SIEMENS

HAL Workbench

DOCUMENTATION

- SW User Guide -

Version: 1.0

Last Change: 2013-08-13

SIEMENS H AU PLM PWD SW DE

Table of contents

List of Figures

List of Tables

1. Introduction

1.1 Overview

Shortcuts The Entitlement of HAL is:

- ***one* person,** instead of a whole team
- **who is a domain expert,** and not necessarily a software developer
- **can code programs** by typing programs in a language specialized to his business
- **click** and make use of an automated creation of the executables
- **& test** by activation of his executables in Connexx
- **by listening** into the hearing system instantaneously.
- **for a defined set of use-cases**

This has to be provided by

- a Language: (2nd item in the list above)
  A so-called domain specific language is designed specially to the needs of the domain, which is in our case: audiology.
  One aspect, which makes the HAL domain specific is the fact, that it provides a *hardware abstraction:* For instance, to set `hi:d8FFbcSs` to the step “fast”, there is no need to know the numerical representation of “fast”, i.e. that the Asic needs the decimal value “2” to set this hiControl to the “fast” mode. Indeed there is an accordant literal representation, so that the following statement leads to the desired result: `hi:d8FFbcSs = d8FFbcSs.fast;`
- The underlying concept, (7th item in the list above)
  which defines a set of use-cases for initiation of the programs like the first fit of the hearing instrument or the program selection to adjust the hearing system to a special listening situation (speech, music, television, ...)
- and the integrated development environment, which helps the domain expert by
  - visual means to enhance readability and editability
    (syntax highlighting, code completion, code unfolding, smart indent)
  - context sensitive help
  - has prepared slots/places to place the code, each for its individual purpose

- indicates errors during editing, thereby avoid to recognize errors later at runtime
- provides an executable “on click of a button”, which makes many software engineering activities (architecture/design; build & integration) unnecessary

This marks also the preferred direction of future developments:

- make the syntax of the language more domain specific
- extend the concept, as there come up new use cases (e.g. Fitting Assistant in Connexx)
- improve the integrated development environment (IDE)
- occasionally in further future: create executables for other use cases.

1.2 Goals & non-Goals

Goals of this document: teach a user how to use hearing aid language to create the behavior of hearing instruments in Connexx

Non-goals of this document:

- teach how to create an MPV or the usage of tools for creation of MPV

2. Setup of Environment

2.1 Preconditions: Installed Connexx & Database

HAL Workbench may be installed without having a Connexx or HiCoss installed previously: In this case HAL Workbench can be used to edit macros, but it is neither possible to create the accordant executable library (DLL) nor to execute it.
Therefore it is recommended to install Connexx and a Hearing Instrument database before installing HAL Workbench.

2.2 Installation of HAL Workbench

Recommendation: For D8 and D9 macro code developing shall only used the latest deployed HAL Workbench version. A development team shall use all the same version of the HAL Workbench (either 7.2 or all 7.3).

For further information please have a look to the RD Wiki:

http://rdwiki.audiology-solutions.net/rdwiki/index.php?title=HAL_Workbench_for_digital8

While installing HAL Workbench, the following dialogs may appear:

- **Required Products**: This Dialog may tell “The current Connexx or HiCoss is not of minimal required version”. This will not lead to a loss or problems, if HAL Workbench is only used to create language constructs provided by the old Connexx Version
- **Destination Folder:** It is highly recommended to keep the destination folder unchanged. This makes it easier for Siemens Audiology Solutions to provide support.
- **Select a Workspace:** It is recommended to keep this folder unchanged as well. It is highly recommended not to place any .hl-files (files containing macros) within
  C:\Program Files\SAT\HAL_Workbench\workspace
  due to a fact, that this is a subfolder of HAL Workbench. If there is the need for any reason (although this use-case is not foreseen anyway) to exchange the HAL-Workbench, the directory of HAL Workbench may accidently replaced *with* the subfolder. Therefore the subdirectory “workspace” shall contain a reference to the .hl-files which are stored in another location.

2.3 Recommended Desktop shortcuts

After the Installation of HAL Workbench it is helpful to create these shortcuts on the Desktop:

- HAL Workbench: C:\Program Files\SAT\HAL_Workbench\HAL_Workbench.exe

- SHS Trace Viewer: C:\Program Files\SAT\Fitting\SHS.SAT.Common.SATTraceViewer.exe

- Connexx: C:\Program Files\SAT\Fitting\SHS.SAT.Fitting.Applications.Fit.exe

These applications will be used frequently when developing HAL macros.

2.4 Manufacturer specific Configuration

2.4.1. Hi Database subdirectory

The manufacturer-specific HiDb-Data are stored in manufacturer-specific subdirectories, for instance

C:\Program Files\SAT\Fitting\HIDB\SI

To enable Connexx to a manufacturer-specific subdirectory, for instance “RX”, the following shall be done:

- creation of a subdirectory, for instance

  C:\Program Files\SAT\Fitting\HIDB\RX

- accordant entry in the file

  C:\Program Files\SAT\Fitting\HIDB\hidata.ini

  beyond the section

  [AsstntDirectories7]

  there has to be a key-value pair, for instance

  1=RX

  and do this accordant to the following table:

Key	Manufacturer
0	Siemens
1	Rexton
2	Hansaton
3	Audioservice
4	Lavis
5	AandM
6	Electone
7	Symphonix
8	MiracleEar
9	Rion
10	Costco

11	Nikon (NIKON supported by RION)
13	Audionova

Table 1: Manufacture keys

2.4.2. Output and naming of HAL Workbench-generated macro DLL

When working with HAL Workbench it is helpful to configure it in a way that the executable DLL is generated in the accordant manufacturer specific subdirectory.
This can be accomplished by editing the HAL_Workbench.ini file in the HAL installation directory.

"C:\Program Files\SAT\HAL_Workbench"

The key **-DAssemblyPath** determines the subdirectory, where the compiled macro dll (library), which is used by Connexx, will be created.
e.g. -DAssemblyPath=**SI** (for Siemens) -DAssemblyPath=**RX** (for Rexton)

C:\Program Files\SAT\Fitting\HIDB**SI** C:\Program Files\SAT\Fitting\HIDB**RX**

The key **-DAssemblyNamespace** determines the macro dll name, which is used by Connexx. This macro dll name must link in the HiDb that Connexx can load this macro dll.

e.g. -DAssemblyNamespace =**SHS.SAT.Fitting.Business.HearingLanguageRuntime** (for Siemens)

-DAssemblyNamespace =**SHS.SAT.Fitting.Business.HearingLanguageRuntime.RX** (for Rexton)

Attention: For other manufacturer as Siemens the macro dll name must be different from the Siemens dll name. Otherwise the fitting application isn't able to handle the manufacturer specific macro dll.

This ini file is **initially** used to setup a user configuration in a new HAL-Workbench workspace. The user configuration can be changed within the HAL-Workbench with the "User Preferences" dialog. The dialog is available over the menu entry "Window -> Preferences".

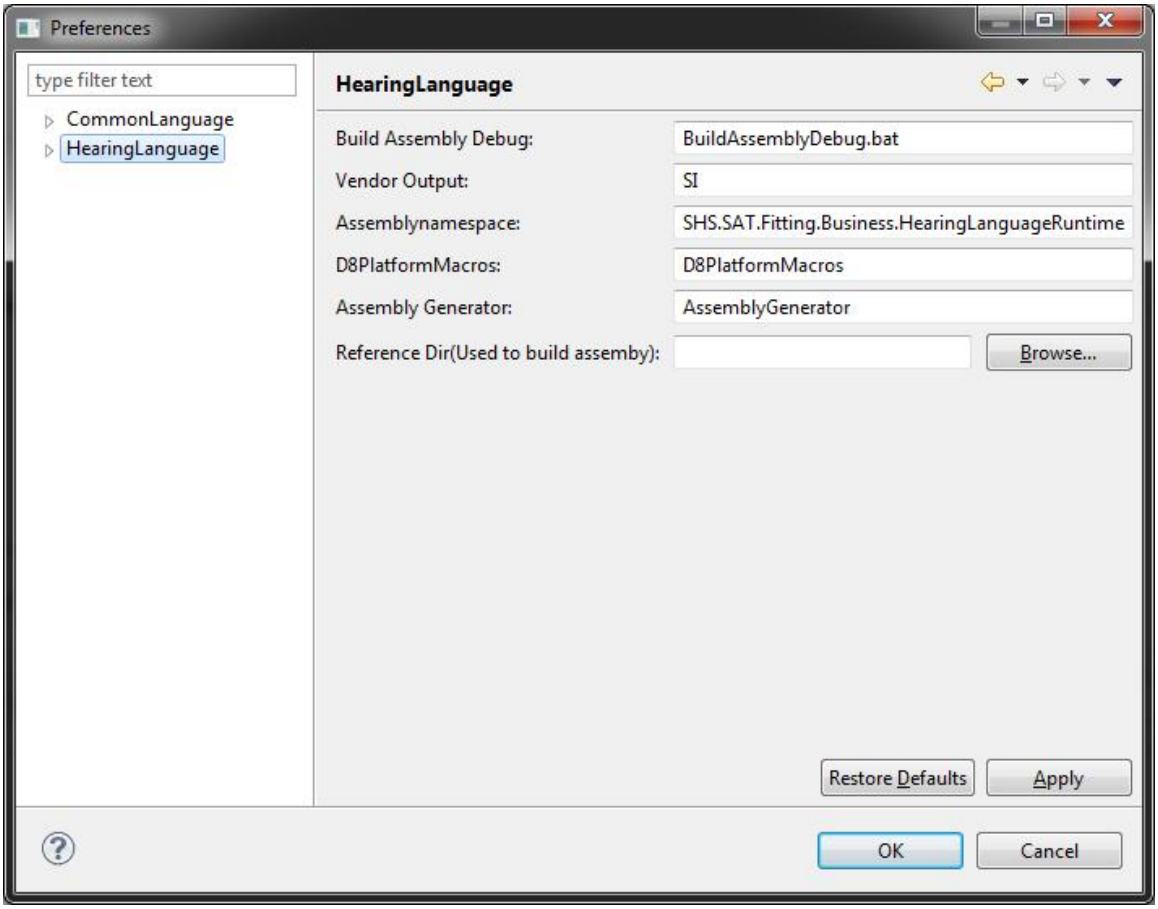


Figure 1: Preferences dialog

If the HAL-DLL is not explicit mentioned in the HIDB, default DLLs for HAL shall be taken (only, if it is not explicit mentioned. The dlls are not cascaded).

