asx

COLLABORATORS

	TITLE : asx	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		October 23, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

asx

1.1 asx.guide

```
==========================================================================

                          A S X

                          v1.22
==========================================================================
```


*** This Manual ***

TABLE OF CONTENTS

1.2 about this manual

About This Manual
==========

Welcome to the AmigaGuide version of the ASX manual.

This manual is a part of the ProAsm manual, which was originally written in TeX and texinfo. You are currently looking at the AmigaGuide version of the ASX part. I used various tools to convert the primal TeX source into AmigaGuide. They did quite a good job, but the result was over 700KBytes of length! As a consequence I had to shorten the whole guide file by hand. Now it is very possible that when browsing through this manual you will find sentences like:

This node (page) intentionally left blank.
or
This node (page) was intentionally shortened.

Please forgive me for that. I had also to skip various tables that can be found in the original printed manual.

Caused by the convertion of the TeX source of the manual into this AmigaGuide file and the need of some small reorganisation of it, some of the links go nowhere. Sorry for that, but I had not the time to check all links. I did check a lot of them, but unfortunately there are still some of these dead links in the guide file. If you should find any, please feel free to inform me (
Daniel Weber
).

If you find some strange (unusal) parts in this AmigaGuide file, please keep in mind, that this guide initially was written as a book. You can order a copy of the printed manual, if you wish. Please read the registration document (register.doc or Registration) for further information.

Daniel Weber
Zurich
November 1995

1.3 notices

Copyrights
==========

This AmigaGuide file reflects version 1.22 of ASX.

Trademarks
==========

Disclaimer
==========

The information, the ASX program, and all the associated utilities are provided "as is" without warranty of any kind, either expressed or implied. The entire risk as to the accuracy of the information herein is assumed by you. Daniel Weber does not warrant, guarantee, or make any representations regarding the use of, or the results of the use of, the information, the ASX program, or the associated utilities in terms of correctness, accuracy, reliability, currentness, or otherwise. In no event will Daniel Weber be liable for direct, indirect, incidental, or consequential damages resulting from any defect in the information, the ASX program, or the associated

utilities even if he has been advised of the possibility of such damages.

1.4 introduction

ASX - The ProAsm User Interface
==========

ASX is a user interface for the ProAsm assembler that is implemented as a commodity. Through the use of the commodities.library it can be installed on any hotkey and controlled fully with the Commodities Exchange program.

ASX loads the ProAsm assembler which than can be accessed using the ARexx interface, the asx.library, or the AppIcon possibility. The asx.library and the AppIcon can optionally be enabled or disabled.

The ARexx commands provide a method of controlling ASX from an external program. These ARexx commands can be used to create an integrated programming/development environment with any application that offers an ARexx interface. For example, a program can be written by a programmer on its favourite ARexx equipped texteditor, then an ARexx command can be sent to ASX to assemble the source code. Any error messages and warnings of the assembly are stored by ASX. Using commands such as NEXTERROR and PREVERROR the texteditor is capable to position the cursor in the line of the first error. After correcting that error a single keystroke can jump to the next error, or another keystroke can jump back to the previous error.

Another method of controlling ASX is the use of the optional asx.library. The various functions that this library offers can be used to design own user interfaces with ease.

The AppIcon possibility is another visual user interface that allows one or more source file icons to be assembled by just dropping them over the ASX AppIcon.

Frequently used include files can be loaded residently and managed by ASX. Such residently loaded include files reduce assembly time since they do not need to be loaded each time the assembler is called.

Preferred include file paths can be added to a database that is managed by ASX. During assembly the assembler uses then this database to know where to look for the include files.

ASX also provides a feature called the source manager, which offers a possibility to manage the current project per hotkey. Through the use of an ARexx script file the user can easily define the action that has to be fulfilled when an entry in the source manager window has been selected. Almost all shooting matches of the source manager can be set by the user to his wishes and needs to allow a wide range of flexibility.

For an easy use of ASX, it comes along with an on-line help feature. Commodore's AmigaGuide is used to display the help text to

the user.

1.5 system requirements

System Requirements
==========

ASX requires Kickstart and Workbench 2.0 or higher.

1.6 pro.guide/Starting from Workbench

Starting from Workbench

The easiest way to run ASX is from the Workbench. A double-click on its icon loads the program. The ENV:asx.prefs configuration file will be loaded and processed immediately, if not otherwise desired (see the LOCALPREFS tooltype below). The configuration file is an ASCII text that can be edited using a texteditor if needed.

If ASX is already running and you invoke it again, it will open the user interface window of the already running version.

When ASX is started from the WorkBench, the following tooltypes are recognized:

CX_PRIORITY=n
Set priority of the commodity to n. By default, ASX uses a priority of 0.

CX_POPKEY=hotkey
This tooltype allows you to specify the hotkey for ASX to hotkey. Hotkey must be a valid key sequence for a hotkey. If you enter an incorrect description, a message will be displayed or a requester is opened to inform you. (The hotkey format is described in the 'AmigaDOS 2.0 manual'). By default, lcommand help is used as hotkey.

If you have defined a hotkey, ASX can be called up at any time by pressing that key sequence.

CX_POPUP=YES|NO
This options allows you to select whether the user interface window of ASX is opened at startup or not. YES must be used to open the user interface window. By default, ASX does not open the window.

APPICON
If this tooltype is selected, ASX displays an AppIcon on the workbench screen. See
The AppIcon
.

ICONX=x
ICONY=y
These tooltypes allow you to specify the position of the AppIcon on the WorkBench screen relative to the top left edge. The x and y coordinates are integer values that describe the top left position.

Normally, the AppIcon will be positioned automatically in an appropriate position on the screen, but you may sometimes want to override this with your own preferred coordinates.

NOLIBRARY
If this tooltype is selected, ASX will not add the asx.library to the system. (See
The asx.library
). By default, the library is installed.

ASMPATH=assembler
Using this tooltype you can specify the location and the name of ProAsm. assembler describes the path and filename of the ProAsm assembler. By default ASX searches the assembler as pro in the following locations: asm: and the current directory.

SWAP
This tooltype tells ASX to load ProAsm each time when used. By default, the assembler is once loaded residently. Loading the assembler residently can be useful on disk based systems. This tooltype is also recommended on systems with little memory.

ERRFILE=filename
This tooltype defines the name of the error file to filename that is automatically used for all assembly task started by ASX (via ARexx and asx.library).

