	<b>Technical Information</b> <b>Entry Tool - Instruction</b>		<b>TI</b>	
Responsibility: <b>APR</b>	Date: <b>2006-08-28</b>	Language: <b>en</b>	Filing system : <b>0.6</b>	Revision: Page: <b>1/23</b>
Issued by:	Approved:	Released:	Area of validity: <b>ABB SW</b>	

## Title: Entry Tool – Instruction

**References:**


- [1] Requirement Specification <Subsystem Code Generator>
- [2] Requirement Specification <Hart Code Generator>
- [3] Requirement Specification <Arm Code Generator>
- [4] Requirement Specification <Entry Tool>
- [5] Requirement Specification <CSV File Format Description>
- [6] Design Description <Common Framework>

## Distribution

**Authors**      **Georg Horst, Martin Dahl** .....Date: **2006-08-28**.....  
**Remarks:**

## Contents

1	Introduction.....	2
2	Design Concept – Code Generating .....	2
3	Prepare Project.....	3
4	Configure the Microsoft .NET Framework.....	4
5	Control Data Entry .....	5
6	Common Subsystem Entry Tool.....	6
6.1	Open Project .....	6
6.2	Project Control .....	6
6.2.1	Open an existing Subsystem: .....	7
6.2.2	Create New Subsystem: .....	7
6.2.3	Delete an existing Subsystem: .....	7
6.2.4	Save Project:.....	7
6.3	Data Input.....	8
6.3.1	Create Subsystem Objects .....	8
6.3.2	View Object defined values.....	12
6.3.3	Create Subsystem Defines .....	12
6.3.4	Create Subsystem Dataclasses.....	13
6.3.5	Create Subsystem Methods.....	14
6.3.6	View Subsystem Information .....	15
6.3.7	Execute Code Generator .....	15
7	HART Commands Entry.....	16
7.1	Create HART Subsystem.....	16
7.2	Edit HART Commands.....	16
7.2.1	Define HART Command Names.....	17
7.2.2	Define HART Commands .....	18
7.2.3	Set HART Command Access Rights .....	19
7.2.4	View HART Command Access Rights.....	19
8	Custom T_DATA_OBJETCS.....	20
8.1	Location of the CSV files.....	20
8.2	Entries in t_data_object_constructor.csv .....	20
8.3	Entries in t_data_object_types.csv .....	21
9	Custom DataClasses.....	22
10	Revision Chart .....	23

	<b>Technical Information</b> <b>Entry Tool - Instruction</b>		<b>TI</b>	
Responsibility: <b>APR</b>	Date: <b>2006-08-28</b>	Language: <b>en</b>	Filing system : <b>0.6</b>	Revision: <b>2/23</b>
Issued by:	Approved:	Released: <b>ABB SW</b>		Area of validity:

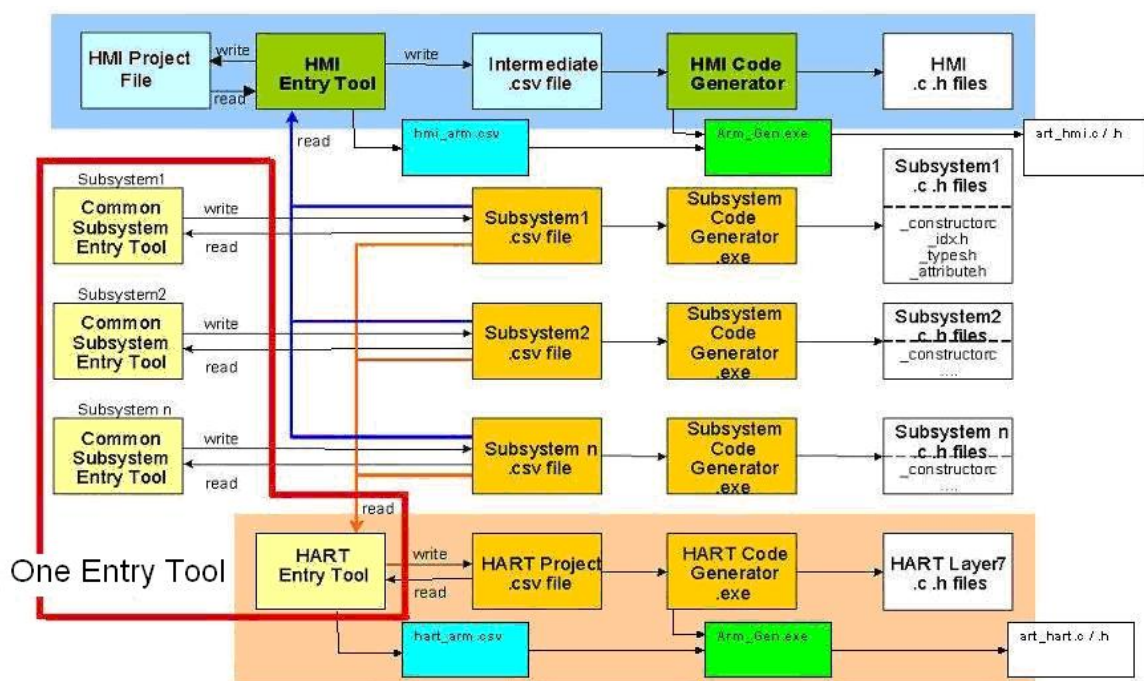
## 1 Introduction

The Entry Tool for the ABB Common Framework was developed with the intention to be able to create on a simple way the needed CSV- files for the SUBSYSTEM code generator, HART code generator and the ARM code generator and to control the user input to reduce mistakes.

To simplify the CSV- file creation the Entry Tool provides a graphical user interface on which the user can create his objects, methods, defines, hart - comandos and hart – access rights. Additionally it provides the possibility to execute the code generators.

This document describes how the project should be prepared to used the Entry Tool (the Subsystem and Hart Entry Tool) with the code generators and it comprised a tutorial how the user can create the needed subsystem data.

## 2 Design Concept – Code Generating



As you see in the picture above, there are at the moment four code generators available (HMI-, Subsystem-, HART- and ARM code generator). All these generators expect several CSV – files as inputs and generate several .c and .h files as outputs. The Entry Tool is able to read and write these files.

Each CSV – file must satisfy the CSV file Format description as defined in [5] and must have the file structure as defined in the requirement documents [1] or [2] or [3].

<b>ABB</b>	<b>Technical Information</b> <b>Entry Tool - Instruction</b>		<b>TI</b>	
Responsibility: <b>APR</b>	Date: <b>2006-08-28</b>	Language: <b>en</b>	Filing system : <b>0.6</b>	Revision: Page: <b>3/23</b>
Issued by:	Approved:	Released: ABB SW		Area of validity:

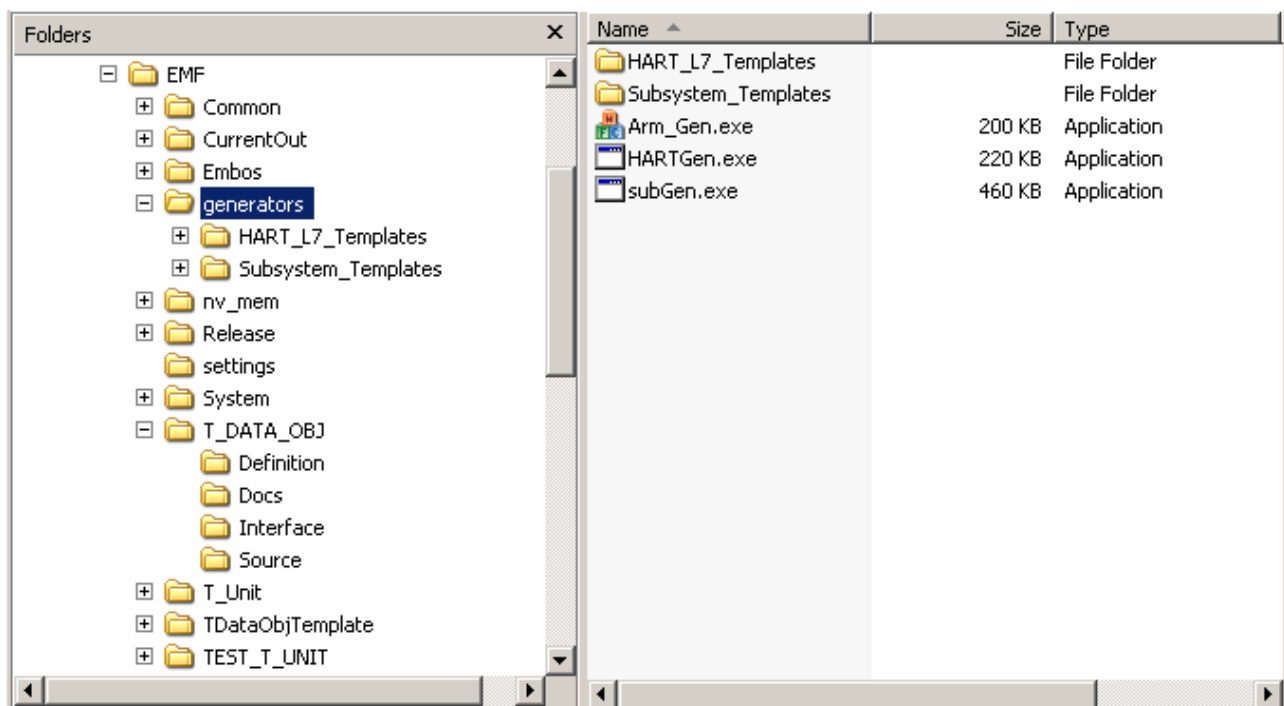
### 3 Prepare Project

The current available version 2.1.1 of the Framework is available at Sharepoint and supports the code generator that is also needed for the Entry Tool.

