

Analyzing Code Generated from a Simulink* Model

Jim Freudenberg

University of Michigan

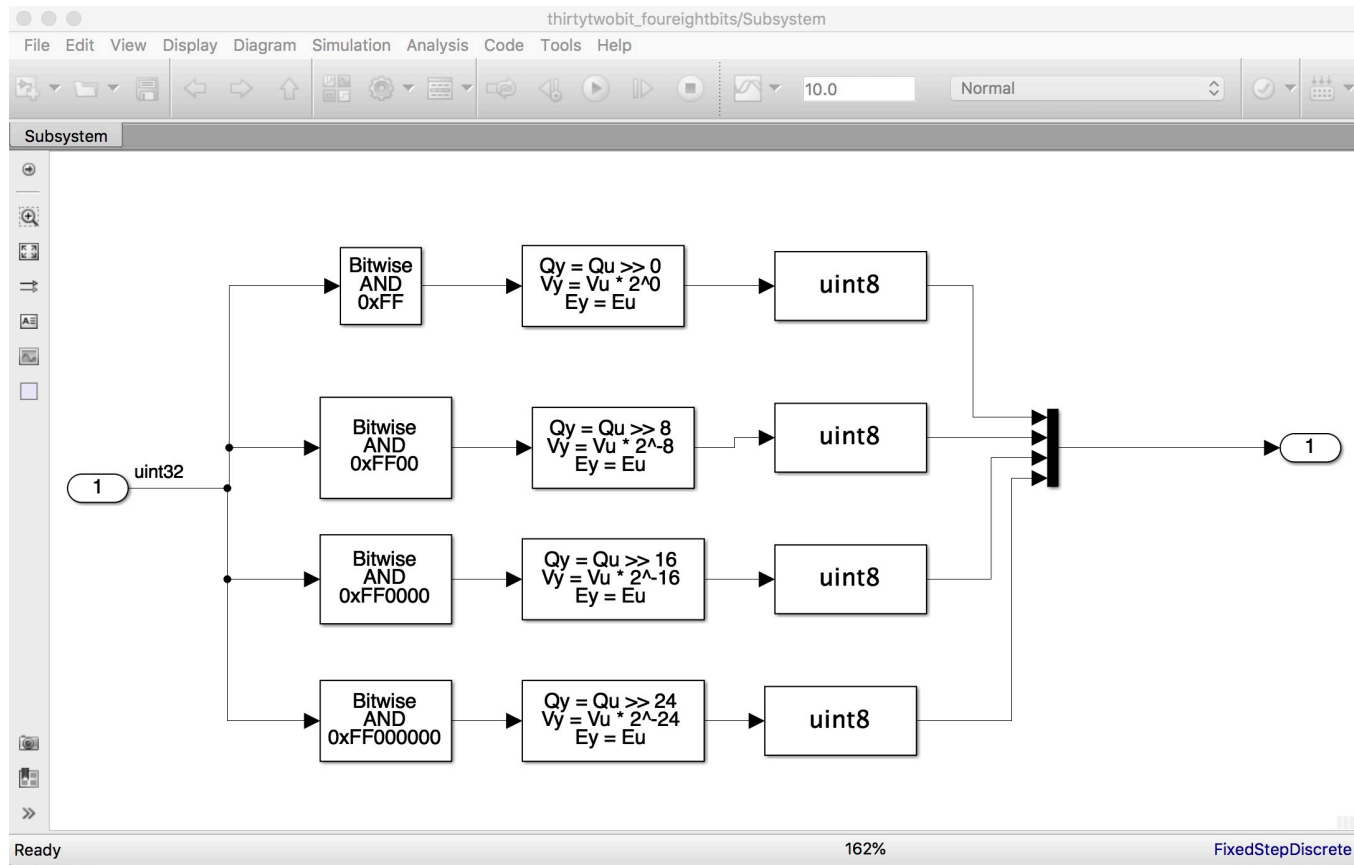
EECS 461: Embedded Control Systems

* Matlab 2018a

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



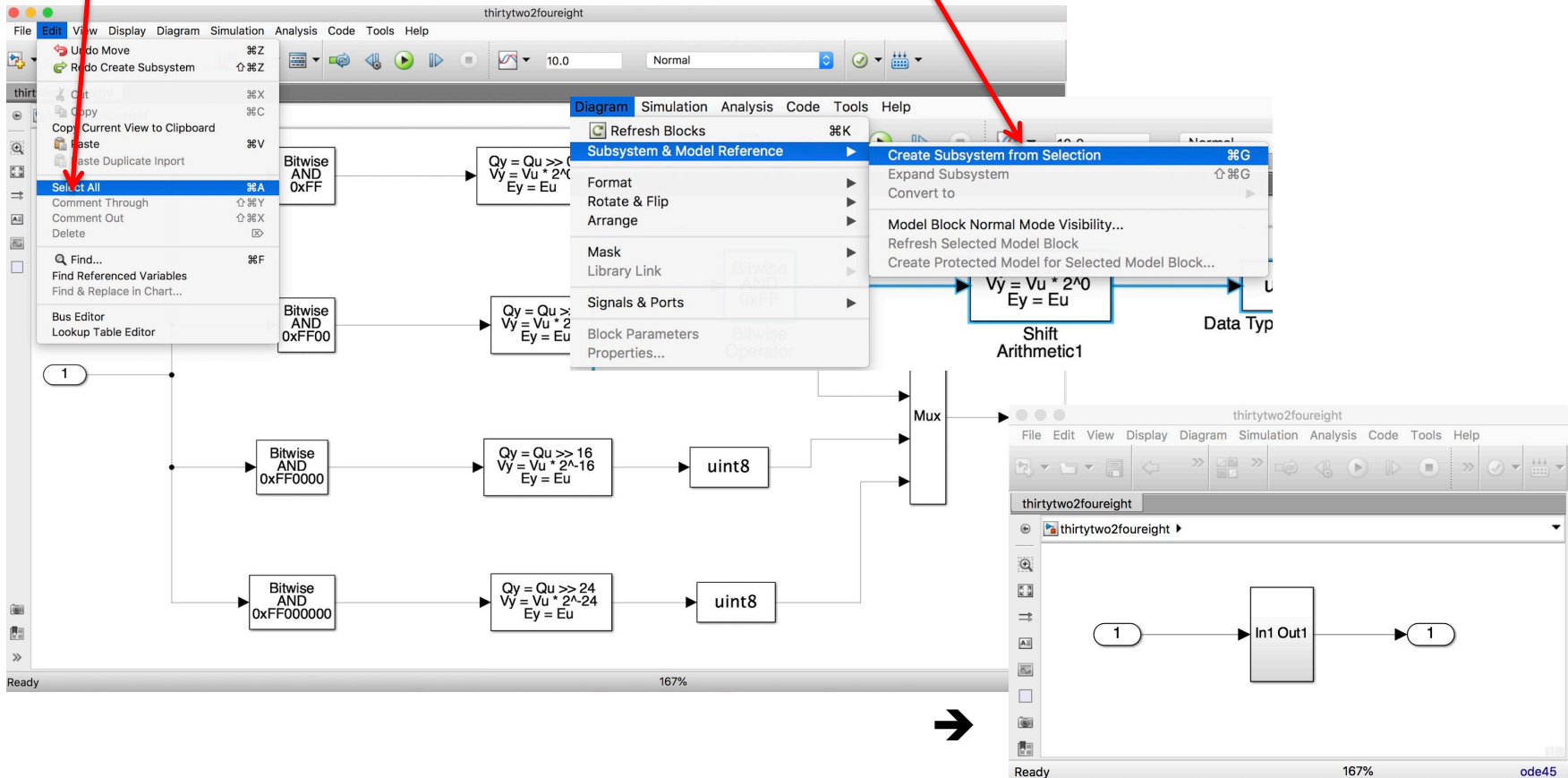
Create a Simulink model of the system depicted in Figure 3 of the handout “Simulink Models for Autocode Generation”. This model takes one 32-bit integer as input and breaks it into four 8-bit integers.

