

Approach to Application of Executable Specifications in Development of Automatic Transmission Control Software

Yoshihiro MATSUYAMA* Kenji HAGIWARA* Satoshi SEINO*

ABSTRACT

Model-based development is being introduced to the development of control software for automatic transmissions in order to increase control reliability and reduce development times. The introduction of model-based development necessitates an enhancement in the readability of executable specifications formulated using MATLAB, i.e., specifications for simulatable models, and the reduction of the time for formulation of these specifications. An original style guide was developed as a supplement to the general description guidelines provided by the MAAB and J-MAAB style guides, and the like, on the basis of a method eschewing connecting lines and division of the model based mainly on Stateflow, and post-processing was applied to the expression of the model when documents were formulated. These methods increased the readability of the specifications. In addition, a block library for automatic transmission control, an automatic modeling tool conducting routine processing, and a tool that automatically sets existing variable information were developed in order to support modeling conforming to the new style guide. These tools have reduced the time for modeling to less than one-twentieth of the time for manual modeling. This has enabled executable specifications to be employed in the development of automatic transmission control software.

1. Introduction

The importance of automotive electronic control technologies is increasing year-by-year, and the scale and complexity of control software continues to grow. Demands on automakers to increase the reliability of control, reduce development times, and lower costs are also increasing. Against this background, the introduction of model-based development (MBD) is today proceeding rapidly as a means of increasing the efficiency of automotive software development^{(1),(2)}.

When MBD is employed, executable specifications, i.e., simulatable control models, are formulated using MATLAB at the design stage, the upstream phase of the development process. These control models are combined with plant models to enable virtual control tests. C codes for implementing can also be automatically generated.

MATLAB offers a high degree of freedom in description, and enables a diverse range of types of modeling to be conducted, even when the same control logic is described. This can cause readability to decline,

and the treatment of MATLAB models as control specifications to be used in the development of mass-produced software has represented a challenge, including for suppliers. In response to this issue, MATLAB users' bodies in automobile industry, the MathWorks Automotive Advisory Board (MAAB) and the Japan MATLAB Automotive Advisory Board (J-MAAB), have formulated MATLAB guidelines and defined the fundamental principles for modeling^{(3),(4)}. However, these guidelines have proven to be insufficient to the description of automatic transmission control, which involves a large amount of switching between different shift stages depending on driving conditions and the driving environment, meaning that numerous condition branches are generated and a great deal of resultant switching between tables and maps is necessary. Original rigorous guidelines were therefore formulated to help ensure readability, but it was discovered that while these guidelines ("the style guide" below) would increase readability, they would also result in an increase in modeling time.

* Automobile R&D Center

2. Development Aims

The following two development aims were established in order to enable MATLAB models to be treated as control specifications for use in the development of mass-produced software:

- (1) Achievement of increased readability for MATLAB models
 - (2) Reduction of modeling time for MATLAB models
- To enable these aims to be realized, the following would also be developed simultaneously:
- (a) A style guide enabling the formulation of high-readability models
 - (b) Support tools to enable the reduction of modeling time

3. Development of Style Guide Enabling the Formulation of High-readability Models

In the development of the style guide, four methods were considered to increase the readability of the models:

- (1) A modeling method based mainly on Stateflow
- (2) A method eschewing connecting lines for exchange of data
- (3) A method eschewing division of the model when searches are conducted for tables and maps
- (4) A post-processing method for the expression of the model when documents are generated

These four methods will be discussed in detail below.

3.1. Modeling Method Based Mainly on Stateflow

Broadly speaking, MATLAB contains two control design tools, Simulink and Stateflow. Simulink's forte is the description of data flows that specify the flow of transformation of input data, while Stateflow's is the description of control flows that specify the order, conditions, and timing of executing those data flows. Examples are presented in Figs. 1 and 2, respectively. The use of both tools in response to control is important in terms of high-readability modeling, but it is claimed that 85% or more of the flows in automotive control are control flows⁽⁵⁾. An analysis of existing mass-production control specifications also showed that the majority of flows were control flows even in control systems for automatic transmissions, and demonstrated that readability declines when the modeling method is based primarily on Simulink. Given this, the project discussed here set out to base the modeling method for automatic transmission control systems primarily on Stateflow.

3.2. Method Eschewing Connecting Lines for Exchange of Data

One of the cumbersome features of modeling control specifications using Simulink and Stateflow, and one that results in a decline in readability, is the use of connecting lines for the exchange of data. The structure of the software may necessitate data reference points (bounded

by red in Fig. 3) that span multiple levels, and when this is the case, it is necessary to use connecting lines for data exchange between all levels. This increases the scale and the number of levels of the model, and the increase in the variables renders the connecting lines more complex and may cause readability to decline.

