JTAG-Booster for

Intel StrongARM SA11x0

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: May 15, 2002
Author: Dieter Fögele
Filename: JTAG_SA-11x0b.doc
Program Version: 4.xx

Table of Contents

1. General

The programs JTAG1100.EXE and JTAG1110.EXE use the JTAG port of the Intel StrongARM SA-11x0 embedded microprocessor 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 an I²C Device
- 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.

The JTAG-BOOSTER' s software is highly optimized to the JTAG chain of a specific target CPU. To give support for all microprocessors of the Intel StrongARM SA-11x0 family, there are two different programs on the distribution disk:

- JTAG1100.EXE : Tool for Intel StrongARM SA-1100
- JTAG1110.EXE : Tool for Intel StrongARM SA-1110

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

1.1. Ordering Information

The following related products are available

- 929 JTAG-Booster Intel StrongARM SA-11x0, 3.3V,
  DOS/Win9x/WinNT,
  delivered with adapter type 285

- 268 Adapter Keith&Koep, adapts the 8 pin connector of JTAG-Booster to 10 pin connector of Keith&Koep Evaluation Platform “ARNOLD II”

1.2. System Requirements

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

- MSDOS, WIN3.x, WIN9x, WinME, WinNT or Win2000
  (WinNT/Win2000 is supported with an additional tool, see chapter 5)

- Intel 80386 or higher

- 205 kByte of free DOS memory

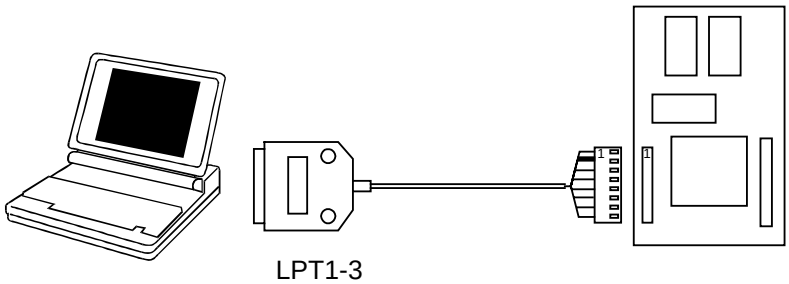
- Parallel Port

1.3. Contents of Distribution Disk

- JTAG1100.EXE
  JTAG1100.OVL — Tool for Intel StrongARM SA-1100
- JTAG1100.INI — Template configuration file for Intel StrongARM SA-1100. See chapter 1.9 "Initialization file JTAG11x0.INI"
- JTAG1110.EXE
  JTAG1110.OVL — Tool for Intel StrongARM SA-1110
- JTAG1110.INI — Template configuration file for Intel StrongARM SA-1110. See chapter 1.9 "Initialization file JTAG11x0.INI"
- HEX2BIN.EXE — Converter program to convert Intel HEX and Motorola S-Record files to binary. See chapter 4 "Converter Program HEX2BIN.EXE"
- WinNT — Support for Windows NT and Windows 2000. See chapter 5 "Support for Windows NT and Windows 2000"
- 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 white 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 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.

For the Keith&Koep Evaluation Platform “ARNOLD II” additional there is a 10 pin connector available to connect the JTAG-Booster (FS part number 268).

+3.3V	+3.3V	GND	GND	GND
2	4	6	8	10
1	3	5	7	9
TRST#	TDI	TDO	TMS	TCK

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:

JTAG11x0 /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 JTAG11x0, press CTRL-Break-Key.

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

1.5. First Example with Intel StrongARM SA-1100

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

```
JTAG1100  /P  MYAPP.BIN
```

at the DOS prompt results in the following output:

```
JTAG1100 --- JTAG utility for Intel StrongARM SA-1100
Copyright © FS FORTH-SYSTEME GmbH, Breisach
Version 4.xx of mm/dd/yyyy

(1)   Configuration loaded from file JTAG11x0.INI
(2)   Target: Keith & Koep, ARNOLD II
(3)   Using LPT at I/O-address 0378h
(4)   JTAG Adapter detected

(5)   1 Device detected in JTAG chain
        Device 0: IDCODE=9108406B   Intel StrongARM SA-1100, E-Step
(6)   Sum of instruction register bits   : 5
(7)   CPU position                       : 0
(8)   Instruction register offset        : 0
(9)   Length of boundary scan reg        : 279

      Looking for a known flash device. Please wait..
(10)  Dual Intel 28F008 FlashFile Smart3/5 detected
(11)  Bus size is 16 Bit
(12)  Erasing Flash-EPROM Block #:0
      Programming File MYAPP.BIN
      65536 Bytes programmed
      Programming ok

      Erase Time          :        1.0 sec
      Programming Time    :       30.3 sec
```

(1) The initialization file JTAG1100.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 Intel StrongARM SA-1100 are switched to bypass mode.

