

Converter from GDB to PRACTICE

Release 02.2025



MANUAL

Converter from GDB to PRACTICE

TRACE32 Online Help

TRACE32 Directory

TRACE32 Index

| | |
|---|---|
| TRACE32 Documents |  |
| GDB Support |  |
| Converter from GDB to PRACTICE | 1 |
| Introduction | 3 |
| Launching Converter | 4 |
| Converter Limitations | 5 |
| Converter-Specific Reserved Identifiers | 6 |
| Using History Convenience Variables in CMM Script | 7 |
| Using PRACTICE Commands from GDB Scripts | 8 |
| Supported Commands | 9 |
| Getting In and Out of GDB (TRACE32) | 9 |
| Running Programs Under GDB (TRACE32) | 9 |
| Stopping and Continuing | 10 |
| Examining the Stack | 13 |
| Examining Source Files | 14 |
| Examining Data | 15 |
| C Preprocessor Macros | 16 |
| Examining the Symbol Table | 17 |
| Using GDB (TRACE32) with Different Languages | 18 |
| Altering Execution | 18 |
| GDB (TRACE32) Files | 18 |
| Specifying a Debugger Target | 19 |
| Controlling GDB (TRACE32) | 19 |
| Command Files | 20 |
| Controlled Output | 20 |
| User Interface | 21 |
| Others | 21 |

Introduction

This document describes how to use the GNU Debugger to PRACTICE Script Converter. The executable file can be found in the TRACE32 installation directory under `~/demo/tools/gdb_converter`.

| | |
|----------------|--|
| Format: | converter [-a=ARCH_NAME{,ARCH_NAME}] [-e <error_file>] [-d] [-h] [-l] <input_file> <output_file> |
| <input_file> | Input GDB script file |
| <output_file> | Output PRACTICE cmm file |
| -a | Select target architectures. This option uses architecture definitions from architectures.def file. Multiple architectures can be defined, however program counter and data memory class definitions are taken only from first defined architecture. |
| -e | Redirect all warning and error messages to <error_file>. |
| -d | Print not supported GDB commands to output file as commented lines instead of generating errors. |
| -l | Print line numbers. |
| -h | Print help. |

Converter Limitations

The converter supports only C expressions in commands.

Following convenience variables are not supported by converter:

`$_exitcode`, `$cdir`, `$tpnum`, `$trace_file`, `$trace_frame`, `$trace_func`, `$trace_line`, `$tracepoint`

| Location | Limitations |
|---------------|---------------|
| FILE:LINENUM | not supported |
| FILE:FUNCTION | not supported |

Other limitations for commands are listed in “chapter 7 – Supported commands”

Converter-Specific Reserved Identifiers

Following identifiers cannot be used for convenience variables:

__V0, __V1, __V2, ...

history_0x0, history_0x1, history_0x2, ...

breakpoints_exist_0x0, breakpoints_exist_0x1, breakpoints_exist_0x2, ...

breakpoints_address_0x0, breakpoints_address_0x1, breakpoints_address_0x2, ...

x_default_address, history_last, history_before_last, history_count, breakpoints_count.

Following identifiers are used for labels in PRACTICE scripts generated by converter. Using these identifiers may be UNPREDICTABLE

__L0, __L1, __L2, ...

update_history_values, get_history_back_value,

breakpoints_set, breakpoints_get_address_and_delete, breakpoints_get_address,

breakpoints_get_exist, breakpoints_print_list

Using History Convenience Variables in CMM Script

GDB history variables - \$, \$\$, \$n, \$\$n – can be accessed in CMM output script by using following equivalents:

| | |
|-------|----------------------|
| \$: | &history_last |
| \$\$: | &history_before_last |
| \$n: | &history_0xN |

(N is hexadecimal representation of n. No leading 0's are allowed in N)

| | |
|--------|--|
| \$\$n: | <p>This variable is read-only and can be accessed by calling subprogram <code>get_history_back_value</code>:</p> <p>GOSUB <code>get_history_back_value</code> n.</p> <p>ENTRY &value</p> |
|--------|--|

&value contains the history value after **get_history_back_value** subprogram returns. “n” must be followed by dot, to be treated as decimal number in PRACTICE script.

Using PRACTICE Commands from GDB Scripts

There is the possibility to execute PRACTICE command directly from GDB script. To execute PRACTICE command from GDB script, use the following construction:

```
#!<practice_command>
```

For example:

```
#!system.up
```

Construction above must be placed in an empty line.

Architecture definitions file format (architectures.def)

architectures.def file allows converter to recognize target registers such as \$pc, \$sp, \$r0 etc. in GDB script file and treat them properly while converting. Names of architectures comes directly from sys.cpu TRACE32 window. Original architectures.def file contains over 1200 architecture names.

