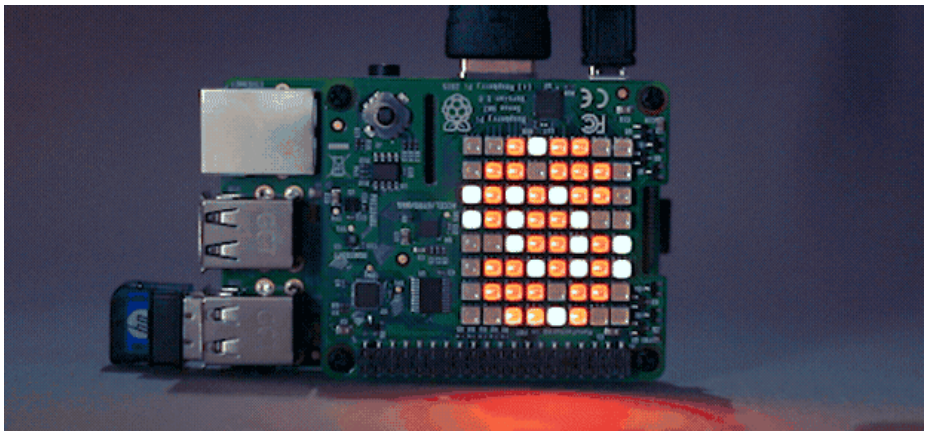
Erweiterte AVR Architekturen

Parameter	Kursinformationen
Veranstaltung:	`Vorlesung Digitale Systeme`
Semester	`Sommersemester 2022`
Hochschule:	`Technische Universität Freiberg`
Inhalte:	`Erweiternde Architekturkonzepte der XMEGA Architektur`
Link auf den GitHub:	https://github.com/TUBAF-IfI-LiaScript/VL_DigitaleSysteme/blob/main/lectures/09_XMEGA.md
Autoren	Sebastian Zug, Karl Fessel & Andrè Dietrich



Ausgangspunkt

Product Family	Pin Count	Program Flash Memory (KB)	SRAM (KB)	Supply Voltage	Speed (MHz) Single Cycle Instruction: MHz = MIPS	Intelligent Analog								Waveform Control			
						ADC (# of bits)	ADC (# of channels)	Comparators	ADC Gain Stage	DAC (# of bits)	Temperature Sensor	Internal Voltage Reference	Zero Cross Detector (ZCD)	8-bit PWM	16-bit PWM	Quadrature Decoder	Waveform Extension (WeX)
ATtiny4/5/9/10	6	0.5–1	0.032	1.8–5.5	12	10[3]	4[3]	✓							2		
ATtiny102/104	8/14	1	0.032	1.8–5.5	12	10	5/8	✓				✓			2		
ATtiny13A	8–20	1	0.064	1.8–5.5	20	10	4	✓							2		
ATtiny20/40	12–20	2/4	0.128/0.256	1.8–5.5	12	10	8/12	✓			✓			2	2		
ATtiny24A/44A/84A	14–20	2–8	Up to 0.512	1.8–5.5	20	10	8	✓	✓		✓	✓		2	2		
ATtiny48/88	28–32	4/8	Up to 0.512	1.8–5.5	16	10	8	✓			✓	✓		1	1		
ATtiny87/167	20–32	8/16	0.512	1.8–5.5	16	10	11	✓			✓	✓		1	2		
ATtiny261A/461A/861A	20–32	2–8	Up to 0.512	1.8–5.5	20	10	11	✓	✓		✓	✓					
ATtiny20x/40x/80x/160x	8–24	2–16	Up to 1	1.8–5.5	20	10	12	✓			✓	✓			2		
ATtiny21x/41x/81x/161x/321x	8–24	2–32	Up to 2	1.8–5.5	20	10	12	✓		8	✓	✓			2		
ATtiny441/841	14–20	4/8	Up to 0.512	1.7–5.5	16	10	12	✓	✓		✓			1	2		
ATtiny2313A	20	2	0.128	1.8–5.5	20	–	–	✓				✓		2	2		
ATmega8A/16A/32A	28–44	8–32	1–2	2.7–5.5	16	10	8	✓						2	1		
ATmega8U2/16U2/32U2	32	8–32	0.5–1	2.7–5.5	16	–	–	✓			✓	✓		4	6		
ATmega16U4/32U4	32	16/32	1/2	2.7–5.5	16	10	12	✓			✓	✓		5			
ATmega48PB/88PB/168PB/328PB	32	4–32	0.5–2	1.8–5.5	20	10	8	✓			✓	✓		4	2/6[6]		
ATmega80x/160x/320x/480x	28–48	8–48	1–6	1.8–5.5	20	10	16	✓			✓	✓		4	3		
ATmega64A/128A	64	64–128	4	2.7–5.5	16	10	8	✓	✓			✓		2	6		
ATmega164PA/324PA/644PA/1284P	44	16–128	1–16	1.8–5.5	20	10	8	✓	✓			✓		4	2/2/4		
ATmega165PA/325PA/645P	44	16–64	1–4	1.8–5.5	16	10	8	✓				✓		4	6		
ATmega169PA/329PA/649P	64	16–64	1–4	1.8–5.5	16	10	8	✓				✓		2	2		
ATmega324PB	44	32	2	1.8–5.5	20	10	8	✓				✓		2	2		
ATmega640/1280/2560/1281/2561	64–100	64–256	8	1.8–5.5	16	10	8/16	✓	✓			✓		4	6/12		
ATmega3290PA/6490P	100	32–64	2–4	1.8–5.5	20	10	8	✓	✓			✓		2	2		
ATmega3250PA/6450P	100	32–64	2–4	1.8–5.5	20	10	8	✓	✓			✓		2	2		
AVR-DA Family	28-64	32-128	4–16	1.8–5.5	24	12	12	✓		10	✓	✓	1-3	9–17	3–6		
ATxmega A1U/A3U/A4U Family	44-100	16–128	2–8	1.6–3.6	32	12	12/16	✓	✓	12	✓	✓			5-8	✓	✓
ATxmega B1/B3 Family	64–100	64–128	4–8	1.6–3.6	32	12	8	✓	✓		✓	✓			2/3	✓	✓
ATxmega C3/D3/C4/D4 Family	44-64	16–384	2–32	1.6–3.6	32	12	12/16	✓	✓		✓	✓			4/5	✓	✓
ATxmega32E5 Family	32	8–32	1–4	1.6–3.6	32	12	16	✓	✓	12	✓	✓			3	✓	✓