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

(7) The position of the Intel StrongARM SA-11x0 in the JTAG chain is checked.

(8) The position of the JTAG instruction register of the Intel StrongARM SA-11x0 is checked

(9) The real length of the boundary scan register is displayed here and compared with the boundary scan register length of a Intel StrongARM SA-1100.

(10) Two Flash-EPROMs Intel 28F008 selected with chip select CS0 are found.

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

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

1.6. First Example with Intel StrongARM SA-1110

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

```
JTAG1110  /P  MYAPP.BIN
```

at the DOS prompt results in the following output:

```
JTAG1110 --- JTAG utility for Intel StrongARM SA-1110
Copyright © FS FORTH-SYSTEME GmbH, Breisach
Version 4.xx of mm/dd/yyyy

(1)   Configuration loaded from file JTAG1110.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=09261013   Intel StrongARM SA-1110, Revision 0
(6)   Sum of instruction register bits  : 5
(7)   CPU position                      : 0
(8)   Instruction register offset       : 0
(9)   Length of boundary scan reg       : 292

      Looking for a known flash device. Please wait..
(10)  Dual Intel 28F008 FlashFile Smart3/5 detected
(11)  Bus size is 16 Bit
(12)  Erasing Flash-EPROM Block #:0
      Programming File MYAPP.BIN
      65536 Bytes programmed
      Programming ok

      Erase Time          :        0.9 sec
      Programming Time    :       31.7 sec
```

(1) The initialization file JTAG1110.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 Intel StrongARM SA-1110 are switched to bypass mode.

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

(7) The position of the Intel StrongARM SA-11x0 in the JTAG chain is checked.

(8) The position of the JTAG instruction register of the Intel StrongARM SA-11x0 is checked

(9) The real length of the boundary scan register is displayed here and compared with the boundary scan register length of a Intel StrongARM SA-1110.

(10) Two Flash-EPROMs Intel 28F008 selected with chip select CS0 are found.

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

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

1.7. 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.

On very fast PCs there could be verify errors. To avoid this, watch for the ´IO recovery time´-switch in the BIOS Setup which must be turned on. Otherwise try to slow down your PC by setting the turbo switch off.

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 1.9 “Initialization file JTAG11x0.INI”). Also the address bits must be defined as output.

Use the option /NOWRSETUP to speed up flash programming.

1.8. Error Messages

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

- **Adapter not connected or target power fail**
  The JTAG-Booster wasn't found. Please check connection to parallel port and connection to target. Check target power. Check your BIOS-Setup.

- **Can't open x:\yyy\zzz\JTAG11x0.OVL**
  The overlay file JTAG11x0.OVL must be in the same directory as JTAG11x0.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.

- **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#

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

- **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 “JTAG11x0 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 “JTAG11x0 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 1.9 ”Initialization file JTAG11x0.INI”

- **illegal Flash Type:**
  The name specified with the option /DEVICE= must be one of the list of chapter 1.10 ”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 1.9 “Initialization file JTAG11x0.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.

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

- **missing option /I2CCLK=**
  Some functions need the option /I2CCLK= to be defined.

- **missing option /I2CDAT=**
  Some functions need the option /I2CDAT= or the options /I2CDATO= and /I2CDATI= to be defined.

- **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 TDI 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 TDI pin stuck at high level**
  A stream of 32 high bits was detected on the pin TDI. TDI 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 Intel StrongARM SA-11x0**
  The option /CPUPOS= points to a part not a Intel StrongARM SA-11x0

- **Pins specified with /I2CCLK= and /I2CDAT= must have different control cells**
  The pin specified with the option /I2CDAT= must be able to be switched to high impedance while the pin specified with option /I2CCLK= is an active output. See chapter 1.9 “Initialization file JTAG11x0.INI”.

