

# MIX – Micronas Interconnect Spec expander

## Introduction

## Management Summary

## Overview

### ***Table of contents:***

MIX – Micronas Interconnect Spec expander.....	1
Introduction.....	1
Management Summary .....	1
Overview .....	1
Table of contents: .....	1
Installation .....	2
Getting Started .....	2
A simple example .....	2
Mix it! .....	3
You get what you typed.....	5
Details .....	7
Common excel worksheet properties.....	7
HIER sheet properties .....	7
HIER columns details .....	8
Special HIER sheet properties.....	8
CONN sheet details .....	9
CONN columns details .....	9
Special CONN sheet signals.....	9
%OPEN% aka. open .....	9
Constants .....	9
Generics and Parameters .....	11
Predefined and user macros.....	11
CONN sheet macros .....	13
Generator statements.....	13
IO sheet.....	13
IO sheet column header.....	14
IO sheet %SEL% rows .....	15
IO sheet pad rows.....	15
IO cell and Pad connections .....	15
IO Parser global configuration .....	16
I2C sheet.....	16
VI2C sheet.....	17
Alarm clock example .....	17
MIX converter man page .....	17
Synopsis .....	17
Command line switches.....	17
Runtime options and configuration .....	18
Misc features .....	18
-delta mode .....	18

Intermediate Excel Sheet .....	18
Output Redirection .....	19
Alarm clock example .....	19
Core logic.....	19
IO logic.....	19
Other examples .....	19
Known Bugs and limitations .....	19
Issue tracking: CADNET -> Issue tracking -> CAD Software -> MIX.....	19
Links.....	19
MIX paper and documentation.....	19
Micronas internal .....	20
Others.....	20
Resources.....	20
Used Software .....	20
Release Notes: .....	20
20030716 .....	20
20030709 .....	20
20030605 .....	20

## Installation

The MIX toolset uses the [Perl](#) scripting language. Please install a recent version of Perl on your workstation, e.g. the freely available [ActiveState Perl](#) from K:\PROJECTS\MIX\PROG\ActivePerl-5.6.1.633-MSWin32-x86.msi.

Apart from that no installation is required.

Start MIX from the network drive like described in the examples below. That will also make sure, that you are using the most recent release automatically. After the Perl installation, you can run MIX. immediately. Open a command shell on your desktop workstation

```
Start -> Ausführen -> cmd
```

and type

```
K:\Projects\MIX\PROG\mix_0.pl foo.xls
```

*Caveat: Currently MIX is available on MS-Windows, only!*

## Getting Started

To receive useful results, MIX reads in various input spreadsheet data. At least you will need to prepare a description of your design hierarchy (HIER) and the connectivity sheet (CONN). Optionally MIX will convert a input/output sheet IO, listing input/output pads and iocells and how these are linked to the core logic into appropriate hierarchy and connection lists.

### ***A simple example***

To understand the usage of the various tables and options, a very simple example is shown and extended step by step. The simple example just has two components ***a*** and ***b***, which are connected by the signal ***sigfoo***. The two instances have a common parent ***chip***

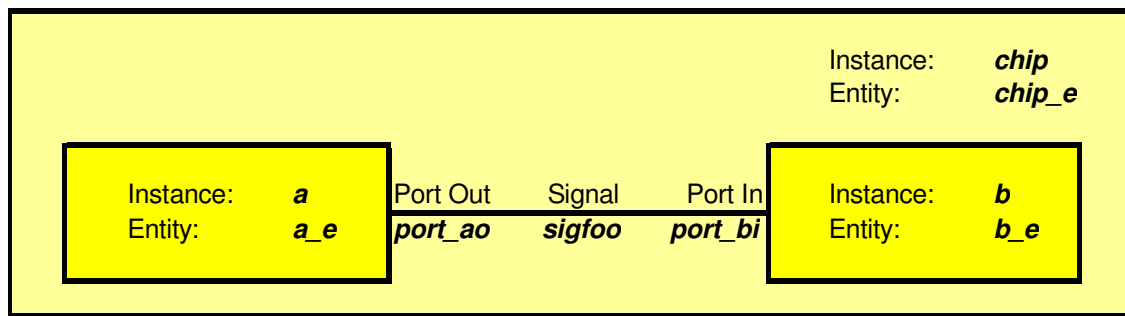


Image 1: Simple mix\_simple.xls

The equivalent description of this simple design in MIX is made up from a worksheet **HIER**.

:::ign	:::gen	:::variants	:::parent	:::inst	:::entity	:::config	:::comment
# Mix SimpleExample, V0.1, 20030630, hierachy							
		Default	TESTBENCH	chip	chip_e	chip_e rtl_conf	
		Default	chip	a	a_e	a_e rtl_conf	
		Default	chip	b	b_e	b_e rtl_conf	

The first row defines the table headers names. The names have to be in the form “:::NAME”. Several of the columns are required, some are optional and you can define additional columns on your own. For HIER sheets the “:::inst” column is the primary key. One design element will be generated for each new name of an “:::inst” row. If a name is defined several times, these lines will be overloaded or summarized, depending on built-in rules of the MIX converter.

The “:::ign” column has to come first. If it starts with a #, the rest of this row will be ignored. All other columns can be added in arbitrary order.

The second required worksheet is **CONN**:

:::ign	:::gen	:::bundle	:::class	:::clock	:::type	:::high	:::low	:::mode	:::name	:::out	:::in	:::descr	:::comment
# Mix SimpleExample, V0.1, 20030630, hierachy													
		Chip	Data	Clk	std_ulogic			S	sigfoo	a/port_ao	b/port_bi	Simple	

As you see, this worksheet also starts with the table header definition line. The primary field is the “:::name” column. The “:::in” and “:::out” columns are used to define the drivers and loads for the signals.

