GDB Server SDK

J-Link GDB Server
Thread Awareness SDK

User Guide & Reference Manual

Document: UM08030
Software Version: 1.0
Revision: 0
Date: June 17, 2016

A product of SEGGER Microcontroller GmbH & Co. KG

www.segger.com

Disclaimer

Specifications written in this document are believed to be accurate, but are not guaranteed to be entirely free of error. The information in this manual is subject to change for functional or performance improvements without notice. Please make sure your manual is the latest edition. While the information herein is assumed to be accurate, SEGGER Microcontroller GmbH & Co. KG (SEGGER) assumes no responsibility for any errors or omissions. SEGGER makes and you receive no warranties or conditions, express, implied, statutory or in any communication with you. SEGGER specifically disclaims any implied warranty of merchantability or fitness for a particular purpose.

Copyright notice

Trademarks

Names mentioned in this manual may be trademarks of their respective companies.

Brand and product names are trademarks or registered trademarks of their respective holders.

Contact address

SEGGER Microcontroller GmbH & Co. KG

In den Weiden 11
D-40721 Hilden

Germany

Tel. +49 2103-2878-0
Fax. +49 2103-2878-28
E-mail: support@segger.com
Internet: www.segger.com

Manual versions

This manual describes the current software version. If you find an error in the manual or a problem in the software, please inform us and we will try to assist you as soon as possible. Contact us for further information on topics or functions that are not yet documented.

Print date: June 17, 2016

Software	Revision	Date	By	Description
1.00	0	160516	AB	Initial version.

About this document

Assumptions

This document assumes that you already have a solid knowledge of the following:

- The software tools used for building your application (assembler, linker, C compiler).
- The C programming language.
- The target processor.
- DOS command line.

If you feel that your knowledge of C is not sufficient, we recommend *The C Programming Language* by Kernighan and Richie (ISBN 0–13–1103628), which describes the standard in C programming and, in newer editions, also covers the ANSI C standard.

How to use this manual

This manual explains all the functions and macros that the product offers. It assumes you have a working knowledge of the C language. Knowledge of assembly programming is not required.

Typographic conventions for syntax

This manual uses the following typographic conventions:

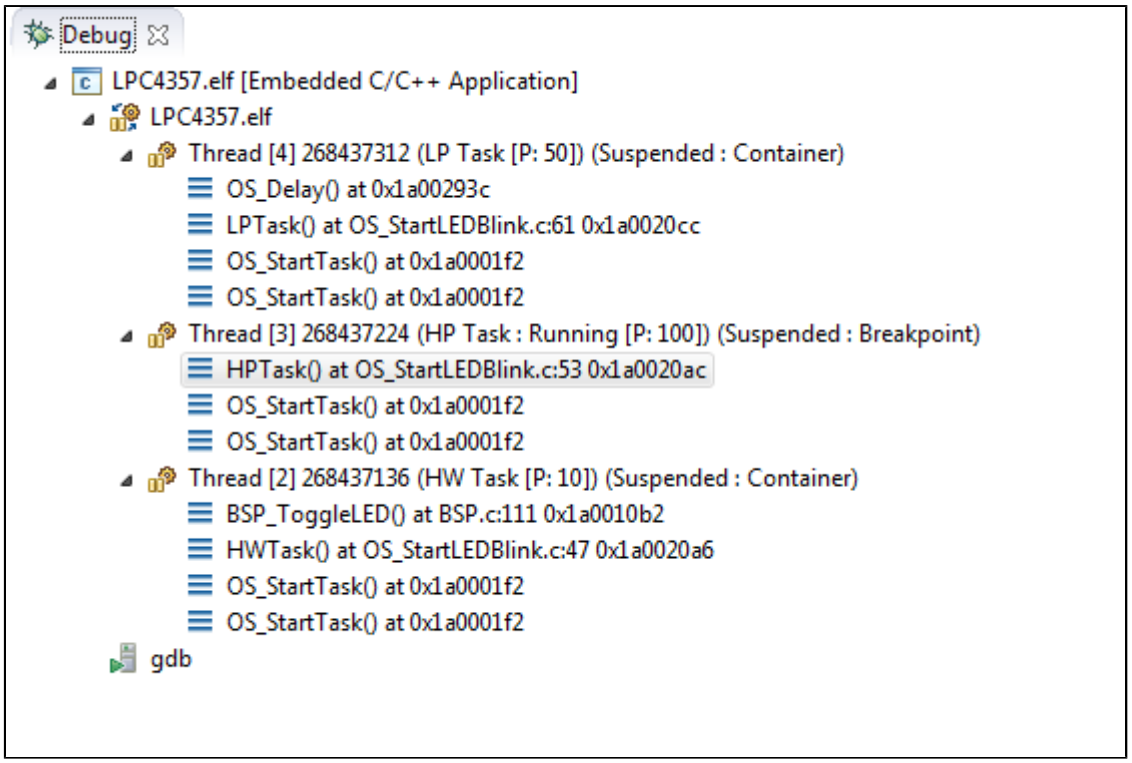
Style	Used for
Body	Body text.
`Keyword`	Text that you enter at the command prompt or that appears on the display (that is system functions, file- or pathnames).
`Parameter`	Parameters in API functions.
`Sample`	Sample code in program examples.
`Sample comment`	Comments in program examples.
Reference	Reference to chapters, sections, tables and figures or other documents.
GUIElement	Buttons, dialog boxes, menu names, menu commands.
Emphasis	Very important sections.