In order to address this issue, attention was focused on a variable management method for Stateflow variables, and principles for the management of the variables described on the Stateflow charts ("the charts" below) were specified in the style guide. Normally, the variables described on the charts are all managed at either the machine level or the chart level (Fig. 4). The values of the variables managed at the machine level can be referred

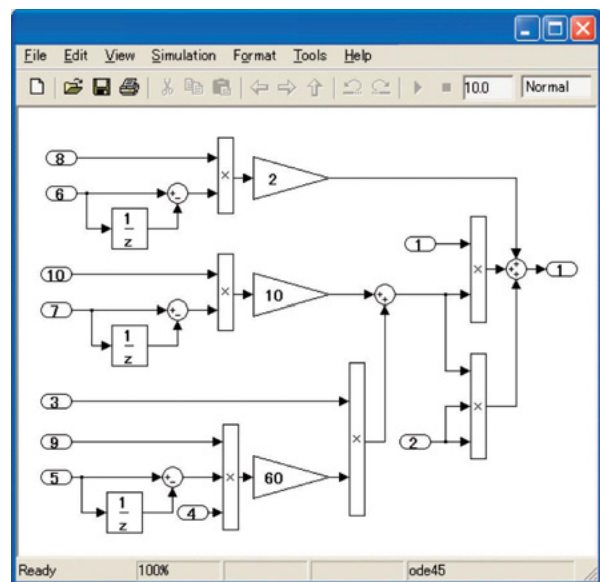


Fig. 1 Sample of Simulink model

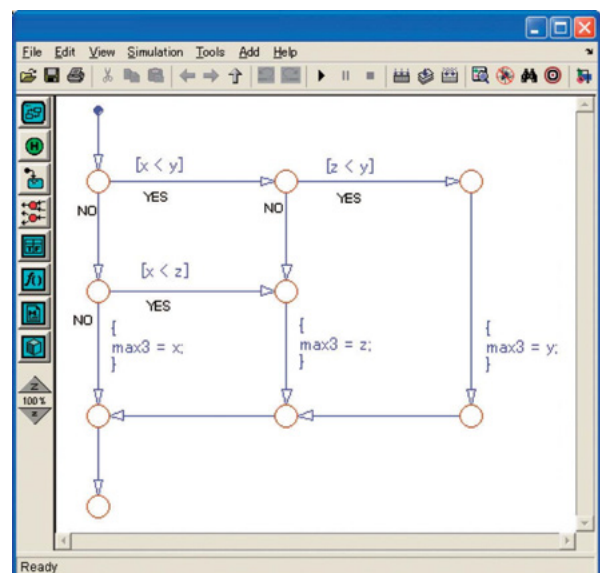


Fig. 2 Sample of Stateflow model

to or revised from any of the charts in the model. By contrast, the values of the variables managed at the chart level can only be referred to or revised from that chart. Exploiting this characteristic, the new principles specify management of variables shared between multiple charts at the machine level, and management of variables used only on a single chart at the chart level. This helps ensure that even when the same variable is used on multiple charts, there is no necessity to exchange data between charts, and therefore no necessity for connecting lines.

This has enabled the realization of high-readability model configurations with few connecting lines. Figure 5 shows the control model produced by application of the variable management method described above to the model shown in Fig. 3.

3.3. Method Eschewing Division of Model when Searches Are Conducted for Tables and Maps

In some control logics, as illustrated by the speed change characteristic logic for an automatic transmission shown in Fig. 6, specific parameters are described in tables and maps, and control is applied using the results of searches for these tables and maps. When modeling this control, the table and map searches cannot be directly described on the charts, and the Lookup Table (LUT) and Lookup Table (2-D) (LUT2D) block shown in Fig. 7 have therefore been used. However, as Fig. 8 shows, this raises the issue of the division of the model during the process of the table or map search, and the consequent decline in the readability of the model.