Make a Subsystem

From the Edit and Diagram menus:

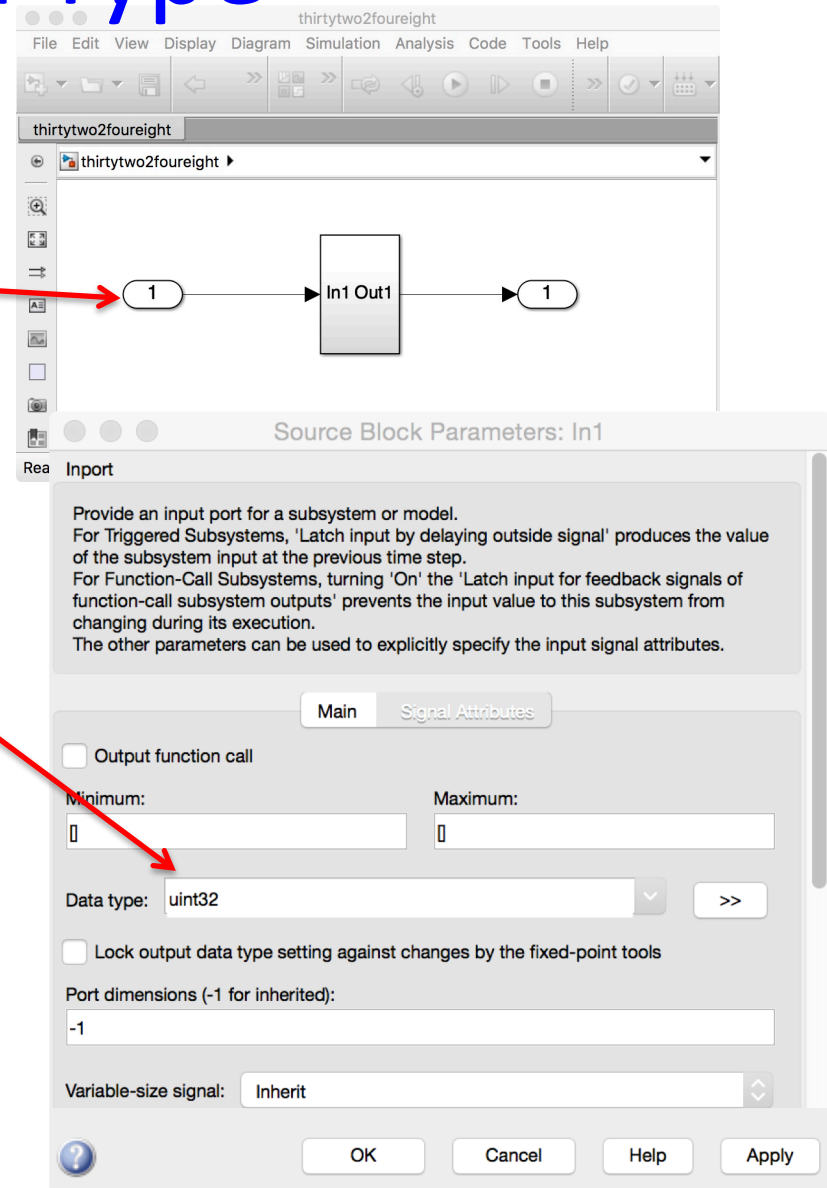
Edit/Select All

Diagram/Subsystem & Model Reference/Create Subsystem from Selection

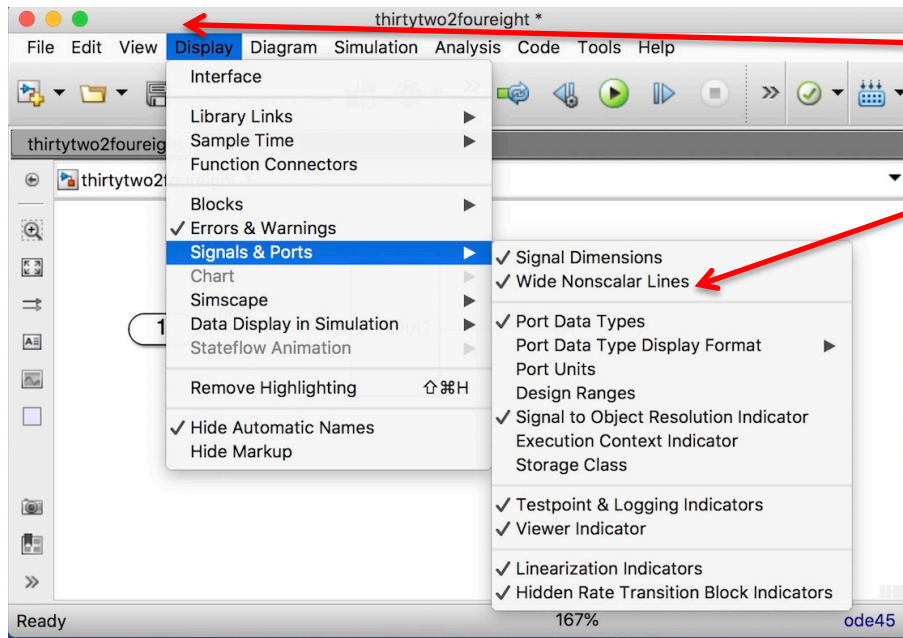


Input Data Type

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



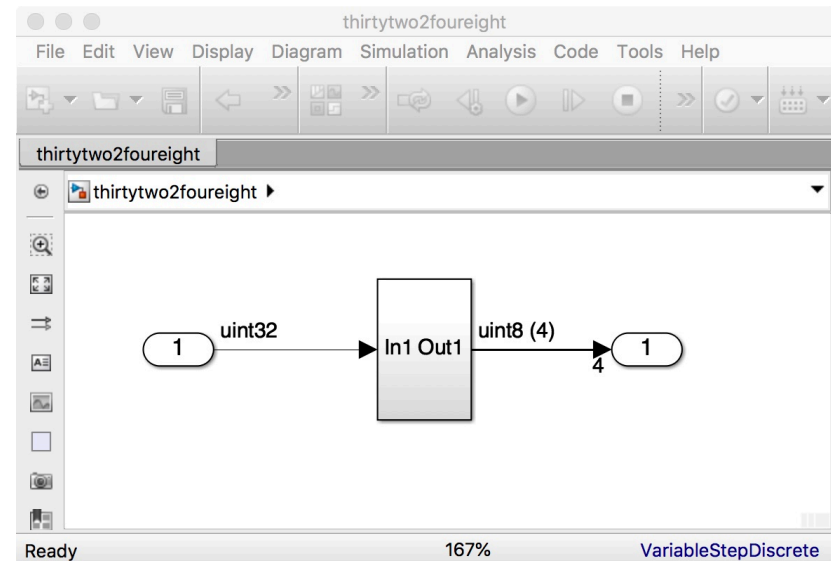
Display Data Types



- Under the Display Menu, select

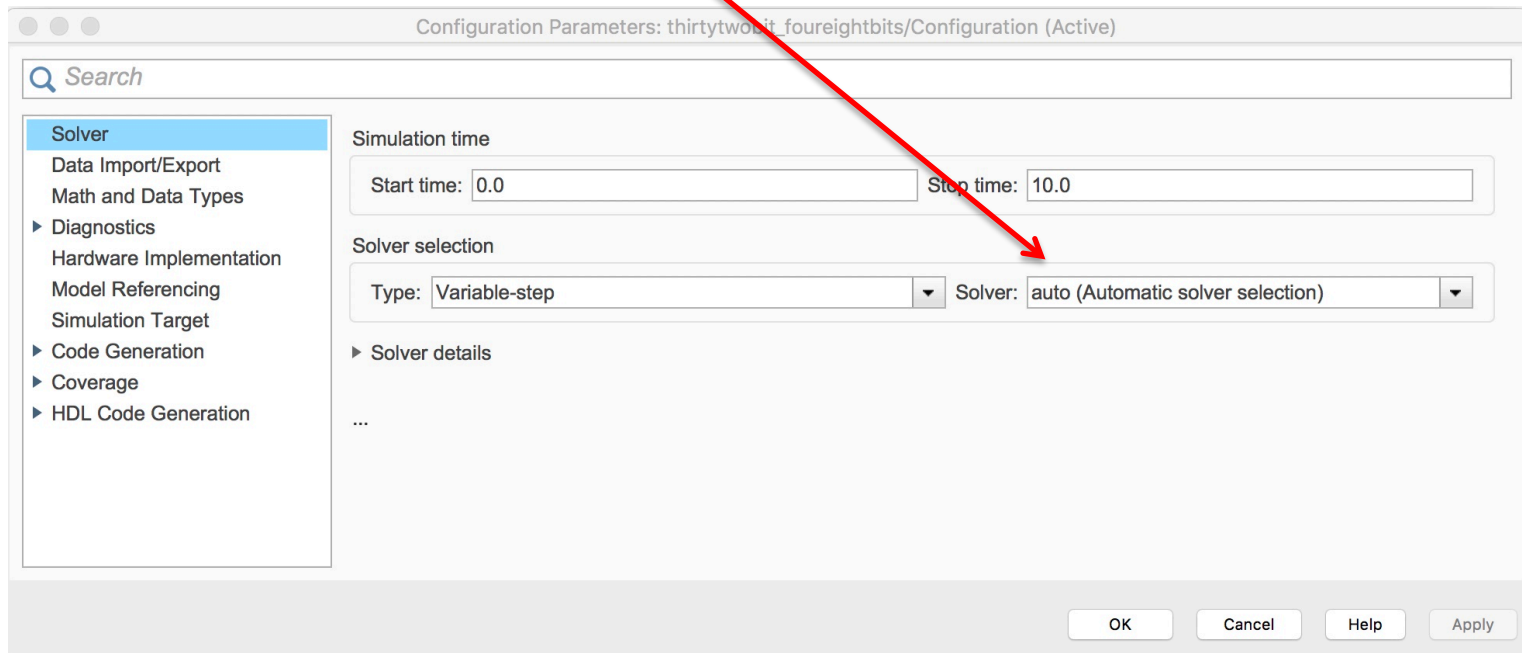
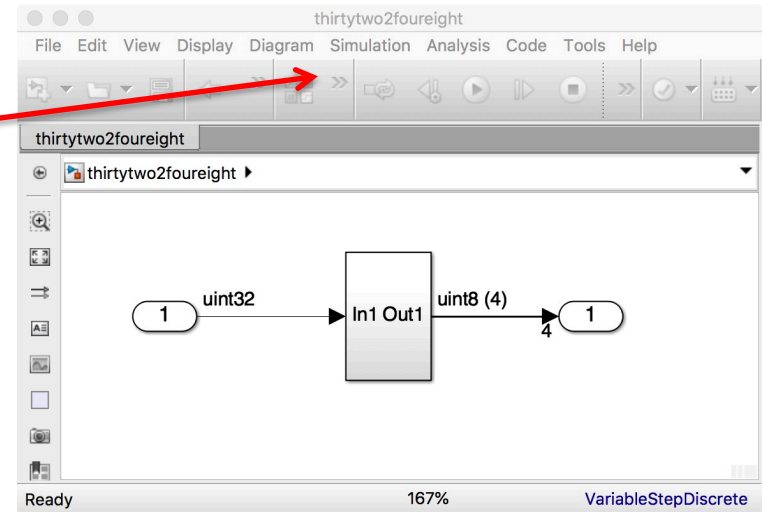
Display/Signals & Ports/Signal Dimensions
Display/Signals & Ports/Wide Nonscalar Lines
Display/Signals & Ports/Port Data Types