Table of contents

Chapter 1

Introduction

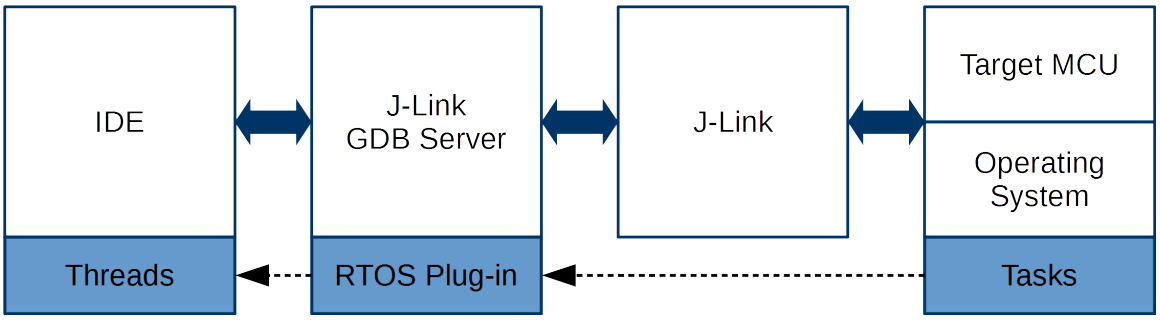
The J-Link GDB Server is able to represent the tasks of an embedded operating system to the GDB client as threads. In order to do this, the GDB server needs an interface to extract the task information from the target. As every OS uses its own algorithms and data structures for storing information about tasks, for every OS a unique plugin is needed.



Chapter 2

Overview

The interface is implemented as a shared library loaded by the GDB server. The GDB server provides an API for various tasks, like reading or writing memory and importing symbols from the GDB client. The library provides a set of functions, which are called from the GDB server when the GDB client requests information about threads.



2.1 Connect sequence

The GDB-Serves calls `RTOS_GetVersion()` immediatly after loading the RTOS plug-in library. If it passes the version check, the plug-in remains loaded and will be used. When a GDB client connects to the J-Link GDB Server with the loaded RTOS plug-in, at first `RTOS_Init()` is called and a pointer to the GDB Server API as well as the target's CPU core are passed. In the next step the GDB client will offer to serve symbol lookup requests. The GDB Server then requests a list of symbols from the RTOS plugin by calling `RTOS_GetSymbols()` and tries to find the memory addresses for these symbols.

Each time the target stops, `RTOS_UpdateThreads()` will be called in order to update the internal state about running tasks. Further calls to the functions depend on what information the GDB client requests.

If the GDB client…

- requests a list of threads, the J-Link GDB Server initially calls `RTOS_GetNumThreads()` and then sequentially `RTOS_GetThreadId()` for each Thread.
- requests the thread ID of the current execution, `RTOS_GetCurrentThreadId()` is called.
- requests information about a thread `RTOS_GetThreadDisplay()` is called.
- requests a register read `RTOS_GetThreadReg()` is called.
- requests a register list read `RTOS_GetThreadRegList()` is called.
- requests a register write `RTOS_SetThreadReg()` is called.
- requests a register list write `RTOS_SetThreadRegList()` is called.

2.2 Project template and sample code

The SDK comes with a project template, including a project file for use with Microsoft Visual Studio 2010 and two Makefiles for building the shared library on Linux and Mac OS X (i386 and x86_64 architecture). The makefiles contain instructions about how to use them with the make tool.

The SDK also includes the source code (RTOSPlugin_embOS.c) of our embOS RTOS plugin.

Chapter 3

The plugin interface

3.1 Data Structures

3.1.1 RTOS_SYMBOLS

```
typedef struct {
  const char *name;
  int optional;
  U32 address;
} RTOS_SYMBOLS;
```

A symbol entry has three properties: A pointer to the name of the symbol, a flag if it is an optional or mandatory symbol and the address. The last entry of the array with symbols must contain `{ NULL, 0, 0 }` as an end marker.

3.1.2 Register lists

In the GDB protocol registers are referenced by an index number and there is a group called "general registers". The plugin uses four functions, which depend on the definitions of the register index numbers and the group of general registers:

- `RTOS_GetThreadReg()`
- `RTOS_GetThreadRegList()`
- `RTOS_SetThreadReg()`
- `RTOS_SetThreadRegList()`

The definitions of register index numbers and the group of general registers are different per target MCU. The following subchapters contain the XML files, which are sent to the GDB client in response to a `qxmlRegisters` command. The group of general registers is specified in the feature node "`org.gnu.gdb.arm.m-profile`". If there is no such block defined, all registers are considered as general registers.

