JTAG-Booster for IDT RC64145



FORTH-SYSTEME GmbH

P.O. Box 11 03 ● D-79200 Breisach, Germany
Kueferstrasse 8 ● D-79206 Breisach, Germany
☎+49 (7667) 908-0 ● Fax +49 (7667) 908-200 ● e-mail: sales@fsforth.de

Copyright © 1995..2000:

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

Release of Document: February 08, 2000
Author: Dieter Fögele
Filename: JTAG145a.doc
Program Version: 3.00

All rights reserved. No part of this document may be copied or reproduced in any form or by any means without the prior written consent of FS FORTH-SYSTEME GmbH.

Table of Contents

1. General 4
   1.1. System Requirements 4
   1.2. Connecting your PC to the target system 5
   1.3. First Example 7
   1.4. Trouble Shooting 9
   1.5. Error Messages 10
   1.6. Initialization file JTAG145.INI 14
   1.7. Supported flash devices 20

2. JTAG145 Parameter Description 25
   2.1. Program a Flash Device 28
   2.2. Read a Flash Device to file 31
   2.3. Verify a Flash Device with file 33
   2.4. Dump target memory 35
   2.5. Program an I²C-Device 36
   2.6. Read an I²C-Device to file 38
   2.7. Verify an I²C-Device with file 40
   2.8. Dump an I²C-Device 42
   2.9. Toggle CPU pins 43
   2.10. Polling CPU pins 44
   2.11. Polling CPU pins while the CPU is running 45
   2.12. List supported Flash Devices 46

3. Implementation Information 47

1. General

The program JTAG145 uses the JTAG port of the IDT RC64145 embedded controller 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.

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

1.1. System Requirements

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

- MSDOS, WIN3.x, WIN9x or WinNT
  (WinNT is supported with an additional support-kit)

- Intel 80386 or higher

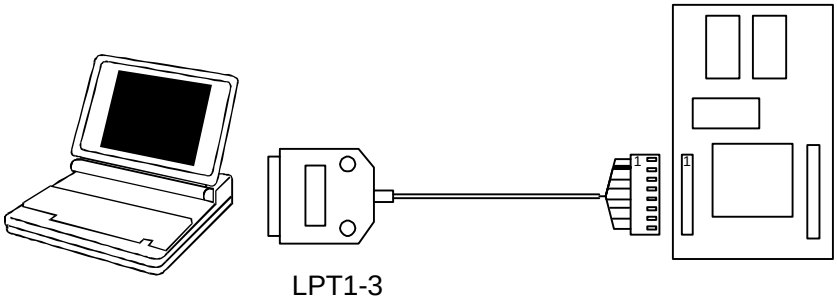
- 205 kByte of free DOS memory

- no extended memory

- Parallel Port

1.2. 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.

Since the JTAG-Booster operates with 5V, in addition a level shifter is necessary to connect the JTAG-Booster to a target operating with 3.3 volts. This level shifter must be ordered separately.

The IDT RC64145 does not have a TRST# pin. The reset of the JTAG interface is connected (internally to the IDT RC64145) to the system reset (Pin SYS_RST#). We recommend **not** to connect the system reset with the TRST# pin of the JTAG connector. For more information see chapter 3. "Implementation Information"

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:

JTAG145 /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 JTAG145, press CTRL-Break-Key.

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

1.3. First Example

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

```
JTAG145  /P  MYAPP.BIN  /VERIFY
```

at the DOS prompt results in the following output:

```
JTAG145 --- JTAG utility for IDT RC64145 (32bit)
Copyright © FS FORTH-SYSTEME GmbH, Breisach
Version 3.00 of mm/dd/yyyy

(1)   Configuration loaded from file JTAG145.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=00004067   IDT RC64145, Revision 0
(6)   Sum of instruction register bits  : 4
(7)   CPU position                      : 0
(8)   Instruction register offset       : 0

(9)   AMD 29LV640 (word mode) detected
(10)  Erasing Flash-EPROM Block #:0
      Programming File MYAPP.BIN
      65536 Bytes programmed
      Programming ok

      Erase Time    :     2.5 sec
      Program Time  :     60.7 sec
```

(1) The initialization file JTAG145.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 IDT RC64145 are switched to bypass mode.

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

(7) The position of the IDT RC64145 in the JTAG chain is checked.

(8) The position of the JTAG instruction register of the IDT RC64145 is checked

(9) One Flash-EPROMs AMD 29LV640 selected with chip select BOOTCS# is found.

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

1.4. 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.

When using older flash devices (nearly maximum erase cycles reached), we propose to use the /VERIFY option. This is also true for the relatively slow 3 Volt only flash devices.

Some newer fast flash devices need a setup time between address/data and the write strobe signal. If programming of this devices fails, try with the option /WRSETUP again.

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

Don't use hardware protected flash memories.

The used chip selects must be defined as output and inactive in the initialization file (see chapter 1.6 “Initialization file JTAG145.INI”). Also the address bits must be defined as output.

1.5. 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\JTAG145.OVL**
  The overlay file JTAG145.OVL must be in the same directory as JTAG145.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.