To use the Entry Tool with the code generators a project- folder must be prepared as follow:

1. Download the Framework Software from Sharepoint and extract the Zip Archive
2. Rename the Framework folder as you need e.g. EMF
3. The actual Framework Version does not include a **generators** folder. This folder should contain all needed code generators.  
Create a **generators** folder in the Framework path.  
(If this folder does not exists you are not able to use the Entry Tool)
4. Download the subsystem code generator SubGen (actual version 2.1.1) from Sharepoint, unzip it and insert it in the **generators** folder
5. Download the HART code generator HartGen (actual version 1.0.0) from Sharepoint , unzip it and insert it in the **generators** folder
6. Download the Arm code generator Arm\_Gen (actual version 1.0.0) from Sharepoint , unzip it and insert it in the **generators** folder
7. If you already have subsystems with created object and methods in CSV-file format (e.g. CurrentOut), than you are able to copy these folders directly to your project path. (Don 't forget, that each Subsystem must includes a **definition** folder which contains all CSV - files)
8. Donwload the Entry Tool (version 1.0) from Sharepoint and extract the Zip Archive.  
The Entry Tool could lie on any path in the Windows file system e.g. also in the **generators** folder.
9. The Entry Tool was developed with the Microsoft .NET Framework 1.1. Without this MS- Framework the Entry Tool could not be executed. If this Framework not exists on your Windows System, download it also from Sharepoint and install it on your PC.

After these steps above the project folder should look like this:



<b>ABB</b>	<b>Technical Information</b> <b>Entry Tool - Instruction</b>		<b>TI</b>	
Responsibility: <b>APR</b>	Date: <b>2006-08-28</b>	Language: <b>en</b>	Filing system : <b>0.6</b>	Revision: Page: <b>4/23</b>
Issued by:	Approved:	Released: ABB SW		Area of validity:

#### 4 Configure the Microsoft .NET Framework

The Entry Tool could stand on any path on the Windows local drives without any necessary .NET Framework configurations.

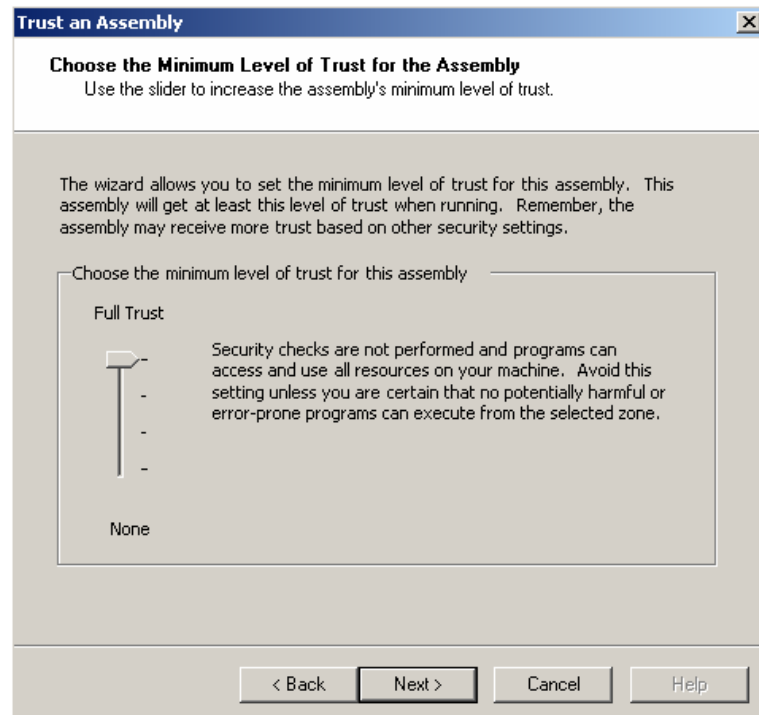
But if the user installed the Entry Tool on a network folder and execute it, a **.NET Security Problem** will arise. This is a part of Microsoft Security Offensive to ban Programs access to user's data.

To avoid this error message the .NET Framework should be configured as follow:


- Open the Windows Control Panel
- Open Administrative Tool
- Open the Microsoft .Net Framework 1.1 Wizards
- Call the Wizards tool "Trust An Assembly"



- Select then the path to the Entry Tool
- Set the Entry Tool trust to "Full Trust" and click on "Next" and than on "Finish"



After this configuration steps you can use the Entry Tool also it is located on network drives.

	<b>Technical Information</b> <b>Entry Tool - Instruction</b>		<b>TI</b>	
Responsibility: <b>APR</b>	Date: <b>2006-08-28</b>	Language: <b>en</b>	Filing system : <b>0.6</b>	Revision: Page: <b>5/23</b>
Issued by:	Approved:	Released: ABB SW		Area of validity:

## 5 Control Data Entry

The Entry Tool controls some user data entries like:

2. If the user tries to define two objects with the same name, the Entry Tool will show an error message and remove the value.
3. If the user tries to define two dataclasses with the same name, the Entry Tool will show an error message and remove the value.
4. If the user tries to define two defines with the same name, the Entry Tool will show an error message and remove the value.
5. Some entries only accept numbers, e.g.:
  - objectCount for Object definition
  - parameterIndex for Methods definition
  - command for Hart Command Names definition
  - slotCode for Hart Command definition
  - order for Hart Command definition

The Entry Tool will show an error message and remove the value if the value is not a number.
6. The Common Framework does not supports arrays of T\_DATA\_OBJECT structure Types, this will be checked by the Entry Tool. If the user tries to define arrays of structure types an error message will be shown as well and the objectCount will be set to 1.
7. Previously defined object defaults, display or descriptor Values would be removed, if the T\_DATA\_OBJECT changed.
8. If the objectCount reduced, e.g. from 20 to 10, than would the last ten previously defined object default values removed.
9. Previously user defined dataclasses that are used by object definition will be automatically removed if the dataclass name change.
10. At reading CSV files, the Entry Tool checks the values in the Columns, which could only contain predefined entries (predefined values can be chosen by user from combo boxes). If the entries do not match with the values in the combo boxes the Entry Tool will remove these entries and show a warning message.
11. If a Subsystem missed some needed CSV- files, than the Entry Tool will create them automatically.

<b>ABB</b>	<b>Technical Information</b> <b>Entry Tool - Instruction</b>		<b>TI</b>	
Responsibility: <b>APR</b>	Date: <b>2006-08-28</b>	Language: <b>en</b>	Filing system : <b>0.6</b>	Revision: Page: <b>6/23</b>
Issued by:	Approved:	Released: ABB SW		Area of validity:

## 6 Common Subsystem Entry Tool

### 6.1 Open Project

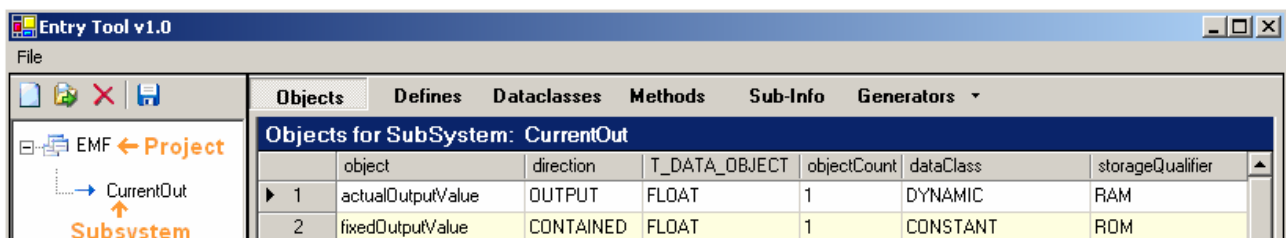
After starting the Entry Tool, you can open a project by click on the Menu Item **File** and then click **Open Project....** Select your Project path and click **OK**.

The Entry Tool checks the user selected path, that means if the chosen folder doesn't include the Framework specified folders like **T\_DATA\_OBJ**, **T\_UNIT**, **System**, **Common** or **generators**, then the Entry Tool shows an error message and the project would not be opened.  
If the path is ok, the Entry Tool will automatically load all existing Subsystem Definition files as well HART & ARM.

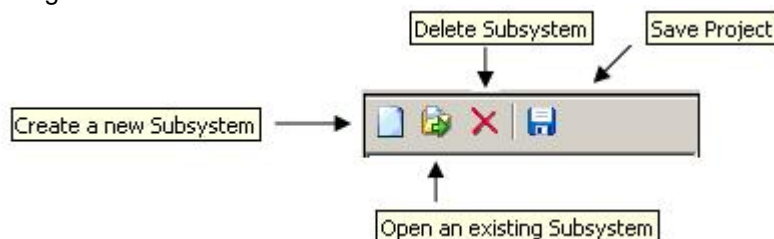
If a project was previously opened you can additionally open it by click on the Menu Item **File** and select between all previously opened projects.