It was therefore decided to introduce a modeling

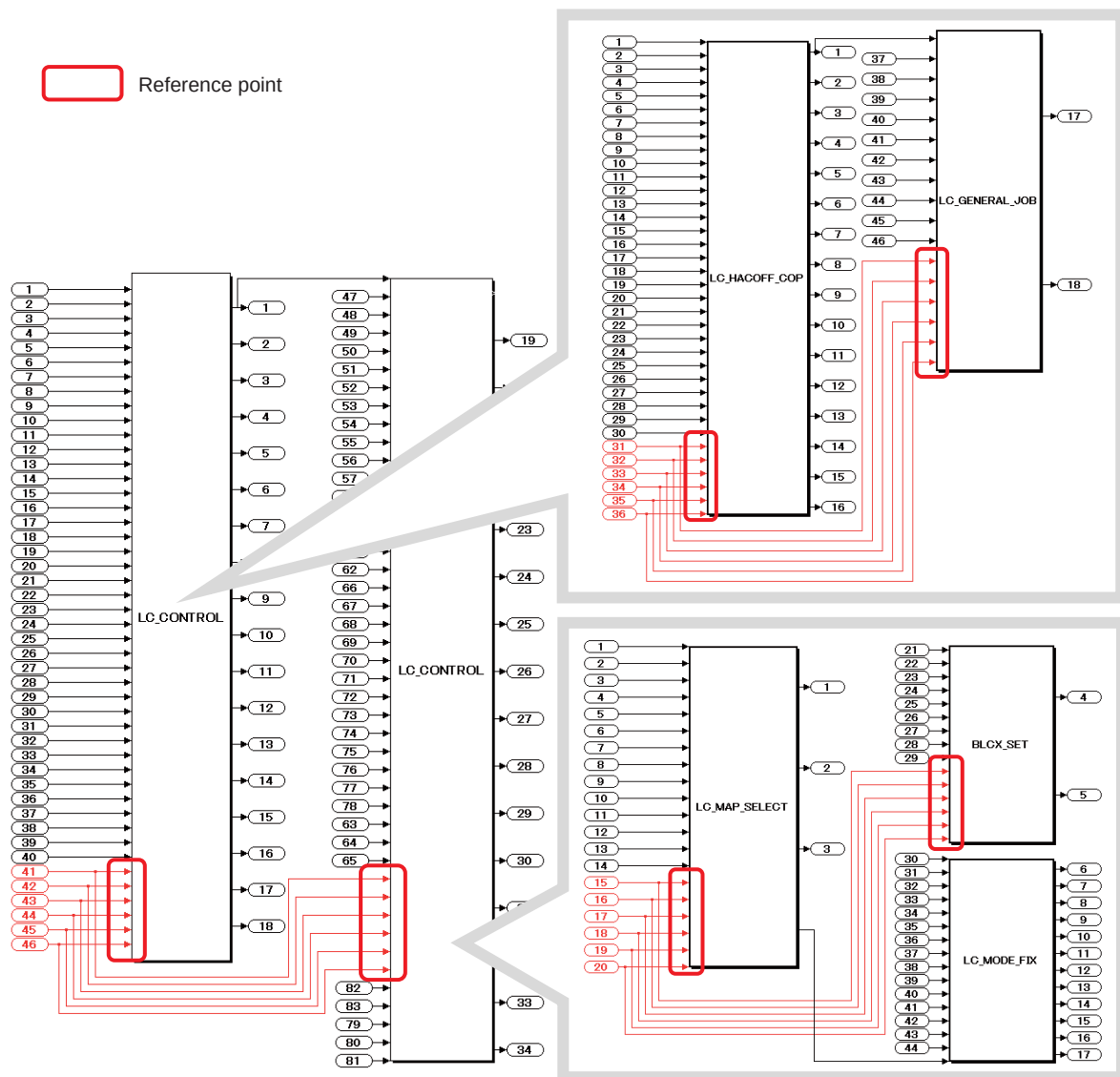


Fig. 3 Complicated model with many connecting lines

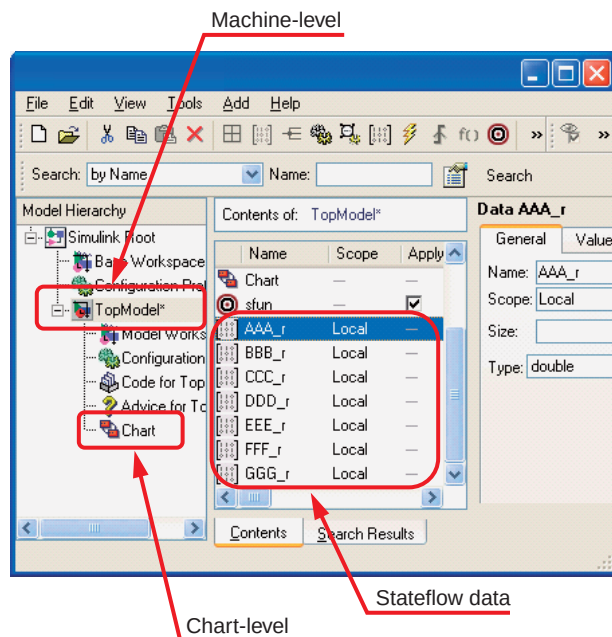


Fig. 4 Machine-level and chart-level

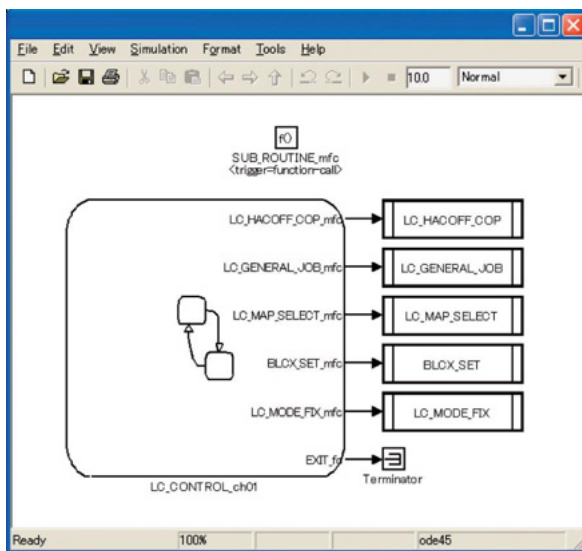


Fig. 5 Simple model with few connecting lines based on machine-level

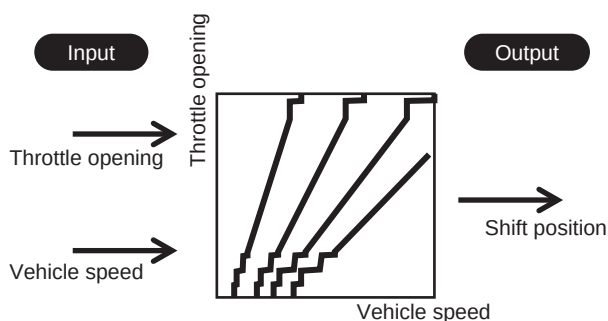


Fig. 6 Speed change characteristic logic of automatic transmission

method using loops of event signals and return values, as shown in Fig. 9. In the proposed method, the table search is conducted by outputting an event signal (Event) that invokes the LUT block in the area in which the table search is desired, and at the same time providing a search parameter, (x); the search results are again returned to the chart as a return value, (y). This enables modeling using table and map searches without division of the model.

3.4. Post-processing Method for Expression of the Model when Documents are Generated

Control specifications for use in the development of mass-produced software are not used only by designers, but are shared among a wide range of individuals involved in development, including testers, bench and vehicle personnel, and suppliers. This has made it necessary to render models into documents that can be viewed in widely used applications such as Microsoft Office Word.



