JTAG-Booster for

NetSilicon NS9xxx

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_NS9xxxb.doc
Program Version: 4.xx

Table of Contents

1. General

The programs JTAG9750.EXE and JTAG9360.EXE use the IEEE 1149.1 JTAG port of the NetSilicon NS9xxx microcontrollers 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, 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 NetSilicon NS9xxx family, there are two different programs on the distribution disk:

- JTAG9750.EXE : Tool for NetSilicon NS9750/NS9775
- JTAG9360.EXE : Tool for NetSilicon NS9360

Please contact us, if you need support for other members of the NetSilicon NS9xxx 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

- 9051 JTAG-Booster NetSilicon NS9xxx, 3.3V,
  NetSilicon NS9750, NS9775, NS9360
  DOS/Win9x/WinNT/Win2000/WinXP,
  delivered with adapter type 285
  and adapter/converter for ARM type 360

1.2. System Requirements

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

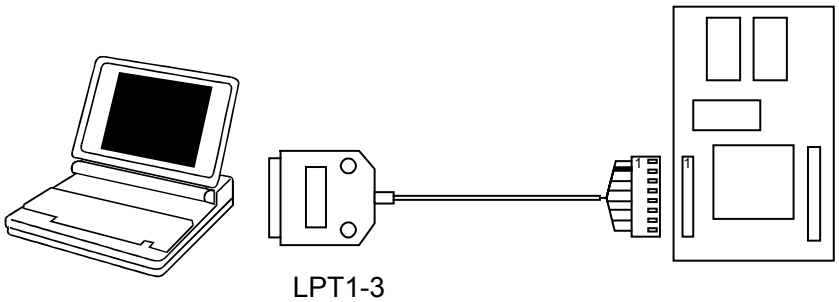
- MSDOS, WIN3.x, WIN9x, WinNT, Win2000 or WindowsXP
  (WinNT/Win2000/WindowsXP is supported with an additional tool, see chapter 5 “Support for Windows NT, Windows 2000 and Windows XP”)
- Intel 80386 or higher
- 205 kByte of free DOS memory
- Parallel Port

1.3. Contents of Distribution Disk

- JTAG9750.EXE
  JTAG9750.OVL — Tool for NetSilicon NS9750/NS9775
- JTAG9750.INI — Template configuration file for NetSilicon NS9750/NS9775. See chapter 1.9 "Initialization file JTAG9xxx.INI"
- JTAG9360.EXE
  JTAG9360.OVL — Tool for NetSilicon NS9360
- JTAG9360.INI — Template configuration file for NetSilicon NS9360. See chapter 1.9 "Initialization file JTAG9xxx.INI"
- WinNT.zip — 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 NS9750 must be configured to work in **scan mode**: CPU signals **BIST_EN#** and SCAN_EN# must be switched to **low level**, while the signal PLL_TEST# must be at high level. (For enabling the **ARM debug** mode, **BIST_EN#** must be switched to **high level**.)

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

JTAG9xxx /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 JTAG9xxx, press CTRL-Break-Key.

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

1.5. First Example with NetSilicon NS9750/NS9775

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

```
JTAG9750  /P  MYAPP.BIN
```

at the DOS prompt results in the following output:

```
JTAG9750 --- JTAG utility for NetSilicon NS9750/NS9775
Copyright © FS FORTH-SYSTEME GmbH, Breisach
Version 4.xx of mm/dd/yyyy

(1)   Configuration loaded from file JTAG9750.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=09104031  NetSilicon NS9750/NS9775, Scan Mode, Revision 0
(6)   Sum of instruction register bits  : 3
(7)   CPU position                      : 0
(8)   Instruction register offset       : 0
(9)   Length of boundary scan reg       : 464
(10)  Boundary Scan used

      Looking for a known flash device. Please wait..
(11)  Dual STM 29W320B, 3.3V, Boot Block Bottom  detected
(12)  Bus size is 32 Bit
(13)  Erasing Flash-EPROM Block #:0  1  2
      Programming File MYAPP.BIN
      65536 Bytes programmed successfully

      Erase Time          :       0.8 sec
      Programming Time    :      xx.8 sec
```

(1) The initialization file JTAG9750.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 NetSilicon NS9750/NS9775 are switched to bypass mode.

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

(7) The position of the NetSilicon NS9750/NS9775 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 NetSilicon NS9750/NS9775 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 NetSilicon NS9750/NS9775.

(10) The NetSilicon NS9750/NS9775 is configured to work in boundary scan mode instead of working in debug mode by having the configuration pins set to: PLL_TEST# = 1, BIST_EN# = 0 and SCAN_EN# = 0).

(11) Two flashes STM 29W320B selected with EXT_CS1# where found.

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

(13) In this example 3 blocks must be erased.