### 6.2 Project Control

After a project was successfully opened the user will see the following surface:



To create a new subsystem, open an existing subsystem or to delete an existing subsystem the user can choose the corresponding Toolbar Buttons:



The user can also add new or open existing Subsystems by click with the right mouse button on the Project name in the tree view (a context menu will pop up).

If you click with the right mouse button on a Subsystem node in the tree view, then you can additionally select from a context menu between **Delete Subsystem** and **Generating** (e.g. execute code generator SubGen).

Additional to the Toolbar- button **Save Project** a project can be saved by click on the Menu Item **File** and then click on **Save Project** or click with the right mouse button on the Project name in the tree view (a context menu will pop up).

<b>ABB</b>	<b>Technical Information</b> <b>Entry Tool - Instruction</b>		<b>TI</b>	
Responsibility: <b>APR</b>	Date: <b>2006-08-28</b>	Language: <b>en</b>	Filing system : <b>0.6</b>	Revision: Page: <b>7/23</b>
Issued by:	Approved:	Released:	Area of validity: <b>ABB SW</b>	

### 6.2.1 Open an existing Subsystem:

You can choose between the Subsystems in the selected Framework folder. The selected subsystem will be added (if is not already opened) with all its data to the Entry Tool.

### 6.2.2 Create New Subsystem:

If **Create new Subsystem** was selected, this Dialog box will pop up, where some general Subsystem information must be entered.

The meaning of these entries:

- **Subsystem Name:** The name of the folder that is created for the subsystem as well as the filenames for the different csv files. Additionally, the name by which references are entered to the subsystem throughout the entry tool.
- **Short:** This name must be a valid "C" identifier, used for prefixing in the generated code. May be identical to the long Subsystem Name.
- **Framework Version:** This indicates the revision of the Framework the subsystem will be designed for. The code generators will check this information to assure that proper code is generated for the specific version. Must be entered in hexadecimal from 0xAABBCC, where AA is the Main revision.

**Create a new Subsystem**

The following Subsystem Data is needed:

Subsystem Name:

Short Subsystem Name:


Framework Version:

### 6.2.3 Delete an existing Subsystem:

By choosing delete Subsystem the Subsystem folder with all its data would be removed to the Recycle Bin.

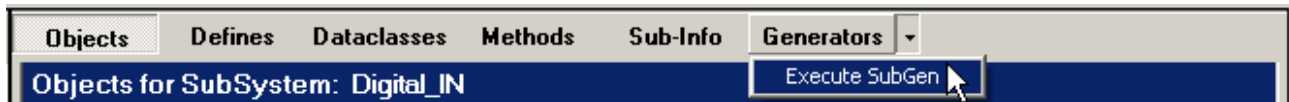
### 6.2.4 Save Project:

If **Save Project** was selected the Entry Tool checks all Subsystem CSV- files for changes. If changes were found, the Entry Tool would save the corresponding CSV- file.

		<b>Technical Information</b> <b>Entry Tool - Instruction</b>			<b>TI</b>	
Responsibility:	Date:	Language:	Filing system :	Revision:	Page:	
APR	2006-08-28	en		0.6	8/23	
Issued by:	Approved:	Released:	Area of validity:			
ABB SW						

### 6.3 Data Input

To create data for a Subsystem the user has to select between the toolbar buttons:



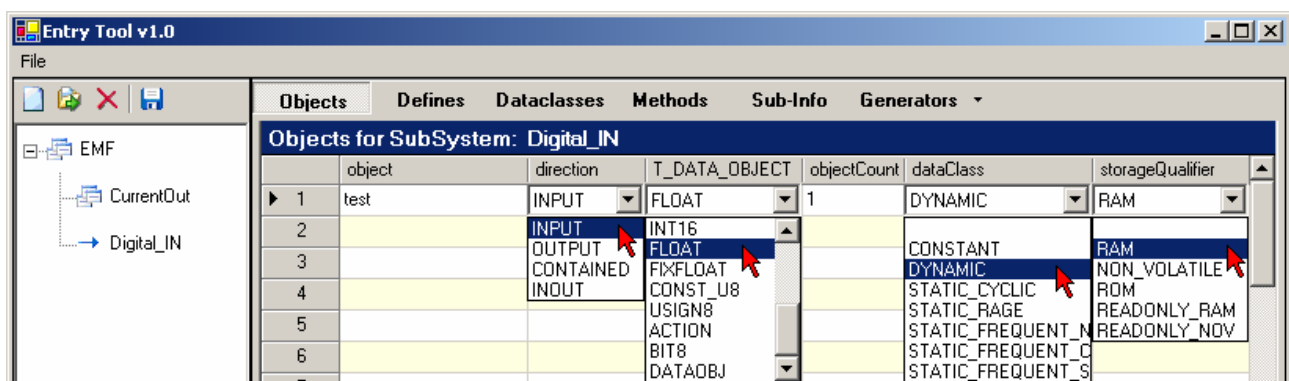
#### 6.3.1 Create Subsystem Objects

To create or edit Subsystem objects click on the toolbar button objects.

The Entry Tool provides the Subsystem data on a table based view. The meaning of each column entry for the object creation is:


- **Object:** Name of the Object, must be a valid "C" identifier
- **direction:** This is an informative field, indicating whether this object is a pure input/output parameter or used as inout. Additionally, inout parameters that are not updated by the device without external trigger should be flagged as contained.
- **T\_DATA\_OBJECT:** All types of possible T\_DATA\_OBJECTS can be selected from the dropdown menu. Custom types can be defined, see section "Custom T\_DATA\_OBJECTS".
- **objectCount:** The number of items , i.e. 1 for single object 2..n for arrays. Note: Structured T\_DATA\_OBJECTS (T\_DATA\_OBJECTS that have attributes) cannot be declared as arrays (this is a framework limitation).
- **dataClass:** A dataclass groups objects that have the handling in certain conditions, such as device startup, reset to default values etc. Custom dataclasses can be defined, see section "Custom DataClasses". For an in-depth discussion see the documentation associated with the Common Framework.
- **storageQualifier:** Indicates how the object is stored.

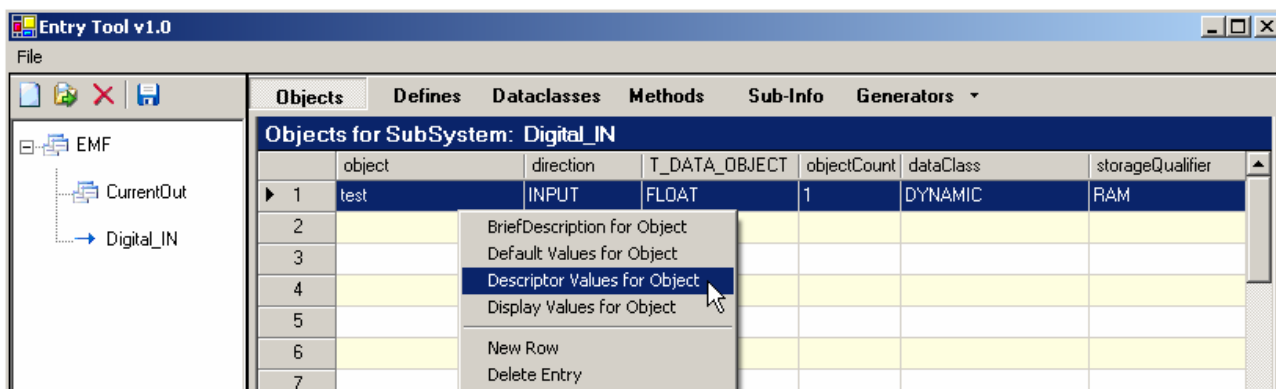
For columns which must contain predefined values, you can select these values from a Combo Box as shown in the picture below:



After the user has entered all the values for an object in the row and he wants to enter additional information for brief description, the default values, descriptor values or display values for an object then he has to click with the right mouse button on the row in which the object is defined. A context menu like this will pop up:



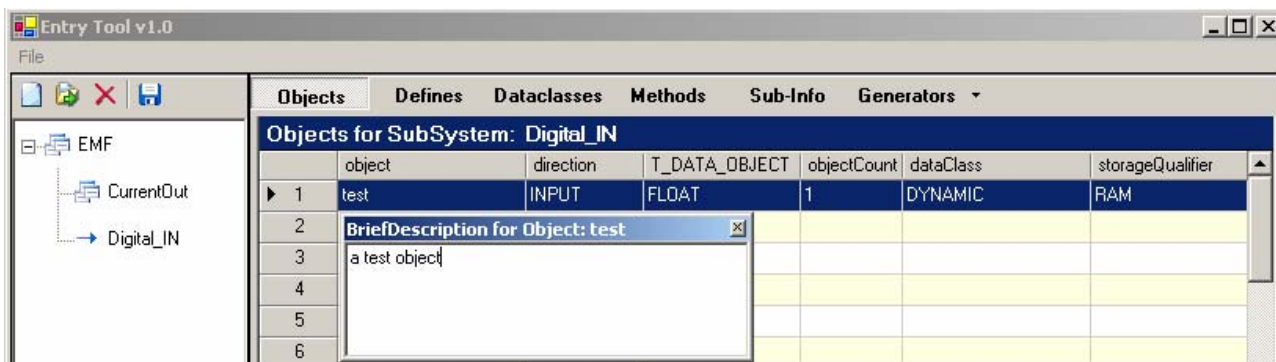
	<b>Technical Information</b> <b>Entry Tool - Instruction</b>		<b>TI</b>	
Responsibility: <b>APR</b>	Date: <b>2006-08-28</b>	Language: <b>en</b>	Filing system : <b>0.6</b>	Revision: <b>9/23</b>
Issued by:	Approved:	Released: <b>ABB SW</b>		Area of validity:



To set the values just select between the context menu items.

