.3.2.4 Inputs: 6-97
6.2.3.2.5 Outputs: 6-97
6.2.4 Block Dialog and Parameters: 6-98
6.2.4.1.1 Block Dependency 6-99
6.2.4.1.2 Block Miscellaneous Details: 6-99
6.2.4.2 PWM ISR Enable 6-99
6.2.4.2.1 Block Name 6-99
6.2.4.2.2 Block Description 6-99

6.2.4.2.3 Block Image 6-99
6.2.4.2.4 Inputs: 6-100
6.2.4.2.5 Outputs: 6-100
6.2.4.2.6 Block Dialog and Parameters: 6-100
6.2.4.2.7 Block Dependency 6-101
6.2.4.2.8 Block Miscellaneous Details: 6-101
6.2.4.3 Simple PWM Output 6-101
6.2.4.3.1 Block Name 6-101
6.2.4.3.2 Block Description 6-101
6.2.4.3.3 Block Image 6-102
6.2.4.3.4 Inputs: 6-102
6.2.4.3.5 Outputs: 6-102
6.2.4.3.6 Block Dialog and Parameters: 6-102
6.2.4.3.7 Block Dependency 6-107
6.2.4.3.8 Block Miscellaneous Details: 6-107
6.2.4.4 Sine Wave Generator 6-107
6.2.4.4.1 Block Name 6-107
6.2.4.4.2 Block Description 6-107
6.2.4.4.3 Block Image 6-107
6.2.4.4.4 Inputs: 6-108
6.2.4.4.5 Outputs: 6-108
6.2.4.4.6 Block Dialog and Parameters: 6-108
6.2.4.4.7 Block Dependency 6-109
6.2.4.4.8 Block Miscellaneous Details: 6-109
6.2.4.5 Three-phase PWM Output 6-109
6.2.4.5.1 Block Name 6-109
6.2.4.5.2 Block Description 6-109
6.2.4.5.3 Block Image 6-109
6.2.4.5.4 Inputs: 6-109
6.2.4.5.5 Outputs: 6-110
6.2.4.5.6 Block Dialog and Parameters: 6-110
6.2.4.5.7 Block Dependency 6-114
6.2.4.5.8 Block Miscellaneous Details: 6-114

6.2.5 Timer Blocks 6-115
6.2.5.1 eTimer Configuration 6-115
6.2.5.1.1 Block Name 6-115
6.2.5.1.2 Block Description 6-115
6.2.5.1.3 Block Image 6-115
6.2.5.1.4 Inputs: 6-115
6.2.5.1.5 Outputs: 6-115
6.2.5.1.6 Block Dialog and Parameters: 6-115
6.2.5.1.7 Block Dependency 6-125
6.2.5.1.8 Block Miscellaneous Details: 6-125
6.2.5.2 eTimer ISR 6-125
6.2.5.2.1 Block Name 6-125
6.2.5.2.2 Block Description 6-125
6.2.5.2.3 Block Image 6-125
6.2.5.2.4 Inputs: 6-126
6.2.5.2.5 Outputs: 6-126
6.2.5.2.6 Block Dialog and Parameters: 6-126
6.2.5.2.7 Block Dependency 6-126
6.2.5.2.8 Block Miscellaneous Details: 6-126
6.2.5.3 eTimer ISR Enable 6-127
6.2.5.3.1 Block Name 6-127
6.2.5.3.2 Block Description 6-127
6.2.5.3.3 Block Image 6-127
6.2.5.3.4 Inputs: 6-127
6.2.5.3.5 Outputs: 6-127
6.2.6 Block Dialog and Parameters: 6-127
6.2.6.1.1 Block Dependency 6-128
6.2.6.1.2 Block Miscellaneous Details: 6-128
6.2.6.2 eTimer Capture 6-128
6.2.6.2.1 Block Name 6-129
6.2.6.2.2 Block Description 6-129
6.2.6.2.3 Block Image 6-129
6.2.6.2.4 Inputs: 6-129

6.2.6.2.5 Outputs: 6-129
6.2.6.2.6 Block Dialog and Parameters: 6-129
6.2.6.2.7 Block Dependency 6-130
6.2.6.2.8 Block Miscellaneous Details: 6-130
6.2.6.3 eTimer Pre-load 6-130
6.2.6.3.1 Block Name 6-130
6.2.6.3.2 Block Description 6-130
6.2.6.3.3 Block Image 6-130
6.2.6.3.4 Inputs: 6-130
6.2.6.3.5 Outputs: 6-130
6.2.6.3.6 Block Dialog and Parameters: 6-131
6.2.6.3.7 Block Dependency 6-131
6.2.6.3.8 Block Miscellaneous Details: 6-131
6.2.6.4 eTimer CNTR Register Read 6-131
6.2.6.4.1 Block Name 6-131
6.2.6.4.2 Block Description 6-131
6.2.6.4.3 Block Image 6-132
6.2.6.4.4 Inputs: 6-132
6.2.6.4.5 Outputs: 6-132
6.2.6.4.6 Block Dialog and Parameters: 6-132
6.2.6.4.7 Block Dependency 6-132
6.2.6.4.8 Block Miscellaneous Details: 6-132
6.2.6.5 eTimer HOLD Register Read 6-133
6.2.6.5.1 Block Name 6-133
6.2.6.5.2 Block Description 6-133
6.2.6.5.3 Block Image 6-133
6.2.6.5.4 Inputs: 6-133
6.2.6.5.5 Outputs: 6-133
6.2.6.5.6 Block Dialog and Parameters: 6-133
6.2.6.5.7 Block Dependency 6-134
6.2.6.5.8 Block Miscellaneous Details: 6-134
6.3 Utility Blocks 6-134
6.3.1 CCP DAQ 6-134

6.3.1.1 Block Name 6-134
6.3.1.2 Block Description 6-134
6.3.1.3 Block Image 6-134
6.3.1.4 Inputs: 6-134
6.3.1.5 Outputs: 6-134
6.3.1.6 Block Dialog and Parameters: 6-134
6.3.1.7 Block Dependency 6-135
6.3.1.8 Block Miscellaneous Details: 6-135
6.3.2 FreeMaster Data Recorder 6-135
6.3.2.1 Block Name 6-135
6.3.2.2 Block Description 6-135
6.3.2.3 Block Image 6-135
6.3.2.4 Inputs: 6-135
6.3.2.5 Outputs: 6-135
6.3.2.6 Block Dialog and Parameters: 6-136
6.3.2.7 Block Dependency 6-136
6.3.2.8 Block Miscellaneous Details: 6-136
6.3.3 Memory Read 6-136
6.3.3.1 Block Name 6-136
6.3.3.2 Block Description 6-136
6.3.3.3 Block Image 6-136
6.3.3.4 Inputs: 6-136
6.3.3.5 Outputs: 6-136
6.3.3.6 Block Dialog and Parameters: 6-136
6.3.3.7 Block Dependency 6-137
6.3.3.8 Block Miscellaneous Details: 6-137
6.3.4 Memory Write 6-137
6.3.4.1 Block Name 6-137
6.3.4.2 Block Description 6-137
6.3.4.3 Block Image 6-137
6.3.4.4 Inputs: 6-138
6.3.4.5 Outputs: 6-138
6.3.4.6 Block Dialog and Parameters: 6-138
6.3.4.7 Block Dependency 6-138
6.3.4.8 Block Miscellaneous Details: 6-138
6.3.5 Profiling 6-139
6.3.5.1 Profiler Instrumentation Block 6-139
6.3.5.1.1 Block Name 6-139
6.3.5.1.2 Block Description 6-139
6.3.5.1.3 Block Image 6-139
6.3.5.1.4 Inputs: 6-139
6.3.5.1.5 Outputs: 6-139
6.3.5.1.6 Block Dialog and Parameters: 6-140
6.3.5.1.7 Block Dependency 6-140

6.3.5.1.8 Block Miscellaneous Details: 6-140

7 Embedded Targets 7-141
7.1 Compilers 7-141
7.1.1 Diab Compiler Switches 7-141
7.1.2 Green Hills Compiler Switches 7-142
7.1.3 Code Warrior Compiler Switches 7-143
7.2 Memory Targets (LSM/DPM) 7-143
7.2.1 Flash Memory Target 7-144
7.2.2 RAM Memory Target 7-144
7.2.3 BAM RAM Memory Target (Bootloader) 7-145
7.3 User-Specific Files Needed for Build 7-145
8 FreeMASTER Interface 8-147
9 Bootloader 9-148
10 PIL/SIL Operational Mode 10-149
10.1 PIL & SIL Automatic Configuration 10-149
10.2 PIL & SIL Manual Configuration 10-151

1 Overview

The Motor Control Development Toolbox is a development toolbox to allow controls engineers to move quickly from the modeling and simulation environment to the target processor environment with ease. The toolbox contains peripheral driver interface blocks, a code generation target for MPC564xL family of processors, optimized target code blocks for the MPC564xL processor, and support for model reference Software-In-the-Loop (SIL) / Processor-In-the-Loop (PIL) code generation and co-simulation testing. It works well in conjunction with the Freescale Application Motor Control Library Blocks which are used for Field Oriented Control applications.

Using the Motor Control Development Toolbox the user can move from the modeling and simulation environment quickly to the target processor environment. The user can then profile execution in the target environment and quickly build and prototype new functions and features understanding the performance and memory cost of the implementation through automatic code generation technology. The toolbox allows the user to perform experiments on determination of best silicon configuration on target for the prototyped application.

2 Installation

Installing the Motor Control Development Toolbox is your first step to getting up and running on the target processor. Please follow the installation steps below, and then explore the examples.

2.1 Minimum Platform Requirements

The minimum recommended PC platform is Windows XP, Vista, 7 or 8 with a minimum CPU Speed of 2.0 GHz and Minimum of 4GB of RAM.

2.2 Installation Steps

1. Run setup.exe
2. Run FreeMASTER Installer from Motor Control Development Toolbox Menu
3. Request and Install license file
4. Setup the MATLAB path for Motor Control Development Toolbox

2.2.1 Run Setup.exe

Install the Motor Control Development Toolbox by running the Setup.exe; this will activate the installer. Click through the dialogs entering in the appropriate location for the installation. It is best if this is NOT installed on a network drive.

2.2.2 Run FreeMASTER Installer

1. START->Programs->Motor Control Development Toolbox for MPC564xL->Installers->FreeMASTER Installer

2.2.3 License Request & Installation

Please refer to *Motor_Control_Toolbox_License_Installation.pdf* for more information on how to request and install the license.

2.2.4 Setting the Path for Motor Control Development Toolbox

In the toolbox the path needs to be setup in the MATLAB environment. This is done by navigating the MATLAB Current Directory to the MC Toolbox/rappid564xl installation directory and running the “rappid_path” script.

```
>> rappid_path
Treating 'D:\MC_Toolbox\rappid564xl' as RAppID installation root.
RAppID path prepended.
Successful.
>>
```

2.2.5 Setting up the Target Compilers

The target compiler for Motor Control Development Toolbox to use will need to be configured. Ensure a system environment variable called <COMPILER_STRING>_TOOL is defined to value as shown below

DIAB_TOOL = C:/WindRiver/diab/5.9.0.0
GHS_TOOL = C:/GHS/multi517
MW_TOOL = C:/Freescale/"CW for MPC55xx and MPC56xx 2.8"
CW_TOOL = C:/Freescale/"CW MCU v10.6"

Note: Pls. use "/" when defining compiler path as shown above and pls. use" " around any directories that have spaces in the names eg: C:/"Program Files"/compiler path. Alternatively, install the compiler into path which does not contain spaces eg: C:/Freescale/CW_MCU_v10_6. Paths shown are for illustration, your installation path may be different. Once Environmental variables are setup you will need to restart MATLAB in order for the environment to see these.

2.2.1 Setting up the PC Compiler for MATLAB

In order to run SIL or PIL MATLAB needs a PC compiler defined so that it can use it to compile the C code on the PC. If you are using 32-bit MATLAB the LCC compiler that comes with MATLAB is what needs to be selected. The following is an example of how to setup you compiler or mex setup.

```
>> mex -setup

Would you like mex to locate installed compilers [y]/n? y

Select a compiler:
[1] Lcc-win32 C 2.4.1 in C:\PROGRA~1\MATLAB\R2013a\sys\lcc

[0] None

Compiler: 1

Please verify your choices:

Compiler: Lcc-win32 C 2.4.1
Location: C:\PROGRA~1\MATLAB\R2013a\sys\lcc

Are these correct [y]/n? y

Done . . .
```

For 64-bit MATLAB you will need to download and install the Microsoft SDK which will be used as the PC compiler. Please follow the procedure below.

1. Install the Microsoft SDK for the compiler: http://www.microsoft.com/en-us/download/details.aspx?id=8279
2. Change you mex setup (by typing mex –setup in the command window) in MATLAB to use the Microsoft SDK vs. the LCC (32-bit only) compiler.
3. If you encounter issues installing the Microsoft SDK please use the following link to help solve installation issues: http://www.mathworks.com/MATLABcentral/answers/95039-why-does-the-sdk-7-1-installation-fail-with-an-installation-failed-message-on-my-windows-system

2.2.2 Setting up the MCU for MCD Toolbox

To prepare the MCU to accept download requests from the MCD Toolbox a boot loader needs to be loaded into flash memory by a programmer/debugger tool. Once

this is done the MCD Toolbox will be able to download the application code generated from the model to perform PIL operations or to execute in stand-alone. The s-record file that needs to be manually programmed into the MCU is MPC564xL.rbf. This s-record contains the code for the boot loader that communicates with the MCD Toolbox. It is located in the MCD Toolbox installation directory under: ...\tools\BootLoader\RBF_Files. Once the boot loader is programmed and the MCU is reset, it is ready to receive application code from the Toolbox. The boot loader will stay resident until erased by the user.

If the user prefers to program the application code generated by toolbox with a separate programmer or debugger then the boot loader is not required. To perform PIL operation, however, the boot loader is required.

2.3 Example Models

The Motor Control Development Toolbox includes many demonstration models showing many different uses of the target peripheral blocks and optimized code blocks. To access these please find ..\rappid564xl\Examples folder at your MC Toolbox install path.

2.3.1 adc_pit_block

This model tests the ADC in different configuration modes. As a sample of the signal used LED's driven from the PIT.

2.3.2 can_simple_block

CAN module configured on 500K speed. Every 100ms it sends out a packet with constant value and ID 2004. Every 100ms it checks incoming data with ID 1010, and update output ports. This data can be viewed through FreeMaster tool.

2.3.3 CTU_ISR

This model tests the CTU trigger interrupt. CTU takes input signals from FlexPWM channel 3 (VAL2 and VAL3) and generates trigger events to External pin. eTimer channel counts input edges and toggles output. Output pin 54 toggles at every Trigger interrupt.

2.3.4 demoParkInvPILBlockRef

Example of Processor-In-the-Loop Block PIL Simulation and compare Simulation Model to On-Target Execution. This example utilized PIL Block Generation for PIL Mode Simulation.

2.3.5 demoParkInvPILModelRef

Example of Processor-In-the-Loop utilizing model reference to compare simulation model execution to on-target execution. This example utilized model reference PIL Mode Simulation.

2.3.6 demoParkInvSILBlockRef

Example of Software-In-the-Loop Simulation with Model Simulation Vs TargetCode Generated Block (SILTest1) in Simulink. This example utilized SIL Block Generation for SIL Mode Simulation.

2.3.7 demoParkInvSILModelRef

Example of Software-In-the-Loop Simulation with Model Simulation Vs TargetCode Generated Model (Processor) in Simulink. This example utilized model reference SIL Mode Simulation.

2.3.8 DigitalIO_block_demo

This model runs in RAM and tests 4 digital inputs and 4 digital outputs using a Stateflow model to generate a square wave as well as 3 PIT channels to control the frequency.

2.3.9 dspi_block

This model tests the DSPI in different configuration modes. As a sample of the signal used Internal Pulse Generator. Output data can be viewed on FreeMaster tool.

2.3.10 etimer_capture

This model tests the eTimer Input Capture Mode.

2.3.11 eTimer_QD

This model tests the eTimer QUADRATURE-COUNT Mode. eTimer channel decodes the primary and secondary inputs taken from FlexPWM0 channel 0 outputs as quadrature encoded signals and toggles output.

2.3.12 flexpwm_ctu_adc_block

This model tests the ADC and external pin output driven by CTU. CTU takes input signals from FlexPWM channel 3 (VAL2 and VAL3) and generates trigger events to External pin and ADC unit A channel 3. ADC generate an interrupt at the end of conversion.

2.3.13 flexpwm_ctu_adc_center

This model tests both ADC Converters, triggered by CTU at the midpoint of a PWM signal.

2.3.14 flexpwm_ctu_etimer

This model tests the eTimer driven by CTU. CTU takes input signals from FlexPWM channel 3 (VAL2 and VAL3) and generates trigger events to External pin and eTimer0 channel 1. eTimer channel counts input edges and toggles output.

2.3.15 FlexPWM_ISR

This model tests for FlexPWM compare interrupt. The model generates a simple phase-shifted PWM signals on A and B outputs of the FlexPWM channel 2. Output pin 54 toggles at every FlexPWM channel 2 compare interrupt to VAL2,VAL3,VAL4,VAL5.

2.3.16 FlexPWM_simple_and_compl

This model tests for FlexPWM simple output mode and FlexPWM complementary mode. The model generates a simple phase-shifted PWM signals on A and B outputs of the FlexPWM channel 2 and complementary PWM signals with dead time insertion on A and B outputs of the FlexPWM channel 3.

2.3.17 FlexPWM_three_phase

This model tests for FlexPWM three phase output. The model generates a three-phase complementary center-aligned PWM signals on A and B outputs of the FlexPWM channels 0,1,2 synchronized from channel 0 Master Sync signal.

2.3.18 MotorControl_0 through 5

These models test the Freescale Application Motor Control Library Blocks.

2.3.19 MPC564xL_FOC_SIL_PIL_Ref

Example of Processor-In-the-Loop utilizing model reference to compare simulation model execution to on-target execution. This example utilized model reference PIL Mode Simulation.

2.3.20 ReadWrite_Demo

This model demonstrates the DataMemRead and DataMemWrite blocks to configure and toggle a digital output.

2.3.21 sim_acc_test

For top model Accelerator mode simulation open sim_acc_test.mdl model and select "Start simulation".

2.3.22 sim_pil_target_block

For PIL block simulation open sim_pil_target_block.mdl model and select "Build Subsystem" for "PIL_block" block. After PIL block is generated open sim_pil_test.mdl block, remove "Model" block and insert generated PIL block instead. Then select "Start simulation".

2.3.23 sim_pil_test

For top model PIL simulation open sim_pil_test.mdl model and select "Start simulation".

2.3.24 sim_sil_target_block

For SIL block simulation open sim_sil_target_block.mdl model and select "Build Subsystem" for "SIL_block" block. After SIL block is generated open sim_sil_test.mdl block, remove "Model" block and insert generated PIL block instead. Then select "Start simulation".

2.3.25 sim_sil_test

For top model SIL simulation open sim_sil_test.mdl model and select "Start simulation".

2.3.26 SineWave_block

This model tests the Sine Wave Generator.

2.3.27 tests564xl

This model runs in Flash and tests 4 digital outputs using a Stateflow model to generate a square wave as well as 3 PIT channels to control the frequency.

3 Processors Supported

The Motor Control Development Toolbox Version 1.2.0 supports the MPC564xL Processor. Testing has been completed on pre-production qualified parts.

4 Required and Recommended Products

The required products are those that are a minimum for the Motor Control Development Toolbox to work. The recommended products are those that the toolbox has been tested with and would fit within the use case of embedded controls development for on-target rapid prototyping.

4.1 MATLAB Required and Recommended Products

Product	*Version Compatibility*	*Required or Recommended*
MATLAB	R2013a, R2013b, R2014a, R2014b, R2015a, R2015b	Required
Simulink	R2013a, R2013b, R2014a, R2014b, R2015a, R2015b	Required
Stateflow	R2013a, R2013b, R2014a, R2014b, R2015a, R2015b	Required
Simulink Coder	R2013a, R2013b, R2014a, R2014b, R2015a, R2015b	Required
Embedded Coder	R2013a, R2013b, R2014a, R2014b, R2015a, R2015b	Required
Fixed Point Blockset	R2013a, R2013b, R2014a, R2014b, R2015a, R2015b	Recommended
Signal Processing Toolbox	R2013a, R2013b, R2014a, R2014b, R2015a, R2015b	Recommended
MATLAB Coder	R2013a, R2013b, R2014a, R2014b, R2014a, R2015b	Recommended

4.2 Compiler Specific Code

Motor Control Development Toolbox supports code generation for the Diab, CodeWarrior and Green Hills compilers. The C and assembly code generated is specific to these compilers.

Compiler	Versions Tested
Wind River Diab	5.9
Green Hills	5.1.7
CodeWarrior Classic	2.8
CodeWarrior Eclipse	10.6

5 MC Toolbox Integration into Simulink

5.1 Target Configuration Block

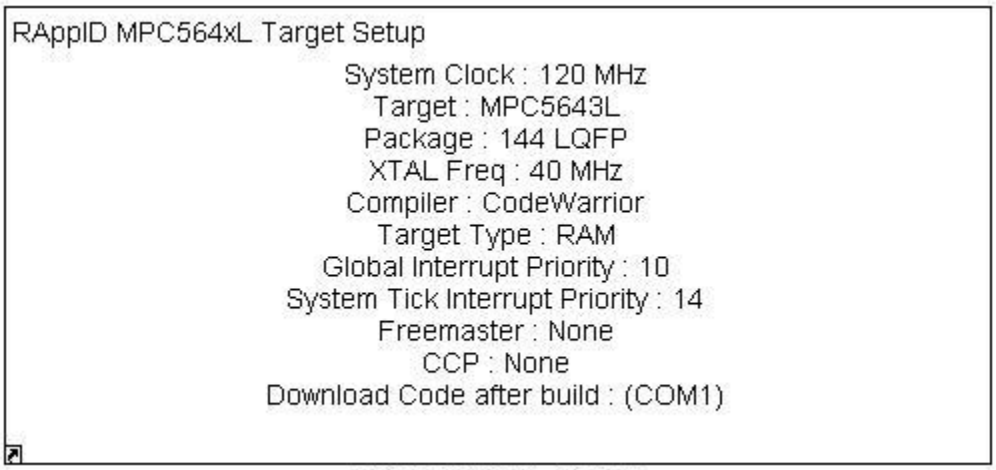
5.1.1 Block Name

Target Configuration Block

5.1.2 Block Description

The main functionality of the block is to configure MPC5643L target.

5.1.3 Block Image



RAppID_564xL_Config

5.1.4 Inputs:

- None

5.1.5 Outputs:

- None

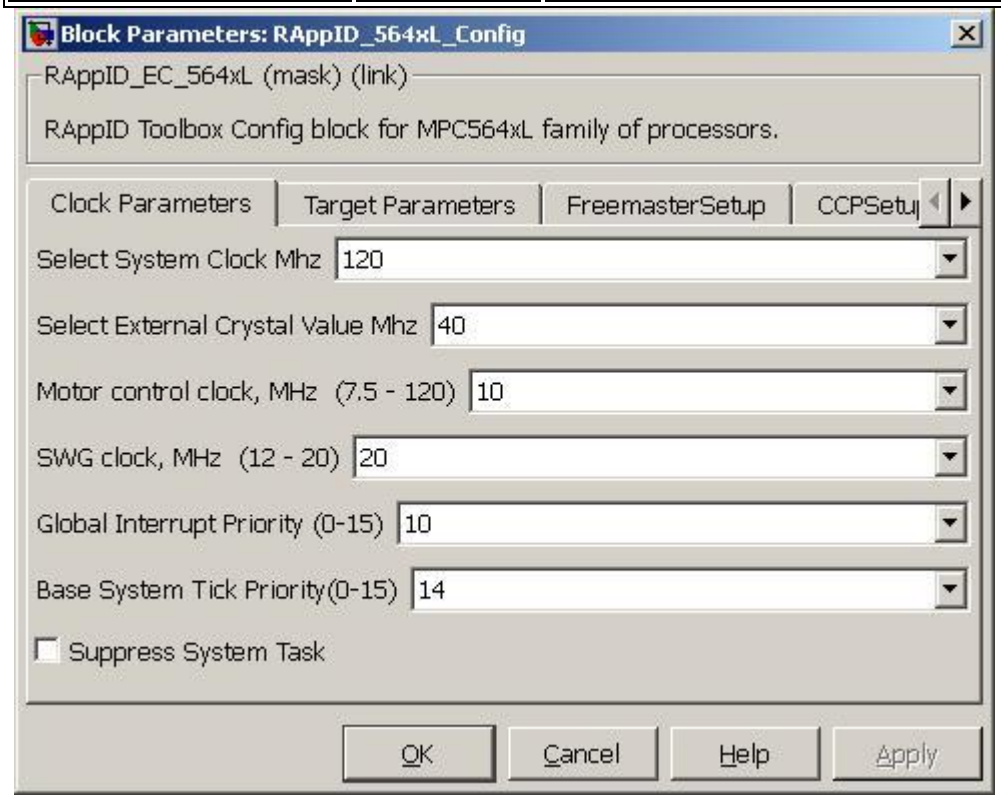
5.1.6 Block Dialog and Parameters:

The block dialog consists of the following tabs:

- Clock Parameters
- Target Parameters
- FreemasterSetup
- CCPSetup
- PIL/BAM Setup
- Diagnostics

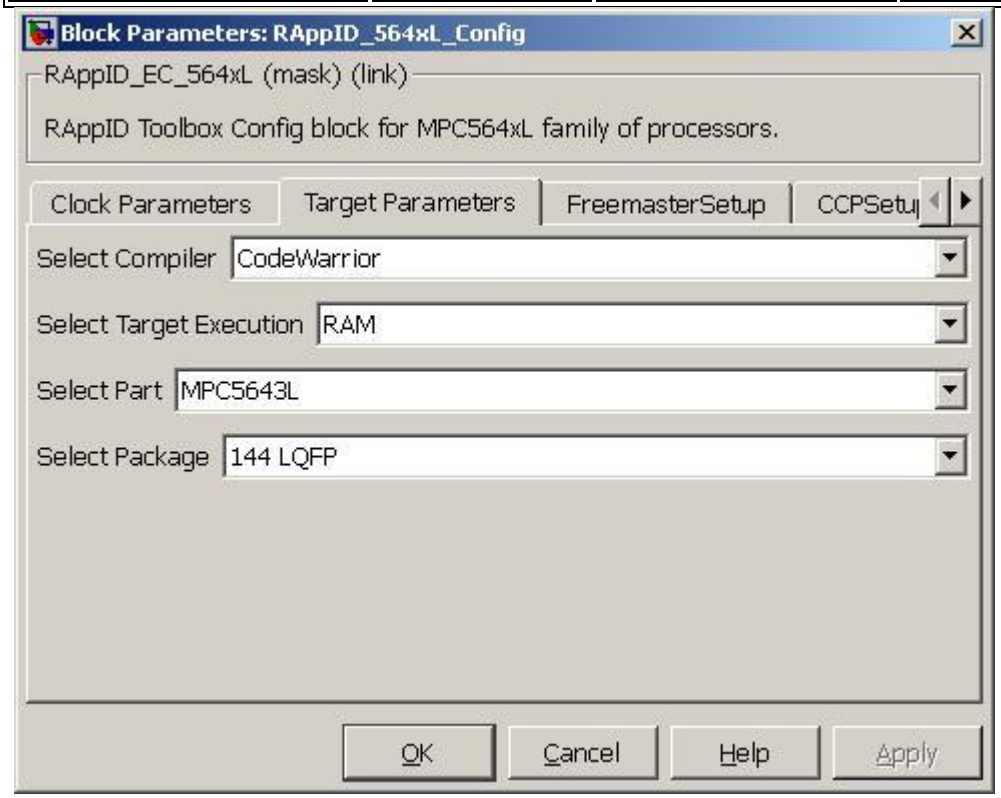
- The Clock Parameters tab contains the following parameters:

Names	Selection Types	Range	Description
Select System Clock Mhz	Pop-up	60, 80, 120	System Clock Value
External Crystal Value (8 or 40 Mhz)	Pop-up	8, 16, 40 Mhz	External Crystal clock
Motor control clock, MHz	Pop-up	System clock divided by 1..16	Motor control clock
SWG clock, MHz	Pop-up	System clock divided by 1..16 in range 12-20 MHz	SWG clock
Global Interrupt Priority (0-15)	Pop-up	0 – 15	Global Minimum Interrupt Priority
Base System Tick Priority (0-15)	Pop-up	0 – 15	Priority level of PIT 0 interrupt
Suppress System Task	Check-box	Enable/Disable	



- The Target Parameters tab contains the following parameters:

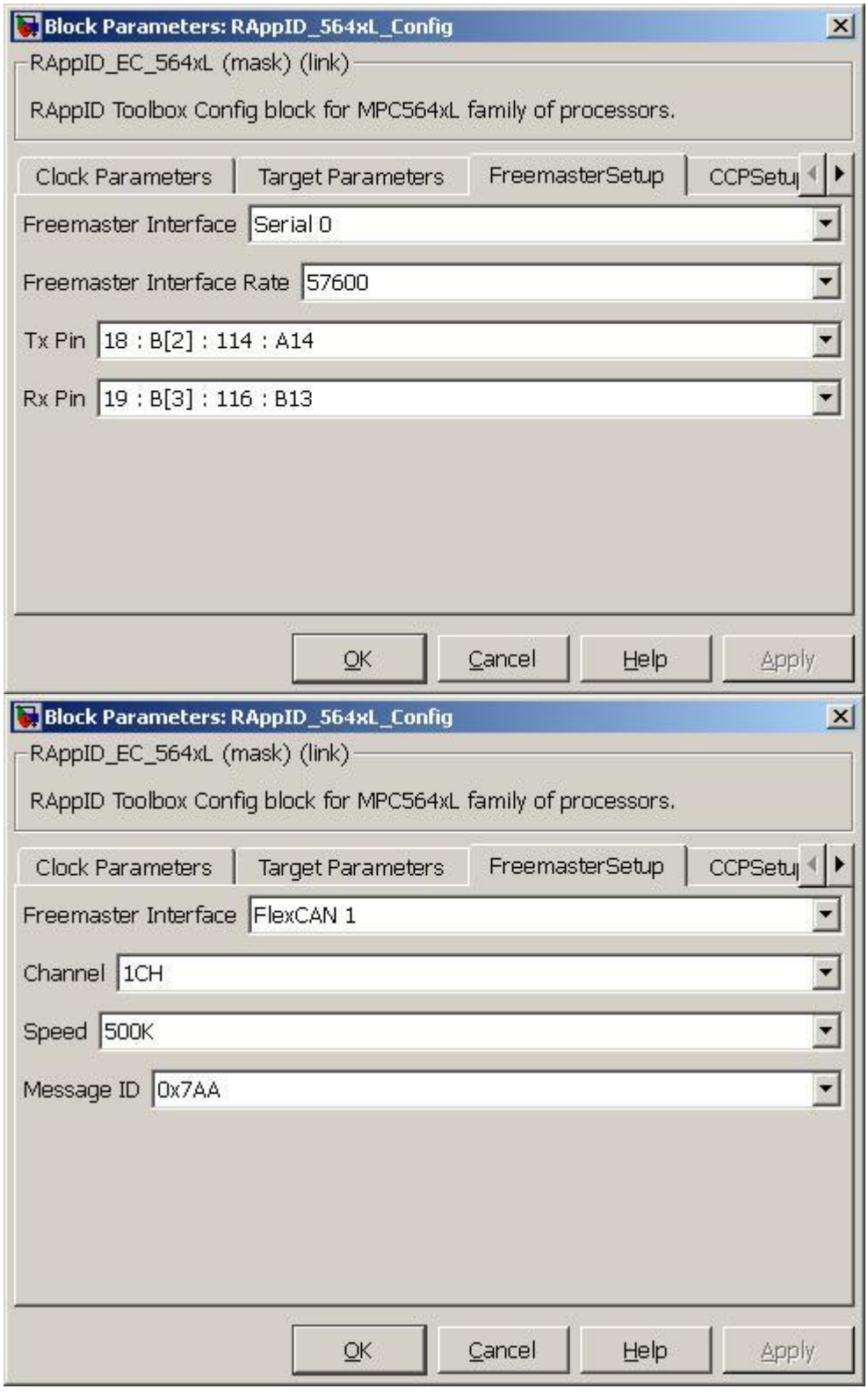
Names	Selection Types	Range	Description
Select Compiler	Pop-up	CodeWarrior Diab GreenHills	
Select Target Execution	Pop-up	RAM FLASH	
Select Part	Pop-up	MPC5643L	
Select Package	Pop-up	144 LQFP 257 MAPBGA	



- The FreemasterSetup tab contains the following parameters:

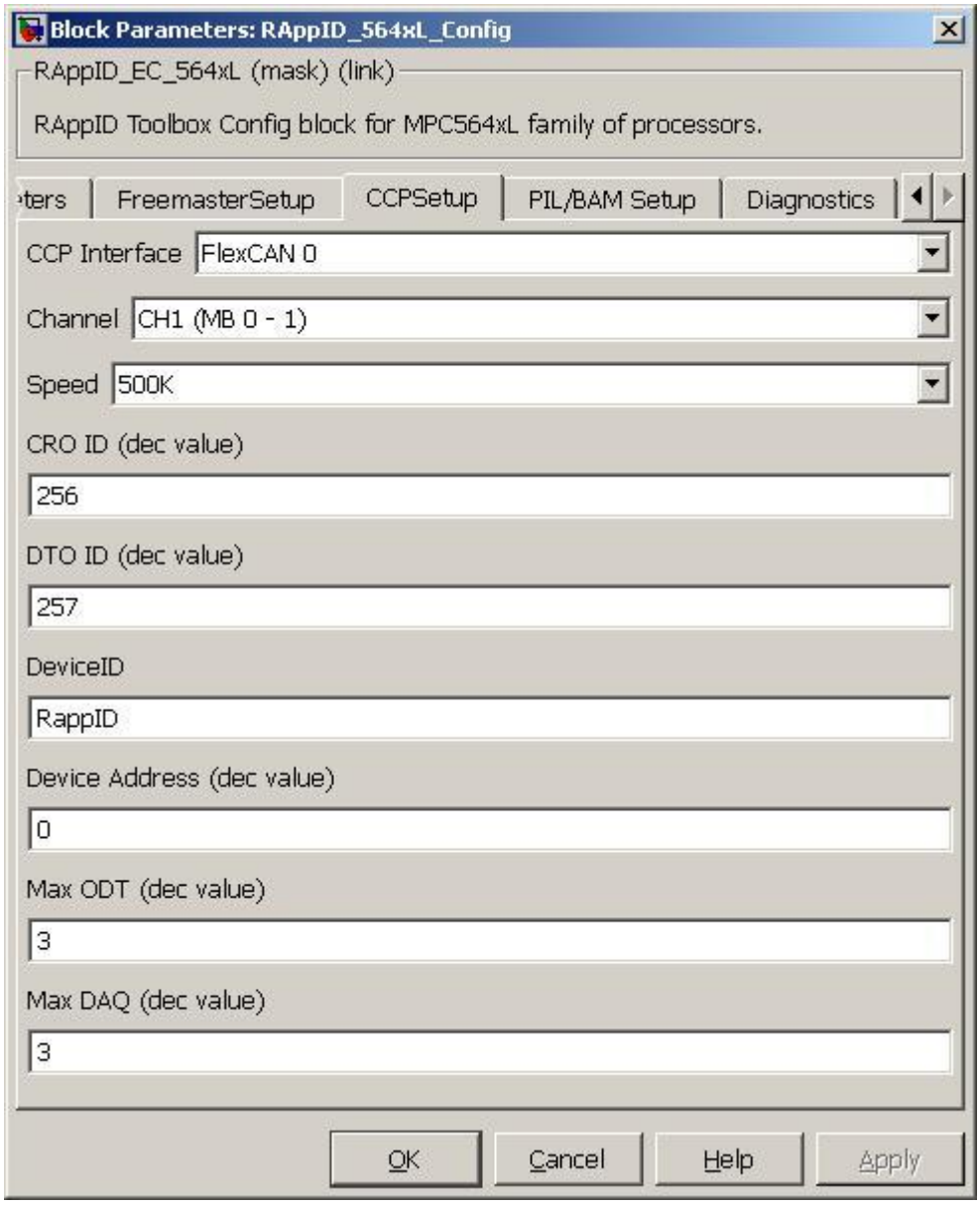
Names	Selection Types	Range	Description
Freemaster Interface	Pop-up	None Serial 0 Serial 1 FlexCAN 0	

