Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung

Institutsleitung
Prof. Dr.-Ing. Dr. h. c. J. Becker
Prof. Dr. rer. nat. W. Stork
Prof. Dr.-Ing. E. Sax

Prof. Dr.-Ing. E. Sax
M. Sc. Jijing Yan

Institut für Technik der Informationsverarbeitung (ITIV)

Einführung in das Praktikum Informationstechnik II

Organisatorisches…

- Anmeldung für PIT: bis 15.05.2018 um 23:55 Uhr
  - Via WiWi Plattform (YouSubscribe)
    - portal.wiwi.kit.edu oder wiwi.link/pit2018
- Nach der Gruppeneinteilung: automatisch zur Lernplattform ILIAS hinzugefügt (man muss nicht selbst der Plattform ILIAS beitreten)
- Lastenheft, Folien für Einführunsveranstaltung I auf der ITIV-Website
  - Download mit Passwort: PITSS18
    - http://www.itiv.kit.edu/60_6070.php
  - Tutorium (Selbststudium) im Kapitel 6 im Lastenheft
    - Lösungen für die Aufgaben findet man später in ILIAS

Organisatorisches…

- Studienleistung
  - Im Studierendenportal an-/abmelden: bis dem Ende der 1. PIT-Woche (03.06.2018)
  - Das PIT kann beliebig widerholt werden (Scheinleistung)
  - Scheinleistung: Bestanden / Nicht-Bestanden
  - Keine separate schriftliche Prüfung, aber der Inhalt wird in IT-Modul Klausur geprüft
    - Projektmanagement
    - Hardwarenahe Programmierung

IT-Veranstaltungsplan SS2018

KW	SW	Datum Mittwoch	9:45 – 11:15 Uhr (Benz)	Datum Donnerstag	14:00 – 15:30 Uhr (Neue Chemie)	Praktikum
16.	1.	18. Apr	VL	19. Apr		
17.	2.	25. Apr	VL	26. Apr		
18.	3.	02. Mai	VL			
19.	4.	09. Mai	VL			P (EV-I)
20.	5.	16. Mai	VL			P (EV-II)
21.	6.	23. Mai	frei			
22.	7.	30. Mai				P
23.	8.	06. Jun	VL			P
24.	9.	13. Jun	VL			P
25.	10.	20. Jun	VL	21. Jun	VL	P
26.	11.	27. Jun	VL	28. Jun	ÜB	P
27.	12.	04. Jul	VL	05. Jul	ÜB	P
28.	13.	11. Jul	VL	12. Jul	ÜB	P (PF)
29.	14.	18. Jul	VL	19. Jul	ÜB	

Anmeldung für PIT: (15.05.2018)

Studienleistung im Studierendenportal: (03.06.2018)

*KW = Kalenderwoche; SW = Semesterwoche; VL = Vorlesung; ÜB = Übung; P = Praktikum

*Zwei Einführungsveranstaltungen für Praktikum: 08.05.2018 und 15.05.2018 jeweils um 17:30 Uhr am HSaF

Ziele des Praktikums

- Praktische Anwendung der Vorlesungsinhalte
  - Passende Algorithmen und Datenstrukturen entsprechend einer bestimmten Aufgabenstellung anwenden
- Praktische Anwendung der Übungsinhalte
  - Umsetzung in hardwarenahen C++ Code
  - Programm strukturiert aufbauen und die Befehle zur Umsetzung der Anforderungen bestimmten Funktionen zuordnen
- Praktische Erfahrung im Projekt (Teamarbeit und Softwareentwicklung)
  - Zerlegen von komplexen Problemen, dargeboten in natürlicher Sprache (Spezifikation), in einfache und übersichtliche Module, sowie Ausdrücken dieser Module mit Hilfe einer Programmiersprache (C++)
  - Erzeugen von hardwarenahem Code unter Einhaltung vorgegebener Qualitätskriterien (Programmierrichtlinien)
  - Einhalten eines vorgegebenen Zeitplans

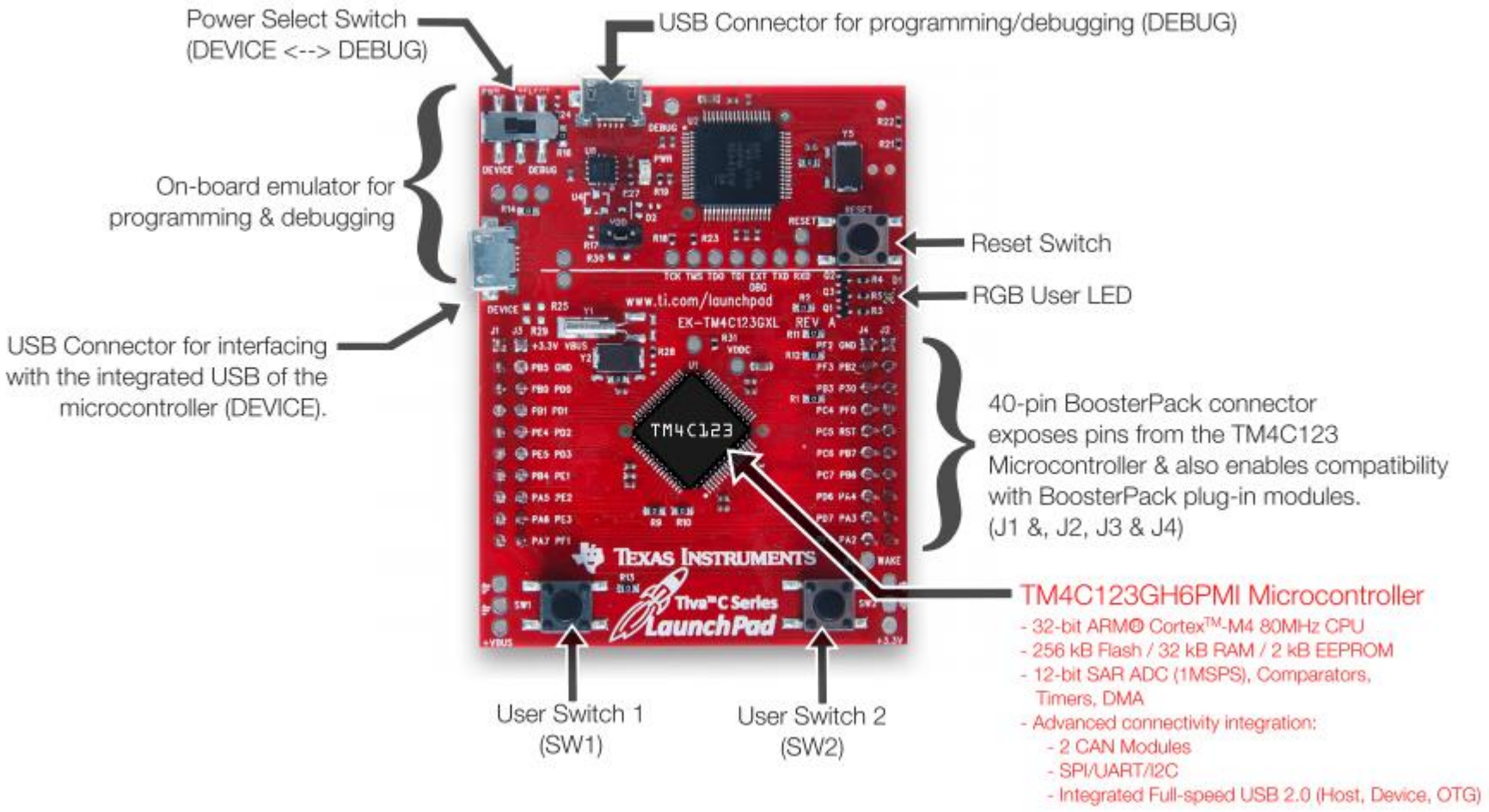
Inhalt

1. Tiva C Series TM4C123G LaunchPad Evaluation Board
2. UART, GPIO, Timer, PWM, ADC, Watchdog im eingebetteten System
3. Einführung in Register
4. Umgang mit Datenblättern
5. Blinky Beispiel