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



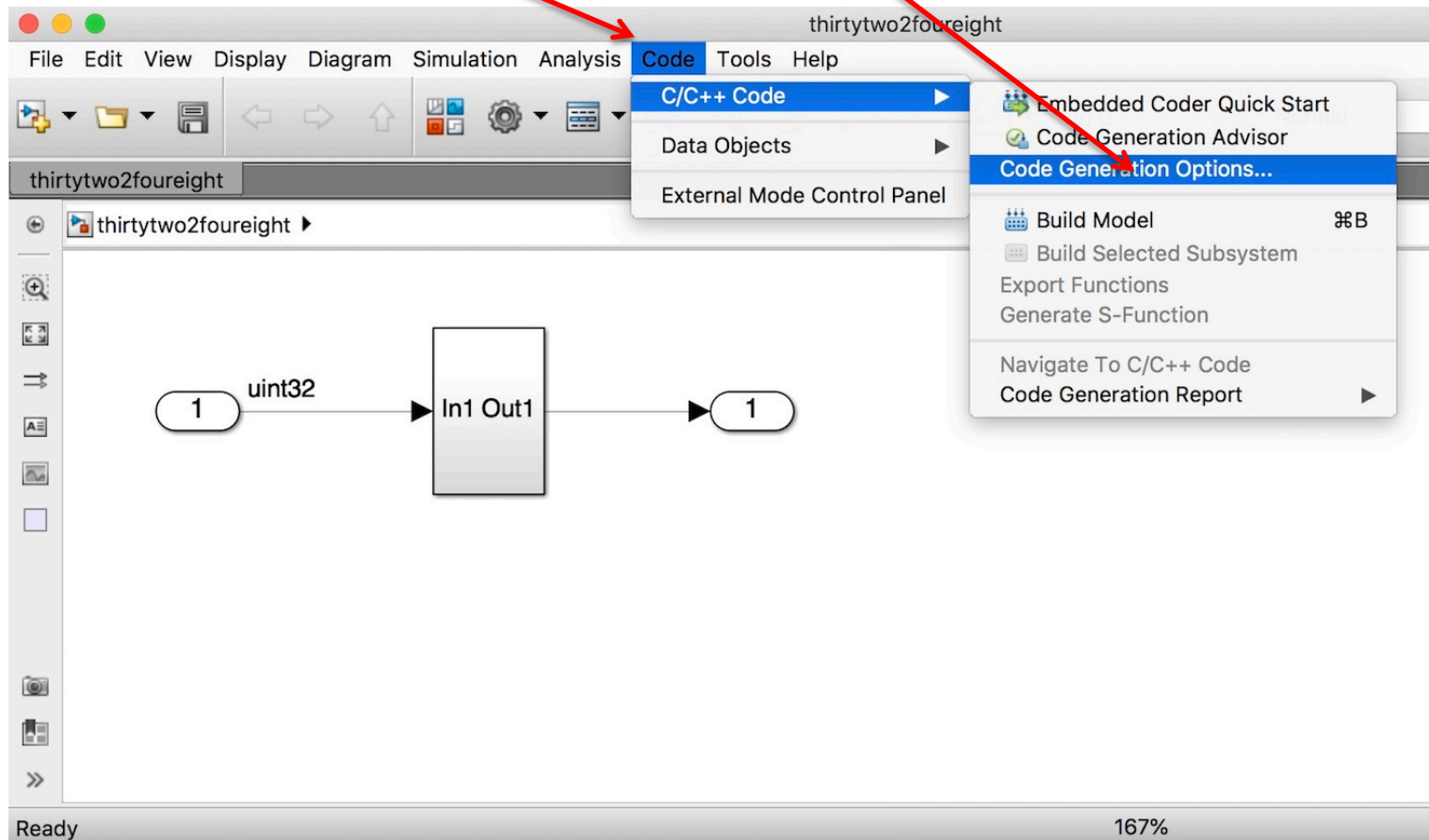
Solver Options

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



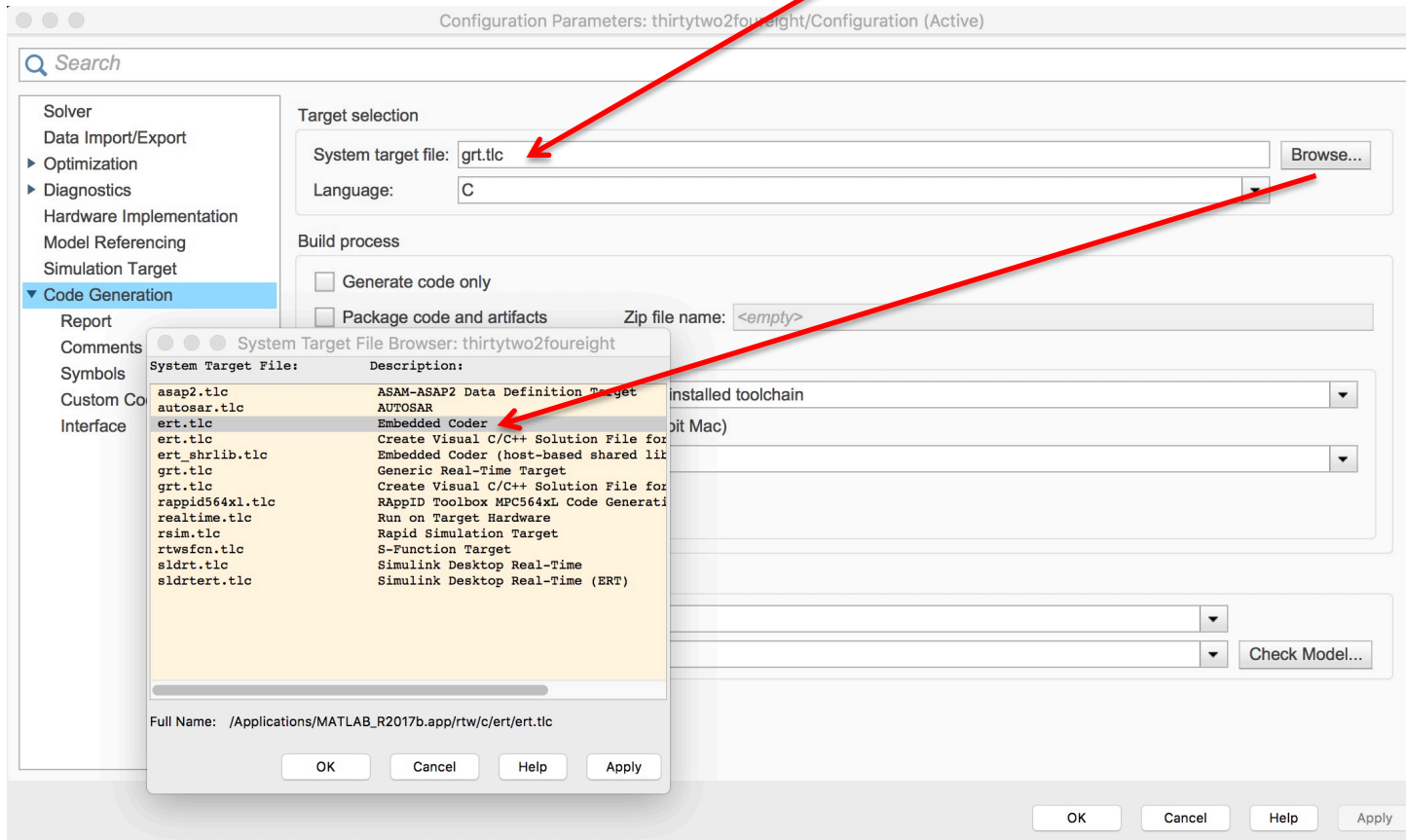