Product Family	Timing and Measurements				Logic, Crypto and Math			Safety and Monitoring				Communications						User Interface			System Flexibility					
	Real-Time Counter	8-bit Timer/Counters	12-bit Timer Counter	16-bit Timer/Counter	CCL	MULT	Crypto (AES/DES)	CRC/SCAN	POR	BOD	WDT	USART	USB	I^2C	SPI	IRCOM	Serial Number	QTouch® Technology	QTouch Technology with PTC [2]	LCD	External Bus Interface	DMA Channels	Event System	SleepWalking	Sleep Modes	picoPower® Technology
ATtiny4/5/9/10				1					✓		✓							✓							4	
ATtiny102/104				2					✓		✓	1													4	
ATtiny13A									✓	✓	✓							✓							3	✓
ATtiny20/40		1		1					✓	✓	✓			1	1			✓							4	
ATtiny24A/44A/84A		1		1					✓	✓	✓			1	1			✓							4	✓
ATtiny48/88		1		1					✓	✓	✓			1	1										3	✓
ATtiny87/167		1		1					✓	✓	✓	1[8]		1	2										4	
ATtiny261A/461A/861A		1		1					✓	✓	✓			1	1			✓							4	✓
ATtiny20x/40x/80x/160x	✓			1	✓	✓		✓	✓	✓	✓	1[1]		1	1		✓						✓	✓	3	✓
ATtiny21x/41x/81x/161x/321x	✓		1	1	✓	✓		✓	✓	✓	✓	1[1]		1	1		✓		✓[4]				✓	✓	3	✓
ATtiny441/841		1		2					✓	✓	✓	2		1	1										4	✓
ATtiny2313A		1		1					✓	✓	✓	1		1	2										3	✓
ATmega8A/16A/32A	✓	2		1		✓			✓	✓	✓	1		1	1			✓							5	
ATmega8U2/16U2/32U2	✓	2		3		✓			✓	✓	✓	2	✓	2	2										6	
ATmega16U4/32U4		1		1		✓			✓	✓	✓	1	✓		1										6	
ATmega48PB/88PB/168PB/328PB	✓	2		1/3[6]		✓			✓	✓	✓	1/2[6]		1/2[6]	1/2[6]			✓	✓[6]						6	
ATmega80x/160x/320x/480x	✓			5	✓	✓		✓	✓	✓	✓	4		1	1		✓						✓	✓	3	✓
ATmega64A/128A		2		2		✓			✓	✓	✓	2		1	1			✓							6	
ATmega164PA/324PA/644PA/1284P	✓	2		1/1/2		✓			✓	✓	✓	2		1	1			✓							6	✓
ATmega165PA/325PA/645P	✓	2		3		✓			✓	✓	✓	3		2	2										6	✓
ATmega169PA/329PA/649P	✓	2		1		✓			✓	✓	✓	1		1	1			✓		✓					5	
ATmega324PB	✓	2		1		✓			✓	✓	✓	1		1	1			✓	✓						5	
ATmega640/1280/2560/1281/2561	✓	2		4		✓			✓	✓	✓	2/4		1	1			✓			✓[5]				6	
ATmega3290PA/6490P	✓	2		1		✓			✓	✓	✓	1		1	1			✓		✓					5	
ATmega3250PA/6450P	✓	2		1		✓			✓	✓	✓	1		1	1			✓							5	
AVR-DA Family	✓		1	1–5	✓	✓		✓	✓	✓	✓	3–6		1–2	2	✓	✓		✓				✓	✓	3	✓
ATxmega A1U/A3U/A4U Family	✓			5-8		✓	✓	✓	✓	✓	✓	5-8	✓	2-4	2-4	✓		✓			✓	4	✓		5	✓
ATxmega B1/B3 Family	✓			2/3		✓	✓	✓	✓	✓	✓	1/2	✓	1	1	✓	✓	✓		✓		2	✓		5	✓
ATxmega C3/D3/C4/D4 Family	✓			4/5		✓		✓	✓	✓	✓	2/3	✓(7)	2	2	✓	✓	✓					✓		5	✓
ATxmega32E5 Family	✓			3	✓	✓		✓	✓	✓	✓	2		1	1	✓	✓	✓				4	✓		5	✓

