

Realtek

RLX Probe User Guide

V2.3.11

May. 8th, 2018

Realtek Semiconductor Corp.

<http://processor.realtek.com.tw>

This document is proprietary and confidential to Realtek Semiconductor Corp.

© 2016 Realtek Semiconductor Corp.

ALL RIGHTS RESERVED.

Contents

Contents	i
List of Tables	iii
List of Figures	iv
Chapter 1 Quick Start	1
Chapter 2 Product Overview	2
2.1 What It Supports	2
2.2 Functions	3
2.3 Snapshot of RLX Probe	4
Chapter 3 Installation And Configure	5
3.1 Installation on Windows	5
3.2 Installation on Linux	9
3.3 Connection	10
3.4 Configuration and Probe Commands	10
3.5 Run Debugger	40
3.6 CPU Register Alias In RLX Probe	40
3.7 FAQ	42
3.8 Debug Multi-Thread Processor with RLX Probe	46
3.9 RLX Probe Hardware Update	47
Chapter 4 GDB & Insight Basic Usage	48
4.1 Some Basic GDB commands	48
4.2 Debug Multi-Thread Processor with GDB	52
4.3 GDB script file	54
4.4 GDB Remote File I/O	56
4.5 Insight Introduction	57
Chapter 5 Operating Environment	58
5.1 Electrical Characters	58
5.2 Operating Temperature	58
Chapter 6 Virtual Probe Usage	59
6.1 Zebu JTAG XTOR	59
6.2 PXP APB XTOR	59
6.3 RTK APB/JTAG XTOR	60
Chapter 7 Support	61
Appendix A	62
Appendix B	91

List of Tables

Table 5.1: RLX Probe operating temperature.....	58
Table A.1: Revision History	91

List of Figures

Figure 2.1: Snapshot of RLX Probe1.0 Board and USB cable	4
Figure 3.1: RLX Probe Driver Setup Wizard Window	5
Figure 3.2: RLX Probe Driver License Window	6
Figure 3.3: RLX Probe Driver Install Location Window.....	6
Figure 3.4: RLX Probe Driver Install Completed Window	7
Figure 3.5: USB Device Driver Install Window 5.....	7
Figure 3.6: USB Device Driver Pre-Install Finish Window.....	8
Figure 3.7: USB Device Driver Window	8
Figure 3.8: USB Device Driver Ready Window	9
Figure 3.9: RLX Probe internal map.....	47
Figure 6.1: Debug overview with Virtual Probe for JTAG XTOR.....	59
Figure 6.2: Debug overview with Virtual Probe for APB XTOR.....	60

Chapter 1 Quick Start

1. Apply for an account with electric table from Realtek internal web.
Download Cygwin (needed only by Windows), RSDK, RLX Probe Driver from <http://processor.realtek.com.tw> and install them.
2. Connect RLX Probe to your target board with EJTAG cable.
3. Open `rlx_probe0.cfg` with any text editor and modify the parameters in it according to your target board.
4. Power on your target board.
5. Connect RLX Probe to your PC with USB cable.
6. Run RLX Probe Driver.
7. Debug your program with `rsdk` for LEXRA/RLX processor, `asdk` for ARM processor, or `msdk` for MIPS processor, also with RLX Debugger.
8. Enjoy your debug.

Chapter 2 Product Overview

2.1 What It Supports

- Processors
 - LX4180/LX5280/LX4280
 - RLX4181/RLX5181
 - RLX4281/RLX5281
 - RLX4081
 - RX3081
 - **Taroko MP**
 - RLX BUS Tracer
 - ARM Cortex-M0/M3/M4/M7
 - ARM Cortex-A9/A7/HAWK
 - ARM Cortex-R5/R7
 - KM0/4 & TM9
 - ARM Cortex-A53/ANANKE (AARCH32/AARCH64)
 - JTAG/SWD mode for ARM
 - ARM JTAG-AP/APB-AP/AHB-AP
 - APB/JTAG XTOR
- Software Platform
 - WIN2K / WINXP/VISTA/WIN7
 - Linux
- PC Port
 - USB1.1 (probe1.0)
 - USB2.0 (probe2.0) (Not supported Yet)
 - Ethernet (probe2.0) (Not supported Yet)

2.2 Functions

- Software and hardware Breakpoint
- Single step execution
- Display/Edit CPU register and memory
- Download program (Verify Load) and Run
- Flash Driver (Not supported Now)
- Read Memory Block to File
- Multi-Processor Debug
- MIPS16 Debug
- Bi-Endian Support
- Remote File I/O
- Memory Test
- Cache Operation
- Command line debug mode
- Debug in IMEM
- Multi-Thread CPU Debug
- PC Sample (Firmware version after V1.6)
- IO voltage changeable (probe2.0)
- Bus Tracer(probe2.0)
- Firmware Auto Update(probe2.0)
- ARM/Thumb Debug

2.3 Snapshot of RLX Probe



Figure 2.2.1: Snapshot of RLX Probe1.0 Board and USB cable

Chapter 3 Installation And Configure

Please apply for an account (License IP) from intranet of Realtek and download RLX Probe Driver from <http://processor.realtek.com.tw>.

3.1 Installation on Windows

Please follow the steps below to install it.

Step1: Double click setup file to start installation of RLX Probe Driver .



Figure 3.3.1: RLX Probe Driver Setup Wizard Window

Please click next to continue.

Step 2 : License Agreement window.



Figure 3.3.2: RLX Probe Driver License Window

The driver is free for use, click “I agree” if you would like to continue installation.

Step 3: Please select install location.



Figure 3.3.3: RLX Probe Driver Install Location Window

Click “install” to continue.

Step 4: Please wait until installation finish.



Figure 3.3.4: RLX Probe Driver Install Completed Window

USB driver is essential for RLX Probe, so please click “Install USB” to install the USB Device Driver the first time you install RLX Probe Driver.

Step 5: Click “Install” to begin the installation of USB Driver.

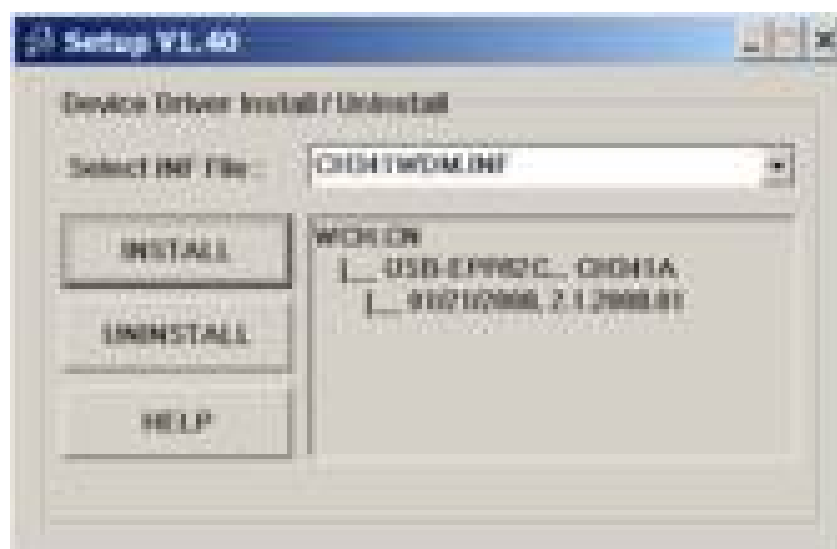


Figure 3.3.5: USB Device Driver Install Window 5

If USB driver has been installed before ,please click the “×” on the top right corner to close the window.

Step 6: The following window will be opened when USB Diver install finished.

Click “OK” to close it .



Figure 3.3.6: USB Device Driver Pre-Install Finish Window

Step 7: Close the USB Device installation window by click the “×” at the top right corner of the following window.

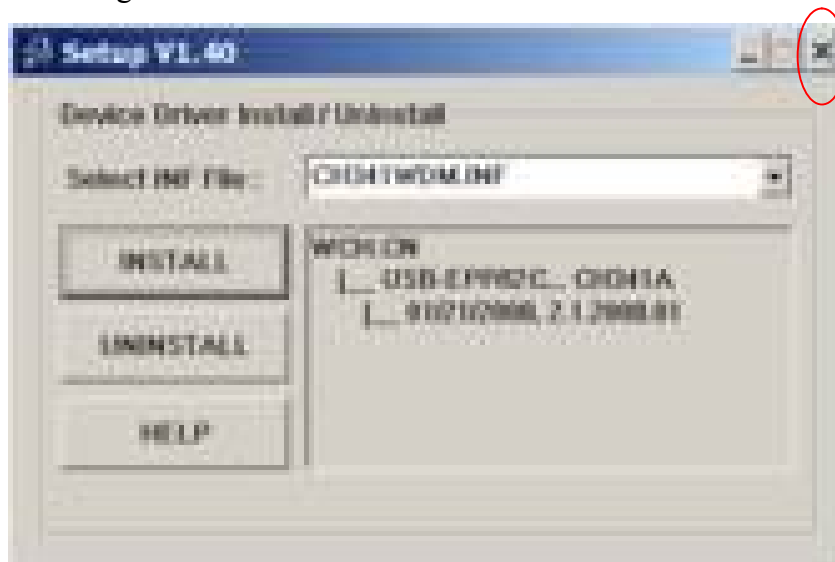


Figure 3.3.7: USB Device Driver Window

The USB Device Driver will not be completed until plug the RLX Probe USB cable in your PC. Please follow step 8 to complete the installation.

Step 8: Please connect RLX Probe to your PC with USB cable.

Wait for the Driver Software Installation finish automatically or select install automatically when new hardware found. Omit the digital signature warning if any.

The following window (WINXP or VISTA system) means USB device ready to use now. Just close it.



Figure 3.3.8: USB Device Driver Ready Window

One external device (USB-EPP....) will appear in device manager of system after correct installation of RLX Probe. The more probes, the more external devices.

For USB20, please use “ftdibus.inf” under probe driver installation directory for installation.

3.2 Installation on Linux

Step1: Download and Uncompress

Download RLX Probe Linux Driver setup file (for example: RLX_Probe_Driver_Linux_v1.3.0.tar.gz) from website:
<http://processor.realtek.com.tw>.

Then, uncompress the file to your destination directory under Linux with command “tar -zxvf filename”.

Step2: Install USB11 device.

1. Please confirm if you have the authority to install soft on the PC.
2. Enter the directory of USB11_Driver and run “insmod ch341_epp.ko”.
3. Run “cat /proc/modules” to check if the device ch341_epp was installed correctly.
4. Check the device file “ch341_p0” with command “find ch341_p0” under “dev” directory.

If more probes connected to the PC, device “ch341_p1, ch341_p2.....” will be found.

If failed to find ch341_p0 after installation, please run

“mknod /dev/ch341_p0 c 200 0” to make a nod for USB Device of RLX Probe.

If failed to install USB driver to your Linux system, please follow the steps below to generate a new module.

1. `cp -r USB_Driver /root`
2. `make`
3. `insmode ch341_epp.ko`

If failed to find `ch341_p0` again, please run

“`mknod /dev/ch341_p0 c 200 0`” to make a nod for USB Device of RLX Probe.

Step3: Install USB20 device . (Please read `readme.dat` in `USB20_Driver` firstly.)

1. As root user copy the following files to `/usr/local/lib`
`cp libftd2xx.so.1.0.2 /usr/local/lib`
2. Change directory to `/usr/local/lib`
`cd /usr/local/lib`
3. make symbolic links to these files using the following commands:
`ln -s libftd2xx.so.1.0.0 libftd2xx.so`
4. Change directory to `/usr/lib`
`cd /usr/lib`
5. make symbolic links to these files using the following commands:
`ln -s /usr/local/lib/libftd2xx.so.1.0.2 libftd2xx.so`

3.3 Connection

There are two kinds of cable to connect your target board, please select the correct cable.

Be careful, the red line of the cable is the pin 1 of the connector, and the signal of the line is “TRST”. Signal define of your connector on target board should follow EJTAG standard.

3.4 Configuration and Probe Commands

For each processor or bus tracer debug, there should be a configure file

corresponding to it.

If multi-processor debug is required, please copy the file `rlx_probe0.cfg` to `rlx_probe1.cfg`, `rlx_probe2.cfg` ..., and modify them according to target processor. Set parameters needed in corresponding configure file (`rlx_probe*.cfg`).

3.4.1 Basic Rules for Configuration

- i. Please use “0x” as header for hexadecimal data, “0n” as header for decimal data, default for decimal data.
- ii. Please bracket string with “” (no need for variable name).
- iii. “YES” and “yes” are both supported; the same with “NO” and “no”.
- iv. Please do not separate a string or a command expression to different lines.
- v. Please use “//” or “/* */” for comments mask or invalidating an option (C language syntax).
- vi. There are two methods to execute probe commands.
 - a. Put commands in the configure file `rlx_probe0.cfg`.
 - b. Put commands in a new script file. For example: `script.txt`, and add a line “`exec script.txt`” or “`fr c script.txt`” to the end of configure file `rlx_probe0.cfg`.
- vii. Please do not define label both in `rlx_probe*.cfg` and script file.
- viii. All configure options can be deleted or masked from the configure file and probe will use default value.
- ix. For variable, “`@$VAR`” is same with “`$VAR`”, it means the value of variable `$VAR`, while “`@@$VAR`” means the data value in memory address pointed by variable “`VAR`”.

3.4.2 Configure Options

1. Processor

Define the target to be debugged.

Example: `PROCESSOR = “RLX5181”;`

Target type supported:

MIPS32 (general MIPS32bit CPU), MIPS4KE, MIPS4KEC, MIPS24K,
RLX5181, RLX4181, LX4180, LX5280, RLX5281, RLX4281,

RLX_BUS_TRACER, AUTO, CORTEX-M3

If “AUTO” being set, RLX Probe will auto detect the processor type.

2. Port

Define TCP port used for GDB to connect to probe.

Example: port = 5181; //decimal, **MUST NOT** add “0x” “0n” in front of the decimal value here.

Make sure GDB also use TCP port 5181 for connection.