HEADERFILE=filename
This tooltype defines the header filename that is automatically used for all assembly task started by ASX (via ARexx and asx.library).

CONFIGFILE=filename
This tooltype defines the configuration filename that is automatically used for all assembly task started by ASX (via ARexx and asx.library).

NOREQ
If this tooltype is selected, ASX will not open any message requesters. This can be useful if you do not want to be interrupted by these requesters. But note that also all error message reported by ASX will be suppressed silently.

NORASTER
To get a stronger 3D-look, ASX fills the background of all its windows with a pattern. If you do not want to have this pattern as background, you can unselect this tooltype and it will be written automatically to the settings file when you save settings. By default, a rastered background is displayed.

```
LOCALPREFS=filename
     This tooltype tells ASX to use the file filename as configuration
     file, instead of the default configuration file named
     ENV:asx.prefs.

NOMASTER
     This tooltype can be used to forbid the source manager.  By
     default, the source manager is enabled.

AREXXSCRIPT=filename
     This tooltype allows you to specify the ARexx script that is
     executed when an entry in the source manager is selected.
     Filename is the complete name and path of the ARexx script.
     By default, rexx:pro/sourcemanager.rexx is used.

SMHOTKEY=hotkey
     This tooltype allows you to specify the hotkey for the source
     manager to hotkey.  Hotkey must be a valid key sequence for a
     hotkey.  If you enter an incorrect description, a message will be
     displayed or a requester is opened to inform you.  (The hotkey
     format is described in the 'AmigaDOS 2.0 manual').  By default,
     lcommand shift help is used as hotkey.

     If you have defined a hotkey, the source manager can be called up
     at any time by pressing that key sequence.

SMPUBSCREEN=screenname
     This tooltype can be used to specify the public screen the source
     manager should be opened on.  Screenname is the name of the
     desired public screen.  By default, the source manager opens its
     window on the front public screen.

NOFALLBACK
     This tooltype forbids the source manager window to open its window
     on the default public screen if the explicitly specified public
     screen is not available.  By default, the source manager does
     "fall back".
```

1.7 pro.guide/Starting from the Shell

```
                Starting from the Shell
-----------------------

    ASX can be started from the shell by typing its name and the
parameters you need.  Enter ASX ? for a shell usage template.

    The ENV:asx.prefs configuration file will be loaded and processed
immediately, if not otherwise desired (see the LOCALPREFS argument
below).  The configuration file is an ASCII text that can be edited
using a texteditor if needed.

    If ASX is already running and you invoke it again, it will open the
```

user interface window of the already running version.

There is no need to use the AmigaDOS Run command to start ASX as it will automatically self-detach from the shell or CLI that it was started from.

The command line options override the default settings of ASX. In the following you find a list of all command line parameters supported by ASX. See also see
Starting from Workbench
.

CX_PRIORITY n
Set priority of the commodity to n. By default, ASX uses a priority of 0.

CX_POPKEY hotkey
This argument allows you to specify the hotkey for ASX to hotkey. Hotkey must be a valid key sequence for a hotkey. If you enter an incorrect description, a message will be displayed or a requester is opened to inform you. (The hotkey format is described in the 'AmigaDOS 2.0 manual'). By default, lcommand help is used as hotkey.

If you have defined a hotkey, ASX can be called up at any time by pressing that key sequence.

CX_POPUP YES|NO
This argument allows you to select if whether the user interface window of ASX is opened or not. YES must be used to open the user interface window. By default, ASX does not open the window.

APPICON
If this argument is given, ASX displays an AppIcon on the workbench screen. See
The AppIcon
.

ICONX x
ICONY y
These arguments allow you to specify the position of the AppIcon on the WorkBench screen relative to the top left edge. The x and y coordinates are integer values that describe the top left position.

Normally, the AppIcon will be positioned automatically in an appropriate position on the screen, but you may sometimes want to override this with your own preferred coordinates.

NOLIBRARY
If this tooltype is selected, ASX will not add the asx.library to the system. (See
The asx.library
). By default, the library is installed.

ASMPATH assembler
this argument allows you to specify the location and the name of

ProAsm. assembler describes the path and filename of the ProAsm assembler. By default ASX searches the assembler as pro in the following locations: asm: and the current directory.

SWAP
This argument tells ASX to load ProAsm each time when used. By default, the assembler is once loaded residently. Loading the assembler residently can be useful on disk based systems. This tooltype is also recommended on systems with little memory.

ERRFILE filename
This argument defines the name of the error file to filename that is automatically used for all assembly task started by ASX (via ARexx and asx.library).

HEADERFILE filename
This argument defines the header filename that is automatically used for all assembly task started by ASX (via ARexx and asx.library).

CONFIGFILE filename
This argument defines the configuration filename that is automatically used for all assembly task started by ASX (via ARexx and asx.library).

NOREQ
If this argument is given, ASX will not open any message requesters. This can be useful if you do not want to be interrupted by these requesters. But note that also all error message reported by ASX will be suppressed silently.

NORASTER
To get a stronger 3D-look, ASX fills the background of all its windows with a pattern. If you do not want to have this pattern as background, you can select this argument and it will be written automatically to the settings file when you save settings. By default, a rastered background is displayed.

LOCALPREFS filename
This argument tells ASX to use the file filename as configuration file, instead of the default configuration file named ENV:asx.prefs.

NOMASTER
This argument can be used to forbid the source manager. By default, the source manager is enabled.

AREXXSCRIPT filename
This argument allows you to specify the ARexx script that is executed when an entry in the source manager is selected. Filename is the complete name and path of the ARexx script. By default, rexx:pro/sourcemanager.rexx is used.

SMHOTKEY hotkey
This command line argument allows you to select the hotkey for the source manager to hotkey. Hotkey must be a valid key sequence for a hotkey. If you enter an incorrect description, then a message

will be displayed or a requester is opened to inform you. (The hotkey format is described in the 'AmigaDOS 2.0 manual'). By default, lcommand shift help is used as hotkey.

If you have defined a hotkey, the source manager can be called up at any time by pressing that key sequence.

SMPUBSCREEN screenname
This argument can be used to specify the public screen the source manager should be opened on. Screenname is the name of the desired public screen. By default, the source manager opens its window on the front public screen.

NOFALLBACK
This argument forbids the source manager window to open its window on the default public screen if the explicitly specified public screen is not available. By default, the source manager does "fall back".

