J-Link GDB Server - GDB protocol extension

Document: UM08036
Software Version: 1.00
Revision: 0
Date: March 27, 2018

A product of SEGGER Microcontroller GmbH

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 (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

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 report it to 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: March 27, 2018

Manual version	Revision	Date	By	Description
1.00	0	180327	AG	Created

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

This is the developer documentation for J-Link GDB Server. This manual documents the SEGGER specific GDB protocol extensions that have been added to the J-Link GDB Server, in order to support features like (ETM/ETB) Trace with code coverage etc., SWO any many others.

For a general documentation about the usage of J-Link GDB Server (configuration, command line options, supported monitor command etc.), please refer to UM08001, section “J-Link GDB Server”.

Chapter 2

Licensing

This chapter describes the different license types of J-Link related software and the legal use of the J-Link software with original SEGGER and OEM products.

2.1 Components requiring a license

J-Link PLUS and higher are fully featured J-Links and come with all licenses included. Other models may do not come with all features enabled. For a detailed overview of the included licenses of the SEGGER debug probes, please refer to:
J-Link Model overview: Licenses

2.2 Legal use of SEGGER J-Link software

The software consists of proprietary programs of SEGGER, protected under copyright and trade secret laws. All rights, title and interest in the software are and shall remain with SEGGER. For details, please refer to the license agreement which needs to be accepted when installing the software. The text of the license agreement is also available as entry in the start menu after installing the software.

Chapter 3

SEGGER-specific GDB protocol extensions

J-Link GDB Server implements some functionality which are not part of the standard GDB remote protocol in general query packets. These SEGGER-specific general query packets can be sent to GDB Server on the low-level of GDB, via maintenance commands, or with a custom client connected to GDB Server. General query packets start with a 'q'. SEGGER-specific general queries are followed by the identifier 'Segger' plus the command group, the actual command and its parameters. Following SEGGER-specific general query packets are available:

Query Packet	Explanation
qSeggerSTRACE:caps	Checks for STRACE caps of connected J-Link/J-Trace
qSeggerSTRACE:config	Configure STRACE for usage.
qSeggerSTRACE:start	Start STRACE.
qSeggerSTRACE:stop	Stop STRACE.
qSeggerSTRACE:read	Read STRACE data.
qSeggerSTRACE:GetInstStats	Get instruction statistics.
qSeggerSWO:start	Starts collecting SWO data.
qSeggerSWO:stop	Stops collecting SWO data.
qSeggerSWO:read	Reads data from SWO buffer.
qSeggerSWO:GetNumBytes	Returns the SWO buffer status.
qSeggerSWO:GetSpeedInfo	Returns info about supported speeds.

3.0.1 qSeggerSTRACE:caps

Returns STRACE capabilities of the current setup (which STRACE commands are supported by GDBServer and connected J-Link/J-Trace).

3.0.1.1 Syntax

`qSeggerSTRACE:caps`

Standard GDB protocol syntax:

- GDB remote protocol
- `ChkSum` is generated/expected as described in GDB protocol spec.
- GDB acknowledge mode active/inactive is taken into account for this command

Byte stream:

```
00000000  24 71 53 65 67 67 65 72  53 54 52 41 43 45 3a 63 $qSegger STRACE:c
00000010  61 70 73 23 37 31 00                           aps#71.
    00000000  2b                                          +
    00000001  24 63 6f 6e 66 69 67 3b  73 74 61 72 74 3b 73 74 $config; start;st
    00000011  6f 70 3b 72 65 61 64 3b  47 65 74 49 6e 73 74 53 op;read; GetInstS
    00000021  74 61 74 73 23 62 66                           tats#bf
```

3.0.1.2 Response

`<Feature0>;<FeatureN>`

Byte stream:

```
00000000  24 71 53 65 67 67 65 72  53 54 52 41 43 45 3a 63 $qSegger STRACE:c
00000010  61 70 73 23 37 31 00                           aps#71.
    00000000  2b                                          +
    00000001  24 63 6f 6e 66 69 67 3b  73 74 61 72 74 3b 73 74 $config; start;st
    00000011  6f 70 3b 72 65 61 64 3b  47 65 74 49 6e 73 74 53 op;read; GetInstS
    00000021  74 61 74 73 23 62 66                           tats#bf
```