1.6. First Example with NetSilicon NS9360

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

```
JTAG9360  /P  MYAPP.BIN
```

at the DOS prompt results in the following output:

```
JTAG9360 --- JTAG utility for NetSilicon NS9360
Copyright © FS FORTH-SYSTEME GmbH, Breisach
Version 4.xx of mm/dd/yyyy

(1)   Configuration loaded from file JTAG9360.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=09105031  NetSilicon NS9360, Scan Mode, Revision 0
(6)   Sum of instruction register bits  : 3
(7)   CPU position                      : 0
(8)   Instruction register offset       : 0
(9)   Length of boundary scan reg       : 312
(10)  Boundary Scan used

      Looking for a known flash device. Please wait..
(11)  Dual STM 29W320B, 3.3V, Boot Block Bottom  detected
(12)  Bus size is 32 Bit
(13)  Erasing Flash-EPROM Block #:0  1  2
      Programming File MYAPP.BIN
      65536 Bytes programmed successfully

      Erase Time          :       0.8 sec
      Programming Time    :      xx.9 sec
```

(1) The initialization file JTAG9360.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 NetSilicon NS9360 are switched to bypass mode.

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

(7) The position of the NetSilicon NS9xxx 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 NetSilicon NS9xxx 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 NetSilicon NS9360.

(10) The NetSilicon NS9360 is configured to work in boundary scan mode instead of working in debug mode by having the configuration pins set to: PLL_TEST# = 1, BIST_EN# = 0 and SCAN_EN# = 0).

(11) Two flashes STM 29W320B selected with EXT_CS1# where found.

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

(13) In this example 3 blocks must be erased.

1.7. Trouble Shooting

The NetSilicon NS9xxx must be configured to work in scan mode instead of having ARM debug mode enabled (PLL_TEST# = 1, BIST_EN# = 0, SCAN_EN# = 0).

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 1.9 “Initialization file JTAG9xxx.INI”). Also the address bits must be defined as output.

Use the option /NOWRSETUP to speed up flash programming.

If you have problems using the option /CFI (Common Flash Interface) add the command line option /CFIDEBUG and redirect the program output into a file. Sending us this file helps in solving problems.

1.8. 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\JTAG9xxx.OVL**
  The overlay file JTAG9xxx.OVL must be in the same directory as JTAG9xxx.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 “JTAG9xxx 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 “JTAG9xxx 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 JTAG9xxx.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 JTAG9xxx.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 NetSilicon NS9xxx**
  The option /CPUPOS= points to a part not a NetSilicon NS9xxx

- **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 1.9 “Initialization file JTAG9xxx.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 1.9 “Initialization file JTAG9xxx.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 1.9 “Initialization file JTAG9xxx.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 3 bits for a NetSilicon NS9xxx**
  The sum of all instruction register bits in the JTAG chain does not fit to the NetSilicon NS9xxx. 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 NetSilicon NS9xxx in the JTAG chain**
  No NetSilicon NS9xxx 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 464 for a NetSilicon NS9750/NS9775. (Should be 312 for a NetSilicon NS9360.)**
  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 NetSilicon NS9xxx. 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 JTAG9xxx.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 NetSilicon NS9xxx. The Target-Entry is used to identify your design which is displayed with most commands.