3.1.2.1 Cortex-M0, Cortex-M0+, Cortex-M1, Cortex-M3

```
<?xml version="1.0"?>
<!-- Copyright (C) 2008 Free Software Foundation, Inc.

     Copying and distribution of this file, with or without modification,
     are permitted in any medium without royalty provided the copyright
     notice and this notice are preserved.  -->

<!DOCTYPE feature SYSTEM "gdb-target.dtd">
<target version="1.0">
  <architecture>arm</architecture>
  <feature name="org.gnu.gdb.arm.m-profile">
    <reg name="r0" bitsize="32" regnum="0" type="uint32" group="general"/>
    <reg name="r1" bitsize="32" regnum="1" type="uint32" group="general"/>
    <reg name="r2" bitsize="32" regnum="2" type="uint32" group="general"/>
    <reg name="r3" bitsize="32" regnum="3" type="uint32" group="general"/>
    <reg name="r4" bitsize="32" regnum="4" type="uint32" group="general"/>
    <reg name="r5" bitsize="32" regnum="5" type="uint32" group="general"/>
    <reg name="r6" bitsize="32" regnum="6" type="uint32" group="general"/>
    <reg name="r7" bitsize="32" regnum="7" type="uint32" group="general"/>
    <reg name="r8" bitsize="32" regnum="8" type="uint32" group="general"/>
    <reg name="r9" bitsize="32" regnum="9" type="uint32" group="general"/>
    <reg name="r10" bitsize="32" regnum="10" type="uint32" group="general"/>
    <reg name="r11" bitsize="32" regnum="11" type="uint32" group="general"/>
    <reg name="r12" bitsize="32" regnum="12" type="uint32" group="general"/>
    <reg name="sp" bitsize="32" regnum="13" type="data_ptr" group="general"/>
    <reg name="lr" bitsize="32" regnum="14" type="uint32" group="general"/>
    <reg name="pc" bitsize="32" regnum="15" type="code_ptr" group="general"/>
    <reg name="xpsr" bitsize="32" regnum="25" type="uint32" group="general"/>
  </feature>
  <feature name="org.gnu.gdb.arm.m-system">
    <reg name="msp" bitsize="32" regnum="26" type="uint32" group="general"/>
    <reg name="psp" bitsize="32" regnum="27" type="uint32" group="general"/>
```



```
    <reg name="primask" bitsize="32" regnum="28" type="uint32" group="general"/>
    <reg name="basepri" bitsize="32" regnum="29" type="uint32" group="general"/>
    <reg name="faultmask" bitsize="32" regnum="30" type="uint32" group="general"/>
    <reg name="control" bitsize="32" regnum="31" type="uint32" group="general"/>
  </feature>
</target>
```

3.1.2.2 Cortex-M4

```
<?xml version="1.0"?>
<!-- Copyright (C) 2008 Free Software Foundation, Inc.

     Copying and distribution of this file, with or without modification,
     are permitted in any medium without royalty provided the copyright
     notice and this notice are preserved.  -->

<!DOCTYPE feature SYSTEM "gdb-target.dtd">
<target version="1.0">
  <architecture>arm</architecture>
  <feature name="org.gnu.gdb.arm.m-profile">
    <reg name="r0" bitsize="32" regnum="0" type="uint32" group="general"/>
    <reg name="r1" bitsize="32" regnum="1" type="uint32" group="general"/>
    <reg name="r2" bitsize="32" regnum="2" type="uint32" group="general"/>
    <reg name="r3" bitsize="32" regnum="3" type="uint32" group="general"/>
    <reg name="r4" bitsize="32" regnum="4" type="uint32" group="general"/>
    <reg name="r5" bitsize="32" regnum="5" type="uint32" group="general"/>
    <reg name="r6" bitsize="32" regnum="6" type="uint32" group="general"/>
    <reg name="r7" bitsize="32" regnum="7" type="uint32" group="general"/>
    <reg name="r8" bitsize="32" regnum="8" type="uint32" group="general"/>
    <reg name="r9" bitsize="32" regnum="9" type="uint32" group="general"/>
    <reg name="r10" bitsize="32" regnum="10" type="uint32" group="general"/>
    <reg name="r11" bitsize="32" regnum="11" type="uint32" group="general"/>
    <reg name="r12" bitsize="32" regnum="12" type="uint32" group="general"/>
    <reg name="sp" bitsize="32" regnum="13" type="data_ptr" group="general"/>
    <reg name="lr" bitsize="32" regnum="14" type="uint32" group="general"/>
    <reg name="pc" bitsize="32" regnum="15" type="code_ptr" group="general"/>
    <reg name="xpsr" bitsize="32" regnum="25" type="uint32" group="general"/>
  </feature>
  <feature name="org.gnu.gdb.arm.m-system">
    <reg name="msp" bitsize="32" regnum="26" type="uint32" group="general"/>
    <reg name="psp" bitsize="32" regnum="27" type="uint32" group="general"/>
    <reg name="primask" bitsize="32" regnum="28" type="uint32" group="general"/>
    <reg name="basepri" bitsize="32" regnum="29" type="uint32" group="general"/>
    <reg name="faultmask" bitsize="32" regnum="30" type="uint32" group="general"/>
    <reg name="control" bitsize="32" regnum="31" type="uint32" group="general"/>
  </feature>
  <feature name="org.gnu.gdb.arm.m-float">
    <reg name="fpscr" bitsize="32" regnum="32" type="uint32" group="float"/>
    <reg name="s0" bitsize="32" regnum="33" type="float" group="float"/>
    <reg name="s1" bitsize="32" regnum="34" type="float" group="float"/>
    <reg name="s2" bitsize="32" regnum="35" type="float" group="float"/>
    <reg name="s3" bitsize="32" regnum="36" type="float" group="float"/>
    <reg name="s4" bitsize="32" regnum="37" type="float" group="float"/>
    <reg name="s5" bitsize="32" regnum="38" type="float" group="float"/>
    <reg name="s6" bitsize="32" regnum="39" type="float" group="float"/>
    <reg name="s7" bitsize="32" regnum="40" type="float" group="float"/>
    <reg name="s8" bitsize="32" regnum="41" type="float" group="float"/>
    <reg name="s9" bitsize="32" regnum="42" type="float" group="float"/>
    <reg name="s10" bitsize="32" regnum="43" type="float" group="float"/>
    <reg name="s11" bitsize="32" regnum="44" type="float" group="float"/>
    <reg name="s12" bitsize="32" regnum="45" type="float" group="float"/>
    <reg name="s13" bitsize="32" regnum="46" type="float" group="float"/>
    <reg name="s14" bitsize="32" regnum="47" type="float" group="float"/>
    <reg name="s15" bitsize="32" regnum="48" type="float" group="float"/>
    <reg name="s16" bitsize="32" regnum="49" type="float" group="float"/>
    <reg name="s17" bitsize="32" regnum="50" type="float" group="float"/>
    <reg name="s18" bitsize="32" regnum="51" type="float" group="float"/>
    <reg name="s19" bitsize="32" regnum="52" type="float" group="float"/>
```



```
    <reg name="s20" bitsize="32" regnum="53" type="float" group="float"/>
    <reg name="s21" bitsize="32" regnum="54" type="float" group="float"/>
    <reg name="s22" bitsize="32" regnum="55" type="float" group="float"/>
    <reg name="s23" bitsize="32" regnum="56" type="float" group="float"/>
    <reg name="s24" bitsize="32" regnum="57" type="float" group="float"/>
    <reg name="s25" bitsize="32" regnum="58" type="float" group="float"/>
    <reg name="s26" bitsize="32" regnum="59" type="float" group="float"/>
    <reg name="s27" bitsize="32" regnum="60" type="float" group="float"/>
    <reg name="s28" bitsize="32" regnum="61" type="float" group="float"/>
    <reg name="s29" bitsize="32" regnum="62" type="float" group="float"/>
    <reg name="s30" bitsize="32" regnum="63" type="float" group="float"/>
    <reg name="s31" bitsize="32" regnum="64" type="float" group="float"/>
  </feature>
</target>
```