1.8 pro.guide/The User Interface

The User Interface

This section describes the user interface of ASX. As user interface, ASX presents six different windows and an AppIcon. Everything in these windows should be fairly self-explanatory. If ASX is started from the Workbench or the command line, no window will be opened except the CX_POPUP=YES tooltype or command line option was selected. The window can also be opened by using the hotkey, the Exchange program that comes along with the Workbench 2.0 or higher, or by starting ASX again.

A window opened by ASX can present itself in two different forms. A large window occupying nearly one third of a normal 640x400 Workbench screen, and a 'zoomed' window, where only the close gadget, the zoom gadget, the depth gadget, and the window title are visible. If you click the zoom gadget, ASX's window will toggle between the large and the 'zoomed' window.

When using ASX, you may get in contact with one or more of the following basic types of gadgets:
action
Action-gadgets appear as raised buttons. Normally the buttons are labeled with a small text that is written inside the gadget. Selecting such a button causes an immediate action. For example, the Quit gadget in the main window causes ASX to quit.

list
listview
Generally list-gadgets appear as a set of control gadgets. A small scrollbar, arrow gadgets, and a string area. ASX may have additional control gadgets attached to its list-gadgets. These

gadgets are a string-gadget, an add-gadget (in one case), del-gadget, a select-gadget, top-, up-, down-, bottom-, and a sort-gadget. Use the scroll bar and the arrow gadgets to position the scrolling area to a specific entry. To add new entries to the list, select the add- or select-gadget (if available) or type the new entry name into the string-gadget (below the scroll area) and press RETURN. Entries can be removed from the list by selecting the entry to be cleared in the scroll area (to copy its name into the string-gadget), and selecting the del-gadget.

The top-, up-, down-, and a bottom-gadget can be used to move the entry to the desired position in the list. Top and bottom move the item directly to either the top end or the bottom end of the list. Up and down move the selected item one position upwards or downwards in the list. The sort-gadget sorts all entries in the list alphabetically. Some gadgets can be disabled if they cannot be used (e.g. on empty lists).

string
String-gadgets appear as raised ridge around a text area. Any string can be entered by clicking (activating) within the text area and typing your string. When you have finished typing, press the RETURN key to confirm or the TAB key to confirm and jump to the next string-gadget.

check
Check-gadgets appear as small rectangular raised buttons, with labeled with a text on the left or the right side of the gadget. When the gadget is selected, a small checkmark appears, or disappears when unselected.

1.9 asx.guide/The_Main_Window

The Main Window
.....

This is the main window that shows up when ASX's user interface is opened. From here you can access the additional windows or handle the ASX concerning settings.

The Gadgets
.....

Assembler
By clicking on this gadget, you will switch to the assembler settings window. See
The Assembler Window
.

IncDir
Go to the IncDir window. There you can specify the directories where the assembler has to search for include files. See

The IncDir Window

.

Residents
: By the selection of this gadget you will switch to the residents settings window. See
The Residents Window
.

Sources
: Selecting this gadget switches to the source manager preference window. See
The Source Manager Preferences Window
.

running assembly jobs
: This listview-gadget shows all running jobs. A job is represented by its ARexx port name and its ID. See
The ARexx Interface
, for
the description of the DefineID, the FindID, and the GetPort commands.

asx.library
: This gadget can be used to control the use of the asx.library. If it is checked (selected) the asx.library is installed. See

The asx.library
.

AppIcon
: If this check-gadget is selected an AppIcon is put on the Workbench screen. See
The AppIcon
.

Hot Key
: This allows you to select what key sequence is used to open the ASX main window when it is hidden, by entering the key description into the string gadget. Any standard commodities key description can be given. If you enter an incorrect description, a message will be displayed or a requester is opened to inform you. (The hotkey format is described in the 'AmigaDOS 2.0 manual'). By default, lcommand help is used as hotkey.

: If you have defined a hotkey, ASX can be called up at any time by pressing that key sequence.

Save
: Stores the current settings to ENVARC:asx.prefs and ENV:asx.prefs.

Hide
: This function closes the ASX user interface, but leaves ASX installed in the background. This allows you to have ASX constantly available, while using the minimum amount of memory possible.

: You can also hide ASX by clicking on the close gadget of the

window.

Quit
This function tells ASX to exit. Usually, ASX can quit immediately. However, sometimes when you try to quit, another program has not ended the ARexx job or closed the asx.library. A requester will inform you by displaying the number of jobs that are still unfinished yet and ASX refuses to quit.

The Menus
....

Frequently used menu items in ASX have keyboard equivalents so that you can use them without having to move from the keyboard to the mouse if you don't want to. The keyboard equivalents are not case sensitive.

Menu items that have a sequence of dots following the item name open a requester as part of their execution. Items with a question mark (?) indicate that they use a checkmark to show that their associated mode is active.

Project

Open...
The Open... (AMIGA-O) command allows you to load an existing configuration file using a file requester.

Save
The Save (AMIGA-S) command stores the current settings to ENVARC:asx.prefs and ENV:asx.prefs.

Save as...
The Save as... (AMIGA-A) command is very similar to the Save command. It allows you to store the current settings to a user specified file. A file requester is opened that allows you to specify the name of the file the configuration is written to. The file requester will always appear with this command, whereas the Save command will assume the predefined names.

About...
The About selection will display the copyright and version notice for ASX, and some information about the used assembler.

Help...

This Help (AMIGA-?) command pops up a brief description of the main window. ASX uses the AmigaGuide for a context-sensitive on-line manual.

Hide

The Hide (AMIGA-H) menu item closes the ASX user interface, but leaves ASX installed in the background. This allows you to have ASX constantly available, while using the minimum amount of memory possible.

Quit

The Quit (AMIGA-Q) menu item tells ASX to exit. Usually, ASX can quit immediately. However, sometimes when you try to quit, another program has not ended the ARexx job or closed the asx.library. A requester will inform you by displaying the number of jobs that are still unfinished yet and ASX refuses to quit.

Window

SnapShot Window

This command tells ASX to remember the current window position. The window will appear in the new position the next time it is opened.

UnSnapShot Window

This command is the opposite of SnapShot Window. It flushes the previously stored window position.

Rastered Background?

To get a stronger 3D-look, ASX fills the background of all windows with a pattern. If you do not want to have this pattern as background, you can unselect this menu item and it will be written automatically to the settings file when you save settings. By default, this menu item is selected.