## Mix it!

Run the MIX converter tool in the directory the excel spread sheet is stored in:

```
$ cd \work\MIX\doc\parts\simple
```

```
$ K:\Projects\MIX\PROG\mix_0.pl mix_simple.xls
```

This reads in the design description and evaluates the various sheets. It creates output files with an intermediate excel design description (mix\_simple-mixed.xls) and the same data in a internal format (mix\_simple.pld). A log file (mix\_0.pl.log) and the HDL output files are written in the same run. Image 2 shows a screenshot of the mix\_simpe.xls conversion. Only the most important errors and warnings are written to the

screen, while a lot of information will be written to the log file. Search for the keywords “ERROR” and “WARNING” to verify proper conversion.

```

C:\WINNT\System32\cmd.exe

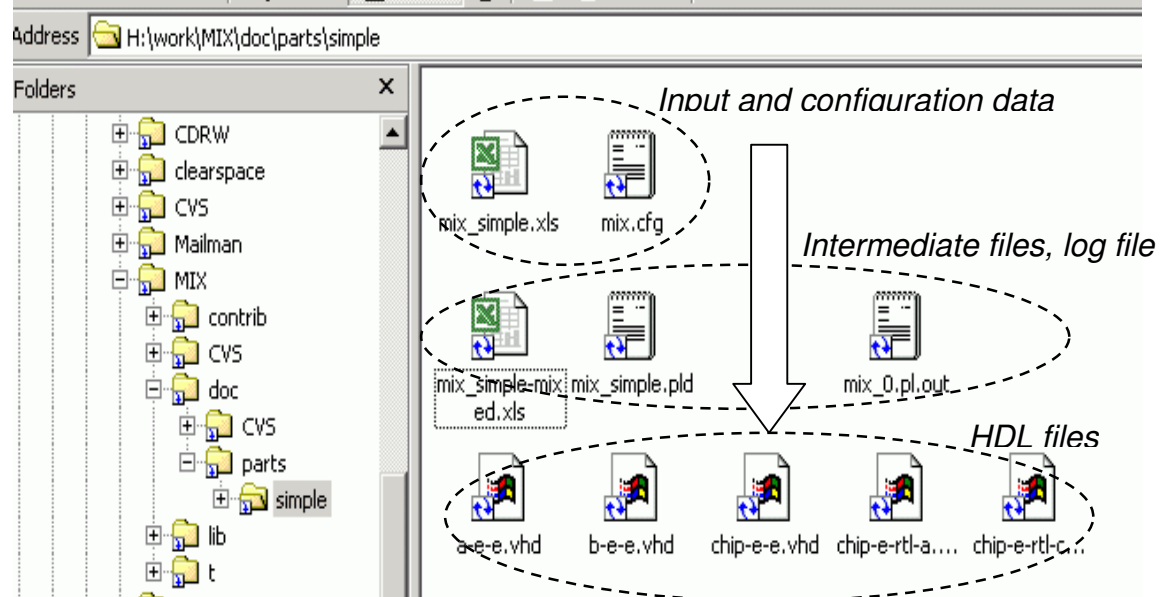
H:\work\MIX\doc\parts\simple>K:\Projects\MIX\PROG\mix_0.pl mix_simple.xls
#####
##### mix_0.pl <Revision Revision: 1.12 >
#####
##### MIX
#####
03/06/30 14:28:15 1336: INFO: Not all lowercase in new inst element TESTBENCH!
03/06/30 14:28:15 1336: file mix_simple.pld already exists! Will be overwritten
03/06/30 14:28:15 1336: File mix_simple-mixed.xls already exists! Contents will
be changed
03/06/30 14:28:15 1336: Rotating 3 old sheets of CONN!
03/06/30 14:28:15 1336: File mix_simple-mixed.xls already exists! Contents will
be changed
03/06/30 14:28:15 1336: Rotating 3 old sheets of HIER!
03/06/30 14:28:16 1336: ===== SUMMARY =====
03/06/30 14:28:16 1336: SUM: Summary of checks and created items:
03/06/30 14:28:16 1336: SUM: checkforce 0
03/06/30 14:28:16 1336: SUM: checkunique 0
03/06/30 14:28:16 1336: SUM: checkwarn 0
03/06/30 14:28:16 1336: SUM: conn 1
03/06/30 14:28:16 1336: SUM: errors 0
03/06/30 14:28:16 1336: SUM: inst 8
03/06/30 14:28:16 1336: SUM: warnings 0
03/06/30 14:28:16 1336: SUM: Number of parsed input tables:
03/06/30 14:28:16 1336: SUM: conf 0
03/06/30 14:28:16 1336: SUM: hier 1
03/06/30 14:28:16 1336: SUM: conn 1
03/06/30 14:28:16 1336: SUM: io 0
03/06/30 14:28:16 1336: SUM: vi2c 0

H:\work\MIX\doc\parts\simple>

```

Image 2: Screenshot mix\_simple conversion

All output files are stored in the current working directory. Old versions of the output files are overwritten. Except the log file that is appended by each converter run. The intermediate excel description file keeps a history of previous HIER and CONN sheets by rotating \_N extended worksheet names.



## You get what you typed

MIX generates various HDL files defined by the input data. If you select VHDL (also the default language) as output description for a hierarchy element, each element results in an appropriate entity, architecture and configuration description. By default MIX writes one file for all entities, one for all architecture and one for all configuration descriptions. Those file names are derived from the last excel input file name by stripping of the .xls extension and attaching a -e.vhd, -a.vhd and -c.vhd.

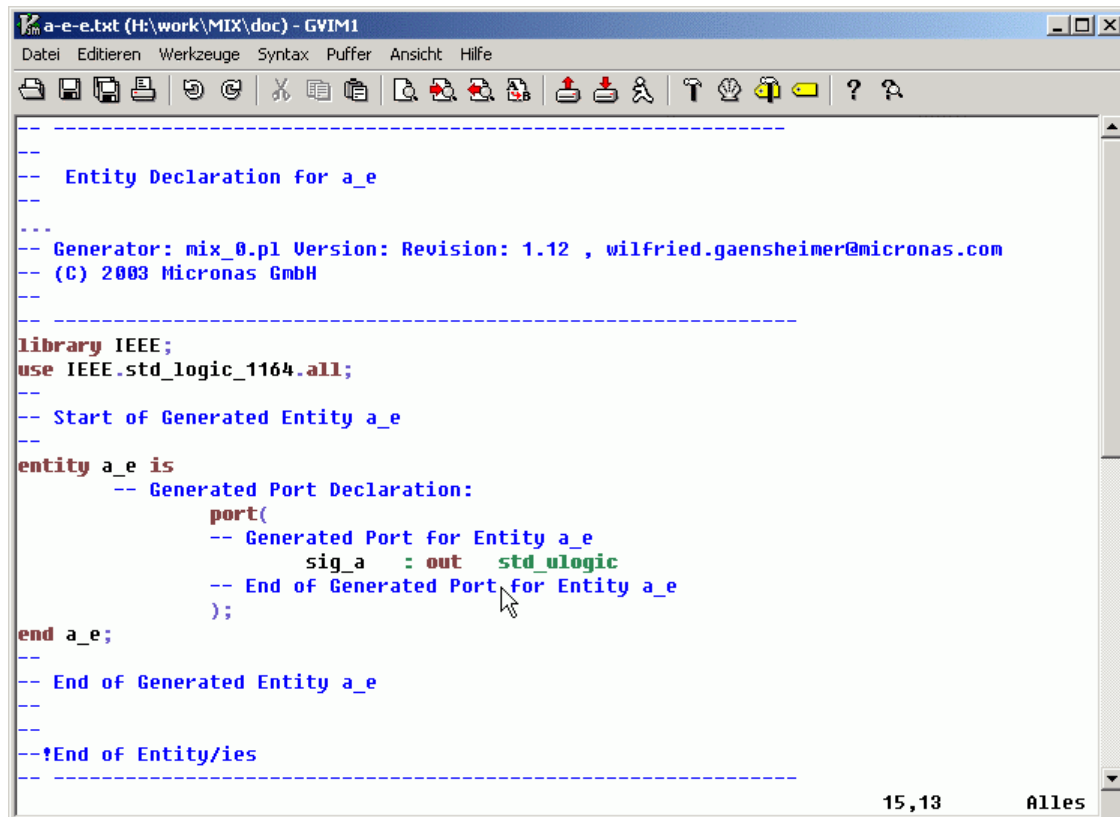
Here the working directory of the simple example contains a file mix.cfg, which is the convenient storage for MIX run-time configuration options. The lines "MIXCFG outarch ARCH", "MIXCFG outenty ENTY", and "MIXCFG outconf CONF" switch MIX outputs to separate files for each entity, architecture and configuration. In this case the file names are defined by the element name:

Type	Basename	Extension	Example
Entity	::entity-column	-e.vhd	<i>chip_e-e.vhd</i>
Architecture	::entity-column	-rtl-a.vhd	<i>chip_e-rtl-a.vhd</i>
Configuration	::conf-column	-c.vhd	<i>chip_e-rtl-conf-c.vhd</i>

"\_" in the element names are converted to -.

By default MIX does not write output data for leaf blocks (instances which are not parent for other instances). Adding a line like "MIXCFG generate.output.arch leaf" into the mix.cfg file changes that.

The generated output files contain head, body and footer sections. See the screenshots of the file a-e-e.vhd and chip-e-rtl-a.vhd for examples of an entity and an architecture definition.

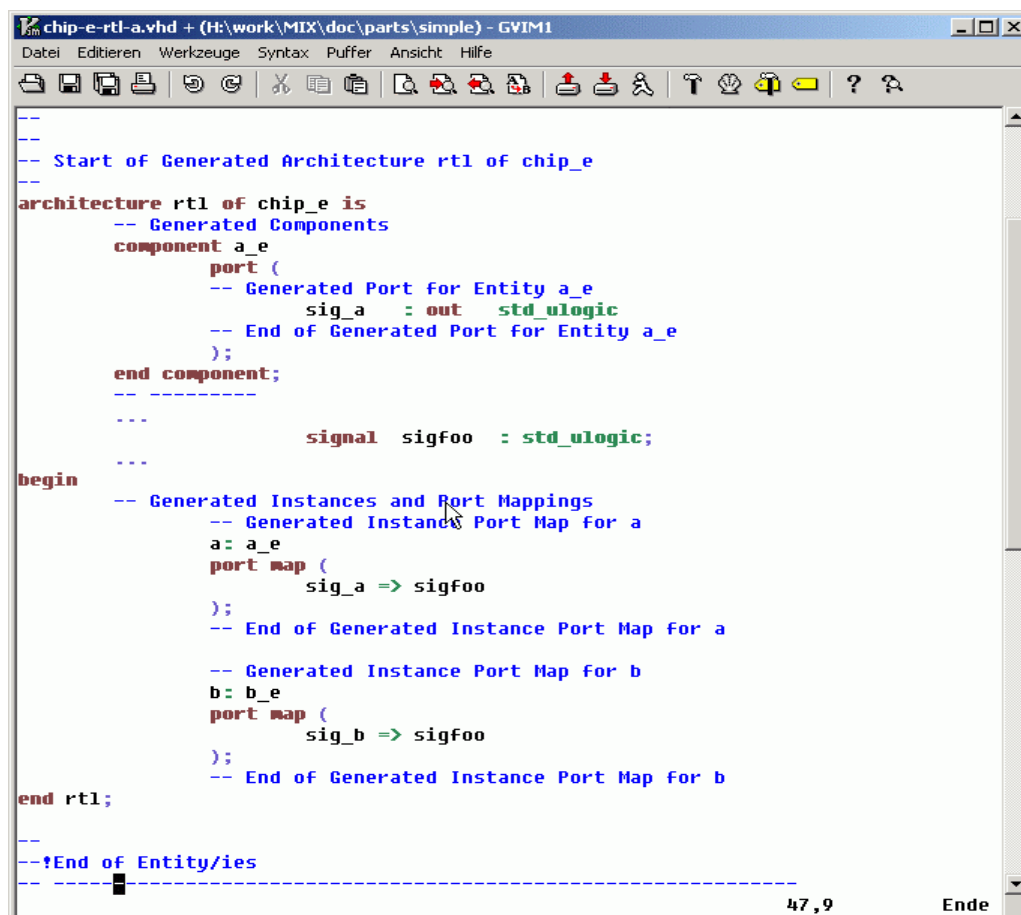


The screenshot shows a text editor window titled "a-e-e.txt (H:\work\MIX\doc) - GVIM1". The menu bar includes "Datei", "Editieren", "Werkzeuge", "Syntax", "Puffer", "Ansicht", and "Hilfe". The toolbar contains various icons for file operations and editing. The main text area displays the following VHDL code:

```
-----  
-- Entity Declaration for a_e  
--  
-- Generator: mix_0.pl Version: Revision: 1.12 , wilfried.gaensheimer@micronas.com  
-- (C) 2003 Micronas GmbH  
--  
-----  
library IEEE;  
use IEEE.std_logic_1164.all;  
--  
-- Start of Generated Entity a_e  
--  
entity a_e is  
    -- Generated Port Declaration:  
    port(  
        -- Generated Port for Entity a_e  
        sig_a : out std_ulogic  
        -- End of Generated Port for Entity a_e  
    );  
end a_e;  
--  
-- End of Generated Entity a_e  
--  
--!End of Entity/ies  
-----
```

The status bar at the bottom right shows "15,13" and "Alles".

