# A State Transition Model Description Language stmc and Its Tools

— an Extension of the C Programming Language for Developing Driver Software and Firmware with Models —

Nobuhiko Ogura†, Ikuta Tanigawa‡, Takuya Todoroki†, Kenji Arai†, and Harumi Watanabe‡

†Tokyo City University, ‡Tokai University

ABSTRACT

We present a state transition model description programming language. It can be translated to pure standard C programs without any OS or handwritten frameworks, hence it is suit for developing low level driver software and firmware, unlike many other automatic software generation tools from software models that usually focuses on higher level models. We show the language and translator to executable software and visual diagram generator, and analysis tools, with embedded software examples.

## OVERVIEW

As a consequence of increasing scale and complexity of embedded softwares, developing techniques with models attract much attention. A large number of studies (DSL approach, MDA, MBD, MBSE, Software factory, etc.) are made to establish model level software design method with appropriate model expression and also with less gap between its models and implementations.

To introduce such techniques to existing developing, we have met following problems:

(i) Developers must be trained on model writing tools, and (for some of them) also have to study legitimate action language, which frequently has different syntax from programming language.

(ii) Model handling mechanism must be adapted to each target or tool. For example, (a) tool integration, and (b) customized startup or model handling framework (sometimes assume some OS) must be prepared, for each target.

These prevent model techniques from spreading for various area of software. Especially, in handling state transition model, the latter (ii-b) makes it difficult to use these techniques in low level (near to hardware) or device driver software.

We propose, a model description language stmc, which has developer friendly notation and is easy to use with existing toolchains. To resolve the above problems, the language is designed by extending the C programing language to handle state transition models in embedded software.

## FEATURE

Stmc source programs are translated into the C programming language and compiled to be downloaded and executed. The language and its translator has the following feature:

### ☐ FRIENDLY to Existing Toolchain

The translator of stmc outputs the standard C program (ISO/ANSI-C known as C90/C89), and most existing embedded toolchain accept them without any modification.

### ☐ FRIENDLY to Legacy Code

As shown in Figure 1, stmc is realized by slight extension of the C programming language, and accept most existing C source codes and header files. For embedded developing environments with nonstandard or implementation-dependent extensions, additional options, such as lazy include mechanism, are available.

### ☐ FRIENDLY to Programming People

In the stmc language, actions and guard conditions of state transition machines are denoted by the C language syntax, and knowledge of different action language does not required. In addition, state transition machine definitions are placed with C language functions and variables declarations. This enables small step migration, and makes introducing model development easier.

file ::= external_definition | file external_definition

external_definition ::= function_definition | declaration
**| *stm_declaration* | *stm_definition***

('file' is the start symbol. Extended part is shown in italic.)

Figure 1: Part of the extended stmc grammar.

## FUNCTIONALITY / TOOLS

Following tools support development embedded software with stmc.

### ☐ Code Generation

By the stmc translator, stmc source programs are translated into the C programming language. It has been applied for targets: Renesas Electronics M16C, R8C, Spansion FM3, STMicroelectronics STM32F10X, Atmel AT91SAM7S256, Microchip PIC16 series, Cypress PSoC1 Series, and x86 Windows.

### ☐ Visualize

Stmc source code can be also translated to model diagram images in PDF, PostScript, png, etc. (as shown in Figure 2.) The model diagrams can be drawn with identifier and exact action codes, and diagrams with description string (not exact code) can be generated.

Diagrams can be colored by description string patterns, and some element can be extracted/excluded by the string patterns. A potential application of this mechanism is distinction of normal/exceptional transition in diagrams. That is an major issue in embedded software.

### ☐ Analyze

We have a tool which computes cyclomatic complexity of transition and actions. Another tool presents event occurrence dependency. These make static analyses of the stmc source codes (models). For using dynamic information, stmc model level debugging environments for CortexM3 with GNU gdb are planned.

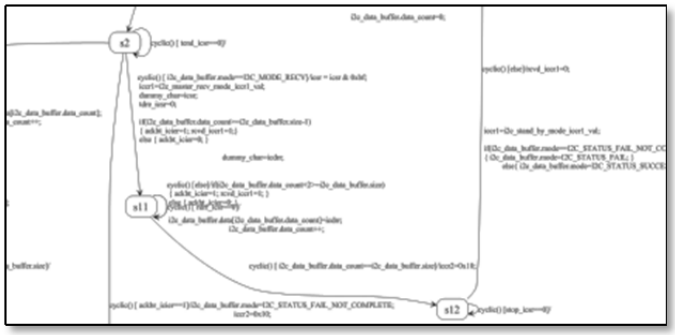


Figure 2: Part of a generated diagram from stmc source code of $I^2C$ communication firmware in R8C processor.