



Smart Vehicle Baseline Report

Embedded Software Analysis and
Generation

In support of the University of California
Open Experimental Platform for DARPA –
MoBIES, DARPA Contract F33615-01-C-1841.

Feb 20, 2001

David Bostic
Ford Motor Company
Dearborn, MI 48121
dbostic@ford.com

William P. Milam
Ford Motor Company
Dearborn, MI 48121
wmilam@ford.com

Yanxin Wang
Ford Motor Company
Dearborn, MI 48121
ywang@ford.com

Jeffrey A. Cook
Ford Motor Company
Dearborn, MI 48121
jcook2@ford.com

Trademark and copyright Notices:

- TargetLink is a trademark of dSPACE Inc.
- Embedded Coder is a trademark of The MathWorks, Inc.
- MATLAB, Simulink, Stateflow, and Real-Time Workshop are registered trademarks of The MathWorks, Inc.

Conventions used within this document:

- Informational messages (visible on command line, pop-up windows etc.) and input command messages (user input strings, selection items, etc.) will be indicated by `courier` font.
- **Bold font** will indicate Simulink, Stateflow, and Targetlink block names, signal names, system, and subsystem names.
- Parameter values, variables, and constant values will be indicated by *an italicized font*.

Table of Contents

1	Introduction.....	5
1.1	Code Generation Tools.....	5
1.1.1	TargetLink (TL).....	5
1.1.2	Real-Time Workshop and Embedded Coder (RTW-EC)	5
2	Motivation.....	6
3	Model Selection	6
4	Methodology.....	6
5	Results.....	7
5.1	Fuel-Air Ratio Model	7
5.1.1	TargetLink Code Generation	8
5.1.1.1	Model Modifications Required for Code Generation.....	8
5.1.1.2	Verification	16
5.1.1.3	Metrics	16
5.1.2	Real-Time Workshop with Embedded Coder (RTW-EC) Code Generation.....	20
5.1.2.1	Model Modifications Required for Code Generation.....	20
5.1.2.2	Verification	23
5.1.2.3	Metrics	25
5.2	Transmission Model	28
5.2.1	TargetLink Code Generation	28
5.2.1.1	Model Modifications Required for Code Generation.....	28
5.2.1.2	Verification	29
5.2.1.3	Metrics	29
5.2.2	Real-Time Workshop with Embedded Coder (RTW-EC) Code Generation.....	32
5.2.1.4	Model Modifications Required for Code Generation.....	32
5.2.1.5	Verification	35
5.2.1.6	Metrics	35
6	Summary.....	39
7	Bibliography	40
8	Appendices	41
8.1	APPENDIX A: Derivation of Discrete Integrator Block Transformation.....	41
8.2	APPENDIX B: Tool Version Information	42
8.3	APPENDIX C: Floating-point C Code Generated by TargetLink Fuel-Air Ratio Model	43
8.4	APPENDIX D: Fixed-Point C Code generated by TargetLink for Fuel-Air Ratio Model	47
8.5	APPENDIX E: Floating-Point C Code Generated by RTW-EC for Fuel-Air Ratio Model.....	52
8.6	APPENDIX F: Floating-Point C Code Generated by TargetLink for Transmission Model	56
8.7	APPENDIX G: Fixed-Point C Code Generated by TargetLink for Transmission Model.....	77
8.8	APPENDIX H: Floating-Point C Code Generated by RTW-EC for Transmission Model	102

List of Tables

Table 1: TargetLink Metrics for Fuel-Air Ratio Control	17
Table 2: RTW Embedded Coder Metrics for Fuel-Air Ratio Control	25
Table 3: TargetLink Metrics for Transmission Control	29
Table 4: RTW Embedded Coder Metrics for Transmission Control	36

List of Figures

Figure 1: TargetLink Block Properties Dialogue Box	9
Figure 2: Original Subsystem: port_airflow	9
Figure 3: Subsystem port_airflow after Product Block Transformation.....	10
Figure 4: Statechart afr_fast_ctl Triggers afr_fast_observer.....	11
Figure 5: afr_fast_ctl State Machine	11
Figure 6: Subsystem with Discrete Time Integrator Block.....	12
Figure 7: Delay Block Representation	12
Figure 8: Replacement System for Discrete Integrator Block	13
Figure 9: Property Manager Screen	14
Figure 10: Properties dialogue with active fixed-point boxes.....	16
Figure 11: Input Port Properties Dialogue and Model	21
Figure 12: Data typing blocks positioned around integrator.....	21
Figure 13: Setting Workspace variable data type to single.....	22
Figure 14: Signal Properties Dialogue	23
Figure 15: Stateflow Properties Dialogue	32
Figure 16: Stateflow Explorer Identifiers.....	33
Figure 17: Diagram with Signal Propagation Characters on Output	34
Figure 18: Name Settings for RTW Coder.....	35

1 Introduction

The purpose of this Baseline Report is to evaluate the capabilities of available tools that allow C-code, suitable for use on an embedded microprocessor, to be generated directly from Simulink, Stateflow, or mixed models. This report makes an assessment against the requirements stated in the [Challenge Problem](#) document. Two of the models presented in the Powertrain Challenge Problem are evaluated: Fuel-Air Ratio Control and Transmission Shift Control. The state of the tools with respect to the requirements, and a subjective assessment of usability are presented. The evaluation is based on the MATLAB version 6.0 toolset and other tools that are compatible with version 6.0.

1.1 Code Generation Tools

This document focuses exclusively on dSPACE TargetLink and The Mathworks Real-Time Workshop and Embedded Coder for automatic code generation from MATLAB-based controller models.

1.1.1 TargetLink (TL)

TargetLink version 1.2 is a C code generation and simulation tool developed by dSPACE Inc. The TargetLink tool suite is designed for use within the MATLAB Simulink/Stateflow modeling environment. TargetLink presents a development environment that enables the creation of generic C-code directly from the model representation for both fixed- and floating-point models. TargetLink also provides automated simulation capabilities to allow side-by-side comparison between the simulation environment and the generated C code.

1.1.2 Real-Time Workshop and Embedded Coder (RTW-EC)

Real-Time Workshop is a product of The MathWorks Inc., the company that produces MATLAB, which is the base application for this investigation. As an integrated component to the simulation environment, Real-Time Workshop generates portable and customized code from a Simulink/Stateflow model. Embedded Coder 1.0 is an add-on product to Real-Time Workshop, used to generate C code for specific target microprocessors, and integrates with other capabilities of MATLAB. Built-in menus and configuration options are available to make various adjustments to the generated code.

2 Motivation

The use of simulation models as the complete specification for software behavior is a basic premise of our approach. A benefit of having an executable specification is that it allows a precise comparison to the output of the available automatic C code generation tools [1]. Given the disparate nature of the platforms on which the generated code must execute, understanding the requirements for moving from a simulation-based controller model to actual hardware is an important stage in the controller development process. In the modeling environment, the idealized controller typically operates on a well-resourced workstation with substantial memory, storage, and computational power, as well as various 'helper' functions in the development tools. In the target environment, execution, resource and timing constraints are important considerations [2]. The ability to integrate automatically generated code with existing software is also critical. In this document we will discuss some of the details of configuring a model and software tool environment to generate C code in a format that is appropriate for an embedded application.

3 Model Selection

As stated above, the focus is on two of the controller models given in the Challenge Problem: Fuel-Air Ratio, and Transmission Control. Together, these models demonstrate characteristics that allow important dimensions of the Code Generation Problem to be discussed. The base models, as well as the annotated Simulink/Stateflow and TargetLink models used for code generation, may be found at the [UC Berkeley MoBIES website](#).

Both models have multiple subsystems, with functionality present at different levels, which permits a discussion of code encapsulation into C functions. Also, the complexity of the models allows a significant amount of code to be generated for inspection. Finally, each model covers enough of the modeling block sets to give a comprehensive perspective on code generation capability; the Fuel-Air Ratio model is primarily Simulink-based, while Transmission covers a sizeable portion of the Stateflow semantic.

4 Methodology

In order to ensure a complete analysis, we established a set of methods for evaluation of the code generators. The context of the experiment is that the models represent subsystems for which code is to be generated and eventually integrated into a larger software structure. The report is organized as follows:

The Fuel-Air Ratio model is described in Section 5.1 for both the TargetLink and Real-Time Workshop tools. The required model modifications for successful code generation are detailed. The discussion covers both simple and complex tool settings. The intent is to give the reader a feel for some of the issues involved with the basic operation of the tools. Additionally, we discuss some of the requirements for controlling the output of the tool and managing the characteristics of the generated C code. The models and generated code are then compared in simulation, and the tool capabilities are tabulated with respect to the challenge problem requirements. The generated code is included in the appendices. Section 5.2 addresses the same issues for Transmission Shift Control.

The data used to verify the behavior of the model are derived from a sixty-second drive cycle provided by UC Berkeley. The drive cycle was generated using the plant model developed to exercise the various controller models described in the Challenge Problem Document. It should be noted that the verification test data represent a single test vector. No attempt was made to exhaustively test the generated code. An assessment of correctness for the generated code is made exclusively by the comparison of simulation results for the model and code over the test cycle.

5 Results

In this section we will discuss the models on which we focused our analysis, and then discuss the capability of the automatic code generation tools: first TargetLink, then RTW-EC.

5.1 Fuel-Air Ratio Model

The Fuel-Air Ratio model is primarily data flow. That is, outputs are derived from inputs almost exclusively by the evaluation of algebraic equations. As such, most of the object blocks are from the base Simulink libraries. As a component of the overall controller, the function of this model is to calculate desired fuel (**dMfc**), in kg/second, by taking the following as inputs to drive the calculation:

- we** - engine speed (in radians/sec)
- map** - manifold absolute pressure (in kilopascals)
- bp** - barometric pressure (in kilopascals)
- tps** - throttle position sensor (in degrees)
- o2s** - oxygen threshold sensor (logical/Boolean)

An analysis of the model shows that only some of the inputs have any dynamic influence on the calculation of **dMfc**. The one state machine in the model, **afr_fast_ctl** is used to conditionally execute the subsystem **afr_fast_observer**, which determines the rate at which recalculation of the output value occurs.

5.1.1 TargetLink Code Generation

5.1.1.1 Model Modifications Required for Code Generation

It is unusual to find models created in the MATLAB simulation environment that are entirely suitable for use in an embedded product, regardless of the code generator which must translate them. Typically, some features included in the simulation environment are unnecessary when applied to the embedded processor module. Conversely, embedded code has requirements that must be explicitly modeled to support automatic code generation. The following discussion describes the necessary modifications to the Fuel-Air Ratio model for code generation using TargetLink.

Reference to Time Variable 't' in State Machine "afr_fast_ctl"

In the Stateflow model, this reference to the variable **t**, which is always available in the MATLAB runtime environment, ordinarily would present the current absolute time to the user at the command window. This variable has no comparable use in the embedded system and, since **t** is not being used for any kind of calculation, may be deleted.

Addition of a Function-Call Block to the Top-Level Trigger Input

When model subsystems are separated from their parent systems, there are typically input signals that are unconnected or must be duplicated for behavioral consistency. In this particular model, a top-level trigger input must be reconnected to a trigger that runs at the same rate as if the model were connected. This is critical in registering the sample frequency of the input data to the execution rate of the system.

Configure Top-Level Port Widths

Again, since this model has been separated from a parent system, some information has to be manually added to account for the disconnection. In the simulation model, the port width of the input ports is automatically derived from the parent system (the port width is accessible via the block properties dialogue for the input port block; a value of '-1' indicates the signal width is to be the same as the connecting signal from the parent subsystem). To generate code, these values must be altered to conform to the data width of the input signals. In the case of Fuel-Air Ratio, the input signals are scalar (width = 1).

Conversion to the TargetLink BlockSet (Automated)

TargetLink does not generate C code directly from the Simulink model; rather, the dSPACE tool automatically replaces most of the blocks in the model with TargetLink-specific blocks and adds several blocks beyond those. The conversion to the TargetLink blockset from the Simulink blockset enables a level of software engineering specification to be applied to the model via the TargetLink block properties dialogue box (see Figure 1). The additional blocks, (such as TargetLink Inport and Outport blocks) can be useful in demarcating function boundaries, interfaces, and return values. The TargetLink Main Dialogue block controls the overall parameters for code generation and simulation. Also, the blockset has the capability to capture simulation data that are used for code validation and numeric scaling.

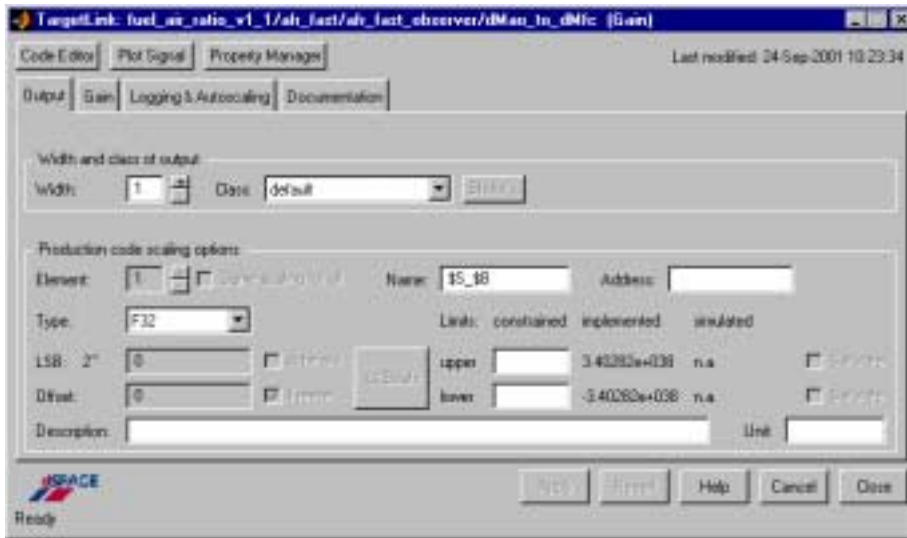


Figure 1: TargetLink Block Properties Dialogue Box

Product and Division Blocks

TargetLink supports both fixed- and floating-point model representations for code generation. For integer or fixed-point calculations, multiplication and division blocks with more than two inputs must be replaced with multiple two-input blocks. For fixed-point code, this permits automatic binary point scaling. This modification is illustrated in Figures 2 and 3.

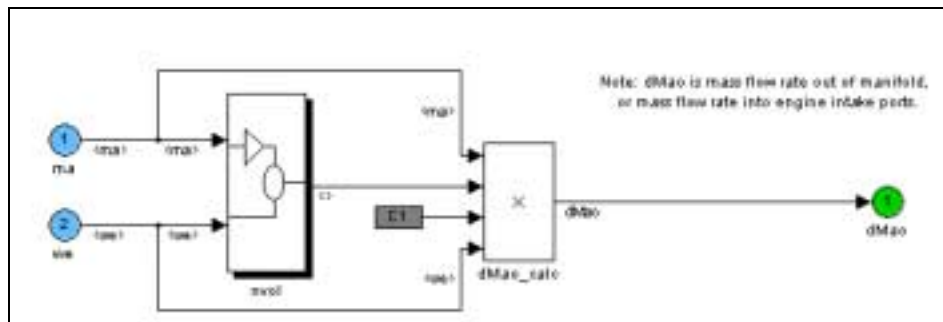


Figure 2: Original Subsystem: port_airflow

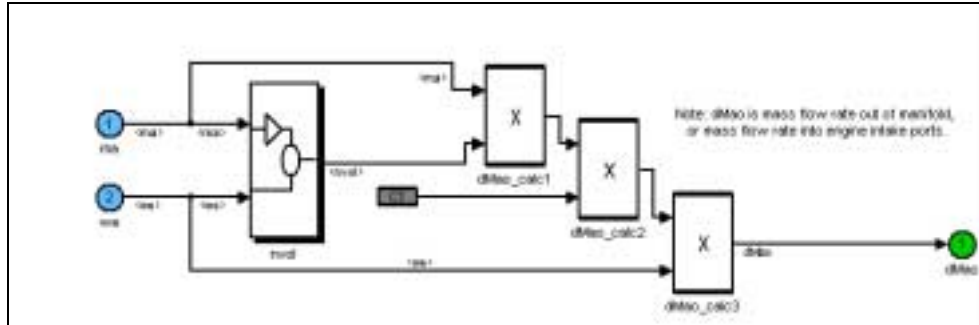


Figure 3: Subsystem port_airflow after Product Block Transformation

Discrete Time Integration

As noted earlier, most of the model has a basic dataflow-driven architecture. There is, however, feedback in one section of the model, and this has interesting implications with respect to code generation. In the modeling and simulation environment, block execution ordering and some timing details are commonly hidden from the user by the modeling tool; in the case of embedded implementations of controllers, these details are critical for repeatability, consistency, and accuracy. In the case of the Fuel-Air Ratio model, the most interesting aspect of the model that must be managed is the execution timing. The top-level trigger signal, **trig_afr_fast**, controls when the calculation for **dmfc** is performed. In Figure 4, one can see that **do_afr_fast_observer** is conditionally executed by **afr_fast_ctl**. Also, one can see in Figure 5 that, since there is only an action associated, the calculation is performed each time, unconditionally, whenever the state machine **afr_fast_control** is triggered. Under these circumstances, the entire model runs in a single time step. In a subsystem subsumed by **afr_fast_observer**, however, there is a calculation using a discrete time integrator block (Figure 6).

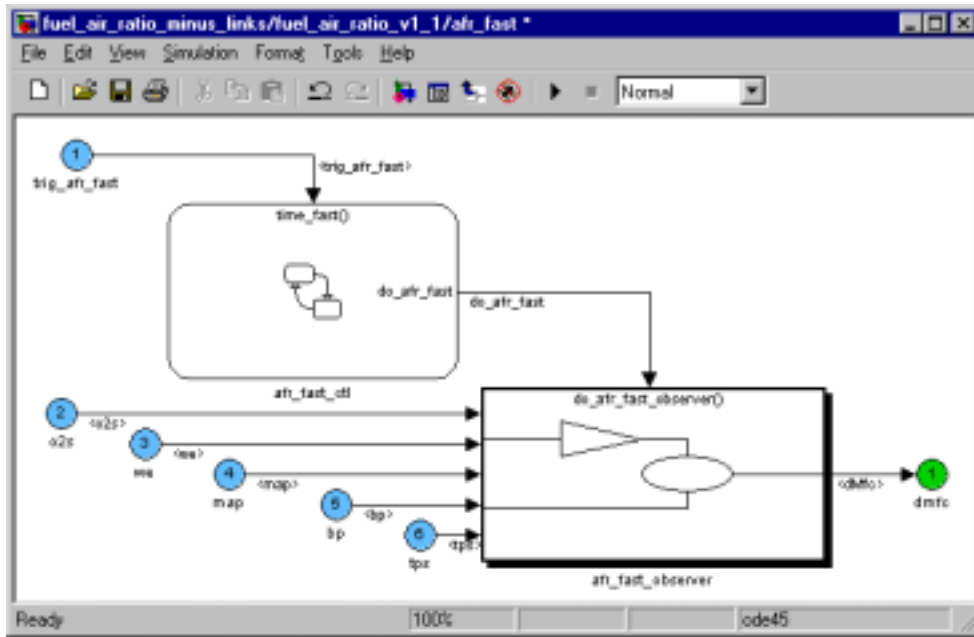


Figure 4: Statechart afr_fast_ctl Triggers afr_fast_observer

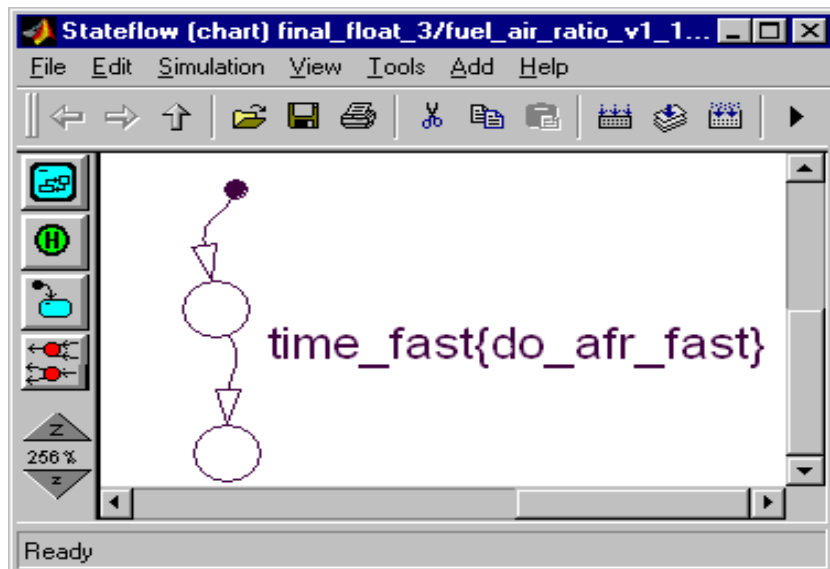


Figure 5: afr_fast_ctl State Machine

When the user attempts to generate code, the following fault indication is given:

"Error #25322: Subsystem/Discrete-Time Integrator: This TargetLink Discrete Integrator block resides in a conditionally executed subsystem."

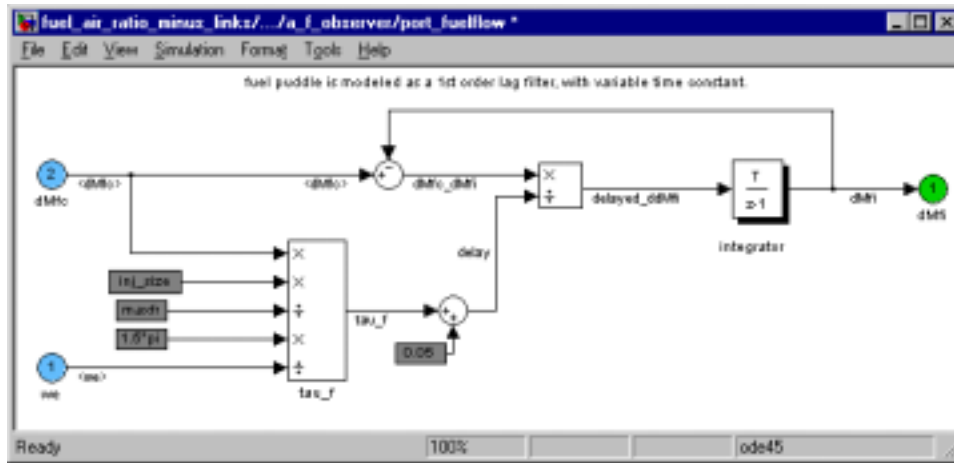


Figure 6: Subsystem with Discrete Time Integrator Block

To understand this error message, one must look at the architecture of the model, the nature of the discrete integrator block, and the requirements for realizing executable code on target hardware.

For many embedded controller applications, unit delay blocks (z^{-1}) are common (see Figure 7). This block operation simply holds the value of the input for one iteration before it is realized at the output of the block. This is sometimes useful in comparing the previous time step value with current time step value for use in various calculations. The implementation of this calculation would be simple because it is time independent; the unit of delay is whatever length of time spans recalculation of the block.

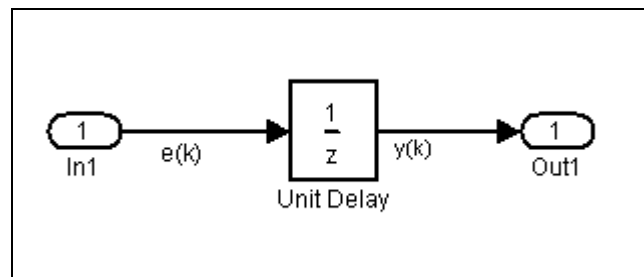


Figure 7: Delay Block Representation

Given some discrete time step k (as represented in Figure 7):

$$\frac{y(k)}{e(k)} = \frac{1}{z} \quad \rightarrow \quad y(k) = e(k-1)$$

The Discrete Time Integrator block, $\frac{T}{z-1}$, is different. This block requires integration that is dependent on the amount of **absolute time** that has lapsed since the last the operation was performed.

So why does TargetLink complain? If the integrator block had not been in a triggered subsystem, the integration time would be derived from the sample time of the model. As it stands, the execution of the block is controlled by a state machine, which could run intermittently depending on the state machine logic. TargetLink detected the existence of this block in a conditionally executed subsystem. In fact, analysis of the model execution shows that all the calculations take place in the same time context as the sampling rate. That this is the case is only coincidental to the architecture of the model, hence the error message.

Typically, embedded application software is free of references to execution, or sample time. Usually each task runs in its own execution context, and this is managed by some external task scheduling mechanism. It is common for the task rates to be fixed, in order to effectively manage processor resources, and as a precaution to ensure schedulability.

This raises the question - how does one translate the model to eliminate this dependency on the Discrete Integrator block without changing the behavior of the model? The following equation gives us an equivalent transformation (see Appendix A for derivation):

$$\frac{y(k)}{e(k)} = \frac{T}{z-1} \quad \rightarrow \quad y(k) = \frac{y(k) + T \cdot e(k)}{z}$$

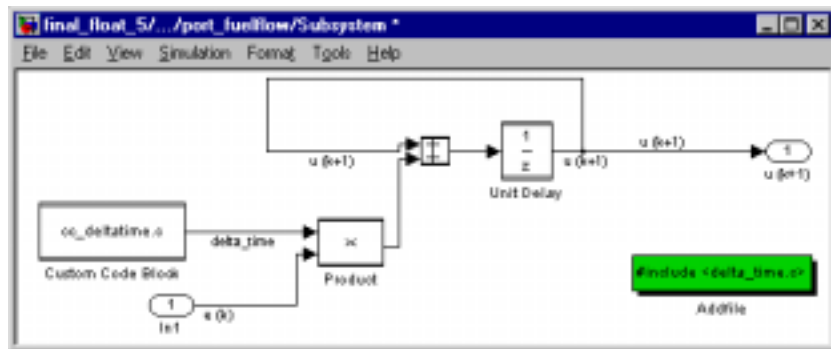


Figure 8: Replacement System for Discrete Integrator Block

Consequently, the discrete integrator block can be replaced by the system shown in Figure 8. The calculation of T originates in the custom code block labeled cc_deltatime. For an embedded implementation, this represents a call to an embedded hardware dependent routine that returns absolute elapsed time since the function was last called. *It is important to note that any transformation of the Discrete Time Integrator is ultimately dependent on an accurate rendering of absolute elapsed time, not simulation or sample time.* TargetLink allows the user to integrate calls

to customized source code into the model to be included in the generated C code, and to run in the simulation mode. A template file for specifying the function prototype is provided.

Beyond the issue of correctness of the implementation, configurability of the tool to produce code with specific software engineering characteristics is an important feature. From this standpoint, TargetLink offers options for adding software detail at the individual block level, and controlling characteristics of the generated code. For the Fuel-Air Ratio code, all data types were converted from the simulation model default 64 bit floating (double) to 32 bit floating-point (F32 float), consistent with typical embedded floating-point implementations.

Data Type Conversion

To select the appropriate output type, the user can select the block properties dialogue for each block. An alternative method for quickly changing block attributes is to use the Property Manager. The TargetLink Property Manager (Figure 9) is a configurable graphical user interface that allows the user to see and/or modify any of the attributes of the blocks and data elements in a model.

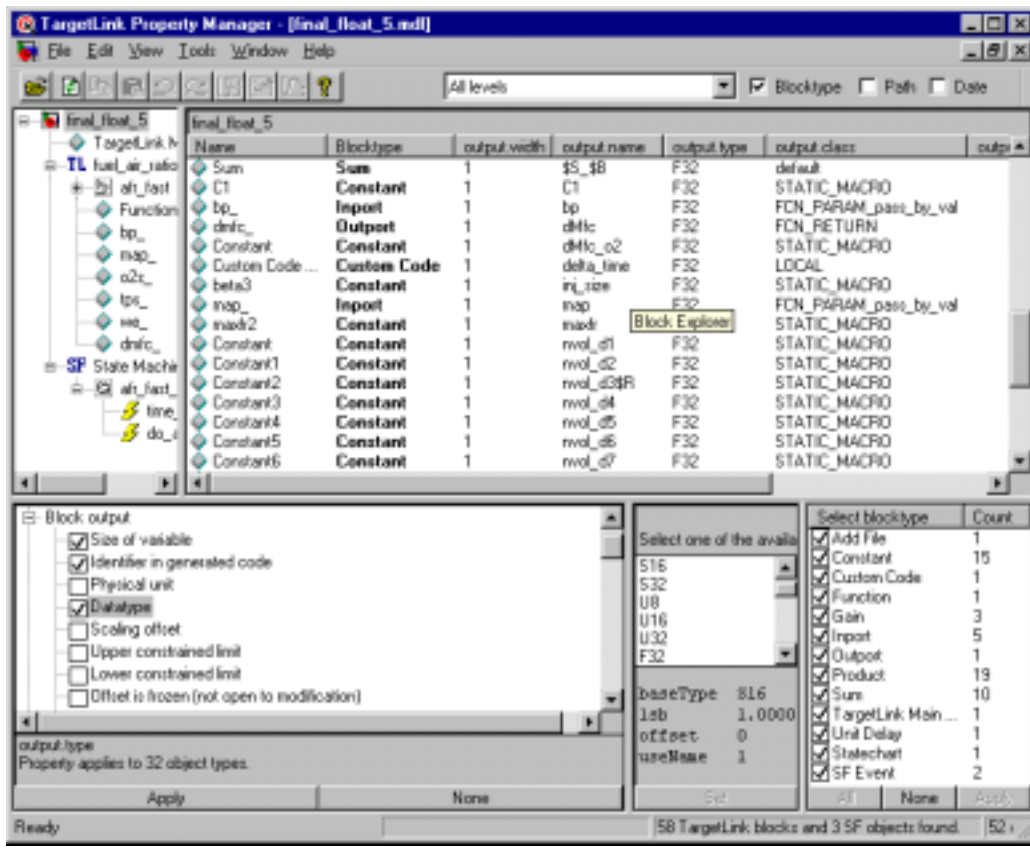


Figure 9: Property Manager Screen

Top-level Inputs as Function Call Parameters (and Return Values)

When a TargetLink system is created, inport and outport blocks are added to define the interface to the system. Within the block properties options, specification of block signals as function arguments and return values is possible.