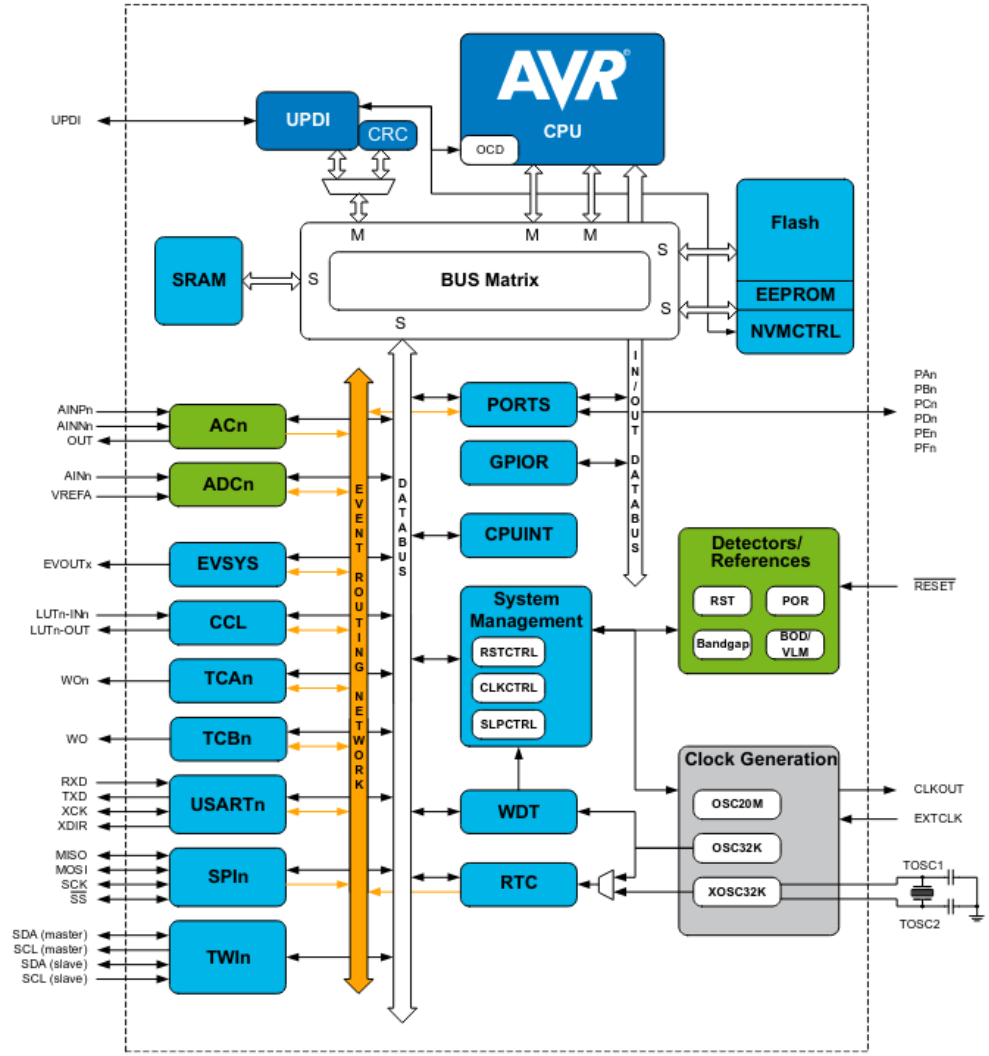
1: LIN port also **2:** Peripheral Touch Controller **3:** Only on the ATtiny5/10 **4:** Not on the ATtiny212/214/412/414/416 **5:** Only on the ATmega1281/2561 **6:** Only on the ATmega328PB **7:** Only on the C3 and C4 **8:** UART only LIN Port also