Settings

Assembler

The Assembler (AMIGA-E) menu item lets you switch over to the assembler settings window. See

The Assembler Window

.

```
    Incdir
            This IncDir (AMIGA-I) command changes to the
          IncDir window.  See
                The IncDir Window
                .



    Residents
            This Residents (AMIGA-R) command switches to the
          residents settings window.  See
                The Residents Window
                .



    Sources
            This  Sources (AMIGA-M) menu item switches to the
          source manager preference window.  See

                The Source Manager Preferences Window
                .
```

1.10 asx.guide/The_Assembler_Window

```
The Assembler Window
....................

    Here you can specify some global assembler settings, that are
automatically used when ASX is used to assemble a source code.

    All string gadgets in this window are connected to a Select gadget.
This gadget opens a file requester allowing the desired file to be
chosen.  The selected file name will instantly be written to the string
gadget.

The Gadgets
...........

ProAsm path
     Here you can specify the path of the ProAsm assembler.  By default
     ASX searches the assembler as pro in the following locations: asm:
     and the current directory.  Note that if you select a new
     assembler, you have to use Reload to force ASX to load it.  Reload
     does not have to be used if the Swap Assembler gadget is set.

Reload
     This gadget forces ASX to reload a new selected assembler.  This
     command must be performed if a new assembler is selected and the
     Swap Assembler is not set.

Swap Assembler
```

If this check-gadget is selected the assembler is loaded each time when used. By default, the assembler is once loaded residently. Loading the assembler residently can be useful on disk based systems. This tooltype is also recommended on system with little memory.

Header file
Using this string gadget you can define the header file that is automatically used for all assembly tasks started by ASX (via ARexx and asx.library).

Error file
Using this string gadget you can define the error file that is automatically used for all assembly tasks started by ASX (via ARexx and asx.library).

Config file
Using this string gadget you can define the configuration file that is automatically used for all assembly tasks started by ASX (via ARexx and asx.library).

Done
This action gadget toggles back to the main window.

The Menus
.....

The assembler window has only two menu items in the Assembler menu:
Help...
The Help (AMIGA-?) command pops up a brief description of the assembler window. ASX uses the AmigaGuide for a context-sensitive on-line manual.

Done
The Done (AMIGA-X) command toggles back to the main window.

1.11 asx.guide/The_IncDir_Window

The IncDir Window
.....

Through this interface, you have the possibility to define directories where the assembler (if run by ASX) searches for include files (see Include Files).

The Gadgets
.....

The listview gadget
This gadget shows you the list of all directories that are currently stored by ASX. Use the scroll bar and the arrow gadgets to position the scrolling area to a specific entry. By selecting an entry in the scroll area of the listview gadget its name is copied to the string-gadget.

The string gadget
     The string-gadget contains usually the name of the selected entry.
     A new entry can also be added to the list of directories by
     entering its name into the string-gadget and pressing RETURN.

Add
     This command allows you to add a directory to the list of
     directories using a file requester.

Delete
     The Delete command deletes the selected directory. A directory
     can be selected by entering its name into the string gadget, or by
     selecting the desired directory in the scroll area of the listview
     gadget.

Flush
     This command deletes all entries in the directory list.

Done
     This command quits the current window and switches back to the
     main window.

The Menus
....

    The incdir window has five menu items in the Edit menu:
Add
     The Add (AMIGA-A) command allows you to add a directory to the
     list of directories using a file requester.

Delete
     The Delete (AMIGA-D) command deletes the selected directory. A
     directory can be selected by entering its name into the string
     gadget, or by selecting the desired directory in the scroll area
     of the listview gadget.

Flush
     This command deletes all entries in the directory list.

Help...
     The Help (AMIGA-?) command pops up a brief description of the
     incdir window. ASX uses the AmigaGuide for a context-sensitive
     on-line manual.
     Note that ASX searches in the following locations for the asx.guide file:

          PROGDIR:
          PROGDIR:Help/<selected language>/   (e.g.: PROGDIR:Help/english/)
          Help:<selected language>/   (e.g.: Help/english/)
          <Current Directory>
          S:

Done
     The Done (AMIGA-X) command quits the current window and switches
     back to the main window.

1.12 asx.guide/The_Residents_Window

```
The Residents Window
....................

    Through this interface, you have the possibility to load include
files residently.  The assembler, run by ASX, can later access these
files without the need of loading them again.  Residently loaded
include files speed up your assembly time but reduce the memory.

The Gadgets
...........

The listview gadget
     This gadget shows you the list of all files that are currently
     kept in memory by ASX.  Use the scroll bar and the arrow gadgets
     to position the scrolling area to a specific entry.  By selecting
     an entry in the scroll area of the listview gadget its name is
     copied to the string-gadget.

The string gadget
     The string-gadget contains normally the name of the selected entry.
     A new entry can also be added to the list of resident files by
     entering its name into the string-gadget and pressing RETURN.

Add
     This command allows you to add a file to the list of residently
     loaded include files using a file requester.

Delete
     The Delete command deletes the selected file and frees allocated
     memory.  A file can be selected by entering its name into the
     string gadget, or by selecting the desired file in the scroll area
     of the listview gadget.

Flush
     This command deletes all entries in the file list, and frees all
     allocated memory.

Done
     This command quits the current window and switches back to the
     main window.

The Menus
.........

    The incdir window has five menu items in the Edit menu:
Add
     The Add (AMIGA-A) command allows you to add a file to the list of
     residently loaded include files using a file requester.

Delete
     The Delete (AMIGA-D) command deletes the selected file and frees
```

allocated memory. A file can be selected by entering its name into the string gadget, or by selecting the desired file in the scroll area of the listview gadget.

Flush
This command deletes all entries in the file list, and frees all allocated memory.

Help...
The Help (AMIGA-?) command pops up a brief description of the residents window. ASX uses the AmigaGuide for a context-sensitive on-line manual.

Done
The Done (AMIGA-X) command quits the current window and switches back to the main window.

1.13 asx.guide/The_Source_Manager_Preferences_Window

The Source Manager Preferences Window
.....

The source manager preferences window offers various settings to configure the source manager (see
The Source Manager Window
).

The Gadgets
.....

The Sources ListView-Gadget
This listview-gadget shows you a list of all entries that are currently stored by ASX. All these entries will also be shown by the source manager. Use the scroll bar and the arrow gadgets to position the scrolling area to a specific entry. By selecting an entry in the scroll area of the listview gadget its name is copied to the string-gadget directly below.