Function Name

The addition of a TargetLink function block to the **afr_fast_observer** block allows the user to specify the name of the function, **calc_fuel_air_ratio**, realized by that subsystem, via a dialogue entry box in the function block properties block.

Calling Parameters

Specific identifier names, in our case the input and output parameter names, can appear uniquely in code by specifying the name and class (as parameters) in the TargetLink inport and outport properties block.

Named Constants

Code can be generated referencing constants by their literal names associated with the appropriate variable class; in this case, MACRO. These names and their values will appear as #define declarations in the resultant C code.

For fixed-point C code generation (integer calculation only, and assuming a 16-bit processor), there are additional modifications that must be made to the model:

Data Type Conversion

All output types and block calculation parameters must be converted to S16 (16 bit single, the same as MATLAB type Int16). This can be done on a block-by-block basis, or handled through the TargetLink Property Manager, as described above.

Discrete Time Integration

A user-defined function must be inserted in the model to generate a fixed-point time calculation. As mentioned previously, a custom code function was added to reference an embedded hardware timer to perform discrete time integration. In order to use this for fixed-point, the return type of the function must be adjusted, and the C code to read the timer must be changed accordingly.

Fixed-point Scaling

The most important aspect of designing fixed-point C code is the determination of the scaling factor, or binary point for each variable. In this respect the user has several options. Each TargetLink block allows the user to set the binpoint, or use a built-in calculator to determine limits and resolution (see Figure 10). If the user has access to floating-point simulation data, the option exists to execute the simulation, log the value ranges at each block, and let TargetLink automatically scale all the values for the system.

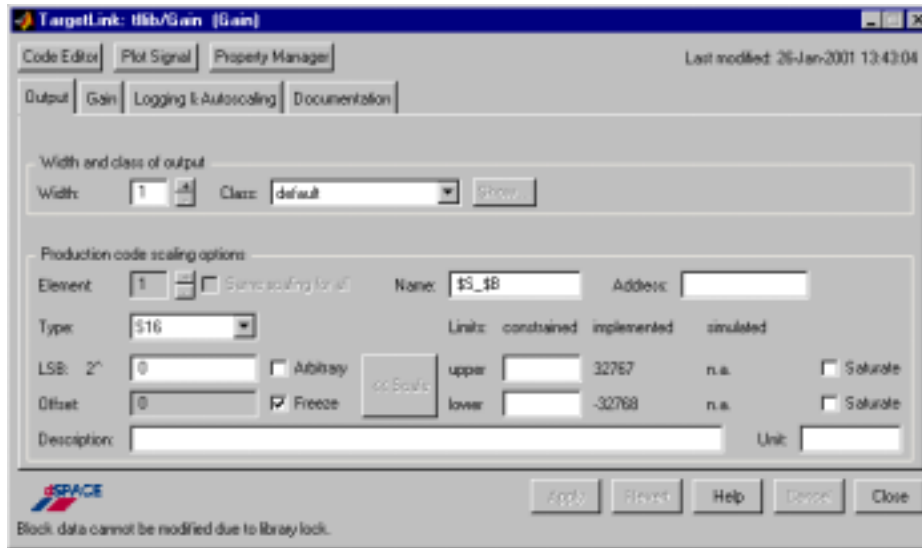


Figure 10: Properties dialogue with active fixed-point boxes

5.1.1.2 Verification

An important aspect of having a tool that automatically generates target C code is the ability to verify the code against the original model. This serves to give the user confidence that the modifications made for software engineering purposes have no significant or unpredictable impact on the function [2].

The TargetLink tool provides a tightly integrated environment that allows code generation and simulation of both the original idealized model and the C code derived from the model.

After the code is generated, the user has several options. TargetLink builds a customized simulation frame that manages all the details of instrumenting the generated C code, and the user has several methods for determining which block outputs to monitor and plot. A separate menu option allows the user to generate code from the idealized (full 64-bit resolution) model for comparison with the target fixed-point code.

5.1.1.3 Metrics

Section 3.2 of the Challenge Problem Document enumerated the required attributes for a code generator. These translate the behavioral specification of the model into a desired C code format. In the table below, these criteria are assessed for the TargetLink tool operating on the Fuel-Air Ratio model.

Table 1: TargetLink Metrics for Fuel-Air Ratio Control


Item	Challenge Problem Criteria	Assessment
	Fuel-Air Ratio Controller	
A	ANSI C (stated compliance)	YES
B	Preserved Functionality (Simulink)	YES
C	Preserved Functionality (Stateflow)	N/A
D	ROM / RAM / CPU Efficiency	200/32/* bytes
E	Traceable / Readable	YES
F	System Independent Functions	YES
G	Automated Usage	YES
H	Discrete Time Support	YES
I	Call to UserCode	YES
J	Call from User Code	YES
K	Variable Name Control	YES
L	Variable Type Control	YES
M	Type Qualifiers	YES
N	Storage Class Control	YES
O	Variable Scoping Control	YES
P	Variable Initialization	N/A
Q	Variable Declaration	N/A
R	Function Partitioning	YES
S	Function Prototyping	YES
T	File Partitioning	YES
U	C Structures	NO
V	User Comments	YES
W	Processor Optimization	YES

Table Note Key:

- A. dSPACE states ANSI Compliance. Per all visual inspections to date, the code generated has complied with the ANSI/ISO 9899-1990 standard, although no formal validation has been performed.
- B. For the tested models from the Powertrain Challenge Problem, no significant difference between the generated code and the model was detected. Some minor differences are attributed to the resolution of the datatyping options selected (see Appendices).
- C. Not Applicable for the Fuel-Air Ratio model, as there is no significant Stateflow usage.
- D. See Appendix B for systems configuration details for these results. Though no execution (CPU) measurements were taken, code size is a broad indicator of execution performance.
- E. For the given model, the correlation between the representation in Simulink and the source code syntax is good. This, however, is dependent on the configuration of the model including

selection of function interfaces and boundaries, and decisions about choosing variable and constant names. In other words, if one lets TargetLink, (which uses methods to prevent variable and system name collisions) make all the naming decisions, it will be more difficult to correlate the resultant code with the original model.

- F. The generated code is completely free of RTOS calls, hardware dependencies, and low-level drivers. Though not tested, options exist in the tool to take advantage of platform and compiler specific optimizations.
- G. Each of the TargetLink blocks has a documented API for reading and setting all of the configuration options that could be altered via the GUI. As well, the TargetLink Main Dialogue Box, the means by which the overall code generation and simulation system is configured, is controllable via a published API.
- H. A majority of the discrete time blocks available in MATLAB Simulink are supported under TargetLink.
- I. Function calls to external code are available via the custom code block. The user must configure a template file that describes the calling interface. The source code must be accessible and resolvable, so that the build process for the entire model can be performed.
- J. The top-level interface for the function that represents the model is definable at the system level. Thus, many type, storage class, and pass-by options can be configured if TargetLink has been made aware of, and supports the particular data configuration.
- K. The user has control over the name that will appear in the code for each block output. If left unselected, the tool uses a macro scheme to uniquely resolve each output.
- L. The user has control over the C data type that will appear in the code for each block output. This must be selected for each block, or else the user will be left with the default data type (usually int16). A configuration file is available to alias data types to names of one's own choosing (e.g. float to F32), which then display as menu options in each block.
- M. The user has control over the storage class, type qualifier and/or macro definition that will appear within the data object initialization/declaration. The user may use the supplied class objects, or create customized classes via a configuration file. All classes are then available from the TargetLink block's property menu.
- N. Please see explanation under criterion 'Type Qualifier'.
- O. Please see explanation under criterion 'Type Qualifier'.
- P. To insure proper model simulation, the Simulink and Stateflow modeling tools handle variable initialization. TargetLink uses those initialization values.

- Q. To insure proper model simulation, variable declaration is handled by the Simulink and Stateflow modeling tools. These variables are then accessible in the TargetLink tool environment.
 - R. A specialty block added to a subsystem allows it to be generated as a separate function, with a name of the user's choice. Control over the function parameter interface is managed by the TargetLink inport/outport blocks.
 - S. Function prototypes are generated automatically in the associated include file for each specified function.
 - T. Specification of the name of the file in which the function should reside may be accomplished using the TargetLink Function Block.
 - U. Support for definition of variables within structures or notation to access C structures is not available.
 - V. Most blocks allow the user to add comments. The user has the option of having the comment appear in the code either at the declaration of the block variable, or at the location of the blocks evaluation.
 - W. For supported compiler/processor pairings, two levels of optimization are available: the first level adjusts ANSI C compliant constructs for enhanced efficiency, while the second level permits non-ANSI C extensions and assembly code.
- 

5.1.2 Real-Time Workshop with Embedded Coder (RTW-EC) Code Generation

5.1.2.1 Model Modifications Required for Code Generation

As with TargetLink, there are certain modifications required to the Fuel-Air Ratio model to enable code generation using Real-Time Workshop with Embedded Coder. Specifically:

Discretize the Simulation Environment

In the Simulation Parameters 'Solver' tab, one must select 'Fixed Step Solver' in the Solver Option Box and 'Mode Single Tasking' in the Mode box. Together, these options inform the code generator that no additional code to manage variations in time step or time contexts is necessary.

Addition of a Function Call Block to the Top Level Trigger Input

As previously discussed, a top-level trigger input must be incorporated to register the sample frequency of the input data to the execution rate of the system.

Select the Appropriate System Target File

In the Real-Time Workshop pane of the Simulation Parameters dialogue box, the user should select `ert.tlc` (Embedded Real-Time Coder) as the TLC (Target Language Compiler) file from which code generation will occur. Of the available coder options, this one is most suitable for generating embedded C code from the model.

It should be noted that selection of the 'Generate code only' checkbox will save the user from having to wait for a complete build and compilation of the model code.

As discussed in section 5.1.1.1, the characteristics identified below are important to realize in the generated C code for, among other things, data element specification, integration, and traceability.

Function Interface Control

Under RTW, there is no practical way to partition software functions in the generated code because the MATLAB code generation framework creates its own calling hierarchy and initialization routines. Neither is there a way to control the prototype, even when functions are created using the atomic unit option, selectable from each subsystem's block properties. Also, if a trigger penetrates a system, atomic function boundaries are disallowed. Furthermore, RTW-EC does not support a stand-alone function architecture that allows the user to uniquely specify return parameters. The framework for executing functions and returning values is embedded in an RTW-specific format.

Data Type Conversion

At the top level, the inputs to the model **we**, **map**, **bp**, **tps**, and **o2s** can be typed by double clicking the port connector bubble, and selecting the desired type (see Figure 11). These type designations (`single`, `double`, `int16`, `uint16`, `int8`, `uint8`, `Boolean`, `int32`, `uint32`,) are specific to MATLAB and cannot be aliased or defined to other type names. In the generated code, a double will be represented by type specifier `real_T`, whose type is defined by MATLAB at compile time.

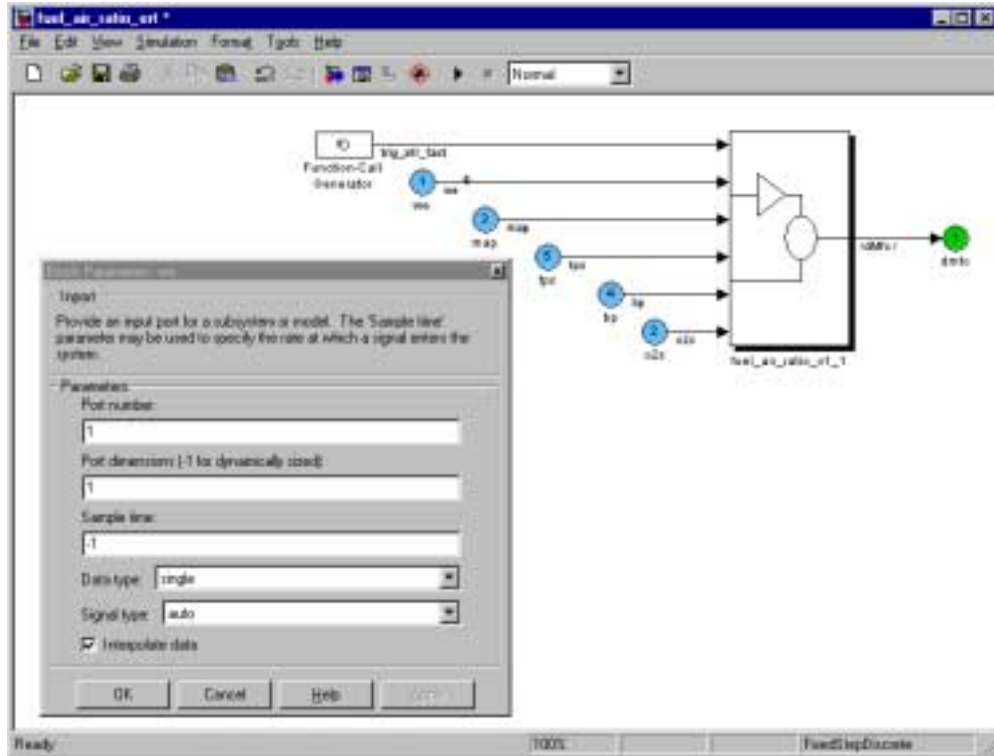


Figure 11: Input Port Properties Dialogue and Model

Certain MATLAB blocks do not accept data-typed signal lines as input, in which case input data must be converted to the type required by the block (see data type conversion block in Figure 12). The output of these blocks must be re-converted to the user-desired type. These modifications are required for the Discrete Time Integrator block in the Fuel-Air Ratio model.

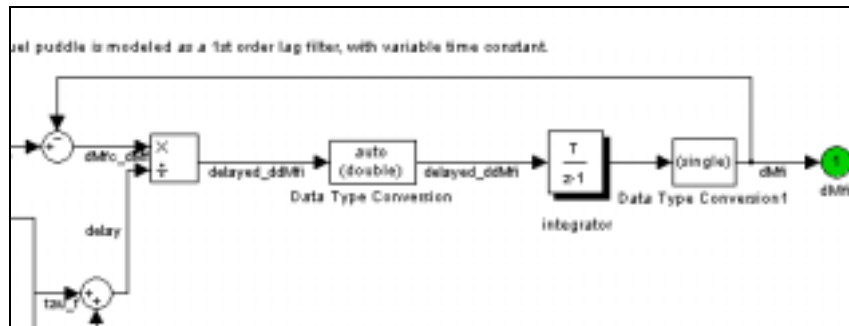


Figure 12: Data typing blocks positioned around integrator

In this particular model, many named constant variables originate in the workspace, and are automatically typed as double. In order to get the data type correct in the generated code, it was necessary to go to the particular block instance and ‘typecast’ the variable. The user interface for this operation is illustrated in Figure 13.

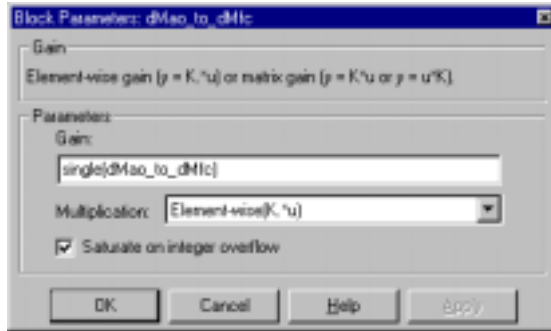


Figure 13: Setting Workspace variable data type to single

Calling Parameters

To make the generated code more traceable, it's desirable to have identifiers in the model represented by the same variable in the generated code. To achieve this, we use the signal properties dialogue. Both options `ImportedExtern`, and `ExportedGlobal` will yield unmangled identifiers in the code, but do not cover the range of storage classes and qualifiers available in C. As shown in Figure 14, the RTW Storage Class Qualifier in the signal properties dialogue box allows the user to type in text to be used as storage class types in the generated code. These strings are unchecked character strings, but will appear in the declarations of the generated code, assuming the user has taken steps to specify the declarations.

Named Constants

This is the desired method for specifying data elements that are unchanging because, as `#defines` they are replaced at compile time and do not consume ROM. There does not seem to be a user-accessible way to make named workspace constants into `#defines`. Under RTW-EC, the best way to get constant variables that are not embedded in a MATLAB structure to appear as named in the generated code is, in the `Advanced` tab of the `Simulation Parameters` dialogue, to select the `Inline Parameters` checkbox. This will enable the `Tunable Parameters` button. The user may then name variables that should be name accessible, for instance a calibration variable. For the unchanging workspace variables, the qualifier `const` was selected. The user may also add information to the variable via the 'storage type qualifier' type-in box, but this is merely a text string prefix, and is not checked against any ANSI C keyword or type.

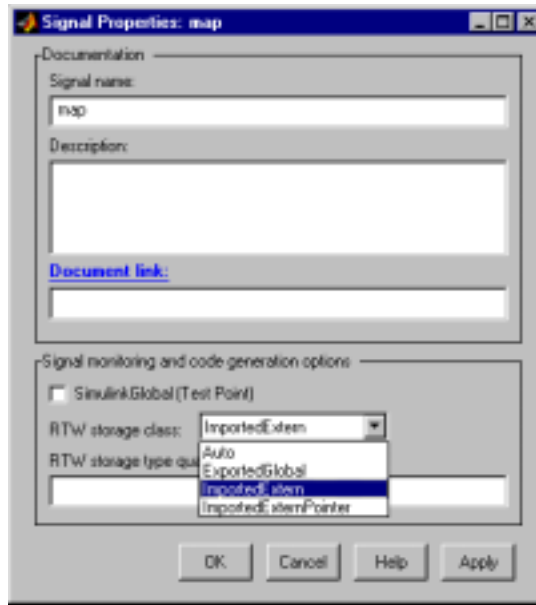


Figure 14: Signal Properties Dialogue

5.1.2.2 Verification

The software engineering detail added to the model in MATLAB allows Real-Time Workshop to customize the characteristics of the generated code. As discussed above, these customizations make it easier to integrate the generated C code into other applications, or reference data elements specified in other modules. The type specification capability allows target-specific data sizing to be performed.

Verification was accomplished by encapsulating the generated code in a MATLAB S-Function for simulation and comparison with the original Simulink model. The process for creating an S-function wrapper for this purpose is well documented in the [Writing S-Functions](#) manual. A number of verification issues are discussed below:

Storage Class Issues

The ability to integrate automatically generated code with existing collections of source code is a critical element for migrating to a more efficient software development process. Thus, for the Fuel-Air Ratio model, it was assumed that this code module would be integrated into some larger software structure. Thus, top-level input signals were designated `ImportedExtern` to indicate they were inputs to the model. There were no errors given when code was generated for this model. When, however, the attempt was made to build the code for simulation, the following error resulted:

```
NMAKE : fatal error U1077: 'D:\MATLABR12EC\bin\win32\mex' : return code '0xff'
```

Backtracking through several steps uncovered the root of the problem. The storage class of the top-level inputs was then changed to 'Exported Global', thus declaring that the system inputs originate internally.

Basic Data Typing

At the top level of the model, it was desired that the inputs be 32-bit floating-point variables (float), the default for an automotive floating-point embedded processor. For simulation, the workspace I/O tab (in Simulation Parameters) referenced a data structure with appropriate input data. Care was taken to cast the data into a 32-bit float format to ensure compatibility. Upon trying to run the simulation, the following error was generated:

```
Invalid data type for inport block 'fuel_air_ratio_ert/we'. The external inputs specified for the block diagram indicate that the data type should be 'double'. Instead, the data type is being set to 'single'
```

It was determined that the data structure must be changed from the MATLAB default:

```
Time
Signals.values
Signals.dimensions = X;
```

To:

```
Time
Signals(1).values
Dimensions = 1;
.
.
.
Signals(X).values
Dimensions = 1;
```

That is, in order for the compiled code to execute as an S-Function in Simulink using 32-bit floating-point inputs, the input data structure had to be changed from the multi-dimensional MATLAB default, to a structure consisting of individual vectors for each input.

The code generated under RTW-EC is targeted for use in a MATLAB-specific environment. Analysis of the code reveals built-in macros and system calls that were unspecified by the user, but are generated to maintain consistency with the MATLAB environment.

Discrete Time Integrator

Embedded Real-Time Coder generates C code for, and builds the executable file for the Fuel-Air ratio model with the Discrete Time Integrator block without complaint. As stated before, it is coincidence that the time integration step is equivalent to the model sample time. If the state machine that drives the calculation were to run conditionally, the result would be incorrect. The generated code must be dependent on an absolute time reference (in target hardware) in order to perform the proper integration.

Conditional Compilation

The simulation results are, of course, obtained from executing compiled source code. Examination of the source (.c) and header (.h) files reveals a dependence on preprocessor defines in order to select the appropriate syntax for compilation. In our circumstance, the variable "ERT" must be defined during the compile to build the correct code.

5.1.2.3 Metrics

Code Generator Feature Comparison Table

In this section, the Challenge Problem Document and metrics are tabulated for the real time workshop with embedded coder for the fuel-air ratio model.

Table 2: RTW Embedded Coder Metrics for Fuel-Air Ratio Control

Item	Challenge Problem Criteria	Assessment
	Fuel-Air Ratio controller	
A	ANSI C (stated compliance)	YES
B	Preserved Functionality (Simulink)	YES*
C	Preserved Functionality (Stateflow)	N/A
D	ROM/RAM/CPU Efficiency	876 / 56 / * bytes
E	Traceable / Readable	NO
F	System Independent Fcns.	NO
G	Automated Usage	NO
H	Discrete Time Support	NO
I	Call to UserCode	NO
J	Call from User Code	NO
K	Variable Name Control	PARTIAL
L	Variable Type Control	PARTIAL
M	Type Qualifiers	PARTIAL
N	Storage Class Control	PARTIAL
O	Variable Scoping Control	NO
P	Variable Initialization	YES
Q	Variable Declaration	YES
R	Function Partitioning	PARTIAL
S	Function Prototyping	NO
T	File Partitioning	PARTIAL
U	C Structures	NO
V	User Comments	NO
W	Processor Optimization	NO

Table Notes:

- A. RTW-EC states ANSI C Compliance. Per visual inspections, the generated code complies with the ANSI/ISO 9899-1990 Standard, although no formal validation has been performed.
- B. Yes, conditionally. See explanation in discussion on RTW Embedded Coder verification.
- C. Not Applicable for Fuel-Air Ratio Model.
- D. See Appendix B for systems configuration details for these results. Though no execution (CPU) measurements were taken, code size is a broad indicator of execution performance.
- E. For the given model, the correlation between the representation in Simulink and the C code is difficult. There are several '.c' and '.h' files generated, and it is not clear which is being called without close examination of all of the generated functions. In addition, there are many functions (and/or macros) interspersed in the files that have no obvious computational correlation to the model. These functions are generated to support the simulation framework for models intended for the MATLAB environment. To determine their effect on the generated code, the user must backtrack through source code located in MATLAB system directories (referenced by named '.h' files). Also, there are sections of C code in the headers that are conditionally compiled, making tracking more difficult.
- F. The generated code is dependent on MATLAB specific system calls and macros/ include files, several of which chain into other include files.
- G. In order to generate useful C code for an embedded application, options in different dialogue boxes and menus must be appropriately configured. Documentation is sometimes less than clear, occasionally resulting in a "trial and error approach" to code generation. For example, when the option "Enable MAT-file logging" is selected, RTW-EC produces C code that executes only for the duration of the simulation time.
- H. Embedded Coder 1.0 will generate C code for models containing Discrete Time Blocks. See Verification discussion section.
- I. No simple method is available, but it can be accomplished by integrating the code into an S-Function. The user must be adept at building S-Functions, generating DLL files, and writing TLC (Target Language Compiler) code in order to get the proper interface to the model code.
- J. No configurable function call interfaces can be generated, and function calls depend on functions/macros generated in other locations.
- K. Signal names may serve as variable names that appear in the code. There is some control for naming variables; the user has control over the name that will appear in the code for each block output.

- L. At the top level of the model, users may select data types of the incoming variables. Type conversion blocks (back to double) may need to be inserted around Simulink blocks that do not support specific data types. There is no way to alias MATLAB data types (e.g. Real_T) to a user specified type (e.g. float or aliased_type).
- M. In the signal properties dialogue box, options for adding qualification strings to identifiers are allowed. These are text strings that will be added to the identifier declaration, but are not understood by the code generator as C keywords.
- N. User can choose between `ImportedExtern`, `ExportedGlobal`, `ExternPointer`, or `Auto`. This is incomplete with respect to available C storage classes. Note: `Auto` is not to be confused with the C storage class of the same name. In MATLAB, `Auto` tells the code generator to select a storage class automatically.
- O. No control for specifying variable scopes is available.
- P. Variables can be initialized using methods typical for simulation in Simulink and Stateflow (for example by using fields in the Stateflow Explorer, constant blocks, and workspace variable definitions).
- Q. Variables can be declared using methods typical for simulation in Simulink and Stateflow (for example by using fields in the Stateflow Explorer, constant blocks, and workspace variable definitions).
- R. Subsystems can be specified as atomic units (and subsequently, functions) via a properties submenu. Care must be taken not to affect the model execution behavior in selecting this option.
- S. There is no way to specify the function prototype for systems demarcated as functions.
- T. For subsystems labeled atomic and declared as functions, C file names may be specified.
- U. C structures exist in the generated code, but are not configurable by the user; most automatically specified data elements are maintained in structures by default.
- V. No facility for controlling the creation of comment fields is available.
- W. No processor or compiler target-specific optimizations are available.



5.2 Transmission Model

The transmission model is an example of a subsystem weighted heavily toward control flow. The model, consequently, consists primarily of state machines represented by Stateflow constructs.

The model describes controls for a 4-speed automatic transmission. At the top level of the model there are 5 inputs:

tps	- throttle angle (in degrees)
vss	- vehicle speed (meter per second)
trig_slow	- slow trigger signal (20 milliseconds)
p_e_switch	- state of the power/economy switch (logical)
a_gear	- actual gear commanded (1,2,3 or 4)

There are also five outputs:

pc1	- clutch1 pressure (kPascals)
pc2	- clutch2 pressure (kPascals)
pc3	- clutch3 pressure (kPascals)
pc4	- clutch4 pressure (kPascals)
pb12	- pressure for the 1-2 band clutch (kPascals)

In addition, the model includes 2-dimensional look-up tables, which contain calibration parameters, such as:

- 1) **shift_speed_12, shift_speed_21, shift_speed_23, shift_speed_32, shift_speed_34, shift_speed_43**. The x values in the tables are throttle angle in degrees. The y values are vehicle speed in km/h at which a shift should occur for the given throttle angle input.
- 2) **c1_table, c2_table, c3_table, c4_table, b12_table**. The y values in the tables are double values that indicate whether a particular clutch is engaged (either 0 or 1). The x values for the tables represent the next gear.

5.2.1 TargetLink Code Generation

5.2.1.1 Model Modifications Required for Code Generation

Many of the modifications necessary to allow the model to simulate and generate embeddable C code in TargetLink were discussed in Section 5.1.1.1 of the Fuel-Air Ratio model. Specifically, the tasks of block translation, data type application, and applying functional interface specifications. Modifications required for TargetLink code generation from the Stateflow transmission model are described below:

Fixed-point Scaling

In section 5.1.1.1, the determination of the scaling factor, or binary point for each variable was managed via parameters available in the properties of each TargetLink block. Fixed-point scaling for Stateflow blocks requires additional modifications.. Stateflow variables must be typed as single

or double. TargetLink then overlays integer type, scaling information, and other code generation attributes on the Stateflow base type. Fixed-point scaling for each variable can be adjusted manually. To scale the variables automatically, the user must provide minimum and maximum value information via Stateflow Explorer.

5.2.1.2 Verification

The generated code was verified by creating a model containing both the original controller and the generated code. In this case, the floating-point TargetLink generated C code was tested. To the extent that the selected test vectors exercised the system, the generated C code exactly replicated the simulation specification. For the general methodology description, see Section 4.

5.2.1.3 Metrics

Section 3.2 of the Challenge Problem documentation enumerated the required attributes for a code generator. In the table below, these criteria are assessed for the TargetLink tool operating on the Transmission model.