Options for Code Generation

From the Code Menu, select:
Code/C/C++ Code/Code Generation Options...



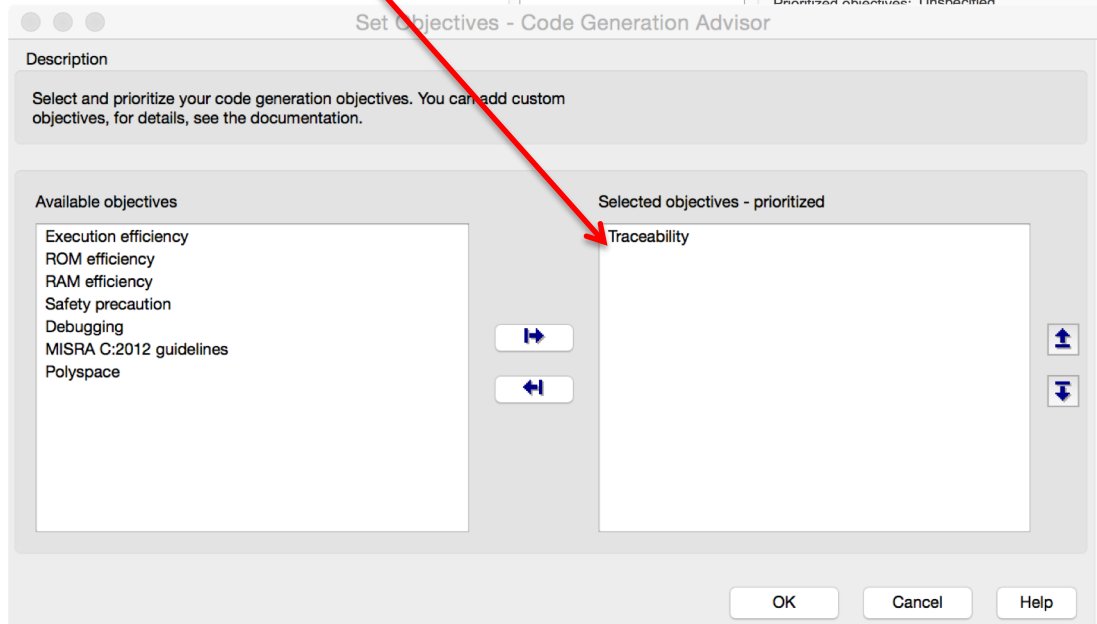
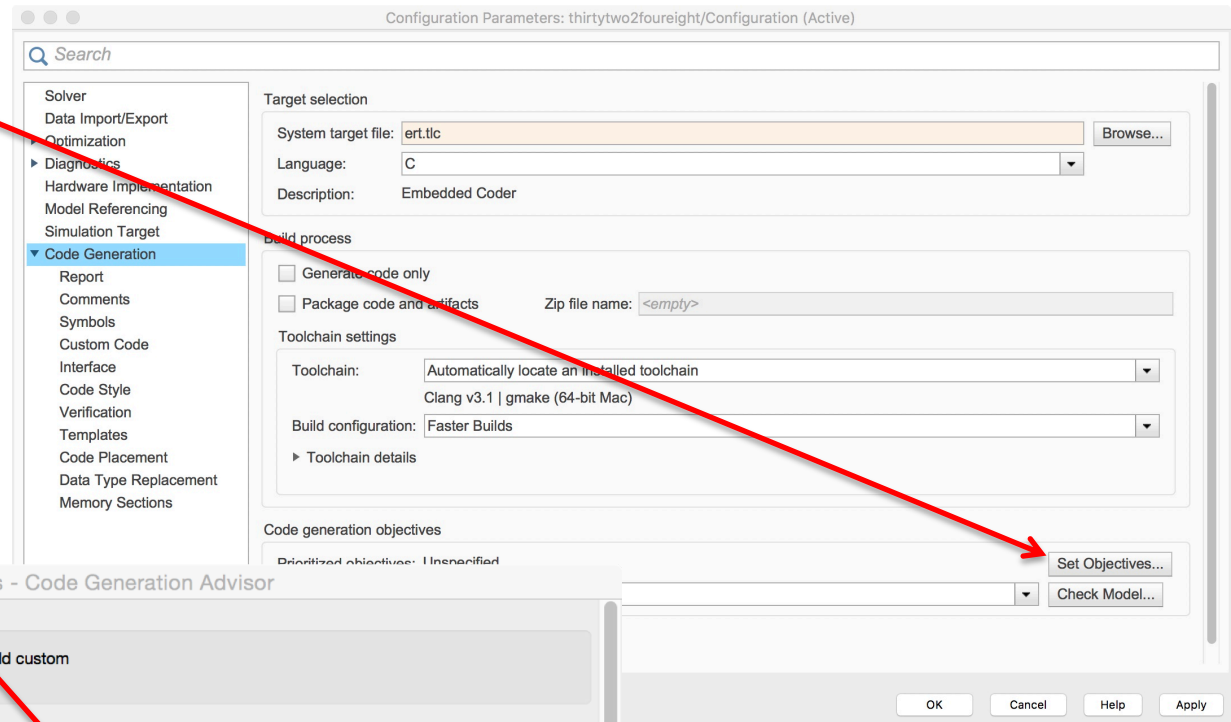
Options Window

