JTAG-Booster for

AMD Alchemy Solutions Processors

P.O: Box 1103
Kueferstrasse 8
Tel. +49 (7667) 908-0
sales@fsforth.de

- D-79200 Breisach, Germany
- D-79206 Breisach, Germany
- Fax +49 (7667) 908-200
- http://www.fsforth.de

FS FORTH-SYSTEME GmbH
Postfach 1103, D-79200 Breisach, Germany

Release of Document: November 22, 2004
Author: Dieter Fögele
Filename: JTAG_alchemyb.doc
Program Version: 4.xx

Table of Contents

1. General

The programs JTAu1000.EXE, JTAu1100.EXE, JTAu1500.EXE and JTAu1550.EXE use the IEEE 1149.1 JTAG port of the AMD Alchemy Solutions Processors in conjunction with the small JTAG-Booster:

- to program data into flash memory
- to verify and read the contents of a flash memory
- to make a memory dump
- to access a serial device (I²C/SPI/MicroWire)
- to test CPU signals

All functions are done without any piece of software running in the target. No firmware or BIOS must be written. Bootstrap software may be downloaded to initially unprogrammed memories.

As this tool uses boundary scan, it is extremely simple and very powerful. It assists you in bringing-up new hardware. Even if there are essential bugs in the hardware (i.e. RAM not reliable working, soldering problems with the BGA package), in many cases you are able to load small test programs into flash, which helps you to analyze hardware problems. Or if you have a flash memory, which is not connected correctly to the CPU (i.e. CPU's A0 is connected to a 16 bit AMD flash), we can support you with a special adapted version of the JTAG-Booster.

The JTAG-BOOSTER' s software is highly optimized to the JTAG chain of a specific target CPU. To give support for all processors of the AMD Alchemy Solutions Processors family, there are three different programs on the distribution disk:

- JTAu1000.EXE : Tool for AMD Alchemy Solutions Au1000 Processor
- JTAu1100.EXE : Tool for AMD Alchemy Solutions Au1100 Processor
- JTAu1500.EXE : Tool for AMD Alchemy Solutions Au1500 Processor
- JTAu1550.EXE : Tool for AMD Alchemy Solutions Au1550 Processor

Please contact us, if you need support for other members of the AMD Alchemy Solutions Processors family.

For latest documentation please refer to the file README.TXT on the distribution disk.

1.1. Ordering Information

The following related products are available

- 9019 JTAG-Booster AMD Alchemy Solutions Processors, 3.3V,
  AMD Au1000, Au1100, Au1500, Au1550
  DOS/Win9x/WinNT/Win2000/WinXP
  delivered with adapter type 285
  and additional cable with single strands TK02206

1.2. System Requirements

To successfully run this tool the following requirements must be met:

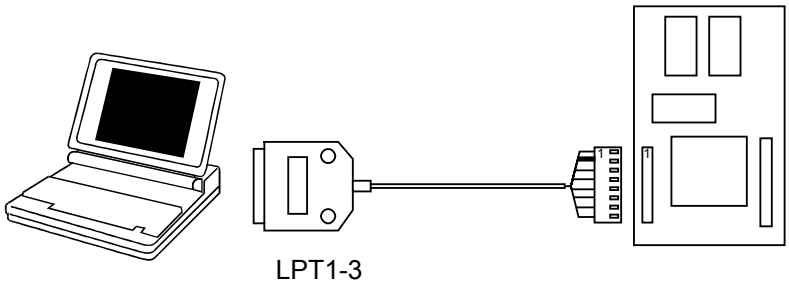
- MSDOS, WIN3.x, WIN9x, WinNT, Win2000 or WindowsXP
  (WinNT/Win2000/WindowsXP is supported with an additional tool, see chapter 5 “Support for Windows NT, Windows 2000 and Windows XP”)
- Intel 80386 or higher
- 205 kByte of free DOS memory
- Parallel Port

1.3. Contents of Distribution Disk

- JTAu1000.EXE JTAu1000.OVL — Tool for AMD Alchemy Solutions Au1000 Processor
- JTAu1000.INI — Template configuration file for AMD Alchemy Solutions Au1000 Processor. See chapter 1.11 "Initialization file JTAuxxxx.INI"
- JTAu1100.EXE JTAu1100.OVL — Tool for AMD Alchemy Solutions Au1100 Processor
- JTAu1100.INI — Template configuration file for AMD Alchemy Solutions Au1100 Processor. See chapter 1.11 "Initialization file JTAuxxxx.INI"
- JTAu1500.EXE JTAu1500.OVL — Tool for AMD Alchemy Solutions Au1500 Processor
- JTAu1500.INI — Template configuration file for AMD Alchemy Solutions Au1500 Processor. See chapter 1.11 "Initialization file JTAuxxxx.INI"
- JTAu1550.EXE JTAu1550.OVL — Tool for AMD Alchemy Solutions Au1550 Processor
- JTAu1550.INI — Template configuration file for AMD Alchemy Solutions Au1550 Processor. See chapter 1.11 "Initialization file JTAuxxxx.INI"
- WinNT — This subdirectory contains the support for Windows NT, Windows 2000 and Windows XP. See chapter 5 "Support for Windows NT, Windows 2000 and Windows XP"
- JTAG_V4xx_FLASHES.pdf — List of all supported Flash devices
- README.txt — Release notes, new features, known problems

1.4. Connecting your PC to the target system

The JTAG-Booster can be plugged into standard parallel ports (LPT1-3) with a DB25-Connector.



LPT1-3

The target end of the cable has the following reference:

1	2*	3	4	5	6	7	8
TCK	GND	TMS	TRST#	NC	TDI	TDO	+3.3V

*PIN 2 can be detected by the thick cable.

To connect your design to the JTAG-BOOSTER you need a single row berg connector with a spacing of 2.54mm on your PCB. The names refer to the target: Pin 7 is the target's TDO pin and is connected to the JTAG-Booster's TDI pin.

The 3.3V version of the JTAG-Booster (FS part number 285) is delivered together with this package. Don't use the 5V version of the JTAG-Booster (FS part number 227) with a 3.3V target. **Don't apply 5V to the 3.3V version of the JTAG-Booster!**

Your target must be able to power the JTAG-Booster, it draws about 100mA.

Before you start the program, the JTAG-BOOSTER must be plugged to a parallel interface of your PC and to the 8 pin JTAG connector on the target.

The utility is started with the general command line format: JTAGxxx

JTAuxxxx /function [filename] [/option_1] ... [/option_n].

Note that the function must be the first argument followed (if needed) by the filename.

If you want to cancel execution of JTAuxxxx, press CTRL-Break-Key.

On any error the program aborts with an MSDOS error level of one.

1.5. First Example with AMD Alchemy Solutions Au1000 Processor

In the following simple example it is assumed that the JTAG-Booster is connected to LPT1 of your PC and target power is on.

Typing

```
JTAu1000  /P  MYAPP.BIN
```

at the DOS prompt results in the following output:

```
JTAu1000 --- JTAG utility for AMD Alchemy Solutions Au1000 Processor
Copyright © FS FORTH-SYSTEME GmbH, Breisach
Version 4.xx of mm/dd/yyyy

(1)   Configuration loaded from file JTAu1000.INI
(2)   Target: Generic Target
(3)   Using LPT at I/O-address 0378h
(4)   JTAG Adapter detected

(5)   1 Device detected in JTAG chain
      Device 0: IDCODE=003E828F  AMD Alchemy Au1000, Revision 0
(6)   Sum of instruction register bits  : 5
(7)   CPU position                      : 0
(8)   Instruction register offset       : 0
(9)   Length of boundary scan reg       : 569

      Looking for a known flash device. Please wait..
(10)  AMD 29LV160B, 3,3V, Boot Block Bottom  detected
(11)  Bus size is 16 Bit
(12)  Erasing Flash-EPROM Block #:0  1  2  3
(13)  Unlock Bypass used
      Programming File MYAPP.BIN
      65536 Bytes programmed successfully

      Erase Time          :       0.8 sec
      Programming Time    :      xx.1 sec
```

(1) The initialization file JTAu1000.INI was found in the current directory.

(2) The target identification line of the initialization file is printed here.

(3) The resulting I/O-address of the parallel port is printed here. With WinNT/Win2000/WinXP you must specify the option /LPT2 to access to the standard address 378h.

(4) A JTAG-Booster is found on the parallel port

(5) The JTAG chain is analyzed. There may be several parts in the JTAG chain. The chain is analyzed and all parts except the AMD Alchemy Solutions Au1000 Processor are switched to bypass mode.

(6) The length of all instruction registers in the JTAG chain are added.

(7) The position of the AMD Alchemy Solutions Au1000 Processor in the JTAG chain is assumed to be zero, if not specified in the command line (see option /CPUPOS=).

(8) The position of the JTAG instruction register of the AMD Alchemy Solutions Au1000 Processor is assumed to be zero, if not specified in the command line (see option /IROFFS=).

(9) The real length of the boundary scan register is displayed here and compared with the boundary scan register length of a AMD Alchemy Solutions Au1000 Processor.

(10) A Flash AMD 29LV160B selected with RCS0# was found.

(11) The resulting data bus size is printed here.

(12) In this example 4 blocks must be erased.

(13) If available, the "Unlock Bypass Mode" is activated. In boundary scan mode, this increases the programming performance by 67..75%.

1.6. First Example with AMD Alchemy Solutions Au1100 Processor

In the following simple example it is assumed that the JTAG-Booster is connected to LPT1 of your PC and target power is on.

Typing

```
JTAu1100  /P  MYAPP.BIN
```

at the DOS prompt results in the following output:

```
JTAu1100 --- JTAG utility for AMD Alchemy Solutions Au1100 Processor
Copyright © FS FORTH-SYSTEME GmbH, Breisach
Version 4.xx of mm/dd/yyyy

(1)   Configuration loaded from file JTAu1100.INI
(2)   Target: Generic Target
(3)   Using LPT at I/O-address 0378h
(4)   JTAG Adapter detected

(5)   1 Device detected in JTAG chain
        Device 0: IDCODE=0020228F  AMD Alchemy Au1100, Revision 0
(6)   Sum of instruction register bits  : 5
(7)   CPU position                      : 0
(8)   Instruction register offset       : 0
(9)   Length of boundary scan reg       : 672

      Looking for a known flash device. Please wait..
(10)  AMD 29LV160B, 3.3V, Boot Block Bottom  detected
(11)  Bus size is 16 Bit
(12)  Erasing Flash-EPROM Block #:0  1  2  3
(13)  Unlock Bypass used
      Programming File MYAPP.BIN
      65536 Bytes programmed successfully

      Erase Time          :       0.8 sec
      Programming Time    :      xx.0 sec
```

(1) The initialization file JTAu1100.INI was found in the current directory.

(2) The target identification line of the initialization file is printed here.

(3) The resulting I/O-address of the parallel port is printed here.

(4) A JTAG-Booster is found on the parallel port

(5) The JTAG chain is analyzed. There may be several parts in the JTAG chain. The chain is analyzed and all parts except the AMD Alchemy Solutions Au1100 Processor are switched to bypass mode.

(6) The length of all instruction registers in the JTAG chain are added.

(7) The position of the AMD Alchemy Solutions Au1100 Processor in the JTAG chain is assumed to be zero, if not specified in the command line (see option /CPUPOS=).

(8) The position of the JTAG instruction register of the AMD Alchemy Solutions Au1100 Processor is assumed to be zero, if not specified in the command line (see option /IROFFS=).

(9) The real length of the boundary scan register is displayed here and compared with the boundary scan register length of a AMD Alchemy Solutions Au1100 Processor.

(10) A Flash AMD 29LV160B selected with RCS0# was found.

(11) The resulting data bus size is printed here.

(12) In this example four block must be erased.

(13) If available, the "Unlock Bypass Mode" is activated. In boundary scan mode, this increases the programming performance by 67..75%.

1.7. First Example with AMD Alchemy Solutions Au1500 Processor

In the following simple example it is assumed that the JTAG-Booster is connected to LPT1 of your PC and target power is on.

Typing

```
JTAu1500  /P  MYAPP.BIN
```

at the DOS prompt results in the following output:

```
JTAu1500 --- JTAG utility for AMD Alchemy Solutions Au1500 Processor
Copyright © FS FORTH-SYSTEME GmbH, Breisach
Version 4.xx of mm/dd/yyyy

(1)   Configuration loaded from file JTAu1500.INI
(2)   Target: Generic Target
(3)   Using LPT at I/O-address 0378h
(4)   JTAG Adapter detected

(5)   1 Device detected in JTAG chain
        Device 0: IDCODE=0010228F  AMD Alchemy Au1500, Revision 0
(6)   Sum of instruction register bits  : 5
(7)   CPU position                      : 0
(8)   Instruction register offset       : 0
(9)   Length of boundary scan reg       : 744

      Looking for a known flash device. Please wait..
(10)  AMD 29LV160B, 3.3V, Boot Block Bottom  detected
(11)  Bus size is 16 Bit
(12)  Erasing Flash-EPROM Block #:0  1  2  3
(13)  Unlock Bypass used
      Programming File MYAPP.BIN
      65536 Bytes programmed successfully

      Erase Time          :       0.8 sec
      Programming Time    :      xx.0 sec
```

(1) The initialization file JTAu1500.INI was found in the current directory.

(2) The target identification line of the initialization file is printed here.

(3) The resulting I/O-address of the parallel port is printed here.

(4) A JTAG-Booster is found on the parallel port

(5) The JTAG chain is analyzed. There may be several parts in the JTAG chain. The chain is analyzed and all parts except the AMD Alchemy Solutions Au1500 Processor are switched to bypass mode.

(6) The length of all instruction registers in the JTAG chain are added.

(7) The position of the AMD Alchemy Solutions Au1500 Processor in the JTAG chain is assumed to be zero, if not specified in the command line (see option /CPUPOS=).

(8) The position of the JTAG instruction register of the AMD Alchemy Solutions Au1500 Processor is assumed to be zero, if not specified in the command line (see option /IROFFS=).

(9) The real length of the boundary scan register is displayed here and compared with the boundary scan register length of a AMD Alchemy Solutions Au1500 Processor.

(10) A Flash AMD 29LV160B selected with RCS0# was found.

(11) The resulting data bus size is printed here.

(12) In this example four block must be erased.