		FlexCAN 1	
Freemaster Interface Rate	Pop-up	19200 – 230400	applicable for Serial interface only
Tx Pin	Pop-up	The list of available pins	applicable for Serial interface only
Rx Pin	Pop-up	The list of available pins	applicable for Serial interface only
Channel	Pop-up	1CH	applicable for CAN interface only
Speed	Pop-up	500K	applicable for CAN interface only
Message ID	Pop-up	0x7AA	applicable for CAN interface only



- The CCP Setup tab contains the following parameters:

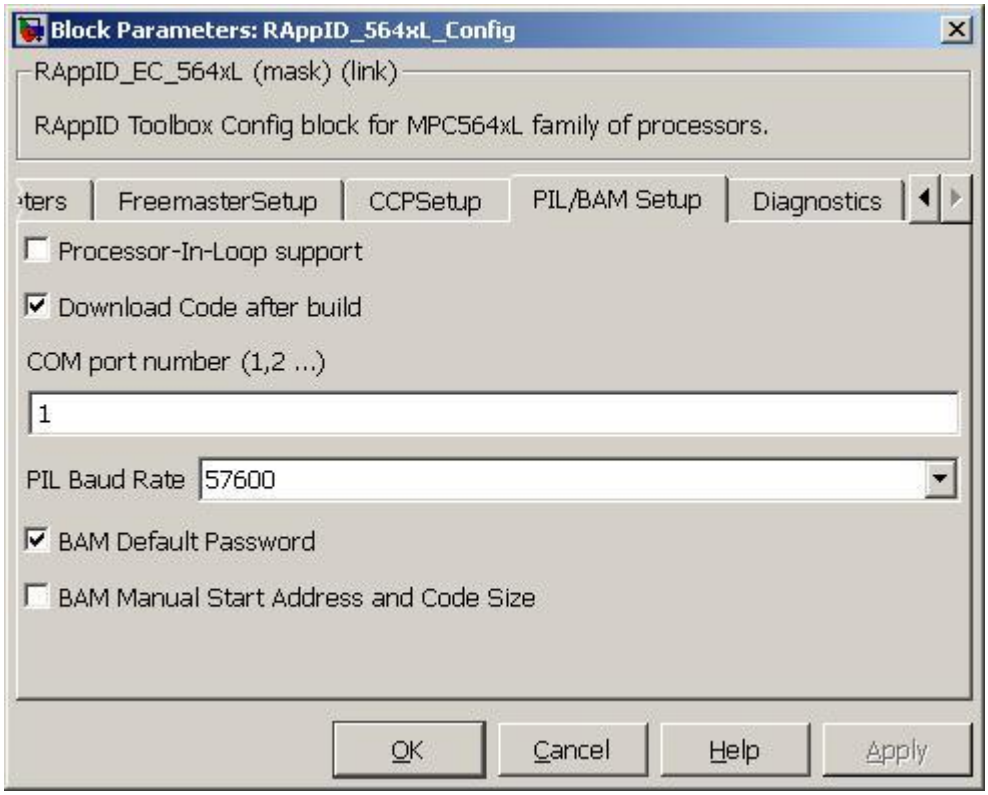
Names	Selection Types	Range	Description
CCP Interface	Pop-up	None FlexCAN 0 FlexCAN 1	
Channel	Pop-up	CH1 (MB 0 - 1) CH1 (MB 2 - 3) CH3 (MB 4 - 5) CH4 (MB 6 - 7) CH5 (MB 8 - 9) CH6 (MB 10 - 11) CH7 (MB 12 - 13) CH8 (MB 14 - 15)	FlexCAN mailbox used for send/receive CCP messages
Speed	Pop-up	500K	
CRO ID (dec value)	Text-box		Income message ID
DTO ID (dec value)	Text-box		Outgoing message ID
DeviceId	Text-box		Device ID
Devica Address (dec value)	Text-box		Devica Address
Max ODT (dec value)	Text-box		Max measurable values in Object description table
Max DAQ (dec value)	Text-box		Max number of Object description tables in the DAQ list



- The PIL/BAM Setup tab contains the following parameters:

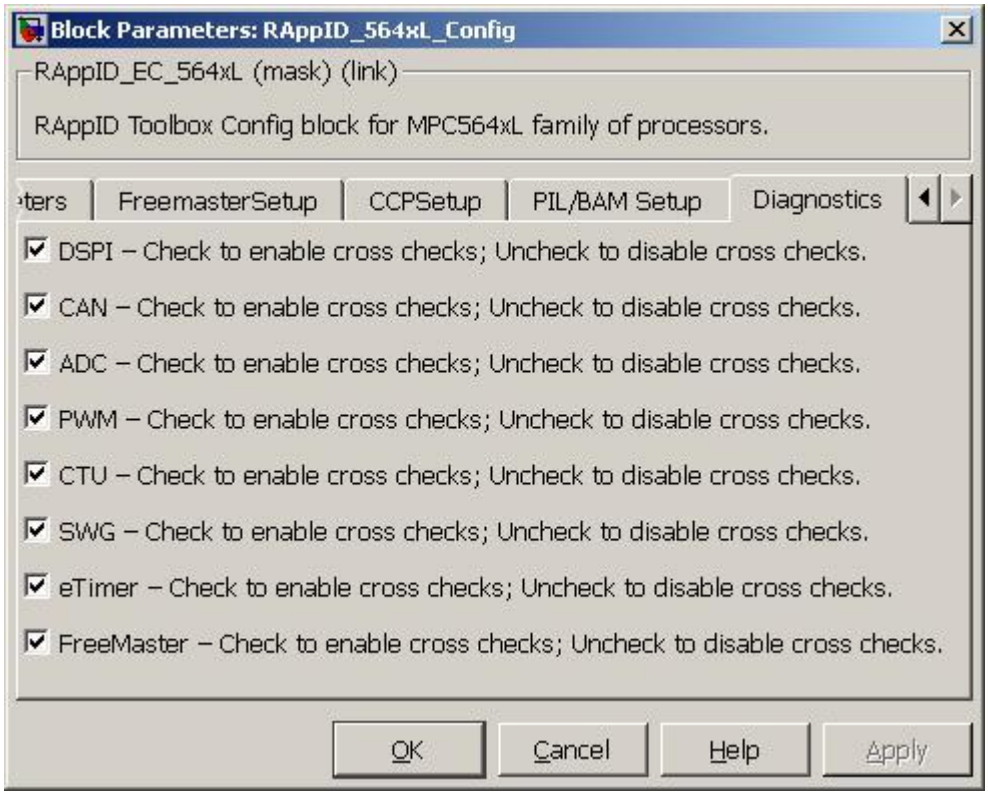
Names	Selection Types	Range	Description
Processor-In-Loop support	Check-box	On/Off	If PIL simulation mode should be supported.
Download Code after build	Check-box	On/Off	If application code should be downloaded

			into target after build
COM port number (1,2 …)	Text-box		COM port for PIL and BootLoader communication
PIL Baud rate	Pop-up	19200, 38400, 57600, 115200, 230400	Baud rate for PIL communication
BAM Default Password	Check-box	On/Off	
BAM Password	Text-box		applicable when “BAM Default Password” option is disabled only
BAM Manual Start Address and Code Size	Check-box	On/Off	
BAM Start Address	Text-box		applicable when “BAM Manual Start Address and Code Size” option is enabled only
BAM Code Size	Text-box		applicable when “BAM Manual Start Address and Code Size” option is enabled only



- The Diagnostics tab contains the following parameters:

Names	Selection Types	Range	Description
DSPI CAN ADC PWM CTU SWG eTimer FreeMaster	Check-box	On/Off	Enable/disable cross-module PCR assignment



5.1.7 Block Dependency

None

5.1.8 Block Miscellaneous Details:

Target Configuration Block must be included into the model. The model can contain only one Target Configuration Block.

5.2 Block Details

Motor Control Development Toolbox provides an easy to use graphic interface allowing configuration of individual peripherals. Each peripheral can be configured by selecting its block on the Motor Control Development Toolbox startup screen. When a block is selected, that peripheral's GUI will be displayed. When finished, click OK and the peripheral's GUI will be saved and closed. More detailed information about specific blocks is found in the tutorial for that block.

5.2.1 Layout of Block GUIs

In the Motor Control Development Toolbox library there are two types of block GUIs: Freescale Application Motor Control Library(application blocks) and Motor Control Development Toolbox for 564xL(peripheral blocks). The Freescale Application Motor Control Library contains independent blocks to be used in your motor control application model. The Motor Control Development Toolbox contains configuration-dependent blocks for controlling the low level drivers and I/O of the target. These blocks are considered configuration-dependent because the RAppID_564xL_Config block **must**

be used when any of the other blocks in the library is used within the model. In some cases, there are additional Config blocks that must be used as well, for example, CAN Config, DSPI Config, ADC Config, and eTimer Config. The blocks within the Motor Control Development Toolbox library are not all supported in all modes. Please use the following table to determine the support of each block.

Library Block Support by Mode

Block	Simulation	SIL	PIL	Target
Freescale Application Motor Control Library Blocks	All Blocks	All Blocks	All Blocks	All Blocks
RAppID_564xL_Config	X	X	X	X
CAN Config				X
CAN ISR				X
CAN Receive Data Trigger				X
CAN Transmit				X
CAN Receive_data				X
DSPI Config				X
DSPI ISR				X
DSPI Receive				X
DSPI Receive Trigger				X
DSPI Transmit				X
Digital Input			X	X
Digital Output			X	X
Periodic Interrupt Timer				X
ADC Channel	X	X	X	X
ADC Config	X	X	X	X
ADC ISR				X
ADC Trigger				X
ADC WatchDog Threshold				X
CTU				X
CTU ISR				X
Complementary PWM Output			X	X
FlexPWM ISR				X
Simple PWM Output			X	X
Sine Wave Generator			X	X
Three Phase PWM Output			X	X
eTimer ISR				X
eTimer capture			X	X
eTimer configuration			X	X
eTimer pre-load				X
Profiler Function			X	X
Memory Read				X
Memory Write				X

6 Peripheral Blocks

A large part of moving from the modeling and simulation environment is connecting to real world sensors and actuators. A major portion of this is using hardware peripherals integrated with the application in a straight forward and simplistic manner. The Motor Control Development Toolbox provides these interface blocks with the objective of allowing the user to do minimal configuration of the peripheral with the ability to completely configure the peripheral as desired. Certain configuration items are assumed in order to have a straight forward method of configuring the peripheral with the application; these constraints are spelled out in the use case of the peripheral.

6.1 Communication Blocks

6.1.1 FlexCAN

Flexible CAN interface to CAN network message transmission and reception. MPC564xL has 2 FlexCAN modules. Each module has 32 buffers to transmit and receive CAN messages

6.1.1.1 FlexCAN Configuration Block

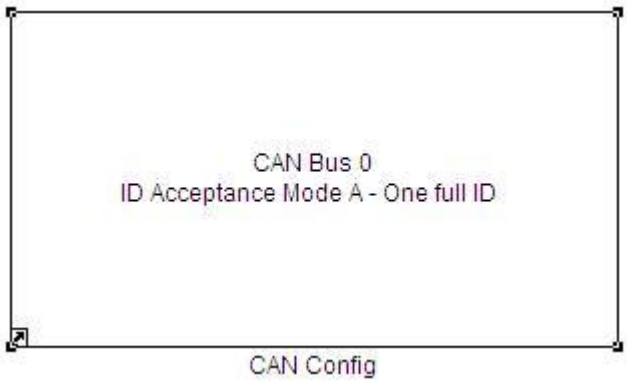
6.1.1.1.1 Block Name

FlexCAN Configuration Block

6.1.1.1.2 Block Description

The main functionality of the block is to configure FlexCAN module.

6.1.1.1.3 Block Image



6.1.1.1.4 Inputs:

- None

6.1.1.1.5 Outputs:

- None

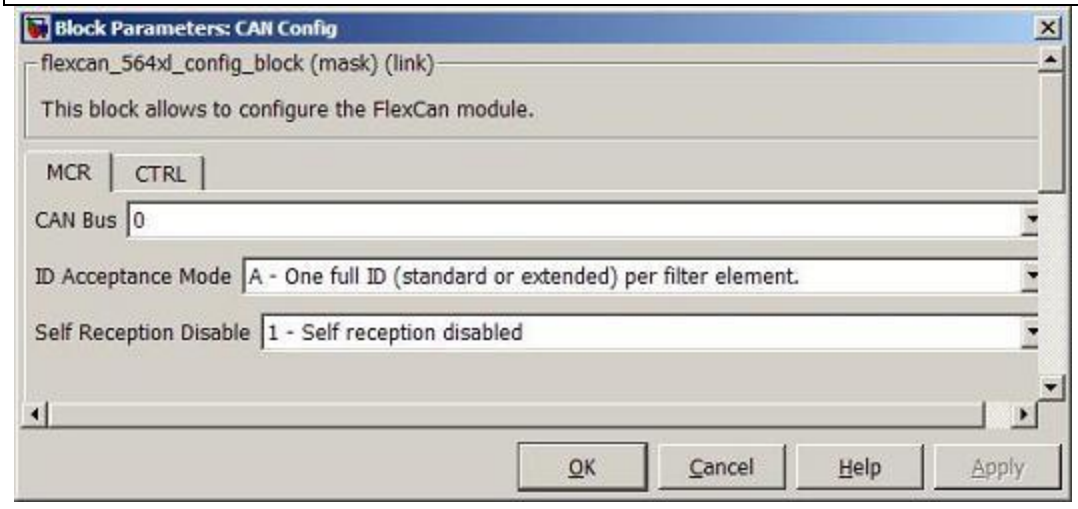
6.1.1.1.6 Block Dialog and Parameters:

The block dialog consists of the following tabs:

- MCR
- CTRL

- The MCR tab contains the following parameters:

Names	Selection Types	Range	Description
CAN Bus	Pop-up	0 or 1	Select the CAN Bus name
ID Acceptance Mode	Pop-up	A, B, C or D	A – One full ID (standard or extended) per filter element. B – Two full standard IDs or two partial 14-bit extended IDs per filter element. C – Four partial 8-bit IDs (standard or extended) per filter element. D – All frames rejected.
Self Reception Disable	Pop-up	0 – 1	0 – Self reception enabled 1 – Self reception disabled



- The CTRL tab contains the following parameters:

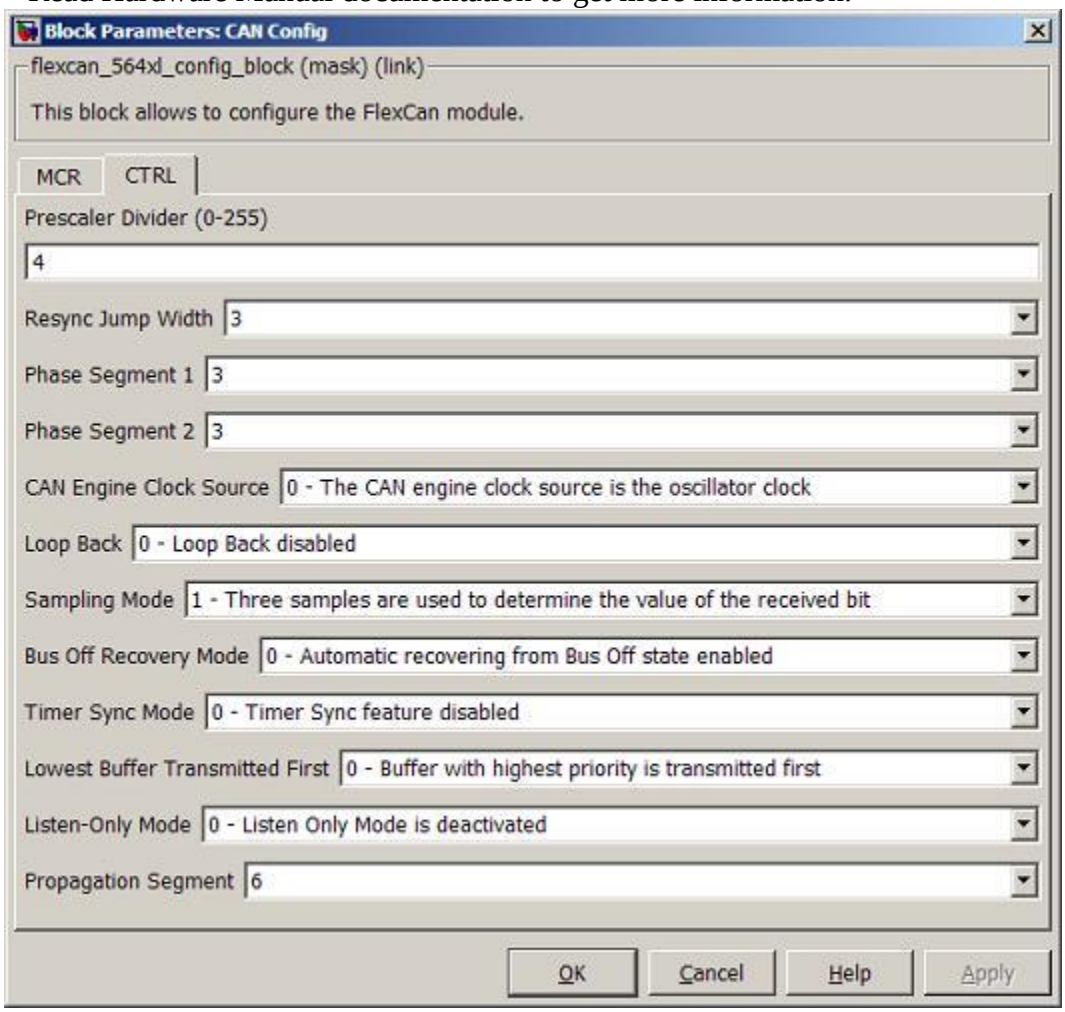
Names	Selection Types	Range	Description
Prescaler Divider (0-255)	Text-box	0 – 255	This 8-bit field defines the ratio between the CPI clock frequency and the Serial Clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the CPI clock frequency. The Maximum value of this register is 0xFF, that gives a minimum Sclock frequency equal to the CPI clock frequency divided by 256
Resync Jump Width	Pop-up	0 – 3	This 2-bit field defines the maximum number of time quanta1 that a bit time can be changed by one re-synchronization
Phase Segment 1	Pop-up	0 – 7	This 3-bit field defines the

			length of Phase Buffer Segment 1 in the bit time. The valid programmable values are 0–7.
Phase Segment 2	Pop-up	0 – 7	This 3-bit field defines the length of Phase Buffer Segment 2 in the bit time. The valid programmable values are 0–7.
CAN Engine Clock Source	Pop-up	0 – The CAN Engine Clock Source is the oscillator clock 1 – The CAN Engine Clock Source is the bus clock	This bit selects the clock source to the CAN Protocol Interface (CPI) to be either the peripheral clock (driven by the FMPLL) or the crystal oscillator clock. The selected clock is the one fed to the prescaler to generate the Serial Clock (Sclock)
Loop Back	Pop-up	0 – Loop Back disabled 1 – Loop Back enabled	This bit configures FlexCAN to operate in Loop-Back Mode. In this mode, FlexCAN performs an internal loop back that can be used for self test operation

Samplig Mode	Pop-up	0 – One sample is used to determine the value of the received bit 1 – Three samples are used to determine the value of the received bit	This bit defines the sampling mode of CAN bits at the Rx input
Bus Off Recovery Mode	Pop-up	0 – Automatic recovery from Bus Off state enabled 1 – Automatic recovery from Bus Off state disabled	This bit defines how FlexCAN recovers from Bus Off state
Timer Sync Mode	Pop-up	0 – Timer Sync feature disabled 1 – Timer Sync feature enabled	This bit enables a mechanism that resets the free-running timer each time a message is received in Message Buffer 0
Lowest Buffer Transmitted First	Pop-up	0 – Buffer with highest priority is transmitted first 1 – Lowest number buffer is transmitted first	This bit defines the ordering mechanism for Message Buffer transmission
Listen-Only Mode	Pop-up	0 – Listen Only Mode is deactivated 1 – FlexCAN module operates in Listen Only Mode	This bit configures FlexCAN to operate in Listen Only Mode. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode
Propagation Segment	Pop-up	0 – 7	This 3-bit field defines the length of the Propagation Segment in the bit time. The

			valid programmable values are 0–7.

* Read Hardware Manual documentation to get more information.



6.1.1.1.7 Block Dependency

Please do the following:

1. Configure the FlexCAN module.

6.1.1.1.8 Block Miscellaneous Details:

FlexCAN module has predefined pin assignment:

FlexCAN Module	Tx pin	Rx pin
0	GPIO[16]	GPIO[17]
1	GPIO[14]	GPIO[15]

6.1.1.2 FlexCAN Receive Data Block

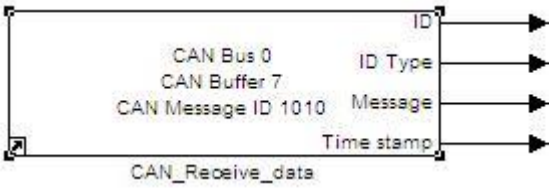
6.1.1.2.1 Block Name

FlexCAN Receive Data Block

6.1.1.2.2 Block Description

The main functionality of the block is receiving data through the FlexCAN interface. The block waiting till message will come.

6.1.1.2.3 Block Image



6.1.1.2.4 Inputs:

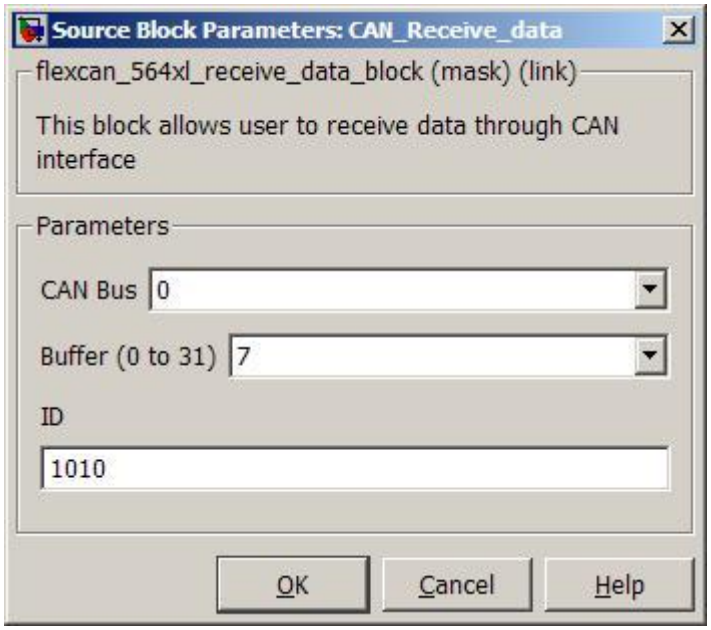
- None

6.1.1.2.5 Outputs:

- ID (uint8)
- ID Type (uint8)
- Message (uint8(8))
- Time stamp (unit16)

6.1.1.2.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
CAN Bus	Pop-up	0 or 1	CAN Module
Buffer	Pop-up	0 – 31	Buffer Number
ID	Text-box		ID of the message to receive



6.1.1.2.7 Block Dependency

Please do the following:

1. Configure FlexCAN

6.1.1.2.8 Block Miscellaneous Details:

Please refer "CAN Configiration block" to get information about pin assignment.

6.1.1.3 FlexCAN Receive Data Trigger Block

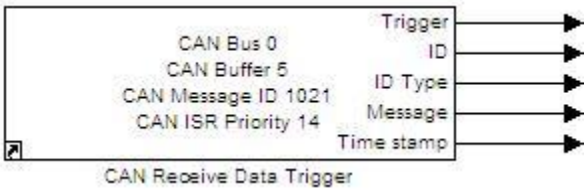
6.1.1.3.1 Block Name

FlexCAN Receive Data Trigger Block

6.1.1.3.2 Block Description

The main functionality of the block is to call user function when FlexCAN message will come.

6.1.1.3.3 Block Image



6.1.1.3.4 Inputs:

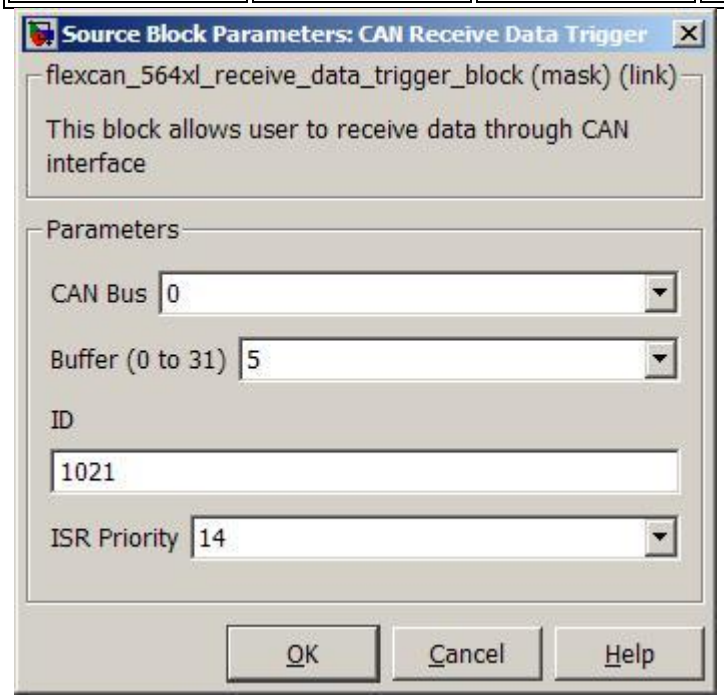
- None

6.1.1.3.5 Outputs:

- Function-Call
- ID (uint8)
- ID Type (uint8)
- Message (uint8(8))
- Time stamp (unit16)

6.1.1.3.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
CAN Bus	Pop-up	0 or 1	CAN Module
Buffer	Pop-up	0 – 31	Buffer Number
ID	Text-box		ID of the message to receive
ISR Priority	Pop-up	0 – 15	Global ISR Priority



6.1.1.3.7 Block Dependency

Please do the following:

1. Configure FlexCAN

6.1.1.3.8 Block Miscellaneous Details:

Please refer "CAN Configiration block" to get information about pin assignment.

6.1.1.4 FlexCAN Transmit Block

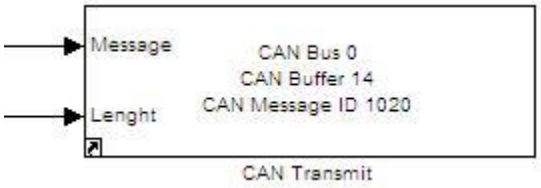
6.1.1.4.1 Block Name

FlexCAN Transmit Data Block

6.1.1.4.2 Block Description

The main functionality of the block is transmitting data through the FlexCAN interface.

6.1.1.4.3 Block Image



6.1.1.4.4 Inputs:

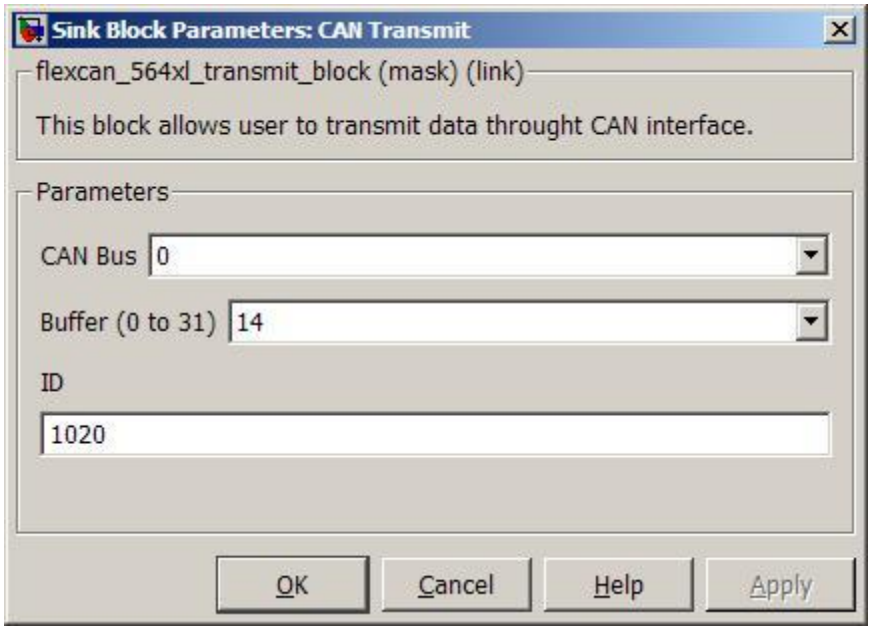
- Message (uint8(8))
- Length (uint8)

6.1.1.4.5 Outputs:

- None

6.1.1.4.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
CAN Bus	Pop-up	0 or 1	CAN Module
Buffer	Pop-up	0 – 31	Buffer Number
ID	Text-box		ID of the message to receive



6.1.1.4.7 Block Dependency

Please do the following:

1. Configure FlexCAN

6.1.1.4.8 Block Miscellaneous Details:

Please refer "CAN Configiration block" to get information about pin assignment.

6.1.1.5 CAN ISR Block

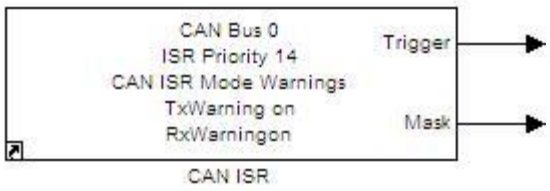
6.1.1.5.1 Block Name

FlexCAN ISR Block

6.1.1.5.2 Block Description

The main functionality of the block is to process FlexCAN ISR's.

6.1.1.5.3 Block Image



6.1.1.5.4 Inputs:

- None

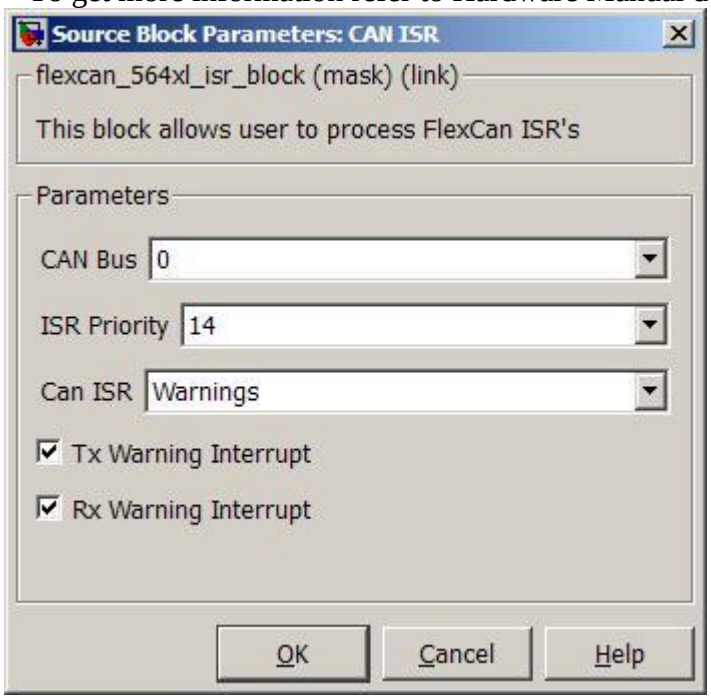
6.1.1.5.5 Outputs:

- Function-call
- Mask (uint32) - contains ESR register value when interrupt occurs.*

6.1.1.5.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
CAN Bus	Pop-up	0 or 1	CAN Module
ISR Priority	Pop-up	0 – 15	Global ISR Priority
CAN ISR	Pop-up	Errors/Warnings	Interrupt source
Tx Warning Interrupt	Check-box	On/Off	applicable if "CAN ISR" is set to "Warnings"
Rx Warning Interrupt	Check-box	On/Off	applicable if "CAN ISR" is set to "Warnings"

* To get more information refer to Hardware Manual documentation.