- **Pins specified with /I2CCLK= and /I2CDATI= must have different control cells**
  The pin specified with the option /I2CDATI= must be able to be switched to high impedance while the pin specified with option /I2CCLK= is an active output. See chapter 1.9 “Initialization file JTAG11x0.INI”.

- **Pins specified with /I2CDATO= and /I2CDATI= must have different control cells**
  The pin specified with the option /I2CDATI= must be able to be switched to high impedance while the pin specified with option /I2CDATO= is an active output. See chapter 1.9 “Initialization file JTAG11x0.INI”.

- **Specify only one of that 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 Intel StrongARM SA-11x0**
  The sum of all instruction register bits in the JTAG chain does not fit to the Intel StrongARM SA-11x0. 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 Intel StrongARM SA-11x0 in the JTAG chain**
  No Intel StrongARM SA-11x0 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 Identifier (xxxx)**
  No valid identifier found. Check the specified chip select signal and the bus width. Try with the option /DEVICE= .

- **Wrong length of boundary scan register. Should be 279 for a Intel StrongARM SA-1100. (Should be 292 for a Intel StrongARM SA-1110.)**
  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 Intel StrongARM SA-11x0. 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.

1.9. Initialization file JTAG11x0.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 Intel StrongARM SA-11x0. The Target-Entry is used to identify your design which is displayed with most commands.

When the program JTAG11x0.EXE is started it scans the current directory for an existing initialization file named JTAG11x0.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 JTAG1100.INI:

```
// Description file for Intel StrongARM SA-1100
Target: Keith & Koep, ARNOLD II
// 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 #.

// Group A:  All pins in this group must be set to the same direction
//           This group is switched between output/active and
//           input/tristate during programming of Flash-EPROMs
D0              Out,Lo    //
D1              Out,Lo    //
D2              Out,Lo    //
D3              Out,Lo    //
D4              Out,Lo    //
D5              Out,Lo    //
D6              Out,Lo    //
D7              Out,Lo    //
D8              Out,Lo    //
D9              Out,Lo    //
D10             Out,Lo    //
D11             Out,Lo    //
D12             Out,Lo    //
D13             Out,Lo    //
D14             Out,Lo    //
D15             Out,Lo    //
D16             Out,Lo    //
D17             Out,Lo    //
D18             Out,Lo    //
D19             Out,Lo    //
D20             Out,Lo    //
D21             Out,Lo    //
D22             Out,Lo    //
D23             Out,Lo    //
D24             Out,Lo    //
D25             Out,Lo    //
D26             Out,Lo    //
D27             Out,Lo    //
D28             Out,Lo    //
D29             Out,Lo    //
D30             Out,Lo    //
```

```
D31             Out,Lo    //

// Group B:  All pins in this group must be set to the same direction
