

# Converter from GDB to PRACTICE



# 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`.

<b>Format:</b>	<b>converter [-a=ARCH_NAME{,ARCH_NAME}] [-e &lt;error_file&gt;] [-d] [-h] [-l] &lt;input_file&gt; &lt;output_file&gt;</b>
<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><b>GOSUB <code>get_history_back_value</code> n.</b></p> <p><b>ENTRY &amp;value</b></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  <b>Limitations:</b> Argument version of 'run' is not supported.
r run	GO  <b>Limitations:</b> Argument version of 'start' is not supported.
start	GO main
pwd	PWD
attach	SYSTEM.MODE.ATTACH  <b>Limitations:</b> Argument version of 'attach' is not supported.
i threads info threads	TASK.THREADS

## Stopping and Continuing

Operations	PRACTICE command
i program info program	<b>IF RUN()</b> <b>PRINT "The debugged program is running."</b> <b>ELSE</b> <b>PRINT "The debugged program is not running."</b>
rwatch	<b>VAR.BREAK.SET &lt;expr&gt; /READ</b>  <b>Limitations:</b> Target registers (\$pc, \$sp, ...) and history \$\$n cannot be used in expression.
watch	<b>VAR.BREAK.SET &lt;expr&gt; /WRITE</b>  <b>Limitations:</b> Target registers (\$pc, \$sp, ...) and history \$\$n cannot be used in expression.
interrupt	<b>BREAK</b>
clear	<b>BREAK.DELETE</b>  <b>Limitations:</b> 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	<b>BREAK.DELETE</b>  <b>Limitations:</b> This commands interacts only with following commands: break, hbreak, tbreak, thbreak.
dis disable disable breakpoints	<b>BREAK.DISABLE</b>  <b>Limitations:</b> This commands interacts only with following commands: break, hbreak, tbreak, thbreak.
enable enable breakpoints	<b>BREAK.ENABLE</b>  <b>Limitations:</b> This commands interacts only with following commands: break, hbreak, tbreak, thbreak.
i breakpoints info breakpoints	<b>BREAK.LIST</b>  <b>Limitations:</b> 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	<b>BREAK.Set &lt;location&gt;</b>  <b>Limitations:</b> 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
<b>hbreak</b>	<b>BREAK.Set &lt;location&gt; /HARD</b>  <b>Limitations:</b> Non-argument version of 'hbreak' is no supported. THREADNUM parameter is not supported by TRACE32. Target registers (\$pc, \$sp, ...) and history values \$\$n are not supported in CONDITION expression.
<b>tbreak</b>	<b>BREAK.Set &lt;location&gt; /DISABLEHIT</b>  <b>Limitations:</b> Non-argument version of 'tbreak' is no 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.
<b>thbreak</b>	<b>BREAK.Set &lt;location&gt; /HARD /DISABLEHIT</b>  <b>Limitations:</b> Non-argument version of 'thbreak' is no 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.
<b>finish</b>	<b>GO.RETURN STEP</b>  <b>Limitations:</b> Printing return value is not supported.
<b>c</b> <b>fg</b> <b>continue</b>	<b>GO</b>  <b>Limitations:</b> Argument version of 'continue' is not supported.
<b>advance</b>	<b>GO &lt;location&gt;</b>  <b>Limitations:</b> Program will not stop after exiting from current stack frame. THREADNUM and CONDITION are not supported.
<b>u</b> <b>until</b>	<b>GO &lt;location&gt;</b>  <b>Limitations:</b> Non-argument version of 'until' is not supported. THREADNUM and CONDITION are not supported.
<b>n</b> <b>next</b>	<b>MODE.HLL</b> <b>[REPEAT &lt;n&gt;] STEP.OVER</b>
<b>s</b> <b>step</b>	<b>MODE.HLL</b> <b>STEP [&lt;n&gt;]</b>
<b>ni</b> <b>nexti</b>	<b>MODE.MIX</b> <b>[REPEAT &lt;n&gt;] STEP.OVER</b>