3. ACC_JTAG_TAP_NUM

Define the JTAG TAP unit number of target processor to be accessed on the JTAG chain. First JTAG tap unit is the one of which TDO signal is connected to RLX Probe whose number should be 1. And do not modify it when you have only one processor and no other JTAG unit on the same JTAG chain.

Example: ACC_JTAG_TAP_NUM = 1.

4. USB_ID

Select the probe number connected to your PC for multi-probes.

Example: USB_ID = 0;

Number 0 means the first probe connected to PC and so on (1,2,3...).

5. BIG_ENDIAN

Define the endian of target processor.

Example: BIG_ENDIAN = “YES”.

“NO” for little endian.

6. TCK_FREQUENCY

Available frequency of TCK is 16MHz, 8MHz, 4MHz, 2MHz, 1MHz,

512 KHz, 256 KHz, 128 KHz, and 16 KHz.

Example: TCK_FREQUENCY = "8M"; (recommended)

7. RESET_TARGET_PROCESSOR

RESET_TARGET_PROCESSOR = "YES".

When this option is enabled, RLX Probe will reset target board and processor before your connection, and probe will reset target processor each time you re-connect to probe driver with GDB.

8. GDB_MESSAGE_SHOW

Show GDB packet in RLX Probe Driver window or not.

Example: GDB_MESSAGE_SHOW = "NO".

Be careful that set this option to "YES" will decrease the speed of debug.

9. DBG_MESSAGE_SHOW

Show debug messages in RLX Probe Driver window or not.

Example: DBG_MESSAGE_SHOW = "YES".

10. GEN_LOG_FILE

Generate log file for debug messages or not.

Example: GEN_LOG_FILE = "YES".

11. DEJAGNU_TEST

There will be some address access limit if you set this option to "YES".

This option is **only** defined for DejaGnu test, so you must set this parameter to "NO" (or do not add the item to configure file) when you debug your own program.

Example: DEJAGNU_TEST = "NO".

12. FLASH DRIVER

FLASH_DRIVER

Enable or disable flash driver.

Example: FLASH_DRIVER = "NO".

If FLASH_DRIVER is set to "YES", two other options: FLASH_BASE and FLASH_FILE are required.

FLASH_TYPE

Example: FLASH_TYPE = 1; // 1 for SPI flash, 0 for parallel flash

FLASH_BASE

Set the base address of flash.

Example: FLASH_BASE = 0xbf00000.

FLASH_FILE

Set path and name of your image file to be downloaded. Please input full path to avoid error. RLX Probe only support image file.

Example: FLASH_FILE = "e:\boot\bootloader.bin".

Note: Actually, we recommend use GDB script and your own flash driver to program your target flash (please refer to examples under installation directory). If no usable random memory on target board, please mask all "mc" region to disable probe download flash driver (avoid execute flash driver from target memory).

13. FLASH_CHIP_ERASE

If you need to erase flash chip but do not download anything to flash, Please set this option to "YES" and set "FLASH_DRIVER" to "NO".

Example: FLASH_CHIP_ERASE = "NO".

14. FLASH_SEC_ERS

This option is for special user to avoid such special connection error as there are some error codes at boot address. Please never set it to "YES" under common use. This option will erase first sector of flash.

Example: FLASH_SEC_ERS = "NO".

15. RST_TARGET_WHEN_DEAD

Enable reset target board when driver found target dead. Be careful that this option will not work until reset of target board was connected to the signal J_RST of the EJTAG connector.

Example: RST_TARGET_WHEN_DEAD = "NO".

16. DMEM & DMEM1

This option defines the DMEM region (in physical address) you used in your program. If no DMEM used in your program, please invalid the option.

Example: DMEM = RANGE (0x00010000, 0x00011fff);

If DMEM set, RLX Probe will access these regions with JAM mode since DMA access will get data from physical memory directly.

17. IMEM & IMEM1

This option defines the IMEM region (in physical address) you used in your program. If no IMEM used in your program, please invalid the option.

Example: IMEM = RANGE (0x00000000, 0x0000ffff);

The address range must be strictly set.

If there is another IMEM in your target processor, please set the range of it with the following format:

Example: IMEM1 = RANGE (0x10000000, 0x10001000);

18. MC

Set the access mode to a memory region while these settings will be **only** valid for read-only region configure.

Abbreviation description:

MC: syntax header.

NON/NONE: neither readable nor writeable. (Only for memory)

RO: Read only memory.

WR: readable and writeable. (for RAM)

WRL: limited read/write. (for some registers)

DMA: read/write this memory region with DMA mode

JAM / PRACC: read/write this memory region by processor execute instruction.

Example: MC 0x00000000, 0x003ffff, RO

The example means setting memory range (0x0-0x3ffff) to read only.

19. Read memory block to file

MEM_READ2FILE

Enable to read memory to file function or not.

Example: MEM_READ2FILE = "NO".

When MEM_READ2FILE is set to "YES", three other options:

MEM_READ_START_ADDR, MEM_READ_END_ADDR and MEM_FILE_NAME are required.

MEM_READ_START_ADDR

Define the start address for memory read.

Example: MEM_READ_START_ADDR = 0xbfc00000.

MEM_READ_END_ADDR

Define the end address for memory read.

Example: MEM_READ_END_ADDR = 0xbfc10000.

MEM_FILE_NAME

Define destination path and file name for the memory read operation.

Example: MEM_FILE_NAME = "e:\Realtek\rosboot.bin".

20. Memory block initialization

MEM_INIT_WRITER

Enable memory block initialization or not.

Example: MEM_INIT_WRITER = "NO".

When MEM_INIT_WRITER is set to “yes”, three other options: MEM_WR_START_ADDR, MEM_WR_END_ADDR and MEM_WR_FILE_NAME are required.

MEM_WR_START_ADDR

Define the end address for memory block initial.

Example: MEM_READ_END_ADDR = 0x80010000.

MEM_WR_END_ADDR

Define the end address for memory block initial write.

Example: MEM_WR_END_ADDR = 0x80020000.

MEM_WR_FILE_NAME

Define destination directory and file name for the memory block initial.

Example: MEM_WR_FILE_NAME = “e:\share\stub_readback.bin”. If File name defined, the “MEM_WR_END_ADDR” will be ignored.

MEM_FILL_VALUE

If you want to fill memory with some defined data (such as 0), you can set the “MEM_FILL_VALUE” in byte mode ,and the file (named by MEM_WR_FILE_NAME) will be ignored and the value takes effect.

Example: **MEM_FILL_VALUE = 0x00;**

21. Enter console mode (command line debug mode)

console/PROG_MODE/TRACE_MODE

If “console”, “PROG_MODE” or “TRACE_MODE” is added to the end of rl_x_probe0.cfg, probe will enter prompt mode. Prompt mode is designed for BUS Tracer debug. In prompt mode, you may input all kinds of RLX Probe command such as “ew” or “mw” and RLX Probe will execute them and show result. In this mode, you can do some simple debug without GDB.

22. FAST_LOAD_DISABLE

If the keyword is added to configure file, then fast-load under no-DMA mode will be disabled (only for processor without EJTAG DMA).

23. DMA_DISABLE

If the keyword DMA_DISABLE add to configure file at any line, EJTAG DMA mode will be disabled.

24. TEST_PROBE_MEM

Enable read/write probe memory test or not.

Example: TEST_SPP = "NO".

25. TEST_TAP

Enable TAP test or not. Probe will read/write EJTAG control registers and check if there is any mismatch when this option is set to "YES".

Example: TEST_TAP = "NO".

26. TEST_DMA

Enable DMA operation test or not. Probe will read/write RAM (found automatically by probe) with DMA mode and check if there is any mismatch. Do not set to "YES" if there isn't DMA support in DSU of target processor.

Example: TEST_DMA = "NO".

27. TEST_MEM

Enable memory address and data line test or not. Probe will test total memory region you set (first write/read region).

Example: TEST_MEM = "NO".

28. TEST_DOWNLOAD_SPEED

Enable probe memory download speed test or not. Download speed will show on the screen after test finished.

Example: TEST_DOWNLOAD_SPEED = "NO".

29. TEST_CONTINUOUS

If this option is set to “YES”, driver will run one test above continuously.

Example: TEST_CONTINUOUS = “NO”.

30. FILE_IO_BASE

Define the file IO base address for remote file IO when you want to use address **other than** default address 0x80000090.

Example: FILE_IO_BASE = 0x80010000.

31. LEXRA_CP0_SHOW

Define the response to GDB registers information fetch. If it’s set to “yes”, RLX Probe Driver will send registers of CP0 GDB.

Be careful that, this option and option 32 are only supported by the newest RSDK version (1.3.6 and 1.4.2 or upper).

32. CP3_REGS_EXIST

Define if there is CP3 registers need to be debug by GDB. If it is added to probe config file, and the “LEXRA_CP0_SHOW” is set to “yes”, RLX Probe will read CP3 for you and send the registers to GDB. You can also write to them by GDB after that.

33. SOFTBP_DISABLE

Disable soft instruction breakpoint when you need to debug code in flash or some read only address. RLX Probe will help GDB to set software instruction breakpoint by setting hardware instruction breakpoint. Please be careful the instruction breakpoint number should not exceed the maximum number of hardware breakpoint number (which will be showed in RLX Probe Driver’s UI).

34. FILE_IO_DISABLE

Disable File IO which will avoid Probe failed to access some bad DEPC (not accessible address).

35. DMA_ENABLE_ALL

Access all address with DMA mode when this option added.

36. QUIT_DEAD_AUTO

RLX Probe will show the error message and wait for user's input for quit if without this option in configure file. If the option added to configure file, RLX Probe will quit when dead without user's intervention.

37. IMEM_READ_WRITE

Enable IMEM load/read.

If this option added to configure file and IMEM region set, RLX Probe will directly access the IMEM when GDB read/write address allocated in IMEM.

38. RECONNECT_INIT&RECONNECT_INIT_OFF

Re-initialize the probe or not (RECONNECT_INIT_OFF) (execute the total configure options and initial commands) when reconnected by GDB.

39. P_CMD_ENABLE&P_CMD_DISABLE

Enable or disable p packet of GDB. This p packet is disabled by probe driver (default) because p packet will decrease the connection speed.

40. IF_TYPE

Define the interface between PC and Probe.

Example: IF_TYPE = "USB11";

Options supported are: USB11 (default)

USB20 (for Probe 2.0)

Ethernet (for Probe 2.0)

41. FIRMWARE_AUTO_UPDATE

Enable/disable firmware auto update through USB2.0 interface (for probe2.0).

Example: FIRMWARE_AUTO_UPDATE=no;

42. FIRMWARE_FILE_NAME

Define the firmware auto update file name.

Example: FIRMWARE_FILE_NAME = "rlx_probe20_usb.xsvf";

43. MSDK_DEBUG

Define the MSDK debug mode for that register number in MSDK GDB packet is different with that in RSDK or SDE. For debugging with MSDK, this option **must** be added.

44. PC/Data Sample

Set the PC Sample file name (in-house processor not supported now).

Example: PCSAMPLE_FILE_NAME = pc_sample.txt;

We can only use maintenance packet of GDB to enable/disable pc/data sample from gdb command line till now.

Example: maint packet qrealtek.pcsample.c (continue and begin pc sample)

Example: maint packet qrealtek.pcsample.stop (stop the pc sample)

Example: maint packet qrealtek.pcsample.setdiv (set clock divider)

45. Interrupt/Soft-reset/TLB mapped address/IMEM Access Control

We can only use maintenance packet(GDB command line) to do some advanced debug.

Example: maint packet qrealtek.interrupt.disable

Example: maint packet qrealtek.interrupt.enable

Example: maint packet qrealtek.softreset.enable

Example: maint packet qrealtek.softreset.disable

TLB mapped address access enable or disable

Example: maint packet yACCTLBMAP

The example will enable TLB mapped address access.

Example: maint packet yPRTTLBMAP

The packet will disable TLB mapped address access by probe.

IMEM read/write with maint packet:

Example: maint packet yimemr: 0x00010000 // read IMEM

Example: maint packet yimemw: 0x00010000=0x12345678
//write to IMEM

46. TC set

We can set TC number on multi-thread processor.

Example: maint packet qrealtek.tcset.(number)

47. NON_INTRUSIVE_DEBUG

When this option is added to RLX Probe configure file, RLX Probe will not trigger CPU to debug mode and access memory (no CPU register can be accessed) through DMA mode. User can debug as same as before, editing memory or memory address mapped register after connecting to probe driver with GDB or directly under console mode.

48. RST_CPU

If RST_TARGET_PROCESSOR not enabled and this option added to configure file, RLX Probe Driver (after 2.2) will reset CPU only when start or re-connect.

49. RST_PERIPHERAL

If RST_TARGET_PROCESSOR not enabled and this option added to configure file, RLX Probe Driver (after 2.2) will reset all peripherals only (without reset CPU) when start or re-connect.

50. L2MEM

Define the physical address range for L2MEM.

Example: L2MEM = RANGE (0x00000000, 0x00007fff);

51. TLB_MAPPED_ADDRESS_PROTECT& TLB_MAPPED_ADDRESS_ACCESS

If the option “TLB_MAPPED_ADDRESS_PROTECT” added to configure file, probe driver will report error when user try to access memory in TLB-Mapped region (default setting). Please use maintenance packet to switch on/off the access permission.

If “TLB_MAPPED_ADDRESS_ACCESS” set, probe driver will access the tlb-mapped memory address and TLB exception will occur in debug mode when the region hasn't been mapped by TLB.

52. TRIG_WHEN_CONNECT

If the option added to configure file, probe driver will not trig processor to debug mode until GDB connect to it. This option is useful for multi-processor debug, such as one processor start firstly and other processors remain in reset state until the first processor initialize over.

53. FPU_REGS_EXIST

When in-house processor has FPU, please add "FPU_REGS_EXIST" to configure file for probe to read/write FPU registers.

54. CM3_FPB_HB_ENABLE

When the option add to configure file, probe driver will Use Flash Patch and Breakpoint component (for CM3) as hardware breakpoints.