Feature	Explanation
`qSeggerSTRACE:config`	`qSeggerSTRACE:config` command is supported
`qSeggerSTRACE:start`	`qSeggerSTRACE:start` command is supported
`qSeggerSTRACE:stop`	`qSeggerSTRACE:stop` command is supported
`qSeggerSTRACE:read`	`qSeggerSTRACE:read` command is supported
`qSeggerSTRACE:GetInstStats`	`qSeggerSTRACE:GetInstStats` command is supported

3.0.1.3 Notes

> **Note**
>
> If this command is not supported by J-Link GDBServer, GDBServer will respond with an empty packet

3.0.2 qSeggerSTRACE:config

Configures STRACE for usage.

3.0.2.1 Syntax

`qSeggerSTRACE:config:<ConfigString>`

Standard GDB protocol syntax:

- GDB remote protocol
- CheckSum is generated/expected as described in GDB protocol spec.
- GDB acknowledge mode active/inactive is taken into account for this command
- Numeric values are transmitted hex-encoded (see GDB protocol spec.)

Parameter	Meaning
`ConfigString`	String containing the configuration data separating settings by ';'.

3.0.2.2 Response

`<ReturnValue>`
ReturnValue is a 4 Byte signed integer.

Value	Meaning
`ReturnValue`	≥ 0 O.K. <0 Error.

3.0.2.3 Notes

> **Note**
>
> For more information please refer to UM08002 (J-Link SDK user guide), chapter STRACE .

3.0.3 qSeggerSTRACE:start

Starts capturing of STRACE data.

3.0.3.1 Syntax

`qSeggerSTRACE:start`

Standard GDB protocol syntax:

- GDB remote protocol
- `ChkSum` is generated/expected as described in GDB protocol spec.
- GDB acknowledge mode active/inactive is taken into account for this command
- Numeric values are transmitted hex-encoded (see GDB protocol spec.)

3.0.3.2 Response

`<ReturnValue>`
ReturnValue is a 4 Byte signed integer.

Value	Meaning
`ReturnValue`	≥ 0 O.K. <0 Error.

3.0.3.3 Notes

> **Note**
>
> For more information please refer to UM08002 (J-Link SDK user guide), chapter STRACE .

3.0.4 qSeggerSTRACE:stop

Stops capturing of STRACE data.

3.0.4.1 Syntax

`qSeggerSTRACE:stop`

Standard GDB protocol syntax:

- GDB remote protocol
- `ChkSum` is generated/expected as described in GDB protocol spec.
- GDB acknowledge mode active/inactive is taken into account for this command
- Numeric values are transmitted hex-encoded (see GDB protocol spec.)

3.0.4.2 Response

`<ReturnValue>`
ReturnValue is a 4 Byte signed integer.

Value	Meaning
`ReturnValue`	≥ 0 O.K. <0 Error.

3.0.4.3 Notes

> **Note**
>
> For more information please refer to UM08002 (J-Link SDK user guide), chapter STRACE .

3.0.5 qSeggerSTRACE:read

Reads the last recently called instruction addresses. The addresses are returned LIFO, meaning the last recent address is returned first.

3.0.5.1 Syntax

`qSeggerSTRACE:read:<NumItems>`

Standard GDB protocol syntax:

- GDB remote protocol
- `ChkSum` is generated/expected as described in GDB protocol spec.
- GDB acknowledge mode active/inactive is taken into account for this command
- Numeric values are transmitted hex-encoded (see GDB protocol spec.)

Parameter	Meaning
`NumItems`	Maximum number of trace data (addresses) to be read. Hexadecimal.

3.0.5.2 Response

`<ReturnValue>[<Item0><Item1>…]` ReturnValue is a 4 Byte signed integer.

Value	Meaning
`ReturnValue`	≥ 0 Number of items read. <0 Error.

3.0.5.3 Notes

Note

For more information please refer to UM08002 (J-Link SDK user guide), chapter STRACE .

