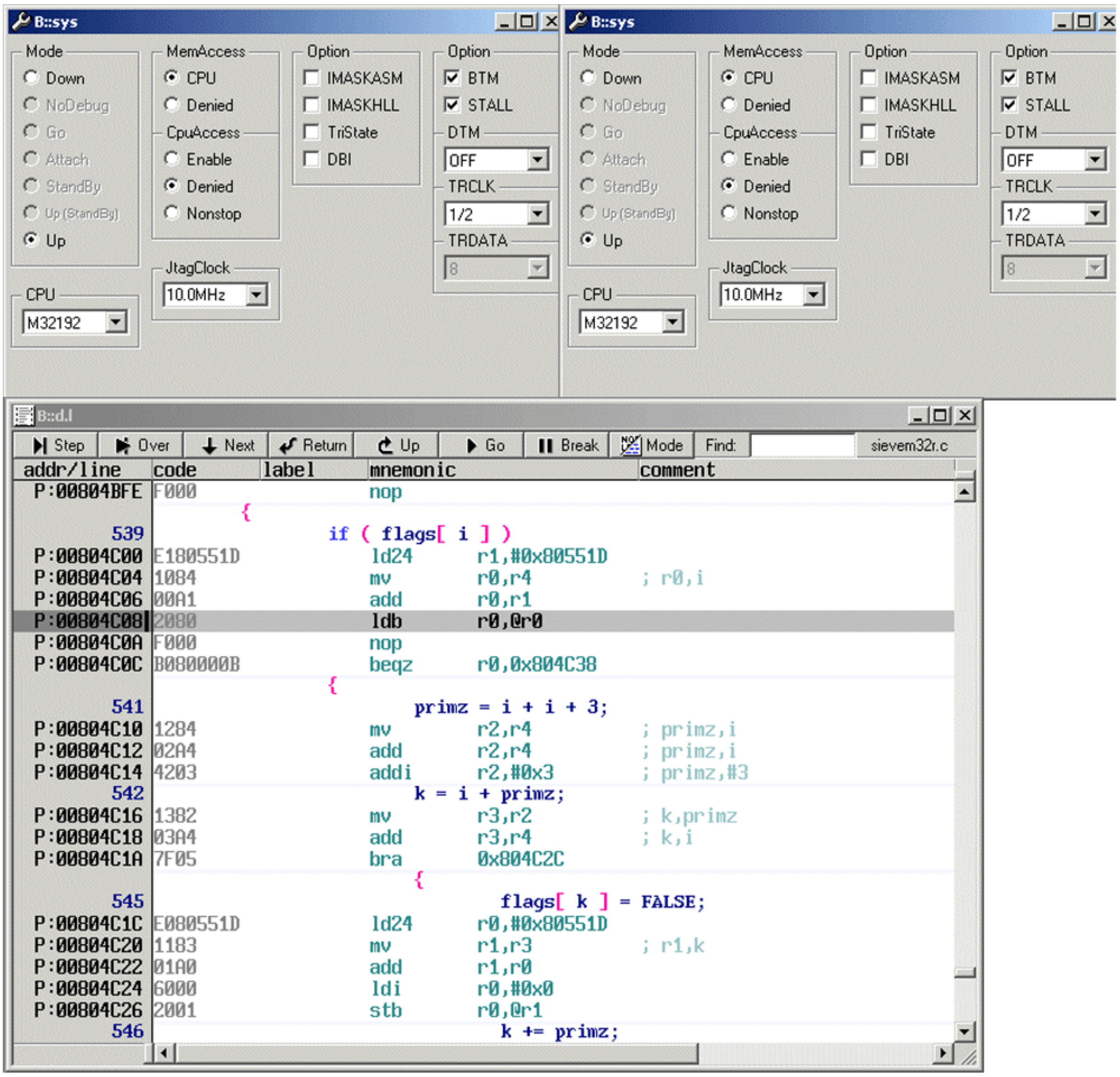
LAUTERBACH
DEVELOPMENT TOOLS

M32R Debugger and Trace

M32R Debugger and Trace

M32R Debugger and Trace

Version 06-Jun-2024



B::sys

Mode: Down, NoDebug, Go, Attach, StandBy, Up (StandBy), [✓ Up]