//           For Flash programming, this group must be set to output
A0              Out,Lo    //
A1              Out,Lo    //
A2              Out,Lo    //
A3              Out,Lo    //
A4              Out,Lo    //
A5              Out,Lo    //
A6              Out,Lo    //
A7              Out,Lo    //
A8              Out,Lo    //
A9              Out,Lo    //
A10             Out,Lo    //
A11             Out,Lo    //
A12             Out,Lo    //
A13             Out,Lo    //
A14             Out,Lo    //
A15             Out,Lo    //
A16             Out,Lo    //
A17             Out,Lo    //
A18             Out,Lo    //
A19             Out,Lo    //
A20             Out,Lo    //
A21             Out,Lo    //
A22             Out,Lo    //
A23             Out,Lo    //
A24             Out,Lo    //
A25             Out,Lo    //
CS0#            Out,Hi    // = chip select 0
CS1#            Out,Hi    // = chip select 1
CS2#            Out,Hi    // = chip select 2
CS3#            Out,Hi    // = chip select 3
CAS0#           Out,Hi    //
CAS1#           Out,Hi    //
CAS2#           Out,Hi    //
CAS3#           Out,Hi    //
RAS0#           Out,Hi    //
RAS1#           Out,Hi    //
RAS2#           Out,Hi    //
RAS3#           Out,Hi    //
OE#             Out,Hi    // = output enable, is switched while accessing Flash
```

```
WE#             Out,Hi     // = write enable, is switched while accessing Flash

// 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 input.
L_FCLK          Inp
L_LCLK          Inp
LDD7            Inp
LDD6            Inp
LDD5            Inp
LDD4            Inp
LDD3            Inp
LDD2            Inp
LDD1            Inp
LDD0            Inp
L_PCLK          Inp
L_BIAS          Inp
GP0             Inp
GP1             Inp
GP2             Inp
GP3             Inp
GP4             Inp
GP5             Inp
GP6             Inp
GP7             Inp
GP8             Inp
GP9             Inp
GP10            Inp
GP11            Inp
GP12            Inp
GP13            Inp
GP14            Inp
GP15            Inp
GP16            Inp
GP17            Inp
GP18            Inp
GP19            Inp
GP20            Inp
GP21            Inp
GP22            Inp
GP23            Inp
GP24            Inp
GP25            Inp
```

```
GP26            Inp
GP27            Inp
TXD_C           Inp
RXD_C           Inp
SCLK_C          Inp
SFRM_C          Inp

// The following pins are output only pins.
// Setting to input (tristate) one of these pins results in an error.
POE#            Out,Hi    // = PCMCIA output enable
PWE#            Out,Hi    // = PCMCIA write enable
PIOR#           Out,Hi    // = PCMCIA IO read
PIOW#           Out,Hi    // = PCMCIA IO write
PSKTSEL         Out,Lo    //
PREG#           Out,Hi    //
PCE2#           Out,Hi    //
PCE1#           Out,Hi    //
RESET_OUT#      Out,Hi    // = Reset output, reset of Flashes connected

// The following pins are input only pins
// Setting to output one of these pins results in an error.
// Declaration of the direction of these pins is optional.
BATT_FAULT      Inp       //
VDD_FAULT       Inp       //
IOIS16#         Inp       //
PWAIT#          Inp       //
RESET#          Inp       //
ROMSEL          Inp       //
```