3.1.2.3 Cortex-A, Cortex-R

```
<?xml version="1.0"?>
<!-- Copyright (C) 2008 Free Software Foundation, Inc.

     Copying and distribution of this file, with or without modification,
     are permitted in any medium without royalty provided the copyright
     notice and this notice are preserved.  -->

<!DOCTYPE feature SYSTEM "gdb-target.dtd">
<target>
  <architecture>arm</architecture>
  <feature name="org.gnu.gdb.arm.core">
    <reg name="r0" bitsize="32" type="uint32" group="general"/>
    <reg name="r1" bitsize="32" type="uint32" group="general"/>
    <reg name="r2" bitsize="32" type="uint32" group="general"/>
    <reg name="r3" bitsize="32" type="uint32" group="general"/>
    <reg name="r4" bitsize="32" type="uint32" group="general"/>
    <reg name="r5" bitsize="32" type="uint32" group="general"/>
    <reg name="r6" bitsize="32" type="uint32" group="general"/>
    <reg name="r7" bitsize="32" type="uint32" group="general"/>
    <reg name="r8" bitsize="32" type="uint32" group="general"/>
    <reg name="r9" bitsize="32" type="uint32" group="general"/>
    <reg name="r10" bitsize="32" type="uint32" group="general"/>
    <reg name="r11" bitsize="32" type="uint32" group="general"/>
    <reg name="r12" bitsize="32" type="uint32" group="general"/>
    <reg name="sp" bitsize="32" type="data_ptr" group="general"/>
    <reg name="lr" bitsize="32" type="uint32" group="general"/>
    <reg name="pc" bitsize="32" type="code_ptr" group="general"/>
    <reg name="cpsr" bitsize="32" type="uint32" group="general"/>
  </feature>
  <feature name="org.gnu.gdb.arm.vfp">
    <reg name="fpscr" bitsize="32" type="uint32" group="float"/>
    <reg name="d0" bitsize="64" type="ieee_double" group="float"/>
    <reg name="d1" bitsize="64" type="ieee_double" group="float"/>
    <reg name="d2" bitsize="64" type="ieee_double" group="float"/>
    <reg name="d3" bitsize="64" type="ieee_double" group="float"/>
    <reg name="d4" bitsize="64" type="ieee_double" group="float"/>
    <reg name="d5" bitsize="64" type="ieee_double" group="float"/>
    <reg name="d6" bitsize="64" type="ieee_double" group="float"/>
    <reg name="d7" bitsize="64" type="ieee_double" group="float"/>
    <reg name="d8" bitsize="64" type="ieee_double" group="float"/>
    <reg name="d9" bitsize="64" type="ieee_double" group="float"/>
    <reg name="d10" bitsize="64" type="ieee_double" group="float"/>
    <reg name="d11" bitsize="64" type="ieee_double" group="float"/>
    <reg name="d12" bitsize="64" type="ieee_double" group="float"/>
    <reg name="d13" bitsize="64" type="ieee_double" group="float"/>
    <reg name="d14" bitsize="64" type="ieee_double" group="float"/>
    <reg name="d15" bitsize="64" type="ieee_double" group="float"/>
  </feature>
  <feature name="segger.core.mode">
    <reg name="r8_usr" bitsize="32" type="uint32" group="mode"/>
    <reg name="r9_usr" bitsize="32" type="uint32" group="mode"/>
```



```
    <reg name="r10_usr" bitsize="32" type="uint32" group="mode"/>
    <reg name="r11_usr" bitsize="32" type="uint32" group="mode"/>
    <reg name="r12_usr" bitsize="32" type="uint32" group="mode"/>
    <reg name="r13_usr" bitsize="32" type="uint32" group="mode"/>
    <reg name="r14_usr" bitsize="32" type="uint32" group="mode"/>
    <reg name="r8_fiq" bitsize="32" type="uint32" group="mode"/>
    <reg name="r9_fiq" bitsize="32" type="uint32" group="mode"/>
    <reg name="r10_fiq" bitsize="32" type="uint32" group="mode"/>
    <reg name="r11_fiq" bitsize="32" type="uint32" group="mode"/>
    <reg name="r12_fiq" bitsize="32" type="uint32" group="mode"/>
    <reg name="r13_fiq" bitsize="32" type="uint32" group="mode"/>
    <reg name="r14_fiq" bitsize="32" type="uint32" group="mode"/>
    <reg name="spsr_fiq" bitsize="32" type="uint32" group="mode"/>
    <reg name="r13_irq" bitsize="32" type="uint32" group="mode"/>
    <reg name="r14_irq" bitsize="32" type="uint32" group="mode"/>
    <reg name="spsr_irq" bitsize="32" type="uint32" group="mode"/>
    <reg name="r13_svc" bitsize="32" type="uint32" group="mode"/>
    <reg name="r14_svc" bitsize="32" type="uint32" group="mode"/>
    <reg name="spsr_svc" bitsize="32" type="uint32" group="mode"/>
    <reg name="r13_abt" bitsize="32" type="uint32" group="mode"/>
    <reg name="r14_abt" bitsize="32" type="uint32" group="mode"/>
    <reg name="spsr_abt" bitsize="32" type="uint32" group="mode"/>
    <reg name="r13_und" bitsize="32" type="uint32" group="mode"/>
    <reg name="r14_und" bitsize="32" type="uint32" group="mode"/>
    <reg name="spsr_und" bitsize="32" type="uint32" group="mode"/>
  </feature>
</target>
```