The default names should be:

SHS.SAT.Fitting.Business.HearingLanguageRuntime**.<manufacturerID>**.D**<Platformnumber>**PlatformMacros.DLL

Where: manufacturerID

0 (Siemens): <manufacturerID> = "" (empty)

!=0: <manufacturerID> = "xx" (e.g. RX or HA)

<Platformnumber> = 8 or 9 or …

2.5 MPV from specific DLL

For development or investigational purposes it may be reasonable to choose a setup that causes Connexx to get the MPV (Manufacturer Presentation View) from a separate DLL. This shall be done to use the examples handed out with this users guide.

In Detail:

The MPV is located in the following DLL:

C:\Program Files\SAT\Fitting\HIDB\SI\TestFamily1.dll

The access to it is accomplished with the following steps

1) copy of the file “developer.config” into the directory of the Connexx-application i.e. herein:
   C:\Program Files\SAT\Fitting
2) copy of the file “hxml.ini” into the following directory
   C:\Program Files\SAT\Configs
3) modification of hxml.ini: the section
   [HxmlPath]
   has to have a key
   Project=
   which has to be set up with the path of a project-hxml file (....**.prj.hxml**) , here: for instance
   Project=C:\SIFIT-Sampleproject\HIData\Life.prj.hxml
4) edit the accordant hearing instrument-hxml file (....**hi.hxml**) file placed in the subdirectory
   HIData\HI\hiFiles
   The accordant file is the file with the same prefix, which is the family name. For instance, if the project-hxml file is
   C:\SIFIT-Sampleproject\HIData**Life**.prj.hxml
   the accordant instrument-hxml file is
   C:\SIFIT-Sampleproject\HIData\HI\hiFiles\Life.hi.hxml
5) within the instrument-hxml file (....**hi.hxml**) the MPVID-subnode of one hearing instrument
   (many XML editors provide finding nodes by giving an XPath-Expression, here: `/hxml/data/v-hi/hi/MPVID`)
   shall indicate the DLL with the MPV, here:

   ```
   <MPVID>TestFamily1</MPVID>
   ```

   which causes Connexx to get the MPV from the following DLL.:

C:\Program Files\SAT\Fitting\HIDB\SI**TestFamily1**.dll

2.6 Standard access for MPV

If a user has set up Connexx as described above in chapter 2.5, it is easy to setup Connexx, so that it makes use of the standard MPV again: Just remove “developer.config” from

C:\Program Files\SAT\Fitting\

and the “old” stage is restored again

3. HAL Language & HAL Workbench: First Steps

3.1 Workspaces & Packages

It is recommended, that the user of HAL Workbench becomes familiar with two terms: the “workspace” and “project”:

The Workspace is the physical location (directory) where HAL Workbench operates in. A ”Workspace” is a commonly used term and in literature often described as place, that allows a user “to gather various files and resources” and: “often represents the complete state of an IDE at a given time a snapshot”. Indeed stores the HAL Workspace user specific settings, such as: Is spell-checking enabled or not.
The user shall import into the workspace only a reference to a project and shall not copy a project into the workspace.

A Project is the largest structural unit used by HAL Workbench. Projects contain folders and files. A project can be opened, closed or built. While the folders and files are the visible parts of a project are probably well known for the user, the project holds also more information which the HAL Workbench needs to translate the macro (.hl) files into a library that Connexx can execute.

For the practical application in daily life a difference between Workspace and Project shall be noteworthy:

- The workspace shall be used to *reference* a project and contains *user-individual* settings. If a workspace is lost the damage is low: It is easy to import the project again and to navigate to a few dialogs and set up settings as personal preferences are
- The project contains the source files, i.e. the macros the domain experts edit. They capture the expertise of the domain experts. They are shared among all domain experts who contribute to the project. If a project is lost, the damage is high.

Therefore the project shall be under source control (there are several mature VCS: Version control systems on the market) while the workspace shall not be under version control !

3.2 Workspaces

A Workspace is the physical location HAL Workbench operates in. In case the HAL Workbench is not started with a parameter, which denominates the workspace (remark: this shall be standard in near future) there appears a dialog which requests to key in the workspace.

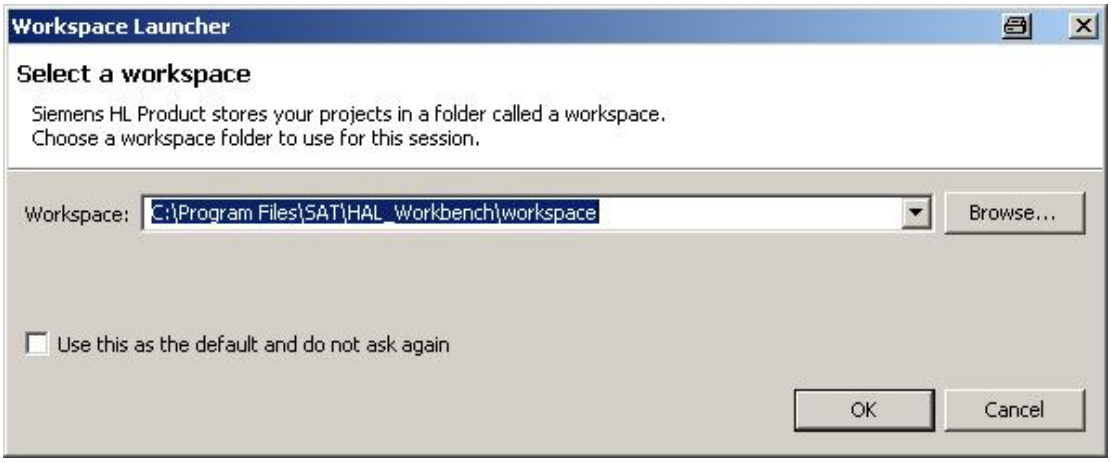


Figure 2: select workspace on opening HAL-Workbench

3.3 Open a Project in a Workspace

With HAL Workbench started,

[main menu] > File > Import

leads to the dialog, that allows the user to “import” a project into the workspace

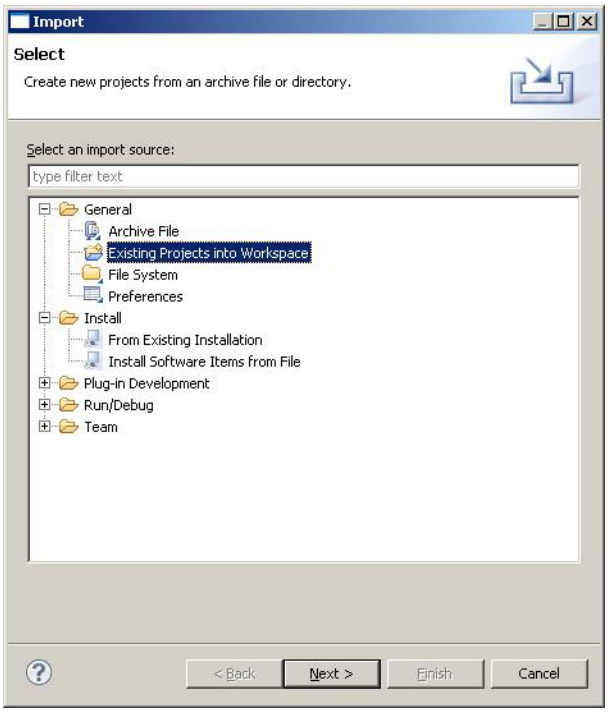


Figure 3: import a project into workspace

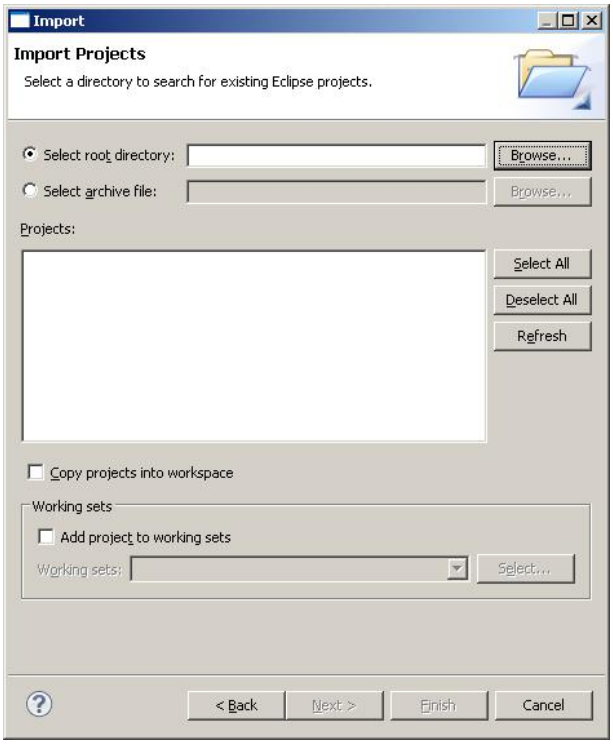


Figure 4: import a project into workspace (2)