When the program JTAG9xxx.EXE is started it scans the current directory for an existing initialization file named JTAG9xxx.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 JTAG9750.INI:

```
// Description file for NetSilicon NS9750/ns9775
Target: Generic Target, 2004/09/22
// 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
DATA0           Inp     // Data Bus for SDRAM, Static Memory, Peripherals
DATA1           Inp     //
DATA2           Inp     //
DATA3           Inp     //
DATA4           Inp     //
DATA5           Inp     //
DATA6           Inp     //
DATA7           Inp     //
DATA8           Inp     //
DATA9           Inp     //
DATA10          Inp     //
DATA11          Inp     //
DATA12          Inp     //
DATA13          Inp     //
DATA14          Inp     //
DATA15          Inp     //
DATA16          Inp     //
DATA17          Inp     //
DATA18          Inp     //
DATA19          Inp     //
DATA20          Inp     //
DATA21          Inp     //
DATA22          Inp     //
DATA23          Inp     //
DATA24          Inp     //
DATA25          Inp     //
```

```
DATA26          Inp     //
DATA27          Inp     //
DATA28          Inp     //
DATA29          Inp     //
DATA30          Inp     //
DATA31          Inp     //

// The following pins are tristateable outputs.
// These pins are tristateable outputs but can not be read back.
// Each pin can be disabled independent of the other pins.
// For Flash Programming these pins must be set to output
ADDR0           Out,Lo  // Data Bus for SDRAM, Static memory, Peripherals
ADDR1           Out,Lo  //
ADDR2           Out,Lo  //
ADDR3           Out,Lo  //
ADDR4           Out,Lo  //
ADDR5           Out,Lo  //
ADDR6           Out,Lo  //
ADDR7           Out,Lo  //
ADDR8           Out,Lo  //
ADDR9           Out,Lo  //
ADDR10          Out,Lo  //
ADDR11          Out,Lo  //
ADDR12          Out,Lo  //
ADDR13          Out,Lo  //
ADDR14          Out,Lo  //
ADDR15          Out,Lo  //
ADDR16          Out,Lo  //
ADDR17          Out,Lo  //
ADDR18          Out,Lo  //
ADDR19          Out,Lo  //
ADDR20          Out,Lo  //
ADDR21          Out,Lo  //
ADDR22          Out,Lo  //
ADDR23          Out,Lo  //
ADDR24          Out,Lo  //
ADDR25          Out,Lo  //
ADDR26          Out,Lo  //
ADDR27          Out,Lo  //
```

```
// 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.
RTCK_OUT        Inp     // ???
RESET_DONE      Inp     // ???
PCI_INTA#       Inp     //
PCI_INTB#       Inp     // CCLKRUN#
PCI_RESET#      Inp     //
PCI_AD0         Inp     //
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_CBE1#       Inp     //
PCI_CBE2#       Inp     //
```

```
PCI_CBE3#       Inp   //
PCI_FRAME#      Inp   //
PCI_IRDY#       Inp   //
PCI_TRDY#       Inp   //
PCI_DEVSEL#     Inp   //
PCI_STOP#       Inp   //
PCI_PERR#       Inp   //
PCI_SERR#       Inp   //
PCI_PAR         Inp   //
BP_STAT0        Inp   //
BP_STAT1        Inp   //
BP_STAT2        Inp   //
BP_STAT3        Inp   //
VSYNC0          Inp   //
VSYNC1          Inp   //
VSYNC2          Inp   //
VSYNC3          Inp   //
PRINT           Inp   //
MII_MDIO        Inp   //
GPIO0           Inp   //
GPIO1           Inp   //
GPIO2           Inp   //
GPIO3           Inp   //
GPIO4           Inp   //
GPIO5           Inp   //
GPIO6           Inp   //
GPIO7           Inp   //
GPIO8           Inp   //
GPIO9           Inp   //
GPIO10          Inp   //
GPIO11          Inp   //
GPIO12          Inp   //
GPIO13          Inp   //
GPIO14          Inp   //
GPIO15          Inp   //
GPIO16          Inp   //
GPIO17          Inp   //
GPIO18          Inp   //
GPIO19          Inp   //
GPIO20          Inp   //
GPIO21          Inp   //
GPIO22          Inp   //
GPIO23          Inp   //
```