Speicherstruktur des ATMega2560 [ATmega640] Seite 20

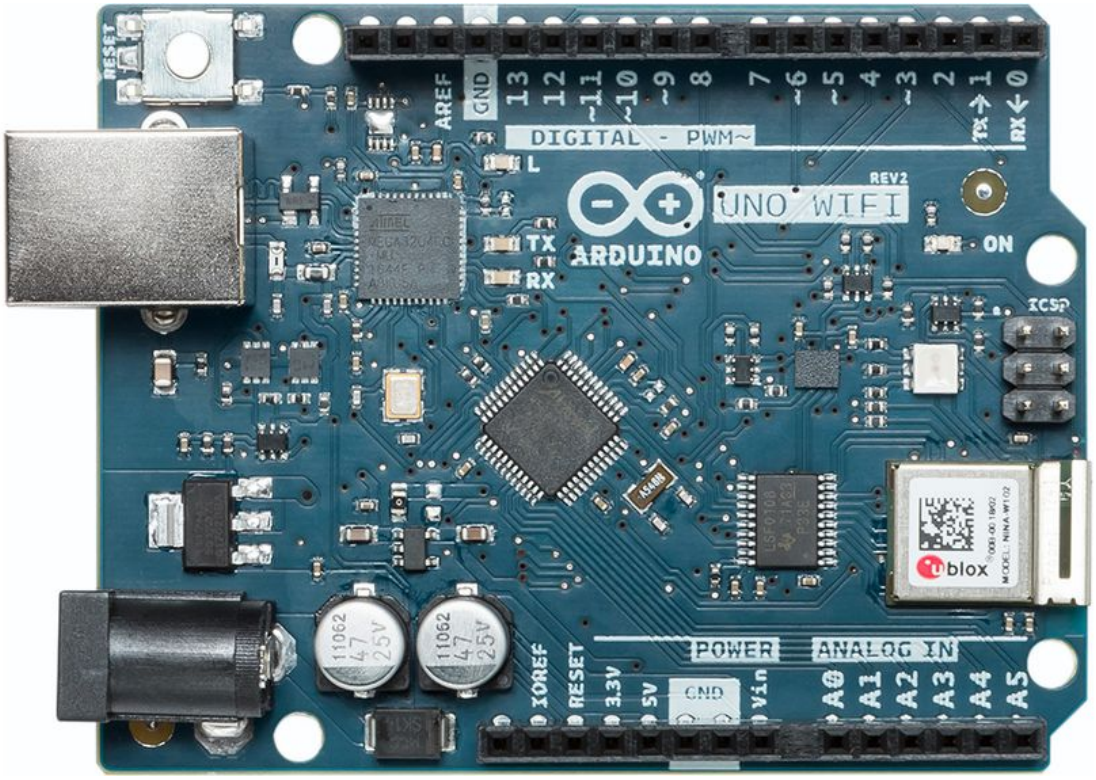
Welche Erweiterungen ergeben sich dabei:

- alle GPIOs können als externe Interrupts genutzt werden
- die Interruptvektortabelle ist nicht mehr fest vordefiniert sondern kann an die Anwendung angepasst werden.
- ein Eventsystem erlaubt die Verknüpfung von peripheren Elementen untereinander, ohne dass die CPU eingreifen muss
- die Taktgeber werden nicht mehr über Fuse-Bits gesetzt sondern können über entsprechende Register konfiguriert werden.
- eine konfigurierbare Logik (CCL) verbindet Eingänge, Perepheriebauteile und Ausgänge mit sequenziellen Schaltwerken
- für den Analog-Digitalwandler stehen 5 interne Referenzspannungen bereit.
- der Controller integriert einen internen Real-Time Oszillator mit 32.768 Hz
- Softwareresets sind über ein eigenes Register möglich.
- UPDI ersetzt die bisherige OneWire Debug Schnittstelle