MemAccess: [✓ CPU], Denied

CpuAccess: Enable, [✓ Denied], Nonstop

Option: IMASKASM ☐, IMASKHLL ☐, TriState ☐, DBI ☐

Option: BTM ✓, STALL ✓

DTM: OFF

TRCLK: 1/2

TRDATA: 8

JtagClock: 10.0MHz

CPU: M32192

B::sys

Mode: Down, NoDebug, Go, Attach, StandBy, Up (StandBy), [✓ Up]

MemAccess: [✓ CPU], Denied

CpuAccess: Enable, [✓ Denied], Nonstop

Option: IMASKASM ☐, IMASKHLL ☐, TriState ☐, DBI ☐

Option: BTM ✓, STALL ✓

DTM: OFF

TRCLK: 1/2

TRDATA: 8

JtagClock: 10.0MHz

CPU: M32192

B::d.l

```
addr/line   code      label  mnemonic                        comment
 P:00804BFE F000             nop
                     {
        539                  if ( flags[ i ] )
 P:00804C00 E180551D         ld24     r1,#0x80551D
 P:00804C04 1084             mv       r0,r4                  ; r0,i
 P:00804C06 00A1             add      r0,r1
 P:00804C08 2080             ldb      r0,@r0
 P:00804C0A F000             nop
 P:00804C0C B080000B         beqz     r0,0x804C38
                             {
        541                      primz = i + i + 3;
 P:00804C10 1284             mv       r2,r4                  ; primz,i
 P:00804C12 02A4             add      r2,r4                  ; primz,i
 P:00804C14 4203             addi     r2,#0x3                ; primz,#3
        542                      k = i + primz;
 P:00804C16 1382             mv       r3,r2                  ; k,primz
 P:00804C18 03A4             add      r3,r4                  ; k,i
 P:00804C1A 7F05             bra      0x804C2C
                                 {
        545                          flags[ k ] = FALSE;
 P:00804C1C E080551D         ld24     r0,#0x80551D
 P:00804C20 1183             mv       r1,r3                  ; r1,k
 P:00804C22 01A0             add      r1,r0
 P:00804C24 6000             ldi      r0,#0x0
 P:00804C26 2001             stb      r0,@r1
        546                          k += primz;
```

Introduction

This document describes the processor specific settings and features of the TRACE32 debugger for the following Renesas M32R CPU families:

- (SDI-3) M32192, M32195, M32196, M32185, M32186
- (SDI-2) M32176, M32180

Please keep in mind that only the **Processor Architecture Manual** (the document you are reading at the moment) is CPU specific, while all other parts of the online help are generic for all CPUs supported by Lauterbach. So if there are questions related to the CPU, the Processor Architecture Manual should be your first choice.

If some of the described functions, options, signals or connections in this Processor Architecture Manual are only valid for a single CPU or for specific families, the names of the families are added in brackets.

Brief Overview of Documents for New Users

Architecture-independent information:

- **"Training Basic Debugging"** (training_debugger.pdf): Get familiar with the basic features of a TRACE32 debugger.
- **"T32Start"** (app_t32start.pdf): T32Start assists you in starting TRACE32 PowerView instances for different configurations of the debugger. T32Start is only available for Windows.
- **"General Commands"** (general_ref_*<x>*.pdf): Alphabetic list of debug commands.

Architecture-specific information:

- **"Processor Architecture Manuals"**: These manuals describe commands that are specific for the processor architecture supported by your Debug Cable. To access the manual for your processor architecture, proceed as follows:
    - Choose **Help** menu > **Processor Architecture Manual**.
- **"OS Awareness Manuals"** (rtos_*<os>*.pdf): TRACE32 PowerView can be extended for operating system-aware debugging. The appropriate OS Awareness manual informs you how to enable the OS-aware debugging.

Demo and Start-up Scripts

Lauterbach provides ready-to-run start-up scripts for known M32R based hardware.

To search for PRACTICE scripts, do one of the following in TRACE32 PowerView:

- Type at the command line: **WELCOME.SCRIPTS**
- or choose **File** menu > **Search for Script**.

  You can now search the demo folder and its subdirectories for PRACTICE start-up scripts (*.cmm) and other demo software.

You can also manually navigate in the `~~/demo/m32r/` subfolder of the system directory of TRACE32.