In the dialog “Import” > [Browse]

the subdirectory

HIData\HI\hiMacro

shall be selected

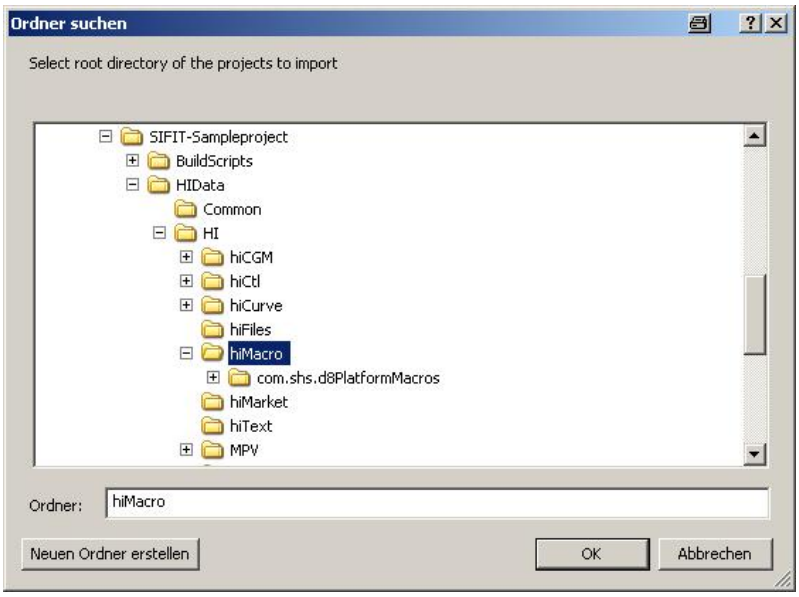


Figure 5: import a project into workspace (3)

Back again in the “Import Projects”-Dialog the checkbox “Copy projects into workspace” shall **not** be activated. *Then* the Workspace will *reference* the project.

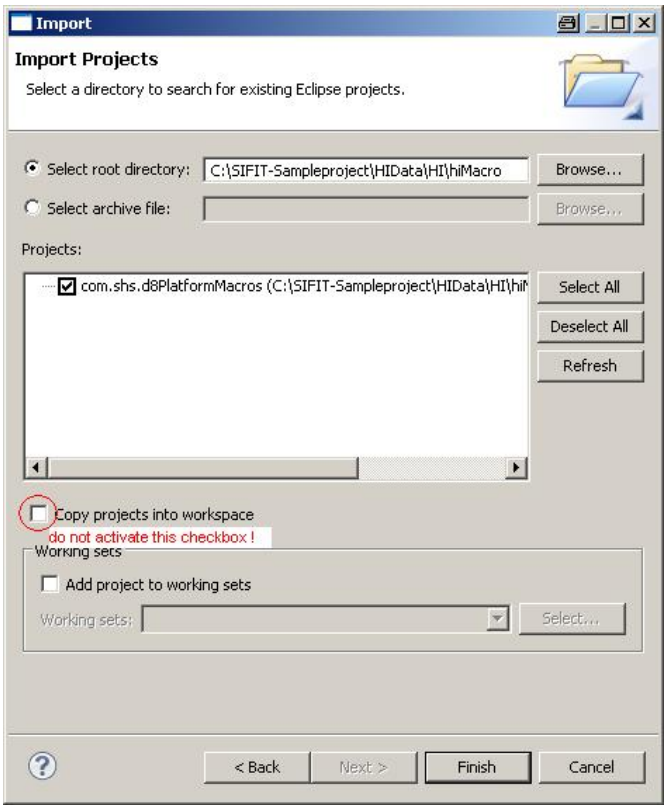


Figure 6: import a project into workspace (4)

In consequence,

the Workspace, which is here located in: C:\Program Files\SAT\HAL_Workbench\workspace,

references the Project which is here located in C:\SIFIT-Sampleproject\HIData\HI\hiMacro.

This is highly recommended, for it can’t be excluded, that HAL-Workbench may be upgraded in special situations manually by replacing the directory C:\Program Files\SAT\HAL_Workbench. In such a case, special care would be necessary not to destroy the Project, if located as a direct subfolder.

Recommendation:

To avoid the ever-time appearing startup-dialog, create a batch-file, which contains a single command:

```
start HAL_Workbench.exe -data "workspace"
```

and create a shortcut to the desktop

3.3.1. Automatic update of Hi Platform on "Project -> Import"

If a project is imported that has a reference to a STDLIB version which is named due to the "old" naming schema a dialog inform the user about that inconsistence.

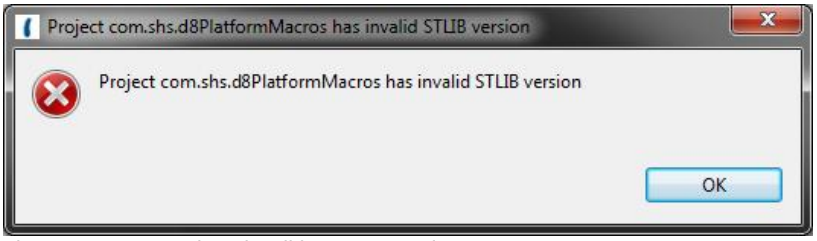


Figure 7: Message about invalid STDLIB version

After the user clicks "OK" a dialog appears where the user can select the platform (STDLIB version) which shall be used for hl-code developing.

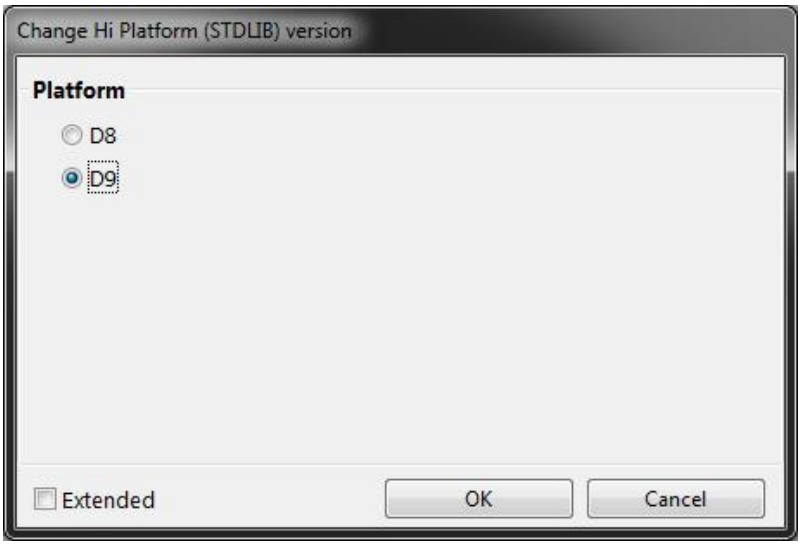


Figure 8: select Hi Platform dialog

The selected platform will be displayed beside the root element of the macro project (see **Figure 9**).