- **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 “JTAG145 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 “JTAG145 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 1.6 ”Initialization file JTAG145.INI”

- **illegal Flash Type:**
  The name specified with the option /DEVICE= must be one of the list of chapter 1.7 ”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 1.6 “Initialization file JTAG145.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 IDT RC64145**
  The option /CPUPOS= points to a part not a IDT RC64145 (32bit). The part may be switched to the 64bit JTAG mode.

- **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 1.6 "Initialization file JTAG145.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 1.6 “Initialization file JTAG145.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 1.6 “Initialization file JTAG145.INI”.

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

- **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 IDT RC64145 in the JTAG chain**
  No IDT RC64145 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= .

1.6. Initialization file JTAG145.INI

This file is used to set the CPU signals for input/output. In case of output signal an additional parameter is used to set the default level to high or low. This file can be used to adapt your own IDT RC64145 design to the JTAG-BOOSTER. The Target-Entry is used to identify your design which is displayed with most commands.

When the JTAG145 software is started it scans the current directory for an existing initialization file named JTAG145.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.

Example of JTAG145.INI:

```
// Description file for IDT RC64145
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.

// 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.
CPU_AD31        Inp     // Address/Data to/from CPU
CPU_AD30        Inp     //
CPU_AD29        Inp     //
CPU_AD28        Inp     //
CPU_AD27        Inp     //
CPU_AD26        Inp     //
CPU_AD25        Inp     //
CPU_AD24        Inp     //
CPU_AD23        Inp     //
CPU_AD22        Inp     //
CPU_AD21        Inp     //
CPU_AD20        Inp     //
CPU_AD19        Inp     //
CPU_AD18        Inp     //
```

```
CPU_AD17        Inp     //
CPU_AD16        Inp     //
CPU_AD15        Inp     // Address/Data to/from CPU
CPU_AD14        Inp     //
CPU_AD13        Inp     //
CPU_AD12        Inp     //
CPU_AD11        Inp     //
CPU_AD10        Inp     //
CPU_AD9         Inp     //
CPU_AD8         Inp     //
CPU_AD7         Inp     //
CPU_AD6         Inp     //
CPU_AD5         Inp     //
CPU_AD4         Inp     //
CPU_AD3         Inp     //
CPU_AD2         Inp     //
CPU_AD1         Inp     //
CPU_AD0         Inp     //
CPU_CMD8        Inp     // Command/Data Identifier to/from CPU
CPU_CMD7        Inp     //
CPU_CMD6        Inp     //
CPU_CMD5        Inp     //
CPU_CMD4        Inp     //
CPU_CMD3        Inp     //
CPU_CMD2        Inp     //
CPU_CMD1        Inp     //
CPU_CMD0        Inp     //
PCI_AD31        Inp     // PCI Address/Data
PCI_AD30        Inp     //
PCI_AD29        Inp     //
PCI_AD28        Inp     //
PCI_AD27        Inp     //
PCI_AD26        Inp     //
PCI_AD25        Inp     //
PCI_AD24        Inp     //
PCI_AD23        Inp     //
PCI_AD22        Inp     //
PCI_AD21        Inp     //
PCI_AD20        Inp     //
PCI_AD19        Inp     //
PCI_AD18        Inp     //
PCI_AD17        Inp     //
PCI_AD16        Inp     //
```

```
PCI_AD15        Inp     //
PCI_AD14        Inp     //
PCI_AD13        Inp     //
PCI_AD12        Inp     //
PCI_AD11        Inp     //
PCI_AD10        Inp     //
PCI_AD9         Inp     //
PCI_AD8         Inp     //
PCI_AD7         Inp     //
PCI_AD6         Inp     //
PCI_AD5         Inp     //
PCI_AD4         Inp     //
PCI_AD3         Inp     //
PCI_AD2         Inp     //
PCI_AD1         Inp     //
PCI_AD0         Inp     //
PCI_CBE3#       Inp     // PCI Command/Byte Enable
PCI_CBE2#       Inp     // PCI Command/Byte Enable
PCI_CBE1#       Inp     // PCI Command/Byte Enable
PCI_CBE0#       Inp     // PCI Command/Byte Enable
PCI_FRAME#      Inp     // PCI Cycle Frame
PCI_IRDY#       Inp     // PCI Initiator Ready
PCI_TRDY#       Inp     // PCI Target Ready
PCI_DEVSEL#     Inp     // PCI Device Select
PCI_STOP#       Inp     // PCI Stop
PCI_PERR#       Inp     // PCI Parity Error
PCI_PAR         Inp     // PCI Parity Bit
PIO12           Inp     //
PIO11           Inp     //
PIO10           Inp     //
PIO9            Inp     //