```
GPIO24          Inp     //
GPIO25          Inp     //
GPIO26          Inp     //
GPIO27          Inp     //
GPIO28          Inp     //
GPIO29          Inp     //
GPIO30          Inp     //
GPIO31          Inp     //
GPIO32          Inp     //
GPIO33          Inp     //
GPIO34          Inp     //
GPIO35          Inp     //
GPIO36          Inp     //
GPIO37          Inp     //
GPIO38          Inp     //
GPIO39          Inp     //
GPIO40          Inp     //
GPIO41          Inp     //
GPIO42          Inp     //
GPIO43          Inp     //
GPIO44          Inp     //
GPIO45          Inp     //
GPIO46          Inp     //
GPIO47          Inp     //
GPIO48          Inp     //
GPIO49          Inp     //
I2C_SCL         Inp     //
I2C_SDA         Inp     //

// The following pins are tristateable outputs.
// These pins are tristateable outputs but can not be read back.
// Each pin can be disabled independent of the other pins.
PCI_GNT1#       Out,Hi  //
PCI_GNT2#       Out,Hi  //
PCI_GNT3#       Out,Hi  //
PCI_CLKOUT      Inp     //
SDM_CLKOUT0     Out,Lo  // SDRAM clock output
SDM_CLKOUT1     Out,Lo  // SDRAM clock output
SDM_CLKOUT2     Out,Lo  // SDRAM clock output
SDM_CLKOUT3     Out,Lo  // SDRAM clock output
EXT_CS0#        Out,Hi  // Peripheral chip select
EXT_CS1#        Out,Hi  // Peripheral chip select
EXT_CS2#        Out,Hi  // Peripheral chip select
```

```
EXT_CS3#        Out,Hi  // Peripheral chip select
SDM_CS0#        Out,Hi  // SDRAM chip select
SDM_CS1#        Out,Hi  // SDRAM chip select
SDM_CS2#        Out,Hi  // SDRAM chip select
SDM_CS3#        Out,Hi  // SDRAM chip select
WE#             Out,Hi  // Write Enable, SDRAM/Peripheral
EXT_OE#         Out,Hi  // Peripheral output enable
SDM_CAS#        Out,Hi  // SDRAM column address strobe
SDM_RAS#        Out,Hi  // SDRAM row address strobe
EXT_BE0#        Out,Lo  // Peripheral Byte Enable
EXT_BE1#        Out,Lo  // Peripheral Byte Enable
EXT_BE2#        Out,Lo  // Peripheral Byte Enable
EXT_BE3#        Out,Lo  // Peripheral Byte Enable
SDM_DQM0        Out,Lo  // SDRAM Data Mask
SDM_DQM1        Out,Lo  // SDRAM Data Mask
SDM_DQM2        Out,Lo  // SDRAM Data Mask
SDM_DQM3        Out,Lo  // SDRAM Data Mask
SDM_CKE0        Out,Hi  // SDRAM clock enable
SDM_CKE1        Out,Hi  // SDRAM clock enable
SDM_CKE2        Out,Hi  // SDRAM clock enable
SDM_CKE3        Out,Hi  // SDRAM clock enable
VIDEO_DATA0     Inp     //
VIDEO_DATA1     Inp     //
VIDEO_DATA2     Inp     //
VIDEO_DATA3     Inp     //
MII_TXD0        Inp     //
MII_TXD1        Inp     //
MII_TXD2        Inp     //
MII_TXD3        Inp     //
MII_TXEN        Inp     //
MII_TXER        Inp     //
MII_MDC         Inp     //
```

```
// 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.
PCI_CENTRAL#    Inp     //
PCI_INTD#       Inp     //
PCI_INTC#       Inp     //
PCI_REQ1#       Inp     //
PCI_REQ2#       Inp     //
PCI_REQ3#       Inp     //
PCI_CLKIN       Inp     //
PCI_IDSEL       Inp     //
SDM_CLKIN0      Inp     //
SDM_CLKIN1      Inp     //
SDM_CLKIN2      Inp     //
SDM_CLKIN3      Inp     //
EXT_TACK        Inp     // slow peripheral transfer acknowledge
LCD_CLK         Inp     // External LCD clock input
HSYNC0          Inp     //
HSYNC1          Inp     //
HSYNC2          Inp     //
HSYNC3          Inp     //
VCLK0           Inp     //
VCLK1           Inp     //
VCLK2           Inp     //
VCLK3           Inp     //
MII_RXCLK       Inp     //
MII_RXD0        Inp     //
MII_RXD1        Inp     //
MII_RXD2        Inp     //
MII_RXD3        Inp     //
MII_RXDV        Inp     //
MII_RXER        Inp     //
MII_TXCLK       Inp     //
MII_CRS         Inp     //
MII_COL         Inp     //
MII_INT#        Inp     // PHY interrupt input
```

