LAUTERBACH
DEVELOPMENT TOOLS

Linux Debugging Reference Card

This reference card gives you an overview of frequently-used **TRACE32®** commands for debugging targets running Linux (stop mode).

Configure Linux for Debugging

Compile kernel with debug info	`CONFIG_DEBUG_INFO=y`
	`#CONFIG_DEBUG_INFO_REDUCED not set`
For module debugging set	`CONFIG_KALLSYMS=y`
Trace and task specific breakpoints (ARM only)	`CONFIG_PID_IN_CONTEXTIDR=y`
Disable KASLR (x86/x64 only)	`#CONFIG_RANDOMIZE_BASE not set`
Compile applications with option	`-g`

Attach to a Running Linux Target

```
;Initialize debugger
SYStem.CPU <cpu>
SYStem.Option <option>

;Enable space IDs
SYStem.Option MMUSPACES ON

;Connect debugger without reset assertion
SYStem.Mode Attach
Break

;Load kernel symbols
Data.LOAD.Elf vmlinux /NoCODE
```

Continue with **Configure Linux Awareness.** Actual sequence heavily depends on target system.

Download and Run Linux with TRACE32

```
;Initialize debugger
SYStem.CPU <cpu>
SYStem.Option <option>

;Enable space IDs
SYStem.Option MMUSPACES ON

;Connect to boot core only
CORE.ASSIGN <physical boot core>

;Connect debugger and try to stop at reset vector
SYStem.Up

;Let bootloader (e.g. U-Boot) initialize the system
Go
WAIT 5.s
Break

;Load FIT image (Flattened-Image-Tree) to physical address
Data.LOAD.Binary image.FIT <phys.addr>

;Load kernel symbols
Data.LOAD.Elf vmlinux /NoCODE

;Continue bootloader – set temporary breakpoint to 'start_kernel'
Go start_kernel /Onchip

;Patch bootargs for debugging (in terminal window)
setenv bootargs … nowatchdog \
  rcupdate.rcu_cpu_stall_suppress=1

;Start FIT image with bootloader (in terminal window)
bootm <phys.addr>

;Wait until breakpoint (at start_kernel) is hit
WAIT !STATE.RUN()
```

Continue with **Configure Linux Awareness.** Actual sequence heavily depends on target system and loading mechanisms.

Configure Linux Awareness

Mandatory for Userspace and Module debugging.
Examples available: `~~/demo/<arch>/kernel/linux/boards`.

MMU Declaration

```
MMU.FORMAT LINUX swapper_pg_dir \
  <virtual address range of swapper_pg_dir>\
  <physical start address>
TRANSlation.TableWalk ON
TRANSlation.ON
```

Define Common Range

```
TRANSlation.COMMON \
  <kernel+module virtual address range>
```

Load Linux Awareness for v3.x/v4.x

```
TASK.CONFIG \
~~/demo/<arch>/kernel/linux/linux-3.x/linux3.t32
MENU.ReProgram \
~~/demo/<arch>/kernel/linux/linux-3.x/linux.men
```

Define Groups

```
GROUP.Create "kernel" \
  <kernel virtual address range> /RED
```

Linux Resource Displays

Processes

```
;Linux menu ⟶ Display Processes
TASK.Process
```

Detailed task info

```
;Linux menu ⟶ Display Processes ⟶ Display Tasks
TASK.DTask "<name>"
```

"ps"

```
;Linux menu ⟶ Display ps-like
TASK.PS
```

File system internals

```
;Linux menu ⟶ Display File System
TASK.FS.*
```

Kernel log buffer

```
;Linux menu ⟶ Display Kernel Log
TASK.DMESG
```

Device tree blob

```
;Linux menu ⟶ Device Tree
TASK.DTB
```

Debug Processes

Display processes

```
;Linux menu ⟶ Display Processes
TASK.Process
```

Configure symbol loader

```
;Linux menu ⟶ Symbol Autoloader
TASK.sYmbol.Option AutoLoad Process
```

Start debugging at main()

```
;Linux menu ⟶ Process Debugging ⟶ Debug new Process
DO ~~/demo/<arch>/kernel/linux/linux-3.x/ \
   app_debug <process>
```

Watch process starts

```
;Linux menu ⟶ Process Debugging ⟶ Watch Processes
TASK.Watch
```

Load symbols for running processes

```
;Linux menu ⟶ Display Processes ⟶ Right click on "magic"
⟶ Load Process Symbols
TASK.sYmbol.LOAD "<name>"
```