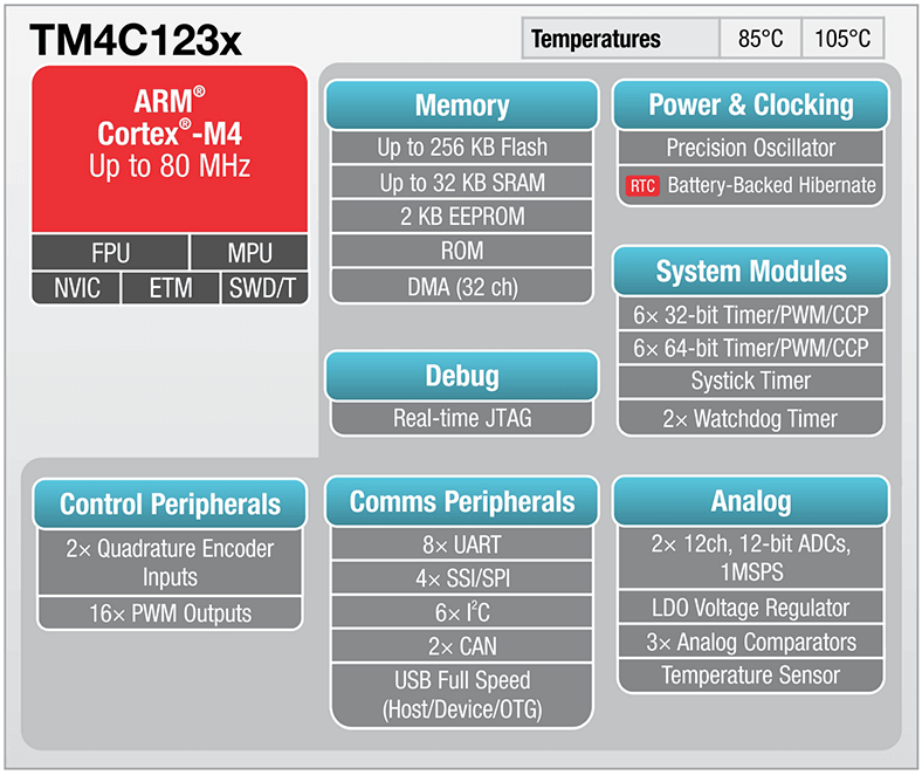
Inhalt

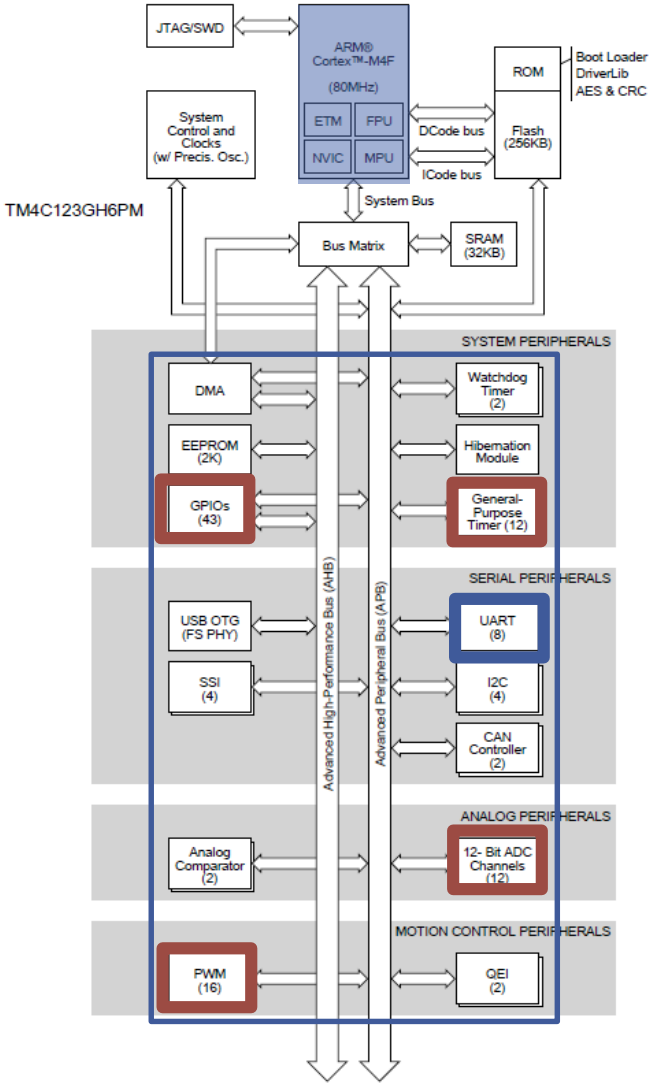
1. **Tiva C Series TM4C123G LaunchPad Evaluation Board**
2. UART, GPIO, Timer, PWM, ADC, Watchdog im eingebetteten System
3. Einführung in Register
4. Umgang mit Datenblättern
5. Blinky Beispiel

Tiva C Series TM4C123G LaunchPad Evaluation Board



TM4C123x Microcontrollers

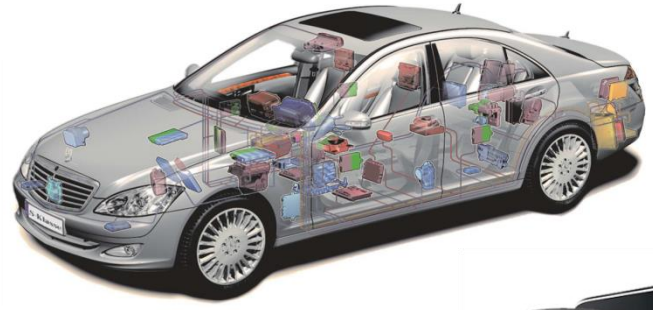


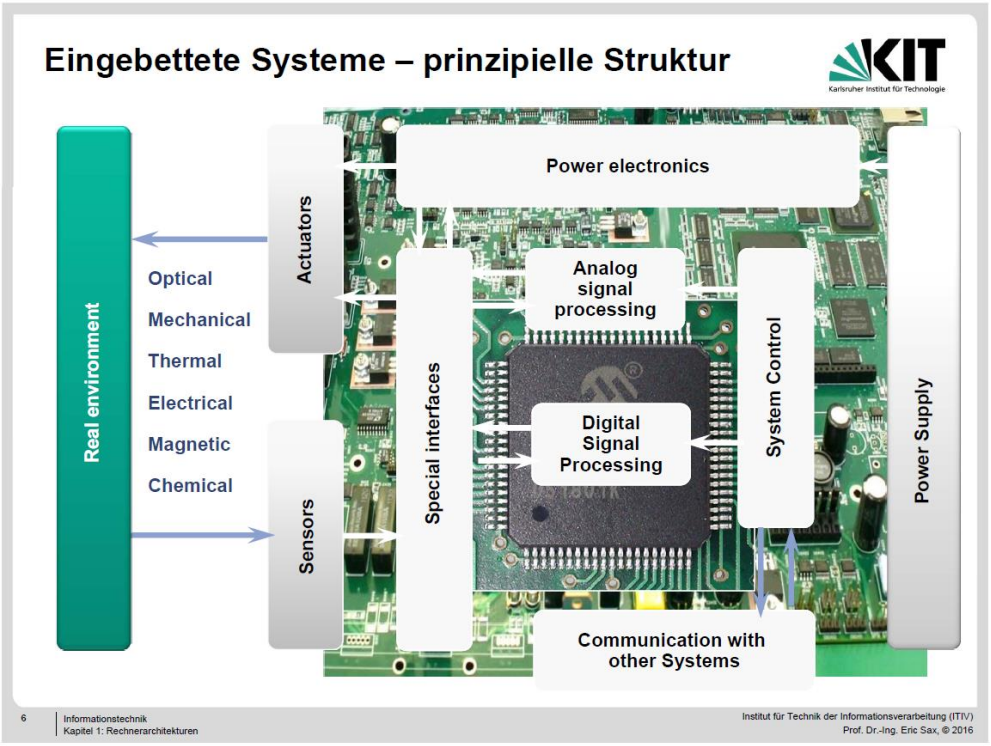


Inhalt

Eingebettete Systeme

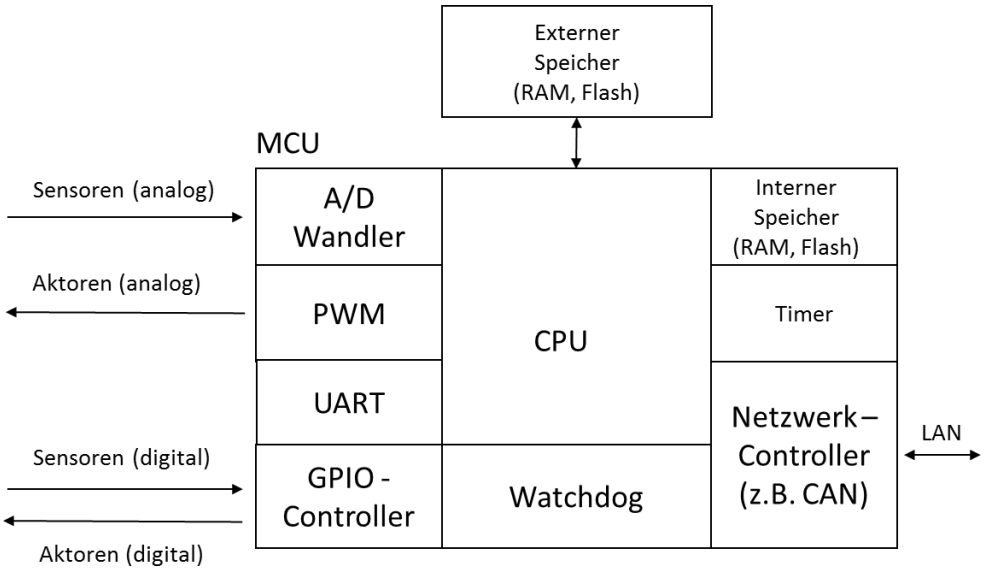
- Hard- & Softwaresystem für komplexe Steuerungs-, Regelungs-, Datenverarbeitungsaufgaben
- Eingebettet in umgebende technische Systeme
- Beispiel: Steuerungseinheit im Auto, Smartphone, Kaffeemaschine…

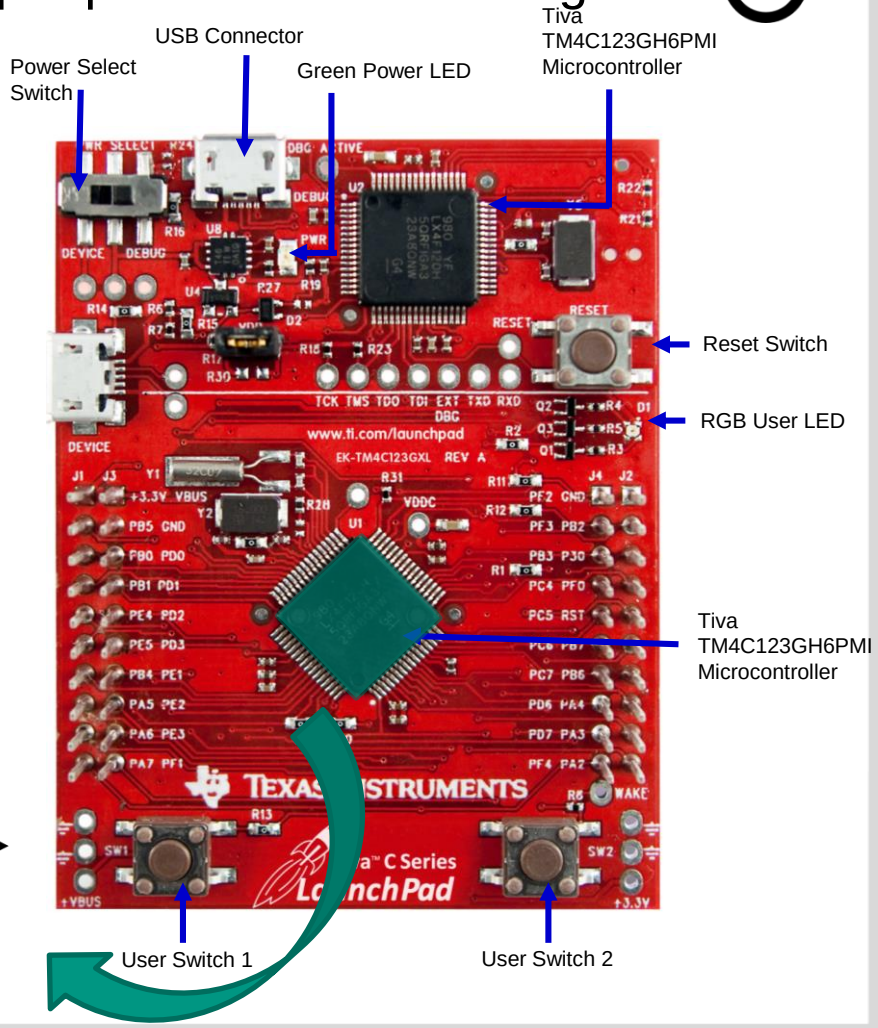




Mikrocontroller

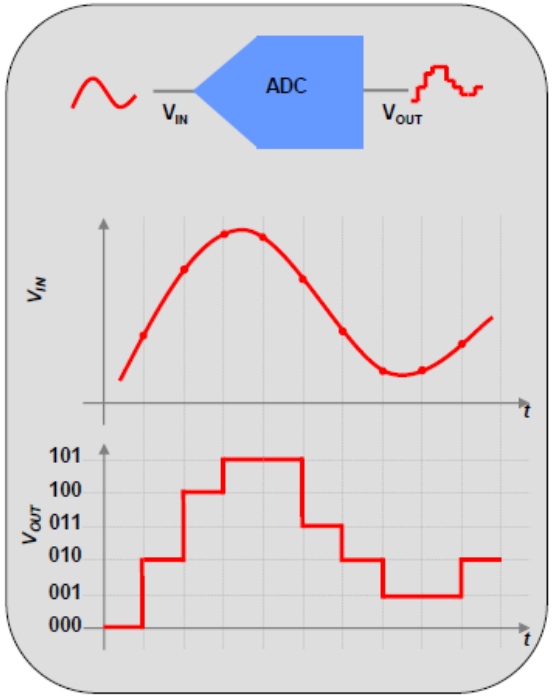
- Mikroprozessor, Speicher & wichtige periphere Einheiten integriert auf einem Chip