Sample File JTAG9360.INI:

```
// Description file for NetSilicon NS9360
Target: Generic Target, 2004/11/05
// 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
SDM_CS0#        Out,Hi  // SDRAM chip select
DATA0           Inp     // Data Bus for SDRAM, Static Memory, Peripherals
DATA1           Inp     //
DATA2           Inp     //
DATA3           Inp     //
DATA4           Inp     //
DATA5           Inp     //
DATA6           Inp     //
DATA7           Inp     //
DATA8           Inp     //
DATA9           Inp     //
DATA10          Inp     //
DATA11          Inp     //
DATA12          Inp     //
DATA13          Inp     //
DATA14          Inp     //
DATA15          Inp     //
DATA16          Inp     //
DATA17          Inp     //
DATA18          Inp     //
DATA19          Inp     //
DATA20          Inp     //
DATA21          Inp     //
DATA22          Inp     //
DATA23          Inp     //
DATA24          Inp     //
```

```
DATA25          Inp     //
DATA26          Inp     //
DATA27          Inp     //
DATA28          Inp     //
DATA29          Inp     //
DATA30          Inp     //
DATA31          Inp     //

// The following pins are tristateable outputs.
// These pins are tristateable outputs but can not be read back.
// Each pin can be disabled independent of the other pins.
// For Flash Programming these pins must be set to output
ADDR0           Out,Lo  // Data Bus for SDRAM, Static memory, Peripherals
ADDR1           Out,Lo  //
ADDR2           Out,Lo  //
ADDR3           Out,Lo  //
ADDR4           Out,Lo  //
ADDR5           Out,Lo  //
ADDR6           Out,Lo  //
ADDR7           Out,Lo  //
ADDR8           Out,Lo  //
ADDR9           Out,Lo  //
ADDR10          Out,Lo  //
ADDR11          Out,Lo  //
ADDR12          Out,Lo  //
ADDR13          Out,Lo  //
ADDR14          Out,Lo  //
ADDR15          Out,Lo  //
ADDR16          Out,Lo  //
ADDR17          Out,Lo  //
ADDR18          Out,Lo  //
ADDR19          Out,Lo  //
ADDR20          Out,Lo  //
ADDR21          Out,Lo  //
```

```
// 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.
RTCK_OUT        Inp     // Return Test Clock
RESET_DONE      Inp     // ???
GPIO0           Inp     // TXDB
GPIO1           Inp     // RXDB
GPIO2           Inp     // RTSB#
GPIO3           Inp     // CTSB#
GPIO4           Inp     // DTRB#
GPIO5           Inp     // DSRB#
GPIO6           Inp     // RIB#
GPIO7           Inp     // DCDB#
GPIO8           Inp     // TXDA
GPIO9           Inp     // RXDA
GPIO10          Inp     // RTSA#
GPIO11          Inp     // CTSA#
GPIO12          Inp     // DTRA#
GPIO13          Inp     // DSRA#
GPIO14          Inp     // RIA#
GPIO15          Inp     // DCDA#
GPIO16          Inp     // USB_OVRCURR
GPIO17          Inp     // USB_PWR
GPIO18          Inp     // ETH_
GPIO19          Inp     // ETH_
GPIO20          Inp     // DTRC#
GPIO21          Inp     // DSRC#
GPIO22          Inp     // RIC#
GPIO23          Inp     // DCDC#
GPIO24          Inp     // DTRD#
GPIO25          Inp     // DSRD#
GPIO26          Inp     // RID#
GPIO27          Inp     // DCDD#
GPIO28          Inp     // EXT_IRQ1
GPIO29          Inp     //
GPIO30          Inp     //
GPIO31          Inp     //
GPIO32          Inp     // EXT_IRQ2
GPIO33          Inp     //
GPIO34          Inp     // IIC_SCL
GPIO35          Inp     // IIC_SDA
GPIO36          Inp     // PWM0
GPIO37          Inp     // PWM1
```

```
GPIO38          Inp     // PWM2
GPIO39          Inp     // PWM3
GPIO40          Inp     // TXDC
GPIO41          Inp     // RXDC
GPIO42          Inp     // RTSC#
GPIO43          Inp     // CTSC#
GPIO44          Inp     // TXDD
GPIO45          Inp     // RXDD
GPIO46          Inp     // RTSD#
GPIO47          Inp     // CTSD#
GPIO48          Inp     // USB_SUSP
GPIO49          Inp     // USB_PHYSPEED
GPIO50          Inp     // MII_MDIO/RMII_MDIO
GPIO51          Inp     // MII_RXDV
GPIO52          Inp     // MII_RXER/RMII_RXER
GPIO53          Inp     // MII_RXD0/RMII_RXD0
GPIO54          Inp     // MII_RXD1/RMII_RXD1/USB_PHYSPEED
GPIO55          Inp     // MII_RXD2/USB_PHYSPEED
GPIO56          Inp     // MII_RXD3/USB_RXD+
GPIO57          Inp     // MII_TXEN/RMII_TXEN/USB_RXD-
GPIO58          Inp     // MII_TXER/
GPIO59          Inp     // MII_TXD0/RMII_TXD0
GPIO60          Inp     // MII_TXD1/RMII_TXD1
GPIO61          Inp     // MII_TXD2
GPIO62          Inp     // MII_TXD3
GPIO63          Inp     // MII_COL
GPIO64          Inp     // MII_CRS/RMII_CRS_DV
GPIO65          Inp     // MII_INT#
GPIO66          Inp     // ADDR22
GPIO67          Inp     // ADDR23
GPIO68          Inp     // ADDR24/SDM_CLKEN0/EXT_IRQ0
GPIO69          Inp     // ADDR25/SDM_CLKEN1/EXT_IRQ1
GPIO70          Inp     // ADDR26/SDM_CLKEN2/IIC_SDA
GPIO71          Inp     // ADDR27/SDM_CLKEN3
GPIO72          Inp     // TA_STRB
```

```
// The following pins are tristateable outputs.
// These pins are tristateable outputs but can not be read back.
// Each pin can be disabled independent of the other pins.
WE#             Out,Hi  // Write Enable, SDRAM/Peripheral
EXT_OE#         Out,Hi  // Peripheral output enable
EXT_CS0#        Out,Hi  // Peripheral chip select
EXT_CS1#        Out,Hi  // Peripheral chip select
EXT_CS2#        Out,Hi  // Peripheral chip select
EXT_CS3#        Out,Hi  // Peripheral chip select
EXT_BE0#        Out,Lo  // Periperal Byte Enable
EXT_BE1#        Out,Lo  // Periperal Byte Enable
EXT_BE2#        Out,Lo  // Periperal Byte Enable
EXT_BE3#        Out,Lo  // Periperal Byte Enable
SDM_CLKOUT0     Out,Lo  // SDRAM clock output
SDM_CLKOUT1     Out,Lo  // SDRAM clock output
SDM_CLKOUT2     Out,Lo  // SDRAM clock output
SDM_CLKOUT3     Out,Lo  // SDRAM clock output
SDM_CS0#        Out,Hi  // SDRAM chip select
SDM_CS1#        Out,Hi  // SDRAM chip select
SDM_CS2#        Out,Hi  // SDRAM chip select
SDM_CS3#        Out,Hi  // SDRAM chip select
SDM_CAS#        Out,Hi  // SDRAM column address strobe
SDM_RAS#        Out,Hi  // SDRAM row address strobe
SDM_DQM0        Out,Lo  // SDRAM Data Mask
SDM_DQM1        Out,Lo  // SDRAM Data Mask
SDM_DQM2        Out,Lo  // SDRAM Data Mask
SDM_DQM3        Out,Lo  // SDRAM Data Mask
MII_MDC         Inp     //

// 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.
SDM_CLKIN0      Inp     //
MII_RXCLK       Inp     // RMII_REFCLK
MII_TXCLK       Inp     //
SRESET#         Inp     //
```

