

Analyzing Code Generated from a Simulink* Model

Jim Freudenberg

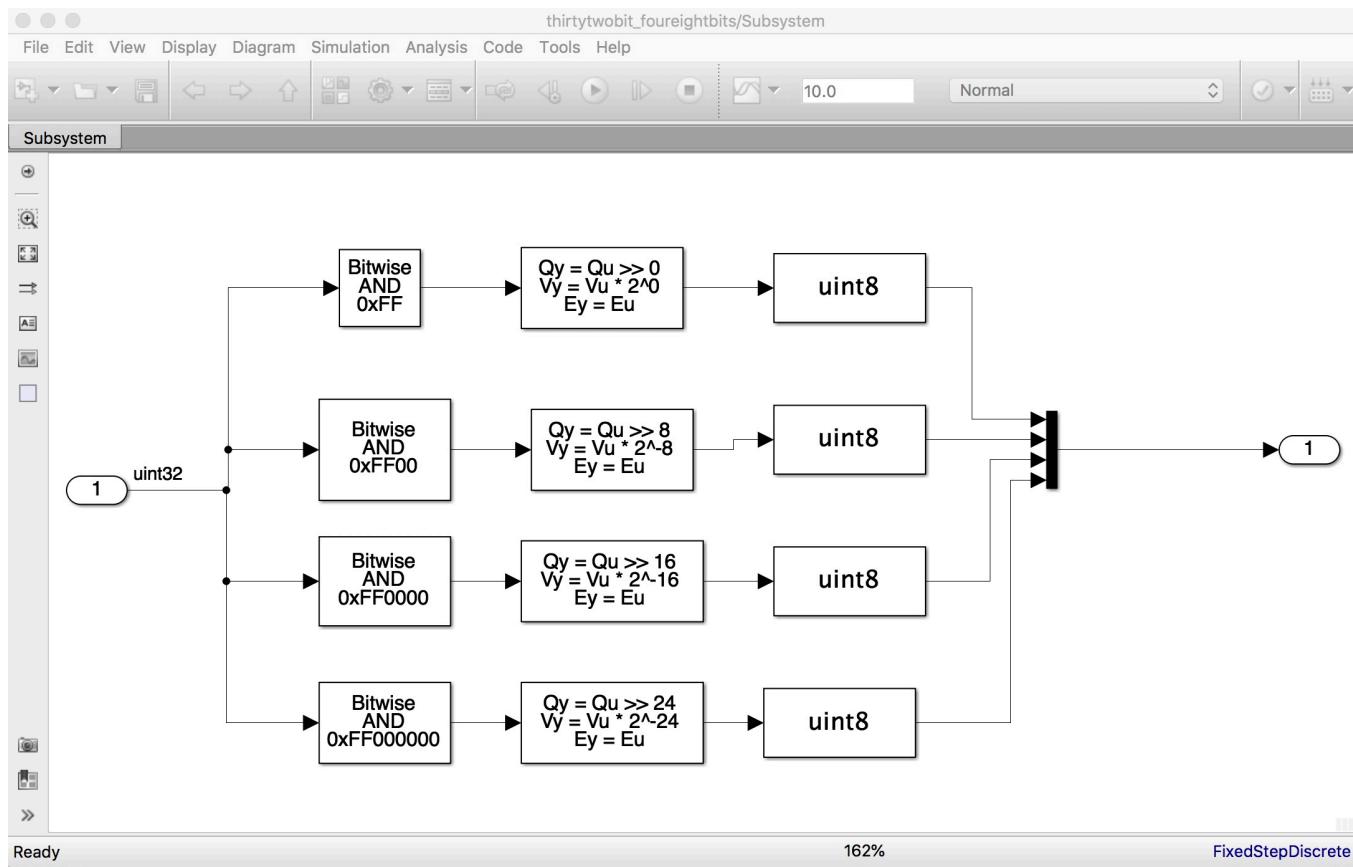
University of Michigan

EECS 461: Embedded Control Systems

Introduction

- This set of slides will show you how generate C code from a simple Simulink model, and analyze the generated code using Embedded Coder utilities.
- In particular, we will use the **traceability** feature to move back and forth from the Simulink blocks to the code generated from them.
- We will use Matlab 2018a, which is the version of Matlab used in the EECS461 lab.
- The slides were made with a Mac – the menus on a PC may look a little different.

Convert 32-bit Integer to 8-bit Integers



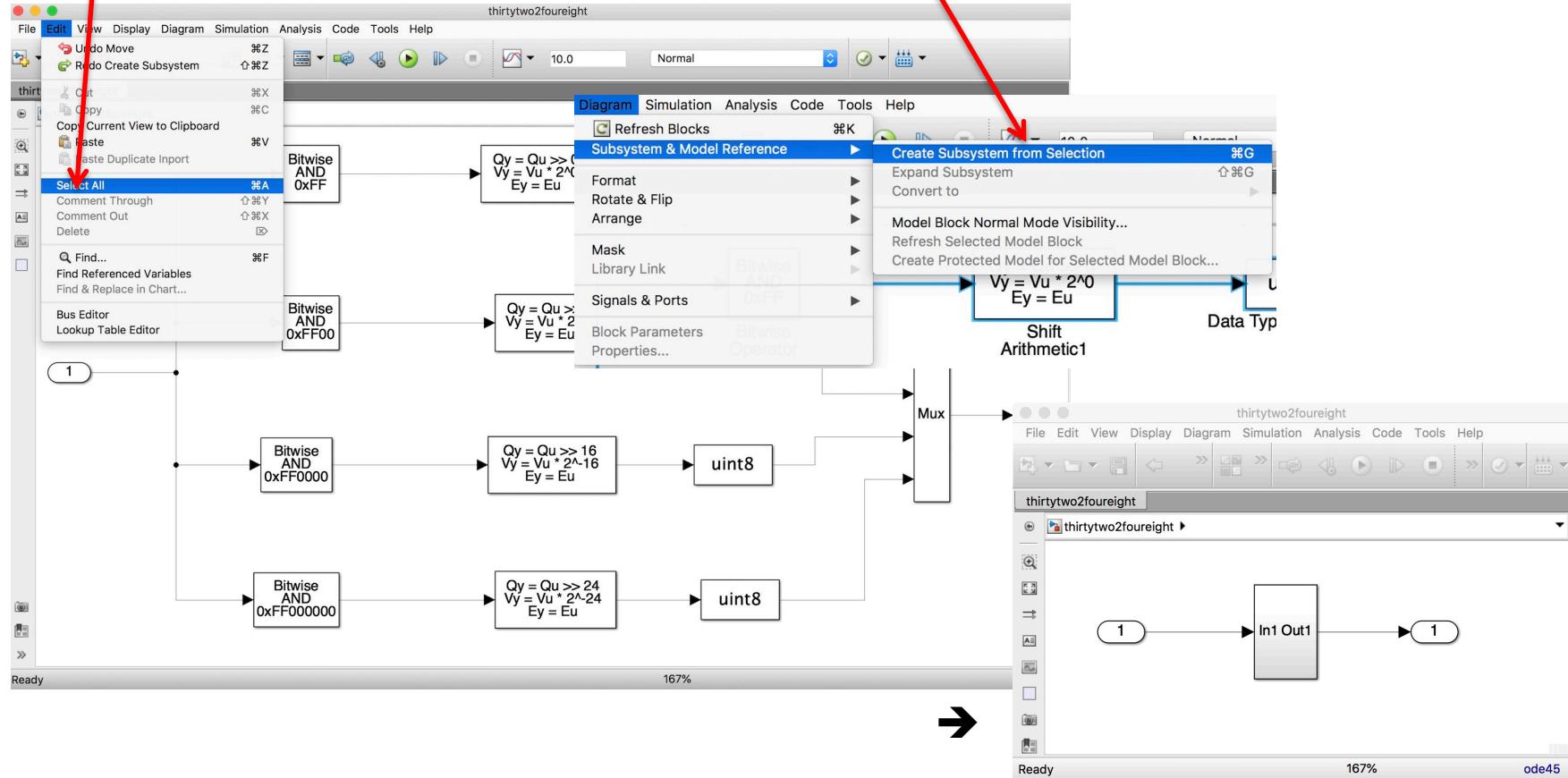
Create a Simulink model of the system depicted in Figure 3 of the handout "Simulink Models for Autocode Generation". This model takes one 32-bit integer as input and breaks it into four 8-bit integers.

Make a Subsystem

From the Edit and Diagram menus:

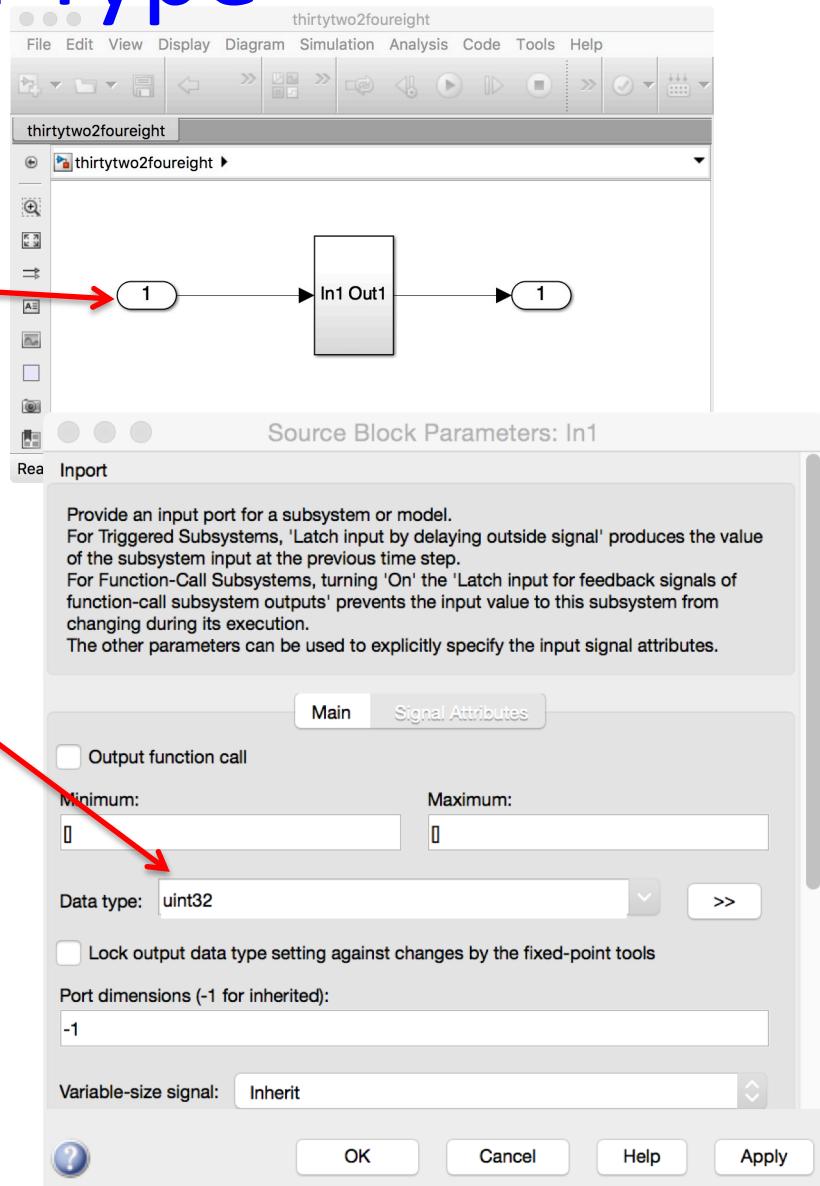
Edit/Select All

Diagram/Subsystem & Model Reference/Create Subsystem from Selection

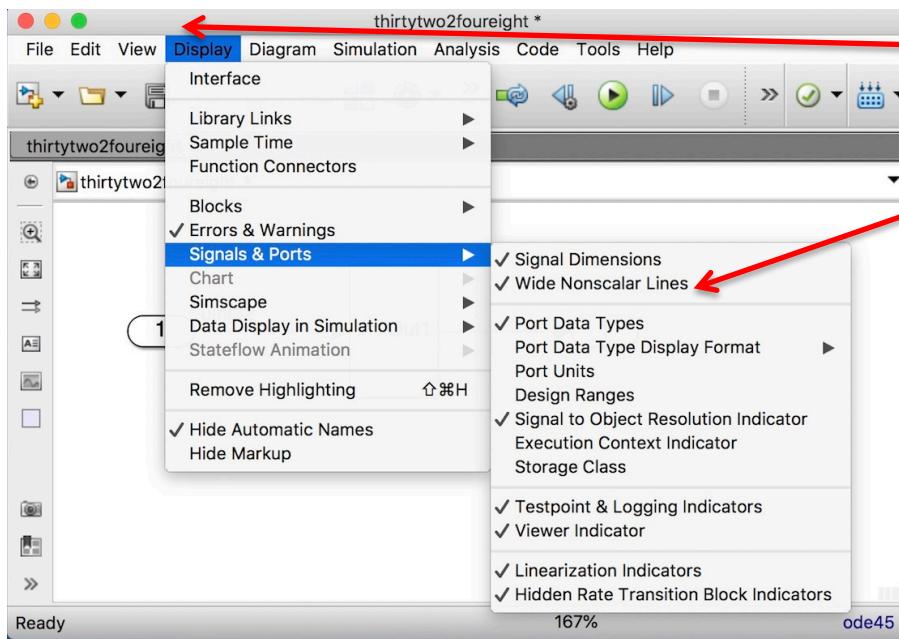


Input Data Type

- Double-click Input Port
- Set data type to `uint32` in the Signal Attributes Menu



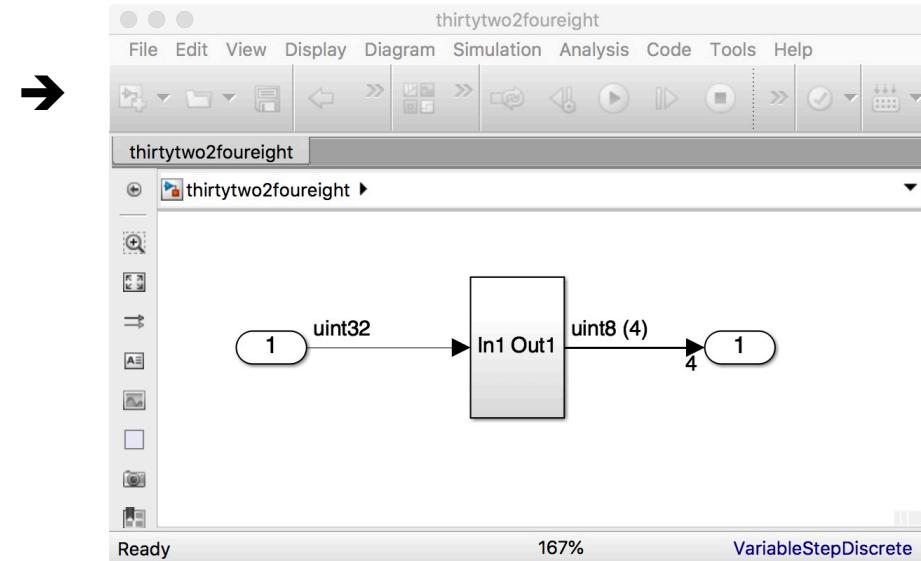
Display Data Types



- Under the Display Menu, select

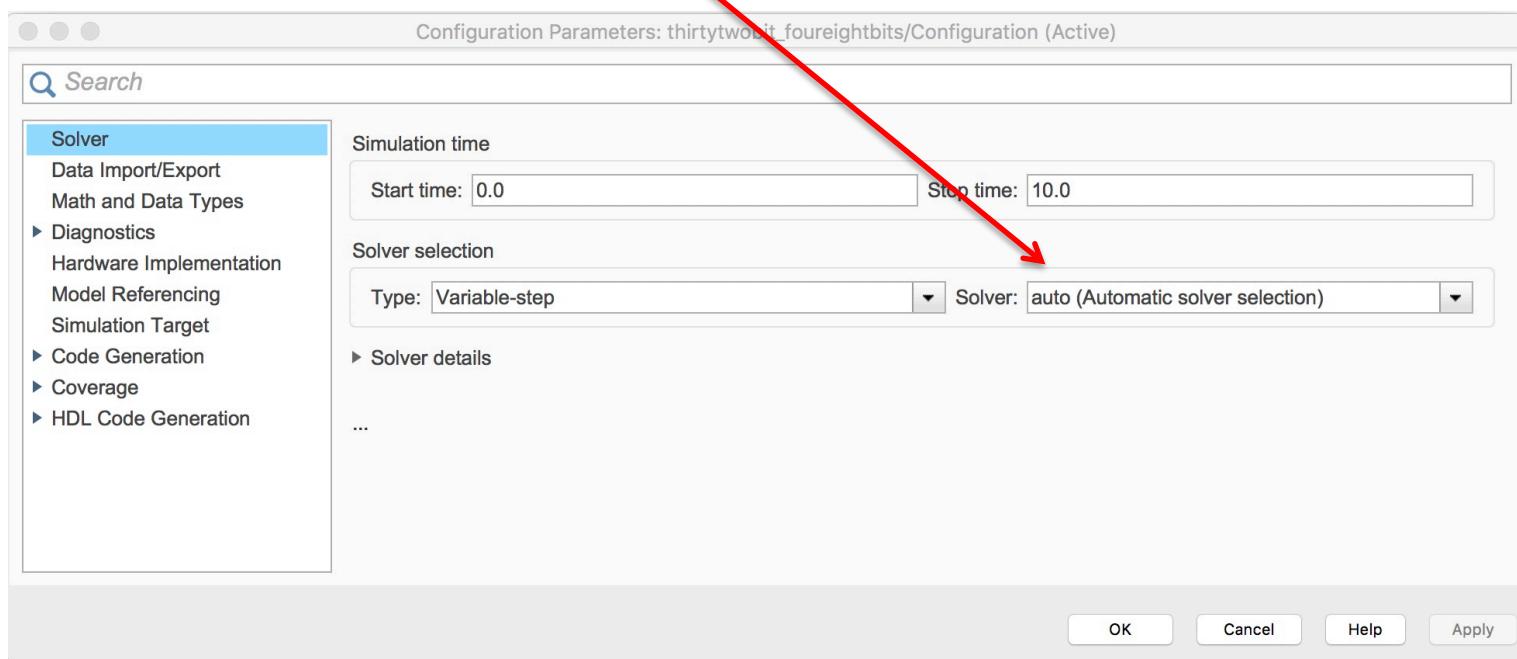
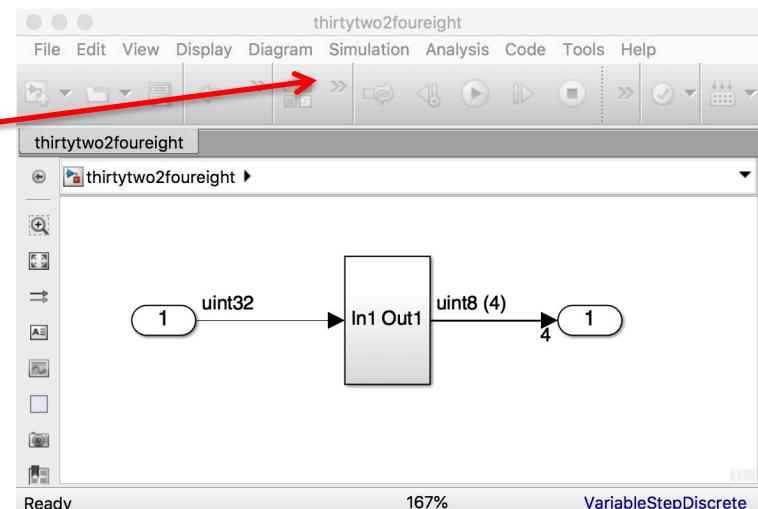
Display/Signals & Ports/Signal Dimensions
Display/Signals & Ports/Wide Nonscalar Lines
Display/Signals & Ports/Port Data Types