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`



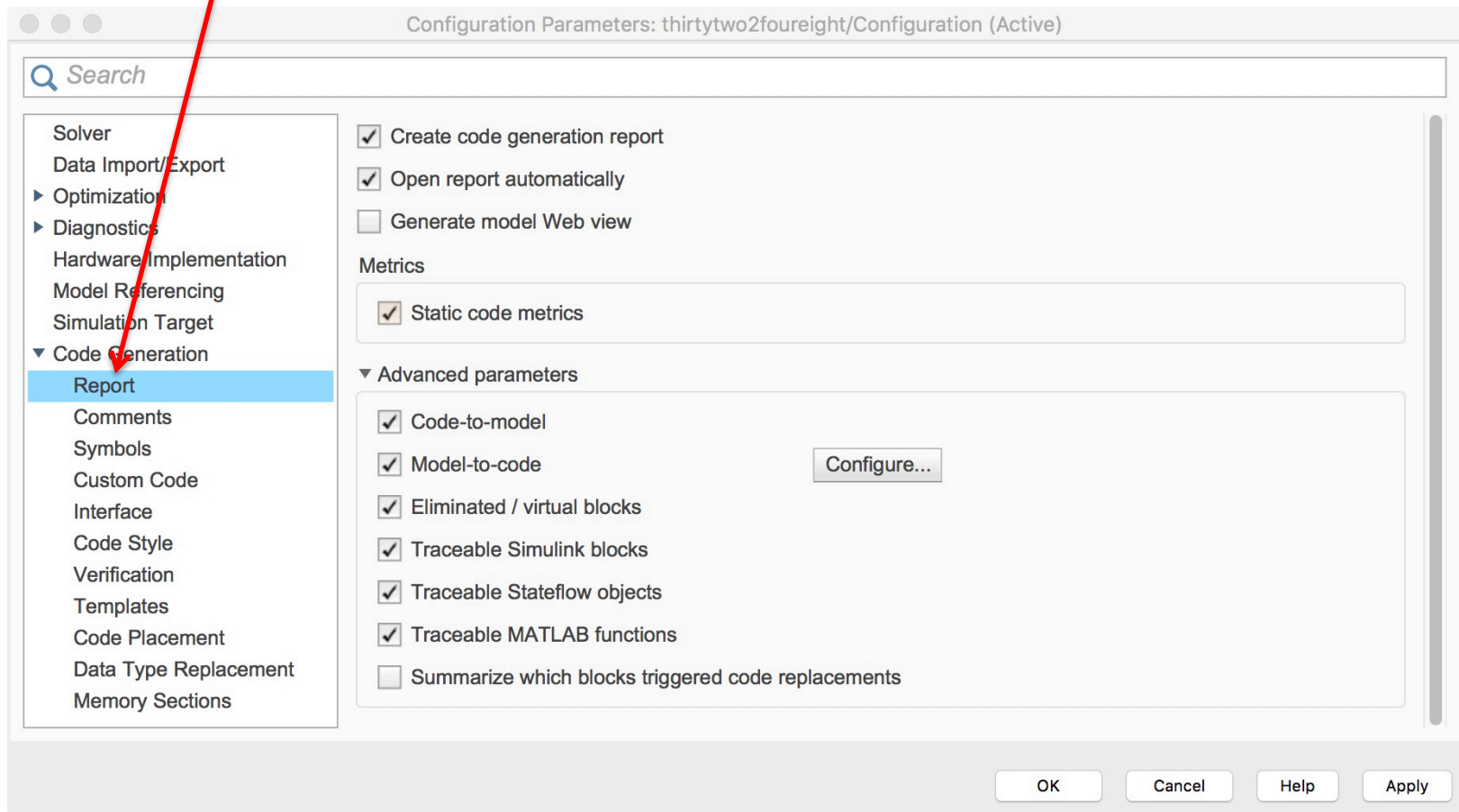
Embedded Coder Options

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



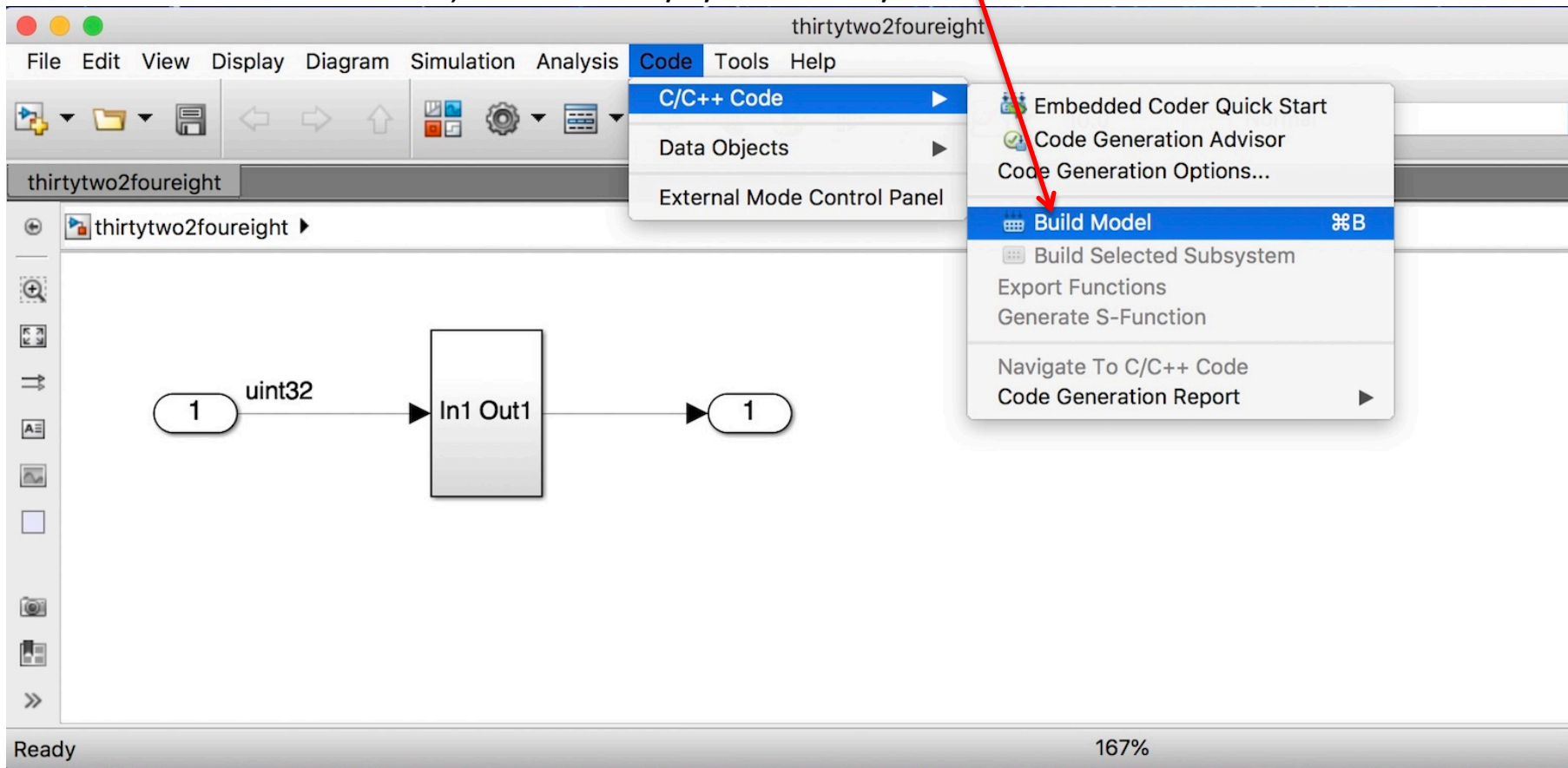
Report Options

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.

The screenshot shows the 'Code Generation Report' window for the model 'thirtytwobit_foureightbits'. The window is divided into a sidebar and a main content area.

Sidebar:

- Contents:** Summary (highlighted), 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).

Main Content Area:

Code Generation Report for 'thirtytwobit_foureightbits'

Model Information

Author	jfr
Last Modified By	jfr
Model Version	1.3
Tasking Mode	SingleTasking

[Configuration settings at time of code generation](#)

Code Information

System Target File	ert.tlc
Hardware Device Type	Intel->x86-64 (Windows64)
Simulink Coder Version	8.14 (R2018a) 06-Feb-2018
Timestamp of Generated Source Code	Mon Sep 9 12:23:28 2019
Location of Generated Source Code	/Users/jfr/Documents/EECS_461/Fall2019/Matlab2019/traceability/thirtytwobit_foureightbits_ert_rtw/
Type of Build	Model
Memory Information	Global Memory: 19(bytes) Maximum Stack: 0(bytes)
Objectives Specified	Traceability

OK Help

- Information about the model settings

Code Interface Report

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

The screenshot shows a window titled "Code Generation Report". On the left is a sidebar with a "Contents" section containing links to "Summary", "Subsystem Report", "Code Interface Report" (highlighted), "Traceability Report", "Static Code Metrics Report", and "Code Replacements Report". Below this is a "Generated Code" section with expandable/collapsible items for "Main file" (containing `ert_main.c`) and "Model files" (containing `thirtytwobit_foureightbits.c`, `thirtytwobit_foureightbits.h`, `thirtytwobit_foureightbits_priv`, and `thirtytwobit_foureightbits_types`). A "Utility files (1)" section is also visible. The main area displays the "Code Interface Report for thirtytwobit_foureightbits". It includes a "Table of Contents" with links to "Entry Point Functions", "Inports", "Outports", "Interface Parameters", and "Data Stores". Under "Entry Point Functions", two functions are listed: `thirtytwobit_foureightbits_initialize` and `thirtytwobit_foureightbits_step`. Each function has a table of details: Prototype, Description, Timing, Arguments, Return value, and Header file.