Fig. 7 Lookup Table and Lookup Table (2-D)

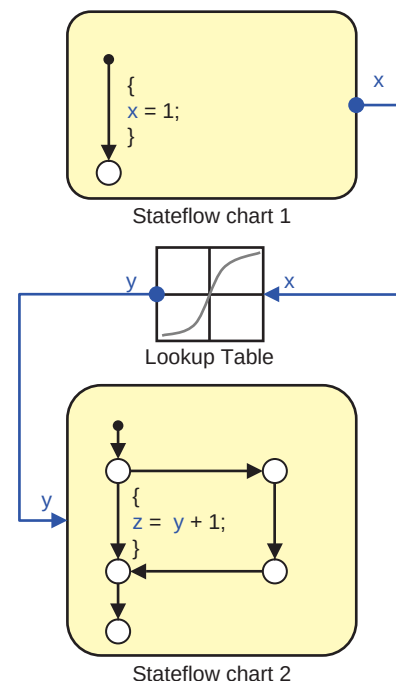


Fig. 8 Model divided into two charts by Lookup Table block

Table 1 Notation rule of variables on document

	RAM	CALIB	WORKRAM
Color	Blue	Red	Blue
Prefix	—	#	—
Capital/ Small	Capital	Capital	Small

Table 2 Naming rule of variables on stateflow chart

	RAM	CALIB	WORKRAM
Suffix	_r	—	—
Capital/ Small	Capital	Capital	Small

Rules for notation, shown in Table 1, were therefore formulated to enable simple visual discrimination of the three types of variables (RAM, CALIB, and WORKRAM) used in control specifications when these documents are created. In addition, points at which conditions branch are notated as diamond shapes. In order to enable documents conforming to these rules of notation to be automatically generated from the model, rules for naming the variables described on the chart were preliminarily specified in the style guide (Table 2). Following this, a post-processing tool was developed to conform the notation of the model to the rules of notation shown in Table 1. This tool was formed by linking MATLAB and the application used in the documents.

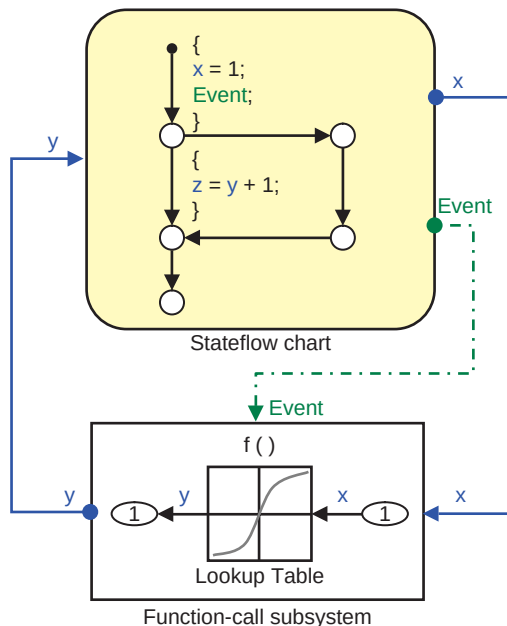


Fig. 9 Modeling method: Searching table data without division

First, MATLAB was programmed to classify the types of variables, specify the points at which conditions branch, and copy the images in the model. The classification of the variables followed the naming rules specified in the style guide (Table 2). The points at which conditions branched were specified from the arrangement of junctions (the circular objects in Fig. 10) and transitions (the arrow-shaped objects in Fig. 10) on the charts. In Fig. 10, if the condition $(X_r < Y)$ above the transition arrow is manifested, processing will proceed in the direction of the arrow $z=1$. If the condition is not manifested, processing will proceed in the direction of the arrow $z=0$. i.e., at a point at which a condition branches, two transitions extend from one junction. The points at which conditions branched were specified by searching for this feature in relation to all the junctions on the chart and analyzing the