6.1.1.5.7 Block Dependency

Please do the following:

1. Configure FlexCAN

6.1.1.5.8 Block Miscellaneous Details:

None

6.1.1.6 FlexCAN ISR Enable/Disable

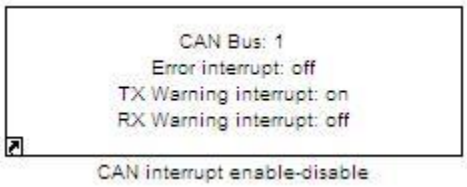
6.1.1.6.1 Block Name

FlexCAN Interrupt Enable/Disable Block

6.1.1.6.2 Block Description

The main functionality of the block is to enable/disable FlexCAN interrupts

6.1.1.6.3 Block Image



6.1.1.6.4 Inputs:

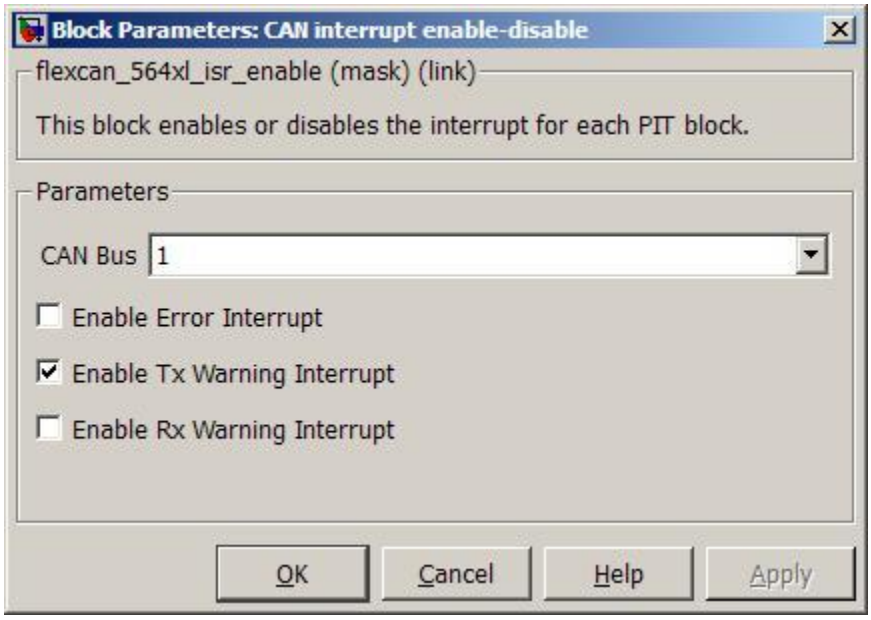
- None

6.1.1.6.5 Outputs:

- None

6.1.1.6.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
CAN Bus	Pop-up	0 or 1	CAN Module
Enable Error interrupt	Check-box	On/Off	Enable/Disable Error interrupt
Enable Tx Warning interrupt	Check-box	On/Off	Enable/Disable Tx Warning interrupt
Enable Rx Warning interrupt	Check-box	On/Off	Enable/Disable Rx Warning interrupt



6.1.1.6.7 Block Dependency

Please do the following:

1. Configure respective FlexCAN and its interrupts

6.1.1.6.8 Block Miscellaneous Details:

None

6.1.2 SPI Interface Blocks

6.1.2.1 DSPI Configuration Block

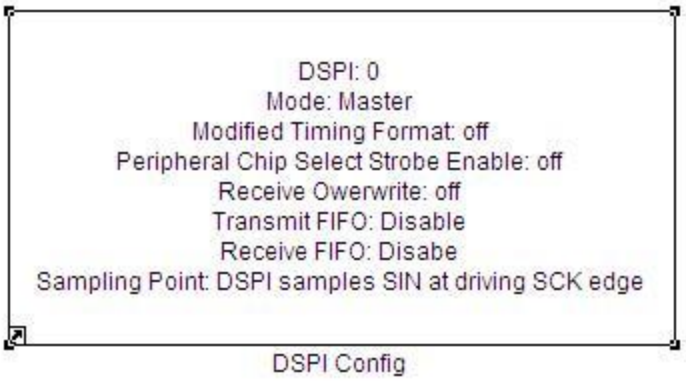
6.1.2.1.1 Block Name

DSPI Configuration Block

6.1.2.1.2 Block Description

This block is used to configure the DSPI module.

6.1.2.1.3 Block Image



DSPI Config

6.1.2.1.4 Inputs:

- None

6.1.2.1.5 Outputs:

- None

6.1.2.1.6 Block Dialog and Parameters:

The block dialog consists of the following tabs:

- General
- Transfer Mode
- Chip Select State

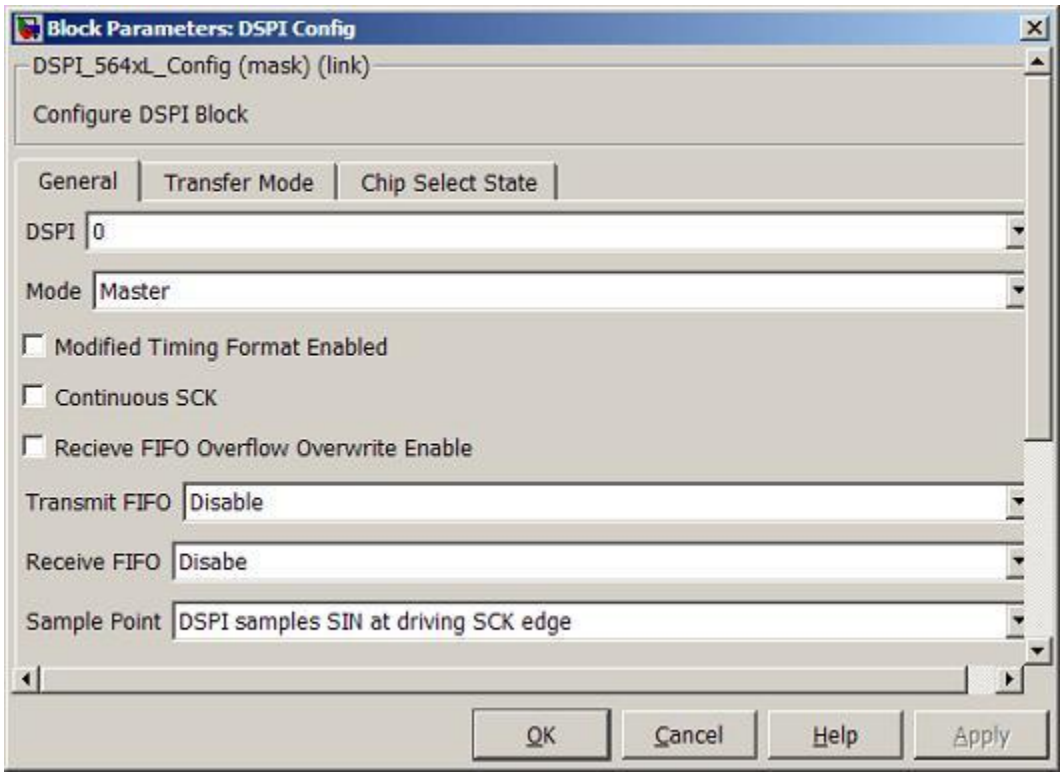
- The General tab contains the following parameters:

Names	Selection Types	Range	Description
DSPI	Pop-up	0 – 2	Select the Serial Communication Interface module which should be used for the receive of the data *
Mode	Pop-up	Master Slave	Mode of operation * When Slave mode is selected all

			unnecessary settings disabled.
Modified Timing Format Enabled	Check-box	Enable/Disable	The MTFE bit enables a modified transfer format to be used *
Continuous SCK	Check-box	Enable/Disable	The Continuous SCK bit enables the Serial Communication Clock (SCK) to run continuously *
Receive FIFO Overflow Overwrite Enable	Check-box	Enable/Disable	Receive FIFO Overflow Overwrite Enable. The ROOE bit enables in RX FIFO overflow condition to ignore the incoming serial data or to overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer, generated the overflow, is ignored or shifted in to the shift register *
Transmit FIFO	Pop-up	Enable Disable	When TXF isn't set, the TX FIFO acts as a single-entry

			(unit depth) FIFO. Therefore, serial operation is performed as if the FIFO has only one valid entry space for serial-word transfer * The FIFO mode doesn't support, should be always Disabled.
Receive FIFO	Pop-up	Enable Disable	When RXF isn't set, the RX FIFO acts as a single-entry (unit depth) FIFO. Therefore, serial operation is performed as if the FIFO has only one valid entry space for serial-word * The FIFO mode doesn't support, should be always Disabled.
Sample Point	Pop-up	– DSPI samples SIN at driving SCK edge – DSPI samples SIN one system clock after driving SCK edge – DSPI samples SIN two system clocks after driving SCK edge	Sample Point. SMPL_PT field controls when the DSPI master samples SIN in Modified Transfer Format *

* Read Hardware Manual documentation to get more information.



- The Transfer Mode tab contains the following parameters:

Names	Selection Types	Range	Description
Frame Size	Pop-up	3 c 16	The number of bits transferred per frame
Clock Polarity	Pop-up	Low High	The CPOL bit selects the inactive state of the Serial Communications Clock (SCK). This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock polarities.

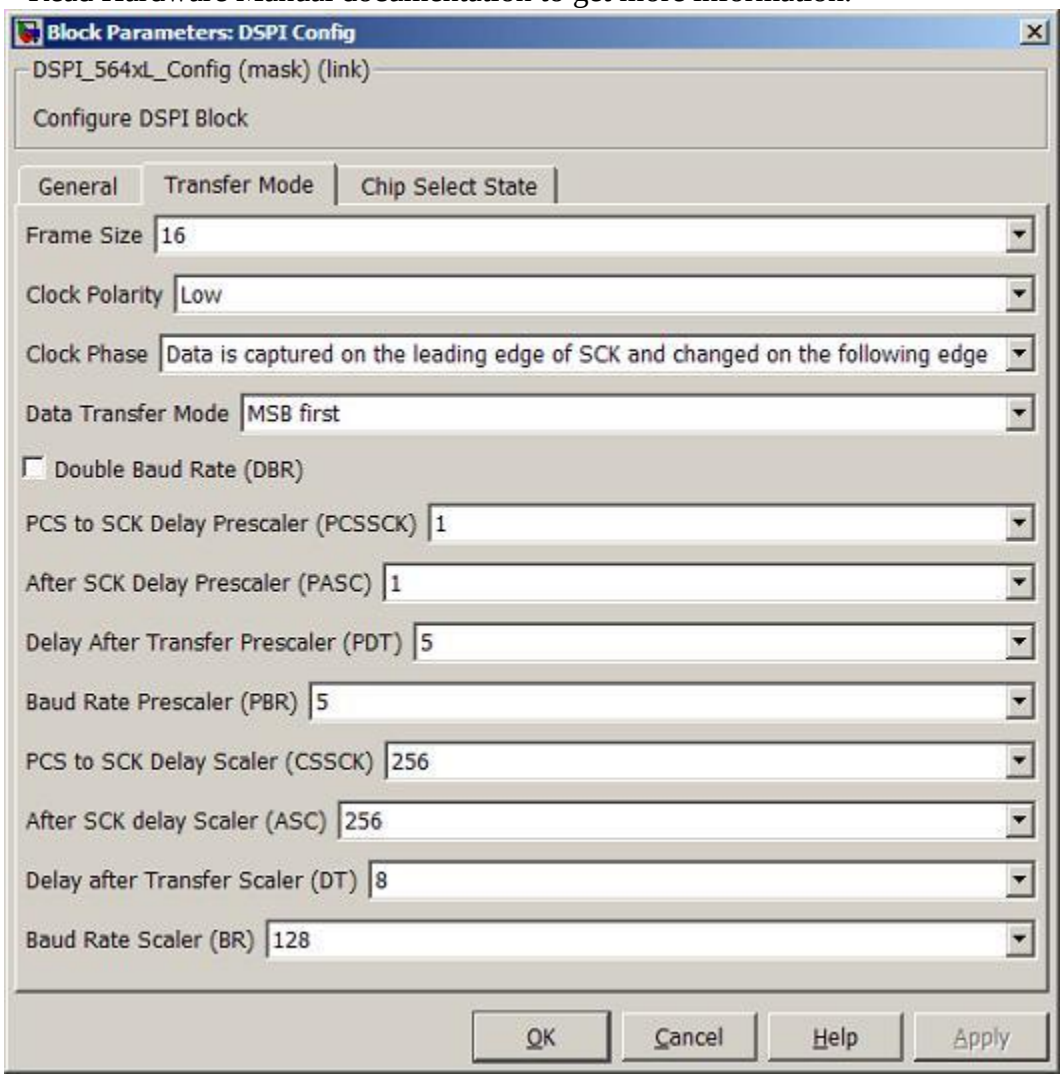
			When the Continuous selection format is selected, switching between clock polarities without stopping the DSPI can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge.
Clock Phase	Pop-up	– Data is captured on the leading edge of SCK and changed on the following edge – Data is changed on the leading edge of SCK and captured on the following edge	The CPHA bit selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode the bit value is ignored and the transfers are done as CPHA bit is set to 1
Data Transfer Mode	Pop-up	MSB first LSB first	Selects LSB or MSB of the

			frame transferred first
Double Baud Rate (DBR)	Check-box	Enable/Disable	Double Baud Rate. The DBR bit doubles the effective baud rate of the Serial Communications Clock (SCK). This field is only used in master mode. It effectively halves the Baud Rate division ratio supporting faster frequencies and odd division ratios for the Serial Communications Clock (SCK). *
PCS to SCK Delay Prescaler (PCSSCK)	Pop-up	1,3,5,7	The PCSSCK field selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK *
After SCK Delay Prescaler (PASC)	Pop-up	1,3,5,7	Selects the prescaler value for the delay between the last edge of SCK and the negation of PCS *
Delay After Transfer Prescaler (PDT)	Pop-up	1,3,5,7	The PDT field selects the prescaler value for the delay between the

			negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The PDT field is only used in master mode *
Baud Rate Prescaler (PBR)	Pop-up	1,3,5,7	The PBR field selects the prescaler value for the baud rate. This field is only used in master mode. The Baud Rate is the frequency of the Serial Communications Clock (SCK). The system clock is divided by the prescaler value before the baud rate selection takes place. *
PCS to SCK Delay Scaler (CSSCK)	Pop-up	2^N, where N in the range of 1 to 16	The CSSCK field selects the scaler value for the PCS to SCK delay. This field is only used in master mode. *
After SCK delay Scaler (ASC)	Pop-up	2^N, where N in the range of 1 to 16	The ASC field selects the scaler value for the After SCK Delay. This field is only used in master mode. *

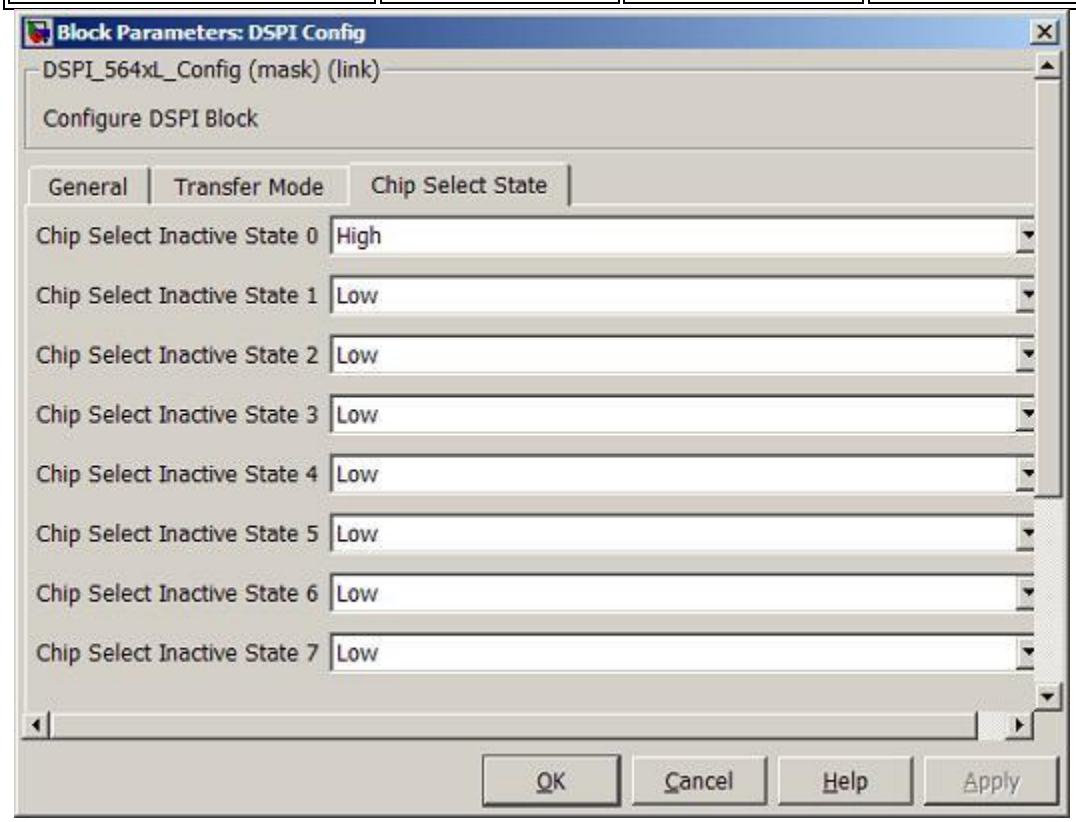
Delay after Transfer Scaler (DT)	Pop-up	2^N, where N in the range of 1 to 16	The DT field selects the Delay after Transfer Scaler. This field is only used in master mode. *
Baud Rate Scaler (BR)	Pop-up	2^N, where N in the range of 1 to 16	The BR field selects the scaler value for the baud rate. This field is only used in master mode. *

* Read Hardware Manual documentation to get more information.



- The Chip Select State tab contains the following parameters:

Names	Selection Types	Range	Description
Chip Select Inactive State *n* *n* if 0 *n* 7	Pop-up	Low High	



6.1.2.1.7 Block Dependency

None

6.1.2.1.8 Block Miscellaneous Details:

None

6.1.2.2 DSPI Receive Block

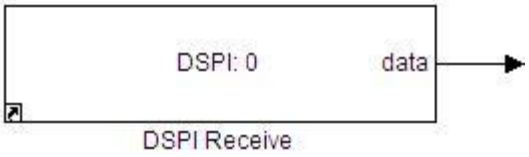
6.1.2.2.1 Block Name

DSPI Receive Block

6.1.2.2.2 Block Description

The main functionality of the block is reading data from a DSPI module.

6.1.2.2.3 Block Image



6.1.2.2.4 Inputs:

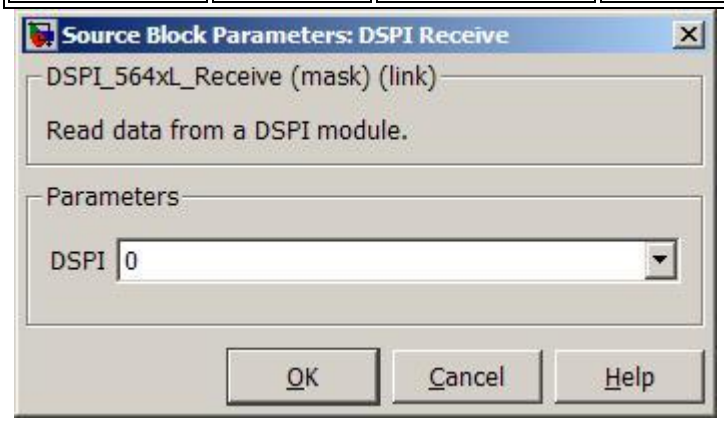
- None

6.1.2.2.5 Outputs:

- Data (uint16)

6.1.2.2.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
DSPI	Pop-up	0,1,2	DSPI Module



6.1.2.2.7 Block Dependency

Please do the following:

1. Configure DSPI

6.1.2.2.8 Block Miscellaneous Details:

None

6.1.2.3 DSPI Receive Trigger Block

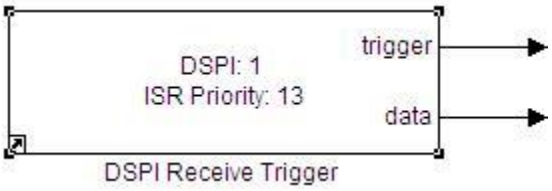
6.1.2.3.1 Block Name

DSPI Receive Trigger Block

6.1.2.3.2 Block Description

The main functionality of the block is to read data when it is available in the DSPI module.

6.1.2.3.3 Block Image



6.1.2.3.4 Inputs:

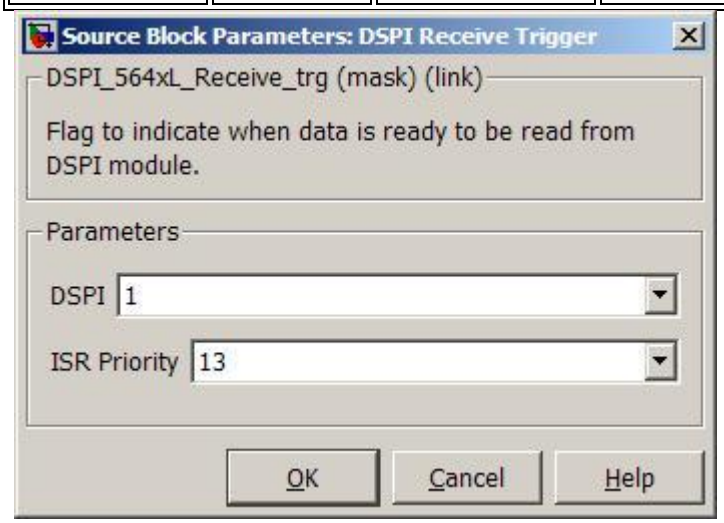
- None

6.1.2.3.5 Outputs:

- Function-call
- Data (uint16)

6.1.2.3.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
DSPI	Pop-up	0,1,2	DSPI Module
ISR Priority	Pop-up	0 – 15	Global ISR Priority



6.1.2.3.7 Block Dependency

Please do the following:

1. Configure DSPI

6.1.2.3.8 Block Miscellaneous Details:

None

6.1.2.4 DSPI Transmit Block

6.1.2.4.1 Block Name

DSPI Transmit Block

6.1.2.4.2 Block Description

This block transmits data through a DSPI module.

6.1.2.4.3 Block Image

6.1.2.4.4 Inputs:

- Data to be transmitted (uint16)
- EOQ (boolean) – available if EOQ is enabled

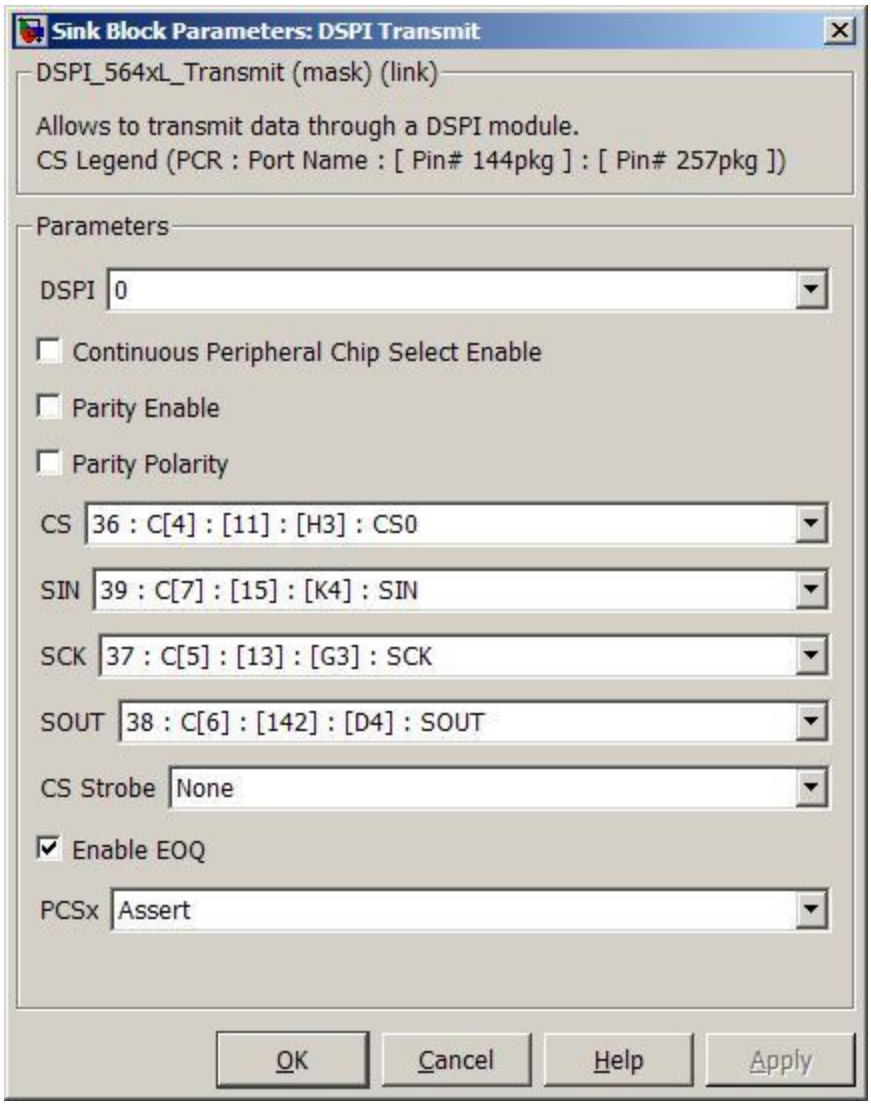
6.1.2.4.5 Outputs:

- None

6.1.2.4.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
DSPI	Pop-up	0,1,2	Select the Serial Communication Interface module which should be used for the transmit of the

			data
Continuous Peripheral Chip Select Enable	Check-box	Enable/Disable	This bit is used in SPI master mode. This bit enables the selected PCS signals to remain asserted between transfers
Parity Enable	Check-box	Enable/Disable	Parity Enable bit enables parity bit transmission and parity reception check for the SPI frame
Parity Polarity	Check-box	Enable/Disable	Parity Polarity bit controls polarity of the parity bit transmitted and checked.
CS	Pop-up	Pin selection	CS pin selection
SIN	Pop-up	Pin selection	SIN pin selection
SCK	Pop-up	Pin selection	SCK pin selection
SOUT	Pop-up	Pin selection	SOUT pin selection
CS Strobe	Pop-up	Pin selection	CS Strobe pin selection
Enable EOQ	Check-box	Enable/Disable	If set EOQ input will be available. The EOQ bit provides a means for host software to signal to the DSPI that the current SPI transfer is the last in a queue



6.1.2.4.7 Block Dependency

Please do the following:

1. Configure DSPI. If DSPI Config module is absent or it configure for slave mode, all unnecessary settings disabled.

6.1.2.4.8 Block Miscellaneous Details:

None

6.1.2.5 DSPI ISR Block

6.1.2.5.1 Block Name

DSPI ISR Block

6.1.2.5.2 Block Description

The main functionality of the block is to call user function when the event is occurring.

6.1.2.5.3 Block Image

6.1.2.5.4 Inputs:

- None

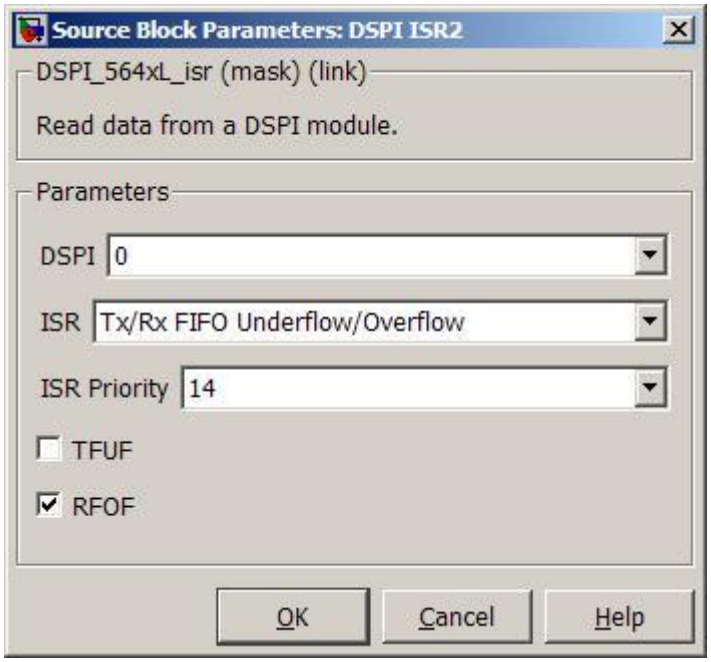
6.1.2.5.5 Outputs:

- Function-Call
- The DSPI_SR register value (uint32) – available if the ISR option is set to "Tx/Rx FIFO Underflow/Overflow"

6.1.2.5.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
DSPI	Pop-up	0,1,2	DSPI Module
ISR	Pop-up	Tx/Rx FIFO Underflow/Overflow EOQF – End of Queue TFFF – Transmit FIFO Full TCF – Transfer Complete RFDF – Rx FIFO Drain	Interrupt source *
ISR Priority	Pop-up	0 – 15	Global ISR Priority
TFUF	Check-box	Enable/Disable	Applicable if ISR is set to "Tx/Rx FIFO Underflow/Overflow"
RFOF	Check-box	Enable/Disable	Applicable if ISR is set to "Tx/Rx FIFO Underflow/Overflow"

* To get more information about DSPI ISR's, refer to Hardware Manual documentation.



6.1.2.5.7 Block Dependency

Please do the following:

1. Configure DSPI

6.1.2.5.8 Block Miscellaneous Details:

None

6.1.3 General Purpose Blocks

6.1.3.1 General Purpose Input

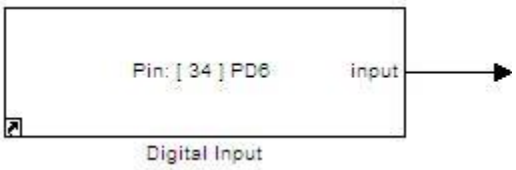
6.1.3.1.1 Block Name

Digital Input Block

6.1.3.1.2 Block Description

The main functionality of the block is to configure a single pin as a General Purpose Input.

6.1.3.1.3 Block Image



6.1.3.1.4 Inputs:

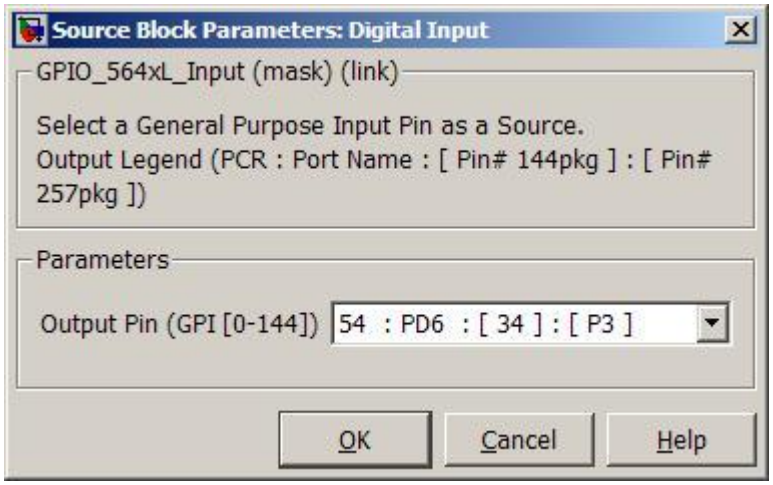
- None

6.1.3.1.5 Outputs:

- Input signal level (boolean)

6.1.3.1.6 Parameters:

- Input Pin selection



6.1.3.1.7 Block Dependency

None

6.1.3.1.8 Block Miscellaneous Details:

None

6.1.3.2 General Purpose Output

6.1.3.2.1 Block Name

Digital Output Block

6.1.3.2.2 Block Description

The main functionality of the block is to configure a single pin as a General Purpose output.

6.1.3.2.3 Block Image

6.1.3.2.4 Inputs:

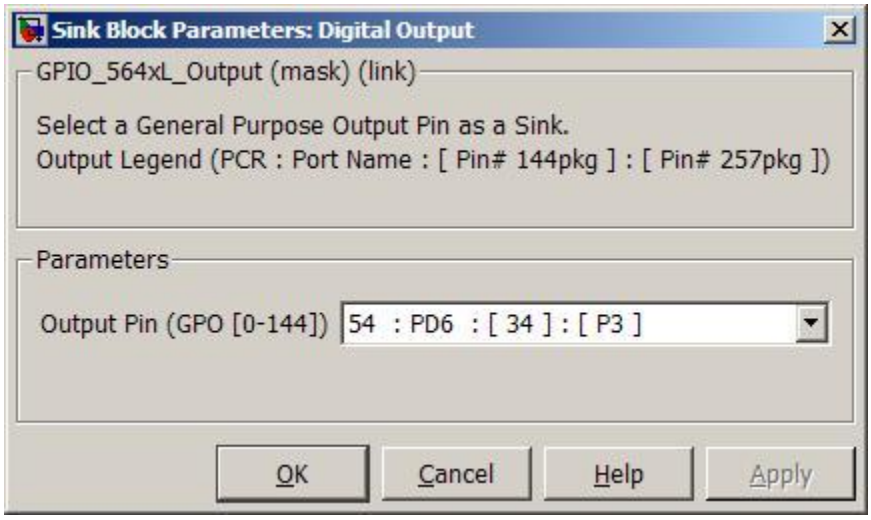
- Output signal level (boolean)

6.1.3.2.5 Outputs:

- None

6.1.3.2.6 Parameters:

- Output Pin selection



6.1.3.2.7 Block Dependency

None

6.1.3.2.8 Block Miscellaneous Details:

None

6.1.3.3 Periodic Interrupt Timer

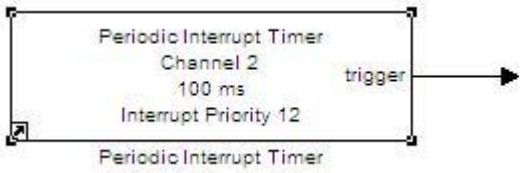
6.1.3.3.1 Block Name

Periodic Interrupt Timer (PIT) Block

6.1.3.3.2 Block Description

This block is used to trigger an interrupt routine periodically.

6.1.3.3.3 Block Image



6.1.3.3.4 Inputs:

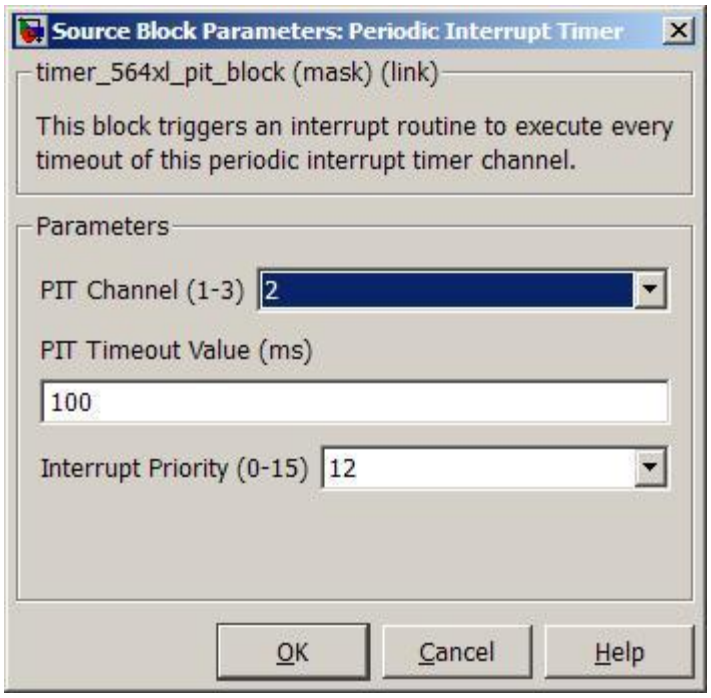
- None

6.1.3.3.5 Outputs:

- Function-call

6.1.3.3.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
PIT Channel	Pop-up	1 – 3	
PIT Timeout Value (ms)	Text-box	0.00000834 – 35791.3941	Timeout period for PIT interrupts
Interrupt Priority	Pop-up	0 – 15	Interrupt priority level



6.1.3.3.7 Block Dependency

None

6.1.3.3.8 Block Miscellaneous Details:

PIT channel 0 is used as system timer and can not be used for PIT block.