→ **BriefDescription for Object:** A comment that will be embedded into the generated source code. The comment describes the purpose of the object in a way that an interested user must not have a look at the source code to gain an overview about the object's use.

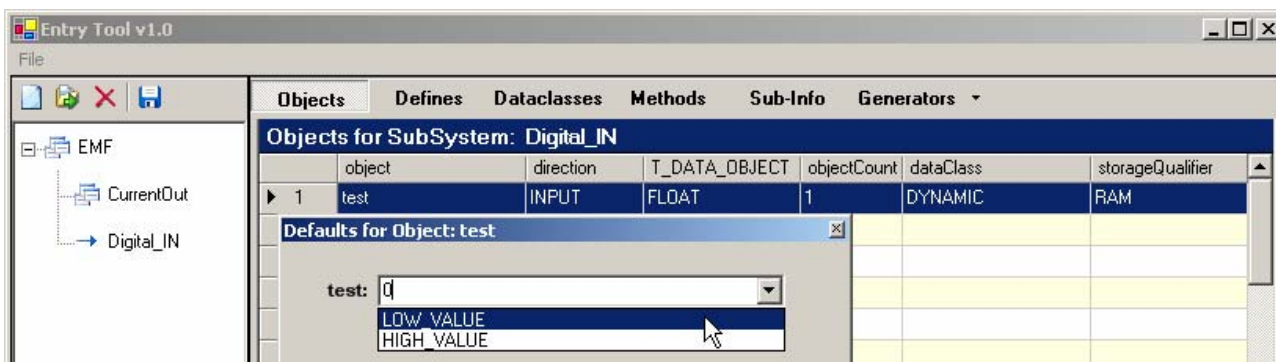
After selecting this Context Menu item the Entry Tool will show the following dialog:




→ **Default Values for Object:** The defaults for each T\_DATA\_OBJECT (for simple, array and structure types). If the created Object is a structure Type, than the Entry Tool will automatically show the user all structure attributes by selecting this menu item.

After selecting this Context Menu item the user has to set the default value for this object:

It is possible to set any user defined value or select between the subsystem defines (defines must be set first under Toolbar button **Defines**).

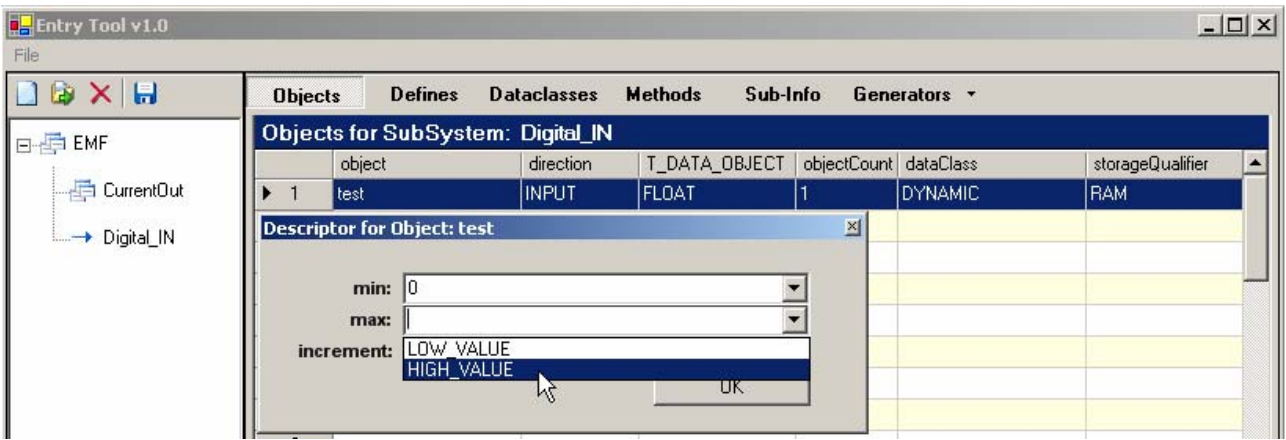



		<b>Technical Information</b> <b>Entry Tool - Instruction</b>			<b>TI</b>	
Responsibility:	Date:	Language:	Filing system :	Revision:	Page:	
<b>APR</b>	<b>2006-08-28</b>	<b>en</b>		<b>0.6</b>		
Issued by:	Approved:	Released:	Area of validity:			
<b>ABB SW</b>						

→ **Descriptor Values for Object:** Set the default values for the descriptor of the T\_DATA\_OBJECT. The Entry Tool automatically searches for the descriptor and shows all attributes after selecting this menu item. This context menu is not accessible if the T\_DATA\_OBJECT has no descriptor.

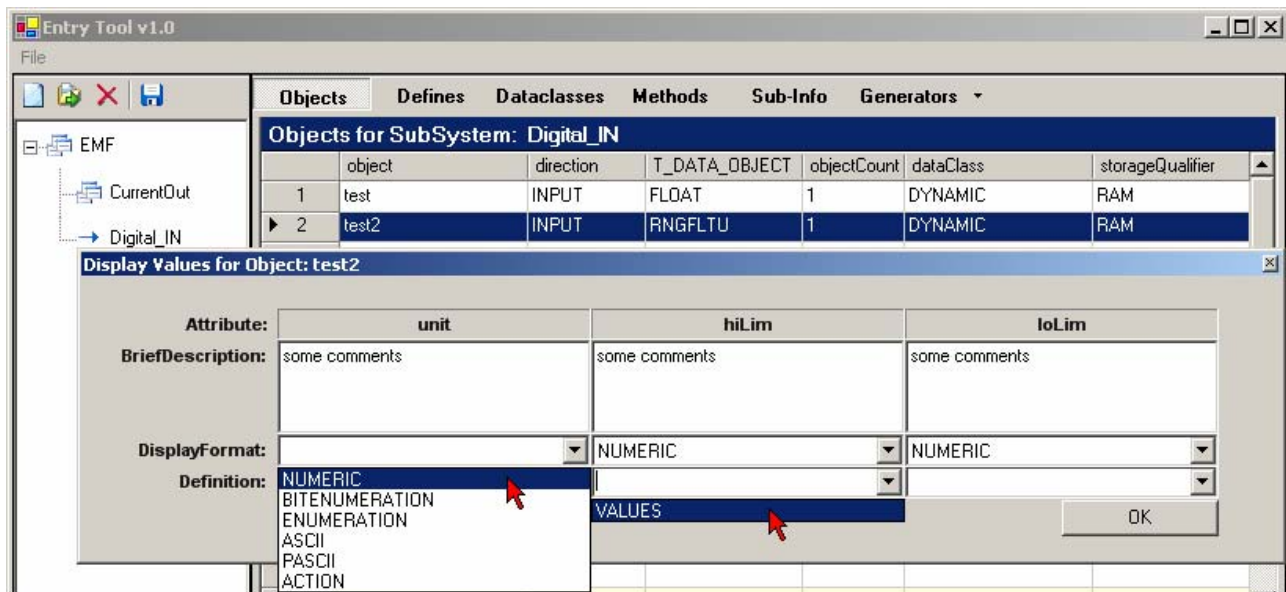
It is possible to set any user defined value or select between the subsystem defines (defines must be set first under Toolbar button **Defines**). For an in-depth discussion of descriptors please see the documentation associated with the Common Framework.

After selecting this context menu item the Entry Tool will search for the Descriptor Type of the T\_DATA\_OBJECT and call on the user to set the values, like this:



	<b>Technical Information</b> <b>Entry Tool - Instruction</b>				<b>TI</b>	
	Responsibility:	Date:	Language:	Filing system :	Revision:	Page:
	<b>APR</b>	<b>2006-08-28</b>	<b>en</b>		<b>0.6</b>	
Issued by:		Approved:		Released:		Area of validity:
				ABB SW		

→ **Display Values for Object:** The display values provide detailed information about individual objects/attributes and its associated definition in case of enumerated and bit-enumerated data. If the created Object is a structure Type, than the Entry Tool will automatically show the user all structure attributes by selecting this menu item as shown in the picture below:




The meaning of the needed entries for each attributes:

- **Attribute:** Each attribute is listed individually (for structured T\_DATA\_OBJECTS)
- **BriefDescription:** Provide a brief description of this value, i.e. what information is transported in case one chooses to display this attribute to a user. (This will be most important for translations)
- **DisplayFormat:** Indicates the class of the attribute, select from the dropdown list
- **Definition:** Provide an easy to follow link to the subsystems definitions, used by other code generators. This link is important for bitenumerated values as well as for enumerated values. (Definitions must be specified first under Toolbar button **Defines**).

