Embedded Systems / Eingebettete Systeme

BSc-Studiengang Informatik
Campus Minden

Matthias König

FH Bielefeld
University of
Applied Sciences

Beispiel einer Anwendung: Blumentopf

- Intelligenter Blumentopf, hier: "Click and Grow"

  - Feuchtigkeitsmesser

  - Pumpe

  - Chip mit Pflanzenklasse

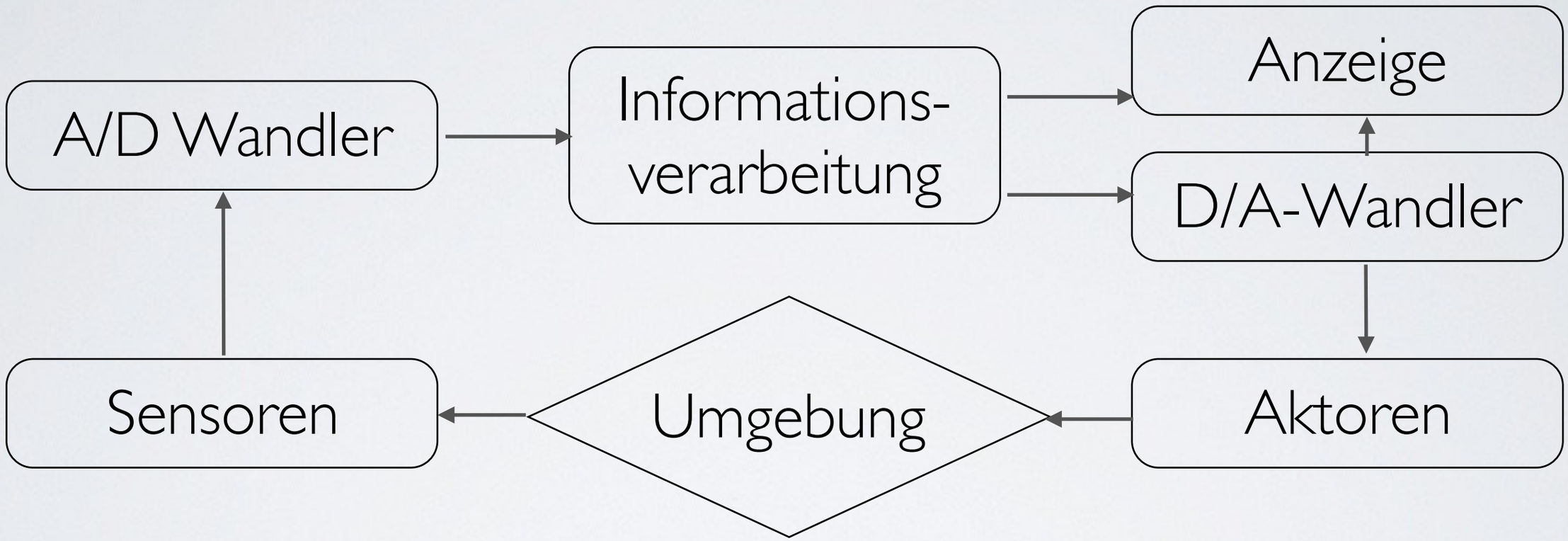
  - LED-Anzeige

  - Mikrocontroller

[Quelle:http://cdn.shopify.com/s/files/1/0156/0137/products/chili-pepper-smartpot_1024x1024.jpg?v=1375114838]

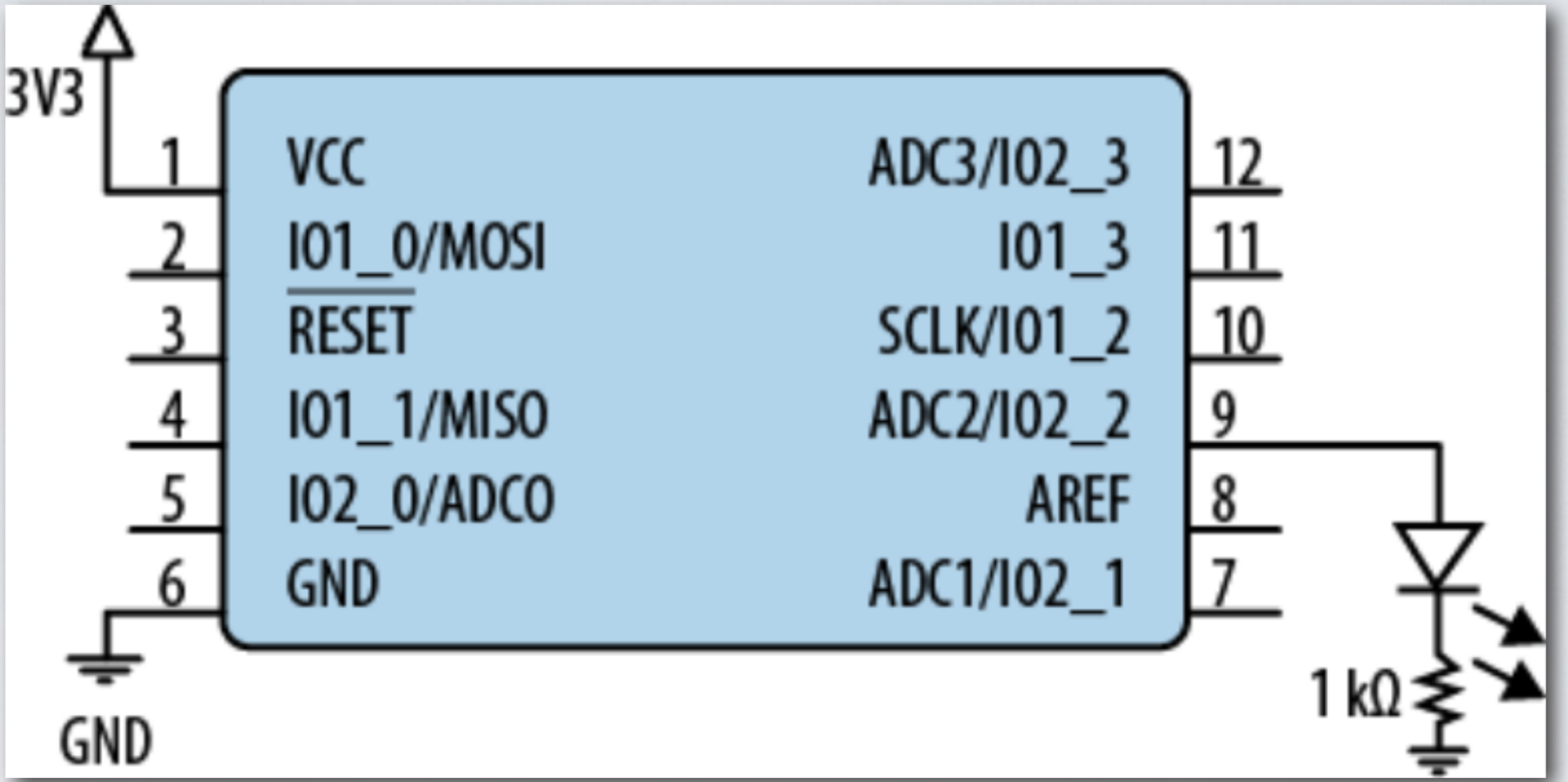
WIEDERHOLUNG

Hardware in a loop



[Quelle: Marwedel, Eingebettete Systeme]

Beispiel: Einschalten der LED



[Quelle: White, Making Embedded Systems]

```
P2DIR |= (1 << 2); // set to output

P2OUT |= (1 << 2); // turn on
```

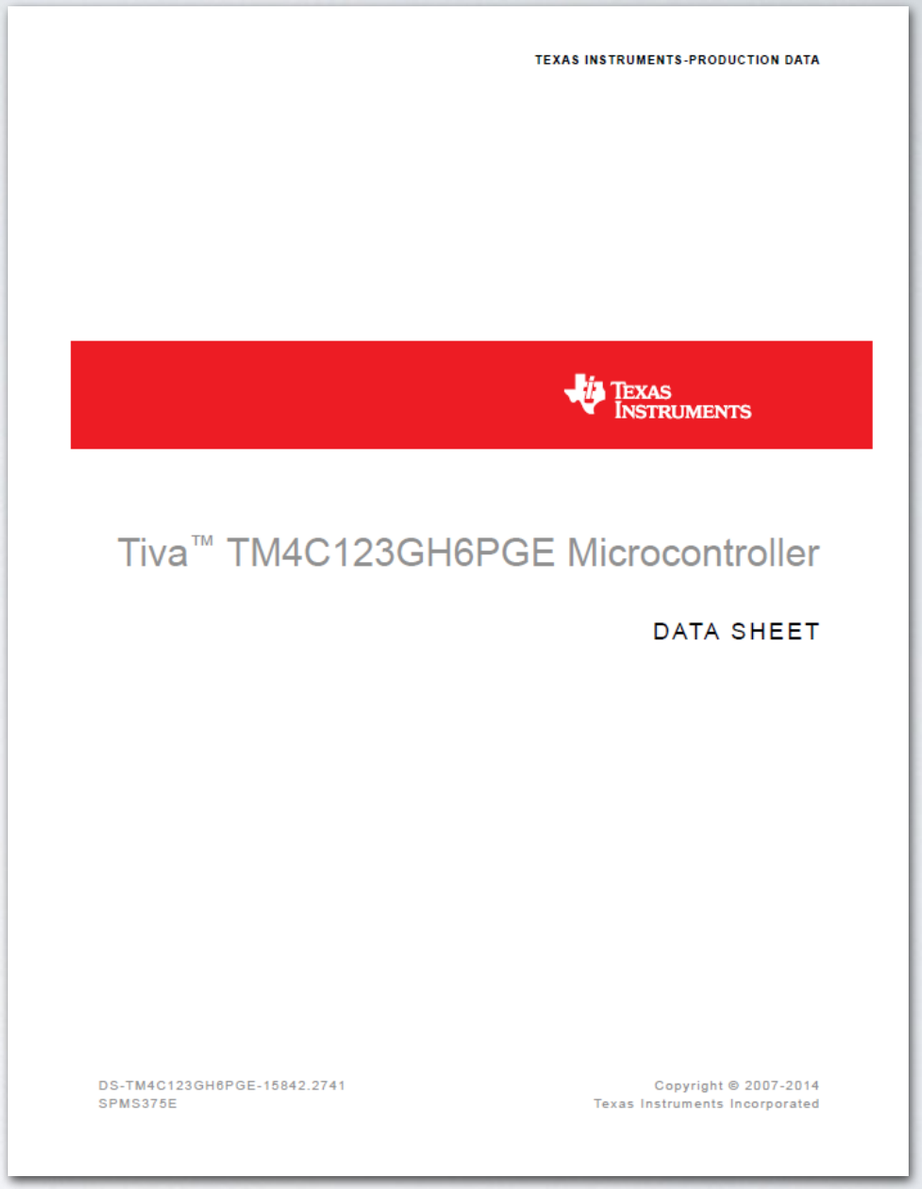
MIKROCONTROLLER-PROGRAMMIERUNG

AM BEISPIEL DES TM4C123GH6PM

Am Beispiel des Mikrocontrollers Cortex-M4F

- Lesen eines Datenblatts
  - Aufbau des Mikrocontrollers
- Programmierung des Mikrocontrollers
  - Assembler
  - C