6.2 Motor Control Blocks

6.2.1 ADC

Analog to Digital Converter is a complex device that utilizes Direct Memory Access for processing at maximum speed with minimal CPU loading. The peripheral block for this assumes the user has configured the conversion command queues for the channels desired and the conversion trigger source type in RAppID_564xL_Config.

6.2.1.1 ADC Channel Block

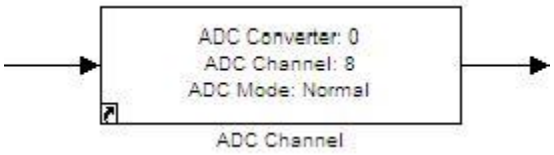
6.2.1.1.1 Block Name

ADC Channel Block

6.2.1.1.2 Block Description

Interface block that provides ability to read data sample from a particular ADC channel. On each update, sets the data outputs as read from ADC. If 'Enable for simulation' is set, than in simulation mode, sets the data outputs as input.

6.2.1.1.3 Block Image



6.2.1.1.4 Inputs:

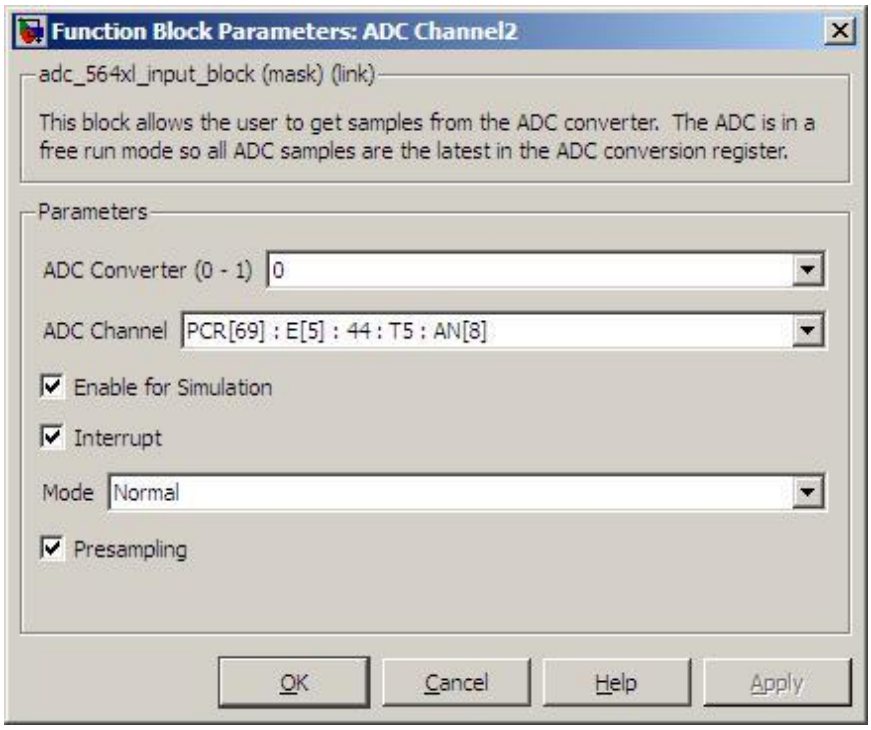
- Input available for simulation mode only.

6.2.1.1.5 Outputs:

- (uint16) 0x0000 to 0x0FFF

6.2.1.1.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
ADC converter number	Pop-up	0 – 1	Select the ADC converter number
ADC Channel	Pop-up	AN0 – AN15	Select the ADC channel number
Enable for Simulation	Check-box	On/Off	Input port available in simulation mode, on each update set output port equal to input.
Interrupt	Check-box	On/Off	Generate interrupt on the end of conversion. ADC ISR block required.
Mode	Pop-up	Normal or Injected	Mode of functionality. Normal – ADC channel works in SingleShot or FreeRun mode. Injected – ADC channel used only for injected data reading.



6.2.1.1.7 Block Dependency

Please do the following:

1. Configure the ADC with ADC Config block.

6.2.1.1.8 Block Miscellaneous Details:

None

6.2.1.2 ADC Configuration Block

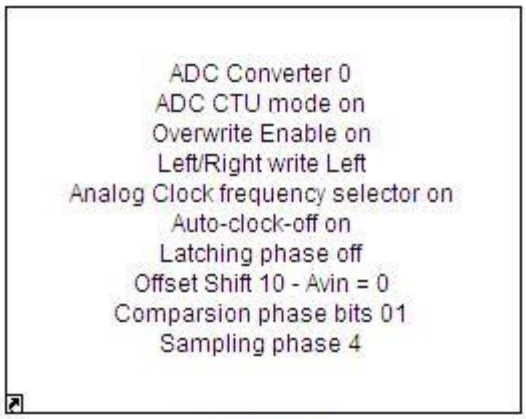
6.2.1.2.1 Block Name

ADC Configuration Block

6.2.1.2.2 Block Description

This block is used to configure the parameters of the ADC.

6.2.1.2.3 Block Image



6.2.1.2.4 Inputs:

- None

6.2.1.2.5 Outputs:

- None

6.2.1.2.6 Block Dialog and Parameters:

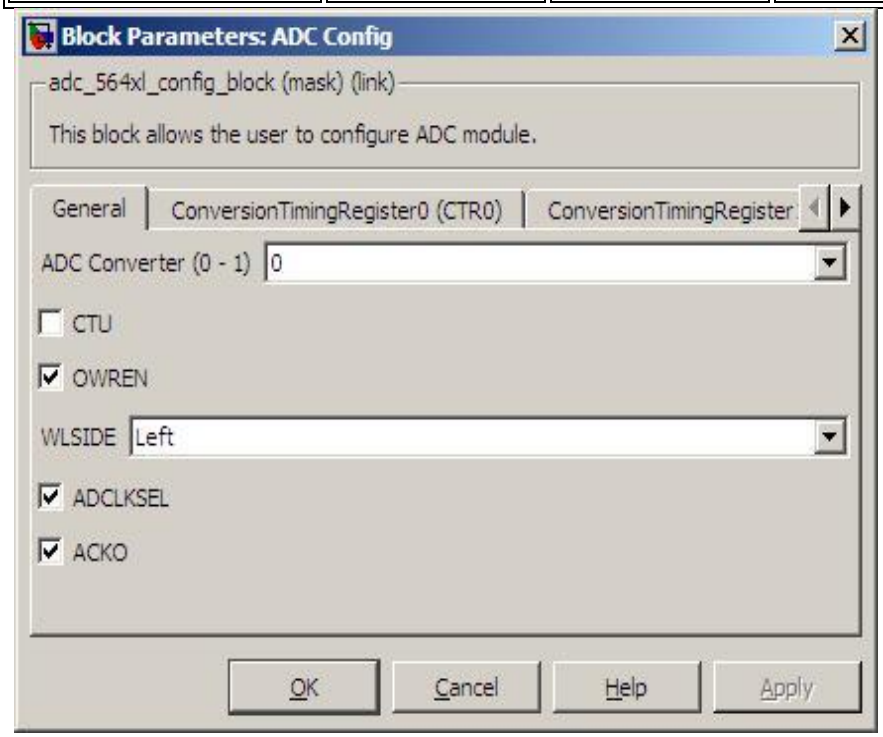
The block dialog consists of the following tabs:

- General
- ConversionTimingRegister0(CTR0)

- The General tab contains the following parameters:

Names	Selection Types	Range	Description
ADC converter number	Pop-up	0 – 1	Select the ADC converter number
CTU	Check-box	Enable/Disable	On/Off CTU mode. If CTU mode on, ADC controls by CTU module.
OWREN	Check-box	Enable/Disable	Overwrite enable. This check-box enables or disables the functionality to overwrite unread converted data.
WLSIDE	Pop-up	Left/Right	Write left/right-aligned

ADCLKSEL	Check-box	Enable/Disable	Analog clock frequency selector If this bit is set the AD_clk frequency is equal to the system clock frequency. Otherwise, it is half of the system clock frequency. This bit can be written in power-down mode only
ACKO	Check-box	Enable/Disable	Auto-clock-off enable

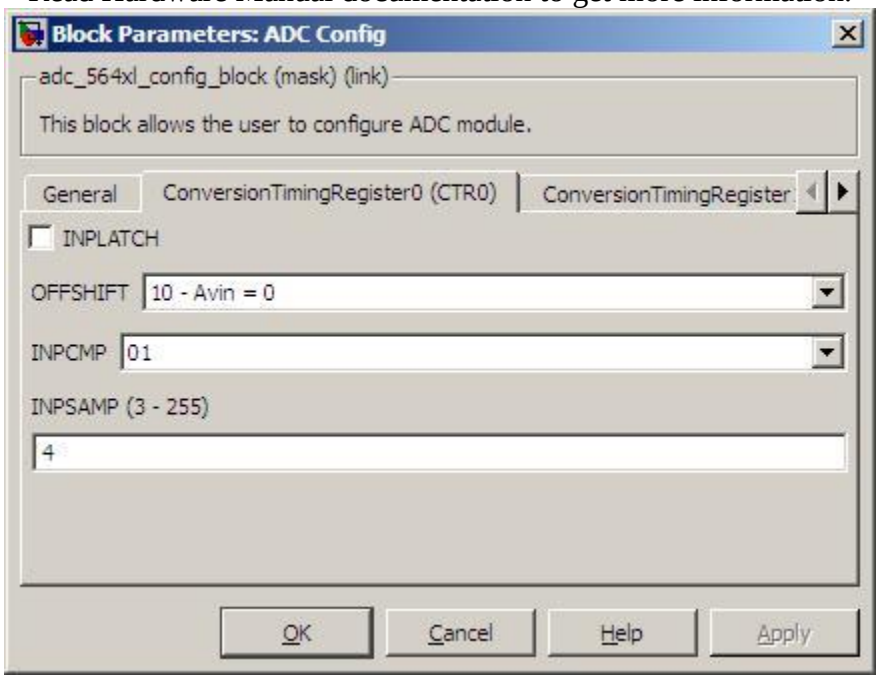


- The ConversionTimingRegister0(CTR0) contains the following parameters:

Names	Selection Types	Range	Description
INPLATCH	Check-box	On/Off	Configuration for latching phase duration. *
OFFSHIFT	Pop-up	00 – NoShift 01 – Avin = 1/2LSB 10 – Avin = 0	Configuration for offset shift characteristic. *
INPCMP	Pop-up	00, 01, 10, 11	Configuration bits for comparison phase duration. *
INPSAMP	Pop-up	0 – 10	Configuration bits for

			sampling phase duration. *

* Read Hardware Manual documentation to get more information.



6.2.1.2.7 Block Dependency

None

6.2.1.2.8 Block Miscellaneous Details:

None

6.2.1.3 ADC Interrupt

6.2.1.3.1 Block Name

ADC ISR Block

6.2.1.3.2 Block Description

This block allows call user function on ADC conversion event.

6.2.1.3.3 Block Image

6.2.1.3.4 Inputs:

- None

6.2.1.3.5 Outputs:

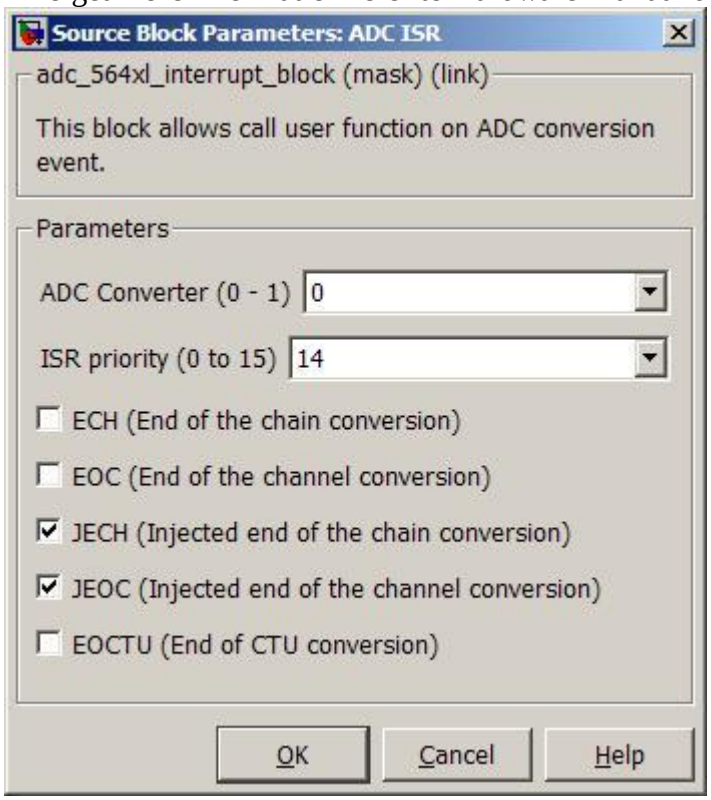
- Function-call
- ISR Mask (uint32). Value of the ADC ISR register.*

6.2.1.3.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
ADC converter number	Pop-up	0 – 1	Select the ADC converter number
ISR Priority	Pop-up	0 – 15	Global ISR priority
ECH (End of the chain conversion)	Check-box	Enable/Disable	Generate interrupt on End of the Chain Conversion. This interrupt is generated when all channels on this ADC Converter had been converted.
EOH (End of the channel conversion)	Check-box	Enable/Disable	Generate interrupt on End of the Chain Conversion ("Interrupt" should be enabled in corresponding ADC Channel block)
JECH (Injected end of the chain conversion)	Check-box	Enable/Disable	The same as for ECH in Injected

			mode.
JEOC (Injected end of the channel conversion)	Check-box	Enable/Disable	The same as for EOCH in Injected mode.
EOCTU (End of CTU conversion)	Check-box	Enable/Disable	ADC finished CTU command conversion.

* To get more information refer to Hardware Manual documentation.



6.2.1.3.7 Block Dependency

Please do the following:

1. Configure the ADC
2. Configure ADC Channel

Note:

1. "Interrupt" check-box in ADC Channel block should be enabled.

6.2.1.3.8 Block Miscellaneous Details:

None

6.2.1.4 ADC Trigger

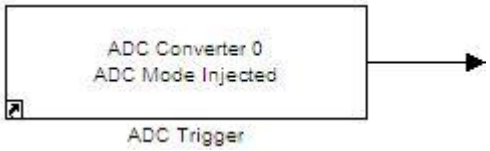
6.2.1.4.1 Block Name

ADC Trigger Block

6.2.1.4.2 Block Description

This block allows get measure in One Shot or Injected Mode. The ADC moved from Free Scan to One Shot/Injected mode get conversion and return it back.*

6.2.1.4.3 Block Image



6.2.1.4.4 Inputs:

- None

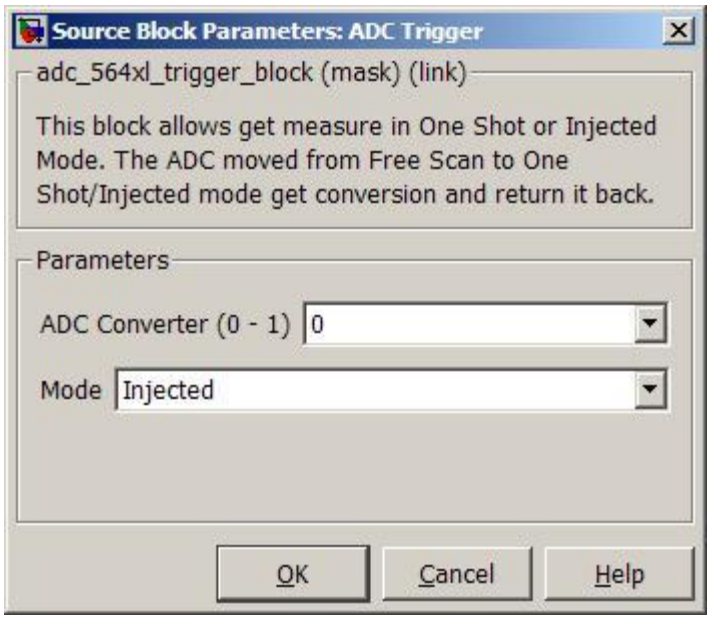
6.2.1.4.5 Outputs:

- Function-call

6.2.1.4.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
ADC converter number	Pop-up	0 – 1	ADC module selection.
Mode	Pop-up	Normal/Injected	Single Shot/Injected mode.

* To get more information about different modes of the ADC, refer to Hardware Manual documentation.



6.2.1.4.7 Block Dependency

Please do the following:

1. Configure the ADC converter.
2. Put inside user function ADC Channel block in the same mode.

Note:

1. ADC trigger Block and ADC Channel inside the user function should be set in one mode.

6.2.1.4.8 Block Miscellaneous Details:

None

6.2.1.5 ADC Watchdog Threshold

6.2.1.5.1 Block Name

ADC Watchdog Threshold Block

6.2.1.5.2 Block Description

This block allow continue hardware monitoring. When watchdog is set it monitors ADC channel and call user function if get value more/less than threshold value.

6.2.1.5.3 Block Image

6.2.1.5.4 Inputs:

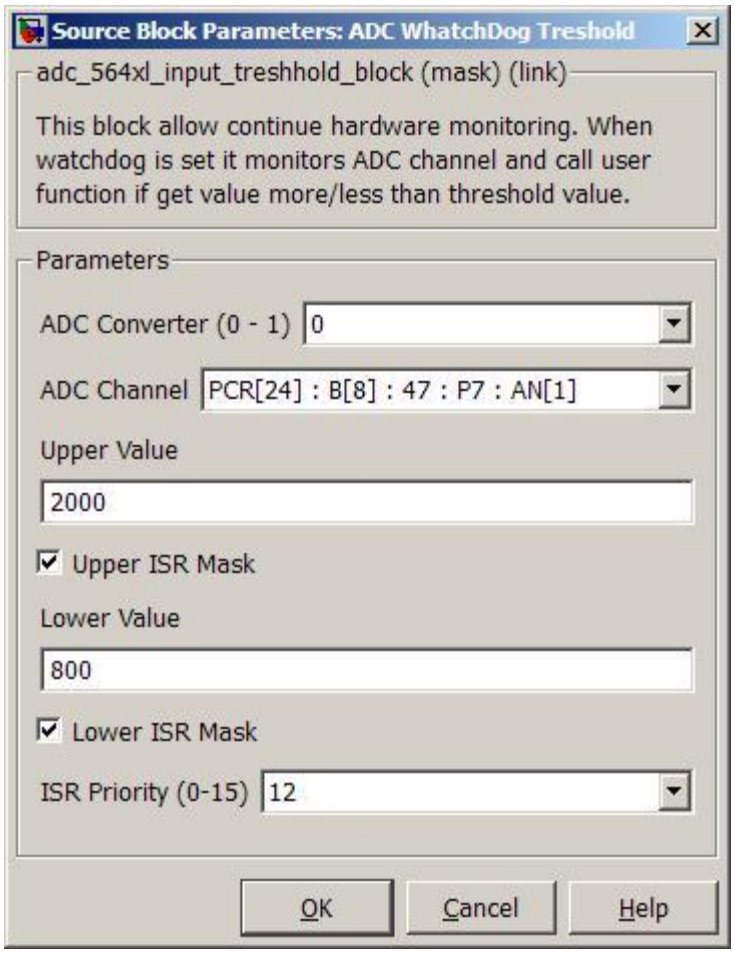
- None

6.2.1.5.5 Outputs:

- Function-call

6.2.1.5.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
ADC converter number	Pop-up	0 – 1	Select the ADC converter number
ADC Channel	Pop-up	AN0 – AN15	Select the ADC channel number
Upper Value	Text-box	0 – 4095	Threshold upper value
ISR Mask	Check-box	On/Off	Enable/Disable ISR on Upper value
Lower Value	Text-box	0 – 4095	Threshold lower value
ISR Mask	Check-box	On/Off	Enable/Disable ISR on Lower value
ISR Priority	Pop-up	0 – 15	Global ISR priority



6.2.1.5.7 Block Dependency

Please do the following:

1. Configure the ADC with ADC Config block.

6.2.1.5.8 Block Miscellaneous Details:

None

6.2.2 CTU

The cross triggering unit (CTU) is intended to completely avoid CPU involvement in the time acquisitions of state variables during the control cycle that can be the PWM cycle, the half PWM cycle or a number of PWM cycles. In such case the pre-setting of the acquisition times needs to be completed during the previous control cycle, where the actual acquisitions are to be made, and a double-buffered structure for the CTU registers is used, in order to activate the new settings at the beginning of the next control cycle. In addition, there are 4 FIFOs inside the CTU available to store the ADC results.

6.2.2.1 CTU Configuration

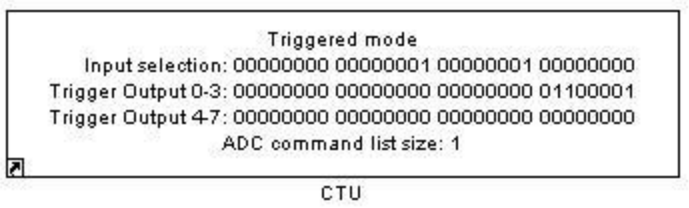
6.2.2.1.1 Block Name

CTU Configuration Block

6.2.2.1.2 Block Description

The main functionality of the block is to configure CTU mode and its interfaces with other peripherals.

6.2.2.1.3 Block Image



6.2.2.1.4 Inputs:

- None

6.2.2.1.5 Outputs:

- None

6.2.2.1.6 Block Dialog and Parameters:

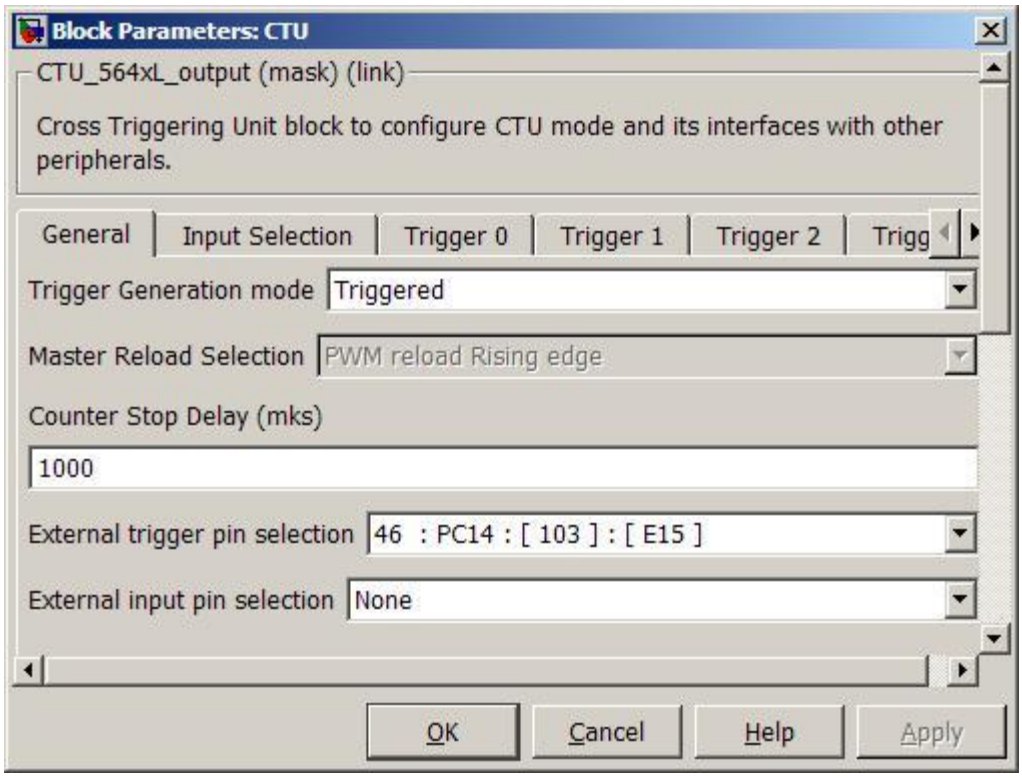
The block dialog consists of the following tabs:

- General
- Input Selection
- Trigger (0-7)
- ADC Commands

- The General tab contains the following parameters:

Names	Selection Types	Range	Description
Trigger Generation mode	Pop-up	Triggered Sequential	Trigger Generation mode
Master Reload Selection	Pop-up	PWM reload Rising edge PWM reload Falling edge	*N* is 0 – 3, specifies

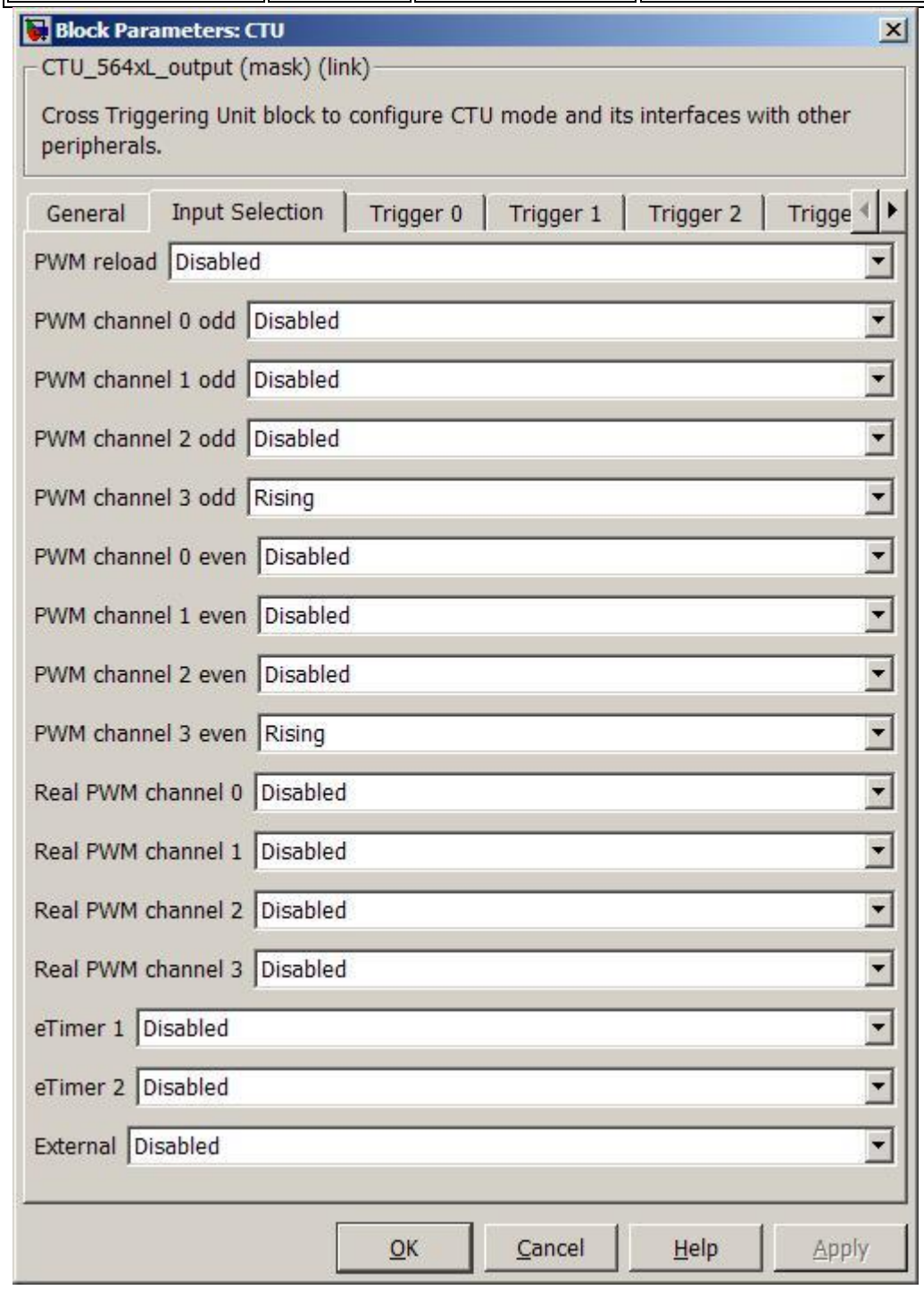
		PWM channel *N* odd Rising edge PWM channel *N* odd Falling edge PWM channel *N* even Rising edge PWM channel *N* even Falling edge Real PWM channel *M* Rising edge Real PWM channel *M* Falling edge eTimer *T* Rising edge eTimer *T* Falling edge External signal Rising edge External signal Falling edge	the PWM channel number *M* is 0 – 3, specifies the Real PWM channel number *T* is 1 – 2, specifies the eTimer
Counter Stop Delay (mks)	Text-box		Defines period when TGS Counter loaded with TGS Counter Reload Value reaches TSG Counter Compare Value
External trigger pin selection	Pop-up	The list of available pins depends on the selected module.	External trigger pin
External input pin selection	Pop-up	The list of available pins depends on the selected module.	External input pin



- The Input Selection tab contains the following parameters:

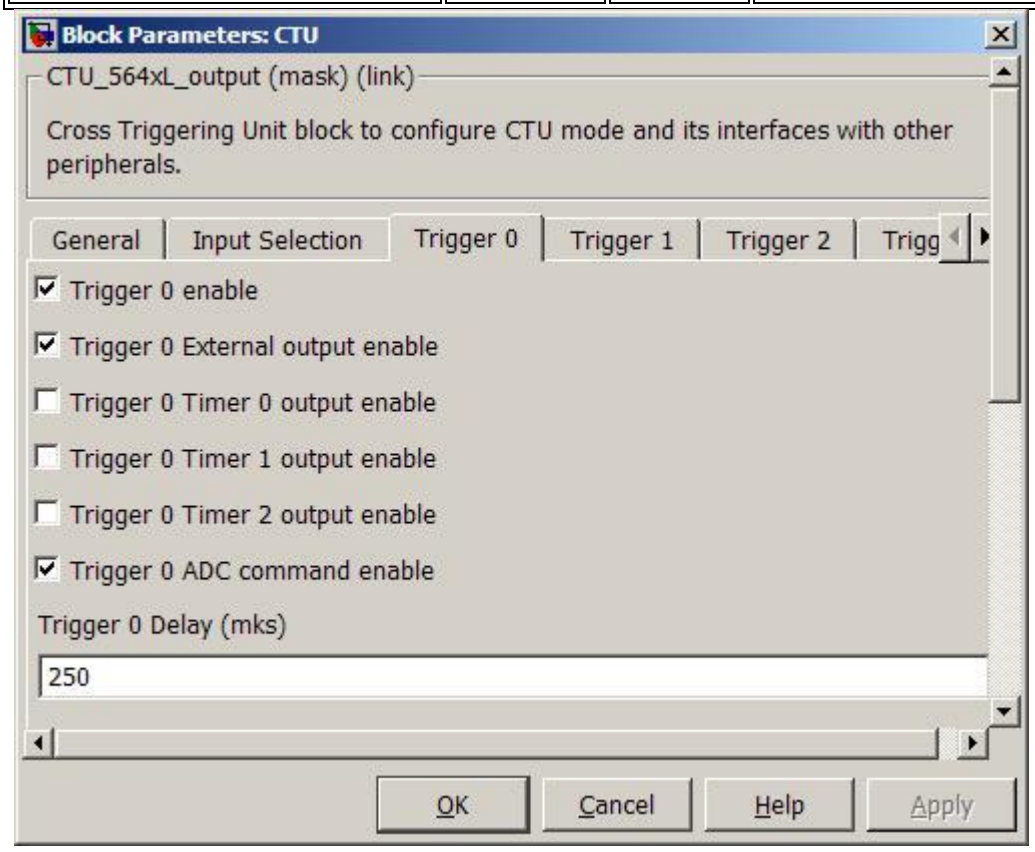
Names	Selection Types	Range	Description
PWM reload	Pop-up	Disabled Rising Falling Rising and Falling	I0_FE, I0_RE bits of TGSISR
PWM channel *n* odd *n* is 0 – 3	Pop-up	Disabled Rising Falling Rising and Falling	I*(n+1)*_FE, I*(n+1)*_RE bits of TGSISR
PWM channel *n* even *n* is 0 – 3	Pop-up	Disabled Rising Falling Rising and Falling	I*(n+5)*_FE, I*(n+5)*_RE bits of TGSISR
Real PWM channel *n* *n* is 0 – 3	Pop-up	Disabled Rising Falling Rising and Falling	I*(n+9)*_FE, I*(n+9)*_RE bits of TGSISR
eTimer *n*	Pop-up	Disabled	I*(n+12)*_FE, I*(n+12)*_RE bits of

n is 1 – 2		Rising Falling Rising and Falling	TGSISR
External	Pop-up	Disabled Rising Falling Rising and Falling	I15_FE, I15_RE bits of TGSISR



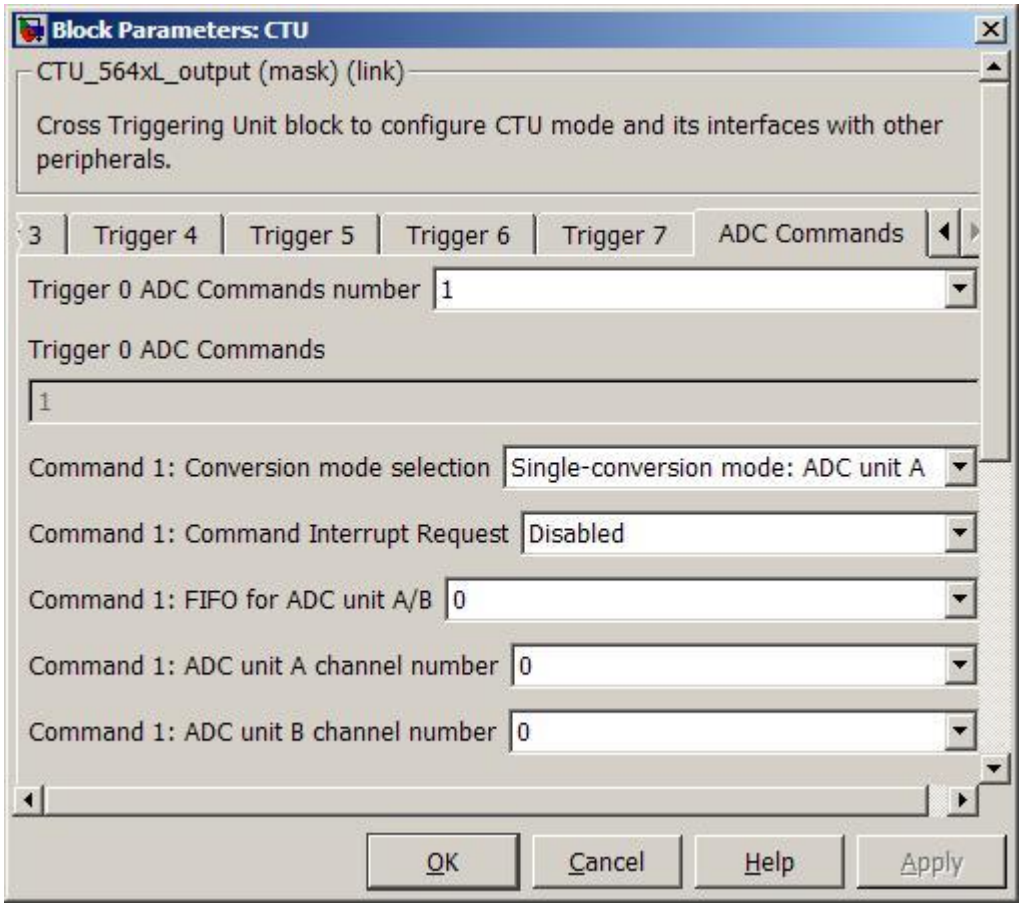
- The Trigger *N* tabs (*N* is 0 – 7) contain the following parameters:

Names	Selection Types	Range	Description
Trigger *n* enable *n* is trigger number	Check-box	Enable Disable	Respective T*n*_E value of THCR1 and THCR2 registers
Trigger *n* External output enable *n* is trigger number	Check-box	Enable Disable	Respective T*n*_ETE value of THCR1 and THCR2 registers
Trigger *n* Timer *m* output enable *n* is trigger number *m* is 0 – 2	Check-box	Enable Disable	Respective T*n*_T*m*E value of THCR1 and THCR2 registers
Trigger *n* ADC command enable *n* is trigger number	Check-box	Enable Disable	Respective T*n*_ADCE value of THCR1 and THCR2 registers
Trigger *n* Delay (mks) *n* is trigger number	Text-box		Defines period when TGS Counter loaded with TGS Counter Reload Value reaches Trigger *n* Compare Value



- The ADC Commands tab contains the following parameters:

Names	Selection Types	Range	Description
Trigger *n* ADC Commands number (for each Trigger *n* where ADC Command is enabled)	Pop-up	0 – 24	Total number of ADC Commands for all Triggers is 24
Command *m*: Conversion mode selection	Pop-up	– Single-conversion mode: ADC unit A – Single-conversion mode: ADC unit B – Dual-conversion mode	for each Command *m* of Trigger *n*
Command *m*: Command Interrupt Request	Pop-up	Enabled Disabled	for each Command *m* of Trigger *n*
Command *m*: FIFO for ADC unit A/B	Pop-up	0 – 7	for each Command *m* of Trigger *n*
Command *m*: ADC unit A channel number	Pop-up	0 – 15	for each Command *m* of Trigger *n*
Command *m*: ADC unit B channel number	Pop-up	0 – 15	for each Command *m* of Trigger *n*



6.2.2.1.7 Block Dependency

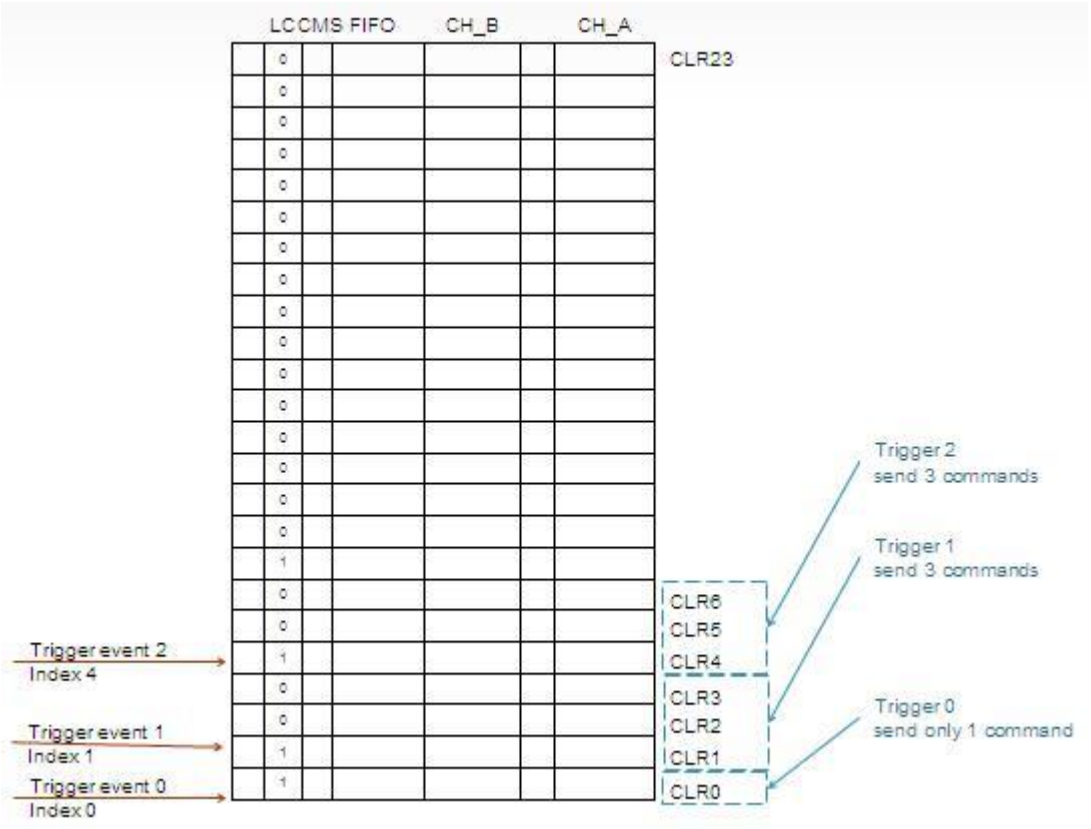
None

6.2.2.1.8 Block Miscellaneous Details:

ATTENTION: The CTU model block has been updated such that a dummy command is no longer required. Please immediately remove any dummy commands that was inserted so that your command list is compatible with the updated block.

1. If LC bit is set in the trigger index command - The command will be executed upon the trigger. This is not recognized as the last command. So, next commands in the command list are executed till a command with LC bit set is found.
2. The command with LC bit set, if not a trigger index command, will not get executed.
    a. a. In a case where there is just one trigger enabled, and single ADC channel conversion required on the trigger - we need a dummy command with LC bit set after the required ADC command.
    b. b. In a case where there are two triggers, one command on first trigger, two commands on the second trigger - we need total four commands(one dummy command at the last with LC bit set). First trigger's commands LC bit may or may not be set. Second trigger's first command LC bit has to be set so that first trigger commands list will stop at this point. Second

trigger's second command will not have LC bit set. Second trigger's third command(last command in the commands list and the dummy command) will have LC bit set. This command will not be executed. If the second trigger's first command LC bit is not set - Three commands will be executed on first trigger and stops at the dummy command with LC bit set. On the second trigger two commands will be executed and stops at the last dummy command whose LC bit is set.



6.2.2.2 CTU ISR

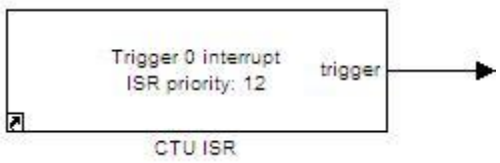
6.2.2.2.1 Block Name

CTU ISR Block

6.2.2.2.2 Block Description

The main functionality of the block is to process CTU ISRs

6.2.2.2.3 Block Image



6.2.2.2.4 Inputs:

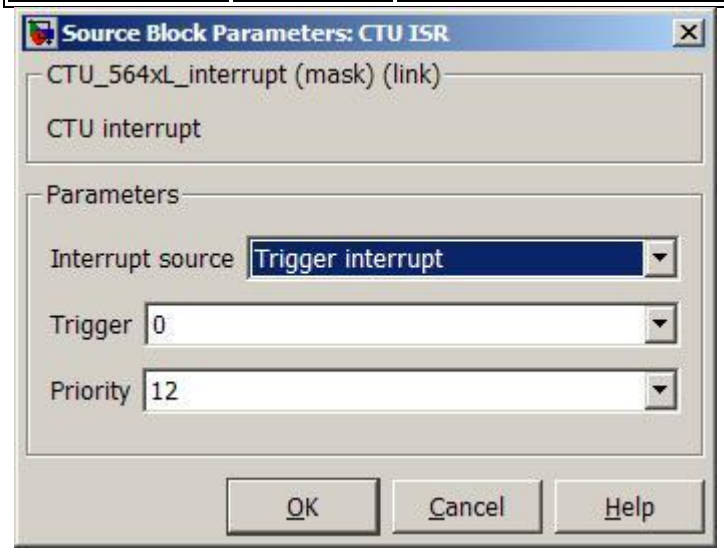
- None

6.2.2.2.5 Outputs:

- Function-call
- CTUEFR value (uint16) (if "Interrupt source" is set to "Error interrupt")

6.2.2.2.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
Interrupt Source	Pop-up	Trigger interrupt MRS interrupt Error interrupt ADC command interrupt	
Trigger	Pop-up	0 – 7	Trigger number causes interrupt request (applicable if "Interrupt source" is set to "Trigger interrupt" only)
Priority	Pop-up	0 – 15	Interrupt priority level



6.2.2.2.7 Block Dependency

None

6.2.2.2.8 Block Miscellaneous Details:

None

6.2.2.3 CTU ISR Enable

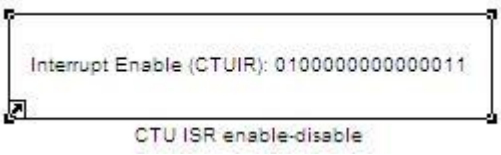
6.2.2.3.1 Block Name

CTU Interrupt Enable/Disable Block

6.2.2.3.2 Block Description

The main functionality of the block is to enable/disable CTU interrupts

6.2.2.3.3 Block Image



6.2.2.3.4 Inputs:

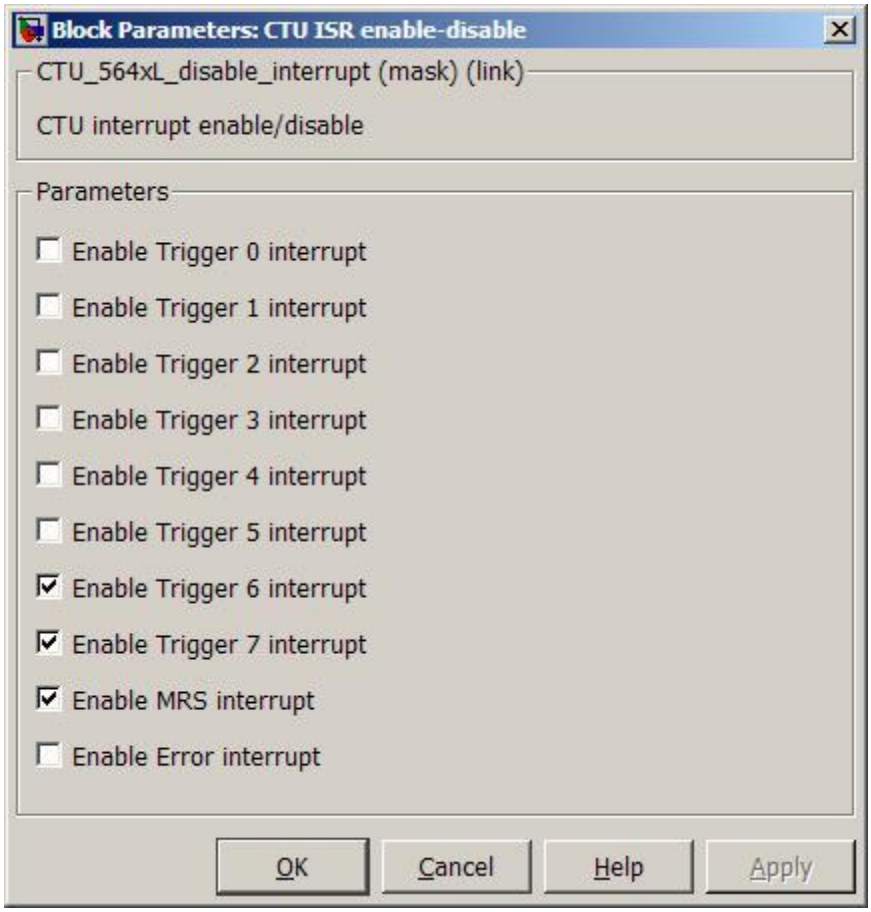
- None

6.2.2.3.5 Outputs:

- None

6.2.2.3.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
Enable Trigger *[n]* Interrupt	Check-box	On/Off	Enable/Disable Trigger *[n]* Interrupt (*n* is 0 – 7)
Enable MRS interrupt	Check-box	On/Off	Enable/Disable MRS interrupt
Enable Error interrupt	Check-box	On/Off	Enable/Disable Error interrupt



6.2.2.3.7 Block Dependency

Please do the following:

1. Configure CTU and its interrupts via CTU Configuration and CTU ISR Blocks

6.2.2.3.8 Block Miscellaneous Details:

None

6.2.3 PWM

Enhanced Modular Input Output System is a timer based system for generation and capture of time based waveform and waveform information.

6.2.3.1 Complementary PWM Output

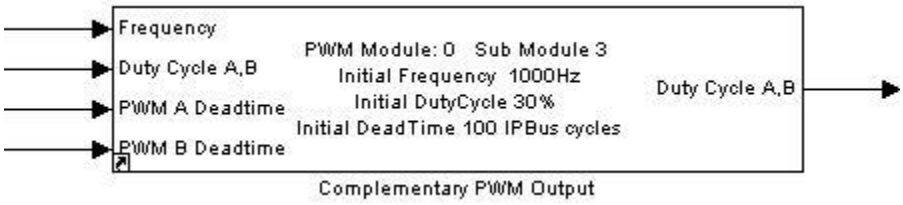
6.2.3.1.1 Block Name

Complementary FlexPWM Output Block

6.2.3.1.2 Block Description

The main functionality of the block is to generate a complementary PWM signals on A and B outputs of the selected FlexPWM module.

6.2.3.1.3 Block Image



6.2.3.1.4 Inputs:

- Frequency (uint32)
- Duty Cycle A,B (uint32)
- PWMA Deadtime (uint32)
- PWMB Deadtime (uint32)
- Duty Cycle A,B (uint32) - if "PWM45 for Output Trigger" is On

6.2.3.1.5 Outputs:

- Duty Cycle (double) - if "Duty Cycle Simulation Output" is On

6.2.3.1.6 Block Dialog and Parameters:

The block dialog consists of the following tabs:

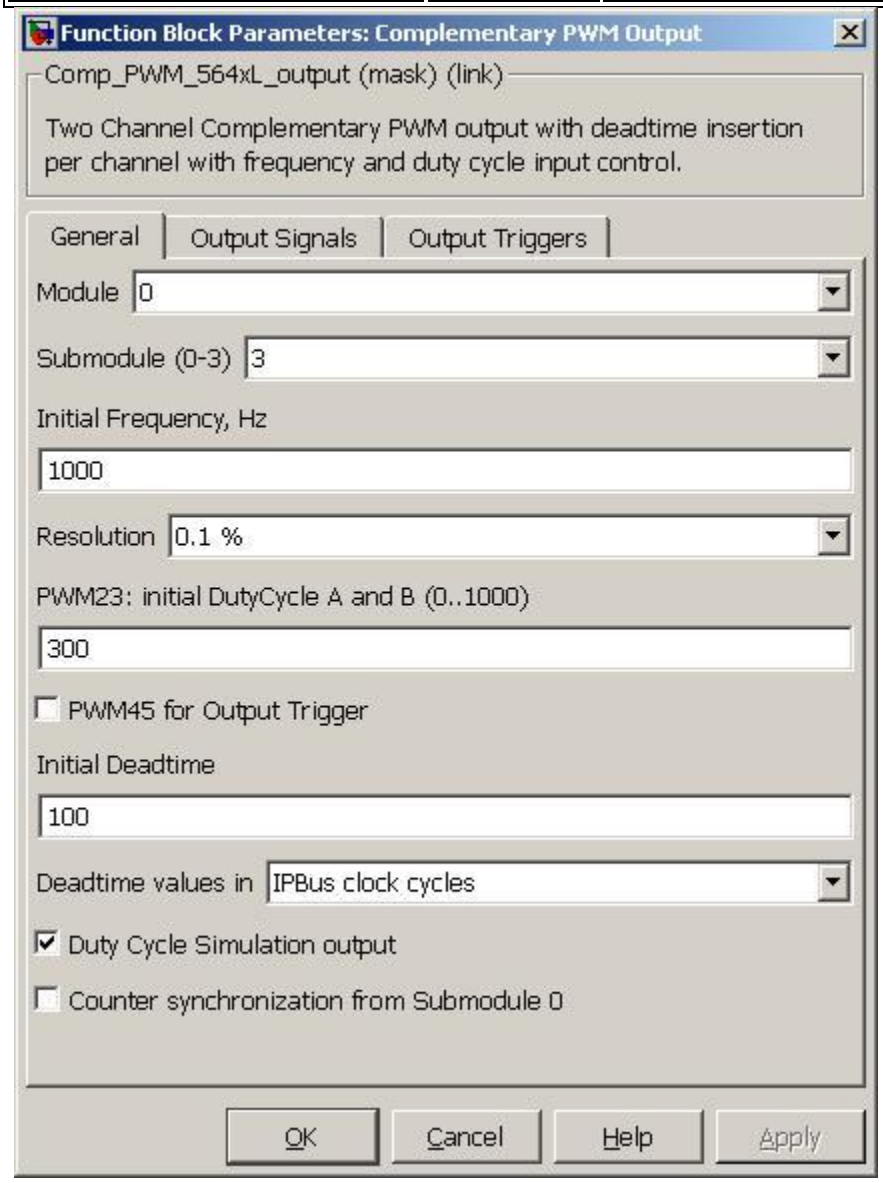
- General
- Output Signals
- Output Triggers

- The General tab contains the following parameters:

Names	Selection Types	Range	Description
Module	Pop-up	0 – 1	FlexPWM module
Submodule (0-3)	Pop-up	0 – 3	FlexPWM channel
Initial Frequency Hz	Text-box	Depends on Motor Control Clock value	Initial frequency of PWM output signals

Resolution	Pop-up	1 % 0.1 % 0.01 % 0.001 %	Duty cycle resolution: actual Duty cycle is equal to Duty cycle parameter or input value multiplied by resolution
PWM23: initial DutyCycle A and B	Text-box	0 – 100/resolution	Initial Duty Cycle of channel A and B output signals
PWM45 for Output Trigger	Check-box	On/Off	If VAL4 and VAL5 is used for Output Trigger generation
PWM45: initial DutyCycle A and B	Text-box	0 – 100/resolution	Initial Duty Cycle for PWM45 (not applicable if "PWM45 for Output Trigger" is Off)
Initial Deadtime	Text-box	0 – 4095 IPBus clock cycles	Deadtime during 0 to 1 transitions of the PWM channels A and B output
Initial Deadtime values in	Pop-up	– IPBus clock cycles – microseconds	Deadtime values unit
Duty Cycle Simulation Output	Check-box	On/Off	If DutyCycle simulation output is used
Counter synchronization from Submodule 0	Check-box	On/Off	If Master Sync is used for counter synchronization (not applicable for Submodule

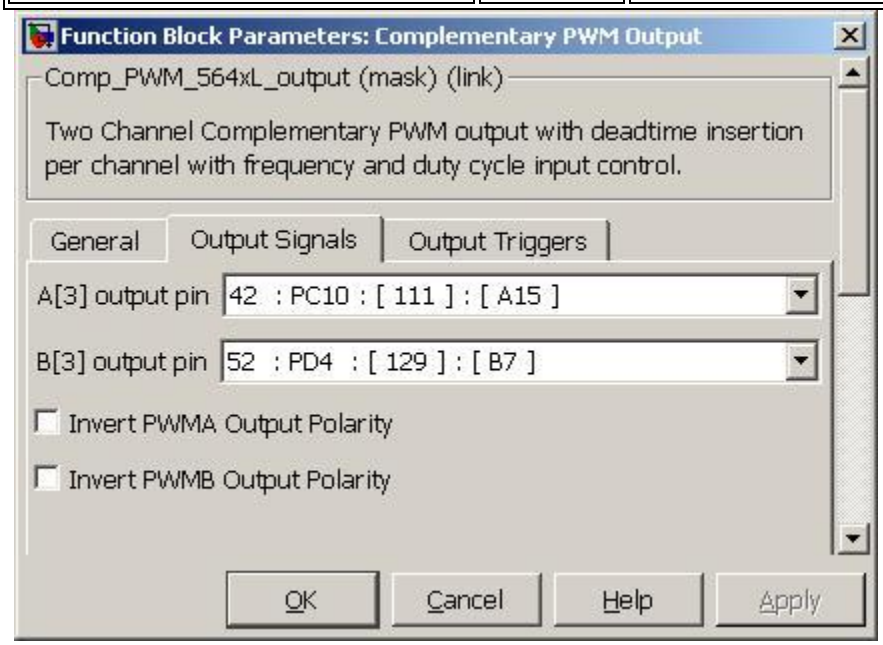
			0)



- The Output Signals tab contains the following parameters:

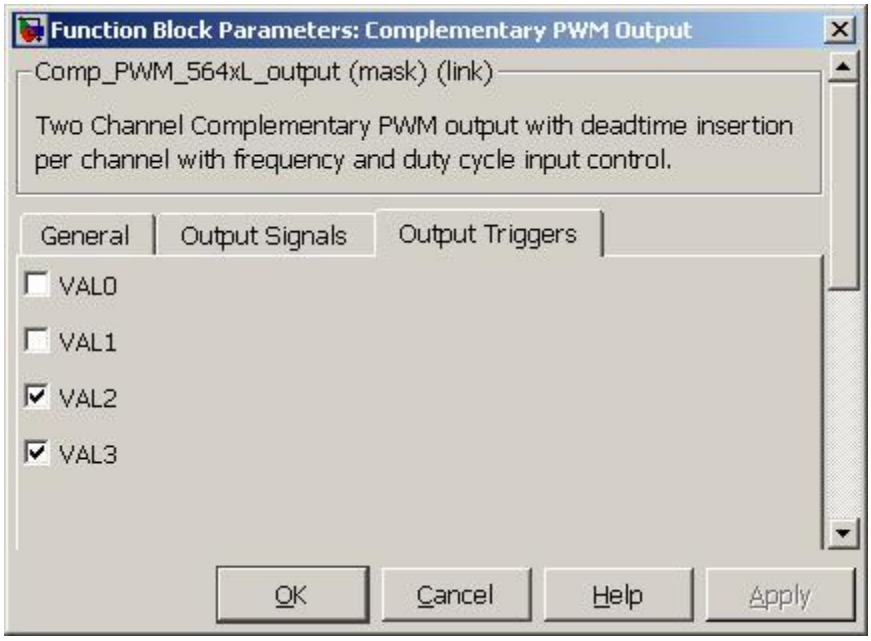
Names	Selection Types	Range	Description
A[n] output pin (where n is the number of the selected submodule)	Pop-up	The list of available pins depends on the selected module.	Channel A pin selection
B[n] output pin (where n is the number of the selected submodule)	Pop-up	The list of available pins depends on the selected module.	Channel B pin selection

Invert PWMA Output Polarity	Check-box	On/Off	
Invert PWMB Output Polarity	Check-box	On/Off	



- The Output Triggers tab contains the following parameters:

Names	Selection Types	Range	Description
VAL*n* *n* is 0 – 5 if "PWM45 for Output Trigger" is On *n* is 0 – 3 if "PWM45 for Output Trigger" is Off	Check-box	On/Off	Enable generation of Output Trigger signal based on the counter value matching VAL*n* register value



6.2.3.1.7 Block Dependency

None

6.2.3.1.8 Block Miscellaneous Details:

Duty Cycle output signal needed for simulation purpose only.

6.2.3.2 FlexPWM ISR

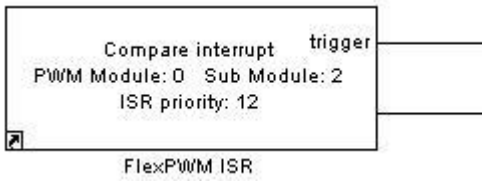
6.2.3.2.1 Block Name

FlexPWM ISR Block

6.2.3.2.2 Block Description

The main functionality of the block is to process FlexPWM ISRs

6.2.3.2.3 Block Image



6.2.3.2.4 Inputs:

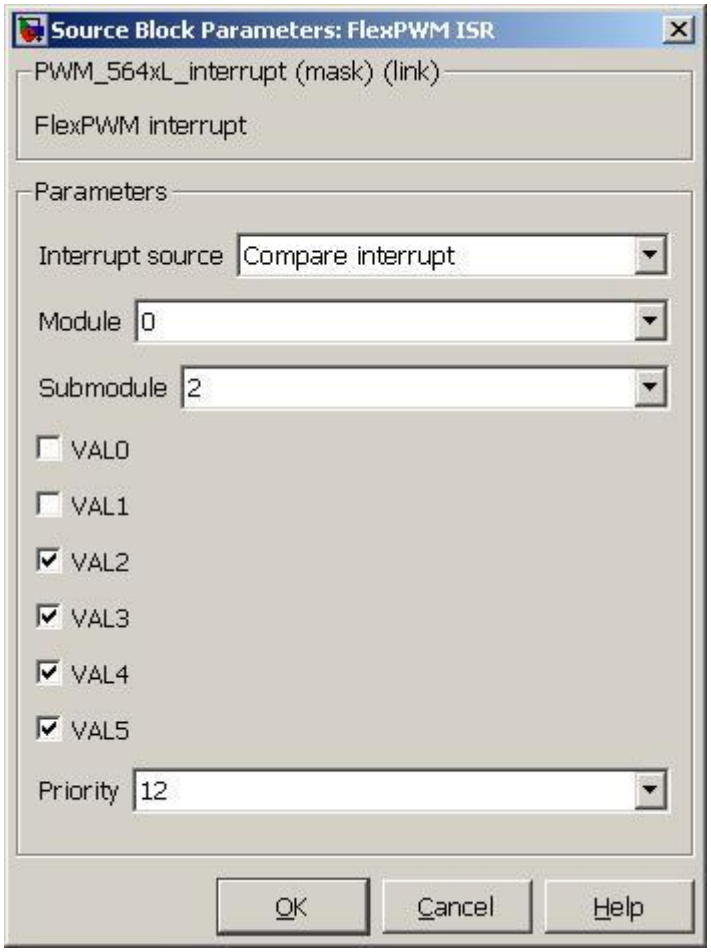
- None

6.2.3.2.5 Outputs:

- Function-call
- Status register (STS) value (uint16) (if "Interrupt source" is set to "Compare interrupt")

6.2.4 Block Dialog and Parameters:

Names	Selection Types	Range	Description
Interrupt Source	Pop-up	Compare interrupt Reload interrupt Reload Error interrupt	
Module	Pop-up	0 – 1	FlexPWM module
Submodule	Pop-up	0 – 3	FlexPWM channel
VAL*x* *x* is 0 – 5	Check-box	Enable/Disable	If counter match to respective VAL*x* register causes interrupt request. applicable for Compare interrupt only
Priority	Pop-up	0 – 15	Interrupt priority level



6.2.4.1.1 Block Dependency

Please do the following:

1. Configure respective FlexPWM submodule via Simple PWM Output, Complementary PWM Output or Three-phase PWM Output Block

6.2.4.1.2 Block Miscellaneous Details:

None

6.2.4.2 PWM ISR Enable

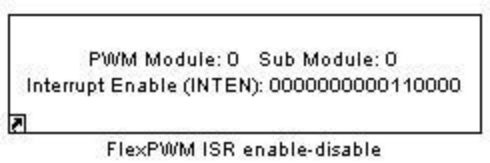
6.2.4.2.1 Block Name

FlexPWM Interrupt Enable/Disable Block

6.2.4.2.2 Block Description

The main functionality of the block is to enable/disable FlexPWM interrupts

6.2.4.2.3 Block Image



FlexPWM ISR enable-disable

6.2.4.2.4 Inputs:

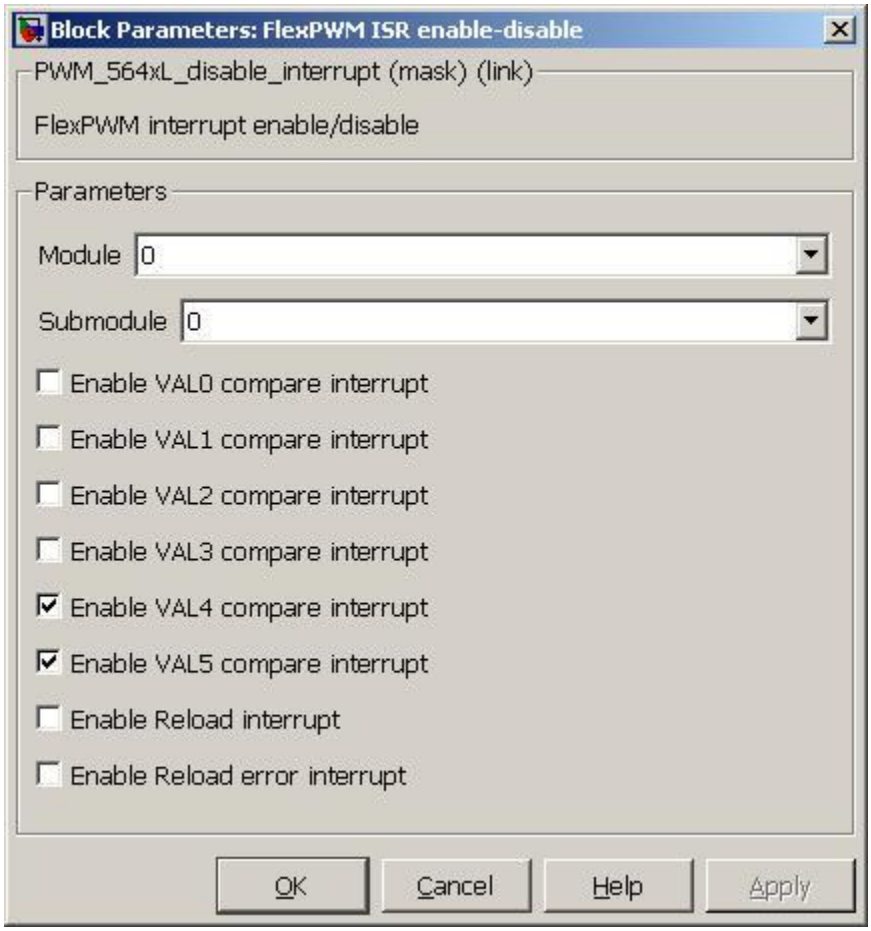
- None

6.2.4.2.5 Outputs:

- None

6.2.4.2.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
Module	Pop-up	0 – 1	FlexPWM module
Submodule	Pop-up	0 – 3	FlexPWM channel
Enable VAL*x* compare interrupt	Check-box	On/Off	Enable/Disable VAL*x* compare interrupt (*x* is 0 – 5)
Enable Reload compare interrupt	Check-box	On/Off	Enable/Disable Reload interrupt
Enable Reload Error compare interrupt	Check-box	On/Off	Enable/Disable Reload Error interrupt



6.2.4.2.7 Block Dependency

Please do the following:

1. Configure respective FlexPWM submodule via Simple PWM Output, Complementary PWM Output or Three-phase PWM Output Block
2. Configure related FlexPWM interrupts via FlexPWM ISR Block

6.2.4.2.8 Block Miscellaneous Details:

None

6.2.4.3 Simple PWM Output

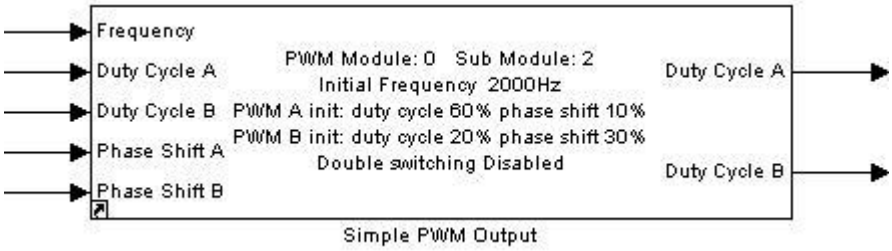
6.2.4.3.1 Block Name

Simple FlexPWM Output Block

6.2.4.3.2 Block Description

The main functionality of the block is to generate a simple center-aligned, edge-aligned, phase-shifted or double-switching PWM output signals on A and B outputs of the selected FlexPWM module. Plase see Block Miscellaneous Details for other information on using this block.