Note

ReturnValue and ItemN are hex-encoded.
e.g. 3 Items read: `0x08000010`, `0x08000014`, `0x08000018`
Response will be: 00000003080000100800001408000018

3.0.6 qSeggerSTRACE:GetInstStats

Gets statistics for the instructions (fetch / executed / skipped count etc.) in the specified address range. This is used for displaying live code profiling / code coverage information and/or execution counters, like SEGGER Ozone does: Ozone trace features

3.0.6.1 Syntax

`qSeggerSTRACE:GetInstStats:<Addr><NumItems><StatType>`

> **Note**
>
> - Must not be used if not reported as supported by *qSeggerSTRACECaps* on page 13.
> - May only be called after qSeggerSTRACE:start has called
> - May be called while the CPU is running or halted
> - Only available for trace probes that support streaming trace: More info
>
> This command is incompatible to the regular GDB protocol. This is done for performance reasons, as the standard GDB protocol format adds significant overhead that is not feasible for commands that are called periodically at a high rate. The format for this command is as follows:
>
> - The command is preceded with an '$' character (compatible to GDB protocol)
> - The command is NOT followed by an '#' character (and there is no checksum following)
> - No acknowledge packet is sent by GDBServer when receiving this command
> - The parameters <Addr>, … are transmitted binary (not hex-encoded!) and little endian
> - No zero termination byte is following this command

Byte stream:

```
    000000E8  00 00 00 00                                             ....
00000040  24 71 53 65 67 67 65 72  53 54 52 41 43 45 3a 47 $qSegger STRACE:G
00000050  65 74 49 6e 73 74 53 74  61 74 73 3a 84 02 00 08 etInstSt ats:....
00000060  07 00 00 00 0b 00 00 00                           ........
    000000FC  18 00 00 00 01 00 00 00  00 00 00 00 01 00 00 00 ........ ........
```

Parameter	Size	Meaning
Addr	32-bit	Start address of range statistics are requested for.
NumItems	32-bit	Number of 16-bit items we want statistics for. General formula: <NumItems> = RangeSizeBytes / 2
StatType	32-bit	Specifies what statistics are actually requested. Determines the format of the response. For more info, see the response sections for this command.

Abbreviation	Meaning
Fetch	Count how many times an instruction was fetched.
Exec	Count how many times an instruction was fetched and executed. Only different to Fetch for conditional instructions. If a conditional branch has been fetched and taken, this is considered as Exec
Skip	Count of how many times an instruction has been skipped/not executed. Only different to Fetch for conditional instructions. If a conditional branch has been fetched but not taken, this is considered as Skip
NumInstSum	Sum of all addresses that have been fetched so far
xxxSum	Sum of all counts of type xxx that have been traced so far

Valid values for StatType :

Value	Symbolic name	Meaning
11	FetchExecSkipSum	Get Fetch , FetchSum , Exec , ExecSum , Skip and SkipSum statistics. The xxxSum statistics always refer to all traced addresses and not to the range requested by <Addr> and <NumItems>.
0	Fetch (deprectaed)	Get Fetch
1	Exec (deprectaed)	Get Exec
2	Skip (deprectaed)	Get Skip
3	Reserved	—
4	FetchSum (deprectaed)	Sum of all Fetch of all addresses that have been traced. <Addr> and <NumItems> are don't care.
5	ExecSum (deprectaed)	Sum of all Exec of all addresses that have been traced. <Addr> and <NumItems> are don't care.
6	SkipSum (deprectaed)	Sum of all Skip of all addresses that have been traced. <Addr> and <NumItems> are don't care.
7	NumInstSum (deprectaed)	Sum of all instructions that have been fetched, based on the addresses that have been traced so far. <Addr> and <NumItems> are don't care.
8	FetchExecSum (deprectaed)	Get Fetch , FetchSum , Exec and ExecSum statistics. The xxxSum values always refer to all traced addresses and not to the range requested by <Addr> and <NumItems>.
9	FetchSkipSum (deprectaed)	Get Fetch , FetchSum , Skip and SkipSum statistics. The xxxSum values always refer to all traced addresses and not to the range requested by <Addr> and <NumItems>.
10	ExecSkipSum (deprectaed)	Get Exec , ExecSum , Skip , SkipSum . The xxxSum values always refer to all traced addresses and not to the range requested by <Addr> and <NumItems>.