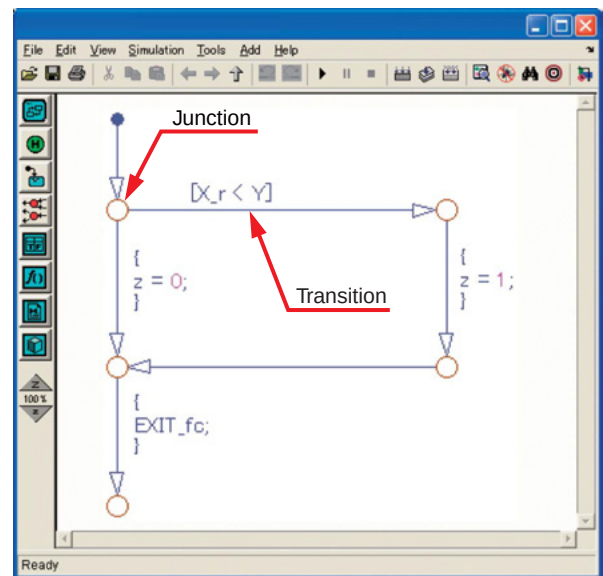


Fig. 10 Junction and transition

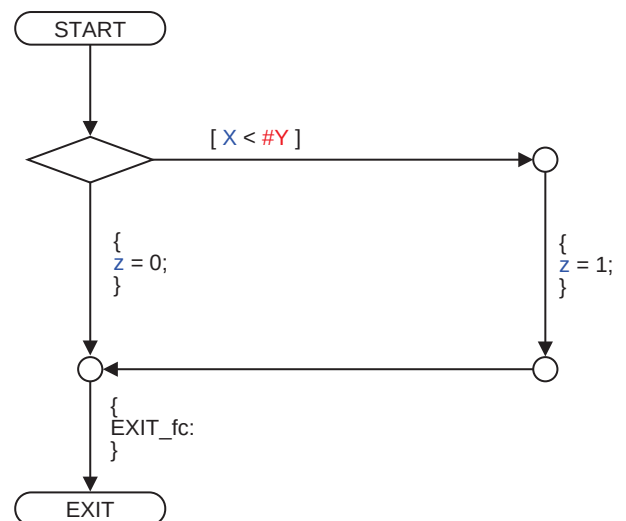


Fig. 11 Document generated from model in Fig. 10

results. A meta format, i.e., a data format in which multiple types of data such as images and text are included, was selected for the copying of the model images, enabling MATLAB functions to be employed and the text data to be retained.

Next, the application used for the documents was booted, and programmed to enable pasting of images from the model and processing of notation. An Active X function supported by MATLAB was employed to enable the document application to be booted from MATLAB. The notation processing – the use of different colors for variables and the substitution of diamond shapes for condition branch points – employed automatic functions of the document application, and was conducted from MATLAB. Figure 11 shows a document automatically generated from the model shown in Fig. 10.

The automatic document generation tool described above, which linked MATLAB and the application employed for the document, enabled the achievement of increased readability in the specifications.

4. Development of Tools to Support Reduction of Modeling Time

Chapter 3 discussed four methods proposed in the development of the style guide in order to increase the readability of models. This chapter will discuss the following three modeling support tools that were developed in order to reduce the time needed for modeling

executable specifications conforming to the style guide:

- (1) A block library for automatic transmission control
- (2) An automatic modeling tool for routine processing
- (3) A tool for the automatic setting of existing variable data

4.1. Block Library for Automatic Transmission Control

As Fig. 12 shows, MATLAB's standard block library contains a diverse range of blocks for use in control design. From these, the present project selected the blocks necessary for the modeling of automatic transmission control in accordance with the style guide, and developed a unique, customized block library (Fig. 13). The blocks in this library were customized to automatically display the data contained in the blocks on masks. Figure 14 shows an example of this. When the axis data name is input to the graphical user interface (GUI), the data contained in the table search block is automatically displayed on a mask.

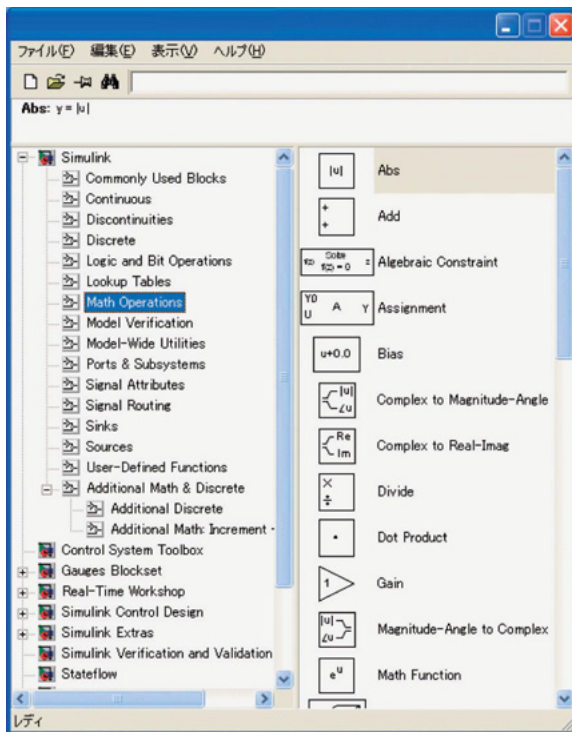


Fig. 12 Simulink library

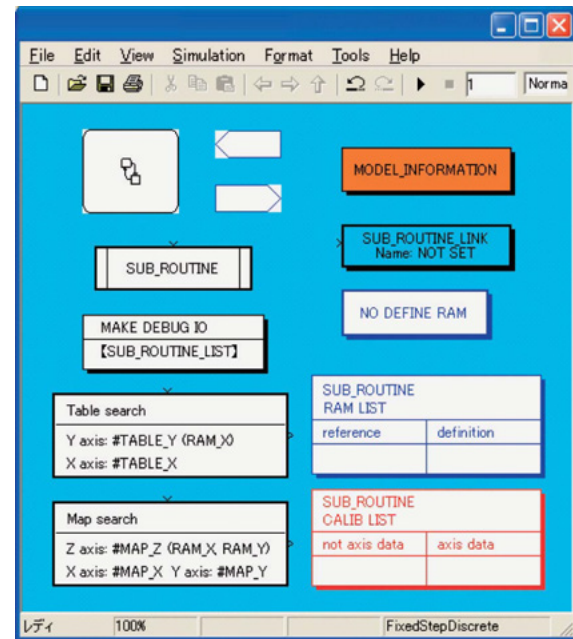


Fig. 13 Automatic transmission modeling library

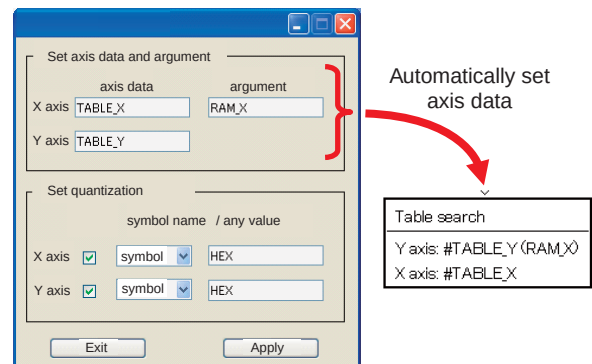


Fig. 14 GUI and customized table block

This function was realized using GUIDE, the MATLAB graphical user interface development environment, to develop a GUI for input of the axis data name and an M-File program for the automatic display of input data on a mask.