A/D-Wandler (ADC)

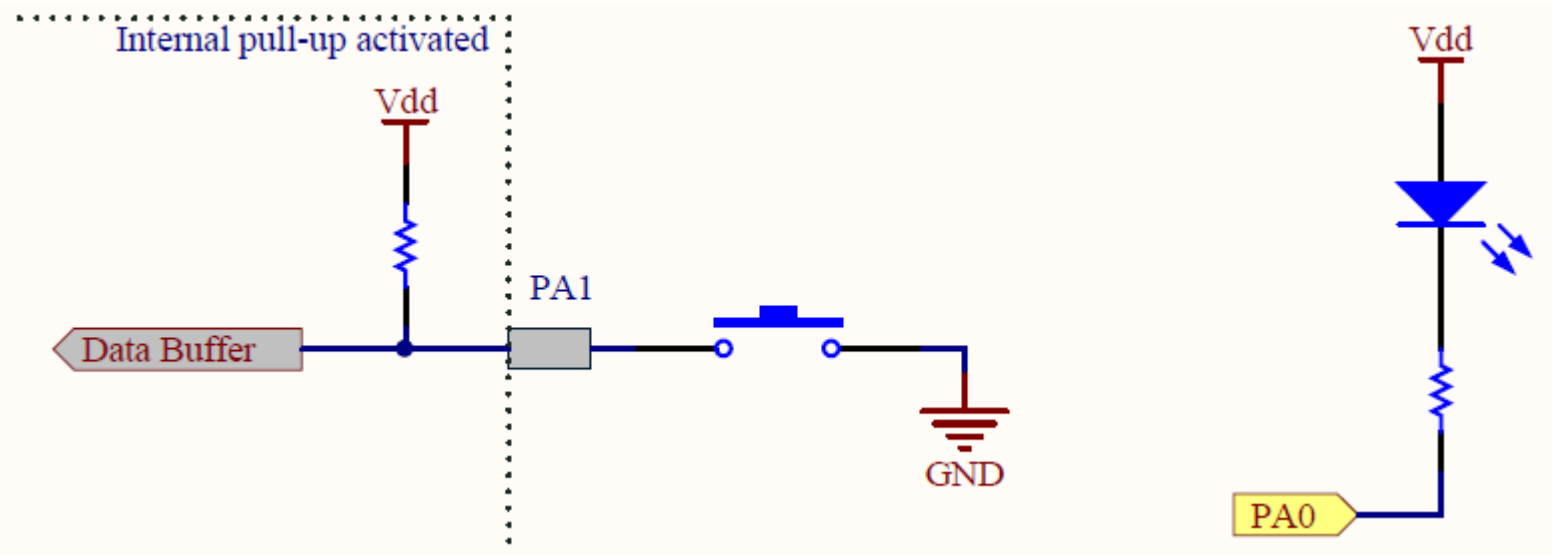
- Wandelt analoge Eingangssignale (z.B. Spannung) in digitale Werte
  Gegenstück: D/A-Wandler



Beispiel: Verarbeiten von Sensor-Eingängen

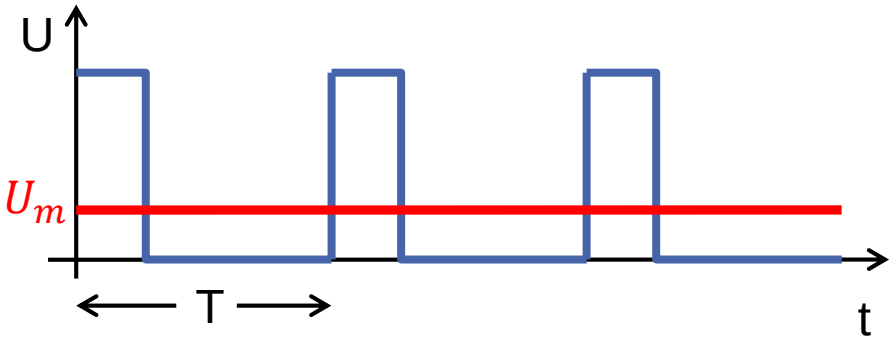
GPIO (General Purpose Input/Output)

- universelle Ein-/Ausgänge, Erkennen/Setzen von High-/Low-Signal
  - Als Eingang geschaltet, kann z.B. der Zustand eines Tasters – offen oder geschlossen – durch den IC abgefragt werden.
  - Als Ausgang, kann z.B. elektrische Elemente wie Leuchtdioden ein- oder ausgeschaltet werden.

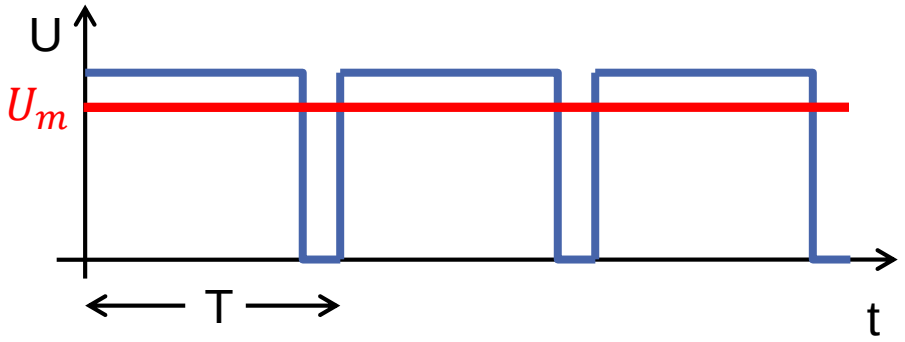


PWM (Pulsweitenmodulation)

- Getacktetes Signal mit definierten An-/Aus-Verhältnis bei fester Grundfrequenz
- Das Verhältnis zwischen der Einschaltzeit t_{ein} und der Periodendauer T = t_{ein} + t_{aus} wird als das Tastverhältnis p bezeichnet (engl. *Duty Cycle*).



25% aktiv

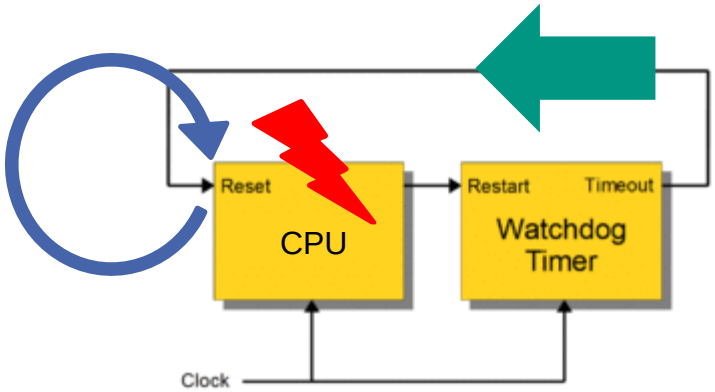


90% aktiv

Beispiel: Motorsteuerung, LED-Dimmer

Watchdog

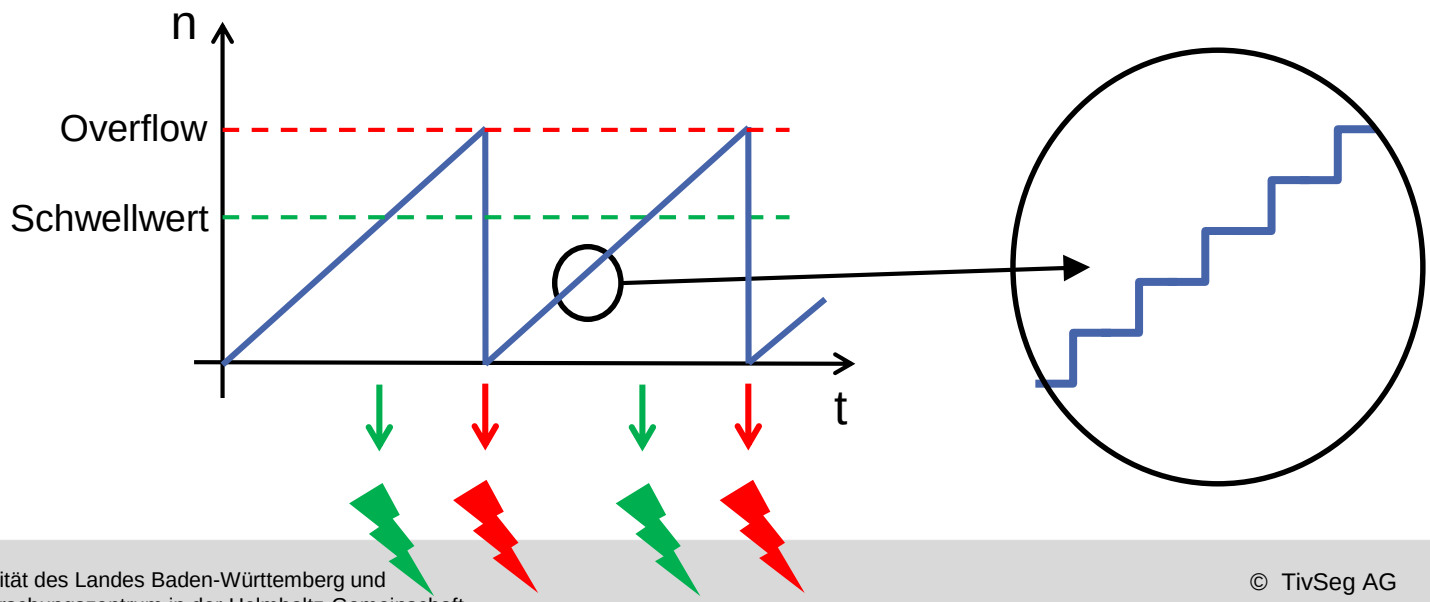
- Ein Watchdog ist eine Schaltung (extern oder im Mikrocontroller integriert), die bei einem Programmabsturz einen Reset auslöst, damit der Mikrocontroller seine Aufgabe wieder erledigen kann-Reset!
- Technisch wird das so realisiert, dass es einen Zeitgeber (z. B. Timer) gibt, den man regelmäßig zurücksetzen muss. Ist der Mikrocontroller abgestürzt, dann kann er das nicht mehr tun und der Watchdog löst den Reset aus.