in aller Kürze


TEXAS INSTRUMENTS-PRODUCTION DATA

Texas Instruments

Tiva™ TM4C123GH6PGE Microcontroller

DATA SHEET

DS-TM4C123GH6PGE-15842.2741
SPMS375E

Copyright © 2007-2014
Texas Instruments Incorporated

Datenblatt: 1409 Seiten

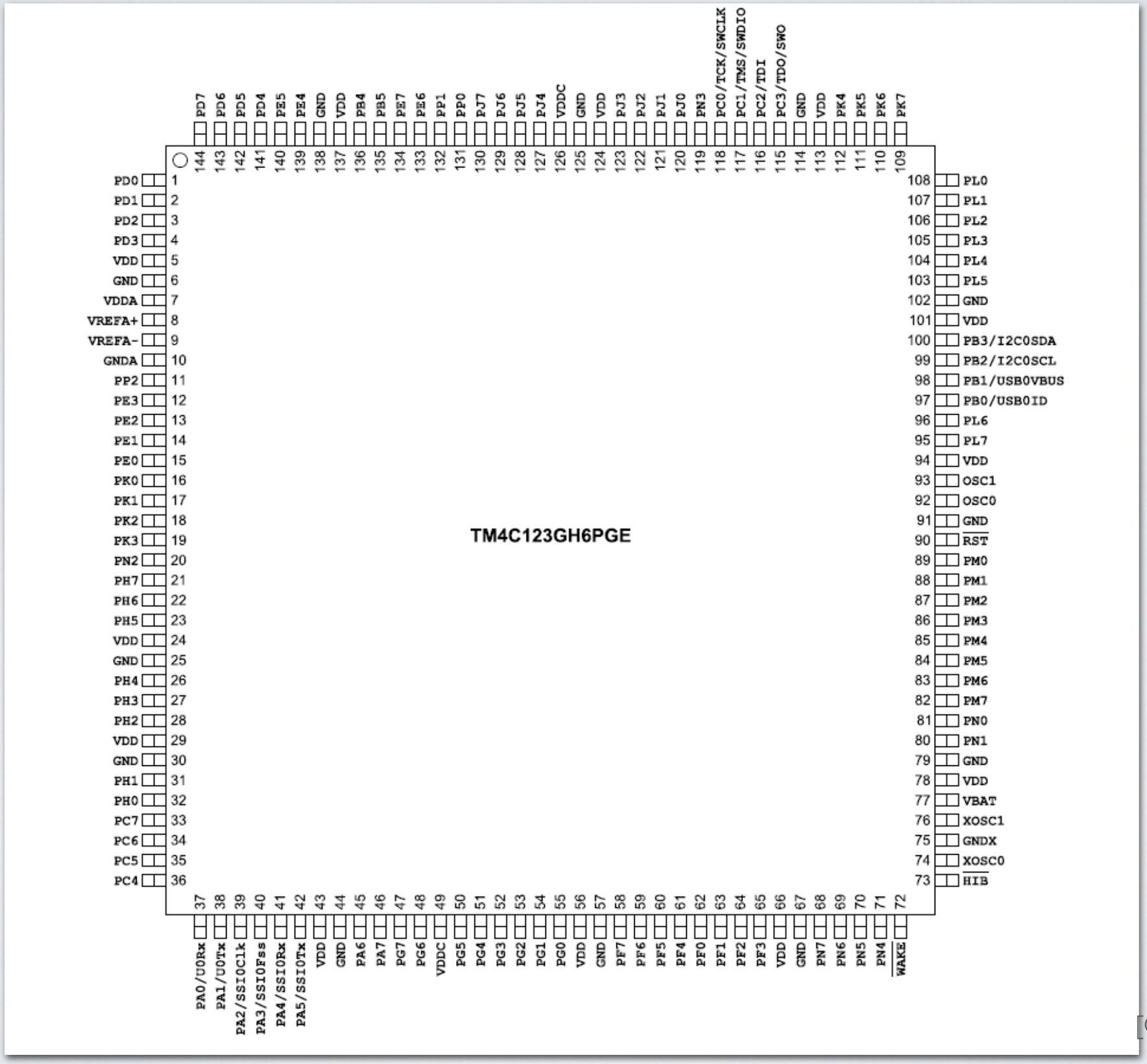
[Quelle: spms376e.pdf]

Datenblatt TM4C123G: Inhalt

1. Architectural Overview
2. The Cortex-M4F Processor
3. Cortex-M4 Peripherals
4. JTAG Interface
5. System Control
6. System Exception Module
7. Hibernation Module
8. Internal Memory
9. Micro Direct Memory Access (µDMA)
10. General-Purpose Input/ Outputs (GPIOs)
11. General-Purpose Timers
12. Watchdog Timers
13. Analog-to-Digital Converter (ADC)
14. Universal Asynchronous Receivers/Transmitters (UARTs)
15. Synchronous Serial Interface (SSI)
16. Inter-Integrated Circuit (I2C) Interface
17. Controller Area Network (CAN) Module
18. Universal Serial Bus (USB) Controller
19. Analog Comparators
20. Pulse Width Modulator (PWM)
21. Quadrature Encoder Interface (QEI)
22. Pin Diagram
23. Signal Tables
24. Electrical Characteristics

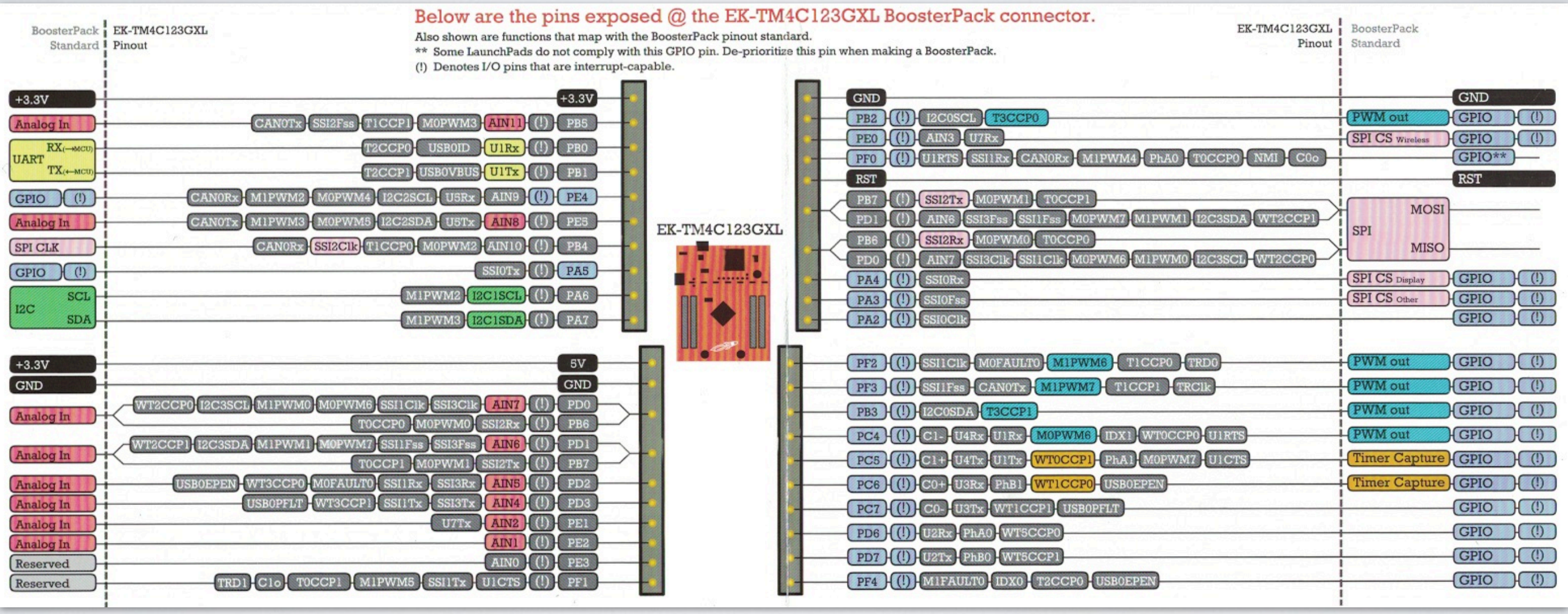
[Quelle: spms376e.pdf]

Pin Configurations



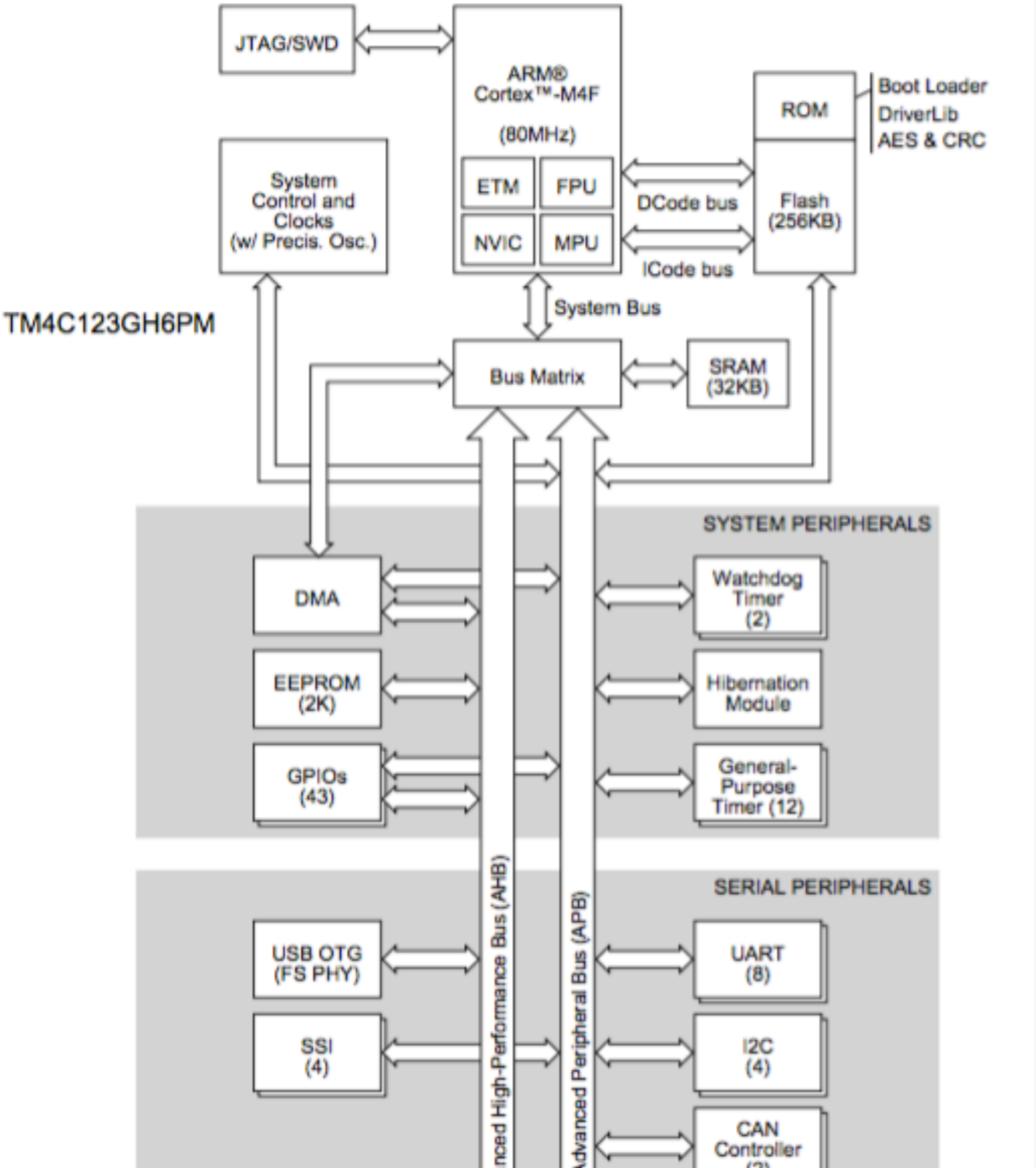
Datenblatt
Seite 1328
[Quelle: spms376e.pdf]