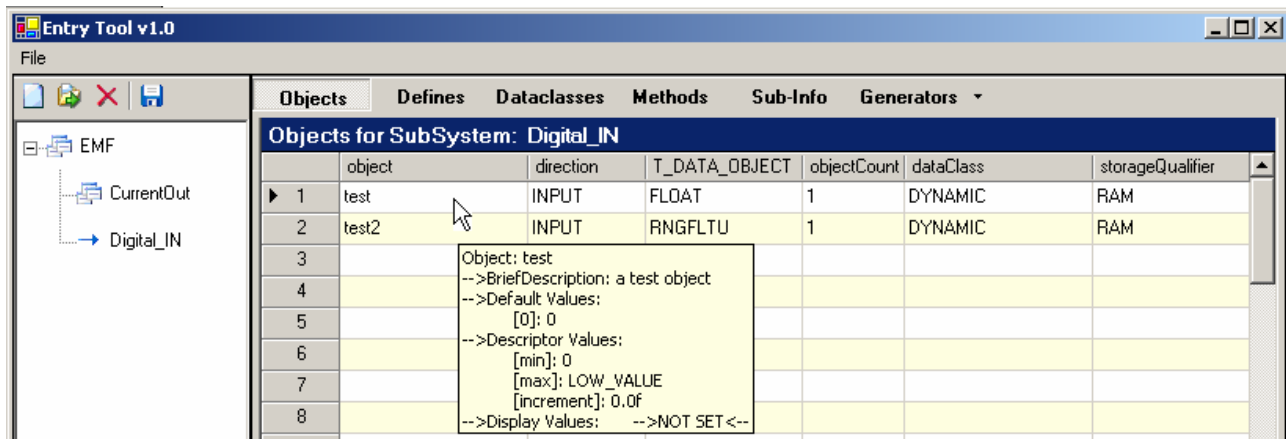
→ **New Row:** A new row for a new object definition will be inserting in the table at the mouse position. A new row can also be created by clicking in the last row (the row with the star on the left side).

→ **Delete Entry:** The current Row will be deleted and also all default, descriptor and display values for a defined object.

		<b>Technical Information</b> <b>Entry Tool - Instruction</b>			<b>TI</b>	
Responsibility:	Date:	Language:	Filing system :	Revision:	Page:	
<b>APR</b>	<b>2006-08-28</b>	<b>en</b>		<b>0.6</b>		
Issued by:	Approved:	Released:	Area of validity:			
<p style="text-align: center;">ABB SW</p>						

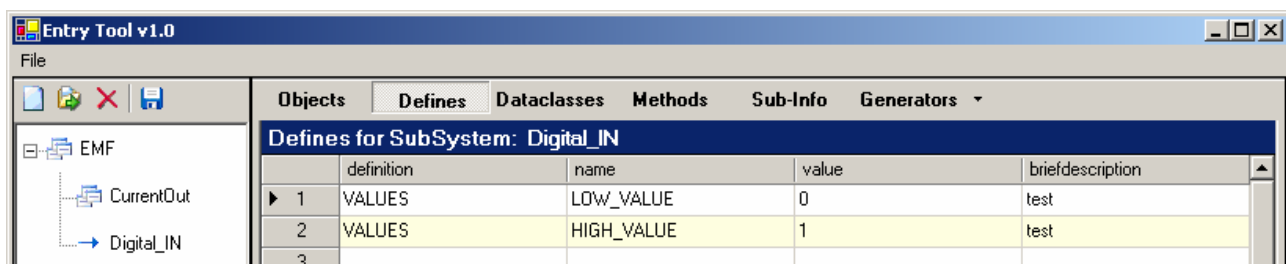
### 6.3.2 View Object defined values

To look which values are already set for the object, the user must just hold with the mouse pointer on the object name. After a short time a Tool tip box will pop up and show all user defined entries for Object brief description, defaults, descriptor and display values.



### 6.3.3 Create Subsystem Defines


This table serves the purpose of providing Subsystem public definitions and symbolic constants. This is typically used for enumerations and bit fields. To set defines the user has to click on the toolbar button **Defines**. The following surface will be show:



All Enumerations and Defines that was created in this Table can be used for Object default, descriptor and display values.

The meaning of each column entry for Defines is:

- **definition:** This name encapsulates definitions into logical groups, i.e. bitenumerated and enumerated values should all have the same entry for "definition". Must be a valid "C" identifier.
- **name:** The "C" name of the definition. This name will be prefixed with short\_subsystem\_name by the code generators. Must be a valid "C" identifier.
- **value:** The actual value of the define. Must be a valid "C" identifier.
- **briefdescription:** Provides a short description of the define. Should have enough information for translation purposes.

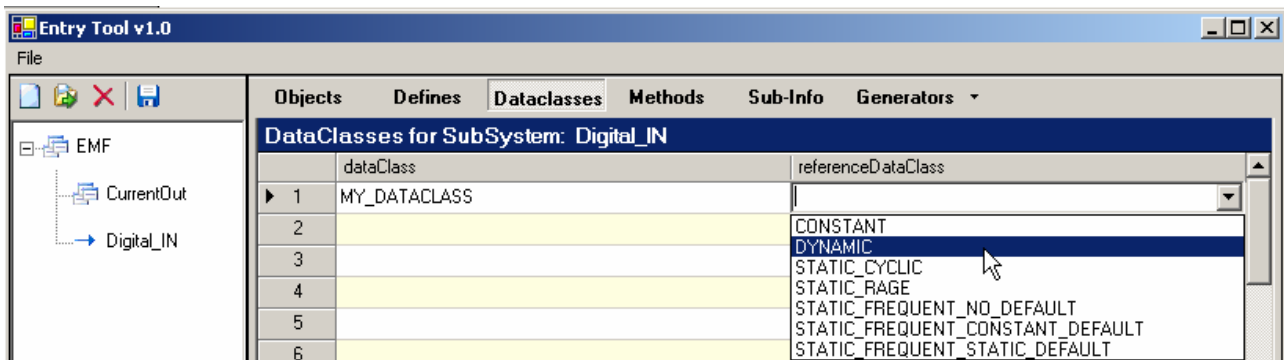
	<div>Technical Information</div> <div>Entry Tool - Instruction</div>				TI
Responsibility:	Date:	Language:	Filing system :	Revision:	Page:
APR	2006-08-28	en		0.6	
Issued by:	Approved:	Released:	Area of validity:		
ABB SW					

### 6.3.4 Create Subsystem Dataclasses

Dataclasses provide a grouping for objects that require the same treatment in different situations, i.e. on device start up, during device reset etc. There is a set of predefined dataclasses that comes with the common framework itself. These predefined dataclasses could be used without additional setup of any kind. Additionally instances of predefined dataclasses can quickly be created for two reasons:

1. There is more than one physical location of objects in a device (e.g. a sensor and main memory).
2. You want to use different names for the default dataclasses.

To define subsystem dataclasses the user has to click on the toolbar button **Dataclasses**.




The meaning of each column entry for Defines is:

- **dataClass:** The name of the dataclass as used in the subsystem. Must be a valid "C" identifier.
- **referenceDataClass:** This indicates the type of the actual dataclass that the new dataclass is an instance of. Custom Dataclasses may be entered as well, but they cannot be selected from the dropdown box. See section "Custom Dataclasses".

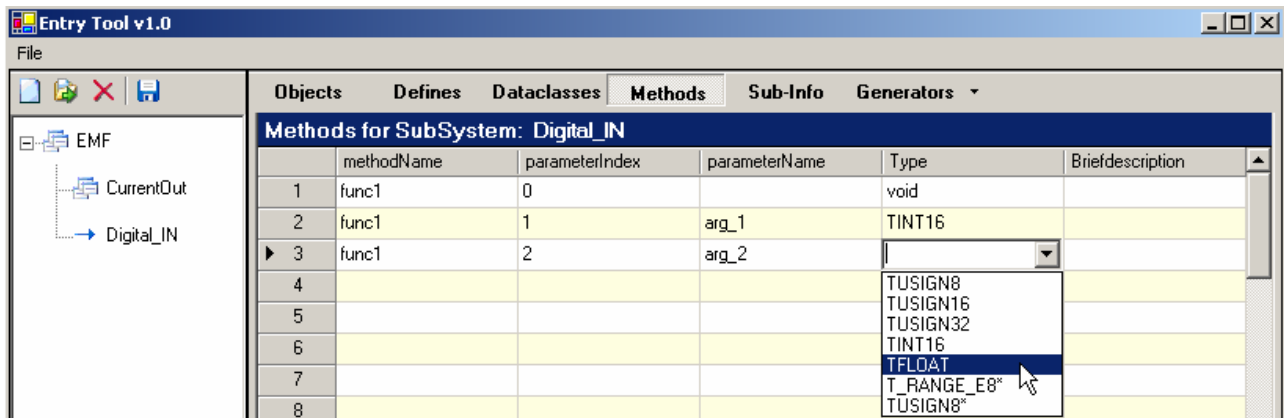
Once created DataClasses could be used for the subsystem's object definitions. It is possible to create multiple instances of the same reference dataclass, i.e. Dynamic1, Dynamic2 Dynamic3...

Note: It is highly recommended to use camel-notation for new dataclasses. i.e. use **"myDataclass"** instead of **"MY\_DATACLASS"** for the new dataclass name. The code generator will create appropriate code only from camel notations.

