Mod Code Generator Manual

Table of Contents

1 Introduction

2 Code of practice for modeling in Modelio environment

Although Modelio itself is used mainly as UML modeling tool, the modeling environment will be used in different way to allows code generation with MCG, i.e. modeling will be focused more on data flow through model elements and interaction of signals (called signal processing or actions), than on representation of model structure and its elements.

The main objective of model package is to define couplings between different model components (data flow from one component to another). The model package is used also to define main input and output interface of entire model.

Each model component is treated as a separate module and MCG will generate separate C source file for each component. Components are used to define operations over model signals (also known as signal processing, actions).

Type package is used only to define allowed types of signals, which will be used later to define interface signals.

Please see below code of practice for modeling in Modelio environment, which will give you some introduction over modeling rules and as well please see Modeling Rules where you can find list of rules that you have to follow during modeling.

2.1 Model Creation

Create New Project

Open Modelio environment, then select File → Switch workspace and select directory, where you would like to place new model.

After that select File → Create a project. In Project name property type desired project name, then click Create the project.

Remove Unused Elements

Select Configuration → Libraries, then remove all Local libraries and Remote libraries.

After that go to Modules tab and disable all modules, then close Project configuration window.

Expand all model elements in model tree. Find and remove LocalModule element.

Rename Model Package.

Locate model package under UML sub-project element, click on it and press F2 on keyboard, then type desired model package name.

2.2 Type Package Creation

Create Type Package

Right-click on UML sub-project element and select Create element → Package. Click on new package, press F2 on keyboard and name it as Types.

Create Model Types

Right-click on type package and select Create element → Data Type. Click on new data type, press F2 on keyboard and name it as INT8

Repeat above process for the rest of required data types:

- INT16
- INT32
- INT64
- UINT8
- UINT16
- UINT32
- UINT64
- FLOAT32
- FLOAT64

2.3 Model Component Creation

Create New Component

Right-click on model package and select Create element → Component. Click on new component element, press F2 on keyboard and then give desired name to it.

Create Component Interfaces

Right-click on component and select Create element → Interface. Click on new interface element, press F2 on keyboard and name it Input Interface, then create two other interfaces for the same component and name them Output Interface and Local Parameters.

Table 2.3.1: Component Interface Types

Interface Type	Objectives
Input Interface	Defines input interface of the component, i.e. list of signals, which are inputs to the component.
Output Interface	Defines output interface of the component, i.e. list of signals, which are outputs from the component.
Local Parameters	Defines local parameters of the component, i.e. list of signals, which are used internally within the component, but are not in the same time part of input or output interface.

Right-click on desired interface and select Create element → Attribute. Double-click on new attribute. In Name property type desired signal name and start typing desired signal type in Type property (see Type Package Creation for list of available data types), then press Ctrl + Space combination on keyboard to see auto-completion list.

Repeat above step to add other signals to desired interface if needed. If you will not be using some interface type then simply do not add signals under it and leave it empty.

Create Component Activity

Right-click on component and select Create diagram, then select Activity diagram from available list of diagram. You should now find Activity element under your component. Remove from Activity two following elements:

1. this: <component name>
2. locals

Once Activity is created, please find Activity diagram under Activity, click on it and press F2 on keyboard, then enter desired name for Activity diagram.

Define Component Activity

Open Activity diagram and drag and drop interface signals (inputs, outputs and locals, see Table 2.3.1: Component Interface Types) into diagram space. If needed, you can drag and drop any signal again if you need to use it more than one time.

Within Control nodes box select Action – Create an Opaque Action, then click on diagram space where you wish to place new action. Double-click on new action.

In Name property type desired action type, please see list of available actions in Table 2.7.1: Allowed Actions).

Some kind of actions require to distinguish which input signal is first input signal in action equation in order to compute correct result.

As example:
Subtraction action y=a-b-c will give different result than y=b-a-c.

Please see *FIRST* marker needed column in Table 2.7.1: Allowed Actions to find out which actions require to distinguish first input signal. If you are using such action, then type in Description field *FIRST* marker, then name of first input signal, then *FIRST* marker again.

As example:
FIRST* first_input_signal_name *FIRST.

You can repeat above steps to create other actions if needed.

Within Flows box select Object Flow – Create an Object Flow, then connect component elements (signals and actions) on diagram space.

Create Next Components

Once component is defined you can create further components following steps since Create New Component.

2.4 Model Package Creation

Create Package Interfaces

Right-click on package and select Create element → Interface. Click on new interface element, press F2 on keyboard and name it Input Interface, then create another interface for the same package and name it Output Interface.

Table 2.4.1: Package Interface Types

Interface Type	Objectives
Input Interface	Defines main input interface of the package and entire model, i.e. list of signals, which are inputs to the model.
Output Interface	Defines main output interface of the package and entire model, i.e. list of signals, which are outputs from the model.

Right-click on desired interface and select Create element → Attribute. Double-click on new attribute. In Name property type desired signal name and start typing desired signal type in Type property (see Type Package Creation for list of available data types), then press Ctrl + Space combination on keyboard to see auto-completion list.

Repeat above step to add other signals to Input and Output Interfaces.