55. FAST_LOAD_ENABLE

If the option "FAST_LOAD_ENABLE" added to configure file, probe driver will use fast load mode to load a binary file to target memory. (Before probe driver v2.3.1, default is enabled; and after v2.3.1, default is disabled)

56. PAUSE/RESUME

If "PAUSE" is added to rl_x_probe0.cfg, probe driver will pause to parse the configure file, and enter prompt mode. If "RESUME" is typed, probe driver will resume to parse the configure file.

57. Enable Exception Debug Trap for ARM Cortex-M

If the option "CM_EXP_ALL_EN" added to configure file, probe driver will enable debug trap on all exceptions for Cortex_M.

If the option "CM_EXP_HARD_BUS_ERR_EN" added to configure file, probe driver will enable debug trap on a HardFault exception or a BusFault exception.

If the option "CM_EXP_HARD_ERR_EN" added to configure file, probe driver will enable debug trap on a HardFault exception.

If the option "CM_EXP_BUS_ERR_EN" added to configure file, probe driver will enable debug trap on a BusFault exception.

If the option "CM_EXP_CHK_ERR_EN" added to configure file, probe driver will enable debug trap on a UsageFault exception due to a checking error.

If the option "CM_EXP_INT_ERR_EN" added to configure file, probe driver will enable debug trap on a fault occurring during an exception entry or return sequence.

If the option "CM_EXP_MEMM_ERR_EN" added to configure file,

probe driver will enable debug trap on a MEMManage exception.

If the option “**CM_EXP_NOCP_ERR_EN**” added to configure file, probe driver will enable debug trap on a UsageFault access to a Coprocessor.

If the option “**CM_EXP_STAT_ERR_EN**” added to configure file, probe driver will enable debug trap on a UsageFault exception due to a state information error.

58. ARM_AHB_BUS_ENABLE

If “**ARM_AHB_BUS_ENABLE**” is added to `rlx_probe0.cfg`, and AHB-AP exists, probe driver will exchange data on AHB bus.

Otherwise, probe driver will exchange data on APB bus.

59. WRITE_ALIGN_4

If “**WRITE_ALIGN_4**” is added to `rlx_probe0.cfg`, probe driver will always write data to memory by word-access mode, that is if a byte-data or a half-word data will be written, probe driver will read a word-data from the write address and modify the data, and then write back to memory.

60. BYTE_HW_READ_ENABLE

If “**BYTE_HW_READ_ENABLE**” is added to `rlx_probe0.cfg`, when a byte data or a half-word data should be read back, probe driver will access the memory by byte or half-word mode. This option is available for MIPS and ARM.

61. ARMA_VFP_ON

If “**ARMA_VFP_ON**” is added to `rlx_probe0.cfg`, and VFP is implemented, probe will enable to read/write VFP registers.

62. ARMA_NEON_ON

If “**ARMA_NEON_ON**” is added to `rlx_probe0.cfg`, and NEON & VFP is implemented, probe will enable to read/write VFP & NEON registers.

63. ARM_SWD_EN

If “**ARM_SWD_EN**” is added to `rlx_probe0.cfg`, probe will enable SWD mode to debug (Only for ARM). And probe debug in JTAG

mode by default.

64. ACC_ARM_AP_NUM

Define the AP number of an ARM Processor to be accessed in the DAP module. Please do not add this option if there is only one MEM-AP which can be accessed. And the setting AP number starts from Zero (0). The right AP number can be show in a init flow of ARM DAP Scan when RLX Probe driver starts.

Example: ACC_ARM_AP_NUM = 0.

65. TAP_IR_LEN

TAP_IR_LEN is used to define the tap ir length for user. If tap ir length is 5-1-5 in turn, and ir length can not be auto-scanned by RLX Probe, this option should be added into your configuration file.

Example: TAP_IR_LEN = (5, 2, 4, 0, 0, 0, 0, 0);

66. XTOR_TCP_PORT/XTOR_TCP_SERVER

Both of these options are used for RTK APB or JTAG XTOR of Virtual Probe. XTOR_TCP_SERVER is used to define the server Name or IP on which virtual environment is running. And XTOR_TCP_PORT is used to define the port number for Virtual Probe connection.

Example: XTOR_TCP_SERVER = 172.29.26.11;

XTOR_TCP_SERVER = rs64_inx222;

XTOR_TCP_PORT = 50001;

67. ARM_JTAG_AP_EN

This option is for a MIPS processor by ARM JTAG-AP connection. And when

68. SHOW_AP_LOG

This option is used to enable to show ARM AP Access log when RLX probe starts the init flow.

69. IDAU_BASE_ADDR

This option is used to set IDAU Base Address which is used to show ARM IDAU Region.

3.4.3 Supported Probe commands or script

1. ew/eh/eb

Write data to target memory or variable.

ew: write a word to memory or a variable.

Example: ew 0xbd011000=0x5f3ffff

Which means write a word value of 0x5f3ffff to address 0xbd011000.

ew \$a = 1; //means set variable “a” to 1.

eh: write half word to address.

Example: eh 0xbd011000=0x5f3f.

eb: write a byte to address .

Example: eb 0xbd011000=0x5f.

Loop write supported.

Example: ew 0xbd011000 1 3 = 0x7f3ffff 0x1234 0x5678

Probe will write data 0x7f3ffff to address 0xbd011000, write data 0x1234 to address 0xbd011004 and write data 0x5678 to address 0xbd011008.

Example for all register writes with register number:

Example: ew .112, 0x12345678 (write 0x12345678 to register 112)

Please refer to Appendix A (all register number map).

2. rw/rh/rb/mw/mh/mb

Read data from an address and store it to a variable.

Rw/mw: read a word

Example: rw \$abc, 0xbd011000;

Which means read a word from address 0xbd011000 and save it to the variable “abc”.

Rh/mh : read half word

Example: rh \$abc, 0xbd011000.

Rb/mb : read a byte

Example: rb \$abc, 0xbd011000.

If no variable need to be set, probe will display the value read from memory or register.

Example: mw 0xbd011000

Example for all register read with register number:

Example: mw .112 (read register 112)

Please refer to Appendix A (all register number map).

3. dw/dh/db

dw/dh/db [op1]

Display word/half-word/byte size objects in specified range.

Example: dw 0x80000000

Display a word in address 0x80000000.

4. delay/w

Delay for a while and driver will do nothing during the time.

Example: delay/w 200

This means delay for 200 ms.

5. dv

Display messages and variables in some format on the terminal.

Example: dv "Hello world! Variable abc = %lx\n", \$abc

It will print "Hello world! Variable abc = " and value of \$abc in hexadecimal to the display terminal.

Print format include %s, %d, %lx .etc, it's same as "printf" function in C language

6. +Q/-Q

Using “+Q” to enable debug message show in console, “-Q” to disable message show.

7. g/c/r/R

g [=op1]

Example: g = 0x80008000

CPU will free run begin at address 0x80008000.

Op1 specifies start address; this command is always used together with “stop”.

In a command script, if there is commands after “g” command, this command will not be executed until CPU get into debug mode (such as : CPU encounter breakpoints) .

R/r

Run the target program. RLX Probe Driver will automatically run begin at the entry of ELF file.

8. C

Continue from the last stop.

9. si/SI/step

Single instruction step.

10. stop/k/sp

Trigger processor to debug mode or stop the target program.

11. cache

cache [op1] [op2]

op1:

i/I : Icache

d/D : Dcache

imem

dmem

op2 :

inv/clr/clear: cache invalidate

wb: data cache write back

wbinv: write back and invalidate

lock1: lock cache to way 1 (for LX/RLX processor)

lockdown: Lock cache down (for LX/RLX processor)

fill: Imem fill

off: Imem/Dmem off

on : Dmem on

12. flash

flash [op1] [op2]

op1 :

e : erase

w : program

op2 :

erase : flash base address

program : filename

op3 :

flash base address

op4 :

flash program destination address

Example : flash e 0xbfc00000

It will erase the flash totally from base address 0xbfc00000.

Example : flash w filename 0xbfc00000 0xbfc10000

It will program data of binary file to flash begin at address 0xbfc10000 while the base address of flash is 0xbfc00000.

13. dump

dump [op1] [op2] [op3]

Dump data of target memory to file.

Op1: base address

op2: data size

op3: file name

Example: dump 0x80000000 0x1000 e:\file.dump

It will dump 0x1000 bytes (from 0x80000000 to 0x80001000) of target memory to the file “e:\file.dump”.

14. load

load [op1] [op2]

Load binary file to target memory.

Op1 : full path name including path

op2 : base address

Example : load e:\file.bin 0x80000000

It will load file “e:\file.bin” to address 0x80000000 of target memory.

15.1

load [op1]

Load elf (executable file) to target memory.

Op1: file name.

Example: l d:\vmlinux.elf

Since address information already exists in the elf file, there is no need to specify target address.

16. exec/source

exec [op1]

Execute the command script file. Default script file postfix is “cmd”.

Op1: command script file full name.

Example: `exec e:\script.txt`

17. exit

Exit the driver.

In a command script, “q” and “q y” will exit the driver, while “q n” will not exit the driver.

18. if

Condition statement.

Example: `if (expr) {statement }`

When the expression is true, the following statement will be executed, or else, not.

Expression and statement must be in the same line.

19. :labelname

Define a label with “:labelname”(a colon must be added before your label). Label is always used together with “goto” command.

20. goto

Example: `goto labelname`

Jump to label and execute commands from there, the label can be defined either above or below the “goto” sentence.

21. mt

`mt [op1] [op2] [op3] [op4]`

Test memory region address from op1 to op2.

Op3: test mode.

Op4: test pattern (only for mode 1).

Op1: start address of the test memory

op2: end address of the test memory

op3: this field is optional, defines the mode of test, default value of op3 is 9.

Op3 = 1, Basic pattern, scan all memory address in the range

op3 = 2, walking “0” & “1” test

op3 = 3, Rotating address test

op3 = 5, partial write test

op3 = 9, default value, run all memory test above

Example: mt 0x80000000 0x81000000

It will run memory test of all modes from address 0x80000000 to 0x81000000 in target memory which is the same with “mt 0x80000000 0x81000000 9”.

Example: mt 0x80000000 0x81000000 1 0x12345678

It will test memory region 0x80000000 to 0x81000000 with pattern 0x12345678.

22. rp

Reset the processor and target board.

23. moni

Monitor specified address in target memory.

Example: moni 0x80000000 1 3 10

Monitor 1,3 and 10 bit in address 0x80000000 of target memory , and show the value when any of the bits changed.

24. fr

fr [op1] [op2] [op3]

op1: “m” or “c”, “m” means load file to memory, “c” to execute a command script file

op2: full name of file.

op3: target memory address when op1 is “m”

Example: “fr m d:\file.load 0x80000000”, load the file “d:\file.load” to target memory address begin at 0x80000000.

Example: “fr c script.txt”, execute the command script file “script.txt”.

25. fw

fw [op1] [op2] [op3]

op1: “o” or “a”, overwrite or append mode for file write

op2: full path name

op3: start address of target memory

op4: size of memory

Read a memory region, and write to the specified file. The default file write mode is overwriting mode.

Example: fw o d:\temp.file 0x80000000 0x100

It will read 0x100 bytes from address 0x80000000 of target memory, and write to file d:\temp.file with overwrite mode.

Example: fw a d:\temp.file 0x80000000 0x100

It will read 0x100 bytes from address 0x80000000 of target memory, and write to file “d:\temp.file” with append mode.

26. fo

Open an ELF file for debug.

27. lf

Load target ELF file to target memory.

28. b/bs

Set breakpoint at address or symbol.

Example: b 0x80000000

Example: b main

29. hb

Set hardware breakpoint at instruction address.

Example: hb 0x80000000

30.hbc

Clear hardware breakpoints/watchpoint (write mode).

Example: hbc (clear all hardware breakpoints)

Example: hbc 0x80000000 (clear hardware breakpoint at address)

31. q/quit/exit

Quit RLX Probe Driver.

32. B

List all breakpoints.

33. D

Delete all breakpoints.

34. config/switch

config/switch configure file number (start from 0)

switch 0 : means to switch to the target defined in rlx_probe0.cfg.

Switch control to another target which defined in the corresponding configure file under command line mode.

35. ei/mi

IMEM read/write.

Example: ei 0x80000000 0x12345678

Write data to IMEM address at 0x80000000.

Example: mi 0x80000000

Read data from IMEM.

36. ldimem/dpimem

Write a image file to IMEM or read IMEM to a file.

Example: ldimem e:\ej_test.bin 0x80000000

Example: dpimem e:\ej_test.bin 0x80000000 0x300 (length)

37. sib

Single step binary code.

38. ej/mj

Read/write EJTAG register.

EJTAG defaults Registers Number and name alias are following:

0x0: "EXTEST"	0x1: "IDCODE"	0x2: "SAMPLE"	0x3: "ImpCode"
0x4: "INTEST"	0x5: "HI-Z"	0x6: "CLAMP"	0x7: "BYPASS1"
0x8: "EJTAG_ADDRESS_IR"	0x9: "EJTAG_DATA_IR"	0xa: "EJTAG_CONTROL_IR"	0xb: "EJTAG_ALL_IR"
0xc: "EJTAG_BOOT_IR"	0xd: "NORMALBOOT_IR"	0xe: "FASTDATA_IR"	0xf: "UNDEFINED0XF"
0x10: "PCTRACE_CTLA"	0x11: "PCTRACE_CTLB"	0x12: "PCTRACE_TCBDATA"	0x13: "PCTRACE_CTLC"
0x14: "UNDEFINED 0X14"	0x15: "PCTRACE_CTLD"	0x16: "PCTRACE_CTLE"	0x17: "UNDEFINED 0X17"
0x18: "UNDEFINED 0X18"	0x19: "UNDEFINED 0X19"	0x1a: "UNDEFINED 0X1A"	
0x1b: "UNDEFINED 0X1B"	0x1c: "UNDEFINED 0X1C"	0x1d: "UNDEFINED 0X1D"	
0x1e: "UNDEFINED 0X1E"	0x1f: "BYPASS2"		

Example: ej 0xa 0x12345678 (write 0x12345678 to EJTAG register 0xa)

Example: mj 0xa (read EJTAG register 0xa)

39. ep/mp

Read/write probe memory.