(13) If available, the "Unlock Bypass Mode" is activated. In boundary scan mode, this increases the programming performance by 67..75%.

1.8. First Example with AMD Alchemy Solutions Au1550 Processor

In the following simple example it is assumed that the JTAG-Booster is connected to LPT1 of your PC and target power is on.

Typing

```
JTAu1550  /P  MYAPP.BIN
```

at the DOS prompt results in the following output:

```
JTAu1550 --- JTAG utility for AMD Alchemy Solutions Au1550 Processor
Copyright © FS FORTH-SYSTEME GmbH, Breisach
Version 4.xx of mm/dd/yyyy

(1)   Configuration loaded from file JTAu1550.INI
(2)   Target: Generic Target
(3)   Using LPT at I/O-address 0378h
(4)   JTAG Adapter detected

(5)   1 Device detected in JTAG chain
        Device 0: IDCODE=0030228F  AMD Alchemy Au1550, Revision 0
(6)   Sum of instruction register bits  : 5
(7)   CPU position                      : 0
(8)   Instruction register offset       : 0
(9)   Length of boundary scan reg       : 773

      Looking for a known flash device. Please wait..
(10)  AMD 29LV160B, 3.3V, Boot Block Bottom  detected
(11)  Bus size is 16 Bit
(12)  Erasing Flash-EPROM Block #:0  1  2  3
(13)  Unlock Bypass used
      Programming File MYAPP.BIN
      65536 Bytes programmed successfully

      Erase Time          :       0.8 sec
      Programming Time    :      xx.0 sec
```

(1) The initialization file JTAu1550.INI was found in the current directory.

(2) The target identification line of the initialization file is printed here.

(3) The resulting I/O-address of the parallel port is printed here.

(4) A JTAG-Booster is found on the parallel port

(5) The JTAG chain is analyzed. There may be several parts in the JTAG chain. The chain is analyzed and all parts except the AMD Alchemy Solutions Au1550 Processor are switched to bypass mode.

(6) The length of all instruction registers in the JTAG chain are added.

(7) The position of the AMD Alchemy Solutions Au1550 Processor in the JTAG chain is assumed to be zero, if not specified in the command line (see option /CPUPOS=).

(8) The position of the JTAG instruction register of the AMD Alchemy Solutions Au1550 Processor is assumed to be zero, if not specified in the command line (see option /IROFFS=).

(9) The real length of the boundary scan register is displayed here and compared with the boundary scan register length of a AMD Alchemy Solutions Au1550 Processor.

(10) A Flash AMD 29LV160B selected with RCS0# was found.

(11) The resulting data bus size is printed here.

(12) In this example four block must be erased.

(13) If available, the "Unlock Bypass Mode" is activated. In boundary scan mode, this increases the programming performance by 67..75%.

1.9. Trouble Shooting

Avoid long distances between your Host-PC and the target. If you are using standard parallel extension cable, the JTAG-BOOSTER may not work. Don't use Dongles between the parallel port and the JTAG-BOOSTER.

Switch off all special modes of your printer port (EPP, ECP, ...) in the BIOS setup. Only standard parallel port (SPP) mode is allowed.

If there are problems with autodetection of the flash devices use the /DEVICE= option. To speed up autodetection specify one of the options /8BIT /16BIT or /32BIT.

Don't use hardware protected flash memories.

The used chip selects must be defined as output and inactive in the initialization file (see chapter 1.11 “Initialization file JTAuxxxx.INI”). Also the address bits must be defined as output.

Use the option /NOWRSETUP to speed up flash programming.

1.10. Error Messages

- **80386 or greater required**
  The JTAG-BOOSTER does not work on a 8088/8086 or a 80286 platform.

- **Cable not connected or target power fail**
  The JTAG-Booster (or one of the simple Parallel Port JTAG adapters selected with the options /LATTICE /WIGGLER /PLS) wasn't found. Please check connection to parallel port and connection to target. Check target power. Check the command line options. Check your BIOS-Setup. If you are using this program with WinNT, Win2000 or WinXP you must specify /LPT2 or /LPT-BASE=378 to get access to the standard printer port.

- **Can't open x:\yyy\zzz\JTAuxxxx.OVL**
  The overlay file JTAuxxxx.OVL must be in the same directory as JTAuxxxx.EXE.

- **Configuration file XYZ not found.**
  The file specified with the option /INI= wasn't found.

- **Device offset out of range**
  The value specified with the option /OFFSET= is greater than the size of the detected flash device.

- **Disk full**
  Writing a output file was aborted as a result of missing disk space.

- **Do not specify option /NOCS with any other chip select**
  There is a conflict in the command line.

- **Do not specify option /BYTE-MODE. *Flash device* does not have a byte mode pin.**
  The flash device specified with the option /DEVICE= does not support switching between 16 (or 32) bit mode and 8 bit mode. In practice it does not have a pin with the name BYTE#

- **Error creating file:**
  The output file could not be opened. Please check free disk space or write protection.

- **Error: *Pin-Name* is an output only pin**
  The specified pin cannot be sampled. Check the command line. Check the initialization file.

- **Error: *Pin-Name* is an input only pin**
  The specified pin cannot be activated. Check the command line. Check the initialization file.

- **Error: *Pin-Name* may not be read back**
  The specified pin can be switched to tristate, but cannot be read back. Check the command line.

- **illegal function:**
  The first parameter of the command line must be a valid function. See chapter 2 “JTAuxxxx Parameter Description” for a list of supported functions.

- **illegal number:**
  The specified number couldn’t be interpret as a valid number. Check the relevant number base.

- **illegal option:**
  See chapter 2 “JTAuxxxx Parameter Description” for a list of supported options.

- **illegal Pin Type:**
  The name specified with the option /PIN= must be one of the list of chapter 1.11 ”Initialization file JTAuxxxx.INI”

- **illegal Flash Type:**
  The name specified with the option /DEVICE= must be one of the list of chapter 1.12 “Supported flash devices”

- **Input file not found:**
  The specified file cannot be found

- **Input file is empty:**
  Files with zero length are not accepted

- **" " is undefined**
  Please check the syntax in your configuration file. (See chapter 1.11 “Initialization file JTAuxxxx.INI”).

- **LPTx not installed**
  The LPT port specified with /LPTx cannot be found. Please check the LPT port or specify a installed LPT port. Check your BIOS setup. If you are using this program with WinNT, Win2000 or WinXP you 1st must install the WinNT support package as described in chapter 5"Support for Windows NT, Windows 2000 and Windows XP

- **missing filename**
  Most functions need a filename as second parameter.

- **missing option /SERCLK=**
  Some functions need the option /SERCLK= to be defined.

- **missing option /SERDAT=**
  Some functions need the option /SERDAT= or the options /SERDATO= and /SERDATI= to be defined.

- **missing option /SERCS=**
  Some functions need the option /SERCS= if the option /SPI or the option /MWIRE is specified.

- **missing option /LENGTH=**
  Some functions need the option /LENGTH= to be defined.

- **missing option /PIN=**
  Some functions need the option /PIN= to be defined.

- **More than 9 devices in the JTAG chain or TDO pin stuck at low level**
  The JTAG chain is limited to 9 parts. Check target power. Check the target’s TDO pin.

- **No devices found in JTAG chain or TDO pin stuck at high level**
  A stream of 32 high bits was detected on the pin TDO. TDO may stuck at high level. Check the connection to your target. Check the target power. Check the target’s TDO pin.

- **Option /CPUPOS= out of range**
  The number specified with the option /CPUPOS= must be less or equal to the number of parts minus 1.

- **Option /IROFFS= out of range**
  Please specify a smaller value

- **Part at specified position is not a AMD Alchemy Solutions Processors**
  The option /CPUPOS= points to a part not a AMD Alchemy Solutions Processors

- **Pins specified with /SERCLK= and /SERDAT= must have different control cells**
  The pin specified with the option /SERDAT= must be able to be switched to high impedance while the pin specified with option /SERCLK= is an active output. See chapter 1.11 “Initialization file JTAuxxxx.INI”.

- **Pins specified with /SERCLK= and /SERDATI= must have different control cells**
  The pin specified with the option /SERDATI= must be able to be switched to high impedance while the pin specified with option /SERCLK= is an active output. See chapter 1.11 “Initialization file JTAuxxxx.INI”.

- **Pins specified with /SERDATO= and /SERDATI= must have different control cells**
  The pin specified with the option /SERDATI= must be able to be switched to high impedance while the pin specified with option /SERDATO= is an active output. See chapter 1.11 “Initialization file JTAuxxxx.INI”.

- **Specify only one of these options:**
  Some options are exclusive (i.e. /8BIT and /16BIT). Don't mix them.

- **Sum of instruction register bits to low. Should be at least 5 bits for a AMD Alchemy Solutions Processors**
  The sum of all instruction register bits in the JTAG chain does not fit to the AMD Alchemy Solutions Processors. Check the target connection. Check the target CPU type. Check the settings for /IROFFS= and /CPUPOS= , if there are several parts in the JTAG chain.

- **Target no longer connected**
  There is a cyclic check of the JTAG chain. Check target power. Check target connection.

- **There are unknown parts in the JTAG chain. Please use the option /IROFFS= to specify the instr. reg. offset of the CPU.**
  If there are unknown parts in the JTAG chain, the program isn't able to determine the logical position of the CPU's instruction register.

- **There is no AMD Alchemy Solutions Processors in the JTAG chain**
  No AMD Alchemy Solutions Processors was found in the JTAG chain. Check the target power. Try with option /DRIVER=4 again.

- **Value of option /FILE-OFFSET out of range**
  The value of the option /FILE-OFFSET= points behind end of file.

- **wrong driver #**
  The value specified with the option /DRIVER= is out of range.

- **Wrong Flash Identifier (xxxx)**
  No valid identifier found. Check the specified chip select signal and the bus width. Try with the option /DEVICE= . Use the option /8BIT or /16BIT or /32BIT to specify the correct data bus size.

- **Wrong length of boundary scan register. Should be 569 for a AMD Alchemy Solutions Au1000 Processor. (Should be 672 for a AMD Alchemy Solutions Au1100 Processor. Should be 744 for a AMD Alchemy Solutions Au1500 Processor Should be 773 for a AMD Alchemy Solutions Au1550 Processor.)**
  The length of the boundary scan register of the selected part (if there are more than one in the chain) does not fit to the AMD Alchemy Solutions Processors. Check the target connection. Check the target CPU type. Check the settings for /IROFFS= and /CPUPOS= , if there are several parts in the JTAG chain.

1.11. Initialization file JTAuxxxx.INI

This file is used to define the default direction and level of all CPU signals. This file **must be carefully adapted** to your design with the AMD Alchemy Solutions Processors. The Target-Entry is used to identify your design which is displayed with most commands.

When the program JTAuxxxx.EXE is started it scans the current directory for an existing initialization file named JTAuxxxx.INI. If no entry is found the default values are used. You may also specify the initialization file with the option /INI= . If the specified file isn't found, the program aborts with an error message.

The CPU pins can also be used with the functions /BLINK (chapter 2.9), /PIN? (chapter 2.10) and /SAMPLE (chapter 2.11) to test the signals on your design.

The sample file below represents the values which are used for default initialization when no initialization file could be found in the current directory and no initialization file is specified with the option /INI=.

Changes to the structure of the file could result in errors. Remarks can be added by using //.

Sample File JTAu1000.INI:

```
// Description file for AMD Au1000
Target: Generic Target, 2003/10/02
// Adapt this file carefully to your design!!
// All chip select signals are set to output and inactive.
// All signals should be defined. Undefined signals are set to their defaults.
// Pin names are defined in upper case.
// Low active signals are signed with a trailing #.

// The following pins are complete bidirectional pins.
// The direction of each pin can be set independent of the other pins.
// Each pin can be used as an input.
// During flash programming these pins are switched between
// input/inactive and output/active.
// For Flash programming and other memory accesses
// these pins should be set to Input
RD0             Inp     // SRAM/IO/PCMCIA/FLASH/ROM/LCD Data Bus
RD1             Inp     //
RD2             Inp     //
RD3             Inp     //
RD4             Inp     //
RD5             Inp     //
RD6             Inp     //
RD7             Inp     //
RD8             Inp     //
RD9             Inp     //
RD10            Inp     //
RD11            Inp     //
RD12            Inp     //
RD13            Inp     //
RD14            Inp     //
RD15            Inp     //
RD16            Inp     //
RD17            Inp     //
RD18            Inp     //
RD19            Inp     //
RD20            Inp     //
RD21            Inp     //
```

```
RD22            Inp     //
RD23            Inp     //
RD24            Inp     //
RD25            Inp     //
RD26            Inp     //
RD27            Inp     //
RD28            Inp     //
RD29            Inp     //
RD30            Inp     //
RD31            Inp     //

// The following pins are complete bidirectional pins.
// The direction of each pin can be set independent of the other pins.
// Each pin can be used as an input.
// For Flash Programming these pins must be set to output
RAD0            Out,Lo  // SRAM/IO/PCMCIA/FLASH/ROM/LCD Address Bus
RAD1            Out,Lo  //
RAD2            Out,Lo  //
RAD3            Out,Lo  //
RAD4            Out,Lo  //
RAD5            Out,Lo  //
RAD6            Out,Lo  //
RAD7            Out,Lo  //
RAD8            Out,Lo  //
RAD9            Out,Lo  //
RAD10           Out,Lo  //
RAD11           Out,Lo  //
RAD12           Out,Lo  //
RAD13           Out,Lo  //
RAD14           Out,Lo  //
RAD15           Out,Lo  //
RAD16           Out,Lo  //
RAD17           Out,Lo  //
RAD18           Out,Lo  //
RAD19           Out,Lo  //
RAD20           Out,Lo  //
RAD21           Out,Lo  //
RAD22           Out,Lo  //
RAD23           Out,Lo  //
```

```
RAD24           Out,Lo  //
RAD25           Out,Lo  //
RAD26           Out,Lo  //
RAD27           Out,Lo  //
RAD28           Out,Lo  //
RAD29           Out,Hi  //
RAD30           Out,Lo  //
RAD31           Out,Hi  //

// Group 552:  All pins in this group must be set to the same direction
//             These pins are bidirectional
USBH1P          Inp     //
USBH1M          Inp     //

// Group 546:  All pins in this group must be set to the same direction
//             These pins are bidirectional
USBDP           Inp     // USBH0P
USBDM           Inp     // USBH0M

// The following pins are complete bidirectional pins.
// The direction of each pin can be set independent of the other pins.
// Each pin can be used as an input.
GPIO10          Inp     // U3DSR#
GPIO11          Inp     // U3DCD#
GPIO12          Inp     // U3RI#
S0DOUT          Inp     // GPIO16
S0CLK           Inp     // GPIO17
S0DEN           Inp     // GPIO18
IRDATX          Inp     // GPIO19
I2SDIO          Inp     // GPIO29
ACRST#          Inp     // S1DEN
ACSYNC          Inp     // S1DOUT
ACDO            Inp     // S1CLK
I2SCLK          Inp     // GPIO30
I2SWORD         Inp     // GPIO31
U0TXD           Inp     // GPIO20
U1TXD           Inp     // GPIO21
U2TXD           Inp     // GPIO22
U3TXD           Inp     // GPIO23
```

```
GPIO13          Inp     // U3RTS#
GPIO14          Inp     // U3DTR#
GPIO15          Inp     // IRFIRSEL
N0TXEN          Inp     //
N0TXD0          Inp     //
N0TXD1          Inp     //
N0TXD2          Inp     //
N0TXD3          Inp     //
N0MDC           Inp     //
N0MDIO          Inp     //
N1TXEN          Inp     // GPIO24
N1TXD0          Inp     // GPIO25
N1TXD1          Inp     // GPIO26
N1TXD2          Inp     // GPIO27
N1TXD3          Inp     // GPIO28
N1MDC           Inp     //
N1MDIO          Inp     //
SDD0            Inp     // SDRAM Data Bus
SDD1            Inp     //
SDD2            Inp     //
SDD3            Inp     //
SDD4            Inp     //
SDD5            Inp     //
SDD6            Inp     //
SDD7            Inp     //
SDD8            Inp     //
SDD9            Inp     //
SDD10           Inp     //
SDD11           Inp     //
SDD12           Inp     //
SDD13           Inp     //
SDD14           Inp     //
SDD15           Inp     //
SDD16           Inp     //
SDD17           Inp     //
SDD18           Inp     //
SDD19           Inp     //
SDD20           Inp     //
SDD21           Inp     //
```