3.1.2.4 MIPS

```
<?xml version="1.0"?>
<!-- Copyright (C) 2007, 2008, 2009, 2010 Free Software Foundation, Inc.

     Copying and distribution of this file, with or without modification,
     are permitted in any medium without royalty provided the copyright
     notice and this notice are preserved.  -->

<!DOCTYPE target SYSTEM "gdb-target.dtd">
<target>
  <architecture>mips</architecture>
  <feature name="org.gnu.gdb.mips.cpu">
    <reg name="r0" bitsize="32" regnum="0"/>
    <reg name="r1" bitsize="32"/>
    <reg name="r2" bitsize="32"/>
    <reg name="r3" bitsize="32"/>
    <reg name="r4" bitsize="32"/>
    <reg name="r5" bitsize="32"/>
    <reg name="r6" bitsize="32"/>
    <reg name="r7" bitsize="32"/>
    <reg name="r8" bitsize="32"/>
    <reg name="r9" bitsize="32"/>
    <reg name="r10" bitsize="32"/>
    <reg name="r11" bitsize="32"/>
    <reg name="r12" bitsize="32"/>
    <reg name="r13" bitsize="32"/>
    <reg name="r14" bitsize="32"/>
    <reg name="r15" bitsize="32"/>
    <reg name="r16" bitsize="32"/>
    <reg name="r17" bitsize="32"/>
    <reg name="r18" bitsize="32"/>
    <reg name="r19" bitsize="32"/>
    <reg name="r20" bitsize="32"/>
    <reg name="r21" bitsize="32"/>
    <reg name="r22" bitsize="32"/>
    <reg name="r23" bitsize="32"/>
    <reg name="r24" bitsize="32"/>
    <reg name="r25" bitsize="32"/>
    <reg name="r26" bitsize="32"/>
```



```
      <reg name="r27" bitsize="32"/>
      <reg name="r28" bitsize="32"/>
      <reg name="r29" bitsize="32"/>
      <reg name="r30" bitsize="32"/>
      <reg name="r31" bitsize="32"/>

      <reg name="lo" bitsize="32" regnum="33"/>
      <reg name="hi" bitsize="32" regnum="34"/>
      <reg name="pc" bitsize="32" regnum="37"/>
    </feature>
    <feature name="org.gnu.gdb.mips.cp0">
      <reg name="status" bitsize="32" regnum="32"/>
      <reg name="badvaddr" bitsize="32" regnum="35"/>
      <reg name="cause" bitsize="32" regnum="36"/>
    </feature>
    <feature name="org.gnu.gdb.mips.fpu">
      <reg name="f0" bitsize="32" type="ieee_single" regnum="38"/>
      <reg name="f1" bitsize="32" type="ieee_single"/>
      <reg name="f2" bitsize="32" type="ieee_single"/>
      <reg name="f3" bitsize="32" type="ieee_single"/>
      <reg name="f4" bitsize="32" type="ieee_single"/>
      <reg name="f5" bitsize="32" type="ieee_single"/>
      <reg name="f6" bitsize="32" type="ieee_single"/>
      <reg name="f7" bitsize="32" type="ieee_single"/>
      <reg name="f8" bitsize="32" type="ieee_single"/>
      <reg name="f9" bitsize="32" type="ieee_single"/>
      <reg name="f10" bitsize="32" type="ieee_single"/>
      <reg name="f11" bitsize="32" type="ieee_single"/>
      <reg name="f12" bitsize="32" type="ieee_single"/>
      <reg name="f13" bitsize="32" type="ieee_single"/>
      <reg name="f14" bitsize="32" type="ieee_single"/>
      <reg name="f15" bitsize="32" type="ieee_single"/>
      <reg name="f16" bitsize="32" type="ieee_single"/>
      <reg name="f17" bitsize="32" type="ieee_single"/>
      <reg name="f18" bitsize="32" type="ieee_single"/>
      <reg name="f19" bitsize="32" type="ieee_single"/>
      <reg name="f20" bitsize="32" type="ieee_single"/>
      <reg name="f21" bitsize="32" type="ieee_single"/>
      <reg name="f22" bitsize="32" type="ieee_single"/>
      <reg name="f23" bitsize="32" type="ieee_single"/>
      <reg name="f24" bitsize="32" type="ieee_single"/>
      <reg name="f25" bitsize="32" type="ieee_single"/>
      <reg name="f26" bitsize="32" type="ieee_single"/>
      <reg name="f27" bitsize="32" type="ieee_single"/>
      <reg name="f28" bitsize="32" type="ieee_single"/>
      <reg name="f29" bitsize="32" type="ieee_single"/>
      <reg name="f30" bitsize="32" type="ieee_single"/>
      <reg name="f31" bitsize="32" type="ieee_single"/>

      <reg name="fcsr" bitsize="32" group="float"/>
      <reg name="fir" bitsize="32" group="float"/>
    </feature>
  </target>
```