40. epr/mpr

Read/write probe internal register.

41. polling_en/polling_dis

Enable/disable probe polling PRACC.

42. pds/pdsp

pds : start PDTrace sampling function in RLX Probe.

Pdsp: stop the PDTrace sampling function in RLX Probe.

43. pddump

pddump : Dump PDTrace data to file.

44. pdparse

Pdparse : parse the PDTrace raw data file to a text file.

Example: pdparse trace_data.bin trace.txt

45. pdover

Check if pdtrace operation over.

46. pdcab

Run pdtrace calibration.

47. expression

RLX Probe support general arithmetic and logic expression for calculation. And “@@0x...” for memory read and write. “@(0x...)” and “@0x...” for memory read expression are supported in driver version later than 2.2.9.

48. rstboard

RLX Probe support to set the JTAG signal ‘Jrst’ low by using the

command 'rstboard' in driver version later than 2.3.4.

Example: rstboard or rt

48. rstjtag

RLX Probe support to set the JTAG signal 'TRST' low by using the command 'rstjtag' in driver version later than 2.3.4.

49. fopen

fopen: open a file to operation

Example: fopen \$ FILE_HDL,"c0.bin",wb (open c0.bin to write data)

fopen \$ FILE_HDL, @\$FILE_NAME (open a file named in \$FILE_NAME to read data)

50. fclose

fclose: close a file

Example: fclose @\$ FILE_HDL

51. fread

fread: read data from file

Example: fread \$DATA,@\$FILE_HDL,@\$OFFSET,@\$DATA_SIZE

Read \$DATA_SIZE bytes data from \$FILE_HDL starting from the offset \$OFFSET, and store data to \$DATA.

52. fwrite

fwrite: write data to a file

Example: fwrite @\$DATA,@\$DATA_SIZE,@\$FILE_HDL

Write \$DATA_SIZE bytes data stored in \$DATA to the file \$FILE_HDL

53. ftell

Example: ftell @\$FILE_HDL,\$FILE_LOC

Get the iostream location from the file \$ FILE_HDL and store in \$FILE_LOC

54. fsize

Example: fsize @\$FILE_HDL

Get the size of the file \$FILE_HDL

55. watch/rwatch

Set watchpoint/rwatchpoint at specific address.

Example: watch 0x80000000

56. rwatchc

Clear rwatchpoint.

Example: rwatchc 0x80000000 (clear hardware breakpoint at address)

57. pmtrace

1)pmtrace start: start pm poptrace and get some trace data.

Example: pmtrace start 0x1000 ir

pmtrace start 0x1000 sw

pmtrace start 0x1000 mem 0xb021fd08 2 0xb021fd04 2

2) pmtrace stop: Reset Probe status, and stop pmtrace.

Example: pmtrace stop

3) pmtrace dump: Dump poptrace data from probe memory.

Example: pmtrace dump 0x1000 pmtrace0.bin

3.4.4 A script file example

```
+Q                                     //show message in console
ew $CAL_MODE                         = 0n1  // create a variable , init it with value 1(decimal)
ew $DRAM_ADR = 0Xa0000000            // create a variable , init it with value 0Xa0000000 (Hex)
```

```
ew $DRAM_VAL = 0x5A5AA5A5    // create a variable , init it with value 0x5A5AA5A5 (Hex)
ew $DDCR_ADR = 0xb8001050    // create a variable , init it with value 0xb8001050 (Hex)
ew $DDCR_VAL = (@$CAL_MODE << 0n31) //create a variable DDCR_VAL
                                //value is left-shift 31 bits of CAL_MODE

ew $i = 0n1                    // create a variable , init it with value 1(decimal )
ew $L0 = 0n0                   // create a variable , init it with value 0(decimal )
ew $R0 = 0n33                  // create a variable , init it with value 33(decimal )
ew $L1 = 0n0                   // create a variable , init it with value 0(decimal )
ew $R1 = 0n33                  // create a variable , init it with value 33(decimal )

dv "Begin Calibration\n"      //print "Begin Calibration\n" in console
ew @$DRAM_ADR = @$DRAM_VAL    //create a variable DRAM_ADR, value is
                                // the same with DRAM_VAL

dv "Calibration for 1st DQS\n" // print "Calibration for 1st DQS\n" in console
:loop0_head                   // create a label named loop0_head
    if (@$i > 0n32) { goto loop0_exit } //if value of i greater than 32, goto loop0_exit.
    Ew @$DDCR_ADR = (@$DDCR_VAL & 0xc0000000) | ((@$i - 1) << 0n25)
    // calculate the expression (@$DDCR_VAL & 0xc0000000) | ((@$i - 1) << 0n25), put the value to variable
    // $DDCR_ADR
    if (@$L0 == 0) { goto blk0_t }
    // if variable $L0 equals to 0, goto blk0_t
    goto blk0_f // goto blk0_f
:blk0_t //create label blk0_t
    if (((@$DRAM_ADR & 0x00ff00ff) == (@$DRAM_VAL & 0x00ff00ff)) { goto blk0_t_t }
    // if two expression equals, goto blk0_t_t, @$DRAM_ADR returns data of address $DRAM_ADR .
    goto blk0_t_f // goto label blk0_t_f
:blk0_t_t // create label blk0_t_t
    ew $L0 = @$i // put value of variable $i to $L0
    goto loop0_end // goto label loop0_end
:blk0_t_f // create label blk0_t_f
    goto loop0_end // goto label loop0_end
:blk0_f //create label blk0_f
    if (((@$DRAM_ADR & 0x00ff00ff) != (@$DRAM_VAL & 0x00ff00ff)) { goto blk0_f_t }
    //if two expression equals, goto blk0_f_t, @$DRAM_ADR returns data of address $DRAM_ADR .
    goto blk0_f_f // goto label blk0_f_f
:blk0_f_t // create label blk0_f_t
    ew $R0 = @$i - 1 // put value of variable $i -1 to $R0
    goto loop0_exit // goto label loop0_exit
:blk0_f_f // create label blk0_f_f
    goto loop0_end // / goto label loop0_end

:loop0_end // create label loop0_end
ew $i = @$i + 0n1 // put value of $i + 0n1 to $i
```

```
goto loop0_head // goto label loop0_head

:loop0_exit // create label loop0_exit

ew $DDCR_VAL = (@$DDCR_VAL & 0Xc0000000) | (((@$L0 + @$R0) >> 0n1) << 0n25)
ew @$DDCR_ADR = (@$DDCR_VAL & 0Xc0000000) | (((@$L0 + @$R0) >> 0n1) << 0n25)
// calculate expression value, and put result to variable
dv "L0 = %d, R0 = %d, C0 = %d\n", @$L0, @$R0, (@$L0 + @$R0) >> 0n1
//print variable value to console
dv "DDCR: 0x%08X\n", @$DDCR_ADR
// "@@$DDCR_ADR" reads data from address $DDCR_ADR of target memory, and returns the data.
```

3.5 Run Debugger

1. Run RLX Probe Driver.
2. Run rsdk-elf-gdb/rsdk-elf-insight or sde-insight/sde-gdb/msdk-elf-gdb (for MIPS processors).
3. Connect to RLX Probe with TCP protocol.

For GUI tool insight:

Select "GDBserver/TCP" in target selection menu.

Input the TCP port number you set in `rlx_probe*.cfg`.

For gdb console mode:

target remote (PC IP) : (TCP port)

4. Debug your program.

3.6 CPU Register Alias In RLX Probe

RLX Probe support to access CPU registers with their alias (no matter with Capital or Lower Case) or register number. There are some examples for access the register:

```
ew .v1 0x12345667 (write 0x12345667 to register v1)
ew .$3 0xabcdef00 (write 0xabcdef00 to register v1)
mw .$2 (read from register v0)
```

Here are general registers' alias and their number:

Register Number	Register Alias	Purpose
0	zero	Zero
1	at	For assembly
2,3	v0,v1	Return value of function
4~7	a0~a3	Parameters for function call
8~15	t0~t7	Temporary register(no need to save)
24,25	t8~t9	Temporary register
16~23	s0~s7	Register variables for function
26,27	k0,k1	For interrupt/exception handler
28	gp	Global pointer
29	sp	Stack pointer
30	fp/s8	Frame pointer
31	Ra	Return address of sub-function

Since the registers' number in GDB are different for different processors , We only list the alias of special registers here(you can find them in GDB with “info register” command) , you can find the detail register number in Appendix A.

Register Alias for your reference:

sr, lo , hi ,
 bad, cause ,
 depc (pc) ,
 ESTATUS, ECAUSE, INTVEC,
 cbs0, cbs1, cbs2, cbe0, cbe1, cbe2, lps0, lpe0, lpc0,
 MMD, m0l31to0, m0l39to32, m0h31to0, m0h39to32, m1l31to0, m1l39to32,
 m1h31to0, m1h39to32, m2l31to0, m2l39to32, m2h31to0, m2h39to32, m3l31to0,
 m3l39to32, m3h31to0, m3h39to32,
 index, random , EntryLo, EntryHi,context, wired, BadVaddr,
 epc, PRID, DREG, DeSave,
 WatchLo0, WatchHi, WatchLo1, WatchHi1,
 WatchLo2, WatchHi2, WatchLo3, WatchHi3, WatchLo4, WatchHi4, WatchLo5,
 WatchHi5, WatchLo6, WatchHi6, WatchLo7, WatchHi7,
 CCTLO, CCTLO, CCTLO2,
 Config, config1, config2, config3, config6, config7,
 DataLo,DataHi, dataLo1,dataHi1,

CK0, CK1, CVSTag, BpCtl,
WMPctl, WMPStatus, WMPVaddr, WMPExMSK0, WMPExMSK1, WMPExMSK2,
WMPExMSK3, WMPExMSK4, WMPExMSK5,
WMPExMSK6, WMPExMSK7,
watchlo, watchhi,
MonCtrl0, MonCtrl1, MonDual,
MonCnt0Lo, MonCnt0Hi, MonCnt1Lo, MonCnt1Hi, MonCnt2Lo, MonCnt2Hi,
MonCnt3Lo, MonCnt3Hi,
IWBase0, IWTop0, IWBase1, IWTop1, DWBase0, DWTop0, DWBase1, DWTop1,
L2MemBase, L2MemTop,
entrylo0, entrylo1,
pagemask, hwrena, count, compare, lladdr,
debug, perfcnt0, errctl, cacheerr,
taglo, taghi, errorepc, intctl, ebase,
srsctl,
perfcnt1, perfcnt2, perfcnt3,
taglo1, taghi1,
srsmmap.

3.7 FAQ

1. connection failed

If you set “DBG_MESSAGE_SHOW” to “YES”, there will be error message shown in the RLX Probe Driver window.

Please take a look at the RLX Probe Driver window and follow the hint in the window.

Some error messages explained as following.

Error 0: No USB probe found!

Please check the cable connection status, if failed again, re-plug the cable. If failed again, please check that if USB Device Driver was correctly installed.

Error 1 : No tap detected!

Please make sure all the following requirements met:

a . The signals define of EJTAG 20pin connector on your target

board follow the EJTAG standard:

pin1: TRST, pin2: GND
pin3: TDI, pin4: GND
pin5: TDO, pin6: GND
pin7: TMS, pin8: GND
pin9: TCK, pin10: GND
pin11: RST, pin12: GND
pin13: PCST0, Pin14: GND
pin15: PCST1, Pin14: GND
pin17: PCST2, Pin14: GND
pin19: DCLK, Pin14: GND

Pins define of EJTAG 14pin connector on RLX Probe Board :

pin1: TRST, pin2: GND
pin3: TDI, pin4: GND
pin5: TDO, pin6: GND
pin7: TMS, pin8: GND
pin9: TCK, pin10: GND
pin11: RST, pin12: NC
pin13: PCST0, pin14: VIO

- b. The read line of the EJTAG Cable of RLX Probe should be connected to the pin1 of EJTAG connector on your target board.
- c. The voltage of EJTAG I/O should be 3.3v.
- d. The connection from Probe to target board should be tight and stable.

If all requirements met, please contact supporter to replace the probe.

Error 2: TAP detection failed!

There are too many taps on the JTAG chain or hardware error.

Error 3: Probe Operation Time out!

That means no response from target processor.

Please set RESET_TARGET_PROCESSOR to “yes” and try again.

If failed again, please reset the target board and close the RLX Probe Driver, then restart the RLX Probe Driver.

Error 4: Probe memory read/write check error!

Please re-plug in the USB cable and try again.

If failed again, please contact supporter for a new RLX Probe.

Error 5: USB read (write) count error!

Please re-plug in the USB cable and try again.

If failed again, please check the connection from RLX Probe to target board.

2. Download speed

If you feel too slow to download your program with RLX Probe, please set GDB_MSG_SHOW to “NO” in `rlx_probeX.cfg` and download RSDK of upper version (at least V1.3.3). If you are sure the connection of RLX Probe and your target board is nice, please set TCK_FREQUENCY to 16MHz.

Download speed will be slow under Non-DMA mode of EJTAG2.6 (used by processors: MIPS processors).

Download speed will also be slow when you download from local to remote target.

Download speed will also be slow on Linux system.

Some network firewall will greatly decrease the download speed of RLX Probe.

3. DMA time out!

For flash or some slow peripheral of processor, RLX Probe will time out if access them with DMA mode. You may add a parameter “DMA_DISABLE” to any line of `rlx_probeX.cfg` for your debug to avoid time out.

4. Uninstall RLX Probe Driver

Please run `uninst.exe` from program group or from installation directory to remove RLX Probe Driver (for Windows).

5. How to verify load

Please run “compare-sections” command to verify load. Be careful that remotetimeout should be set to very large to avoid time out when read back for verify. GDB command for this is “set remotetimeout 10000”.

6. Can’t step or run?

If processor go to exception address when you step first time or failed to run, that means exception/interrupt enabled and happened before you download your program. So, please mask all interrupts or re-download your program and try again.

7. Go dead when continue after stopped at a breakpoint

Please update firmware of RLX Probe (see p3.7).

