

Linux Debugging

- ARM[®]/Cortex[™]
- Intel[®] Atom[™]
- MIPS[®] Architecture
- Power Architecture[®]
- and others





Linux Debugging Environment

TRACE32 supports all Linux distributions, bundles and platforms using the standard Linux kernel (e.g. Android). It offers:

- Linux-aware debugging of single core systems
- Linux-aware debugging of SMP systems

Stop-mode Debugging

A JTAG debugger works with Stop-mode debugging. This means the processor and thus the whole system is stopped whenever a breakpoint is hit. This behavior makes it possible to analyze the overall system state at a specific point in time.

Benefits

- The only requirement for Stop-mode debugging is a functioning JTAG interface. This enables debugging from the reset vector.
- Debugging of the kernel and beyond process boundaries is possible because the TRACE32 debugger offers both Linux and MMU support.
- If the software ceases to react, the processor can be stopped to find out the point in the code where the processor crashed.
- If the processor is stopped, neither the kernel nor any other process can interfere with your analysis.

Linux Window Help		
Display <u>P</u>rocesses Display <u>p</u> s-like Display <u>T</u> asks Display <u>M</u> odules Display <u>F</u> ile System	•	
Process Debugging	Þ	Load Symbols
Module Debugging	F	Delete Symbols
Library Debugging	•	Debug Process on main
Symbol <u>A</u> utoloader	•	Watch Processes
<u>D</u> isplay Kernel Log		Scan Process MMU Pages Scan All MMU Tables
Linux Terminal	1	
Configure Terminal		
Generate ramdump		
Help Linux Awareness		

List of Linux processes									
magic	command	#thr	state	spaceid	pids				
C08EA540 🗄	swapper/0	56.	running	0000	0. 2.	3.4.	6.7.	8.9.	10. 🔺
EFFDFBC0	init	-	sleeping	0001	1.				
EFC7DC00	ueventd	-	sleeping	0359	857.				=
EFC82960	sh	-	sleeping	04F3	1267.				
EFC7F0A0	servicemanager	-	sleeping	04F4	1268.				
EFF769E0 🗄	vold	3.	sleeping	04F5	1269.	1288.	1316.		
EFC7F380 🗄	netd	7.	sleeping	04F6	1270.	1473.	1474.	1477.	147
EFF76140	debuggerd	-	sleeping	04F7	1271.				
EFCCC6C0 🗄	surfacetlinger	8.	running	04F8	1272.	1488.	1491.	1492.	149
EFC/9BE0 🗄	zygote	4.	sleeping	04F9	12/3.	1880.	1881.	1882.	
EFC/9060 🗄	drmserver	2.	sleeping	04FA	12/4.	1483.			
EFF42960 🕀	mediaserver	5.	sleeping	04FB	12/5.	1484.	1485.	1486.	154
EFF423A0	dbus-daemon	-	sleeping	04FC	12/6.				
EFF420C0	installo	-	sleeping	04FD	12//.				
EFD0F620	keystore	-	sleeping	04FE	12/8.				
	adbd	1	sleeping	0501	1400	1407	1/02	1/00	
	aubu	67	steeping	0502	1000	1517	1512	1514	1 5 1
	com android syst	10	currenc	0624	1572	1576	1577	1578	157
	android process	12	slooping	0633	1587	1501	1507	1593	150
ED425600	com android innu	10	sleeping	0641	1601	1607	1609	1610	161
EED19960	com android phon	21	sleening	0652	1618	1623	1624	1625	162
ED425320	com, android, laun	12.	sleeping	0650	1628.	1635.	1637.	1640.	164
FED8B6C0	com, android, smsn	10	sleeping	0680	1676	1686	1688	1689	169
ED5290A0 🕀	android.process.	15.	sleeping	0692	1682.	1691.	1692.	1694.	169
ED912900 🕀	com, android, desk	12.	sleeping	06BC	1724.	1726.	1728.	1730.	173
ED529940 🕀	com, android, prov	12.	sleeping	06D7	1751.	1753.	1754.	1756.	175
ECF36C60 🕀	com.android.exch	14.	sleeping	06EA	1770.	1774.	1775.	1776.	177 -
4									- F - 2





Debugging via GDB protocol is Run-mode debugging. This means only the selected process is stopped, while the kernel and all other processes are not influenced.

List of Linux processes									
name	id	space		sel	stop	1			
OMAP_UART3:	403.	403.	0x0193						
kpsmoused:	489.	489.	0x01E9						
irg/363-rtc0:	495.	495.	0x01EF						
kworker/u_2:	513.	513.	0x0201						
sieve:	545.	545.	0x0221		•	_			
gdbserver:	546.	546.	0x0222						
hello:	549.	549.	0x0225	V		-			
4									

Benefits

- Ideal for pure application process debugging
- Communication interfaces remain active (e.g. Ethernet, RS232)
- As a special feature, TRACE32 allows the user to concurrently use Run- and Stop-mode debugging.

Linux-aware Tracing

Trace technologies allow an in-depth analysis of the behavior and the timing characteristics of the embedded system. Core trace modules generate information on the instruction execution and running processes for their related core.

Off-chip parallel or serial trace ports allow a TRACE32 trace tool to record this information for Linux-aware trace analysis, providing effective troubleshooting, comprehensive profiling and quality assurance.



Hardware-based Products

Power Debug **Debugger via JTAG** Mainly Stop-mode debugging Combined Run- and Stop-mode debugging possible





Real-Time Trace Linux-aware single core and multicore tracing Stop-mode debugging only

Power Trace

Software-only Products

Benefit from the same user interface on all products.

Front-End

Debugger for Virtual Targets Mainly Stop-mode debugging Run-mode debugging possible





Debugger via GDB Protocol Run-mode debugging GDB and KGDB

Front-End

For more information visit: www.lauterbach.com/rtoslinux.html