Pin Configurations



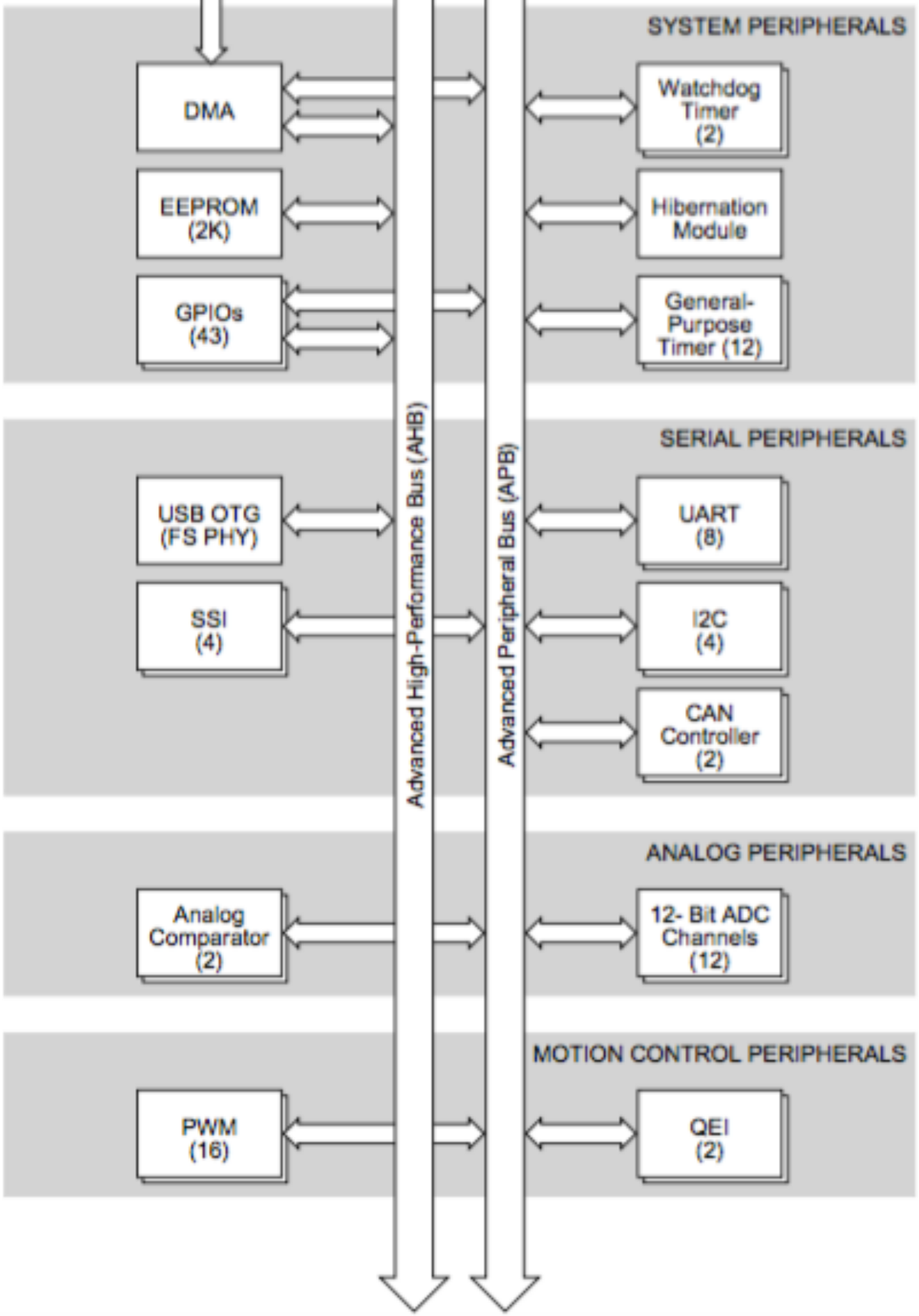
Quelle: Beilage zu LaunchPad Evaluation Kit
Part Number EK-TM4C123GXL

Blockdiagramm eines Cortex-M4



Datenblatt
Seite 48
[Quelle: spms376e.pdf]

Blockdiagramm eines Cortex-M4



Datenblatt
Seite 48
[Quelle: spms376e.pdf]

Speichergröße

1.3.2 On-Chip Memory

The TM4C123GH6PGE microcontroller is integrated with the following set of on-chip memory and features:

- 32 KB single-cycle SRAM
- 256 KB Flash memory
- 2KB EEPROM
- Internal ROM loaded with TivaWare™ for C Series software:
  - TivaWare™ Peripheral Driver Library
  - TivaWare Boot Loader
  - Advanced Encryption Standard (AES) cryptography tables
  - Cyclic Redundancy Check (CRC) error detection functionality

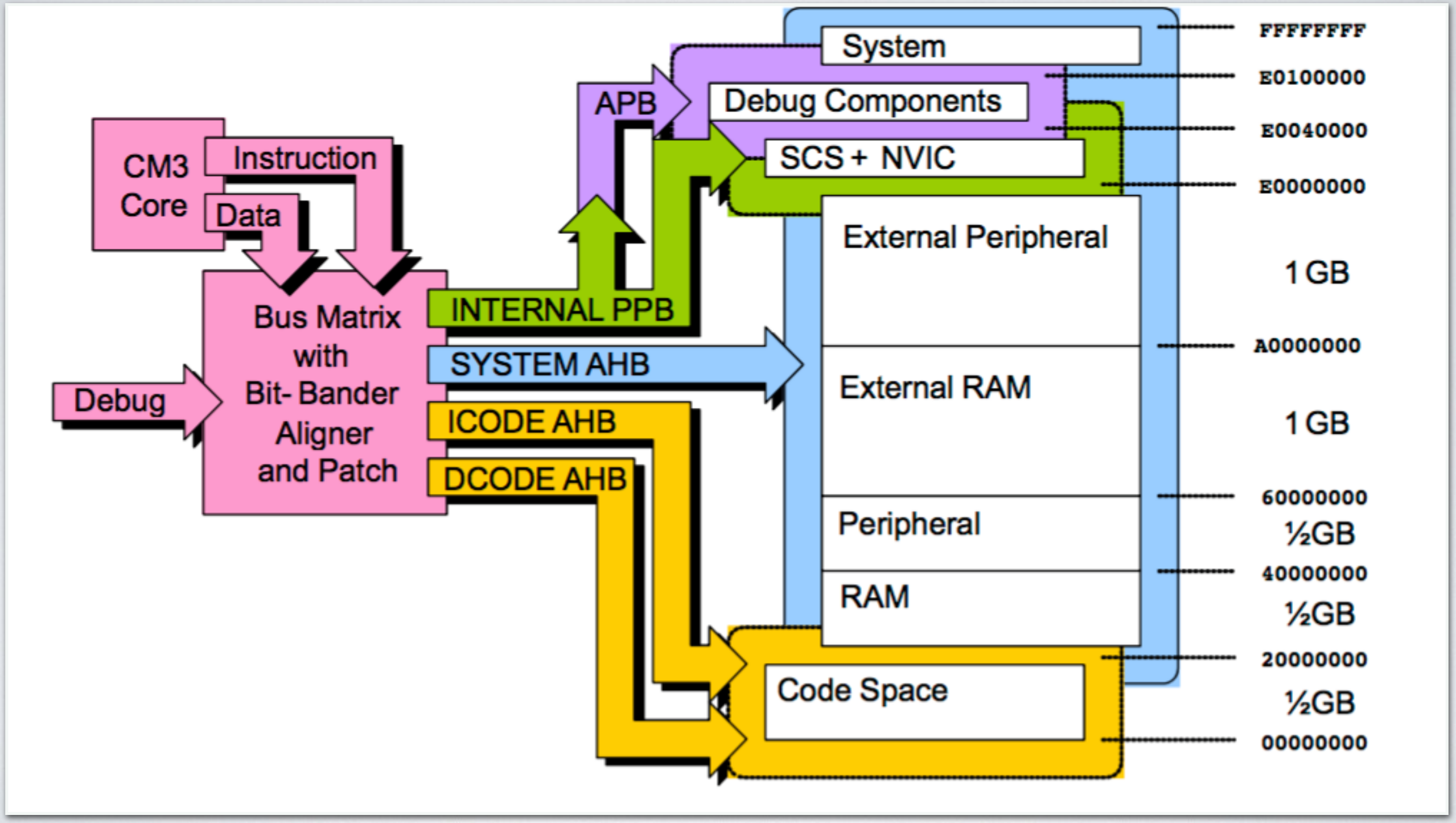
Datenblatt
Seite 51
[Quelle: spms376e.pdf]

General Purpose IO

- Zusammengesetzt aus 6 physikalischen GPIO-Blöcken (Port A bis F)
- Bis zu 43 GPIOs (Konfigurationsabhängig)
- Ports A bis F werden vom Advanced Peripheral Bus (APB) erreicht
- Besondere GPIO Pins: UART0, SSI0, I2C0, JTAG/SWD

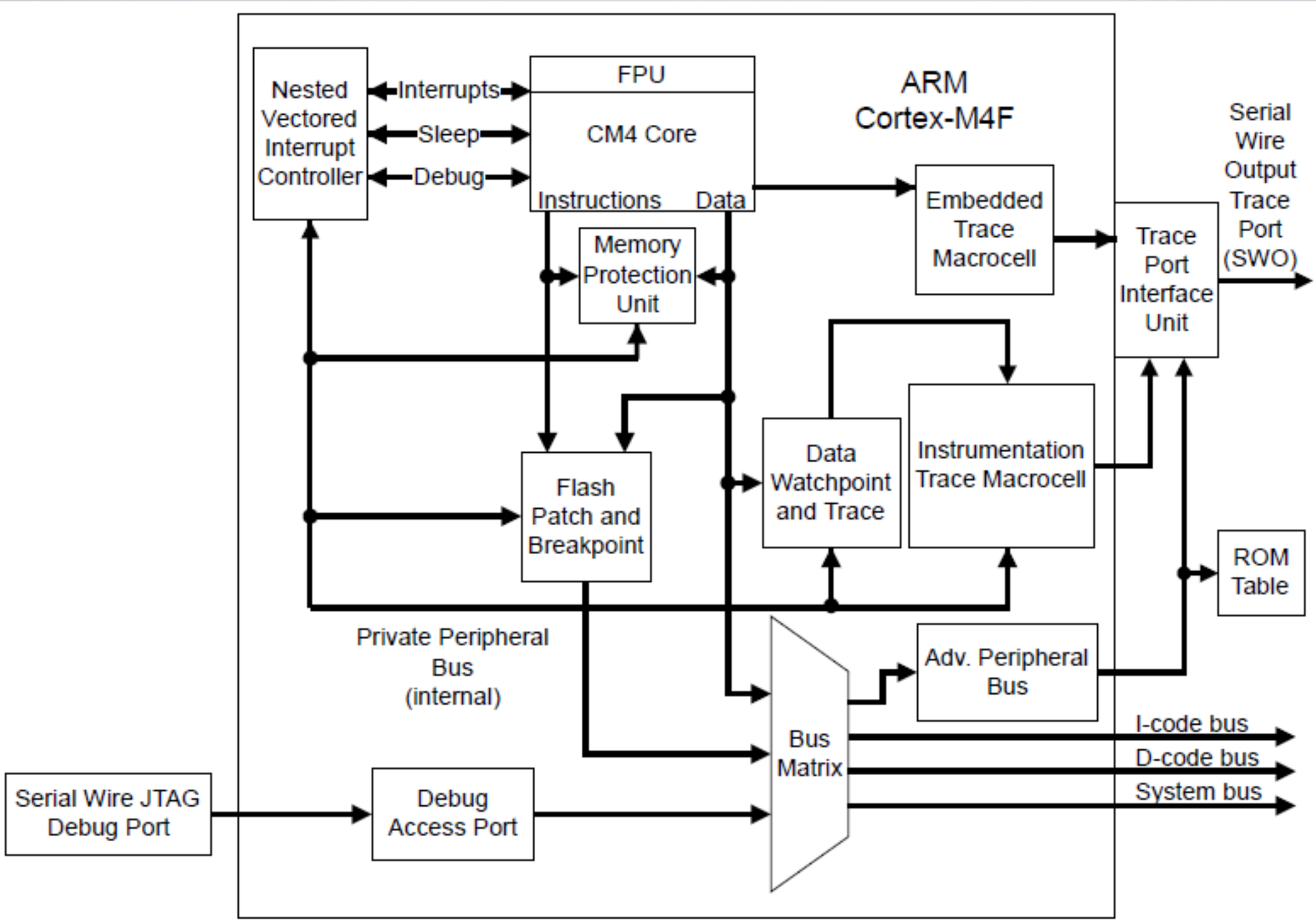
Datenblatt
Seite 77, 649
[Quelle: spms376e.pdf]