SDD22	Inp	//
SDD23	Inp	//
SDD24	Inp	//
SDD25	Inp	//
SDD26	Inp	//
SDD27	Inp	//
SDD28	Inp	//
SDD29	Inp	//
SDD30	Inp	//
SDD31	Inp	//
SDCLK0	Out,Lo	//
SDCLK1	Out,Lo	//
SDCLK2	Out,Lo	//
GPIO0	Inp	//
GPIO1	Inp	//
GPIO2	Inp	// EXTCLK0
GPIO3	Inp	// EXTCLK1
GPIO4	Inp	// DMA_REQ0
GPIO5	Inp	// DMA_REQ1
GPIO6	Inp	// SMROMCKE
GPIO7	Inp	//
GPIO8	Inp	// I2SDI
GPIO9	Inp	// U3CTS#
PIOW#	Out,Hi	// PCMCIA Write Cycle Indication
PIOR#	Out,Hi	// PCMCIA Read Cycle Indication
POE#	Out,Hi	// PCMCIA Output Enable
PREG#	Inp	// PCMCIA register-only access signal
PCE1#	Out,Hi	// PCMCIA Card Enable
PCE2#	Out,Hi	// PCMCIA Card Enable
PWE#	Inp	// PCMCIA write enable
LRD0#	Inp	// LCD Controller Chip Interface read indicator
LRD1#	Inp	// LCD Controller Chip Interface read indicator
LWR0#	Inp	// LCD Controller Chip Interface write indicator
LWR1#	Inp	// LCD Controller Chip Interface write indicator
RBEN0#	Out,Lo	// SRAM/IO/PCMCIA/FLASH/ROM/LCD Byte Enable
RBEN1#	Out,Lo	//
RBEN2#	Out,Lo	//
RBEN3#	Out,Lo	//
RWE#	Out,Hi	// SRAM/IO/PCMCIA/FLASH/ROM/LCD Write Enable

```
ROE#        Out,Hi  // SRAM/IO/PCMCIA/FLASH/ROM/LCD Output Enable
RCS0#       Out,Hi  // SRAM/IO/PCMCIA/FLASH/ROM/LCD Chip Select
RCS1#       Out,Hi  //
RCS2#       Out,Hi  //
RCS3#       Out,Hi  //

// The following pins are output only pins.
// Setting to input (tristate) one of these pins results in an error.
SDCS0#      Out,Hi  // Programmable Chip Select
SDCS1#      Out,Hi  //
SDCS2#      Out,Hi  //
SDWE#       Out,Hi  // SDRAM command Output
SDRAS#      Out,Hi  // SDRAM command Output
SDCAS#      Out,Hi  // SDRAM command Output
SDCKE       Out,Lo  // SDRAM Clock enable
SDQM0#      Out,Hi  // SDRAM Byte Mask
SDQM1#      Out,Hi  // SDRAM Byte Mask
SDQM2#      Out,Hi  // SDRAM Byte Mask
SDQM3#      Out,Hi  // SDRAM Byte Mask
SDBA0       Out,Lo  // SDRAM Bak Address
SDBA1       Out,Lo  // SDRAM Bak Address
SDA0        Out,Lo  // SDRAM Address Output
SDA1        Out,Lo  //
SDA2        Out,Lo  //
SDA3        Out,Lo  //
SDA4        Out,Lo  //
SDA5        Out,Lo  //
SDA6        Out,Lo  //
SDA7        Out,Lo  //
SDA8        Out,Lo  //
SDA9        Out,Lo  //
SDA10       Out,Lo  //
SDA11       Out,Lo  //
SDA12       Out,Lo  //
RESETOUT#   Out,Hi  // Buffered Output of CPU Reset Input
LCLK        Out,Lo  // LCD Controller Chip Interface Clock
```

```
// The following pins are input only.
// Setting to output of one of these pins results in an error.
// Declaration of the direction of these pins is optional.

U0RXD       Inp     // UART0 receive
U1RXD       Inp     // UART1 receive
U2RXD       Inp     // UART2 receive
IRDARX      Inp     // Serial IrDA input
U3RXD       Inp     // UART3 receive
ACDI        Inp     // AC-Link TDM input stream
ACBCLK      Inp     // S1DIN
N0CRS       Inp     //
N0RXD0      Inp     //
N0RXD1      Inp     //
N0RXD2      Inp     //
N0RXD3      Inp     //
N0RXCLK     Inp     //
N0RXDV      Inp     //
N0COL       Inp     //
N0TXCLK     Inp     //
N1CRS       Inp     //
N1RXD0      Inp     //
N1RXD1      Inp     //
N1RXD2      Inp     //
N1RXD3      Inp     //
N1RXCLK     Inp     //
N1RXDV      Inp     //
N1COL       Inp     //
N1TXCLK     Inp     //
VDDXOK      Inp     // VDDX stable
TESTEN      Inp     // Test Enable, should be pulled low
ROMSEL      Inp     // Boot from ROM or SMROM
ROMSIZE     Inp     // ROM width 16 or 32 bit
RESETIN#    Inp     // CPU Reset input
TC0         Inp     // Test Clock Input, should be pulled low
TC1         Inp     // Test Clock Input, should be pulled low
TC2         Inp     // Test Clock Input, should be pulled low
TC3         Inp     // Test Clock Input, should be pulled low
PIOS16#     Inp     // PCMCIA I/O is 16 Bit
```

PWAIT#	Inp	// PCMCIA entend Cycle
LWAIT#	Inp	// LCD Controller Chip Interface Extend Cycle
EWAIT#	Inp	// SRAM/IO/PCMCIA/FLASH/ROM Extend Cycle

Sample File JTAu1100.INI:

```
// Description file for AMD Au1100
Target: Generic Target, 2003/10/02
// Adapt this file carefully to your design!!
// All chip select signals are set to output and inactive.
// All signals should be defined. Undefined signals are set to their defaults.
// Pin names are defined in upper case.
// Low active signals are signed with a trailing #.

// The following pins are complete bidirectional pins.
// The direction of each pin can be set independent of the other pins.
// Each pin can be used as an input.
// During flash programming these pins are switched between
// input/inactive and output/active.
// For Flash programming and other memory accesses
// these pins should be set to Input
RD0             Inp     // SRAM/IO/PCMCIA/FLASH/ROM/LCD Data Bus
RD1             Inp     //
RD2             Inp     //
RD3             Inp     //
RD4             Inp     //
RD5             Inp     //
RD6             Inp     //
RD7             Inp     //
RD8             Inp     //
RD9             Inp     //
RD10            Inp     //
RD11            Inp     //
RD12            Inp     //
RD13            Inp     //
RD14            Inp     //
RD15            Inp     //
RD16            Inp     //
RD17            Inp     //
RD18            Inp     //
RD19            Inp     //
RD20            Inp     //
RD21            Inp     //
```

```
RD22            Inp     //
RD23            Inp     //
RD24            Inp     //
RD25            Inp     //
RD26            Inp     //
RD27            Inp     //
RD28            Inp     //
RD29            Inp     //
RD30            Inp     //
RD31            Inp     //

// The following pins are complete bidirectional pins.
// The direction of each pin can be set independent of the other pins.
// Each pin can be used as an input.
// For Flash Programming these pins must be set to output
RAD0            Out,Lo  // SRAM/IO/PCMCIA/FLASH/ROM/LCD Address Bus
RAD1            Out,Lo  //
RAD2            Out,Lo  //
RAD3            Out,Lo  //
RAD4            Out,Lo  //
RAD5            Out,Lo  //
RAD6            Out,Lo  //
RAD7            Out,Lo  //
RAD8            Out,Lo  //
RAD9            Out,Lo  //
RAD10           Out,Lo  //
RAD11           Out,Lo  //
RAD12           Out,Lo  //
RAD13           Out,Lo  //
RAD14           Out,Lo  //
RAD15           Out,Lo  //
RAD16           Out,Lo  //
RAD17           Out,Lo  //
RAD18           Out,Lo  //
RAD19           Out,Lo  //
RAD20           Out,Lo  //
RAD21           Out,Lo  //
RAD22           Out,Lo  //
RAD23           Out,Lo  //
```

```
RAD24           Out,Lo  //
RAD25           Out,Lo  //
RAD26           Out,Lo  //
RAD27           Out,Lo  //
RAD28           Out,Lo  //
RAD29           Out,Hi  //
RAD30           Out,Lo  //
RAD31           Out,Hi  //

// Group 666:  All pins in this group must be set to the same direction
//             These pins are bidirectional
USBH1P          Inp     //
USBH1M          Inp     //

// Group 659:  All pins in this group must be set to the same direction
//             These pins are bidirectional
USBDP           Inp     //
USBDM           Inp     //

// The following pins are complete bidirectional pins.
// The direction of each pin can be set independent of the other pins.
// Each pin can be used as an input.
GPIO10          Inp     // U3DSR#
GPIO11          Inp     // U3DCD#
GPIO12          Inp     // U3RI#
S0DOUT          Inp     // GPIO208
S0CLK           Inp     // GPIO209
S0DEN           Inp     // GPIO210
IRDATX          Inp     // GPIO211
I2SDIO          Inp     // GPIO29
ACRST#          Inp     // S1DEN#
ACSYNC          Inp     // S1DOUT
ACDO            Inp     // S1CLK
I2SCLK          Inp     // GPIO30
I2SWORD         Inp     // GPIO31
U0TXD           Inp     // GPIO212
U1TXD           Inp     // GPIO213
U3TXD           Inp     // GPIO214
GPIO13          Inp     // U3RTS#
```

```
GPIO14          Inp     // U3DTR#
GPIO15          Inp     // IRFIRSEL
N0TXEN          Inp     // GPIO24
N0TXD0          Inp     // GPIO25
N0TXD1          Inp     // GPIO26
N0TXD2          Inp     // GPIO27
N0TXD3          Inp     // GPIO28
N0MDC           Inp     // GPIO215
N0MDIO          Inp     //
GPIO16          Inp     //
GPIO17          Inp     //
GPIO18          Inp     //
GPIO19          Inp     //
GPIO20          Inp     //
GPIO21          Inp     //
GPIO22          Inp     //
GPIO23          Inp     //
RESVD_3         Inp     // Reserved, should be left open
LCD_PWM0        Inp     // Pulse Width Modulation Clock 0
LCD_PWM1        Inp     // Pulse Width Modulation Clock 1
SDMS0_CLK       Out,Lo  // SD Card 0 Interface Clock
SDMS0_CMD       Inp     // SD Card 0 Half Duplex Command and Response
SDMS0_DAT0      Inp     // SD Card 0 Data Bus
SDMS0_DAT1      Inp     // SD Card 0 Data Bus
SDMS0_DAT2      Inp     // SD Card 0 Data Bus
SDMS0_DAT3      Inp     // SD Card 0 Data Bus
SDMS1_CLK       Out,Lo  // SD Card 1 Interface Clock
SDMS1_CMD       Inp     // SD Card 1 Half Duplex Command and Response
SDMS1_DAT0      Inp     // SD Card 1 Data Bus
SDMS1_DAT1      Inp     // SD Card 1 Data Bus
SDMS1_DAT2      Inp     // SD Card 1 Data Bus
SDMS1_DAT3      Inp     // SD Card 1 Data Bus
SDD0            Inp     // SDRAM Data Bus
SDD1            Inp     //
SDD2            Inp     //
SDD3            Inp     //
SDD4            Inp     //
SDD5            Inp     //
SDD6            Inp     //
```

```
SDD7        Inp     //
SDD8        Inp     //
SDD9        Inp     //
SDD10       Inp     //
SDD11       Inp     //
SDD12       Inp     //
SDD13       Inp     //
SDD14       Inp     //
SDD15       Inp     //
SDD16       Inp     //
SDD17       Inp     //
SDD18       Inp     //
SDD19       Inp     //
SDD20       Inp     //
SDD21       Inp     //
SDD22       Inp     //
SDD23       Inp     //
SDD24       Inp     //
SDD25       Inp     //
SDD26       Inp     //
SDD27       Inp     //
SDD28       Inp     //
SDD29       Inp     //
SDD30       Inp     //
SDD31       Inp     //
SDCLK0      Out,Lo  //
SDCLK1      Out,Lo  //
SDCLK2      Out,Lo  //
GPIO0       Inp     //
GPIO1       Inp     //
GPIO2       Inp     // EXTCLK0
GPIO3       Inp     // EXTCLK1
GPIO4       Inp     // DMA_REQ0
GPIO5       Inp     // DMA_REQ1
GPIO6       Inp     // SMROMCKE
GPIO7       Inp     //
GPIO8       Inp     // I2SDI
GPIO9       Inp     // U3CTS#
PIOW#       Out,Hi  // PCMCIA Write Cycle Indication
```

```
PIOR#       Out,Hi  // PCMCIA Read Cycle Indication
POE#        Out,Hi  // PCMCIA Output Enable
PREG#       Inp     // GPIO204
PCE1#       Out,Hi  // GPIO205
PCE2#       Out,Hi  // GPIO206
PWE#        Inp     // GPIO207
LRD0#       Inp     // GPIO200
LRD1#       Inp     // GPIO201
LWR0#       Inp     // GPIO202
LWR1#       Inp     // GPIO203
RBEN0#      Out,Lo  // SRAM/IO/PCMCIA/FLASH/ROM/LCD Byte Enable
RBEN1#      Out,Lo  //
RBEN2#      Out,Lo  //
RBEN3#      Out,Lo  //
RWE#        Out,Hi  // SRAM/IO/PCMCIA/FLASH/ROM/LCD Write Enable
ROE#        Out,Hi  // SRAM/IO/PCMCIA/FLASH/ROM/LCD Output Enable
RCS0#       Out,Hi  // SRAM/IO/PCMCIA/FLASH/ROM/LCD Chip Select
RCS1#       Out,Hi  //
RCS2#       Out,Hi  //
RCS3#       Out,Hi  //
LCD_D0      Out,Lo  // LCD Data Output
LCD_D1      Out,Lo  //
LCD_D2      Out,Lo  //
LCD_D3      Out,Lo  //
LCD_D4      Out,Lo  //
LCD_D5      Out,Lo  //
LCD_D6      Out,Lo  //
LCD_D7      Out,Lo  //
LCD_D8      Out,Lo  //
LCD_D9      Out,Lo  //
LCD_D10     Out,Lo  //
LCD_D11     Out,Lo  //
LCD_D12     Out,Lo  //
LCD_D13     Out,Lo  //
LCD_D14     Out,Lo  //
LCD_D15     Out,Lo  //
LCD_LCK     Out,Lo  // LCD Line Clock
LCD_FCK     Out,Lo  // LCD Frame Clock
LCD_BIAS    Out,Lo  // LCD Bias Clock
```

```
LCD_LEND      Out,Lo   // LCD Line End

// The following pins are output only pins.
// Setting to input (tristate) one of these pins results in an error.
SDCS0#        Out,Hi   // Programmable Chip Select
SDCS1#        Out,Hi   //
SDCS2#        Out,Hi   //
SDWE#         Out,Hi   // SDRAM command Output
SDRAS#        Out,Hi   // SDRAM command Output
SDCAS#        Out,Hi   // SDRAM command Output
SDCKE         Out,Lo   // SDRAM Clock enable
SDQM0#        Out,Hi   // SDRAM Byte Mask
SDQM1#        Out,Hi   // SDRAM Byte Mask
SDQM2#        Out,Hi   // SDRAM Byte Mask
SDQM3#        Out,Hi   // SDRAM Byte Mask
SDBA0         Out,Lo   // SDRAM Bak Address
SDBA1         Out,Lo   // SDRAM Bak Address
SDA0          Out,Lo   // SDRAM Address Output
SDA1          Out,Lo   //
SDA2          Out,Lo   //
SDA3          Out,Lo   //
SDA4          Out,Lo   //
SDA5          Out,Lo   //
SDA6          Out,Lo   //
SDA7          Out,Lo   //
SDA8          Out,Lo   //
SDA9          Out,Lo   //
SDA10         Out,Lo   //
SDA11         Out,Lo   //
SDA12         Out,Lo   //
RESETOUT#     Out,Hi   // Buffered Output of CPU Reset Input
LCLK          Out,Lo   // LCD Controller Chip Interface Clock
LCD_PCK       Out,Lo   // LCD Pixel Clock

// The following pins are input only.
// Setting to output of one of these pins results in an error.
// Declaration of the direction of these pins is optional.
U0RXD         Inp      // UART0 receive
U1RXD         Inp      // UART1 receive
```

```
IRDARX      Inp     // Serial IrDA input
U3RXD       Inp     // UART3 receive
ACDI        Inp     // AC-Link TDM input stream
ACBCLK      Inp     // S1DIN
N0CRS       Inp     //
N0RXD0      Inp     //
N0RXD1      Inp     //
N0RXD2      Inp     //
N0RXD3      Inp     //
N0RXCLK     Inp     //
N0RXDV      Inp     //
N0COL       Inp     //
N0TXCLK     Inp     //
RESVD_0     Inp     // Reserved, should be pulled low
RESVD_1     Inp     // Reserved, should be pulled low
RESVD_2     Inp     // Reserved, should be pulled low
RESVD_4     Inp     // Reserved, should be pulled low
RESVD_5     Inp     // Reserved, should be pulled low
VSEL        Inp     // External SDRAM Voltage Type: 0=2.5V, 1=3.3V
TESTEN      Inp     // Test Enable, should be pulled low
ROMSEL      Inp     // Boot from ROM or SMROM
ROMSIZE     Inp     // ROM width 16 or 32 bit
RESETIN#    Inp     // CPU Reset input
TC0         Inp     // Test Clock Input, should be pulled low
TC1         Inp     // Test Clock Input, should be pulled low
TC2         Inp     // Test Clock Input, should be pulled low
TC3         Inp     // Test Clock Input, should be pulled low
PIOS16#     Inp     // PCMCIA I/O is 16 Bit
PWAIT#      Inp     // PCMCIA entend Cycle
LWAIT#      Inp     // LCD Controller Chip Interface Extend Cycle
EWAIT#      Inp     // SRAM/IO/PCMCIA/FLASH/ROM Extend Cycle
```

