User Manual

Decoder Debugger

PROCITEC GmbH

Imprint

PROCITEC GmbH
CEO: Dipl.-Ing. (FH) Dipl.-Inf. (FH) Jens Heyen
Rastatter Strasse 41
D-75179 Pforzheim
Germany

Phone: +49 7231 15561 0
www.go2signals.de
go2signals@procitec.de

Register: Mannheim HRB 504702
Tax ID: DE 203 881 534

Printed: 20. November 2018

Contents

1 General

1.1 Welcome to the Decoder Debugger

The Decoder Debugger supports decoder development based on the *Decoder Description Language DDL* for go2SIGNALS systems.

The graphical user interface of these systems (SDA) already provides basic tools to create and modify decoders, however, the Decoder Debugger allows for detailed program analysis including the monitoring of variables, data input/output and the setting of breakpoints. Decoders can be tested and debugged in two different modes.

In Offline Mode, pure decoder functions can be investigated without any system overhead during runtime. The system configuration required is shown in Figure 1.

To observe the behavior interaction with demodulators and the automatic production control as well, run the debugger in Online Mode as graphical user interface of the APC program. This requires the system depicted in Figure 4.

In either mode, decoder programs can be executed and monitored in various single-step modes as well as continuously with or without breakpoints.

1.2 Software Environment

All functions described are embedded in the application "APC.exe", which also contains the complete signal processing environment of go2DECODE. During normal operation, APC runs as a background process without any visible user interface. However, a user interface for decoder debugging purposes may be activated during APC runtime via the SDA user interface. This mode allows for full control online debugging of decoders and automatic processing behavior. This **<Decoder Debugger Online>** is started via the SDA's **<Extras>** menu.

You can also start APC in sole decoder debugging mode without any signal processing functions. In this mode, only a decoder interpreter is activated, and controlled by a similar user interface. APC is started in this mode via the link **<DecDebug>** or the start icon  when running the **<Decoder Debugger Offline>**.

2 Offline Mode

2.1 Offline Operating Concept

In this mode, the decoder is operated within an isolated environment, i.e. without being affected by APC control during runtime. This operating platform is provided for initial decoder development operations. It is easy to handle and decoders most often can be run faster than under real-time conditions, which is a helpful feature with time-consuming slow baud rates. The basic approach for this mode can be described as follows:

Under real operating conditions, the decoder will be fed by the demodulator output. For debugging, the decoder will now access a recording of this output. Therefore, in a first step, it is necessary to produce a record file as described in chapter Producing a Demodulator Output Record on page 5.

The recording will contain the complete set of information produced for decoder input during recording runtime, such as bit stream, symbol and channel arrangements, burst information, time of arrival etc. To start debugging, load only this record file in connection with the compiled decoder including the source code.

The **<Decoder Debugger Offline>** mode, however, does not allow for monitoring any interactions with external APC functions such as:

- Manipulation of demodulation parameters initiated by the decoder
- The automatic modem detection process
- The effect of decoder results for the go2DECODE system

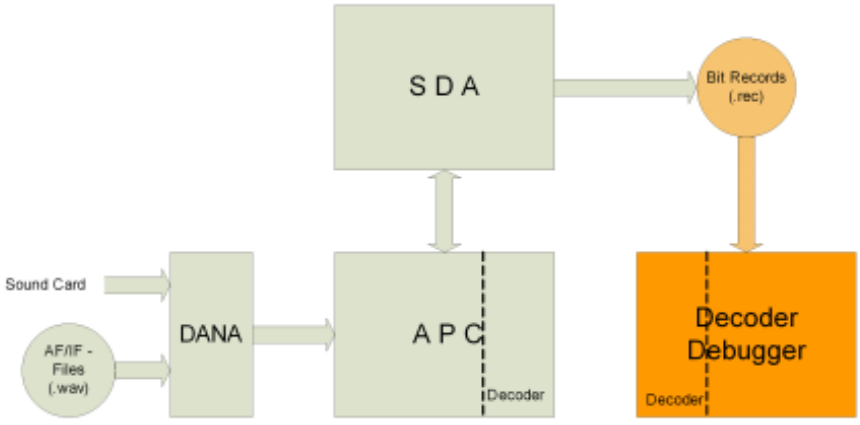

Figure 1: Decoder Debugger – Offline Operating Environment

2.2 Decoder Debugger - Offline Main Window

The **<Decoder Debugger Offline>** user interface elements are shown in the Figure below, followed by a brief description:

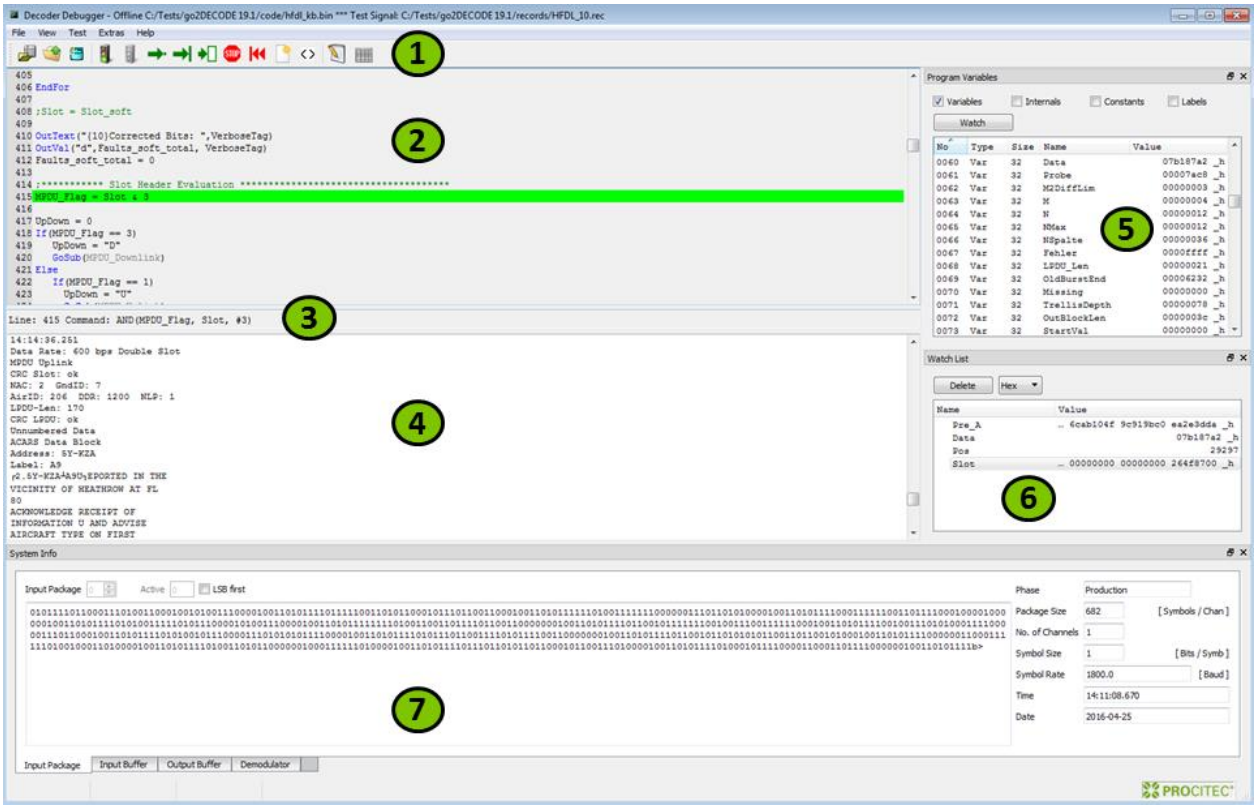

Figure 2: Decoder Debugger – Offline Main Window

1. Menu Bar and Toolbar
   Central functions can be accessed via the menu bar. For most of these functions, toolbar icons are provided as well.
2. Source Display
   The source code of a loaded decoder is displayed in this window. Breakpoints and the subsequent program line are highlighted in different colors. This display is read-only. Use the editor for modifying source codes.
3. Command Line Display
   Each source code corresponds to a single, or a set of, runtime commands which are composed during compilation. The next runtime command is displayed in this line.
4. Result Display
   The decoding results are displayed in this pane. Results may be raw, XML text or filtered.
5. Variables List
   This pane displays all program variables and their current values.
6. Watch List
   This pane may contain a selected subset of the Variables List. Values can be indicated in different formats.

7. Buffer and Parameter Display Tabs
   The panes on these tabs provide information on:
   - Last input package received from demodulator
   - Input buffer with current read pointer indication
   - Output buffer containing current output command results
   - Demodulator parameters
   - Additional display for selectable long variables

2.3 Producing a Demodulator Output Record

Demodulator output records can be produced via the go2DECODE user interface SDA. These records are images of the real-time decoder input data streams.

The correct demodulator has to be prepared and must be run in manual production mode, i.e. the automatic signal processing (button **<Automat>**) must be switched off. No decoder is required, but allowed while recording. For details on the running of modems, please refer to the go2DECODE Instruction Manual. The recording can be started by the icon  . It will run until the modem is stopped. The selected record file name can be used as Decoder Debugger data input.

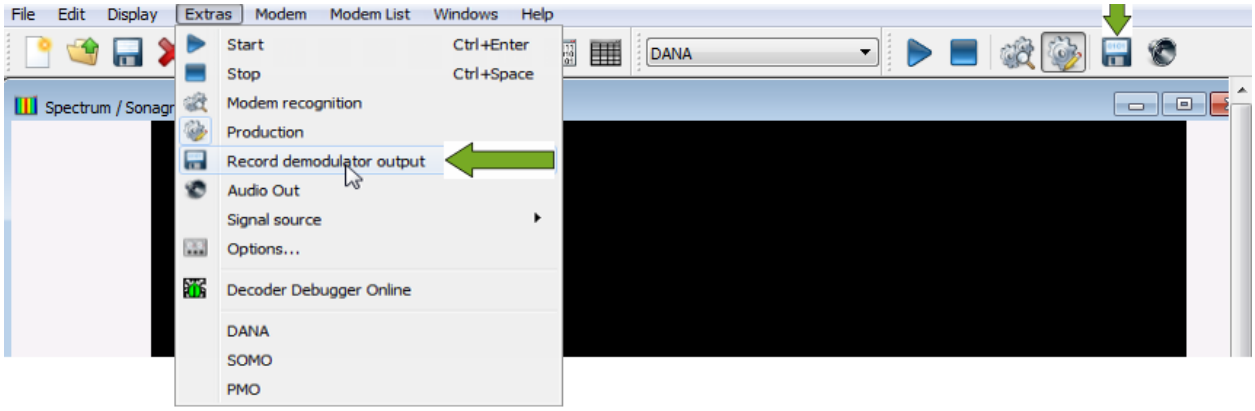

Figure 3: Starting Demodulator Output Records

2.4 Loading Record File and Decoder

Decoder debugger operation requires compiled decoders and recorded demodulator outputs. If only source code is available, it must be loaded via the decoder editor and compiled first.

Existing decoders are loaded via:

- **<File><Open><Decoder>**, Ctrl+O, 

The displayed file browser offers all files with the extension "*.bin", i.e. compiled decoder codes. If the corresponding source code (extension "*.txt") is found in the same directory, it will automatically be loaded as well. An alert message is displayed if the source code has been changed since the last compilation.

Pre-recorded data are loaded via:

- **<File><Open><Data>**, Ctrl+D, 

The displayed file browser will offer all files with the extension "*.rec" or an alternative (obsolete) format "*.bit".

3 Online Mode

3.1 Operating Concept of Online Debugging

All the interactions between the decoder and other go2DECODE functions can be examined online during normal operation. In this mode, the decoder runs in its original environment but is observed and controlled by the APC's debugging user interface (see Figure below) In order to have a running system, the applications SDA and DANA must be started and operated as well, i.e. the system can be run as the original go2DECODE via SDA as long as there is no interrupt by the decoder debugger user interface. There are no restrictions with regard to the signal source, but loss-free single step operation is only possible with file source input.