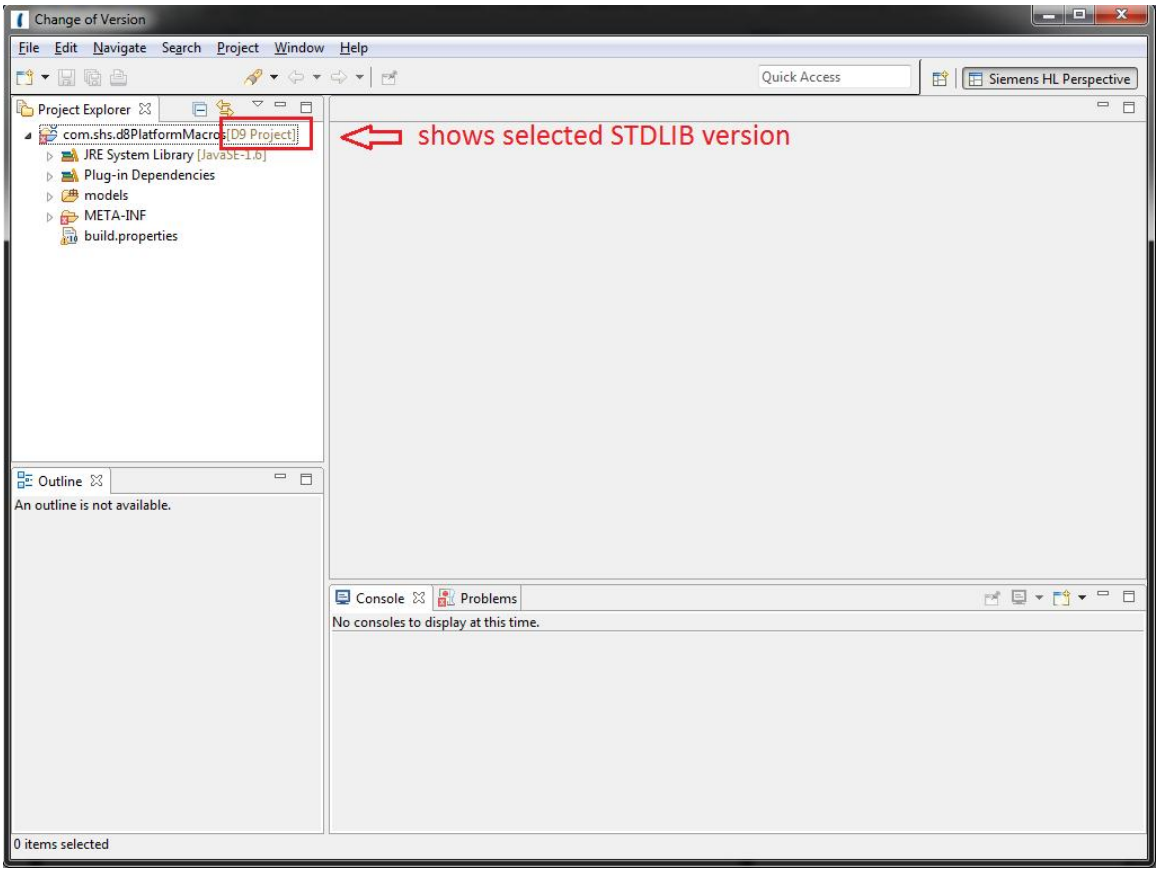


Figure 9: Hi Platform identification

3.4 Workspaces

3.4.1. Workspaces in an iterative project development

In a project which an iterative development cycles the developers work “time boxed” which means, that there are stable development results at certain time points (at the end of a “time box”) and these time points typically occur every 3 or 4 weeks.
Such an iterative, “agile” development process is used for Connexx development. The deployments (installation packages) of Connexx unveil the number of the iteration with their name. For instance, Connexx “B_Fitting_7.0.0.**IT20**.0437_2012-05-02” is a result of iteration number **20**.
As Connexx implements a new feature of HAL with such iterations, it is recommended to organize the HiDb/Sifit- and the “macro” development in the same manner, i.e. in iterations as well.

HAL Workbench offers a convenient mechanism to access different, iteration-related development stages, by creation of an accordant set of Workspaces.

Please note: the Workspace does not contain the macros itself; it references the macros, which shall reside in the file system in another location than the workspace.

It is recommended to use this Workspace concept. This can be accomplished with the following steps:

1) Create the following directory:
   C:\%HomePath%\HAL-Workspaces\
   this can be done, for instance, with the following steps:
   a) activate [Windows Start-Button] > execute (german: [Start] > Ausführen)
   b) in the dialog now enter: C:\%HomePath%\
   c) the Windows Explorer opens the Home-Directory. Herein create a Subdirectory “HAL-Workspaces\”
2) now in directory “C:\%HomePath%\HAL-Workspaces\” create a subdirectory for each iteration, for instance, the subdirectories
   - Sifit-IT19
   - Sifit-IT20
   - Sifit-IT21

   so that it looks like this:

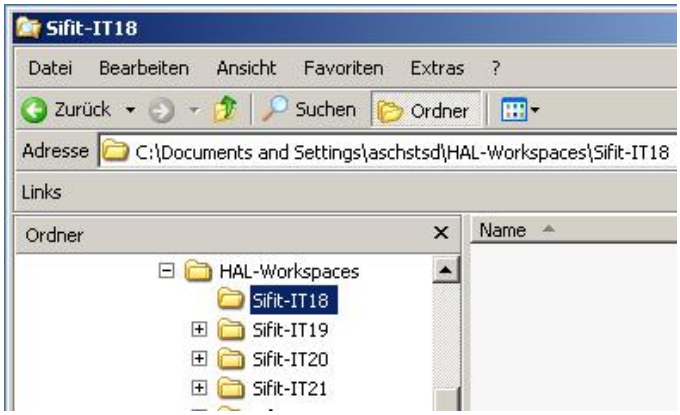


Figure 10: subdirectories aligned to iterations

3) Start HAL-Workbench. If the Workspace Launcher appears it may look like this:

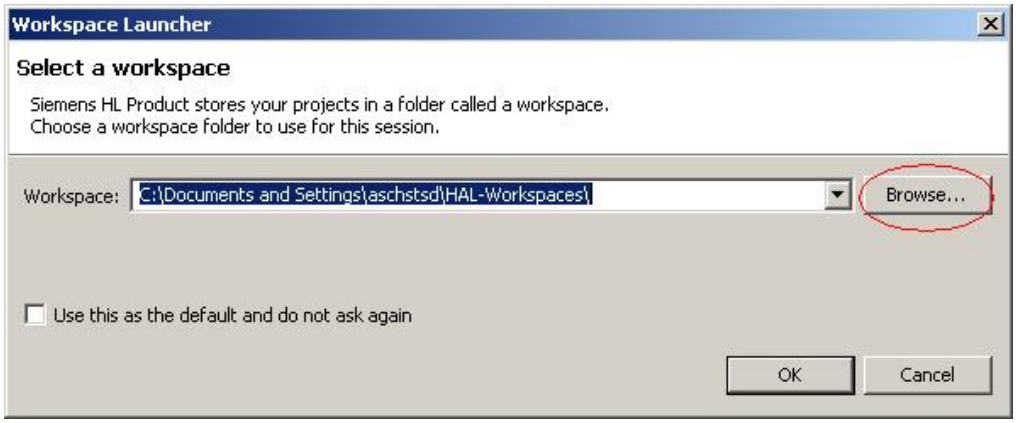


Figure 11: Workspace Launcher (1)

And use the means to select one of the iteration-related subdirectories.

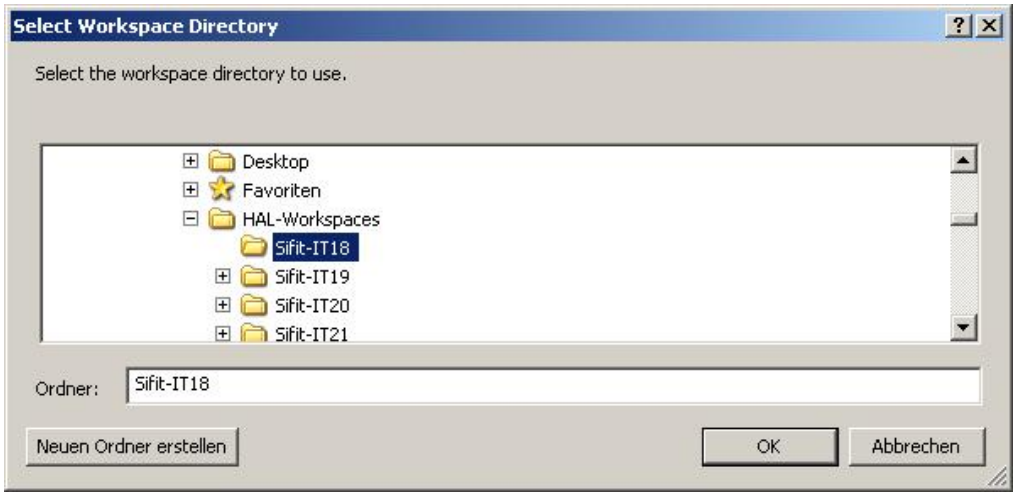


Figure 12: Workspace Launcher (2)

In this example depicted, the user has decided to work with Sifit-data that are related to iteration number 18.

4) As soon as HAL Workbench has started, use

   [main menu] > File > Import

   to import the project as described in the last chapter.

   It is now possible to work with the .hl-files (“macros”) as usual.

5) A s a benefit of this procedure HAL Workbench builds up a List of the recently used Workspaces and they can be found in

   [main menu] > File > Switch Workspace

   as depicted here:

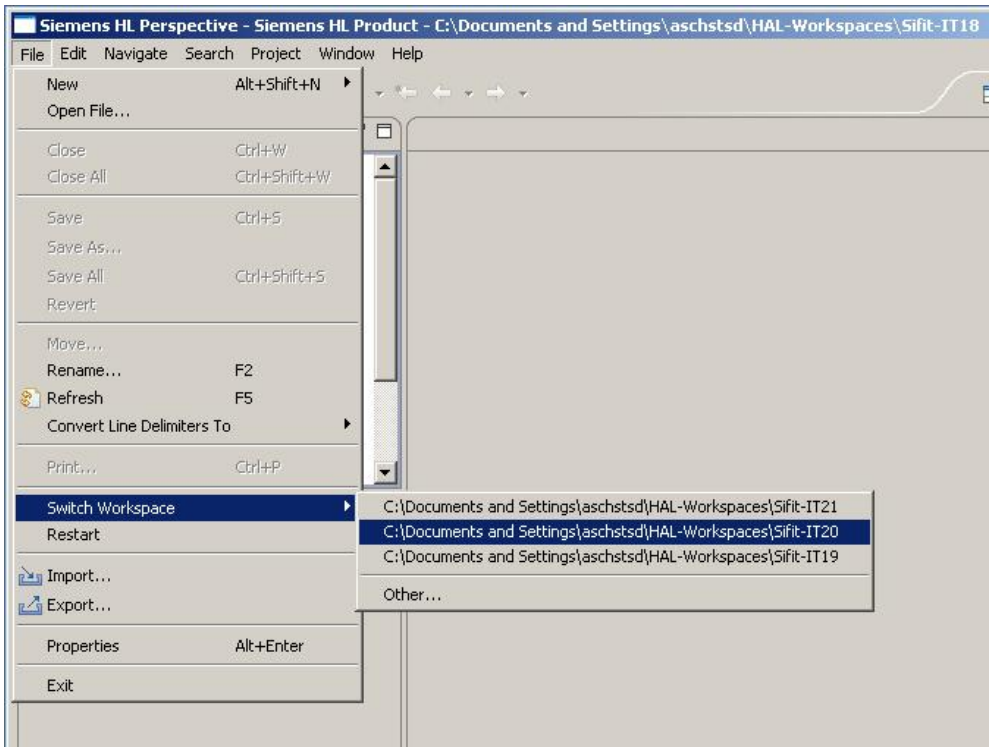


Figure 13: List of Workspaces

Remark: The Workspace, which is currently opened, is given in the windows title border (see upper right corner).