2. Block Diagram



Speicherstruktur des ATMega2560 [ATmega640] Seite 20



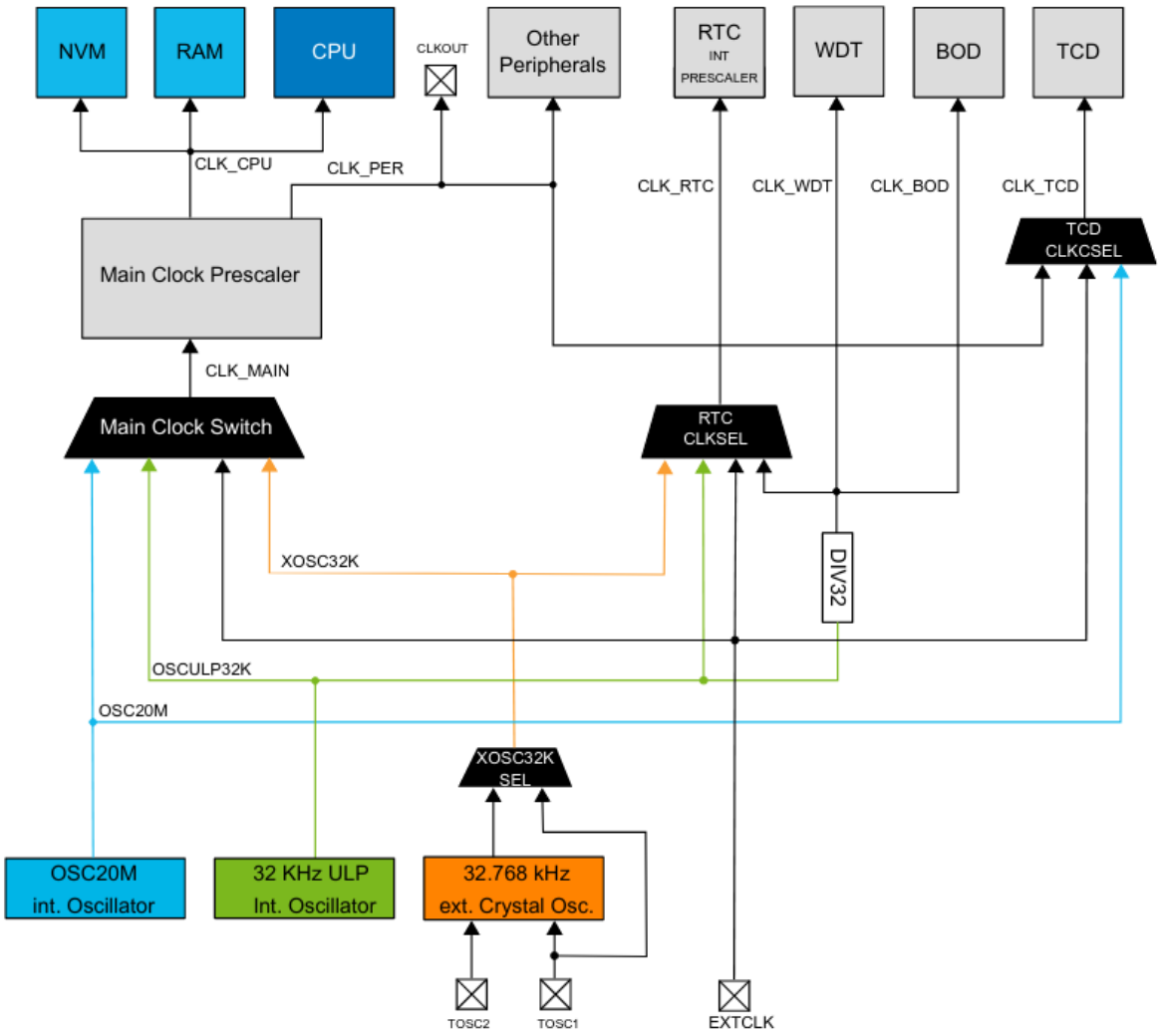
Atmel ATmega4809 auf dem Arduino Uno Wifi Rev. 2

[Microchip4809] Firma Microchip, ATmega4808/4809 Data Sheet, Link

Hardware

10.2.1 Block Diagram - CLKCTRL

Figure 10-1. CLKCTRL Block Diagram



Clock-System des ATmega4809 [Microchip4809] Seite 85

Der Haupttakt versorgt die CPU, den RAM, den Flash, den I/O-Bus und allen am I/O-Bus angeschlossenen Peripheriegeräten und wird vom Taktcontroller vorskaliert und verteilt. Dem asynchronen Takt folgen die Realtime-Clock (RTC), der Watchdog-Timer (WDT), die Brown-out-Detection (BOD) und asynchrone Timer Counter (TCD). Die asynchronen Taktquellen werden über Register in der jeweiligen Peripherie konfiguriert.