Warning

WARNING:	To prevent debugger and target from damage it is recommended to connect or disconnect the Debug Cable only while the target power is OFF. Recommendation for the software start: 1. Disconnect the Debug Cable from the target while the target power is off. 2. Connect the host system, the TRACE32 hardware and the Debug Cable. 3. Power ON the TRACE32 hardware. 4. Start the TRACE32 software to load the debugger firmware. 5. Connect the Debug Cable to the target. 6. Switch the target power ON. 7. Configure your debugger e.g. via a start-up script. Power down: 1. Switch off the target power. 2. Disconnect the Debug Cable from the target. 3. Close the TRACE32 software. 4. Power OFF the TRACE32 hardware.

Quick Start

Starting up the debugger is done as follows:

1. Select the device prompt for the ICD Debugger and reset the system.

```
B::

RESet
```

   The device prompt `B::` is normally already selected in the TRACE32 command line. If this is not the case, enter `B::` to set the correct device prompt. The **RESet** command is only necessary if you do not start directly after booting the TRACE32 development tool.

2. Specify the CPU specific settings.

```
SYStem.CPU <cpu_type>
```

   The default values of all other options are set in such a way that it should be possible to work without modification. Please consider that this is probably not the best configuration for your target.

3. Set up data for electrical interface.

```
SYStem.JtagClock <frequency>
```

   Normally the default value is 10.0 MHz, but the it can be increased up to 25 MHz.

4. Inform the debugger about read only and none-readable address ranges (ROM, FLASH).

```
MAP.DenyAccess

MAP.NoDenyAccess <range>

MAP.BOnchip <range>
```

   The **BreakOnchip** information is necessary to decide where on-chip breakpoints must be used. On-chip breakpoints are necessary to set program breakpoints to FLASH/ROM. The sections of FLASH and ROM depend on the specific CPU and its chip selects. Accesses to invalid addresses can cause unrecoverable bus errors. To avoid bus errors from the debugger side use the subcommands of MAP to define inaccessible memory areas. Bus errors can be removed by executing SYStem.Up. Make sure that there isn't any TRACE32 window open which accesses to a inaccessible memory that is not masked out, otherwise the bus error can occur again.

5. Enter debug mode.

```
SYStem.Up
```

This command resets the CPU and enters debug mode. After this command is executed, it is possible to access memory and registers.

6. Configure chip according application.

Before loading binary data into the processor memory, the memory should be made writable for the debugger. Therefore processor configuration registers have to be set e.g. chip select register.

7. Load the program.

```
Data.LOAD.SR program.abs /Verify      ; SR specifies the format,
                                      ; program.abs is the file name
```

The format of the **Data.LOAD** command depends on the file format generated by the compiler. It is recommended to use the option /Verify that verifies all written data. This test discovers a problem with the electrical connection, wrong chip configurations or linker command file settings.

For a detailed description of the **Data.LOAD** command and all available options, see **"Data"** in **"General Commands Reference Guide D"** (general_ref_d.pdf).

A typical start sequence for the MSC8101 is shown below. This sequence can be written to a PRACTICE script file (*.cmm, ASCII format) and executed with the command **DO** *<file>*. Other sequences can be found in the directory ~~/demo/m32r.

```
B::                                  ; Select the ICD device prompt

WinClear                             ; Clear all windows

SYS.CPU M32196                       ; Select CPU

SYS.JC 15000000.                     ; Choose JTAG frequency

SYStem.Up                            ; Reset the target and enter debug
                                     ; mode

MAP.DENYACCESS                       ; Forbid any access to the memory in
                                     ; general

MAP.BONCHIP 0x0000--0x007FFF         ; Specifies the program memory where
; ROM                                ; on-chip breakpoints must be used

Data.LOAD.SR Sieve.abs /Verify       ; Load the application, verify the
                                     ; process

Go main                              ; Run and break at main()

List.Mix                             ; Open source window               *)

Register.view /SpotLight             ; Open register window             *)

Var.Local                            ; Open window with local variables *)
```

*) These commands open windows on the screen. The window position can be specified with the **WinPOS** command.

Troubleshooting

SYStem.Up Errors

The **SYStem.Up** command is the first command of a debug session where communication with the target is required. If you receive error messages while executing this command this may have the following reasons.