- You may need to run the model in order to see the update



Solver Options

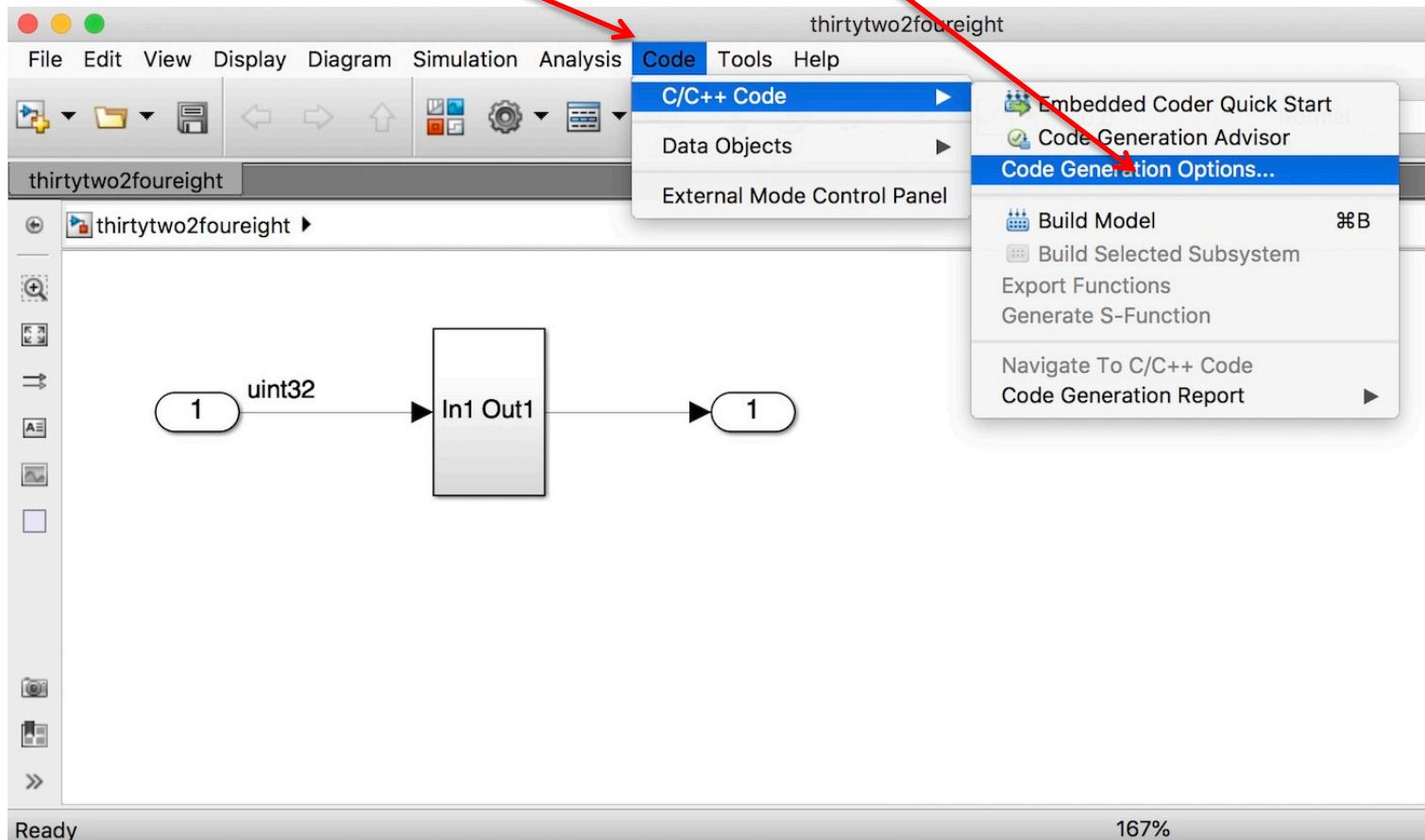
In the
Simulation/Model Configuration Parameters/Solver
Menu, select “Fixed Step” solver and
“discrete (no continuous states)”



Options for Code Generation

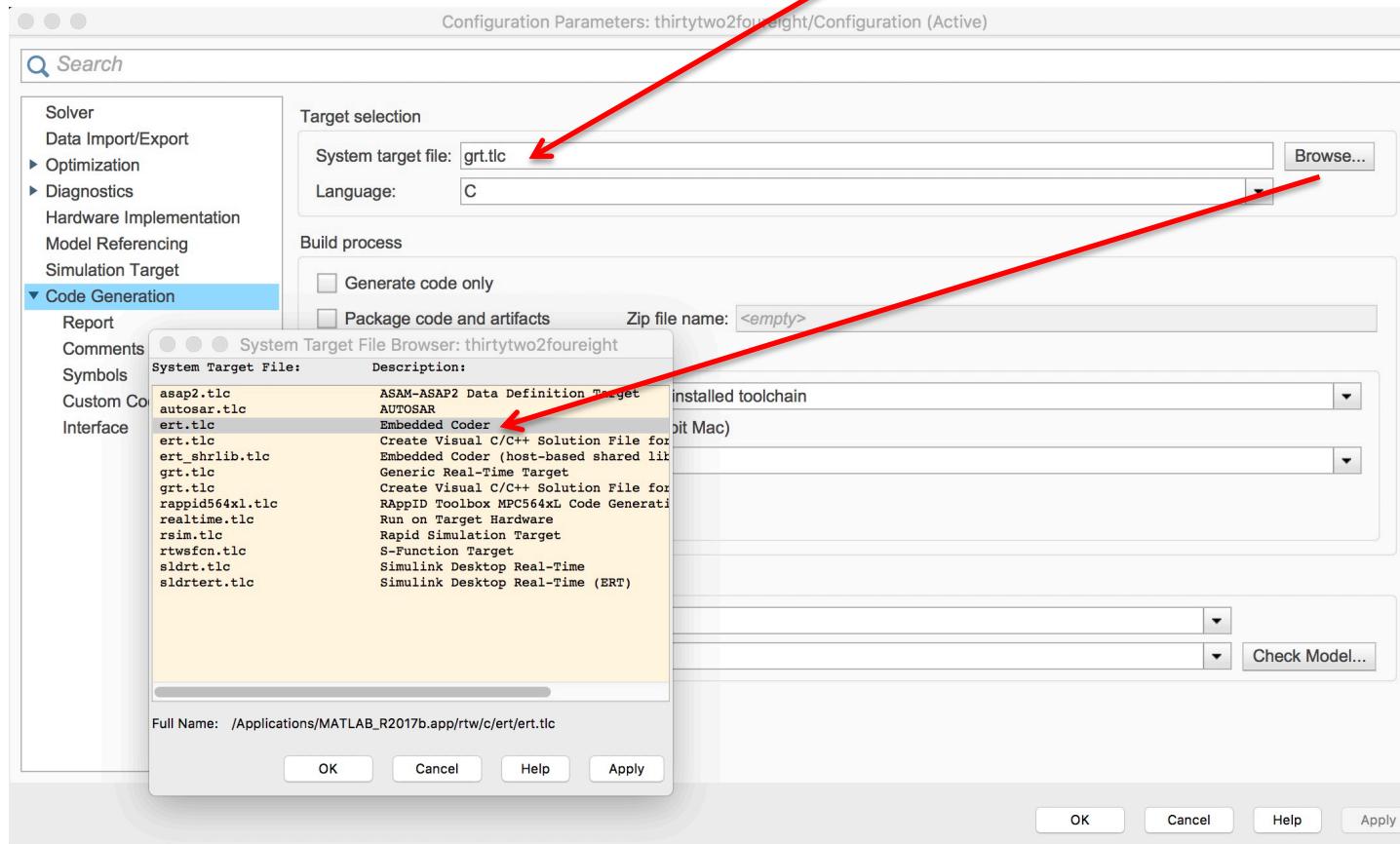
From the Code Menu, select:

Code/C/C++ Code/Code Generation Options...