Timer (Counter)

- Ein Timer ist ein bestimmtes Register im Mikrocontroller, das hardwaregesteuert fortlaufend um 1 erhöht (oder verringert) wird (**inkrementieren/ dekrementieren**)
- Mit ihrer Hilfe ist es möglich, in regelmäßigen Zeitabständen Aktionen zu veranlassen: Interrupt, Signal an einem Pin ausgeben

Bsp.: Quarzoszillator Systemtakt mit 4 MHz
8-Bit Timer: 0~ 2^8-1= 255
4000000 / 256 = 15625 Overflows in einer Sekunde



Inhalt

Register

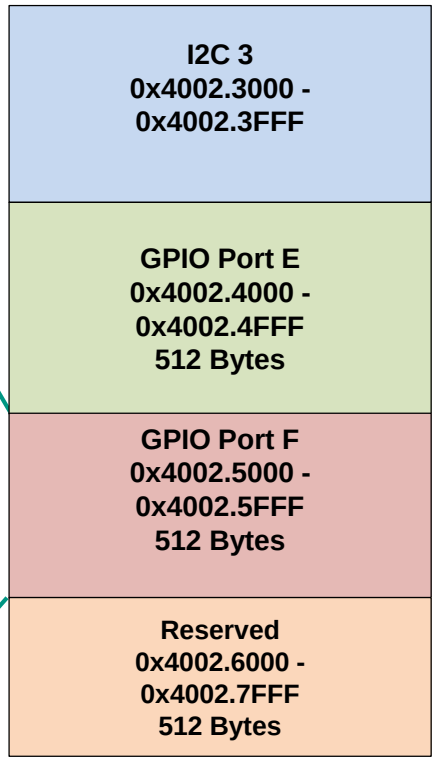
- Speicherbereich, der direkt mit der eigentlichen Recheneinheit verbunden ist und die unmittelbaren Operanden und Ergebnisse aller Berechnungen aufnimmt (flüchtig)
- Register sind in der Regel höchstens so groß wie die Wortgröße des Prozessorkerns (8,16, 32, 64 Bit).
- Die Gesamtheit aller Register bezeichnet man als dessen Registersatz.
- Verschiedene Register dienen zum Zwischenspeichern von Befehlen, Speicheradressen, Rechenoperanden usw.
- Adresse für Register setzt sich zusammen aus:
  - Basisadresse+ Offsets
  - Bsp. GPIO Port F Interrupt Mask (IM)
    - Basisadresse GPIO F: 0x40025000
    - Offset IM: 0x410
    - → Adresse: 0x40025410

Register Adressierung

- Adresse für Register setzt sich zusammen aus: Basisadresse+ Offsets. Z.B. für GPIO Port F Interrupt Mask (IM)
  - Basisadresse GPIO F: 0x40025000
  - Offset IM: 0x410
  - → Adresse: 0x40025410

Table 10-6. GPIO Register Map

Offset	Name	Type	Reset	Description
0x000	GPIODATA	RW	0x0000.0000	GPIO Data
0x400	GPIODIR	RW	0x0000.0000	GPIO Direction
0x404	GPIOIS	RW	0x0000.0000	GPIO Interrupt Sense
0x408	GPIOIBE	RW	0x0000.0000	GPIO Interrupt Both Edges
0x40C	GPIOIEV	RW	0x0000.0000	GPIO Interrupt Event
0x410	GPIOIM	RW	0x0000.0000	GPIO Interrupt Mask
0x414	GPIORIS	RO	0x0000.0000	GPIO Raw Interrupt Status
0x418	GPIOMIS	RO	0x0000.0000	GPIO Masked Interrupt Status
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear
0x420	GPIOAFSEL	RW	-	GPIO Alternate Function Select
0x500	GPIODR2R	RW	0x0000.00FF	GPIO 2-mA Drive Select
0x504	GPIODR4R	RW	0x0000.0000	GPIO 4-mA Drive Select
0x508	GPIODR8R	RW	0x0000.0000	GPIO 8-mA Drive Select
0x50C	GPIOODR	RW	0x0000.0000	GPIO Open Drain Select



Zeiger (Pointer)

- Speichern die Adressen von Speicherzellen
- Deklaration durch Anhängen von ***** an den Variablentyp
  - int a = 2; //normale Variable
  - int* b = &a; // Adressvariable, Operator & bedeutet „Adresse von“
  - *b += 3; // Dereferenzierung:* vor Adressvariable, wird zum Inhalt, welcher an dieser Adresse gespeichert ist
- Zeiger auf Register
  - volatile unsigned int* address_ADC = (unsigned int*) 0xFFFF1150;
  - *address_ADC = 0b10000;

„**volatile** zwingt den Compiler den Wert an der entsprechenden Speicheradresse bei jedem Zugriff erneut zu lesen bzw. zu schreiben. Dies ist nötig, da sich der Register-Wert, der über die Adresse aufgelöst wird, unabhängig vom Programm ändern kann. Im konkreten Fall ändert der ADC sein Status-Register zur Laufzeit unabhängig vom Programm (d.h. rein Hardware-gesteuert), sobald z.B. eine Wandlung abgeschlossen ist.“

Bsp.: Zeiger (Pointer) in Register

- Adresse für GPIO Port F Interrupt Mask (IM)

```
// Zeiger auf den GPIO
const char* GPIOF_ADDRESS = ( char* ) 0x40025000;
// Offset zum Interrupt Mask Register des GPIO Port F
const unsigned int OFFSET_IM = 0x0410;
// Zeiger auf den GPIO
volatile unsigned int* GPIOF_IM_ADDRESS = ( unsigned
int* )(GPIOF_ADDRESS + OFFSET_IM);
```

⇒**Registeradresse = 0x40025410**

Rechenoperationen: bitweise logisch

- `int a = 0b1010;`
  `a &= ~(0b1100); // ~ entspricht logischem NICHT`

```
  1010
 &0011
 ------
  0010
```

- `#define GPIO_PORTF_DATA_R (*((volatile unsigned long *)0x400253FC))`
  `unsigned long SW1; // input aus PF4`
  `SW1 = GPIO_PORTF_DATA_R&0x10; // lesen PF4 in SW1`

```
  10101111
 &00010000
 ---------
  00000000
```

Der #define-Befehl wird benutzt, um Ersetzungen in der ganzen Datei, in der er steht, vorzunehmen. Der Compiler geht also die ganze Datei durch und ersetzt jedes Vorkommen des Makronamen durch dessen Ersatzbefehl

Rechenoperationen: Bit shifting


```
int a = ( 1 << 5 ); // 1
int b = ( a >> 6 ); // 2
```

0 0 0 0 0 0 0 1

a = 0 0 1 0 0 0 0 0

b = 0 0 0 0 0 0 0 0

```
// Setzen der Bits 1,2 und 3
// Binär 00001110 = Hexadezimal 0x0E
GPIO_PORTF_DIR_R = 0x0E;     /* direkte Zuweisung - unübersichtlich */
/* Ausführliche Schreibweise: identische Funktionalität, mehr Tipparbeit, aber übersichtlicher und selbsterklärend: */
GPIO_PORTF_DIR_R = (1 << 1)|(1 << 2)|(1 << 3);
```

0 0 0 0 0 0 1 0

0 0 0 0 1 0 0

0 0 0 1 0 0 0

Zusammenfassung: Zugriff auf Register

- Adressierung eines Registers
  - volatile unsigned int* address_OVR = (unsigned int*) 0xFFFF1150;
- Wert in Register schreiben
  - *address_REG = 0xFF;
- Einzelnes Bit setzen
  - *address_REG |= (1 << 5);
- Einzelnes Bit löschen
  - *address_REG &= ~(1 << 5);
- Alternativ: Verwendung bitmacros.h

Verwendung bitmacros.h

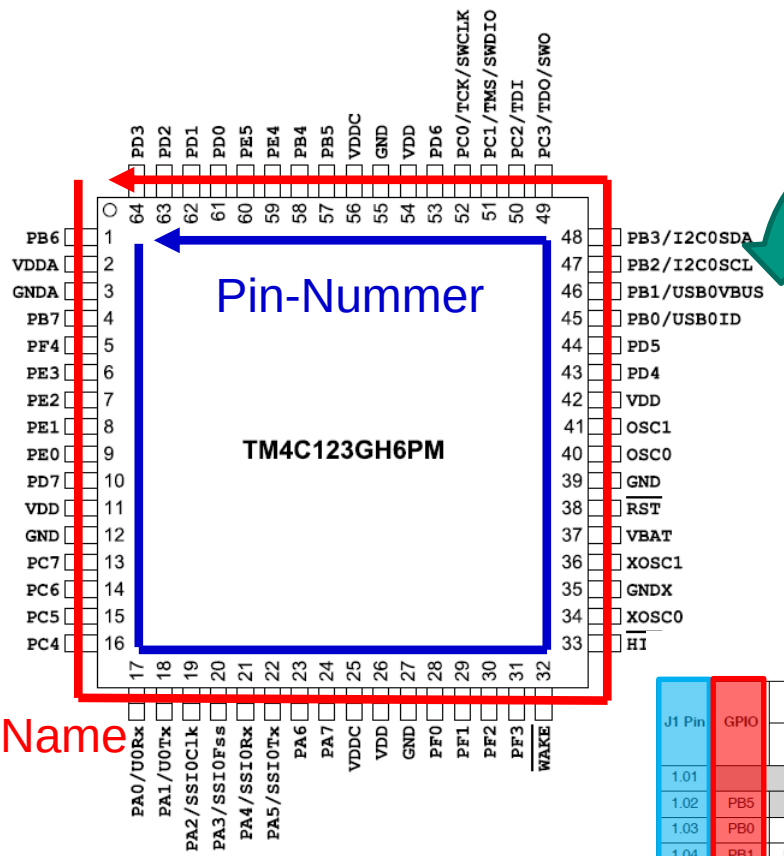
- SET_BIT(*Adresse, Bitposition);
- CLEAR_BIT(*Adresse, Bitposition);
- SET_BITS(*Adresse, (Bitmuster << Bitposition));
- CLEAR_BITS(*Adresse, (Bitmuster << Bitposition));
- BIT_IS_SET(*Adresse, Bitposition);
- BIT_IS_CLEAR(*Adresse, Bitposition);
- HWACC(Adresse)