Table 3: TargetLink Metrics for Transmission Control

Item	Challenge Problem Criteria	Assessment
	Transmission Model	
A	ANSI C (stated compliance)	YES
B	Preserves Functionality (Simulink)	YES
C	Preserves Functionality (Stateflow)	YES
D	ROM / RAM / CPU Efficiency	3372 / 80 / * bytes
E	Traceable / Readable	YES
F	System Independent Functions	YES
G	Automated Usage	YES
H	Discrete Time Support	N/A
I	Call to User Code	N/A
J	Call from User Code	YES
K	Variable Name Control	YES
L	Variable Type Control	YES
M	Type Qualifiers	YES
N	Storage Class Control	YES
O	Variable Scoping Control	YES
P	Variable Initialization	N/A
Q	Variable Declaration	YES
R	Function Partitioning	YES
S	Function Prototyping	YES
T	File Partitioning	YES
U	C Structures	NO
V	User Comments	YES
W	Processor Optimization	YES

Table Note Key:

- A. dSPACE states ANSI Compliance. Per all visual inspections to date, the code generated has complied with the ANSI/ISO 9899-1990 standard, although no formal validation has been performed.
- B. For the tested models from the Powertrain Challenge Problem, no significant difference between the generated code and the model was detected. Some minor differences are attributed to the resolution of the data typing options selected.
- C. Per the validation experiment, functionality of the generated C code appears to match the behavior of the model.
- D. See Appendix B for systems configuration details for these results. Though no execution (CPU) measurements were taken, code size is a broad indicator of execution performance.
- E. For the given model, the correlation between the representation in Simulink and the source code syntax is good. This, however, is dependent on the configuration of the model, including selection of function interfaces and boundaries and decisions about choosing variable and constant names. In other words, if one lets TargetLink (which uses methods to prevent variable and system name collisions) make all the naming decisions, it will be more difficult to correlate the resultant code with the original model. Correlation of Stateflow diagrams with multiple sub-states and transitions is a complex process under any circumstances; the intricacies and subtleties of the graphical notation of Stateflow will not usually result in a simple implementation in C code. Correlation is simplified by the TargetLink naming scheme, which prepends identifiers with logical subsystems and statechart tags.
- F. The generated code is completely free of RTOS, driver, or hardware-dependent calls (see 'Optimization' item for the exceptions).
- G. Each of the TargetLink blocks has a documented API for reading and setting all of the configuration options that could be altered via the GUI. Data elements in Stateflow can be accessed and/or modified via a documented API. The TargetLink Main Dialogue Box, the means by which the overall code generation and simulation system is configured, is controllable via a published API.
- H. A majority of the discrete time blocks available in the MATLAB/Simulink libraries are supported under TargetLink, though no discrete time elements are present in the Transmission model.
- I. No external function calls are present in the Transmission Model. TargetLink can, however, utilize the native Stateflow mechanism for accessing external functions with some additional configuration.

- J. The top-level interface for the function that represents the model is definable at the system level. The TargetLink inport and outport blocks allow specification of the function-calling interface. Thus, many type, storage class, and pass-by options can be configured. For the transmission model, an atomic function boundary is defined for subsystem **trans_slow_torque**. In the translation process to the TargetLink blockset, the function boundary attributes are preserved.
- K. The user has control over the name that will appear in the code for each block output. If left unselected, the tool uses a macro scheme to uniquely resolve each output. Identifiers named in Stateflow Explorer may even be aliased to other code-generated names at the users discretion.
- L. The user has control over the C data type that will appear in the code for each block output. This must be selected for each element, or else the user will be left with the default data type (usually int16). A configuration file is available to alias data types to names of one's own choosing (e.g. float to F32), which are then selectable as menu options in each block. In Stateflow, only data types of single or double can be aliased to TargetLink types.
- M. The user has control over the storage class, type qualifier and/or macro definition that will appear within the data object initialization/declaration. The user may use the supplied class objects, or modify the add entries to create customized classes via a configuration file. All classes are then available from the TargetLink block's property menu.
- N. Please see explanation under criterion 'Type Qualifier'.
- O. Please see explanation under criterion 'Type Qualifier'.
- P. To insure proper model simulation, the Simulink and Stateflow modeling tools handle variable initialization. TargetLink uses those initialization values.
- Q. Variable declaration is handled by the Simulink (via signal properties) and Stateflow (via Explorer) modeling tools. These variables are then accessible in the TargetLink tool environment. TargetLink can create new variables as outputs
- R. A specialty block added to a subsystem allows functions to be generated separately, with a name of the user's choice. Control over the function parameter interface is managed by the TargetLink inport and outport blocks.
- S. Function prototypes are generated automatically in the associated include file for each specified function.
- T. Specification of the name of the file in which the function should reside may be accomplished via options available in the TargetLink function block.
- U. Support for definition of variables within structures, or notation to access C structures is not available.

- V. Most blocks allow the user to add comments. The user has the option of having the comment appear in the code either at the declaration of the block variable, or where the block is evaluated. In Stateflow, data objects can have text descriptions appear at their instantiation.
- W. For supported compiler/processor pairings, two levels of optimization are available: the first level adjusts ANSI C compliant constructs for enhanced efficiency, while the second level permits non-ANSI C extensions and assembly code.

5.2.2 Real-Time Workshop with Embedded Coder (RTW-EC) Code Generation

5.2.1.4 Model Modifications Required for Code Generation

Several of the configuration items necessary for code generation under RTW-EC were covered in the discussion on the Fuel-Air Ratio model. We refer the reader to Section 5.1.2.1 for a discussion of those items. Here, we will discuss the changes necessary to generate code that are unique to the Transmission model.

Configuration of Stateflow Typing Properties

The default settings for data elements in both Simulink and Stateflow is double, a 64-bit floating-point data type. In order to enforce consistent data typing for other types, the user option 'Use Strong Data Typing with Simulink I/O' is recommended (Figure 15). If this option is checked, the statechart block will accept and generate signals of any data type supported by Simulink. The input signal type will be **required** to match the type of the corresponding chart input data item, otherwise, a type conflict error will be generated. This will help prevent unintentional type conversion to doubles on the part of Stateflow.



Figure 15: Stateflow Properties Dialogue

Data Type Conversion

Since many elements of data typing under RTW-EC were discussed in the Section 5.1.1.1 Fuel-Air Ratio Model, we will forego a lengthy discussion here, but there are a few items of note. The originally double input data elements (e.g. **shift_speed_21**) for the Transmission model were explicitly recast to "single" via Stateflow Explorer, typical for an embedded target implementation. Several other elements (e.g. **gear** or **ctr**) had type-specific information available. The inputs at the top level of the model were then typed to correspond to the information in Stateflow Explorer, since they were directly connected.

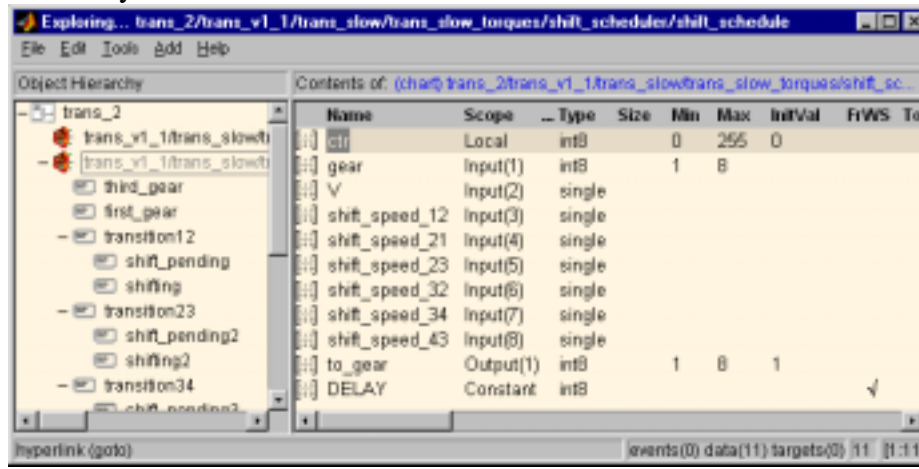


Figure 16: Stateflow Explorer Identifiers

Remove Propagation Symbols From Named Signals

As stated earlier, we have assumed each model was a component of a larger software structure, and would be suitably generated as a C code function. To accomplish this, we would define the model inputs as storage class `ImportedExtern`, and define the outputs as `ExportedGlobal`. When an attempt to generate code was made, the following error was given:

```
The name '<' specified for the signal from output port 1 of block 'trans_2/trans_v1_1' is not a valid identifier. Valid Identifiers start with an alphabetic character, followed by at most 31 alpha numeric or '_' characters...etc.
```

An analysis of the model revealed that the propagation symbols in the signal names were not understood by the code generator. Once the propagation symbols were removed, the error vanished.

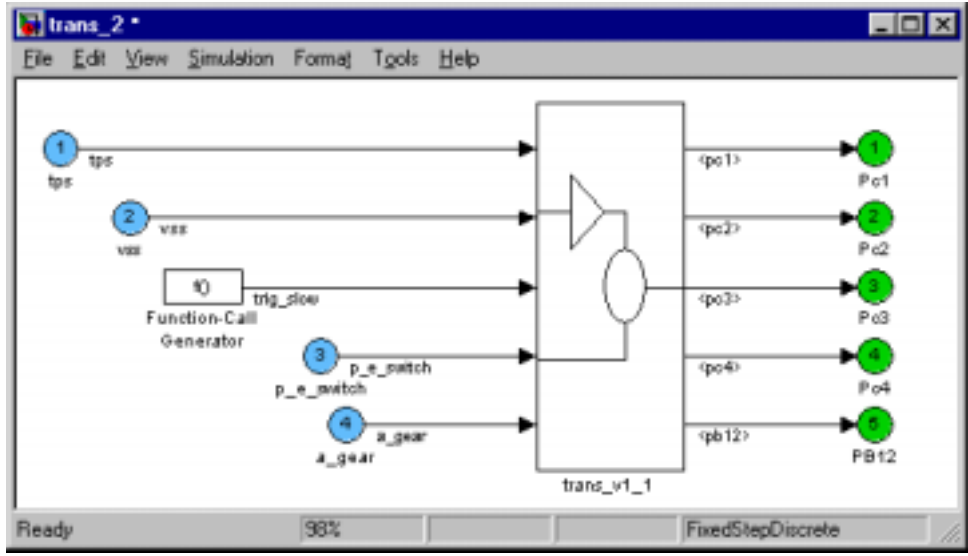


Figure 17: Diagram with Signal Propagation Characters on Output

Control of Stateflow Naming Conventions

As a default configuration, Real-Time Workshop Embedded Coder (RTW-EC) does not use names defined in Stateflow Explorer for the elements specified in the generated C code. This fact applies to both identifiers and statechart names. To change this, select from the stateflow chart 'Tools' menu, 'Open RTW Target'; then select the button 'Coder Options'. The user is then presented with a dialogue box (see Figure 18) and several check boxes.

The items checked below all contribute to improving the traceability and readability of the code. Data elements, however, are still likely to be defined in RTW-EC-named structures and not all symbol names are preserved. For example, in the transmission model, the "from Simulink"-specified variable `to_gear`, was rendered as `trans_2_B.s7_SFunction_o2`.

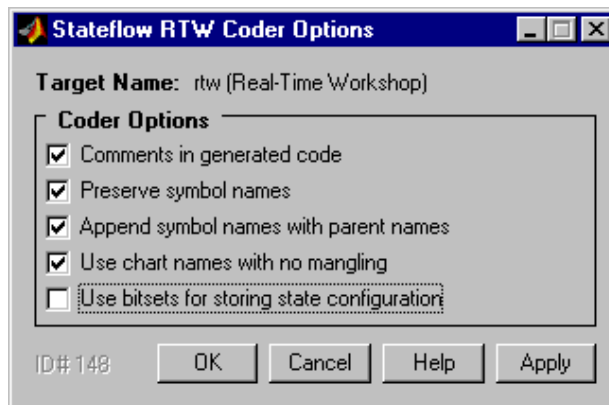


Figure 18: Name Settings for RTW Coder

5.2.1.5 Verification

The verification methods for the generated code were discussed in Section 4. Specific requirements associated with RTW-ET were described in Section 5.1.2.2.

5.2.1.6 Metrics

Section 3.2 of the Challenge Problem Document outlined a list of desirable attributes for a code generator. These criteria are specific to the code generation problem, in that they provide the necessary means to translate the behavioral specification of the model into a desired C code format. Below is listed a table of criteria that was named as being important in evaluating Automatic Code Generation tools.

Table 4: RTW Embedded Coder Metrics for Transmission Control

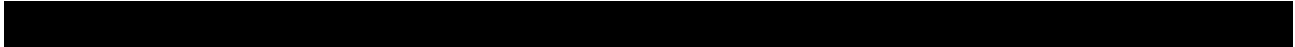
Item #	Challenge Problem Criteria	Assessment
	Transmission Model	
A	ANSI C (stated compliance)	YES
B	Preserved Functionality (Simulink)	YES
C	Preserved Functionality (Stateflow)	YES
D	ROM/RAM/CPU Efficiency	2456 / 48 /* bytes
E	Traceable / Readable	NO
F	System Independent Fcns.	NO
G	Automated Usage	NO
H	Discrete Time Support	N/A
I	Call to UserCode	YES
J	Call from User Code	NO
K	Variable Name Control	PARTIAL
L	Variable type Control	PARTIAL
M	Type Qualifiers	PARTIAL
N	Storage Class Control	PARTIAL
O	Variable Scoping Control	PARTIAL
P	Variable Initialization	YES
Q	Variable Declaration	YES
R	Function Partitioning	PARTIAL
S	Function Prototyping	NO
T	File Partitioning	PARTIAL
U	C Structures	NO
V	User Comments	YES
W	Processor Optimization	NO

Table Notes:

- A. MathWorks states ANSI C Compliance. Per all visual inspections to date, the code generated has complied with the ANSI/ISO 9899-1990 Standard, although no formal validation has been performed.
- B. Yes, conditionally. See explanation in discussion on RTW Embedded Coder validation.
- C. Functionality for Stateflow portion of model is preserved.
- D. See Appendix B for systems configuration details for these results. Though no execution (CPU) measurements were derived, code size is a broad indicator of execution performance.

- E. For the given model, the correlation between the representation in Simulink and the C code is difficult. There are several '.c' and '.h' files generated, and among them, it is not clear which is being called without close examination of all of the functions generated. In addition, there are many functions (and/or macros) interspersed in the files that have no obvious computational correlation to the model. These functions are generated to support the simulation framework for models intended for the MATLAB environment. To determine their effects on the generated code, the user must backtrack through source code located in MATLAB system directories (referenced by named '.h' files). Also, there are sections of C code in the headers that are conditionally compiled, making tracking more difficult. The inability to control the variable names assigned by RTW-EC makes the correlation of Stateflow diagrams to the resultant code difficult. The use of pointers to reference states does not enhance traceability.
- F. The generated code is dependent on MATLAB specific system calls and macros/include files, several of which chain into other include files.
- G. See comments in Section 5.1.2.3. There is no published API for accessing Stateflow components.
- H. There are no discrete time blocks in the Transmission model.
- I. Calls to external C code functions can be created from within Stateflow diagrams.
- J. No configurable calling interface to the functions is available, and calls depend on functions/macros generated in other locations.
- K. Signal names may serve as variable names that appear in the code. There is some control for naming variables; the user has control over the name that will appear in the code for each block output.
- L. At the top level of the model, users may select data types of the incoming variables. Intermediately, type conversion blocks (back to double) may need to be inserted around Simulink blocks that do not support specific data types. There is no way to alias MATLAB data types (e.g. Real_T) to a user specified type (e.g. float or 'custom_aliased_type').
- M. In the signal properties dialogue box, options for adding qualification strings to identifiers are allowed. These are text strings that will be added to the identifier declaration, but are not understood by the code generator as C keywords.
- N. User can choose between ImportedExtern, ExternGlobal, or ImportedExtern Pointer. This selection is incomplete with respect to available C storage classes.
- O. Variables declared in Stateflow are declared locally within each chart. To have global visibility (and representation in the C code) to all charts, variables must be specified at the stateflow machine level.

- P. Variables can be initialized using methods typical for simulation in Simulink and Stateflow (using fields in the Stateflow Explorer/ Constant blocks, and workspace variable definitions).
- Q. Variables can be declared using methods typical for simulation in Simulink and Stateflow (using fields in the Stateflow Explorer/ Constant blocks, and workspace variable definitions).
- R. Subsystems can be specified as atomic units (and subsequently, functions) via a properties submenu. Care must be taken not to affect the model execution behavior in selecting this option.
- S. There is no way to specify the function prototype for subsystems specified as functions.
- T. For subsystems labeled atomic and declared as functions, file names may be specified.
- U. C structures exist in the generated code, but are not configurable by the user; most automatically specified data elements are maintained in structures by default.
- V. User comments (denoted by text surrounded by /* */) in the Stateflow action language will appear beside the generated C code.
- W. No processor or compiler target-specific optimizations are available.



6 Summary

This report assesses two automatic code generation tools with respect to critical attributes identified in the [Challenge Problem Document](#). dSPACE TargetLink and The Mathworks RTW-EC were evaluated for two models developed by UC Berkeley for the DARPA MoBIES program representing examples of mixed Simulink-State Flow software specifications.

The results of this study are applicable only to the specific models evaluated using the particular tool versions identified in the Appendix B of this report.

7 Bibliography

- [1] Butts, K., et. al., "Automotive Powertrain Control Development Using CACSD," Perspectives in Control: New Concepts and Application, Tariq Samad (ed.), IEEE Press, 1999.
- [2] Toeppe, S., Ranville, S., Bostic, D., Rzeimen, K., "Automatic Code Generation Requirements For Production Automotive Powertrain Applications," IEEE International Symposium on Computer Aided Control System Design 1999
- [3] Toeppe, S., Ranville, S., Bostic, D., "Automating Software Specification, Design and Synthesis for Computer Aided Control System Design Tools," Proceedings of the 19th AIAA/IEEE/SAE Digital Avionics System Conference 2000
- [4] Toeppe, S., Ranville, S., Bostic, D., Wang, Y., "Practical Validation of Model Based Code Generation for Automotive Applications," Proceedings of the 18th AIAA/IEEE/SAE Digital Avionics System Conference 1999

8.1 APPENDIX A: Derivation of Discrete Integrator Block Transformation

We begin with the Discrete Integrator Equation:

$$\frac{y(k)}{e(k)} = \frac{T}{z-1}$$

if we cross multiply, we get:

$$z \cdot y(k) - y(k) = T \cdot e(k)$$

which is also

$$z \cdot y(k) = y(k) + T \cdot e(k)$$

So, now if we perform a transformation using z on the left side of the equation, we get:

$$y(k+1) = y(k) + T \cdot e(k)$$

$$y(k) = y(k-1) + T \cdot e(k-1)$$

$$y(k) = \frac{y(k) + T \cdot e(k)}{z}$$

or similarly,

$$y(k) = z^{-1} \cdot (y(k) + T \cdot e(k))$$

**SYSTEM CONFIGURATION FOR EVALUATION OF CODE GENERATORS AND
GENERATED CODE ROM AND RAM SIZE**

TargetLink Version: 1.2p1
MATLAB Version: 6.0
Stateflow Version: 4.0
Simulink Version: 4.0
LCC

Host Hardware Platform:
WindowsNT 4.0 Service Pack 6
Dell Precision 420 : 1GHz Pentium 4: 512 MB RAM

Compiler:
GreenHills C-PowerPC compiler, version 1.8.8

Compiler options:
-ANSI -C -G -OS -X442 -Z1019 -strict=nolinkerr -sda
(see compiler documentation for explanation of switches; switch selection is similar to settings used for production builds).

Host Hardware Platform :
Dec Alpha, 333MHz, 256 MB
Operating System Platform: DEC UNIX (OSF/1) V3.2C

8.3 APPENDIX C: Floating-point C Code Generated by TargetLink Fuel-Air Ratio Model

```

/*****\

Model code file      : fuel_air_ratio_v1_1.c
for TargetLink model : final_float_4/fuel_air_ratio_v1_1

Generated by TargetLink, the dSPACE production quality code generator
Wed Oct 24 15:10:28 2001

CODE GENERATOR OPTIONS:
Target                : Generic
ANSI-C compatible code : yes
Optimization level    : 5
Constant style        : decimal
Clean code option     : enabled
Logging mode          : Do not log anything
Linker sections       : enabled
Interrupt functions   : enabled
User attributes       : enabled
Assembler statements  : disabled
Variable name length  : 31 chars
Separate lookup search function : disabled
Use global bitfields  : disabled
Stateflow: use of bitfields : enabled
State activity encoding limit : 5
Omit zero inits in restart function: disabled
Share fcns between TL subsystems : disabled
Generate 64bit functions : enabled
Inlining Threshold    : 6
Line break limit      : 100
Constant suffix       : disabled
Target optimized boolean data type : enabled

Subsys  Corresponding Simulink Subsystem
Sa1     fuel_air_ratio_v1_1
Sa2     fuel_air_ratio_v1_1/afr_fast
Sa3     fuel_air_ratio_v1_1/afr_fast/afr_fast_observer
Sa4     fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer
Sa5     fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
Sa6     fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow/nvol
Sa7     fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_fuelflow
Sa8     fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_fuelflow/Subsystem
Sa9     fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/fuel_transient_comp
Sa10    fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/o2_sensor_corr

SF-Node Corresponding Stateflow Node                description
Ca0     fuel_air_ratio_v1_1/Stateflow machine [final_float_4]
Ca1     fuel_air_ratio_v1_1/afr_fast/afr_fast_ctl

TargetLink version   : 1.2p1 from 19-Jul-2001
Codegenerator version : 1.2.P1   from Jul 19 2001, 17:45:00
Copyright (c) 1999-2001 by dSPACE GmbH
/*****\

#define _FUEL_AIR_RATIO_V1_1_C_          /* identifier for this file */

/*****\
includes
/*****\

#include <delta_time.c>                /*
User included file from block: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_f
uelflow/Subsystem/Addfile */
#include "fuel_air_ratio_v1_1.h"        /* Include of own header file */
#include "fuel_air_ratio_v1_1.udt.h"    /* type definition of the user data types */
#include "tl_types.h"                  /* definition of base data types */

/*****\
STATIC_MACRO: module-local preprocessor macros
/*****\
#define dmfc_o2                0.F

```

```

#define p_gain_fuel_trans      ((F32)3.F)
#define dMao_to_dMfc          ((F32)0.06802721088F)
#define maxfr                 0.3335F
#define inj_size              14.64F
#define nvol_d9               0.372363F
#define nvol_d8               0.00038888F
#define nvol_d7               -1.0387e-006F
#define nvol_d6               64.99F
#define nvol_d5               0.665964F
#define nvol_d3               13102.67146F
#define nvol_d3_a             13102.67146F
#define nvol_d2               -214.66574F
#define nvol_d1               0.0828512F
#define C1                     0.112F
#define inv_Ma2Pm              ((F32)3.104840101e-005F)

/*****\
  defaultClass_slStaticGlobalInit: Default storage class for global static variables with initial
  value
*****/
static F32      X_Sa8_Unit_Delay = 0.0003492637697F; /* 32 bit floating-point variable */
static F32      Sa3_dMfc_ = 0.F; /* 32 bit floating-point variable */

/*****\
  defaultClass_slStaticGlobal: Default storage class for global static variables
*****/
static F32      AUX_a_o2s;
static F32      AUX_a_we;
static F32      AUX_a_map;
static F32      AUX_a_bp;
static F32      AUX_a_tps;

/*****\
  Stateflow functions
*****/

/*****\
  Model step function of Stateflow chart: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_ctl
*****/
void Cal_af_r_fast_ctl(void)
{
  /* LOCAL: local variable */
  F32 delta_time; /* 32 bit floating-point variable */
  /* defaultClass_slLocal: Default storage class for local variables */
  F32 Sa3_dMao_to_dMfc; /* 32 bit floating-point variable */
  F32 Sa6_Product; /* 32 bit floating-point variable */
  F32 Sa4_Gain; /* 32 bit floating-point variable */
  F32 Sa3_Sum1; /* 32 bit floating-point variable */

  /* Product: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_airflow/nvol/Product
  */
  Sa6_Product = AUX_a_we * AUX_a_we;

  /* Gain: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/Gain */
  Sa4_Gain = AUX_a_map * inv_Ma2Pm;

  /* Gain: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/dMao_to_dMfc

  # combined # Product: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_airflow
  /dMao_calc3

  # combined # Product: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_airflow
  /dMao_calc2

  # combined # Product: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_airflow
  /dMao_calc1

  # combined # Sum: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_airflow/
  nvol/Sum3

  # combined # Product: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_airflow
  /nvol/Product4

  # combined # Product: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_airflow
  /nvol/Product1

```

```

# combined # Sum: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow/
nvol/Sum

# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/nvol/Product2

# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/nvol/Product3

# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/nvol/Product7

# combined # Sum: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow/
nvol/Sum1

# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/nvol/Product5

# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/nvol/Product6

# combined # Sum: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow/
nvol/Sum2

# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/nvol/Product8

# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/nvol/Product9 */
Sa3_dMao_to_dMfc = Sa4_Gain * ((Sa4_Gain * Sa4_Gain * ((nvol_d1 * Sa6_Product) + (nvol_d2 *
AUX_a_we)
+ nvol_d3_a)) + (Sa4_Gain * ((nvol_d3 * Sa6_Product) + (nvol_d5 * AUX_a_we) + nvol_d6)) +
(nvol_d7
* Sa6_Product) + (nvol_d8 * AUX_a_we) + nvol_d9) * C1 * AUX_a_we * dMao_to_dMfc;

/*
Custom code block: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_fuelflow/
Subsystem/Custom Code Block << output code >> */
{
delta_time = DeltaTimeSec();
}

/* Sum: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/Sum1
# combined # Sum: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/Sum2

# combined # Gain: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/fuel_transient_comp/p_gain_f
uel_trans
# combined # Sum: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/fuel_transient_comp/Sum3 */
Sa3_Sum1 = dMfc_o2 + Sa3_dMao_to_dMfc + ((Sa3_dMao_to_dMfc - X_Sa8_Unit_Delay) *
p_gain_fuel_trans);

/* Outport: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/dMfc */
Sa3_dMfc_ = Sa3_Sum1;

/* ---- Update code of subsystem: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer ---- */

/*
Unit delay: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_fuelflow/Subsystem/Unit Delay

# combined # Sum: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_fuelflow/
Subsystem/Sum

# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_fuelflo
w/Subsystem/Product

# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_fuelflo
w/Product

# combined # Sum: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_fuelflow/
Sum

# combined # Sum: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_fuelflow/
Sum2

# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_fuelflo
w/Product3

# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_fuelflo

```

```

w/Product4

# combined # Product: fuel_air_ratio_v1_1/afrc_fast/afrc_fast_observer/a_f_observer/port_fuelflo
w/Product2

# combined # Product: fuel_air_ratio_v1_1/afrc_fast/afrc_fast_observer/a_f_observer/port_fuelflo
w/Product1 */
X_Sa8_Unit_Delay = X_Sa8_Unit_Delay + (delta_time * ((Sa3_Sum1 - X_Sa8_Unit_Delay) /
(((Sa3_Sum1
* inj_size) / maxfr) * 4.71238898F) / AUX_a_we) + 0.05F));
}

/*****\
Model step function of subsystem(s): fuel_air_ratio_v1_1
\*****/
F32 fuel_air_ratio_v1_1(
F32 we /* 32 bit floating-point variable */,
F32 map /* 32 bit floating-point variable */,
F32 tps /* 32 bit floating-point variable */,
F32 bp /* 32 bit floating-point variable */,
F32 o2s /* 32 bit floating-point variable */)
{
/* FCN_RETURN: function return value (STACK) */
F32 dMfc; /* 32 bit floating-point variable */

AUX_a_we = we;
AUX_a_map = map;
AUX_a_tps = tps;
AUX_a_bp = bp;
AUX_a_o2s = o2s;

/* TargetLink outpost used for signal rescaling: fuel_air_ratio_v1_1/dmfc_ */
dMfc = Sa3_dMfc_;

/* ---- Update code of subsystem: fuel_air_ratio_v1_1 ---- */
return dMfc;
}

#undef _FUEL_AIR_RATIO_V1_1_C_

```

8.4 APPENDIX D: Fixed-Point C Code generated by TargetLink for Fuel-Air Ratio Model

```

/*****\

Model code file      : fuel_air_ratio_v1_1.c
for TargetLink model : fixed/fuel_air_ratio_v1_1

Generated by TargetLink, the dSPACE production quality code generator
Wed Oct 31 09:26:25 2001

CODE GENERATOR OPTIONS:
Target                : Generic
ANSI-C compatible code : yes
Optimization level    : 5
Constant style        : decimal
Clean code option     : enabled
Logging mode          : Do not log anything
Linker sections       : enabled
Interrupt functions   : enabled
User attributes       : enabled
Assembler statements  : disabled
Variable name length  : 31 chars
Separate lookup search function : disabled
Use global bitfields  : disabled
Stateflow: use of bitfields : enabled
State activity encoding limit : 5
Omit zero inits in restart function: disabled
Share fcns between TL subsystems : disabled
Generate 64bit functions : enabled
Inlining Threshold    : 6
Line break limit      : 100
Constant suffix       : disabled
Target optimized boolean data type : enabled

Subsys  Corresponding Simulink Subsystem
Sa1     fuel_air_ratio_v1_1
Sa2     fuel_air_ratio_v1_1/afr_fast
Sa3     fuel_air_ratio_v1_1/afr_fast/afr_fast_observer
Sa4     fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer
Sa5     fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
Sa6     fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow/nvol
Sa7     fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_fuelflow
Sa8     fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_fuelflow/Subsystem
Sa9     fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/fuel_transient_comp
Sa10    fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/o2_sensor_corr

SF-Node  Corresponding Stateflow Node          description
Ca0     fuel_air_ratio_v1_1/Stateflow machine [fixed]
Ca1     fuel_air_ratio_v1_1/afr_fast/afr_fast_ctl

TargetLink version   : 1.2p1 from 19-Jul-2001
Codegenerator version : 1.2.P1   from Jul 19 2001, 17:45:00
Copyright (c) 1999-2001 by dSPACE GmbH
/*****\

#define _FUEL_AIR_RATIO_V1_1_C_          /* identifier for this file */

/*****\
  includes
/*****\

#include <delta_time.c>                /*
  User included file from block: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_f
  uelflow/Subsystem/Addfile */
#include "fuel_air_ratio_v1_1.h"        /* Include of own header file */
#include "fuel_air_ratio_v1_1.udt.h"    /* type definition of the user data types */
#include "tl_types.h" /* definition of base data types */

/*****\
  STATIC_MACRO: module-local preprocessor macros
/*****\
#define dmfc_o2          0

/* LSB: 2^-13 OFF:  0 MIN/MAX: -4 .. 3.99987793 */
#define p_gain_fuel_trans  ((S16)24576 /* 3. */)