1.10. Supported flash devices

Type JTAG9xxx /LIST [optionlist]

to get a online list of all flash types which could be used with the /DEVICE= option. In addition newer flash devices are supported by using the option /CFI.

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

2. JTAG9xxx Parameter Description

When you start JTAG9xxx.EXE without any parameters the following help screen with all possible functions and options is displayed:

```
JTAG9xxx --- JTAG utility for NetSilicon NS9xxx
Copyright © FS FORTH-SYSTEME GmbH, Breisach
Version 4.xx of mm/dd/yyyy

Programming of Flash-EPROMs and hardware tests on targets with the
NetSilicon NS9xxx.

The JTAG-Booster is needed to connect the parallel port of the PC
to the JTAG port of the NetSilicon NS9xxx.

Usage: JTAG9xxx /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
        /PNAND      : Program a NAND Flash Device
        /RNAND      : Read a NAND Flash Device to file
        /VNAND      : Verify a NAND Flash Device with file
        /DUMPNAND   : Make a dump of NAND Flash Device
        /PSER       : Program an I2C/SPI/MicroWire Device with file
        /RSER       : Read an I2C/SPI/MicroWire Device to file
        /VSER       : Verify an I2C/SPI/MicroWire Device with file
        /DUMPSER    : Make a dump of an 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	/WIGGLER	/PLS
/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:

/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 JTAG9xxx.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 JTAG9xxx.INI”).
Note: The initialization file is not loaded for the functions /SAMPLE (chapter 2.11) and /SNAP (chapter 2.12).
Default: /INI=JTAG9xxx.INI

/LATTICE /WIGGLER /PLS
Besides the standard JTAG-Booster interface there are several simple "Parallel-Port-JTAG" interfaces supported. With this interfaces the programming performance, of course, is reduced.

/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 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 NetSilicon NS9xxx instruction register within the JTAG chain. In most cases this option is not needed.
Default: /IROFFS=0

/CPUPOS=
Specifies the position of the NetSilicon NS9xxx within the JTAG chain.
Default: /CPUPOS=0

2.1. Program a Flash Device

Usage: JTAG9xxx /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 1.10 “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 1.9 “Initialization file JTAG9xxx.INI”.)
Default: /CS1

/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 NetSilicon NS9xxx 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:

JTAG9xxx /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.

JTAG9xxx /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: JTAG9xxx /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 1.10 “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:

JTAG9xxx /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: JTAG9xxx /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 1.10 “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:

JTAG9xxx /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: JTAG9xxx /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: /CS1

[3]hhhhhh=number base is hex

Example:

JTAG9xxx /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: JTAG9xxx /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:

JTAG9xxx /PSER EEPROM.CFG /SERCLK=FLAG0 /SERDAT=FLAG1
This example loads the file EEPROM.CFG to a I²C EEPROM connected to the pins FLAG0 and FLAG1 of the NetSilicon NS9xxx

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

Usage: JTAG9xxx /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:

JTAG9xxx /RSER EEPROM.CFG /SERCLK=GP26 /SERDAT=GP27 /L=100
This example reads 256 bytes from a I²C EEPROM to the file EEPROM.CFG. The serial EEPROM is connected to the pins CP26 and GP27 of the NetSilicon NS9xxx.

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

Usage: JTAG9xxx /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 /SER (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 /SER (Chapter 2.5)