DMA_REQ3#       Inp     // DMA Request Input = PIO7
DMA_REQ2#       Inp     // DMA Request Input = PIO6
DMA_REQ1#       Inp     // DMA Request Input = PIO5
DMA_REQ0#       Inp     // DMA Request Input = PIO4
SYS_WAIT        Inp     // Wait          = PIO8
DMA_ACK3#       Out,Hi  // DMA Acknowledge Output = PIO3
DMA_ACK2#       Out,Hi  // DMA Acknowledge Output = PIO2
DMA_ACK1#       Out,Hi  // DMA Acknowledge Output = PIO1
DMA_ACK0#       Out,Hi  // DMA Acknowledge Output = PIO0
SYS_DATA31      Inp     // System Interface Data
SYS_DATA30      Inp     //
SYS_DATA29      Inp     //
```

```
SYS_DATA28          Inp     //
SYS_DATA27          Inp     //
SYS_DATA26          Inp     //
SYS_DATA25          Inp     //
SYS_DATA24          Inp     //
SYS_DATA23          Inp     //
SYS_DATA22          Inp     //
SYS_DATA21          Inp     //
SYS_DATA20          Inp     //
SYS_DATA19          Inp     //
SYS_DATA18          Inp     //
SYS_DATA17          Inp     //
SYS_DATA16          Inp     //
SYS_DATA15          Inp     //
SYS_DATA14          Inp     //
SYS_DATA13          Inp     //
SYS_DATA12          Inp     //
SYS_DATA11          Inp     //
SYS_DATA10          Inp     //
SYS_DATA9           Inp     //
SYS_DATA8           Inp     //
SYS_DATA7           Inp     //
SYS_DATA6           Inp     //
SYS_DATA5           Inp     //
SYS_DATA4           Inp     //
SYS_DATA3           Inp     //
SYS_DATA2           Inp     //
SYS_DATA1           Inp     //
SYS_DATA0           Inp     //

// The following pin is tristateable, but can not be read back
PCI_REQ#            Out,Hi  // PCI Bus Request

// The following pins are output only pins.
// Setting to input (tristate) one of that pins results in an error.
VALIDIN#            Out,Hi  // RC64145 drives valid Data
CPU_RDY#            Out,Hi  // Ready, connects to the CPU
CPU_INT#            Out,Hi  // Interrupt Output to CPU
PCI_SERR#           Out,Hi  // PCI System Error, open drain output
PCI_INT#            Out,Hi  // PCI Interrupt Request, open drain output
UART0_SOUT          Out,Hi  // UART0 Serial Data Out
UART0_RTS#          Out,Hi  // UART0 Request to Send
UART0_DTR#          Out,Hi  // UART0 Data Terminal Ready
```

```
UART1_SOUT      Out,Hi   // UART1 Serial Data Out
SYS_ALE         Out,Lo   // Address Latch Enable, High->Latch transparent
SPI_CS#         Out,Hi   // SPI Chip Select, No connect???
SPI_MOSI        Out,Lo   // SPI Master Out Slave In, No connect????
SPI_CLK         Out,Lo   // SPI Clock, No connect????
XCVRDIR_WR#     Out,Hi   // External Transceiver Direction
XCVROE#         Out,Hi   // External Transceiver Output Enable
SYS_BOOTCS#     Out,Hi   // Boot Chip Select
SYS_DEVCS0#     Out,Hi   // Device Chip Select
SYS_DEVCS1#     Out,Hi   // Device Chip Select
SYS_DEVCS2#     Out,Hi   // Device Chip Select
SYS_DEVCS3#     Out,Hi   // Device Chip Select
SYS_DEVOE#      Out,Hi   // Device Output Enable
SD_WE#          Out,Hi   // SDRAM Write Enable
SD_CAS#         Out,Hi   // SDRAM CAS#
SD_RAS#         Out,Hi   // SDRAM RAS#
SD_BA1          Out,Lo   // SDRAM Bank Address
SD_BA0          Out,Lo   // SDRAM Bank Address
SD_CS3#         Out,Hi   // SDRAM Chip Select
SD_CS2#         Out,Hi   // SDRAM Chip Select
SD_CS1#         Out,Hi   // SDRAM Chip Select
SD_CS0#         Out,Hi   // SDRAM Chip Select
SYS_DQM_WE3#    Out,Hi   // Data Mask
SYS_DQM_WE6#    Out,Hi   // Data Mask
SYS_DQM_WE1#    Out,Hi   // Data Mask
SYS_DQM_WE4#    Out,Hi   // Data Mask
SYS_ADR11       Out,Lo   //
SYS_ADR10       Out,Lo   //
SYS_ADR9        Out,Lo   //
SYS_ADR8        Out,Lo   //
SYS_ADR7        Out,Lo   //
SYS_ADR6        Out,Lo   //
SYS_ADR5        Out,Lo   //
SYS_ADR4        Out,Lo   //
SYS_ADR3        Out,Lo   //
SYS_ADR2        Out,Lo   //
SYS_ADR1        Out,Lo   //
SYS_ADR0        Out,Lo   //
```

```
// 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.
VALIDOUT#       Inp     // CPU drives valid Data
SYS_CLK         Inp     // Input Clock to RC64145
PCI_IDSEL       Inp     // PCI Initialization Device Select
PCI_LOCK#       Inp     // PCI Locked Cycle
PCI_GNT#        Inp     // PCI Bus Grant
PCI_CLK         Inp     // PCI Clock
UART1_SIN       Inp     // UART1 Serial Data In
UART0_CTS#      Inp     // UART0 Clear to Send
UART0_DCD#      Inp     // UART0 Data Carrier Detect
UART0_SIN       Inp     // UART0 Serial Data In
SPI_MISO        Inp     // SPI Master In Slave Out, VCC CORE????
```

This example is equal to the default initialization which is used 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 //.

