

Inhaltsverzeichnis

1	Einleitung	1
2	Hauptschleife	3
2.1	Seriellles EEPROM	3
2.2	Emulator	4
2.3	JTAG-Adapter	4
3	Betriebssysteme	5
3.1	Prozesse, Threads	5
3.2	Der Scheduler	6
3.3	Kernelspace, Userspace, Kernelmode, Usermode	6
3.4	Zusammenspiel mit der MMU	7
4	QNX	9
4.1	Microkernel	9
4.2	Resource-Manager	10
4.3	Komponentensystem	11
5	Linux	13
5.1	Linux-Build	15
5.2	Linux als Zielplattform	21
5.3	Kernel verkleinern	26
6	Startphase eines Systems	29
6.1	Allgemeine Betrachtungen	32
6.1.1	Ausführungsmodell einer CPU	32
6.1.2	Phase 1: Power-On, IPL	32
6.1.3	Phase 2: Bootloader, Ausführung des Startup Image	35
6.1.4	Phase 3: Start des OS	37
6.1.5	Booten: Zusammenfassung	37
6.2	Booten unter QNX	38
6.2.1	Beispiel QNX-IPL	38

6.2.2	Erkennen der Systemkomponenten und Konfiguration	39
6.2.3	Kernel	41
6.3	QNX Imagevarianten	42
6.3.1	IFS	42
6.3.2	Flash-Filesystem	43
6.3.3	EFS	44
6.3.4	ETFS	44
6.4	Beispiel-Build für ein BeagleBoard, QNX	44
6.4.1	BeagleBoard	44
6.4.2	Bootvorgang	45
6.4.3	Minimal-Buildfile	46
6.4.4	Beispiel-Konfigurationen	47
6.4.5	Formatieren einer ETFS-Partition	48
6.4.6	MMC partitionieren und formatieren	49
6.4.7	Flashen von X-Loader und U-Boot2	50
6.4.8	Flashen von IPL und IFS-Image	51
6.4.9	Logfile Startup X-Loader und U-Boot, QNX	51
6.4.10	Logfile Startup IPL/QNX	52
6.5	Booten unter Linux	53
7	Speichermodell für die Applikation	55
8	Reset und On/off	57
8.1	Reset bei On	57
8.2	Reset durch Watchdog	60
8.2.1	SW-Watchdog	61
8.2.2	Priorität des WD-Threads	61
8.2.3	Reset-Strategie	62
8.2.4	Welche Fehler-Situationen soll ein WD lösen	62
8.2.5	SW-DUMMY-Watchdog	63
8.2.6	Watchdog-Reset, Analysen	64
9	Umgang mit Flash-Memory	67
9.1	Flash-Probleme	67
9.2	Reclaim	71
9.2.1	Snippet eines Reclaimers (QNX)	71
9.3	Notfall-Persistenz	72
9.3.1	Persistenz restaurieren	72
10	HDD	75
10.1	HDD-Probleme	75
10.2	Fazit	76