Image 3: Entity a-e-e.vhd



The screenshot shows a text editor window titled "chip-e-rtl-a.vhd + (H:\work\MIX\doc\parts\simple) - GVIM1". The menu bar and toolbar are the same as in Image 3. The main text area displays the following VHDL code:

```
--  
-- Start of Generated Architecture rtl of chip_e  
--  
architecture rtl of chip_e is  
    -- Generated Components  
    component a_e  
        port (  
            -- Generated Port for Entity a_e  
            sig_a : out std_ulogic  
            -- End of Generated Port for Entity a_e  
        );  
    end component;  
    --  
    signal sigfoo : std_ulogic;  
    --  
begin  
    -- Generated Instances and Port Mappings  
    -- Generated Instance Port Map for a  
    a: a_e  
    port map (  
        sig_a => sigfoo  
    );  
    -- End of Generated Instance Port Map for a  
    -- Generated Instance Port Map for b  
    b: b_e  
    port map (  
        sig_b => sigfoo  
    );  
    -- End of Generated Instance Port Map for b  
end rtl;  
--  
--!End of Entity/ies  
-----
```

The status bar at the bottom right shows "47,9" and "Ende".

Image 4: architecture chip-e-rtl-a.vhd

## Details

### Common excel worksheet properties

All excel worksheets parsed by MIX share some common properties.

There needs to be a header line consisting only of keywords with leading double colon. All data before the header line is ignored. Only the first header line will be evaluated. Data in columns with no header or malformed headers will be ignored.

Commonly understood table headings are `::ign` and `::comment`. The `::ign` column is special, because it needs to be the first column of a sheet. If a cell in the `::ign` column starts with a #, the complete row is ignored.

The `::comment` column can contain user or program generated comments for a given row. It's data will be appended to it's contents as it appears.

MIX reads the excel cell values. This is what you see, not the real contents of the cells. Thus all excel formulas can be used to define the cell values.

A lot of predefined *text macros* are understood and converted by MIX. A text macro is made up by a name surrounded by % signs. Retrieve a complete list of macros with the `-listconf` command line switch and grep all lines starting with "MIXCFG macro". The table %macro% gives a list of predefined macros, their default settings and whether this macro can be used by the user. You can use text macros inside of any cell.

::ign	::gen	::variants	::parent	::inst	::lang	::entity	::config	::comment
#Mx SimpleExample, Macro Expansion, V0.1, 20030630, hierachy								
		Default	TESTBENCH	chip	VHDL	%:inst% e	%:inst% e rtl conf	
		Default	chip	a	VHDL	%:inst% e	%:inst% e rtl conf	
		Default	chip	b	%LANGUAGE%	%:inst% e	%:inst% e rtl conf	

A second type of macros are available to reference other columns of a like through `::NAME%`. This will be replaced by the contents of the `::NAME` column of this row. `::NAME` has to be defined for the current worksheet, obviously. In the above example `::inst%_e` will evaluate to `chip_e`, `a_e` and `b_e` accordingly. `%LANGUAGE%` becomes `vhdl`. Macro expansion happens just before the intermediate design data is written out, after evaluation of all input data. Recursive macro expansion is not implemented. Macros in primary keys like signal and instance names are evaluated at signal or instance creation time.

### HIER sheet properties

The hierarchy of a design is defined in the HIER sheet. By default the HIER sheet is named HIER, but you can set the configuration value `hier.xls` to a Perl regular expression. MIX will then consider all worksheets which names match the perl regular expression, to be HIER definitions. In that case you are free to use different header definitions on different sheets.

The HIER sheets require at least the columns marked man. (mandatory) in the following table.

Column name	Description	Default value	Req.	Example
::ign	Ignore line	<empty>	man.	# comm.
::gen	Generator and match	<empty>	man.	see below
::variants	Variant selector	Default	opt.	Var1



::parent	Instance name of this instances parent cell	W_NO_PARENT	man.	chip
::inst	Instance name, primary key!	n/a	man.	a_i1
::lang	Language definition	VHDL	opt.	vhdl
::entity	Entity name	W_NO_ENTITY	man.	a
::config	Configuration name	W_NO_CONFIG	man.	a_rtl_conf
::shortname	Short name	<empty>	opt.	text
::use	Additional, project specific libraries	<empty>	opt.	padlib.foo
::comment	Comment field	<empty>	opt.	text

Internally the keywords ::debug, ::hierarchy, ::skip and ::default are used. Please do not use them. Apart from that you are free to add columns of your own. User defined columns are usable in %::NAME% macro expansion and are listed in the intermediate design data output. Only the ::inst column has to contain a value in each row (which could be a %::NAME% macro, though). All other columns will receive more or less reasonable default values in case they are left empty.

### ***HIER columns details***

- ::gen If a cell here is not empty, the line will be considered as generator. See description of generator statements below.
- ::variants Select a line depending on the –variant command line switch. If –variant VAR is set, only lines whose ::variant cell contains the keyword VAR, Default or empty, are selected and read in. Several variants may be given in one cell, separated by “,”. Without specifying the –variant switch, the “Default” and empty ::variant cells are read in and evaluated.
- ::inst Defines the instance name. If the same name appears in several rows, the resulting row will be overloaded from all input rows. The exact behaviour depends on the column name. Some are concatenated, some are replaced.
- ::lang HDL language selection, case insensitive. If this column is omitted or empty, VHDL output is generated. The default value can be changed by means of the macro.%LANGUAGE% macro. Currently only VHDL and Verilog are supported (XXX: Verilog output still t.b.d.).

Additional columns:

- ::arch If no ::arch column is given, architecture will default to "rtl" (configuration hier.field::arch.[3] = rtl; cannot be changed globally)
- ::use Add project specific libraries and work packages to the HDL description files. See the ::use details below.

### ***Special HIER sheet properties***

%TOP%



## ***CONN sheet details***

### ***CONN columns details***

#### ***Special CONN sheet signals***

Special signal names:

%LOW%, %HIGH%, %LOW\_BUS%, %HIGH\_BUS%

#### **%OPEN% aka. open**

Used the %OPEN% signal to leave some pins of module `bar port a open` MIX has no knowledge about ports. Everything is defined in terms of signals and instances. Use the "open" pseudo signal to define the extra pins.

E.g.: wire `foo/a port` to `bar/a port`. `bar/a` has extra pins.

`signal_a, ::high 7, ::low 0, ::out foo/a ::in bar/a`

`%OPEN%, ::high 1, ::low 0, ::out -, ::out bar/a(9:8)`

XXX write this in spreadsheet XXX

You could also force the `::in` pins to high or low, instead, e.g. use %HIGH%,

%HIGH\_BUS%, %LOW

or %LOW\_BUS%. Or a constant.