Memory Map



[Quelle: https://www.arm.com/files/pdf/CortexM3_Uni_Intro.pdf, Zugriff 05.04.16]

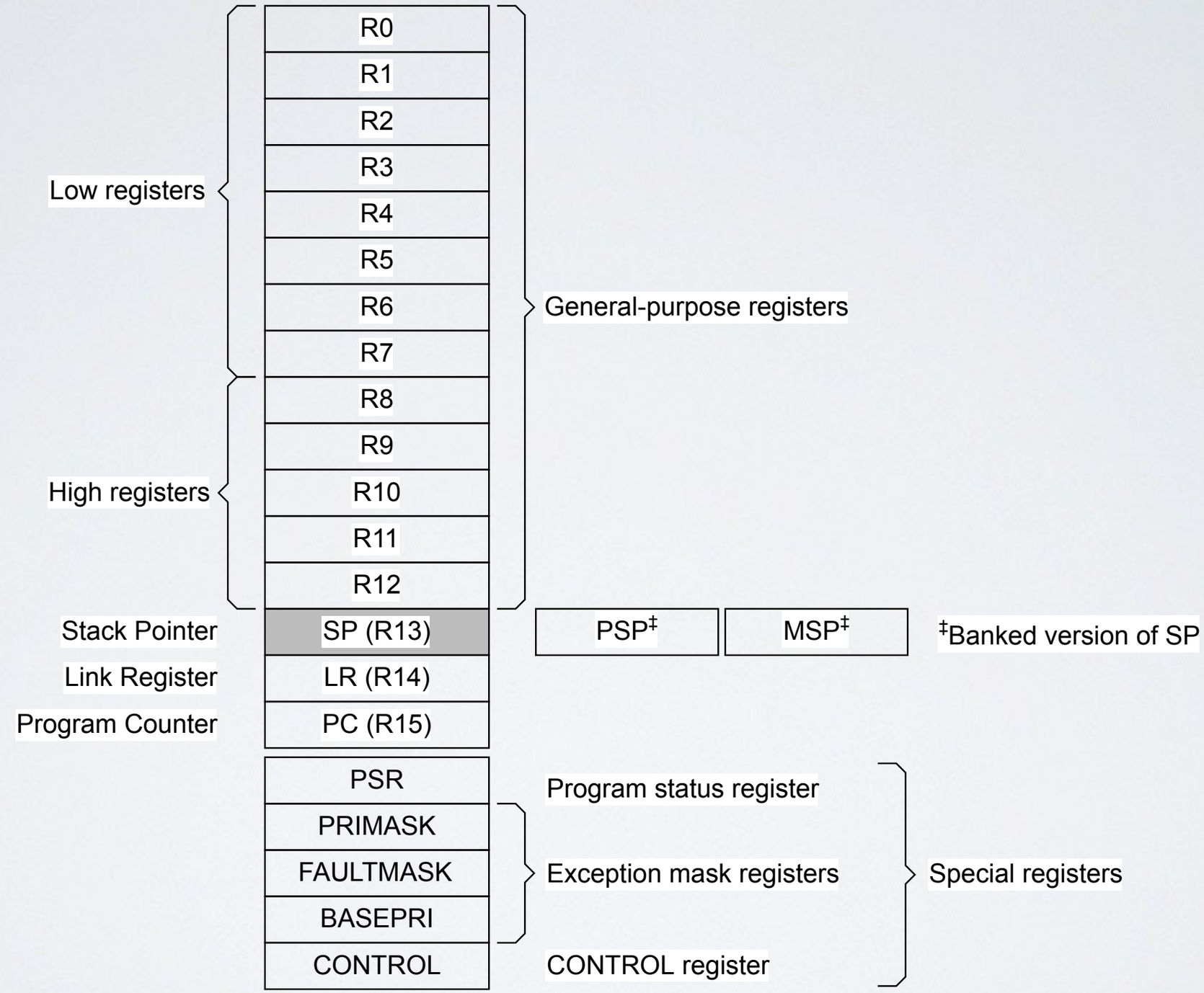
Prozessorkern



Datenblatt
Seite 71
[Quelle: spms376e.pdf]

Register

Figure 2-3. Cortex-M4F Register Set



Datenblatt
Seite 76
[Quelle: spms376e.pdf]

“Wichtigste” Register

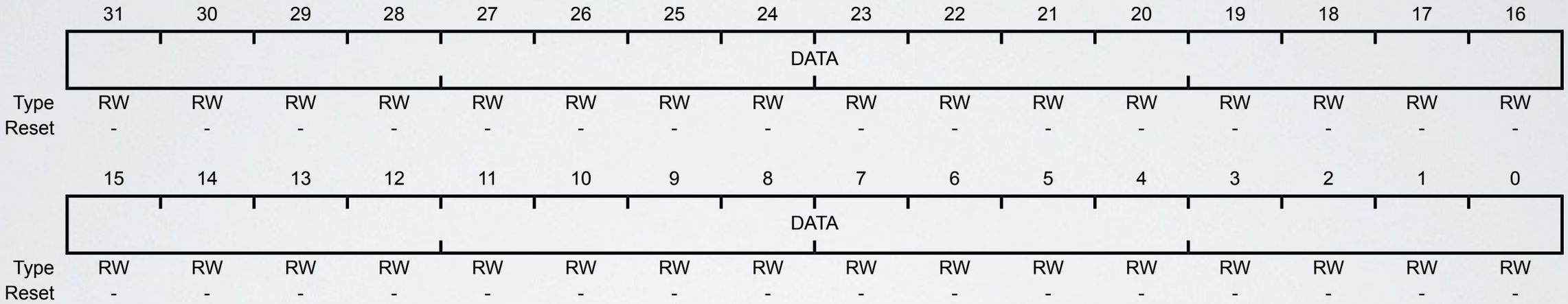
- Program Counter PC (Seite 80)
- Program Status Register PSR (Seite 81)
- Cortex General-Purpose Register R0 bis R12 (Seite 77)
- Stack Pointer SP (Seite 78)
- Link Register LR (Seite 79)
- Control Register (Seite 88)
- Watchdog Register (Seite 777)

Datenblatt
Seiten 75-76
[Quelle: spms376e.pdf]

General Purpose Register

Cortex General-Purpose Register 0 (R0)

Type RW, reset -



Bit/Field	Name	Type	Reset	Description
31:0	DATA	RW	-	Register data.

Datenblatt
Seite 77
[Quelle: spms376e.pdf]

Stack Pointer

Register 14: Stack Pointer (SP)

The **Stack Pointer (SP)** is register R13. In Thread mode, the function of this register changes depending on the `ASP` bit in the **Control Register (CONTROL)** register. When the `ASP` bit is clear, this register is the **Main Stack Pointer (MSP)**. When the `ASP` bit is set, this register is the **Process Stack Pointer (PSP)**. On reset, the `ASP` bit is clear, and the processor loads the **MSP** with the value from address 0x0000.0000. The **MSP** can only be accessed in privileged mode; the **PSP** can be accessed in either privileged or unprivileged mode.

Stack Pointer (SP)

Type RW, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SP															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SP															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:0	SP	RW	-	This field is the address of the stack pointer.

Datenblatt
Seite 78
[Quelle: spms376e.pdf]

Link Register

Register 15: Link Register (LR)

The **Link Register (LR**) is register R14, and it stores the return information for subroutines, function calls, and exceptions. The Link Register can be accessed from either privileged or unprivileged mode.

`EXC_RETURN` is loaded into the **LR** on exception entry. See Table 2-10 on page 111 for the values and description.

Link Register (LR)

Type RW, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	LINK															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LINK															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:0	LINK	RW	0xFFFF.FFFF	This field is the return address.

Datenblatt
Seite 79
[Quelle: spms376e.pdf]

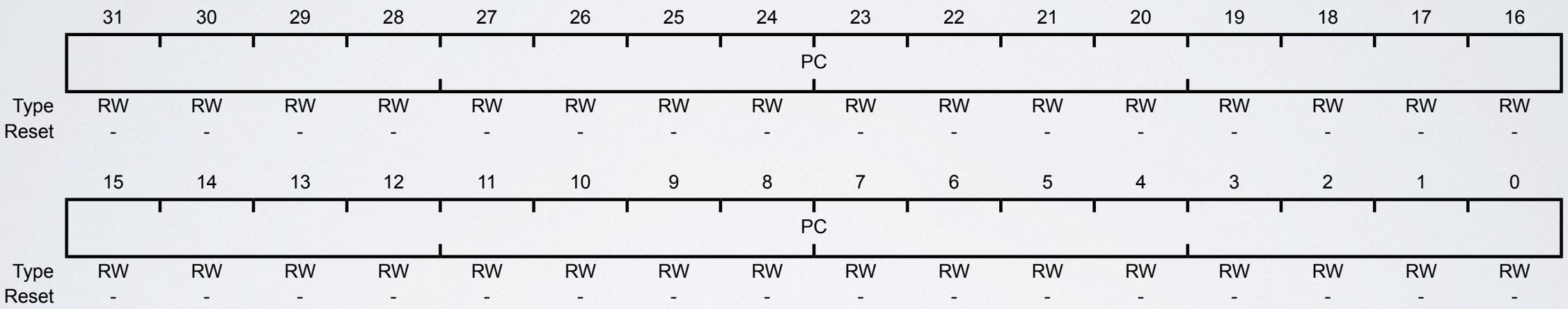
Program Counter

Register 16: Program Counter (PC)

The **Program Counter (PC)** is register R15, and it contains the current program address. On reset, the processor loads the **PC** with the value of the reset vector, which is at address 0x0000.0004. Bit 0 of the reset vector is loaded into the `THUMB` bit of the **EPSR** at reset and must be 1. The **PC** register can be accessed in either privileged or unprivileged mode.

Program Counter (PC)

Type RW, reset -



Bit/Field	Name	Type	Reset	Description
31:0	PC	RW	-	This field is the current program address.