Code Interface Report for thirtytwobit_foureightbits

Table of Contents

- [Entry Point Functions](#)
- [Inports](#)
- [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	thirtytwobit_foureightbits.h

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	thirtytwobit_foureightbits.h

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_foueightbits.c](#)
[thirtytwobit_foueightbits.h](#)
[thirtytwobit_foueightbits_private.h](#)
[thirtytwobit_foueightbits_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

1. [File Information](#)

2. [Global Variables](#)

3. [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

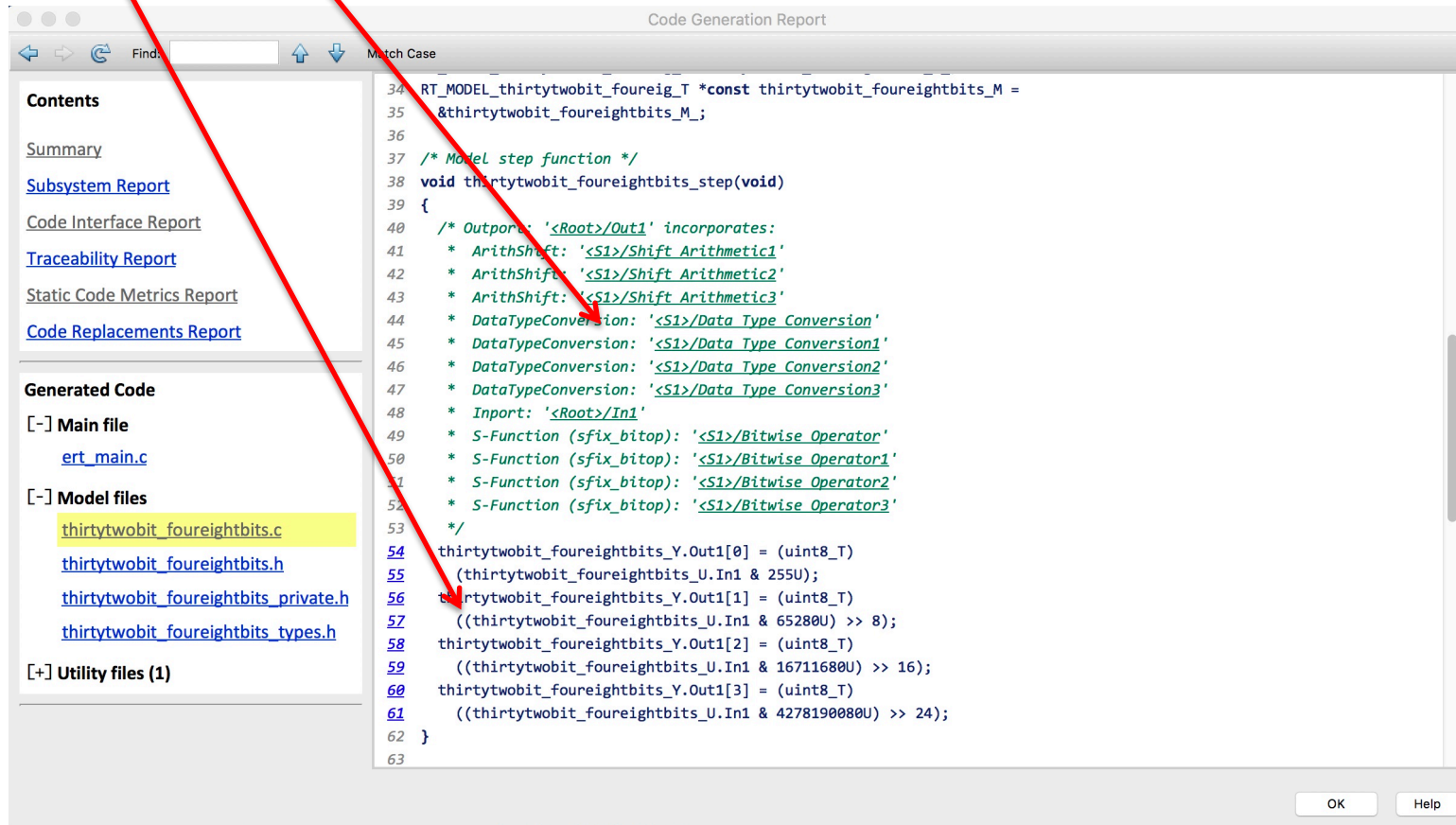
File Name	Lines of Code	Lines	Generated On
rtwtypes.h	81	161	09/09/2019 12:23 PM
thirtytwobit_foueightbits.h	37	109	09/09/2019 12:23 PM
thirtytwobit_foueightbits.c	31	95	09/09/2019 12:23 PM
thirtytwobit_foueightbits_private.h	4	30	09/09/2019 12:23 PM
thirtytwobit_foueightbits_types.h	4	33	09/09/2019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

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_foueightbits.c](#)
 - [thirtytwobit_foueightbits.h](#)
 - [thirtytwobit_foueightbits_private.h](#)
 - [thirtytwobit_foueightbits_types.h](#)
- ☐ Utility files (1)

```
34 RT_MODEL_thirtytwobit_foueight_T *const thirtytwobit_foueightbits_M =
35 &thirtytwobit_foueightbits_M_;
36
37 /* Model step function */
38 void thirtytwobit_foueightbits_step(void)
39 {
40     /* Output: '<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      * Input: '<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_foueightbits_Y.Out1[0] = (uint8_T)
55     ((thirtytwobit_foueightbits_U.In1 & 255U);
56     thirtytwobit_foueightbits_Y.Out1[1] = (uint8_T)
57     ((thirtytwobit_foueightbits_U.In1 & 65280U) >> 8);
58     thirtytwobit_foueightbits_Y.Out1[2] = (uint8_T)
59     ((thirtytwobit_foueightbits_U.In1 & 16711680U) >> 16);
60     thirtytwobit_foueightbits_Y.Out1[3] = (uint8_T)
61     ((thirtytwobit_foueightbits_U.In1 & 4278190080U) >> 24);
62 }
63
```

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)

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_fouereightbits.c](#)
 - [thirtytwobit_fouereightbits.h](#)
 - [thirtytwobit_fouereightbits_private.h](#)

Traceability Report for thirtytwobit_fouereightbits

Table of Contents

- [Eliminated / Virtual Blocks](#)
- [Traceable Simulink Blocks / Stateflow Objects / MATLAB Functions](#)
 - [thirtytwobit_fouereightbits](#)
 - [thirtytwobit_fouereightbits/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	Output

Traceable Simulink Blocks / Stateflow Objects / MATLAB Functions

OK Help

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

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_foueightbits.c](#)
- [thirtytwobit_foueightbits.h](#)
- [thirtytwobit_foueightbits_private.h](#)
- [thirtytwobit_foueightbits_types.h](#)

[+] Utility files (1)

Code Generation Report

Find: Match Case

Object Name	Code Location
<Root>/In1	thirtytwobit_foueightbits.c:48, 55, 57, 59, 61 thirtytwobit_foueightbits.h:49
<Root>/Out1	thirtytwobit_foueightbits.c:40, 54, 56, 58, 60 thirtytwobit_foueightbits.h:54

Subsystem: [thirtytwobit_foueightbits/Subsystem](#)

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

OK Help

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

Traceability

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

The screenshot displays the MATLAB/Simulink Code Generation Report interface. On the left, the 'Contents' pane lists various reports, with 'Traceability Report' highlighted. Below it, the 'Generated Code' section shows a list of files, including 'thirtytwobit_foueightbits.c'. The main pane shows the 'Code Generation Report' for the 'thirtytwobit_foueightbits/Subsystem'.

The report is divided into two sections: 'Object Name' and 'Code Location'. The 'Object Name' section lists the following objects:

- <Root>/In1
- <Root>/Out1
- <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 Arithmetic
- <S1>/Shift Arithmetic2
- <S1>/Shift Arithmetic3

The 'Code Location' section shows the corresponding code locations for these objects. The 'Shift Arithmetic' block is highlighted in blue, and a red arrow points from its name in the 'Object Name' list to the corresponding Simulink block in the diagram.

The Simulink diagram shows a subsystem with four parallel processing paths. Each path starts with a 'Bitwise AND' block (0xFF, 0xFF00, 0xFF0000, and 0xFF000000 respectively) followed by a 'Shift Arithmetic' block. The 'Shift Arithmetic' block for the second path is highlighted in blue, matching the one in the report. The output of each path is a 'uint8' block, which are then combined into a 'uint8 (4)' block. The diagram also shows a 'uint32' input block and a 'uint8' output block.