See
The User Interface
, for more information about a listview-gadget and how to handle it. New to this listview-gadget is, that it has a Sort gadget attached to it. This gadget can be used to sort all entries in alphabetical order.

An entry is a simple string that is later used by the ARexx script (see below).

The Public Screen Names ListView-Gadget
This listview-gadgets shows all currently opened public screens plus <front PubScreen> as first entry. Below the listview-gadget is a string-gadget that contains the name of the selected public screen. Use the scroll bar and the arrow gadgets to position the scrolling area to a specific entry. By selecting an entry in the

scroll area of the listview gadget its name is copied to the string-gadget directly below.

The source manager can be limited to be opened on only one public screen by selecting the desired screen. If <front PubScreen> is selected, the source manager will popup on the front public screen. Which, in the most cases, will be the screen of the program you currently work with.

fall back
When this check-gadget is selected, the source manager window does "fall back" to open its window on the default public screen if the explicitly specified public screen is not available. Otherwise the source manager does silently abort.

ARexx Script
These gadgets allow you to specify the ARexx script that is executed when an entry in the source manager is selected (see

The Source Manager Window

). This can be done by either entering the name and the complete path into the string gadget, or by selecting the Select gadget. The Select gadget opens a file requester with whom the desired script can be selected more easily.

When an entry is selected using the source manager (see

The Source Manager Window

), its string and the name of the public screen the window is opened on are passed to the ARexx script as arguments. The public screen name can be used to determine the program the source manager is used from. You can use the PARSE command of ARexx to put the arguments into ARexx variables. Consider the following example:

```
/* ARexx programs must start with a comment */
    OPTIONS RESULTS
    PARSE ARG source,pubscreen   /* get the arguments */
```

In the above example the source variable contains the selected string and pubscreen the public screen name. Look at the sample script file SourceManager.rexx for an example how the source manger can be used.

By default, rexx:pro/sourcemanager.rexx is used as script.

enabled
Using this check-gadget you can control the allowance of the source manager. If selected the source manager is enabled and its window will be opened when its hotkey is pressed. By default, the source manager is enabled.

Hot Key
This allows you to select what key sequence is used to open the source manager window, by entering the key description into the string gadget. Any standard commodities key description can be given. If you enter an incorrect description, a message will be displayed or a requester is opened to inform you. (The hotkey format is described in the ‘AmigaDOS 2.0 manual’). By default,

lcommand shift help is used as hotkey.

If you have defined a hotkey, the source manager can be called up at any time by pressing that key sequence.

Done
This action gadget toggles back to the main window.

The Menus
....

The source manager preferences window has the following five menu items:
Add source
The Add source (AMIGA-A) command allows you to add a file name to the list of source codes using a file requester.

Delete source
The Delete (AMIGA-D)command deletes the selected entry. An entry can be selected by entering its name into the string gadget, or by selecting the desired entry in the scroll area of the 'Sources' listview-gadget.

Flush source
This command deletes all entries in the 'Sources' list.

Help
The Help (AMIGA-?) command pops up a brief description of the source manager preferences window. ASX uses the AmigaGuide for a context-sensitive on-line manual.

Done
The Done (AMIGA-X) command toggles back to the main window.

1.14 pro.guide/The Source Manager Window

The Source Manager Window
....

The source manager provides a small project manager for a programming/development environment. Using the source manager you can handle your current projects in a very easy way. The interface between the manager and the application program builds an ARexx script. An ARexx script has the advantage that it allows also more than one application to be managed, even applications without an ARexx interface.

The source manager window is probably the simplest window. It pops up on the specified or on the front most public screen when you pressed the source manager hotkey defined in the source manager preferences window (see
The Source Manager Window
). Note that the window of the source manager can only be opened on public screens (see

The Source Manager Preferences Window
). The window presents itself
only with a listview-gadget, a window close-gadget, and a menu.

The Gadgets
....

The ListView-Gadget
The listview-gadget contains the same entries as defined in the source manager preferences (see
The Source Manager Window
). When
an entry is selected, by either a doubleclick or the RETURN key, the specified ARexx script is executed with the selected entry and the name of the public screen as arguments. For more details about this ARexx script and the argument passing, see

The Source Manager Window
.

The Window Close-Gadget
Select this gadget to cancel. It also closes the source manager window.

The Keyboard
....

The source manager window is also keyboard controllable:
up and down arrow keys
You may use this keys to select an entry. They have the same function as the small up and down arrow-gadgets attached to the listview-gadget.

RETURN
ENTER
One of these keys can be used to execute the ARexx script with the selected entry and the public screen name the window is opened on as arguments. In their function, they are identical to a double-click on the selected entry.

ESC
Press the ESC key to cancel. It also closes the source manager window.

Hotkey
Pressing the hotkey while the source manager window is open, the window is forced to the front, and it will be activated.

The Menu
....

The source manager window has a very short menu. The only menu item is CANCEL.

The CANCEL (AMIGA-C) command closes the source manager window and cancels the operation.

1.15 pro.guide/The AppIcon

```
The AppIcon
...........
```

    If the AppIcon gadget in the main window is selected, an AppIcon is displayed on the Workbench screen. If ASX was started from Workbench, the AppIcon has the same icon as ASX. Otherwise the built in icon is used.

    The operations that can be performed with the AppIcon are very easy:

Show Interface
     A double-click on the icon forces ASX to open the main window when it is hidden. It has the same effect as pressing the ASX's hotkey.

Assemble Files
     Just select one or more file icons (no drawers) on the Workbench, drag them to the ASX AppIcon and drop them. ASX will then assemble one file after the other. Note that no report window is opened to observe the assembly progress. Use the ERRFILE directive (see The Error File, for its description) to let the assembler generate error files that can be looked at later.

1.16 pro.guide/The ARexx Interface

```
The ARexx Interface
-------------------
```

    ARexx is an interpreted intercommunication language for the Amiga with the ability to send commands to and receive commands from other programs. ASX has an ARexx interface built into it that allows external programs to control ASX. The commands offered by ASX can be used to create an integrated programming/development environment with any ARexx equipped application.