Show thread context

```
;Linux menu ⟶ Display Processes ⟶ Right click on "magic"
⟶ Display Stack Frame
Frame /TASK "<name>"
```

Thread specific breakpoints

```
TrOnchip.ContextID ON ;Set to ON (ARM only)
;Select breakpoint ⟶ advanced ⟶ TASK
Break.Set <addr> /TASK "<name>"
```

Debug Libraries

Display libraries

```
;Linux menu ⟶ Display Processes ⟶ right-click on "magic"
⟶ Display maps
TASK.MAPS "<name>"
```

Configure symbol loader

```
;Linux menu ⟶ Symbol Autoloader
TASK.sYmbol.Option AutoLoad Library
```

Load symbols

```
;Display maps ⟶ Right-click on library "magic"
⟶ Load Library Symbols
TASK.sYmbol.LOADLib <proc> <lib>
```

Debug Kernel Modules

Display modules

```
;Linux menu ⟶ Display Modules
TASK.MODule
```

Configure symbol loader

```
;Linux menu ⟶ Symbol Autoloader
TASK.sYmbol.Option AutoLoad Module
```

Start debugging at module init routine

```
;Linux menu ⟶ Module Debugging ⟶ Debug Module on init
DO ~~/demo/<arch>/kernel/linux/linux-3.x/ \
   mod_debug <module>
```

Load module symbols

```
;Linux menu ⟶ Display Modules ⟶ Right click on "magic"
⟶ Load Module Symbols
TASK.sYmbol.LOADMod "<name>"
```

SMP Support

Setup	**Attach Debugging** The debugger automatically selects all cores when executing `SYStem.CPU <cpu>`. **Start Debugging** Use `CORE.ASSIGN` to select the physical boot core and switch to SMP after `smp_init_done` is called.
View	Debugger shows context of one core. State line shows current core number. Data/Register windows change color.
Change core	Right-click on core number in state line. Use `CORE.select <x>` command. Use `/CORE <x>` option in window context.
Breakpoints	Are set on all cores, current view switches automatically to core that hits the breakpoint.

```
Go smp_init_done
WAIT !STATE.RUN()

SYStem.Mode Down
CORE.ASSIGN 1. 2. …
SYStem.Mode Attach
```

Configure Trace-based Task Detection

Using data trace (e.g. PowerPC)

```
Break.Set TASK.CONFIG(magic) /Write /TraceData
```

Using Context ID (ARM only)

```
ETM.ContextID 32
```

Filter Trace Recording

Generic syntax

```
Break.Set <space ID>:<virtual address range>
  /TraceEnable
```

Example: Filter Kernel function `__switch_to`

```
Var.Break.Set __switch_to /TraceEnable
```

Example: Filter Kernel, Modules, Userspace running in Task context

```
;Linux menu ⟶ Display Processes ⟶ right-click on "magic"
⟶ trace this task
Break.Set <space ID>:0x0--0xffffffffffffffff
  /TraceEnable
;Use TASK.MAPS "<name>" to adapt range e.g. to
;task+libraries
```

Trace-based Task Profiling

Display task switches

```
Trace.List List.TASK /DEFault
```

Statistics

```
;Perf menu ⟶ Task runtime ⟶ Show numerical
Trace.STATistic.TASK
```

Charts

```
;Perf menu ⟶ Task runtime ⟶ Show graphical
Trace.Chart.TASK
```

Trace-based Function Profiling

Statistics

```
;Perf menu ⟶ Task function runtime ⟶ Show numerical
Trace.STATistic.Func

;Filter specific task in the recorded trace e.g. offchip-trace
Trace.STATistic.Func /TASK "<name>"
```

Charts

```
;Perf menu ⟶ Task function runtime ⟶ Show graphical
Trace.Chart.Func

;Filter specific task in the recorded trace e.g. offchip-trace
Trace.Chart.Func /TASK "<name>"
```

Documentation / Help / Support

User manuals	pdf/rtos_linux_stop.pdf pdf/rtos_linux_run.pdf
Training manual	pdf/training_rtos_linux.pdf pdf/training_rtos_linux_x86.pdf
Example scripts	demo/<arch>/kernel/linux
Support	local sales offices, rtoslinux-support@lauterbach.com

Lauterbach GmbH • Altlaufstraße 40 • 85635 Höhenkirchen • info@lauterbach.com