#### **Constants**

Constant values can be defined and used in several ways in the CONN worksheet. Constants can be marked with a C in the `::mode` column.

Basically anything resembling a number written in the `::out` column will be considered to be a constant value. String and bit vector constants are enclosed in single or double quotes. Remember to type two single quotes in excel to start a single quote string. Excel takes the first quote character to prevent the following string to be interpolated by excel.

If you do not name a constant (leave the `::name` field empty), MIX will generate a name like `mix_const_N`. N starts by one and increments for each new constant value.

Constant values can be assigned to instance ports in the `::in` column of that constant.

Depending on the form of a constant and the output language, MIX tries to convert the constant value into something suitable. See the constant.xls example (XXXLINK).

::ign	::gen	::bundle	::class	::clock	::type	::high	::low	::mode	::name	::out	::in
		CONSTANT			std_ulogic			C	const_01	0	inst_aa/const_01_p
		CONSTANT			std_ulogic			C	const_02	1	inst_aa/const_02_p
		const_signal	c_sig		std_ulogic_vector	6	0		const_03	%CONST%/'64'	inst_aa/const_03
		const_signal	c_sig		std_ulogic_vector	3	0		const_04	%CONST%/'32'	inst_aa/const_04
		const_signal	c_sig		std_ulogic				const_05	1	inst_aa/const_05
		const_signal	c_sig		std_ulogic_vector	6	0		const_06	0xf	inst_aa/const_06_p
# Allowed formats: 0x[0-9a-f], 0[0-7], 10101, 'xxxx', 'zzzz', "abc"											
		Formats			std_ulogic					0	inst_aa/zero_p
		Formats			std_ulogic					1	inst_aa/one_p
		Formats			integer					10	inst_aa/integer_p
		Formats			real				10.2		inst_aa/real_p
		Formats			real				1_000_000.0		inst_aa/under_p
		Formats			integer				16#FF#		inst_aa/vhdl_basehex_p
		Formats			integer				2#1010_1010#		inst_aa/vhdlbase2_p
		Formats			real				2.2E-6		inst_aa/reale_p
		Formats			time				10 ns		inst_aa/int_time_p
		Formats			time				2.27us		inst_aa/real_time_p
		Formats			string				"ein string"		inst_aa/string_p
		Formats			bit_vector	7	0		"11111111"		inst_aa/bit_vector_p
		Formats			bit_vector	7	0		"1010"		inst_aa/bad_width_p
		Formats			std_ulogic_vector	7	0		'01010101'		inst_aa/std_ulogic_vector_p
# Various other constant formats											
		Std_ulogic			std_ulogic_vector	7	0		16#FF#		inst_aa/std_u_logic_vport
		Std_ulogic			std_ulogic_vector	7	0		16#11#		inst_aa/std_u_11_vport
		Std_ulogic			std_ulogic_vector	10	0		16#FF#		inst_aa/std_u_logic_vport_ext
		Std_ulogic			std_ulogic_vector	7	0		0xff		inst_aa/std_u_logic_port_02
		Std_ulogic			std_ulogic_vector	7	0		0b01010101		inst_aa/std_u_logic_bin_p
		Std_ulogic			std_ulogic_vector	7	0		8#07#		inst_aa/std_u_logic_octv_p
		Std_ulogic			std_ulogic_vector	7	0		2#11001100#		inst_aa/std_u_logic_binv_p
		Std_ulogic			std_ulogic_vector	7	0		4#3030#		inst_aa/std_u_logic_quadv_p
		Std_ulogic			std_ulogic_vector	3	0		16#ee#		inst_aa/std_u_logic_hexerr_p

Image 5 Examples for constants

If the target language is VHDL, XXXXX  
MIX looks like in the following screenshot:

```

...
-- Generated Signal List, Excerpt ...
    constant const_01_c : std_ulogic := '0';
    signal const_01 : std_ulogic;
    constant const_02_c : std_ulogic := '1';
    signal const_02 : std_ulogic;
    constant const_03_c : std_ulogic_vector(6 downto 0) := "64"; -- __I_VectorConv
    signal const_03 : std_ulogic_vector(6 downto 0);
    constant mix_const_10_c : string := "ein string"; -- __I_ConstNoconv
    signal mix_const_10 : string;
    constant mix_const_14_c : std_ulogic_vector(7 downto 0) := "11111111"; -- __I_ConvConstant: 16#FF#
    signal mix_const_14 : std_ulogic_vector(7 downto 0);
    constant mix_const_15_c : std_ulogic_vector(7 downto 0) := "00010001"; -- __I_ConvConstant: 16#11#
    signal mix_const_15 : std_ulogic_vector(7 downto 0);
    constant mix_const_16_c : std_ulogic_vector(10 downto 0) := "0001111111"; -- __I_ConvConstant: 16#FF#
    signal mix_const_16 : std_ulogic_vector(10 downto 0);

...
begin
-- Generated Signal Assignments
    const_01 <= const_01_c;
    const_02 <= const_02_c;
    const_03 <= const_03_c;
    mix_const_10 <= mix_const_10_c;
    mix_const_14 <= mix_const_14_c;
    mix_const_15 <= mix_const_15_c;
    mix_const_16 <= mix_const_16_c;

...
-- Generated Instance Port Map for inst_aa
inst_aa: inst_aa_e
port map (
    const_01_p => const_01,
    const_02_p => const_02,
    const_03_p => const_03,
    string_p => mix_const_10,
    std_u_11_vport => mix_const_15,
    std_u_logic_vport => mix_const_14,
    std_u_logic_vport_ext => mix_const_16,
...
);
-- End of Generated Instance Port Map for inst_aa

```

Image 6 Example constant output

Search for lines with `__E` in the comments to detect constants MIX was not able to translate properly. Verilog constant output is not very mature as of today.

## Generics and Parameters

### Predefined and user macros

The following table contains a list of predefined macros. Some of them are defined at run time (e.g. the current date), some are for internal purposes only. macros marked with a yes in the “User” column can be set freely by the user on the command line or in the `mix.cfg` configuration file.