NOTE:

There is one exception with architecture names: 64-bit MIPS names contains '-64' suffix to distinguish them from 32-bit MIPS, for example MIPS5K-64.

Format of each entry in architectures.def:

```
<cpu>
  <name>ARM7TDMI</name>
  <data_memory_class>d</data_memory_class>
  <program_counter>pc</program_counter>
  <reg>pc</reg>
  <reg>r0</reg>
  <reg>r1</reg>
  ...
</cpu>
```

If register set for some architecture is the same as for previously defined architecture, **<reg_reference>**referenced_architecture_name</reg_reference> can be used instead of sequence of **<reg></reg>**

```
<cpu>
  <name>ARM7TDMI</name>
  <data_memory_class>d</data_memory_class>
  <program_counter>pc</program_counter>
  <reg_reference>ARM7</reg_reference>
</cpu>
```


Supported Commands

Getting In and Out of GDB (TRACE32)

| Operations | PRACTICE command |
|------------------|---------------------------|
| quit | QUIT |
| shell | OS |
| set logging file | AREA.OPEN A000 <log_file> |

Running Programs Under GDB (TRACE32)

| Operations | PRACTICE command |
|---------------------------|--|
| cd | CD Limitations: Argument version of 'run' is not supported. |
| r run | GO Limitations: Argument version of 'start' is not supported. |
| start | GO main |
| pwd | PWD |
| attach | SYSTEM.MODE.ATTACH Limitations: Argument version of 'attach' is not supported. |
| i threads info threads | TASK.THREADS |

Stopping and Continuing

| Operations | PRACTICE command |
|---------------------------------------|--|
| i program info program | IF RUN() PRINT "The debugged program is running." ELSE PRINT "The debugged program is not running." |
| rwatch | VAR.BREAK.SET <expr> /READ Limitations: Target registers (\$pc, \$sp, ...) and history \$\$n cannot be used in expression. |
| watch | VAR.BREAK.SET <expr> /WRITE Limitations: Target registers (\$pc, \$sp, ...) and history \$\$n cannot be used in expression. |
| interrupt | BREAK |
| clear | BREAK.DELETE Limitations: Non-argument version of 'clear' is not supported. 'Clear' command doesn't interact with 'info breakpoints' command – Listing of breakpoints contains cleared breakpoints. |
| d delete delete breakpoints | BREAK.DELETE Limitations: This commands interacts only with following commands: break, hbreak, tbreak, thbreak. |
| dis disable disable breakpoints | BREAK.DISABLE Limitations: This commands interacts only with following commands: break, hbreak, tbreak, thbreak. |
| enable enable breakpoints | BREAK.ENABLE Limitations: This commands interacts only with following commands: break, hbreak, tbreak, thbreak. |
| i breakpoints info breakpoints | BREAK.LIST Limitations: If last breakpoint listed has been set using line number, default examine address command cannot be used by "x" command. Using this address will cause PRACTICE script error. Only non-argument version of 'info breakpoints' is supported. |
| b break | BREAK.Set <location> Limitations: Non-argument version of 'break' is no supported. THREADNUM parameter is not supported by TRACE32. Target registers (\$pc, \$sp, ...) and history values \$\$n are not supported in CONDITION expression. |

| Operations | PRACTICE command |
|--|--|
| hbreak | BREAK.Set <location> /HARD Limitations: Non-argument version of 'hbreak' is not supported. THREADNUM parameter is not supported by TRACE32. Target registers (\$pc, \$sp, ...) and history values \$\$n are not supported in CONDITION expression. |
| tbreak | BREAK.Set <location> /DISABLEHIT Limitations: Non-argument version of 'tbreak' is not supported. Temporary breakpoint deleting after hit is not supported by TRACE32. Breakpoint will be disabled instead. THREADNUM parameter is not supported by TRACE32. Target registers (\$pc, \$sp, ...) and history values \$\$n are not supported in CONDITION expression. |
| thbreak | BREAK.Set <location> /HARD /DISABLEHIT Limitations: Non-argument version of 'thbreak' is not supported. Temporary breakpoint deleting after hit is not supported by TRACE32. Breakpoint will be disabled instead. THREADNUM parameter is not supported by TRACE32. Target registers (\$pc, \$sp, ...) and history values \$\$n are not supported in CONDITION expression. |
| finish | GO.RETURN STEP Limitations: Printing return value is not supported. |
| c fg continue | GO Limitations: Argument version of 'continue' is not supported. |
| advance | GO <location> Limitations: Program will not stop after exiting from current stack frame. THREADNUM and CONDITION are not supported. |
| u until | GO <location> Limitations: Non-argument version of 'until' is not supported. THREADNUM and CONDITION are not supported. |
| n next | MODE.HLL [REPEAT <n>] STEP.OVER |
| s step | MODE.HLL STEP [<n>] |
| ni nexti | MODE.MIX [REPEAT <n>] STEP.OVER |

| Operations | PRACTICE command |
|-----------------------------------|--|
| si stepi | MODE.MIX STEP [<i><n></i>] |
| i watchpoints info watchpoints | BREAK.LIST Limitations: Argument version of 'info watchpoints' is not supported. |
| awatch | VAR.BREAK.SET <i><expr></i> /R /W Limitations: Target registers (\$pc, \$sp, ...) and history values \$\$n cannot be used in <i><expr></i> . |