Create Package Activity

Right-click on package and select Create diagram, then select Activity diagram from available list of diagram. You should now find Activity element under your package. Remove from Activity the locals element.

Once Activity is created, please find Activity diagram under Activity, click on it and press F2 on keyboard, then enter desired name for Activity diagram.

Define Package Activity

Open Activity diagram and drag and drop Input and Output Interface model elements signals (see Table 2.4.1: Package Interface Types) into diagram space.

Once Input and Output Interface are placed on diagram space, drag and drop model component elements into it.

Within Flows box select Object Flow – Create an Object Flow, then connect model elements (interfaces and components) on diagram space. Data flow between model elements will be recognized basing on signal names and types.

As example:

If model package input interface and model component input interface have interface signal of same name and type and both model elements are connected on diagram, then data flow will be recognized between these two model elements.

2.5 Modeling Rules

Structure Rules

SR#1 RULE
Each model created within Modelio environment **shall** have following generic layout in order to allow proper code generation with MCG:

1. <model package>
    a) <model component 1>
    b) <model component 2>
    c) ...
    d) <model component n>
2. <type package>

SR#2 RULE
Each model **shall** have exactly one model package.

SR#3 RULE
Each model **shall** have exactly one type package.

SR#4 RULE
Each model package **shall** have at least one model component.

SR#5 RULE
<type package> **shall** be named Types.

SR#6 RULE
<type package> **shall** have following types:

- INT8
- INT16
- INT32
- INT64
- UINT8
- UINT16
- UINT32
- UINT64
- FLOAT32
- FLOAT64

Component Interface Rules

CIR#1 RULE
Model component **shall** always have three interface types as per Table 2.3.1: Component Interface Types.

CIR#2 RULE
If particular interface of model component is not used, then it **shall** be left empty.

CIR#3 RULE
Signals of model component **shall** be created only under three available interface types as per Table 2.3.1: Component Interface Types.

CIR#4 RULE
Each input signal of model component **shall** have unique name in comparison to input signals of other model components, i.e. other component **shall not** have input signal with same name.

CIR#5 RULE
Each output signal of model component **shall** have unique name in comparison to output signals of other model components, i.e. other component **shall not** have output signal with same name.

CIR#6 RULE
Each interface signal of model component **shall** have unique name in comparison to other interface signals of same model components.

Package Interface Rules

PIR#1 RULE
Model package **shall** always have two interface types as per Table 2.4.1: Package Interface Types.

PIR#2 RULE
Each interface of model package **shall** have at least one signal.

PIR#3 RULE
Signals of model package **shall** be created only under two available interface types as per Table 2.4.1: Package Interface Types.

PIR#3 RULE
Each interface signal of model package **shall** have unique name in comparison to other interface signals of same model package.

Component Activity Rules

CAR#1 RULE
Only allowed actions as per Table 2.7.1: Allowed Actions **shall** be used to define signal processing on diagram space between signals.

CAR#2 RULE

There **shall** be exactly one space separation between *FIRST* markers and name of first input signal.

CAR#3 RULE
Input Interface signal **shall** be connected only as input of another signal or action, i.e. that kind of signal **shall not** have any source in form of another signal or action within scope of diagram.

CAR#4 RULE
Output Interface signal **shall** be connected only as output of another signal or action, i.e. that kind of signal **shall not** have any consumer in form of another signal or action within scope of diagram.

CAR#5 RULE
Local Parameters signal can be connected as input or output of another signals or action, however that kind of signal **shall have** other source and consumer in form of another signal or action within scope of diagram.

CAR#6 RULE
Any signal **shall** have only one source, apart from Input Interface signals, which **shall not** have any source within scope of diagram.

Data Flow and Integration Rules

DFIR#1 RULE
Please keep in mind that MCG CC recognizes data flow between two different components basing on naming convention, i.e. signals on both sides (output in one component and input in another component) **shall** have same name and type. Therefore please select name of signals and their types carefully.

Naming Convention Rules

NCR#1 RULE

Names of model elements **shall** contain only allowed characters listed in Table 2.6.1: Allowed Characters.

NCR#2 RULE
Names of model elements **shall** start with upper or lower case letters only.

NCR#3 RULE
Names of model elements **shall not** contain white spaces.

2.6 Naming Convention

Table 2.6.1: Allowed Characters defines all allowed characters in name of any model element:

Table 2.6.1: Allowed Characters

Character type	Allowed characters
Upper case letters	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Lower case letters	a b c d e f g h i j k l m n o p q r s t u v w x y z
Digits	1 2 3 4 5 6 7 8 9 0
Special characters	_ -

2.7 List of Actions

Table 2.7.1: Allowed Actions defines list of allowed interactions (actions) between signals on diagram space of component element.

Table 2.7.1: Allowed Actions

Action type	Definition	*FIRST* marker needed
ADD	Addition arithmetic operation. Requires at least two input signals[1)]. Requires exactly one output signal[2)].	
SUB	Subtraction arithmetic operation. Requires at least two input signals[1)]. Requires exactly one output signal[2)].	Yes

1) Input Interface or Local Parameters signal could be connected as input signal to that action.
2) Local Parameters or Output Interface signal could be connected as output signal from that action.

3 Mod Code Generator (MCG)

4 Modeling example with code generation