Sample File JTAu1500.INI:

```
// Description file for AMD Alchemy Solutions Au1500 Processor
Target: Generic Target, 2003/10/02
// Adapt this file carefully to your design!!
// All chip select signals are set to output and inactive.
// All signals should be defined. Undefined signals are set to their defaults.
// Pin names are defined in upper case.
// Low active signals are signed with a trailing #.

// The following pins are complete bidirectional pins.
// The direction of each pin can be set independent of the other pins.
// Each pin can be used as an input.
// During flash programming these pins are switched between
// input/inactive and output/active.
// For Flash programming and other memory accesses
// these pins should be set to Input
RD0             Inp     // SRAM/IO/PCMCIA/FLASH/ROM/LCD Data Bus
RD1             Inp     //
RD2             Inp     //
RD3             Inp     //
RD4             Inp     //
RD5             Inp     //
RD6             Inp     //
RD7             Inp     //
RD8             Inp     //
RD9             Inp     //
RD10            Inp     //
RD11            Inp     //
RD12            Inp     //
RD13            Inp     //
RD14            Inp     //
RD15            Inp     //
RD16            Inp     //
RD17            Inp     //
RD18            Inp     //
RD19            Inp     //
RD20            Inp     //
RD21            Inp     //
```

```
RD22            Inp     //
RD23            Inp     //
RD24            Inp     //
RD25            Inp     //
RD26            Inp     //
RD27            Inp     //
RD28            Inp     //
RD29            Inp     //
RD30            Inp     //
RD31            Inp     //

// The following pins are complete bidirectional pins.
// The direction of each pin can be set independent of the other pins.
// Each pin can be used as an input.
// For Flash Programming these pins must be set to output
RAD0            Out,Lo  // SRAM/IO/PCMCIA/FLASH/ROM/LCD Address Bus
RAD1            Out,Lo  //
RAD2            Out,Lo  //
RAD3            Out,Lo  //
RAD4            Out,Lo  //
RAD5            Out,Lo  //
RAD6            Out,Lo  //
RAD7            Out,Lo  //
RAD8            Out,Lo  //
RAD9            Out,Lo  //
RAD10           Out,Lo  //
RAD11           Out,Lo  //
RAD12           Out,Lo  //
RAD13           Out,Lo  //
RAD14           Out,Lo  //
RAD15           Out,Lo  //
RAD16           Out,Lo  //
RAD17           Out,Lo  //
RAD18           Out,Lo  //
RAD19           Out,Lo  //
RAD20           Out,Lo  //
RAD21           Out,Lo  //
RAD22           Out,Lo  //
RAD23           Out,Lo  //
```

```
RAD24           Out,Lo  //
RAD25           Out,Lo  //
RAD26           Out,Lo  //
RAD27           Out,Lo  //
RAD28           Out,Lo  //
RAD29           Out,Hi  //
RAD30           Out,Lo  //
RAD31           Out,Hi  //

// Group 151:  All pins in this group must be set to the same direction
//             These pins are bidirectional
USBH1P          Inp     //
USBH1M          Inp     //

// Group 146:  All pins in this group must be set to the same direction
//             These pins are bidirectional
USBDP           Inp     //
USBDM           Inp     //

// The following pins are complete bidirectional pins.
// The direction of each pin can be set independent of the other pins.
// Each pin can be used as an input.
GPIO10          Inp     // U3DSR#
GPIO11          Inp     // U3DCD#
GPIO12          Inp     // U3RI#
GPIO200         Inp     // PCI_RSTO#
GPIO201         Inp     //
GPIO202         Inp     //
GPIO203         Inp     //
GPIO204         Inp     //
GPIO205         Inp     //
GPIO206         Inp     //
GPIO207         Inp     //
GPIO208         Inp     // DMA_REQ2
GPIO209         Inp     // DMA_REQ3
GPIO210         Inp     //
GPIO211         Inp     //
GPIO212         Inp     //
GPIO213         Inp     //
```

```
GPIO214         Inp     //
GPIO215         Inp     //
ACRST#          Inp     // AC-Link CODEC reset
ACSYNC          Inp     // AC-Link fixed rate sample sync
ACDO            Inp     // AC-Link TDM output stream
U0TXD           Inp     // GPIO20
U3TXD           Inp     // GPIO23
GPIO13          Inp     // U3RTS#
GPIO14          Inp     // U3DTR#
GPIO15          Inp     //
N0TXEN          Inp     //
N0TXD0          Inp     //
N0TXD1          Inp     //
N0TXD2          Inp     //
N0TXD3          Inp     //
N0MDC           Inp     //
N0MDIO          Inp     //
N1TXEN          Inp     //
N1TXD0          Inp     //
N1TXD1          Inp     //
N1TXD2          Inp     //
N1TXD3          Inp     //
N1MDC           Inp     //
N1MDIO          Inp     //
SDD0            Inp     // SDRAM Data Bus
SDD1            Inp     //
SDD2            Inp     //
SDD3            Inp     //
SDD4            Inp     //
SDD5            Inp     //
SDD6            Inp     //
SDD7            Inp     //
SDD8            Inp     //
SDD9            Inp     //
SDD10           Inp     //
SDD11           Inp     //
SDD12           Inp     //
SDD13           Inp     //
SDD14           Inp     //
```

```
SDD15       Inp     //
SDD16       Inp     //
SDD17       Inp     //
SDD18       Inp     //
SDD19       Inp     //
SDD20       Inp     //
SDD21       Inp     //
SDD22       Inp     //
SDD23       Inp     //
SDD24       Inp     //
SDD25       Inp     //
SDD26       Inp     //
SDD27       Inp     //
SDD28       Inp     //
SDD29       Inp     //
SDD30       Inp     //
SDD31       Inp     //
SDCLK0      Out,Lo  //
SDCLK1      Out,Lo  //
SDCLK2      Out,Lo  //
GPIO0       Inp     //
GPIO1       Inp     //
GPIO2       Inp     // EXTCLK0
GPIO3       Inp     // EXTCLK1
GPIO4       Inp     // DMA_REQ0
GPIO5       Inp     // DMA_REQ1
GPIO6       Inp     // SMROMCKE
GPIO7       Inp     //
GPIO8       Inp     //
GPIO9       Inp     // U3CTS#
PIOW#       Out,Hi  // PCMCIA Write Cycle Indication
PIOR#       Out,Hi  // PCMCIA Read Cycle Indication
POE#        Out,Hi  // PCMCIA Output Enable
PREG#       Out,Hi  // PCMCIA register-only access signal
PCE1#       Out,Hi  // PCMCIA card enable
PCE2#       Out,Hi  // PCMCIA card enable
PWE#        Out,Hi  // PCMCIA write enable
LRD0#       Out,Hi  // LCD controller chip interface read indicator
LRD1#       Out,Hi  // LCD controller chip interface read indicator
```

```
LWR0#          Out,Hi  // LCD controller chip interface write indicator
LWR1#          Out,Hi  // LCD controller chip interface write indicator
RBEN0#         Out,Lo  // SRAM/IO/PCMCIA/FLASH/ROM/LCD Byte Enable
RBEN1#         Out,Lo  //
RBEN2#         Out,Lo  //
RBEN3#         Out,Lo  //
RWE#           Out,Hi  // SRAM/IO/PCMCIA/FLASH/ROM/LCD Write Enable
ROE#           Out,Hi  // SRAM/IO/PCMCIA/FLASH/ROM/LCD Output Enable
RCS0#          Out,Hi  // SRAM/IO/PCMCIA/FLASH/ROM/LCD Chip Select
RCS1#          Out,Hi  //
RCS2#          Out,Hi  //
RCS3#          Out,Hi  //
PCI_DEVSEL#    Inp     // PCI device select
PCI_STOP#      Inp     // PCI target stop
PCI_FRAME#     Inp     // PCI frame
PCI_IRDY#      Inp     // PCI initiator ready
PCI_TRDY#      Inp     // PCI target ready
PCI_PAR        Inp     // PCI parity
PCI_GNT0#      Inp     // PCI arbiter bus grant output
PCI_GNT1#      Inp     // PCI arbiter bus grant output
PCI_GNT2#      Inp     // PCI arbiter bus grant output
PCI_GNT3#      Inp     // PCI arbiter bus grant output
PCI_PERR#      Inp     // PCI parity error
PCI_SERR#      Inp     // PCI system error
PCI_AD0        Inp     // PCI address/data
PCI_AD1        Inp     //
PCI_AD2        Inp     //
PCI_AD3        Inp     //
PCI_AD4        Inp     //
PCI_AD5        Inp     //
PCI_AD6        Inp     //
PCI_AD7        Inp     //
PCI_AD8        Inp     //
PCI_AD9        Inp     //
PCI_AD10       Inp     //
PCI_AD11       Inp     //
PCI_AD12       Inp     //
PCI_AD13       Inp     //
PCI_AD14       Inp     //
```

```
PCI_AD15        Inp     //
PCI_AD16        Inp     //
PCI_AD17        Inp     //
PCI_AD18        Inp     //
PCI_AD19        Inp     //
PCI_AD20        Inp     //
PCI_AD21        Inp     //
PCI_AD22        Inp     //
PCI_AD23        Inp     //
PCI_AD24        Inp     //
PCI_AD25        Inp     //
PCI_AD26        Inp     //
PCI_AD27        Inp     //
PCI_AD28        Inp     //
PCI_AD29        Inp     //
PCI_AD30        Inp     //
PCI_AD31        Inp     //
PCI_CBE0#       Inp     // PCI bus command and byte enable
PCI_CBE1#       Inp     //
PCI_CBE2#       Inp     //
PCI_CBE3#       Inp     //

// The following pins are output only pins.
// Setting to input (tristate) one of these pins results in an error.
SDCS0#          Out,Hi  // SDRAM Programmable Chip Select
SDCS1#          Out,Hi  //
SDCS2#          Out,Hi  //
SDWE#           Out,Hi  // SDRAM command Output
SDRAS#          Out,Hi  // SDRAM command Output
SDCAS#          Out,Hi  // SDRAM command Output
SDCKE           Out,Lo  // SDRAM Clock enable
SDQM0#          Out,Hi  // SDRAM Byte Mask
SDQM1#          Out,Hi  // SDRAM Byte Mask
SDQM2#          Out,Hi  // SDRAM Byte Mask
SDQM3#          Out,Hi  // SDRAM Byte Mask
SDBA0           Out,Lo  // SDRAM Bank Address
SDBA1           Out,Lo  // SDRAM Bank Address
SDA0            Out,Lo  // SDRAM Address Output
SDA1            Out,Lo  //
```

```
SDA2            Out,Lo  //
SDA3            Out,Lo  //
SDA4            Out,Lo  //
SDA5            Out,Lo  //
SDA6            Out,Lo  //
SDA7            Out,Lo  //
SDA8            Out,Lo  //
SDA9            Out,Lo  //
SDA10           Out,Lo  //
SDA11           Out,Lo  //
SDA12           Out,Lo  //
RESETOUT#       Out,Hi  // Buffered Output of CPU Reset Input
LCLK            Out,Lo  // LCD Controller Chip Interface Clock
PCI_CLKO        Out,Lo  // PCI clock output

// The following pins are input only.
// Setting to output of one of these pins results in an error.
// Declaration of the direction of these pins is optional.
U0RXD           Inp     // UART0 receive
U3RXD           Inp     // UART3 receive
PCI_LOCK#       Inp     // PCI lock fopr atomic operations
PCI_CLK         Inp     // PCI input clock
PCI_REQ0#       Inp     // PCI arbiter bus request input
PCI_REQ1#       Inp     // PCI arbiter bus request input
PCI_REQ2#       Inp     // PCI arbiter bus request input
PCI_REQ3#       Inp     // PCI arbiter bus request input
PCI_CFG         Inp     // PCI configuration, 0->sattelite mode, 1->host mode
PCI_IDSEL       Inp     // PCI initialization device select input
PCI_INTA#       Inp     // PCI interrupt input
PCI_INTB#       Inp     // PCI interrupt input
PCI_INTC#       Inp     // PCI interrupt input
PCI_INTD#       Inp     // PCI interrupt input
PCI_RST#        Inp     // PCI reset input
ACDI            Inp     // AC-Link TDM input stream
ACBCLK          Inp     // AC-Link serial data clock
N0CRS           Inp     //
N0RXD0          Inp     //
N0RXD1          Inp     //
N0RXD2          Inp     //
```

```
N0RXD3      Inp     //
N0RXCLK     Inp     //
N0RXDV      Inp     //
N0COL       Inp     //
N0TXCLK     Inp     //
N1CRS       Inp     //
N1RXD0      Inp     //
N1RXD1      Inp     //
N1RXD2      Inp     //
N1RXD3      Inp     //
N1RXCLK     Inp     //
N1RXDV      Inp     //
N1COL       Inp     //
N1TXCLK     Inp     //
VDDXOK      Inp     // VDDX stable
TESTEN      Inp     // Test enable, should be pulled low
ROMSEL      Inp     // Boot from ROM or SMROM
ROMSIZE     Inp     // ROM width 16 or 32 bit
RESETIN#    Inp     // CPU Reset input
TC0         Inp     // Test Clock Input, should be pulled low
TC1         Inp     // Test Clock Input, should be pulled low
TC2         Inp     // Test Clock Input, should be pulled low
TC3         Inp     // Test Clock Input, should be pulled low
PIOS16#     Inp     // PCMCIA I/O is 16 Bit
PWAIT#      Inp     // PCMCIA entend Cycle
LWAIT#      Inp     // LCD Controller Chip Interface Extend Cycle
EWAIT#      Inp     // SRAM/IO/Flash/ROM Extend Cycle
```