Als Taktquellen sind vorgesehen:

- Interne Oszillatoren (16/20 MHz Oszillator, 32KHz Oszillator)
- Externe Ozillatoren (via External Clock Pin, 32.768 kHz Quarz Oszillator)

Die Konfiguration wird über zwei Register `MCLKCTRLA` (Taktressource) und `MCLKCTRLB` (Prescaler) vorgenommen.

[Microchip4809] Firma Microchip, ATmega4808/4809 Data Sheet, Link

Variable Konfiguration von Pin-Belegungen

Ein zentraler Unterschied des bisherig ATmega328 zum ATmega4809 ist die Möglichkeit der variablen Zuordnung bestimmter Funktionalitäten zu einzelnen Pins. Der Port-Multiplexer (`PORTMUX`) kann entweder die Funktionalität von Pins aktivieren oder deaktivieren, oder zwischen Standard- und alternativen Pin-Positionen schalten.

4.1 Multiplexed Signals

TQFP48/ UQFN48	PDIP40(4)	TQFP32/ VQFN32	SSOP28	Pin name(1,2)	Special	ADC0	AC0	USARTn	SPI0	TWI0	TCA0	TCBn	EVSYS	CCL-LUTn
44	33	30	22	PA0	EXTCLK			0,TxD			0-WO0			0-IN0
45	34	31	23	PA1				0,RxD			0-WO1			0-IN1
46	35	32	24	PA2	TWI			0,XCK		SDA(MS)	0-WO2	0-WO	EVOUTA	0-IN2
47	36	1	25	PA3	TWI			0,XDIR		SCL(MS)	0-WO3	1-WO		0-OUT
48	37	2	26	PA4				0,TxD(3)	MOSI		0-WO4			
1	38	3	27	PA5				0,RxD(3)	MISO		0-WO5			
2	39	4	28	PA6				0,XCK(3)	SCK					0-OUT(3)
3	40	5	1	PA7	CLKOUT		OUT	0,XDIR(3)	SS				EVOUTA(3)	
4				PB0				3,TxD			0-WO0(3)			
5				PB1				3,RxD			0-WO1(3)			
6				PB2				3,XCK			0-WO2(3)		EVOUTB	
7				PB3				3,XDIR			0-WO3(3)			
8				PB4				3,TxD(3)			0-WO4(3)	2-WO(3)		
9				PB5				3,RxD(3)			0-WO5(3)	3-WO		
10	1	6	2	PC0				1,TxD	MOSI(3)		0-WO0(3)	2-WO		1-IN0
11	2	7	3	PC1				1,RxD	MISO(3)		0-WO1(3)	3-WO(3)		1-IN1
12	3	8	4	PC2	TWI			1,XCK	SCK(3)	SDA(MS)(3)	0-WO2(3)		EVOUTC	1-IN2
13	4	9	5	PC3	TWI			1,XDIR	SS(3)	SCL(MS)(3)	0-WO3(3)			1-OUT
14	5			VDD										

IO-Multiplexing des ATmega4809 [Microchip4809] Seite 18

Die Grafik zeigt, das zum Beispiel das der USART3 standardmäßig mit PB0 und PB1 verknüpft ist. Durch die Rekonfiguration im `PORTMUX` Register kann diese Zuordnung nach PB4 und PB5 verschoben werden.

> Recherchieren Sie die Beschaltung der Seriellen Schnittstelle des aktuellen Boards. Werden hier die alternativen Pins benutzt? Welche Konfigurationen sind entsprechend zu treffen?