The development of the block library for automatic transmission control use increased the readability of the model and reduced the time for modeling.

4.2. Automatic Modeling Tool for Routine Processing

As indicated in Chapter 3, control logics employing table searches and map searches may, depending on control conditions, need to switch between multiple tables and maps. When this is the case, a routinized model of the type shown in Fig. 15, formed from switch case blocks, merge blocks, and the customized table and map blocks discussed in Section 4.1. is used. However, the greater the amount of switching between tables and maps, the more the number of elementary modeling tasks increases, and this also increases modeling error.

An automatic modeling tool was therefore developed in order to enable this type of routine modeling to be conducted automatically. This tool is made up of a GUI for data input by designers, and an automatic modeling program.

First, for the design of the GUI, the switching patterns

in existing mass-produced automatic transmission control specifications were analyzed. The results of this analysis showed five representative switching patterns. These are, in addition to table switching and map switching, CALIB data switching, RAM data switching, and map switching under multiple conditions. In addition, the results also indicated that the number of times switching is conducted and the switching conditions vary in response to the type of control applied, necessitating a high degree of freedom in design. Therefore, as Fig. 16 shows, the GUI was designed to enable designers to freely select from the five representative patterns, and to easily set the number of times switching is conducted as well as switching conditions.

Next, a program enabling automatic modeling using Simulink or Stateflow based on the data selected and input using the GUI was developed. This program automatically executes the following sequence of processes:

- (1) The blocks, which are necessary for automatic modeling of the switching pattern selected using the GUI, are

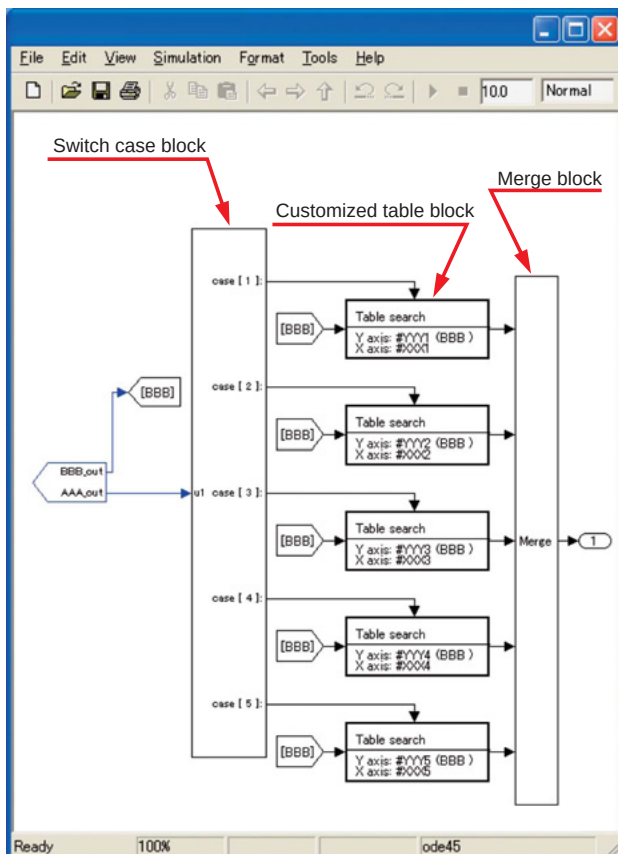


Fig. 15 Table switching model

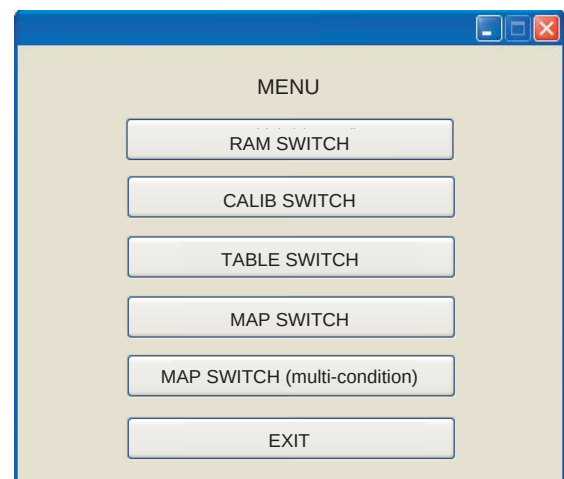


Fig. 16 GUI of auto-modeling tool

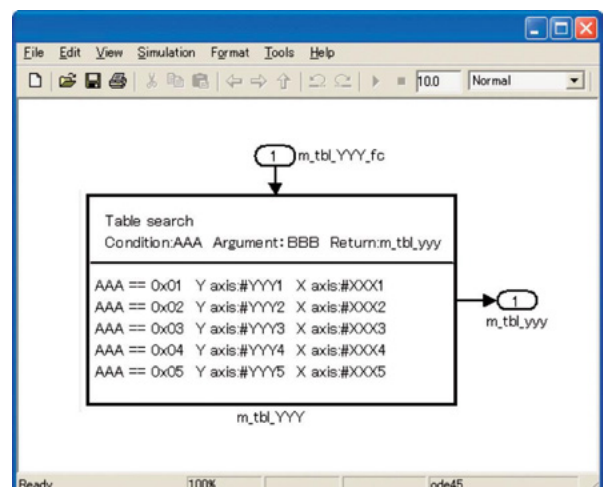


Fig. 17 Result of auto-modeling

obtained from the block library discussed in Section 4.1., and placed in a function-call subsystem.