Datenblatt
Seite 80
[Quelle: spms376e.pdf]

Program Status-Register

Program Status Register (PSR)

Type RW, reset 0x0100.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	N	Z	C	V	Q	ICI / IT		THUMB	reserved				GE			
Type	RW	RW	RW	RW	RW	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ICI / IT						reserved		ISRNUM							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APSR

N: Negative or Less Flag

Z: Zero Flag

C: Carry or Borrow Flag

V: Overflow Flag

Q: DSP Overflow and Saturation Flag

EPSR

ICI/IT: Interruptible-Continuable Instruction State

THUMB: Thumb State

GE: Greater Than or Equal Flags

IPSR

ISRNUM: Exception Type Number of current ISR

Datenblatt
Seite 81-84
[Quelle: spms376e.pdf]

Speicherstruktur

Address Range	Memory Region	Memory Type	Description
0x0000.0000 - 0x1FFF.FFFF	Code	Normal	Program Code (also Data)
0x2000.0000 - 0x3FFF.FFFF	SRAM	Normal	Data (also Program Code)
0x4000.000 - 0x5FFF.FFFF	Peripheral	Device	Bit Band, Bit Band Alias Areas
0x6000.0000 - 0x9FFF.FFFF	External RAM	Normal	Data
0xA000.0000 - 0xDFFF.FFFF	External Device	Device	External Device Memory
0xE000.0000 - 0xE00F.FFFF	Private Peripheral Bus	Strongly Ordered	NVIC, System Timer, System Control Block
0xE010.0000 - 0xFFFF.FFFF	Reserved		

Datenblatt
Seiten 95-97
[Quelle: spms376e.pdf]

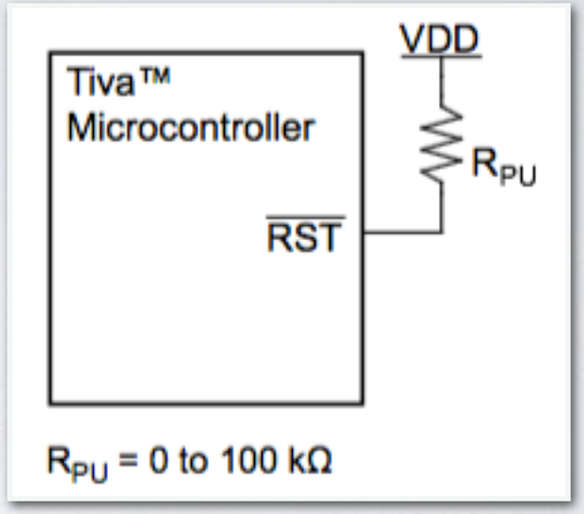
Exception number	IRQ number	Offset	Vector
154	138	0x0268	IRQ131
.	.	.	.
.	.	.	.
.	.	.	.
18	2	0x004C	IRQ2
17	1	0x0048	IRQ1
16	0	0x0044	IRQ0
15	-1	0x0040	Systick
14	-2	0x003C	PendSV
13		0x0038	Reserved
12			Reserved for Debug
11	-5	0x002C	SVCall
10			Reserved
9			
8			
7			
6	-10	0x0018	Usage fault
5	-11	0x0014	Bus fault
4	-12	0x0010	Memory management fault
3	-13	0x000C	Hard fault
2	-14	0x0008	NMI
1		0x0004	Reset
		0x0000	Initial SP value

Interrupt Vectors

Datenblatt
Seiten 106-108
[Quelle: spms376e.pdf]

Reset Control

- Resetquellen:
  - Power-On reset (POR) (Seite 214)
  - Extern reset input pin (RST) (Seite 215)
  - Watchdog (Seite 217)
  - Brown-Out (BOR0, BOR1) (Seite 216)
    - Externe Spannungsversorgung unterschreitet das spezifizierte V_{DD}
  - Software-initiated reset (Seite 217)
  - MSOC failure (Seite 218)



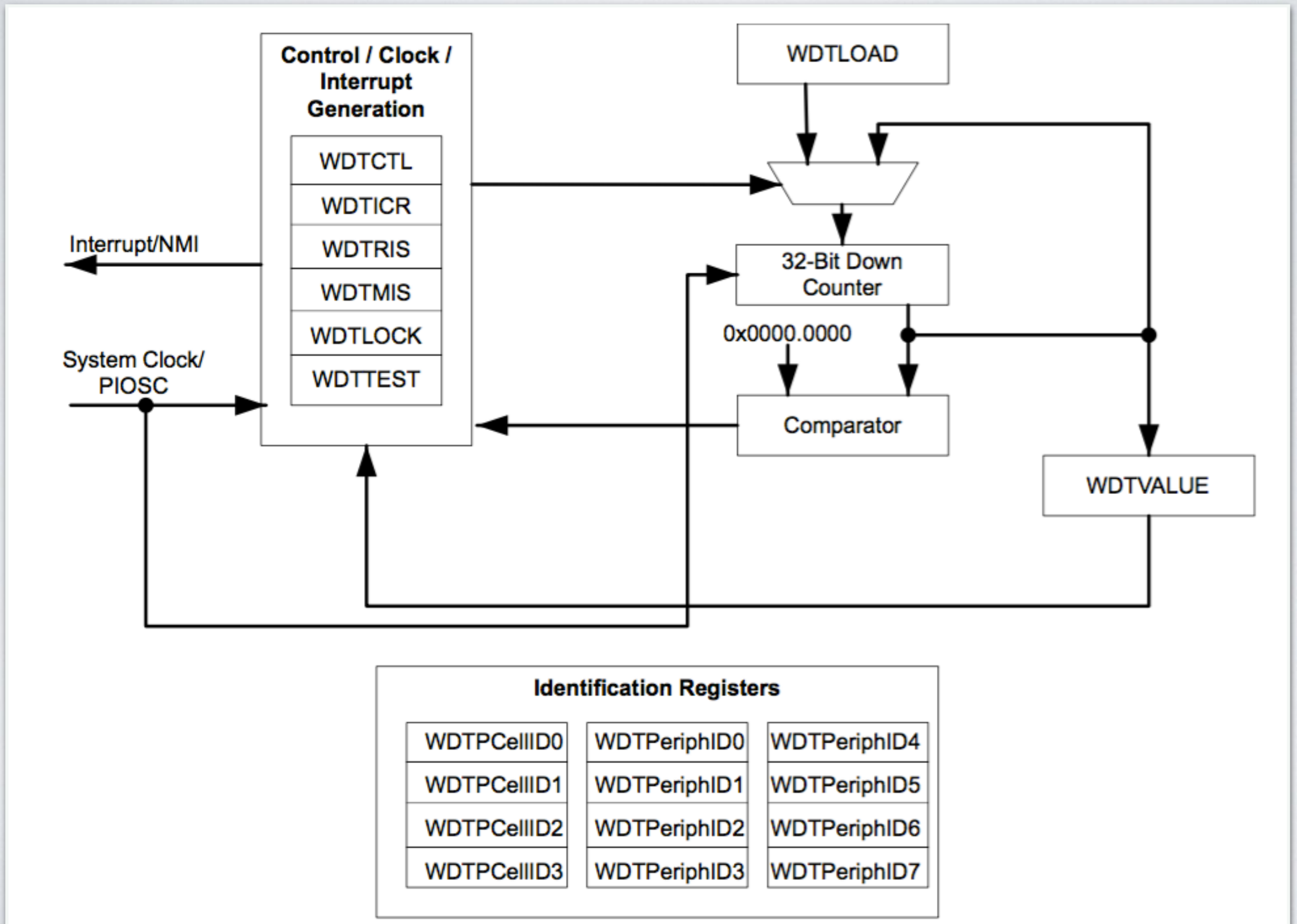
Datenblatt
Seiten 213-217,
1370ff
[Quelle: spms376e.pdf]

Reset Control

Reset Source	Core Reset?	JTAG Reset?	On-Chip Peripherals Reset?
Power-On Reset	Yes	Yes	Yes
`RST`	Yes	Pin Config Only	Yes
Brown-Out Reset	Yes	Pin Config Only	Yes
Software System Request Reset using the `SYSRESREQ` bit in the **APINT** register.	Yes	Pin Config Only	Yes
Software System Request Reset using the `VECTRESET` bit in the **APINT** register.	Yes	Pin Config Only	No
Software Peripheral Reset	No	Pin Config Only	Programmable SRCR
Watchdog Reset	Yes	Pin Config Only	Yes
MOSC Failure Reset	Yes	Pin Config Only	Yes

Datenblatt
Seiten 213-217,
1370ff
[Quelle: spms376e.pdf]

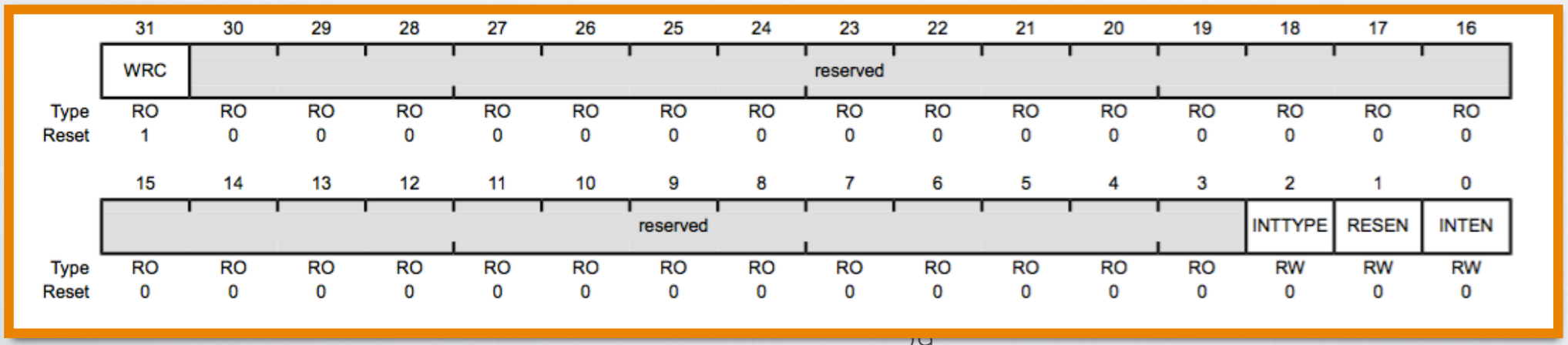
Cortex-M4F Watchdog



Datenblatt
Seite 774-798
[Quelle: spms376e.pdf]

Watchdog Control-Register (WDTCTL)

Offset	Name	Type	Reset	Description
0x000	WDTLOAD	RW	0xFFFF.FFFF	Watchdog Load
0x004	WDTVALUE	RO	0xFFFF.FFFF	Watchdog Value
0x008	WDTCTL	RW	0x0000.0000 (WDT0) 0x8000.0000 (WDT1)	Watchdog Control
0x00C	WDTICR	WO	-	Watchdog Interrupt Clear
0x010	WDTRIS	RO	0x0000.0000	Watchdog Raw Interrupt Status
0x014	WDTMIS	RO	0x0000.0000	Watchdog Masked Interrupt Status
0x418	WDTTEST	RW	0x0000.0000	Watchdog Test
0xC00	WDTLOCK	RW	0x0000.0000	Watchdog Lock
…	…	…	…	…



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WRC	reserved														
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													INTTYPE	RESEN	INTEN
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