3.2 GDB Server API

3.2.1 SYS_MEM_Free()

Description

Frees a previously allocated memory block.

Prototype

```
void SYS_MEM_Free(void *p);
```

Parameter	Description
p	Pointer to the memory block.

3.2.2 SYS_MEM_Alloc()

Description

Allocates a memory block from the heap memory.

Prototype

```
void* SYS_MEM_Alloc(unsigned Size);
```

Parameter	Description
Size	Size of memory block.

Return value

Pointer to the allocated memory block. NULL, if the memory could not be allocated.

3.2.3 SYS_MEM_Realloc()

Description

Reallocates a memory block.

Prototype

```
void* SYS_MEM_Realloc(void *p, unsigned Size);
```

Parameter	Description
p	Pointer to old memory block.
Size	New size of memory block.

Return value

Pointer to the new memory block. NULL, if the memory could not be allocated.

Additional information

The memory block may be moved to a new location, whose address then is returned. The content of the original memory block is preserved by copying it to the new location, if the block had to be moved.

3.2.4 APP_DebugOutf()

Description

Outputs a formatted log message to the debug channel.

Prototype

```
void APP_LogOutf(const char* sFormat, …);
```

Parameter	Description
sFormat	Text string.
…	Values to be formatted.

Additional information

Outputs to the debug channel are suppressed in non-debug builds of the J-Link GDB Server.

3.2.5 APP_LogOutf()

Description

Outputs a formatted log message to J-Link GDB server window.

Prototype

```
void APP_LogOutf(const char* sFormat, …);
```

Parameter	Description
sFormat	Text string.
…	Values to be formatted.

Additional information

If a log file is specified, the message will also be printed to the log file.

3.2.6 APP_WarnOutf()

Description

Outputs a formatted warning message to J-Link GDB server window.

Prototype

```
void APP_LogOutf(const char* sFormat, …);
```

Parameter	Description
sFormat	Text string.
…	Values to be formatted.

Additional information

The line starts with “WARNING: ”. If a log file is specified, the message will also be printed to the log file.

3.2.7 APP_ErrorOutf()

Description

Outputs a formatted error message to J-Link GDB server window.

Prototype

```
void APP_LogOutf(const char* sFormat, …);
```

Parameter	Description
sFormat	Text string.
…	Values to be formatted.

Additional information

The line starts with "ERROR: ". If a log file is specified, the message will also be printed to the log file.

3.2.8 TARGET_ReadMem()

Description

The function reads memory from the target system.