- The JTAG lines are not connected correctly.
- The target has no power.
- The pull-up resistor between the JTAG[VCCS] pin and the target VCC is too large.
- The target is in reset:

  The debugger controls the processor reset and use the RESET line to reset the CPU on every SYStem.Up. Therefore no external R-C combination or external reset controller is allowed.
- There is logic added to the JTAG state machine:

  By default the debugger supports only one processor in one JTAG chain. If the processor is only one member of a JTAG chain the debugger has to be informed about the target JTAG chain configuration. Use the SYStem.CONFIG command to specify the position of the device in the JTAG-chain.
- There are additional loads or capacities on the JTAG lines.

Memory Access Errors

After system up is completed successfully, data can be written to or read from memory. Trying to access memory not belonging to the memory map of the processor will be refused with the error message

```
no memory mapped at address           D:XXXXXXXX
```

and When a unrecoverable bus error occurs the target processor has to be reset.

```
bus error generated by CPU
```

FAQ

Please refer to https://support.lauterbach.com/kb.

CPU specific SYStem Settings and Restrictions

NOTE:	All trace related settings described here are only relevant, if the device provides trace capabilities!

Trace features can only be used, if a special device and /or a special adapter board (Pitch-Converter) is used. Both products are provided by Renesas.

SYStem.CONFIG — Configure debugger according to target topology

The **SYSTem.CONFIG** command group is not supported for the M32R.

SYStem.CPU — Select target CPU

Format:	**SYStem.CPU** *<cpu>*
<cpu>:	**M32192** | **M32192FPU** | **M32176** | **M32180**

Selects the processor type.

The processor type must be selected by the **SYStem.CPU** command before issuing any other target related commands.

SYStem.JtagClock   Define JTAG clock

Format:	**SYStem.JtagClock** [<*frequency*>] **SYStem.BdmClock** (deprecated)
<*frequency*>:	**6 kHz**…**25 MHz** **1250000.** | **2500000.** | **5000000.** | **10000000.**

Default frequency: 10 MHz.

Selects the JTAG port frequency (TCK) used by the debugger to communicate with the processor. The frequency affects e.g. the download speed. It could be required to reduce the JTAG frequency if there are buffers, additional loads or high capacities on the JTAG lines or if VTREF is very low. A very high frequency will not work on all systems and will result in an erroneous data transfer. Therefore we recommend to use the default setting if possible.

<*frequency*>	The debugger cannot select all frequencies accurately. It chooses the next possible frequency and displays the real value in the **SYStem.state** window. Besides a decimal number like “100000.’ short forms like”10kHz” or “15MHz” can also be used. The short forms imply a decimal value, although no “.” is used.

When the debugger is not working correctly (e.g. memory is flickering) decrease the JtagClock.

SYStem.LOCK   Lock and tristate the debug port

Format:	**SYStem.LOCK** [**ON** | **OFF**]

Default: OFF.

If the system is locked, no access to the debug port will be performed by the debugger. While locked, the debug connector of the debugger is tristated. The main intention of the **SYStem.LOCK** command is to give debug access to another tool.

SYStem.MemAccess — Select run-time memory access method

Format:	**SYStem.MemAccess** *<mode>*
<mode>:	**CPU** **StopAndGo** **Denied**

Default: Enable.

Enable **CPU** (deprecated)	Provides access to memory while the core is running.
StopAndGo	Temporarily halts the core(s) to perform the memory access. Each stop takes some time depending on the speed of the JTAG port, the number of the assigned cores, and the operations that should be performed. For more information, see below.
Denied	No access to memory while the core is running.

SYStem.Mode — Establish the communication with the target

Format:	**SYStem.Mode** <*mode*>
	SYStem.Down (alias for SYStem.Mode Down) **SYStem.Up** (alias for SYStem.Mode Up)
<*mode*>:	**Down** **Up**

Down	Disables the debugger (default). The state of the CPU remains unchanged. The JTAG port is tristated if **SYStem.Option.TriState** is checked.In other case the debugger drives JTAG signals and Reset.
Up	Resets the target, sets the CPU to debug mode and stops the CPU. After the execution of this command the CPU is stopped and all register are set to the default level.
Attach **Go** **StandBy**	Not available.

SYStem.Option — Display SYStem window

It has the same effect as **SYStem.state**

SYStem.Option.DBI — Enables program break via debug interrupt

Format:	**SYStem.Option.DBI** [**ON** | **OFF**]

Default: OFF.

When DBI is ON, the chip will stop faster rather than via SW control, provided the CPU offers DBI capability.