Sample File JTAG1110.INI:

```
// Description file for Intel StrongARM SA-1110
Target: Generic Target
// 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 #.

// Group A:  All pins in this group must be set to the same direction
//           This group is switched between output/active and
//           input/tristate during programming of Flash-EPROMs
D0              Out,Lo    //
D1              Out,Lo    //
D2              Out,Lo    //
D3              Out,Lo    //
D4              Out,Lo    //
D5              Out,Lo    //
D6              Out,Lo    //
D7              Out,Lo    //
D8              Out,Lo    //
D9              Out,Lo    //
D10             Out,Lo    //
D11             Out,Lo    //
D12             Out,Lo    //
D13             Out,Lo    //
D14             Out,Lo    //
D15             Out,Lo    //
D16             Out,Lo    //
D17             Out,Lo    //
D18             Out,Lo    //
D19             Out,Lo    //
D20             Out,Lo    //
D21             Out,Lo    //
D22             Out,Lo    //
D23             Out,Lo    //
D24             Out,Lo    //
D25             Out,Lo    //
D26             Out,Lo    //
D27             Out,Lo    //
D28             Out,Lo    //
D29             Out,Lo    //
D30             Out,Lo    //
```

```
D31             Out,Lo    //

// Group B:  All pins in this group must be set to the same direction
//           For Flash programming, this group must be set to output
A0              Out,Lo    //
A1              Out,Lo    //
A2              Out,Lo    //
A3              Out,Lo    //
A4              Out,Lo    //
A5              Out,Lo    //
A6              Out,Lo    //
A7              Out,Lo    //
A8              Out,Lo    //
A9              Out,Lo    //
A10             Out,Lo    //
A11             Out,Lo    //
A12             Out,Lo    //
A13             Out,Lo    //
A14             Out,Lo    //
A15             Out,Lo    //
A16             Out,Lo    //
A17             Out,Lo    //
A18             Out,Lo    //
A19             Out,Lo    //
A20             Out,Lo    //
A21             Out,Lo    //
A22             Out,Lo    //
A23             Out,Lo    //
A24             Out,Lo    //
A25             Out,Lo    //
CAS0#           Out,Hi    //
CAS1#           Out,Hi    //
CAS2#           Out,Hi    //
CAS3#           Out,Hi    //
RAS0#           Out,Hi    //
SDCAS#          Out,Hi    //
SDRAS#          Out,Hi    //
OE#             Out,Hi    // = output enable, is switched while programming
WE#             Out,Hi    // = write enable, is switched while programming
```

```
// 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 input.
TXD_3           Inp
RXD_3           Inp
TXD_2           Inp
RXD_2           Inp
TXD_1           Inp
RXD_1           Inp
UDC+            Inp
UDC-            Inp
L_FCLK          Inp
L_LCLK          Inp
LDD7            Inp
LDD6            Inp
LDD5            Inp
LDD4            Inp
LDD3            Inp
LDD2            Inp
LDD1            Inp
LDD0            Inp
L_PCLK          Inp
L_BIAS          Inp
GP0             Inp
GP1             Inp
GP2             Inp
GP3             Inp
GP4             Inp
GP5             Inp
GP6             Inp
GP7             Inp
GP8             Inp
GP9             Inp
GP10            Inp
GP11            Inp
GP12            Inp
GP13            Inp
GP14            Inp
GP15            Inp
GP16            Inp
GP17            Inp
GP18            Inp
GP19            Inp
```

```
GP20            Inp
GP21            Inp
GP22            Inp
GP23            Inp
GP24            Inp
GP25            Inp
GP26            Inp
GP27            Inp
TXD_C           Inp
RXD_C           Inp
SCLK_C          Inp
SFRM_C          Inp

// The following pins are output only pins.
// Setting to input (tristate) one of these pins results in an error.
SDCLK2          Out,Lo  //
SDCKE1          Out,Lo  //
SDCKE0          Out,Lo  //
SDCLK0          Out,Lo  //
POE#            Out,Hi  // = PCMCIA output enable
PWE#            Out,Hi  // = PCMCIA write enable
PIOR#           Out,Hi  // = PCMCIA IO read
PIOW#           Out,Hi  // = PCMCIA IO write
PSKTSEL         Out,Lo  // = select socket A
PREG#           Out,Hi  // = select memory space
PCE2#           Out,Hi  //
PCE1#           Out,Hi  //
RAS3#           Out,Hi  //
RAS2#           Out,Hi  //
RAS1#           Out,Hi  //
RD_WR#          Out,Hi  //
CS0#            Out,Hi  // = chip select 0
CS1#            Out,Hi  // = chip select 1
CS2#            Out,Hi  // = chip select 2
CS3#            Out,Hi  // = chip select 3
CS4#            Out,Hi  // = chip select 4
CS5#            Out,Hi  // = chip select 5
RESET_OUT#      Out,Hi  // = Reset output, reset of Flashes connected
```

```
// 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.
BATT_FAULT   Inp      //
VDD_FAULT    Inp      //
SMROM_EN     Inp      //
IOIS16#      Inp      //
PWAIT#       Inp      //
RDY          Inp      //
RESET#       Inp      //
ROMSEL       Inp      //
```