8. New device found when probe plugged in your PC

Please download and install new version of RLX Probe Driver. If probe driver have being installed, please select auto-install when new hardware found.

9. Run failed while Step Ok

The possible reason may be:

1. Cache should be flushed;
2. Delay not enough;
3. Data is modified in interrupt routine.

10. Can’t run forward from a breakpoint?

If you just can’t run forward from a breakpoint hit, you should check if the interrupt of target board happened frequently and try step to see if it’s ok.

11. Run Time debug fail

When you try to connect to target board with probe when it’s running, but you failed to connect , that means CPU is stalled at somewhere and can’t response

to RLX Probe .

If the connection will reset the target board, please try hot plug in the RLX Probe (USB cable has connected the Probe and PC) to target board.

12. Can't access memory address?

Please check if the address is TLB mapped. RLX Probe support auto check the TLB (do not support MIPS processors) or SMMU for the accessible memory address. If the address located in TLB mapped address, but the entry is invalid or dirty or the map not setup, RLX Probe will response error to GDB to avoid dead.

13. Any other problems

Please update your driver and hardware or contact supporter.

Please set GDB_MSG_SHOW and WR_LOG_FILE to “YES” in configure file, and get the log file, then inform technical supporter.

3.8 Debug Multi-Thread Processor with RLX Probe

User should follow the following rules to config RLX Probe:

1. Define the correct processor type in `rlx_probe0.cfg`;
2. MSDK is recommended for debug multi-thread processor, so, add “MSDK_DEBUG” to `rlx_probe0.cfg` because the register number sequence doesn't match that of RSDK or SDE.
3. User should only define a `rlx_probe0.cfg` for each processor not VPE .
4. Only define the first TAP number in `rlx_probe0.cfg` for multi-vpe processor since it looks like there are 2 taps for a multi-VPE processor.

Example: `ACC_JTAG_TAP_NUM = 1;`

3.9 RLX Probe Hardware Update

Please download hardware bitstream (rlx_probe.mcs) from processor website.

Open the shell of RLX Probe with screwdriver. Plug JTAG download cable to the connector J3. The pin define (top to bottom) of J3 is:

1. VCC; 2.GND; 3.TCK; 4.TDO; 5.TDI; 6.TMS.

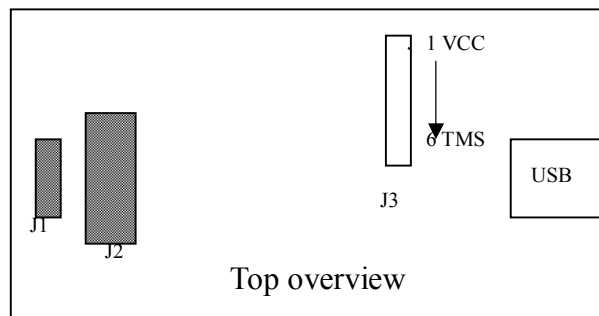


Figure 3.3.9: RLX Probe internal map

Run Xilinx FPGA download program (Impact) and assign the bitstream file to the PROM unit, then program it.

Chapter 4 GDB & Insight Basic Usage

4.1 Some Basic GDB commands

There are some debug commands for you under command line mode.

1. set debug remote 1

Show GDB packet when debug.

Set debug remote 0: don't show GDB packet.

2. target remote

Example 1: target remote 172.29.20.70:55555

Description: to connect IP 172.29.20.70 on port 55555

3. load

Download program to target

4. c

To continue your program.

5. step

Step execute a C sentence.

6. stepi

Step one instruction (assembly) .

7. hb

Set hardware break point to address or line

Example 1: hb 11; set hardware break point to line 11.

Example 2: hb *0x80000000: set a hardware break point to address 0x80000000.

8. p

Read/write register or memory.

Example (read): p *0x80000000 or p /x *0x80000000

Example (write): p *0x80000000=0xffffffff

Register read: p /x \$pc

9. `disas function_name`

Disassemble program.

Example: `disas func`

10. `break`

`break function`

`break +offset`

`break -offset`

`break line number`

`break filename:line number`

`break *address`

`info break` : to inquire the break information

11. `delete breakpoints`

To delete breakpoints.

12. `disable / enable`

Disable or enable breakpoint.

a) `watch/rwatch/awatch` (hardware breakpoint)

Example 1: `watch ABC /watch *0x80000000`

Set a watchpoint for the expression of ABC. GDB will break when the expression is written into and its value changes.

“awatch” means stop when a expression is read or wrote.

Example 2: `rwatch *0x80000000`

Set a read/write watchpoint to the address 0x80000000.

If you need to watch data or instruction with larger size than 4bytes, please use the GDB command “`maint packet Z1/Z2/Z3, address, size`” as following.

Example3: `maint packet Z1, 80000000, 10`

This GDB command should be executed when you want to set a wide range (16bytes width) hardware instruction breakpoint. Processor will break at the instruction located within 0x80000000 to 0x800000010.

Example 4: `maint packet Z2, 80080000, 10`

This GDB command should be executed when you want to set a wide range

(16bytes width) hardware data breakpoint. Processor will break when access the data located within 0x80000000 to 0x80000010.

13. info watchpoint

Print a list of watchpoints, breakpoints, and catchpoints.

b) compare-sections

This command will verify sections loaded to memory.

If no sections followed by the command, all sections will be compared.

14. set remotetimeout 1000000

To avoid timeout when verify load, please set long remotetimeout time (unit: ms).

For more information, please refer to “Debug with GDB.pdf”.

15. x address

Display memory.

16. finish

Run until function return.

17. list

Display source code around the line number you gave out.

18. disassemble function

Dump of assembler code for function.

19. info registers/info all-registers

Display registers.

20. Copy between memory and a file

dump [format] memory filename start-address end-address

dump [format] value filename expr

Dump the contents of memory from start addr to end addr, or the value of expr, to filename in the given format. The format parameter may be any one of:

binary Raw binary form.

lhex Intel hex format.

Tekhex Tektronix Hex format.

Gdb uses the same definitions of these formats as the gnu binary utilities, like

‘objdump’ and ‘objcopy’. If format is omitted, gdb dumps the data in raw binary form.

Example: dump memory /cygdrive/e/dump.bin 0x80000000 0x80100000

append [binary] memory filename start_addr end_addr

append [binary] value filename expr

Append the contents of memory from start addr to end addr, or the value of expr, to filename, in raw binary form. (gdb can only append data to files in raw binary form.)

restore filename [binary] bias start end

Restore the contents of file filename into memory. The restore command can automatically recognize any known bfd file format, except for raw binary. To restore a raw binary file you must specify the optional keyword binary after the filename.

If bias is non-zero, its value will be added to the addresses contained in the file. Binary files always start at address zero, so they will be restored at address bias. Other .bfd files have a built-in location; they will be restored at offset bias from that location.

If start and/or end are non-zero, then only data between file offset start and file offset end will be restored. These offsets are relative to the addresses in the file, before the bias argument is applied.

21. set

Add a variable for GDB console or script use.

Example: set \$var = 0x1234

22. if

Takes a single argument, which is an expression to evaluate. It is followed by a series of commands that are executed only if the expression is true (nonzero). There can then optionally be a line else, followed by a series of commands that are only executed if the expression was false. The end of the list is marked by a line containing end.

23. while

The syntax is similar to if: the command takes a single argument, which is an expression to evaluate, and must be followed by the commands to execute, one per line, terminated by an end. The commands are executed repeatedly as long as the expression evaluates to true.

Loop_break: This command exits the while loop in whose body it is included. Execution of the script continues after that while's end line.

Loop_continue:

This command skips the execution of the rest of the body of commands in the while loop in whose body it is included. Execution branches to the beginning of the while loop, where it evaluates the controlling expression.

End:

Terminate the block of commands that are the body of if, else, or while flow-control commands.

24. define

25. hook

26. printf string, expressions...

Print the values of the expressions under the control of string. The expressions are separated by commas and may be either numbers or pointers. Their values are printed as specified by string, exactly as if your program were to execute the C subroutine printf(string, expressions...);

For example, you can print two values in hex like this:

```
printf "foo, bar-foo = 0x%x, 0x%x\n", foo, bar-foo
```

The only backslash-escape sequences that you can use in the format string are the simple ones that consist of backslash followed by a letter.

27. source filename

Execute the GDB command file filename.

28. maint packet (raw GDB packet mode)

Example1: maint packet Z1, 80000000, 10

This GDB command should be executed when you want to set a wide range (16bytes width) hardware instruction breakpoint. Processor will break at the instruction located within 0x80000000 to 0x800000010.

Example2: maint packet Z2, 80080000, 10

This GDB command should be executed when you want to set a wide range (16bytes width) hardware data breakpoint. Processor will break when access the data located within 0x80000000 to 0x800000010.

4.2 Debug Multi-Thread Processor with GDB

When you need to debug multi-thread processor with GDB and RLX Probe, you may use the following GDB commands and follow some basic rules .

1. info threads

This command will show you how many hardware threads in processor and the current thread number and status of each thread. And you should input the command before you can access memory/register resource of other threads.

Example:

```
(gdb) info threads
```

```
2 Thread 2 (Blocked) thread2_entry () at test.S :483
```

```
*1 Thread 1(Runnable) thread1_entry () at test.S: 472
```

In the example, a “*” will be show at the header of the current thread information line which means the current thread.

2. thread [thread ID]

Switch to other threads for register/memory edit. Do not forget to switch back to the current runnable thread because the switch only enable you to access the registers of the target thread ,not really switch thread context to the target thread. And thread 0 stands for any thread while thread -1 stands for all threads.

```
(gdb) thread 2
```

```
[Switching to thread 2 (Thread 2)]#0 thread2_entry () at test.S:483
```

```
483 nop
```

3. info registers

Read the total registers of the thread.

4. break line/address thread [thread ID]

When you try to set breakpoint at some target thread, please append “thread [thread ID]” to the tail of your breakpoint setting command.

5. hb line/address thread [thread ID]

When you try to set hardware breakpoint at some target thread, but don't stop at other threads, please append “thread [thread ID]” to the tail of your command.

6. watch/rwatch/awatch variable/address thread [thread ID]

When you try to watch some variable or address range at some target thread,

7. set scheduler-locking on|off|step

You may set scheduler-locking on|off|step by send raw packet in GDB command line.

```
Maint packet Qscheduler-locking:on // set the other threads locking
maint packet Qscheduler-locking:off //set the other threads free
maint packet Qscheduler-locking:step //set other threads to single step
```

8. hardware breakpoint enhancement

You may want to set hb/watch point at some target TC or VPE or ASID, now, RLX Probe support it. The format of raw packet should be:

```
maint packet Z1,80000000,4,[TC number]:[ASID];[VPE number] (for
instruction hardware breakpoint)
```

```
maint packet Z2,80000000,4,[TC number]:[ASID];[VPE number] (for data
watch breakpoint)
```

```
maint packet Z3,80000000,4,[TC number]:[ASID];[VPE number] (for data
read watch breakpoint)
```

9. thread apply [thread ID][thread ID][...] [command]

You may apply some GDB commands to one or more threads with only one command. (Not supported now)

4.3 GDB script file

GDB can execute script file you edit with GDB commands. The “source filename” command is for this usage.

An GDB script file can work with your program in memory. The following GDB script file example will store data from a file to memory and GDB script work with the program “rlx_ep_fw_spi_flash.exe” (which will be in the install directory).

Example:

```
#set height to 0 to avoid “return” press needed for scroll
set height 0

#set file name
file rlx_ep_fw_spi_flash.exe

#connection
target remote : 5181
```

```
#load your file to memory
load

#set breakpoint to symbol in your file
b    debug_spi_fast_wr_end

#set the memory buffer size
set $size = 31*1024

#set loop time
set $loop_num = 27

#set buffer size
set $buf_size = 0x7c00

#set start address
set $start_addr = 0

#set end address of data load
set $end_addr = $buf_size - 1

#set registers to be used in your file
p /x $t4 = $pc + 1024

#set registers to be used in your file
p /x $gp = $size

#set registers to be used in your file
p /x $a0 = 0x80009000


#execute loop
while $loop_num > 0
    #read data to memory from a file
    restore E:\proj\EJTAG\IP\FPGA\rlx_ep.ngd binary 0x80000400
        $start_addr $end_addr

    #continue

c

p /x $gp = $size
```

```
p /x $a0 = $a0 + $buf_size
set $start_addr = $start_addr + $buf_size
set $end_addr = $start_addr + $buf_size - 1
#decrease the counter
set $loop_num = $loop_num - 1
p $loop_num
p /x *$start_addr
end
#printf your message
printf "spi flash success !"
#quit GDB
quit
#confirm for GDB input
y
```

4.4 GDB Remote File I/O

4.4.1 Introduction

RLX Probe support at least half-duplex data exchange under the GDB Remote Serial Protocol (RSP).

The GDB remote I/O library uses a predefined address, default 0x80000090, for passing parameters. This address can be configurable with function "rlx_gdb_set_param_addr". Please make sure the application does not use the memory within 1K range starting at the address you defined.

The GDB remote I/O library is a set of subroutines that bases on the RSP protocol to emulate I/O interactions during remote debugging. It allows the target to use the host's file system and console I/O to perform various system calls.

This simulates file system operations even on targets that lack file systems. If there's any error finishing the I/O call, the library tries to set the global variable errno. In order to link with the library, add "-lrlx" to your link flags. Be careful that, you would better use library from RLX Probe Driver installation destination directory and set the

correct link path in makefile.

The parameter “FILE_IO_BASE” defines the file IO base address for remote file IO. If you want to use address other than default address 0x80000090, please set it to your favorite address and keep it at least 1K memory size available start from that address for RLX Probe use.