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

## Examining the Stack

Operations	PRACTICE command
up	<b>GO.UP [&lt;n&gt;]</b>  <b>Limitations:</b> Printing stack frame is not supported.
up-silently	<b>GO.UP [&lt;n&gt;]</b>
i args info args	<b>VAR.FRAME /ARGS</b>
i frame info frame	<b>VAR.FRAME /LOCALS /CALLER /ARGS</b>  <b>Limitations:</b> Argument version of 'info frame' is not supported.
bt backtrace	<b>VAR.FRAME /NOVAR /NOCALLER [/LOCALS]</b>  <b>Limitations:</b> Only non-argument version of 'backtrace' or 'backtrace full' is supported.
i stack info stack i s info s	<b>VAR.FRAME /NOVAR /NOCALLER</b>  <b>Limitations:</b> Only non-argument version of 'info stack' is supported.
where	<b>VAR.FRAME /NOVAR /NOCALLER [/LOCALS]</b>  <b>Limitations:</b> Only non-argument version of 'where' or 'where full' is supported.
i locals info locals	<b>VAR.LOCAL</b>  <b>Limitations:</b> 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>  <b>Limitations:</b> S1RECORD is used instead of SRECORD.
dump tekhex memory	DATA.SAVE.TEKHEX <file> <address_range>
x	<b>PRINT</b> or <b>DATA.PRINT</b>  <b>Limitations:</b> 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	<b>VAR.LOG &lt;expr&gt; /ONBREAK</b>  <b>Limitations:</b> 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	<b>PRINT</b>  <b>Limitations:</b> Behavior the same as 'print' command.
p print	<b>PRINT</b>  <b>Limitations:</b> 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  <b>Limitations:</b> 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  <b>Limitations:</b> '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  <b>Limitations:</b> 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	<b>SYMBOL.LIST.TYPE</b>  <b>Limitations:</b> Only non-argument version of 'info types' is supported.
i address info address	<b>DATA.PRINT V.ADDRESS</b> ( <i>&lt;symbol_name&gt;</i> )
i symbol info symbol	<b>SYMBOL.INFO</b> <i>&lt;symbol&gt;</i>   <i>&lt;address&gt;</i>
i classes info classes	<b>SYMBOL.CLASS</b> <i>&lt;class_name&gt;</i>  <b>Limitations:</b> 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	<b>SYMBOL.LIST.FUNCTIONS</b>  <b>Limitations:</b> Only non-argument version of 'info functions' is supported.
i sources info sources	<b>SYMBOL.LIST.SOURCE</b>
ptype	<b>VAR.TYPE</b> <i>&lt;type&gt;</i>  <b>Limitations:</b> Only argument version of 'ptype' is supported.
whatis	<b>VAR.TYPE</b> <i>&lt;expr&gt;</i>  <b>Limitations:</b> Target registers (\$pc, \$sp, ...) and history values \$\$n cannot be used in <i>&lt;expr&gt;</i> .

# Using GDB (TRACE32) with Different Languages

Operations	PRACTICE command
Set language	SYMBOL.LANGUAGE <language_name>
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 <jump_address> <line_number>
call	VAR <function_call>

## GDB (TRACE32) Files

Operations	PRACTICE command
symbol-file	DATA.LOAD.AUTO <file> /NoCODE or SYMBOL.DELETE (for non-argument version)
file	DATA.LOAD.AUTO <file>  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
<b>if</b> <b>else</b> <b>end</b> <b>while</b> <b>loop_break</b> <b>loop_continue</b>	This set of commands is fully supported by converter by using <b>IF</b> , <b>GOTO</b> and <b>__Ln</b> labels (n=0,1,2,...) in PRACTICE scripts.
<b>source</b>	<b>DO</b> <cmn_script_file>  <b>Notice:</b> 'source' argument must point to cmn practice script

## Controlled Output

---

Operations	PRACTICE command
<b>echo</b>	<b>PRINT</b>  <b>Limitations:</b> Only following backslash-escape sequences are supported: \\, \n, \t, \". New line is always printed after text.
<b>output</b>	Behavior the same as 'print' command.
<b>printf</b>	<b>PRINT</b>  <b>Limitations:</b> 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: <b>DATA.LIST</b> layout asm: <b>MODE.MIX, DATA.LIST</b> layout split: <b>MODE.MIX, DATA.LIST</b> layout regs: <b>REGISTER</b>  <b>Limitations:</b> “prev” and “next” parameters are not supported.
refresh	<b>SCREEN</b>
update	<b>SCREEN</b>
tabset	<b>SETUP.TABSIZE</b> <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	<b>VERSION</b>