1.10. Supported flash devices

Type JTAG11x0 /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. JTAG11x0 Parameter Description

When you start JTAG11x0.EXE without any parameters the following help screen with all possible functions and options is displayed:

```
JTAG11x0 --- JTAG utility for Intel StrongARM SA-11x0
Copyright © FS FORTH-SYSTEME GmbH, Breisach
Version 4.xx of mm/dd/yyyy

Programming of Flash-EPROMs and hardware tests on targets with the
Intel StrongARM SA-11x0.

The JTAG-Booster is needed to connect the parallel port of the PC
to the JTAG port of the Intel StrongARM SA-11x0.

Usage: JTAG11x0 /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
    /PI2C       : Program an I2C Device with file
    /RI2C       : Read an I2C Device to file
    /VI2C       : Verify an I2C Device with file
    /DUMPI2C    : Make a dump of an I2C 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	/CS4
/CS5	/CARD	/CARDIO	/CARDREG	/NOCS
/NOWRSETUP	/TOP	/BYTE-MODE	/BM	/PAUSE
/P	/NODUMP	/NOERASE	/LATTICE	/ERASEALL
/LPT1	/LPT2	/LPT3	/LPT-BASE=	/32BIT
/16BIT	/8BIT	/NOMAN	/LENGTH=	L=
/FILE-OFFSET=	/FO=	/OFFSET=	/O=	DELAY=
/DEVICE-BASE=	/DB=	/DRIVER=	/DATA-MASK=	/DM=
/IROFFS=	/CPUPOS=	/DEVICE=	/PIN=	/I2CCLK=
/I2CDAT=	/I2CDATI=	/I2CDATO=	/I2CBIG	/WATCH=
/OUT=	/INI=	/REP		

The following options are valid for most functions:

/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 JTAG11x0.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 1.9 “Initialization file JTAG11x0.INI”).
Default: /INI=JTAG11x0.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.
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 or Win2000, 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 operation 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 Intel StrongARM SA-11x0 instruction register within the JTAG chain. In most cases this option is not needed.
Default: /IROFFS=0

/CPUPOS=
Specifies the position of the Intel StrongARM SA-11x0 within the JTAG chain.
Default: /CPUPOS=0

2.1. Program a Flash Device

Usage: JTAG11x0 /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 devices are shown in chapter 1.10 “Supported flash 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.

/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.

/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 by 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=

/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=

[1]hhhhhh=number base is hex

/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 (Intel StrongARM SA-1100)
/CS0 /CS1 /CS2 /CS3 /CS4 /CS5 (Intel StrongARM SA-1110)
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 1.9 “Initialization file JTAG11x0.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 Intel StrongARM SA-11x0 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:

JTAG11x0 /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.

JTAG11x0 /P CE.ROM /32BIT /CS1
This example programs the file CE.ROM to the 32 Bit Flash-EPROM connected to CS1#.

2.2. Read a Flash Device to file

Usage: JTAG11x0 /R filename [optionlist]

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

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 1.10 “Supported flash devices”.

Options:

/DEVICE=devicename
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)

/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=

[2]hhhhhh=number base is hex

/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 /NOCS (Intel StrongARM SA-1100)
/CS0 /CS1 /CS2 /CS3 /CS4 /CS5 /NOCS (Intel StrongARM SA-1110)
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:

JTAG11x0 /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: JTAG11x0 /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 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 1.10 “Supported flash devices”.

Options:

/DEVICE=devicename
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 /NOCS (Intel StrongARM SA-1100)
/CS0 /CS1 /CS2 /CS3 /CS4 /CS5 /NOCS (Intel StrongARM SA-1110)
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:

JTAG11x0 /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: JTAG11x0 /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: /16BIT

/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 /NOCS (Intel StrongARM SA-1100)
/CS0 /CS1 /CS2 /CS3 /CS4 /CS5 /NOCS (Intel StrongARM SA-1110)
See function /P (Chapter 2.1)
Default: /CS0

[3]hhhhhh=number base is hex