Sample File JTAu1550.INI:

```
// Description file for AMD Alchemy Solutions Au1550 Processor
Target: Generic Target, 2004/11/19
// Adapt this file carefully to your design!!
// All chip select signals are set to output and inactive.
// All signals should be defined. Undefined signals are set to their defaults.
// Pin names are defined in upper case.
// Low active signals are signed with a trailing #.

// The following pins are complete bidirectional pins.
// The direction of each pin can be set independent of the other pins.
// Each pin can be used as an input.
// During flash programming these pins are switched between
// input/inactive and output/active.
// For Flash programming and other memory accesses
// these pins should be set to Input
RD0             Inp     // SRAM/IO/PCMCIA/FLASH/ROM/NAND Data Bus
RD1             Inp     //
RD2             Inp     //
RD3             Inp     //
RD4             Inp     //
RD5             Inp     //
RD6             Inp     //
RD7             Inp     //
RD8             Inp     //
RD9             Inp     //
RD10            Inp     //
RD11            Inp     //
RD12            Inp     //
RD13            Inp     //
RD14            Inp     //
RD15            Inp     //
RD16            Inp     //
RD17            Inp     //
RD18            Inp     //
RD19            Inp     //
RD20            Inp     //
RD21            Inp     //
RD22            Inp     //
RD23            Inp     //
RD24            Inp     //
RD25            Inp     //
```

```
RD26            Inp     //
RD27            Inp     //
RD28            Inp     //
RD29            Inp     //
RD30            Inp     //
RD31            Inp     //

// The following pins are complete bidirectional pins.
// The direction of each pin can be set independent of the other pins.
// Each pin can be used as an input.
// For Flash Programming these pins must be set to output
RAD0            Out,Lo  // SRAM/IO/PCMCIA/FLASH/ROM/NAND Address Bus
RAD1            Out,Lo  //
RAD2            Out,Lo  //
RAD3            Out,Lo  //
RAD4            Out,Lo  //
RAD5            Out,Lo  //
RAD6            Out,Lo  //
RAD7            Out,Lo  //
RAD8            Out,Lo  //
RAD9            Out,Lo  //
RAD10           Out,Lo  //
RAD11           Out,Lo  //
RAD12           Out,Lo  //
RAD13           Out,Lo  //
RAD14           Out,Lo  //
RAD15           Out,Lo  //
RAD16           Out,Lo  //
RAD17           Out,Lo  //
RAD18           Out,Lo  //
RAD19           Out,Lo  //
RAD20           Out,Lo  //
RAD21           Out,Lo  //
RAD22           Out,Lo  //
RAD23           Out,Lo  //
RAD24           Out,Lo  //
RAD25           Out,Lo  //
RAD26           Out,Lo  //
RAD27           Out,Lo  //
RAD28           Out,Lo  //

// Group 765:  All pins in this group must be set to the same direction
//             These pins are bidirectional
```

```
USBH0P          Inp     //
USBH0M          Inp     //

// Group 760:  All pins in this group must be set to the same direction
//             These pins are bidirectional
USBH1P          Inp     //
USBH1M          Inp     //

// Group 770:  All pins in this group must be set to the same direction
//             These pins are bidirectional
USBDP           Inp     //
USBDM           Inp     //

// The following pins are complete bidirectional pins.
// The direction of each pin can be set independent of the other pins.
// Each pin can be used as an input.
GPIO200         Inp     // PCI_RSTO#
GPIO201         Inp     // PSC0_EXTCLK
GPIO202         Inp     // PSC1_EXTCLK
GPIO203         Inp     // PSC2_EXTCLK
GPIO204         Inp     // PSC3_EXTCLK
GPIO205         Inp     //
GPIO206         Inp     // PSC2_SYNC1
GPIO207         Inp     // PSC2_SYNC0
GPIO208         Inp     // PSC2_D1/DMA_REQ2
GPIO209         Inp     // PSC2_D0/DMA_REQ3
GPIO210         Inp     // PSC2_CLK
GPIO211         Inp     // PSC3_SYNC1
GPIO212         Inp     // PSC3_SYNC0
GPIO213         Inp     // PSC3_D1
GPIO214         Inp     // PSC3_D0
GPIO215         Inp     // PSC3_CLK
U0TXD           Inp     // GPIO20
U1TXD           Inp     // GPIO21
U1RTS#          Inp     // GPIO22
U3TXD           Inp     // GPIO23
N0TXEN          Inp     //
N0TXD0          Inp     //
N0TXD1          Inp     //
N0TXD2          Inp     //
N0TXD3          Inp     //
N0MDC           Inp     //
N0MDIO          Inp     //
```

```
N1TXEN          Inp     // GPIO24
N1TXD0          Inp     // GPIO25
N1TXD1          Inp     // GPIO26
N1TXD2          Inp     // GPIO27
N1TXD3          Inp     // GPIO28
N1MDC           Inp     //
N1MDIO          Inp     //
DDQ0            Inp     // SDRAM Data Bus
DDQ1            Inp     //
DDQ2            Inp     //
DDQ3            Inp     //
DDQ4            Inp     //
DDQ5            Inp     //
DDQ6            Inp     //
DDQ7            Inp     //
DDQ8            Inp     //
DDQ9            Inp     //
DDQ10           Inp     //
DDQ11           Inp     //
DDQ12           Inp     //
DDQ13           Inp     //
DDQ14           Inp     //
DDQ15           Inp     //
DDQ16           Inp     //
DDQ17           Inp     //
DDQ18           Inp     //
DDQ19           Inp     //
DDQ20           Inp     //
DDQ21           Inp     //
DDQ22           Inp     //
DDQ23           Inp     //
DDQ24           Inp     //
DDQ25           Inp     //
DDQ26           Inp     //
DDQ27           Inp     //
DDQ28           Inp     //
DDQ29           Inp     //
DDQ30           Inp     //
DDQ31           Inp     //
DDQS0           Inp     // DDR SDRAM Strobe
DDQS1           Inp     //
DDQS2           Inp     //
DDQS3           Inp     //
```

```
GPIO0           Inp     //
GPIO1           Inp     //
GPIO2           Inp     // EXTCLK0
GPIO3           Inp     // EXTCLK1
GPIO4           Inp     // DMA_REQ0
GPIO5           Inp     // DMA_REQ1
GPIO6           Inp     // SMROMCKE
GPIO7           Inp     //
GPIO8           Inp     //
GPIO9           Inp     // U3CTS#
GPIO10          Inp     // U3DSR#
GPIO11          Inp     // U3DCD#
GPIO12          Inp     // U3RI#
GPIO13          Inp     // U3RTS#
GPIO14          Inp     // U3DTR#
GPIO15          Inp     //
PIOW#           Out,Hi  // PCMCIA Write Cycle Indication
PIOR#           Out,Hi  // PCMCIA Read Cycle Indication
POE#            Out,Hi  // PCMCIA Output Enable
PREG#           Out,Hi  // PCMCIA register-only access signal
PCE1#           Out,Hi  // PCMCIA card enable
PCE2#           Out,Hi  // PCMCIA card enable
PWE#            Out,Hi  // PCMCIA write enable
RBEN0#          Out,Lo  // SRAM/IO/PCMCIA/FLASH/ROM/NAND Byte Enable
RBEN1#          Out,Lo  //
RBEN2#          Out,Lo  //
RBEN3#          Out,Lo  //
RWE#            Out,Hi  // SRAM/IO/PCMCIA/FLASH/ROM/NAND Write Enable
ROE#            Out,Hi  // SRAM/IO/PCMCIA/FLASH/ROM/NAND Output Enable
RCS0#           Out,Hi  // SRAM/IO/PCMCIA/FLASH/ROM/NAND Chip Select
RCS1#           Out,Hi  //
RCS2#           Out,Hi  //
RCS3#           Out,Hi  //
RALE            Out,Lo  // NAND Address Latch Enable
RCLE            Out,Lo  // NAND Command Latch Enable
PCI_DEVSEL#     Inp     // PCI device select
PCI_STOP#       Inp     // PCI target stop
PCI_FRAME#      Inp     // PCI frame
PCI_IRDY#       Inp     // PCI initiator ready
PCI_TRDY#       Inp     // PCI target ready
PCI_PAR         Inp     // PCI parity
PCI_GNT0#       Inp     // PCI arbiter bus grant output
PCI_GNT1#       Inp     // PCI arbiter bus grant output
```

```
PCI_GNT2#       Inp     // PCI arbiter bus grant output
PCI_GNT3#       Inp     // PCI arbiter bus grant output
PCI_PERR#       Inp     // PCI parity error
PCI_SERR#       Inp     // PCI system error
PCI_AD0         Inp     // PCI address/data
PCI_AD1         Inp     //
PCI_AD2         Inp     //
PCI_AD3         Inp     //
PCI_AD4         Inp     //
PCI_AD5         Inp     //
PCI_AD6         Inp     //
PCI_AD7         Inp     //
PCI_AD8         Inp     //
PCI_AD9         Inp     //
PCI_AD10        Inp     //
PCI_AD11        Inp     //
PCI_AD12        Inp     //
PCI_AD13        Inp     //
PCI_AD14        Inp     //
PCI_AD15        Inp     //
PCI_AD16        Inp     //
PCI_AD17        Inp     //
PCI_AD18        Inp     //
PCI_AD19        Inp     //
PCI_AD20        Inp     //
PCI_AD21        Inp     //
PCI_AD22        Inp     //
PCI_AD23        Inp     //
PCI_AD24        Inp     //
PCI_AD25        Inp     //
PCI_AD26        Inp     //
PCI_AD27        Inp     //
PCI_AD28        Inp     //
PCI_AD29        Inp     //
PCI_AD30        Inp     //
PCI_AD31        Inp     //
PCI_CBE0#       Inp     // PCI bus command and byte enable
PCI_CBE1#       Inp     //
PCI_CBE2#       Inp     //
PCI_CBE3#       Inp     //
PSC0_CLK        Inp     // PSC0 clock
PSC0_SYNC0      Inp     //
PSC0_SYNC1      Inp     // GPIO16
```

```
PSC0_D0         Inp     //
PSC0_D1         Inp     //
PSC1_CLK        Inp     // PSC0 clock
PSC1_SYNC0      Inp     //
PSC1_SYNC1      Inp     // GPIO17
PSC1_D0         Inp     //
PSC1_D1         Inp     //

// The following pins are output only pins.
// Setting to input (tristate) one of these pins results in an error.
RESETOUT#       Out,Hi  // Buffered Output of CPU Reset Input
DA0             Out,Lo  // SDRAM Address Output
DA1             Out,Lo  //
DA2             Out,Lo  //
DA3             Out,Lo  //
DA4             Out,Lo  //
DA5             Out,Lo  //
DA6             Out,Lo  //
DA7             Out,Lo  //
DA8             Out,Lo  //
DA9             Out,Lo  //
DA10            Out,Lo  //
DA11            Out,Lo  //
DA12            Out,Lo  //
DA13            Out,Lo  //
DBA0            Out,Lo  // SDRAM Bank Address
DBA1            Out,Lo  // SDRAM Bank Address
DCS0#           Out,Hi  // SDRAM Programmable Chip Select
DCS1#           Out,Hi  //
DCS2#           Out,Hi  //
DCK0            Out,Lo  //
DCK0#           Out,Lo  //
DCK1            Out,Lo  //
DCK1#           Out,Lo  //
DCKE            Out,Lo  // SDRAM Clock enable
DRAS#           Out,Hi  // SDRAM command Output
DCAS#           Out,Hi  // SDRAM command Output
DWE#            Out,Hi  // SDRAM command Output
DDM0            Out,Hi  // SDRAM Byte Mask
DDM1            Out,Hi  // SDRAM Byte Mask
DDM2            Out,Hi  // SDRAM Byte Mask
DDM3            Out,Hi  // SDRAM Byte Mask
RCLK            Out,Lo  // Bus clock output for synchrous mode
```

```
PCI_CLKO        Out,Lo  // PCI clock output

// The following pins are input only.
// Setting to output of one of these pins results in an error.
// Declaration of the direction of these pins is optional.
DRVSEL          Inp     // SDRAM drive strength select
U0RXD           Inp     // UART0 receive
U1RXD           Inp     // UART1 receive
U1CTS#          Inp     // UART1 CTS
U3RXD           Inp     // UART3 receive
PCI_LOCK#       Inp     // PCI lock fopr atomic operations
PCI_CLK         Inp     // PCI input clock
PCI_REQ0#       Inp     // PCI arbiter bus request input
PCI_REQ1#       Inp     // PCI arbiter bus request input
PCI_REQ2#       Inp     // PCI arbiter bus request input
PCI_REQ3#       Inp     // PCI arbiter bus request input
PCI_CFG         Inp     // PCI configuration, 0->sattelite mode, 1->host mode
PCI_IDSEL       Inp     // PCI initialization device select input
PCI_INTA#       Inp     // PCI interrupt input
PCI_INTB#       Inp     // PCI interrupt input
PCI_INTC#       Inp     // PCI interrupt input
PCI_INTD#       Inp     // PCI interrupt input
PCI_RST#        Inp     // PCI reset input
N0CRS           Inp     //
N0RXD0          Inp     //
N0RXD1          Inp     //
N0RXD2          Inp     //
N0RXD3          Inp     //
N0RXCLK         Inp     //
N0RXDV          Inp     //
N0COL           Inp     //
N0TXCLK         Inp     //
N1CRS           Inp     //
N1RXD0          Inp     //
N1RXD1          Inp     //
N1RXD2          Inp     //
N1RXD3          Inp     //
N1RXCLK         Inp     //
N1RXDV          Inp     //
N1COL           Inp     //
N1TXCLK         Inp     //
VDDXOK          Inp     // VDDX stable
WAKE#           Inp     // Wake from hibernate request
```

```
FWTOY#          Inp     // Firewall the TOY clock signal
BOOT0           Inp     // Boot Memory Select
BOOT1           Inp     //
BOOT2           Inp     //
RESETIN#        Inp     // CPU Reset input
TESTEN          Inp     // Test enable, should be pulled low
TC0             Inp     // Test Clock Input, should be pulled low
TC1             Inp     // Test Clock Input, should be pulled low
TC2             Inp     // Test Clock Input, should be pulled low
TC3             Inp     // Test Clock Input, should be pulled low
PIOS16#         Inp     // PCMCIA I/O is 16 Bit
PWAIT#          Inp     // PCMCIA extend Cycle
EWAIT#          Inp     // SRAM/IO/Flash/ROM Extend Cycle
R/B#            Inp     // NAND Ready/not Busy
```