Macro Name	Default Value	Description	User
%0%	mix_0.pl	Program name	run time
%ARGV%	K:\Projects\MIX\PROG\mix_0.pl – listoncf	Program arguments	run time

%BUFFER%	buffer		yes
%BUS_TYPE%	std_ulogic_vector		yes
%CONST%	__CONST__		yes
%DATE%	Mon Jun 30 16:22:41 2003-06-30	Current date	<i>run time</i>
%DEFAULT_MODE%	S	Signal default mode	yes
%EMPTY%		Empty string	yes
%GENERIC%	__GENERIC__		-
%H%	\$	internal	-
%HIGH%	mix_logic1	Name of logic high value	yes
%HIGH_BUS%	mix_logic1_bus	Name of logic high value bus	yes
%HOME%	H:\		-
%IOCELL_SELECT_PORT%	select		yes
%IOCELL_TYPE%	__E_DEFAULT_IOCELL__		-
%IOCR%	\n		-
%LANGUAGE%	vhdl		yes
%LOW%	mix_logic0		yes
%LOW_BUS%	mix_logic0_bus		yes
%NULL%	0		-
%OPEN%	open		-
%PAD_CLASS%	PAD		yes
%PAD_TYPE%	__E_DEFAULT_PAD__		-
%PARAMETER%	__PARAMETER__		-
%PROJECT%	NO_PROJECT_SET		yes
%SIGNAL%	std_ulogic		yes
%SPACE%			yes
%TAB%	\t		-
%TBD%	__W_TO_BE_DEFINED__		-
%TOP%	__TOP__		yes
%UNDEF%	ERROR_UNDEF		-
%UNDEF_1%	ERROR_UNDEF_1		-
%UNDEF_2%	ERROR_UNDEF_2		-
%UNDEF_3%	ERROR_UNDEF_3		-
%UNDEF_4%	ERROR_UNDEF_4		-
%USER%	wig		<i>run time</i>
%VERILOG_TIMESCALE%	'timescale 1ns / 1ns;		yes
%VERSION%	Revision: 1.12		-
%VHDL_NOPROJ%	-- No project specific VHDL libraries		-
%VHDL_USE%	-- No project specific VHDL libraries		yes
%VHDL_USE_ARCH%	%VHDL_USE_DEFAULT%\n%VHDL_USE%		yes
%VHDL_USE_CONF%	%VHDL_USE_DEFAULT%\n%VHDL_USE%		yes
%VHDL_USE_DEFAULT%	library IEEE;\nuse IEEE.std_logic_1164.all;\n		yes
%VHDL_USE_ENTY%	%VHDL_USE_DEFAULT%\n%VHDL_USE%		yes

The default values can be changed by using

```
-conf macro.%THIS_MACRO%=my_value
```

command line switch. New macros can be defined the same way.

Alternatively a line like

```
MIXCFG macro.%THIS_MACRO% my_value
```

to the mix.cfg configuration file will achieve the same result.

## CONN sheet macros

not text macros, but MH, MD and MX generators:

MH: Macro header: needs to match against the MX line (\$X defines variables)

MD: Body of macro. Will be inserted after variable replacement.

MX: Triggers macro expansion. Will be matched against all MH lines known. Defines variable values.

## Generator statements

### IO sheet

A third category of input specification is the IO sheet. The contents of the IO sheet is parsed and translated into instances of io cell blocks and pad cells and connections of the io logic with the design core logic.

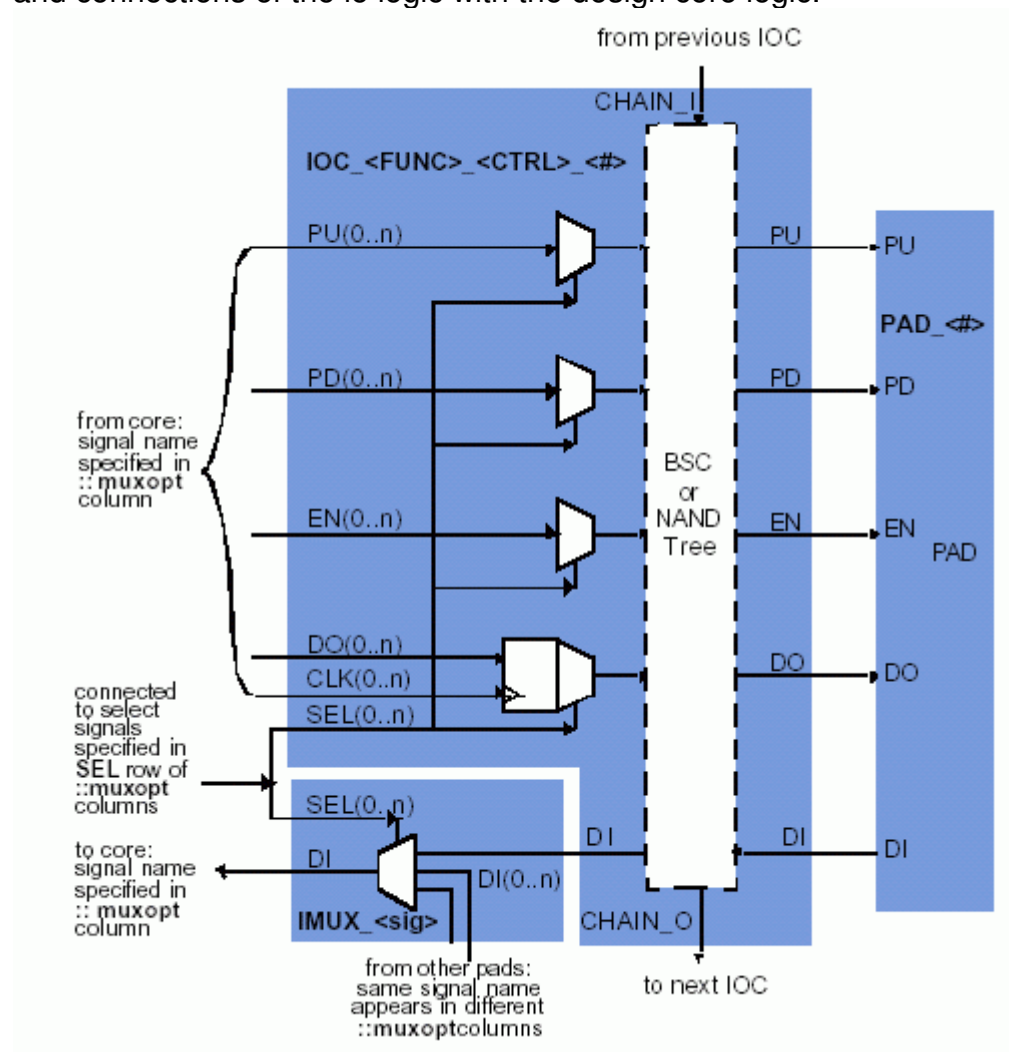


Image 7: IO Cell and Pad Layout

The IO sheet is a simple way to specify the connections of the IO cell to the design core logic. MIX will derive the connections of the DI(0..n), DO(0..n), PU(0..n), PD(0..n), EN(0..n) and the accompanying select lines. MIX will not add special purpose connections and the IO cell to Pad cell connections. This should be done by using the macro and generator statements in the CONN sheets.