Example:

JTAG11x0 /DUMP /BMS
This example makes a memory dump of the first 256 bytes of the Boot-EPROM.

2.5. Program an I²C-Device

Usage: JTAG11x0 /PI2C filename [/I2CBIG] [optionlist]

The specified file is programmed to an I²C-Device (i.e. a serial EEPROM) connected to pins of the CPU. Finally a complete verify is done. If the verify fails, the contents of the I²C-Device is written to a file with the extension DMP.

Two methods to connect the I²C-Device to the CPU are supported. The first method is to use two CPU pins, one pin for clock output (I2CCLK) and one pin for serial data input and output (I2CDAT). The second method is to use one pin for clock output (I2CCLK), one for serial data input (I2CDATI) and one for serial data output (I2CDATO).

Options:

/I2CBIG
Specify this option if there is a device which needs a three byte address instead of a two byte address.
This option must be the first option after the filename.

/DEVICE-BASE=hhhhhh
This option specifies an I²C device starting address. The default values are chosen to access an serial EEPROM.
Default: /DEVICE-BASE=5000 (if option /I2CBIG omitted)
Default: /DEVICE-BASE=500000 (if option /I2CBIG specified)

/OFFSET=hhhhhh
The programming starts at an offset of hhhhhh relative to the start address of the I²C-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.

/I2CCLK=pin_name
Specifies the CPU pin used for serial clock output.

/I2CDAT=pin_name
Specifies the CPU pin used for serial data input and output. 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 /I2CDATO= and /I2CDATI= .

/I2CDATO=pin_name
Specifies the CPU pin used for serial data output. Pin_name must specify a output pin otherwise an error message occurs.

/I2CDATI=pin_name
Specifies the CPU pin used for serial data input. Pin_name must specify a input pin otherwise an error message occurs.

Example:

JTAG11x0 /PI2C EEPROM.CFG /I2CCLK=GP26 /I2CDAT=GP27
This example loads the file EEPROM.CFG to a serial EEPROM connected to the pins GP26 and GP27 of the Intel StrongARM SA-11x0.

2.6. Read an I²C-Device to file

Usage: JTAG11x0 /RI2C filename [/I2CBIG] /L=hhhhhh [optionlist]

The contents of an I²C-Device (i.e. a serial EEPROM) is read and written to a file. The option /LENGTH= must be specified.

Options:

/I2CBIG
This option must be the first option after the filename.
See function /PI2C (Chapter 2.5)

/DEVICE-BASE=hhhhhh
See function /PI2C (Chapter 2.5)

/OFFSET=hhhhhh
Reading of the I²C-Device starts at an offset of hhhhhh relative to the start address of the I²C-Device.
Default: /OFFSET=0
Abbreviation: /O=

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

/I2CCLK=pin_name
See function /PI2C (Chapter 2.5)

/I2CDAT=pin_name
See function /PI2C (Chapter 2.5)

/I2CDATO=pin_name
See function /PI2C (Chapter 2.5)

/I2CDATI=pin_name
See function /PI2C (Chapter 2.5)

Example:

JTAG11x0 /RI2C EEPROM.CFG /I2CCLK=GP26 /I2CDAT=GP27 /L=100
This example reads 256 bytes from a serial EEPROM to the file EEPROM.CFG. The serial EEPROM is connected to the pins CP26 and GP27 of the Intel StrongARM SA-11x0.

2.7. Verify an I²C-Device with file

Usage: JTAG11x0 /VI2C filename [I2CBIG] [optionlist]

The contents of an I²C-Device (i.e. a serial 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:

/I2CBIG
This option must be the first option after the filename.
See function /PI2C (Chapter 2.5)

/DEVICE-BASE=hhhhhh
See function /PI2C (Chapter 2.5)

/OFFSET=hhhhhh
See function /PI2C (Chapter 2.5)

/FILE-OFFSET=hhhhhh
See function /PI2C (Chapter 2.5)

/LENGTH=hhhhhh
See function /PI2C (Chapter 2.5)

/NODUMP
See function /PI2C (Chapter 2.5)

/I2CCLK=pin_name
See function /PI2C (Chapter 2.5)

/I2CDAT=pin_name
See function /PI2C (Chapter 2.5)

/I2CDATO=pin_name
See function /PI2C (Chapter 2.5)

/I2CDATI=pin_name
See function /PI2C (Chapter 2.5)

Example:

JTAG11x0 /VI2C EEPROM.CFG /I2CCLK=GP26 /I2CDAT=GP27
This example verifies 256 bytes from a serial EEPROM with the file EEPROM.CFG. The serial EEPROM is connected to the pins CP26 and GP27 of the Intel StrongARM SA-11x0.

2.8. Dump an I²C-Device

Usage: JTAG11x0 /DUMPI2C [I2CBIG] [optionlist]

A Hex-Dump of an I²C-Device is printed on the screen, if not redirected to file or device.

Options:

/I2CBIG
This option must be the first option.
See function /PI2C (Chapter 2.5)

/DEVICE-BASE=hhhhhh
See function /PI2C (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=