1.12. Supported flash devices

Type JTAuxxxx /LIST [optionlist]

to get a online list of all flash types which could be used with the /DEVICE= option.

See separate file JTAG_V4xx_FLASHES.pdf to get a complete list of supported flash types.

2. JTAuxxxx Parameter Description

When you start JTAuxxxx.EXE without any parameters the following help screen with all possible functions and options is displayed:

```
JTAuxxxx --- JTAG utility for AMD Alchemy Solutions Processors
Copyright © FS FORTH-SYSTEME GmbH, Breisach
Version 4.xx of mm/dd/yyyy

Programming of Flash-EPROMs and hardware tests on targets with the
AMD Alchemy Solutions Processors.

The JTAG-Booster is needed to connect the parallel port of the PC
to the JTAG port of the AMD Alchemy Solutions Processors.

Usage: JTAuxxxx /function [filename] [/option_1] ... [/option_n]
Supported functions:
        /P          : Program a Flash Device
        /R          : Read a Flash Device to file
        /V          : Verify a Flash Device with file
        /DUMP       : Make a target dump
        /PSER       : Program a I2C/SPI/MicroWire Device with file
        /RSER       : Read a I2C/SPI/MicroWire Device to file
        /VSER       : Verify a I2C/SPI/MicroWire Device with file
        /DUMPSER    : Make a dump of a I2C/SPI/MicroWire Device
        /BLINK      : Toggle a CPU pin
        /PIN?       : Test a CPU pin
        /SAMPLE     : Test a CPU pin while the CPU is running
        /SNAP       : Test all CPU pins while CPU is running
        /LIST       : Print a list of supported Flash devices
```

Supported Options:

/CS0	/CS1	/CS2	/CS3	/BIG
/NOCS	/NOWRSETUP	/TOP	/BYTE-MODE	/BM
/CFI	/CFIDEBUG	/PAUSE	/P	/NODUMP
/NOERASE	/ERASEALL	/LATTICE	/LPT1	/LPT2
/LPT3	/LPT-BASE=	/32BIT	/16BIT	/8BIT
/NOMAN	/LENGTH=	/L=	/FILE-OFFSET=	/FO=
/OFFSET=	/O=	/DELAY=	/DEVICE-BASE=	/DB=
/DRIVER=	/IROFFS=	/CPUPOS=	/DEVICE=	/PIN=
/SERCS=	/SERCLK=	/SERDAT=	/SERDATI=	/SERDATO=
/SERBUFF=	/SERBIG	/SPI	/MWIRE	/LSB1ST
/SPIERA	/WATCH=	/OUT=	/INI=	/REP

The following options are valid for most functions:

/BIG
This option switches the byte ordering to big endian mode. This option must fit to the target's endianess
Default: Little Endian

/DRIVER=x with x = 1,2,3,4
A driver for the interface to the JTAG-BOOSTER on the parallel port may be specified. /DRIVER=1 selects the fastest available driver, /DRIVER=4 selects the slowest one. Use a slower driver if there are problems with JTAG-BOOSTER.
Default: /DRIVER=3

/INI=file
An initialization file may be specified. By default the current directory is searched for the file JTAuxxxx.INI. If this file is not found and no initialization file is specified in the command line, default initialization values are used (see also chapter 1.11 “Initialization file JTAuxxxx.INI”).
Note: The initialization file is not loaded for the functions /SAMPLE (chapter 2.11) and /SNAP (chapter 2.12).
Default: /INI=JTAuxxxx.INI

/LATTICE
For demonstration purposes this software works with the Lattice ispLSI-Adapter, too. With the option /LATTICE you can simulate the speed achievable with the simple ispLSI-Adapter.

/LPT1 /LPT2 /LPT3
A printer port may be specified where the JTAG-Booster resides. If you are using this program with WinNT, Win2000 or WinXP you must specify /LPT2 or /LPT-BASE=378 to get access to the standard printer port.
Default: /LPT1

/LPT-BASE
The physical I/O-Address of printer port may be specified instead of the logical printer name. Useful option, if you work with WinNT, Win2000 or WinXP, because the standard printer port is mapped as LPT2 here. Use the option /LPT-BASE=378 to get a command line which works independent of the operating system.

/OUT=file_or_device
All screen outputs are redirected to the specified file or device. Note that you can't redirect to the same parallel port where the JTAG-Booster resides.
Default: /OUT=CON

/PAUSE
With the option /PAUSE you can force the program to stop after each screen. Please do not use this option if you redirect the output to a file.
Abbreviation: /P

/WATCH=
With the option /WATCH= a pin can be specified, which is toggled twice per second, while the program is active. This pin may be the trigger of a watchdog. This pin must be specified as output in the initialization file.

/IROFFS=
Specifies the position of the AMD Alchemy Solutions Processors instruction register within the JTAG chain. In most cases this option is not needed.
Default: /IROFFS=0

/CPUPOS=
Specifies the position of the AMD Alchemy Solutions Processors within the JTAG chain.
Default: /CPUPOS=0

2.1. Program a Flash Device

Usage: JTAuxxxx /P filename [optionlist]

The specified file is programmed into the flash memory. The flash status is polled after programming of each cell (cell=8, 16 or 32 bit, depending on current data bus width). In case of a programming error, the contents of the flash memory is written to a file with the extension DMP.

If you want a complete verify after programming, please use an additional command line with the verify function. See chapter 2.3 “Verify a Flash Device with file”. In most cases this additional verify step is not needed.

The type of the flash device is normally detected by the software. When autodetection fails you should use the /DEVICE= option together with /8BIT or /16BIT or /32BIT to set the right flash device and configuration. The known flash devices are shown in chapter 1.12 “Supported flash devices”. Use the option /CFI if the flash is not in the list of know devices.

Options:

/DEVICE=devicename
The flash device is detected automatically by switching to autoselect mode. In case of trouble you should select the flash device by using this parameter to avoid autodetection. Combine this option with one of the following options which specify the data bus width and the option /BYTE-MODE if applicable.

/CFI
To be prepared for future flash chips, the JTAG-Booster integrates support for flashes which contain the CFI (Common Flash Interface) information structure. The CFI support is activated by simply adding the option /CFI to the command line. The JTAG-Booster then automatically searches in all available bus widths for all possible flash types and configurations instead of searching for the JEDEC identification code.
In case of an error add the command line option /CFIDEBUG and redirect the program output into a file. Sending us this file helps in solving problems.

/8BIT /16BIT /32BIT
Specifies the data bus width to the target flash device. You can speed up autodetection, if you specify the correct data bus size. You need this option together with the option /DEVICE= to explicit specify a specific flash configuration.

/BYTE-MODE
If there is a flash device connected to the CPU which does have a byte mode pin (8 bit and 16/32 bit bus mode), you can force it to be used as 8 bit mode with the option /BYTE-MODE. In most cases this option will not be needed.
Abbreviation: /BM

/NOMAN
If you use a flash device which is identical to one of the supported parts, but is from a different manufacturer, with this option you can suppress the comparison of the manufacturer identification code. We recommend to use this option together with the /DEVICE= option to avoid failures in autodetection.

/DEVICE-BASE=hhhhhh[1]
Here you can specify a flash device starting address. In most cases, where the flash device is selected with one of the CPUs chip select pins, this parameter is not needed. But if there is any decoding logic in your hardware, this option will be needed. Especially, if there are several flash banks connected to one chip select and a sub decoding logic generates chip selects for these flash banks, this option can be used to select a specific flash bank.
Default: /DEVICE-BASE=0
Abbreviation: /DB=

[1]hhhhhh=number base is hex

/OFFSET=hhhhhh
The programming starts at an offset of hhhhhh relative to the start address of the flash device. If the offset is negative, the offset specifies an address relative to the end of the flash device. See also option /TOP
Default: /OFFSET=0
Abbreviation: /O=

/TOP
If the option /TOP is used the option /OFFSET= specifies the address where the programming ends (plus one) instead of the starting address. This option is very important for Intel CPU architectures, because target execution always starts at the top of the address space.

/FILE-OFFSET=hhhhhh
If FILE-OFFSET is specified, the first hhhhhh bytes of the file are skipped and not programmed to target.
Default: /FILE-OFFSET=0
Abbreviation: /FO=

/LENGTH=hhhhhh
The number of programmed bytes may be limited to LENGTH. If no LENGTH is specified the whole file is programmed.
Default: /LENGTH=4000000 (64 MByte)
Abbreviation: /L=

/NODUMP
In case of a verify error the contents of the flash memory is written to a file with the extension .DMP. With /NODUMP you can suppress this feature.

/ERASEALL
Erase the whole flash device. If this option isn't set, only those blocks are erased where new data should be written to.

/NOERASE
This option prevents the flash device from being erased.