The status bar at the bottom indicates 'Ready', 'View diagnostics', '162%', and 'FixedStepDiscrete'.

Traceability: .c file

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

The screenshot displays the 'Code Generation Report' interface. On the left, a 'Contents' pane lists various report sections, with 'Traceability Report' highlighted. Below it, the 'Generated Code' section shows a tree structure with 'Main file' (ert_main.c) and 'Model files' (thirtytwobit_foueightbits.c, thirtytwobit_foueightbits.h, thirtytwobit_foueightbits_priv, thirtytwobit_foueightbits_type). The main pane shows a table with two columns: 'Object Name' and 'Code Location'. The table lists several objects, including Bitwise Operators and Data Type Conversions, each with a link to its C code location. A red arrow points from the '<S1>/Shift Arithmetic1' entry in the table to a corresponding entry in the 'Generated Code' tree. Another red arrow points from the '<S1>/Shift Arithmetic1' entry in the table to the C code snippet in the 'Generated Code' pane, which shows the implementation of the shift operation.

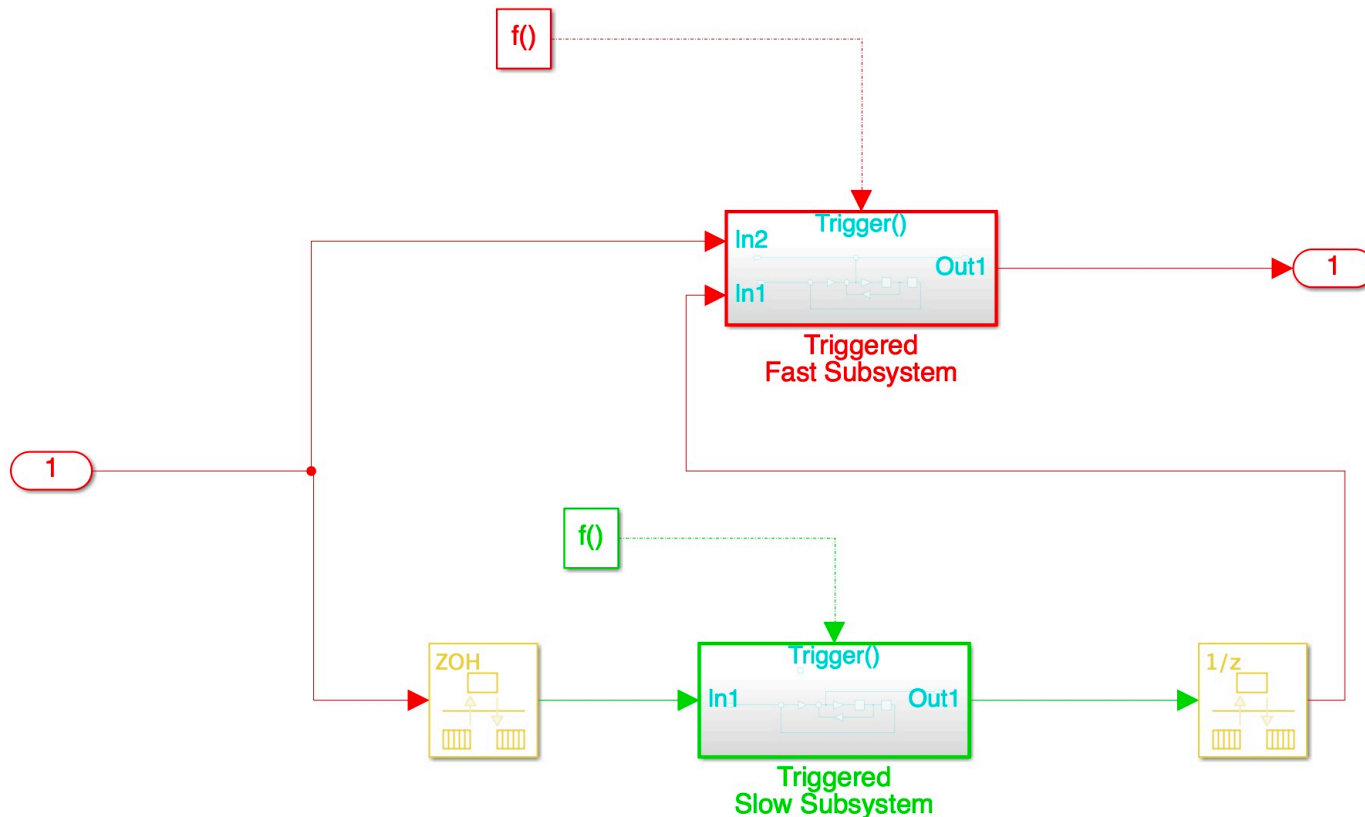
Object Name	Code Location
<S1>/Bitwise Operator	thirtytwobit_foueightbits.c:49, 55
<S1>/Bitwise Operator1	thirtytwobit_foueightbits.c:50, 57
<S1>/Bitwise Operator2	thirtytwobit_foueightbits.c:51, 59
<S1>/Bitwise Operator3	thirtytwobit_foueightbits.c:52, 61
<S1>/Data Type Conversion	thirtytwobit_foueightbits.c:44,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

```
31
32 /* Real-time model */
33 RT_MODEL_thirtytwobit_foueight_T thirtytwobit_foueightbits_M;
34 RT_MODEL_thirtytwobit_foueight_T *const thirtytwobit_foueightbits_M =
35 &thirtytwobit_foueightbits_M;
36
37 /* Model step function */
38 void thirtytwobit_foueightbits_step(void)
39 {
40     /* Output: '<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      * Input: '<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_foueightbits_Y.Out1[0] = (uint8_T)
55     (thirtytwobit_foueightbits_U.In1 & 255U);
56     thirtytwobit_foueightbits_Y.Out1[1] = (uint8_T)
57     ((thirtytwobit_foueightbits_U.In1 & 65280U) >> 8);
58     thirtytwobit_foueightbits_Y.Out1[2] = (uint8_T)
59     ((thirtytwobit_foueightbits_U.In1 & 16711680U) >> 16);
60     thirtytwobit_foueightbits_M.Out1[0] = (uint8_T)
```