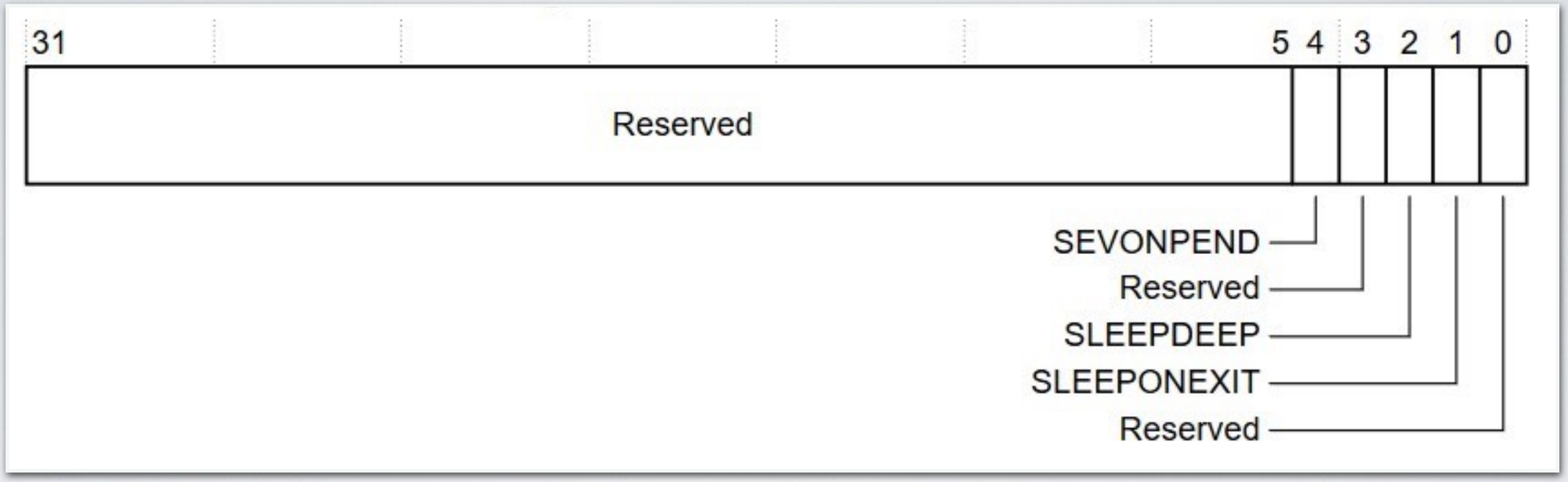
Datenblatt Seiten 777-779
[Quelle: spms376e.pdf]

Power Management

- Unterscheidet sich in Sleep- und Deep-Sleep-Mode
- Sleep-Mode hält den Prozessortakt an
- Deep-Sleep-Mode stoppt System-Clock und schaltet die PLL und den Speicher aus
- Genaueres s. Datenblatt

Datenblatt
Seite 114-115
[Quelle: spms376e.pdf]

Power Management - SCR



SEVONPEND: Send Event on Pending

SLEEPDEEP: 0 = Sleep-Mode, 1 = Deep-Sleep-Mode

SLEEPONEXIT: Indicates sleep-on-exit

Datenblatt
Seite 237-242
[Quelle: spms376e.pdf]

Table 3-8. Peripherals Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x208	PEND2	RW	0x0000.0000	Interrupt 64-95 Set Pending	146
0x20C	PEND3	RW	0x0000.0000	Interrupt 96-127 Set Pending	146
0x210	PEND4	RW	0x0000.0000	Interrupt 128-138 Set Pending	147
0x280	UNPEND0	RW	0x0000.0000	Interrupt 0-31 Clear Pending	148
0x284	UNPEND1	RW	0x0000.0000	Interrupt 32-63 Clear Pending	148
0x288	UNPEND2	RW	0x0000.0000	Interrupt 64-95 Clear Pending	148
0x28C	UNPEND3	RW	0x0000.0000	Interrupt 96-127 Clear Pending	148
0x290	UNPEND4	RW	0x0000.0000	Interrupt 128-138 Clear Pending	149
0x300	ACTIVE0	RO	0x0000.0000	Interrupt 0-31 Active Bit	150
0x304	ACTIVE1	RO	0x0000.0000	Interrupt 32-63 Active Bit	150
0x308	ACTIVE2	RO	0x0000.0000	Interrupt 64-95 Active Bit	150
0x30C	ACTIVE3	RO	0x0000.0000	Interrupt 96-127 Active Bit	150
0x310	ACTIVE4	RO	0x0000.0000	Interrupt 128-138 Active Bit	151
0x400	PRI0	RW	0x0000.0000	Interrupt 0-3 Priority	152
0x404	PRI1	RW	0x0000.0000	Interrupt 4-7 Priority	152
0x408	PRI2	RW	0x0000.0000	Interrupt 8-11 Priority	152
0x40C	PRI3	RW	0x0000.0000	Interrupt 12-15 Priority	152
0x410	PRI4	RW	0x0000.0000	Interrupt 16-19 Priority	152
0x414	PRI5	RW	0x0000.0000	Interrupt 20-23 Priority	152
0x418	PRI6	RW	0x0000.0000	Interrupt 24-27 Priority	152
0x41C	PRI7	RW	0x0000.0000	Interrupt 28-31 Priority	152
0x420	PRI8	RW	0x0000.0000	Interrupt 32-35 Priority	152
0x424	PRI9	RW	0x0000.0000	Interrupt 36-39 Priority	152
0x428	PRI10	RW	0x0000.0000	Interrupt 40-43 Priority	152
0x42C	PRI11	RW	0x0000.0000	Interrupt 44-47 Priority	152
0x430	PRI12	RW	0x0000.0000	Interrupt 48-51 Priority	152
0x434	PRI13	RW	0x0000.0000	Interrupt 52-55 Priority	152
0x438	PRI14	RW	0x0000.0000	Interrupt 56-59 Priority	152
0x43C	PRI15	RW	0x0000.0000	Interrupt 60-63 Priority	152
0x440	PRI16	RW	0x0000.0000	Interrupt 64-67 Priority	154
0x444	PRI17	RW	0x0000.0000	Interrupt 68-71 Priority	154
0x448	PRI18	RW	0x0000.0000	Interrupt 72-75 Priority	154

Table 3-8. Peripherals Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x44C	PRI19	RW	0x0000.0000	Interrupt 76-79 Priority	154
0x450	PRI20	RW	0x0000.0000	Interrupt 80-83 Priority	154
0x454	PRI21	RW	0x0000.0000	Interrupt 84-87 Priority	154
0x458	PRI22	RW	0x0000.0000	Interrupt 88-91 Priority	154
0x45C	PRI23	RW	0x0000.0000	Interrupt 92-95 Priority	154
0x460	PRI24	RW	0x0000.0000	Interrupt 96-99 Priority	154
0x464	PRI25	RW	0x0000.0000	Interrupt 100-103 Priority	154
0x468	PRI26	RW	0x0000.0000	Interrupt 104-107 Priority	154
0x46C	PRI27	RW	0x0000.0000	Interrupt 108-111 Priority	154
0x470	PRI28	RW	0x0000.0000	Interrupt 112-115 Priority	154
0x474	PRI29	RW	0x0000.0000	Interrupt 116-119 Priority	154
0x478	PRI30	RW	0x0000.0000	Interrupt 120-123 Priority	154
0x47C	PRI31	RW	0x0000.0000	Interrupt 124-127 Priority	154
0x480	PRI32	RW	0x0000.0000	Interrupt 128-131 Priority	154
0x484	PRI33	RW	0x0000.0000	Interrupt 132-135 Priority	154
0x488	PRI34	RW	0x0000.0000	Interrupt 136-138 Priority	154
0xF00	SWTRIG	WO	0x0000.0000	Software Trigger Interrupt	156
System Control Block (SCB) Registers					
0x008	ACTLR	RW	0x0000.0000	Auxiliary Control	157
0xD00	CPUID	RO	0x410F.C241	CPU ID Base	159
0xD04	INTCTRL	RW	0x0000.0000	Interrupt Control and State	160
0xD08	VTABLE	RW	0x0000.0000	Vector Table Offset	163
0xD0C	APINT	RW	0xFA05.0000	Application Interrupt and Reset Control	164
0xD10	SYSCTRL	RW	0x0000.0000	System Control	166
0xD14	CFGCTRL	RW	0x0000.0200	Configuration and Control	168
0xD18	SYSPRI1	RW	0x0000.0000	System Handler Priority 1	170
0xD1C	SYSPRI2	RW	0x0000.0000	System Handler Priority 2	171
0xD20	SYSPRI3	RW	0x0000.0000	System Handler Priority 3	172
0xD24	SYSHNDCTRL	RW	0x0000.0000	System Handler Control and State	173
0xD28	FAULTSTAT	RW1C	0x0000.0000	Configurable Fault Status	177
0xD2C	HFAULTSTAT	RW1C	0x0000.0000	Hard Fault Status	183
0xD34	MMADDR	RW	-	Memory Management Fault Address	184

Register-Summary

Datenblatt
Seite 74 - 75,
113, 134-137
[Quelle: spms376e.pdf]

Befehlsübersicht

Operation	Description	Assembler	Cycles
Move	Register	MOV Rd, <op2>	1
	16-bit immediate	MOVW Rd, #<imm>	1
	Immediate into top	MOVT Rd, #<imm>	1
	To PC	MOV PC, Rm	1 + P
Add	Add	ADD Rd, Rn, <op2>	1
	Add to PC	ADD PC, PC, Rm	1 + P
	Add with carry	ADC Rd, Rn, <op2>	1
	Form address	ADR Rd, <label>	1
Subtract	Subtract	SUB Rd, Rn, <op2>	1
	Subtract with borrow	SBC Rd, Rn, <op2>	1
	Reverse	RSB Rd, Rn, <op2>	1
Multiply	Multiply	MUL Rd, Rn, Rm	1
	Multiply accumulate	MLA Rd, Rn, Rm	2
	Multiply subtract	MLS Rd, Rn, Rm	2
	Long signed	SMULL RdLo, RdHi, Rn, Rm	1
	Long unsigned	UMULL RdLo, RdHi, Rn, Rm	1
	Long signed accumulate	SMLAL RdLo, RdHi, Rn, Rm	1
	Long unsigned accumulate	UMLAL RdLo, RdHi, Rn, Rm	1
Divide	Signed	SDIV Rd, Rn, Rm	2 to 12[a]
	Unsigned	UDIV Rd, Rn, Rm	2 to 12[a]
Saturate	Signed	SSAT Rd, #<imm>, <op2>	1
	Unsigned	USAT Rd, #<imm>, <op2>	1
Compare	Compare	CMP Rn, <op2>	1
	Negative	CMN Rn, <op2>	1

Datenblatt Seite (3-4) - (3-8)

[Quelle: Cortex-M4 Technical Reference Manual Revision r0p0]

Befehlsübersicht

Logical	AND	AND Rd, Rn, <op2>	1
	Exclusive OR	EOR Rd, Rn, <op2>	1
	OR	ORR Rd, Rn, <op2>	1
	OR NOT	ORN Rd, Rn, <op2>	1
	Bit clear	BIC Rd, Rn, <op2>	1
	Move NOT	MVN Rd, <op2>	1
	AND test	TST Rn, <op2>	1
	Exclusive OR test	TEQ Rn, <op1>	
Shift	Logical shift left	LSL Rd, Rn, #<imm>	1
	Logical shift left	LSL Rd, Rn, Rs	1
	Logical shift right	LSR Rd, Rn, #<imm>	1
	Logical shift right	LSR Rd, Rn, Rs	1
	Arithmetic shift right	ASR Rd, Rn, #<imm>	1
	Arithmetic shift right	ASR Rd, Rn, Rs	1
Rotate	Rotate right	ROR Rd, Rn, #<imm>	1
	Rotate right	ROR Rd, Rn, Rs	1
	With extension	RRX Rd, Rn	1
Count	Leading zeroes	CLZ Rd, Rn	1
Load	Word	LDR Rd, [Rn, <op2>]	2b
	To PC	LDR PC, [Rn, <op2>]	2b + P
	Halfword	LDRH Rd, [Rn, <op2>]	2b
	Byte	LDRB Rd, [Rn, <op2>]	2b
	Signed halfword	LDRSH Rd, [Rn, <op2>]	2b
	Signed byte	LDRSB Rd, [Rn, <op2>]	2b
	User word	LDRT Rd, [Rn, #<imm>]	2b
	User halfword	LDRHT Rd, [Rn, #<imm>]	2b
	User byte	LDRBT Rd, [Rn, #<imm>]	2b

Datenblatt Seite (3-4) - (3-8)

[Quelle: Cortex-M4 Technical Reference Manual Revision r0p0]