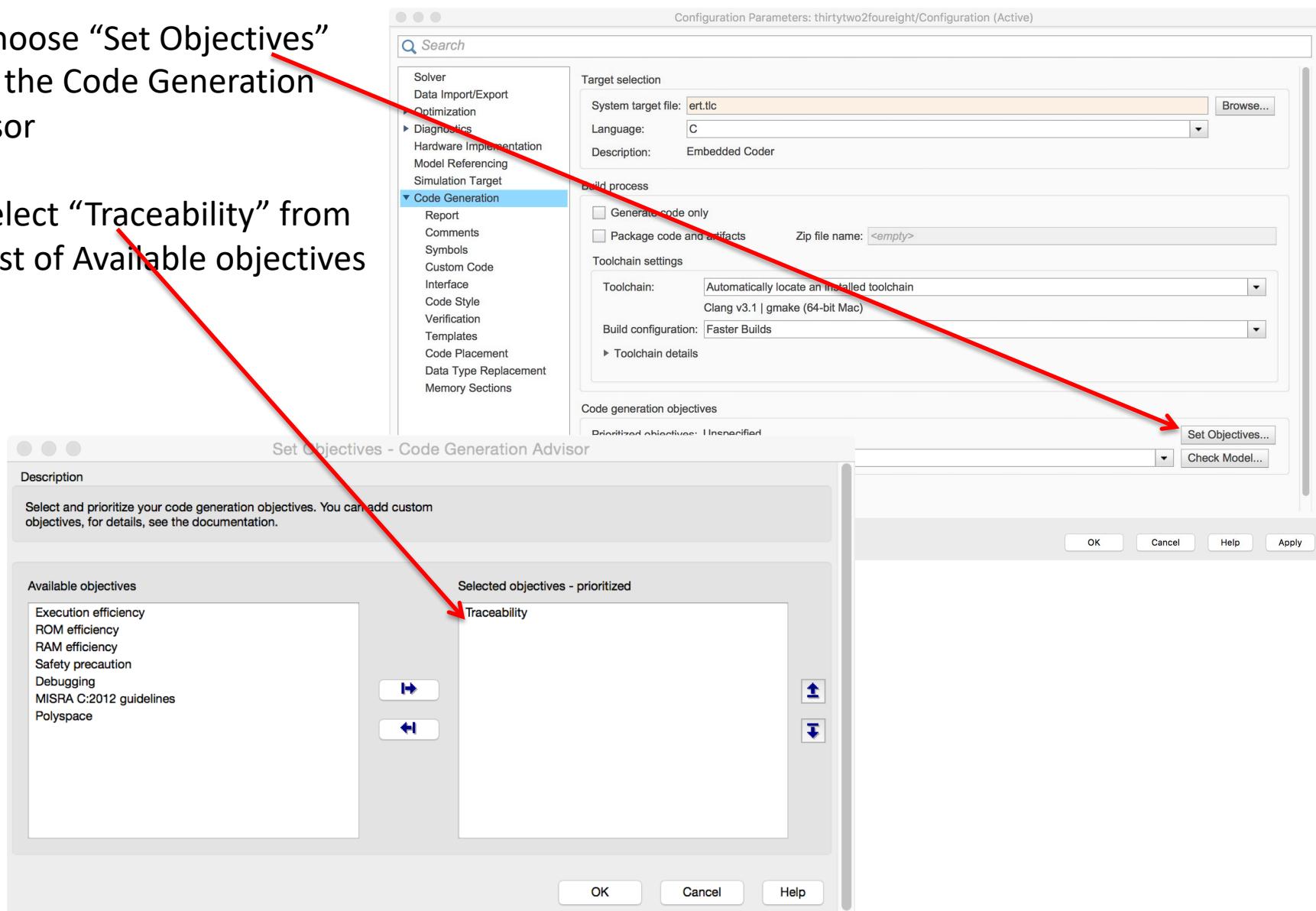
Options Window

The default target option is “Generic Real-Time Target”, grt.tlc
For more efficient code and additional features, Browse to select
“Embedded Coder”, ert.tlc



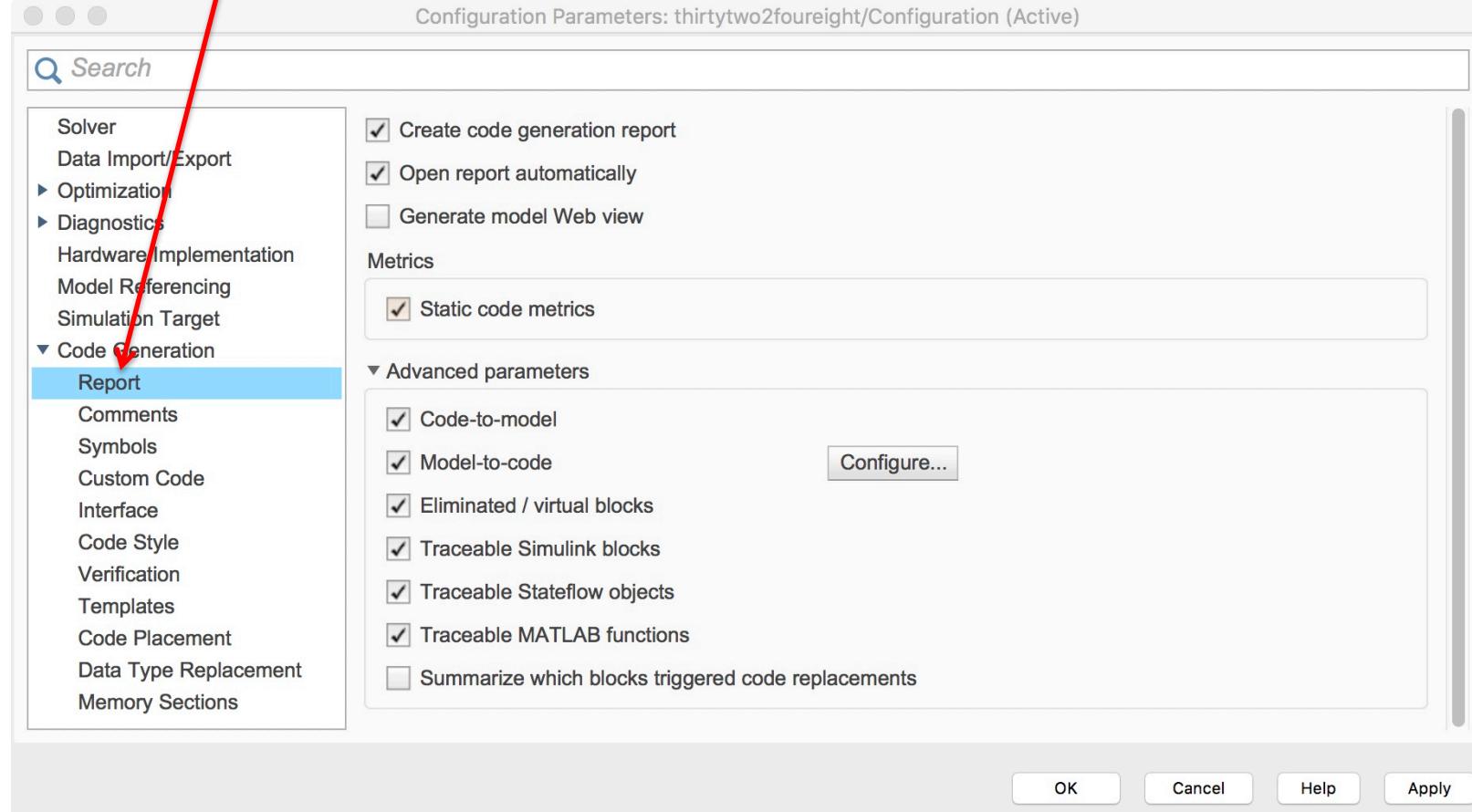
Embedded Coder Options

- Choose “Set Objectives” from the Code Generation Advisor
- Select “Traceability” from the list of Available objectives



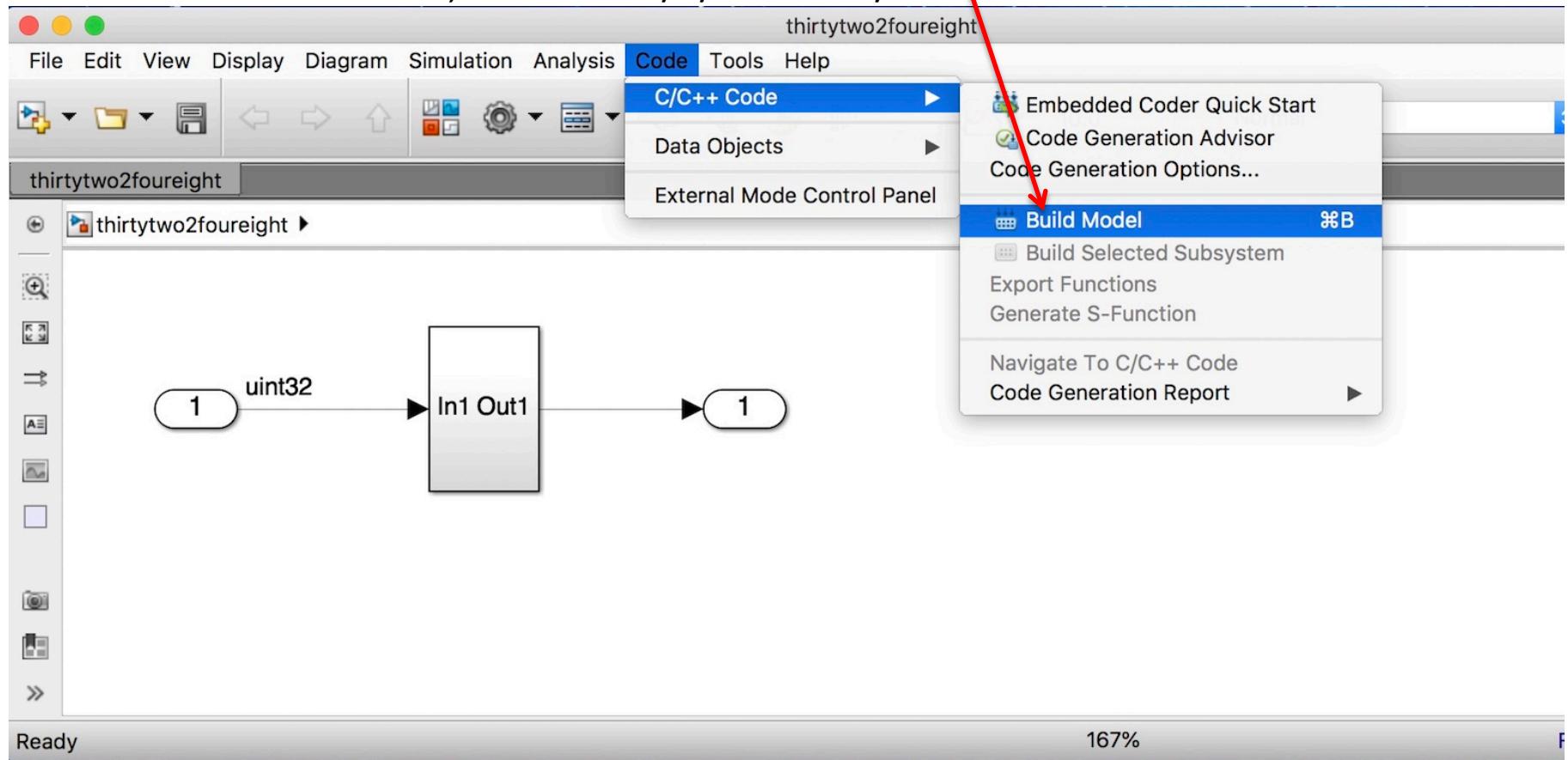
Report Options

Select Code Generation/Report and check the indicated options.