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

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

/LSB1ST
See function /PSER (Chapter 2.5)

Example:

JTAG9xxx /VSER EEPROM.CFG /SERCLK=GP26 /SERDAT=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 NetSilicon NS9xxx.

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

Usage: JTAG9xxx /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 /SER (Chapter 2.5)

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

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

/LSB1ST
See function /PSER (Chapter 2.5)

Example:

JTAG9xxx /DUMPSER /SERCLK=FLAG0 /SERDAT=FLAG1
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: JTAG9xxx /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 NetSilicon NS9xxx 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 JTAG9xxx.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:

JTAG9xxx /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: JTAG9xxx /PIN? /PIN=pinname [optionlist]

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

Please Note: Not every pin of the NetSilicon NS9xxx 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 JTAG9xxx.INI” can be used. If you type /PIN= without any pin declaration a list of the CPU pins is displayed.

Example:

JTAG9xxx /PIN? /PIN=RESET#
This example samples the reset pin of the NetSilicon NS9xxx.

2.11. Polling CPU pins while the CPU is running

Usage: JTAG9xxx /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 JTAG9xxx.INI” can be used. If you type /PIN= without any pin declaration a list of the CPU pins is displayed.

Example:

JTAG9xxx /SAMPLE /PIN=FLAG3
This example samples the state of the port pin FLAG3 while the NetSilicon NS9xxx is running.

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

Usage: JTAG9xxx /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 NetSilicon NS9750/NS9775

0 RTCK_OUT	0 PCI_CENTRAL#	0 RESET_DONE	0 PCI_INTD#
0 PCI_INTC#	0 PCI_INTB#	0 PCI_INTA#	0 PCI_GNT3#
0 PCI_REQ3#	0 PCI_GNT2#	0 PCI_REQ2#	0 PCI_GNT1#
0 PCI_REQ1#	0 PCI_RESET#	0 PCI_CLKOUT	0 PCI_CLKIN
0 PCI_AD31	0 PCI_AD30	0 PCI_AD29	0 PCI_AD28
0 PCI_AD27	0 PCI_AD26	0 PCI_AD25	0 PCI_AD24
0 PCI_CBE3#	0 PCI_IDSEL	0 PCI_AD23	0 PCI_AD22
0 PCI_AD21	0 PCI_AD20	0 PCI_AD19	0 PCI_AD18
0 PCI_AD17	0 PCI_AD16	0 PCI_CBE2#	0 PCI_FRAME#
0 PCI_IRDY#	0 PCI_TRDY#	0 PCI_DEVSEL#	0 PCI_STOP#
0 PCI_PERR#	0 PCI_SERR#	0 PCI_PAR	0 PCI_CBE1#
0 PCI_AD15	0 PCI_AD14	0 PCI_AD13	0 PCI_AD12
0 PCI_AD11	0 PCI_AD10	0 PCI_AD9	0 PCI_AD8
0 PCI_CBE0#	0 PCI_AD7	0 PCI_AD6	0 PCI_AD5
0 PCI_AD4	0 PCI_AD3	0 PCI_AD2	0 PCI_AD1
0 PCI_AD0	0 DATA0	0 DATA1	0 DATA2
0 DATA3	0 DATA4	0 DATA5	0 DATA6
0 DATA7	0 DATA8	0 DATA9	0 DATA10
0 DATA11	0 DATA12	0 DATA13	0 DATA14
0 DATA15	0 DATA16	0 DATA17	0 DATA18
0 DATA19	0 DATA20	0 DATA21	0 DATA22
0 DATA23	0 DATA24	0 DATA25	0 DATA26
0 DATA27	0 DATA28	0 DATA29	0 DATA30
0 DATA31	0 ADDR0	0 ADDR1	0 ADDR2
0 ADDR3	0 ADDR4	0 ADDR5	0 ADDR6
0 ADDR7	0 ADDR8	0 ADDR9	0 ADDR10
0 ADDR11	0 ADDR12	0 ADDR13	0 ADDR14
0 ADDR15	0 ADDR16	0 ADDR17	0 ADDR18
0 ADDR19	0 ADDR20	0 ADDR21	0 ADDR22
0 ADDR23	0 ADDR24	0 ADDR25	0 ADDR26
0 ADDR27	0 SDM_CLKOUT0	0 EXT_CS0#	0 EXT_CS1#
0 SDM_CLKOUT1	0 EXT_CS2#	0 EXT_CS3#	0 SDM_CS0#
0 SDM_CS1#	0 SDM_CS2#	0 SDM_CS3#	0 WE#
0 SDM_CLKOUT2	0 EXT_OE#	0 SDM_CAS#	0 SDM_CLKOUT3
0 SDM_RAS#	0 SDM_CLKIN0	0 EXT_BE0#	0 SDM_CLKIN1
0 SDM_CLKIN2	0 EXT_BE1#	0 EXT_BE2#	0 SDM_CLKIN3
0 EXT_BE3#	0 SDM_DQM0	0 SDM_DQM1	0 SDM_DQM2
0 SDM_DQM3	0 SDM_CKE0	0 SDM_CKE1	0 SDM_CKE2
0 SDM_CKE3	0 EXT_RP??	0 EXT_TACK	0 LCD_CLK

0 HSYNC0	0 HSYNC1	0 HSYNC2	0 HSYNC3
0 VIDEO_DATA0	0 VIDEO_DATA1	0 VIDEO_DATA2	0 VIDEO_DATA3
0 VSYNC0	0 VSYNC1	0 VSYNC2	0 VSYNC3
0 BP_STAT0	0 BP_STAT1	0 BP_STAT2	0 BP_STAT3
0 VCLK0	0 VCLK1	0 VCLK2	0 VCLK3
0 PRINT	0 MII_RXCLK	0 MII_RXD3	0 MII_RXD2
0 MII_RXD1	0 MII_RXD0	0 MII_RXDV	0 MII_RXER
0 MII_TXCLK	0 MII_TXD3	0 MII_TXD2	0 MII_TXD1
0 MII_TXD0	0 MII_TXEN	0 MII_TXER	0 MII_CRS
0 MII_COL	0 MII_MDC	0 MII_MDIO	0 MII_INT#
0 GPIO49	0 GPIO48	0 GPIO47	0 GPIO46
0 GPIO45	0 GPIO44	0 GPIO43	0 GPIO42