1.7. Supported flash devices

The following names could be used with the /DEVICE= option:

AM29F010	: AMD 29F010	
AM29F010*2	: Dual AMD 29F010	
* AM29LV010	: AMD 29LV010	(3.3V)
* AM29LV010*2	: Dual AMD 29LV010	(3.3V)
* AM29F100BW	: AMD 29F100B	(word mode)
* AM29F100BB	: AMD 29F100B	(byte mode)
* AM29F100TW	: AMD 29F100T	(word mode)
* AM29F100TB	: AMD 29F100T	(byte mode)
* AM29F200BW	: AMD 29F200B	(word mode)
* AM29F200BB	: AMD 29F200B	(byte mode)
* AM29F200TW	: AMD 29F200T	(word mode)
* AM29F200TB	: AMD 29F200T	(byte mode)
* AM29LV200BW	: AMD 29F200B	(3.3V, word mode)
* AM29LV200BB	: AMD 29F200B	(3.3V, byte mode)
* AM29LV200TW	: AMD 29F200T	(3.3V, word mode)
* AM29LV200TB	: AMD 29F200T	(3.3V, byte mode)
AM29F040	: AMD 29F040	
AM29F040*2	: Dual AMD 29F040	
* AM29F400BW	: AMD 29F400B	(word mode)
* AM29F400BB	: AMD 29F400B	(byte mode)
AM29F400TW	: AMD 29F400T	(word mode)
AM29F400TB	: AMD 29F400T	(byte mode)
* AM29LV400BW	: AMD 29LV400B	(3.3V, word mode)
* AM29LV400BB	: AMD 29LV400B	(3.3V, byte mode)
* AM29LV400TW	: AMD 29LV400T	(3.3V, word mode)
* AM29LV400TB	: AMD 29LV400T	(3.3V, byte mode)
* AM29LV004B	: AMD 29LV004B	(3.3V)
* AM29LV004T	: AMD 29LV004T	(3.3V)
AM29F080	: AMD 29F080	
* AM29F080*2	: Dual AMD 29F080	
* AM29LV081	: AMD 29LV081	(3.3V)
* AM29LV081*2	: Dual 29LV081	(3.3V)
* AM29F800BW	: AMD 29F800B	(word mode)
* AM29F800BB	: AMD 29F800B	(byte mode)
AM29F800TW	: AMD 29F800T	(word mode)
* AM29F800TB	: AMD 29F800T	(byte mode)
* AM29LV800BW	: AMD 29LV800B	(3.3V, word mode)
* AM29LV800BB	: AMD 29LV800B	(3.3V, byte mode)

AM29LV800TW	: AMD 29LV800T	(3.3V, word mode)
* AM29LV800TB	: AMD 29LV800T	(3.3V, byte mode)
* AM29LV008T	: AMD 29LV008T	(3.3V)
* AM29LV008B	: AMD 29LV008B	(3.3V)
AM29F016	: AMD 29F016	
AM29F016*2	: Dual AMD 29F016	
* AM29F017	: AMD 29F017	
AM29F017*2	: Dual AMD 29F017	
AM29LV017*2	: Dual AMD 29LV017	(3.3V)
* AM29F160BW	: AMD 29F160B	(word mode)
* AM29F160BB	: AMD 29F160B	(byte mode)
* AM29F160TW	: AMD 29F160T	(word mode)
* AM29F160TB	: AMD 29F160T	(byte mode)
* AM29F160BB*2	: Dual AMD 29F160B	(byte mode)
* AM29F160TB*2	: Dual AMD 29F160T	(byte mode)
AM29LV160BW	: AMD 29LV160B	(3.3V, word mode)
* AM29LV160BB	: AMD 29LV160B	(3.3V, byte mode)
* AM29LV160TW	: AMD 29LV160T	(3.3V, word mode)
* AM29LV160TB	: AMD 29LV160T	(3.3V, byte mode)
AM29LV160BB*2	: Dual AMD 29LV160B	(3.3V, byte mode)
* AM29LV160TB*2	: Dual AMD 29LV160T	(3.3V, byte mode)
AM29F032	: AMD 29F032	
AM29F032*2	: Dual AMD 29F032	
* AM29LV033	: AMD 29LV033	(3.3V)
* AM29LV033*2	: Dual AMD 29F033	(3.3V)
* AM29LV640	: AMD 29LV160	(3.3V, word mode)
* I28F001T	: Intel 28F001T	
* I28F001T*2	: Dual Intel 28F001T	
* I28F001B	: Intel 28F001B	
* I28F200BW	: Intel 28F200B	(word mode)
* I28F200BB	: Intel 28F200B	(byte mode)
* I28F200TW	: Intel 28F200T	(word mode)
* I28F200TB	: Intel 28F200T	(byte mode)
* I28F002B	: Intel 28F002B	
* I28F002T	: Intel 28F002T	
* I28F400BW	: Intel 28F400B	(word mode)
* I28F400BB	: Intel 28F400B	(byte mode)
I28F400TW	: Intel 28F400T	(word mode)
I28F400TB	: Intel 28F400T	(byte mode)
* I28F004B	: Intel 28F004B	
* I28F004T	: Intel 28F004T	
* I28F400B3B	: Intel 28F400B3B	Boot Block Smart3 Bottom