SYStem.Option.IMASKASM   Disable interrupts while single stepping

Format: **SYStem.Option.IMASKASM** [**ON** | **OFF**]

Default: OFF.

If enabled, the interrupt mask bits of the CPU will be set during assembler single-step operations. The interrupt routine is not executed during single-step operations. After single step the interrupt mask bits are restored to the value before the step.

SYStem.Option.IMASKHLL   Disable interrupts while HLL single stepping

Format: **SYStem.Option.IMASKHLL** [**ON** | **OFF**]

Default: OFF.

If enabled, the interrupt mask bits of the CPU will be set during HLL single-step operations. The interrupt routine is not executed during single-step operations. After single step the interrupt mask bits are restored to the value before the step.

Format:	**SYStem.Option.Keycode** [1…32 Byte keycode]

Default: 12 times 0xFF.

Some of the devices support Code Protection ID feature. Without a valid ID code, there is no access to the device by the debugger.

Use the **AREA** window to get further information about the Security status after startup.

Use the following sequence in all your startup scripts or enter it in the command line one time in order to get access to the device:

SYStem.Option.KEYCODE	up to 32 byte representing your keycode

By default use:

```
SYS.OPTION KEYCODE 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF\
0xFF 0xFF
```

If the device is blank, the debugger automatically uses 12 time 0xFF per default. Then no SYS.OPTION KEYCODE command is needed. The number and location of bytes depends on the use MCU. It is normally hard coded!

```
; Source code example for Renesas Compiler (CPU 32192, code location
; 0x00084)

.SECTION PROTECTID, DATA, ALIGN=1

; H'0000 0084 Protect ID
.DATA.B H'FF

.DATA.B H'FF,H'FF,H'FF,H'FF,H'FF,H'FF,H'FF

.DATA.B H'FF,H'FF,H'FF,H'FF
```

SYStem.Option.TriState — Allow debugger to drive JTAG and reset

Format:	**SYStem.Option.TriState** [**ON** | **OFF**]

Default: OFF.

If this option is OFF the JTAG signals and nRST line are never driven by the debugger.

SYStem.state — Display SYStem.state window

Format:	**SYStem.state**

Displays the **SYStem.state** window.

Trace specific Commands

SYStem.Option.BTM — Enables program trace messages

Format:	**SYStem.Option.BTM** [**ON** | **OFF**]

Default: ON.

The option can be switched when the chip has trace support. When BTM is ON, the chip delivers program trace messages.

SYStem.Option.DTM — Enables data trace messages

Format:	**SYStem.Option.DTM** [**OFF** | **Read** | **Write** | **ReadWrite**]

Default: OFF.

The option can be used if the chip has trace support. When the option is set to READ|WRITE|READWRITE, the CPU generates data trace messages, according to the selected access type.

SYStem.Option.STALL — Trace message overrun control

Format:	**SYStem.Option.STALL** [**ON** | **OFF**]

Default: OFF.

The option can be set when the chip has trace support and defines the behavior that becomes active when the chip intern trace message FIFO buffer gets full. Stall OFF will cause losing of messages when the buffer overruns.

SYStem.Option.TRCLK Trace output clock ratio

Format:	**SYStem.Option.TRCLK** [**1/8** | **1/4** | **1/3** | **1/2** | *** 1** | ***2** | ***3** | ***4**]

Default: 1/2.

The option can be set when the chip has trace support and defines the frequency of the trace output clock based on the processor frequency. High frequencies can cause electrical connection problems during the record of trace messages.

SYStem.Option.TRDATA Trace port width

Format:	**SYStem.Option.TRDATA** [**4** | **8**]

Default: 8.

The option can be set when the chip has trace support and defines port width of the trace data. The maximum is defined by the derivatives maximum trace pin count.

TrOnchip

The OCE unit of the M32R allows to set on-chip breakpoints. The registers are controlled by TRACE32. TRACE32 uses the on-chip trigger registers to perform on-chip breakpoints, which can be set in the Data.List window or in the dialog Breakpoint.Set. The current user interface of TRACE32 offers many possible configurations of the OCE unit. However the usable number of breakpoints is depending on the device.

- Up to 32 program address breakpoints

  (M32192: 4 program breakpoints)

- Up to 4 data address breakpoints

  (M32192 : 4 data breakpoints)

The amount of range breakpoints is limited that's why it is sometimes useful to set the **Break.CONFIG.InexactAddress** option. When enabled, this option let transform range breakpoints into normal, if necessary. The OCE can perform more operations than TRACE32 offers with it's user interface e.g. build a chain of breakpoints.