11	Treiber	77
11.1	Systemaufrufe	78
11.2	Linux-Treiber	78
11.2.1	Treiber/Kernel-Modul	79
11.2.2	Build des hello-Beispiels	81
11.3	Geräte, Devices	83
11.3.1	Character-Devices	83
11.4	Kopierfunktionen zum Überwinden der Speicherkapselung	91
11.5	Mapping-Funktion zum Überwinden der Speicherkapselung	91
11.5.1	Warteschlange zum blockierenden Lesen	93
11.6	Treiber-Snippets für Linux	94
11.6.1	Beispiel eines einfachen Systemaufrufs	94
11.6.2	Beispiel einer einfachen chardev-Implementierung	96
11.6.3	procfs, Datenaustausch	100
11.7	QNX-Treiber mit Resource-Manager Implementierung	107
11.7.1	Registrieren des Resource-Managers	108
11.7.2	Implementieren der Dateioperationen	110
11.7.3	Füllen des Antwort-Buffers:	111
11.7.4	Der Rückgabewert	111
12	Interrupts	113
12.1	Interrupt-Latenz	114
12.1.1	Anforderungen an Interrupt-Service-Routinen (ISR)	115
12.1.2	Implementierungen der ISR	116
12.1.3	Nested-Interrupts	117
12.2	Shared-Interrupts	118
12.2.1	Flankensteuerung	118
12.2.2	Pegelsteuerung	120
12.2.3	Zusammenfassung	121
13	Interrupts unter Linux	123
13.1	Implementierungskonzepte	124
13.2	Interrupt-Ablauf unter Linux	125
13.3	SW-Snippets für Linux	126
13.3.1	Registrierung eines Interrupt-Handlers	126
13.3.2	Deregistrierung	127
13.3.3	Interrupt-Handler	127
13.3.4	Shared-Interrupt-Handler	127
13.3.5	Design-Regeln	128
13.3.6	Kritische Bereiche schützen	128
13.3.7	Ringpuffer oder Doppelpuffer	129
13.3.8	Interrupts sperren	130
13.3.9	Einzelne Interrupts sperren/maskieren	131
13.3.10	Spinlocks	132

13.4	Bottom-Half Implementierungen	134
13.4.1	SoftIrq	134
13.4.2	Tasklets	135
13.4.3	Work-Queues	136
13.4.4	Auftrag schedulen (beauftragen)	137
13.4.5	Threaded-Interrupts	138
13.4.6	Welche Bottom-Half	139
14	Interrupts unter QNX	141
14.1	Die ISR unter QNX	141
14.2	SW-Snippets für QNX	143
14.2.1	Registrieren der ISR	143
14.2.2	Beispiel eines Interrupt-Handlers	145
14.2.3	InterruptAttachEvent	146
14.2.4	Deregistrieren	147
15	MultiCore-Systeme	149
15.1	Embedded MultiCore-Systeme	149
15.2	AMP – Asymmetric Multiprocessing	150
15.3	SMP – Symmetric Multiprocessing	151
15.4	BMP – Bound Multiprocessing	151
15.4.1	BMP mit CPU-Affinities: Linux	152
15.4.2	BMP mit Interrupt-Affinities: Linux	152
15.4.3	BMP mit CPU-Affinities: QNX	153
15.5	MultiCore-Scheduling	154
15.5.1	Memory-Hierarchien, exklusiver und gemeinsamer Speicher	154
15.5.2	Memory – Konsistenz und Kohärenz	155
15.5.3	Neue Möglichkeiten mit BMP	156
16	Virtuelle Maschinen	157
16.1	Kategorien virtueller Maschinen	159
16.2	Virtualisierungsvarianten	159
16.2.1	Application-Virtualisierung	159
16.2.2	Para-Virtualisierung	160
16.2.3	Binary-Translation	161
16.2.4	OS-Level-Virtualisierung	162
16.2.5	Speicher-Virtualisierung	162
16.2.6	Ressourcen-Virtualisierung	163
16.3	CPU mit Virtualisierungs-Unterstützung	164
16.4	Sicherheit von Virtualisierungs-Lösungen	165
17	Zusammenspiel zwischen MultiCore-Konzept und virtuellen Maschinen	167