/CS0 /CS1 /CS2 /CS3
This options may be used to specify one or more chip select signals to the flash memory. The used chip selects must be defined as output and inactive in the initialization file. (See chapter 1.11 “Initialization file JTAuxxxx.INI”.)
Default: /CS0

/NOCS
Use this option to switch off all chip select signals. This may be necessary if the device's chip select is generated via a normal decoder instead of using the AMD Alchemy Solutions Processors chip select unit.

/NOWRSETUP
By default write cycles to the Flash EPROM are realized with three steps: 1. set address/data 2. write strobe active 3. write strobe inactive. **In most cases** it is possible to set the write strobe coincident with setting of address and data by specifying the option /NOWRSETUP. **This increases the programming speed by 50%.**

Examples:

JTAuxxxx /P ROMDOS.ROM /L=20000 /TOP
This example programs up to 128 Kbytes of the file ROMDOS.ROM (with i.e. 512 Kbytes) to the top of the boot flash memory.

JTAuxxxx /P CE.ROM /32BIT /CS1
This example programs the file CE.ROM to the 32 Bit Flash-EPROM connected to RCS1#.

2.2. Read a Flash Device to file

Usage: JTAuxxxx /R filename [optionlist]

The contents of a flash device is read and written to a file.

The type of the flash device is normally detected by the software. When autodetection fails you should use the /DEVICE= option together with /8BIT or /16BIT or /32BIT to set the right flash device and configuration. The known devices are shown in chapter 1.12 “Supported flash devices”. Use the option /CFI if the flash is not in the list of know devices.

Options:

/DEVICE=devicename
See function /P (Chapter 2.1)

/CFI
See function /P (Chapter 2.1)

/8BIT /16BIT /32BIT
See function /P (Chapter 2.1)

/BYTE-MODE
See function /P (Chapter 2.1)

/NOMAN
See function /P (Chapter 2.1)

/DEVICE-BASE=hhhhhh[2]
See function /P (Chapter 2.1)

[2]hhhhhh=number base is hex

/OFFSET=hhhhhh
Reading of the flash memory starts at an offset of hhhhhh relative to the start address of the flash device. If the offset is negative, the offset specifies a address relative to the end of the flash device.
See also option /TOP.
Default: /OFFSET=0
Abbreviation: /O=

/TOP
If the option /TOP is used the option /OFFSET= specifies the address where reading ends (plus one) instead of the starting address.

/LENGTH=hhhhhh
The number of read bytes may be limited to LENGTH. If no LENGTH is specified the whole flash device is read (if no offset is specified).

/CS0 /CS1 /CS2 /CS3
See function /P (Chapter 2.1)

/NOWRSETUP
See function /P (Chapter 2.1)
Please note: In the function /R write cycles are needed to detect the type of the flash memory.

Example:

JTAuxxxx /R BIOS.ABS /L=10000 /TOP
This example may be used to read the upper most 64 Kbyte of the flash memory to the file BIOS.ABS.

2.3. Verify a Flash Device with file

Usage: JTAuxxxx /V filename [optionlist]

The contents of a flash device is compared with the specified file. If there are differences the memory is dumped to a file with the extension DMP.

The type of flash device is normally detected by the software. When autodetect fails you should use the /DEVICE= option together with /8BIT or /16BIT or /32BIT to set the right flash device and configuration. The known devices are shown in chapter 1.12 “Supported flash devices”. Use the option /CFI if the flash is not in the list of know devices.

Options:

/DEVICE=devicename
See function /P (Chapter 2.1)

/CFI
See function /P (Chapter 2.1)

/8BIT /16BIT /32BIT
See function /P (Chapter 2.1)

/BYTE-MODE
See function /P (Chapter 2.1)

/NOMAN
See function /P (Chapter 2.1)

/DEVICE-BASE=hhhhhh
See function /P (Chapter 2.1)

/OFFSET=hhhhhh
See function /P (Chapter 2.1)

/TOP
See function /P (Chapter 2.1)

/FILE-OFFSET=hhhhhh
See function /P (Chapter 2.1)

/LENGTH=hhhhhh
See function /P (Chapter 2.1)

/NODUMP
See function /P (Chapter 2.1)

/CS0 /CS1 /CS2 /CS3
See function /P (Chapter 2.1)

/NOWRSETUP
See function /P (Chapter 2.1)
Please note: In the function /V write cycles are needed to detect the type of the flash memory.

Example:

JTAuxxxx /V ROMDOS.ROM /L=20000 /TOP
This example may be used to verify the upper most 128 Kbytes of the flash memory with the file ROMDOS.ROM (with i.e. 512 Kbytes).

2.4. Dump target memory

Usage: JTAuxxxx /DUMP [optionlist]

A Hex-Dump of the target memory is printed on the screen, if not redirected to file or device.

Options:

/8BIT /16BIT /32BIT
Default: /32BIT

/OFFSET=hhhhhh
The memory dump starts at an offset of hhhhhh plus the device start address (see option /DEVICE-BASE=).
Default: /OFFSET=0
Abbreviation: /O=

/DEVICE-BASE=hhhhhh[3]
The device start address is used as an additional offset. This gives the function /DUMP the same behavior as function /P /V and /R.
Default: /DEVICE-BASE=0
Abbreviation: /DB=

/TOP
If the option /TOP is used the option /OFFSET= specifies the address where the dump ends (plus one) instead of the starting address

/LENGTH=hhhhhh
Default: /LENGTH=100
Abbreviation: /L=

/CS0 /CS1 /CS2 /CS3
See function /P (Chapter 2.1)
Default: /CS0

[3]hhhhhh=number base is hex

Example:

JTAuxxxx /DUMP
This example makes a memory dump of the first 256 bytes of the Boot-EPROM.

2.5. Program a Serial Device (I²C/SPI/MicroWire)

Usage: JTAuxxxx /PSER filename [/SERBIG] [optionlist]

The specified file is programmed to a serial device (i.e. EEPROM) connected to pins of the CPU. Finally a complete verify is done. If the verify fails, the contents of the serial device is written to a file with the extension DMP.

For an I²C device there are two different methods how to connect it to the CPU. The first method uses two CPU pins, one pin for clock output (SERCLK) and one pin for serial data input/output (SERDAT). The second method uses one pin for clock output (SERCLK), one for serial data input (SERDATI) and one for serial data output (SERDATO).

Connecting a SPI/MicroWire device needs four different CPU pins: SERCS is the chip select output of the CPU, SERCLK is the clock output of the CPU, SERDATO is the serial data output of the CPU and must be connected to the SI input at the SPI/MicroWire device and SERDATI is the serial data input to the CPU and must be connected to the SO output of the SPI/MicroWire device.

Options:

/SERBIG
Specify this option if there is a device which needs a three byte address instead of a two byte address. For SPI devices this option is normally needed for devices with more than or equal to 64 kBytes. For I²C devices this option is normally needed for devices with more than 2 kByte.
This option must be the first option after the filename.

/SPI
Specify this option, if there is a SPI device connected instead of an I²C device.

/MWIRE
Specify this option, if there is a MicroWire device connected instead of an I²C device.
Please Note: Actually only the M93C06 and the M93C46 and only in 16 Bit mode are supported.

/DEVICE-BASE=hhhhhh
This option specifies an I²C device starting address. The default values are chosen to access a serial EEPROM. By changing the device starting address different devices can be selected. As SPI/MicroWire devices are selected by the chip select signal instead of an address, this option does not make sense for SPI/MicroWire devices.
Default: /DEVICE-BASE=5000 (if option /SERBIG omitted)
Default: /DEVICE-BASE=500000 (if option /SERBIG specified)
Default: /DEVICE-BASE=0 (for SPI/MicroWire devices)

/OFFSET=hhhhhh
The programming starts at an offset of hhhhhh relative to the start address of the serial device.
Default: /OFFSET=0
Abbreviation: /O=

/FILE-OFFSET=hhhhhh
If FILE-OFFSET is specified, the first hhhhhh bytes of the file are skipped and not programmed to target.
Default: /FILE-OFFSET=0
Abbreviation: /FO=

/LENGTH=hhhhhh
The number of programmed bytes may be limited to LENGTH. If no LENGTH is specified the whole file is programmed.
Abbreviation: /L=

/NODUMP
In case of a verify error the contents of the I²C-Device is written to a file with the extension .DMP. With option /NODUMP you can suppress this feature.

/SERCS=pin_name (SPI/MicroWire mode only)
Specifies the CPU pin used to select the serial device.
In SPI mode SERCS is treated as low active.
In MicroWire mode SERCS is treated as high active.

/SERCLK=pin_name
Specifies the CPU pin used for serial clock output.

/SERDAT=pin_name (I²C only)
Specifies the CPU pin used for serial data input and output for an I²C device. Pin_name must specify a bidirectional pin otherwise an error message occurs. Instead of one bidirectional pin one pin for serial data input and one for serial data output may be used. See option /SERDATO= and /SERDATI= .

/SERDATO=pin_name
Specifies the CPU pin used for serial data output. Pin_name must specify a output pin otherwise an error message occurs. This pin must be connected to the serial data **input** of a SPI/MicroWire device.

/SERDATI=pin_name
Specifies the CPU pin used for serial data input. Pin_name must specify a input pin otherwise an error message occurs. This pin must be connected to the serial data **output** of a SPI/MicroWire device.

/SERBUFF= hhhhhh
For I²C and SPI devices the write page mode can be activated by specifying the option /SERBUFF=. Using this feature increases the programming performance. Please note: Some SPI devices do not support single byte write mode. For these devices the option /SERBUFF= must be specified in the command line.

/LSB1ST
Some devices need the least significant data bit sent/received first (i.e. Altera ECS1 configuration device for FPGAs). Addresses are still sent/received most significant bit first. This option does not affect the behavior of accessing I²C devices.

/SPIERA
Some SPI devices need to be erased before programming. Add option /SPIERA to the command line to perform a chip erase procedure before programming.

Example:

JTAuxxxx /PSER EEPROM.CFG /SERCLK=GPIO0 /SERDAT=GPIO
This example loads the file EEPROM.CFG to a I²C EEPROM connected to the pins GPIO0 and GPIO1 of the AMD Alchemy Solutions Processors

2.6. Read a Serial Device to file (I²C/SPI/MicroWire)

Usage: JTAuxxxx /RSER filename [/SERBIG] /L=hhhhhh [optionlist]

The contents of a serial device (i.e. EEPROM) is read and written to a file. The option /LENGTH= must be specified.

Options:

/SERBIG
This option must be the first option after the filename.
See function /PSER (Chapter 2.5)

/SPI
Specify this option, if there is a SPI device connected instead of a I²C device.

/MWIRE
Specify this option, if there is a MicroWire device connected instead of an I²C device.

/DEVICE-BASE=hhhhhh
See function /PSER (Chapter 2.5)

/OFFSET=hhhhhh
Reading of the serial device starts at an offset of hhhhhh relative to the start address of the serial device.
Default: /OFFSET=0
Abbreviation: /O=

/LENGTH=hhhhhh
The number of read bytes must be specified otherwise an error message occurs.
Abbreviation: /L=

/SERCS=pin_name
See function /PSER (Chapter 2.5)

/SERCLK=pin_name
See function /PSER (Chapter 2.5)

/SERDAT=pin_name
See function /PSER (Chapter 2.5)

/SERDATO=pin_name
See function /PSER (Chapter 2.5)

/SERDATI=pin_name
See function /PSER (Chapter 2.5)

/LSB1ST
See function /PSER (Chapter 2.5)

Example:

JTAuxxxx /RSER EEPROM.CFG /SERCLK=GPIO0 /SERDAT=GPIO1 /L=100
This example reads 256 bytes from a I²C EEPROM to the file EEPROM.CFG. The serial EEPROM is connected to the pins CPIO0 and GPIO1 of the AMD Alchemy Solutions Processors.

2.7. Verify a Serial Device with file (I²C/SPI/MicroWire)

Usage: JTAuxxxx /VSER filename [/SERBIG] [optionlist]

The contents of a serial device (i.e. EEPROM) is compared with the specified file. If there are differences the contents of the I²C -Device is written to a file with the extension DMP.

Options:

/SERBIG
This option must be the first option after the filename.
See function /PSER (Chapter 2.5)

/SPI
Specify this option, if there is a SPI device connected instead of a I²C device.

/MWIRE
Specify this option, if there is a MicroWire device connected instead of an I²C device.

/DEVICE-BASE=hhhhhh
See function /PSER (Chapter 2.5)

/OFFSET=hhhhhh
See function /PSER (Chapter 2.5)

/FILE-OFFSET=hhhhhh
See function /PSER (Chapter 2.5)

/LENGTH=hhhhhh
See function /PSER (Chapter 2.5)

/NODUMP
See function /PSER (Chapter 2.5)

/SERCS=pin_name
See function /PSER (Chapter 2.5)

/SERCLK=pin_name
See function /PSER (Chapter 2.5)

/SERDAT=pin_name
See function /PSER (Chapter 2.5)

/SERDATO=pin_name
See function /PSER (Chapter 2.5)

/SERDATI=pin_name
See function /PSER (Chapter 2.5)

/LSB1ST
See function /PSER (Chapter 2.5)

Example:

JTAuxxxx /VSER EEPROM.CFG /SERCLK=GPIO0 /SERDAT=GPIO1
This example verifies 256 bytes from a serial EEPROM with the file EEPROM.CFG. The serial EEPROM is connected to the pins GPIO0 and GPIO1 of the AMD Alchemy Solutions Processors.

2.8. Dump a Serial Device (I²C/SPI/MicroWire)

Usage: JTAuxxxx /DUMPSER [/SERBIG] [optionlist]

A Hex-Dump of serial device (i.e. EEPROM) is printed on the screen, if not redirected to file or device.

Options:

/SERBIG
This option must be the first option.
See function /PSER (Chapter 2.5)

/SPI
Specify this option, if there is a SPI device connected instead of a I²C device.

/MWIRE
Specify this option, if there is a MicroWire device connected instead of an I²C device.