3.0.6.2 Example

```
    1 25 void main(void) {
    1          0800 0284  PUSH          {R4}
      26   register int i;
      27
    1 28   i = 0;
    1          0800 0286  MOVS          R4, #0x00
      29   do {
  100 30     i++;
  100          0800 0288  ADDS          R4, #0x01
    1          0800 0290  B             0x08000288                ; <main>+0x04
  100 31     if (i > 99) {
  100          0800 028A  CMP           R4, #0x63
   99          0800 028C  BLE           0x08000288                ; <main>+0x04
    1 32       i = 0;
    1          0800 028E  MOVS          R4, #0x00
      33     }
      34   } while (1);
      35 }
```

```
qSeggerSTRACE:GetInstStats:0x08000284 7 0/1/.../11
0x24                  $
0x84 0x02 0x00 0x08   Addr = 0x08000284
0x07 0x00 0x00 0x00   NumItems = 7 (0x7)
0x0B 0x00 0x00 0x00   StatType = 11 (0xB)
```


3.0.6.3 Responses

Value	Size	Meaning
RetVal	32-bit	≥ O.K., < 0 Error
ItemN	64-bit	Fetch /Exec /Skip counter value for <Addr> + N * 2.
xxxSum	64-bit	Sum of all Fetch /Exec /Skip counter value. Does NOT refer to the given address range but to ALL addresses that have been traced so far.

Command:

```
qSeggerSTRACE:GetInstStats:0x08000284 7 0/1/.../11
0x24                    $
0x84 0x02 0x00 0x08   Addr = 0x08000284
0x07 0x00 0x00 0x00   NumItems = 7 (0x7)
0x0B 0x00 0x00 0x00   StatType = 11 (0xB)
```


Response for <StatType> == FetchExecSkipSum

(Data based on example given earlier in this section)

```
<RetVal> NumItems * <FetchCnt> <FetchSum> NumItems * <ExecCnt> <ExecSum>
NumItems * <SkipCnt> <SkipSum>
```

> **Note**
>
> Needed buffer size: 4 + (8 * (3 * (NumItems + 1)))

```
0x00 0x00 0x00 0x00                     RetVal 0 => O.K.
0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...284 FCnt =   1 (0x01)
0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...286 FCnt =   1 (0x01)
0x64 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...288 FCnt = 100 (0x64)
0x64 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...28A FCnt = 100 (0x64)
0x64 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...28C FCnt = 100 (0x64)
0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...28E FCnt =   1 (0x01)*
0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...290 FCnt =   1 (0x01)*
0x30 0x01 0x00 0x00 0x00 0x00 0x00 0x00                ESum = 304 (0x130)
* Fetched for i == 100

0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...284 ECnt =   1 (0x01)
0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...286 ECnt =   1 (0x01)
0x64 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...288 ECnt = 100 (0x64)
0x64 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...28A ECnt = 100 (0x64)
0x63 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...28C ECnt =  99 (0x63)*
0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...28E ECnt =   1 (0x01)**
0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...290 ECnt =   1 (0x01)**
0x2F 0x01 0x00 0x00 0x00 0x00 0x00 0x00                ESum = 303 (0x12F)
*  Branch not taken for i == 100
** Executed for i == 100

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...284 SCnt =   0 (0x0)
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...286 SCnt =   0 (0x0)
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...288 SCnt =   0 (0x0)
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...28A SCnt =   0 (0x0)
0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...28C SCnt =   1 (0x01)*
0x63 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...28E SCnt =   0 (0x63)**
0x63 0x00 0x00 0x00 0x00 0x00 0x00 0x00 Addr 0x...290 SCnt =   0 (0x63)**
0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00                SSum =   1 (0x01)
*  Branch not taken for i == 100
** Only fetched for i == 100, so not even fetched but skipped for i != 100
```

Response for <StatType> == Fetch (deprecated)

```
<RetVal> NumItems * <FetchCnt>
```

> **Note**
>
> Needed buffer size: 4 + (8 * NumItems)

Response for <StatType> == Exec (deprecated)

```
<RetVal> NumItems * <ExecCnt>
```