15.2 Register Summary - PORTMUX

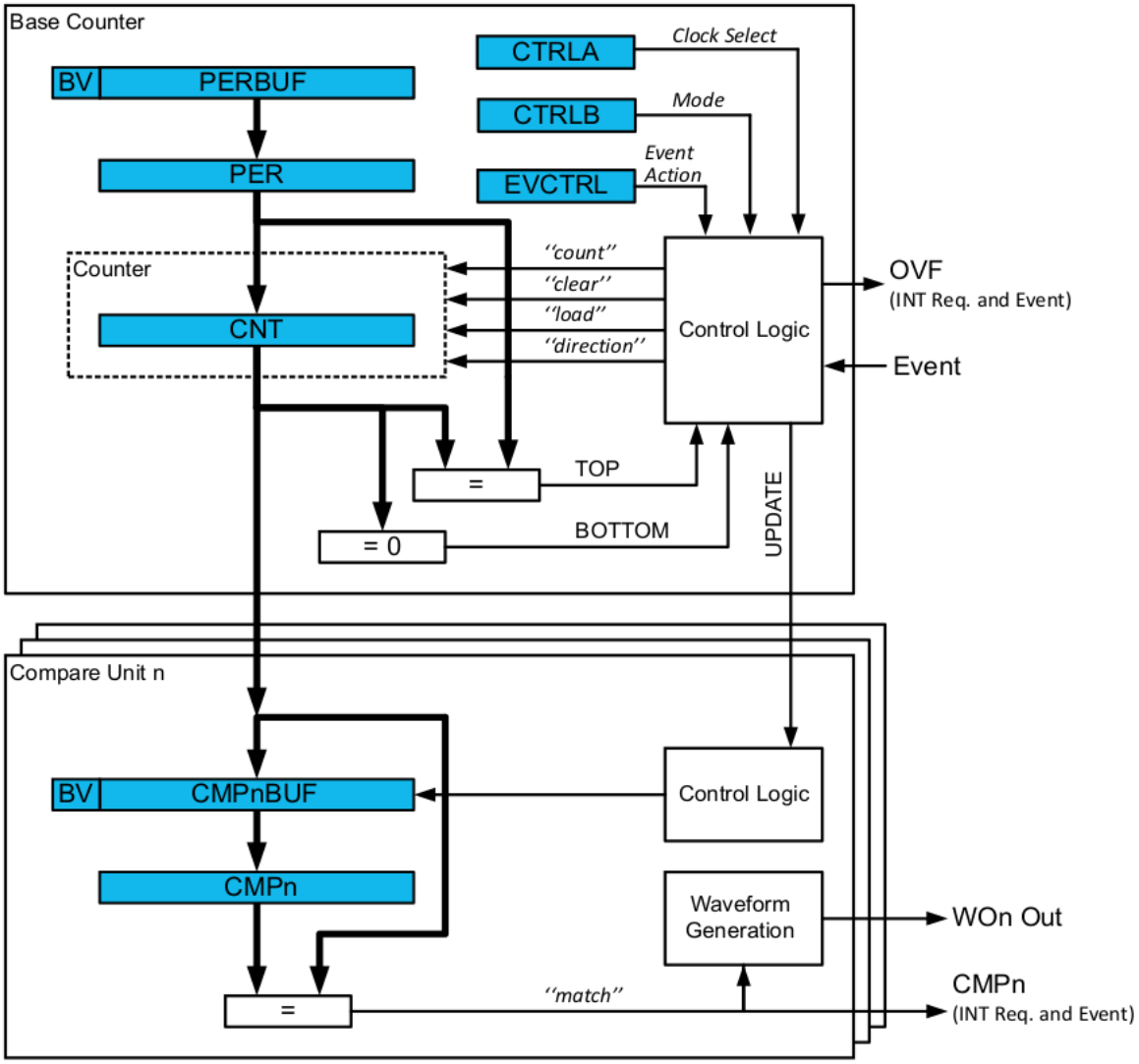
Offset	Name	Bit Pos.								
0x00	EVSYSROUTEA	7:0			EVOUTF	EVOUTE	EVOUTD	EVOUTC	EVOUTB	EVOUTA
0x01	CCLROUTEA	7:0					LUT3	LUT2	LUT1	LUT0
0x02	USARTROUTEA	7:0	USART3[1:0]		USART2[1:0]		USART1[1:0]		USART0[1:0]	
0x03	TWISPIROUTEA	7:0			TWI0[1:0]				SPI0[1:0]	
0x04	TCAROUTEA	7:0						TCA0[2:0]		
0x05	TCBROUTEA	7:0					TCB3	TCB2	TCB1	TCB0

PORTMUX_Register [Microchip4809] Seite 134

Bezeichnung	Bedeutung
`EVOUTx`	Event Output Pin
`LUTn`	Look-Up Tables Output Pins
`USARTn`	
`TWIn`	I2C Schnittstellen Pins
`SPIn`	Serial Peripheral Interface Pins
`TCAn`	Timer Counter A Output
`TCBn`	Timer Counter B Output

Frage: Warum brauchen wir 6 Ausgabekanäle für lediglich 3 Compare-Einheiten?