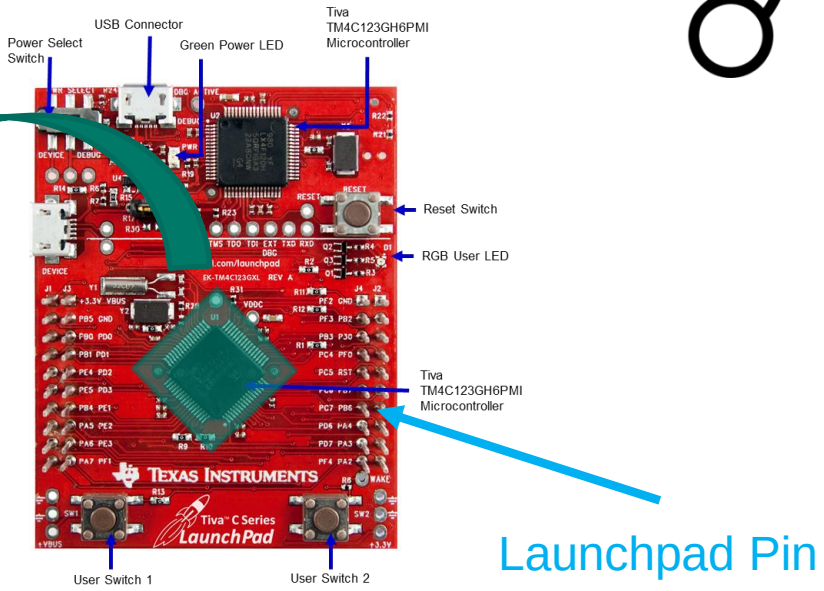
bitmacros.h

```
1 #ifndef BITMACROS_H
2 #define BITMACROS_H
3
4
5 /* set bit */
6 #define SET_BIT(var, bit) ((var) |= (1u << (bit)))
7
8 /* set bits */
9 #define SET_BITS(var, bits) ((var) |= (bits))
10
11 /* clear bit */
12 #define CLEAR_BIT(var, bit) ((var) &= (unsigned)~(1u << (bit)))
13
14 /* clear bits */
15 #define CLEAR_BITS(var, bits) ((var) &= (unsigned)~(bits))
16
17 /* bit is set/clear */
18 #define BIT_IS_SET(var, bit) ((var) & (1u << (bit)))
19 #define BIT_IS_CLEAR(var, bit) !BIT_IS_SET(var, bit)
20
21 /* Use these macros for direct access to HW-Registers */
22 #define HWACC(x)
23         (*((volatile unsigned int *)(x)))
24
25 /* Direct acces via BitBanding Region */
26 //#define HWREGBITW(x, b)
27 //        HWREG(((unsigned int)(x) & 0xF0000000) | 0x02000000 |
28 //              (((unsigned int)(x) & 0x000FFFFF) << 5) | ((b) << 2))
29
30 #endif /* BITMACROS_H */
```

Inhalt

Pin-Nummer, Pin-Name, Pin-Funktion, GPIO-Port





GPIO port :
Port A [0:7], Port B [0:7], Port C [0:7], Port D [0:7], Port E [0:7], Port F [0:4]

TM4C123G LaunchPad User Guide.pdf

Table 2-3. J1 Connector[1]

J1 Pin	GPIO	Analog Function / GPIO AMSEL	On-board Function	Tiva C Series MCU Pin	GPIOPCTL Register Setting 1	2	3	4	5	6	7	8	9	14	15
1.01					3.3 V										
1.02	PB5	AIN11	–	57	–	SSI2Fss	–	M0PWM3	–	–	T1CCP1	CAN0Tx	–	–	–
1.03	PB0	USB0ID	–	45	U1Rx	–	–	–	–	–	T2CCP0	–	–	–	–
1.04	PB1	USB0VBUS	–	46	U1Tx	–	–	–	–	–	T2CCP1	–	–	–	–
1.05	PE4	AIN9	–	59	U5Rx	–	I2C2SCL	M0PWM4	M1PWM2	–	–	CAN0Rx	–	–	–
1.06	PE5	AIN8	–	60	U5Tx	–	I2C2SDA	M0PWM5	M1PWM3	–	–	CAN0Tx	–	–	–
1.07	PB4	AIN10	–	58	–	SSI2Clk	–	M0PWM2	–	–	T1CCP0	CAN0Rx	–	–	–
1.08	PA5	–	–	22	–	SSI0Tx	–	–	–	–	–	–	–	–	–
1.09	PA6	–	–	23	–	–	I2C1SCL	–	M1PWM2	–	–	–	–	–	–
1.10	PA7	–	–	24	–	–	I2C1SDA	–	M1PWM3	–	–	–	–	–	–

(1) Shaded cells indicate configuration for compatibility with the MSP430 LaunchPad.

Tiva C Series TM4C123G LaunchPad Pin Map

Table 2-3. J1 Connector[1]

J1 Pin	GPIO	Analog Function	On-board Function	Tiva C Series MCU Pin	GPIOPCTL Register Setting										
		GPIO AMSEL			1	2	3	4	5	6	7	8	9	14	15
1.01	3.3 V														
1.02	PB5	AIN11	–	57	–	SSI2Fss	–	M0PWM3	–	–	T1CCP1	CAN0Tx	–	–	–
1.03	PB0	USB0ID	–	45	U1Rx	–	–	–	–	–	T2CCP0	–	–	–	–
1.04	PB1	USB0VBUS	–	46	U1Tx	–	–	–	–	–	T2CCP1	–	–	–	–
1.05	PE4	AIN9	–	59	U5Rx	–	I2C2SCL	M0PWM4	M1PWM2	–	–	CAN0Rx	–	–	–
1.06	PE5	AIN8	–	60	U5Tx	–	I2C2SDA	M0PWM5	M1PWM3	–	–	CAN0Tx	–	–	–
1.07	PB4	AIN10	–	58	–	SSI2Clk	–	M0PWM2	–	–	T1CCP0	CAN0Rx	–	–	–
1.08	PA5	–	–	22	–	SSI0Tx	–	–	–	–	–	–	–	–	–
1.09	PA6	–	–	23	–	–	I2C1SCL	–	M1PWM2	–	–	–	–	–	–
1.10	PA7	–	–	24	–	–	I2C1SDA	–	M1PWM3	–	–	–	–	–	–

[1] Shaded cells indicate configuration for compatibility with the MSP430 LaunchPad.

TM4C123G LaunchPad User Guide.pdf

Table 2-4. J2 Connector[1]

J2 Pin	GPIO	Analog Function	On-board Function	Tiva C Series MCU Pin	GPIOPCTL Register Setting										
		GPIO AMSEL			1	2	3	4	5	6	7	8	9	14	15
2.01	GND														
2.02	PB2	–	–	47	–	–	I2C0SCL	–	–	–	T3CCP0	–	–	–	–
2.03	PE0	AIN3	–	9	U7Rx	–	–	–	–	–	–	–	–	–	–
2.04	PF0	–	USR_SW2/ WAKE (R1)	28	U1RTS	SSI1Rx	CAN0Rx	–	M1PWM4	PhA0	T0CCP0	NMI	C0o	–	–
2.05	RESET														
2.06	PB7	–	–	4	–	SSI2Tx	–	M0PWM1	–	–	T0CCP1	–	–	–	–
	PD1	AIN6	Connected for MSP430 Compatibility (R10)	62	SSI3Fss	SSI1Fss	I2C3SDA	M0PWM7	M1PWM1	–	WT2CCP1	–	–	–	–
2.07	PB6	–	–	1	–	SSI2Rx	–	M0PWM0	–	–	T0CCP0	–	–	–	–
	PD0	AIN7	Connected for MSP430 Compatibility (R9)	61	SSI3Clk	SSI1Clk	I2C3SCL	M0PWM6	M1PWM0	–	WT2CCP0	–	–	–	–
2.08	PA4	–	–	21	–	SSI0Rx	–	–	–	–	–	–	–	–	–
2.09	PA3	–	–	20	–	SSI0Fss	–	–	–	–	–	–	–	–	–
2.10	PA2	–	–	19	–	SSI0Clk	–	–	–	–	–	–	–	–	–

[1] Shaded cells indicate configuration for compatibility with the MSP430 LaunchPad.

Table 2-5. J3 Connector[1]

J3 Pin	GPIO	Analog Function	On-board Function	Tiva C Series MCU Pin	GPIOPCTL Register Setting										
		GPIO AMSEL			1	2	3	4	5	6	7	8	9	14	15
3.01	5.0 V														
3.02	GND														
3.03	PD0	AIN7	–	61	SSI3Clk	SSI1Clk	I2C3SCL	M0PWM6	M1PWM0	–	WT2CCP0	–	–	–	–
	PB6	–	Connected for MSP430 Compatibility (R9)	1	–	SSI2Rx	–	M0PWM0		–	T0CCP0	–	–	–	–
3.04	PD1	AIN6	–	92	SSI3Fss	SSI1Fss	I2C3SDA	M0PWM7	M1PWM1	–	WT2CCP1	–	–	–	–
	PB7	–	Connected for MSP430 Compatibility (R10)	4	–	SSI2Tx	–	M0PWM1	–	–	T0CCP1	–	–	–	–
3.05	PD2	AIN5		63	SSI3Rx	SSI1Rx	–	M0FAULT0	–	–	WT3CCP0	USB0EPEN			
3.06	PD3	AIN4	–	64	SSI3Tx	SSI1Tx	–	–	–	–	WT3CCP1	USB0PFLT	–	–	–
3.07	PE1	AIN2	–	8	U7Tx	–	–	–	–	–		–	–	–	–
3.08	PE2	AIN1	–	7	–	–	–	–	–	–	–	–	–	–	–
3.09	PE3	AIN0	–	6	–	–	–	–	–	–	–	–	–	–	–
3.10	PF1	–	–	29	U1CTS	SSI1Tx	–	–	M1PWM5	–	T0CCP1	–	C1o	TRD1	–