> **Note**
>
> Needed buffer size: 4 + (8 * NumItems)

Response for <StatType> == Skip (deprecated)

`<RetVal> NumItems * <SkipCnt>`

> **Note**
>
> Needed buffer size: 4 + (8 * NumItems)

Response for <StatType> == FetchSum (deprecated)

`<RetVal><FetchSum>`

> **Note**
>
> Needed buffer size: 4 + 8

Response for <StatType> == ExecSum (deprecated)

`<RetVal><ExecSum>`

> **Note**
>
> Needed buffer size: 4 + 8

Response for <StatType> == SkipSum (deprecated)

`<RetVal><SkipSum>`

> **Note**
>
> Needed buffer size: 4 + 8

Response for <StatType> == NumInstSum (deprecated)

See `FetchSum`

Response for <StatType> == FetchExecSum (deprecated)

`<RetVal> NumItems * <FetchCnt> <FetchSum> NumItems * <ExecCnt> <ExecSum>`

> **Note**
>
> Needed buffer size: 4 + (8 * (2 * (NumItems + 1)))

Response for <StatType> == FetchSkipSum (deprecated)

`<RetVal> NumItems * <FetchCnt> <FetchSum> NumItems * <SkipCnt> <SkipSum>`

> **Note**
>
> Needed buffer size: 4 + (8 * (2 * (NumItems + 1)))

Response for <StatType> == ExecSkipSum (deprecated)

`<RetVal> NumItems * <ExecCnt> <ExecSum> NumItems * <SkipCnt> <SkipSum>`

> **Note**

> Needed buffer size: 4 + (8 * (2 * (NumItems + 1)))

3.0.7 qSeggerSWO:start

Starts collecting SWO data with the desired interface speed. The target is not being touched in any way, therefore you are responsible for doing the necessary target setup afterwards.

Syntax

`qSeggerSWO:start:<Enc>:<Freq>`

Parameter	Meaning
`Enc`	Encoding type, only 0 (“UART encoding”) is allowed. Hexadecimal.
`Freq`	The desired interface speed. Hexadecimal.

Response

`<ReturnValue>`
ReturnValue is “OK” or empty on error.

3.0.8 qSeggerSWO:stop

Stops collecting SWO data and returns the remaining bytes to be read from the buffer.

Syntax

`qSeggerSWO:stop`

Response

`<ReturnValue>`
ReturnValue is the hexadecimal number of bytes in the buffer or empty on error.

3.0.9 qSeggerSWO:read

Reads the specified number of SWO data bytes from the buffer.

Syntax

`qSeggerSWO:read:<NumBytes>`

Parameter	Meaning
`NumBytes`	Number of bytes to read (up to max. 64MB).

Response

`<ReturnValue>`
ReturnValue is a hex-encoded string or empty on error.

> **Note**
>
> The function will always return as much data bytes as requested. If more bytes than available are requested, excessive data has undefined values.

3.0.10 qSeggerSWO:GetNumBytes

Returns the amount of available bytes in the buffer.

Syntax

`qSeggerSWO:GetNumBytes`

Response

`<ReturnValue>`
ReturnValue is the hexadecimal number of bytes in the buffer or empty on error.

3.0.11 qSeggerSWO:GetSpeedInfo

Returns the base frequency and the minimum divider of the connected J-Link. With this information, the available SWO speeds for J-Link can be calculated and the matching one for the target CPU frequency can be selected.

Syntax

`qSeggerSTRACE:GetSpeedInfo:<Enc>`

Parameter	Meaning
`Enc`	Encoding type, only 0 (“UART encoding”) is allowed. Hexadecimal.

Response

`<BaseFreq>,<MinDiv>`

Value	Meaning
`BaseFreq`	Base frequency of the connected J-Link.
`MinDiv`	Minimum divider of the connected J-Link.

ReturnValue is empty on error.

Chapter 4

Support

Before contacting support, please make sure that the SEGGER wiki has been checked: SEGGER wiki

Support is provided via e-mail only. For more information regarding contact information, please refer to *here* on page 27

Chapter 5

Contact

SEGGER Microcontroller GmbH

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