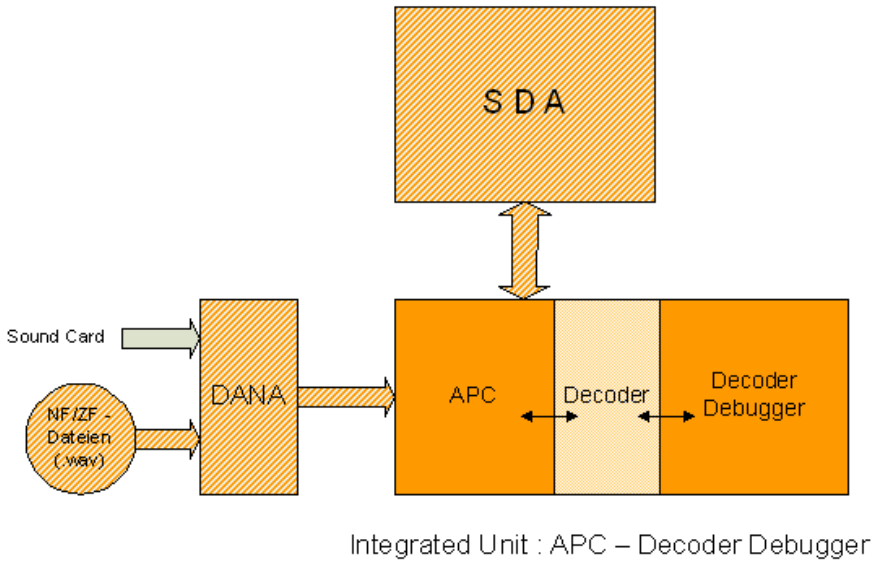

Figure 4: Online Debugging Concept

Start the go2DECODE system as usual. The decoder and modem to be observed must be present in the loaded modem type list. Start the production manually or automatically. From the debugger user interface, now select the complete modem from the modem type list rather than just the decoder. This list is displayed in an additional window in the debugger user interface, representing a copy of the SDA modem type list.

Subsequently, debug the decoder or modem, resp., during automatic production, both in search phase and in production phase. Define breakpoints to be active in both search and production phase, or in production phase only. On reaching the breakpoints, single stepping as well as system monitoring can be done the same way as in pure **<Decoder Debugger**

Offline> mode, and the effects of decoder commands, editing of demodulator parameters or automatic production control can be observed as well.

3.2 Selecting the Decoder Debugger – Online User Interface

Open the online debugging user interface via menu item **<Extras><Decoder Debugger Online>** :

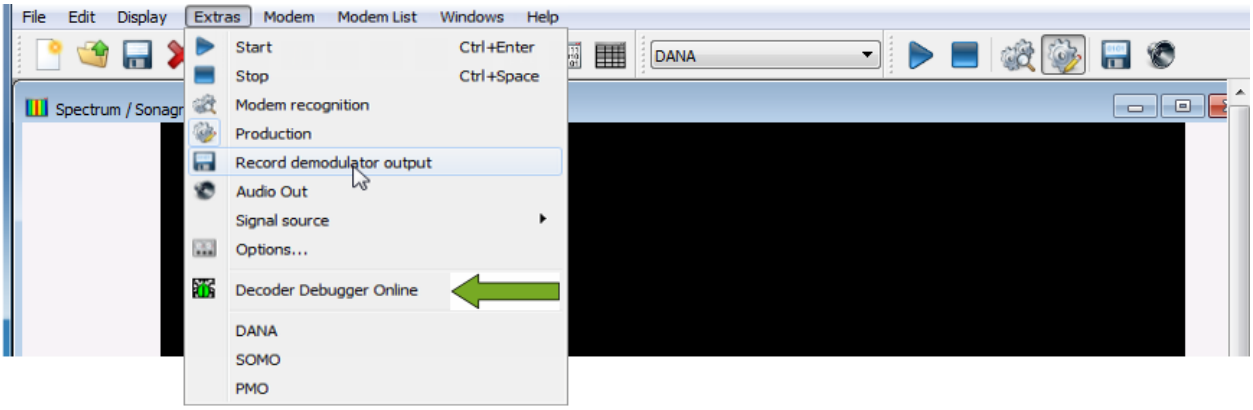

Figure 5: Select Decoder Debugger

The following main window will appear, a slightly modified window compared to one in **<Decoder Debugger Offline>** mode:

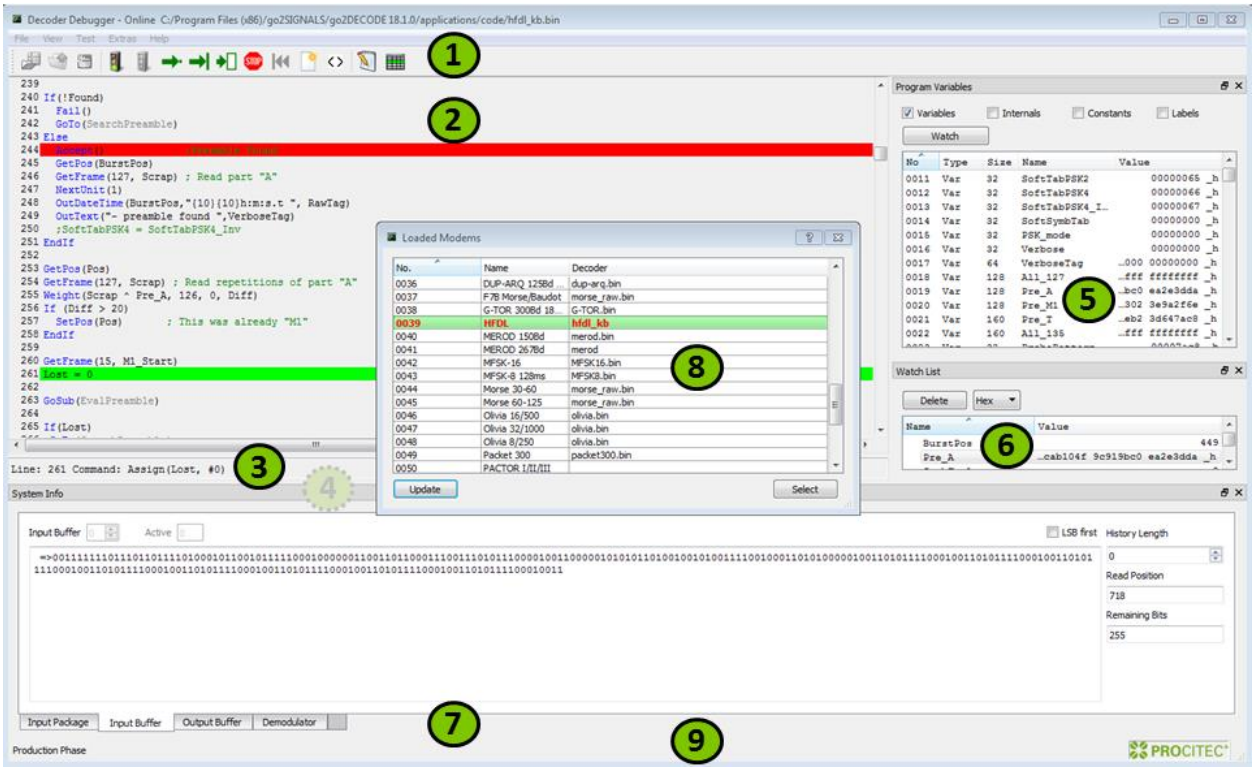

Figure 6: Decoder Debugger – Online Main Window

1. Menu Bar and Toolbar
   Central functions can be accessed via the menu bar. For most of these functions toolbar icons are provided as well.
2. Source Display
   The source code of a loaded decoder is displayed in this window. Breakpoints and the subsequent program line are highlighted in different colors. This display is read-only. Use the editor for modifying source codes.
3. Command Line Display
   Each source code corresponds to a single, or a set of, runtime commands which are composed during compilation. The next runtime command is displayed in this line.
4. Result Display
   → missing
5. Variables List
   This pane displays all program variables and their current values.
6. Watch List
   This pane may contain a selected subset of the Variables List. Values can be displayed in different formats.
7. Buffer and Parameter Display Tabs
   The panes on these tabs contain:
   - Last input package received from demodulator
   - Input buffer with current read pointer indication
   - Output buffer containing current output command results
   - Demodulator parameters
   - Additional display for selectable long variables
8. List of Loaded Modems
   This list corresponds to the modem type list loaded via the application SDA. Once these modems are loaded into the APC (now as part of the Decoder Debugger), they will be displayed in this list. Modems can be marked, indicating their current operating status.

Bold:	The modem/decoder is selected for monitoring
Yellow background:	The modem is processed in search mode
Green background:	The modem is processed in production mode
Red characters:	Processing has stopped at a breakpoint within this modem/decoder

9. Status Fields
   These fields indicate the operating status of the internal APC, and the connection status to the external applications SDA and DANA.

3.3 Selecting a Decoder for Debugging

As the decoder will be selected in conjunction with a complete modem via the modem list, remember to make the following preparations before selecting the decoder:

- Start SDA if it is not running. Successful connection to the SDA will be confirmed in the status fields.
- Load a modem type list via SDA containing the decoder of interest. On successful loading, a copy of this modem will be visible above the main window. If the list is hidden, select:
- **<View><Modem List>**, 

Press the button **<Update>** to ensure the latest changes are updated as well.

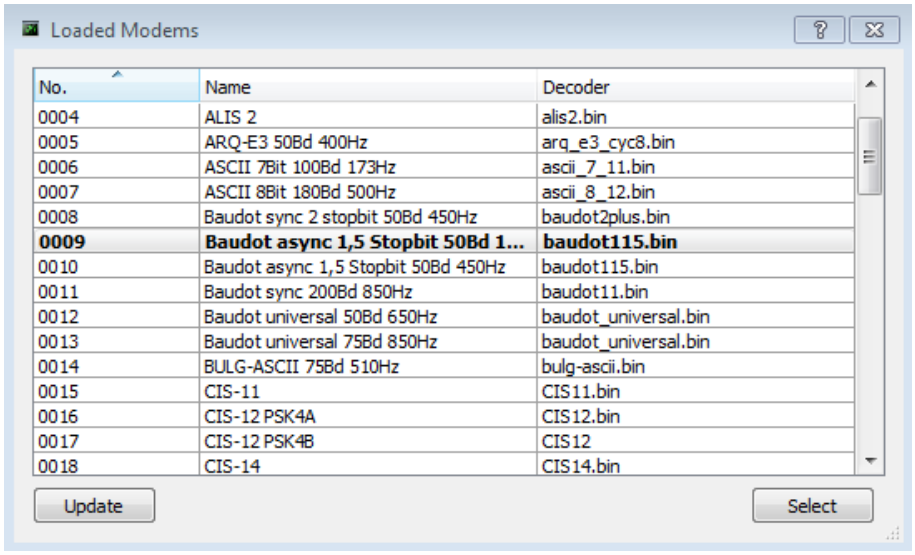

Figure 7: Loaded Modems List

To select a decoder as part of a modem, use **<Select>** or just double click the respective modem. As a result, the decoder source code will be shown in the source display.

Start the production via SDA as in the standard go2DECODE environment. The selected decoder can be accessed by setting breakpoints in any code line in the debugger source display. Subsequently, apply the debugging functions described in the next chapter.

4 Running and Analyzing Decoder Programs

4.1 Free Run

Decoders are started and executed continuously via:

- **<Test><Run>**, F5, 

To stop the running decoder before the end of input data, use:

- **<Test><Hold>**, Esc, 

4.2 Breakpoints

Breakpoints can be set in any source code line. Either double click the desired line number or select any valid code line (no blank or comment lines) and use:

- **<Test><Breakpoint>**, F9, 

The decoder application will run until it reaches the selected line. It can be continued in a stepping mode or continuous mode. Set as many breakpoints as required. To remove a single breakpoint, repeat the procedure in the same line.

In *Online* mode, breakpoints can be set in more than one decoder. They will be active whether the decoder is currently selected or not. Also in *Online* mode, you may specify whether the breakpoint shall be valid in production mode only or also during the search phase in automatic production, by activating the checkbox before:

- **<Test><Breakpoints incl. Search>**

Alternatively, the decoder program can run up to a dedicated line by setting the cursor to this line and selecting:

- **<Test><Run to Cursor>**

To remove all breakpoints, either in the current decoder or all breakpoints in all decoders, enter:

- **<Test><Delete All Breakpoints><In this Decoder>**, Shift+F9
- **<Test><Delete All Breakpoints><In all Decoders>**, Ctrl+Shift+F9

4.3 Stepping Modes

Once a decoder is stopped, i.e. at the beginning or at a breakpoint, it can be continued in three different stepping modes.

