
Inhaltsverzeichnis

1	Einleitung	1
----------	-------------------------	----------

Teil I Bausteine eines Embedded Frameworks

2	Ein Infotainment-System	7
2.1	Die Headunit	8
2.2	Geräte/Devices im System	9
2.3	Begriffsklärung: Was ist ein Framework	10
2.4	Fokus unserer Framework-Betrachtungen	11
3	Anforderungen an ein Embedded Automotive Framework	13
3.1	Verteilte Entwicklung	13
3.2	MVC, Organisationen	13
3.3	Speicheranforderungen	15
3.4	Startzeiten	15
3.5	Menüwechselzeiten	16
3.6	Konfiguration versus Dynamik	16
3.7	Datenfluss/Kontrollfluss/Zustandssteuerung	16
4	Betriebssysteme	17
4.1	Prozesse, Tasks, Threads	18
4.2	Der Scheduler	19
4.2.1	Präemptive Prioritätssteuerung	19
4.2.2	Round-Robin	19
4.2.3	Prioritäts-Inversion, Prioritäts-Vererbung	19
4.2.4	Task-Switch, Kontextwechsel	20
4.3	Zusammenspiel mit der MMU	20

5	Design-Rules und Konventionen	21
5.1	Namenskonventionen	22
5.2	Shared-Verzeichnisse, Libraries	22
5.3	Eigene Definitionen im Shared	22
5.3.1	Standard-Typen	22
5.3.2	Alignment	23
5.3.3	Little-Endian, Big-Endian	24
5.3.4	Zahlengrenzen	25
5.3.5	Const Definitionen	25
5.3.6	Eigene Assertions	26
5.3.7	Exceptions und Signale	27
5.3.8	Eigene Ausgaben	27
6	Speicherverwaltung	29
6.1	Statische Speicherverwaltung	31
6.2	Dynamische Speicherverwaltung	31
6.3	Start-Up-Strategien	33
6.4	Speicherbedarf und kleine Pittfalls	33
6.4.1	Welche Variable wo?	33
6.4.2	Alternative Strategie 1: Späte Konstruktion	36
6.4.3	Alternative Strategie 2: Späte Initialisierung	38
6.4.4	Alternative Strategie 3: Lokale statische Variablen	39
6.4.5	Speicherort von Konstanten	40
6.4.6	Speicherbedarf eines einfachen Objekts	43
6.4.7	Speicherbedarf eines einfachen Objekts mit virtueller Methode	44
6.4.8	Speicherbedarf eines Objekts einer abgeleiteten einfachen Klasse	46
6.4.9	Speicherbedarf eines Objekts einer abgeleiteten Klasse mit virtueller Methode	48
6.4.10	Speicherbedarf eines Objekts einer abgeleiteten Klasse mit virtueller Basis-Methode	50
6.4.11	Mehrfachvererbung und Pittfall	52
6.4.12	Speicherbedarf dynamischer Variablen	55
6.4.13	Schlechte Parameterübergabe	56
6.4.14	Probleme mit dem Kopierkonstruktor	57
6.4.15	Probleme mit dem Zuweisungsoperator	57
6.4.16	Schlechte Rückgaben	57
6.4.17	Padding-Probleme	59
6.4.18	Alignment-Probleme, Endian-Probleme	60
6.4.19	Speicherfragmentierung	60
6.4.20	Vektoren, STL	61