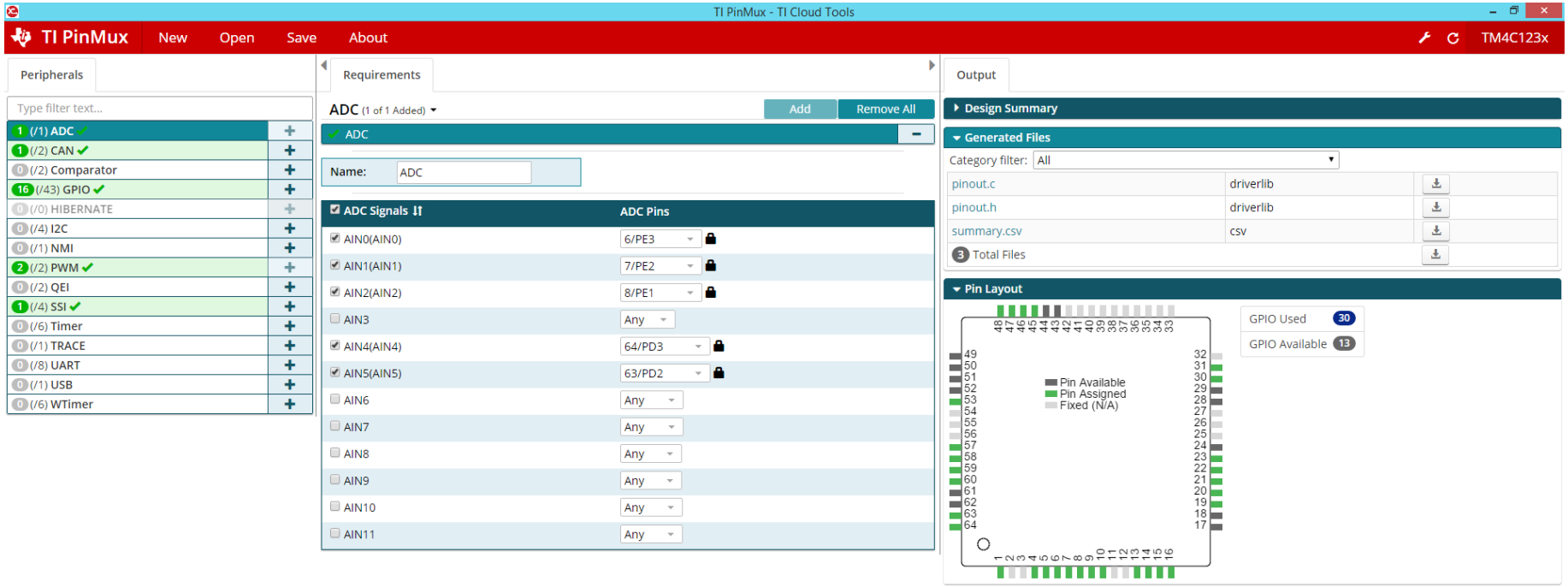
[1] Shaded cells indicate configuration for compatibility with the MSP430 LaunchPad.

Table 2-6. J4 Connector

J4 Pin	GPIO	Analog Function	On-board Function	Tiva C Series MCU Pin	GPIOPCTL Register Setting										
		GPIO AMSEL			1	2	3	4	5	6	7	8	9	14	15
4.01	PF2	–	Blue LED (R11)	30	–	SSI1Clk	–	M0FAULT0	M1PWM6	–	T1CCP0	–	–	–	TRD0
4.02	PF3	–	Green LED (R12)	31	–	SSI1Fss	CAN0Tx	–	M1PWM7	–	T1CCP1	–	–	–	TRCLK
4.03	PB3	–	–	48	–	–	I2C0SDA	–	–	–	T3CCP1	–	–	–	–
4.04	PC4	C1–	–	16	U4Rx	U1Rx	–	M0PWM6	–	IDX1	WT0CCP0	U1RTS	–	–	–
4.05	PC5	C1+	–	15	U4Tx	U1Tx	–	M0PWM7	–	PhA1	WT0CCP1	U1CTS	–	–	–
4.06	PC6	C0+	–	14	U3Rx	–	–	–	–	PhB1	WT1CCP0	USB0EPEN	–	–	–
4.07	PC7	C0–	–	13	U3Tx	–	–	–	–	–	WT1CCP1	USB0PFLT	–	–	–
4.08	PD6	–	–	53	U2Rx	–	–	–	–	PhA0	WT5CCP0	–	–	–	–
4.09	PD7	–	–	10	U2Tx	–	–	–	–	PhB0	WT5CCP1	NMI	–	–	–
4.10	PF4	–	USR_SW1 (R13)	5	–	–	–	–	M1FAULT0	IDX0	T2CCP0	USB0EPEN	–	–	–

Tools für Verwaltung Peripherie: TI PinMux

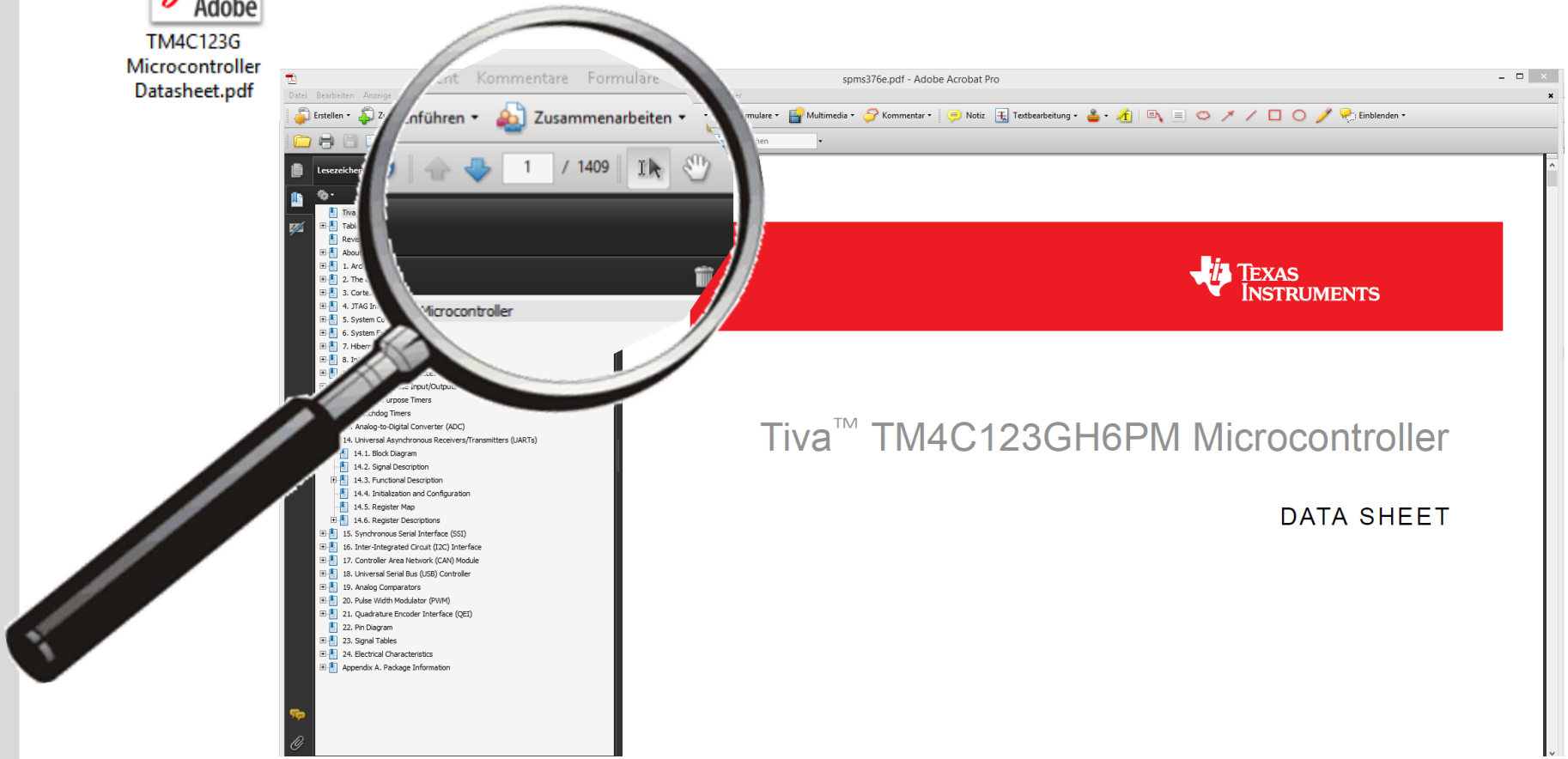
- Graphisch konfigurieren Pin Out
- Automatisch generieren Sources und Header Files für IDE



Download: http://processors.wiki.ti.com/index.php/TI_PinMux_Tool

Umgang mit Datenblättern

TM4C123G Microcontroller Datasheet.pdf



Vor der Programmierung: Umgang mit Datenblättern

- Recherchieren für einen Tiva™ TM4C123GH6PM Microcontroller
  - Basisadressen für A/D Wandler, PWM, GPIO Port F
  - zu jedem Peripheriegerät die Adresse für das Interrupt-Enable-Register (IER) oder Interrupt-Mask-Register (IM) angeben

Start	End	Description
Peripherals		
0x4002.0000	0x4002.0FFF	I²C 0
0x4002.1000	0x4002.1FFF	I²C 1
0x4002.2000	0x4002.2FFF	I²C 2
0x4002.3000	0x4002.3FFF	I²C 3
0x4002.4000	0x4002.4FFF	GPIO Port E
0x4002.5000	0x4002.5FFF	GPIO Port F
0x4002.6000	0x4002.7FFF	Reserved
0x4002.8000	0x4002.8FFF	PWM 0
0x4002.9000	0x4002.9FFF	PWM 1
0x4002.A000	0x4002.BFFF	Reserved
0x4002.C000	0x4002.CFFF	QEI0
0x4002.D000	0x4002.DFFF	QEI1
0x4002.E000	0x4002.FFFF	Reserved
0x4003.0000	0x4003.0FFF	16/32-bit Timer 0
0x4003.1000	0x4003.1FFF	16/32-bit Timer 1

Seite 92~94

Table 10-6. GPIO Register Map

Offset	Name	Type	Reset	Description
0x000	GPIODATA	RW	0x0000.0000	GPIO Data
0x400	GPIODIR	RW	0x0000.0000	GPIO Direction
0x404	GPIOIS	RW	0x0000.0000	GPIO Interrupt Sense
0x408	GPIOIBE	RW	0x0000.0000	GPIO Interrupt Both Edges
0x40C	GPIOIEV	RW	0x0000.0000	GPIO Interrupt Event
0x410	GPIOIM	RW	0x0000.0000	GPIO Interrupt Mask
0x414	GPIORIS	RO	0x0000.0000	GPIO Raw Interrupt Status
0x418	GPIOMIS	RO	0x0000.0000	GPIO Masked Interrupt Status
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear
0x420	GPIOAFSEL	RW	-	GPIO Alternate Function Select
0x500	GPIODR2R	RW	0x0000.00FF	GPIO 2-mA Drive Select
0x504	GPIODR4R	RW	0x0000.0000	GPIO 4-mA Drive Select
0x508	GPIODR8R	RW	0x0000.0000	GPIO 8-mA Drive Select
0x50C	GPIOODR	RW	0x0000.0000	GPIO Open Drain Select