Examining the Stack

| Operations | PRACTICE command |
|--|---|
| up | GO.UP [<n>] Limitations: Printing stack frame is not supported. |
| up-silently | GO.UP [<n>] |
| i args info args | VAR.FRAME /ARGS |
| i frame info frame | VAR.FRAME /LOCALS /CALLER /ARGS Limitations: Argument version of 'info frame' is not supported. |
| bt backtrace | VAR.FRAME /NOVAR /NOCALLER [/LOCALS] Limitations: Only non-argument version of 'backtrace' or 'backtrace full' is supported. |
| i stack info stack i s info s | VAR.FRAME /NOVAR /NOCALLER Limitations: Only non-argument version of 'info stack' is supported. |
| where | VAR.FRAME /NOVAR /NOCALLER [/LOCALS] Limitations: Only non-argument version of 'where' or 'where full' is supported. |
| i locals info locals | VAR.LOCAL Limitations: Only non-argument version of 'where' or 'where full' is supported. |

| Operations | PRACTICE command |
|------------------|---|
| disassemble | MODE.MIX DATA.LIST |
| l list | DATA.LIST Limitations: Non argument-version of 'list' is not supported. |
| show directories | SYMBOL.SOURCEPATH.LIST |
| directory | SYMBOL.SOURCEPATH.SET <directory> Limitations: Convenience variable \$cdir is not supported as <directory>. |

| Operations | PRACTICE command |
|-----------------------------------|--|
| i dache info dcache | CTS.CACHE.STATE |
| dump memory dump binary memory | DATA.SAVE.BINARY <file> <address_range> |
| dump ihex memory | DATA.SAVE.INTELHEX <file> <address_range> |
| dump srec memory | DATA.SAVE.S1RECORD <file> <address_range> Limitations: S1RECORD is used instead of SRECORD. |
| dump tekhex memory | DATA.SAVE.TEKHEX <file> <address_range> |
| x | PRINT or DATA.PRINT Limitations: Default parameters for 'x' command are constants (n=1, f=x, u=w) and cannot be changed. This means that previous 'x' or 'print' command has no influence on default format used by 'x' command. 'a' and 'i' formats are not supported. For 'f' format unit size can be either 'w' or 'g'. Other unit sizes are not supported. For 'x', 'u', 't', 'c', 's' formats examined values are printed on TRACE32 AREA window. For 'd', 'o', 'f' formats PRACTICE script uses separate DATA.PRINT window for each 'x' command. |
| display | VAR.LOG <expr> /ONBREAK Limitations: Because VAR.LOG needs to provide all expression at once following rules applies when 'display' command is being used: - All 'display' commands have to be placed in sequence, one after another, without any blank lines between them. - Next sequence of 'display' commands (on script execution flow), discards previous 'display' sequence. Expressions from this previous sequence are no longer displayed. Only following formats of 'display' are supported: /x, /u, /t, /c, /s. |
| show convenience | PMACRO.LIST |
| inspect | PRINT Limitations: Behavior the same as 'print' command. |
| p print | PRINT Limitations: Only 'x', 'u', 't', 'c', 's' formats are supported. Values are always printed with new lines. |
| set print asm-demangle off | SYMBOL.DEMANGLE OFF OFF |
| set print demangle off | SYMBOL.DEMANGLE OFF OFF |
| set print asm-demangle on | SYMBOL.DEMANGLE ON ON |

| Operations | PRACTICE command |
|--|--|
| set print demangle on | SYMBOL.DEMANGLE ON ON |
| i all-registers info all-registers i registers info registers | REGISTER (for non-argument version) PRINT register_name REGISTER(register_name) |
| dump value dump binary value | OPEN #1 <file> /CREATE WRITE #1 %BINARY <value> CLOSE #1 Limitations: Raw binary format is not supported. Values are written to file as binary numbers. |
| dump ihex value dump tekhex value | OPEN #1 <file> /CREATE WRITE #1 %HEX <value> CLOSE #1 Limitations: 'ihex' and 'tekhex' formats are not supported. Values are written to file as hexadecimal numbers. |
| append value append binary value | IF OS.FILE(<file>) OPEN #1 <file> /WRITE ELSE OPEN #1 <file> /CREATE WRITE #1 %BINARY <value> CLOSE #1 Limitations: Raw binary format is not supported. Values are written to file as binary numbers. |