- (2) The data input using the GUI are reflected in the blocks.
- (3) The individual blocks are connected.
- (4) The switching logic is notated on a mask in the function-call subsystem.

Figure 17 shows a result obtained from automatic modeling using this tool.

The use of the automatic modeling tool discussed above, made up of a GUI and an automatic modeling program, enabled modeling time to be reduced.

4.3. Tool for Automatic Setting of Existing Variable Data

For all of the variables treated in control specifications for use in the development of mass-produced software, it is necessary to set data scale (LSB) and data type information, which specifies the accuracy of the variables. This is also the case when MATLAB models are treated as mass-production control specifications. In the case of large-scale control models, which contain a vast amount of variables, a large amount of time is required to input data for each variable.

However, the existing variables treated in mass-produced software are managed in databases, and this makes this data easy to reuse.

A GUI was therefore developed that would link to databases, automatically obtain existing variable data, input it in bundles, and display the results of setting the data in the model as a list. In addition, a specialized block was developed that would enable the variable data to be retained in the MATLAB model.

To realize the ability to link with databases, i.e., the ability to automatically obtain existing variable data and input the data as a bundle, the system was preliminarily

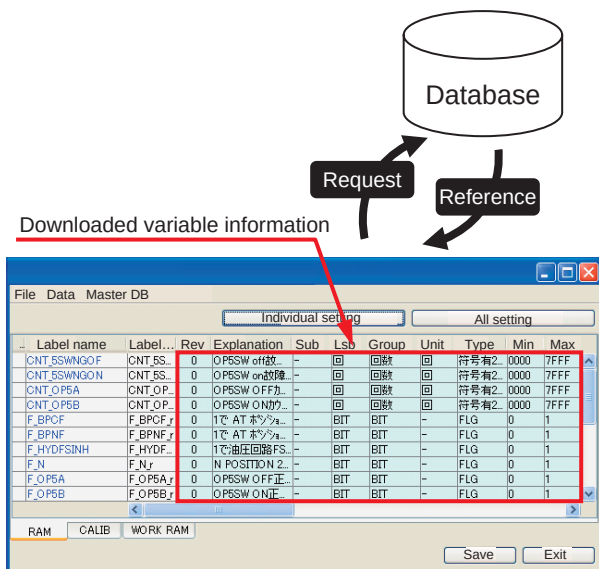


Fig. 18 Link system between database and GUI managing variable information

programmed to enable the model to be automatically analyzed in accordance with the naming rules (Table 2) specified in the style guide, and the variables, for which it was necessary to be set in the model, to be automatically identified by their variable names, when the GUI was booted. The system was then programmed to enable it to refer to variable data that is on the database and associated with the variable names using the database language SQL (Fig. 18).

To enable the variable data that had been input in a bundle to be retained in the MATLAB model, the system was programmed to enable the variable data to be stored in a special block properties column. The use of MIME, also used in E-mail and other applications, as the data format for the storage of the data enabled matrix data, formed from multiple types of variable data, to be stored in a single block (Fig. 19).

The tool for the automatic setting of existing variable data described above reduced modeling time.

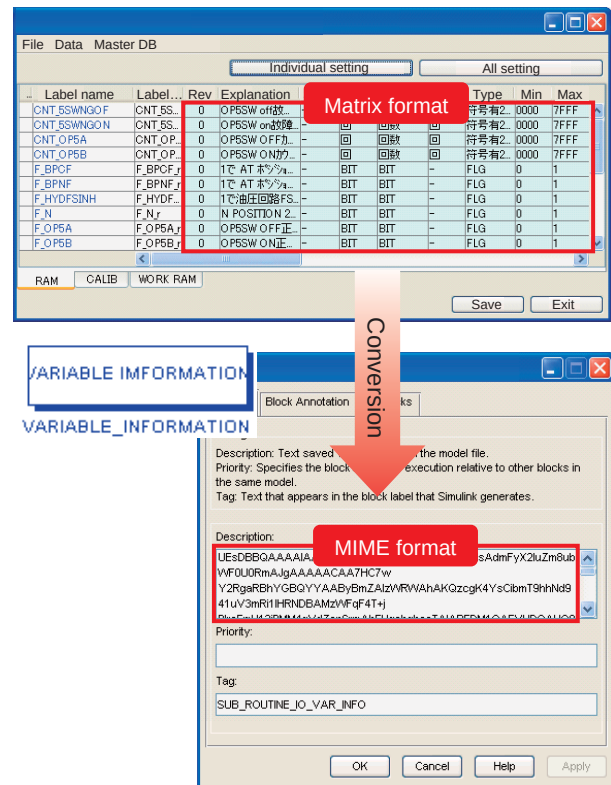


Fig. 19 Conversion of variable information format

5. Example of Actual Application

Figure 20 shows executable specifications for automatic transmission control software that were formulated using the developed style guide and modeling support tools. The introduction of the new tools enabled a MATLAB model, which is able to be treated as control