    If you want to send commands from an ARexx equipped program to ASX, ARexx has to be informed. This is done using the ADDRESS command:

```
            ADDRESS 'asx_rexx'
```

    Be aware that case is important. ARexx will not find the port if the name is not spelled correctly.

```
An ARexx Example
................
```

    Here is a small example ARexx program that talks to ASX. In the

following example all plain ARexx commands are written in upper case, and the ASX ARexx commands in mixed case: For a description of the plain ARexx commands refer to the 'AmigaDOS manual'.

```
/* ARexx programs must start with a comment */

OPTIONS RESULTS
PARSE ARG SourceFile,OutputFile     /* get arguments */
ADDRESS 'asx_rexx'                  /* talk to ASX */

BeginAssembly                       /* start an ASX Daemon Task */
DaemonPort = RESULT

IF DaemonPort~='' THEN DO
        ADDRESS (''||DaemonPort)    /* now talk to the daemon task */
        DefineID 'foo'              /* optionally define an ID */
        Assemble SourceFile -o OutputFile -w /* Assemble... */
END

EXIT(0)
```

For more sophisticated examples see the ARexx scripts that come along with ASX.

1.17 pro.guide/ARexx Commands

ARexx Commands
....

This section lists all ARexx commands of ASX in alphabetical order, with their syntax, arguments, and result descriptions.

The ARexx commands of ASX have the following general format. The name of the command comes first, followed by one or more arguments if required. Arguments in square brackets are optional.

All ASX commands when called from ARexx respond by setting two variables: the RC (the error status code) variable and the RESULT variable. RC contains an integer value that indicates the severity of an error:

0	No error occurred.
5	A warning only.
10	Something went wrong.
20	Complete or severe failure occurred.

RESULT contains the result of the command if a result is returned. RESULT can be a string or an integer value according to the called command. Before using a command that returns a result, it is important to turn on results with the option results command. After this command

has been executed, any results returned by ASX or the daemon task will be in the RESULT variable.

Following you find the description of all ARexx commands that have to be send to the ASX ARexx interface (asx_rexx).

ARexx Commands Overview
.....

- : AddIncDir path
  This command adds the specified path to the list of pathes stored in ASX. A path name containing one or more whitespaces must be enclosed in double or single quotes.

  RESULT
  True (1) if succeeded, false (0) otherwise.

  RC
  Error status code.

  See also RemIncDir and FlushIncDir.

- : AddResident file
  This command loads the specified file to ASX's resident list. A file name containing one or more whitespace must be enclosed in double or single quotes.

  RESULT
  None.

  RC
  Error status code.

  See also RemResident and FlushResident.

- : AddSource string
  This command adds the name of the specified string to the source name list of ASX's source manager. A string name containing one or more whitespace must be enclosed in double or single quotes.

  Note that string can be a string of any format.

  RESULT
  None.

  RC
  Error status code.

  See also RemSource, FlushSource.

- : BegOfAssembly
  This command starts a new daemon process that awaits assembly jobs. BegOfAssembly must be used before any source code can be assembled.

The following example shows the use of the BegOfAssembly command and the addressing of the new created task:

```
ADDRESS 'asx_rexx'
BegOfAssembly
daemon = RESULT
...
ADDRESS (''||daemon)    /* ARexx weirdness */
...
EndOfAssembly
```

Note the strange argument of the second ADDRESS command. To address the new created daemon task, the result of the BegOfAssembly command must be used as argument of the ADDRESS command. Either as ADDRESS VALUE port, or due to a ARexx weirdness, the argument can have the following format: ("||port).

RESULT
ARexx port name of the new created daemon task.

RC
Error status code.

See also EndOfAssembly.

- : DisableFallBack
This command forbids the "fall back" mode for the source manager window. If "fall back" is disabled, the source manager silently aborts if its window cannot be opened on the explicitly specified public screen.

RESULT
None.

RC
Error status code.

See also EnableFallBack.

- : DisableManager
This command can be used to disable the source manager via ARexx.

RESULT
None.

RC
Error status code.

See also EnableManager.

- : EnableFallBack
This command enables the "fall back" mode for the source manager window. If "fall back" is enabled, the source manager window opens its window on the default public screen if the explicitly specified public screen is not available.

RESULT
None.

RC
Error status code.

See also DisableFallBack

- : EnableManager
This is the opposite of the DisableManager command and it can be used to enable the source manager via ARexx.

RESULT
None.

RC
Error status code.

See also DisableManager.

- : FindID idstring
This command searches the daemon process spawned by ASX which carries idstring as its ID. An ID can be specified for a daemon process using the DefineID command.

RESULT
None.

RC
Error status code.

See also DefineID and GetID.

- : FlushIncDir
This command deletes all entries in the include directory list stored in ASX.

RESULT
None.

RC
Error status code.

See also AddIncDir and RemIncDir.

- : FlushResident
This command deletes all residently loaded include files in the ASX's resident list.

RESULT
None.

RC
Error status code.

See also AddResident and RemResident.

- : FlushSource
This command deletes all entries in the source name list of ASX's source manager.

RESULT
None.

RC
Error status code.

See also AddSource and RemSource.

- : GetPort num
This command returns the name of port number num. This command seems to be a bit strange, since you do not know the port sequence. The GetPort command is merely implemented to have a possibility to walk through the port list and to access all ports, even the unknown ones.

RESULT
Name of the port or an empty string, if the port could not be found.

RC
Error status code.

- : Hide
The Hide command closes the ASX user interface if opened, but leaves ASX installed in the background. This allows you to have ASX constantly available, while using the minimum amount of memory possible.

RESULT
None.

RC
Error status code.

- : IsIncDir path
The IsIncDir command searches the given path in the include directory list stored in ASX.

RESULT
A true (1) value is returned if path could be found in the list. Otherwise a false (0) is returned.

RC
Error status code.

- : IsResident file
This command searches for the given file in the list of residently loaded include files.

RESULT
True (1) is returned when the search was successful. Otherwise false (0) is returned.

RC
Error status code.

- : IsSource string
The IsSource command searches for the given string in the source name list of ASX's source manager.

RESULT
True (1) is returned when the search was successful. Otherwise false (0) is returned.

RC
Error status code.

- : IsThereError
The IsThereError returns the port name of the next daemon process with a non-empty error list.

RESULT
Daemon's port name or an empty string if no daemon process could be found.

RC
Error status code.

- : IsTherePort
This command returns the port name of the next demon process. This command can be used to traverse through the complete daemon port list.

RESULT
Daemon's port name or an empty string if no daemon process could be found.

RC
Error status code.