7	Embedded Speicherkonzepte, spezielle Muster	63
7.1	Statische Variablen	63
7.1.1	Reihenfolge der Konstruktoren	63
7.2	Quasi-statische Allokation aus festem Pufferspeicher	66
7.2.1	Anlegen eines statischen Puffers	67
7.2.2	Placement-new	67
7.2.3	Makros zur Allokation	68
7.3	Allokator	70
7.3.1	Statische Allokation	73
7.3.2	Dynamische Allokation mit temporärem Heap	75
7.3.3	Object-Pool-Allokation	78
7.4	Datencontainer	82
7.5	Gemeinsam benutzte Objekte (Shared-Objects)	82
7.6	Referenzzähler (reference counting)	83
7.7	Garbage-Collection	85
7.8	Die Captain-Oates Strategie	85
7.9	Speicherlimit	86
7.10	Partieller Fehler (Partial Failure)	87
7.11	Fazit	87
8	Prozesse und POSIX-Threads	89
8.1	Prozesse	90
8.1.1	Der Lebenszyklus eines Prozesses	90
8.1.2	Prozesszustände	91
8.1.3	Zombies	92
8.1.4	Bestandteile	92
8.1.5	Prozess mit fork erzeugen	93
8.1.6	Prozess mit exec erzeugen	94
8.1.7	Prozess mit spawn erzeugen	96
8.1.8	Prozessvererbung	97
8.1.9	Prozessattribute	97
8.1.10	Über Prozessgrenzen hinweg	98
8.2	Threads	99
8.2.1	Erzeugung von POSIX-Threads	99
8.2.2	Allgemeine Gestaltung eines Threads in eingebetteten Anwendungen	100
8.2.3	Thread-Attribute	101
8.2.4	Über Threadgrenzen hinweg	104
8.2.5	Thread-spezifische Daten	106
8.2.6	Thread-Träger und Thread-Objekte	108
8.3	Designentscheidung	118

9	Inter-Prozess-Kommunikationskanäle, IPC	121
9.1	Pipe	121
9.2	FIFO, Named-Pipe	125
9.3	Socket, local	125
9.4	Socket-Verbindung zwischen Prozessoren	127
9.5	Message-Queues	134
9.5.1	System V Message-Queues	134
9.5.2	POSIX Message-Queues	134
9.6	QNX-Message	137
9.7	Shared-Memory	137
9.8	Shared-Memory gegen Überschreiben schützen	144
9.9	Zeitverhalten	149
9.10	Designentscheidung und prinzipielle Implementierung	150
10	Gemeinsame Nutzung von Code	151
10.1	DLLs	151
10.2	Shared-Code	151
10.3	Designentscheidung und prinzipielle Implementierung	152
11	Synchronisierungsmechanismen für Prozesse und Threads	153
11.1	Semaphore	153
11.2	Unbenannter Semaphor	157
11.3	Benannter Semaphor	160
11.4	Mutex, rekursiver Mutex	161
11.5	Mutex-Guard	167
11.6	Signale	168
11.7	Bedingungsvariablen (Condition Variables)	168
11.8	Zeitverhalten	177
11.9	Deadlock, Livelock und Prioritätsinversion	177
11.10	Zusammenfassung	182
12	Kommunikation per Events	183
12.1	Wichtige Event-Typen	184
12.1.1	Signal-Events	184
12.1.2	Events mit Cookies	185
12.1.3	Call-Events	185
12.1.4	Timer-Events	185
12.1.5	Change-Events (Notifikation - Das Hollywood-Prinzip)	186
12.2	Realisierungen von Events	186
12.2.1	Einfache Events	186
12.2.2	C-Strukturen	187
12.2.3	Funktionszeiger	189
12.2.4	Event-Klassen und Objekte	190

	12.2.5 Zeiger auf Event-Objekte verschicken	192
	12.2.6 Events am Beispiel von MOST	195
	12.3 Designentscheidung und prinzipielle Implementierung	196
13	Event-Verarbeitung	197
13.1	Queues	197
	13.1.1 Externe Queue	199
	13.1.2 Interne Queue	204
	13.1.3 System-Queue	205
	13.1.4 Queue mit verbesserter Inversions-Eigenschaft	205
13.2	Dispatcher	211
	13.2.1 Der System-Dispatcher	211
	13.2.2 Der lokale Dispatcher	212
	13.2.3 Signalisation im Dispatcher	213
	13.2.4 Registrierung im Dispatcher	213
13.3	Synchrone Events	214
14	Zustandsautomaten	215
14.1	Motivation	215
14.2	Kommunikation	216
14.3	Automaten	217
14.4	Statechart-Elemente	217
14.5	Probleme mit Statecharts und Ersatzlösungen	221
14.6	Implementierung dynamischen Verhaltens	223
	14.6.1 Definition der Events	224
	14.6.2 Implementierung der Aktionen	225
14.7	Implementierung der Statecharts	226
	14.7.1 Implementierung durch Aufzählung für Zustände und Unterzustände	226
	14.7.2 Zustandsmuster (State-Pattern)	231
	14.7.3 Verbesserte Aufzählungsmethode	240
14.8	Implementierung nach Samek	246
15	Externer Kommunikationskanal: MOST-Bus	253
15.1	Der synchrone Kanal	254
15.2	Der asynchrone Kanal	255
15.3	Der Kontrollkanal	255
15.4	Geräte/Devices im Framework	255
	15.4.1 Logische Geräte, Modelle	257
	15.4.2 Methoden	258
	15.4.3 Properties	258
	15.4.4 MOST-Adressierung	259
	15.4.5 Adressierungsbeispiele	265
	15.4.6 MOST-Events, prinzipielle Dekodierung des Headers	266