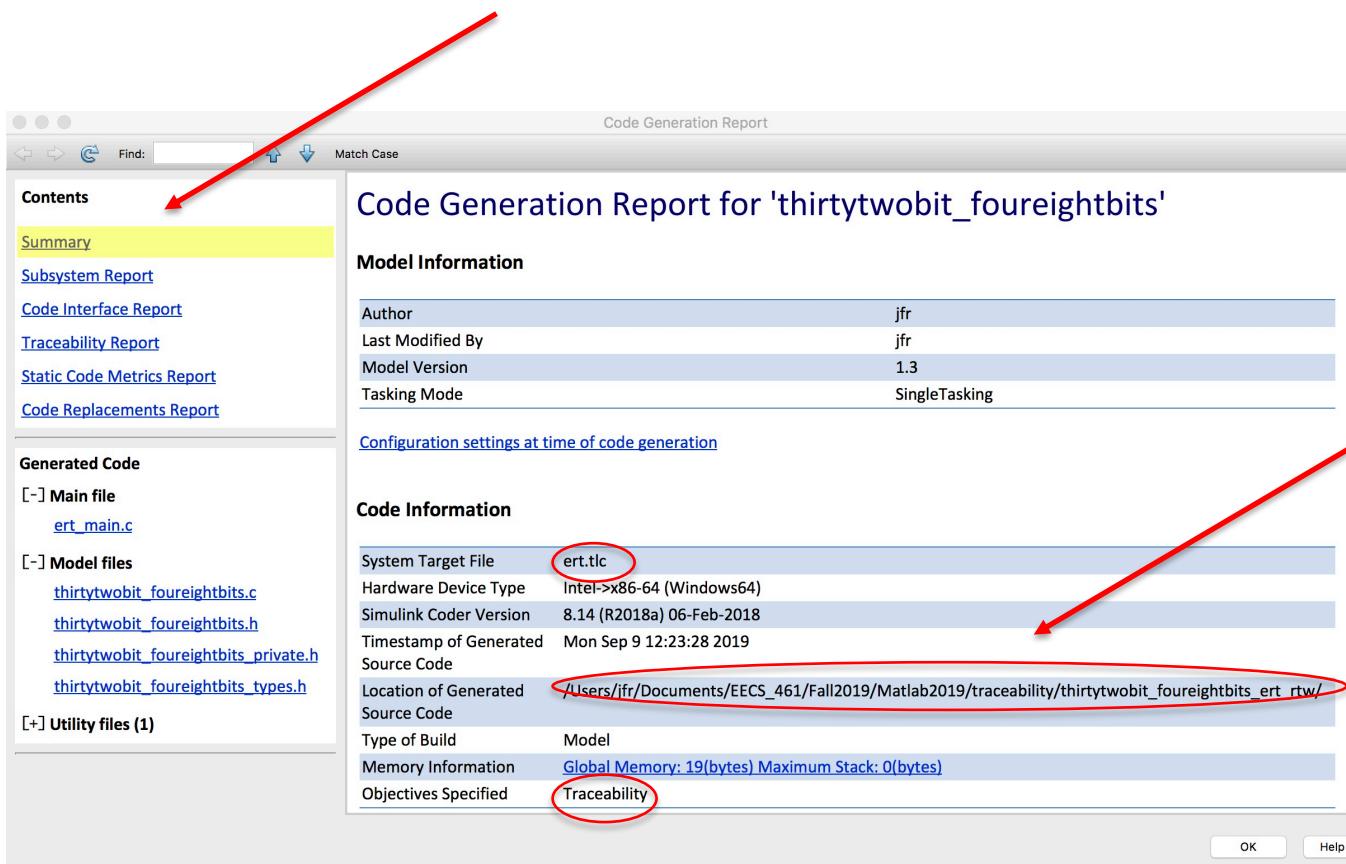
Generate C Code

- To generate C code, we must “Build” the model
- **Be sure the Matlab working directory is the same directory that contains the Simulink model!!!!**
- Under the Code Menu, select Code/C/C++ Code/Build Model



Code Generation Report

- If the Build is successful, a Code Generation Report window will open
- Note the many menus - we will take a look at some of these.



- Information about the model settings

Code Interface Report

Describes the generated functions, their arguments, and how often they are executed

Code Generation Report

Find: Match Case

Contents

- Summary
- Subsystem Report
- Code Interface Report**
- Traceability Report
- Static Code Metrics Report
- Code Replacements Report

Generated Code

- [-] Main file
 - ert_main.c
- [-] Model files
 - thirtytwobit_foureightbits.c
 - thirtytwobit_foureightbits.h
 - thirtytwobit_foureightbits_priv.h
 - thirtytwobit_foureightbits_types.h
- [+] Utility files (1)

Code Interface Report for thirtytwobit_foureightbits

Table of Contents

- [Entry Point Functions](#)
- [Imports](#)
- [Outports](#)
- [Interface Parameters](#)
- [Data Stores](#)

Entry Point Functions

Function: thirtytwobit_foureightbits_initialize

Prototype	void thirtytwobit_foureightbits_initialize(void)
Description	Initialization entry point of generated code
Timing	Must be called exactly once
Arguments	None
Return value	None
Header file	<u>thirtytwobit_foureightbits.h</u>

Function: thirtytwobit_foureightbits_step

Prototype	void thirtytwobit_foureightbits_step(void)
Description	Output entry point of generated code
Timing	Must be called periodically, every 0.2 seconds
Arguments	None
Return value	None
Header file	<u>thirtytwobit_foureightbits.h</u>

OK Help

Q: why does the function `thirtytwo_bit_foureightbits_step` execute every 0.2 seconds?

Static Code Metrics Report

- Number of lines of code in the generated .c and .h files
- Total lines of code, including comments

Code Generation Report

Find: Match Case

Contents

- [Summary](#)
- [Subsystem Report](#)
- [Code Interface Report](#)
- [Traceability Report](#)
- [Static Code Metrics Report](#)
- [Code Replacements Report](#)

Generated Code

[–] **Main file**
[ert_main.c](#)

[–] **Model files**
[thirtytwobit_foureightbits.c](#)
[thirtytwobit_foureightbits.h](#)
[thirtytwobit_foureightbits_private.h](#)
[thirtytwobit_foureightbits_types.h](#)

[+] **Utility files (1)**

Static Code Metrics Report

The static code metrics report provides statistics of the generated code. Metrics are estimated from static analysis of the generated code using the C data types specified in the 'Device details' section of the **Configuration Parameter > Hardware Implementation** pane: **char 8, short 16, int 32, long 32, float 32, double 64, pointer 64** bits. If your model contains a Variant block, the Static Code Metrics Report does not contain data for the inactive variant. Actual object code metrics might differ due to target specific compiler and platform settings. Consult the **Code Generation Advisor** for options to improve code efficiency.

Table of Contents

- [File Information](#)
- [Global Variables](#)
- [Function Information](#)

1. File Information [hide]

[–] Summary (excludes ert_main.c)

Number of .c files : 1
Number of .h files : 4
Lines of code : 157
Lines : 428

[–] File details

File Name	Lines of Code	Lines	Generated On
rtwtypes.h	81	161	09/09/2019 12:23 PM
thirtytwobit_foureightbits.h	37	109	09/09/2019 12:23 PM
thirtytwobit_foureightbits.c	31	95	09/09/2019 12:23 PM
thirtytwobit_foureightbits_private.h	4	30	09/09/2019 12:23 PM
thirtytwobit_foureightbits_types.h	4	33	09/09/2019 12:23 PM

OK Help

Function Definitions

- Function definitions are found in `thirtytwobitto8bit.c`
- The function `thirtytwobitto8bit_step` performs the desired bit manipulations
- C code for bit masking and shifting
- Simulink blocks implemented in C code
- Note the block `<S1>/Shift Arithmetic` is not on the list

Code Generation Report

Find: Match Case

Contents

Summary

Subsystem Report

Code Interface Report

Traceability Report

Static Code Metrics Report

Code Replacements Report

Generated Code

[-] Main file

[ert_main.c](#)

[-] Model files

[thirtytwobit_foureightbits.c](#)

[thirtytwobit_foureightbits.h](#)

[thirtytwobit_foureightbits_private.h](#)

[thirtytwobit_foureightbits_types.h](#)

[+] Utility files (1)

```
34 RT_MODEL_thirtytwobit_foureig_T *const thirtytwobit_foureightbits_M =
35   &thirtytwobit_foureightbits_M_;
36
37 /* Model step function */
38 void thirtytwobit_foureightbits_step(void)
39 {
40   /* Outport: '<Root>/Out1' incorporates:
41    * ArithShift: '<S1>/Shift Arithmetic1'
42    * ArithShift: '<S1>/Shift Arithmetic2'
43    * ArithShift: '<S1>/Shift Arithmetic3'
44    * DataTypeConversion: '<S1>/Data Type Conversion'
45    * DataTypeConversion: '<S1>/Data Type Conversion1'
46    * DataTypeConversion: '<S1>/Data Type Conversion2'
47    * DataTypeConversion: '<S1>/Data Type Conversion3'
48    * Inport: '<Root>/In1'
49    * S-Function (sfix_bitop): '<S1>/Bitwise Operator'
50    * S-Function (sfix_bitop): '<S1>/Bitwise Operator1'
51    * S-Function (sfix_bitop): '<S1>/Bitwise Operator2'
52    * S-Function (sfix_bitop): '<S1>/Bitwise Operator3'
53   */
54   thirtytwobit_foureightbits_Y.Out1[0] = (uint8_T)
55     ((thirtytwobit_foureightbits_U.In1 & 255U));
56   thirtytwobit_foureightbits_Y.Out1[1] = (uint8_T)
57     (((thirtytwobit_foureightbits_U.In1 & 65280U) >> 8));
58   thirtytwobit_foureightbits_Y.Out1[2] = (uint8_T)
59     (((thirtytwobit_foureightbits_U.In1 & 16711680U) >> 16));
60   thirtytwobit_foureightbits_Y.Out1[3] = (uint8_T)
61     (((thirtytwobit_foureightbits_U.In1 & 4278190080U) >> 24));
62 }
63 }
```