/I2CCLK=pin_name
Specifies the CPU pin used for serial clock output.

/I2CDAT=pin_name
Specifies the CPU pin used for serial data input and output. 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 /I2CDATO= and /I2CDATI= .

/I2CDATO=pin_name
Specifies the CPU pin used for serial data output. Pin_name must specify a output pin otherwise an error message occurs.

[4]hhhhhh=number base is hex

/I2CDATI=pin_name
Specifies the CPU pin used for serial data input. Pin_name must specify a input pin otherwise an error message occurs.

Example:

JTAG11x0 /DUMPI2C /I2CCLK=FLAG0 /I2CDAT=FLAG1
This example makes a memory dump of the first 100h bytes of a serial EEPROM connected to the CPU.

2.9. Toggle CPU pins

Usage: JTAG11x0 /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 Intel StrongARM SA-11x0 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 1.9 “Initialization file JTAG11x0.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:

JTAG11x0 /BLINK /PIN=FLAG3 /DELAY=0
This example toggles the GP26 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: JTAG11x0 /PIN? /PIN=pinname [optionlist]

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

Please Note: Not every pin of the Intel StrongARM SA-11x0 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 1.9 “Initialization file JTAG11x0.INI” can be used. If you type /PIN= without any pin declaration a list of the CPU pins is displayed.

Example:

JTAG11x0 /PIN? /PIN=RESET#
This example samples the reset pin of the Intel StrongARM SA-11x0.

2.11. Polling CPU pins while the CPU is running

Usage: JTAG11x0 /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 1.9 “Initialization file JTAG11x0.INI” can be used. If you type /PIN= without any pin declaration a list of the CPU pins is displayed.

Example:

JTAG11x0 /SAMPLE /PIN=FLAG3
This example samples the state of the port pin FLAG3 while the Intel StrongARM SA-11x0 is running.

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

Usage: JTAG11x0 /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 Intel StrongARM SA-1110

0 BATT_FAULT	0 VDD_FAULT	1 SFRM_C	1 SCLK_C
1 RXD_C	1 TXD_C	1 D0	1 D1
1 D2	1 D3	1 D4	1 D5
1 D6	1 D7	0 D8	1 D9
0 D10	1 D11	0 D12	1 D13
1 D14	1 D15	0 D16	0 D17
0 D18	0 D19	0 D20	0 D21
0 D22	0 D23	0 D24	0 D25
0 D26	0 D27	0 D28	0 D29
0 D30	0 D31	0 GP27	0 GP26
0 GP25	0 GP24	0 GP23	0 GP22
0 GP21	1 GP20	1 GP19	1 GP18
1 GP17	1 GP16	1 GP15	1 GP14
1 GP13	1 GP12	1 GP11	1 GP10
1 GP9	1 GP8	1 GP7	1 GP6
1 GP5	1 GP4	1 GP3	1 GP2
1 GP1	1 GP0	1 L_BIAS	1 L_PCLK
1 LDD0	1 LDD1	1 LDD2	1 LDD3
1 LDD4	1 LDD5	1 LDD6	1 LDD7
1 L_LCLK	1 L_FCLK	1 POE#	1 PWE#
1 PIOR#	1 PIOW#	0 PSKTSEL	1 IOIS16#
1 PWAIT#	0 PREG#	1 PCE2#	1 PCE1#
1 WE#	0 OE#	1 RAS3#	1 RAS2#
1 RAS1#	1 RAS0#	1 CAS3#	1 CAS2#
1 CAS1#	1 CAS0#	1 CS3#	1 CS2#
1 CS1#	0 CS0#	0 A25	0 A24
0 A23	0 A22	0 A21	0 A20
0 A19	0 A18	0 A17	0 A16
0 A15	0 A14	0 A13	0 A12
0 A11	0 A10	1 A9	0 A8
1 A7	0 A6	0 A5	0 A4
1 A3	1 A2	1 A1	0 A0
1 UDC-	1 UDC+	1 RXD_1	1 TXD_1
1 RXD_2	1 TXD_2	1 RXD_3	1 TXD_3
1 RESET#	1 RESET_OUT#	0 ROMSEL	

3. Implementation Information

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

- The JTAG-Booster uses the EXTEST function of the JTAG-Interface to perform Flash programming.

- Refer to the following table for connecting Flash-EPROMs to the Intel StrongARM SA-11x0:

SA-1100 signal	8 Bit Flash	16 Bit Flash	32 Bit Flash
CS0# CS1# CS2# CS3#	CS#	CS#	CS#
OE#	OE#	OE#	OE#
WE#	WE#	WE#	WE#
D0..7	D0..7	-	-
D0..15	-	D0..15	-
D0..31	-	-	D0..31

SA-1110 signal	8 Bit Flash	16 Bit Flash	32 Bit Flash
CS0# CS1# CS2# CS3# CS4# CS5#	CS#	CS#	CS#
OE#	OE#	OE#	OE#
WE#	WE#	WE#	WE#
D0..7	D0..7	-	-
D0..15	-	D0..15	-
D0..31	-	-	D0..31

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 and Windows 2000

A configured run time version of the "Kithara DOS Enabler, Version 5.1" is used to give support for some of our DOS based tools (like the JTAG-Booster) for Windows NT and Windows 2000. After installation of the "DOS Enabler" the accesses to the LPT or COM 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.

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 a previous version 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 5.xx as described above.

5.3. De-Installation version 5.xx:

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

- use: Settings - Control-Panel - Add/Remove Programs
  and remove the
  "WinNT support for JTAG-Booster and FLASH166"

- Reboot your PC