```

```

/* LSB: 2^-18 OFF: 0 MIN/MAX: -0.125 .. 0.1249961853 */
#define dMao_to_dMfc ((S16)17833 /* 0.06802721088 */)

/* LSB: 2^-16 OFF: 0 MIN/MAX: -0.5 .. 0.4999847412 */
#define maxfr 21856 /* 0.3335 */

/* LSB: 2^-11 OFF: 0 MIN/MAX: -16 .. 15.99951172 */
#define inj_size 29983 /* 14.64 */

/* LSB: 2^-16 OFF: 0 MIN/MAX: -0.5 .. 0.4999847412 */
#define nvol_d9 24403 /* 0.372363 */

/* LSB: 2^-26 OFF: 0 MIN/MAX: -0.00048828125 .. 0.0004882663488 */
#define nvol_d8 26097 /* 0.00038888 */

/* LSB: 2^-34 OFF: 0 MIN/MAX: -1.907348633e-006 .. 1.907290425e-006 */
#define nvol_d7 -17845 /* -1.0387e-006 */

/* LSB: 2^-8 OFF: 0 MIN/MAX: -128 .. 127.9960938 */
#define nvol_d6 16637 /* 64.99 */

/* LSB: 2^-15 OFF: 0 MIN/MAX: -1 .. 0.9999694824 */
#define nvol_d5 21822 /* 0.665964 */

/* LSB: 2^-29 OFF: 0 MIN/MAX: -6.103515625e-005 .. 6.10332936e-005 */
#define nvol_d4 -21968 /* -4.09185e-005 */

/* LSB: 2^-1 OFF: 0 MIN/MAX: -16384 .. 16383.5 */
#define nvol_d3 26205 /* 13102.67146 */

/* LSB: 2^-7 OFF: 0 MIN/MAX: -256 .. 255.9921875 */
#define nvol_d2 -27477 /* -214.66574 */

/* LSB: 2^-18 OFF: 0 MIN/MAX: -0.125 .. 0.1249961853 */
#define nvol_d1 21719 /* 0.0828512 */

/* LSB: 2^-18 OFF: 0 MIN/MAX: -0.125 .. 0.1249961853 */
#define C1 29360 /* 0.112 */

/* LSB: 2^-29 OFF: 0 MIN/MAX: -6.103515625e-005 .. 6.10332936e-005 */
#define inv_Ma2Pm ((S16)16669 /* 3.104840101e-005 */)

/*****\
 defaultClass_slStaticGlobalInit: Default storage class for global static variables with initial
 value
 *****/
static S16 X_Sa8_Unit_Delay = 366 /* 0.0003492637697 */; /* LSB: 2^-20 OFF: 0 MIN/MAX: -
0.03125 .. 0.03124904633 */
static S16 Sa3_dmfc_ = 0; /* LSB: 2^-20 OFF: 0 MIN/MAX: -0.03125 .. 0.03124904633 */

/*****\
 defaultClass_slStaticGlobal: Default storage class for global static variables
 *****/
static S16 AUX_a_we;
static S16 AUX_a_map;
static S16 AUX_a_bp;
static S16 AUX_a_tps;
static S8 AUX_a_o2s;

/*****\
 Stateflow functions
 *****/

/*****\
 Model step function of Stateflow chart: fuel_air_ratio_vl_1/afr_fast/afr_fast_ctl
 *****/
void Cal_afr_fast_ctl(void)
{
 /* LOCAL: local variable */
 S16 delta_time; /* LSB: 2^-21 OFF: 0 MIN/MAX: -0.015625 .. 0.01562452316 */
 /* defaultClass_slLocal: Default storage class for local variables */
 S16 Sa3_dMao_to_dMfc; /* LSB: 2^-20 OFF: 0 MIN/MAX: -0.03125 .. 0.03124904633 */
 S16 Sa7_Product4; /* LSB: 2^-13 OFF: 0 MIN/MAX: -4 .. 3.99987793 */
 S16 Sa7_Product3; /* LSB: 2^-22 OFF: 0 MIN/MAX: -0.0078125 .. 0.007812261581 */
 S16 Sa7_Product2; /* LSB: 2^-15 OFF: 0 MIN/MAX: -1 .. 0.9999694824 */
 S16 Sa7_Product1; /* LSB: 2^-16 OFF: 0 MIN/MAX: -0.5 .. 0.4999847412 */
 S16 Sa6_Product; /* LSB: 2^5 OFF: 0 MIN/MAX: -1048576 .. 1048544 */
 S16 Sa4_Gain; /* LSB: 2^-22 OFF: 0 MIN/MAX: -0.0078125 .. 0.007812261581 */
 S16 Sa3_Sum1; /* LSB: 2^-20 OFF: 0 MIN/MAX: -0.03125 .. 0.03124904633 */

```

```

S16 AUX_a_S16_d_a;
S16 AUX_a_S16_e_a;
S16 AUX_a_S16_f_a;
S16 AUX_a_S16_g_a;

/* Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow/nvol/Product
*/
Sa6_Product = (S16)((((S32)AUX_a_we) * ((S32)AUX_a_we)) >> 13);

/* Gain: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/Gain */
Sa4_Gain = (S16)((((S32)AUX_a_map) * ((S32)inv_Ma2Pm)) >> 14);

/* Gain: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/dMao_to_dMfc

# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/nvol/Product2 */
AUX_a_S16_d_a = (S16)((((S16)((((S32)nvol_d1) * ((S32)Sa6_Product)) >> 14)) >> 2);

/*
# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/nvol/Product3 */
AUX_a_S16_d_a += (S16)((((S32)nvol_d2) * ((S32)AUX_a_we)) >> 14);

/*
# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/nvol/Product5 */
AUX_a_S16_e_a = (S16)((((S16)((((S32)nvol_d4) * ((S32)Sa6_Product)) >> 14)) >> 5);

/*
# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/nvol/Product6 */
AUX_a_S16_e_a += (S16)((((S32)nvol_d5) * ((S32)AUX_a_we)) >> 14);

/*
# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/nvol/Product8 */
AUX_a_S16_f_a = (S16)((((S32)nvol_d7) * ((S32)Sa6_Product)) >> 14);

/*
# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/nvol/Product9 */
AUX_a_S16_f_a += (S16)((((S16)((((S32)nvol_d8) * ((S32)AUX_a_we)) >> 14)) >> 1);

/*
# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/nvol/Product4

# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/nvol/Product1

# combined # Sum: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow/
nvol/Sum */
AUX_a_S16_g_a = (S16)((((S32)(S16)((((S32)Sa4_Gain) * ((S32)Sa4_Gain)) >> 14)) *
((S32)(S16)((((U16)AUX_a_S16_d_a)
+ ((U16)(S16)(nvol_d3 >> 4)))))) >> 13);

/*
# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/nvol/Product7

# combined # Sum: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow/
nvol/Sum1 */
AUX_a_S16_g_a += (S16)((((S16)((((S32)Sa4_Gain) * ((S32)(S16)((((U16)AUX_a_S16_e_a) +
((U16)(S16)(nvol_d6 >> 3)))))) >> 14)) << 1);

/*
# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/dMao_calc3

# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/dMao_calc2

# combined # Product: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow
/dMao_calc1

# combined # Sum: fuel_air_ratio_v1_1/afr_fast/afr_fast_observer/a_f_observer/port_airflow/
nvol/Sum3

```

```

    # combined # Sum: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_airflow/
    nvol/Sum2 */
    Sa3_dMao_to_dMfc = (S16)((((S32)(S16)((((S32)(S16)((((S32)(S16)((((S32)Sa4_Gain) *
((S32)(S16)((U16)AUX_a_S16_g_a)
+ ((U16)(S16)((S16)((U16)AUX_a_S16_f_a) + ((U16)(S16)(nvol_d9 >> 1)))) >> 1)))))) >> 14)) *
((S32)c1)) >> 15))
    * ((S32)AUX_a_we) >> 13)) * ((S32)dMao_to_dMfc) >> 14);

/*
Custom code block: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_fuelflow/
Subsystem/Custom Code Block << output code >> */
{

    delta_time = S16_DeltaTimeSec();

}

/* Sum: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/Sum1
# combined # Sum: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/Sum2

# combined # Gain: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/fuel_transient_comp/p_gain_f
uel_trans
# combined # Sum: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/fuel_transient_comp/Sum3 */
Sa3_Sum1 = (S16)((U16)(S16)(dMfc_o2 << 20)) + ((U16)(S16)((U16)Sa3_dMao_to_dMfc) +
((U16)(S16)((
(S16)((((S32)(S16)((S16)((S32)Sa3_dMao_to_dMfc) - ((S32)X_Sa8_Unit_Delay))) << 6)) *
((S32)p_gain_fuel_trans)) >> 15)) >> 4)))));

/* Outport: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/dMfc */
Sa3_dMfc_ = Sa3_Sum1;

/* Product: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_fuelflow/Product1 */
Sa7_Product1 = (S16)((((S32)Sa3_Sum1) * ((S32)inj_size) >> 15);

/* Product: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_fuelflow/Product2 */
if ( maxfr != 0 ) {
    Sa7_Product2 = (S16)((S32)((S32)Sa7_Product1) << 15)) / maxfr;
} else {
    if ( Sa7_Product1 < 0 ) {
        Sa7_Product2 = -32768;
    } else {
        Sa7_Product2 = 32767;
    }
}

/* Product: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_fuelflow/Product4 */
Sa7_Product4 = (S16)((((S32)Sa7_Product2) * 19301) >> 14);

/* Product: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_fuelflow/Product3 */
if ( AUX_a_we != 0 ) {
    Sa7_Product3 = (S16)((S32)((S32)Sa7_Product4) << 13)) / AUX_a_we;
} else {
    if ( Sa7_Product4 < 0 ) {
        Sa7_Product3 = -32768;
    } else {
        Sa7_Product3 = 32767;
    }
}

/* ---- Update code of subsystem: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer ---- */

/*
Unit delay: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_fuelflow/Subsyste
m/Unit Delay

# combined # Sum: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_fuelflow/
Subsystem/Sum

# combined # Product: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_fuelflo
w/Subsystem/Product

# combined # Product: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_fuelflo
w/Product
fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_fuelflow/Product: Range of
denominator does not contain zero. Omitted division by zero check.

# combined # Sum: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_fuelflow/
Sum

# combined # Sum: fuel_air_ratio_v1_1/af_r_fast/af_r_fast_observer/a_f_observer/port_fuelflow/
Sum2 */

```

```

X_Sa8_Unit_Delay = (S16)((U16)X_Sa8_Unit_Delay) + ((U16)(S16)((S16)((S32)delta_time) *
((S32)(S16)((
(S32)((S32)(S16)((S16)((S32)Sa3_Sum1) - ((S32)X_Sa8_Unit_Delay))) << 4)) << 13)) /
((S16)((S16)(
Sa7_Product3 >> 4)) + 13107)))) >> 14)) >> 6));
}

```

```

/*****\
Model step function of subsystem(s): fuel_air_ratio_v1_1
\*****/

```

```

S16 calc_fuel_air_ratio(
S16 we /* LSB: 2^-4 OFF: 0 MIN/MAX: -2048 .. 2047.9375 */,
S16 map /* LSB: 2^-7 OFF: 0 MIN/MAX: -256 .. 255.9921875 */,
S16 tps /* LSB: 2^-13 OFF: 0 MIN/MAX: -4 .. 3.99987793 */,
S16 bp /* LSB: 2^-7 OFF: 0 MIN/MAX: -256 .. 255.9921875 */,
S8 o2s /* LSB: 2^-4 OFF: 0 MIN/MAX: -8 .. 7.9375 */)
{
/* FCN_RETURN: function return value (STACK) */
S16 dMfc; /* LSB: 2^-20 OFF: 0 MIN/MAX: -0.03125 .. 0.03124904633 */

```

```

AUX_a_we = we;
AUX_a_map = map;
AUX_a_tps = tps;
AUX_a_bp = bp;
AUX_a_o2s = o2s;

```

```

/* TargetLink output used for signal rescaling: fuel_air_ratio_v1_1/dmfc_ */
dMfc = Sa3_dMfc_;

```

```

/* ---- Update code of subsystem: fuel_air_ratio_v1_1 ---- */
return dMfc;
}

```

```

#undef _FUEL_AIR_RATIO_V1_1_C_

```

8.5 APPENDIX E: Floating-Point C Code Generated by RTW-EC for Fuel-Air Ratio Model

```

/*
 * fuel_air_ratio_ert.c
 *
 * Real-Time Workshop code generation for Simulink model "fuel_air_ratio_ert.mdl".
 *
 * Model Version                : 1.23
 * Real-Time Workshop file version : 4.0 $Date: 2000/09/19 19:45:27 $
 * Real-Time Workshop file generated on : Mon Oct 29 14:37:02 2001
 * TLC version                  : 4.0 (Aug 21 2000)
 * C source code generated on    : Mon Oct 29 14:37:02 2001
 *
 * Relevant TLC Options:
 *   InlineParameters           = 1
 *   RollThreshold              = 5
 *   CodeFormat                 = Embedded-C
 *
 * Simulink model settings:
 *   Solver                     : FixedStep
 *   StartTime                  : 0.0 s
 *   StopTime                   : 1.0 s
 *   FixedStep                  : 0.01 s
 */

#include "fuel_air_ratio_ert.h"
#include "fuel_air_ratio_ert_prm.h"

/* Start of Functions in model "fuel_air_ratio_ert" */

/* model step function */
void fuel_air_ratio_ert_step(int_T tid)
{
    /* update absolute time */
    if (ssIsSampleHit(fuel_air_ratio_ert_rt0, 0, tid)) {
        ssUpdateRealAbsoluteTime(fuel_air_ratio_ert_rt0);

        if (ssIsSpecialSampleHit(fuel_air_ratio_ert_rt0, 1, 0, tid)) {
            ssUpdateSubrateTaskTime(fuel_air_ratio_ert_rt0, 1);
        }
    }

    if (ssIsSampleHit(fuel_air_ratio_ert_rt0, 1, tid)) { /* Sample time: [1.0, 0.0] */
        /* S-Function (fcncallgen) Block: <Root>/Function-Call Generator */

        /* Output and update for function-call system: <S2>/afr_fast_ctl */

        /* Stateflow Chart Code: <S2>/afr_fast_ctl */
        {
            SFafr_fast_ctlInstanceStruct *chartInstance = (SFafr_fast_ctlInstanceStruct
*)fuel_air_ratio_ert_DWork.s3_SFunction_PWORK.ChartInstance;

#define event_time_fast          0

            {
                int previousEvent;
                previousEvent = _sfEvent_fuel_air_ratio_ert_;
                /* Broadcast of input event(s) */

                _sfEvent_fuel_air_ratio_ert_ = event_time_fast;
                {
                    if(_sfEvent_fuel_air_ratio_ert_ == event_time_fast) {
                        {
                            int_T tid = 1;

                            /* Output and update for function-call system: <S2>/afr_fast_observer */
                            /* Gain Block: <S5>/Gain */
                            fuel_air_ratio_ert_B.temp6 = map * ((real32_T)( inv_Ma2Pm ));

                            /* Product Block: <S10>/Product1 */
                            fuel_air_ratio_ert_B.temp4 = fuel_air_ratio_ert_B.temp6 *
                                fuel_air_ratio_ert_B.temp6;

                            /* Product Block: <S10>/Product */
                            fuel_air_ratio_ert_B.temp5 = we *

```

```

we;

/* Product Block: <S10>/Product2 */
fuel_air_ratio_ert_B.temp8 = ((real32_T)( nvol_d1 )) *
    fuel_air_ratio_ert_B.temp5;

/* Product Block: <S10>/Product3 */
fuel_air_ratio_ert_B.temp7 = ((real32_T)( nvol_d2 )) *
    we;

/* Sum Block: <S10>/Sum */
fuel_air_ratio_ert_B.temp8 = fuel_air_ratio_ert_B.temp8
    + fuel_air_ratio_ert_B.temp7 + ((real32_T)( nvol_d3 ));

/* Product Block: <S10>/Product4 */
fuel_air_ratio_ert_B.temp4 = fuel_air_ratio_ert_B.temp4 *
    fuel_air_ratio_ert_B.temp8;

/* Product Block: <S10>/Product5 */
fuel_air_ratio_ert_B.temp7 = ((real32_T)( nvol_d4 )) *
    fuel_air_ratio_ert_B.temp5;

/* Product Block: <S10>/Product6 */
fuel_air_ratio_ert_B.temp8 = ((real32_T)( nvol_d5 )) *
    we;

/* Sum Block: <S10>/Sum1 */
fuel_air_ratio_ert_B.temp7 = fuel_air_ratio_ert_B.temp7
    + fuel_air_ratio_ert_B.temp8 + ((real32_T)( nvol_d6 ));

/* Product Block: <S10>/Product7 */
fuel_air_ratio_ert_B.temp7 = fuel_air_ratio_ert_B.temp6 *
    fuel_air_ratio_ert_B.temp7;

/* Product Block: <S10>/Product8 */
fuel_air_ratio_ert_B.temp5 = ((real32_T)( nvol_d7 )) *
    fuel_air_ratio_ert_B.temp5;

/* Product Block: <S10>/Product9 */
fuel_air_ratio_ert_B.temp8 = ((real32_T)( nvol_d8 )) *
    we;

/* Sum Block: <S10>/Sum2 */
fuel_air_ratio_ert_B.temp5 = fuel_air_ratio_ert_B.temp5
    + fuel_air_ratio_ert_B.temp8 + ((real32_T)( nvol_d9 ));

/* Sum Block: <S10>/Sum3 */
fuel_air_ratio_ert_B.temp4 = fuel_air_ratio_ert_B.temp4
    + fuel_air_ratio_ert_B.temp7 + fuel_air_ratio_ert_B.temp5;

/* Product Block: <S8>/dMao_calc */
fuel_air_ratio_ert_B.temp6 = fuel_air_ratio_ert_B.temp6 *
    fuel_air_ratio_ert_B.temp4 *
    ((real32_T)( C1 )) *
    we;

/* Gain Block: <S4>/dMao_to_dMfc */
fuel_air_ratio_ert_B.temp6 *= ((real32_T)( dMao_to_dMfc ));

/* DiscreteIntegrator Block: <S9>/Discrete-Time Integrator1 */
fuel_air_ratio_ert_B.s9_dMfi_b =
fuel_air_ratio_ert_DWork.s9_Discrete_Time_Integra_DSTATE;

/* DataTypeConversion Block: <S9>/Data Type Conversion1 */
fuel_air_ratio_ert_B.temp7 = (real32_T)fuel_air_ratio_ert_B.s9_dMfi_b;

/* Sum Block: <S6>/Sum3 */
fuel_air_ratio_ert_B.temp8 = fuel_air_ratio_ert_B.temp6
    - fuel_air_ratio_ert_B.temp7;

/* Gain Block: <S6>/p_gain_fuel_trans */
fuel_air_ratio_ert_B.temp8 *= ((real32_T)( p_gain_fuel_trans ));

/* Sum Block: <S4>/Sum2 */
fuel_air_ratio_ert_B.temp6 = fuel_air_ratio_ert_B.temp6
    + fuel_air_ratio_ert_B.temp8;

/* Sum Block: <S4>/Sum1 */
fuel_air_ratio_ert_B.s4_dMfc = ((real32_T)( dMfc_o2 ))
    + fuel_air_ratio_ert_B.temp6;

```

```

/* Sum Block: <S9>/Sum */
fuel_air_ratio_ert_B.temp7 = - fuel_air_ratio_ert_B.temp7
+ fuel_air_ratio_ert_B.s4_dMfc;

/* Product Block: <S9>/tau_f */
fuel_air_ratio_ert_B.temp8 = fuel_air_ratio_ert_B.s4_dMfc *
((real32_T)( inj_size )) /
((real32_T)( maxfr )) *
(4.712389F) /
we;

/* Sum Block: <S9>/Sum2 */
fuel_air_ratio_ert_B.temp8 = fuel_air_ratio_ert_B.temp8 + (0.050000001F);

/* Product Block: <S9>/Product */
fuel_air_ratio_ert_B.temp7 = fuel_air_ratio_ert_B.temp7 /
fuel_air_ratio_ert_B.temp8;

/* DataTypeConversion Block: <S9>/Data Type Conversion */
fuel_air_ratio_ert_B.s9_delayed_ddMfi_b = (real_T)fuel_air_ratio_ert_B.temp7;

/* DiscreteIntegrator Block: <S9>/Discrete-Time Integrator1 */
{
real_T currentT = ssGetTaskTime(fuel_air_ratio_ert_rt0,tid);
real_T dT = currentT -
fuel_air_ratio_ert_DWork.s9_Discrete_Time_Integra_RWORK.PrevT;
fuel_air_ratio_ert_DWork.s9_Discrete_Time_Integra_RWORK.PrevT = currentT;
fuel_air_ratio_ert_DWork.s9_Discrete_Time_Integra_DSTATE =
fuel_air_ratio_ert_DWork.s9_Discrete_Time_Integra_DSTATE + dT *
fuel_air_ratio_ert_B.s9_delayed_ddMfi_b;
}
}
}
}
_sfEvent_fuel_air_ratio_ert_ = previousEvent;
}

#undef event_time_fast
}
}

if (ssIsSampleHit(fuel_air_ratio_ert_rt0, 1, tid)) { /* Sample time: [1.0, 0.0] */
/* Outport Block: <Root>/dmfc */
fuel_air_ratio_ert_Y.root_dmfc = fuel_air_ratio_ert_B.s4_dMfc;
}

/* (no update code required) */
}

/* End of Functions in model "fuel_air_ratio_ert" */

#include "fuel_air_ratio_ert_reg.h"

/* [EOF] fuel_air_ratio_ert.c */

#define IN_SF_MACHINE_SOURCE 1
#include "fuel_air_ratio_ert_rtw.h"

#include "afr_fast_ctl.h"

int8_T _sfEvent_fuel_air_ratio_ert_=0;
void fuel_air_ratio_ert_initializer( void )
{
/* Initialize machine's broadcast event variable */
_sfEvent_fuel_air_ratio_ert_ = CALL_EVENT;
}

void fuel_air_ratio_ert_terminator(void)
{
}
extern void fuel_air_ratio_ert_terminator(void);
void fuel_air_ratio_ert_machine_global_terminator(void)
{
fuel_air_ratio_ert_terminator();
return;
}

```

```
extern void fuel_air_ratio_ert_initializer(void);
void fuel_air_ratio_ert_machine_global_initializer(void)
{
    fuel_air_ratio_ert_initializer();
    return;
}
```

8.6 APPENDIX F: Floating-Point C Code Generated by TargetLink for Transmission Model

```
/******\
Model code file      : sepfile.c
for TargetLink model : tl_transfixed/trans_v1_1

Generated by TargetLink, the dSPACE production quality code generator
Tue Feb 26 10:39:21 2002

CODE GENERATOR OPTIONS:
Target                : Generic
ANSI-C compatible code : yes
Optimization level    : 5
Constant style        : decimal
Clean code option     : enabled
Logging mode          : Do not log anything
Linker sections       : enabled
Interrupt functions   : enabled
User attributes       : enabled
Assembler statements  : disabled
Variable name length  : 31 chars
Separate lookup search function : disabled
Use global bitfields  : disabled
Stateflow: use of bitfields : enabled
State activity encoding limit : 5
Omit zero inits in restart function: disabled
Share fcns between TL subsystems : disabled
Generate 64bit functions : enabled
Inlining Threshold    : 6
Line break limit      : 100
Constant suffix       : disabled
Target optimized boolean data type : enabled

Subsys  Corresponding Simulink Subsystem
Sa1     trans_v1_1
Sa2     trans_v1_1/trans_slow
Sa3     trans_v1_1/trans_slow/trans_slow_torques
Sa4     trans_v1_1/trans_slow/trans_slow_torques/clutch_pressure
Sa5     trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler

SF-Node  Corresponding Stateflow Node
          description
Ca0      trans_v1_1/Stateflow machine [tl_transfixed]

Ca1      trans_v1_1/trans_slow/trans_slow_ctl

Ca2      trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule

Ca3
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.first_gear

Ca4
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.fourth_gear

Ca5
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.second_gear

Ca6
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.third_gear

Ca7
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition12

Ca8
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition12.shi
```

```

    ft_pending
Ca9
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition12.shi
fting
Ca10
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition21

Ca11
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition21.shi
ft_pending
Ca12
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition21.shi
fting
Ca13
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition23

Ca14
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition23.shi
ft_pending2
Ca15
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition23.shi
fting2
Ca16
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition32

Ca17
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition32.shi
ft_pending
Ca18
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition32.shi
fting
Ca19
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition34

Ca20
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition34.shi
ft_pending3
Ca21
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition34.shi
fting3
Ca22
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition43

Ca23
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition43.shi
ft_pending
Ca24
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition43.shi
fting

```

```

TargetLink version      : 1.2p1 from 19-Jul-2001
Codegenerator version  : 1.2.P1 from Jul 19 2001, 17:45:00
Copyright (c) 1999-2001 by dSPACE GmbH

```

```

\*****/

```

```

#define _SEPFILE_C_ /* identifier for this file */

```

```

/*****\
includes
\*****/

```

```

#include <ppc_context.h> /* standard header files */
#include <lib_utils.h>
#include "trans_v1_1_udt.h" /* type definition of the user data types */
#include "tl_types.h" /* definition of base data types */
#include "sepfile.h" /* Include of own header file */
#include "trans_v1_1.h" /* Include of the root system header */

```

```

/*****\
TL.CG.MACROCL.STATIC: Default macro class for macros with module extent.
\*****/

```

```

#define Ca16_transition32_id      9
#define Ca3_first_gear_id      2
#define Ca4_fourth_gear_id      6
#define Ca19_transition34_id      5
#define Ca13_transition23_id      4
#define Ca10_transition21_id      10
#define Ca6_third_gear_id      1
#define Ca7_transition12_id      3
#define Ca5_second_gear_id      7
#define Ca22_transition43_id      8

/*****\
  defaultClass_sfStaticGlobal: Default storage class for global static variables
\*****/
static S8      Ca2_to_gear = 1;      /* LSB: 2^0 OFF: 0 MIN/MAX: -128 .. 127 */

/*****\
  defaultClass_slStaticGlobal: Default storage class for global static variables
\*****/
static F32      AUX_a_Sa5_m_s_to_km_h;      /* zaxxonnia */
static F32      AUX_a_Sa5_Product;
static F32      AUX_a_Sa5_Product1;
static F32      AUX_a_Sa5_Product2;
static F32      AUX_a_Sa5_Product4;
static F32      AUX_a_Sa5_Product5;

/*****\
  defaultClass_slGlobalInit: Default storage class for global variables with initial
value.
\*****/
F32      s43_x_table_array[8][2] =
{
  { /* [0][0..1] */ 10.F, 29.F},
  { /* [1][0..1] */ 25.F, 29.F},
  { /* [2][0..1] */ 30.F, 41.F},
  { /* [3][0..1] */ 40.F, 63.F},
  { /* [4][0..1] */ 60.F, 109.F},
  { /* [5][0..1] */ 70.F, 127.F},
  { /* [6][0..1] */ 80.F, 127.F},
  { /* [7][0..1] */ 100.F, 127.F}
};      /* 32 bit floating-point variable */
F32      s34_x_table_array[8][2] =
{
  { /* [0][0..1] */ 10.F, 49.F},
  { /* [1][0..1] */ 25.F, 49.F},
  { /* [2][0..1] */ 30.F, 77.F},
  { /* [3][0..1] */ 40.F, 98.F},
  { /* [4][0..1] */ 60.F, 127.F},
  { /* [5][0..1] */ 70.F, 127.F},
  { /* [6][0..1] */ 80.F, 127.F},
  { /* [7][0..1] */ 100.F, 127.F}
};      /* 32 bit floating-point variable */
F32      s32_x_table_array[8][2] =
{
  { /* [0][0..1] */ 10.F, 21.F},
  { /* [1][0..1] */ 25.F, 21.F},
  { /* [2][0..1] */ 30.F, 21.F},
  { /* [3][0..1] */ 40.F, 29.F},
  { /* [4][0..1] */ 60.F, 44.F},
  { /* [5][0..1] */ 70.F, 52.F},
  { /* [6][0..1] */ 80.F, 60.F},
  { /* [7][0..1] */ 100.F, 84.F}
};      /* 32 bit floating-point variable */
F32      s23_x_table_array[8][2] =
{
  { /* [0][0..1] */ 10.F, 34.F},
  { /* [1][0..1] */ 25.F, 34.F},
  { /* [2][0..1] */ 30.F, 56.F},
  { /* [3][0..1] */ 40.F, 70.F},
  { /* [4][0..1] */ 60.F, 98.F},
  { /* [5][0..1] */ 70.F, 126.F},
  { /* [6][0..1] */ 80.F, 127.F},