- : Quit
This command tells ASX to exit. Usually, ASX can quit immediately. However, sometimes when you try to quit, another program has not ended the ARexx job or closed the asx.library yet. In such a case, ASX refuses to quit.

RESULT
None.

RC
Error status code.

- : RemIncDir path
This command removes the specified path from the list of pathes stored in ASX. A path name containing one or more whitespace must be enclosed in double or single quotes. If the given path is not in the list, the operation will be ignored silently.

RESULT
None.

RC
Error status code.

See also AddIncDir and FlushIncDir.

- : RemResident file
This command removes the specified file from the resident list. A file name containing one or more whitespace must be enclosed in double or single quotes. If the given file is not in the list, the operation will be ignored silently.

RESULT
None.

RC
Error status code.

See also AddResident and FlushResident.

- : RemSource string
This command removes the specified string from the source name list of ASX's source manager. A string name containing one or more whitespace must be enclosed in double or single quotes. If the given string is not in the list, the operation will be ignored silently.

Note that string can be a string of any format.

RESULT
None.

RC
Error status code.

See also AddSource and FlushSource.

- : SetARexxScript filename
This command specifies the ARexx script that is executed when an entry in the source manager is selected. Filename is the name of the ARexx script file. The filename must be enclosed in double or single quotes if it contains any whitespaces.

RESULT
None.

RC
Error status code.

- : SetPubScreen ScreenName
This command can be used to specify the name of the public screen on which the source manager should open its window.

RESULT
None.

RC
Error status code.

See also EnableFallBack.

```
- : Status number
    This command allows you to find out the values of the 21 internal
    variables of ASX.

    Some of the variables whose values are returned hold a true (1) or
    false value (0).  Those that return true or false values have a
    question mark (?) after them in the table below.  Other variables
    return numbers or strings.

    The given number argument specifies the number of the internal
    variable you want information on.  Valid numbers currently range
    from 0 to 21.  Following you find a list of all the internal
    variables and their numbers:
   0
         This returns the version string of ASX.

   1
         The number of jobs running.

   2
         The number of resident files loaded.

   3
         The number of include pathes stored.

   4
         The number of error lists stored by ASX that are not looked
         at.

   5
         ASX user interface hidden?

   6
         This returns the version string of ProAsm.

   7
         asx.library installed?

   8
         This returns the open count value of the asx.library.

   9
         AppIcon displayed?

   10
         The number of source file names in the source list of the
         source manager.

   11
         This returns the name of the specified pubscreen for the
         source manager.

   12
         FallBack enabled?

   13
```

Source manager enabled?

14
This returns the name of the ARexx script file that is executed when a source is selected in the source manager.

15
The string of the current ASX hotkey definition.

16
The string of the current source manager hotkey definition.

17
This returns the path name of ProAsm.

18
This returns the name of the specified header file.

19
This returns the name of the specified configuration file.

20
This returns the name of the specified error file.

21
Swap assembler?

The following two variables reflect the result of the Status command:

RESULT
Contents of the desired internal variable.

RC
Error status code.

ARexx Commands Overview For The Daemon Process
.....

Following you find all ARexx commands that have to be send to the daemon process that is spawned by the BegOfAssembly command (see also BegOfAssembly). Such a demon process must be started, before any source code can be assembled. Consider the following example that starts such a daemon process:

```
ADDRESS 'asx_rexx'
BegOfAssembly
daemon=port
...
ADDRESS (''||daemon)    /* ARexx weirdness */
...
EndOfAssembly
```

\- : Assemble arguments
This command executes the ProAsm assembler. Arguments can be any of the command line arguments supported by ProAsm. For the

argument syntax the same syntax rules are valid as for the command line.

A few texteditors have the possibility to return the start address of the source code in memory. In this case you may use the -A/ADDRESS command line argument of ProAsm. For example:

```
Assemble ADDRESS the_start_address
asmresult = RESULT
```

Using another editor, that does not support this feature, you can cause the editor to save a temporary file that is assembled. The temporary file can then be deleted after the assembly is done. For example:

```
Assemble the_temp_file
asmresult = RESULT
ADDRESS COMMAND "c:delete "||the_temp_file
```

RESULT

The RESULT variable contains the message string returned by the assembler. The strings informs you about the number of errors (E), warnings (W), and optimizations encountered (O). It informs you likewise about the number of bytes saved by optimizations (the second value succeeding the (O) in the format string), the code size (C), and the workspace used (S). In general, it is the same message that is displayed in the shell after the assembler has finished its job. The returned string is of the following format:

```
E num W num C num O num num S num
```

You can use the PARSE command of ARexx to split the message in various parts and put the different values into ARexx variables:

```
Assemble foo
PARSE VAR RESULT . err . warn . opt obytes . size . spc
/* Where:
    err:    #of errors
    warn:   #of warning messages
    opt:    #of optimizations encountered
    obytes: #of bytes saved by optimizations
    size:   code size in bytes
    spc:    used workspace (in bytes)
*/
```

The values are stored in the different variables in the same order as described above. Refer to the 'AmigaDOS manual' for the description of the ARexx commands.

RC

Error status code.

\- : DefineID

This command allows you to define an ID string to mark your assembly job. Usually this is used to identify a programming/development environment.

For example:

```
ADDRESS 'asx_rexx'
FindID 'TextEditor'
port = result
```

```
DO WHILE port~=''
  ADDRESS (''||port)
  EndOfAssembly         /* quit all open jobs */
  ADDRESS 'asx_rexx'
  FindID 'TextEditor'
  port = result
END

BegOfAssembly           /* begin a new job    */
port = result

IF port~='' THEN DO
  ADDRESS (''||port)    /* ARexx weirdness    */
  DefineID 'TextEditor' /* set ID             */
  ...
END
```

The ARexx fragment above shows you a possible use of the DefineID and FindID commands. The examples assume a programming/development environment that allows only one ARexx script, one job respectively, to be executed at the time. The example can easily be extended to do the same for more than one source code.

RESULT
None.

RC
Error status code.

See also FindID and GetID.

- : EndOfAssembly
This command tells ASX that the current assembly job is finished. The daemon process is then quitted, and all the memory and resources used is returned to the system.

The EndOfAssembly command should be used if the job has finished and the error list managed by ASX is not used anymore. In a programming/development environment this may have to be managed differently. You can define the same ID for all assembly jobs started by this environment (using the DefineID command), and then instead of quitting each job separately, you can write a function that quits all jobs at once. This has the advantage that the error lists stored by ASX are still available until all jobs are ended.