Seite 660

Vor der Programmierung: Umgang mit Datenblättern

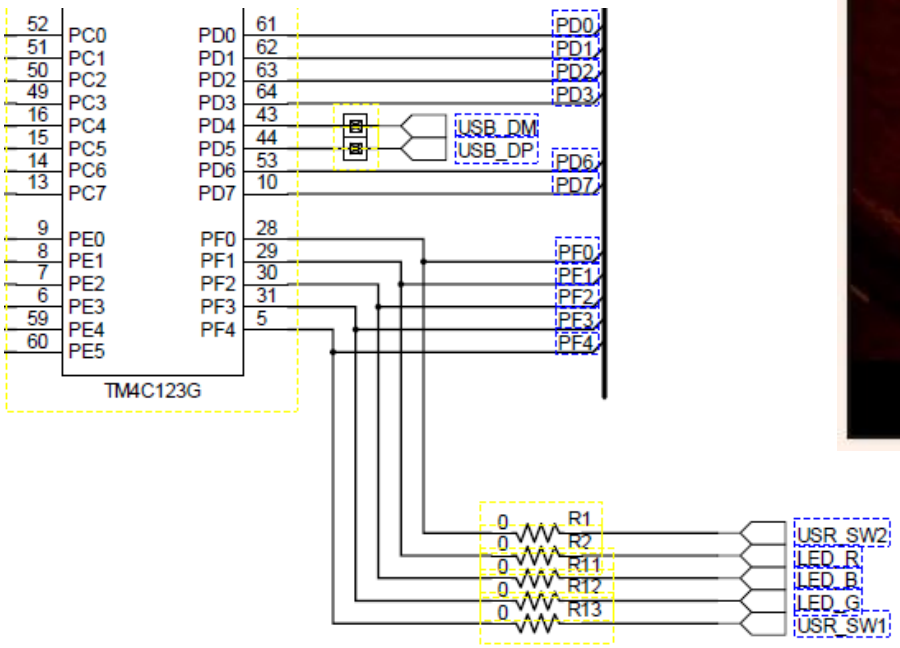
- Recherchieren für einen Tiva™ TM4C123GH6PM Microcontroller
  - Basisadressen für A/D Wandler, PWM, GPIO Port F
  - zu jedem Peripheriegerät die Adresse für das Interrupt-Enable-Register (IER) oder Interrupt-Mask-Register (IM) angeben

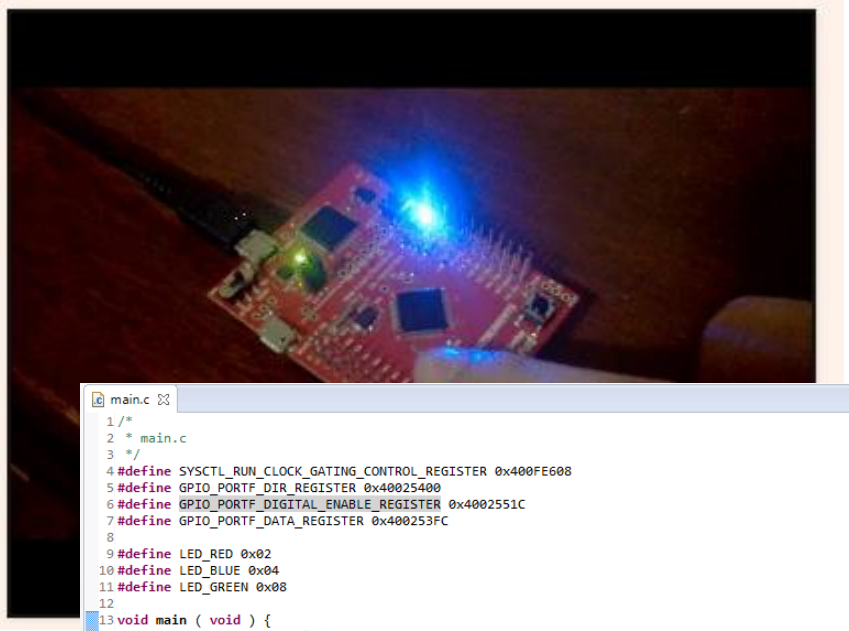
Peripherie	Basisadresse	Offset für IER/IM	Gesamtadresse
GPIO_PORTF_BASE	0x4002.5000	0x410	0x4002.5410
PWM0_BASE	0x4002.8000	0x014	0x4002.8014
ADC0_BASE	0x4003.8000	0x008	0x4003.8008
Timer0_BASE	0x4003.0000	0x018	0x4003.0018
USART0_BASE	0x4000.C000	0x038	0x4000.C038

Inhalt

Blinky: „Hello world!“

- Blinken die RGB LED an Pin PF1,PF2, PF3





main.c

```
/*
 * main.c
 */
#define SYSCTL_RUN_CLOCK_GATING_CONTROL_REGISTER 0x400FE608
#define GPIO_PORTF_DIR_REGISTER 0x40025400
#define GPIO_PORTF_DIGITAL_ENABLE_REGISTER 0x4002551C
#define GPIO_PORTF_DATA_REGISTER 0x400253FC

#define LED_RED 0x02
#define LED_BLUE 0x04
#define LED_GREEN 0x08

void main ( void ) {
    // enable System - clock to GPIO Port F
    *( volatile unsigned int*) SYSCTL_RUN_CLOCK_GATING_CONTROL_REGISTER |= (1u << 5);
    // Set Port F LED Pins (1, 2, 3) as outputs
    *( volatile unsigned int*) GPIO_PORTF_DIR_REGISTER |= ( LED_RED | LED_BLUE | LED_GREEN );
    // Set Pins 1,2,3 digital Enable on Port F
    *( volatile unsigned int*) GPIO_PORTF_DIGITAL_ENABLE_REGISTER |= ( LED_RED | LED_BLUE | LED_GREEN );
    // Clear Port F LED Pins
    *( volatile unsigned int*) GPIO_PORTF_DATA_REGISTER &= ~( LED_RED | LED_BLUE | LED_GREEN );
    // Set Pins High ( LEDs on)
    *( volatile unsigned int*) GPIO_PORTF_DATA_REGISTER |= ( LED_RED | LED_BLUE | LED_GREEN );

    while (42) {
    }
}
```

Blinky: „Hello world!“

- Blinken der LED an Pin PF1,PF2, PF3:
  - Clock einstellen: (Datasheet Seite 219)

Clock Sources

Phase-Lock-Loop (PLL)

Multiplexer

Fundamental Clock Sources

Precision Internal Oscillator (PIOSC)
- 16 MHz ± 3%

Main Oscillator (MOSC) using...
- An external single-ended clock source
- An external crystal

Internal 30 kHz Oscillator
- 30 kHz ± 50%
- Intended for use during Deep-Sleep power-saving modes

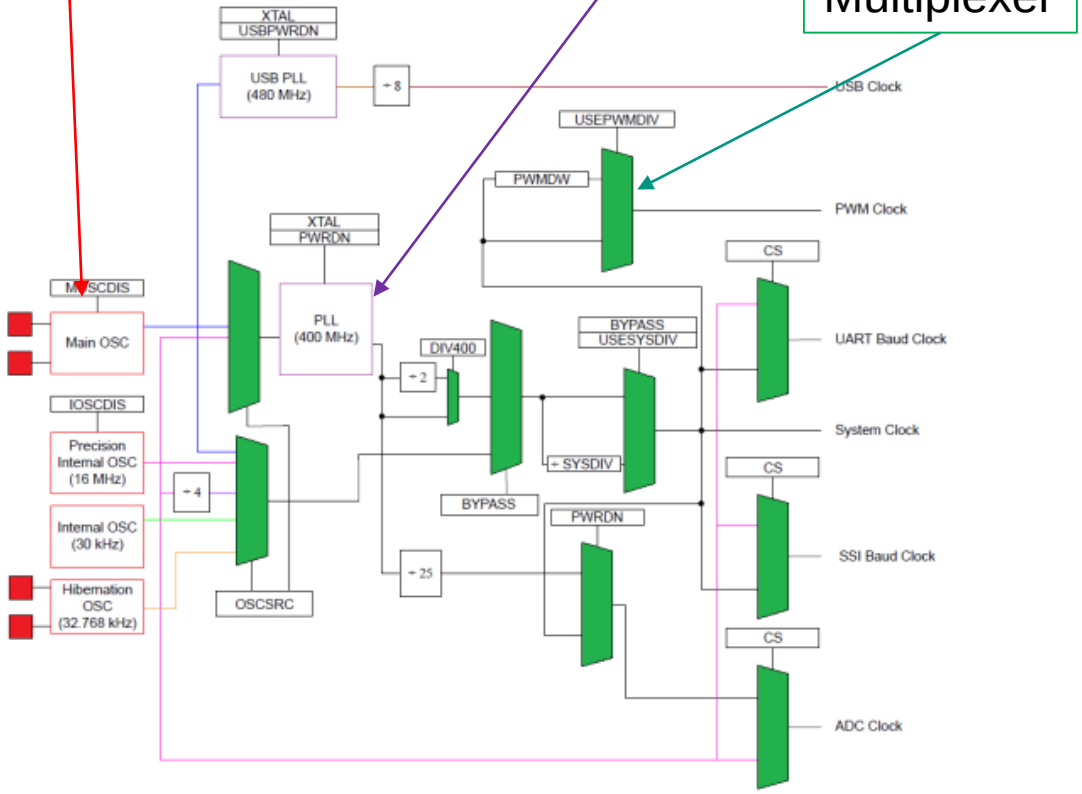
Hibernation Module Clock Source
- 32,768Hz crystal
- Intended to provide the system with a real-time clock source

System (CPU) Clock Sources

The CPU can be driven by any of the fundamental clocks ...
- Internal 16 MHz
- Main
- Internal 30 kHz
- External Real-Time