```

```

    { /* [7][0..1] */ 100.F, 127.F}
}; /* 32 bit floating-point variable */
F32 s21_x_table_array[8][2] =
{
    { /* [0][0..1] */ 10.F, 12.F},
    { /* [1][0..1] */ 25.F, 12.F},
    { /* [2][0..1] */ 30.F, 12.F},
    { /* [3][0..1] */ 40.F, 12.F},
    { /* [4][0..1] */ 60.F, 12.F},
    { /* [5][0..1] */ 70.F, 12.F},
    { /* [6][0..1] */ 80.F, 12.F},
    { /* [7][0..1] */ 100.F, 50.F}
}; /* 32 bit floating-point variable */
F32 c4_x_table_array[4][2] =
{
    { /* [0][0..1] */ 1.F, 0.F},
    { /* [1][0..1] */ 2.F, 0.F},
    { /* [2][0..1] */ 3.F, 0.F},
    { /* [3][0..1] */ 4.F, 1.F}
}; /* 32 bit floating-point variable */
F32 c3_x_table_array[4][2] =
{
    { /* [0][0..1] */ 1.F, 0.F},
    { /* [1][0..1] */ 2.F, 0.F},
    { /* [2][0..1] */ 3.F, 1.F},
    { /* [3][0..1] */ 4.F, 1.F}
}; /* 32 bit floating-point variable */
F32 c2_x_table_array[4][2] =
{
    { /* [0][0..1] */ 1.F, 0.F},
    { /* [1][0..1] */ 2.F, 1.F},
    { /* [2][0..1] */ 3.F, 1.F},
    { /* [3][0..1] */ 4.F, 1.F}
}; /* 32 bit floating-point variable */
F32 c1_x_table_array[4][2] =
{
    { /* [0][0..1] */ 1.F, 1.F},
    { /* [1][0..1] */ 2.F, 1.F},
    { /* [2][0..1] */ 3.F, 0.F},
    { /* [3][0..1] */ 4.F, 0.F}
}; /* 32 bit floating-point variable */
F32 b12_x_table_array[4][2] =
{
    { /* [0][0..1] */ 1.F, 1.F},
    { /* [1][0..1] */ 2.F, 1.F},
    { /* [2][0..1] */ 3.F, 0.F},
    { /* [3][0..1] */ 4.F, 0.F}
}; /* 32 bit floating-point variable */

/*****\
GLOBAL: global variables (RAM)
\*****/
TABLE_2D s43_x_table =
{
    8 /*
    comp : size
    Description: No of x,y pairs
    LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
    &((s43_x_table_array[0][0])) /*
    comp : xy_pairs
    Description: Pointer to start of x,y array
    pointer to 32 bit floating-point variable */
}; /* structure type */
TABLE_2D s34_x_table =
{
    8 /*
    comp : size
    Description: No of x,y pairs
    LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
    &((s34_x_table_array[0][0])) /*
    comp : xy_pairs
    Description: Pointer to start of x,y array

```

```

        pointer to 32 bit floating-point variable */
}; /* structure type */
TABLE_2D    s32_x_table =
{
    8 /*
        comp      : size
        Description: No of x,y pairs
        LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
    &((s32_x_table_array[0][0])) /*
        comp      : xy_pairs
        Description: Pointer to start of x,y array
        pointer to 32 bit floating-point variable */
}; /* structure type */
TABLE_2D    s23_x_table =
{
    8 /*
        comp      : size
        Description: No of x,y pairs
        LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
    &((s23_x_table_array[0][0])) /*
        comp      : xy_pairs
        Description: Pointer to start of x,y array
        pointer to 32 bit floating-point variable */
}; /* structure type */
TABLE_2D    s21_x_table =
{
    8 /*
        comp      : size
        Description: No of x,y pairs
        LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
    &((s21_x_table_array[0][0])) /*
        comp      : xy_pairs
        Description: Pointer to start of x,y array
        pointer to 32 bit floating-point variable */
}; /* structure type */
TABLE_2D    s12_x_table =
{
    8 /*
        comp      : size
        Description: No of x,y pairs
        LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
    &((s12_x_table_array[0][0])) /*
        comp      : xy_pairs
        Description: Pointer to start of x,y array
        pointer to 32 bit floating-point variable */
}; /* structure type */
TABLE_2D    c4_x_table =
{
    4 /*
        comp      : size
        Description: No of x,y pairs
        LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
    &((c4_x_table_array[0][0])) /*
        comp      : xy_pairs
        Description: Pointer to start of x,y array
        pointer to 32 bit floating-point variable */
}; /* structure type */
TABLE_2D    c3_x_table =
{
    4 /*
        comp      : size
        Description: No of x,y pairs
        LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
    &((c3_x_table_array[0][0])) /*
        comp      : xy_pairs
        Description: Pointer to start of x,y array
        pointer to 32 bit floating-point variable */
}; /* structure type */
TABLE_2D    c2_x_table =
{
    4 /*
        comp      : size

```

```

        Description: No of x,y pairs
        LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
    &((c2_x_table_array[0][0])) /*
        comp      : xy_pairs
        Description: Pointer to start of x,y array
        pointer to 32 bit floating-point variable */
};
/* structure type */
TABLE_2D      c1_x_table =
{
    4 /*
        comp      : size
        Description: No of x,y pairs
        LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
    &((c1_x_table_array[0][0])) /*
        comp      : xy_pairs
        Description: Pointer to start of x,y array
        pointer to 32 bit floating-point variable */
};
/* structure type */
TABLE_2D      b12_x_table =
{
    4 /*
        comp      : size
        Description: No of x,y pairs
        LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
    &((b12_x_table_array[0][0])) /*
        comp      : xy_pairs
        Description: Pointer to start of x,y array
        pointer to 32 bit floating-point variable */
};
/* structure type */

/*****\
    defaultClass_slGlobal: Default storage class for global variables.
\*****/
BFa9_tp      BFa9 =
{
    0 /*
        comp: Ca24_shifting
        boolean type */,
    0 /*
        comp: Ca23_shift_pending
        boolean type */,
    0 /*
        comp: Ca17_shift_pending
        boolean type */,
    0 /*
        comp: Ca21_shifting3
        boolean type */,
    0 /*
        comp: Ca20_shift_pending3
        boolean type */,
    0 /*
        comp: Ca15_shifting2
        boolean type */,
    0 /*
        comp: Ca18_shifting
        boolean type */,
    0 /*
        comp: Ca9_shifting
        boolean type */,
    0 /*
        comp: Ca14_shift_pending2
        boolean type */,
    0 /*
        comp: Ca11_shift_pending
        boolean type */,
    0 /*
        comp: Ca12_shifting
        boolean type */,
    0 /*
        comp: Ca8_shift_pending
        boolean type */,
    0 /*

```

```

    comp: Ca2_shift_schedule_ns
    LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 15 */,
0 /*
    comp: Ca2_shift_schedule
    boolean type */
}; /* structure type */

/*****\
  Model step function of subsystem(s): trans_v1_1/trans_slow/trans_slow_torques
  \*****/
void calc_trans_slow_torques(
  F32 Sa3____c /* 32 bit floating-point variable */,
  F32 Sa3____b /* 32 bit floating-point variable */,
  Bool8 Sa3____a /* boolean type */,
  S8 Sa3____ /* LSB: 2^0 OFF: 0 MIN/MAX: -128 .. 127 */,
  F32 * Sa3__pc1_ /* pointer to 32 bit floating-point variable */,
  F32 * Sa3__pc2_ /* pointer to 32 bit floating-point variable */,
  F32 * Sa3__pc3_ /* pointer to 32 bit floating-point variable */,
  F32 * Sa3__pc4_ /* pointer to 32 bit floating-point variable */,
  F32 * Sa3__pb12_ /* pointer to 32 bit floating-point variable */)
{
  /* defaultClass_slLocal: Default storage class for local variables */
  F32 Sa5_4_3shift; /* 32 bit floating-point variable */
  F32 Sa5_3_4_shift; /* 32 bit floating-point variable */
  F32 Sa5_3_2_shift; /* 32 bit floating-point variable */
  F32 Sa5_2_3_shift; /* 32 bit floating-point variable */
  F32 Sa5_2_1_shift; /* 32 bit floating-point variable */
  F32 Sa5_1_2_shift; /* 32 bit floating-point variable */
  F32 Sa4_c4_table; /* 32 bit floating-point variable */
  F32 Sa4_c3_table; /* 32 bit floating-point variable */
  F32 Sa4_c2_table; /* 32 bit floating-point variable */
  F32 Sa4_c1_table; /* 32 bit floating-point variable */
  F32 Sa4_b12_table; /* 32 bit floating-point variable */

  /*
  1D-CustomTableLookup:
  trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/1_2_shift
  evaluate 2D lookup function (replaces dSPACE Tab1D...) */
  Sa5_1_2_shift = lookup_2d(
    &(s12_x_table),
    Sa3____c);

  /*
  1D-CustomTableLookup:
  trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/2_1_shift
  evaluate 2D lookup function (replaces dSPACE Tab1D...) */
  Sa5_2_1_shift = lookup_2d(
    &(s21_x_table),
    Sa3____c);

  /*
  1D-CustomTableLookup:
  trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/2_3_shift
  evaluate 2D lookup function (replaces dSPACE Tab1D...) */
  Sa5_2_3_shift = lookup_2d(
    &(s23_x_table),
    Sa3____c);

  /*
  1D-CustomTableLookup:
  trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/3_2_shift
  evaluate 2D lookup function (replaces dSPACE Tab1D...) */
  Sa5_3_2_shift = lookup_2d(
    &(s32_x_table),
    Sa3____c);

  /*

```

```

1D-CustomTableLookup:
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/3_4_shift
  evaluate 2D lookup function (replaces dSPACE Tab1D...) */
  Sa5_3_4_shift = lookup_2d(
    &(s34_x_table),
    Sa3____c);

/*
1D-CustomTableLookup:
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/4_3shift
  evaluate 2D lookup function (replaces dSPACE Tab1D...) */
  Sa5_4_3shift = lookup_2d(
    &(s43_x_table),
    Sa3____c);

/* # combined # Gain:
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/m_s_to_km_h */
  AUX_a_Sa5_m_s_to_km_h = Sa3____b * 3.6F;

/*
# combined # Product:
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/Product
# combined # # combined # Switch:
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
  Switch
  Omitted comparison with constant. */
  AUX_a_Sa5_Product = Sa5_1_2_shift * Sa3____a;

/*
# combined # Product:
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/Product1
# combined # # combined # Switch:
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
  Switch
  Omitted comparison with constant. */
  AUX_a_Sa5_Product1 = Sa5_2_1_shift * Sa3____a;

/*
# combined # Product:
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/Product2
# combined # # combined # Switch:
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
  Switch
  Omitted comparison with constant. */
  AUX_a_Sa5_Product2 = Sa5_2_3_shift * Sa3____a;

/*
# combined # Product:
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/Product4
# combined # # combined # Switch:
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
  Switch
  Omitted comparison with constant. */
  AUX_a_Sa5_Product4 = Sa5_3_4_shift * Sa3____a;

/*
# combined # Product:
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/Product5
# combined # # combined # Switch:
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
  Switch
  Omitted comparison with constant. */
  AUX_a_Sa5_Product5 = Sa5_4_3shift * Sa3____a;
/* Stateflow: trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schеду
*/
/*
  Begin execution of chart
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schеду
  le */
  switch ( BFa9.Ca2_shift_schedule_ns ) {
    case Ca3_first_gear_id: {
      /*

```

```

Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.first_gear */

/* furry animals */
if ( AUX_a_Sa5_m_s_to_km_h > AUX_a_Sa5_Product ) {
/*
State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.first_gear to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.transition12.shift_pending */
BFa9.Ca2_shift_schedule_ns = Ca7_transition12_id;
BFa9.Ca8_shift_pending = 1;
ctr = 0;
Ca2_to_gear = 1;
}
/*
End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
hedule.first_gear */
break;
}

case Ca4_fourth_gear_id: {
/*
Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.fourth_gear */
if ( AUX_a_Sa5_m_s_to_km_h <= AUX_a_Sa5_Product5 ) {
/*
State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.fourth_gear to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.transition43.shift_pending */
BFa9.Ca2_shift_schedule_ns = Ca22_transition43_id;
BFa9.Ca23_shift_pending = 1;
ctr = 0;
Ca2_to_gear = 4;
}
/*
End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
hedule.fourth_gear */
break;
}

case Ca5_second_gear_id: {
/*
Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.second_gear */
if ( AUX_a_Sa5_m_s_to_km_h > AUX_a_Sa5_Product2 ) {
/*
State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.second_gear to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.transition23.shift_pending2 */
BFa9.Ca2_shift_schedule_ns = Ca13_transition23_id;
BFa9.Ca14_shift_pending2 = 1;
ctr = 0;
Ca2_to_gear = 2;
} else {
if ( AUX_a_Sa5_m_s_to_km_h <= AUX_a_Sa5_Product1 ) {
/*
State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.second_gear to
trans_vl_1/trans_slow/trans_slow_torques/shift_sched
uler/shift_schedule.transition21.shift_pending */

```

```

        BFa9.Ca2_shift_schedule_ns = Ca10_transition21_id;
        BFa9.Ca11_shift_pending = 1;
        ctr = 0;
        Ca2_to_gear = 2;
    }
}
/*
    End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
    hedule.second_gear */
    break;
}

case Ca6_third_gear_id: {
    /*
        Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
    schedule.third_gear */
    if ( AUX_a_Sa5_m_s_to_km_h > AUX_a_Sa5_Product4 ) {
        /*
            State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
    schedule.third_gear to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
    shift_schedule.transition34.shift_pending3 */
        BFa9.Ca2_shift_schedule_ns = Ca19_transition34_id;
        BFa9.Ca20_shift_pending3 = 1;
        ctr = 0;
        Ca2_to_gear = 3;
    } else {

        /*
            # combined # # combined # Product:
trans_vl_1/trans_slow/trans_slow_torques/shift_sch
    eduler/Product3
            # combined # # combined # Switch:
trans_vl_1/trans_slow/trans_slow_torques/shift_sche
    duler/Switch
            Omitted comparison with constant. */
        if ( AUX_a_Sa5_m_s_to_km_h <= (Sa5_3_2_shift * Sa3____a) ) {
            /*
                State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
    shift_schedule.third_gear to
trans_vl_1/trans_slow/trans_slow_torques/shift_schedu
    ler/shift_schedule.transition32.shift_pending */
            BFa9.Ca2_shift_schedule_ns = Ca16_transition32_id;
            BFa9.Ca17_shift_pending = 1;
            ctr = 0;
            Ca2_to_gear = 3;
        }
    }
}
/*
    End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
    hedule.third_gear */
    break;
}

case Ca7_transition12_id: {
    /*
        Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
    schedule.transition12 */
    if ( AUX_a_Sa5_m_s_to_km_h <= AUX_a_Sa5_Product1 ) {
        /*
            State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
    schedule.transition12 to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
    shift_schedule.first_gear */
        if ( BFa9.Ca8_shift_pending ) {

```

```

        BFa9.Ca8_shift_pending = 0;
    } else {
        if ( BFa9.Ca9_shifting ) {
            BFa9.Ca9_shifting = 0;
        }
    }
    BFa9.Ca2_shift_schedule_ns = Ca3_first_gear_id;
    Ca2_to_gear = 1;
} else {
    if ( BFa9.Ca8_shift_pending ) {
        /*
        Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
        shift_schedule.transition12.shift_pending */
        if ( ctr > Ca2_DELAY ) {
            /*
            State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
            shift_schedule.transition12.shift_pending to
trans_vl_1/trans_slow/trans_slow_t
            orques/shift_scheduler/shift_schedule.transition12.shifting */
            BFa9.Ca8_shift_pending = 0;
            BFa9.Ca9_shifting = 1;
            Ca2_to_gear = 2;
        } else {
            ctr = ctr + 1;
        }
    }
    /*
    End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
    shift_schedule.transition12.shift_pending */
} else {
    if ( BFa9.Ca9_shifting ) {
        /*
        Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
        er/shift_schedule.transition12.shifting */
        if ( Sa3___ == 2 ) {
            /*
            State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
            er/shift_schedule.transition12.shifting to
trans_vl_1/trans_slow/trans_slow_
            torques/shift_scheduler/shift_schedule.second_gear */
            if ( BFa9.Ca8_shift_pending ) {
                BFa9.Ca8_shift_pending = 0;
            } else {
                if ( BFa9.Ca9_shifting ) {
                    BFa9.Ca9_shifting = 0;
                }
            }
            BFa9.Ca2_shift_schedule_ns = Ca5_second_gear_id;
            Ca2_to_gear = 2;
        }
    }
    /*
    End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler
    /shift_schedule.transition12.shifting */
}
}
}
/*
End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
chedule.transition12 */
break;
}

case Ca10_transition21_id: {
    /*
    Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_

```

```

        schedule.transition21 */
    if ( AUX_a_Sa5_m_s_to_km_h > AUX_a_Sa5_Product ) {
        /*
            State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
        schedule.transition21 to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
            shift_schedule.second_gear */
        if ( BFa9.Call_shift_pending ) {
            BFa9.Call_shift_pending = 0;
        } else {
            if ( BFa9.Cal2_shifting ) {
                BFa9.Cal2_shifting = 0;
            }
        }
        BFa9.Ca2_shift_schedule_ns = Ca5_second_gear_id;
        Ca2_to_gear = 2;
    } else {
        if ( BFa9.Call_shift_pending ) {
            /*
                Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
            shift_schedule.transition21.shift_pending */
            if ( ctr > Ca2_DELAY ) {
                /*
                    State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
                shift_schedule.transition21.shift_pending to
trans_vl_1/trans_slow/trans_slow_t
                    orques/shift_scheduler/shift_schedule.transition21.shifting */
                BFa9.Call_shift_pending = 0;
                BFa9.Cal2_shifting = 1;
                Ca2_to_gear = 1;
            } else {
                ctr = ctr + 1;
            }
        }
        /*
            End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
        shift_schedule.transition21.shift_pending */
    } else {
        if ( BFa9.Cal2_shifting ) {
            /*
                Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
            er/shift_schedule.transition21.shifting */
            if ( Sa3___ == 1 ) {
                /*
                    State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
                er/shift_schedule.transition21.shifting to
trans_vl_1/trans_slow/trans_slow_
                    torques/shift_scheduler/shift_schedule.first_gear */
                if ( BFa9.Call_shift_pending ) {
                    BFa9.Call_shift_pending = 0;
                } else {
                    if ( BFa9.Cal2_shifting ) {
                        BFa9.Cal2_shifting = 0;
                    }
                }
                BFa9.Ca2_shift_schedule_ns = Ca3_first_gear_id;
                Ca2_to_gear = 1;
            }
        }
        /*
            End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler
        /shift_schedule.transition21.shifting */
    }
}
}
/*

```

```

        End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
        hedule.transition21 */
        break;
    }

    case Ca13_transition23_id: {
        /*
        Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
        schedule.transition23 */
        if ( AUX_a_Sa5_m_s_to_km_h <= AUX_a_Sa5_Product2 ) {
            /*
            State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
            schedule.transition23 to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
            shift_schedule.second_gear */
            if ( BFa9.Ca14_shift_pending2 ) {
                BFa9.Ca14_shift_pending2 = 0;
            } else {
                if ( BFa9.Ca15_shifting2 ) {
                    BFa9.Ca15_shifting2 = 0;
                }
            }
            BFa9.Ca2_shift_schedule_ns = Ca5_second_gear_id;
            Ca2_to_gear = 2;
        } else {
            if ( BFa9.Ca14_shift_pending2 ) {
                /*
                Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
                shift_schedule.transition23.shift_pending2 */
                if ( ctr > Ca2_DELAY ) {
                    /*
                    State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
                    shift_schedule.transition23.shift_pending2 to
trans_vl_1/trans_slow/trans_slow_
                    torques/shift_scheduler/shift_schedule.transition23.shifting2 */
                    BFa9.Ca14_shift_pending2 = 0;
                    BFa9.Ca15_shifting2 = 1;
                    Ca2_to_gear = 3;
                } else {
                    ctr = ctr + 1;
                }
            }
            /*
            End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
            shift_schedule.transition23.shift_pending2 */
        } else {
            if ( BFa9.Ca15_shifting2 ) {
                /*
                Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
                er/shift_schedule.transition23.shifting2 */
                if ( Sa3___ == 3 ) {
                    /*
                    State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
                    er/shift_schedule.transition23.shifting2 to
trans_vl_1/trans_slow/trans_slow_
                    _torques/shift_scheduler/shift_schedule.third_gear */
                    if ( BFa9.Ca14_shift_pending2 ) {
                        BFa9.Ca14_shift_pending2 = 0;
                    } else {
                        if ( BFa9.Ca15_shifting2 ) {
                            BFa9.Ca15_shifting2 = 0;
                        }
                    }
                }
                BFa9.Ca2_shift_schedule_ns = Ca6_third_gear_id;
                Ca2_to_gear = 3;
            }
        }
    }
}

```

```

        }
        /*
        End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler
        /shift_schedule.transition23.shifting2 */
    }
}
/*
End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
chedule.transition23 */
break;
}

case Ca16_transition32_id: {
    /*
    Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
chedule.transition32 */
    if ( AUX_a_Sa5_m_s_to_km_h > AUX_a_Sa5_Product2 ) {
        /*
        State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
chedule.transition32 to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.third_gear */
        if ( BFa9.Ca17_shift_pending ) {
            BFa9.Ca17_shift_pending = 0;
        } else {
            if ( BFa9.Ca18_shifting ) {
                BFa9.Ca18_shifting = 0;
            }
        }
        BFa9.Ca2_shift_schedule_ns = Ca6_third_gear_id;
        Ca2_to_gear = 3;
    } else {
        if ( BFa9.Ca17_shift_pending ) {
            /*
            Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.transition32.shift_pending */
            if ( ctr > Ca2_DELAY ) {
                /*
                State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.transition32.shift_pending to
trans_vl_1/trans_slow/trans_slow_t
orques/shift_scheduler/shift_schedule.transition32.shifting */
                BFa9.Ca17_shift_pending = 0;
                BFa9.Ca18_shifting = 1;
                Ca2_to_gear = 2;
            } else {
                ctr = ctr + 1;
            }
        }
        /*
        End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.transition32.shift_pending */
    } else {
        if ( BFa9.Ca18_shifting ) {
            /*
            Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
er/shift_schedule.transition32.shifting */
            if ( Sa3___ == 2 ) {
                /*
                State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
er/shift_schedule.transition32.shifting to
trans_vl_1/trans_slow/trans_slow_
torques/shift_scheduler/shift_schedule.second_gear */

```

```

        if ( BFa9.Ca17_shift_pending ) {
            BFa9.Ca17_shift_pending = 0;
        } else {
            if ( BFa9.Ca18_shifting ) {
                BFa9.Ca18_shifting = 0;
            }
        }
        BFa9.Ca2_shift_schedule_ns = Ca5_second_gear_id;
        Ca2_to_gear = 2;
    }
    /*
    End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler
    /shift_schedule.transition32.shifting */
    }
}
/*
End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
hedule.transition32 */
break;
}

case Ca19_transition34_id: {
    /*
    Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.transition34 */
    if ( AUX_a_Sa5_m_s_to_km_h <= AUX_a_Sa5_Product5 ) {
        /*
        State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.transition34 to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.third_gear */
        if ( BFa9.Ca20_shift_pending3 ) {
            BFa9.Ca20_shift_pending3 = 0;
        } else {
            if ( BFa9.Ca21_shifting3 ) {
                BFa9.Ca21_shifting3 = 0;
            }
        }
        BFa9.Ca2_shift_schedule_ns = Ca6_third_gear_id;
        Ca2_to_gear = 3;
    } else {
        if ( BFa9.Ca20_shift_pending3 ) {
            /*
            Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.transition34.shift_pending3 */
            if ( ctr > Ca2_DELAY ) {
                /*
                State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.transition34.shift_pending3 to
trans_vl_1/trans_slow/trans_slow_
torques/shift_scheduler/shift_schedule.transition34.shifting3 */
                BFa9.Ca20_shift_pending3 = 0;
                BFa9.Ca21_shifting3 = 1;
                Ca2_to_gear = 4;
            } else {
                ctr = ctr + 1;
            }
        }
        /*
        End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.transition34.shift_pending3 */
    } else {
        if ( BFa9.Ca21_shifting3 ) {
            /*

```

```

Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition34.shifting3 */
if ( Sa3___ == 4 ) {
/*
State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition34.shifting3 to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.fourth_gear */
if ( BFa9.Ca20_shift_pending3 ) {
BFa9.Ca20_shift_pending3 = 0;
} else {
if ( BFa9.Ca21_shifting3 ) {
BFa9.Ca21_shifting3 = 0;
}
}
BFa9.Ca2_shift_schedule_ns = Ca4_fourth_gear_id;
Ca2_to_gear = 4;
}
/*
End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition34.shifting3 */
}
}
/*
End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition34 */
break;
}

case Ca22_transition43_id: {
/*
Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition43 */
if ( AUX_a_Sa5_m_s_to_km_h > AUX_a_Sa5_Product4 ) {
/*
State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition43 to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.fourth_gear */
if ( BFa9.Ca23_shift_pending ) {
BFa9.Ca23_shift_pending = 0;
} else {
if ( BFa9.Ca24_shifting ) {
BFa9.Ca24_shifting = 0;
}
}
BFa9.Ca2_shift_schedule_ns = Ca4_fourth_gear_id;
Ca2_to_gear = 4;
} else {
if ( BFa9.Ca23_shift_pending ) {
/*
Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition43.shift_pending */
if ( ctr > Ca2_DELAY ) {
/*
State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition43.shift_pending to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition43.shifting */
BFa9.Ca23_shift_pending = 0;
BFa9.Ca24_shifting = 1;
Ca2_to_gear = 3;
} else {

```

```

        ctr = ctr + 1;
    }
    /*
    End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
    shift_schedule.transition43.shift_pending */
    } else {
        if ( BFa9.Ca24_shifting ) {
            /*
            Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
            er/shift_schedule.transition43.shifting */
            if ( Sa3___ == 3 ) {
                /*
                State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
                er/shift_schedule.transition43.shifting to
trans_vl_1/trans_slow/trans_slow_
                torques/shift_scheduler/shift_schedule.third_gear */
                if ( BFa9.Ca23_shift_pending ) {
                    BFa9.Ca23_shift_pending = 0;
                } else {
                    if ( BFa9.Ca24_shifting ) {
                        BFa9.Ca24_shifting = 0;
                    }
                }
                BFa9.Ca2_shift_schedule_ns = Ca6_third_gear_id;
                Ca2_to_gear = 3;
            }
            /*
            End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler
            /shift_schedule.transition43.shifting */
        }
    }
}
/*
End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
hedule.transition43 */
break;
}

default: {
    if ( !(BFa9.Ca2_shift_schedule) ) {
        BFa9.Ca2_shift_schedule = 1;
        BFa9.Ca2_shift_schedule_ns = Ca3_first_gear_id;
        Ca2_to_gear = 1;
    }
}
}
/* End execution of chart
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule
*/

/*
1D-CustomTableLookup:
trans_vl_1/trans_slow/trans_slow_torques/clutch_pressure/b12_table
evaluate 2D lookup function (replaces dSPACE Tab1D...) */
Sa4_b12_table = lookup_2d(
    &(b12_x_table),
    (F32)Ca2_to_gear);

/* Outport: trans_vl_1/trans_slow/trans_slow_torques/PB12
# combined # Gain: trans_vl_1/trans_slow/trans_slow_torques/clutch_pressure/Gain4
*/
*Sa3_pb12_ = Sa4_b12_table * 1000.F;

/*
1D-CustomTableLookup:
trans_vl_1/trans_slow/trans_slow_torques/clutch_pressure/cl_table

```

```

        evaluate 2D lookup function (replaces dSPACE Tab1D...) */
Sa4_c1_table = lookup_2d(
    &(c1_x_table),
    (F32)Ca2_to_gear);

/* Outport: trans_v1_1/trans_slow/trans_slow_torques/Pc1
# combined # Gain: trans_v1_1/trans_slow/trans_slow_torques/clutch_pressure/Gain1
*/
*Sa3_pc1_ = Sa4_c1_table * 1000.F;

/*
1D-CustomTableLookup:
trans_v1_1/trans_slow/trans_slow_torques/clutch_pressure/c2_table
evaluate 2D lookup function (replaces dSPACE Tab1D...) */
Sa4_c2_table = lookup_2d(
    &(c2_x_table),
    (F32)Ca2_to_gear);

/* Outport: trans_v1_1/trans_slow/trans_slow_torques/Pc2
# combined # Gain: trans_v1_1/trans_slow/trans_slow_torques/clutch_pressure/Gain1
*/
*Sa3_pc2_ = Sa4_c2_table * 1000.F;

/*
1D-CustomTableLookup:
trans_v1_1/trans_slow/trans_slow_torques/clutch_pressure/c3_table
evaluate 2D lookup function (replaces dSPACE Tab1D...) */
Sa4_c3_table = lookup_2d(
    &(c3_x_table),
    (F32)Ca2_to_gear);

/* Outport: trans_v1_1/trans_slow/trans_slow_torques/Pc3
# combined # Gain: trans_v1_1/trans_slow/trans_slow_torques/clutch_pressure/Gain2
*/
*Sa3_pc3_ = Sa4_c3_table * 1000.F;

/*
1D-CustomTableLookup:
trans_v1_1/trans_slow/trans_slow_torques/clutch_pressure/c4_table
evaluate 2D lookup function (replaces dSPACE Tab1D...) */
Sa4_c4_table = lookup_2d(
    &(c4_x_table),
    (F32)Ca2_to_gear);

/* Outport: trans_v1_1/trans_slow/trans_slow_torques/Pc4
# combined # Gain: trans_v1_1/trans_slow/trans_slow_torques/clutch_pressure/Gain3
*/
*Sa3_pc4_ = Sa4_c4_table * 1000.F;
}