Prototype

```
int TARGET_ReadMem(U32 Addr, char* pData, unsigned int NumBytes);
```

Parameter	Description
Addr	Target address to read from.
pData	Pointer to buffer for target memory.
NumBytes	Number of bytes to read.

Return value

= 0 Reading memory Ok.
< 0 Reading memory failed.

Additional information

If necessary, the target CPU is halted in order to read memory.

3.2.9 TARGET_ReadU8()

Description

The function reads one byte from the target system.

Prototype

```
int TARGET_ReadU8(U32 Addr, U8* pData);
```

Parameter	Description
Addr	Target address to read from.
pData	Pointer to buffer for target memory.

Return value

= 0 Reading memory Ok.
< 0 Reading memory failed.

Additional information

If necessary, the target CPU is halted in order to read memory.

3.2.10 TARGET_ReadU16()

Description

The function reads two bytes from the target system.

Prototype

```
int TARGET_ReadU16(U32 Addr, U16* pData);
```

Parameter	Description
Addr	Target address to read from.
pData	Pointer to buffer for target memory.

Return value

= 0 Reading memory Ok.
< 0 Reading memory failed.

Additional information

If necessary, the target CPU is halted in order to read memory.

3.2.11 TARGET_ReadU32()

Description

The function reads 4 bytes from the target system.

Prototype

```
int TARGET_ReadU32(U32 Addr, U32* pData);
```

Parameter	Description
Addr	Target address to read from.
pData	Pointer to buffer for target memory.

Return value

= 0 Reading memory Ok.
< 0 Reading memory failed.

Additional information

If necessary, the target CPU is halted in order to read memory.

3.2.12 TARGET_WriteMem()

Description

The function writes memory to the target system.

Prototype

```
int TARGET_WriteMem(U32 Addr, const char* pData, unsigned int NumBytes);
```

Parameter	Description
Addr	Target memory address to write to.
pData	Pointer to buffer containing the data bytes to write.
NumBytes	Number of bytes to write.

Return value

= 0 Writing memory Ok.
< 0 Writing memory failed.

Additional information

If necessary, the target CPU is halted in order to write to the memory.

3.2.13 TARGET_WriteU8()

Description

The function writes one byte to the target system.

Prototype

```
int TARGET_WriteU8(U32 Addr, U8* pData);
```

Parameter	Description
Addr	Target memory address to write to.
pData	Pointer to buffer containing the data byte to write.

Return value

= 0 Writing memory Ok.
< 0 Writing memory failed.

Additional information

If necessary, the target CPU is halted in order to write to the memory.

3.2.14 TARGET_WriteU16()

Description

The function writes one byte to the target system.

Prototype

```
int TARGET_WriteU16(U32 Addr, U16* pData);
```

Parameter	Description
Addr	Target memory address to write to.
pData	Pointer to buffer containing the data bytes to write.

Return value

= 0 Writing memory Ok.
< 0 Writing memory failed.

Additional information

If necessary, the target CPU is halted in order to write to the memory.

3.2.15 TARGET_WriteU32()

Description

The function writes four bytes to the target system.

Prototype

```
int TARGET_WriteU32(U32 Addr, U32* pData);
```

Parameter	Description
Addr	Target memory address to write to.
pData	Pointer to buffer containing the data bytes to write.

Return value

= 0 Writing memory Ok.
< 0 Writing memory failed.

Additional information

If necessary, the target CPU is halted in order to write to the memory.

3.2.16 MISC_Load16TE()

Description

The function loads two bytes from a memory buffer according to the target's endianess.

Prototype

```
U32 MISC_Load16TE(const U8* p);
```

Parameter	Description
p	Pointer to memory buffer.

Return value

The memory value.

Additional information

This function is intented to be used in conjunction with TARGET_ReadMem() or TARGET_WriteMem(). When reading single integer values, the endianess is automatically taken into account.

3.2.17 MISC_Load24TE()

Description

The function loads three bytes from a memory buffer according to the target's endianess.

Prototype

```
U32 MISC_Load24TE(const U8* p);
```

Parameter	Description
p	Pointer to memory buffer.

Return value

The memory value.

Additional information

This function is intented to be used in conjunction with TARGET_ReadMem() or TARGET_WriteMem(). When reading single integer values, the endianess is automatically taken into account.

3.2.18 MISC_Load32TE()

Description

The function loads four bytes from a memory buffer according to the target's endianess.

Prototype

```
U32 MISC_Load32TE(const U8* p);
```

Parameter	Description
p	Pointer to memory buffer.

Return value

The memory value.

Additional information

This function is intented to be used in conjunction with TARGET_ReadMem() or TARGET_WriteMem(). When reading single integer values, the endianess is automatically taken into account.

3.3 Plug-in Client API

3.3.1 RTOS_Init()

Description

Initializes RTOS plug-in for further usage.

Prototype

```
int RTOS_Init(const GDB_API * pAPI,
                    U32       core);
```

Parameters

Parameter	Description
pAPI	Pointer to API functions table.
core	JLINK_CORE_ constant identifying the target's core.

Return value

= 0 Initialization failed.
= 1 Initialized successfully.

Additional information

If the plug-in does not support the CPU architecture, it should return 0. The pointer to the API functions table should be stored locally. API funtions can be used later by the plug-in to perform special functions like reading or writing to target memory.