A simple example illustrates the various input fields and their usage.

:::ign	:::class	:::ispin	:::pin	:::pad	:::type	:::iocell	:::port	:::name	:::muxopt	:::muxopt	:::muxopt	:::muxopt	
%SEL%								sel	pad	iosel_0	iosel_1	iosel_2	iosel_3
	DATA_I	1	1	1	w_pad_i	ioc_g_i	di	data_i1	data_i1.0	data_i1.1	data_i1.2	data_i1.3	
	DATA_O		1	2	w_pad_o	ioc_g_o	do	data_o1	data_o1.0	data_o1.1	data_o1.2	data_o1.3	
%SEL%								sel	pad	iosel_disp	iosel_ls_min		
	DISPLAY	1	12	12	w_disp	ioc_r_io	di, do, en	disp_2	di2.0, disp2.0, disp2_en.0	, dis_ls_min.0, dis_ls_en			
	DISPLAY	1	13	13	w_disp	ioc_r_io	di, do, en	disp_3	di2.1, disp2.1, disp2_en.1	, dis_ls_min.1, dis_ls_en			
	DISPLAY	1	14	14	w_disp	ioc_r_io	di, do, en	disp_4	di2.3, disp2.3, disp2_en.3	, dis_ls_min.2, dis_ls_en			
	DISPLAY	1	15	15	w_disp	ioc_r_io	di, do, en	disp_5	di2.4, disp2.4, disp2_en.4	, dis_ls_min.3, dis_ls_en			

Image 8: IO Sheet Example

### IO sheet column header

The IO sheet starts with the usual header line, defining mandatory and optional columns:

- :::ign A # marks a comment line. Has to be first column. Optional.
- :::class The class name is forwarded to the :::class field of the CONN sheet. (opt.)
- :::ispin Set to one if this pad is actually bonded (ignored by MIX).
- :::pin Number or name of the pin. (ignored by MIX)
- :::pad The primary key of the IO sheet. Has to be an integer value. The pad numbers do not have to be consecutive. Mandatory column.
- :::type Defines the entity of the generated pad cell.
- :::iocell Defines name and entity of the generated io cell
- :::port Define the io cell port name towards the core logic. Port names are separated by , and/or <Alt><CR>.
- :::name pad name
- :::muxopt Connection matrix of io cell ports to core logic. Signals are separated by , and/or <Alt>-<CR>. Signals may be single bits or a one bit bus slice. Type core\_sig.N or core\_sig(N) for bit N of the core signal core\_sig. :::muxopt is allowed to appear several times. The number of signals and the order matches the port names in the :::port field.

### ***IO sheet %SEL% rows***

The rows with the key %SEL% in the ::class field define the wiring of the IO cells multiplexer select lines. The name in the ::port field defined the io cell port to connect to. The ::name field is ignored. The signal names listed in the ::muxopt columns are connected from the designs core with the appropriate slice of the io cell multiplexer select lines (one hot select lines). The leftmost ::muxopt connects to bit 0, the next to bit 1 and so forth. core\_sel.N or core\_cel(N) . can be used to wire select buses. The number of non-empty consecutive ::muxopt fields also define the width of the multiplexer. The actual multiplexer width is stored in the %::\_muxwidth\_% field.

The given values for the select lines are valid until the last row or another %SEL% line is found.

%NOSEL% will stop usage of in/out multiplexer and select lines.

Setting the configuration switch "iocell.select" to "bus" changes from a one-hot architecture to a select bus architecture. The select bus name is take in the leftmost ::muxopt column, an appropriate width is calculated by the number of defined ::muxopt columns.

### ***IO sheet pad rows***

All other rows make up the connection matrix. For each row an io cell is instantiated. By default this io cell's name is composed by the type listed in the ::type field with the ::pad number attached, separated by a \_.

Secondly a pad cell is instantiated. By default this pad cell's name is composed from the prefix pad\_ and the pad number as given in the ::pad field.

The default naming for both io cell and pad is given by the configuration variables

```
pad.name = %PREFIX_PAD_GEN%::pad%
iocell.name = %::iocell%_%::pad%
```

You can leave unused ::muxopt columns empty. Then MIX will reduce the number of select lines accordingly.

### ***IO cell and Pad connections***

MIX IO sheet parser connects the IO cell internal interface towards the design core logic, only. Connections between pad and io cell need to be defined explicitly, usually be means of ::gen match operators.

Additonal wires for NAND tree or boundary scan are specified the same way. Signal busses need to be defined properly, esp. the width and type. MIX will derive these properties from the definition.

The generated pad and iocells need to be linked into the design hierarchy properly, e.g. by adding ::gen match operators in the HIER sheet.



HER								
:sign	:gen	:variants	:parent	:inst	:lang	:entity	:config	:comment
# Pads, name is pad_NN, NN is the number from :pad column in IO								
		Default	%ICP%	padframe	vnd	padframe_e	:%entity% conf	
/pad (d+)		Default	padframe	pad \$1	vnd		:%entity% conf	
/ioc (.*)		Default	padframe	ioc \$1	vnd		:%entity% conf	

CONN												
:sign	:gen	:bundle	:class	:clock	:type	:high	:low	:mode	:name	:out	:in	:descr
# Standard Pad Cells wiring: do, en, di; triggered by iocell name which is of the form ioc_[rg]_[io...]. N will be derived by the IO..												
/ioc (w_ w?w*) (d+)	PAD_CTRL	IO	multiple	std_logic					pad do \$2	ioc \$1 \$2/p do	pad \$2/do	data out
/ioc (w_ w?w*) (d+)	PAD_CTRL	IO	multiple	std_logic					pad en \$2	ioc \$1 \$2/p en	pad \$2/en	pad enable
/ioc (w_ w?w*) (d+)	PAD_CTRL	IO	multiple	std_logic					pad di \$2	pad \$2/d	ioc \$1 \$2/p di	data in

Image 9: Defining IO and pad cell connections

## IO Parser global configuration

Some global configuration are available to tailor the IO sheet parser behaviour:

iocell.name	Rule determines the naming of the io cell instances. Default: '%::iocell%_%::pad%'
iocell.auto	If set to "bus", create busses as needed. Otherwise MIX relies on appropriate definition of busses by the CONN sheets.
iocell.bus	If iocell.auto is set to bus, MIX will attach the keyword listed here ( _vector) to signal type definitions.
iocell.defaultdir	Defines default iocell port direction. Default: in
iocell.in	comma and/or white-space separated list of in ports Default: do, en, pu, pd, xout
iocell.out	list of io-cells out ports Default: di, xin
iocell.select	List of keywords defining the select multiplexer connections: onehot        use one hot architecture for select lines bus            use a select line bus auto           calculate width of select bus or one-hot based on actually wired ports const          calculate width of select bus or one-hot based on %SEL% width Default: onehot,auto
pad.name	Rule determines the naming of the pad instances. Default: '%PREFIX_PAD_GEN%::pad%'

## I2C sheet

t.b.d.