4.3.1 Single Line

The *Single Line* command will process the code covered by the next source code line.

- **<Test><Single Line>**, F10, 

4.3.2 Single Step

The *Single Step* command will process the next step compiled and displayed in the command line.

- **<Test><Single Step>**, F11, 

4.3.3 Single Package

Decoder input data is received in packages of various sizes. The sizes are determined by the demodulator. New packages are requested by the decoder whenever this is required in order to continue. Consequently, this function will continue until the next input package is requested.

- **<Test><Single Package>**, Shift+F10, 

4.4 Restarting the Decoder

This function is available in Offline Mode. The decoder and input data are reset to their start condition by the command:

- **<Test><Reset>**, F12, 

4.5 Clear Result

This function will clear the result display which is used in Offline Mode.

- **<Test><Clear Result>**, F8, 

5 Monitoring Variables

Variables can be monitored each time the run is stopped due to breakpoints or step commands.

5.1 Variables List

The Variables List contains all variables, labels and constants used in the program:

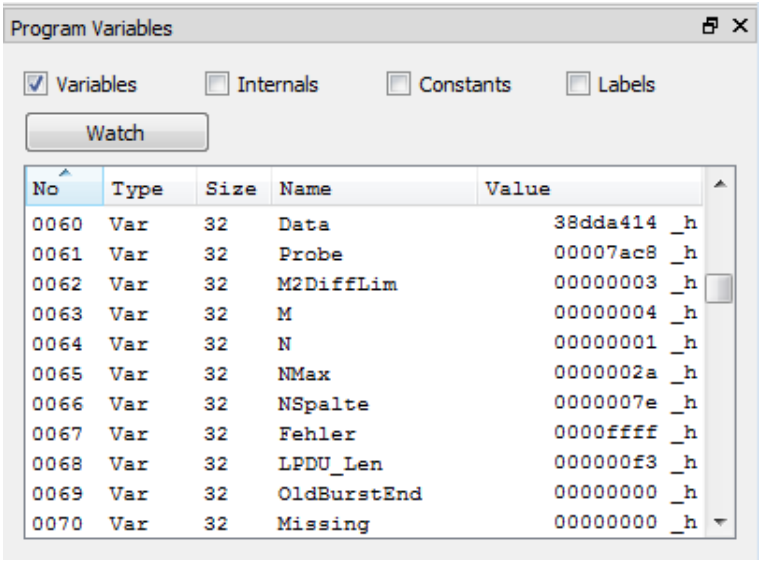

Figure 8: Variables List

Each variable is described by the following attributes:

Attribute	Description
No	Variable number as used in the compiler listing (i.e. file with extension .list produced during compilation)
Type	Variable type as described in chapter Variable Types
Size	Size of variable in bits
Name	Variable name
Value	Variable value

Table 1: Variables Attributes

5.1.1 Variable Types

There are the following types of variables:

- Declared program variables, shown as 'Var'.
- Internal variables, shown as 'Intern'. These variables are used either for system control or as auxiliary variables, generated during compilation, e.g. resolving nested operations.
- Constants, shown as 'Const', as used in the source code.
- Labels as used in the source code or generated during compilation, e.g. resolving branch constructions. The values correspond to program step numbers used internally.

5.1.2 Selecting Variables

Single variables can be transferred to the watch list. To do so, select the variable line and press the **<Watch>** button, or use the context menu. Alternatively, select the desired variables via double click.

Long variables which cannot be fully displayed may be transferred to the Long Variables Display on the buffer display tabs. Select the variable with the right mouse button and the context menu item **<full value size>**.

5.1.3 Changing Variable Values

You may change the variable values with types 'Var' and 'Intern' at any time. Use the item **<edit value>** in the context menu.

5.2 Watch List

The Watch List contains a subset of the Variables List. Variables are transferred to the watch list:

- From the Variables List as described in chapter Selecting Variables on page 14

  or
- From the source display via double click & drag

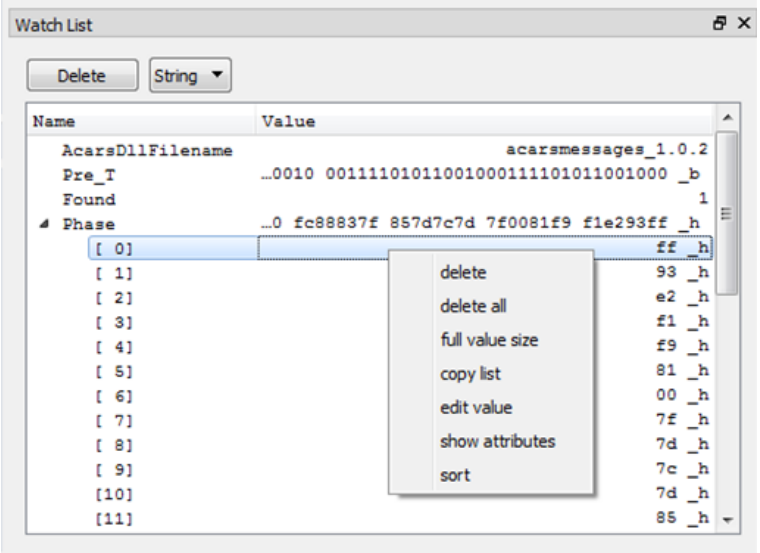

Figure 9: Watch List

These variables value annotation formats can be selected individually for each variable:

Selection	Description
Hex	hexadecimal
Dec	decimal (first 32 bits)
Oct	octal
Bin	binary
String	ASCII-String

Table 2: Annotation Settings

Array variables can be expanded to see each element separately. The list can be further managed via the right mouse button context menu:

Function	Description
delete	delete selected variable
delete all	clear the watch list
full value size	copy the selected variable to the Long Variables Display
Copy list	copy the list to the clipboard
edit value	modify the selected variables value. The values can be editeddirectlyas well.
show attributes	show the variables size and array dimensions
sort	toggle between automatic and manual sorting methods. Automatic sorting will sort the table based on a selectable column (name or value). Manual sorting allows placing a new variable via drag & dropping at a dedicated position. The table can be sorted by later internal drag & drop operations as well.