* I28F400B3T	: Intel 28F400B3T	Boot Block Smart3 Top
* I28F004S5	: Intel 28F004	FlashFile Smart3/5
* I28F800BW	: Intel 28F800B	(word mode)
* I28F800BB	: Intel 28F800B	(byte mode)
I28F800TW	: Intel 28F800T	(word mode)
* I28F800TB	: Intel 28F800T	(byte mode)
* I28F800B3B	: Intel 28F800B3B	Boot Block Smart3 Bottom
* I28F800B3T	: Intel 28F800B3T	Boot Block Smart3 Top
* I28F008B	: Intel 28F008B	
* I28F008T	: Intel 28F008T	
I28F008SA	: Intel 28F008	FlashFile 12V
I28F008SA*2	: Dual Intel 28F008	FlashFile 12V
I28F008S5	: Intel 28F008	FlashFile Smart3/5
I28F008S5*2	: Dual Intel 28F008	FlashFile Smart3/5
I28F016W	: Intel 28F016	FlashFile (word mode)
* I28F016B	: Intel 28F016	FlashFile (byte mode)
* I28F016S5	: Intel 28F016	FlashFile Smart3/5
* I28F160B3B	: Intel 28F160	Boot Block Smart3 Bottom
* I28F160B3T	: Intel 28F160	Boot Block Smart3 Top
* I28F160S5W	: Intel 28F160	FlashFile Smart5 (word mode)
* I28F160S5B	: Intel 28F160	FlashFile Smart5 (byte mode)
I28F320S5W	: Intel 28F320	FlashFile Smart3/5 (word mode)
* I28F320S5B	: Intel 28F320	FlashFile Smart3/5 (byte mode)
* I28F320J5W	: Intel 28F320	StrataFlash (word mode)
* I28F320J5B	: Intel 28F320	StrataFlash (byte mode)
* I28F320J3W	: Intel 28F320	StrataFlash (3.3V, word mode)
* I28F320J3B	: Intel 28F320	StrataFlash (3.3V, byte mode)
* I28F640J5W	: Intel 28F640	StrataFlash (word mode)
* I28F640J5B	: Intel 28F640	StrataFlash (byte mode)
* I28F640J3W	: Intel 28F640	StrataFlash (3.3V, word mode)
* I28F640J3B	: Intel 28F640	StrataFlash (3.3V, byte mode)
* I28F128J3W	: Intel 28F128	StrataFlash (3.3V, word mode)
* I28F128J3B	: Intel 28F128	StrataFlash (3.3V, byte mode)
* MBM29F200BW	: Fujitsu 29F200B	(word mode)
* MBM29F200BB	: Fujitsu 29F200B	(byte mode)
* MBM29F200TW	: Fujitsu 29F200T	(word mode)
* MBM29F200TB	: Fujitsu 29F200T	(byte mode)
* MBM29LV200BW	: Fujitsu 29LV200B	(3.3V, word mode)
* MBM29LV200BB	: Fujitsu 29LV200B	(3.3V, byte mode)
* MBM29LV200TW	: Fujitsu 29LV200T	(3.3V, word mode)
* MBM29LV200TB	: Fujitsu 29LV200T	(3.3V, byte mode)
* MBM29F002B	: Fujitsu 29F002B	

* MBM29F002SB	: Fujitsu 29F002SB	(TSOP40)
* MBM29LV002B	: Fujitsu 29LV002B	(3.3V)
* MBM29F002T	: Fujitsu 29F002T	
* MBM29F002ST	: Fujitsu 29F002ST	(TSOP40)
* MBM29LV002T	: Fujitsu 29LV002T	(3.3V)
* MBM29F040	: Fujitsu 29F040	
* MBM29F400BW	: Fujitsu 29F400B	(word mode)
* MBM29F400BB	: Fujitsu 29F400B	(byte mode)
* MBM29F400TW	: Fujitsu 29F400T	(word mode)
* MBM29F400TB	: Fujitsu 29F400T	(byte mode)
* MBM29LV400BW	: Fujitsu 29LV400B	(3.3V, word mode)
* MBM29LV400BB	: Fujitsu 29LV400B	(3.3V, byte mode)
* MBM29LV400TW	: Fujitsu 29LV400T	(3.3V, word mode)
* MBM29LV400TB	: Fujitsu 29LV400T	(3.3V, byte mode)
* MBM29LV004B	: Fujitsu 29LV004B	(3.3V)
* MBM29LV004T	: Fujitsu 29LV004T	(3.3V)
* MBM29F080	: Fujitsu 29F080	
* MBM29LV080	: Fujitsu 29LV080	(3.3V)
* MBM29F800BW	: Fujitsu 29F800B	(word mode)
* MBM29F800BB	: Fujitsu 29F800B	(byte mode)
* MBM29F800TW	: Fujitsu 29F800T	(word mode)
* MBM29F800TB	: Fujitsu 29F800T	(byte mode)
* MBM29LV800BW	: Fujitsu 29LV800B	(3.3V, word mode)
* MBM29LV800BB	: Fujitsu 29LV800B	(3.3V, byte mode)
* MBM29LV800TW	: Fujitsu 29LV800T	(3.3V, word mode)
* MBM29LV800TB	: Fujitsu 29LV800T	(3.3V, byte mode)
* MBM29LV008B	: Fujitsu 29LV008B	(3.3V)
* MBM29LV008T	: Fujitsu 29LV008T	(3.3V)
* MBM29F016	: Fujitsu 29F016	
* MBM29F016*2	: Dual Fujitsu 29F016	
* MBM29F017	: Fujitsu 29F017	
* MBM29F017*2	: Dual Fujitsu 29F017	
* MBM29F160BW	: Fujitsu 29LV160B	(word mode)
* MBM29F160BB	: Fujitsu 29LV160B	(byte mode)
* MBM29F160TW	: Fujitsu 29LV160T	(word mode)
* MBM29F160TB	: Fujitsu 29LV160T	(byte mode)
* MBM29F160BB*2	: Dual Fujitsu 29LV160B	(byte mode)
* MBM29F160TB*2	: Dual Fujitsu 29LV160T	(byte mode)
* MBM29LV160BW	: Fujitsu 29LV160B	(3.3V, word mode)
* MBM29LV160BB	: Fujitsu 29LV160B	(3.3V, byte mode)
* MBM29LV160TW	: Fujitsu 29LV160T	(3.3V, word mode)
* MBM29LV160TB	: Fujitsu 29LV160T	(3.3V, byte mode)
MBM29LV160BB*2	: Dual Fujitsu 29LV160B	(3.3V, byte mode)