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

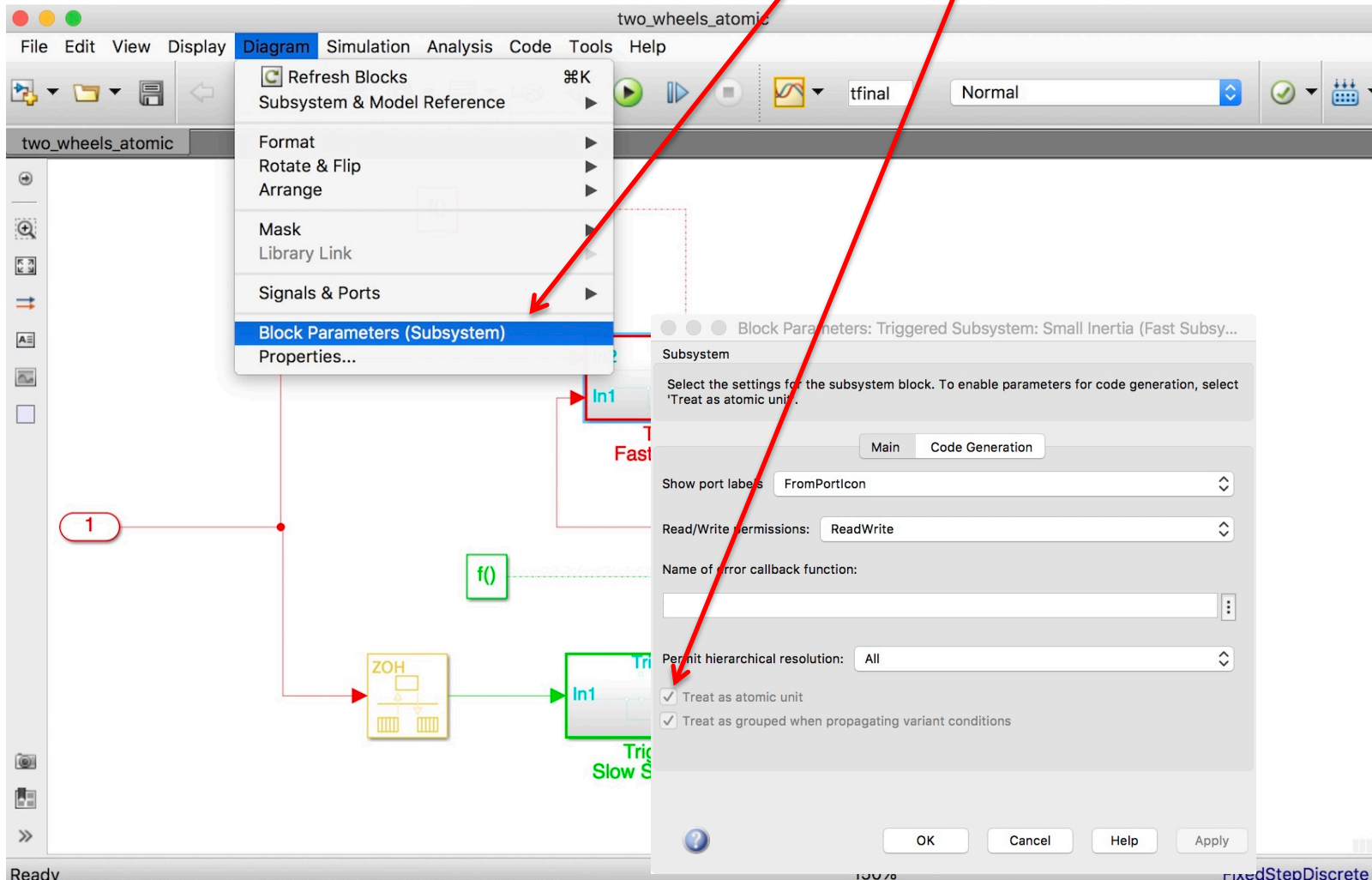
- [-] Main file**
 - [ert_main.c](#)
- [-] Model files**
 - [two_wheels_NOT_atomic.c](#)
 - [two_wheels_NOT_atomic.h](#)
 - [two_wheels_NOT_atomic_private.h](#)
 - [two_wheels_NOT_atomic_types.h](#)
- [-] Utility files**
 - [rtwtypes.h](#)

```
71
72 /* S-Function (fcnallgen): '<Root>/Function-Call Generator' incorporates:
73  * SubSystem: '<Root>/TriggeredFastSubsystem'
74  */
75 if (two_wheels_NOT_atomic_DW.TriggeredFastSubsystem_RESET_EL) {
76   TriggeredFastSubsystem_ELAPS_T = 0U;
77 } else {
78   TriggeredFastSubsystem_ELAPS_T = two_wheels_NOT_atomic_M->Timing.clockTick0
79   - two_wheels_NOT_atomic_DW.TriggeredFastSubsystem_PREV_T;
80 }
81
82 two_wheels_NOT_atomic_DW.TriggeredFastSubsystem_PREV_T =
83   two_wheels_NOT_atomic_M->Timing.clockTick0;
84 two_wheels_NOT_atomic_DW.TriggeredFastSubsystem_RESET_EL = false;
85
86 /* DiscreteIntegrator: '<S1>/Discrete-Time Integrator' */
87 if (two_wheels_NOT_atomic_DW.DiscreteTimeIntegrator_SYSTEM_E == 0) {
88   two_wheels_NOT_atomic_DW.DiscreteTimeIntegrator_DSTATE += 0.002 * (real_T)
89   TriggeredFastSubsystem_ELAPS_T
90   * two_wheels_NOT_atomic_DW.DiscreteTimeIntegrator_PREV_U;
91 }
92
93 /* End of DiscreteIntegrator: '<S1>/Discrete-Time Integrator' */
94
95 /* DiscreteIntegrator: '<S1>/Discrete-Time Integrator1' */
96 if (two_wheels_NOT_atomic_DW.DiscreteTimeIntegrator1_SYSTEM_ == 0) {
97   two_wheels_NOT_atomic_DW.DiscreteTimeIntegrator1_DSTATE += 0.002 * (real_T)
98   TriggeredFastSubsystem_ELAPS_T
99   * two_wheels_NOT_atomic_DW.DiscreteTimeIntegrator1_PREV_U;
100 }
101
102 /* End of DiscreteIntegrator: '<S1>/Discrete-Time Integrator1' */
103
104 /* Subsystem: '<Root>/Out1' incorporates:
```

OK Help

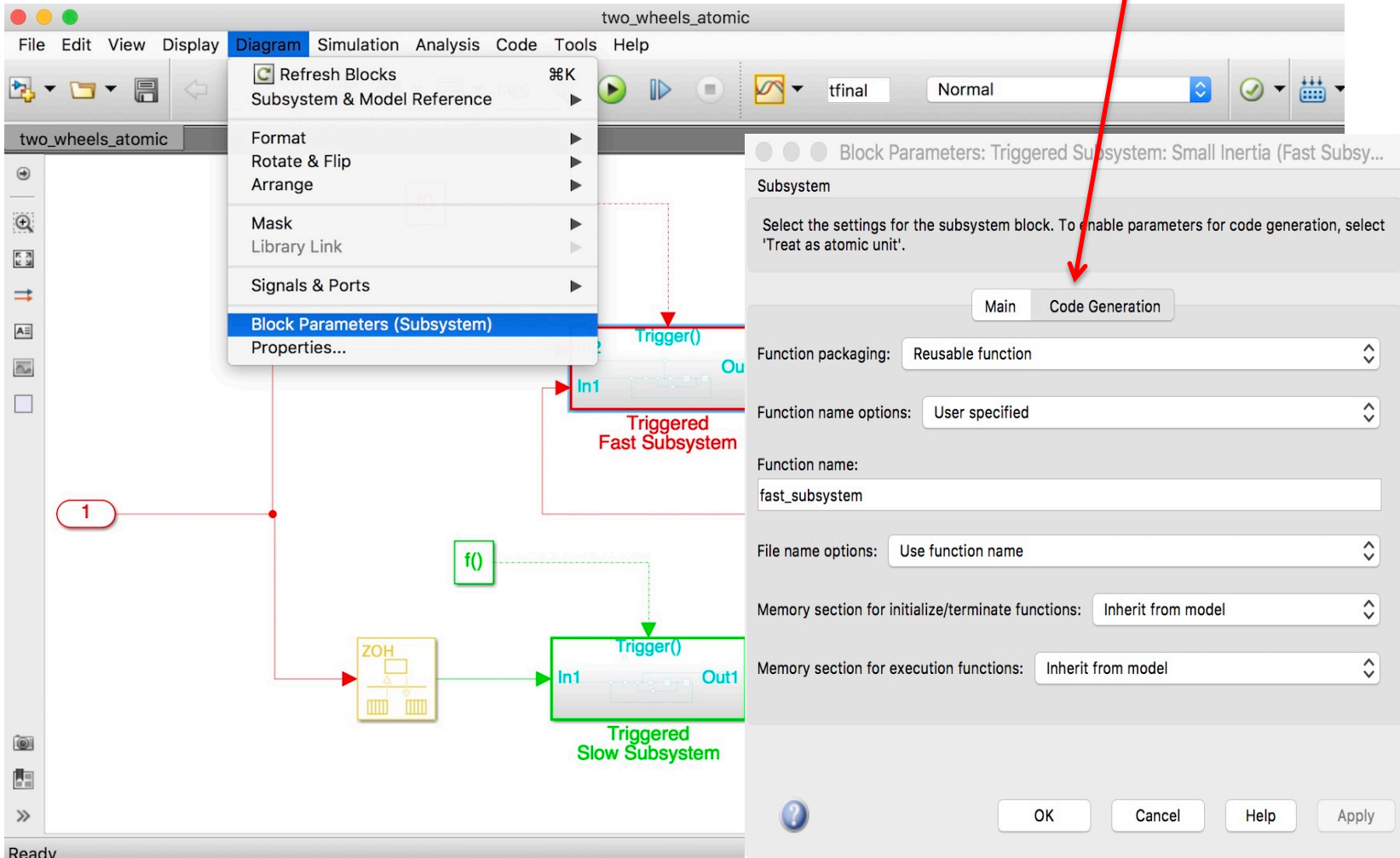
Making subsystems atomic

- 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”

The screenshot displays the 'Code Generation Report' window. On the left, a sidebar lists various reports and generated code files. Under 'Generated Code', the 'Subsystem files' section is expanded, showing a list of files including 'fast_subsystem.c', which is highlighted in yellow. Two red arrows originate from this list: one points to the 'File: fast_subsystem.c' header in the main report area, and the other points to the 'fast_subsystem.c' entry in the 'Subsystem files' list. The main report area shows the C code for 'fast_subsystem.c', which includes a license notice, version information, target selection, and the start of the 'fast_subsystem_Init' function.

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!