4.4.2 Interface

The APIs for the GDB Remote I/O library are summarized as follows:

- **rlx_gdb_open** – remote file open
- **rlx_gdb_close** – remote file close
- **rlx_gdb_read** – remote file read
- **rlx_gdb_write** – remote file write
- **rlx_gdb_lseek** – remote file lseek
- **rlx_gdb_rename** – remote file rename
- **rlx_gdb_unlink** – remote file unlink
- **rlx_gdb_stat/fstat** – remote file stat/fstat
- **rlx_gdb_gettimeofday** – remote gettimeofday
- **rlx_gdb_isatty** – remote isatty
- **rlx_gdb_system** – remote system
- **rlx_gdb_printf** – remote printf
- **rlx_gdb_fprintf** – remote fprintf

More information of file IO, please refer to library.pdf in installation directory.

4.5 Insight Introduction

Please refer to Insight_Guide.doc in the installation directory.

Chapter 5 Operating Environment

5.1 Electrical Characters

Power: PC USB port (< 300mA)

Voltage: 5V (Supply voltage of USB port should not fall down under 3.9V)

Target processor I/O supply voltage:

RLX Probe 1.0: 3.3V/2.5V

Make sure your probe's I/O voltage not exceed 2.6V before connect to Xilinx V6 FPGA, or else probe may damage FPGA.

RLX Probe 2.0: 3.3V, 2.5V, 1.8V, 1.5V, 1.2V and changeable.

5.2 Operating Temperature

Working Temperature range	0 ~ 70 °C
Store Temperature range	-40 ~ 80 °C
Opposite Humidity	10 ~ 90 %

Table 5.1: RLX Probe operating temperature

Chapter 6 Virtual Probe Usage

Virtual Probe is a co-simulation tools based on JTAG XTOR or APB XTOR without hardware probe. Now Virtual Probe supports Zebu JTAG XTOR, PXP APB XTOR, and RTK JTAG/APB XTOR.

6.1 ZeBu JTAG XTOR

The ZeBu JTAG XTOR developed by Synopsis implements an interface to drive a JTAG port of a design implemented in ZeBu. This JTAG interface is managed by a software layer controlling all signals (TCK, TMS, TDI and TDO).

The Co-simulation overview:

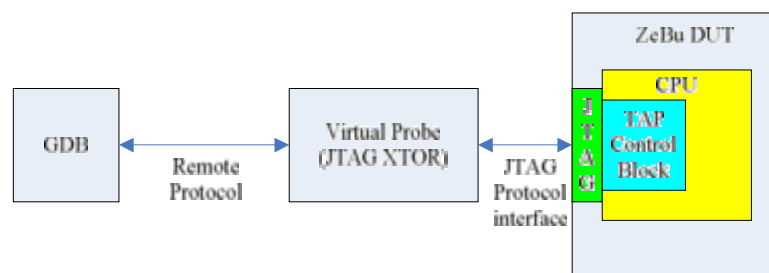


Figure 6.1: Debug overview with Virtual Probe for JTAG XTOR

Virtual Probe for ZeBu JTAG XTOR supports all debugging functions: single step, register access, memory access, hardware breakpoint, software breakpoint, and watchpoint.

6.2 PXP APB XTOR

Palladium XP APB XTOR developed by Candence is based on ARM AMBA APB XTOR. Virtual Probe for PXP APB XTOR communicates with CPU by AMBA APB Interface without JTAG signals.

The Co-simulation overview:

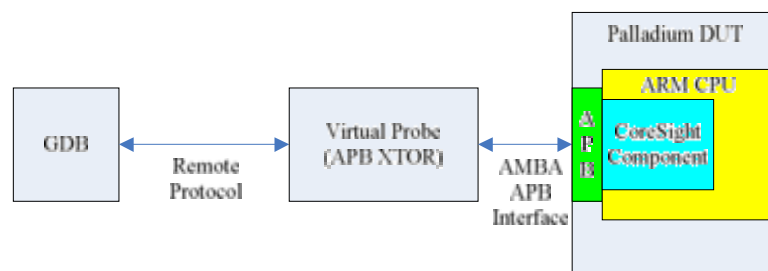


Figure 6.2: Debug overview with Virtual Probe for APB XTOR

Virtual Probe for ZeBu JTAG XTOR supports all debugging functions: single step, register access, memory access, hardware breakpoint, software breakpoint, and watchpoint.

6.3 RTK APB/JTAG XTOR

The RTK APB/JTAG XTOR is an in-house co-simulation environment for designer. Virtual Probe for RTK JTAG XTOR support JTAG protocol. Otherwise Virtual Probe for RTK APB XTOR supports AMBA APB XTOR. Both of those supports all debugging functions: single step, register access, memory access, hardware breakpoint, software breakpoint, and watchpoint.

The overviews of Virtual Probe are the same as PXP/ZeBu XTOR.

Chapter 7 Support

Tel: +86-512-62588966-6134 (internal call from Realtek, please dial 80-50-6134)

Fax: +86-512-62588160

Email : yf_chen@realtek.com.cn (Realsil: 6134)

Email: dong0127_fang@realtek.com.cn (Realsil: 6704)

Email: cwchen02@realtek.com.tw (Realtek: 5548)

Email: nsli@realtek.com.tw (Realtek: 5733)

Web Site: <http://processor.realtek.com.tw>

Thanks for using RLX Probe!

Appendix A

Register Number Map (For MIPS)

Reg number										Lexra (Taroko)		Reg number
in G packet		4181	4180	5181/5280	Temp for 5281	MIPS (MSDK)						in G packet
0-31	General Register	General Register	General Register	General Register	General Register	General Register				General		0-31
32	STATUS CP0_select_0 12\$	STATUS CP0_select_0 12\$	STATUS CP0_select_0 12\$	STATUS CP0_select_0 12\$	STATUS CP0_select_0 12\$	STATUS CP0_select_0 12\$				Status		32
33	LO CP0_select_0 18\$	LO CP0_select_0 18\$	LO CP0_select_0 18\$	LO CP0_select_0 18\$	LO CP0_select_0 18\$	LO CP0_select_0 18\$				LO		33
34	Hi CP0_select_0 19\$	Hi CP0_select_0 19\$	Hi CP0_select_0 19\$	Hi CP0_select_0 19\$	Hi CP0_select_0 19\$	Hi CP0_select_0 19\$				HI		34
35	BADVADDR CP0_select_0 8\$	BADVADDR CP0_select_0 8\$	BADVADDR CP0_select_0 8\$	BADVADDR CP0_select_0 8\$	BADVADDR CP0_select_0 8\$	BADVADDR CP0_select_0 8\$				BadVaddr		35
36	CAUSE CP0_select_0 13\$	CAUSE CP0_select_0 13\$	CAUSE CP0_select_0 13\$	CAUSE CP0_select_0 13\$	CAUSE CP0_select_0 13\$	CAUSE CP0_select_0 13\$				Cause		36
37	DEPC CP0_select_0 17\$	DEPC CP0_select_0 17\$	DEPC CP0_select_0 17\$	DEPC CP0_select_0 17\$	DEPC CP0_select_0 17\$	DEPC CP0_select_0 17\$				PC(DEPC)		37
38-69	CP0 Select 0 Register	float point registers	float point registers	CP0 Select 0 Register	Reserved					FloatingPoint(Reserved)		38-69
70-72	Reserved	Reserved	Reserved	Reserved	Reserved					FloatingPoint(Reserved)		70-72
73	LXCP0_ESTATUS		LXCP0_ESTATUS	LXCP0_ESTATUS	Reserved					LXCP0_ESTATUS		73
74	LXCP0_ECAUSE		LXCP0_ECAUSE	LXCP0_ECAUSE	CP0 select 0, \$0					LXCP0_ECAUSE		74
75	LXCP0_INTVEC		LXCP0_INTVEC	LXCP0_INTVEC	CP0 select 0, \$1					LXCP0_INTVEC		75
76			cbs0	LXCP0_BPCTL	CP0 select 0, \$2					cbs0		76
77			cbs1	LXCP0_WMPCTL	CP0 select 0, \$3					cbs1		77
78			cbs2	LXCP0_WMPSTATUS	CP0 select 0, \$4					cbs2		78

79		cbe0	LXCP0_WMPVADDR	CP0 select 0, \$5	cbe0	79
80		cbe1	LXCP0_WMPEXTRAMASK	CP0 select 0, \$6	cbe1	80
81		cbe2	Reserved	CP0 select 0, \$7	cbe2	81
82		lps0	Reserved	CP0 select 0, \$8	lps0	82
83	0x53	lpe0	CP0 select 1, \$18	CP0 select 0, \$9	lpe0	83
84		lpc0	CP0 select 1, \$19	CP0 select 0, \$10	lpc0	84
85		mmd	CP0 select 1, \$20	CP0 select 0, \$11	mmd	85
86		m0l	CP0 select 2, \$18	CP0 select 0, \$12	m0l	86
87		m0l high 8bit	CP0 select 2, \$19	CP0 select 0, \$13	m0l	87
88		m0h	CP0 select 3, \$18	CP0 select 0, \$14	m0h	88
89		m0h high 8bit	CP0 select 3, \$19	CP0 select 0, \$15	m0h	89
90		m1l	CP0 select 4, \$18	CP0 select 0, \$16	m1l	90
91		m1l high 8bit	CP0 select 4, \$19	CP0 select 0, \$17	m1l high 8bit	91
92		m1h	CP0 select 5, \$18	CP0 select 0, \$18	m1h	92
93		m1h high 8bit	CP0 select 5, \$19	CP0 select 0, \$19	m1h high 8bit	93
94		m2l	CP0 select 6, \$18	CP0 select 0, \$20	m2l	94
95		m2l high 8bit	CP0 select 6, \$19	CP0 select 0, \$21	m2l high 8bit	95
96		m2h	CP0 select 7, \$18	CP0 select 0, \$22	m2h	96
97		m2h high 8bit	CP0 select 7, \$19	CP0 select 0, \$23	m2h high 8bit	97
98		m3l		CP0 select 0, \$24	m3l	98
99		m3l high 8bit		CP0 select 0, \$25	m3l high 8bit	99
100	0x64	m3h		CP0 select 0, \$26	m3h	100

101	m3h high 8bit	CP0 select 0, \$27	m3h high 8bit	101
102		CP0 select 0, \$28	CP0 select0, \$0	102
103		CP0 select 0, \$29	cp0 seletc0 \$1	103
104		CP0 select 0, \$30	cp0 seletc0 \$2	104
105		CP0 select 0, \$31	cp0 seletc0 \$3	105
106		CP0 select 1, \$0	cp0 seletc0 \$4	106
107		CP0 select 1, \$1	cp0 seletc0 \$5	107
108		CP0 select 1, \$2	cp0 seletc0 \$6	108
109		CP0 select 1, \$3	cp0 seletc0 \$7	109
110		CP0 select 1, \$4	cp0 seletc0 \$8	110
111		CP0 select 1, \$5	cp0 seletc0 \$9	111
112		CP0 select 1, \$6	cp0 seletc0 \$10	112
113		CP0 select 1, \$7	cp0 seletc0 \$11	113
114		CP0 select 1, \$8	cp0 seletc0 \$12	114
115	0x73	CP0 select 1, \$9	cp0 seletc0 \$13	115
116		CP0 select 1, \$10	cp0 seletc0 \$14	
117		CP0 select 1, \$11	cp0 seletc0 \$15	
118		CP0 select 1, \$12	cp0 seletc0 \$16	
119		CP0 select 1, \$13	cp0 seletc0 \$17	
120		CP0 select 1, \$14	cp0 seletc0 \$18	
121		CP0 select 1, \$15	cp0 seletc0 \$19	
122		CP0 select 1, \$16	cp0 seletc0 \$20	

123	CP0 select 1, \$17	cp0 select0 \$21	
124	CP0 select 1, \$18	cp0 select0 \$22	
125	CP0 select 1, \$19	cp0 select0 \$23	
126	CP0 select 1, \$20	cp0 select0 \$24	
127	CP0 select 1, \$21	cp0 select0 \$25	
128	CP0 select 1, \$22	cp0 select0 \$26	
129	CP0 select 1, \$23	cp0 select0 \$27	
130	CP0 select 1, \$24	cp0 select0 \$28	
131	CP0 select 1, \$25	cp0 select0 \$29	
132	CP0 select 1, \$26	cp0 select0 \$30	
133	CP0 select 1, \$27	cp0 select0, \$31	133
134	CP0 select 1, \$28	CP0 select 1, \$18	134
135	CP0 select 1, \$29	CP0 select 1, \$19	135
136	CP0 select 1, \$30	CP0 select 1, \$20	136
137	CP0 select 1, \$31	CP0 select 2, \$18	137
138	CP0 select 2, \$0	CP0 select 2, \$19	138
139	CP0 select 2, \$1	CP0 select 2, \$20	139
140	CP0 select 2, \$2	CP0 select 3, \$18	140
141	CP0 select 2, \$3	CP0 select 3, \$19	141
142	CP0 select 2, \$4	reserved	142
143	CP0 select 2, \$5	CP0 select 4, \$18	143
144	CP0 select 2, \$6	CP0 select 4, \$19	144

145		CP0 select 2, \$7	reserved	145
146		CP0 select 2, \$8	CP0 select 5, \$18	146
147	0x93	CP0 select 2, \$9	CP0 select 5, \$19	147
148		CP0 select 2, \$10	reserved	148
149		CP0 select 2, \$11	CP0 select 6, \$18	149
150		CP0 select 2, \$12	CP0 select 6, \$19	150
151		CP0 select 2, \$13	reserved	151
152		CP0 select 2, \$14	CP0 select 7, \$18	152
153		CP0 select 2, \$15	CP0 select 7, \$19	153
154		CP0 select 2, \$16	reserved	154
155		CP0 select 2, \$17	LX CP0 select 0, \$0	155
156		CP0 select 2, \$18	LX CP0 select 0, \$1	156
157		CP0 select 2, \$19	LX CP0 select 0, \$2	
158		CP0 select 2, \$20	LX CP0 select 0, \$3	
159		CP0 select 2, \$21	LX CP0 select 0, \$4	
160		CP0 select 2, \$22	LX CP0 select 0, \$5	
161		CP0 select 2, \$23	LX CP0 select 0, \$6	
162		CP0 select 2, \$24	LX CP0 select 0, \$7	
163		CP0 select 2, \$25	LX CP0 select 0, \$8	
164		CP0 select 2, \$26	LX CP0 select 0, \$9	
165		CP0 select 2, \$27	LX CP0 select 0, \$10	
166		CP0 select 2, \$28	LX CP0 select 0, \$11	