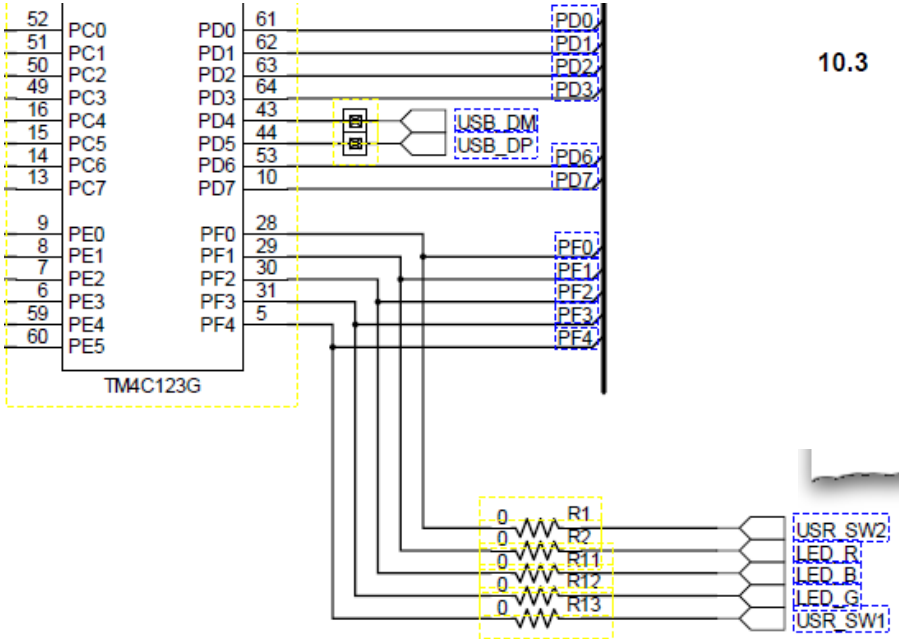
- Plus -
- The internal PLL (400 MHz)
- The internal 16MHz oscillator divided by four (4MHz ± 3%)

Clock Source	Drive PLL?	Used as SysClk?
Internal 16MHz	Yes	Yes
Internal 16Mhz/4	No	Yes
Main Oscillator	Yes	Yes
Internal 30 kHz	No	Yes
Hibernation Module	No	Yes
PLL	-	Yes



Blinky: „Hello world!“

- Blinken der LED an Pin PF1,PF2, PF3:
  - GPIO Initialisierung und Konfiguration (Datasheet Seite 656)
    - Aktivieren Port’s Clock in **RCGCGPIO** (General-Purpose Input/Output Run Mode Clock Gating)
    - Einstellen Pin als Input/Output Pin in **GPIODIR** Register (Direction)
    - Löschen die Bits im Register **GPIOAFSEL** für die Auswahl alternativer Funktionen (AFSEL)
    - Einstellen Pin als digitales Pin durch Schreiben in **GPIODEN** Register (Digital Enable)
    - Schreiben die Datenbits in **GPIODATA** register (Data)



10.3 Initialization and Configuration

The GPIO modules may be accessed via two different memory apertures. The legacy aperture, the Advanced Peripheral Bus (APB), is backwards-compatible with previous devices. The other aperture, the Advanced High-Performance Bus (AHB), offers the same register map but provides better back-to-back access performance than the APB bus. These apertures are mutually exclusive. The aperture enabled for a given GPIO port is controlled by the appropriate bit in the GPIOHBCTL register (see page 258). Note that GPIO can only be accessed through the AHB aperture.

To configure the GPIO pins of a particular port, follow these steps:

1. Enable the clock to the port by setting the appropriate bits in the RCGCGPIO register (see page 340). In addition, the SCGCGPIO and DCGCGPIO registers can be programmed in the same manner to enable clocking in Sleep and Deep-Sleep modes.
2. Set the direction of the GPIO port pins by programming the GPIODIR register. A write of a 1 indicates output and a write of a 0 indicates input.

Blinky: „Hello world!“

- Blinken der LED an Pin PF1,PF2, PF3:
  - Aktivieren Port’s Clock : (Seite 340)

Register 60: General-Purpose Input/Output Run Mode Clock G (RCGCGPIO), offset 0x608

The **RCGCGPIO** register provides software the capability to enable and disable Run mode. When enabled, a module is provided a clock and accesses to modul allowed. When disabled, the clock is disabled to save power and accesses to m generate a bus fault. This register provides the same capability as the legacy **Ru Gating Control Register n RCGCn** registers specifically for the watchdog modu same bit polarity as the corresponding **RCGCn** bits.

General-Purpose Input/Output Run Mode Clock Gating Control (RCGCGPIO)

Base 0x400F.E000
Offset 0x608
Type RW, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19
reserved												
Type 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

15	14	13	12	11	10	9	8	7	6	5	4	3
reserved										R5	R4	R3
Type 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

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	R5	RW	0	GPIO Port F Run Mode Clock Gating Control Value Description 0 GPIO Port F is disabled. 1 Enable and provide a clock to GPIO Port F in Run mode.

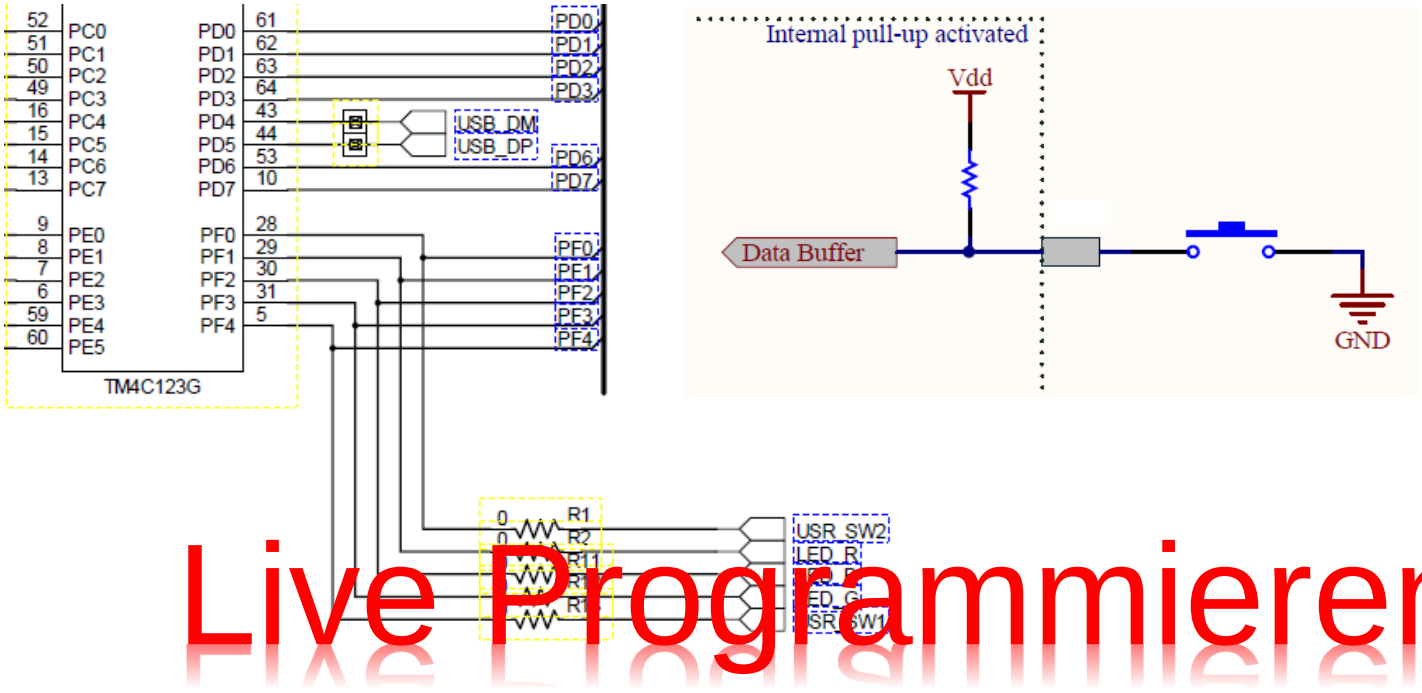
main.c

```
1 /*
2  * main.c
3  */
4 #define SYSCTL_RUN_CLOCK_GATING_CONTROL_REGISTER    *( volatile unsigned int*)0x400FE608
5 #define GPIO_PORTF_DIR_REGISTER                     *( volatile unsigned int*)0x40025400
6 #define GPIO_PORTF_DIGITAL_ENABLE_REGISTER          *( volatile unsigned int*)0x4002551C
7 #define GPIO_PORTF_DATA_REGISTER                    *( volatile unsigned int*)0x400253FC
8
9 #define LED_RED 0x02
10 #define LED_BLUE 0x04
11 #define LED_GREEN 0x08
12
13 void main ( void ) {
14     // enable System - clock to GPIO Port F
15     SYSCTL_RUN_CLOCK_GATING_CONTROL_REGISTER |= (1u << 5);
16     // Set Port F LED Pins (1, 2, 3) as outputs
17     GPIO_PORTF_DIR_REGISTER |= ( LED_RED | LED_BLUE | LED_GREEN );
18     // Set Pins 1,2,3 digital Enable on Port F
19     GPIO_PORTF_DIGITAL_ENABLE_REGISTER |= ( LED_RED | LED_BLUE | LED_GREEN );
20     // Clear Port F LED Pins
21     GPIO_PORTF_DATA_REGISTER &= ~( LED_RED | LED_BLUE | LED_GREEN );
22     // Set Pins High ( LEDs on)
23     GPIO_PORTF_DATA_REGISTER |= ( LED_RED | LED_BLUE | LED_GREEN );
24
25     while (1) {
26     }
27 }
28
```

Live Programmieren

Blinky+ Switch Beispiel

- Blinken der LED an Pin PF1,PF2, PF3:
  - GPIO Initialisierung und Konfiguration zusätzlich
    - Einstellen Pull-Up in **GPIOPUR** (GPIO Pull-Up Select)
    - Unlock Pin in **GPIOLOCK** Register (Lock)
    - Enable Commit in **GPIOCR** Register (Commit)



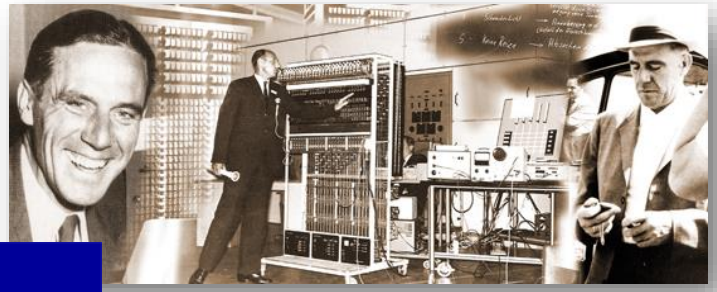
Live Programmieren

Zum Weiterlesen

- https://www.mikrocontroller.net
- http://www.ti.com/tool/ek-tm4c123gxl

Support

- Tutoren
- Betreuer
  - M. Sc. Jijing Yan, yan@kit.edu
  - Prof. Dr.-Ing. E. Sax, Eric.Sax@kit edu



…der Geburtsort der Informatik