/DEVICE-BASE=hhhhhh
See function /PSER (Chapter 2.5)

/OFFSET=hhhhhh[4]
The memory dump starts at an offset of hhhhhh.
Default: /OFFSET=0
Abbreviation: /O=

/LENGTH=hhhhhh
Default: /LENGTH=100
Abbreviation: /L=

/SERCS=pin_name
See function /PSER (Chapter 2.5)

/SERCLK=pin_name
See function /PSER (Chapter 2.5)

[4]hhhhhh=number base is hex

/SERDAT=pin_name
See function /PSER (Chapter 2.5)

/SERDATO=pin_name
See function /PSER (Chapter 2.5)

/SERDATI=pin_name
See function /PSER (Chapter 2.5)

/LSB1ST
See function /PSER (Chapter 2.5)

Example:

JTAuxxxx /DUMPSER /SERCLK=GPIO0 /SERDAT=GPIO1
This example makes a memory dump of the first 100h bytes of a I²C EEPROM connected to the CPU.

2.9. Toggle CPU pins

Usage: JTAuxxxx /BLINK /PIN=pinname [optionlist]

This command allows to test the hardware by blinking with LEDs or toggling CPU signals. Faster signals can be generated by setting the delay option to zero. This can be a very helpful feature to watch signals on an oscilloscope.

The signal on the defined pin has an duty cycle of 1/2: The level is 67% high and 33% low.

Please Note: Not every pin of the AMD Alchemy Solutions Processors may be specified as an output pin.

Options:

/PIN=pin_name
CPU pin to toggle. If the option /PIN= is not specified an error message occurs. Most pins of the list in chapter 1.11 “Initialization file JTAuxxxx.INI” can be used. If you type /PIN= without any pin declaration a list of the CPU pins is displayed.

/DELAY=dddddd[5]
Time to wait to next change of signal. This option can be adjusted to get optimum signals for measures with the oscilloscope.
Default: /DELAY=10000

Example:

JTAuxxxx /BLINK /PIN=FLAG3 /DELAY=0
This example toggles the FLAG3 pin very fast which can be followed by the use of an oscilloscope.

[5]dddddd=number base is decimal

2.10. Polling CPU pins

Usage: JTAuxxxx /PIN? /PIN=pinname [optionlist]

This command allows to test the hardware by polling CPU signals.

Please Note: Not every pin of the AMD Alchemy Solutions Processors may be specified as an input pin.

Options:

/PIN=pin_name
CPU pin to poll. If the option /PIN= is not specified an error message occurs. Most pins of the list in chapter 1.11 “Initialization file JTAuxxxx.INI” can be used. If you type /PIN= without any pin declaration a list of the CPU pins is displayed.

Example:

JTAuxxxx /PIN? /PIN=RESET#
This example samples the reset pin of the AMD Alchemy Solutions Processors.

2.11. Polling CPU pins while the CPU is running

Usage: JTAuxxxx /SAMPLE /PIN=pinname [optionlist]

This command is similar to the function /PIN?. But with this function any pin can be observed, independent of the pin direction. Furthermore the CPU remains in normal operation.

Options:

/PIN=pin_name
CPU pin to poll. If the option /PIN= is not specified an error message occurs. All pins of the list in chapter 1.11 “Initialization file JTAuxxxx.INI” can be used. If you type /PIN= without any pin declaration a list of the CPU pins is displayed.

Example:

JTAuxxxx /SAMPLE /PIN=FLAG3
This example samples the state of the port pin FLAG3 while the AMD Alchemy Solutions Processors is running.

2.12. Show status of all CPU pins while the CPU is running

Usage: JTAuxxxx /SNAP [optionlist]

This function is similar to the function /SAMPLE, but displays the status of all CPU pins on the screen. The CPU remains in normal operation.

The behavior of the function /SNAP depends on the option /REP: With this option specified, the JTAG-Booster samples and displays the state of the CPU pins repetitive. Without this option the status of the pins is displayed only once.

Options:

/PAUSE
Use this option to stop the output after each displayed screen. Don't use this option together with the option /REP or if the output is redirected to a file.
Abbreviation /P

/REP
If this option is specified the status of the pins is sampled and displayed repetitive. In case of many signals the display is separated into several screens. Therefor we recommend to use a video mode with 43 or 50 lines. Use the '+' and the '-' key to switch between different screens. Any other key terminates the program.

Sample output:
This is a sample output for a AMD Alchemy Solutions Au1100 Processor

0 GPIO10	0 USBH1P	0 USBH1M	0 U0RXD
0 S0DIN	0 USBDP	0 USBDM	0 GPIO11
0 GPIO12	0 S0DOUT	0 S0CLK	0 S0DEN
0 U1RXD	0 IRDARX	0 IRDATX	0 I2SDIO
0 U3RXD	0 ACDI	0 ACRST#	0 ACSYNC
0 ACBCLK	0 ACDO	0 I2SCLK	0 I2SWORD
0 N0CRS	0 N0RXD0	0 U0TXD	0 U1TXD
0 U3TXD	0 GPIO13	0 N0RXD1	0 N0RXD2
0 GPIO14	0 GPIO15	0 N0RXD3	0 N0RXCLK
0 N0TXEN	0 N0TXD0	0 N0RXDV	0 N0COL
0 N0TXD1	0 N0TXD2	0 N0TXD3	0 N0MDC
0 N0MDIO	0 N0TXCLK	0 RESVD_0	0 RESVD_1
0 GPIO16	0 GPIO17	0 GPIO18	0 GPIO19
0 RESVD_2	0 GPIO20	0 GPIO21	0 GPIO22
0 GPIO23	0 RESVD_3	0 LCD_PWM0	0 RESVD_4
0 RESVD_5	0 LCD_PWM1	0 SDMS_MS_EN	0 SDMS0_CLK
0 SDMS0_CMD	0 SDMS0_DAT0	0 SDMS0_DAT1	0 SDMS0_DAT2
0 SDMS0_DAT3	0 SDMS1_CMD	0 SDMS1_CLK	0 SDMS1_DAT0
0 SDMS1_DAT1	0 SDMS1_DAT2	0 SDMS1_DAT3	0 SDD0
0 SDD1	0 SDD2	1 SDD3	1 SDD4
1 SDD5	1 SDD6	1 SDD7	1 SDD8
0 SDD9	1 SDD10	0 SDD11	1 SDD12
0 SDD13	1 SDD14	0 SDD15	1 SDD16
0 SDD17	1 SDD18	1 SDD19	1 SDD20
1 SDD21	1 SDD22	1 SDD23	1 SDD24
1 SDD25	1 SDD26	1 SDD27	1 SDD28
1 SDD29	1 SDCLK0	1 SDD30	1 SDD31
1 SDCLK1	1 SDCS0#	1 SDCS1#	1 SDCS2#
1 SDWE#	1 SDCLK2	1 SDRAS#	1 SDCKE
1 SDCAS#	1 SDQM0#	1 SDQM1#	1 SDQM2#
1 SDQM3#	1 SDBA0	1 SDBA1	1 SDA0
1 SDA1	1 SDA2	1 SDA3	1 SDA4
1 SDA5	1 SDA6	1 SDA7	1 SDA8
1 SDA9	1 SDA10	1 SDA11	1 SDA12
1 RESETOUT#	1 VSEL	1 TESTEN	1 ROMSEL
1 ROMSIZE	0 RESETIN#	0 VDDXOK	0 GPIO0
0 GPIO1	0 TC0	0 TC1	0 GPIO2
0 GPIO3	0 GPIO4	0 GPIO5	0 TC2
0 TC3	0 PIOW#	0 GPIO6	0 GPIO7

1 PIOR#	0 PIOS16#	0 PWE#	1 POE#
0 PREG#	0 PWAIT#	0 PCE1#	0 PCE2#
0 LRD0#	0 LRD1#	1 LWR0#	0 LWAIT#
0 EWAIT#	1 LCLK	0 LWR1#	1 RD0
1 RD1	0 RD2	1 RD3	1 RD4
1 RD5	1 RD6	1 RD7	1 RD8
1 RD9	1 RD10	1 RD11	1 RD12
1 RD13	1 RD14	1 RD15	1 RD16
1 RD17	0 RD18	0 RD19	0 RD20
0 RD21	0 RD22	0 RD23	1 RD24
0 RD25	0 RD26	0 RD27	0 RD28
0 RD29	1 RD30	1 RD31	1 RAD0
1 RAD1	1 RAD2	1 RAD3	1 RAD4
1 RAD5	1 RAD6	1 RAD7	1 RAD8
1 RAD9	1 RAD10	1 RAD11	1 RAD12
1 RAD13	1 RAD14	1 RAD15	1 RAD16
1 RAD17	1 RAD18	1 RAD19	1 RAD20
1 RAD21	1 RAD22	1 RAD23	1 RAD24
1 RAD25	1 RAD26	1 RAD27	1 RAD28
1 RAD29	1 RAD30	1 RAD31	0 RBEN0#
0 RBEN1#	0 RBEN2#	0 RBEN3#	1 RWE#
1 ROE#	1 RCS0#	1 RCS1#	1 RCS2#
1 RCS3#	1 LCD_PCK	1 LCD_D0	1 LCD_D1
1 LCD_D2	1 LCD_LCK	1 LCD_D3	1 LCD_D4
1 LCD_D5	1 LCD_D6	1 LCD_FCK	1 LCD_D7
0 LCD_D8	1 LCD_D9	0 LCD_D10	1 LCD_D11
0 LCD_D12	0 LCD_D13	1 LCD_D14	0 LCD_D15
1 LCD_BIAS	1 LCD_LEND	1 GPIO8	1 GPIO9

3. Implementation Information

This chapter summarizes some information about the implementation of the JTAG-Booster and describes some restrictions.

- The JTAG-Booster currently uses Boundary Scan to perform Flash programming. EJTAG is not used.

- The software assumes the following scheme for connecting the Flash-EPROM to the AMD Alchemy Solutions Processors. Please contact us, if you need support for a different method.

AMD Au1x00 signal	8 Bit Flash	16 Bit Flash	32 Bit Flash
RCS0# RCS1# RCS2# RCS3#	CS#	CS#	CS#
ROE#	OE#	OE#	OE#
RWE#	WE#	WE#	WE#
RAD0	A0	-	-
RAD1	A1	A1	-
RAD2..28	A2..28	A2..28	A2..28
RD0..7	D0..7	-	-
RD0..15	-	D0..15	-
RD0..31	-	-	D0..31

1.) All other signals are hold static during flash programming. The state of these signals is defined in the Initialization file.

- Default endianess is "Little Endian". Use the option /BIG for big endian support.

- Physical address range for static memory and peripherals is limited to 512MByte (MIPS32 specific). For memory accesses with boundary scan, the address lines RAD29, RAD30 and RAD31 are not modified.

- Support for NAND flashes is not included as standard functionality, but can be provided on customers request (AMD Alchemy Solutions Au1550 Processor only).

4. Converter Program HEX2BIN.EXE

Since the JTAG-Booster software is not able to handle Intel-HEX or Motorola S-Record files, an separate converter tool is delivered with this product package.

Five types of HEX formats can be converted to BIN file:

- I : INTEL HEX format (BYTE oriented)
- D : Digital Research
- M : MOTOROLA S HEX format (BYTE oriented)
- T : TEKTRONICS HEX format (BYTE oriented)
- H : Intel HEX-32

Maximum conversion size is 256 kBytes. A 4th parameter for starting address can be specified to skip out the leading garbage and you will maintain a small size of output binary file.

If you start the HEX2BIN without any additional parameter all necessary parameters will be asked for in a prompt mode:

```
HEX2BIN
Input HEX file name: MYAPP.H86
Output BIN file name[MYAPP.BIN]:
HEX file format
<I>ntel  /<M>otorola  /<D>igital Research  /<T>ektronics  /[H] Intel HEX-32[I] : H
Input CODE segment start address[0000000]: 10000
Input CODE segment end address[FFFFFFF]:
Unused bytes will be <1>00  <2>FF  [1]  : 2
```

Instead of using the prompt mode, you can directly specify all necessary parameters in the command line. This is essential for making batch files:

```
HEX2BIN  MYAPP.H86  MYAPP.BIN  H  0010000  FFFFFFF  2
```

It is very important to fill unused bytes with 0xFF, because this are simply skipped by the JTAG-Boosters software and so it speeds up the programming performance.

Please Note: **"CODE segment start address"** is interpreted as a Intel x86 architecture segment address: You have to specify a start address of 10000 to start the conversion at 1 MByte.

This converter is a relatively old DOS tool and therefor it has problems with non DOS compliant file and directory names. Avoid names with spaces, limit names to eight characters. Otherwise the converter does not convert the input file, without any error message!!

5. Support for Windows NT, Windows 2000 and Windows XP

A configured run time version of the "Kithara DOS Enabler, Version 6.x" is used to give support for some of our DOS based tools (like the JTAG-Booster) for Windows NT, Windows 2000 and Windows XP. After installation of the "DOS Enabler" the accesses to the LPT ports are allowed for the all programs listed in file Readme_WinNT.txt

Note: Accesses to the ports are only allowed for the programs listed in file Readme_WinNT.txt. If you rename one of our tools, the DOS Enabler does not work.

Important: You need administrator rights to install or de-install this program.

5.1. Installation on a clean system

If you have a clean system without having installed a previous version of the "Kithara Tool Center", this tool is really simple to install. Extract the ZIP file to a new folder and start KSETUP.EXE. Everything is done within a few seconds. No additional input is needed. Now reboot your PC.

5.2. Installation with already installed version 5.x/6.x of Kithara

If you have already installed an older WinNT support (Kithara Version 5.x or 6.x), you have to de-install it 1st as described in chapter 5.4.

After rebooting your PC you can install the Kithara 6.x as described above.

5.3. Installation with already installed version 4.x of Kithara

Important!! If you have already installed an older WinNT support, you have to deinstall it completely!!!

- Start kcenter
- Select Register "Einstellungen" (=Settings)
  and deactivate "VDD benutzen"
  and "speziellen seriellen Treiber benutzen".
- Stop Kernel

- exit the kcenter program
- Now you can deinstall the Kithara Package with:
  Settings - Control Panel.
  All unused parts must be removed.
- Reboot your PC
- Now you can install the Kithara 6.x as described above.

5.4. De-Installation version 5.x/6.x:

For deinstallation of the runtime version of the "Kithara DOS-Enabler Version 5.x/6.x":

- use: Settings - Control-Panel - Add/Remove Programs
  and remove the
  “FS FORTH-SYSTEME WinNT Support”
  and/or
  “WinNT Support for JTAG-Booster and FLASH166”
- Reboot your PC