* MBM29LV160TB*2	: Dual Fujitsu 29LV160T	(3.3V, byte mode)
MBM29F032	: Fujitsu 29F032	
MBM29F032*2	: Dual Fujitsu 29F032*2	
* M29F100BW	: ST 29F100B	(word mode)
* M29F100BB	: ST 29F100B	(byte mode)
* M29F100TW	: ST 29F100T	(word mode)
* M29F100TB	: ST 29F100T	(byte mode)
M29F200BW	: ST 29F200B	(word mode)
* M29F200BB	: ST 29F200B	(byte mode)
* M29F200TW	: ST 29F200T	(word mode)
* M29F200TB	: ST 29F200T	(byte mode)
* M29F040	: ST 29F040	
* M29F040*2	: Dual ST 29F040	
* M29W040	: ST 29W040	(3.3V)
* M29W040*2	: Dual ST 29W040	(3.3V)
* M29F400BW	: ST 29F400B	(word mode)
* M29F400BB	: ST 29F400B	(byte mode)
* M29F400TW	: ST 29F400T	(word mode)
* M29F400TB	: ST 29F400T	(byte mode)
* M29W400BW	: ST 29W400B	(3.3V, word mode)
* M29W400BB	: ST 29W400B	(3.3V, byte mode)
* M29W400TW	: ST 29W400T	(3.3V, word mode)
* M29W400TB	: ST 29W400T	(3.3V, byte mode)
* M29W004B	: ST 29W004B	(3.3V)
* M29W004T	: ST 29W004T	(3.3V)
* M29F800BW	: ST 29F800B	(word mode)
* M29F800BB	: ST 29F800B	(byte mode)
* M29F800TW	: ST 29F800T	(word mode)
* M29F800TB	: ST 29F800T	(byte mode)

The flash devices signed with ´*´ are not yet tested.

2. JTAG145 Parameter Description

When you start JTAG145.EXE without any parameters the following help screen with all possible functions and options is displayed:

```
JTAG145 --- JTAG utility for the IDT RC64145 (32bit)
Copyright © FS FORTH-SYSTEME GmbH, Breisach
Version 3.00 of mm/dd/yyyy

Programming of Flash-EPROMs and Debugging on targets with the
IDT RC64145.

The JTAG-Booster is needed to connect the parallel port of the PC
to the JTAG port of the IDT RC64145.

Usage: JTAG145 /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 CPU is running
    /LIST       : Print a list of supported Flash devices

Supported Options:
/BOOTCS     /DEVCS0     /DEVCS1     /DEVCS2     /DEVCS3
/NOCS       /WRSETUP    /VERIFY     /TOP        /PAUSE
/P          /NODUMP     /NOERASE    /LATTICE    /ERASEALL
/LPT1       /LPT2       /LPT3       /LPT-BASE   /16BIT
/8BIT       /NOMAN      /32H        /32L        /LENGTH=
/L=         /FILE-OFFSET=           /FO=        /OFFSET=
/O=         /DELAY=     /DEVICE-BASE=           /DB=
/DRIVER=    /DATA-MASK= /DM=        /IROFFS=    /CPUPOS=
/DEVICE=    /PIN=       /I2CCLK=    /I2CDAT=    /I2CDATI=
/I2CDATO=   /WATCH=     /OUT=       /INI=
```

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 JTAG145.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 1.6 “Initialization file JTAG145.INI”).
Default: /INI=JTAG145.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.

/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.

2.1. Program a Flash Device

Usage: JTAG145 /P filename [optionlist]

The specified file is programmed to the flash memory. Finally a complete verify is done, if the option /VERIFY is omitted. If the verify fails, the contents of the flash memory is written to a file with the extension DMP.