OK Help

Traceability Report: Eliminated Blocks

Certain Simulink blocks are used only to improve model readability, and do not affect the generated code. These are eliminated (in this case, the subsystem block, Mux block, and redundant input and output port blocks)

The screenshot shows the 'Code Generation Report' window with the following details:

- Contents:** Summary, Subsystem Report, Code Interface Report, **Traceability Report** (highlighted in yellow), Static Code Metrics Report, Code Replacements Report.
- Generated Code:** [-] Main file: ert_main.c; [-] Model files: thirtytwobit_foureightbits.c, thirtytwobit_foureightbits.h, thirtytwobit_foureightbits_private.h
- Traceability Report for thirtytwobit_foureightbits:**
 - Table of Contents:** 1. Eliminated / Virtual Blocks, 2. Traceable Simulink Blocks / Stateflow Objects / MATLAB Functions
 - thirtytwobit_foureightbits
 - thirtytwobit_foureightbits/Subsystem
 - Eliminated / Virtual Blocks:**

Block Name	Comment
<Root>/Subsystem	Virtual SubSystem
<S1>/In1	Inport
<S1>/Mux	Mux
<S1>/Shift Arithmetic	Eliminated trivial shift
<S1>/Out1	Outport
 - Traceable Simulink Blocks / Stateflow Objects / MATLAB Functions**

Why is the Shift Arithmetic block eliminated?

<Root>: Top Level Simulink diagram

<S1>: Subsystem that contains the rest of the Simulink blocks

Traceability Report: Blocks to Code

Simulink block name

Lines of code in associated .c and .h files

The screenshot shows a 'Code Generation Report' window. On the left, a sidebar lists report types: 'Contents' (Summary, Subsystem Report, Code Interface Report, **Traceability Report**, Static Code Metrics Report, Code Replacements Report), 'Generated Code' (Main file: `ert_main.c`, Model files: `thirtytwobit_foureightbits.c`, `thirtytwobit_foureightbits.h`, `thirtytwobit_foureightbits_private.h`, `thirtytwobit_foureightbits_types.h`, Utility files (1)), and a '+' sign for Utility files (1). Two red arrows point from the 'Simulink block name' and 'Lines of code in associated .c and .h files' text labels to the 'Object Name' and 'Code Location' columns respectively in the main table. The main table has two sections: 'Object Name' and 'Code Location' for the root blocks, and 'Object Name' and 'Code Location' for the subsystem 'thirtytwobit_foureightbits/Subsystem'. The subsystem section lists 15 objects and their corresponding code locations.

Object Name	Code Location
<code><Root>/In1</code>	thirtytwobit_foureightbits.c:48, 55, 57, 59, 61 thirtytwobit_foureightbits.h:49
<code><Root>/Out1</code>	thirtytwobit_foureightbits.c:40, 54, 56, 58, 60 thirtytwobit_foureightbits.h:54
Subsystem: thirtytwobit_foureightbits/Subsystem	
Object Name	Code Location
<code><S1>/Bitwise Operator</code>	thirtytwobit_foureightbits.c:49, 55
<code><S1>/Bitwise Operator1</code>	thirtytwobit_foureightbits.c:50, 57
<code><S1>/Bitwise Operator2</code>	thirtytwobit_foureightbits.c:51, 59
<code><S1>/Bitwise Operator3</code>	thirtytwobit_foureightbits.c:52, 61
<code><S1>/Data Type Conversion</code>	thirtytwobit_foureightbits.c:44, 54
<code><S1>/Data Type Conversion1</code>	thirtytwobit_foureightbits.c:45, 56
<code><S1>/Data Type Conversion2</code>	thirtytwobit_foureightbits.c:46, 58
<code><S1>/Data Type Conversion3</code>	thirtytwobit_foureightbits.c:47, 60
<code><S1>/Shift Arithmetic1</code>	thirtytwobit_foureightbits.c:41, 57 thirtytwobit_foureightbits.h:42
<code><S1>/Shift Arithmetic2</code>	thirtytwobit_foureightbits.c:42, 59 thirtytwobit_foureightbits.h:43
<code><S1>/Shift Arithmetic3</code>	thirtytwobit_foureightbits.c:43, 61 thirtytwobit_foureightbits.h:44

Q: why are there only lines of code for three `<S1>/Shift Arithmetic` blocks in the file `thirtytwobitfoureightbits.c`?

Traceability

Click on block name `< S1>/Shift Arithmetic`
Corresponding Simulink block is highlighted

Code Generation Report

Contents

- [Summary](#)
- [Subsystem Report](#)
- [Code Interface Report](#)
- [Traceability Report](#)
- [Static Code Metrics Report](#)
- [Code Replacements Report](#)

Generated Code

- [\[-\] Main file](#)
 - [ert_main.c](#)
- [\[-\] Model files](#)
 - [thirtytwobit_foureightbits.c](#)
 - [thirtytwobit_foureightbits.h](#)
 - [thirtytwobit_foureightbits_priv](#)
 - [thirtytwobit_foureightbits_type](#)
- [\[+\] Utility files \(1\)](#)

Object Name

Object Name	Code Location
<code><Root>/In1</code>	<code>thirtytwobit_foureightbits.c:48, 55, 57, 59, 61</code>
<code><Root>/Out1</code>	<code>thirtytwobit_foureightbits.c:48, 55, 57, 59, 61</code>
<code><Root>/Out2</code>	<code>thirtytwobit_foureightbits.c:48, 55, 57, 59, 61</code>
<code><Root>/Out3</code>	<code>thirtytwobit_foureightbits.c:48, 55, 57, 59, 61</code>
<code><Root>/Out4</code>	<code>thirtytwobit_foureightbits.c:48, 55, 57, 59, 61</code>

Code Location

Subsystem: `thirtytwobit_foureightbits/Subsystem`

Object Name

- `<S1>/Bitwise Operator`
- `<S1>/Bitwise Operator1`
- `<S1>/Bitwise Operator2`
- `<S1>/Bitwise Operator3`
- `<S1>/Data Type Conversion`
- `<S1>/Data Type Conversion1`
- `<S1>/Data Type Conversion2`
- `<S1>/Data Type Conversion3`
- `<S1>/Shift Arithmetic1`
- `<S1>/Shift Arithmetic2`
- `<S1>/Shift Arithmetic3`

Code Location

File Edit View Display Diagram Simulation Analysis Code Tools Help

Normal 10.0

Subsystem

Block diagram details:
Input: uint32 (1)
Output: uint8(4) (1)
Blocks:

- Bitwise AND 0xFF
- Bitwise AND 0xFF00
- Bitwise AND 0xFF0000
- Bitwise AND 0xFF000000
- Shift Arithmetic (highlighted in blue):
 - Qy = Qu >> 0
Vy = Vu * 2^0
Ey = Eu
 - Qy = Qu >> 8
Vy = Vu * 2^8
Ey = Eu
 - Qy = Qu >> 16
Vy = Vu * 2^16
Ey = Eu
 - Qy = Qu >> 24
Vy = Vu * 2^24
Ey = Eu
- Conversion to uint8
- Summing junction

Ready View diagnostics 162% FixedStepDiscrete

OK Help

Traceability: .c file

- Click on .c filename name next to Simulink block name
- Location of C code for that block is shown

Code Generation Report

Find: Match Case

Contents

Summary

Subsystem Report

Code Interface Report

Traceability Report

Static Code Metrics Report

Code Replacements Report

Generated Code

[-] Main file

[ert_main.c](#)

Object Name

Code Location

<code><S1>/Bitwise Operator</code>	<code>thirtytwobit_foureightbits.c:49, 55</code>
<code><S1>/Bitwise Operator1</code>	<code>thirtytwobit_foureightbits.c:50, 57</code>
<code><S1>/Bitwise Operator2</code>	<code>thirtytwobit_foureightbits.c:51, 59</code>
<code><S1>/Bitwise Operator3</code>	<code>thirtytwobit_foureightbits.c:52, 61</code>
<code><S1>/Data Type Conversion</code>	<code>thirtytwobit_foureightbits.c:44, 54</code>
<code><S1>/Data Type Conversion1</code>	<code>thirtytwobit_foureightbits.c:45, 56</code>
<code><S1>/Data Type Conversion2</code>	<code>thirtytwobit_foureightbits.c:46, 58</code>
<code><S1>/Data Type Conversion3</code>	<code>thirtytwobit_foureightbits.c:47, 60</code>
<code><S1>/Shift Arithmetic1</code>	<code>thirtytwobit_foureightbits.c:41, 57</code>
<code><S1>/Shift Arithmetic2</code>	
<code><S1>/Shift Arithmetic3</code>	

Code Generation Report

Find: Match Case

Contents

Summary

Subsystem Report

Code Interface Report

Traceability Report

Static Code Metrics Report

Code Replacements Report

Generated Code

[-] Main file

[ert_main.c](#)

[-] Model files

[thirtytwobit_foureightbits.c](#)

[thirtytwobit_foureightbits.h](#)

[thirtytwobit_foureightbits_priv.h](#)

[thirtytwobit_foureightbits_type.h](#)

[+] Utility files (1)