18 HMI	169
18.1 Einführung	169
18.2 HMI-Entwicklung	169
18.3 Mensch-Maschine-Schnittstelle	170
18.4 Aktuelle Trends in der HMI-Entwicklung	171
18.4.1 OpenGL	171
18.4.2 Adobe Flash	171
18.4.3 HTML (HTML5), JavaScript, CSS	172
18.4.4 Qt	172
18.5 Einsatz von Grafik-Paketen in embedded Systemen	172
18.5.1 Variante 1, Extra-Thread	173
18.5.2 Variante 2, HMI-System	174
18.5.3 Variante 3, zwei FSMs	175
18.5.4 Variante 4, Remote-Display	176
18.5.5 Variante 5, Terminal-Mode	176
18.6 HMI in OpenGL, Snippets	176
18.6.1 Variante 1 in OpenGL	177
18.6.2 Variante 2 in OpenGL	181
18.6.3 Variante 3 in OpenGL	184
18.6.4 Variante 4 in OpenGL	184
18.6.5 Variante 5, Nokia-Terminal-Mode	192
18.7 HMI in Qt, Snippets	194
18.7.1 Qt-Features	195
18.7.2 Signale und Slots	196
18.7.3 Ereignisse (Events) in Qt	197
18.7.4 UI-Entwicklung	198
18.7.5 Anbindung einer GUI an eigene Applikationen	198
18.7.6 Variante 1 in Qt	199
18.7.7 Variante 2 in Qt	201
18.7.8 Variante 3 in Qt	205
19 Java für Embedded Systeme	207
19.1 Grafik	207
19.1.1 Performance-Überlegungen	207
19.1.2 Garbage-Collection	208
19.2 Java-HMI Anbindung	209
19.2.1 Sockets	210
19.2.2 Pipes	210
19.2.3 Marshalling für serielle IPC	210
19.2.4 Shared-Memory	211
19.2.5 JNI-Anbindung	212
19.2.6 Performance-Tipps	218
20 Einfaches Multimedia-Framework, Linux	231

21 Fehler	237
21.1 Gründe für Fehler	237
21.2 Fehlerarten	238
21.2.1 Einfache Applikationsfehler	238
21.2.2 Teure Applikationsfehler	238
21.2.3 System-Fehler	239
21.3 Debugging	244
21.3.1 Remote-Debugging	244
21.4 Weitere Tools und Methoden zur Fehlersuche/Findung	246
21.4.1 Banale aber wirksame Debugging-Methoden	246
21.4.2 Reviews	247
21.4.3 Trace-Client, Trace-Server	247
21.4.4 Performance-Tools, Profiler	248
21.4.5 pidin (QNX)	249
21.4.6 malloc-Library	249
21.4.7 Library-Interposer	252
21.4.8 Post-mortem-Analyse, Core-Dump	253
21.4.9 Kernel-Traces in QNX	257
21.5 Kernel-Traces in Linux	269
21.5.1 LTT	269
21.6 Eigene Tools (QNX)	274
21.6.1 CyclicCheck	274
21.6.2 Speichercheck	275
21.6.3 Speicherversteck (QNX)	276
21.6.4 Speicherversteck (Linux)	277
21.6.5 CPU-Last	277
21.6.6 IP-Scanner	279
21.6.7 Proc-Tracker	280
21.6.8 Snoopy (QNX)	280
21.6.9 Script-Launcher	281
21.7 Neue Probleme bei MultiCore	281
22 Anhang	283
22.1 /procfs-Beispiel	283
22.2 Producer-Consumer-Beispiel mit Semaphore	285
22.3 tint-Beispiel mit procfs	290
22.4 tint-Beispiel mit chardev und Nutzung von cdev	294
22.5 Treiber mit Threaded-Interrupt und Speichermapping	298
22.5.1 User-Prozess	298
22.5.2 Kernel-Prozess	299
22.6 QNX Treiber (ISR + Resource-Manager)	302
22.7 QNX Treiber (ISR + Resource-Manager) für BeagleBoard	307
22.8 Links	314
22.8.1 QNX	314
22.8.2 BeagleBoard	315

22.8.3	Java	315
22.8.4	GStreamer	316
22.8.5	Linux	316
22.8.6	Arbeiten am ICM, h_da	316
Literatur		317
Sachverzeichnis		319

Embedded Technologies

Vom Treiber bis zur Grafik-Anbindung

Wietzke, J.

2012, XXVI, 321 S. 114 Abb., 20 Abb. in Farbe.,

Hardcover

ISBN: 978-3-642-23995-3