Table 3: Context Menu Functions

5.3 Long Variable Display

This display is provided to show the value of long variables to their full extent. Variables are transferred to the long variable list:

- From the Variables List as described in chapter Selecting Variables on page 14 or
- From the Watch List in the same way

- From the source display by double clicking & dragging it to the display's tab label

As a result, the tab label will assume the variable's name and the value will be shown in the selected format. Possible display formats are binary, hexadecimal and ASCII text. The box can be edited and the values will be updated on activation of **<Apply>**.

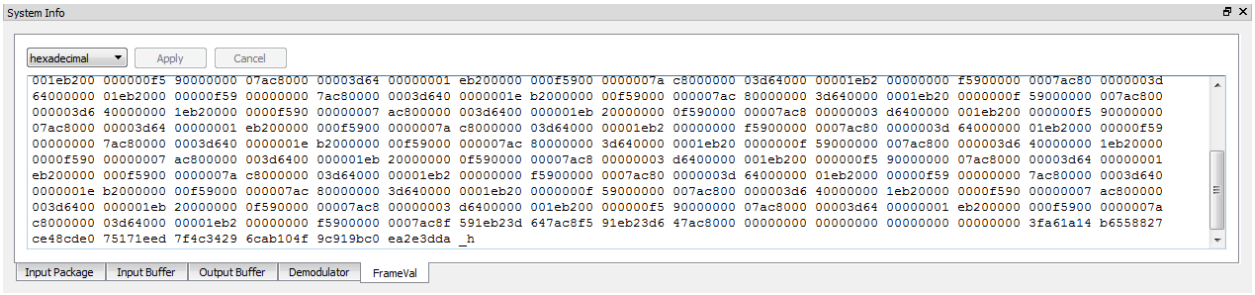

Figure 10: Long Variable Display

6 Displays

6.1 Buffers

Incoming and outgoing data pass through intermediate buffers as displayed in Figure 10. The input package contains the demodulated bit stream received from the demodulator. This package is transferred to the input buffer after optional pre-processing. The main program receives all inputs (e.g. for commands *Search…* and *Get…)* from the input buffer. Commands *Out...* will write to the output buffer which is flushed before the next input package is received.

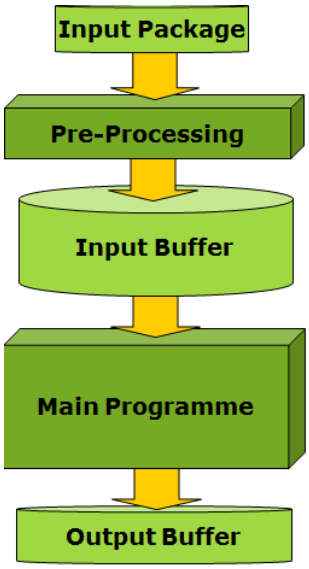

Figure 11: Buffering Concept

6.1.1 Input Package

The input package bit stream is shown in conjunction with a set of parameters received in parallel. The bit stream can be displayed in most or least significant bit first order, controlled via the checkbox **<LSB first>**. The display also shows intermediate results of pre-processing steps, which can modify the package in question. The additional parameters on the right of the display are as follows:

6.1.1.1 Phase

This parameter indicates whether the package is received during search or production phase. The search phase is the initial phase of automatic production process, when all pos-

sible modems are tested. This phase only can be observed in **<Decoder Debugger Online>** mode. The production phase follows the search phase if a positively responding modem has been found. In **<Decoder Debugger>** mode, only the production phase is simulated.

6.1.1.2 Package Size

This value describes the current number of input symbols per input channel contained in this input package. To obtain the total number of received bits, this value must be multiplied with the symbol size and the number of channels.

6.1.1.3 No. of Channels

Number of demodulator channels which has been defined as a demodulator parameter for the respective modem.

6.1.1.4 Symbol Size

This indicates the number of bits per incoming input symbol. This value corresponds to the modulation order (or "valence") defined with the demodulator.

6.1.1.5 Symbol Rate

This value is the result of the actual symbol rate measured. It may deviate from the nominal symbol rate defined with the demodulator.

6.1.1.6 Time/Date

This time designates the arrival time of the first symbol of this package. It can be referred to the system clock or file time, depending on the settings in DANA.

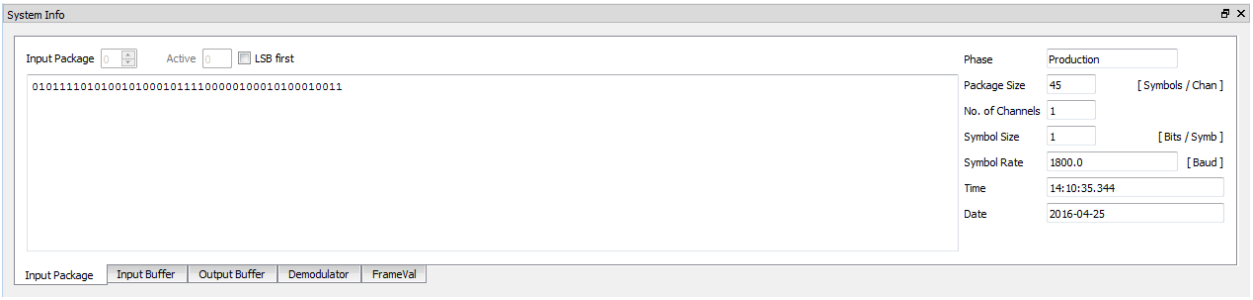

Figure 12: Input Package Display

Multiple input packages will be present when the decoder executes commands to split a multi-channel input package into single channel packages. In this case, the spin box **<Input Package>** is enabled to switch the display between channels. The package currently treated by pre-processing is indicated in the box **<Active>**.