```

```

#undef _SEPFILC_
/*****\

```

```

Model code file      : trans_v1_1.c
for TargetLink model : tl_transfixed/trans_v1_1

```

```

Generated by TargetLink, the dSPACE production quality code generator
Tue Feb 26 10:39:20 2002

```

```

CODE GENERATOR OPTIONS:
Target                : Generic
ANSI-C compatible code : yes
Optimization level    : 5
Constant style        : decimal
Clean code option     : enabled
Logging mode          : Do not log anything
Linker sections       : enabled
Interrupt functions   : enabled
User attributes       : enabled
Assembler statements  : disabled

```

```

Variable name length      : 31 chars
Separate lookup search function : disabled
Use global bitfields      : disabled
Stateflow: use of bitfields : enabled
State activity encoding limit : 5
Omit zero inits in restart function: disabled
Share fcns between TL subsystems : disabled
Generate 64bit functions  : enabled
Inlining Threshold       : 6
Line break limit         : 100
Constant suffix          : disabled
Target optimized boolean data type : enabled

Subsys  Corresponding Simulink Subsystem
Sa1     trans_v1_1
Sa2     trans_v1_1/trans_slow
Sa3     trans_v1_1/trans_slow/trans_slow_torques
Sa4     trans_v1_1/trans_slow/trans_slow_torques/clutch_pressure
Sa5     trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler

SF-Node  Corresponding Stateflow Node
         description
Ca0     trans_v1_1/Stateflow machine [tl_transfixed]

Ca1     trans_v1_1/trans_slow/trans_slow_ctl

Ca2     trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule

Ca3
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.first_gear

Ca4
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.fourth_gear

Ca5
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.second_gear

Ca6
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.third_gear

Ca7
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition12

Ca8
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition12.shi
ft_pending

Ca9
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition12.shi
fting

Ca10
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition21

Ca11
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition21.shi
ft_pending

Ca12
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition21.shi
fting

Ca13
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition23

Ca14
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition23.shi
ft_pending2

Ca15
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition23.shi
fting2

Ca16
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition32

Ca17
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition32.shi

```

```

    ft_pending
    Ca18
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition32.shi
    ftng
    Ca19
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition34

    Ca20
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition34.shi
    ft_pending3
    Ca21
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition34.shi
    ftng3
    Ca22
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition43

    Ca23
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition43.shi
    ft_pending
    Ca24
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition43.shi
    ftng

    TargetLink version    : 1.2p1 from 19-Jul-2001
    Codegenerator version : 1.2.P1 from Jul 19 2001, 17:45:00
    Copyright (c) 1999-2001 by dSPACE GmbH
\*****

#define _TRANS_Vl_1_C_    /* identifier for this file */

\*****\
    includes
\*****/

#include "tl_types.h"      /* definition of base data types */
#include "trans_vl_1.h"    /* Include of own header file */
#include "trans_vl_1_udt.h" /* type definition of the user data types */
#include "sepfile.h"

\*****\
    defaultClass_sfGlobalInit: Default storage class for global variables with initial
value.
\*****/
S8    ctr = 0;          /* LSB: 2^0 OFF:  0 MIN/MAX: -128 .. 127 */

\*****\
    defaultClass_slStaticGlobalInit: Default storage class for global static variables
with initial
value
\*****/
static F32    Sa3__pc1_ = 0.F;    /* 32 bit floating-point variable */
static F32    Sa3__pc2_ = 0.F;    /* 32 bit floating-point variable */
static F32    Sa3__pc3_ = 0.F;    /* 32 bit floating-point variable */
static F32    Sa3__pc4_ = 0.F;    /* 32 bit floating-point variable */
static F32    Sa3__pb12_ = 0.F;   /* 32 bit floating-point variable */

\*****\
    defaultClass_slStaticGlobal: Default storage class for global static variables
\*****/
static F32    AUX_a_tps;
static F32    AUX_a_vss;
static Bool8  AUX_a_p_e_switch;
static S8     AUX_a_a_gear;

\*****\
    Stateflow functions
\*****/

\*****\
    Model step function of Stateflow chart: trans_vl_1/trans_slow/trans_slow_ctl

```

```

\*****/
void Cal_trans_slow_ctl(void)
{
    /* Call of function: calc_trans_slow_torques */
    calc_trans_slow_torques(
        AUX_a_tps,
        AUX_a_vss,
        AUX_a_p_e_switch,
        AUX_a_a_gear,
        &(Sa3__pc1_),
        &(Sa3__pc2_),
        &(Sa3__pc3_),
        &(Sa3__pc4_),
        &(Sa3__pb12_));
}

\*****\
Model step function of subsystem(s): trans_v1_1
\*****/
void trans_v1_1(
    F32 tps /* 32 bit floating-point variable */,
    F32 vss /* 32 bit floating-point variable */,
    Bool8 p_e_switch /* boolean type */,
    S8 a_gear /* LSB: 2^0 OFF: 0 MIN/MAX: -128 .. 127 */,
    F32 * pc1 /* pointer to 32 bit floating-point variable */,
    F32 * pc2 /* pointer to 32 bit floating-point variable */,
    F32 * pc3 /* pointer to 32 bit floating-point variable */,
    F32 * pc4 /* pointer to 32 bit floating-point variable */,
    F32 * pc5 /* pointer to 32 bit floating-point variable */)
{
    AUX_a_tps = tps;
    AUX_a_vss = vss;
    AUX_a_p_e_switch = p_e_switch;
    AUX_a_a_gear = a_gear;

    /* TargetLink outpost: trans_v1_1/Pc1_ */
    *pc1 = Sa3__pc1_;

    /* TargetLink outpost: trans_v1_1/Pc2_ */
    *pc2 = Sa3__pc2_;

    /* TargetLink outpost: trans_v1_1/Pc3_ */
    *pc3 = Sa3__pc3_;

    /* TargetLink outpost: trans_v1_1/Pc4_ */
    *pc4 = Sa3__pc4_;

    /* TargetLink outpost: trans_v1_1/PB12_ */
    *pc5 = Sa3__pb12_;
}

#undef _TRANS_V1_1_C_

```

8.7 APPENDIX G: Fixed-Point C Code Generated by TargetLink for Transmission Model

```
/******\
Model code file      : trans_v1_1.c
for TargetLink model : tl_transfixed/trans_v1_1

Generated by TargetLink, the dSPACE production quality code generator
Tue Mar 12 09:54:44 2002

CODE GENERATOR OPTIONS:
Target                : Generic
ANSI-C compatible code : yes
Optimization level    : 5
Constant style        : decimal
Clean code option     : enabled
Logging mode          : Do not log anything
Linker sections       : enabled
Interrupt functions   : enabled
User attributes       : enabled
Assembler statements  : disabled
Variable name length  : 31 chars
Separate lookup search function : disabled
Use global bitfields  : disabled
Stateflow: use of bitfields : enabled
State activity encoding limit : 5
Omit zero inits in restart function: disabled
Share fcns between TL subsystems : disabled
Generate 64bit functions : enabled
Inlining Threshold    : 6
Line break limit      : 100
Constant suffix       : disabled
Target optimized boolean data type : enabled

Subsys  Corresponding Simulink Subsystem
Sa1      trans_v1_1
Sa2      trans_v1_1/trans_slow
Sa3      trans_v1_1/trans_slow/trans_slow_torques
Sa4      trans_v1_1/trans_slow/trans_slow_torques/clutch_pressure
Sa5      trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler

SF-Node  Corresponding Stateflow Node
          description
Ca0      trans_v1_1/Stateflow machine [tl_transfixed]

Ca1      trans_v1_1/trans_slow/trans_slow_ctl

Ca2      trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule

Ca3
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.first_gear

Ca4
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.fourth_gear

Ca5
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.second_gear

Ca6
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.third_gear

Ca7
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition12

Ca8
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition12.shi
ft_pending

Ca9
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition12.shi
fting
```

```

Ca10
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition21

Ca11
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition21.shi
ft_pending

Ca12
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition21.shi
fting

Ca13
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition23

Ca14
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition23.shi
ft_pending2

Ca15
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition23.shi
fting2

Ca16
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition32

Ca17
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition32.shi
ft_pending

Ca18
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition32.shi
fting

Ca19
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition34

Ca20
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition34.shi
ft_pending3

Ca21
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition34.shi
fting3

Ca22
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition43

Ca23
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition43.shi
ft_pending

Ca24
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition43.shi
fting

```

```

TargetLink version      : 1.2p1 from 19-Jul-2001
Codegenerator version  : 1.2.P1 from Jul 19 2001, 17:45:00
Copyright (c) 1999-2001 by dSPACE GmbH
\*****/

#define _TRANS_Vl_1_C_    /* identifier for this file */

\*****\
  includes
\*****/

#include "tl_types.h"      /* definition of base data types */
#include "trans_vl_1.h"    /* Include of own header file */
#include "trans_vl_1_udt.h" /* type definition of the user data types */
#include "sepfiler.h"
#include "DSFxp.h"

\*****\
  defaultClass_sfGlobalInit: Default storage class for global variables with initial
value.
\*****/
S8      ctr = 0;          /* LSB: 2^0 OFF:  0 MIN/MAX: -128 .. 127 */

\*****\

```

```

    defaultClass_slStaticGlobalInit: Default storage class for global static variables
with initial
    value
\*****/
static S16 Sa3__pc1_ = 0; /* LSB: 2^-4 OFF: 0 MIN/MAX: -2048 .. 2047.9375 */
static S16 Sa3__pc2_ = 0; /* LSB: 2^-4 OFF: 0 MIN/MAX: -2048 .. 2047.9375 */
static S16 Sa3__pc3_ = 0; /* LSB: 2^-4 OFF: 0 MIN/MAX: -2048 .. 2047.9375 */
static S16 Sa3__pc4_ = 0; /* LSB: 2^-4 OFF: 0 MIN/MAX: -2048 .. 2047.9375 */
static S16 Sa3__pb12_ = 0; /* LSB: 2^-4 OFF: 0 MIN/MAX: -2048 .. 2047.9375 */

/*****\
    defaultClass_slStaticGlobal: Default storage class for global static variables
\*****/
static S16 AUX_a_tps;
static S16 AUX_a_vss;
static Bool8 AUX_a_p_e_switch;
static S8 AUX_a_a_gear;

/*****\
    Stateflow functions
\*****/

/*****\
    Model step function of Stateflow chart: trans_vl_1/trans_slow/trans_slow_ctl
\*****/
void Cal_trans_slow_ctl(void)
{
    /* Call of function: calc_trans_slow_torques */
    calc_trans_slow_torques(
        AUX_a_tps,
        AUX_a_vss,
        AUX_a_p_e_switch,
        AUX_a_a_gear,
        &(Sa3__pc1_),
        &(Sa3__pc2_),
        &(Sa3__pc3_),
        &(Sa3__pc4_),
        &(Sa3__pb12_));
}

/*****\
    table lookup functions
\*****/

/*****\
*
* FUNCTION:
* Tab1DS48I80T28676_a
*
* DESCRIPTION:
* Generated function for look-up tables.
*
* PARAMETERS:
* type name description
* ~~~~~
* MAP_Tab1DS48I80T28676_a* map struct with all necessary informations
* S16 x table input
*
* RETURNS:
* U8 interpolated z-value.
*
* TABLE SETTINGS:
* Search algorithm.....: ascending linear search.
* Behaviour at boundaries.....: extrapolate
* Behaviour between defined points...: interpolate
* Distances fit into given bitlengths: no
* Boundary points.....: no

```

```

*   Starting point.....: - (non-equidistant implementation)
*   Scaling unit.....: - (non-equidistant implementation)
*
\*****/
U8 Tab1DS48I80T28676_a(
  const MAP_Tab1DS48I80T28676_a *   map   /* pointer to structure type */,
  S16 x   /* LSB: 2^0 OFF: 0 MIN/MAX: -32768 .. 32767 */
{
  /* VCM_VARCL_SL_LUT_LOCAL_DEFAULT: Default storage class for local variables */
  const S16 *   x_table; /*
    Scaling may differ through function reuse.
    pointer to LSB: 2^-8 OFF: 0 MIN/MAX: -128 .. 127.9960938 */
  const U8 * z_table; /*
    Scaling may differ through function reuse.
    pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */

  x_table = map->x_table;
  z_table = map->z_table;

  /* Extrapolation */
  if ( x >= (x_table[(map->N) - 1] ) ) {
    x_table += (map->N) - 2;
    z_table += (map->N) - 2;
  } else {
    x_table++;
    /* Linear search, start low. */

    while ( x > *(x_table++) ) {
      z_table++;
    }
    x_table -= 2;
  }
  /* division by 0 protection */
  if ( *x_table == *(x_table+1) )
    return (U8)*z_table ;

  { /* begin interpolation formula */

    S16 x1 = *(x_table++),          /* x1 <= x <= x2 */
        x2 = *x_table;
    U8  z1 = *(z_table++),          /* z1 <= z <= z2 */
        z2 = *z_table;

    S32      z      ; /* returned table value          */
    U16      deltaz ; /* (z2-z1) or (z1-z2)          */
    U16      deltax ; /* (x - x1)                  */
    U16      x2x1   ; /* intermediate x-difference. */
    unsigned int neg = 0; /* sign-flag, 0=pos. 1=neg. */

    if ( x < x1 )                /* calculate deltax and deltaz */
    {
      deltax = x1 - x;
      neg    = 1;
    }
    else
      deltax = x - x1;

    if ( z1 < z2 )
      deltaz = z2 - z1;
    else
    {
      deltaz = z1 - z2;
      neg    ^= 1;
    }

    x2x1 = (S16)x2 - (S16)x1;

    z = ( (U32)deltaz * (U32)deltax) / x2x1;
    z &= (INT32MAX >> 1); /* 2^30, saturate for next sub/add */
    z = (neg) ? (S32)z1 - z : z + (S32)z1;
  }
}

```

```

        return C__U8FITI32_SAT (z, UINT8MAX);
    } /* end interpolation formula */
}

/*****\
*
* FUNCTION:
*   Tab1DS16I80T28672_a
*
* DESCRIPTION:
*   Generated function for look-up tables.
*
* PARAMETERS:
*   type                name  description
*   ~~~~~
*   MAP_Tab1DS16I80T28672_a*  map  struct with all necessary informations
*   U8                       x    table input
*
* RETURNS:
*   U8 interpolated z-value.
*
* TABLE SETTINGS:
*   Search algorithm.....: ascending linear search.
*   Behaviour at boundaries.....: saturate
*   Behaviour between defined points...: interpolate
*   Distances fit into given bitlengths: no
*   Boundary points.....: no
*   Starting point.....: - (non-equidistant implementation)
*   Scaling unit.....: - (non-equidistant implementation)
*
\*****/
U8 Tab1DS16I80T28672_a(
const MAP_Tab1DS16I80T28672_a *      map /* pointer to structure type */,
U8 x /* LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */)
{
    /* VCM_VARCL_SL_LUT_LOCAL_DEFAULT: Default storage class for local variables */
    const U8 *x_table; /*
        Scaling may differ through function reuse.
        pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */
    const U8 *z_table; /*
        Scaling may differ through function reuse.
        pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */

    x_table = map->x_table;
    z_table = map->z_table;

    /* Saturation */
    if ( x <= (x_table[0]) ) {
        return z_table[0];
    }
    if ( x >= (x_table[(map->N) - 1]) ) {
        return z_table[(map->N) - 1];
    }
    x_table++;
    /* Linear search, start low. */

    while ( x > (*(x_table++)) ) {
        z_table++;
    }
    x_table -= 2;
    { /* begin interpolation formula */

        U8 d_0, d_1;

```

```

    /* These difference cannot overflow */
    /* if the x,y pairs are ordered correctly */
    d_0 = x - x_table[0];
    d_1 = x_table[1] - x;

    return (U8)( (U16)
        ( (U16)( (U16)d_1 * (U16)(z_table[0]) )
          + (U16)( (U16)d_0 * (U16)(z_table[1]) ) )
        / (U8)(d_0 + d_1) );
} /* end interpolation formula */
}

```

```

/*****\
*
* FUNCTION:
*   Tab1DS48I80T28672_a
*
* DESCRIPTION:
*   Generated function for look-up tables.
*
* PARAMETERS:
*   type           name  description
*   ~~~~~
*   MAP_Tab1DS48I80T28672_a*  map  struct with all necessary informations
*   U8                      x    table input
*
* RETURNS:
*   U8 interpolated z-value.
*
* TABLE SETTINGS:
*   Search algorithm.....: ascending linear search.
*   Behaviour at boundaries.....: extrapolate
*   Behaviour between defined points...: interpolate
*   Distances fit into given bitlengths: no
*   Boundary points.....: no
*   Starting point.....: - (non-equidistant implementation)
*   Scaling unit.....: - (non-equidistant implementation)
*

```

```

\*****/
U8 Tab1DS48I80T28672_a(
    const MAP_Tab1DS48I80T28672_a * map /* pointer to structure type */,
    U8 x /* LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */)
{
    /* VCM_VARCL_SL_LUT_LOCAL_DEFAULT: Default storage class for local variables */
    const U8 *x_table; /*
        Scaling may differ through function reuse.
        pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */
    const U8 *z_table; /*
        Scaling may differ through function reuse.
        pointer to LSB: 2^-2 OFF: 0 MIN/MAX: 0 .. 63.75 */

    x_table = map->x_table;
    z_table = map->z_table;

    /* Extrapolation */
    if ( x >= (x_table[(map->N) - 1]) ) {
        x_table += (map->N) - 2;
        z_table += (map->N) - 2;
    } else {
        x_table++;
        /* Linear search, start low. */
        while ( x > *(x_table++) ) {
            z_table++;
        }
        x_table -= 2;
    }
}

```

```

}
/* division by 0 protection */
if ( *x_table == *(x_table+1) )
    return (U8)*z_table ;

{ /* begin interpolation formula */

    U8 x1 = *(x_table++),          /* x1 <= x <= x2 */
      x2 = *x_table;
    U8 z1 = *(z_table++),          /* z1 <= z <= z2 */
      z2 = *z_table;

    S16      z      ; /* returned table value          */
    U8      deltaz; /* (z2-z1) or (z1-z2)          */
    U8      deltax; /* (x - x1)                    */
    unsigned int neg = 0; /* sign-flag, 0=pos. 1=neg.    */

    if (x < x1)                    /* calculate deltax and deltaz */
    {
        deltax = x1 - x;
        neg     = 1;
    }
    else
        deltax = x - x1;

    if (z1 < z2)
        deltaz = z2 - z1;
    else
    {
        deltaz = z1 - z2;
        neg     ^= 1;
    }

    z = ( (U16)deltaz * (U16)deltax) / (x2 - x1);
    z &= (INT16MAX >> 1); /* 2^14, saturate for next sub/add */
    z = (neg) ? (S16)z1 - z : z + (S16)z1;

    return C__U8FIT16_SAT (z, UINT8MAX);

} /* end interpolation formula */

}

/*****\
  Model step function of subsystem(s): trans_v1_1
  \*****/
void trans_v1_1(
    S16 tps /* LSB: 2^-7 OFF: 0 MIN/MAX: -256 .. 255.9921875 */,
    S16 vss /* LSB: 2^-8 OFF: 0 MIN/MAX: -128 .. 127.9960938 */,
    Bool8 p_e_switch /* boolean type */,
    S8 a_gear /* LSB: 2^-3 OFF: 0 MIN/MAX: -16 .. 15.875 */,
    S16 * pc1 /* pointer to LSB: 2^-4 OFF: 0 MIN/MAX: -2048 .. 2047.9375 */,
    S16 * pc2 /* pointer to LSB: 2^-4 OFF: 0 MIN/MAX: -2048 .. 2047.9375 */,
    S16 * pc3 /* pointer to LSB: 2^-4 OFF: 0 MIN/MAX: -2048 .. 2047.9375 */,
    S16 * pc4 /* pointer to LSB: 2^-4 OFF: 0 MIN/MAX: -2048 .. 2047.9375 */,
    S16 * pc5 /* pointer to LSB: 2^-4 OFF: 0 MIN/MAX: -2048 .. 2047.9375 */)
{
    AUX_a_tps = tps;
    AUX_a_vss = vss;
    AUX_a_p_e_switch = p_e_switch;
    AUX_a_a_gear = a_gear;

    /* TargetLink outport: trans_v1_1/Pc1_ */
    *pc1 = Sa3__pc1_;

    /* TargetLink outport: trans_v1_1/Pc2_ */
    *pc2 = Sa3__pc2_;

    /* TargetLink outport: trans_v1_1/Pc3_ */
    *pc3 = Sa3__pc3_;
}

```

```

/* TargetLink outpost: trans_v1_1/Pc4_ */
*pc4 = Sa3_pc4_;

/* TargetLink outpost: trans_v1_1/PB12_ */
*pc5 = Sa3_pb12_;
}

```

```
#undef _TRANS_V1_1_C_
```

```
/*****\
```

```

Model code file      : sepfile.c
for TargetLink model : tl_transfixed/trans_v1_1

```

```

Generated by TargetLink, the dSPACE production quality code generator
Tue Mar 12 09:54:44 2002

```

```
CODE GENERATOR OPTIONS:
```

```

Target                : Generic
ANSI-C compatible code : yes
Optimization level    : 5
Constant style        : decimal
Clean code option     : enabled
Logging mode          : Do not log anything
Linker sections       : enabled
Interrupt functions   : enabled
User attributes       : enabled
Assembler statements  : disabled
Variable name length  : 31 chars
Separate lookup search function : disabled
Use global bitfields  : disabled
Stateflow: use of bitfields : enabled
State activity encoding limit : 5
Omit zero inits in restart function: disabled
Share fcns between TL subsystems : disabled
Generate 64bit functions : enabled
Inlining Threshold    : 6
Line break limit      : 100
Constant suffix       : disabled
Target optimized boolean data type : enabled

```

```
Subsys Corresponding Simulink Subsystem
```

```

Sa1 trans_v1_1
Sa2 trans_v1_1/trans_slow
Sa3 trans_v1_1/trans_slow/trans_slow_torques
Sa4 trans_v1_1/trans_slow/trans_slow_torques/clutch_pressure
Sa5 trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler

```

```
SF-Node Corresponding Stateflow Node
description
```

```

Ca0 trans_v1_1/Stateflow machine [tl_transfixed]

Ca1 trans_v1_1/trans_slow/trans_slow_ctl

Ca2 trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule

Ca3
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.first_gear

Ca4
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.fourth_gear

Ca5
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.second_gear

Ca6
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.third_gear

```

```

Ca7
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition12

Ca8
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition12.shi
ft_pending

Ca9
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition12.shi
fting

Ca10
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition21

Ca11
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition21.shi
ft_pending

Ca12
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition21.shi
fting

Ca13
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition23

Ca14
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition23.shi
ft_pending2

Ca15
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition23.shi
fting2

Ca16
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition32

Ca17
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition32.shi
ft_pending

Ca18
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition32.shi
fting

Ca19
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition34

Ca20
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition34.shi
ft_pending3

Ca21
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition34.shi
fting3

Ca22
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition43

Ca23
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition43.shi
ft_pending

Ca24
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule.transition43.shi
fting

TargetLink version      : 1.2p1 from 19-Jul-2001
Codegenerator version  : 1.2.P1 from Jul 19 2001, 17:45:00
Copyright (c) 1999-2001 by dSPACE GmbH
\*****/

#define _SEPFILE_C_ /* identifier for this file */

\*****\
  includes
\*****/

#include "tl_types.h"      /* definition of base data types */
#include "sepfile.h"/* Include of own header file */
#include "trans_v1_1.udt.h" /* type definition of the user data types */
#include "trans_v1_1.h"    /* Include of the root system header */

```

```

\*****\
  TL.CG.MACROCL.STATIC: Default macro class for macros with module extent.
\*****/
#define Ca22_transition43_id      8
#define Ca6_third_gear_id        1
#define Ca10_transition21_id     10
#define Ca19_transition34_id     5
#define Ca16_transition32_id     9
#define Ca13_transition23_id     4
#define Ca3_first_gear_id        2
#define Ca7_transition12_id      3
#define Ca4_fourth_gear_id      6
#define Ca5_second_gear_id      7

\*****\
  defaultClass_slGlobalConst: Default storage class for global const variables
\*****/
const S16      s32_x_table[8] =
{
  /* [0..7] */ 2560, 6400, 7680, 10240, 15360, 17920, 20480, 25600
  /* 10., 25., 30., 40., 60., 70., 80., 100. */
}; /* LSB: 2^-8 OFF: 0 MIN/MAX: -128 .. 127.9960938 */
const U8      s43_x_table[8] =
{
  /* [0..7] */ 20, 50, 60, 80, 120, 140, 160, 200
  /* 10., 25., 30., 40., 60., 70., 80., 100. */
}; /* LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */
const U8      Sa5_4_3shift_z_table[8] =
{
  /* [0..7] */ 29, 29, 41, 63, 109, 131, 154, 154
  /* 29., 29., 41., 63., 109., 131., 154., 154. */
}; /* LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */
const U8      s34_x_table[8] =
{
  /* [0..7] */ 20, 50, 60, 80, 120, 140, 160, 200
  /* 10., 25., 30., 40., 60., 70., 80., 100. */
}; /* LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */
const U8      Sa5_3_4_shift_z_table[8] =
{
  /* [0..7] */ 49, 49, 77, 98, 154, 176, 198, 198
  /* 49., 49., 77., 98., 154., 176., 198., 198. */
}; /* LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */
const U8      Sa5_3_2_shift_z_table[8] =
{
  /* [0..7] */ 42, 42, 42, 58, 89, 104, 120, 168
  /* 21., 21., 21., 29., 44.5, 52., 60., 84. */
}; /* LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */
const U8      s23_x_table[8] =
{
  /* [0..7] */ 20, 50, 60, 80, 120, 140, 160, 200
  /* 10., 25., 30., 40., 60., 70., 80., 100. */
}; /* LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */
const U8      Sa5_2_3_shift_z_table[8] =
{
  /* [0..7] */ 34, 34, 56, 70, 98, 126, 144, 144
  /* 34., 34., 56., 70., 98., 126., 144., 144. */
}; /* LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */
const U8      s21_x_table[8] =
{
  /* [0..7] */ 20, 50, 60, 80, 120, 140, 160, 200
  /* 10., 25., 30., 40., 60., 70., 80., 100. */
}; /* LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */
const U8      Sa5_2_1_shift_z_table[8] =
{
  /* [0..7] */ 48, 48, 48, 48, 48, 48, 48, 202
  /* 12., 12., 12., 12., 12., 12., 12., 50.5 */
}; /* LSB: 2^-2 OFF: 0 MIN/MAX: 0 .. 63.75 */
const U8      s12_x_table[8] =
{
  /* [0..7] */ 20, 50, 60, 80, 120, 140, 160, 200
  /* 10., 25., 30., 40., 60., 70., 80., 100. */
}

```

```

}; /* LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */
const U8 Sa5_1_2_shift_z_table[8] =
{
    /* [0..7] */ 28, 28, 58, 77, 116, 136, 136, 136
    /* 14., 14., 29., 38.5, 58., 68., 68., 68. */
}; /* LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */
const U8 c4_x_table[4] =
{
    /* [0..3] */ 32, 64, 96, 128
    /* 1., 2., 3., 4. */
}; /* LSB: 2^-5 OFF: 0 MIN/MAX: 0 .. 7.96875 */
const U8 Sa4_c4_table_z_table[4] =
{
    /* [0..3] */ 64, 64, 64, 192
    /* 0.00390625, 0.00390625, 0.00390625, 1.00390625 */
}; /* LSB: 2^-7 OFF: -0.4961 MIN/MAX: -0.49609375 .. 1.49609375 */
const U8 c3_x_table[4] =
{
    /* [0..3] */ 32, 64, 96, 128
    /* 1., 2., 3., 4. */
}; /* LSB: 2^-5 OFF: 0 MIN/MAX: 0 .. 7.96875 */
const U8 Sa4_c3_table_z_table[4] =
{
    /* [0..3] */ 0, 0, 128, 128
    /* 0., 0., 1., 1. */
}; /* LSB: 2^-7 OFF: 0 MIN/MAX: 0 .. 1.9921875 */
const U8 c2_x_table[4] =
{
    /* [0..3] */ 32, 64, 96, 128
    /* 1., 2., 3., 4. */
}; /* LSB: 2^-5 OFF: 0 MIN/MAX: 0 .. 7.96875 */
const U8 Sa4_c2_table_z_table[4] =
{
    /* [0..3] */ 0, 128, 128, 128
    /* 0., 1., 1., 1. */
}; /* LSB: 2^-7 OFF: 0 MIN/MAX: 0 .. 1.9921875 */
const U8 c1_x_table[4] =
{
    /* [0..3] */ 32, 64, 96, 128
    /* 1., 2., 3., 4. */
}; /* LSB: 2^-5 OFF: 0 MIN/MAX: 0 .. 7.96875 */
const U8 Sa4_c1_table_z_table[4] =
{
    /* [0..3] */ 128, 128, 0, 0
    /* 1., 1., 0., 0. */
}; /* LSB: 2^-7 OFF: 0 MIN/MAX: 0 .. 1.9921875 */
const U8 b12_x_table[4] =
{
    /* [0..3] */ 32, 64, 96, 128
    /* 1., 2., 3., 4. */
}; /* LSB: 2^-5 OFF: 0 MIN/MAX: 0 .. 7.96875 */
const U8 Sa4_b12_table_z_table[4] =
{
    /* [0..3] */ 128, 128, 0, 0
    /* 1., 1., 0., 0. */
}; /* LSB: 2^-7 OFF: 0 MIN/MAX: 0 .. 1.9921875 */
const MAP_Tab1DS16I80T28672_a Sa5_1_2_shift_map =
{
    8 /*
        number of values in x-axis vector
        comp: N
        LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */
    (const U8 *)s12_x_table /*
        pointer to x-axis vector
        comp: x_table
        pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */
    (const U8 *)Sa5_1_2_shift_z_table /*
        pointer to y-axis vector
        comp: z_table
        pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */
}; /* structure type */
const MAP_Tab1DS48I80T28672_a Sa5_2_1_shift_map =

```