167		CP0 select 2, \$29	LX CP0 select 0, \$12	
168		CP0 select 2, \$30	LX CP0 select 0, \$13	
169		CP0 select 2, \$31	LX CP0 select 0, \$14	
170		CP0 select 3, \$0	LX CP0 select 0, \$15	
171		CP0 select 3, \$1	LX CP0 select 0, \$16	
172		CP0 select 3, \$2	LX CP0 select 0, \$17	
173		CP0 select 3, \$3	LX CP0 select 0, \$18	
174		CP0 select 3, \$4	LX CP0 select 0, \$19	
175		CP0 select 3, \$5	LX CP0 select 0, \$20	
176		CP0 select 3, \$6	LX CP0 select 0, \$21	
177		CP0 select 3, \$7	LX CP0 select 0, \$22	
178		CP0 select 3, \$8	LX CP0 select 0, \$23	
179	0xb3	CP0 select 3, \$9	LX CP0 select 0, \$24	
180		CP0 select 3, \$10	LX CP0 select 0, \$25	
181		CP0 select 3, \$11	LX CP0 select 0, \$26	
182		cp0 ,select 3,\$12	LX CP0 select 0, \$27	
183		CP0 select 3, \$13	LX CP0 select 0, \$28	
184		CP0 select 3, \$14	LX CP0 select 0, \$29	
185		CP0 select 3, \$15	LX CP0 select 0, \$30	
186		cp0 ,select 3,\$16	LX CP0 select 0, \$31	186
187			LX CP0 select 1, \$19	187
188		cp0 ,select 3,\$18	LX CP0 select 2, \$19	188

189	cp0 ,select 3,\$19	LX CP0 select 3, \$19	189
190		LX CP0 select 4, \$19	190
191		LX CP0 select 5, \$19	191
192		LX CP0 select 6, \$19	192
193	cp0 ,select 3,\$23	LX CP0 select 7, \$19	193
194		LX CP0 select 1, \$10	194
195	cp0 ,select 3,\$25		
196			
197			
198	cp0 ,select 3,\$28		
199			
200			
201	cp0 ,select 3,\$31		
202	cp0 ,select 4,\$0		
203			
204			
205			
206			
207			
208			
209			
210			

211	0xd3		
212			
213			
214			
215			
216			
217			
218			
219			
220		cp0,s 4,\$18	
221		cp0,s 4,\$19	
222			
223			
224			
225		cp0,s 4,\$23	CP3 control, \$31 225
226			CP3 general, \$0 226
227			
228			
229			
230		cp0,s 4,\$28	
231			
232			

233		0	
234			
235			
236			
237			
238			
239			
240			
241			
242			
243			
244			
245			
246			
247			
248			
249			
250			
251			
252		cp0,s 5,\$18	
253		cp0,s 5,\$19	
254			

255			
256			
257	cp0,s 5,\$23	CP3 general, \$31	257
258		END	258
259			
260			
261			
262	cp0,s 5,\$28		
263	cp0,s 5,\$29		
264			
265	0		
266			
267			
268			
269			
270			
271			
272			
273			
274			
275			
276			
277			

278

279

280

281

282

cp0 ,s6,\$16

283

284

cp0 ,s6,\$18

285

cp0 ,s6,\$19

286

287

288

289

290

291

292

293

294

295

296

297

0

298

299

300

301	
302	
303	
304	
305	
306	
307	
308	
309	
310	
311	
312	
313	
314	cp0,s7,\$16
315	
316	cp0,s7,\$18
317	cp0,s7,\$19
318	
319	
320	
321	
322	
323	

324

325

326

327

328

329

cp0,s7,\$31 (END)

Register Number Map (For ARM)

Reg numer in G packet	ARMv7.x	ARMv8 aarch32	ARMv8 aarch64	Cortex-M3
0	R0	R0	X0	r0
1	R1	R1	X1	r1
2	R2	R2	X2	r2
3	R3	R3	X3	r3
4	R4	R4	X4	r4
5	R5	R5	X5	r5
6	R6	R6	X6	R6
7	R7	R7	X7	r7
8	R8	R8	X8	r8
9	R9	R9	X9	r9
10	R10	R10	X10	r10
11	R11	R11	X11	r11
12	R12	R12	X12	r12

13	SP	SP	X13	sp
14	LR	LR	X14	lr
15	PC	PC	X15	pc
16	-	-	X16	
17	-	-	X17	
18	-	-	X18	
19	-	-	X19	
20	-	-	X20	
21	-	-	X21	
22	-	-	X22	
23	-	-	X23	
24	-	-	X24	
25	CPSR	CPSR	X25	xpcr
26	SPSR_IRQ	R8_USR	X26	MSP
27	SP_IRQ	R9_USR	X27	PSP
28	LR_IRQ	R10_USR	X28	PRIMASK
29	SPSR_FIQ	R11_USR	X29	BASEPRI
30	R8_FIQ	R12_USR	X30	FAULTMASK
31	R9_FIQ	SP_USR	SP	CONTROL
32	R10_FIQ	LR_USR	PC	
33	R11_FIQ	R8_FIQ	CPSR	
34	R12_FIQ	R9_FIQ	FPSR	

35	SP_FIQ	R10_FIQ	FPCR	
36	LR_FIQ	R11_FIQ	CURRENT_EL	
37	SPSP_UND	R12_FIQ	DAIF	
38	SP_UND	SP_FIQ	DLR_EL0	
39	LR_UND	LR_FIQ	DSPSR_EL0	
40	SPSR_ABT	SP_IRQ	ELR_EL1	
41	SP_ABT	LR_IRQ	ELR_EL2	
42	LR_ABT	SP_SVC	ELR_EL3	
43	SPSR_SVC	LR_SVC	NZCV	
44	SP_SVC	SP_ABT	SP_EL0	
45	LR_SVC	LR_ABT	SP_EL1	
46	R8_USR	SP_UND	SP_EL2	
47	R9_USR	LR_UND	SP_EL3	
48	R10_USR	SP_MON	SPSEL	
49	R11_USR	LR_MON	SPSR_EL1	
50	R12_USR	SP_HYP	SPSR_EL2	
51	SP_USR	ELR_HYP	SPSR_EL3	
52	LR_USR	SPSR_FIQ	SPSR_ABT	
53	SPSR_MON	SPSR_IRQ	SPSR_FIQ	
54	SP_MON	SPSR_SVC	SPSR_IRQ	
55	LR_MON	SPSR_ABT	SPSR_UND	
56	SPSR_HYP	SPSR_UND		

57	SP_HYP	SPSR_MON		
58	LR_HYP	SPSR_HYP		
59				
60	MIDR		ACTLR_EL1	
61	CTR	ACTLR	ACTLR_EL2	
62	TCMTR	ADFSR	ACTLR_EL3	
63	TLBTR	AIDR	AFSR0_EL1	
64	MPUIR	AIFSR	AFSR0_EL2	
65	MPIDR	AMAIR0	AFSR0_EL3	
66	REVIDR	AMAIR1	AFSR1_EL1	
67	AMIDR	ATS12NSOPR	AFSR1_EL2	
68	ID_PFR0	ATS12NSOPW	AFSR1_EL2	
69	ID_PFR1	ATS12NSOUR	AIDR_EL1	
70	ID_DFR0	ATS12NSOUW	AMAIR_EL1	
71	ID_AFR0	ATS1CPR	AMAIR_EL2	
72	ID_MMFR0	ATS1CPW	AMAIR_EL3	
73	ID_MMFR1	ATS1CUR	CSSIDR_EL1	
74	ID_MMFR2	ATS1CUW	CLIDR_EL1	
75	ID_MMFR3	ATS1HR	CONTEXTIDR_EL1	
76	ID_ISAR0	ATS1HW	CPACR_EL1	
77	ID_ISAR1	BPIALL	CPTR_EL2	
78	ID_ISAR2	BPIALLIS	CPTR_EL3	

79	ID_ISAR3	BPIMVA	CSSELR_EL1	
80	ID_ISAR4	CSSIDR	CTR_EL0	
81	ID_ISAR5	CLIDR	DACR32_EL2	
82	CSSIDR	CONTEXTIDR	DCZID_EL0	
83	CLIDR	CP15DMB	ESR_EL1	
84	AIDR	CP15DSB	ESR_EL2	
85	CSSELR	CP15ISB	ESR_EL3	
86	VPIDR	CPACR	FAR_EL1	
87	VMPIDR	CSSELR	FAR_EL2	
88	SCTLR	CTR	FAR_EL3	
89	ACTLR	DACR	FPEXC32_EL2	
90	CPACR	DCCIMVAC	HACR_EL2	
91	SCR	DCCISW	HCR_EL2	
92	SDER	DCCMVAC	HPFAR_EL2	
93	NSACR	DCCMVAU	HSTR_EL2	
94	HSCCLR	DCCSW	ID_AA64AFR0_EL1	
95	HACTLR	DCIMVAC	ID_AA64AFR1_EL1	
96	HCR	DCISW	ID_AA64DFR0_EL1	
97	HDCLR	DFAR	ID_AA64DFR1_EL1	
98	HCPTR	DFSR	ID_AA64ISAR0_EL1	
99	HSTR	DTLBIALL	ID_AA64ISAR1_EL1	
100	HACR	DTLBIASID	ID_AA64MMFR0_EL1	

101	TTBR0	DTLBIMVA	ID_AA64MMFR1_EL1	
102	TTBR1	FCSEIDR	ID_AA64PFR0_EL1	
103	TTBCR	HACR	ID_AA64PFR1_EL1	
104	HTCR	HACTLR	ID_AFR0_EL1	
105	VTCR	HADFSR	ID_DFR0_EL1	
106	DACR	HAIFSR	ID_ISAR0_EL1	
107	TTBR0_64	HAMAIR0	ID_ISAR1_EL1	
108	TTBR1_64	HAMAIR1	ID_ISAR2_EL1	
109	HTTBR	HCPtr	ID_ISAR3_EL1	
110	VTTBR	HCR	ID_ISAR4_EL1	
111	DFSR	HCR2	ID_ISAR5_EL1	
112	IFSR	HDFAR	ID_MMFR0_EL1	
113	ADFSR	HIFAR	ID_MMFR1_EL1	
114	AIFSR	HMAIR0	ID_MMFR2_EL1	
115	HADFSR	HMAIR1	ID_MMFR3_EL1	
116	HAIFSR	HPFAR	ID_PFR0_EL1	
117	HSR	HRMR	ID_PFR1_EL1	
118	DFAR	HSCTLR	IFSR32_EL1	
119	IFAR	HSR	ISR_EL1	
120	HDFAR	HSTR	MAIR_EL1	
121	HIFAR	HCTR	MAIR_EL2	
122	HPFAR	HTPIDR	MAIR_EL3	

123	WFI	HVBAR	MIDR_EL1	
124	ICIALLUIS	VBAR	MPIDR_EL1	
125	BPIALLIS	ICIALLU	MVFR0_EL1	
126	PAR	ICIALLUIS	MVFR1_EL1	
127	ICIALLU	ICIMVAU	MVFR2_EL1	
128	ICIMVAU	ID_AFR0	PAR_EL1	
129	ISB	ID_DFR0	REVIDR_EL1	
130	BPIALL	ID_ISAR0	RMR_EL1	
131	BPIMVA	ID_ISAR1	RMR_EL2	
132	DCIMVAC	ID_ISAR2	RMR_EL3	
133	DCISW	ID_ISAR3	RVBAR_EL1	
134	ATS1CPR	ID_ISAR4	RVBAR_EL2	
135	ATS1CPW	ID_ISAR5	RVBAR_EL3	
136	ATS1CUR	ID_MMFR0	SCR_EL3	
137	ATS1CUW	ID_MMFR1	SCTLR_EL1	
138	ATS12NSOPR	ID_MMFR2	SCTLR_EL2	
139	ATS12NSOPW	ID_MMFR3	SCTLR_EL3	
140	ATS12NSOUR	ID_PFR0	TCR_EL1	
141	ATS12NSOUW	ID_PFR1	TCR_EL2	
142	DCCMVAC	IFAR	TCR_EL3	
143	DCCSW	IFSR	TPIDR_EL0	
144	DSB	ISR	TPIDR_EL1	

145	DMB	ITLBIALL	TPIDR_EL2	
146	DCCMVAU	ITLBIASID	TPIDR_EL3	
147	DCCIMVAC	ITLBIMVA	TPIDRRO_EL0	
148	DCCISW	JIDR	TTBR0_EL1	
149	ATS1HR	JMCR	TTBR0_EL2	
150	ATS1HW	JOSCR	TTBR0_EL3	
151	PAR_64	MAIR0	TTBR1_EL1	
152	TLBIALLIS	MAIR1	VBAR_EL1	
153	TLBIMVAIS	MIDR	VBAR_EL2	
154	TLBIASIDIS	MPIDR	VBAR_EL3	
155	TLBIMVAAIS	MVBAR	VMPIDR_EL2	
156	ITLBIALL	NMRR	VPIDR_EL2	
157	ITLBIMVA	NSACR	VTCR_EL2	
158	ITLBIASID	PAR	VTTBR_EL2	
159	DTLBIALL	PRRR	MDCCINT_EL1	
160	DTLBIMVA	REVIDR	MDCCSR_EL0	
161	DTLBIASID	RMR	MDCR_EL2	
162	TLBIMVA	RVBAR	MDRAR_EL1	
163	TLBIASID	SCR	MDSCR_EL1	
164	TLBIMVA	SCTLR	OSDLR_EL1	
165	TLBIMVAA	TCMTR	OSDTRRX_EL1	
166	TLBIALLHIS	TLBIALL	OSDTRTX_EL1	