6.1.2 Input Buffer

The input buffer is displayed in most significant bit first or least significant bit first order. The input buffer pane additionally shows the current position of the read pointer after "=>" respectively before "<=", and as an absolute value in the box **<Read Position>**. The number of bits from the read pointer to the last incoming bits is displayed in the box **<Remaining Bits>**. The displayed number of bits before the read pointer can be set in the spin box **<History Length>**. **<Read Position>** is the absolute value of the read pointer position, designating the number of bits since the start of input bit stream.

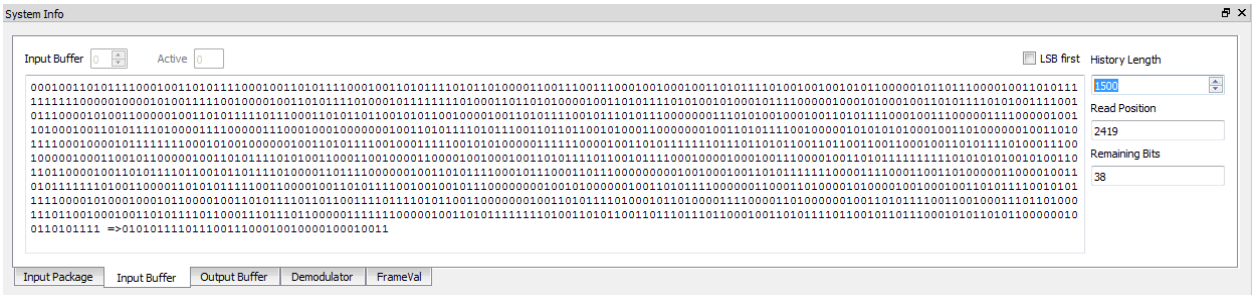

Figure 13: Input Buffer Display

The characters „<B" in the input buffer pane marks the beginning of a detected burst, "b>" designates a burst end. Input symbols exceeding 1 bit are separated by blanks.

6.1.3 Output Buffer

The output buffer display shows the result of commands *Out...* in unfiltered XML format. It is flushed at the time of data exchange, i.e. the time a new input package is requested to continue decoder processing.

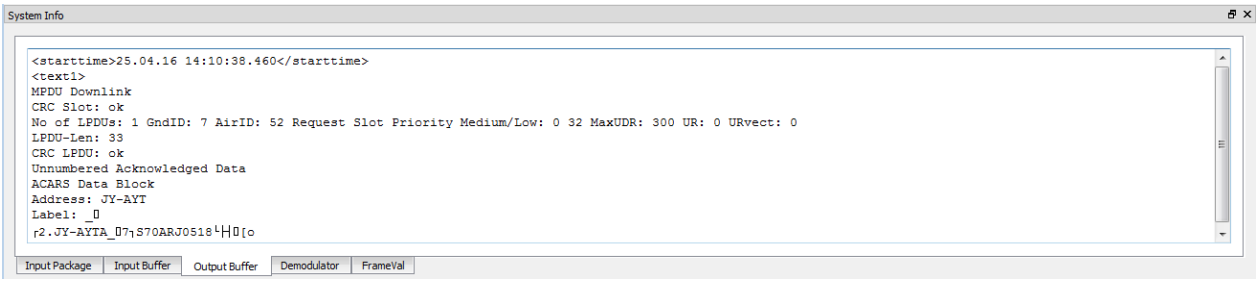

Figure 14: Output Buffer Display

6.2 Demodulator Parameters

The tab **<Demodulator>** shows the current demodulator settings as they would be read by the decoder at this point. Parameters which are not relevant for the current type of demodulator are shaded in grey. They are displayed nevertheless as they may be relevant when changing the type of demodulator during the decoding program.

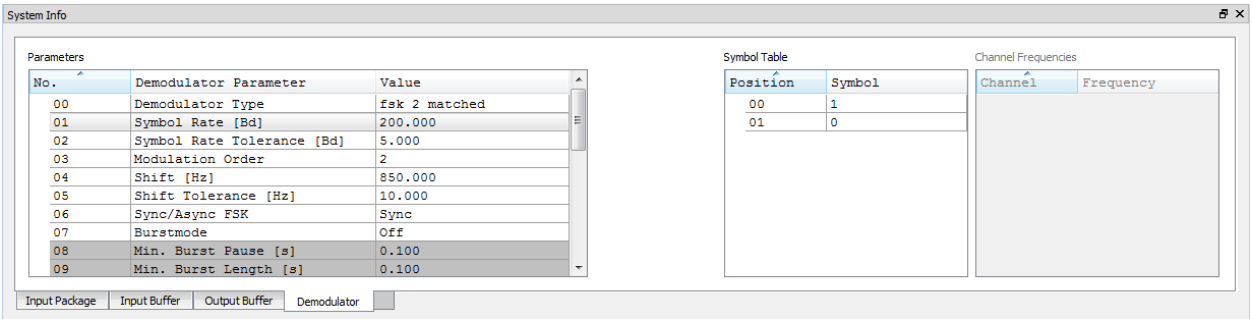

Figure 15: Demodulator Parameters Display

6.3 Result

This pane shows the decoded text results once the output buffer has been flushed. It will be present exclusively in Offline Mode as in Online Mode the software will use the SDA result fields.

The results can be displayed as complete raw XML text or previously pass a definable output filter. Call the filter table by:

- **<View><XML Filter>**, Ctrl+F,

A list of standard XML tags is shown which can be modified as required. Add or delete tags, and interpret the list via the selection in the top left drop-down list box.