RESULT
None.

RC
Error status code.

See also BefOfAssembly.

- : FirstError
This command returns the first error or warning message reported by ProAsm to ASX.

```
    RESULT
          Error or warning message.  If neither an error nor a warning
          message is reported, an empty string is returned.

    RC
          Error status code.

     See also NextError and PrevError.

 - : GetID
     The GetID command returns the ID string of the current assembly
     job.

    RESULT
          ID string.

    RC
          Error status code.

     See also DefineID and FindID.

 - : NextError
     This command returns the next error or warning message in the
     error list reported by ProAsm.  NextError begins with the first
     reported message, except the FirstError command was executed
     before.

    RESULT
          Error or warning message.  An empty string is returned, if
          the end of the message list has been reached.

    RC
          Error status code.

     See also FirstError and PrevError.

 - : PrevError
     This command returns the previous error or warning message in the
     error list reported by ProAsm.

    RESULT
          Error or warning message.  An empty string is returned, if
          the start of the message list has been reached.

    RC
          Error status code.

     See also FirstError and NextError.
```

1.18 pro.guide/The asx.library

```
The asx.library
---------------
```

With the asx.library you have access on the basic capabilities of ASX: assemble, manage the residently loaded files, include pathes, and the sources from the source manager.

This library is added to the system only when the asx.library gadget in the main window of ASX is selected. This can be done by using the graphical user interface, or , at startup, by suppressing the NOLIBRARY command line argument, tooltype respectively (the library is installed by default).

Using the library function, you can build your own user interface for ProAsm. These library functions are quite similar to their ARexx equivalent.

In the following you find an alphabetical list of all asx.library functions:

```
 - : asx_Assemble()
                  result := asx_Assemble(ArgStr, SpecialList)

                   d0                     a0,        a1
```

The asx_Assemble() function executes the ProAsm assembler, with the ArgStr and the SpecialList as Arguments.

Input:

a0

ArgStr - is a pointer to the command line arguments. Any command line arguments supported by ProAsm will be accepted. For Example:

```
                lea     ArgStr(pc),a0
                ...
     ArgStr: DC.B    "-W -F",0
                EVEN
```

a1

SpecialList - refer to asx.i for further information about this structure. To work properly, only two fields must be set: the ax_Source and ax_StdOut field.

Result:

d0

This register contains a boolean value whether the function succeeded (true) or not (false). The result of the assembly, if succeeded, is returned in the ax_Errors, ax_Warnings, ... fields of the SpecialList.

Take also a look at the Pro68.s source code as an example for the use of this function.

```
 - : asx_AddIncDir()
                        result := asx_AddIncDir(path)

                         d0                    a0
```

This function adds the specified path to the list of pathes stored in ASX.

Input:
a0
path - pointer to a pathname that should be stored by ASX. ASX makes a copy of the supplied pathname.

Result:
d0
A boolean value is returned: true if succeeded, false (0) otherwise.

- : asx_AddResident()

```
result := asx_AddResident(filename)

 d0                          a0
```

The asx_AddResident() function loads the specified file to ASX's resident list.

Input:
a0
filename - pointer to the name of the file that should be loaded residently by ASX.

Result:
d0
A boolean value is returned: true if succeeded, false (0) otherwise.

- : asx_AddSource()

```
result := asx_AddSource(string)

 d0                      a0
```

This function adds the name of the specified string to the source name list of ASX's source manager.

Note that string can be a string of any format.

Input:
a0
filename - pointer to a filename or a string that should be copied to ASX source manager's source list.

Result:
d0
A boolean value is returned: true if succeeded, false (0) otherwise.

- : asx_FlushIncDir()

```
asx_FlushIncDir()
```

This function deletes all entries in the include directory list stored in ASX.

```
- : asx_FlushResident()
                              asx_FlushResident()
```

The asx_FlushResident() function deletes all residently loaded include files in the ASX's resident list.

```
- : asx_FlushSource()
                               asx_FlushSource()
```

This function deletes all entries in the source name list of ASX's source manager.

```
- : asx_IsIncDir()
                         result := asx_IsIncDir(path)

                          d0                    a0
```

The asx_IsIncDir function searches for the given path in the include directory list stored in ASX.

Input:
a0
path - pointer to the pathname to search for.

Result:
d0
A boolean value is returned: true if path could be found, false (0) otherwise.

```
- : asx_IsResident()
                      result := asx_IsResident(filename)

                       d0                        a0
```

This function searches for the given file in the list of residently loaded include files.

Input:
a0
filename - pointer to the filename to search for.

Result:
d0
A boolean value is returned: true if the file could be found, false (0) otherwise.

```
- : asx_IsSource()
                        result := asx_IsSource(string)
```

```
                          d0                    a0
```

The IsSource function searches for the given string in the source name list of ASX's source manager.

Input:
a0
string - pointer to the string to search for.

Result:
d0
A boolean value is returned: true if the string could be found, false (0) otherwise.

- : asx_RemIncDir()

```
                              asx_RemIncDir(path)

                                      a0
```

This function removes the specified path from the list of pathes stored in ASX. If the given path is not in the list, the operation will be ignored silently.

Input:
a0
path - pointer to the pathname that should be removed from the include directory list of ASX.

Result:
None.

- : asx_RemResident()

```
                           asx_RemResident(filename)

                                      a0
```

This function removes the specified file from the resident list. If the given file is not in the list, the operation will be ignored silently.

Input:
a0
filename - pointer to a filename that should be removed from the resident list.

Result:
None.

- : asx_RemSource()

```
                             asx_RemSource(string)

                                      a0
```

```
     This function removes the specified string from the source name
     list of ASX's source manager.  If the given string is not in the
     list, the operation will be ignored silently.

    Input:
         a0
               string - pointer to the string that should be removed
               from the source list.

    Result:
         None.
```

1.19 author

```
Author
======

If you have bugreports, questions, ideas, flames or complaints (constructive
criticism is always welcome), or if you just want to contact me, write
or send a letter to:
```


```
                        Daniel Weber

        Internet:       dweber@amiga.icu.net.ch   (preferred)
                        dweber@iiic.ethz.ch

        Mail:           Daniel Weber
                        Hoeflistrasse 32
                        CH-8135 Langnau
                        Switzerland.
```