The type of the flash device is normally detected by the software. When autodetect fails you should use the /DEVICE= option to set the right flash device. The known devices are shown in chapter 1.7 “Supported flash devices”.

Options:

/DEVICE=devicename
The device is detected automatically by switching to autoselect mode. In case of trouble you should select the device by using this parameter to avoid autodetection.

/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]
By default a device start address of 3000000h (=48MByte) is used for accesses to the flash device.
Default: /DEVICE-BASE=3000000
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=

[1]hhhhhh=number base is hex

/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=

/VERIFY
If this option is specified, the result is verified on a cell by cell (cell=8bit or 16bit) basis instead of a complete verify after programming. If you want both a cell by cell verify and a complete verify after programming, please use a additional command line with the verify function. See chapter 2.3 “Verify a Flash Device with file”.

/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.

/BOOTCS /DEVCS0 /DEVCS1 /DEVCS2 /DEVCS3
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 1.6 “Initialization file JTAG145.INI”.)
Default: /BOOTCS

/NOCS
Use this option to switch all chip select signals off. This may be necessary if the device's chip select is generated via a normal decoder instead of using the IDT RC64145 chip select unit.

/32L /32H
The JTAG-Booster is not able to handle the interface to a dual 16bit ROM (32bit-ROM). With this two options you can select one 16bit half of a 32bit-ROM. The option /32L selects the low word and /32H selects the high word. Do not specify the option /8BIT together with /32L or /32H.

/WRSETUP
By default write cycles to the Flash EPROM start with address, data and write strobe set at the same time. With the option /WRSETUP you can force the program to generate a setup time for address and data with respect to the write strobe.

Examples:

JTAG145 /P BIOS.ROM /VERIFY /O=-400000
This example programs the file BIOS.ROM to the flash memory with an offset of 4 MByte to the top of the boot flash. This is the point, where the MIPS Core fetches the first instructions.

JTAG145 /P CE.EVN /VERIFY /32L /DEVCS0
This example programs the file CE.EVN to the lower word of a 32bit-ROM connected to DEVCS0#.

2.2. Read a Flash Device to file

Usage: JTAG145 /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 to set the right flash device. The known devices are shown in chapter 1.7 “Supported flash devices”.

Options:

/DEVICE=devicename
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=

/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).

[2]hhhhhh=number base is hex

/BOOTCS /DEVCS0 /DEVCS1 /DEVCS2 /DEVCS3
See function /P (Chapter 2.1)

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

/32L /32H
See function /P (Chapter 2.1)

/WRSETUP
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: JTAG145 /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: JTAG145 /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 to set the right flash device. The known devices are shown in chapter 1.7 “Supported flash devices”.

Options:

/DEVICE=devicename
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)

/BOOTCS /DEVCS0 /DEVCS1 /DEVCS2 /DEVCS3
See function /P (Chapter 2.1)

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

/32L /32H
See function /P (Chapter 2.1)

/WRSETUP
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: JTAG145 /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: JTAG145 /DUMP [optionlist]

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

Options:

/OFFSET=hhhhhh[3]
The memory dump starts at an offset of hhhhhh.
Default: /OFFSET=0
Abbreviation: /O=

/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=

/BOOTCS /DEVCS0 /DEVCS1 /DEVCS2 /DEVCS3
See function /P (Chapter 2.1)
Default: /BOOTCS

/8BIT /16BIT
Default: /16BIT

/32L /32H
See function /P (Chapter 2.1)

Example: JTAG145 /DUMP /BOOTCS

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

[3]hhhhhh=number base is hex

2.5. Program an I²C-Device

Usage: JTAG145 /PI2C filename [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:

/DEVICE-BASE=hhhhhh[4]
By default a device start address of 3000000h (=48MByte) is used for accesses to the flash device.
Default: /DEVICE-BASE=3000000
Abbreviation: /DB=

/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=

[4]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.
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:

```
JTAG145 /PI2C EEPROM.CFG /I2CCLK=PIO10 /I2CDAT=PIO11
```

This example loads the file EEPROM.CFG to a serial EEPROM connected to the pins PIO10 and PIO11 of the IDT RC64145

2.6. Read an I²C-Device to file

Usage: JTAG145 /RI2C filename /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:

/DEVICE-BASE=hhhhhh[5]
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)

[5]hhhhhh=number base is hex

Example:

```
JTAG145 /RI2C EEPROM.CFG /I2CCLK=PIO10 /I2CDAT=PIO11 /L=100
```

This example reads 256 bytes from a serial EEPROM to the file EEPROM.CFG. The serial EEPROM is connected to the pins PIO10 and PIO11 of the IDT RC64145.

2.7. Verify an I²C-Device with file

Usage: JTAG145 /VI2C filename [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:

/DEVICE-BASE= hhhhhh[6]
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)

[6]hhhhhh=number base is hex

Example:

```
JTAG145 /VI2C EEPROM.CFG /I2CCLK=PIO10 /I2CDAT=PIO11
```

This example verifies 256 bytes from a serial EEPROM with the file EEPROM.CFG. The serial EEPROM is connected to the pins PIO10 and PIO11 of the IDT RC64145.