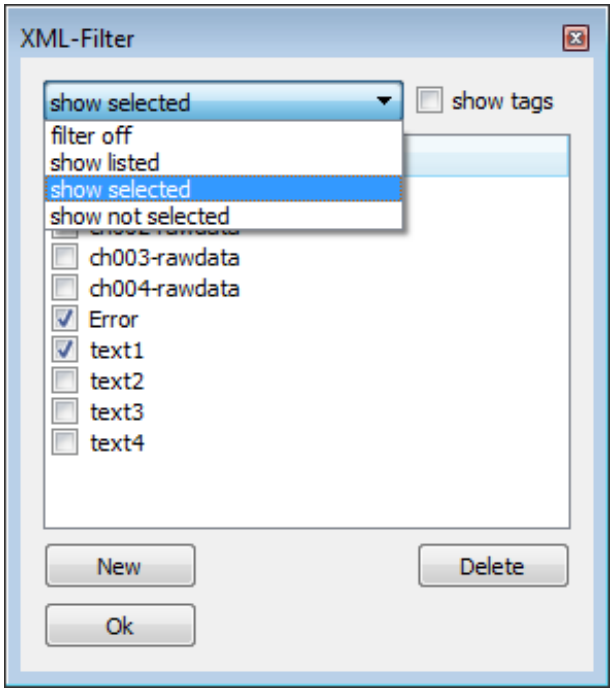

Figure 16: XML Filter Table

The content of a result filed can be deleted at any time by:

- **<Test><Clear Result>**, F8,

7 Decoder Editor

This integrated editor serves to modify and compile the decoder source code. For a detailed description, please refer to the go2DECODE Instruction Manual. The decoder editor is called by:

- **<View><Editor>**, Ctrl+E, 

The editor dialogue window can be freely positioned on the screen.

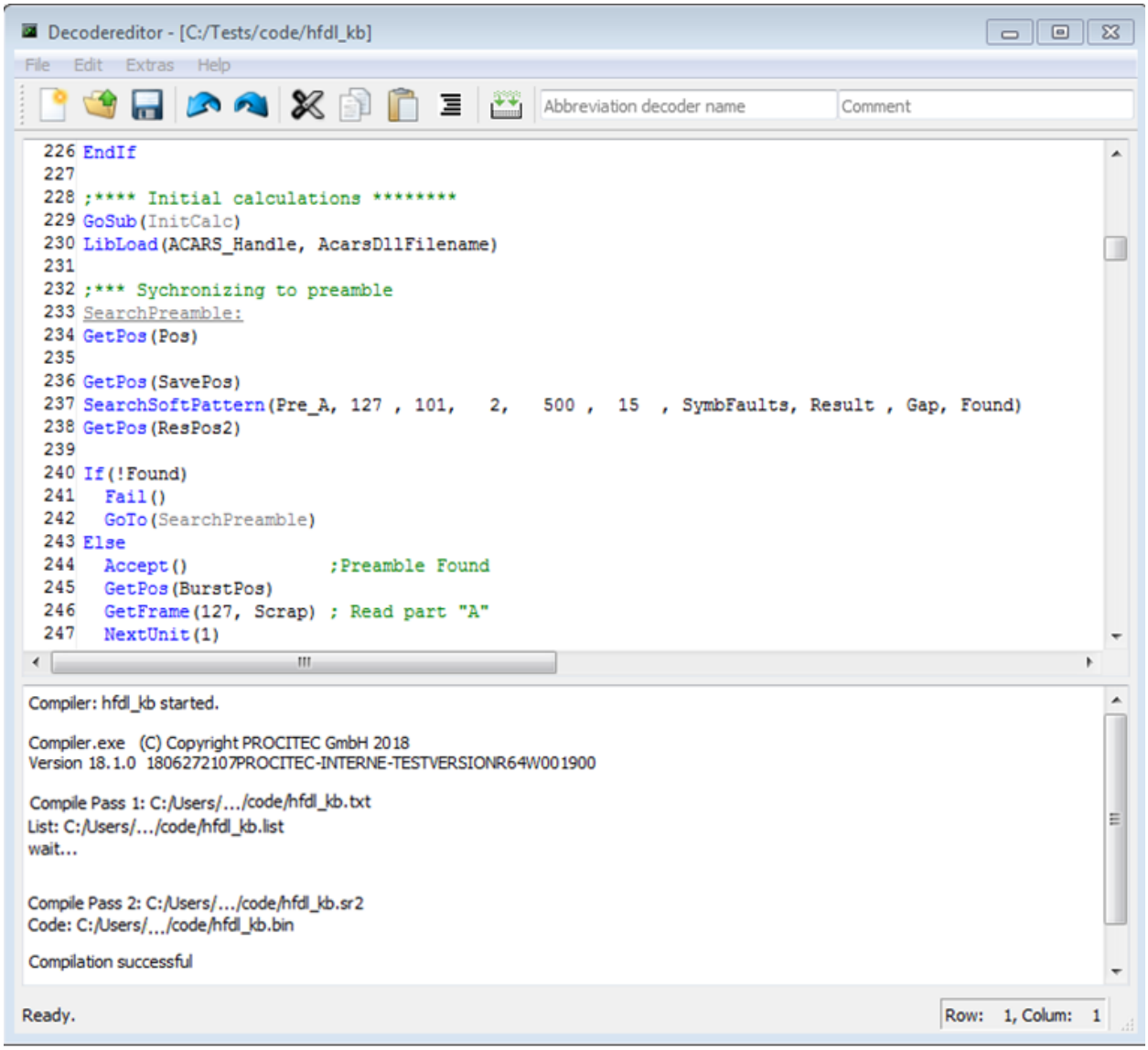

Figure 17: Decoder Editor Dialogue Window

Note: Any changes in the source text will not be applied until successful compilation, started via the icon and an update of the runtime environment. To update, proceed as shown in chapter Offline Mode.

7.1 Offline Mode

Use the command:

- **<File><Update>**,

or simply close the editor and acknowledge the dialogue requesting an update.

7.2 Online Mode

Make sure the source code has been selected from, and compiled into, the designated subdirectory */code*. The update is made on restarting the manual or automatic production via SDA.

8 Appendix

8.1 Support and Document-ID

Requests and suggestions?

Any requests and suggestions about our products will be highly appreciated. We would be glad to receive your information.

Further questions? We are pleased to support you!

If you still have any questions, don't hesitate to ask your friendly and helpful Support.

For a rapid assistance please have the following required information on hand:

- Document-ID GODDDE19010020181120
- Product name and und version
- License
- Dongle No.
- Use case
- Operating system
- Language of the operating system
- Other applications running
- Screenshot

8.2 Contact

You can reach us at:

PROCITEC GmbH
Rastatter Strasse 41
D-75179 Pforzheim

Phone: +49 7231 15561 0
www.go2signals.de
service@procitec.de

9 List of Figures

10 List of Tables

11 Index