0 GPIO41	0 GPIO40	0 GPIO39	0 GPIO38
0 GPIO37	0 GPIO36	0 GPIO35	0 GPIO34
0 GPIO33	0 GPIO32	0 GPIO31	0 GPIO30
0 GPIO29	0 GPIO28	0 GPIO27	0 GPIO26
0 GPIO25	0 GPIO24	0 GPIO23	0 GPIO22
0 GPIO21	0 GPIO20	0 GPIO19	0 GPIO18
0 GPIO17	0 GPIO16	0 GPIO15	0 GPIO14
0 GPIO13	0 GPIO12	0 GPIO11	0 GPIO10
0 GPIO9	0 GPIO8	0 I2C_SCL	0 I2C_SDA
0 GPIO7	0 GPIO6	0 GPIO5	0 GPIO4
0 GPIO3	0 GPIO2	0 GPIO1	0 GPIO0

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. The Debug Interface of the NetSilicon NS9xxx is not used.

- Some board designs do not have a pullup resistors at the TSRT# signal. As there is a weak pull down on the NetSilicon NS9xxx chip, you will get the error message:
  "No devices found in JTAG chain or TDO pin stuck at high level".
  To avoid this error message you need to connect the TSRT# output of the JTAG-Booster with the TRST# input of the NetSilicon NS9xxx. Normally this is the case if you use the standard 8 pin JTAG-Booster connector.
  If you use the JT_ARM adapter (FS part number 360), the jumper JP2 must be set.

3.1. Implementation Information NetSilicon NS9750/NS9775

- The software assumes the following scheme for connecting the (NOR-) Flash-EPROM to the NetSilicon NS9750/NS9775. Please contact us, if you have used a different method.

NetSilicon NS9750/NS9775 signal	8 Bit (NOR-) Flash	16 Bit (NOR-) Flash	32 Bit (NOR-) Flash
EXT_CS0# EXT_CS1# EXT_CS2# EXT_CS3#	CS#	CS#	CS#
EXT_OE#	OE#	OE#	OE#
WE#	WE#	WE#	WE#
ADDR0..26	A0..26	-	-
ADDR0..26	-	A1..27	-
ADDR0..26	-	-	A2..28
DATA0..7	D0..7	-	-
DATA0..15	-	D0..15	-
DATA0..31	-	-	D0..31

1.) The NetSilicon NS9750/NS9775 does adjust the addresses driven out through the address lines according to the selected bus size, this means: While A0 selects between bytes if the bus size is 8 bit, A0 selects between 32 bit words if the bus size is 32 bit.

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

- The software assumes the following scheme for connecting the NAND Flash-EPROM to the NetSilicon NS9750/NS9775. Only 8 bit NAND flashes are supported. Please contact us, if you have used a different method.

NetSilicon NS9750/NS9775 signal	**8 Bit NAND Flash**
EXT_CS1#	FCE#
EXT_OE#	FRE#
WE#	FWE#
ADDR13	ALE
ADDR14	CLE
DATA0..7	D0..7
GPIO49	R/B#

1.) This scheme fits to the module A9M9750 from FS Forth-Systeme.

2.) EXT_OE# and WE# are set according to the memory cycle.

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

3.2. Implementation Information NetSilicon NS9360

- The software assumes the following scheme for connecting the (NOR-) Flash-EPROM to the NetSilicon NS9360. Please contact us, if you have used a different method.

NetSilicon NS9360 signal	8 Bit (NOR-) Flash	16 Bit (NOR-) Flash	32 Bit (NOR-) Flash
EXT_CS0# EXT_CS1# EXT_CS2# EXT_CS3#	CS#	CS#	CS#
EXT_OE#	OE#	OE#	OE#
WE#	WE#	WE#	WE#
ADDR0..21	A0..21	-	-
ADDR0..21	-	A1..22	-
ADDR0..21	-	-	A2..23
DATA0..7	D0..7	-	-
DATA0..15	-	D0..15	-
DATA0..31	-	-	D0..31

1.) The NetSilicon NS9360 does adjust the addresses driven out through the address lines according to the selected bus size, this means: While A0 selects between bytes if the bus size is 8 bit, A0 selects between 32 bit words if the bus size is 32 bit.

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

3.) Flash size is actually limited to
4 MByte in 8 bit mode
8 MByte in 16 bit mode
16 MByte in 32 bit mode.
ADDR23..27 are multiplexed with GPIO66..71. Actually no address information is driven on these CPU pins.

- The software assumes the following scheme for connecting the NAND Flash-EPROM to the NetSilicon NS9360. Only 8 bit NAND flashes are supported. Please contact us, if you have used a different method.

NetSilicon NS9360 signal	8 Bit NAND Flash
EXT_CS1#	FCE#
EXT_OE#	FRE#
WE#	FWE#
ADDR13	ALE
ADDR14	CLE
DATA0..7	D0..7
GPIO??	R/B#

1.) This scheme fits to the module A9M9360 from FS Forth-Systeme.

2.) EXT_OE# and WE# are set according to the memory cycle.

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

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