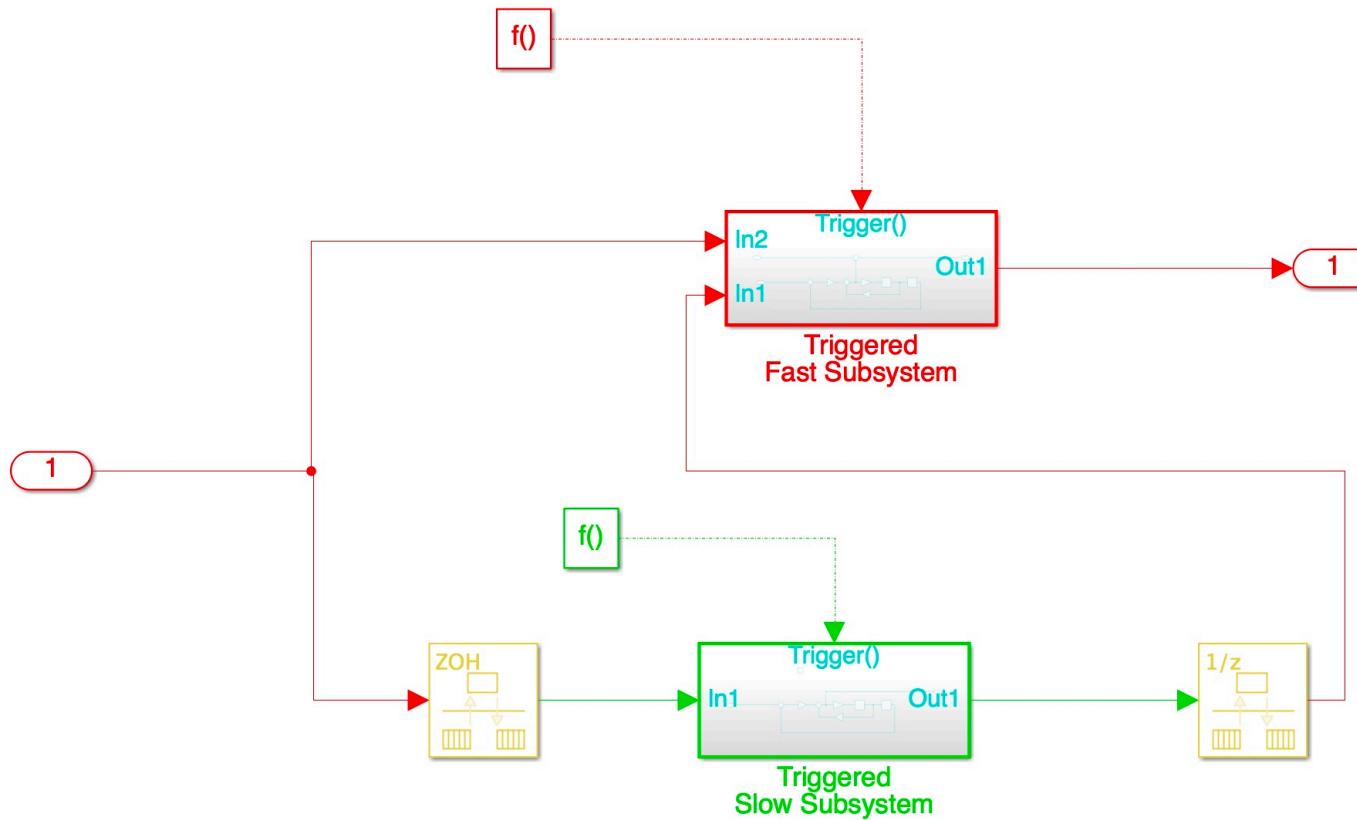
31 */* Real-time model */*
32 RT_MODEL_thirtytwobit_foureig_T thirtytwobit_foureightbits_M;
33 RT_MODEL_thirtytwobit_foureig_T *const thirtytwobit_foureightbits_M =
34 &thirtytwobit_foureightbits_M;
35
36
37 */* Model step function */*
38 void thirtytwobit_foureightbits_step(void)
39 {
40 */* Outport: '<Root>/Out1' incorporates:*
41 ** ArithShift: '<S1>/Shift Arithmetic1'*
42 ** ArithShift: '<S1>/Shift Arithmetic2'*
43 ** ArithShift: '<S1>/Shift Arithmetic3'*
44 ** DataTypeConversion: '<S1>/Data Type Conversion'*
45 ** DataTypeConversion: '<S1>/Data Type Conversion1'*
46 ** DataTypeConversion: '<S1>/Data Type Conversion2'*
47 ** DataTypeConversion: '<S1>/Data Type Conversion3'*
48 ** Inport: '<Root>/In1'*
49 ** S-Function (sfix_bitop): '<S1>/Bitwise Operator'*
50 ** S-Function (sfix_bitop): '<S1>/Bitwise Operator1'*
51 ** S-Function (sfix_bitop): '<S1>/Bitwise Operator2'*
52 ** S-Function (sfix_bitop): '<S1>/Bitwise Operator3'*
53 **/*
54 thirtytwobit_foureightbits_Y.Out1[0] = (uint8_T)
55 (thirtytwobit_foureightbits_U.In1 & 255U);
56 thirtytwobit_foureightbits_Y.Out1[1] = (uint8_T)
57 ((thirtytwobit_foureightbits_U.In1 & 65280U) >> 8);
58 thirtytwobit_foureightbits_Y.Out1[2] = (uint8_T)
59 ((thirtytwobit_foureightbits_U.In1 & 16711680U) >> 16);
60

OK Help

Q: what are the hex equivalents of the decimal numbers 255, 65280, 16711680, and 4278190080?

Atomic Subsystems

- Sometimes it is useful to have a separate file generated for each subsystem in a model.
 - For example, it makes the code easier to read.
 - In order to do this, you need to specify each subsystem as an “atomic unit”.
- Example: Two virtual wheels connected to the haptic wheel



Default code (nonatomic subsystems)

- Default setting is for all code to be in one file, including separate functions to update both subsystems, e.g., the code to update the fast subsystem

Code Generation Report

Find: Match Case

Contents

- [Summary](#)
- [Subsystem Report](#)
- [Code Interface Report](#)
- [Traceability Report](#)
- [Static Code Metrics Report](#)
- [Code Replacements Report](#)

Generated Code

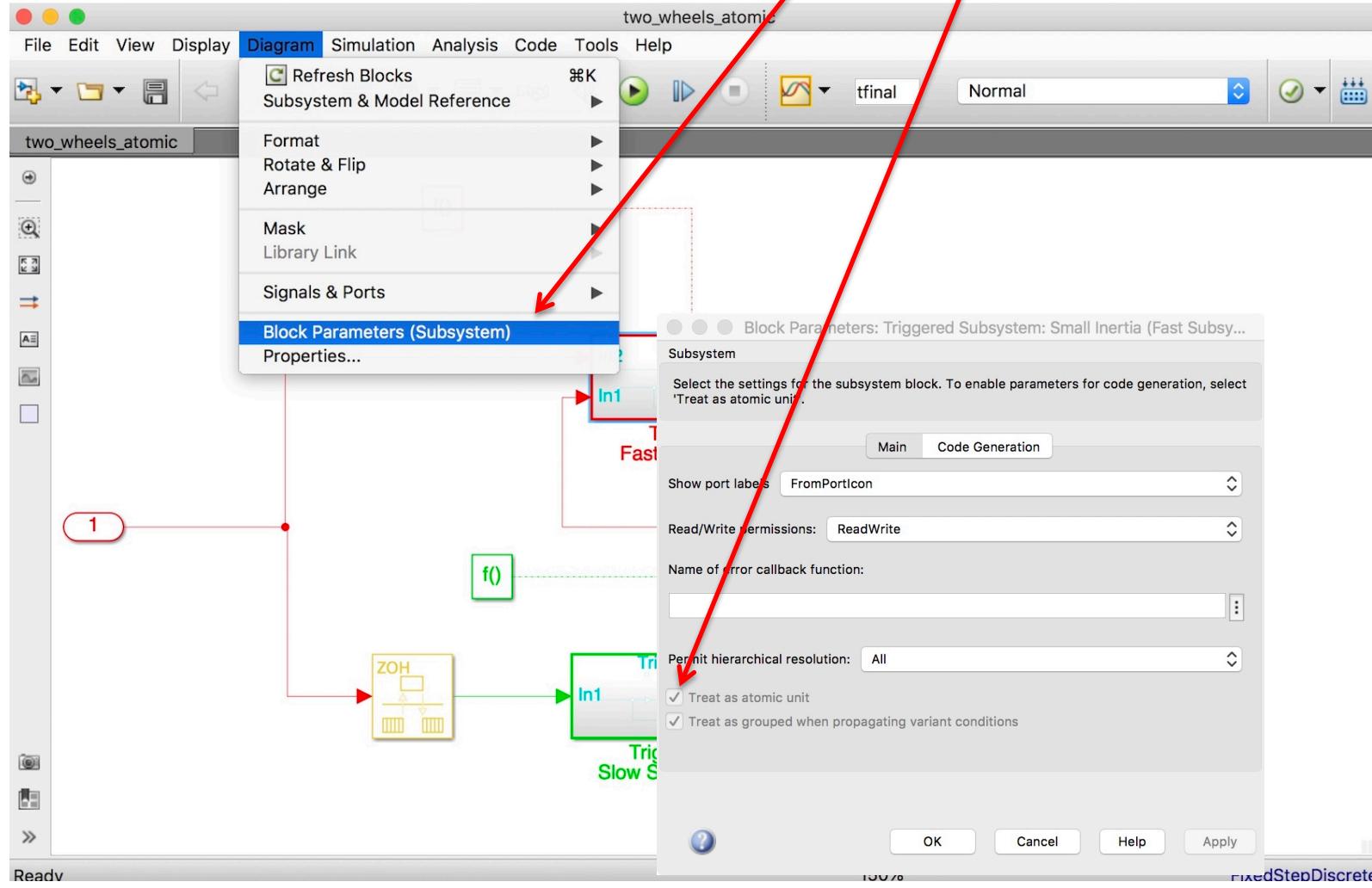
- [\[-\] Main file](#)
 - [ert_main.c](#)
- [\[-\] Model files](#)
 - [two_wheels_NOT_atomic.c](#)
 - [two_wheels_NOT_atomic.h](#)
 - [two_wheels_NOT_atomic_private.h](#)
 - [two_wheels_NOT_atomic_types.h](#)
- [\[-\] Utility files](#)
 - [rtwtypes.h](#)

```
/* S-Function (fcncallgen): '<Root>/Function-Call Generator' incorporates:
 * SubSystem: '<Root>/TriggeredFastSubsystem'
 */
72 if (two_wheels_NOT_atomic_DW.TriggeredFastSubsystem_RESET_EL) {
73     TriggeredFastSubsystem_ELAPS_T = 0U;
74 } else {
75     TriggeredFastSubsystem_ELAPS_T = two_wheels_NOT_atomic_M->Timing.clockTick0
76     - two_wheels_NOT_atomic_DW.TriggeredFastSubsystem_PREV_T;
77 }
78 two_wheels_NOT_atomic_DW.TriggeredFastSubsystem_PREV_T =
79     two_wheels_NOT_atomic_M->Timing.clockTick0;
80 two_wheels_NOT_atomic_DW.TriggeredFastSubsystem_RESET_EL = false;
81
82 /* DiscreteIntegrator: '<S1>/Discrete-Time Integrator' */
83 if (two_wheels_NOT_atomic_DW.DiscreteTimeIntegrator_SYSTEM_E == 0) {
84     two_wheels_NOT_atomic_DW.DiscreteTimeIntegrator_DSTATE += 0.002 * (real_T)
85     TriggeredFastSubsystem_ELAPS_T
86     * two_wheels_NOT_atomic_DW.DiscreteTimeIntegrator_PREV_U;
87 }
88
89 /* End of DiscreteIntegrator: '<S1>/Discrete-Time Integrator' */
90
91 /* DiscreteIntegrator: '<S1>/Discrete-Time Integrator1' */
92 if (two_wheels_NOT_atomic_DW.DiscreteTimeIntegrator1_SYSTEM_ == 0) {
93     two_wheels_NOT_atomic_DW.DiscreteTimeIntegrator1_DSTATE += 0.002 * (real_T)
94     TriggeredFastSubsystem_ELAPS_T
95     * two_wheels_NOT_atomic_DW.DiscreteTimeIntegrator1_PREV_U;
96 }
97
98 /* End of DiscreteIntegrator: '<S1>/Discrete-Time Integrator1' */
99
100
101
102
103
```