6.2.4.3.3 Block Image



6.2.4.3.4 Inputs:

- Frequency (uint32)
- Duty Cycle A (uint32)
- Duty Cycle B (uint32)
- Phase Shift A (uint32) - if "PWM signals" is "A - edge aligned B - phase shifted" or "A B - phase shifted"
- Phase Shift B (uint32) - if "PWM signals" is "A B - phase shifted"

6.2.4.3.5 Outputs:

- Duty Cycle A (double) - if "Duty Cycle Simulation Output" is On
- Duty Cycle B (double) - if "Duty Cycle Simulation Output" is On

6.2.4.3.6 Block Dialog and Parameters:

The block dialog consists of the following tabs:

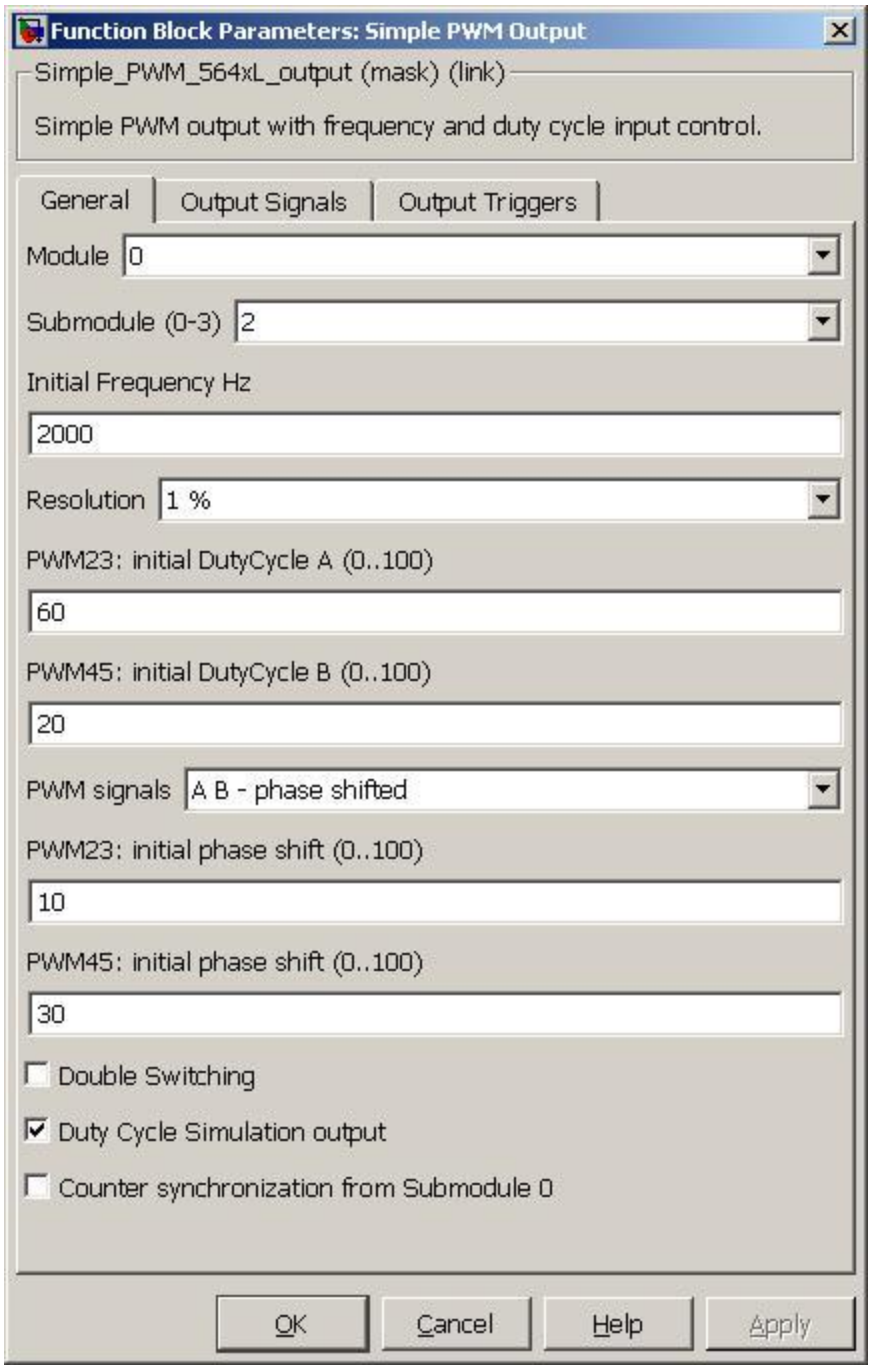
- General
- Output Signals
- Output Triggers

- The General tab contains the following parameters:

Names	Selection Types	Range	Description
Module	Pop-up	0 – 1	FlexPWM module
Submodule (0-3)	Pop-up	0 – 3	FlexPWM channel

Initial Frequency Hz	Text-box	Depends on Motor Control Clock value	Initial frequency of PWM output signals
Resolution	Pop-up	1 % 0.1 % 0.01 % 0.001 %	Duty cycle resolution: actual Duty cycle is equal to Duty cycle parameter or input value multiplied by resolution
PWM23: initial DutyCycle A	Text-box	0 – 100/resolution	Initial Duty Cycle of channel A output signals
PWM45: initial DutyCycle B	Text-box	0 – 100/resolution	Initial Duty Cycle of channel B output signals
PWM signals	Pop-up	A - edge aligned B - phase shifted A B - center aligned A B - edge aligned A B - phase shifted	
PWM23: initial phase shift A	Text-box	0 – 100/resolution	Initial Phase Shift of channels A output signals (applicable if "PWM signals" is "A - edge aligned B - phase shifted" or "A B - phase shifted" only)
PWM45: initial phase shift B	Text-box	0 – 100/resolution	Initial Phase Shift of channels B output signals (applicable if "PWM signals"

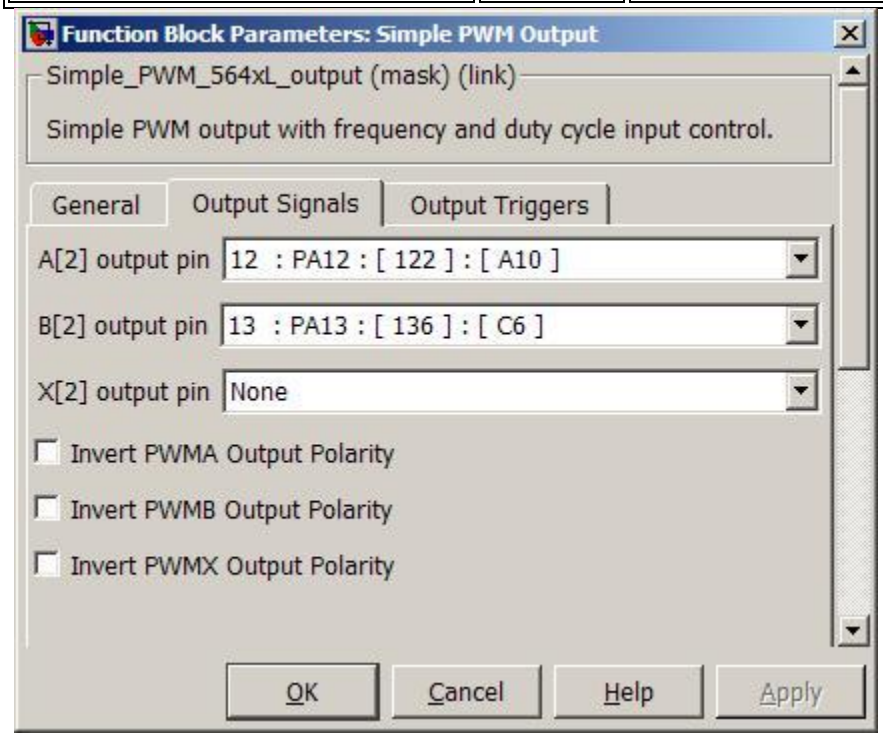
			is "A B - phase shifted" only)
Double Switching	Check-box	On/Off	Double switching PWM behavior
Duty Cycle Simulation Output	Check-box	On/Off	If DutyCycle simulation output is used
Counter synchronization from Submodule 0	Check-box	On/Off	If Master Sync is used for counter synchronization (not applicable for Submodule 0)



- The Output Signals tab contains the following parameters:

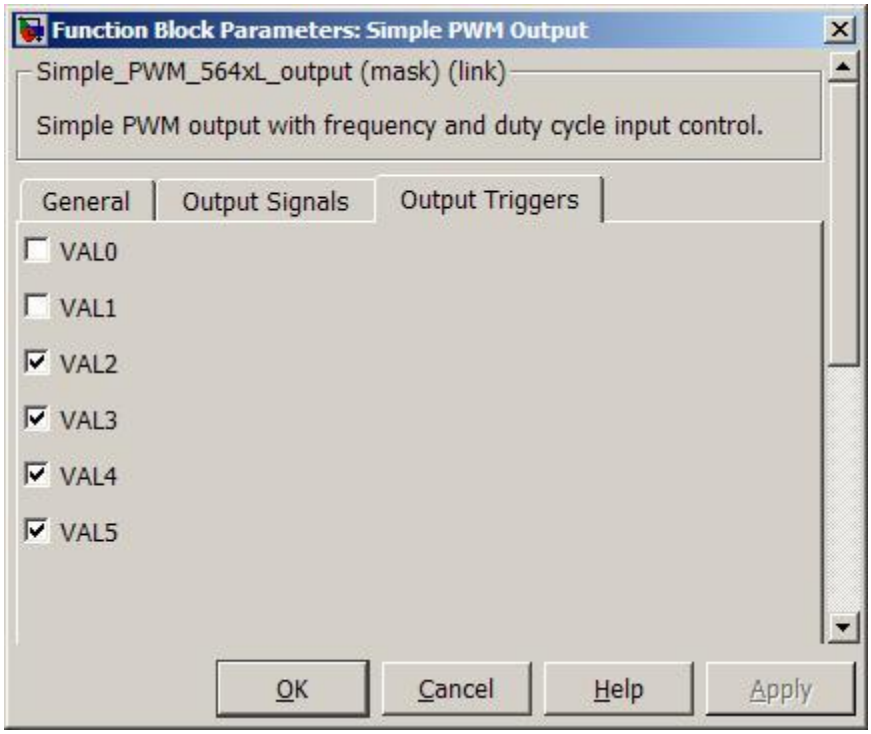
Names	Selection Types	Range	Description
A[n] output pin (where n is the number of the selected submodule)	Pop-up	The list of available pins depends on the selected module.	Channel A pin selection

B[n] output pin (where n is the number of the selected submodule)	Pop-up	The list of available pins depends on the selected module.	Channel B pin selection
X[n] output pin (where n is the number of the selected submodule)	Pop-up	The list of available pins depends on the selected module.	X output pin selection
Invert PWMA Output Polarity	Check-box	On/Off	
Invert PWMB Output Polarity	Check-box	On/Off	
Invert PWMX Output Polarity	Check-box	On/Off	



- The Output Triggers tab contains the following parameters:

Names	Selection Types	Range	Description
VAL*n* *n* is 0 – 5	Check-box	On/Off	Enable generation of Output Trigger signal based on the counter value matching VAL*n* register value



6.2.4.3.7 Block Dependency

None

6.2.4.3.8 Block Miscellaneous Details:

1. Duty Cycle output signal needed for simulation purpose only.
2. If it is desired to only use one of the output pair than select None for the pin selection of the unused output and tie the block input for the unused output to a constant value or tie it to the same input as the other output that is used.

6.2.4.4 Sine Wave Generator

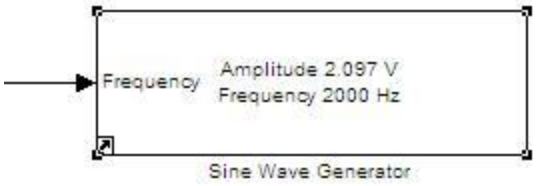
6.2.4.4.1 Block Name

Sine Wave Generator Block

6.2.4.4.2 Block Description

The main functionality of the block is to generate sinusoidal voltage signal

6.2.4.4.3 Block Image



6.2.4.4.4 Inputs:

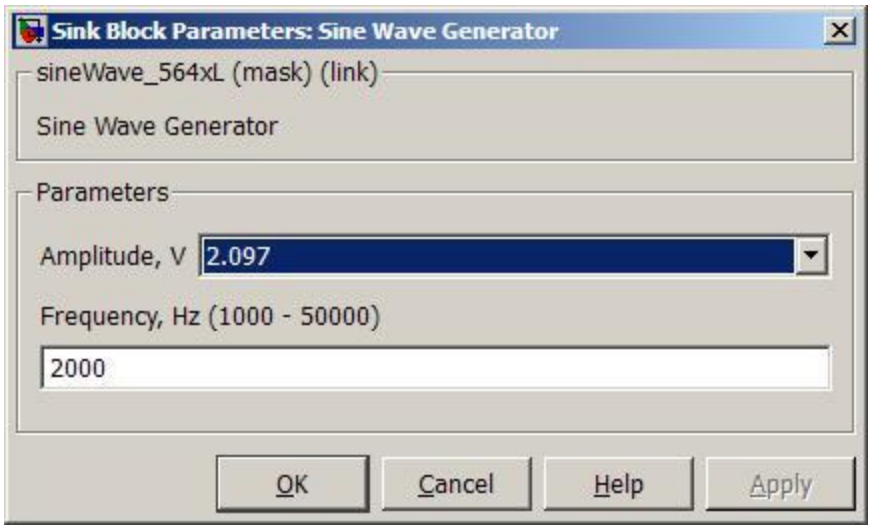
- Output Signal Frequency (uint16)

6.2.4.4.5 Outputs:

- None

6.2.4.4.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
Amplitude, V	Pop-up	0.467 0.538 0.611 0.682 0.754 0.826 0.898 0.969 1.090 1.258 1.426 1.595 1.761 1.929 2.097 2.264	Peak-to-peak amplitude
Frequency, Hz (1000 - 50000)	Text-box	1000 – 50000	Initial signal frequency



6.2.4.4.7 Block Dependency

None

6.2.4.4.8 Block Miscellaneous Details:

None

6.2.4.5 Three-phase PWM Output

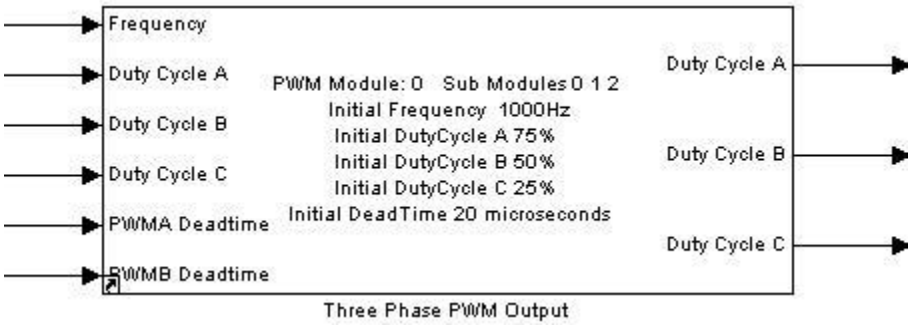
6.2.4.5.1 Block Name

Three-phase FlexPWM Output Block

6.2.4.5.2 Block Description

The main functionality of the block is to generate a three-phase complementary center-aligned PWM signals on A and B outputs of the three selected FlexPWM modules.

6.2.4.5.3 Block Image



6.2.4.5.4 Inputs:

- Frequency (uint32)
- Duty Cycle A (uint32)
- Duty Cycle B (uint32)
- Duty Cycle C (uint32)
- PWMA Deadtime (uint32)
- PWMB Deadtime (uint32)

6.2.4.5.5 Outputs:

- Duty Cycle A (double) - if "Duty Cycle Simulation Output" is On
- Duty Cycle B (double) - if "Duty Cycle Simulation Output" is On
- Duty Cycle C (double) - if "Duty Cycle Simulation Output" is On

6.2.4.5.6 Block Dialog and Parameters:

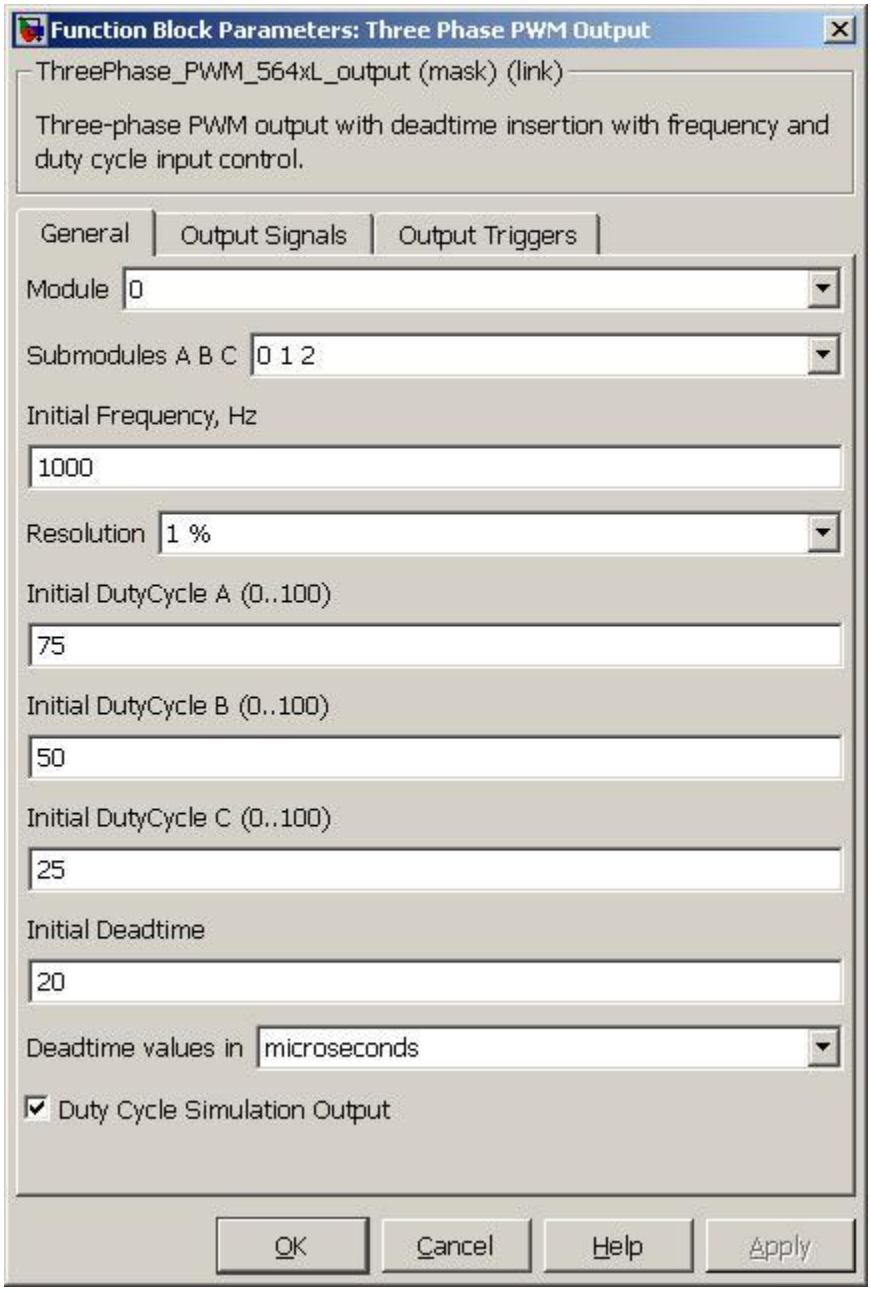
The block dialog consists of the following tabs:

- General
- Output Signals
- Output Triggers

- The General tab contains the following parameters:

Names	Selection Types	Range	Description
Module	Pop-up	0 – 1	FlexPWM module
Submodules	Pop-up	0 1 2 0 1 3 0 2 3	FlexPWM submodules (A B C)
Initial Frequency Hz	Text-box	Depends on Motor Control Clock value	Initial frequency of PWM output signals
Resolution	Pop-up	1 % 0.1 % 0.01 % 0.001 %	Duty cycle resolution: actual Duty cycle is equal to Duty cycle parameter or input value

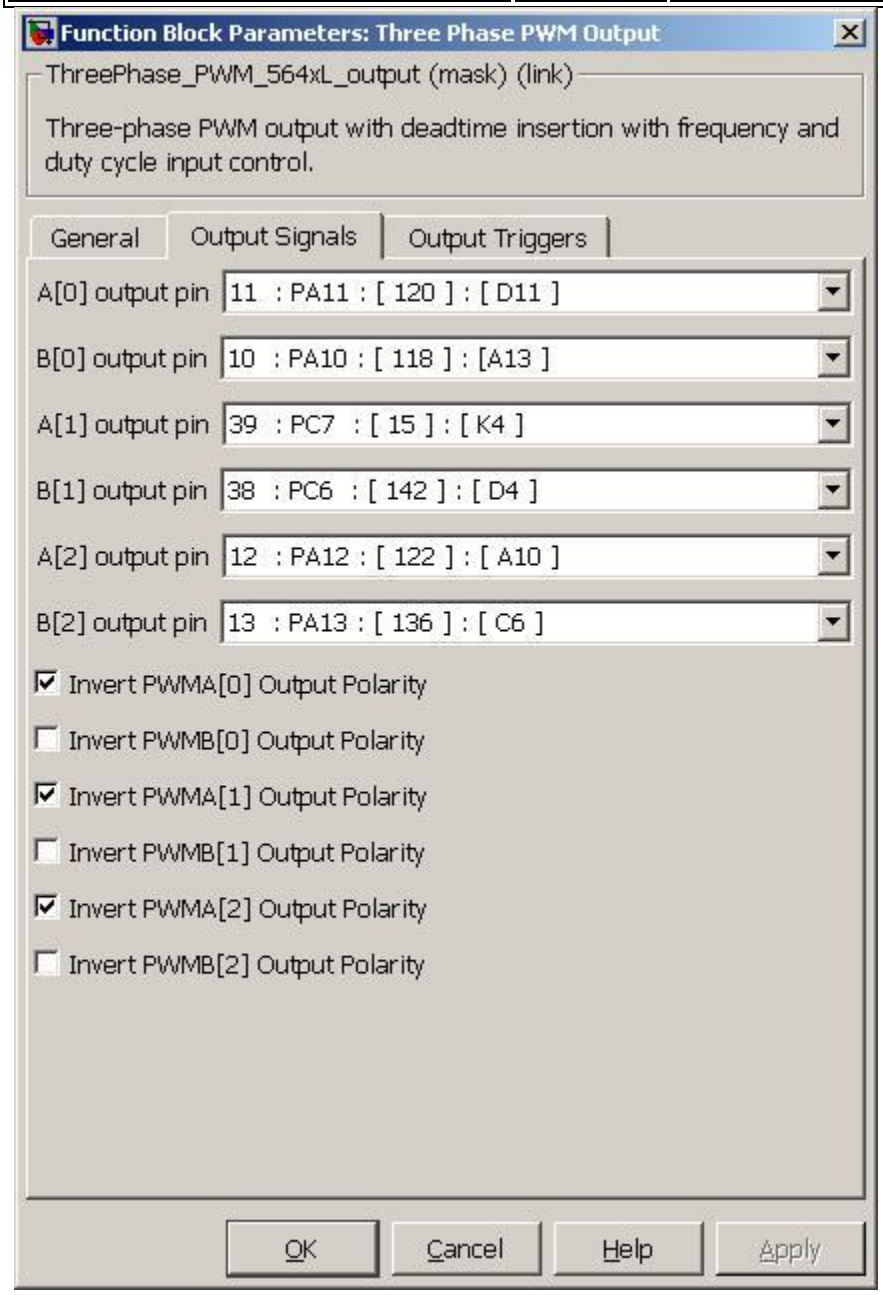
			multiplied by resolution
Initial DutyCycle A	Text-box	0 – 100/resolution	Initial Duty Cycle of submodule A output signals
Initial DutyCycle B	Text-box	0 – 100/resolution	Initial Duty Cycle of submodule B output signals
Initial DutyCycle C	Text-box	0 – 100/resolution	Initial Duty Cycle of submodule C output signals
Initial Deadtime	Text-box	0 – 4095 IPBus clock cycles	Deadtime during 0 to 1 transitions of the PWM outputs
Initial Deadtime values in	Pop-up	– IPBus clock cycles – microseconds	Deadtime values unit
Duty Cycle Simulation Output	Check-box	On/Off	If DutyCycle simulation output is used



- The Output Signals tab contains the following parameters:

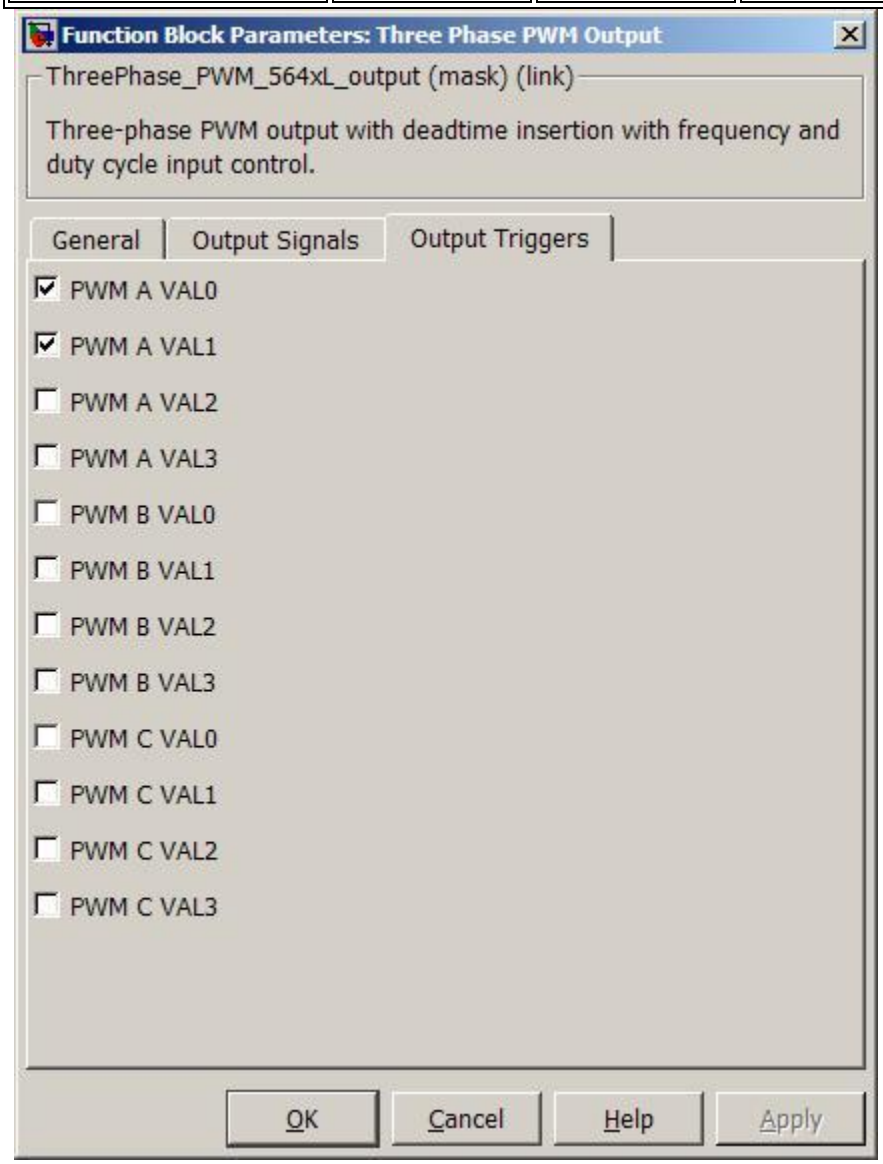
Names	Selection Types	Range	Description
A[n] output pin (where n is the number of the selected submodule)	Pop-up	The list of available pins depends on the selected module.	Channel A pin selection
B[n] output pin (where n is the number of the selected submodule)	Pop-up	The list of available pins depends on the selected	Channel B pin

		module.	selection
Invert PWMA[n] Output Polarity (where n is the number of the selected submodule)	Check-box	On/Off	
Invert PWMB[n] Output Polarity (where n is the number of the selected submodule)	Check-box	On/Off	



- The Output Triggers tab contains the following parameters:

Names	Selection Types	Range	Description
PWM *x* VAL*n* *n* is 0 – 3 *x* is A,B,C	Check-box	On/Off	Enable generation of Output Trigger signal based on the counter value matching VAL*n* register value



6.2.4.5.7 Block Dependency

None

6.2.4.5.8 Block Miscellaneous Details:

Duty Cycle A, Duty Cycle B and Duty Cycle C output signals needed for simulation purpose only.

6.2.5 Timer Blocks

6.2.5.1 eTimer Configuration

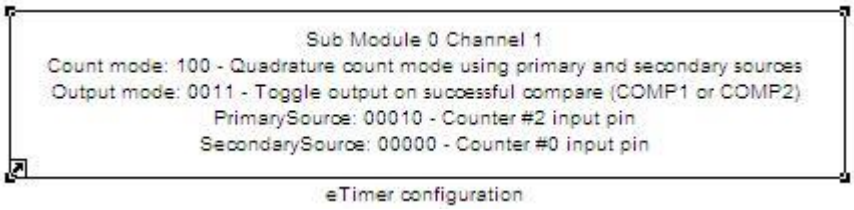
6.2.5.1.1 Block Name

eTimer Configuration Block

6.2.5.1.2 Block Description

This block is used to configure the eTimer.

6.2.5.1.3 Block Image



6.2.5.1.4 Inputs:

- None

6.2.5.1.5 Outputs:

- None

6.2.5.1.6 Block Dialog and Parameters:

The block dialog consists of the following tabs:

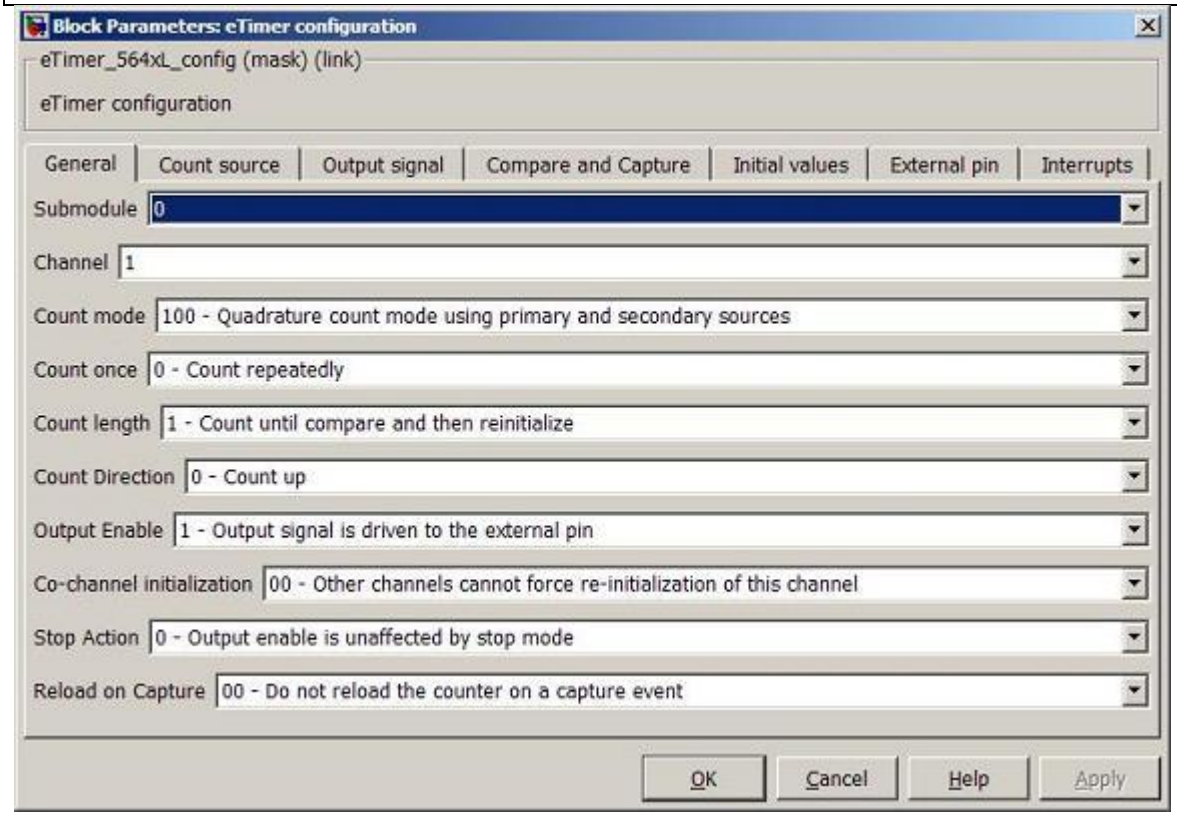
- General
- Count Source
- Output Signal
- Compare and Capture
- Initial Values
- External Pin
- Interrupts

- The General tab contains the following parameters:

Names	Selection Types	Range	Description
Submodule	Pop-up	0 – 2	

Channel	Pop-up	0 – 5	
Count mode	Pop-up	000 – No Operation 001 – Count rising edges of primary source 010 – Count rising and falling edges of primary source 011 – Count rising edges of primary source while secondary input high active 100 – Quadrature count mode using primary and secondary sources 101 – Count primary source rising edges while secondary source specifies direction 110 – Edge of secondary source triggers primary count till compare 111 – Cascaded counter mode	CNTMODE value of CTRL1 Register
Count once	Pop-up	0 – Count repeatedly 1 – Count until compare and then stop	ONCE value of CTRL1 Register
Count length	Pop-up	0 – Continue counting to roll over 1 – Count until compare and then reinitialize	LENGTH value of CTRL1 Register
Count Direction	Pop-up	0 – Count up 1 – Count down	
Output Enable	Pop-up	0 – The external pin is configured as an input 1 – Output signal is driven to the external pin	OEN value of CTRL2 Register
Co-channel initialization	Pop-up	00 – Other channels cannot force re-initialization of this channel 01 – Other channels may force re-initialization of channel counter with LOAD 10 – Other channels may force re-initialization of channel counter with CMPLD1 or CMPLD2	COINIT value of CTRL2 Register
Stop Action	Pop-up	0 – Output enable is unaffected by stop mode 1 – Output enable is disabled during stop mode	STPEN value of CTRL3 Register
Reload on Capture	Pop-up	00 – Do not reload the counter on a capture event 01 – Reload the counter on a capture 1 event 10 – Reload the counter on a capture 2 event 11 – Reload the counter on both a capture 1	ROC value of CTRL3 Register

		event and a capture 2 event	



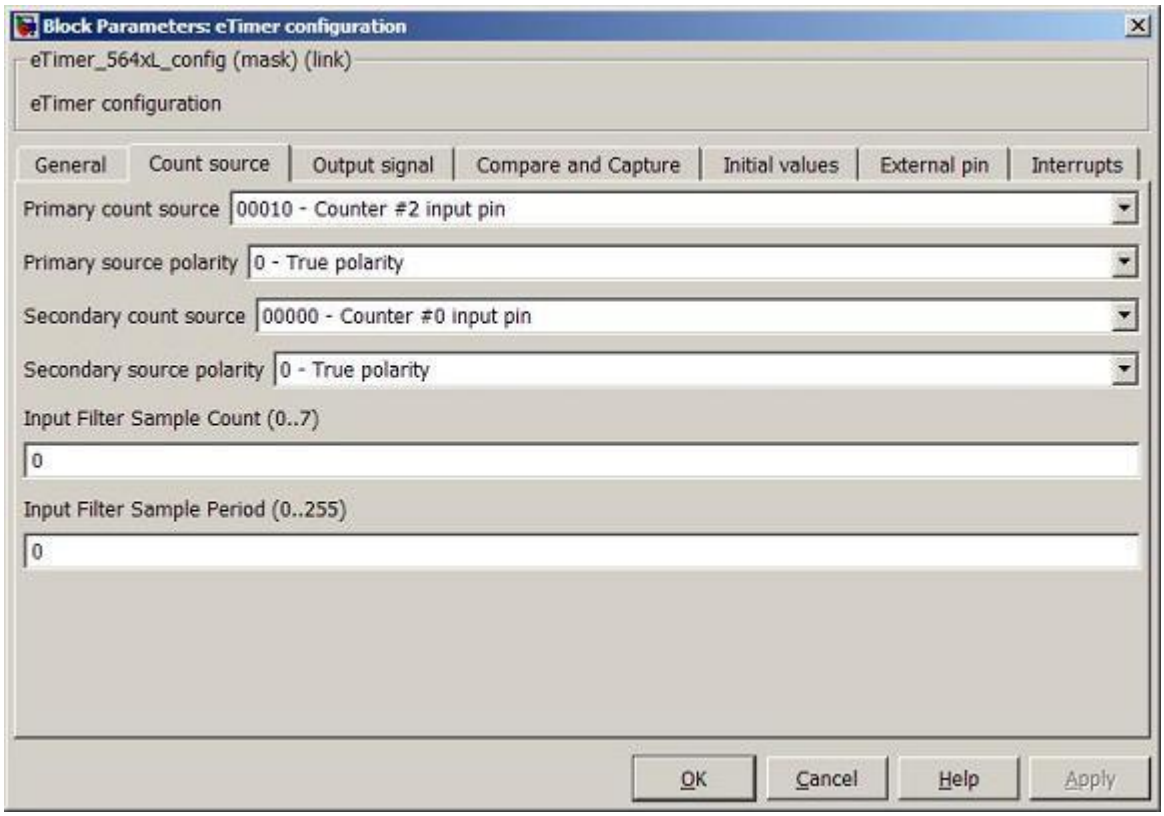
- The Count Source tab contains the following parameters:

Names	Selection Types	Range	Description
Primary count source	Pop-up	See count source values table	PRISRC value of CTRL1 Register
Primary source polarity	Pop-up	0 – True polarity 1 – Inverted polarity	PIPS value of CTRL2 Register
Secondary count source	Pop-up	See count source values table	SECSRC value of CTRL1 Register
Secondary source polarity	Pop-up	0 – True polarity 1 – Inverted polarity	SIPS value of CTRL2 Register
Input Filter Sample Count (0..7)	Text-box	0 – 7	FILT_CNT

			value of FILT Register
Input Filter Sample Period (0..255)	Text-box	0 – 255	FILT_PER value of FILT Register

- Count source values:

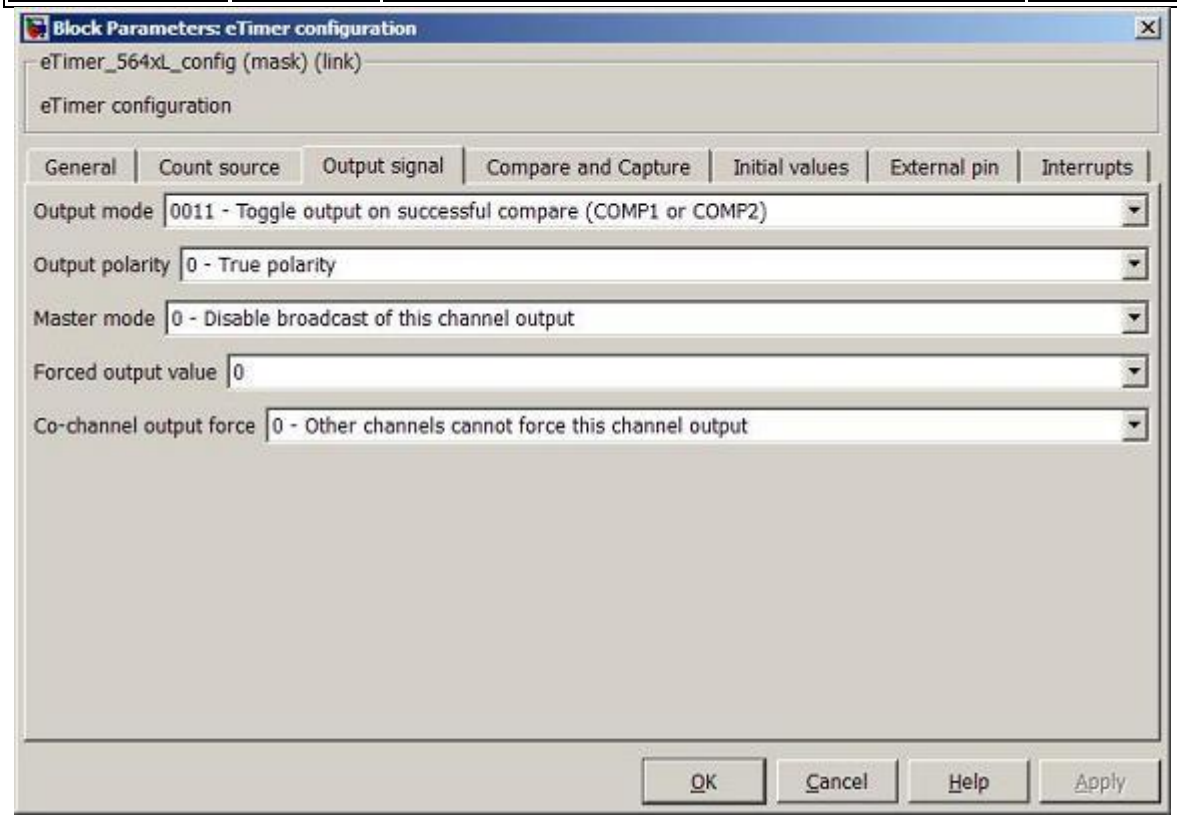
00000 – Counter #0 input pin
00001 – Counter #1 input pin
00010 – Counter #2 input pin
00011 – Counter #3 input pin
00100 – Counter #4 input pin
00101 – Counter #5 input pin
01000 – Auxiliary input #0 pin
01001 – Auxiliary input #1 pin
01010 – Auxiliary input #2 pin
10000 – Counter #0 output
10001 – Counter #1 output
10010 – Counter #2 output
10011 – Counter #3 output
10100 – Counter #4 output
10101 – Counter #5 output
11000 – IP Bus clock divide by 1 prescaler
11001 – IP Bus clock divide by 2 prescaler
11010 – IP Bus clock divide by 4 prescaler
11011 – IP Bus clock divide by 8 prescaler
11100 – IP Bus clock divide by 16 prescaler
11101 – IP Bus clock divide by 32 prescaler
11110 – IP Bus clock divide by 64 prescaler
11111 – IP Bus clock divide by 128 prescaler



- The Output Signal tab contains the following parameters:

Names	Selection Types	Range	Description
Output mode	Pop-up	0000 – Software controlled 0001 – Clear output on successful compare (COMP1 or COMP2) 0010 – Set output on successful compare (COMP1 or COMP2) 0011 – Toggle output on successful compare (COMP1 or COMP2) 0100 – Toggle output using alternating compare registers 0101 – Set output on compare with COMP1 and clear on secondary source input edge 0110 – Set output on compare with COMP2 and clear on secondary source input edge 0111 – Set output on compare and clear on counter roll over 1000 – Set output on compare on COMP1 and clear on compare on COMP2 1001 – Asserted while counter is active and	OUTMODE value of CTRL2 Register

		cleared when counter is stopped 1010 – Asserted when counting up and cleared when counting down 1111 – Enable gated clock output while counter is active	
Output polarity	Pop-up	0 – True polarity 1 – Inverted polarity	OPS value of CTRL2 Register
Master mode	Pop-up	0 – Disable broadcast of this channel output 1 – Enable broadcast of this channel output	MSTR value of CTRL2 Register
Forced output value	Pop-up	0,1	VAL value of CTRL2 Register
Co-channel output force	Pop-up	0 – Other channels cannot force this channel output 1 – Other channels may force this channel output	COFRC value of CTRL2 Register



- Compare and Capture tab contains the following parameters:

Names	Selection Types	Range	Description
Compare Load 1	Pop-up	See Compare Load values table	CLC1 value of CCCTRL Register
Compare Load 2	Pop-up	See Compare Load values table	CLC2 value of CCCTRL Register
Compare Mode	Pop-up	00 – COMP1 and COMP2 are used when the counter is counting up 01 – COMP1 is used when the counter is counting down and COMP2 when counting up 10 – COMP1 is used when the counter is counting up and COMP2 when counting down 11 – COMP1 and COMP2 are used when the counter is counting down	CMPMODE value of CCCTRL Register
Capture 1 Mode	Pop-up	See Capture modes table	CPT1MODE value of CCCTRL Register
Capture 2 Mode	Pop-up	See Capture modes table	CPT2MODE value of CCCTRL Register
One Shot Capture Mode	Pop-up	0 – Free running mode is selected 1 – One shot mode is selected	ONESHOT value of CCCTRL Register
Capture FIFO water mark	Pop-up	0 – 3	CFWM value of CCCTRL Register

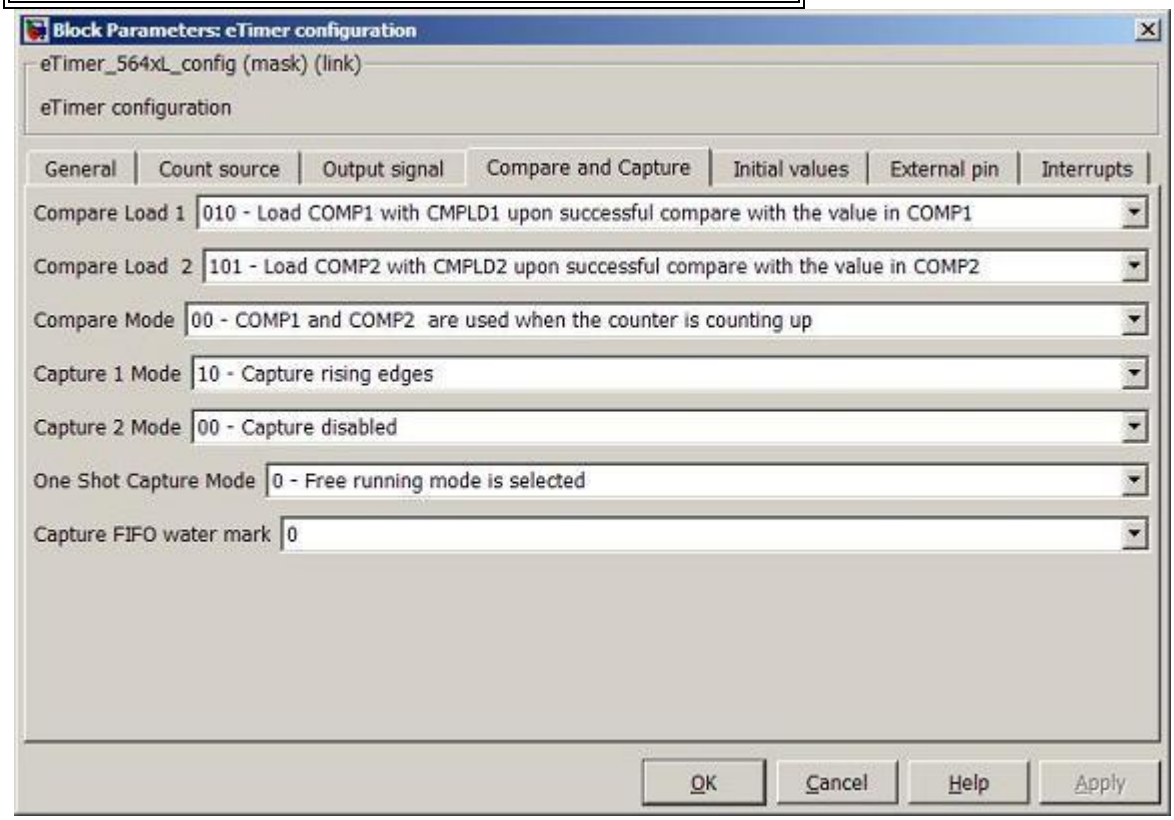
- Compare Load values:

000 – COMP1 is never preload
010 – Load COMP1 with CMPLD1 upon successful compare with the value in COMP1
011 – Load COMP1 with CMPLD1 upon successful compare with the value in COMP2
100 – Load COMP1 with CMPLD2 upon successful compare with the value in COMP1
101 – Load COMP1 with CMPLD2 upon successful compare with the value in COMP2

110 – Load CNTR with CMPLD1 upon successful compare with the value in COMP1
111 – Load CNTR with CMPLD1 upon successful compare with the value in COMP2

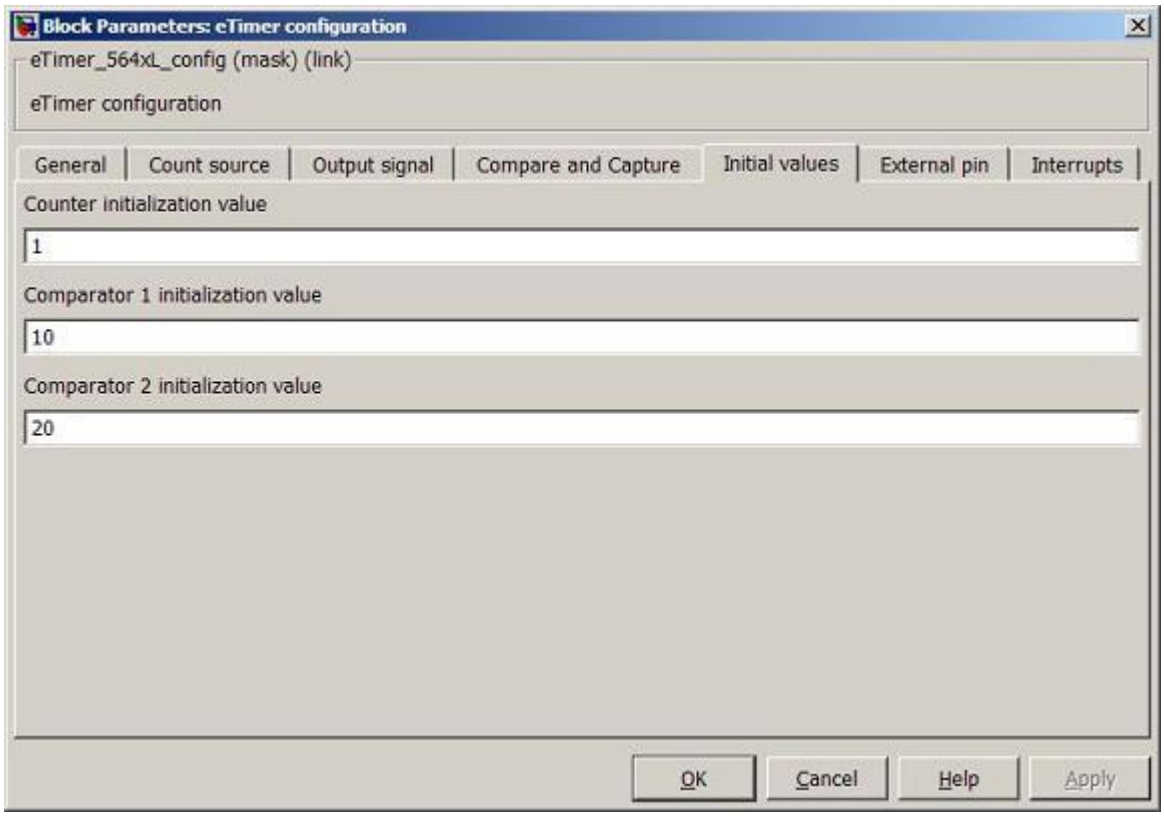
- Capture modes:

00 – Capture disabled
01 – Capture falling edges
10 – Capture rising edges
11 – Capture any edge



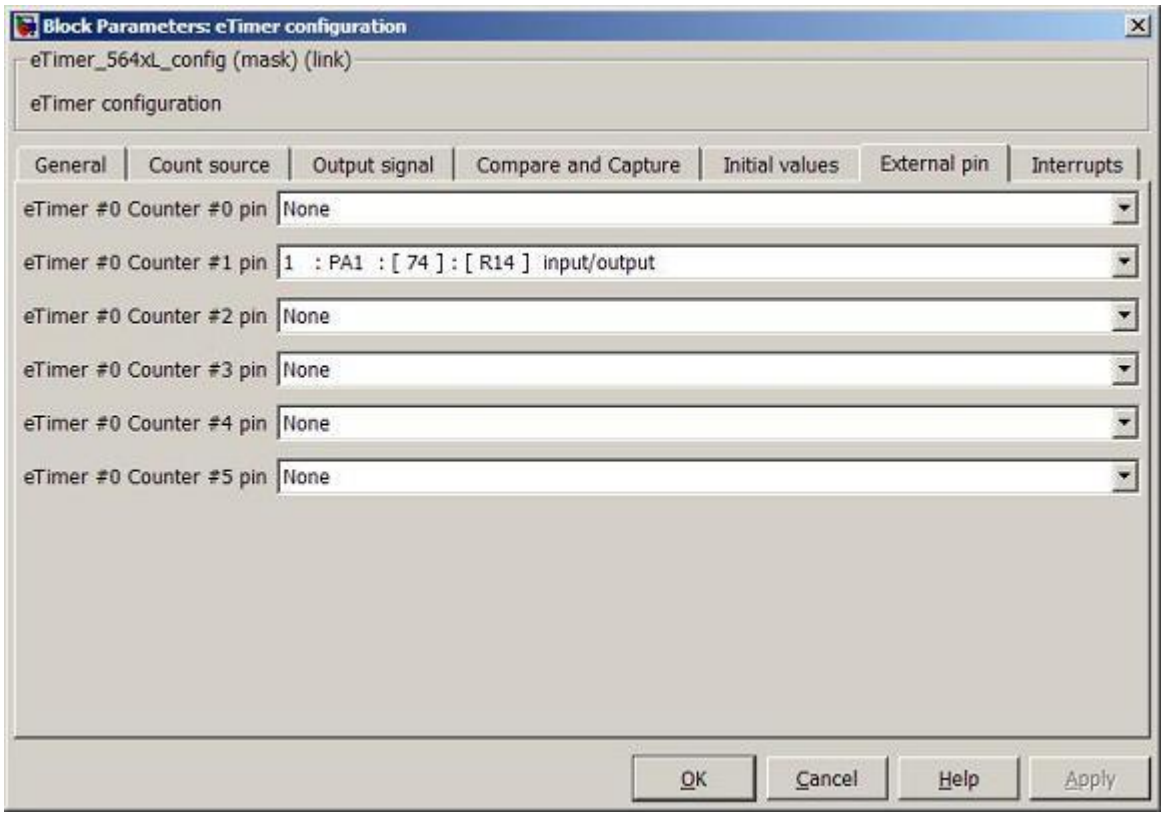
- Initial Values tab contains the following parameters:

Names	Selection Types	Range	Description
Counter initialization value	Text-box	0 – 65535	
Comparator 1 initialization value	Text-box	0 – 65535	
Comparator 2 initialization value	Text-box	0 – 65535	



- External Pin tab contains the following parameters:

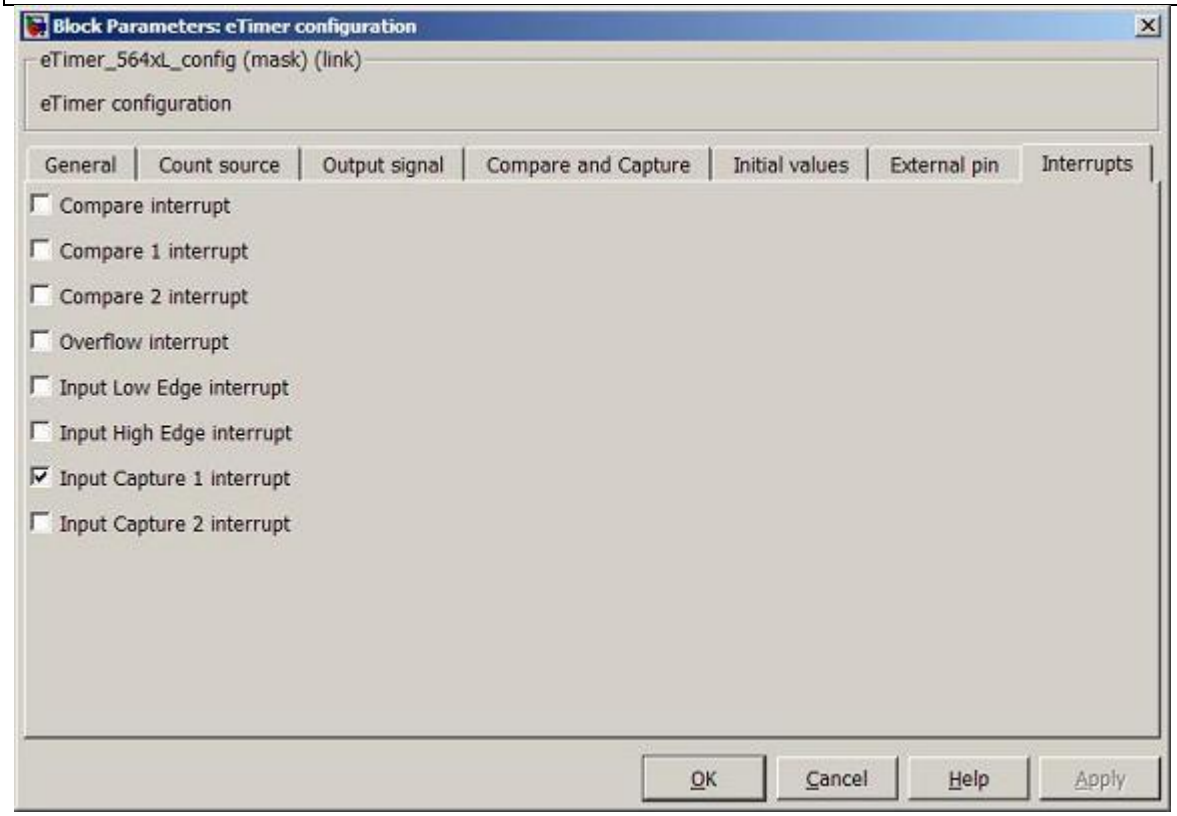
Names	Selection Types	Range	Description
eTimer #*n* Counter #*x* pin *n* is submodule number *x* is 0 – 5	Pop-up	The list of available pins depends on the selected module.	eTimer #*n* Counter #*x* pin selection



- Interrupts tab contains the following parameters:

Names	Selection Types	Range	Description
Compare interrupt	Check-box	Enable/Disable	TCFIE value of INTDMA register
Compare 1 interrupt	Check-box	Enable/Disable	TCF1IE value of INTDMA register
Compare 2 interrupt	Check-box	Enable/Disable	TCF2IE value of INTDMA register
Overflow interrupt	Check-box	Enable/Disable	TOFIE value of INTDMA register
Input Low Edge interrupt	Check-box	Enable/Disable	IELFIE value of INTDMA register
Input High Edge interrup	Check-box	Enable/Disable	IEHFIE value of INTDMA register
Input Capture 1 interrupt	Check-box	Enable/Disable	ICF1IE value of INTDMA register
Input Capture 2 interrupt	Check-box	Enable/Disable	ICF2IE value of INTDMA

			register



6.2.5.1.7 Block Dependency

None

6.2.5.1.8 Block Miscellaneous Details:

None

6.2.5.2 eTimer ISR

6.2.5.2.1 Block Name

eTimer ISR Block

6.2.5.2.2 Block Description

The main functionality of the block is to process eTimer ISRs.

6.2.5.2.3 Block Image

6.2.5.2.4 Inputs:

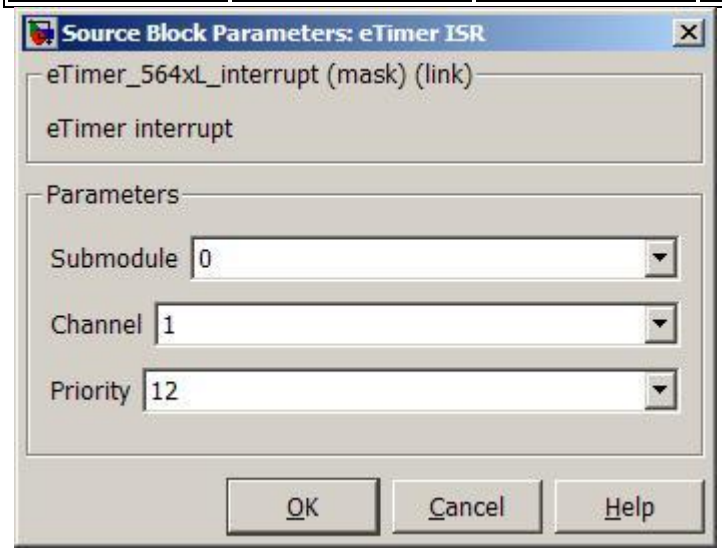
- None

6.2.5.2.5 Outputs:

- Function-Call
- Status Register (STS) value (uint16)

6.2.5.2.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
Submodule	Pop-up	0 – 2	
Channel	Pop-up	0 – 5	
Priority	Pop-up	0 – 15	Interrupt priority level



6.2.5.2.7 Block Dependency

Please do the following:

1. Configure related eTimer submodule and channel via eTimer Configuration Block

6.2.5.2.8 Block Miscellaneous Details:

None

6.2.5.3 eTimer ISR Enable

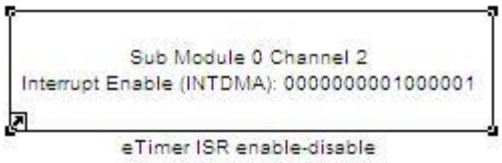
6.2.5.3.1 Block Name

eTimer Interrupt Enable/Disable Block

6.2.5.3.2 Block Description

The main functionality of the block is to enable/disable eTimer interrupts

6.2.5.3.3 Block Image



6.2.5.3.4 Inputs:

- None

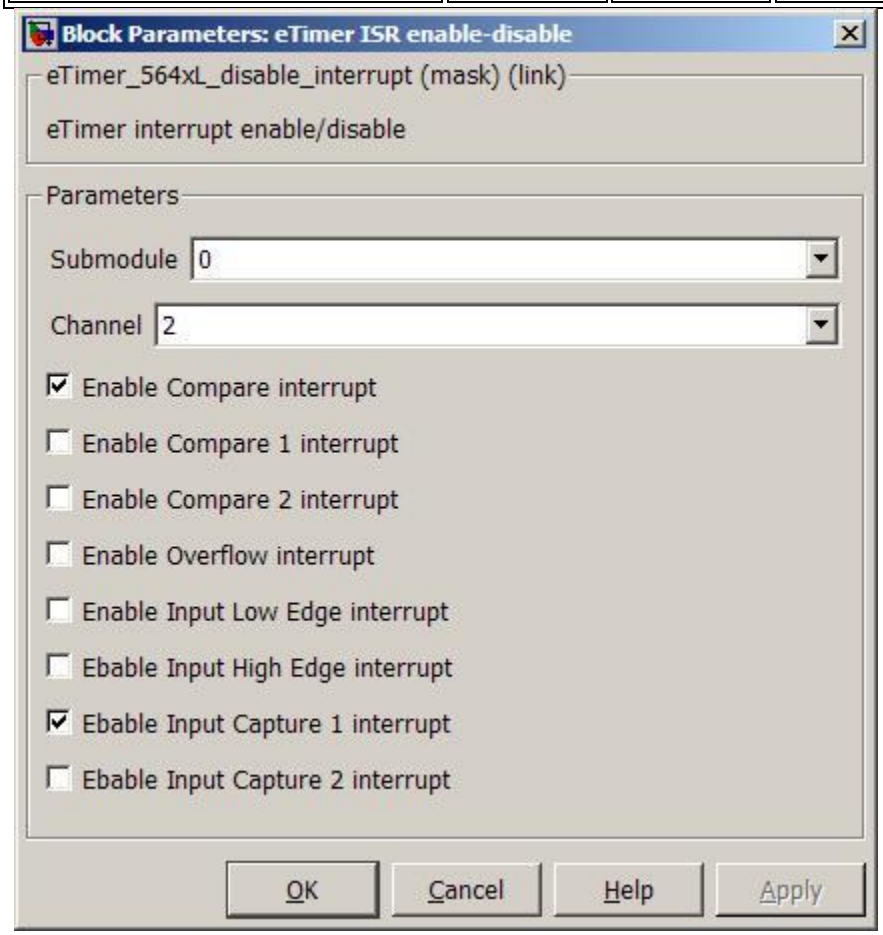
6.2.5.3.5 Outputs:

- None

6.2.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
Submodule	Pop-up	0 – 2	
Channel	Pop-up	0 – 5	
Enable Compare interrupt	Check-box	On/Off	Enable/Disable Compare interrupt
Enable Compare 1 interrupt	Check-box	On/Off	Enable/Disable Compare 1 interrupt
Enable Compare 2 interrupt	Check-box	On/Off	Enable/Disable Compare 2 interrupt
Enable Overflow interrupt	Check-box	On/Off	Enable/Disable Overflow interrupt
Enable Input Low Edge interrupt	Check-box	On/Off	Enable/Disable Input Low Edge interrupt

Enable Input Low High interrupt	Check-box	On/Off	Enable/Disable Input High Edge interrupt
Enable Input Capture 1 interrupt	Check-box	On/Off	Enable/Disable Input Capture 1 interrupt
Enable Input Capture 2 interrupt	Check-box	On/Off	Enable/Disable Input Capture 2 interrupt



6.2.6.1.1 Block Dependency

Please do the following:

1. Configure related eTimer submodule and channel via eTimer Configuration Block
2. Configure related eTimer interrupts via eTimer ISR Block

6.2.6.1.2 Block Miscellaneous Details:

None

6.2.6.2 eTimer Capture

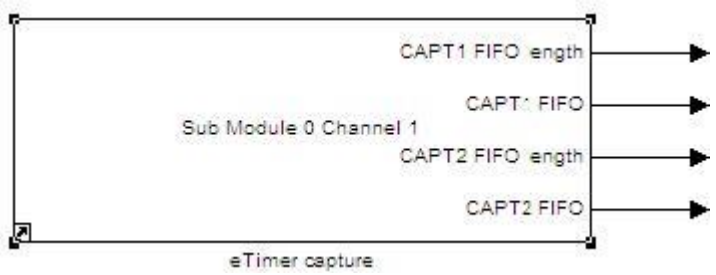
6.2.6.2.1 Block Name

eTimer Capture Block

6.2.6.2.2 Block Description

The main functionality of the block is to read eTimer Capture FIFOs.

6.2.6.2.3 Block Image



6.2.6.2.4 Inputs:

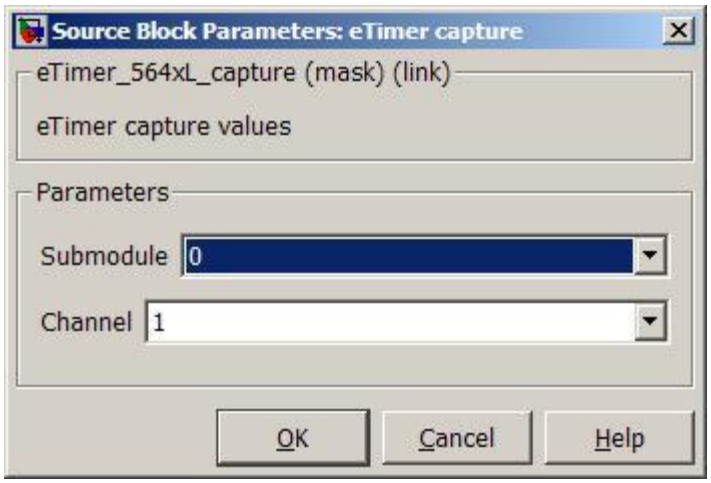
- None

6.2.6.2.5 Outputs:

- Number of words in CAPT1 FIFO (uint16)
- CAPT1 FIFO (uint16(8))
- Number of words in CAPT2 FIFO (uint16)
- CAPT2 FIFO (uint16(8))

6.2.6.2.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
Submodule	Pop-up	0 – 2	
Channel	Pop-up	0 – 5	



6.2.6.2.7 Block Dependency

Please do the following:

1. Configure related eTimer submodule and channel via eTimer Configuration Block

6.2.6.2.8 Block Miscellaneous Details:

None

6.2.6.3 eTimer Pre-load

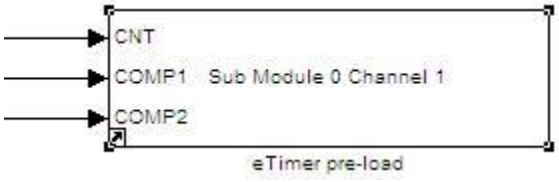
6.2.6.3.1 Block Name

eTimer Pre-load Block

6.2.6.3.2 Block Description

The main functionality of the block is eTimer counter and comparators pre-loading.

6.2.6.3.3 Block Image



6.2.6.3.4 Inputs:

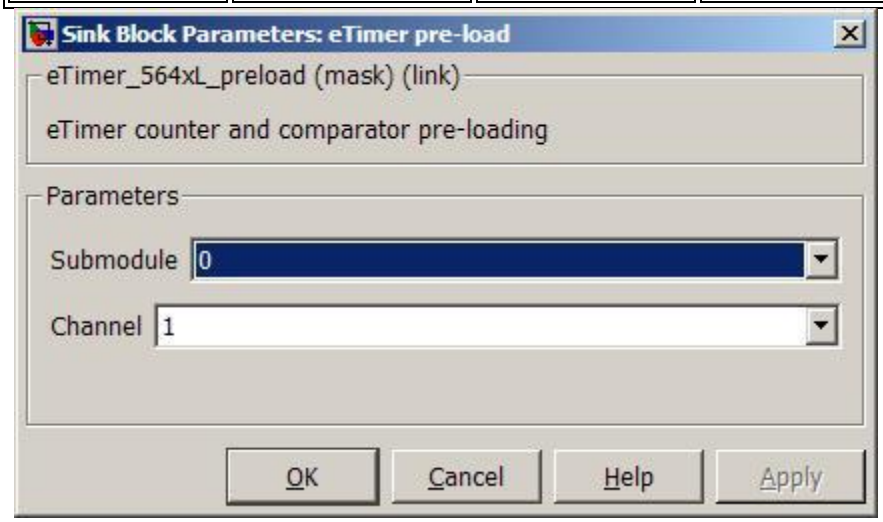
- None

6.2.6.3.5 Outputs:

- Counter initialization value (uint16)
- Comparator 1 initialization value (uint16)
- Comparator 2 initialization value (uint16)

6.2.6.3.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
Submodule	Pop-up	0 – 2	
Channel	Pop-up	0 – 5	



6.2.6.3.7 Block Dependency

Please do the following:

1. Configure related eTimer submodule and channel via eTimer Configuration Block

6.2.6.3.8 Block Miscellaneous Details:

None

6.2.6.4 eTimer CNTR Register Read

6.2.6.4.1 Block Name

eTimer CNTR Register Read Block

6.2.6.4.2 Block Description

The main functionality of the block is direct read from eTimer Counter register.