3.4.2. Workspaces for two developing two hi platforms TODO

TODO

3.5 Close a Project in a Workspace

To close a Project again, select the project in the “Project Explorer” View (com.shs.d8PlatformMacros) and press the “delete” button. In the dialog, that appears now, do **not** check “Delete project contents on disk”

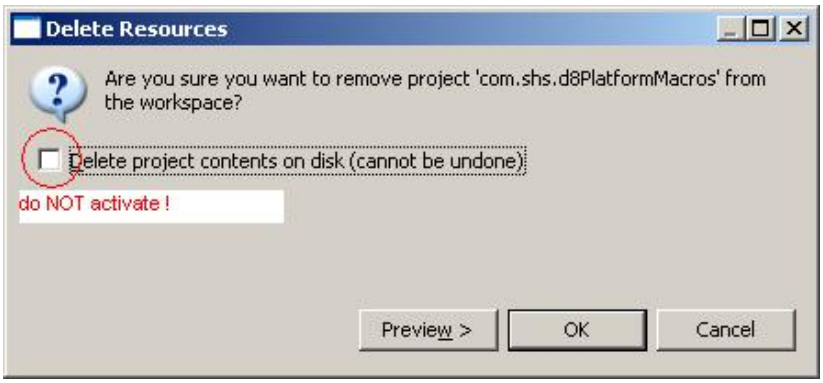


Figure 14: remove project from workspace

As a result, the project will be closed and the “Project Explorer” View is empty.

3.6 The HAL Workbench Editor

3.6.1. Overview

The areas in HAL Workbench which have a tab-Alias (german: “Tabreiter”) are called “Views” while the whole arrangement is called “Perspective”. Some menu entries help to get familiar with the Perspectives and to restore the default Perspective if the original perspective is accidently lost:

[main menu] > Window > Close all Perspectives closes the opened perspective

[main menu] > Window > Open Perspective > Other leads to the following dialog

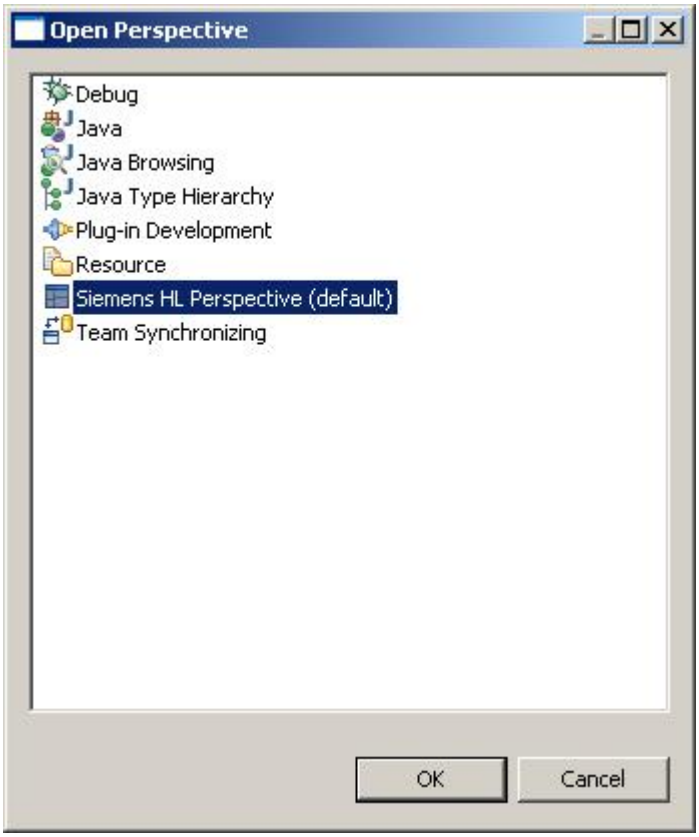


Figure 15: restoring Siemens HL Perspective

The selection of

Siemens HL Perspective (default)

causes HAL Workbench to arrange the default Perspective again.

3.6.2. Show Line numbers

HAL Workbench can display line numbers. This may be helpful especially in situations, where a discussion about some details in a .hl-file shall take place.
To activate the display of line numbers make a mouse “right-click” on the bar beside the .hl file.

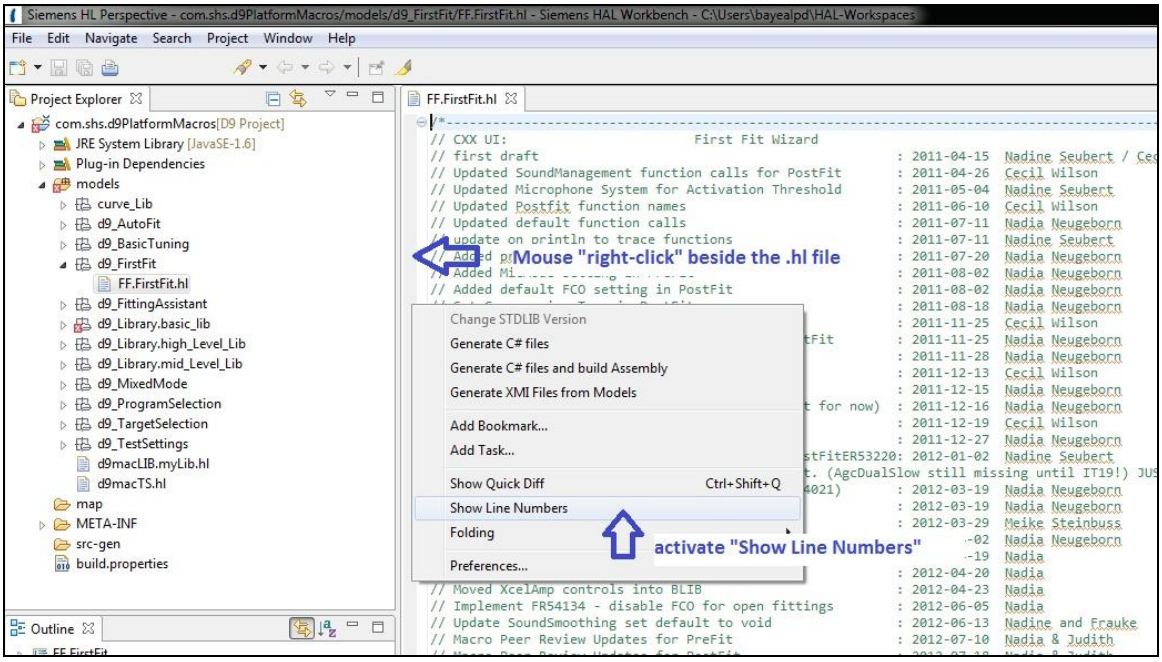


Figure 16: activate line numbers within editor

3.6.3. Synchronizing Project Explorer View and Content View

The two perspectives can be synchronized, so that a selection of a file in the Package Explorer (usually in the left side) displays the file also in the Content Perspective (right side) and vice versa. This can be accomplished by pressing the button with the icon with the two yellow arrows:

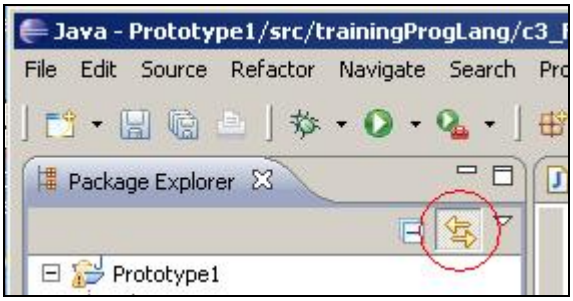


Figure 17: synchronizing Project Explorer View and Content View

3.6.4. Search, find and replace

Searching, finding and replacement are the most commonly used features in editors. In HAL

Workbench, this can be accomplished by

[main menu] > Edit > Find/Replace or: <ctrl><f>

find or replace text in the opened file

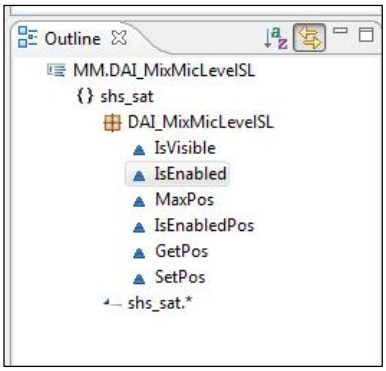
[main menu] > Search > Search or: <ctrl><h> tab [file search]

find text in files

with this function it is possible, to search for strings in the files of the entire project.