		<b>Technical Information</b> <b>Entry Tool - Instruction</b>			<b>TI</b>	
Responsibility:	Date:	Language:	Filing system :	Revision:	Page:	
<b>APR</b>	<b>2006-08-28</b>	<b>en</b>		<b>0.6</b>		
Issued by:	Approved:	Released:	Area of validity:			
		ABB SW				


### 6.3.5 Create Subsystem Methods

To define the Subsystem EXE and SRV methods the user has to click on the toolbar button **Methods**. The methods defined by the tooling are the public methods of the subsystem that are available through the T\_UNIT interface. There is no need to specify all of the subsystems private (hidden) methods.



The meaning of each column entry for Defines is:

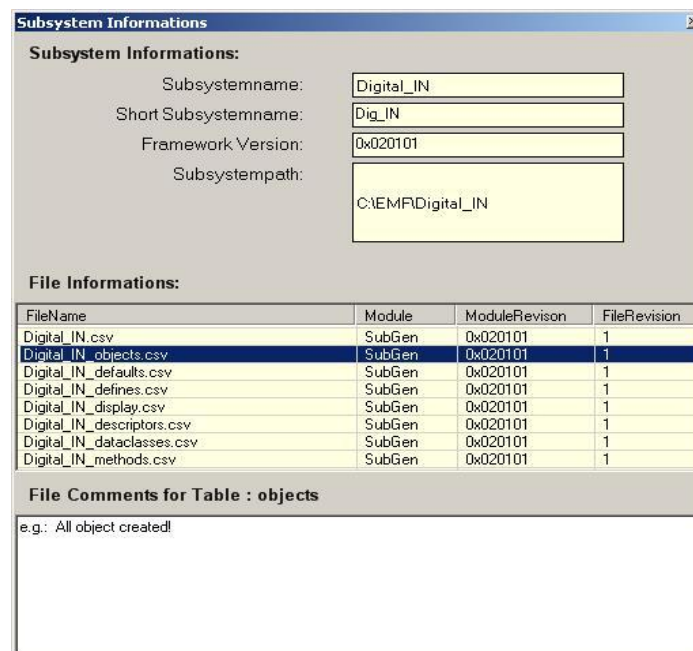
- **methodName:** The name of the method. Must be a valid "C" identifier.
- **parameterIndex:** The index of the parameter. Index 0 is the return type and must always be specified. Indices 2..n describe the actual parameters of the method.
- **parameterName:** Shall be omitted for parameterIndex 0 (return type), otherwise this is the name of the parameter. Must be a valid "C" identifier.
- **Type:** This is the actual type of the parameter (or the returntype for parameterIndex 0). It is possible to specify any "C" type, however you should use only the simple types defined by the common framework for a maximum of compatibility!
- **Briefdescription:** Gives a brief description of the parameter or – in case of parameterIndex 0 – a brief description for the whole method.

		<b>Technical Information</b> <b>Entry Tool - Instruction</b>		<b>TI</b>	
Responsibility:	Date:	Language:	Filing system :	Revision:	Page:
<b>APR</b>	<b>2006-08-28</b>	<b>en</b>		<b>0.6</b>	
Issued by:	Approved:	Released:	Area of validity:		
<p style="text-align: center;">ABB SW</p>					

### 6.3.6 View Subsystem Information

The toolbar button **Sub-Info** shows the global subsystem information's like the Name of the Subsystem , the folder path, etc...

Additionally the user can see all CSV- files that belongs to the Subsystem, and with which code generators these files would be created. The File-Revision for each CSV- File will be incremented with every save. For each CSV- file you can insert a comment. This comment will be saved in the Reserved Column of the CSV- file.



**Subsystem Informations**

**Subsystem Informations:**

Subsystemname:

Short Subsystemname:

Framework Version:

Subsystempath:

**File Informations:**

FileName	Module	ModuleRevision	FileRevision
Digital_IN.csv	SubGen	0x020101	1
Digital_IN_objects.csv	SubGen	0x020101	1
Digital_IN_defaults.csv	SubGen	0x020101	1
Digital_IN_defines.csv	SubGen	0x020101	1
Digital_IN_display.csv	SubGen	0x020101	1
Digital_IN_descriptors.csv	SubGen	0x020101	1
Digital_IN_dataclasses.csv	SubGen	0x020101	1
Digital_IN_methods.csv	SubGen	0x020101	1

**File Comments for Table : objects**

e.g.: All object created!

### 6.3.7 Execute Code Generator

Click on the toolbar button **Generators** and then click on **Execute SubGen** to call the subsystem code generator that creates the .c and .h source files.

The Code Generators checks the data input and if some errors, warnings or information was found then the Entry Tool will show them to the user.

<b>ABB</b>	<b>Technical Information</b> <b>Entry Tool - Instruction</b>		<b>TI</b>	
Responsibility: <b>APR</b>	Date: <b>2006-08-28</b>	Language: <b>en</b>	Filing system : <b>0.6</b>	Revision: Page:
Issued by:	Approved:	Released:	Area of validity: <b>ABB SW</b>	

## 7 HART Commands Entry

The Hart Subsystem includes all CSV –files as each other subsystem additive two CSV- Files for Hart Commands and one CSV- file for Hart Access Rights.

The entry of HART objects, defines, dataclasses and methods is the same as described in chapter 5.

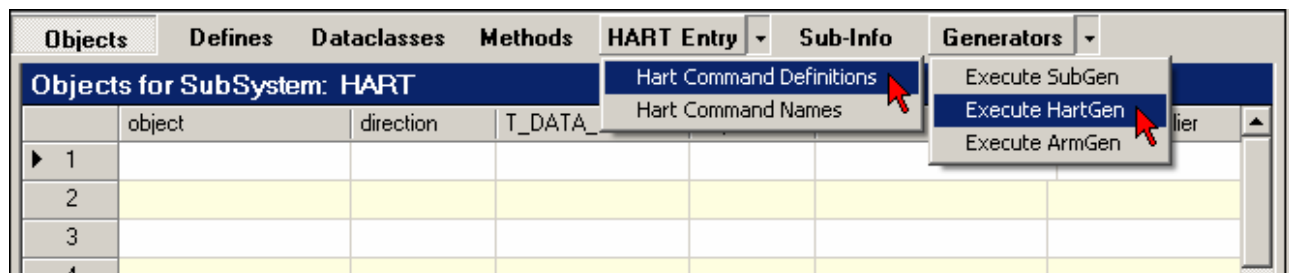
### 7.1 Create HART Subsystem

On Entry Tool start up it would be controlled if the HART subsystem with all his files exists (If not all files was found, the Entry Tool create the needed files automatically).

If the Hart subsystem wasn't found it could be created by click on **Create new Subsystem** as described in chapter 5, The Name of this Subsystem should be always "HART".

### 7.2 Edit HART Commands

After selecting the Hart Subsystem in the project tree view in the toolbar appears a button **HART Entry**. If the user clicks on this button he can select between entry **HART Command Definitions** and **HART Command Names** as shown below:




Also appears under the toolbar button **Generators** to new items:

- **Execute HartGen:** To execute the Code generator for the HART Commands
- **Execute ArmGen:** To execute the Code generator for the HART Command Access Rights

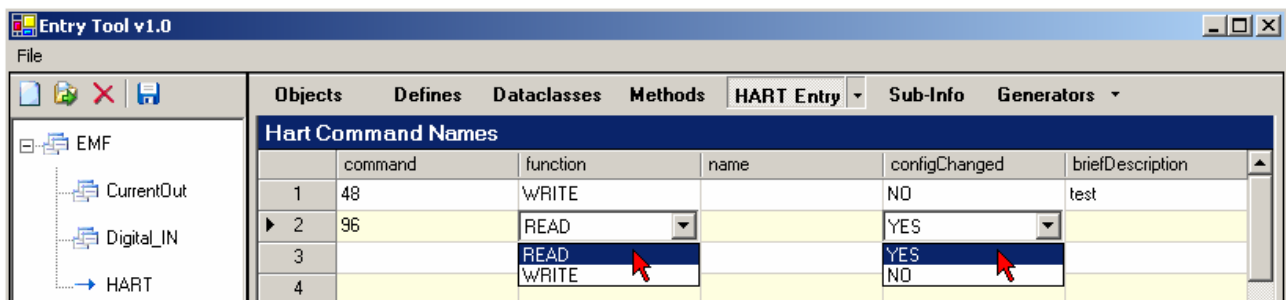
Additional to the toolbar button **Generators** the user can execute the needed code generator by click with the right mouse button on the Hart Subsystem in the project tree view.



		<b>Technical Information</b> <b>Entry Tool - Instruction</b>			<b>TI</b>	
Responsibility:	Date:	Language:	Filing system :	Revision:	Page:	
<b>APR</b>	<b>2006-08-28</b>	<b>en</b>		<b>0.6</b>		
Issued by:	Approved:	Released:	Area of validity:			
<b>ABB SW</b>						

### 7.2.1 Define HART Command Names

To define the HART Commands the user has to click on the toolbar button **HART Command Names**.