6.2.6.4.3 Block Image

6.2.6.4.4 Inputs:

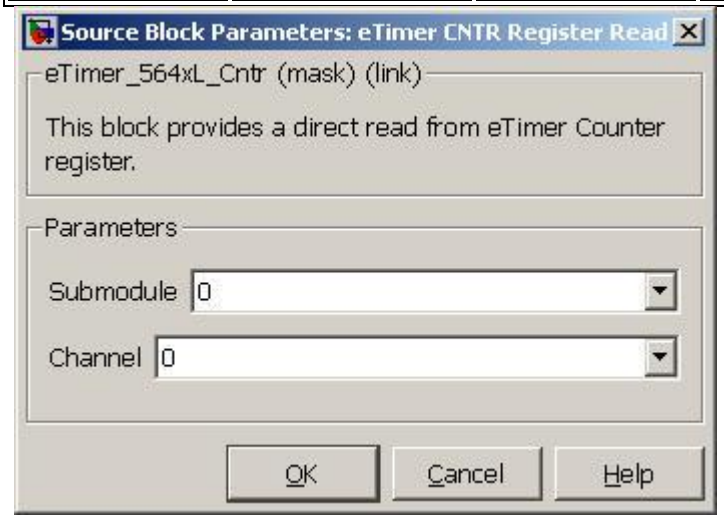
- None

6.2.6.4.5 Outputs:

- eTimer Counter register value (uint16)

6.2.6.4.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
Submodule	Pop-up	0 – 2	
Channel	Pop-up	0 – 5	



6.2.6.4.7 Block Dependency

Please do the following:

1. Configure related eTimer submodule and channel via eTimer Configuration Block

6.2.6.4.8 Block Miscellaneous Details:

None

6.2.6.5 eTimer HOLD Register Read

6.2.6.5.1 Block Name

eTimer HOLD Register Read Block

6.2.6.5.2 Block Description

The main functionality of the block is direct read from eTimer Hold register.

6.2.6.5.3 Block Image

6.2.6.5.4 Inputs:

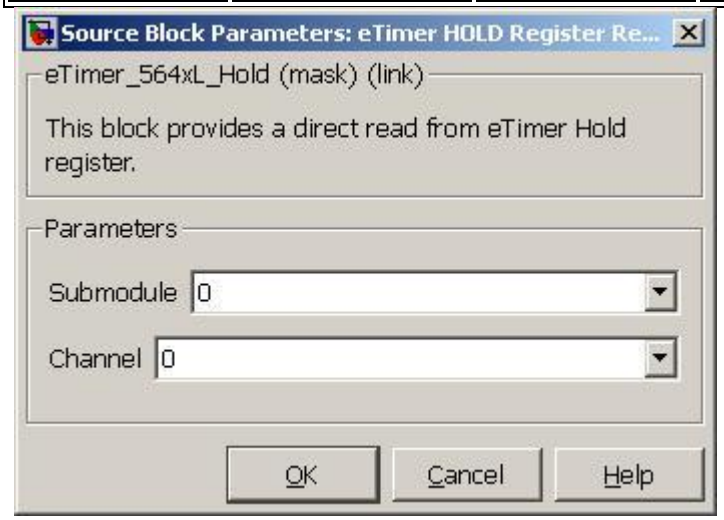
- None

6.2.6.5.5 Outputs:

- eTimer Hold register value (uint16)

6.2.6.5.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
Submodule	Pop-up	0 – 2	
Channel	Pop-up	0 – 5	



6.2.6.5.7 Block Dependency

Please do the following:

1. Configure related eTimer submodule and channel via eTimer Configuration Block

6.2.6.5.8 Block Miscellaneous Details:

None

6.3 Utility Blocks

The Motor Control Development Toolbox provides utility blocks for use with your targeted application. The utility blocks allow the user to leverage certain processor capabilities for software engineering and configuration optimization tasks. The user can use the profiling block to determine the execution time of a specific function in code.

6.3.1 CCP DAQ

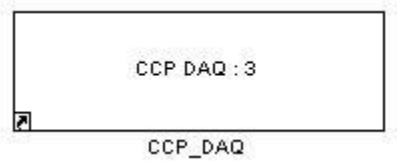
6.3.1.1 Block Name

CCP DAQ Block

6.3.1.2 Block Description

This block generates DAQ trigger request for CCP.

6.3.1.3 Block Image



6.3.1.4 Inputs:

- None

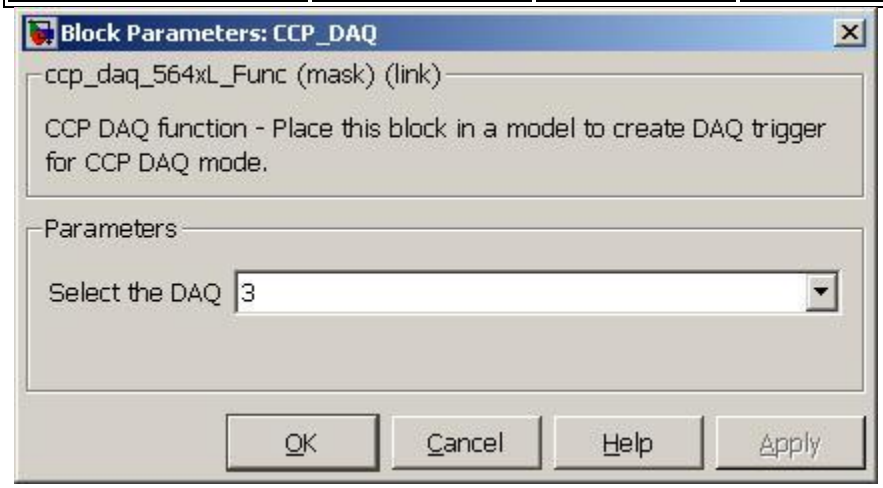
6.3.1.5 Outputs:

- None

6.3.1.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description

Select the DAQ	Pop-up	0 – 255	Index of DAQ request



6.3.1.7 Block Dependency

CCP should be enabled in RAppID_564xL_Config block.

6.3.1.8 Block Miscellaneous Details:

Place this block in a model to create DAQ trigger for CCP DAQ mode.

6.3.2 FreeMaster Data Recorder

6.3.2.1 Block Name

FreeMaster Data Recorder Block

6.3.2.2 Block Description

This block allows the user to store variables in internal memory and download them through FreeMaster tool.

6.3.2.3 Block Image

6.3.2.4 Inputs:

- None

6.3.2.5 Outputs:

- None

6.3.2.6 Block Dialog and Parameters:

None

6.3.2.7 Block Dependency

FreeMaster should be enabled in RAppID_564xL_Config block.

6.3.2.8 Block Miscellaneous Details:

None

6.3.3 Memory Read

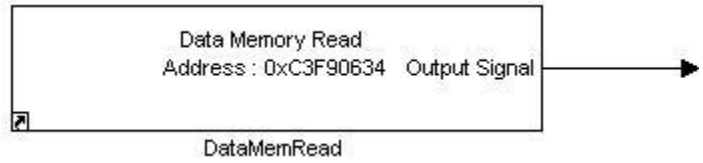
6.3.3.1 Block Name

Memory Read Block

6.3.3.2 Block Description

The main functionality of the block is to read data at memory location specified by the base address and the offset value. That is reading data at memory location (Base Address + Offest.)

6.3.3.3 Block Image



6.3.3.4 Inputs:

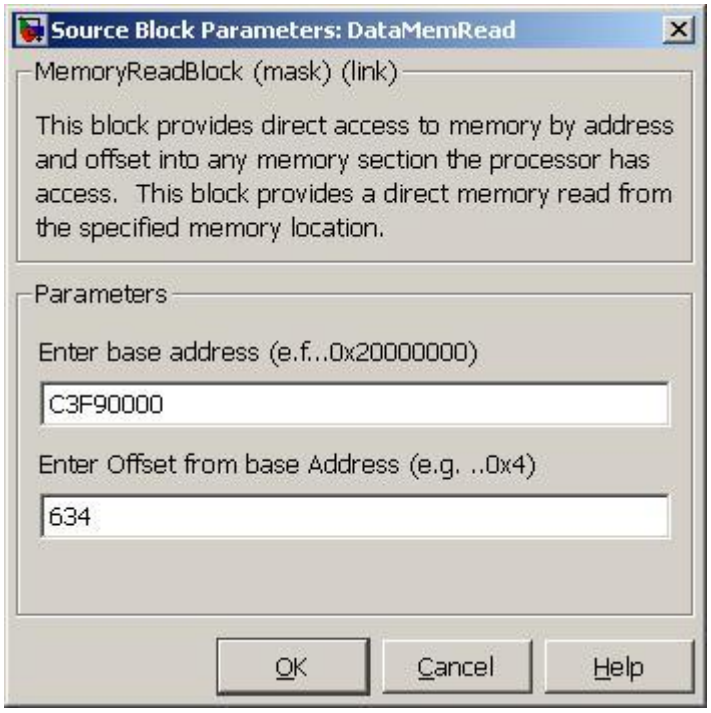
- None

6.3.3.5 Outputs:

- 32bit Data at the specified 32bit memory location.

6.3.3.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
Enter base Address	Text-box	32bit Hex value	
Enter Offset from the base address	Text-box	32bit Hex value	



6.3.3.7 Block Dependency

None

6.3.3.8 Block Miscellaneous Details:

The block requires that the memory address (base address + offset) must be within 32bit boundary.

6.3.4 Memory Write

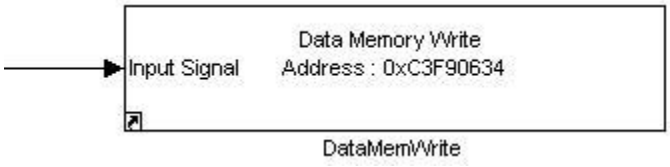
6.3.4.1 Block Name

Memory Write Block

6.3.4.2 Block Description

The main functionality of the block is to write data at memory location specified by the base address and the offset value. That is reading data at memory location (Base Address + Offest.) This block writes 32bit data at a time for the specified memory location. Therefore the address must be within the 32bit boundary.

6.3.4.3 Block Image



6.3.4.4 Inputs:

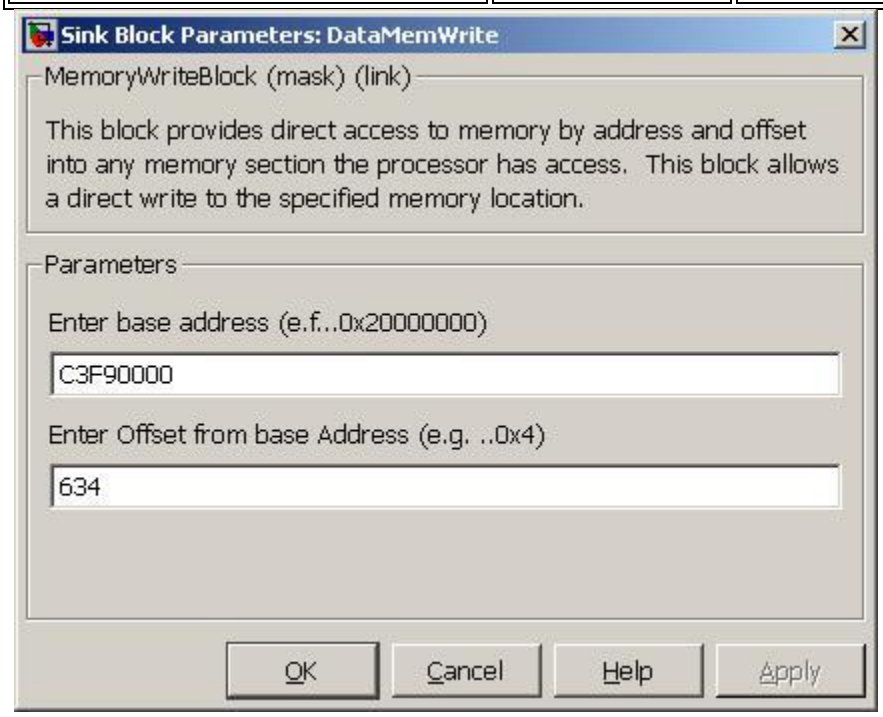
- A 32bit Data at the specified 32bit memory location to write to.

6.3.4.5 Outputs:

- None

6.3.4.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
Enter base Address	Text-box	32bit Hex value	
Enter Offset from the base address	Text-box	32bit Hex value	



6.3.4.7 Block Dependency

None

6.3.4.8 Block Miscellaneous Details:

The block requires that the memory address (base address + offset) must be within 32bit boundary.

6.3.5 Profiling

The profiling block works with atomic subsystems that have been deemed to generate function code. The profiling function instruments the generated code by placing a call just inside the top of the function and just before exiting the function at the bottom. The profiling function takes a snap shot of the Time Base Incrementer (TBI) register at the top and bottom of the function. Then at the end of the function finds the difference between these two values, because the TBI runs at the system clock rate by using this as a timing device the number of clock cycles used during execution is captured faithfully. This will account for things like interrupts and such while the function is executing. The overhead of the code for capturing of this register is approximately 40 clock cycles. This value is NOT removed from the raw numbers of the profile for the function. The use of the profile send function at the end of profiling for sending data over the serial interface add significantly more cycles in overhead to the process. The overhead is incurred after the measurement is taken but will show up when profile function blocks are embedded with hierarchical subsystems being measured, the parent subsystems will inherit the overhead.

The profiling block is supported for PIL simulation and on Target execution only, not supported in SIL simulation or normal simulation.

6.3.5.1 Profiler Instrumentation Block

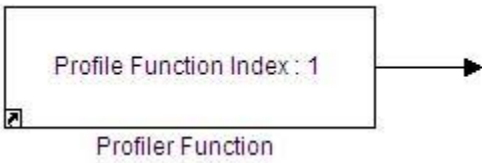
6.3.5.1.1 Block Name

Profiler Function Block

6.3.5.1.2 Block Description

This block profiles the execution time of a function. It places profiling code at the beginning and the end of the function this block is placed in. Up to 100 functions can be profiled at once so a profile index must be selected for each block.

6.3.5.1.3 Block Image



6.3.5.1.4 Inputs:

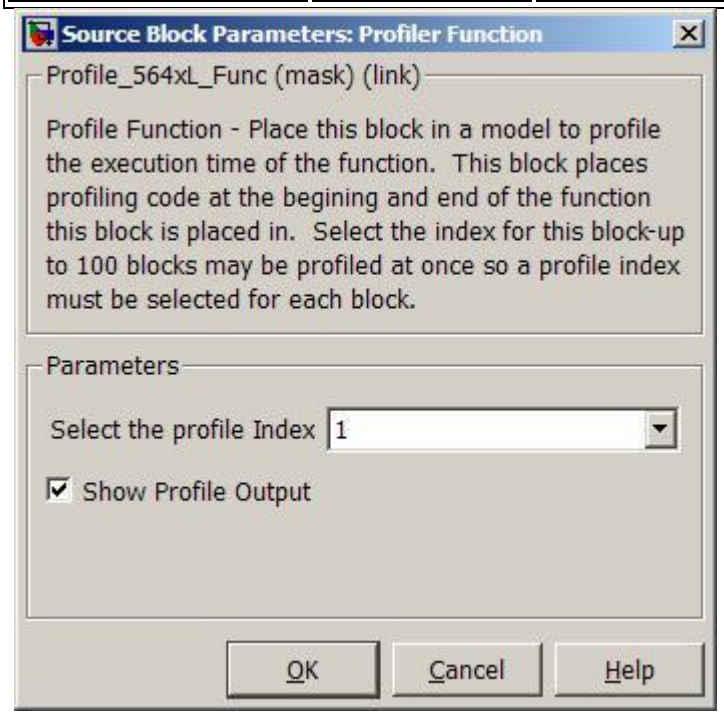
- None

6.3.5.1.5 Outputs:

- Execution time length in system clock ticks (UINT32) (if "Show Profile Output" is set)

6.3.5.1.6 Block Dialog and Parameters:

Names	Selection Types	Range	Description
Profile index	Pop-up	0 – 99	Index of Profile buffer
Show Profile Output	Check-box	On/Off	



6.3.5.1.7 Block Dependency

None

6.3.5.1.8 Block Miscellaneous Details:

None

7 Embedded Targets

The embedded target for RTW/EC provides for selection of either a RAM memory or Flash memory configuration. The MC Toolbox target is derived from RTW/EC ERT target so all code generation options for the MC Toolbox target are available just as the Embedded Real-Time Target (ERT) from MathWorks. All code generation options from ERT are available to the MC Toolbox target. There is also support for including user-specific files into the build.

7.1 Compilers

The Motor Control Development Toolbox supports the Wind River Diab compiler version 5.8.1.0, the Green Hills Multi compiler 5.1.7 and the Freescale Code Warrior compiler for the MPC56xx microcontrollers version 2.8. The default compiler switches used are noted in the tables below.

7.1.1 Diab Compiler Switches

Option	Summary
-tPPCE200Z4VEF:simple	Acrhitecture: PowerPC VLE Processor: PPCE200Z4V Object Format: PowerPC ELF EABI Floating Point Mode: Single Hardware / Double Software Environment: simple (Only character I/O)
-@E+err.log	Redirect and append standard error output to err.log
-c	Stop after the assembly step and produce an object file with default file extension .o
-g	Generate symbolic debugger information
-Xenum-is-best	Use the smallest signed or unsigned integer type permitted by the range of values for an enumeration, that is, the first of signed char, unsigned char, short, unsigned short, int, unsigned int, long, or unsigned long sufficient to represent the values of the enumeration constants.
-Xlint	Generate warnings when suspicious and non-portable C code is encountered.
-Xkeywords=0x1f	If the option -Xkeywords=x is used with the least significant bit set in x (e.g., -Xkeywords=0x1), the compiler recognizes the keyword **extended** as a synonym for long double.
-Xnested-interrupts	In order to support nested interrupts, when entering an interrupt function (a function named in a #pragma interrupt directive or declared with the interrupt or __interrupt__ keywords), save the ssr0 and ssr1 registers, and restore them on returning from the function.
-Xlocals-on-stack	If the -Xlocals-on-stack option is given, only register variables are allocated to registers
-Xdebug-local-cie	Generate a local Common Information Entry (CIE) for

	each unit. This option, which eliminates the dependency on the debug library libg.a, is applicable only with DWARF 2 or DWARF 3 debug information.
-Xclib-optim-off	Direct the compiler to disregard all knowledge of ANSI C library functions. By default, the compiler automatically includes, before all other header files, the file lpragma.h, which contains pure_function, no_return, and no_side_effects pragmas and other statements that allow optimization of calls to C library functions.
-Xc-new	Compile using a compiler frontend derived from one produced by the Edison Design Group. By default, invoking -Xc-new also invokes -Xdialect-c99. Supported only with the :rtp execution environment.

7.1.2 Green Hills Compiler Switches

Option	Summary
-cpu=ppc564xl	Specifies code generation for a particular target processor.
-G	Generate MULTI debugging information
-vle	Enables VLE code generation and linkage with VLE libraries.
-w	Disables assembler warnings.
-floatsingle	Treats double types as float, so that no 64-bit instructions are required.
-Oslowcompile	Enables optimizations that improve compiling time
-Ospeed	Enables optimizations that improve both size and performance and that improve performance at the expense of size.
-Omax	Controls the aggressiveness with which the compiler will pursue optimizations, without regard for compile time.
-OI	Enables all the General Use optimizations together with two-pass inlining.
-OL	Enables loop optimizations for individual functions.
-Ounroll	Controls the Loop Unrolling optimization. This optimization duplicates the code in innermost loops to produce more straightline code.
-Ounrollbig	Controls the size of the loops that the compiler will consider for unrolling. Consider larger loops for unrolling.
-preprocess_assembly_files	Controls whether assembly files with standard extensions such as .s and .asm are preprocessed.
-g	Generates source-level debugging information.
-dwarf2	Enables the generation of DWARF debugging information in the object file
-noSPE	Disables the use of SPE and vector floating point instructions by the compiler, and C and C++ extensions to

	use the SPE (SPE PIM). -noSPE also passes -noSPE to the assembler

7.1.3 Code Warrior Compiler Switches

Option	Summary
`-gdwarf-2`	Generate DWARF 2.x debugging information
`-fp spfp_only`	Single precision hardware floating point / double precision software floating point emulation
`-char unsigned`	Chars are unsigned
`-proc Zen`	Set target processor to Zen
`-gccincludes`	Adopt GCC #include semantics: add '-I' paths to system list if '-I-' is not specified, and search directory of referencing file first for #includes.
`-nostdinc`	Use standard system include paths (specified by the environment variable %MWCIncludes%); added after all system '-I' paths.
`-nosyspath`	Treat #include <...> like #include "..."; always search both user and system path lists.
`-ansi off`	Allow non-standard keywords. Use the minimal-sized type for enumerations. ANSI strictness checking is off.
`-opt all`	Optimize for speed, perform peephole optimization, schedule instructions, auto-inline small functions (without 'inline' explicitly specified), align functions on 16 byte boundary.
`-Cpp_exceptions off`	Disable C++ exceptions.
`-nostdlib`	Don’t link against standard libraries by default.
`-srec`	Generate an S-record file.
`-bool on`	Use Boolean data type
`-spe_vector`	Use SPE instructions
`-vle`	Use VLE instructions instead of Book E

7.2 Memory Targets (LSM/DPM)

The toolbox provides linker command files with specific memory map setups for RAM based and Flash based targets. These targets only consider internal RAM and internal Flash as target environments. These both are user configurable by changing the provided template make files and the linker command files.

While the actual size of RAM on the 564xL is 128k, the RAM memory size supported by Motor Control Development Toolbox for Lock Step Mode (LSM) and Dual Parallel Mode (DPM) is only the first 64k, where the addressing is the same between the two modes. Additionally, in DPM only one core is used.

7.2.1 Flash Memory Target

The toolbox default FLASH memory target is configured as setup below.

GreenHills

Memory	Start Address	Length (Bytes)	Description
rcw	0x00000000	8	
init	0x00000020	8160	
int_flash	0x00002000	909312	
exception_handlers	0x000E0000	8192	
vector_table	0x000E2000	8192	
int_sram	0x40000000	63488	
stack_ram	0x4000F800	2048	

DIAB

Memory	Start Address	Length (Bytes)	Description
word	0x00000000	8	
vector_table	0x00002000	4096	
int_flash	0x00003000	53248	
exception_handlers	0x00010000	8192	
int_sram	0x40000000	63488	
stack_ram	0x4000F800	2048	

CodeWarrior

Memory	Start Address	Length (Bytes)	Description
word	0x00000000	8	
exception_handlers	0x00001000	4096	
vector_table	0x00002000	4096	
int_flash	0x00003000	53248	
int_sram	0x40000000	63488	
stack_ram	0x4000F800	2048	

7.2.2 RAM Memory Target

The toolbox default RAM memory target is configured as setup below.

GreenHills

Memory	Start Address	Length (Bytes)	Description
exception_handlers	0x40000000	8192	
vector_table	0x40002000	4096	
int_sram	0x40003000	51200	
stack_ram	0x4000F800	2048	

DIAB

Memory	Start Address	Length (Bytes)	Description
exception_handlers	0x40000000	8192	

vector_table	0x40002000	4096	
int_sram	0x40003000	47104	
stack_ram	0x4000F800	2048	

CodeWarrior

Memory	Start Address	Length (Bytes)	Description
init	0x40000000	4064	
exception_handlers	0x40001000	4096	
vector_table	0x40002000	4096	
int_flash	0x40003000	28672	
int_sram	0x4000A000	22528	
stack_ram	0x4000F800	2048	

7.2.3 BAM RAM Memory Target (Bootloader)

The toolbox default Boot Assist Module (BAM) RAM memory target is configured as setup below.

GreenHills

Memory	Start Address	Length (Bytes)	Description
exception_handlers	0x40005000	8192	
vector_table	0x40007000	4096	
int_sram	0x40008000	30720	
stack_ram	0x4000F800	2048	

DIAB

Memory	Start Address	Length (Bytes)	Description
exception_handlers	0x40005000	8192	
vector_table	0x40007000	4096	
int_sram	0x40008000	30720	
stack_ram	0x4000F800	2048	

CodeWarrior

Memory	Start Address	Length (Bytes)	Description
init	0x40005000	4064	
exception_handlers	0x40006000	4096	
vector_table	0x40007000	4096	
int_flash	0x40008000	20480	
int_sram	0x4000D000	10240	
stack_ram	0x4000F800	2048	

7.3 User-Specific Files Needed for Build

There may be instances where variables are used from code that is not part of the model but are instead hand coded. In these instances, user specific files will need to be

included in the build directory created by toolbox. The user can specify which files to include in the build directory by creating/modifying a file called rappid_564xl_user_copy_required_files.m and locate it somewhere in one of the Matlab paths. There is an example of this file located in the 'mscripts' directory that demonstrates how to include user specific files.

8 FreeMASTER Interface

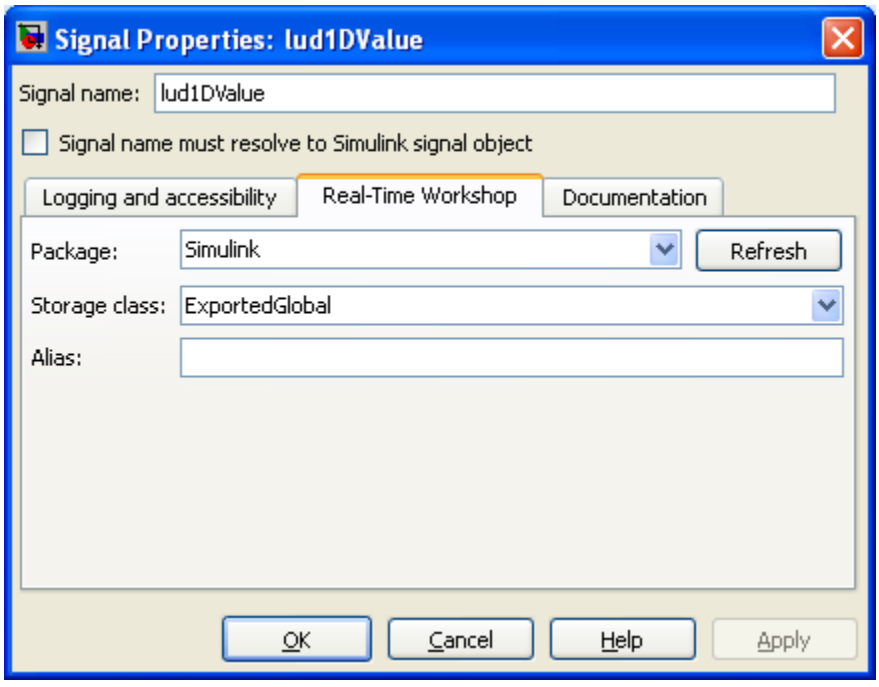
The FreeMASTER is a real-time monitor pc-based application that connects to Your embedded target through various communication links.

The FreeMASTER Interface works with systems that have been deemed to generate function code. It incorporates FreeMASTER serial communication protocol files targeted for MPC56xx family platform. The FreeMASTER function provides communication interfaces b/w the Target and the host PC using FreeMASTER protocol.

Either SCI or CAN communication links can be used with FreeMASTER. Currently, both the Bootloader and the PIL mode use the SCI communication link. Due to the limitations of the SCI link, FreeMASTER cannot be used on SCI at the same time as either Bootloader or PIL. Additionally, many applications use the CAN link for node to node communications, making the CAN link more generally supported throughout the development cycle. In the future, the Bootloader will support CAN communications. Multiple interfaces can use the CAN link concurrently. For these reasons, the CAN communication link is the recommended connection for FreeMASTER.

To configure your Motor Control Development Toolbox enhanced Simulink model for FreeMASTER, complete the following steps:

1. Identify the signals to monitor with FreeMASTER and provide them with a unique name.
2. In Signal Properties, Real-Time Workshop tab:
    a. Package: ‘Simulink’
    b. Storage class: ‘ExportedGlobal’



3. In RAppID_564xL_Config block, FreemasterSetup tab
    a. Freemaster Interface: ‘Serial’
    b. Freemaster Interface Rate: <select your preferred rate from the list>
    c. Remaining Serial interface settings are not user-selectable

    Or

    d. Freemaster Interface: ‘CAN’
    e. Remaining CAN interface settings are not user-selectable
    f.

For more information on FreeMASTER, see *pcm_um.pdf*.

9 Bootloader

The RAppID Boot Loader works with several Freescale Qorivva & PX series family of parts. The Boot Loader supports the operational modes of the different part family silicon instantiations. The Boot Loader supports the operational modes of the different part family BAM silicon instantiations. The Boot Loader provides a streamlined method for programming code into FLASH or RAM on either target EVBs or custom boards. Once programming is complete the application code automatically starts. For more information, refer to the Boot Loader User Manual *RAppidBL_UserManual.pdf*

10 PIL/SIL Operational Mode

Motor Control Development Toolbox provides support for both Processor-In-the-Loop (PIL) and Software-In-the-Loop (SIL) simulations. PIL simulation allows execution of the production source code on the target. SIL simulation provides the same benefit of verifying the production source code, but without the hardware.

10.1 PIL & SIL Automatic Configuration

Motor Control Development Toolbox provides via the Tools pull-down menu for easy transition between simulation modes as well as generation of PIL & SIL blocks.

There is support for the following simulation mode transitions:

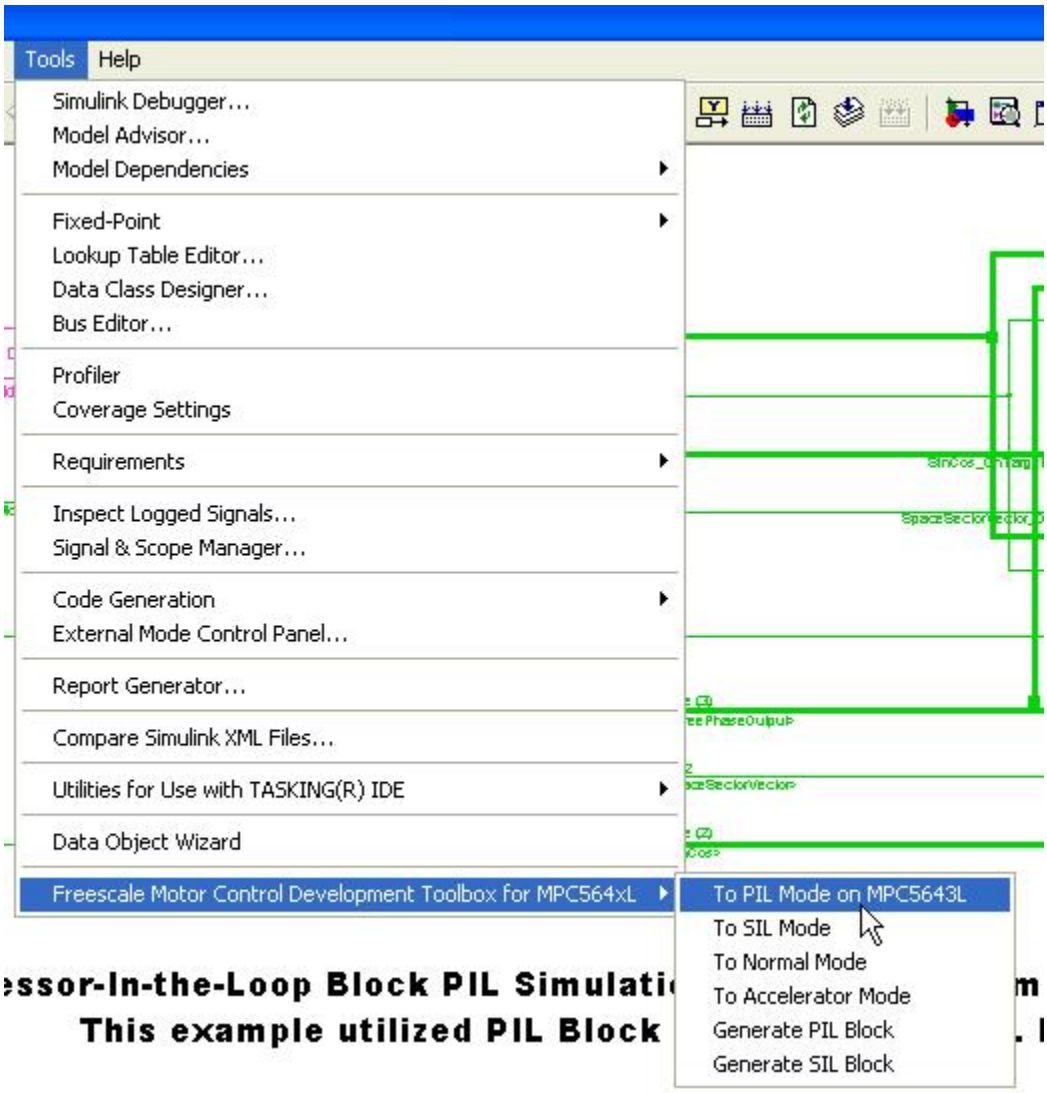
1) To PIL Mode on MPC5643L
2) To SIL Mode
3) To Normal Mode
4) To Accelerator Mode

Note: These four options only support the Model Block PIL mode and can only be executed from the model containing the model reference block.

There is support for the following Build Model actions:

1) Generate PIL Block
2) Generate SIL Block

Note: These menu items are only supported for Block PIL and this needs to be used from the model that contains the logic to be used in PIL mode.



Note: In addition to executing the script, it is necessary to set the Processor-In-Loop option in the RAppID_564xL_Config block settings when transitioning to PIL Mode.

10.2 PIL & SIL Manual Configuration

Alternatively, it is possible to change the settings manually. The following table identifies the additional settings required to transition to PIL & SIL modes (these settings are ignored in Normal and Accelerator Modes). These settings apply regardless of whether you are using PIL/SIL block or Model block (using model-reference) PIL/SIL method:

	PIL	SIL
'Configuration Parameters'		
'Code Generation' - 'System target file'	'rappid564xl.tlc'	'ert.tlc'
'Code Generation' – 'Interface' – 'absolute time'	off	Off
'Code Generation' – 'Symbols' – 'Maximum identifier length'	63	63
'Hardware Implementation' – 'Device Vendor'	'Freescale'	'Generic'
'Hardware Implementation' – 'Device type'	'32-bit PowerPC'	'Custom'
'Hardware Implementation' – 'Byte ordering'	'BigEndian'	'BigEndian'
'Hardware Implementation' – 'Signed integer division rounds'	'Zero'	'Zero'
Set 'ModelReferenceCompliant' in Command Window: set_param(<model_name>, 'ModelReferenceCompliant', 'on');	on	On
RAppID_564xL_Config block		
Processor-In-Loop	on	Off

Next, perform the following steps depending on which method you have chosen:

- PIL/SIL block
    1. Set from Tools pulldown menu Code Generation -> Options -> SIL and PIL Verification section -> Create block to either SIL or PIL.
    2. Build Model

- Model block PIL/SIL
    1. Change the Simulation Mode in ModelReference Parameters menu to PIL or SIL.

Now you are ready to simulate your PIL/SIL model.

Note: PIL block & SIL block are supported in all versions of MATLAB currently supported by Motor Control Development Toolbox.