167	TLBIMVAHIS	TLBIALLH	OSECCR_EL1	
168	TLBIALLNSNHIS	TLBIALLHIS	OSLAR_EL1	
169	TLBIALLH	TLBIALLIS	OSLSR_EL1	
170	TLBIMVAH	TLBIALLNSNH	SDER32_EL3	
171	TLBIALLNSNH	TLBIALLNSNHIS		
172	PMCR	TLBIASID		
173	PMCNTENSET	TLBIASIDIS		
174	PMCNTENCLR	TLBIIPAS2		
175	PMOVS	TLBIIPAS2IS		
176	PMSWINC	TLBIIPAS2L		
177	PMSELR	TLBIIPAS2LIS		
178	PMCEID0	TLBIMVA		
179	PMCEID1	TLBIMVAA		
180	PMCCNTR	TLBIMVAAIS		
181	PMXEVTYPER	TLBIMVAAL		
182	PMXVCNTR	TLBIMVAALIS		
183	PMUSERENR	TLBIMVAH		
184	PMINTENSET	TLBIMVAHIS		
185	PMINTENCLR	TLBIMVAIS		
186	PMOVSET	TLBIMVAL		
187	PRRR	TLBIMVALH		
188	NMRR	TLBIMVALHIS		

189	AMAIRO	TLBIMVALIS		
190	AMAIR1	TLBTR		
191	HMAIRO	TPIDRPRW		
192	HMAIR1	TPIDRPURO		
193	HAMAIRO	TPIDRURW		
194	HMAIR1	TTBCR		
195	VBAR	TTBR0		
196	MVBAR	TTBR1		
197	ISR	HTTBR		
198	HVBAR	VMPIDR		
199	FCSEIDR	VPIDR		
200	CONTEXTIDR	VTCR		
201	TPIDRURW	VTTBR		
202	TPIDRURO	DLR		
203	TPIDRPRW	DSPSR		
204	HTPIDR	HDCR		
205	CNTRFRQ	SDCR		
206	CNTKCTL	SDER		
207	CNTP_TVAL	PMCCFILTR		
208	CNTP_CTL	PMCCNTR		
209	CNTV_TVAL	PMCEID0		
210	CNTV_CTL	PMCEID1		

211	CNTHCTL	PMCNTENCLR		
212	CNTHP_TVAL	PMCNTENSET		
213	CNTHP_CTL	PMCR		
214	CNTPCT	PMINTENCLR		
215	CNTVCT	PMINTENSET		
216	CNTP_CVAL	PMOVSr		
217	CNTV_CVAL	PMOVSSET		
218	CNTVOFF	PMSELR		
219	CNTHP_CVAL	PMSWINC		
220	DRBAR	PMUSERENR		
221	IRBAR	PMXEVCNTR		
222	DRSR	PMCEVTYPER		
223	IRSR	CNTRFQ		
224	DRACR	CNTHCTL		
225	IRACR	CNTHP_CTL		
226	RGNR	CNTHP_CVAL		
227	L2CTLR	CNTHP_TVAL		
228	L2ECTLR	CNTKCTL		
229	MAIR0	CNTP_CTL		
230	MAIR1	CNTP_CVAL		
231	CDBGDR0	CNTP_TVAL		
232	CDBGDR1	CNTPCT		

233	CDBGDR2	CNTV_CTL		
234	CDBGDCT	CNTV_CVAL		
235	CDBGICT	CNTV_TVAL		
236	CDBGDCD	CNTVCT		
237	CDBGICD	CNTVOFF		
238	CDBGTD			
239	CBAR			
240	TLBLR			
241	LTLBAR			
242	LTLBVAR			
243	LTLBPAR			
244	SLTLBER			
245	SLTLBEW			
246	V2PCWPR			
247	V2PCWPW			
248	V2PCWUR			
249	V2PCWUW			
250	V2POWPR			
251	V2POWPW			
252	V2POWUR			
253	V2POWUW			
254	PLEIDR			

255	PLEASR			
256	PLEFSR			
257	PLEUAR			
258	PLEPCR			
259	TLBCR			
260	PCR			
261	NEONBR			
262	VCR			
263	VIR			
264	SCUCTLR			
265	L2MRERRSR			
266	DCCIALl			
267	FILASTARTR			
268	FILAENDR			
269-426				
427	S0			
428	S1			
429	S2			
430	S3			
431	S4			
432	S5			
433	S6			

434	S7			
435	S8			
436	S9			
437	S10			
438	S11			
439	S12			
440	S13			
441	S14			
442	S15			
443	S16			
444	S17			
445	S18			
446	S19			
447	S20			
448	S21			
449	S22			
450	S23			
451	S24			
452	S25			
453	S26			
454	S27			
455	S28			

456	S29			
457	S30			
458	S31			
459	D0			
460	D1			
461	D2			
462	D3			
463	D4			
464	D5			
465	D6			
466	D7			
467	D8			
468	D9			
469	D10			
470	D11			
471	D12			
472	D13			
473	D14			
474	D15			
475	D16			
476	D17			
477	D18			

478	D19			
479	D20			
480	D21			
481	D22			
482	D23			
483	D24			
484	D25			
485	D26			
486	D27			
487	D28			
488	D29			
489	D30			
490	D31			
491	FPEXC			
492	FPSCR			
493	FPSID			
494	MVFR0			
495	MVFR1			
496	Q0			
497	Q1			
498	Q2			
499	Q3			

500	Q4			
501	Q5			
502	Q6			
503	Q7			
504	Q8			
505	Q9			
506	Q10			
507	Q11			
508	Q12			
509	Q13			
510	Q14			
511	Q15			

Appendix B

Revision History

Table A.1: Revision History

Revision	Date	Description
Beta 1.0	Nov 29 th , 2007	Beta 1.0 version
V1.0	Dec 17 th , 2007	Version 1.0
V1.1	Jan 7 th , 2008	Add initial memory block write function.
V1.2	Mar 18 th , 2008	Add introduction to GDB compare-sections and set remotetimeout commands .Add “dv”, “mw” commands to probe driver.
V1.2.1.0	Mar 27 th , 2008	Add Linux Driver installation guide. Add multi-probe support. Fixed some syntax error.
V1.2.1.1	Apr. 3 rd , 2008	Add rwatch introduction and FAQ of probe error. Modified the FAQ.
V1.2.2	Apr. 15 th , 2008	Add “ew” loop command support. Add prompt mode introduction. Add introduction to keyword FAST_LOAD_DISABLE and DMA_DISABLE.
V1.2.2.1	Apr. 18 th , 2008	Add detail introduction to GDB watch command.

		Add FAQ of step fail.
V1.2.3	May. 8 th ,2008	Add introduction to cache and flash operation. Add introduction to dump/load/exit operation. Fixed some type error.
V1.3.0	Jul. 7 th ,2008	Add introduction to all supported commands and provide command script example. Fixed flash and memory write/read error. Add introduction to probe hardware update.
V1.3.1	Oct.29 th ,2008	Add support for CPU RLX5281, RLX4281. Add introduction to more commands of GDB. Provide Read/Write function of IMEM range.
V1.3.2beta1	Dec.19 th ,2008	Add introduce to dump, append, restore functions of GDB. Correct some syntax error. Redefined the sequence of “ACC_TAP_NUM”. Add register map of processor.
V1.3.2beta2	Jan 23th,2009	Add introduction to GDB file IO. Add configure address for file IO.
V1.3.2	Mar 20 th ,2009	Add introduction to GDB commands “while/if/source/loop_break/loop_continue/set” . Add introduction to GDB script file and an example. Add default commands script file postfix.
V1.3.2.1	Mar 30, 2009	Add FAQ of “DMA time out”.
V1.3.2.2	May 14, 2009	Correct the introduction to “dump” command of GDB.
V1.3.2.3	May 15, 2009	Unified font. Add introduction to commands “r/R/si/SI/k/fo/lf/step/c”.

		Fixed some number. Add a FAQ for “run failed at a breakpoint”.
V1.3.3.0	Jun 17, 2009	Add introduction to wide range hardware breakpoint setting.
V1.3.3.1	Aug 21,2009	Add FAQ of Run-Time debug.
V1.3.4	Aug 28,2009	Add support to low JTAG frequency. Moved USB init behind TCP init. Fixed flash region write failed. Fixed hardware breakpoint range mask. Add target board reset detection. Add a bit for reset low edge detection and a clear reset flag bit. Add management of reset out of probe in driver. Avoid opening file with empty string name. Avoid “load” command conflict with BUS Tracer. Add support to general MIPS32bit cpu type. Fixed syntax.
V1.3.4.1	Sep 14,2009	Add introduction to option “LEXRA_CP0_SHOW” and “CP3_REGS_EXIST” for CP0 & CP3 registers debug.
V1.3.4.2	Sep 28,2009	Add more detail introduction to “MC” and “ACCESS_TAP_NUM”.
V1.3.4.2	Oct 10,2009	Add introduction to RLX Probe configure item “ SOFTBP_DISABLE”.
V1.3.4.5	Nov 13,2009	Add introduction to probe configure options: “DMA_ENABLE_ALL” and “FILE_IO_DISABLE”.
V1.3.4.6	Nov 16,2009	Add introduction to new probe configure option “QUIT_DEAD_AUTO” .

V1.3.4.7	Mar 4 th ,2010	Fixed some error. Add FAQ of “can’t access memory”.
V1.3.4.8	May 5 th ,2010	Add introduction to “ei/mi/ldimem/dpimem/IMEM_READ_WRITE/ RECONNECT_INIT”.
V1.3.5	May 31 st ,2010	Add support to RLX4081. Remove some information.
V1.3.5p5	Oct 11 st ,2010	Add introduction to “P_CMD_ENABLE”.
V2.0beta	Dec 3 rd ,2010	Add introduction to RLX Probe2.0. Add introduction to options “MSDK_DEBUG”, “IF_TYPE”, “FIRMWARE_AUTO_UPDATE”, “FIRMWARE_FILE_NAME”.
V2.0	Dec 23,2010	Add introduction to pc sample and interrupt control.
V2.0p1	May 26,2011	Add introduction to IMEM1 setting requirement in configure file.
V2.1p13	Jul. 13,2011	Add introduction to “ej/mj” command and EJTAG register number and alias. Add introduction to access all register with register number. Add all register number map.
V2.1p13	Aug.10,2011	Add introduction to “NON_INTRUSIVE_DEBUG” mode and setting. Fixed some typo.
V2.1p14	Aug 26,2011	Add introduction to “Debug multi-thread with GDB and RLX Probe”.
V2.2	Dec 8 th , 2011	Add more introductions to GDB File IO. Add introduction to RST_CPU and RST_PERIPHERAL. Add support to RX3081.
V2.2.1	Dec 16,2011	Add detail pin define for RLX Probe 1.0 20pin & 14pin connectors.
V2.2.2	Dec 18,2011	Add introduction to register alias in RLX Probe.
V2.2.3	Feb 10,2012	Add introduction to pdtrace commands.

		<p>Add introduction to memory fill.</p> <p>Add introduction to polling enable/disable.</p> <p>Add introduction to access probe memory/registers.</p>
V2.2.4	Mar 21,2012	<p>Add missing introduction to “b/bs/hb/hbc” console commands.</p> <p>Add introduction to “pdover/pdcab”.</p>
V2.2.8	Sep 5,2012	<p>Add introduction to configure options: “TRIG_WHEN_CONNECT”, “FPU_REGS_EXIST” , “RECONNECT_INIT_OFF”, “TLB_MAPPED_ADDRESS_PROTECT”, “TLB_MAPPED_ADDRESS_ACCESS” .</p> <p>Add introduction to maintenance packet for TLB mapped address protection.</p> <p>Reword some lines.</p>
V2.2.10	Nov 20 th ,2012	<p>Add introduction to command “switch” and more detail for “config” command.</p> <p>Add introduction to expression.</p>
V2.3	May 14 th , 2013	<p>Add support to processor ARM Cortex-M3</p>
V2.3.1	Jun 3 rd , 2013	<p>Add support to Hardware Breakpoint by FPB for ARM Cortex-M3.</p> <p>Add introduction to configure options:”CM3_FPB_HB_ENABLE”, “FAST_LOAD_ENABLE”.</p> <p>Add support to GDB D-packet for CM3.</p> <p>Add support to PCSample for CM3.</p>
V2.3.2	Jul 16 th , 2013	<p>Add introduction to configure options: “PAUSE” “RESUME”.</p> <p>Add introduction to configure options: “ARM_AHB_BUS_ENABLE”</p> <p>Add support to enable debug trap on exceptions: HARDERR, INTERR, BUSERR, STATERR, CHKERR, NOCPERR, and MMERR.</p>
V2.3.2p1	Jul 23 rd , 2013	<p>Add introduction to configure options: “WRITE_ALIGN_4” “BYTE_HW_READ_ENABLE”.</p>

V2.3.3	Sep 6 th , 2013	Add support to Cortex-A9 Add register number mapping for ARM Add introduction to configure options: “ARMA_VFP_ON” “ARMA_NEON_ON”.
V2.3.4	Oct 9 th , 2013	Add introduction to command: “rstboard” & “rstjtag”. Update Register Number Map for ARM Cortex-A series.
V2.3.5	Dec 20 th , 2013	Add introduction to command: “fpen”, “fclose”, “fsize”, “fread”, “fwrite” & “ftell” Add support to new processor HAWK
V2.3.6	Jun 17 th , 2014	Add support to ARM SWD mode. Add introduction to configure options: “ARM_SWD_EN”. Add introduction to command: “watch”, “rwatch”, “rwatchc”.
V2.3.7	Jan 12 th , 2015	Add support to Cortex-A53 (ARMv8 aarch32/aarch64). Add support to JTAG Master for performance monitor POPtrace function.
V2.3.8Beta	Jan 29 th , 2015	Add support to ARMv8 aarch32 System control registers. Add support to APB XTOR for ARMv7.
V2.3.9	Apr 29 th , 2016	Add support to ARM Cortex-M4/M7. Add support to KM Processor.
V2.3.10	Jun 26 th , 2016	Add support to JTAG-AP. Add support to ARMv8-m.
V2.3.11	May 8 th , 2018	Add support to Cortex-R. Add support to ANANKE. Add support to Secure/Non-secure for ARM. Add introduction to configure options: “IDAU_BASE_ADDR”.

		Update MIPS CPU Register list in Appendix A.
--	--	--