2.8. Dump an I²C-Device

Usage: JTAG145 /DUMPI2C [optionlist]

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

Options:

/OFFSET=hhhhhh[7]
The memory dump starts at an offset of hhhhhh.
Default: /OFFSET=0
Abbreviation: /O=

/LENGTH=hhhhhh
Default: /LENGTH=100
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: JTAG145 /DUMPI2C /I2CCLK=PIO10 /I2CDAT=PIO11

This example makes a memory dump of the first 100h bytes of a serial EEPROM connected to the CPU.

[7]hhhhhh=number base is hex

2.9. Toggle CPU pins

Usage: JTAG145 /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.

Please Note: Not every pin of the IDT RC64145 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 1.6 “Initialization file JTAG145.INI” can be used. If you type /PIN= without any pin declaration a list of the controller pins is displayed.

/DELAY=dddddd[8]
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: JTAG145 /BLINK /PIN=DEVCS0# /DELAY=0

This example toggles the DEVCS0# pin very fast which can be followed by the use of an oscilloscope.

[8]dddddd=number base is decimal

2.10. Polling CPU pins

Usage: JTAG145 /PIN? /PIN=pinname [optionlist]

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

Please Note: Not every pin of the IDT RC64145 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 1.6 “Initialization file JTAG145.INI” can be used. If you type /PIN= without any pin declaration a list of the controller pins is displayed.

Example: JTAG145 /PIN? /PIN=RESET#

This example samples the reset pin of the IDT RC64145.

2.11. Polling CPU pins while the CPU is running

Usage: JTAG145 /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. Additional 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 1.6 “Initialization file JTAG145.INI” can be used. If you type /PIN= without any pin declaration a list of the CPU pins is displayed.

Example: JTAG145 /SAMPLE /PIN=PIO11

This example samples the state of the port pin PIO11 while the IDT RC64145 is running.

2.12. List supported Flash Devices

Usage: JTAG145 /LIST [optionlist]

This command lists all supported flash devices to screen if not redirected to file or device. Flash devices signed with ´*´ are not yet tested.

3. Implementation Information

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

- In the IDT RC64145 a system reset does also reset the JTAG interface. Observing the pin SYS_RST# (i.e. with function /PIN? or /SAMPLE) is not possible.

- The IDT RC64145 does not have a TRST# pin. The reset of the JTAG interface is connected (internally to the IDT RC64145) to the system reset (Pin SYS_RST#). We recommend **not** to connect the system reset with the TRST# pin of the JTAG connector. Otherwise the target is reset on any starting of the JTAG-Booster and the function /SAMPLE (see chapter 2.11) does not work.

- The JTAG-Booster uses the 32 Bit mode of the JTAG chain. To configure the JTAG interface a pullup resistor is needed at the TDI pin. At end of the system reset (signal SYS_RST#) the TDI pin is sampled to configure the length of the chain. The IDT 64145 should be the first part in the JTAG chain (nearest to TDI) to get best control of the TDI pin.

- If the IDT RC64145 is configured to the wrong mode, the JTAG booster aborts with an error message. In this case the target power should be cycled while the JTAG-Booster is not attached to the target.

- The pin SYS_DEVOE# must be connected to the output enable pin of the Flash-EPROM.

- The pin SYS_DQMWE1# must be connected to the write enable pin of the Flash-EPROM. The pin SYS_DQMWE0# is not available in the 32Bit JTAG chain and can not be used.

- Even if there are byte wide devices used, connect SYS_DQMWE1# to all write enable inputs of the Flash-EPROMs.

- In case of programming 8 Bit EPROMs there must be a switch/multiplexer on your board: For normal operation the signal SYS_DQMWE0# must be connected to the write enable input of the Flash-EPROM. For programming with the JTAG-Booster the signal SYS_DQMWE1# must be connected to the Flash-EPROM

- As a result of the latched addresses only Flash devices with short unlock vectors are supported. This means that the old AMD 29F010 which needs the unlock addresses 0x5555 and 0x2AAA are not supported, the newer versions AM29F010B have short vectors (0x555 and 0x2AAA) and can be programmed.

- Refer to the following table for connecting Flash-EPROMs to the IDT RC64145:

64145 signal	**8 Bit Flash**	**16 Bit Flash**	**32 Bit Flash**
SYS_DEVOE#	OE#	OE#	OE#
SYS_BOOTCS# **1)**	CS#	CS#	CS#
SYS_DQMWE1#	WE#	WE#	WE#
SYS_DATA0..7	D0..7	D0..7	D0..7
SYS_DATA8..15	-	D8..15	D8..15
SYS_DATA16..31	-	-	D16..31
SYS_ADR0..11	A0..11 **3)**	A0..11 **3)**	A0..11 **3)**
SYS_ADR12..23 **2)**	A12..A23	A12..23	A12..23

1) Or one of the other chip select signals: SYS_DEVCS0#, SYS_DEVCS1#, SYS_DEVCS2#, SYS_DEVCS3#

2) These signals are the latched SYS_ADR0..11.

3) SYS_ADR0..23 represent always a cell address. SYS_ADR0 should always be connected to the LSB of the device address, regardless of bank width.