## VI2C sheet

### Alarm clock example

### MIX converter man page

#### *Synopsis*

#### **Command line switches**

-out OUTPUTFILE.ext defines output filename and type

-outenty OUT-e.vhd|ENTY|COMB  
-- Write all entities into OUT-e.vhd.  
-- If argument is ENTY, each entity will be written into a file called *entityname-e.vhd*.. (The exact naming depends on changeable rules).  
-- If argument is COMB, entity, architecture and configuration will all be written into one file called *entityname.vhd*

-outarch OUT-rtl-a.vhd|ARCH|COMB  
*See description of outenty option.*

-outconf OUT-c.vhd|CONF|COMB  
*See description of outenty option.*

-combine  
write entity, architecture and configuration into one file for each entity. Shortcut for setting -out[enty|arch|conf] to COMB individually.

-dir DIRECTORY  
write intermediate, internal and backend data into the given DIRECTORY. By default MIX writes to the current working directory

-top TOPCELL  
use TOPCELL as top. Default is TESTBENCH or daughter of TESTBENCH.

-adump  
dump internal data in ASCII format, too (debugging, use with small data set).

-variant VAR1  
Select VAR1 from the HIER worksheet.

-conf key.key.key=value  
Overload \$EH{key}{key}{key} with value or add a new configuration variable.

-listconf  
Print out all available/predefined configurations options

-delta  
Output will be compared against previous runs.

-sheet SHEET=MATCH  
SHEET can be one of "hier", "conn", "vi2c". MATCH is a perl regular expression.

# Add your options here ....

"Standard" options:

```
my @stdopts = qw(help|h! verbose|v! quiet|q! nobanner! debug:i  
makeopts=s@ gmakeopts=s@);  
Caveat: the -h option will not work on MS-Windows
```

## ***Runtime options and configuration***

Runtime configuration is controlled by (increasing precedence):

- ❑ built-in default values
- ❑ mix.cfg files  
if found \$HOME, \$PROJECT and/or in current directory.  
Format is: MIXCFG name.of.conf value
- ❑ CONF sheet found in input xls files
- ❑ command line switch  
-conf foo.bar=value
- ❑ dedicated command line options

MIX reads in mix.cfg configuration files in the following locations:

1. \$ENV{HOME}  
'HOMEDRIVE', 'HOMEPATH', 'USERPROFILE' or 'C:\  
(only from the first matching location)
2. \$ENV{PROJECT}
3. . (cwd)

Order: HOME / HOME / cwd() / -conf (last wins)

## ***Misc features***

### **-delta mode**

Do not change output files, but report number of changes. Adds extra sheets DIFF\_CONN and DIFF\_HIER (and old versions of them) to the intermediate output. The FOO.pld internal output gets overwritten, though. All messages are appended to mix\_0.pl

If a new HDL-file needs to be created by the changes, the -delta mode is not be applied for that file, but it is generated as new. If you never have written a FOO-mixed.xls intermediate output, there is no HIER or CONN sheet generated.

By adding the following two lines to your configuration (e.g. ./mix.cfg), delta mode will be the default:

```
MIXCFG output.generate.delta 1  
MIXCFG output.delta sort,remove
```

See the configuration option description for more details.

-nodelta switches delta mode off, then.

### **Intermediate Excel Sheet**

Format of intermediate xls sheet will be kept as is as long as the number and order of columns is unchanged.

MIX saves three old versions of the generated HIER and CONN sheets. The worksheets names are rotated by a trailing \_ and number.

## Output Redirection

MIX writes all output into the current working directory. By using the

```
-dir DIRECTORY
```

option, you can set the output directory for all intermediate, internal and HDL files to DIRECTORY. Absolute path names defined by other options (e.g. – out) are not changed, though. If DIRECTORY does not exist it will be created.

To have separate output directories, use the mix.cfg file:

```
MIXCFG output.path HDLDIRS  
MIXCFG internal.path MIXINTERNAL  
MIXCFG intermediate MIXINTERMEDIATE
```

writes internal, intermediate and HDL files to the given pathes. All directories have to exist and will not be created.

## Alarm clock example

### *Core logic*

### *IO logic*

## Other examples

## Known Bugs and limitations

MS-Win/UNIX end-of-line issue:

Some EDA tools are not able to cope with the different end-of-line (CR vs LF/CR) of UNIX and MS-Windows. Use

```
$ module load freeware; recode "pc..lat1" *.vhd
```

to fix that after transferring data.

## Issue tracking: CADNET -> Issue tracking -> CAD Software -> MIX

If you find unexpected or buggy behavior, please issue a trouble ticket. Don't forget to add a short description. The test case should contain all source files and also log files, the exact command line switches and configuration files applied.

*Please provide a comprehensive description of issues found including all required input data and command line switch to allow fast debugging and fixing.*

## Links

### *MIX paper and documentation*

EDP\_2003\_final\_030331.pdf

MIX\_Specification.xls (see [\\Galaxy\Development\PROJECTS\MIX](#))

MIX\_Intro.ppt (see [\\Galaxy\Development\PROJECTS\MIX](#))

## ***Micronas internal***

Micronas HDL coding guidelines.pdf

## ***Others***

## ***Resources***

Downloads for:

- IO examples
- Standard bus examples
- NAND Tree
- BS example
- miscellaneous useful macros

## ***Used Software***

Perl  
Kommodo  
Vim  
WinCVS

## ***Release Notes:***

### **20030716**

see doc\release\_20030716.txt

### **20030709**

see doc\release\_20030709.txt

### **20030605**

See doc\release\_20030605.txt