```

{
  8 /*
     number of values in x-axis vector
     comp: N
     LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
  (const U8 *)s21_x_table /*
     pointer to x-axis vector
     comp: x_table
     pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */,
  (const U8 *)Sa5_2_1_shift_z_table /*
     pointer to y-axis vector
     comp: z_table
     pointer to LSB: 2^-2 OFF: 0 MIN/MAX: 0 .. 63.75 */
}; /* structure type */
const MAP_Tab1DS48I80T28672_a Sa5_2_3_shift_map =
{
  8 /*
     number of values in x-axis vector
     comp: N
     LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
  (const U8 *)s23_x_table /*
     pointer to x-axis vector
     comp: x_table
     pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */,
  (const U8 *)Sa5_2_3_shift_z_table /*
     pointer to y-axis vector
     comp: z_table
     pointer to LSB: 2^-2 OFF: 0 MIN/MAX: 0 .. 63.75 */
}; /* structure type */
const MAP_Tab1DS48I80T28676_a Sa5_3_2_shift_map =
{
  8 /*
     number of values in x-axis vector
     comp: N
     LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
  (const S16 *)s32_x_table /*
     pointer to x-axis vector
     comp: x_table
     pointer to LSB: 2^-8 OFF: 0 MIN/MAX: -128 .. 127.9960938 */,
  (const U8 *)Sa5_3_2_shift_z_table /*
     pointer to y-axis vector
     comp: z_table
     pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */
}; /* structure type */
const MAP_Tab1DS16I80T28672_a Sa5_3_4_shift_map =
{
  8 /*
     number of values in x-axis vector
     comp: N
     LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
  (const U8 *)s34_x_table /*
     pointer to x-axis vector
     comp: x_table
     pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */,
  (const U8 *)Sa5_3_4_shift_z_table /*
     pointer to y-axis vector
     comp: z_table
     pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */
}; /* structure type */
const MAP_Tab1DS16I80T28672_a Sa5_4_3shift_map =
{
  8 /*
     number of values in x-axis vector
     comp: N
     LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
  (const U8 *)s43_x_table /*
     pointer to x-axis vector
     comp: x_table
     pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */,
  (const U8 *)Sa5_4_3shift_z_table /*
     pointer to y-axis vector
     comp: z_table

```

```

    pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */
};
/* structure type */
const MAP_Tab1DS16I80T28672_a Sa4_b12_table_map =
{
    4 /*
    number of values in x-axis vector
    comp: N
    LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
    (const U8 *)b12_x_table /*
    pointer to x-axis vector
    comp: x_table
    pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */,
    (const U8 *)Sa4_b12_table_z_table /*
    pointer to y-axis vector
    comp: z_table
    pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */
};
/* structure type */
const MAP_Tab1DS16I80T28672_a Sa4_c1_table_map =
{
    4 /*
    number of values in x-axis vector
    comp: N
    LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
    (const U8 *)c1_x_table /*
    pointer to x-axis vector
    comp: x_table
    pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */,
    (const U8 *)Sa4_c1_table_z_table /*
    pointer to y-axis vector
    comp: z_table
    pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */
};
/* structure type */
const MAP_Tab1DS16I80T28672_a Sa4_c2_table_map =
{
    4 /*
    number of values in x-axis vector
    comp: N
    LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
    (const U8 *)c2_x_table /*
    pointer to x-axis vector
    comp: x_table
    pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */,
    (const U8 *)Sa4_c2_table_z_table /*
    pointer to y-axis vector
    comp: z_table
    pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */
};
/* structure type */
const MAP_Tab1DS16I80T28672_a Sa4_c3_table_map =
{
    4 /*
    number of values in x-axis vector
    comp: N
    LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
    (const U8 *)c3_x_table /*
    pointer to x-axis vector
    comp: x_table
    pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */,
    (const U8 *)Sa4_c3_table_z_table /*
    pointer to y-axis vector
    comp: z_table
    pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */
};
/* structure type */
const MAP_Tab1DS48I80T28672_a Sa4_c4_table_map =
{
    4 /*
    number of values in x-axis vector
    comp: N
    LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 255 */,
    (const U8 *)c4_x_table /*
    pointer to x-axis vector
    comp: x_table
    pointer to LSB: 2^-1 OFF: 0 MIN/MAX: 0 .. 127.5 */,

```

```

    (const U8 *)Sa4_c4_table_z_table      /*
        pointer to y-axis vector
        comp: z_table
        pointer to LSB: 2^-2 OFF: 0 MIN/MAX: 0 .. 63.75 */
};    /* structure type */

/*****\
GLOBAL: global variables (RAM)
\*****/
U8    Sa5_Switch_threshold = 128 /* 0.5 */; /* LSB: 2^-8 OFF: 0 MIN/MAX: 0 ..
0.99609375 */
U8    Sa4_c2_table; /* LSB: 2^-7 OFF: 0 MIN/MAX: 0 .. 1.9921875 */

/*****\
defaultClass_sfStaticGlobal: Default storage class for global static variables
\*****/
static S16    Ca2_V; /* LSB: 2^-6 OFF: 0 MIN/MAX: -512 .. 511.984375 */
static S8     daves_to_gear = 1; /* LSB: 2^0 OFF: 0 MIN/MAX: -128 .. 127 */

/*****\
defaultClass_slStaticGlobal: Default storage class for global static variables
\*****/
static S16    AUX_a_Sa5_Product;
static S16    AUX_a_Sa5_Product1;
static S16    AUX_a_Sa5_Product2;
static S16    AUX_a_Sa5_Product4;
static S16    AUX_a_Sa5_Product5;

/*****\
defaultClass_slGlobal: Default storage class for global variables.
\*****/
BFa9_tp      BFa9 =
{
    0 /*
        comp: Ca24_shifting
        boolean type */,
    0 /*
        comp: Ca23_shift_pending
        boolean type */,
    0 /*
        comp: Ca17_shift_pending
        boolean type */,
    0 /*
        comp: Ca21_shifting3
        boolean type */,
    0 /*
        comp: Ca20_shift_pending3
        boolean type */,
    0 /*
        comp: Ca15_shifting2
        boolean type */,
    0 /*
        comp: Ca18_shifting
        boolean type */,
    0 /*
        comp: Ca9_shifting
        boolean type */,
    0 /*
        comp: Ca14_shift_pending2
        boolean type */,
    0 /*
        comp: Ca11_shift_pending
        boolean type */,
    0 /*
        comp: Ca12_shifting
        boolean type */,
    0 /*
        comp: Ca8_shift_pending
        boolean type */,
    0 /*
        comp: Ca2_shift_schedule_ns
        LSB: 2^0 OFF: 0 MIN/MAX: 0 .. 15 */,

```

```

0 /*
  comp: Ca2_shift_schedule
  boolean type */
}; /* structure type */

/*****\
  Model step function of subsystem(s): trans_v1_1/trans_slow/trans_slow_torques
\*****/
void calc_trans_slow_torques(
  S16 Sa3____c /* LSB: 2^-7 OFF: 0 MIN/MAX: -256 .. 255.9921875 *//,
  S16 Sa3____b /* LSB: 2^-8 OFF: 0 MIN/MAX: -128 .. 127.9960938 *//,
  Bool8 Sa3____a /* boolean type *//,
  S8 Sa3____ /* LSB: 2^-3 OFF: 0 MIN/MAX: -16 .. 15.875 *//,
  S16 * Sa3_pc1_ /* pointer to LSB: 2^-4 OFF: 0 MIN/MAX: -2048 .. 2047.9375
*/,
  S16 * Sa3_pc2_ /* pointer to LSB: 2^-4 OFF: 0 MIN/MAX: -2048 .. 2047.9375
*/,
  S16 * Sa3_pc3_ /* pointer to LSB: 2^-4 OFF: 0 MIN/MAX: -2048 .. 2047.9375
*/,
  S16 * Sa3_pc4_ /* pointer to LSB: 2^-4 OFF: 0 MIN/MAX: -2048 .. 2047.9375
*/,
  S16 * Sa3_pbl2_ /* pointer to LSB: 2^-4 OFF: 0 MIN/MAX: -2048 .. 2047.9375
*/)
{
  /* defaultClass_slLocal: Default storage class for local variables */
  S16 Sa4_Gain5; /* LSB: 2^-11 OFF: 0 MIN/MAX: -16 .. 15.99951172 */

  /* # combined # Product:
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/Product

  # combined # 1D-TableLookup:
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/1_2_shift
t */
  AUX_a_Sa5_Product = (S16)((((U32)(Tab1DS16I80T28672_a(
  (const MAP_Tab1DS16I80T28672_a *)(&(Sa5_1_2_shift_map))),
  (U8)(S8)(Sa3____c >> 6)))) * ((U32)8192 /* 1. */) >> 7);

  /* # combined # Product:
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/Product1

  # combined # 1D-TableLookup:
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/2_1_shift
t */
  AUX_a_Sa5_Product1 = (S16)((((U32)(Tab1DS48I80T28672_a(
  (const MAP_Tab1DS48I80T28672_a *)(&(Sa5_2_1_shift_map))),
  (U8)(S8)(Sa3____c >> 6)))) * ((U32)8192 /* 1. */) >> 6);

  /* # combined # Product:
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/Product2

  # combined # 1D-TableLookup:
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/2_3_shift
t */
  AUX_a_Sa5_Product2 = (S16)((((U32)(Tab1DS48I80T28672_a(
  (const MAP_Tab1DS48I80T28672_a *)(&(Sa5_2_3_shift_map))),
  (U8)(S8)(Sa3____c >> 6)))) * ((U32)8192 /* 1. */) >> 7);

  /* # combined # Product:
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/Product4

  # combined # 1D-TableLookup:
trans_v1_1/trans_slow/trans_slow_torques/shift_scheduler/3_4_shift
t */
  AUX_a_Sa5_Product4 = (S16)((((U32)(Tab1DS16I80T28672_a(
  (const MAP_Tab1DS16I80T28672_a *)(&(Sa5_3_4_shift_map))),
  (U8)(S8)(Sa3____c >> 6)))) * ((U32)8192 /* 1. */) >> 7);

```

```

/* # combined # Product:
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/Product5
# combined # 1D-TableLookup:
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/4_3shift
*/
AUX_a_Sa5_Product5 = (S16)((((U32)(Tab1DS16I80T28672_a(
(const MAP_Tab1DS16I80T28672_a *)(&(Sa5_4_3shift_map)),
(U8)(S8)(Sa3_____c >> 6)))) * ((U32)8192 /* 1. */) >> 7);
/* Stateflow: trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule
*/

/* # combined # Gain:
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/m_s_to_km_h */
Ca2_V = (S16)((((S32)Sa3_____b) * 29491) >> 15);
/*
Begin execution of chart
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sche
le */
switch ( BFa9.Ca2_shift_schedule_ns ) {
case Ca3_first_gear_id: {
/*
Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.first_gear */

/* furry animals */
if ( Ca2_V > ((S16)(AUX_a_Sa5_Product >> 1)) ) {
/*
State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.first_gear to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.transition12.shift_pending */
BFa9.Ca2_shift_schedule_ns = Ca7_transition12_id;
BFa9.Ca8_shift_pending = 1;
ctr = 0;
daves_to_gear = 1;
}
/*
End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
hedule.first_gear */
break;
}

case Ca4_fourth_gear_id: {
/*
Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.fourth_gear */
if ( Ca2_V <= AUX_a_Sa5_Product5 ) {
/*
State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.fourth_gear to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.transition43.shift_pending */
BFa9.Ca2_shift_schedule_ns = Ca22_transition43_id;
BFa9.Ca23_shift_pending = 1;
ctr = 0;
daves_to_gear = 4;
}
/*
End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
hedule.fourth_gear */
break;
}

case Ca5_second_gear_id: {
/*

```

```

Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.second_gear */
if ( Ca2_V > AUX_a_Sa5_Product2 ) {
/*
State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.second_gear to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.transition23.shift_pending2 */
BFa9.Ca2_shift_schedule_ns = Ca13_transition23_id;
BFa9.Ca14_shift_pending2 = 1;
ctr = 0;
daves_to_gear = 2;
} else {
if ( Ca2_V <= ((S16)(AUX_a_Sa5_Product1 >> 3)) ) {
/*
State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.second_gear to
trans_vl_1/trans_slow/trans_slow_torques/shift_sched
uler/shift_schedule.transition21.shift_pending */
BFa9.Ca2_shift_schedule_ns = Ca10_transition21_id;
BFa9.Ca11_shift_pending = 1;
ctr = 0;
daves_to_gear = 2;
}
}
/*
End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
hedule.second_gear */
break;
}

case Ca6_third_gear_id: {
/*
Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.third_gear */
if ( Ca2_V > AUX_a_Sa5_Product4 ) {
/*
State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.third_gear to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.transition34.shift_pending3 */
BFa9.Ca2_shift_schedule_ns = Ca19_transition34_id;
BFa9.Ca20_shift_pending3 = 1;
ctr = 0;
daves_to_gear = 3;
} else {
/*
# combined # # combined # Product:
trans_vl_1/trans_slow/trans_slow_torques/shift_sch
eduler/Product6

# combined # 1D-TableLookup:
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler
/3_2_shift */
if ( Ca2_V <= ((S16)(((S16)((U32)(Tab1DS48I80T28676_a(
(const MAP_Tab1DS48I80T28676_a *)(&Sa5_3_2_shift_map)),
(S16)(Sa3_____c << 1)))) * ((U32)8192 /* 1. */) >> 7)) >> 1)) ) {
/*
State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.third_gear to
trans_vl_1/trans_slow/trans_slow_torques/shift_schedu
ler/shift_schedule.transition32.shift_pending */
BFa9.Ca2_shift_schedule_ns = Ca16_transition32_id;
BFa9.Ca17_shift_pending = 1;

```

```

        ctr = 0;
        daves_to_gear = 3;
    }
}
/*
    End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
    hedule.third_gear */
    break;
}

case Ca7_transition12_id: {
    /*
        Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
    schedule.transition12 */
        if ( Ca2_V <= ((S16)(AUX_a_Sa5_Product1 >> 3)) ) {
            /*
                State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
    schedule.transition12 to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
    shift_schedule.first_gear */
                if ( BFa9.Ca8_shift_pending ) {
                    BFa9.Ca8_shift_pending = 0;
                } else {
                    if ( BFa9.Ca9_shifting ) {
                        BFa9.Ca9_shifting = 0;
                    }
                }
                BFa9.Ca2_shift_schedule_ns = Ca3_first_gear_id;
                daves_to_gear = 1;
            } else {
                if ( BFa9.Ca8_shift_pending ) {
                    /*
                        Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
    shift_schedule.transition12.shift_pending */
                        if ( ctr > Ca2_DELAY ) {
                            /*
                                State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
    shift_schedule.transition12.shift_pending to
trans_vl_1/trans_slow/trans_slow_t
    orques/shift_scheduler/shift_schedule.transition12.shifting */
                                BFa9.Ca8_shift_pending = 0;
                                BFa9.Ca9_shifting = 1;
                                daves_to_gear = 2;
                            } else {
                                ctr = ctr + 1;
                            }
                        }
                    /*
                        End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
    shift_schedule.transition12.shift_pending */
                } else {
                    if ( BFa9.Ca9_shifting ) {
                        /*
                            Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
    er/shift_schedule.transition12.shifting */
                            if ( Sa3___ == 16 /* 2. */ ) {
                                /*
                                    State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
    er/shift_schedule.transition12.shifting to
trans_vl_1/trans_slow/trans_slow_
    torques/shift_scheduler/shift_schedule.second_gear */
                                    if ( BFa9.Ca8_shift_pending ) {
                                        BFa9.Ca8_shift_pending = 0;
                                    } else {
                                        if ( BFa9.Ca9_shifting ) {

```

```

        BFa9.Ca9_shifting = 0;
    }
}
BFa9.Ca2_shift_schedule_ns = Ca5_second_gear_id;
daves_to_gear = 2;
}
/*
    End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler
    /shift_schedule.transition12.shifting */
}
}
}
/*
    End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
    hedule.transition12 */
break;
}

case Ca10_transition21_id: {
/*
    Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
    schedule.transition21 */
    if ( Ca2_V > ((S16)(AUX_a_Sa5_Product >> 1)) ) {
/*
        State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
    schedule.transition21 to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
    shift_schedule.second_gear */
        if ( BFa9.Call_shift_pending ) {
            BFa9.Call_shift_pending = 0;
        } else {
            if ( BFa9.Ca12_shifting ) {
                BFa9.Ca12_shifting = 0;
            }
        }
        BFa9.Ca2_shift_schedule_ns = Ca5_second_gear_id;
        daves_to_gear = 2;
    } else {
        if ( BFa9.Call_shift_pending ) {
/*
            Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
    shift_schedule.transition21.shift_pending */
            if ( ctr > Ca2_DELAY ) {
/*
                State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
    shift_schedule.transition21.shift_pending to
trans_vl_1/trans_slow/trans_slow_t
    orques/shift_scheduler/shift_schedule.transition21.shifting */
                BFa9.Call_shift_pending = 0;
                BFa9.Ca12_shifting = 1;
                daves_to_gear = 1;
            } else {
                ctr = ctr + 1;
            }
        }
/*
            End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
    shift_schedule.transition21.shift_pending */
    } else {
        if ( BFa9.Ca12_shifting ) {
/*
            Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
    er/shift_schedule.transition21.shifting */
            if ( Sa3___ == 8 /* 1. */ ) {
/*

```

```

        State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
er/shift_schedule.transition21.shifting to
trans_vl_1/trans_slow/trans_slow_
torques/shift_scheduler/shift_schedule.first_gear */
        if ( BFa9.Call_shift_pending ) {
            BFa9.Call_shift_pending = 0;
        } else {
            if ( BFa9.Ca12_shifting ) {
                BFa9.Ca12_shifting = 0;
            }
        }
        BFa9.Ca2_shift_schedule_ns = Ca3_first_gear_id;
        daves_to_gear = 1;
    }
    /*
        End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler
/shift_schedule.transition21.shifting */
    }
}
}
/*
    End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
hedule.transition21 */
    break;
}

    case Ca13_transition23_id: {
        /*
            Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.transition23 */
        if ( Ca2_V <= AUX_a_Sa5_Product2 ) {
            /*
                State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.transition23 to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.second_gear */
                if ( BFa9.Ca14_shift_pending2 ) {
                    BFa9.Ca14_shift_pending2 = 0;
                } else {
                    if ( BFa9.Ca15_shifting2 ) {
                        BFa9.Ca15_shifting2 = 0;
                    }
                }
                BFa9.Ca2_shift_schedule_ns = Ca5_second_gear_id;
                daves_to_gear = 2;
            } else {
                if ( BFa9.Ca14_shift_pending2 ) {
                    /*
                        Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.transition23.shift_pending2 */
                        if ( ctr > Ca2_DELAY ) {
                            /*
                                State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.transition23.shift_pending2 to
trans_vl_1/trans_slow/trans_slow_
torques/shift_scheduler/shift_schedule.transition23.shifting2 */
                                BFa9.Ca14_shift_pending2 = 0;
                                BFa9.Ca15_shifting2 = 1;
                                daves_to_gear = 3;
                            } else {
                                ctr = ctr + 1;
                            }
                        }
                    /*
                        End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/

```

```

        shift_schedule.transition23.shift_pending2 */
    } else {
        if ( BFa9.Ca15_shifting2 ) {
            /*
            Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
er/shift_schedule.transition23.shifting2 */
            if ( Sa3___ == 24 /* 3. */ ) {
                /*
                State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
er/shift_schedule.transition23.shifting2 to
trans_vl_1/trans_slow/trans_slow
_torques/shift_scheduler/shift_schedule.third_gear */
                if ( BFa9.Ca14_shift_pending2 ) {
                    BFa9.Ca14_shift_pending2 = 0;
                } else {
                    if ( BFa9.Ca15_shifting2 ) {
                        BFa9.Ca15_shifting2 = 0;
                    }
                }
                BFa9.Ca2_shift_schedule_ns = Ca6_third_gear_id;
                daves_to_gear = 3;
            }
            /*
            End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler
/shift_schedule.transition23.shifting2 */
        }
    }
}
/*
End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
hedule.transition23 */
break;
}

case Ca16_transition32_id: {
    /*
    Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.transition32 */
    if ( Ca2_V > AUX_a_Sa5_Product2 ) {
        /*
        State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.transition32 to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.third_gear */
        if ( BFa9.Ca17_shift_pending ) {
            BFa9.Ca17_shift_pending = 0;
        } else {
            if ( BFa9.Ca18_shifting ) {
                BFa9.Ca18_shifting = 0;
            }
        }
        BFa9.Ca2_shift_schedule_ns = Ca6_third_gear_id;
        daves_to_gear = 3;
    } else {
        if ( BFa9.Ca17_shift_pending ) {
            /*
            Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.transition32.shift_pending */
            if ( ctr > Ca2_DELAY ) {
                /*
                State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
shift_schedule.transition32.shift_pending to
trans_vl_1/trans_slow/trans_slow_t
orques/shift_scheduler/shift_schedule.transition32.shifting */

```

```

        BFa9.Ca17_shift_pending = 0;
        BFa9.Ca18_shifting = 1;
        daves_to_gear = 2;
    } else {
        ctr = ctr + 1;
    }
    /*
    End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
    shift_schedule.transition32.shift_pending */
    } else {
        if ( BFa9.Ca18_shifting ) {
            /*
            Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
            er/shift_schedule.transition32.shifting */
            if ( Sa3___ == 16 /* 2. */ ) {
                /*
            State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
            er/shift_schedule.transition32.shifting to
trans_vl_1/trans_slow/trans_slow_
            torques/shift_scheduler/shift_schedule.second_gear */
                if ( BFa9.Ca17_shift_pending ) {
                    BFa9.Ca17_shift_pending = 0;
                } else {
                    if ( BFa9.Ca18_shifting ) {
                        BFa9.Ca18_shifting = 0;
                    }
                }
                BFa9.Ca2_shift_schedule_ns = Ca5_second_gear_id;
                daves_to_gear = 2;
            }
            /*
            End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler
            /shift_schedule.transition32.shifting */
        }
    }
    /*
    End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
    hedule.transition32 */
    break;
}

case Ca19_transition34_id: {
    /*
    Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
    schedule.transition34 */
    if ( Ca2_V <= AUX_a_Sa5_Product5 ) {
        /*
        State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
        schedule.transition34 to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
        shift_schedule.third_gear */
        if ( BFa9.Ca20_shift_pending3 ) {
            BFa9.Ca20_shift_pending3 = 0;
        } else {
            if ( BFa9.Ca21_shifting3 ) {
                BFa9.Ca21_shifting3 = 0;
            }
        }
        BFa9.Ca2_shift_schedule_ns = Ca6_third_gear_id;
        daves_to_gear = 3;
    } else {
        if ( BFa9.Ca20_shift_pending3 ) {
            /*

```

```

Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
  shift_schedule.transition34.shift_pending3 */
  if ( ctr > Ca2_DELAY ) {
    /*
      State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
  shift_schedule.transition34.shift_pending3 to
trans_vl_1/trans_slow/trans_slow_
  torques/shift_scheduler/shift_schedule.transition34.shifting3 */
    BFa9.Ca20_shift_pending3 = 0;
    BFa9.Ca21_shifting3 = 1;
    daves_to_gear = 4;
  } else {
    ctr = ctr + 1;
  }
  /*
End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
  shift_schedule.transition34.shift_pending3 */
  } else {
    if ( BFa9.Ca21_shifting3 ) {
      /*
Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
er/shift_schedule.transition34.shifting3 */
      if ( Sa3___ == 32 /* 4. */ ) {
        /*
State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
er/shift_schedule.transition34.shifting3 to
trans_vl_1/trans_slow/trans_slow_
  _torques/shift_scheduler/shift_schedule.fourth_gear */
        if ( BFa9.Ca20_shift_pending3 ) {
          BFa9.Ca20_shift_pending3 = 0;
        } else {
          if ( BFa9.Ca21_shifting3 ) {
            BFa9.Ca21_shifting3 = 0;
          }
        }
        BFa9.Ca2_shift_schedule_ns = Ca4_fourth_gear_id;
        daves_to_gear = 4;
      }
      /*
End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler
  /shift_schedule.transition34.shifting3 */
    }
  }
  /*
End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
hedule.transition34 */
  break;
}

case Ca22_transition43_id: {
  /*
Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.transition43 */
  if ( Ca2_V > AUX_a_Sa5_Product4 ) {
    /*
State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_
schedule.transition43 to
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
  shift_schedule.fourth_gear */
    if ( BFa9.Ca23_shift_pending ) {
      BFa9.Ca23_shift_pending = 0;
    } else {

```

```

        if ( BFa9.Ca24_shifting ) {
            BFa9.Ca24_shifting = 0;
        }
    }
    BFa9.Ca2_shift_schedule_ns = Ca4_fourth_gear_id;
    daves_to_gear = 4;
} else {
    if ( BFa9.Ca23_shift_pending ) {
        /*
        Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
        shift_schedule.transition43.shift_pending */
        if ( ctr > Ca2_DELAY ) {
            /*
            State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
            shift_schedule.transition43.shift_pending to
trans_vl_1/trans_slow/trans_slow_t
            orques/shift_scheduler/shift_schedule.transition43.shifting */
            BFa9.Ca23_shift_pending = 0;
            BFa9.Ca24_shifting = 1;
            daves_to_gear = 3;
        } else {
            ctr = ctr + 1;
        }
        /*
        End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/
        shift_schedule.transition43.shift_pending */
    } else {
        if ( BFa9.Ca24_shifting ) {
            /*
            Begin execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
            er/shift_schedule.transition43.shifting */
            if ( Sa3___ == 24 /* 3. */ ) {
                /*
                State transition from
trans_vl_1/trans_slow/trans_slow_torques/shift_schedul
                er/shift_schedule.transition43.shifting to
trans_vl_1/trans_slow/trans_slow_
                torques/shift_scheduler/shift_schedule.third_gear */
                if ( BFa9.Ca23_shift_pending ) {
                    BFa9.Ca23_shift_pending = 0;
                } else {
                    if ( BFa9.Ca24_shifting ) {
                        BFa9.Ca24_shifting = 0;
                    }
                }
                BFa9.Ca2_shift_schedule_ns = Ca6_third_gear_id;
                daves_to_gear = 3;
            }
            /*
            End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler
            /shift_schedule.transition43.shifting */
        }
    }
}
/*
End execution of state
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_sc
hedule.transition43 */
break;
}

default: {
    if ( !(BFa9.Ca2_shift_schedule) ) {
        BFa9.Ca2_shift_schedule = 1;
        BFa9.Ca2_shift_schedule_ns = Ca3_first_gear_id;
        daves_to_gear = 1;
    }
}

```

```

    }
}
/* End execution of chart
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule
*/

/* Gain: trans_vl_1/trans_slow/trans_slow_torques/clutch_pressure/Gain5 */
Sa4_Gain5 = (S16)(((S16)daves_to_gear) << 11);

/* Outport: trans_vl_1/trans_slow/trans_slow_torques/PB12
# combined # Gain: trans_vl_1/trans_slow/trans_slow_torques/clutch_pressure/Gain4
# combined # 1D-TableLookup:
trans_vl_1/trans_slow/trans_slow_torques/clutch_pressure/b12_table
*/
*Sa3_pb12_ = ((S16)(Tab1DS16I80T28672_a(
(const MAP_Tab1DS16I80T28672_a *)(&(Sa4_b12_table_map)),
(U8)(S8)(Sa4_Gain5 >> 6)))) * 125;

/* Outport: trans_vl_1/trans_slow/trans_slow_torques/Pc1
# combined # Gain: trans_vl_1/trans_slow/trans_slow_torques/clutch_pressure/Gain4
# combined # 1D-TableLookup:
trans_vl_1/trans_slow/trans_slow_torques/clutch_pressure/c1_table
*/
*Sa3_pc1_ = ((S16)(Tab1DS16I80T28672_a(
(const MAP_Tab1DS16I80T28672_a *)(&(Sa4_c1_table_map)),
(U8)(S8)(Sa4_Gain5 >> 6)))) * 125;