C Preprocessor Macros

| Operations | PRACTICE command |
|--------------|--------------------------|
| macro list | SYMBOL.LIST.MACRO |
| macro define | SYMBOL.NEW.MACRO <macro> |

Examining the Symbol Table

| Operations | PRACTICE command |
|-------------------------------|--|
| i types info types | SYMBOL.LIST.TYPE Limitations: Only non-argument version of 'info types' is supported. |
| i address info address | DATA.PRINT V.ADDRESS(<symbol_name>) |
| i symbol info symbol | SYMBOL.INFO <symbol> <address> |
| i classes info classes | SYMBOL.CLASS <class_name> Limitations: Non-argument version of 'info classes' is not supported. Only strict class_name are supported – class_name cannot be regular expression. |
| i functions info functions | SYMBOL.LIST.FUNCTIONS Limitations: Only non-argument version of 'info functions' is supported. |
| i sources info sources | SYMBOL.LIST.SOURCE |
| ptype | VAR.TYPE <type> Limitations: Only argument version of 'ptype' is supported. |
| whatis | VAR.TYPE <expr> Limitations: Target registers (\$pc, \$sp, ...) and history values \$\$n cannot be used in <expr>. |

Using GDB (TRACE32) with Different Languages

| Operations | PRACTICE command |
|---------------|---|
| Set language | SYMBOL.LANGUAGE <i><language_name></i> |
| show language | SYMBOL.LANGUAGE |

Altering Execution

| Operations | PRACTICE command |
|-----------------------------------|--|
| set set variable | Argument of 'set' command (assignment expression) is evaluated in PRACTICE script. |
| jump | IF RUN() BREAK REGISTER.SET PC <i><jump_address></i> <i><line_number></i> |
| call | VAR <i><function_call></i> |

GDB (TRACE32) Files

| Operations | PRACTICE command |
|--------------------|---|
| symbol-file | DATA.LOAD.AUTO <i><file></i> /NoCODE or SYMBOL.DELETE (for non-argument version) |
| file | DATA.LOAD.AUTO <i><file></i> Limitations: Non-argument version of 'file' is not supported. |

Specifying a Debugger Target

| Operations | PRACTICE command |
|-----------------------------------|--|
| load | DATA.LOAD.AUTO <file> [<offset>] |
| processor show architecture | PRINT CPU() |
| show endian | IF SYSTEM.BIGENDIAN() PRINT "Current endianness is big endian." ELSE PRINT "Current endianness is little endian." |
| set architecture set processor | SYSTEM.CPU <architecture_name> Limitations: Architecture name 'auto' is not supported. |
| set endian little | SYSTEM.OPTION.BIGENDIAN OFF |
| set endian big | SYSTEM.OPTION.BIGENDIAN ON |

Controlling GDB (TRACE32)

| Operations | PRACTICE command |
|---|--|
| set history size | HISTORY.SIZE |
| show commands | HISTORY.TYPE Limitations: Argument version of 'show commands' is not supported. |
| set history filename set history save off set history save on set history save | This commands are not supported in TRACE32 with exactly the same behavior in as in GDB. This commands are printed to output PRACTICE script as commented lines. |

Command Files

| Operations | PRACTICE command |
|---|---|
| if else end while loop_break loop_continue | This set of commands is fully supported by converter by using IF , GOTO and __Ln labels (n=0,1,2,...) in PRACTICE scripts. |
| source | DO <cmn_script_file> Notice: 'source' argument must point to cmn practice script |

Controlled Output

| Operations | PRACTICE command |
|---------------|---|
| echo | PRINT Limitations: Only following backslash-escape sequences are supported: \\, \n, \t, \". New line is always printed after text. |
| output | Behavior the same as 'print' command. |
| printf | PRINT Limitations: Only following backslash-escape sequences are supported: \\, \n, \t, \". Only following % formats are supported: %x, %u, %c, %s. New line is always printed after text. |

User Interface

| Operations | PRACTICE command |
|------------|---|
| layout | layout src: DATA.LIST layout asm: MODE.MIX, DATA.LIST layout split: MODE.MIX, DATA.LIST layout regs: REGISTER Limitations: “prev” and “next” parameters are not supported. |
| refresh | SCREEN |
| update | SCREEN |
| tabset | SETUP.TABSIZE <n> |

Others

| Operations | PRACTICE command |
|--------------|---|
| show | This commands calls following supported 'show' subcommands: show architecture show commands show convenience show directories show endian show language show version |
| show version | VERSION |