The on-chip trigger unit events can be also used to control the trace. The possible actions can be defined in the **Break.Set** dialog. To control the trace unit an appropriate action has to be chosen for the **Break.Set** command.

```
b.s flags /TraceData                    ; Set up a filter for Data Trace
                                        ; (only with DTM option set to on)

b.s flags /TraceEnable                  ; Set up a filter for Program Trace

b.s main /TraceTrigger                  ; Set Watchpoint message to generate
                                        ; Trigger for the Trace analyzer

b.s main /BusTrigger                    ; Set Watchpoint message to generate a
                                        ; trigger pulse on the PodBus

b.s main /BusCount                      ; Set Watchpoint message to allow
                                        ; frequency counter feature
```

On-chip Breakpoints can stimulate the EVENT0 … 2 pins. These signals can be used as input events for the Simple Trigger Unit (STU).

TrOnchip.RESet — Resets all TO settings

Format:	**TrOnchip.RESet**

Resets the trigger system to the default state.

Format:	**TrOnchip.state**

Control panel to configure the on-chip breakpoint and trace registers. The details are described in section **TrOnchip**.

Security Levels of the M32R Family

Security Level

Depending on the verification result and the security level, the following accesses to the device is possible:

- **Security ID code matches:** Any access is possibly, there are no limitations
- **Security ID code does not match:**

  Security Level 0: There is no access at all, even flashing is not possible. The debugger generates an error message and remains in down state.

  Security Level 1: The debugger reaches up state, but all read/write access to the Flash memory, to RAM, to registers and peripherals are blocked. No command will be accepted, except a special command to erase the complete Flash area. Any attempt to execute a command (except the Flash erase command) will cause an error message.

  The special Flash erase command takes care that just a complete erased Flash allows access to the device by the default Security ID code.

Flash Erase if Device is secured

How to manage Flash erase if Security Level 1 is activated and the security key code is unknown?

- Close all windows on the screen and perform a SYstem.up.
- Enter **diag 0x3000 0xF5** and wait until erasing is ready.
- Enter the default Security Code and system up the debugger or
  just system up the debugger (default Security Code is used implicitly).

Another way is to clear the flash memory of the CPU by using the instruction

…/unsecure.

Now the device is open with a cleared Flash RAM.

⚠	The JTAG clock must be limited to 1/2 of the M32 core clock.

⚠	Buffers, additional loads or high capacities on the JTAG/COP lines reduce the debug speed.

⚠	Trace related options only in case the device provides Trace capabilities.

System.Up duration	System.Up takes 1 … 2 s caused by the target CPU. If a new trace port width is selected (4->8 or 8->4), next SYStem.Up command takes a bit longer due to probe CPLD reprogramming.
ASM debugging in hardware loops - stepping	The debugger tries to step over delay slots. If the debugger is not successful, set a software breakpoint after the hardware loop and use go to step over the hardware loop.
HLL debugging in optimized code	HLL debugging in optimized code is restricted. Source lines may be assigned wrong, local variables may not be displayed.
Debugging with interrupts	When IMASKHLL or IMASKASM is enabled the debugger won't update correctly the interrupt disable bit in the SR register in case the code executed the DI instruction. Use **SYStem.Option.IPLDI** to switch the behavior.
Ignore RESET Monitoring	Normally the debugger monitors RESET and stops operation if RESET is asserted. If one wants to disable RESET monitoring, he has to enter **DIAG 0x3000 0xB1 1** To allow RESET monitoring again, enter **DIAG 0x3000 0xB1 0** (default after startup)
External Watchdog Timer	An external WDT must normally be turned off. For the case that it is not possible, there are 2 solutions. 1. For the case the WDT can be feed by toggling a CPU pin: **DIAG 0x3000 0xEA** *<pin>* **(Example: DIAG 0x3000 0xEA 124.)** **DIAG 0x3000 0xEB** [**0** | **1**] **(Example DIAG: 0x3000 0xEB 1** for on**)** 2. For the case the WDT must be feed by anyhow: Refer to **DATA.TIMER.SEQUENC**E and similar instructions **By default external WDT support is not enabled.**

Floating Point Formats

F24	Fractional fixed point 24 bit
F48	Fractional fixed point 48 bit
F16	Fractional fixed point 16 bit
F32	Fractional fixed point 32 bit

NOTE:	Fractional floating point numbers are always displays with a fixed precision, i.e. a fixed number of digits. Small fractional numbers can have many non relevant digits displayed.