15.4.7	MOST-Events, prinzipielle Dekodierung der Daten	268
15.4.8	MOST-Konstanten	269
15.4.9	MOST-Probleme	271

Teil II Das Framework

16	Das Framework	275
17	OS-Grundmechanismen	277
18	Komponentenarchitektur	279
18.1	Event-Handler und Event-Formate	281
18.2	Implementierung von Schnittstellen, Proxy und Handler	285
18.3	Komponentenarchitektur	287
18.3.1	Komponentenkontext und Daten im Shared-Memory	288
18.3.2	Erzeugung der Kontexte	295
18.3.3	Main-/Admin-Task	300
18.3.4	Signale, Mechanismen über den Kontext	302
18.3.5	Komponenten-Dispatcher	308
18.3.6	Softtimer und Timer-Manager	310
18.3.7	Registrierung	318
18.3.8	Die Basisklasse <code>AComponentBase</code> und das Zusammenspiel mit der Umgebung	319
18.4	Zusammenfassung	327
19	Main-Dispatcher-Komponente	331
19.1	Dispatcher-Receiver	331
19.2	Dispatcher-Main-Thread	336
20	Modell-Komponenten, logische Geräte	341
20.1	Aufgaben des logischen Geräts	342
20.2	Was erwartet die HMI von einem logischen Gerät?	343
20.3	Zustände des Geräts	345
20.4	Gültigkeit der Daten	346
20.5	Kommunikationsanforderung	347
20.6	Struktur eines logischen Geräts	347
20.7	Datencontainer versus Event-Prinzip	347
20.8	Architektur eines Datencontainers	349
20.8.1	Struktur des Datencontainers	350
20.8.2	Lesezugriffe der HMI	356
20.8.3	Versenden eines HMI-Befehls	357
20.8.4	Menge der erlaubten High-Level-Aktionen	358
20.8.5	Abstrakte Klasse <code>ADataContainerAccessor</code>	359
20.9	MOST-Codec	363

20.9.1	Vorbereitung	363
20.9.2	Einfache Implementierung eines MOST-Decoders	367
20.9.3	Objektorientierter Ansatz	370
20.9.4	Objekte zur Kompilierzeit	372
20.9.5	Konstante Objekte zur Dekodierung der MOST-Nachrichten	374
20.10	Zusammenfassung	387
21	HMI-Komponente	389
22	Persistenz-Controller und On/Off-Konzepte	391
23	Codegenerierung	395
23.1	Erstellen der XML-Eingabe-Dateien	395
23.2	Erzeugen des Quellcodes	397
23.3	Übersicht der erzeugten Dateien	398
23.4	Übersetzen des Quellcodes	404
24	Sonstige Aspekte	405
24.1	Internes non-MOST-Device, Beispiel Mediaplayer	405
24.2	Internes MOST-Device, Beispiel Tastatur	405
24.3	Treiber im Framework	406
24.4	Einfaches Debugging-Konzept	406
24.5	Überlastverhalten des Systems	408
24.6	Anmerkungen zur Komplexität und zum Testing	409
Anhang		411
25.1	Timer und Zeitmessung	412
25.1.1	Sleep	413
25.1.2	Timeout	414
25.2	Socket	414
25.2.1	InetAddr	414
25.2.2	CSockAcceptor	415
25.2.3	CSockConnector	418
25.2.4	CSockStream	420
25.3	Keyboard	422
25.4	Shared-Memory	424
25.5	CBinarySemaphore	432
25.6	Extern const	439
Literatur		441
Index		443

Automotive Embedded Systeme
Effizientes Framework - Vom Design zur
Implementierung

Wietzke, J.; Tran, M.T.

2005, XXIII, 445 S., Hardcover

ISBN: 978-3-540-24339-7