The result of these searches is displayed in a “Search” View near the bottom of the Perspective.

3.6.5. “Outline” window icons and their meaning



The HAL Workbench provides a powerful “Outline” window. The window shows the currently opened hl file as root element. Within the displayed tree the used HAL grammar statements are listed as sub nodes. These HAL grammar statements and their meaning are defined in the following table.

The user can simply navigate to the corresponding element in the hl file with a mouse-click on the displayed “Outline” window element.

HAL language construct	Icon	HAL Language example
namespace		`namespace shs_sat{…`
using		`using builtins.*;`
package		`package FirstFit FF{…`
application function		`application function void PreFit(){…`
public function		`public function int FeedbackStopper(){…`
protected function		`protected function void MpoType_Set(){…`
test function		`test function void TestGetHiParam(){…`

Table 2: Outline window icons

3.6.6. Change Hi Platform (STDLIB) version

It is possible to change the STDLIB version of an opened project with the HAL Workbench editor. It is not recommended to change a STDLIB version from a newer platform (STDLIB version) to an older one. Therefore a warning message appears to inform the user (see **Figure 18**).

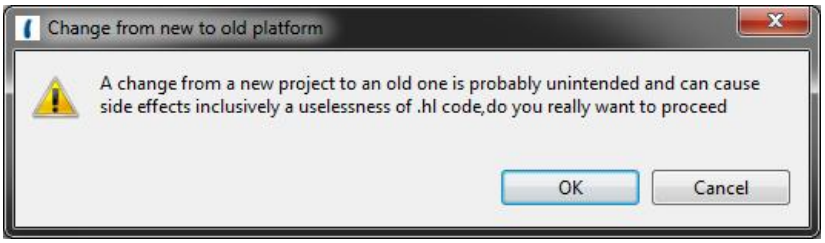


Figure 18: Change Hi Platform version from newer to older platform

The user can change the platform (STDLIB) reference of a project in this way:

1. With the context menu entry “Change Hi Platform (STDLIB) version…” (see **Figure 19**)

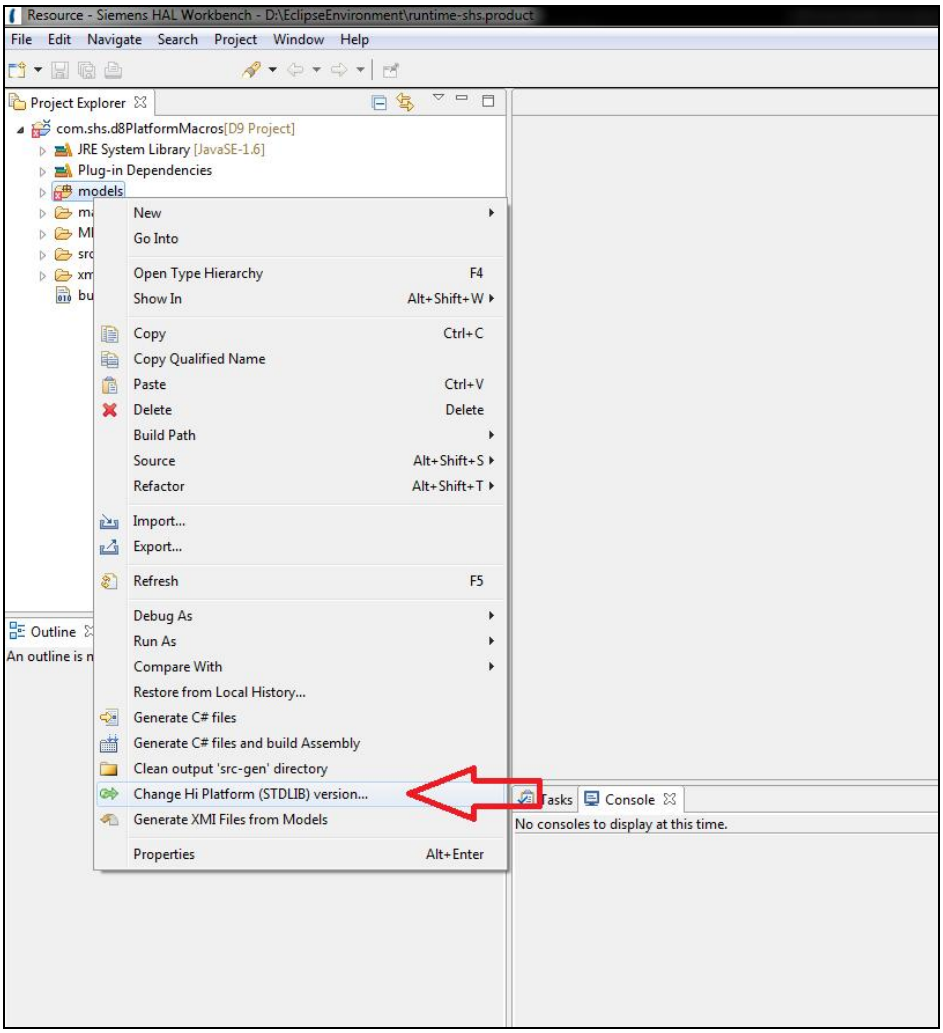


Figure 19: Context menu "Change Hi Platform (STDLIB) version"

After activating the entry the dialog (see **Figure 20**) appears and the user can select the needed platform.

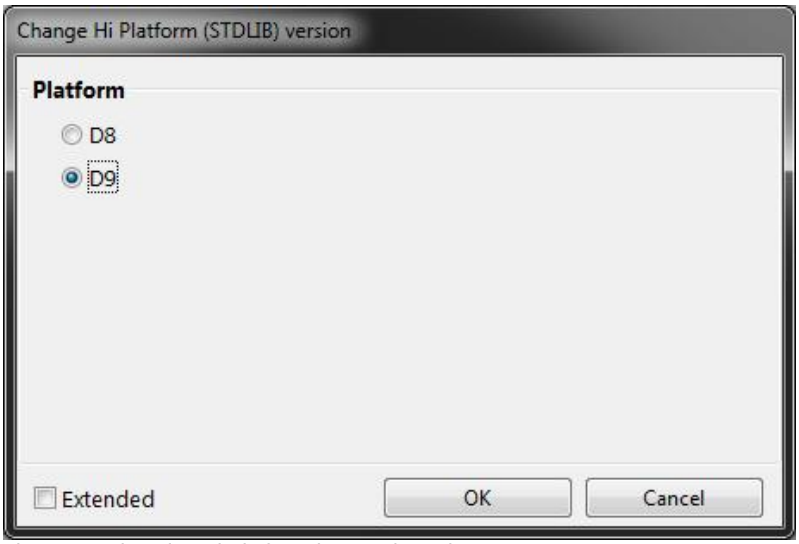


Figure 20: Changing Hi Platform (STDLIB) version

The selected platform will be displayed beside the root element of the macro project (see **Figure 9**).

4. Packages, Files, Functions & application Functions

4.1 Packages

All Files, which contain macros, are organized in “packages”. A package is a logical unit designed to group files which macros that are designed for a special purpose. The reader may know this special purpose as “macro type”. It is recommended to create a subdirectory each for all macro files of one package, for instance:

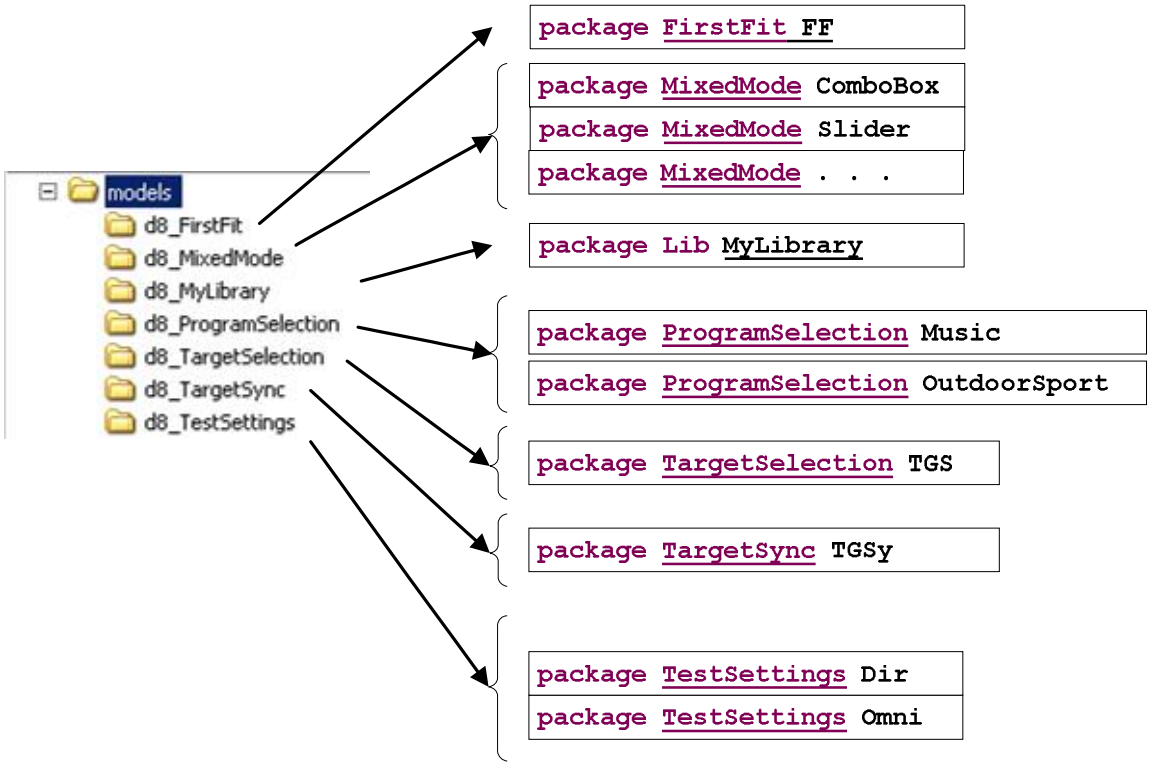


Figure 21: Project structure and packages

This organization of different .hl-Files in directories is recommended for clearness. However, if Connexx makes use of a certain package, for instance “Program Selection”, it takes all files containing a “package ProgramSelection” declaration regardless in which subdirectory they reside.