Integer Access Keywords

Word	Word (16 bit)
TByte	Triple byte (24 bit)
Long	Double Word (32 bit), upper and lower word swapped
HByte	Hexabyte (48 bit)
Quad	Tertiary Word (64 bit), upper and lower word swapped

JTAG Connection

Mechanical Description of the 10-pin Debug Cable

This connector is defined by ARM and we recommend this connector for all future designs. Our debugger “JTAG Debugger for StarCore” (LA-7845) is supplied with this connector:

Signal	Pin	Pin	Signal
TCK	1	2	GND
TDI	3	4	TDO
TMS	5	6	TRST-
DBI	7	8	VCCTRB (trace buffer)
VCC	9	10	RST-

This is a standard 10 pin double row connector (pin-to-pin spacing: 0.100 in.). We strongly recommend to use a connector on your target with housing and having a center polarization (e.g. AMP: 2-827745-0). A connection the other way around indeed causes damage to the output driver of the debugger.

Electrical Description of the 10-pin Debug Cable

- The input and output signals are 3.3 V TTL compatible.

  VTREF is used as a sense line for the target voltage. It is also used as supply voltage for the supply translating transceiver of the ICD interface to make an adaptation to the target voltage (1.5 V) 1.8 … 3.3 V (3.6 V). On the newer debug cables (September 2003 and newer) it is used as sense line, only.
- $\overline{\text{RST}}$, TDI, TMS, TCK: In normal operation mode the driver is enabled, but it can be disabled to give another tool access to the JTAG port. In environments where multiple tools can access the JTAG port, it is required that there is a pull-up or pull-down resistor at TCK. This is to ensure that TCK maintains its level during a hand-over between different tools.
- TDO is an ICD input. It is connected to the supply translating transceiver.
- $\overline{\text{RST}}$ (reset) is used by the debugger to reset the target CPU or to detect a reset on the target. It is driven by an open collector buffer. A pull-up resistor is included in the ICD connector. The debugger will only assert a pulse on nSRST when the SYStem.UP, the SYStem.Mode Go or the SYStem.RESetOUT command is executed.
- DBI is an output which can force the CPU into debug mode by hardware.
- VCCTRB is an output and supplies the trace date buffer on the target. Normally 1.8 V.
- N/C (= Vsupply) is not connected in the ICD. This pin is used by debuggers of other manufacturers for supply voltage input. The ICD is self-powered.

Mechanical Description of the 20-pin Trace Connector

Signal	Pin	Pin	Signal
TRCK	1	2	VSS
TRSYNC	3	4	TRDATA0
TRDATA1	5	6	VSS
TRDATA2	7	8	TRDATA3
VSS	9	10	TRDATA4
TRDATA5	11	12	VSS
TRDATA6	13	14	TRDATA7
VCC	15	16	EVENT0
EVENT1	17	18	EVENT2
EVENT3	19	20	N/C

This connector is the standard for single M32R targets. For pure debug features, this connector is not needed. Not using this connector does not impact debug features at all.

Pins	Connection	Description	Recommendations
1	TRCLK	Trace Clock	
2	VSS	System Ground Plan	Connect to digital ground.
3	TRSYNC	TRace Sync.	Strobe for valid Trace data.
4	TRDATA0	Trace data 0	
5	TRDATA1	Trace data 1	
6	VSS	System Ground Plan	Connect to digital ground.
7	TRDATA2	Trace data 2	
8	TRDATA3	Trace data 3	
9	VSS	System Ground Plan	Connect to digital ground.
10	TRDATA4	Trace data 4	
11	TRDATA5	Trace data 5	
12	VSS	System Ground Plan	Connect to digital ground.
13	TRDATA6	Trace data 5	
14	TRDATA7	Trace data 6	

15	VCC	Target VCC	Just used for voltage reference.
16	EVENT0	Event output	
17	EVENT1	Event output	
18	EVENT2	Event output	
19	EVENT3	Event output	
20	N/C		

Memory Classes

Memory Class	Description
D,C	Data memory. Memory seen from the cores point of view.
P	Program memory.