3.3.2 RTOS_GetVersion()

Description

Returns the RTOS plugin version.

Prototype

```
U32 RTOS_GetVersion();
```

Return value

The plugin version number as unsigned integer: 100 * [major] + [minor].

Additional information

Will be called before any other function. The J-Link GDB server only checks the RTOS plug-in's major version number. The minor version number is freely choosable, it is printed in the GDB server's log file but it is not evaluated.

3.3.3 RTOS_GetSymbols()

Description

Returns a pointer to the RTOS symbol table.

Prototype

```
RTOS_SYMBOLS *RTOS_GetSymbols();
```

Return value

Pointer to the RTOS symbol table.

Additional information

The J-Link GDB server tries to find addresses of all of the symbols specified in the RTOS symbol table. If it is found, the address is written into RTOS_SYMBOLS.address, otherwise NULL is written into the address field. If any of the symbols, declared as mandatory, is not found, the plug-in is not being used by the GDB server. So it is ensured for the following functions, that every mandatory symbol has a valid address entry.

3.3.4 RTOS_GetNumThreads()

Description

Returns the number of threads.

Prototype

```
U32 RTOS_GetNumThreads();
```

Return value

Number of threads.

Additional information

After calling this function, the GDB server will request the thread ID by calling RTOS_GetThreadId() for every thread.

3.3.5 RTOS_GetThreadId()

Description

Returns the ID of the thread with `index` number n.

Prototype

```
U32 RTOS_GetThreadId(U32 index);
```

Parameters

Parameter	Description
`index`	Index number of the thread.

Return value

ID of the thread.

Additional information

Index numbers for threads run from 0..[m-1], where m is the number of threads returned by RTOS_GetNumThreads().

3.3.6 RTOS_GetCurrentThreadId()

Description

Returns the ID of the currently running thread.

Prototype

```
U32 RTOS_GetCurrentThreadId();
```

Return value

ID of the currently running thread.

3.3.7 RTOS_GetThreadDisplay()

Description

Prints the thread's name to pDisplay. The name may contain extra information about the tread's status (running/suspended, priority, etc.).

Prototype

```
int RTOS_GetThreadDisplay(char * pDisplay,
                          U32    threadid);
```

Parameters

Parameter	Description
pDisplay	Pointer to the string, the name has to be copied to.
threadid	ID of the thread.

Return value

Length of the name string.

Additional information

The space reserved for the name is 256 bytes (including terminating zero)

3.3.8 RTOS_GetThreadReg()

Description

Copys the thread’s register value to pRegValue as HEX string.

If the register value has to be read directly from the CPU, the function must return a value <0. The register value is then read from the CPU by the GDB server itself.

Prototype

```
int RTOS_GetThreadReg(char * pHexRegVal,
                      U32    RegIndex,
                      U32    threadid);
```

Parameters

Parameter	Description
pHexRegVal	Pointer to the string, the value has to be copied to.
RegIndex	Index of the register.
threadid	ID of the thread.

Return value

= 0 Reading register Ok.
< 0 Reading register failed.

3.3.9 RTOS_GetThreadRegList()

Description

Copys the thread's general registers to `pHexRegList` as HEX string.

If the register values have to be read directly from the CPU, the function must return a value <0. The register values are then read from the CPU by the GDB server itself.

Prototype

```
int RTOS_GetThreadRegList(char * pHexRegList,
                          U32    threadid);
```

Parameters

Parameter	Description
`pHexRegList`	Pointer to the string, the values have to be copied to.
`threadid`	ID of the thread.

Return value

= 0 Reading registers Ok.
< 0 Reading registers failed.

3.3.10 RTOS_SetThreadReg()

Description

Sets the thread’s register to pRegValue, given as HEX string.

If the register value has to be written directly to the CPU, the function must return a value <0. The register value is then written to the CPU by the GDB server itself.

Prototype

```
int RTOS_SetThreadReg(char * pHexRegVal,
                      U32    RegIndex,
                      U32    threadid);
```

Parameters

Parameter	Description
pHexRegVal	Pointer to the string, containing the value to write.
RegIndex	Index of the register.
threadid	ID of the thread.

Return value

= 0 Writing register Ok.
< 0 Writing register failed.

3.3.11 RTOS_SetThreadRegList()

Description

Sets the thread's registers to pHexRegList, given as HEX string.

If the register values have to be written directly to the CPU, the function must return a value <0. The register values are then written to the CPU by the GDB server itself.

Prototype

```
int RTOS_SetThreadRegList(char * pHexRegList,
                          U32    threadid);
```

Parameters

Parameter	Description
pHexRegList	Pointer to the string, containing the values to write.
threadid	ID of the thread.

Return value

= 0 Writing registers Ok.
< 0 Writing registers failed.

3.3.12 RTOS_UpdateThreads()

Description

Updates the thread information from the target.

For efficiency purposes, the plug-in should read all required information within this function at once, so later requests can be served without further communication to the target.

Prototype

```
int RTOS_UpdateThreads();
```

Return value

= 0	Updating threads Ok.
< 0	Updating threads failed.

Chapter 4

Indexes

4.1 Index of functions