OK Help

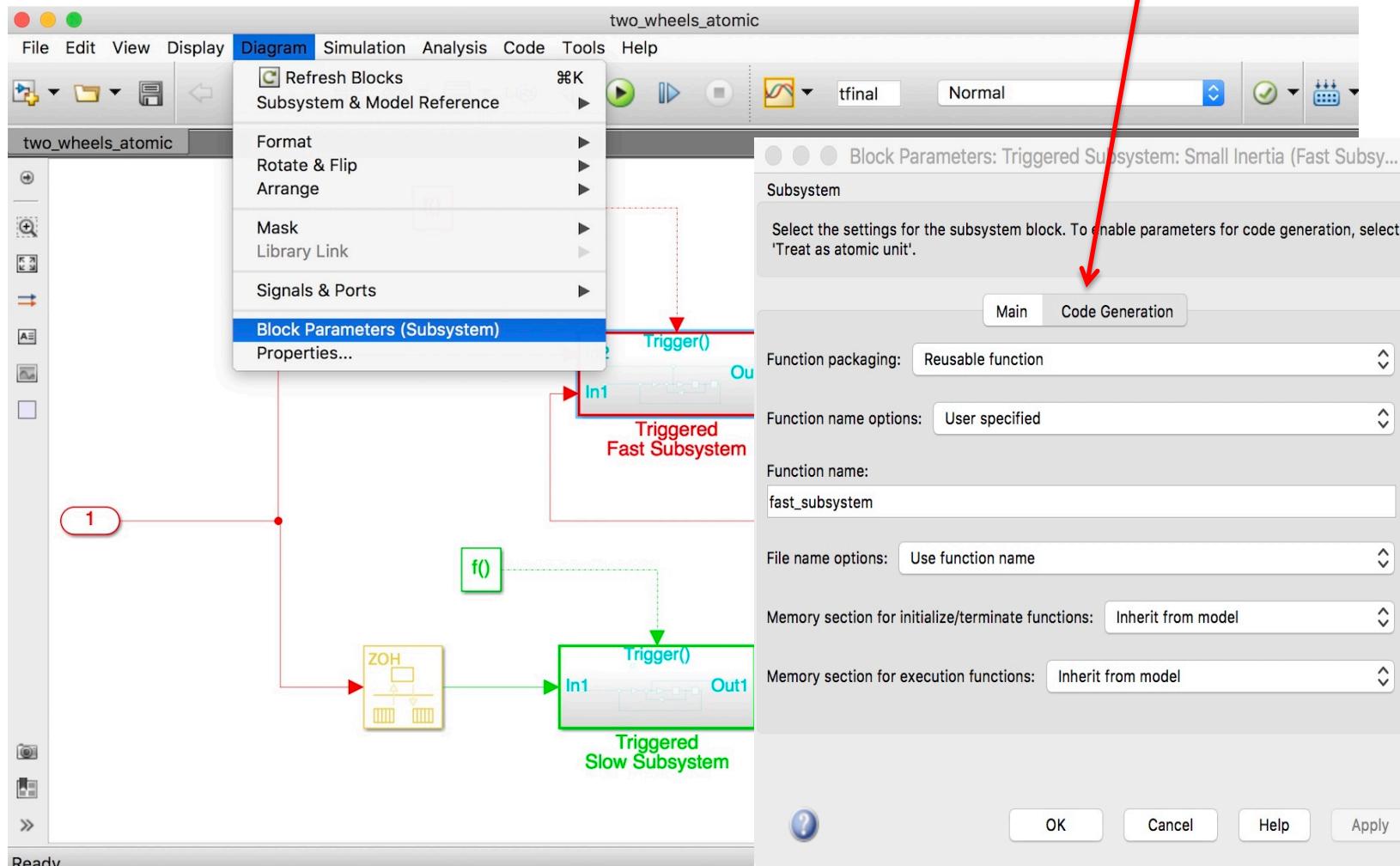
Making subsystems atomic

- Select a subsystem, and navigate to the menu Diagram/Block Parameters (Subsystem)/Main
- For a Triggered Subsystem block, the "Treat as atomic unit" box will be automatically checked and grayed out.



Name the function

- From the menu Diagram/Block Parameters (Subsystem)/Code Generation
- Select “Reusable function” and “User specified” function name
- Name the function, and specify “Use function name” for the file name



Code with atomic subsystems

- Separate functions and files “fast_subsystem” and “slow_subsystem”

Code Generation Report

Find: Match Case

[Subsystem Report](#)

[Code Interface Report](#)

[Traceability Report](#)

[Static Code Metrics Report](#)

[Code Replacements Report](#)

Generated Code

[-] Main file

[ert_main.c](#)

[-] Model files

[two_wheels_atomic.c](#)

[two_wheels_atomic.h](#)

[two_wheels_atomic_private.h](#)

[two_wheels_atomic_types.h](#)

[-] Subsystem files

[fast_subsystem.c](#)  

[fast_subsystem.h](#)

[slow_subsystem.c](#)

[slow_subsystem.h](#)

[+] Utility files (1)

File: fast_subsystem.c

```
1  /*
2  * Academic License - for use in teaching, academic research, and meeting
3  * course requirements at degree granting institutions only. Not for
4  * government, commercial, or other organizational use.
5  *
6  * File: fast_subsystem.c
7  *
8  * Code generated for Simulink model 'two_wheels_atomic'.
9  *
10 * Model version : 1.6
11 * Simulink Coder version : 8.14 (R2018a) 06-Feb-2018
12 * C/C++ source code generated on : Tue Sep 10 10:19:39 2019
13 *
14 * Target selection: ert.tlc
15 * Embedded hardware selection: Intel->x86-64 (Windows64)
16 * Code generation objective: Traceability
17 * Validation result: Not run
18 */
19
20 #include "fast_subsystem.h"
21
22 /* Include model header file for global data */
23 #include "two_wheels_atomic.h"
24 #include "two_wheels_atomic_private.h"
25
26 /* System initialize for function-call system: '<Root>/Triggered Fast Subsystem' */
27 void fast_subsystem_Init(DW_fast_subsystem_T *localDW)
28 {
```

OK Help

Triggered and Atomic Subsystems

There are several ways to make a subsystem that is both triggered and atomic:

- Use a Triggered Subsystem block from the Ports & Subsystems menu of the Simulink Library Browser.
 - This block will automatically be atomic and triggered.
- Use an Atomic Subsystem block from the Ports & Subsystems menu of the Simulink Library Browser.
 - This block will automatically be atomic. Add a Trigger block to make it triggered.
- Use a regular Subsystem block either from the Ports & Subsystems menu of the Simulink Library Browser, or by using the “Create Subsystem from Selection” option from the Subsystem & Model Reference menu of a Simulink diagram.
 - Make the subsystem atomic by checking the “Treat as atomic unit box” of the Diagram/Block Parameters (Subsystem)/Main menu and then add a Trigger block to the subsystem.
 - Make the subsystem triggered by adding a Trigger and then make the subsystem atomic by checking the “Treat as atomic unit box” of the Diagram/Block Parameters (Subsystem)/Main menu.

NOTE: In Matlab 2018a this last option will cause the “atomic unit” box to be grayed out but not checked. This is a bug. THE SUBSYSTEM IS STILL ATOMIC!