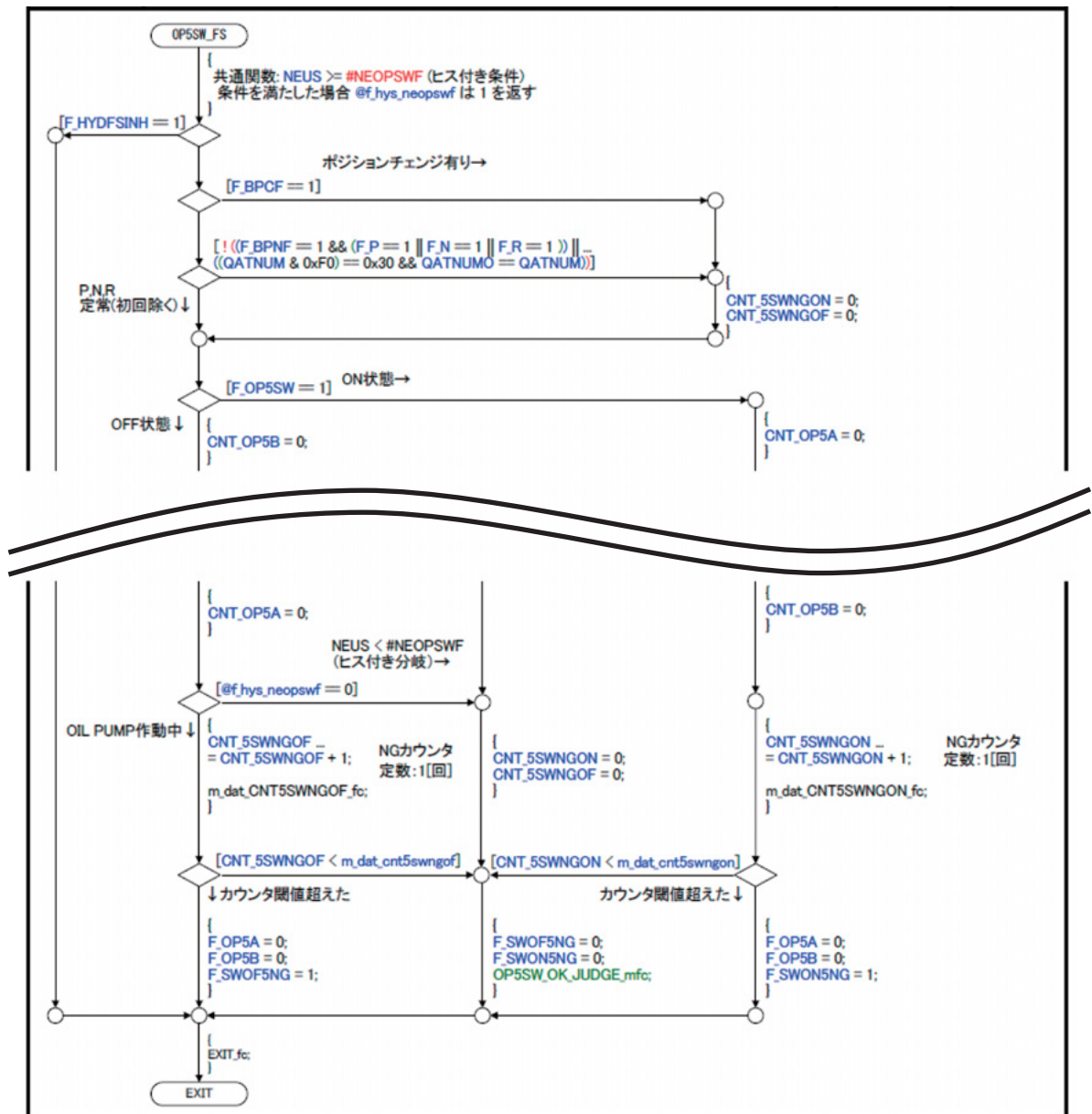


Fig. 20 Example of application

specifications for the development of mass-produced software, to be formulated in less than one-twentieth of the time previously needed.

6. Conclusion

- (1) The development of a style guide based mainly on Stateflow that eschewed connecting lines and did not necessitate division of the model, and the development of a post-processing tool for notation of the model when documents are generated, has increased the readability of control specifications for automatic transmissions.

- (2) The development of a block library for automatic transmission control, an automatic modeling tool for routine processing, and a tool for the automatic setting of existing variable data has reduced the time needed for modeling in conformance with the abovementioned style guide to less than one-twentieth of the time previously needed.

The developed style guide and tools are therefore able to be employed in the formulation of executable specifications for use in the development of control software for automatic transmissions.

References

- (1) Ohtsuka, K., Oi, M.: Deployment of Model Based Development for Automotive Software, Journal of Society of Automotive Engineers of Japan, Vol. 60, No. 6, p. 80-85 (2006)
- (2) Minowa, S., Hirota, T., Yokoyama, H., Ishiguro, T., Ohtsuka, S.: Autocoding of Power Plant Control Software, Honda R&D Technical Review, Vol. 16, No. 1, p. 243-248
- (3) <http://www.mathworks.com/industries/auto/maab.html>
- (4) <http://j-maab.cybernet.jp/>
- (5) http://www.cybernet.co.jp/documents/pdf/products/magazine/cybernet_news/no87.pdf

■ Author ■



Yoshihiro MATSUYAMA



Kenji HAGIWARA



Satoshi SEINO