Excercise 4-1:

Create a new folder in SIFIT-Sampleproject and name it “d8_MyLibrary”. This folder shall become visible in HAL-Workbench

Hint:

If the windows file system has changed, select the “Project Explorer” in HAL Workbench and press <F5>, which causes the “ProjectExplorer” to update itself.

4.2 Files

All macros are stored in Test Files which carry the extension "**.hl**". Although a .hl-file may be created with a standard text editor, it is recommended to create .hl files with HAL-Workbench. This can be accomplished by:
[main menu] > File > New > New HL File

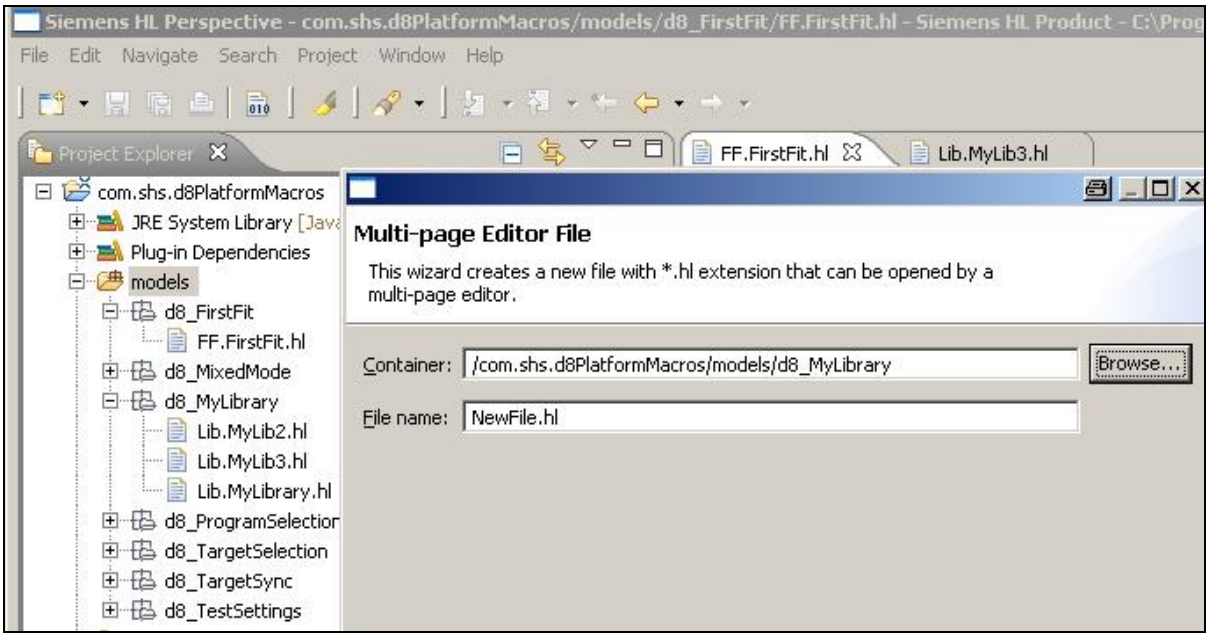


Figure 22: creation of new .hl files

The container (subdirectory) and the File name have to be specified.
The specified .hl-File is created immediately.

Remark: It is possible to create files with external editors as well or to copy a file in the file system from one directory to another. In this case, after switching back to HAL Workbench, the "Project Explorer"-View shall be selected and <F5> pressed. This causes the "ProjectExplorer" to update itself.

5. Macro use cases

5.1 Target Gain & Best Gain calculation

Connexx calculates the best gain settings for the hearing instrument. Macros located in the packages “TargetSelection” (before and after Connexx’ TargetGain calculation) and in “FirstFit” (before and after Connexx’ BestGain calculation & setting) are activated. In case of binaural fitting a macro of package “TargetSync” is activated also.
The sequence of macro activation is shown in detail in the picture below.

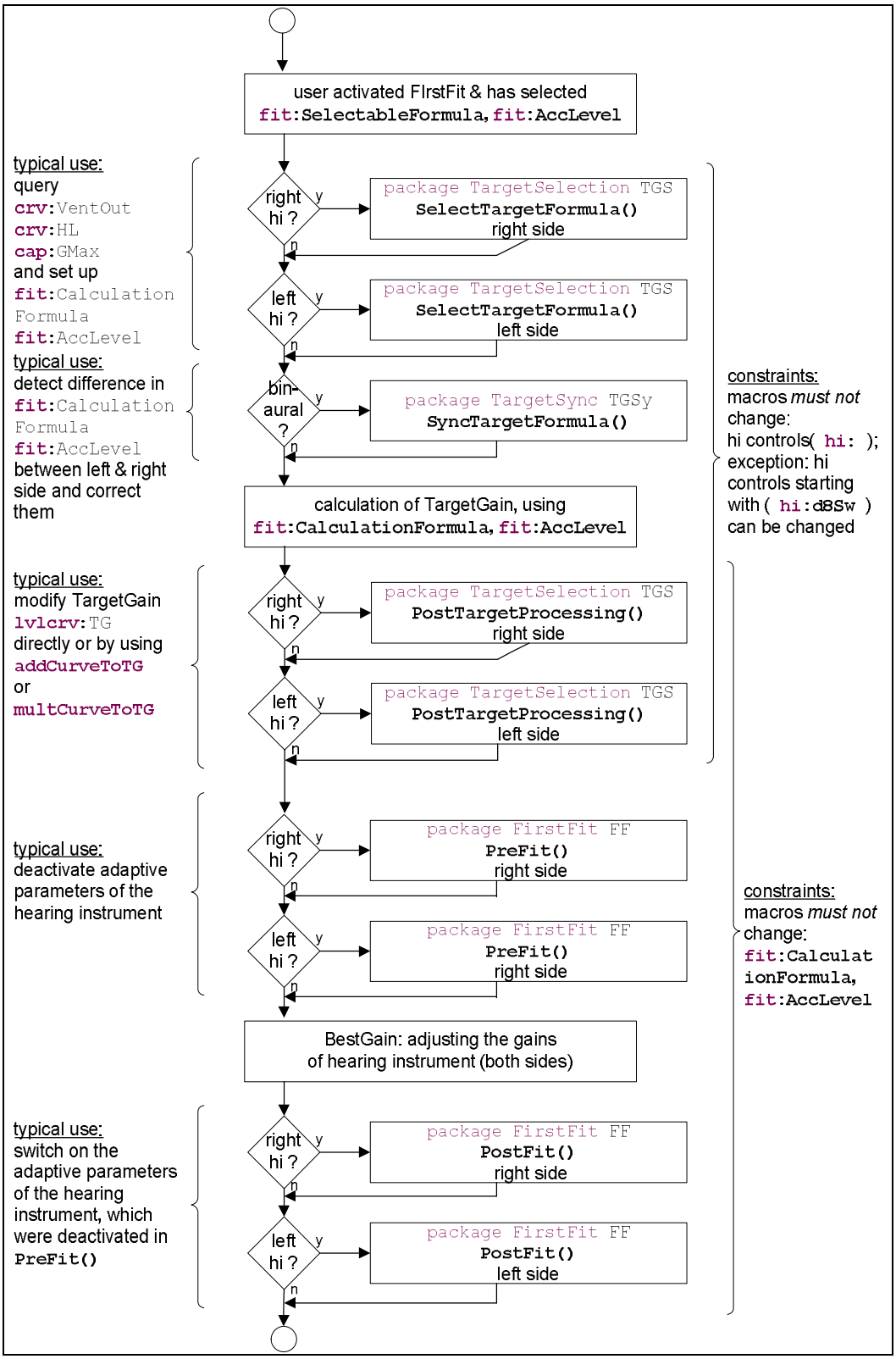


Figure 23: sequence of macro activation in Connexx FirstFit

Exercise 5-1:

set up the macros taking part in the Target- & BestGain calculation in a way, that each macro produces an output in SAT-Trace-Viewer. Use Connexx and

- simulate a hearing instrument on one side
- simulate a hearing instrument on both sides

cause Connexx to execute the first fit and view the output in SAT-Trace-Viewer

5.2 Mixed Mode

It is not the purpose of this document to teach, *how* an MPV has to be set up. Here only a reminder: Each user interface control (e.g. ComboBox) has a name, e.g.:

```
<ComboBox    x:Name="MAC_ComboBox"
```

and if this name starts with “MAC”, the control activates a macro (here: `package MixedMode ComboBox`). Connexx creates and updates the state of the user interface control by executing several application functions of this macro:

Connexx runs these macros to get information about the user interface control as whole:

- IsVisible() to determine if the whole ui control is visible or not

and

- IsEnabled() to determine, if the whole ui control is enabled or not

and {

- MaxPos() the maximum number of visible positions, i.e. of positions, that are switched as “visible” by function “IsVisiblePhysicalPos()”

or

- MaxPhysicalPos() the maximum number of all positions, *regardless* if they are switched as “visible” by function “IsVisiblePhysicalPos()” or not

}

Connexx runs these macros to get information about each single position of a user interface control:

- IsVisiblePhysicalPos() to make a single position of the ui control visible or invisible

and {

- IsEnabledPos(int pos) is the position with the given index -which is the index of the sub-set (*only* the positions, that are visible, i.e. IsVisiblePhysicalPos() returns “true”)- enabled or disabled Connexx)

or

- IsEnabledPhysicalPos(int pos) is the position with the given index -which is the index of the super-set (*all* positions regardless if visible or invisible)- enabled or disabled

} and

- SetPos(int pos) to set the ui control to the given position

and

- GetPos() to get the current position (to update the ui control in Connexx accordingly)

Remark: In contrast to Connexx 6, there is not only a function SetPos(), but also a function GetPos()

5.3 Program Selection

The macro application functions that are called by Connexx are specified in HAL SDS. Please take a look into that document.