Befehlsübersicht

Store	Word	STR Rd, [Rn, <op2>]	2b
	Halfword	STRH Rd, [Rn, <op2>]	2b
	Byte	STRB Rd, [Rn, <op2>]	2b
	Signed halfword	STRSH Rd, [Rn, <op2>]	2b
	Signed byte	STRSB Rd, [Rn, <op2>]	2b
	User word	STRT Rd, [Rn, #<imm>]	2b
	User halfword	STRHT Rd, [Rn, #<imm>]	2b
	User byte	STRBT Rd, [Rn, #<imm>]	2b
	User signed halfword	STRSHT Rd, [Rn, #<imm>]	2b
	User signed byte	STRSBT Rd, [Rn, #<imm>]	2b
	Doubleword	STRD Rd, Rd, [Rn, #<imm>]	1 + N
	Multiple	STM Rn, {<reglist>}	1 + N
Push	Push	PUSH {<reglist>}	1 + N
	Push with link register	PUSH {<reglist>, LR}	1 + N
Pop	Pop	POP {<reglist>}	1 + N
	Pop and return	POP {<reglist>, PC}	1 + N + P
Semaphore	Load exclusive	LDREX Rd, [Rn, #<imm>]	2
	Load exclusive half	LDREXH Rd, [Rn]	2
	Load exclusive byte	LDREXB Rd, [Rn]	2
	Store exclusive	STREX Rd, Rt, [Rn, #<imm>]	2
	Store exclusive half	STREXH Rd, Rt, [Rn]	2
	Store exclusive byte	STREXB Rd, Rt, [Rn]	2
	Clear exclusive monitor	CLREX	1

Datenblatt Seite (3-4) - (3-8)

[Quelle: Cortex-M4 Technical Reference Manual Revision r0p0]

Befehlsübersicht

Branch	Conditional	B<cc> <label>	1 or 1 + P_c
	Unconditional	B <label>	1 + P
	With link	BL <label>	1 + P
	With exchange	BX Rm	1 + P
	With link and exchange	BLX Rm	1 + P
	Branch if zero	CBZ Rn, <label>	1 or 1 + Pc
	Branch if non-zero	CBNZ Rn, <label>	1 or 1 + Pc
	Byte table branch	TBB [Rn, Rm]	2 + P
	Halfword table branch	TBH [Rn, Rm, LSL#1]	2 + P
State change	Supervisor call	SVC #<imm>	-
	If-then-else	IT... <cond>	1d
	Disable interrupts	CPSID <flags>	1 or 2
	Enable interrupts	CPSIE <flags>	1 or 2
	Read special register	MRS Rd, <specreg>	1 or 2
	Write special register	MSR <specreg>, Rn	1 or 2
	Breakpoint	BKPT #<imm>	-
Extend	Signed halfword to word	SXTH Rd, <op2>	1
	Signed byte to word	SXTB Rd, <op2>	1
	Unsigned halfword	UXTH Rd, <op2>	1
	Unsigned byte	UXTB Rd, <op2>	1
Bit field	Extract unsigned	UBFX Rd, Rn, #<imm>, #<imm>	1
	Extract signed	SBFX Rd, Rn, #<imm>, #<imm>	1
	Clear	BFC Rd, Rn, #<imm>, #<imm>	1
	Insert	BFI Rd, Rn, #<imm>, #<imm>	1

Datenblatt Seite (3-4) - (3-8)

[Quelle: Cortex-M4 Technical Reference Manual Revision r0p0]

Register-Summary

Address	Name	Type	Reset	Description
0xE000E008	ACTLR	RW	0x00000000	Auxiliary Control Register, ACTLR on page 4-5
0xE000E010	STCSR	RW	0x00000000	SysTick Control and Status Register
0xE000E014	STRVR	RW	Unknown	SysTick Reload Value Register
0xE000E018	STCVR	RW clear	Unknown	SysTick Current Value Register
0xE000E01C	STCR	RO	STCALIB	SysTick Calibration Value Register
0xE000ED00	CPUID	RO	0x410FC240	CPUID Base Register, CPUID on page 4-5
0xE000ED04	ICSR	RW or RO	0x00000000	Interrupt Control and State Register
0xE000ED08	VTOR	RW	0x00000000	Vector Table Offset Register
0xE000ED0C	AIRCR	RW	0x00000000a	Application Interrupt and Reset Control Register
0xE000ED10	SCR	RW	0x00000000	System Control Register
0xE000ED14	CCR	RW	0x00000200	Configuration and Control Register.
0xE000ED18	SHPR1	RW	0x00000000	System Handler Priority Register 1
0xE000ED1C	SHPR2	RW	0x00000000	System Handler Priority Register 2
0xE000ED20	SHPR3	RW	0x00000000	System Handler Priority Register 3
0xE000ED24	SHCSR	RW	0x00000000	System Handler Control and State Register
0xE000ED28	CFSR	RW	0x00000000	Configurable Fault Status Registers
0xE000ED2C	HFSR	RW	0x00000000	HardFault Status register
0xE000ED30	DFSR	RW	0x00000000	Debug Fault Status Register
0xE000ED34	MMFAR	RW	Unknown	MemManage Address Registerb
0xE000ED38	BFAR	RW	Unknown	BusFault Address Registerb
0xE000ED3C	AFSR	RW	0x00000000	Auxiliary Fault Status Register, AFSR on page 4-6
0xE000ED40	ID_PFR0	RO	0x00000030	Processor Feature Register 0
0xE000ED44	ID_PFR1	RO	0x00000200	Processor Feature Register 1
0xE000ED48	ID_DFR0	RO	0x00100000	Debug Features Register 0

Datenblatt
Seite (4-3) - (4-4)

[Quelle: Cortex-M4 Technical Reference Manual Revision r0p0]

0xE000ED08	VTOR	RW	0x00000000	Vector Table Offset Register
0xE000ED0C	AIRCR	RW	0x00000000a	Application Interrupt and Reset Control Register
0xE000ED10	SCR	RW	0x00000000	System Control Register
0xE000ED14	CCR	RW	0x00000200	Configuration and Control Register.
0xE000ED18	SHPR1	RW	0x00000000	System Handler Priority Register 1
0xE000ED1C	SHPR2	RW	0x00000000	System Handler Priority Register 2
0xE000ED20	SHPR3	RW	0x00000000	System Handler Priority Register 3
0xE000ED24	SHCSR	RW	0x00000000	System Handler Control and State Register
0xE000ED28	CFSR	RW	0x00000000	Configurable Fault Status Registers
0xE000ED2C	HFSR	RW	0x00000000	HardFault Status register
0xE000ED30	DFSR	RW	0x00000000	Debug Fault Status Register
0xE000ED34	MMFAR	RW	Unknown	MemManage Address Registerb
0xE000ED38	BFAR	RW	Unknown	BusFault Address Registerb
0xE000ED3C	AFSR	RW	0x00000000	Auxiliary Fault Status Register, AFSR on page 4-6
0xE000ED40	ID_PFR0	RO	0x00000030	Processor Feature Register 0
0xE000ED44	ID_PFR1	RO	0x00000200	Processor Feature Register 1
0xE000ED48	ID_DFR0	RO	0x00100000	Debug Features Register 0
0xE000ED4C	ID_AFR0	RO	0x00000000	Auxiliary Features Register 0
0xE000ED50	ID_MMFR0	RO	0x00000030	Memory Model Feature Register 0
0xE000ED54	ID_ MMFR1	RO	0x00000000	Memory Model Feature Register 1
0xE000ED58	ID_MMFR2	RO	0x00000000	Memory Model Feature Register 2
0xE000ED5C	ID_MMFR3	RO	0x00000000	Memory Model Feature Register 3
0xE000ED60	ID_ISAR0	RO	0x01141110	Instruction Set Attributes Register 0
0xE000ED64	ID_ISAR1	RO	0x02112000	Instruction Set Attributes Register 1
0xE000ED68	ID_ISAR2	RO	0x21232231	Instruction Set Attributes Register 2
0xE000ED6C	ID_ISAR3	RO	0x01111131	Instruction Set Attributes Register 3
0xE000ED70	ID_ISAR4	RO	0x01310102	Instruction Set Attributes Register 4
0xE000ED88	CPACR	RW	-	Coprocessor Access Control Register
0xE000EF00	STIR	WO	0x00000000	Software Triggered Interrupt Register

Datenblatt
Seite (4-3) - (4-4)

[Quelle: Cortex-M4 Technical Reference Manual Revision r0p0]

Instruction Set

P	The number of cycles required for a pipeline refill. This ranges from 1 to 3 depending on the alignment and width of the target instruction, and whether the processor manages to speculate the address early.
B	The number of cycles required to perform the barrier operation. For DSB and DMB, the minimum number of cycles is zero. For ISB, the minimum number of cycles is equivalent to the number required for a pipeline refill.
N	The number of registers in the register list to be loaded or stored, including PC or LR.
W	The number of cycles spent waiting for an appropriate event.
Rd	Destination (and source) register
Rn, Rm	Source register
PC	Programcounter before this line
NOP	No-Operation

Datenblatt
Seite (3-4)

[Quelle: Cortex-M4 Technical Reference Manual Revision r0p0]

Memory accesses

Offset addressing

- The offset value is added to or subtracted from an address obtained from the base register. The result is used as the address for the memory access. The base register is unaltered.

- The assembly language syntax for this mode is: [<Rn>,<offset>]

Pre-indexed addressing

- The offset value is applied to an address obtained from the base register. The result is used as the address for the memory access, and written back into the base register.

- The assembly language syntax for this mode is: [<Rn>,<offset>]!

Post-indexed addressing

- The address obtained from the base register is used, unaltered, as the address for the memory access. The offset value is applied to the address, and written back into the base register.

- The assembly language syntax for this mode is: [<Rn>],<offset>

In each case, <Rn> is the base register. <offset> can be:

- an immediate constant, such as <imm8> or <imm12>

- an index register, <Rm>

- a shifted index register, such as <Rm>, LSL #<shift>.

[Quelle: ARMv7-M Architecture Reference Manual, Chapter A6.5]

Conditional execution

cond	Mnemonic extension	Meaning (integer)	Meaning (floating-point) [ab]	Condition flags
0000	EQ	Equal	Equal	Z == 1
0001	NE	Not equal	Not equal, or unordered	Z == 0
0010	CS [c]	Carry set	Greater than, equal, or unordered	C == 1
0011	CC [d]	Carry clear	Less than	C == 0
0100	MI	Minus, negative	Less than	N == 1
0101	PL	Plus, positive or zero	Greater than, equal, or unordered	N == 0
0110	VS	Overflow	Unordered	V == 1
0111	VC	No overflow	Not unordered	V == 0
1000	HI	Unsigned higher	Greater than, or unordered	C == 1 and Z == 0
1001	LS	Unsigned lower or same	Less than or equal	C == 0 or Z == 1
1010	GE	Signed greater than or equal	Greater than or equal	N == V
1011	LT	Signed less than	Less than, or unordered	N != V
1100	GT	Signed greater than	Greater than	Z == 0 and N == V
1101	LE	Signed less than or equal	Less than, equal, or unordered	Z == 1 or N != V
1110	None (AL) [e]	Always (unconditional)	Always (unconditional)	Any

[Quelle: ARMv7-M Architecture Reference Manual, Chapter A6.3]

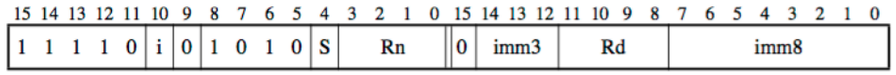
ARM-Thumb-Assembler-Beispielbefehl

A6.7.1 ADC (immediate)

Add with Carry (immediate) adds an immediate value and the carry flag value to a register value, and writes the result to the destination register. It can optionally update the condition flags based on the result.

