**The lecture 11**

**Working with Data Types**

The term *data type* refers to the way in which a computer represents numbers in memory. A data type determines the amount of storage allocated to a number, the method used to encode the number’s value as a pattern of binary digits, and the operations available for manipulating the type. Most computers provide a choice of data types for representing numbers, each with specific advantages in the areas of precision, dynamic range, performance, and memory

usage. To enable you to take advantage of data typing to optimize the performance of MATLAB programs, MATLAB allows you to specify the data types of MATLAB variables. Simulink builds on this capability by allowing you to specify the data types of Simulink signals and block parameters. The ability to specify the data types of a model’s signals and block parameters is particularly useful in real-time control applications. For example, it allows a Simulink model to specify the optimal data types to use to represent signals and block parameters in code generated from a model by automatic code-generation tools, such as Real-Time Workshop available from The MathWorks. By choosing the most appropriate data types for your model’s signals and parameters, you can dramatically increase performance and decrease the size of the code generated from the model.

Simulink performs extensive checking before and during a simulation to ensure that your model is *typesafe*, that is, that code generated from the model will not overflow or underflow and thus produce incorrect results. Simulink models that use the default data type (double) are inherently typesafe. Thus, if you never plan to generate code from your model or use a nondefault data type in your models, you can skip the remainder of this section.

On the other hand, if you plan to generate code from your models and use nondefault data types, read the remainder of this section carefully, especially the section on data type rules. In that way, you can avoid introducing data type errors that prevent your model from running to completion or simulating at all.

**Data Types Supported by Simulink**

Simulink supports all built-in MATLAB data types except int64 and uint64. The term *built-in data type* refers to data types defined by MATLAB itself as opposed to data types defined by MATLAB users. Unless otherwise specified, the term data type in the Simulink documentation refers to built-in data types. The following table lists the built-in MATLAB data types supported by Simulink.



Besides the built-in types, Simulink defines a boolean (1 or 0) type, instances of which are represented internally by uint8 values. Many Simulink blocks also support fixed-point data types. If the documentation for a block does not specify a data type, the block inputs or

outputs only data of type double.

**Fixed-Point Data**

Simulink allows you to create models that use fixed-point numbers to represent signals and parameter values. Use of fixed-point data can reduce the memory requirements and increase the speed of code generated from a model. To simulate a fixed-point model, you must have the Simulink Fixed Point product installed on your system. If Simulink Fixed Point is not installed on your system, you can simulate a fixed-point model as a floating-point model by

enabling automatic conversion of fixed-point data to floating-point data during simulation.

If you do not have Simulink Fixed Point installed and do not enable automatic conversion of fixed-point to floating-point data, Simulink displays an error when you try to simulate a fixed-point model.

You can edit a model containing fixed-point blocks without Simulink Fixed Point. However, you must have Simulink Fixed Point to

**•**Update a Simulink diagram (**Ctrl+D**) containing fixed-point data types

**•**Run a model containing fixed-point data types

**•**Generate code from a model containing fixed-point data types

**•**Log the minimum and maximum values produced by a simulation

**•**Automatically scale the output of a model using the autoscaling tool

**Fixed-Point Settings Interface**

Most of the functionality in the Fixed-Point Settings interface is for use with

the Simulink Fixed Point product. However, even if you do not have Simulink

Fixed Point, you can use the Fixed-Point Settings interface to perform a data

type override that allows you to work with a fixed-point model.

If you do not have Simulink Fixed Point, you can work with a model containing

Simulink blocks with fixed-point settings by doing the following:

**1** Access the **Fixed-Point Settings** interface from the model by selecting

**Tools -> Fixed-Point Settings**.

**2** Set the **Logging mode** parameter to Force off model wide.

**3** Set the **Data type override** parameter to True doubles or True singles

model wide.

This procedure allows you to share fixed-point Simulink models among people

in your company who may or may not have Simulink Fixed Point.

**Specifying Block Parameter Data Types**

When entering block parameters whose data type is user-specifiable, use the syntax type(value) to specify the parameter, where type is the name of the data type and value is

the parameter value. The following examples illustrate this syntax.



You can specify any MATLAB built-in data type supported by Simulink as the data type of a parameter. You cannot specify fixed-point data types as parameter data types.

**Creating Signals of a Specific Data Type**

You can introduce a signal of a specific data type into a model in any of the following ways:

**•**Load signal data of the desired type from the MATLAB workspace into your model via a root-level inport or a From Workspace block.

**•**Create a Constant block in your model and set its parameter to the desired type.

**•**Use a Data Type Conversion block to convert a signal to the desired data type.

**Data Type Propagation**

Whenever you start a simulation, enable display of port data types, or refresh the port data type display, Simulink performs a processing step called data type propagation. This step involves determining the types of signals whose type is not otherwise specified and checking the types of signals and input ports to ensure that they do not conflict. If type conflicts arise, Simulink displays an error dialog that specifies the signal and port whose data types conflict.

Simulink also highlights the signal path that creates the type conflict.