6. Interface Data & Built-In Functions

6.1 Interface Data

There is a wide range of interface data that can be used by the macro developer to query or modify the state of the hearing system.

- all interface data are preceded by a scope-operator, there are
  - `pat:` interface to patient's data
  - `hi:` interface to hearing instrument controls
  - `cap:` interface to hearing instrument's capabilities
  - `crv:` interface to curves (like hearing loss, uncomfortable level)
  - `levelcrv:` interface to a levelcurve (i.e. curves for 3 levels)
  - `fit:` interface to data, that determine behavior of Connexx' FirstFit calculation
  - `env:` interface to environmental data

- all interface data have either read or read/write access, described in detail in HAL SDS
- there are constraints, i.e. conditions specific for single interface data that allow or prohibit the usage in a certain use-case (“macro type”) or not. This is described in detail in HAL SDS.

6.2 Built-In Functions

All Built-In functions are specified in HAL SDS.

A small set of Exercises may help to explore these functions:

Excercise 6-1: determine, if hearing level curve is defined.
Place some code in the TargetSelection, SelectTargetFormula() macro, that produces a Trace-Output if the HL (hearing loss) curve is defined or not.

Excercise 6-2: create a MPV-View with a combo-box. The combo-box shall activate a mixed-mode macro, which sets the hiControl `hi:d8SwMacroVar23` to the position according to the selected line. The application function `GetPos()` shall produce a trace-output, which indicates, if `hi:d8SwMacroVar23` is at it’s minimum position or not

7. Hello World: Run macros

With HAL Workbench, Connexx and SHS Trace Viewer it should be easy to get macros run and to view their effects:

- first, start the “SAT Trace Viewer” (its located here: C:\Program Files\SAT\Fitting\SHS.SAT.Common.SATTraceViewer.exe")

- in SAT Trace Viewer, select

  [main menu] > Edit > Filter

  and set up

  Field: **Message**

  Keyword: **[HAL]**

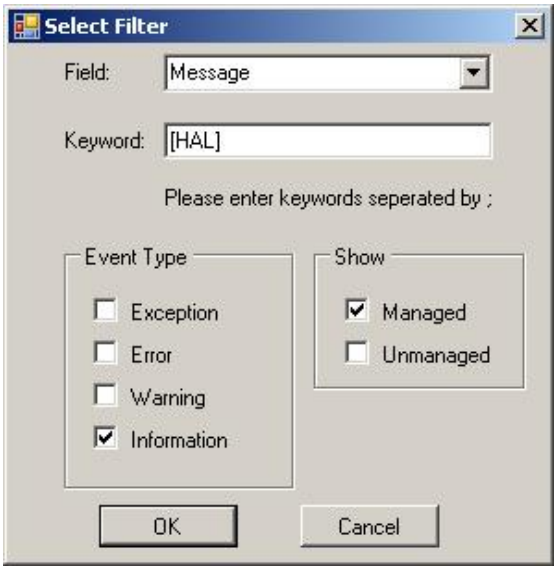


Figure 24: Filter Settings in SHS Trace View

In consequence, the Trace Viewer will display messages, but only those, which start with “[HAL]”

- with HAL Workbench, place some println-statements in macros that print a text starting with “[HAL]”. For instance:

```
int hp=env:Prog;
println('[HAL]MixedMode ComboBox.IsVisible() in HP='+hp);
```

- with HAL Workbench create an Assembly, that can be executed by Connexx. This shall be accomplished with the following steps:

  1 - [main menu] > Project > Clean

2 - in Project Explorer, select to subdir “src-gen”, unfold, delete all files

Remark: its a known bug, that step1 and step2 are necessary. This shall be accomplished in near future, so that only step3 shall be necessary

3 - in Project Explorer, select subdir “models” > [right mouse button] > Generate C#-Files and build Assembly

- start Connexx and simulate a hearing instrument

- in SAT Trace Viewer, select
  [main menu] > Capture > Start Capture

- when now acting in the user interface of Connexx, SAT Trace Viewer shall display a Trace Output

8. Error Handling

In case of an Error Connexx displays a universal, non-narrative Error message. Currently the macro developer does get no hint, where the root cause may be located. Currently (2012-Feb) the feature request FR52486 is accepted by D8ProgramBoard for implementation in Connexx7.0. With this feature request, error messages will give detailed information about

- the use case / macro type, which causes the error
- the file with the code causing the error
- root cause of the error (invalid array access, invalid control position, exception in built-in function or unknown interface data (e.g. a curve is missing in HiDb))
- further details in a separate description

9. Appendix A: Definitions, Abbreviations, References

9.1 Definitions

Definitions

- API — application programming interface: in the context of the hearing aid language an interface that the domain expert respectively HAL can use to perform some special operations (see chapter 4.3.)
- HAL — Hearing Aid Language: the domain specific language to modify hearing instruments in the workflow of the SATs Fitting Software Connexx
- HAL program — a program written in HAL. These programs are a successor of the SHIMAL-macros (Conexx6 and earlier, compare chapter 2.2.).
- HAL Runtime — the “runtime engine” for programs written in HAL, i.e. the Software, that executes the code written in HAL.
- instance — a concrete expression of a template (tabling, boilerplate). example: the datatype “int” may be regarded as an instance. “int j” creates a data field j, which is a concrete expression of the datatype “int”. The word “instance” is often used in Software development
- Instantiation — the act of creating an instance from a template.
- Macro — the same as a HAL program. The term “macro” is so common among developers at SAT RD, that this term is used also now to describe programs written in HAL.

9.2 Abbreviations

API	Application Programming Interface
Cxx	Connexx
hi	hearing instrument
DSL	Domain Specific Language
HAL	Hearing Aid Language

BTE	Behind The Ear hearing instrument

Table 3: Abbreviations

9.3 References

- HAL SDS (Agile DocID: D00093267)
- Manufacturer Presentation View SDS [genaue Bezeichnung noch von Christoph erwartet]

10. Appendix B: Solutions of Exercises

This chapter contains the solutions to the exercises in this document.

10.1 Exercise 6-1:

Exercise 6-1: determine, if hearing level curve is defined.
Place some code in the TargetSelection, SelectTargetFormula() macro, that produces a Trace-Output if the HL (hearing loss) curve is defined or not.
Solution:

The following lines may be placed in the SIFIT-Sampleproject, file MM.ComboBox.hl, in the function

```
application function GetPos(){
```

10.2 Exercise 6-2:

Exercise 6-2: create a MPV-View with a combo-box. The combo-box shall activate a mixed-mode macro, which sets the hiControl `hi:d8SwMacroVar23` to the position according to the selected line. The application function `GetPos()` shall produce a trace-output, which indicates, if `hi:d8SwMacroVar23` is at it's minimum position or not

Solution:
The following lines may be placed in the SIFIT-Sampleproject, file MM.ComboBox.hl, in the function

```
application function GetPos(){
:
//Excercise 6-2
if(isMin(hi:d8SwMacroVar23)){
  println('[HAL]MixedMode isMin(hi:d8SwMacroVar23) == true ');
}
else{
  println('[HAL]MixedMode isMin(hi:d8SwMacroVar23) == false ');
}
```

11. History

Version	Description of change	Date <YYYY-MM-DD>	Created by
0.1	initial draft:	2011-12-08	Aschoff
0.7	first deployable version	2012-02-15	Aschoff
0.8	new chapter 3.1 inserted: "Workspaces & Packages"	2012-02-23	Aschoff
0.9	new chapter 3.5 inserted: "Workspaces in an iterative project development"	2012-05-04	Aschoff
0.91	Update chapter 2.4.2 for OEM specific macro dll names and paths. Use new template.	2012-10-08	Alexander Bayer
0.92	Added chapter "3.6.6 Outline Window icons and their meaning"	2012-10-24	Alexander Bayer
0.93	Added "Preferences" dialog and macro dll naming pattern.	2012-10-31	Alexander Bayer
0.94	Added D8/D9 features and workflow	2013-07-16	Alexander Bayer
1.0	User Guide for Fitting 7.3	2013-08-13	Alexander Bayer

Table 4: Document History