/* 1D-TableLookup: trans_vl_1/trans_slow/trans_slow_torques/clutch_pressure/c2_table
*/
Sa4_c2_table = Tab1DS16I80T28672_a(
(const MAP_Tab1DS16I80T28672_a *)(&(Sa4_c2_table_map)),
(U8)(S8)(Sa4_Gain5 >> 6));

/* Outport: trans_vl_1/trans_slow/trans_slow_torques/Pc2
# combined # Gain: trans_vl_1/trans_slow/trans_slow_torques/clutch_pressure/Gain1
*/
*Sa3_pc2_ = ((S16)Sa4_c2_table) * 125;

/* Outport: trans_vl_1/trans_slow/trans_slow_torques/Pc3
# combined # Gain: trans_vl_1/trans_slow/trans_slow_torques/clutch_pressure/Gain2
# combined # 1D-TableLookup:
trans_vl_1/trans_slow/trans_slow_torques/clutch_pressure/c3_table
*/
*Sa3_pc3_ = ((S16)(Tab1DS16I80T28672_a(
(const MAP_Tab1DS16I80T28672_a *)(&(Sa4_c3_table_map)),
(U8)(S8)(Sa4_Gain5 >> 6)))) * 125;

/* Outport: trans_vl_1/trans_slow/trans_slow_torques/Pc4
# combined # Gain: trans_vl_1/trans_slow/trans_slow_torques/clutch_pressure/Gain3
# combined # 1D-TableLookup:
trans_vl_1/trans_slow/trans_slow_torques/clutch_pressure/c4_table
*/
*Sa3_pc4_ = (S16)((((U16)((U16)(Tab1DS48I80T28672_a(
(const MAP_Tab1DS48I80T28672_a *)(&(Sa4_c4_table_map)),
(U8)(S8)(Sa4_Gain5 >> 6)))) * 125)) - 7938);
}

#undef _SEPPFILE_C_

```

8.8 APPENDIX H: Floating-Point C Code Generated by RTW-EC for Transmission Model

```
#define IN_SF_MACHINE_SOURCE          1
#include "trans_2_rtw.h"

#include "shift_schedule.h"
#include "trans_slow_ctl.h"

int8_T _sfEvent_trans_2=0;
void trans_2_initializer( void )
{
    /* Initialize machine's broadcast event variable */
    _sfEvent_trans_2 = CALL_EVENT;
}

void trans_2_terminator(void)
{
}
extern void trans_2_terminator(void);
void trans_2_machine_global_terminator(void)
{
    trans_2_terminator();
    return;
}
extern void trans_2_initializer(void);
void trans_2_machine_global_initializer(void)
{
    trans_2_initializer();
    return;
}

/*
 * Auto-generated file: trans_st.c
 *
 * Real-Time Workshop code generation for Simulink system "<S2>/trans_slow_torques"
 *
 * Model : trans_2
 * Model Version : 1.15
 * Real-Time Workshop file version : 4.0 $Date: 2000/09/19 19:45:27 $
 * Real-Time Workshop file generated on : Thu Jan 24 23:22:52 2002
 * TLC version : 4.0 (Aug 21 2000)
 * C source code generated on : Thu Jan 24 23:22:54 2002
 *
 * Relevant TLC Options:
 *   InlineParameters = 0
 *   RollThreshold = 5
 *   CodeFormat = Embedded-C
 *
 * Simulink model settings:
 *   Solver : FixedStep
 *   StartTime : 0.0 s
 *   StopTime : 60.0 s
 *   FixedStep : 0.02 s
 *
 * Note that the functions contained in this file are part of a Simulink
 * model, and are not self-contained algorithms. Related data and functions
 * are found in these locations.
 *
 * Internal data definitions: trans_2.h
 * Internal data declarations: trans_2_prm.h
 * Imported data and functions: trans_2.h
 * Exported data and functions: trans_2_export.h
 * Initialization code: trans_2_reg.h
 * Model (step) code: trans_2.c
 */
#include "trans_st.h"
```

```

/* Start of Functions for System "<S2>/trans_slow_torques"*/

/* Initial conditions for function-call system: <S2>/trans_slow_torques */
void trans_slow_torques_Initialize(void)
{
    /* Stateflow Chart Initializer Code: <S6>/shift_schedule */
    {
        SFshift_scheduleInstanceStruct *chartInstance = (SFshift_scheduleInstanceStruct
*)trans_2_DWork.s7_SFunction_PWORK.ChartInstance;

        trans_2_B.s7_SFunction_o2 = (int8_T)1;
    }
}

/* Output and update for function-call system: <S2>/trans_slow_torques */
void trans_slow_torques(void)
{
    /* Gain Block: <S6>/m_s_to_km_h */
    trans_2_B.s6_vss_kmh = vss * (trans_2_P.s6_m_s_to_km_h_Gain);

    /* Lookup Block: <S6>/1_2_shift */
    trans_2_B.temp9 = rt_Lookup32(&trans_2_P.s6_1_2_shift_XData[0],
    8,
    tps,
    &trans_2_P.s6_1_2_shift_YData[0]);

    /* Switch Block: <S6>/Switch */
    if (trans_2_U.root_p_e_switch) {
        trans_2_B.temp8 = (trans_2_P.s6_power_Value);
    } else {
        trans_2_B.temp8 = (trans_2_P.s6_economy_Value);
    }

    /* Product Block: <S6>/Product */
    trans_2_B.s6_Product = trans_2_B.temp9 *
    trans_2_B.temp8;

    /* Lookup Block: <S6>/2_1_shift */
    trans_2_B.temp9 = rt_Lookup32(&trans_2_P.s6_2_1_shift_XData[0],
    8,
    tps,
    &trans_2_P.s6_2_1_shift_YData[0]);

    /* Product Block: <S6>/Product1 */
    trans_2_B.s6_Product1 = trans_2_B.temp9 *
    trans_2_B.temp8;

    /* Lookup Block: <S6>/2_3_shift */
    trans_2_B.temp9 = rt_Lookup32(&trans_2_P.s6_2_3_shift_XData[0],
    8,
    tps,
    &trans_2_P.s6_2_3_shift_YData[0]);

    /* Product Block: <S6>/Product2 */
    trans_2_B.s6_Product2 = trans_2_B.temp9 *
    trans_2_B.temp8;

    /* Lookup Block: <S6>/3_2_shift */
    trans_2_B.temp9 = rt_Lookup32(&trans_2_P.s6_3_2_shift_XData[0],
    8,
    tps,
    &trans_2_P.s6_3_2_shift_YData[0]);

    /* Product Block: <S6>/Product3 */
    trans_2_B.s6_Product3 = trans_2_B.temp9 *
    trans_2_B.temp8;

    /* Lookup Block: <S6>/3_4_shift */
    trans_2_B.temp9 = rt_Lookup32(&trans_2_P.s6_3_4_shift_XData[0],
    8,
    tps,

```

```

    &trans_2_P.s6_3_4_shift_YData[0]);

/* Product Block: <S6>/Product4 */
trans_2_B.s6_Product4 = trans_2_B.temp9 *
    trans_2_B.temp8;

/* Lookup Block: <S6>/4_3shift */
trans_2_B.temp9 = rt_Lookup32(&trans_2_P.s6_4_3shift_XData[0],
    8,
    tps,
    &trans_2_P.s6_4_3shift_YData[0]);

/* Product Block: <S6>/Product5 */
trans_2_B.s6_Product5 = trans_2_B.temp9 *
    trans_2_B.temp8;

/* Stateflow Chart Code: <S6>/shift_schedule */
{
    SFshift_scheduleInstanceStruct *chartInstance = (SFshift_scheduleInstanceStruct
*)trans_2_DWork.s7_SFunction_PWORK.ChartInstance;

#define IN_NO_ACTIVE_CHILD                (0)
#define IN_first_gear_in_shift_schedule  1
#define IN_fourth_gear_in_shift_schedule  2
#define IN_second_gear_in_shift_schedule  3
#define IN_third_gear_in_shift_schedule   4
#define IN_transition12_in_shift_schedule  5
#define IN_transition21_in_shift_schedule  6
#define IN_transition23_in_shift_schedule  7
#define IN_transition32_in_shift_schedule  8
#define IN_transition34_in_shift_schedule  9
#define IN_transition43_in_shift_schedule 10
#define IN_shift_pending_in_transition12  1
#define IN_shifting_in_transition12       2
#define IN_shift_pending_in_transition21  1
#define IN_shifting_in_transition21       2
#define IN_shift_pending2_in_transition23  1
#define IN_shifting2_in_transition23      2
#define IN_shift_pending_in_transition32  1
#define IN_shifting_in_transition32       2
#define IN_shift_pending3_in_transition34  1
#define IN_shifting3_in_transition34      2
#define IN_shift_pending_in_transition43  1
#define IN_shifting_in_transition43       2
#define DELAY_in_shift_schedule           1

    {
        int previousEvent;
        previousEvent = _sfEvent_trans_2_;
        /* Call this chart */
        _sfEvent_trans_2_ = CALL_EVENT;
        {
            /* During Start
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule */
            if(chartInstance->State.is_active_shift_schedule==0) {
                /* Entry Atomic Start
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule */
                chartInstance->State.is_active_shift_schedule=1;
                /* Entry Internal Start
trans_vl_1/trans_slow/trans_slow_torques/shift_scheduler/shift_schedule */
                /* Entry Atomic Start first_gear */
                chartInstance->State.is_shift_schedule=IN_first_gear_in_shift_schedule;
                trans_2_B.s7_SFunction_o2 = 1;
            } else {
                switch(chartInstance->State.is_shift_schedule) {
                    case IN_NO_ACTIVE_CHILD:
                        break;
                    case IN_first_gear_in_shift_schedule:
                        /* During Start first_gear */
                        if(trans_2_B.s6_vss_kmh > trans_2_B.s6_Product) {
                            /* Exit Atomic Start first_gear */
                            /* Entry Atomic Start transition12 */

```

```

        chartInstance->State.is_shift_schedule=IN_transition12_in_shift_schedule;
        /* Entry Atomic Start shift_pending */
        chartInstance-
>State.is_transition12_in_shift_schedule=IN_shift_pending_in_transition12;
        chartInstance->LocalData.ctr = 0;
        trans_2_B.s7_SFunction_o2 = 1;
    }
    break;
case IN_fourth_gear_in_shift_schedule:
    /* During Start fourth_gear */
    if(trans_2_B.s6_vss_kmh <= trans_2_B.s6_Product5) {
        /* Exit Atomic Start fourth_gear */
        /* Entry Atomic Start transition43 */
        chartInstance->State.is_shift_schedule=IN_transition43_in_shift_schedule;
        /* Entry Atomic Start shift_pending */
        chartInstance-
>State.is_transition43_in_shift_schedule=IN_shift_pending_in_transition43;
        chartInstance->LocalData.ctr = 0;
        trans_2_B.s7_SFunction_o2 = 4;
    }
    break;
case IN_second_gear_in_shift_schedule:
    /* During Start second_gear */
    if(trans_2_B.s6_vss_kmh > trans_2_B.s6_Product2) {
        /* Exit Atomic Start second_gear */
        /* Entry Atomic Start transition23 */
        chartInstance->State.is_shift_schedule=IN_transition23_in_shift_schedule;
        /* Entry Atomic Start shift_pending2 */
        chartInstance-
>State.is_transition23_in_shift_schedule=IN_shift_pending2_in_transition23;
        chartInstance->LocalData.ctr = 0;
        trans_2_B.s7_SFunction_o2 = 2;
    } else if(trans_2_B.s6_vss_kmh <= trans_2_B.s6_Product1) {
        /* Exit Atomic Start second_gear */
        /* Entry Atomic Start transition21 */
        chartInstance->State.is_shift_schedule=IN_transition21_in_shift_schedule;
        /* Entry Atomic Start shift_pending */
        chartInstance-
>State.is_transition21_in_shift_schedule=IN_shift_pending_in_transition21;
        chartInstance->LocalData.ctr = 0;
        trans_2_B.s7_SFunction_o2 = 2;
    }
    break;
case IN_third_gear_in_shift_schedule:
    /* During Start third_gear */
    if(trans_2_B.s6_vss_kmh > trans_2_B.s6_Product4) {
        /* Exit Atomic Start third_gear */
        /* Entry Atomic Start transition34 */
        chartInstance->State.is_shift_schedule=IN_transition34_in_shift_schedule;
        /* Entry Atomic Start shift_pending3 */
        chartInstance-
>State.is_transition34_in_shift_schedule=IN_shift_pending3_in_transition34;
        chartInstance->LocalData.ctr = 0;
        trans_2_B.s7_SFunction_o2 = 3;
    } else if(trans_2_B.s6_vss_kmh <= trans_2_B.s6_Product3) {
        /* Exit Atomic Start third_gear */
        /* Entry Atomic Start transition32 */
        chartInstance->State.is_shift_schedule=IN_transition32_in_shift_schedule;
        /* Entry Atomic Start shift_pending */
        chartInstance-
>State.is_transition32_in_shift_schedule=IN_shift_pending_in_transition32;
        chartInstance->LocalData.ctr = 0;
        trans_2_B.s7_SFunction_o2 = 3;
    }
    break;
case IN_transition12_in_shift_schedule:
    /* During Start transition12 */
    if(trans_2_B.s6_vss_kmh <= trans_2_B.s6_Product1) {
        /* Exit Atomic Start shift_pending */
        chartInstance->State.is_transition12_in_shift_schedule=IN_NO_ACTIVE_CHILD;
        /* Exit Atomic Start transition12 */
        /* Entry Atomic Start first_gear */

```

```

    chartInstance->State.is_shift_schedule=IN_first_gear_in_shift_schedule;
    trans_2_B.s7_SFunction_o2 = 1;
  } else {
    switch(chartInstance->State.is_transition12_in_shift_schedule) {
      case IN_NO_ACTIVE_CHILD:
        break;
      case IN_shift_pending_in_transition12:
        /* During Start shift_pending */
        if(chartInstance->LocalData.ctr > DELAY_in_shift_schedule) {
          /* Exit Atomic Start shift_pending */
          /* Entry Atomic Start shifting */
          chartInstance-
>State.is_transition12_in_shift_schedule=IN_shifting_in_transition12;
          trans_2_B.s7_SFunction_o2 = 2;
        } else {
          chartInstance->LocalData.ctr = (int8_T )(chartInstance->LocalData.ctr
+ 1);
        }
        break;
      case IN_shifting_in_transition12:
        /* During Start shifting */
        if(a_gear == 2) {
          /* Exit Atomic Start shifting */
          chartInstance-
>State.is_transition12_in_shift_schedule=IN_NO_ACTIVE_CHILD;
          /* Exit Atomic Start transition12 */
          /* Entry Atomic Start second_gear */
          chartInstance-
>State.is_shift_schedule=IN_second_gear_in_shift_schedule;
          trans_2_B.s7_SFunction_o2 = 2;
        }
        break;
    }
  }
  break;
case IN_transition21_in_shift_schedule:
  /* During Start transition21 */
  if(trans_2_B.s6_vss_kmh > trans_2_B.s6_Product) {
    /* Exit Atomic Start shift_pending */
    chartInstance->State.is_transition21_in_shift_schedule=IN_NO_ACTIVE_CHILD;
    /* Exit Atomic Start transition21 */
    /* Entry Atomic Start second_gear */
    chartInstance->State.is_shift_schedule=IN_second_gear_in_shift_schedule;
    trans_2_B.s7_SFunction_o2 = 2;
  } else {
    switch(chartInstance->State.is_transition21_in_shift_schedule) {
      case IN_NO_ACTIVE_CHILD:
        break;
      case IN_shift_pending_in_transition21:
        /* During Start shift_pending */
        if(chartInstance->LocalData.ctr > DELAY_in_shift_schedule) {
          /* Exit Atomic Start shift_pending */
          /* Entry Atomic Start shifting */
          chartInstance-
>State.is_transition21_in_shift_schedule=IN_shifting_in_transition21;
          trans_2_B.s7_SFunction_o2 = 1;
        } else {
          chartInstance->LocalData.ctr = (int8_T )(chartInstance->LocalData.ctr
+ 1);
        }
        break;
      case IN_shifting_in_transition21:
        /* During Start shifting */
        if(a_gear == 1) {
          /* Exit Atomic Start shifting */
          chartInstance-
>State.is_transition21_in_shift_schedule=IN_NO_ACTIVE_CHILD;
          /* Exit Atomic Start transition21 */
          /* Entry Atomic Start first_gear */
          chartInstance-
>State.is_shift_schedule=IN_first_gear_in_shift_schedule;
          trans_2_B.s7_SFunction_o2 = 1;

```

```

    }
    break;
}
}
break;
case IN_transition23_in_shift_schedule:
/* During Start transition23 */
if(trans_2_B.s6_vss_kmh <= trans_2_B.s6_Product2) {
/* Exit Atomic Start shift_pending2 */
chartInstance->State.is_transition23_in_shift_schedule=IN_NO_ACTIVE_CHILD;
/* Exit Atomic Start transition23 */
/* Entry Atomic Start second_gear */
chartInstance->State.is_shift_schedule=IN_second_gear_in_shift_schedule;
trans_2_B.s7_SFunction_o2 = 2;
} else {
switch(chartInstance->State.is_transition23_in_shift_schedule) {
case IN_NO_ACTIVE_CHILD:
break;
case IN_shift_pending2_in_transition23:
/* During Start shift_pending2 */
if(chartInstance->LocalData.ctr > DELAY_in_shift_schedule) {
/* Exit Atomic Start shift_pending2 */
/* Entry Atomic Start shifting2 */
chartInstance-
>State.is_transition23_in_shift_schedule=IN_shifting2_in_transition23;
trans_2_B.s7_SFunction_o2 = 3;
} else {
chartInstance->LocalData.ctr = (int8_T )(chartInstance->LocalData.ctr
+ 1);
}
break;
case IN_shifting2_in_transition23:
/* During Start shifting2 */
if(a_gear == 3) {
/* Exit Atomic Start shifting2 */
chartInstance-
>State.is_transition23_in_shift_schedule=IN_NO_ACTIVE_CHILD;
/* Exit Atomic Start transition23 */
/* Entry Atomic Start third_gear */
chartInstance-
>State.is_shift_schedule=IN_third_gear_in_shift_schedule;
trans_2_B.s7_SFunction_o2 = 3;
}
break;
}
}
break;
case IN_transition32_in_shift_schedule:
/* During Start transition32 */
if(trans_2_B.s6_vss_kmh > trans_2_B.s6_Product2) {
/* Exit Atomic Start shift_pending */
chartInstance->State.is_transition32_in_shift_schedule=IN_NO_ACTIVE_CHILD;
/* Exit Atomic Start transition32 */
/* Entry Atomic Start third_gear */
chartInstance->State.is_shift_schedule=IN_third_gear_in_shift_schedule;
trans_2_B.s7_SFunction_o2 = 3;
} else {
switch(chartInstance->State.is_transition32_in_shift_schedule) {
case IN_NO_ACTIVE_CHILD:
break;
case IN_shift_pending_in_transition32:
/* During Start shift_pending */
if(chartInstance->LocalData.ctr > DELAY_in_shift_schedule) {
/* Exit Atomic Start shift_pending */
/* Entry Atomic Start shifting */
chartInstance-
>State.is_transition32_in_shift_schedule=IN_shifting_in_transition32;
trans_2_B.s7_SFunction_o2 = 2;
} else {
chartInstance->LocalData.ctr = (int8_T )(chartInstance->LocalData.ctr
+ 1);
}
}
}

```

```

        break;
    case IN_shifting_in_transition32:
        /* During Start shifting */
        if(a_gear == 2) {
            /* Exit Atomic Start shifting */
            chartInstance-
>State.is_transition32_in_shift_schedule=IN_NO_ACTIVE_CHILD;
            /* Exit Atomic Start transition32 */
            /* Entry Atomic Start second_gear */
            chartInstance-
>State.is_shift_schedule=IN_second_gear_in_shift_schedule;
            trans_2_B.s7_SFunction_o2 = 2;
        }
        break;
    }
}
break;
case IN_transition34_in_shift_schedule:
    /* During Start transition34 */
    if(trans_2_B.s6_vss_kmh <= trans_2_B.s6_Product5) {
        /* Exit Atomic Start shift_pending3 */
        chartInstance->State.is_transition34_in_shift_schedule=IN_NO_ACTIVE_CHILD;
        /* Exit Atomic Start transition34 */
        /* Entry Atomic Start third_gear */
        chartInstance->State.is_shift_schedule=IN_third_gear_in_shift_schedule;
        trans_2_B.s7_SFunction_o2 = 3;
    } else {
        switch(chartInstance->State.is_transition34_in_shift_schedule) {
            case IN_NO_ACTIVE_CHILD:
                break;
            case IN_shift_pending3_in_transition34:
                /* During Start shift_pending3 */
                if(chartInstance->LocalData.ctr > DELAY_in_shift_schedule) {
                    /* Exit Atomic Start shift_pending3 */
                    /* Entry Atomic Start shifting3 */
                    chartInstance-
>State.is_transition34_in_shift_schedule=IN_shifting3_in_transition34;
                    trans_2_B.s7_SFunction_o2 = 4;
                } else {
                    chartInstance->LocalData.ctr = (int8_T )(chartInstance->LocalData.ctr
+ 1);
                }
                break;
            case IN_shifting3_in_transition34:
                /* During Start shifting3 */
                if(a_gear == 4) {
                    /* Exit Atomic Start shifting3 */
                    chartInstance-
>State.is_transition34_in_shift_schedule=IN_NO_ACTIVE_CHILD;
                    /* Exit Atomic Start transition34 */
                    /* Entry Atomic Start fourth_gear */
                    chartInstance-
>State.is_shift_schedule=IN_fourth_gear_in_shift_schedule;
                    trans_2_B.s7_SFunction_o2 = 4;
                }
                break;
        }
    }
}
break;
case IN_transition43_in_shift_schedule:
    /* During Start transition43 */
    if(trans_2_B.s6_vss_kmh > trans_2_B.s6_Product4) {
        /* Exit Atomic Start shift_pending */
        chartInstance->State.is_transition43_in_shift_schedule=IN_NO_ACTIVE_CHILD;
        /* Exit Atomic Start transition43 */
        /* Entry Atomic Start fourth_gear */
        chartInstance->State.is_shift_schedule=IN_fourth_gear_in_shift_schedule;
        trans_2_B.s7_SFunction_o2 = 4;
    } else {
        switch(chartInstance->State.is_transition43_in_shift_schedule) {
            case IN_NO_ACTIVE_CHILD:
                break;

```

```

        case IN_shift_pending_in_transition43:
            /* During Start shift_pending */
            if(chartInstance->LocalData.ctr > DELAY_in_shift_schedule) {
                /* Exit Atomic Start shift_pending */
                /* Entry Atomic Start shifting */
                chartInstance-
>State.is_transition43_in_shift_schedule=IN_shifting_in_transition43;
                trans_2_B.s7_SFunction_o2 = 3;
            } else {
                chartInstance->LocalData.ctr = (int8_T )(chartInstance->LocalData.ctr
+ 1);
            }
            break;
        case IN_shifting_in_transition43:
            /* During Start shifting */
            if(a_gear == 3) {
                /* Exit Atomic Start shifting */
                chartInstance-
>State.is_transition43_in_shift_schedule=IN_NO_ACTIVE_CHILD;
                /* Exit Atomic Start transition43 */
                /* Entry Atomic Start third_gear */
                chartInstance-
>State.is_shift_schedule=IN_third_gear_in_shift_schedule;
                trans_2_B.s7_SFunction_o2 = 3;
            }
            break;
        }
    }
    break;
}
}
}
}

_sfEvent_trans_2_ = previousEvent;
}

#undef IN_first_gear_in_shift_schedule
#undef IN_fourth_gear_in_shift_schedule
#undef IN_second_gear_in_shift_schedule
#undef IN_third_gear_in_shift_schedule
#undef IN_transition12_in_shift_schedule
#undef IN_transition21_in_shift_schedule
#undef IN_transition23_in_shift_schedule
#undef IN_transition32_in_shift_schedule
#undef IN_transition34_in_shift_schedule
#undef IN_transition43_in_shift_schedule
#undef IN_shift_pending_in_transition12
#undef IN_shifting_in_transition12
#undef IN_shift_pending_in_transition21
#undef IN_shifting_in_transition21
#undef IN_shift_pending2_in_transition23
#undef IN_shifting2_in_transition23
#undef IN_shift_pending_in_transition32
#undef IN_shifting_in_transition32
#undef IN_shift_pending3_in_transition34
#undef IN_shifting3_in_transition34
#undef IN_shift_pending_in_transition43
#undef IN_shifting_in_transition43
#undef DELAY_in_shift_schedule
}
/* DataTypeConversion Block: <S5>/Data Type Conversion */
trans_2_B.temp8 = (real32_T)trans_2_B.s7_SFunction_o2;

/* Lookup Block: <S5>/c1_table */
trans_2_B.temp9 = rt_Lookup32(&trans_2_P.s5_c1_table_XData[0],
4,
trans_2_B.temp8,
&trans_2_P.s5_c1_table_YData[0]);

/* Gain Block: <S5>/Gain */
pcl = trans_2_B.temp9 * (trans_2_P.s5_Gain_Gain);

```

```

/* Lookup Block: <S5>/c2_table */
trans_2_B.temp9 = rt_Lookup32(&trans_2_P.s5_c2_table_XData[0],
4,
trans_2_B.temp8,
&trans_2_P.s5_c2_table_YData[0]);

/* Gain Block: <S5>/Gain1 */
pc2 = trans_2_B.temp9 * (trans_2_P.s5_Gain1_Gain);

/* Lookup Block: <S5>/c3_table */
trans_2_B.temp9 = rt_Lookup32(&trans_2_P.s5_c3_table_XData[0],
4,
trans_2_B.temp8,
&trans_2_P.s5_c3_table_YData[0]);

/* Gain Block: <S5>/Gain2 */
pc3 = trans_2_B.temp9 * (trans_2_P.s5_Gain2_Gain);

/* Lookup Block: <S5>/c4_table */
trans_2_B.temp9 = rt_Lookup32(&trans_2_P.s5_c4_table_XData[0],
4,
trans_2_B.temp8,
&trans_2_P.s5_c4_table_YData[0]);

/* Gain Block: <S5>/Gain3 */
pc4 = trans_2_B.temp9 * (trans_2_P.s5_Gain3_Gain);

/* Lookup Block: <S5>/b12_table */
trans_2_B.temp8 = rt_Lookup32(&trans_2_P.s5_b12_table_XData[0],
4,
trans_2_B.temp8,
&trans_2_P.s5_b12_table_YData[0]);

/* Gain Block: <S5>/Gain4 */
pb12 = trans_2_B.temp8 * (trans_2_P.s5_Gain4_Gain);
}

/* End of Functions for System "<S2>/trans_slow_torques" */

/*
* trans_2.c
*
* Real-Time Workshop code generation for Simulink model "trans_2.mdl".
*
* Model Version : 1.15
* Real-Time Workshop file version : 4.0 $Date: 2000/09/19 19:45:27 $
* Real-Time Workshop file generated on : Thu Jan 24 23:22:52 2002
* TLC version : 4.0 (Aug 21 2000)
* C source code generated on : Thu Jan 24 23:22:54 2002
*
* Relevant TLC Options:
* InlineParameters = 0
* RollThreshold = 5
* CodeFormat = Embedded-C
*
* Simulink model settings:
* Solver : FixedStep
* StartTime : 0.0 s
* StopTime : 60.0 s
* FixedStep : 0.02 s
*/

#include "trans_2.h"
#include "trans_2_prm.h"

/* Start of Functions in model "trans_2" */

/* model step function */
void trans_2_step(void)
{
/* S-Function (fcncallgen) Block: <Root>/Function-Call Generator */

```

```

/* Output and update for function-call system: <S2>/trans_slow_ctl */
/* Stateflow Chart Code: <S2>/trans_slow_ctl */
{
  SFtrans_slow_ctlInstanceStruct *chartInstance = (SFtrans_slow_ctlInstanceStruct
*)trans_2_DWork.s3_SFunction_PWORK.ChartInstance;
#define event_time_slow_in_trans_slow_ctl 1
  {
    int previousEvent;
    previousEvent = _sfEvent_trans_2_;
    /* Broadcast of input event(s) */

    _sfEvent_trans_2_ = event_time_slow_in_trans_slow_ctl;
    {
      /* During Start trans_v1_1/trans_slow/trans_slow_ctl */
      /* Entry Internal Start trans_v1_1/trans_slow/trans_slow_ctl */
      if(_sfEvent_trans_2_ == event_time_slow_in_trans_slow_ctl) {
        trans_slow_torques();
      }
    }

    _sfEvent_trans_2_ = previousEvent;
  }
#undef event_time_slow_in_trans_slow_ctl
}

/* Outport Block: <Root>/Pc1 */
trans_2_Y.root_Pc1 = pc1;

/* Outport Block: <Root>/Pc2 */
trans_2_Y.root_Pc2 = pc2;

/* Outport Block: <Root>/Pc3 */
trans_2_Y.root_Pc3 = pc3;

/* Outport Block: <Root>/Pc4 */
trans_2_Y.root_Pc4 = pc4;

/* Outport Block: <Root>/PB12 */
trans_2_Y.root_PB12 = pb12;

/* (no update code required) */
}

/* End of Functions in model "trans_2" */
#include "trans_2_reg.h"
/* [EOF] trans_2.c */

```