The meaning of each column entry for HART Command names is:

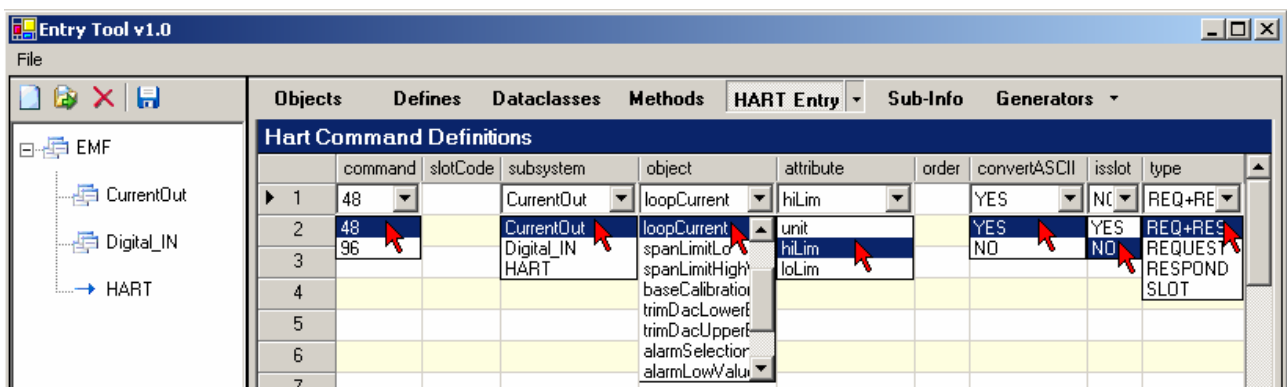
- **command:** The HART cmd number.
- **function:** Indicates whether this is a read or write command.
- **name:** This should be a short description that describes the purpose of the command.
- **configChanged:** This flag is required by the HART 1.0 subsystem. It indicates that this command would trigger a configuration changed flag.
- **briefDescription:** This provides a long description of the command.

Once created Hart Commands could be used for the HART Command Definitions.

## 7.2.2 Define HART Commands


To define the HART Commands the user has to click on the toolbar button **HART Command Definitions**.

Most of the entries in this the surface below the user can select from predefined values in the combo boxes. E.g. the user can select for the value in the subsystem column from all subsystems in your project, in the Object Column the user can then select between all objects from the subsystem that was been chosen before and if the T\_DATA\_OBJECT of the selected object is a structure type, then the Entry Tool will search for the structure attributes and make them available.



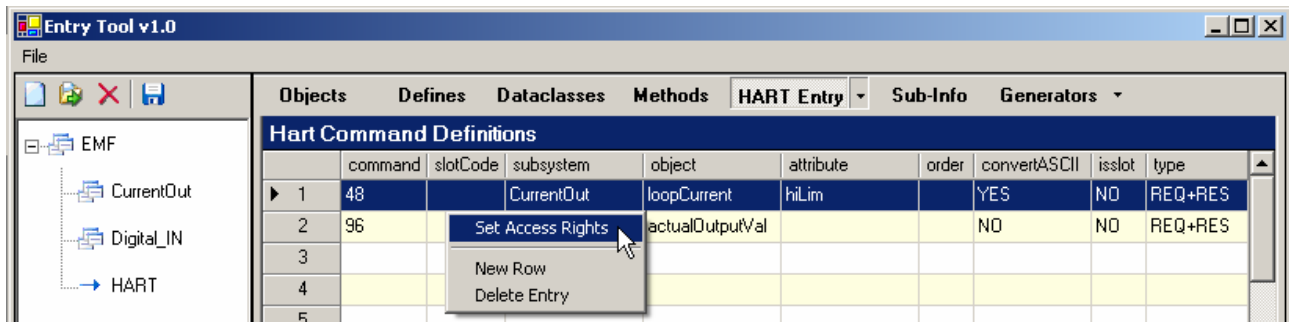
The meaning of each column entry for HART Command names is:

- **command:** The command number as defined in "Command\_Names"
- **slotCode:** required for Description of slotcodes only, this is a second level command description.
- **subsystem:** Any of the available Subsystems can be selected from the dropdown list.
- **object:** After selecting a subsystem you can select any of the subsystems objects.
- **attribute:** For structured T\_DATA\_OBJECTs each attribute must be specified individually. If the selected object is a structured one all of its attributes are selectable via the dropdown list.  
*This field is mandatory for these structured T\_DATA\_OBJECTs.*
- **order:** This indicates the logical position (0..n) of the object inside the command. The byte positions used for hart communications are calculated by the HART code generator. The order must be consecutively numbered inside a command.
- **convertASCII:** Flag that indicates that the object is a string stored with 8 bits in the device but should be communicated as packed ASCII via HART.
- **isslot:** indicates whether the object is a slot object. As of HART subsystem 1.0 this can only be the 4 predefined objects slot1..slot4 of the HART subsystem itself.
- **type:** This indicates the type of the description. Non-slot command always have a single description for requests as well as for responds, therefore the type is "REQ+RES". Slot commands require definition of the structure of the request and respond individually (even if both are identical). Additionally, slot commands require the definition of the individual slotcodes, these are using the "SLOT" type.

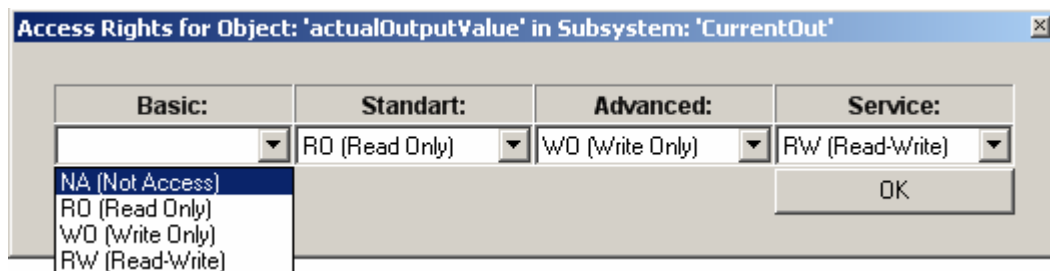
		<b>Technical Information</b> <b>Entry Tool - Instruction</b>			<b>TI</b>	
Responsibility:	Date:	Language:	Filing system :	Revision:	Page:	
<b>APR</b>	<b>2006-08-28</b>	<b>en</b>		<b>0.6</b>		
Issued by:	Approved:	Released:	Area of validity:			
ABB SW						

### 7.2.3 Set HART Command Access Rights

To select Access Rights for a Hart Command the user has just to click with the right mouse button on the Hart command and select **Set Access Rights**.



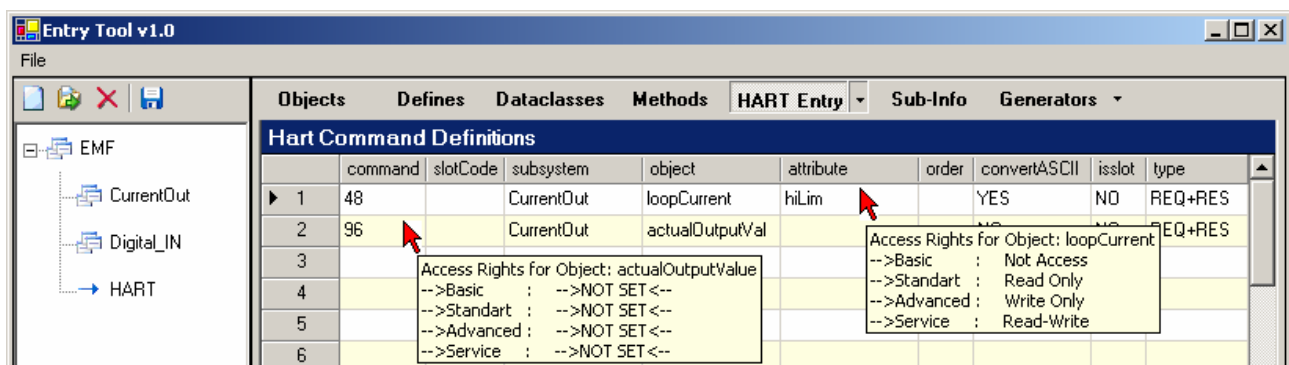
The following dialog will pop up, where the user can set the Access Rights:




Previously defined access rights would be removed, if the subsystem, the object or the attribute entry changed.

### 7.2.4 View HART Command Access Rights

To look which access rights are already set for a specific hart command, the user must point with the mouse pointer on the row. After a short time a Tool tip box will pop up and show all user defined access rights:



	<b>Technical Information</b> <b>Entry Tool - Instruction</b>		<b>TI</b>	
Responsibility: <b>APR</b>	Date: <b>2006-08-28</b>	Language: <b>en</b>	Filing system : <b>0.6</b>	Revision: Page:
Issued by:	Approved:	Released: ABB SW		Area of validity:

## 8 Custom T\_DATA\_OBJETCS

The tooling for the common framework is not limited to the T\_DATA\_OBJECTS that are provided with the framework. Any T\_DATA\_OBJECT can be made available to the tooling by means of providing the data structure and constructor to the tooling.

The following section does not describe how to code a new T\_DATA\_OBJECT but rather how to provide the linkage with the tooling.