Encoding T1 ARMv7-M

```
ADC{S}<c> <Rd>,<Rn>,#<const>
```




```
d = UInt(Rd);  n = UInt(Rn);  setflags = (S == '1');  imm32 = ThumbExpandImm(i:imm3:imm8);
if d IN {13,15} || n IN {13,15} then UNPREDICTABLE;
```


Copyright © 2006-2008 ARM Limited. All rights reserved.
Non-Confidential

Assembler syntax

```
ADC{S}<c><q>  {<Rd>,} <Rn>, #<const>
```

where:

S — If present, specifies that the instruction updates the flags. Otherwise, the instruction does not update the flags.

<c><q> — See *Standard assembler syntax fields* on page A6-7.

<Rd> — Specifies the destination register. If <Rd> is omitted, this register is the same as <Rn>.

<Rn> — Specifies the register that contains the first operand.

<const> — Specifies the immediate value to be added to the value obtained from <Rn>. See *Modified immediate constants in Thumb instructions* on page A5-15 for the range of allowed values.

The pre-UAL syntax ADC<c>S is equivalent to ADCS<c>.

Operation

```
if ConditionPassed() then
    EncodingSpecificOperations();
    (result, carry, overflow) = AddWithCarry(R[n], imm32, APSR.C);
    R[d] = result;
    if setflags then
        APSR.N = result<31>;
        APSR.Z = IsZeroBit(result);
        APSR.C = carry;
        APSR.V = overflow;
```

Exceptions

None.

Copyright © 2006-2008 ARM Limited. All rights reserved.
Non-Confidential

[Quelle: ARMv7-M Architecture Reference Manual, Chapter A6.7]

ARM-Thumb-Assembler-Beispielbefehl

A6.7.1 ADC (immediate)

Add with Carry (immediate) adds an immediate value and the carry flag value to a register value, and writes the result to the destination register. It can optionally update the condition flags based on the result.

Encoding T1 ARMv7-M

```
ADC{S}<c> <Rd>,<Rn>,#<const>
```

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	i	0	1	0	1	0	S	Rn				0	imm3			Rd				imm8							

```
d = UInt(Rd);  n = UInt(Rn);  setflags = (S == '1');  imm32 = ThumbExpandImm(i:imm3:imm8);
if d IN {13,15} || n IN {13,15} then UNPREDICTABLE;
```

Assembl

```
ADC{S}<c>
```

where:

S

<c><q>

<Rd>

<Rn>

<const>

The pre-U

Operati

```
if Condit
    Encod
    (resu
    R[d]
    if se
        A
        A
        A
        A
```

Excepti

None.

[Quelle: ARMv7-M Architecture Reference Manual, Chapter A6.7]

ARM-Thumb-Assembler-Beispielbefehl

d writes
t.

2 1 0

Assembler syntax

```
ADC{S}<c><q>  {<Rd>,} <Rn>, #<const>
```

where:

S	If present, specifies that the instruction updates the flags. Otherwise, the instruction does not update the flags.
<c><q>	See *Standard assembler syntax fields* on page A6-7.
<Rd>	Specifies the destination register. If <Rd> is omitted, this register is the same as <Rn>.
<Rn>	Specifies the register that contains the first operand.
<const>	Specifies the immediate value to be added to the value obtained from <Rn>. See *Modified immediate constants in Thumb instructions* on page A5-15 for the range of allowed values.

The pre-UAL syntax ADC<c>S is equivalent to ADCS<c>.

Operation

```
if ConditionPassed() then
    EncodingSpecificOperations();
    (result, carry, overflow) = AddWithCarry(R[n], imm32, APSR.C);
    R[d] = result;
    if setflags then
        APSR.N = result<31>;
        APSR.Z = IsZeroBit(result);
        APSR.C = carry;
        APSR.V = overflow;
```

Exceptions

None.

[Quelle: ARMv7-M Architecture Reference Manual, Chapter A6.7]

Launchpad - ARM-Assembler

- Hinweise zu Assembler mit der ARM Toolchain:
  - https://launchpad.net/gcc-arm-embedded
  - http://energia.nu/
- Assembler-Programm enthält
  - ARM-Befehle und
  - Anweisungen für Linker, z.B. `.section, global, .end.`

Launchpad - Blink ARM-Assembler

120 Byte

```
@ Blink the LED on the Launchpad ARM Cortex M4 board.
@ Directives
        .thumb
        .syntax unified
@ Equates
        .equ SYSCTL_RCGC2_R, 0x400fe108
        .equ GPIO_PORTF_DIR_R, 0x40025400
        .equ GPIO_PORTF_DEN_R, 0x4002551c
        .equ GPIO_PORTF_DATA_R, 0x400253fc
        .equ STACKINIT,   0x20005000
        .equ LEDDELAY,    800000
.section .text
        .org 0
@ Vectors
vectors:
        .word STACKINIT         @ stack pointer value
        .word _start + 1        @ reset vector
        .word _nmi_handler + 1  @
        .word _hard_fault  + 1  @
        .word _memory_fault + 1 @
        .word _bus_fault + 1    @
        .word _usage_fault + 1  @
_start:
        @ enable  gpio
        ldr r6, = SYSCTL_RCGC2_R
        mov r0, 0x20
        str r0, [r6]

        ldr r6, = GPIO_PORTF_DIR_R
        mov r0, 0x8
        str r0, [r6]
        ldr r6, = GPIO_PORTF_DEN_R
        str r0, [r6]

        ldr r6, = GPIO_PORTF_DATA_R    @ point to port F
loop:
        ldr r2, [r6]
        orr r2, #0x8
        str r2, [r6]            @  turn on LED
        ldr r1, = LEDDELAY
delay1:
        subs r1, 1
        bne delay1

        ldr r2, [r6]
        and r2, #0xfffffff7
        str r2, [r6]            @ turn off LED
        ldr r1, = LEDDELAY
delay2:
        subs r1, 1
        bne delay2

        b loop                  @ continue forever

_dummy:                         @ just hang in a loop
_nmi_handler:
_hard_fault:
_memory_fault:
_bus_fault:
_usage_fault:
        add r0, 1
        add r1, 1
        b _dummy
```

Launchpad - Blink.c in C

1056 Byte (-Os)

```c
#include <stdint.h>
#include "inc/tm4c123gh6pm.h"

int main(void) {
    volatile uint32_t ui32Loop;

    // Enable the GPIO port that is used for the on-board LED.
    SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;

    // Do a dummy read to insert a few cycles after enabling the peripheral.
    ui32Loop = SYSCTL_RCGC2_R;

    // Enable the GPIO pin for the LED.  Set the direction as output, and
    // enable the GPIO pin for digital function.
    GPIO_PORTF_DIR_R = 0x08;
    GPIO_PORTF_DEN_R = 0x08;

    // Loop
    for (;;) {
        // Turn on the LED.
        GPIO_PORTF_DATA_R |= 0x08;

        // Delay for a bit.
        for(ui32Loop = 0; ui32Loop < 200000; ui32Loop++){}

        // Turn off the LED.
        GPIO_PORTF_DATA_R &= ~(0x08);

        // Delay for a bit.
        for(ui32Loop = 0; ui32Loop < 200000; ui32Loop++){}
    }
}
```

inclusive startup_gcc.c

Launchpad - startup_gcc.c in C

```c
#include <stdint.h>
#include "inc/hw_nvic.h"
#include "inc/hw_types.h"

void ResetISR(void);
static void NmiSR(void);
static void FaultISR(void);
static void IntDefaultHandler(void);
extern int main(void);

static uint32_t pui32Stack[64];

// The vector table.  Note that the proper constructs must be placed
on this to
__attribute__ ((section(".isr_vector")))
void (* const g_pfnVectors[])(void) = {
    (void (*)(void))((uint32_t)pui32Stack + sizeof(pui32Stack)),
                                            // The initial stack
pointer
    ResetISR,                               // The reset handler
    NmiSR,                                  // The NMI handler
    FaultISR,                               // The hard fault handler
    IntDefaultHandler,                      // The MPU fault handler
    ...
    IntDefaultHandler                       // PWM 1 Fault
};

// for linker
extern uint32_t _etext;
extern uint32_t _data;
extern uint32_t _edata;
extern uint32_t _bss;
extern uint32_t _ebss;
```

```c
void ResetISR(void) {
    uint32_t *pui32Src, *pui32Dest;

    pui32Src = &_etext;
    for(pui32Dest = &_data; pui32Dest < &_edata; )    {
        *pui32Dest++ = *pui32Src++;
    }

    __asm("    ldr     r0, =_bss\n"
          "    ldr     r1, =_ebss\n"
          "    mov     r2, #0\n"
          "    .thumb_func\n"
          "zero_loop:\n"
          "        cmp     r0, r1\n"
          "        it      lt\n"
          "        strlt   r2, [r0], #4\n"
          "        blt     zero_loop");

    HWREG(NVIC_CPAC) = ((HWREG(NVIC_CPAC) &
                         ~(NVIC_CPAC_CP10_M | NVIC_CPAC_CP11_M)) |
                        NVIC_CPAC_CP10_FULL | NVIC_CPAC_CP11_FULL);
    main();
}

static void NmiSR(void) {
    while(1) {}
}

static void FaultISR(void) {
    while(1){}
}

static void IntDefaultHandler(void) {
    while(1){}
}
```

Launchpad - Inline-Assembler

1064 Byte (-Os)

```
#include <stdint.h>
#include "inc/tm4c123gh6pm.h"

int main(void) {
    volatile uint32_t ui32Loop;

    // Enable the GPIO port that is used for the on-board LED.
    SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;

    // Do a dummy read to insert a few cycles after enabling the peripheral.
    ui32Loop = SYSCTL_RCGC2_R;

    // Enable the GPIO pin for the LED (PG2).  Set the direction as output, and
    // enable the GPIO pin for digital function.
    asm volatile(
    "ldr r6, = 0x40025400 \n\t"
    "mov r0, 0x8 \n\t"
    "str r0, [r6] \n\t"

    "ldr r6, = 0x4002551c \n\t"
    "str r0, [r6] \n\t"
    );

    // Loop
    for (;;) {
        // Turn on the LED.
        GPIO_PORTF_DATA_R |= 0x08;

        // Delay for a bit.
        for(ui32Loop = 0; ui32Loop < 200000; ui32Loop++){}

        // Turn off the LED.
        GPIO_PORTF_DATA_R &= ~(0x08);

        // Delay for a bit.
        for(ui32Loop = 0; ui32Loop < 200000; ui32Loop++){}
    }
}
```

Inline Assembler

Zusammenfassung

- Lesen eines Datenblatts
  - Aufbau des Mikrocontrollers
- Programmierung des Mikrocontrollers
  - Assembler
  - C

in aller Kürze

Beispiel einer Anwendung: Bratenthermometer

- Bratenthermometer mit Bluetooth, hier "iGrill"
  - Temperaturmesser
  - Bluetooth
  - Mikrocontroller

[Quelle: http://idevicesinc.com/igrill/images/sections/igrill_family/sect2_img1.jpg]

Literatur / Quellen

- **ARM**, *Cortex-M4 Technical Reference Manual Revision r0p0,* 2009, 2010
- **ARM**, *ARMv7-M Architecture Reference Manual*, 2010
- **Texas Instruments**, *Tiva TM4C123GH6PM, Data Sheet*, URL: http://www.ti.com/lit/ds/spms376e/spms376e.pdf
- Click and Grow, *Smart Pot mit Chili Peper,* URL: http://www.clickandgrow.com/collections/general/products/starter-kit-with-chili-pepper
- iDevices Inc, *iGrill*, URL: http://idevicesinc.com/igrill/
- **Stand aller Internetquellen: 27.04.2016**