The T\_DATA\_OBJECT TRANGEFLTU is already shipped with the framework but it will serve as an example for a custom dataclass. The relevant information is typically provided in the header file of the T\_DATA\_OBJECT, in this case "t\_data\_object\_rngflt.h":

```
typedef struct
{
    TFLOAT maxHi;
    TFLOAT minLo;
    const T_RANGE_E8 SLOW* ptrEnumTable;
} T_TRANGEFLTU_DESCRIPTOR;
```

```
typedef struct
{
    TUSIGN8 unit;
    TFLOAT hiLim;
    TFLOAT loLim;
} T_RANGE_FLT_U;
```

```
#define CONSTRUCT_STRUCT_RNGFLTU(w,y,z)
{ (void FAST*)(w),3,sizeof(T_RANGE_FLT_U), y, STRUCT, z, &Init_RNGFLTU }
```

The above code snippets contain all the information that is required for a T\_DATA\_OBJECT from the tooling's point of view, namely the name of the constructor, the type of the T\_DATA\_OBJECT itself and the type of its descriptor.

### 8.1 Location of the CSV files

T\_DATA\_OBJs may be defined in two ways: either globally and usable for the whole project or locally inside a subsystem – usable by that subsystem alone. The proper position of the CSV files for global T\_DATA\_OBJs

is "\T\_DATA\_OBJ\Definition" while local T\_DATA\_OBJs are defined in "\subsystem\Definition".


### 8.2 Entries in t\_data\_object\_constructor.csv

The constructor must be entered into t\_data\_object\_constructor.csv:

reserved	T_DATA_OBJECT	constructorName	ptrValue	numberOfAttributes	...
	RNGFLTU	CONSTRUCT_STRUCT_RNGFLTU	_GEN	_IGNORE	...

...	objectLength	storageQualifier	typeQualifier	ptrDescriptor	type	descriptorType
	_IGNORE	_GEN	_IGNORE	_GEN	T_RNGFLTU	T_RNGFLTU_RANGE

	<b>Technical Information</b> <b>Entry Tool - Instruction</b>		<b>TI</b>	
Responsibility: <b>APR</b>	Date: <b>2006-08-28</b>	Language: <b>en</b>	Filing system : <b>0.6</b>	Revision: Page:
Issued by:	Approved:	Released:	Area of validity: <b>ABB SW</b>	

The columns basically have a one-to-one relationship with the constructor's parameters (the constructor is expected to fill a `_T_DATA_OBJ` structure). Either static or generic information can be specified in the CSV file. The generic information can be one of the following strings:

- **\_IGNORE** means this parameter must not be passed to the constructor at all.
- **\_GEN** means that the code generator should provide a proper parameter. The actual parameter that is created by the code generator of `_GEN` depends on the column.

The individual columns should be provided with the following information:

- **constructorName** must be the same as provided by the header file
- **ptrValue** `_GEN` creates a pointer to the variable in the object's dataclass, usually `_GEN`
- **numberOfAttributes** `_GEN` creates the number of Attributes, usually `_IGNORE`
- **objectLength** `_GEN` passes the Number specified in `subsystem_constructor.csv`, usually `_IGNORE`
- **typeQualifier** `_GEN` passes the qualifier specified in `subsystem_constructor.csv`, usually `_GEN`
- **ptrDescriptor** `_GEN` creates a pointer to the automatically created descriptor, usually `_GEN`
- **type** must be a reference to a type that is available in `t_data_object_types.csv`.
- **descriptorType** must be a reference to a type that is available in `t_data_object_types.csv`.<sup>1</sup>

Note: Structured `T_DATA_OBJ` (such as the example) cannot be instantiated as `ARRAY`. However if the `T_DATA_OBJs` type is a simple type you can define two constructors in the table, one for the single instance constructor (typically `objectLength = _IGNORE`) and one for the `ARRAY` constructor (`objectLength = _GEN`).


### 8.3 Entries in `t_data_object_types.csv`

The type of the `T_DATA_OBJECT` and its descriptor must be entered into `t_data_object_types.csv`:

reserved	type	Attribute	basicType	externalType
	T_FIXFLOAT		TUSIGN32	TFLOAT
	T_RNGFLTU	Unit	TUSIGN8	TUSIGN8
	T_RNGFLTU	hiLim	TFLOAT	TFLOAT
	T_RNGFLTU	loLim	TFLOAT	TFLOAT
	T_RNGFLTU_DESCRIPTOR	maxHi	TFLOAT	TFLOAT
	T_RNGFLTU_DESCRIPTOR	maxLo	TFLOAT	TFLOAT
	T_RNGFLTU_DESCRIPTOR	ptrEnumTable	T_RANGE_E8*	T_RANGE_E8*

The above example defines a simple type 'T\_FIXFLOAT', a structured type 'T\_RNGFLTU' and its descriptor 'T\_RNGFLTU\_DESCRIPTOR' (see the code snippet in chapter 8 for a C representation of these types). Descriptors cannot have different basic/external types while `T_DATA_OBJECTS` can have external types. For example `T_FIXFLOAT` will be stored internally as `TUSIGN32` but are transformed to `TFLOAT` when communicated through the object's put/get methods.

<sup>1</sup> The name of both the type and descriptor type is a logical link only and the actual name is never referenced in any source code, therefore it may differ from the name of the type in the `T_DATA_OBJs` source code. However, for the sake of clarity you should use the same name in the source code and CSV file.

	<b>Technical Information</b> <b>Entry Tool - Instruction</b>		<b>TI</b>	
Responsibility: <b>APR</b>	Date: <b>2006-08-28</b>	Language: <b>en</b>	Filing system : <b>0.6</b>	Revision: Page:
Issued by:	Approved:	Released: <b>ABB SW</b>		Area of validity:

## 9 Custom DataClasses

The need to write a custom Dataclass will rarely ever arise as the entry tool provides easy access to multiple instances of the framework's predefined dataclasses.

If custom dataclasses are yet required, they must be created according to [6]. Additionally, to provide the linkage of custom dataclasses with the code generator, the subsystem templates must be enhanced with information on the new dataclasses. The templates of interest are located in the files 'Frame\_attributes.h' and 'Frame\_constructor.c' in the 'Generators\Subsystem\_Templates\Source' folder.

Locate the following template<sup>2</sup> in the file 'Frame\_attributes.h':

```
//@SubGen template start@<DATACLASS STATIC_FREQUENT_STATIC_DEFAULT>
//-----
//! definition of Template
#ifdef WIN32
    #pragma bss_seg( ".AAA$FRAME_DATA" )
    extern const FAST T_FRAME_TEMPLATE frameTemplate;
    extern const FAST T_FRAME_TEMPLATE frameTemplateDefault;
    #pragma bss_seg()
#else
    #pragma dataseg=FAST FRAME_TEMPLATE
    extern const FAST __no_init T_FRAME_TEMPLATE frameTemplate;
    extern const FAST __no_init T_FRAME_TEMPLATE frameTemplateDefault;
    #pragma dataseg=default
#endif
extern const SLOW T_FRAME_TEMPLATE frameTemplateInitializer;
//-----
//@SubGen template end@<DATACLASS STATIC_FREQUENT_STATIC_DEFAULT >
```

Make a copy of this template, replace 'STATIC\_FREQUENT\_STATIC\_DEFAULT' with the custom dataclass name and insert the custom extern definitions required by the dataclass.

Note: There is another template (@SubGen template start@<DATACLASS>) in the same file that acts as default template. If this default template is appropriate for the custom dataclass then there is no need to make any modifications at all.

A similar template has to be provided in 'Frame\_constructor.c'. Locate and edit the template '@SubGen template start@<DATACLASS STATIC\_FREQUENT\_STATIC\_DEFAULT>' as described above for 'Frame\_attributes.h'. Again, there is a default template and if this suits the custom dataclass there is no need to provide a custom definition.


Additionally the constructor of the new dataclass has to be defined via a template.

Locate the following template in 'Frame\_constructor.c':

```
//@SubGen template start@ <DATACLASSLIST CONSTANT>
//lint -e{708} union initialization is OK
CONSTRUCTOR_DC_CONSTANT(frameTemplate)
//@SubGen template end@ <DATACLASSLIST CONSTANT>
```

Make a copy of this template, replace 'CONSTANT' with the custom dataclass name and rename the constructor. The parameters to the constructor may need tuning as well.

<sup>2</sup> 'Template' will be expanded to the dataclass name while 'Frame' will be expanded to the subsystem name.

	<b>Technical Information</b> <b>Entry Tool - Instruction</b>		<b>TI</b>	
Responsibility: <b>APR</b>	Date: <b>2006-08-28</b>	Language: <b>en</b>	Filing system : <b>0.6</b>	Revision: Page:
Issued by:	Approved:	Released: <b>ABB SW</b>		Area of validity:

## 10 Revision Chart

Rev.	Description of Version/Changes	Primary Author(s)	Date
0.1	First draft	Georg Horst	2006-04-28
0.2	Added detailed table descriptions	Martin Dahl	2006-05-29
0.3	Added Chapter "Custom T_DATA_OBJECTS"	Martin Dahl	2006-06-19
0.4	Added Chapter "Configure .NET Framework"	Georg Horst	2006-07-26
0.5	Changed the state of the ArmGen code generator	Georg Horst	2006-08-21
0.6	Added "Custom Dataclasses", edited "Custom T_DATA_OBJECTS" and "Create Subsystem Dataclasses"	Martin Dahl	2006-08-28