Figure 20-2. Timer/Counter Block Diagram

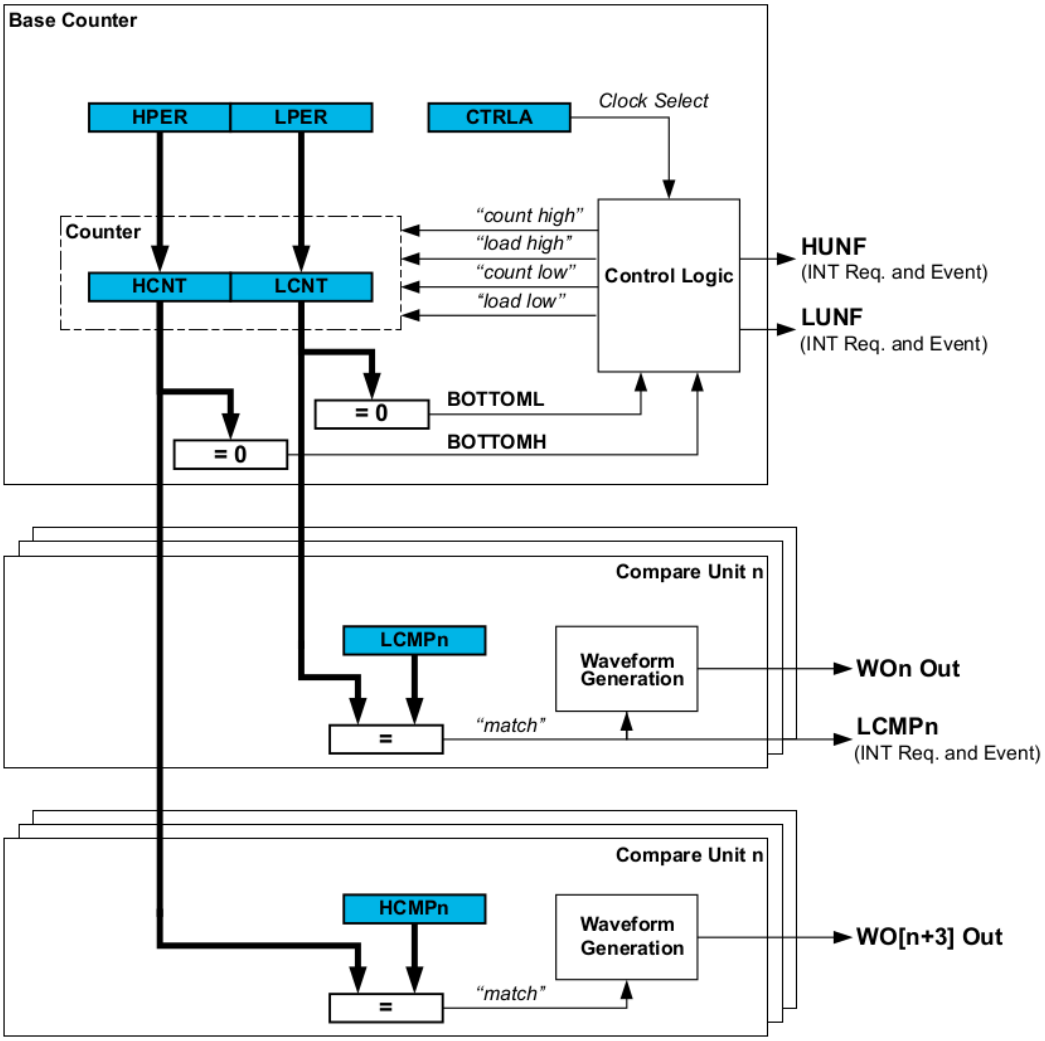


Timer-Counter Modul A des ATmega4809 [Microchip4809] Seite 187

Die 16 Bit Counter des 4809 können in einen 8bit Modus umgeschalten werden.

Block Diagram

Figure 20-13. Timer/Counter Block Diagram Split Mode



8 Bit Modus des Timer-Counter Modul A [Microchip4809] Seite 187

[Microchip4809] Firma Microchip, ATmega4808/4809 Data Sheet, Link

Interrupts

Achtung: Das Interruptsystem des 4809 unterscheidet sich deutlich von dem des ATmega328. Bisher waren wir dort auf eine statisch konfigurierte Priorisierung angewiesen.

Interruptsystem des 4809 [Microchip4809] Seite 111

Die Interrupt-Erzeugung muss global aktiviert werden, indem eine '1' in das Global Interrupt Enable Bit (I) im CPU-Statusregister `CPU.SREG` geschrieben wird. Dieses Bit wird nicht gelöscht, wenn ein Interrupt quittiert wird.

Ablauf der Interruptverarbeitung [Microchip4809] Seite 114

Wenn ein Interrupt aktiviert ist und die Interrupt-Bedingung eintritt, empfängt die CPUINT die Interrupt-Anforderung. Wenn eine Interrupt-Anforderung von der CPUINT bestätigt wird, wird der Programmzähler so gesetzt, dass er auf den Interrupt-Vektor zeigt. Der Interrupt-Vektor ist ein Sprung zum Interrupt-Handler. Nach der Rückkehr vom Interrupt-Handler wird die Programmausführung an der Stelle fortgesetzt, an der sie vor dem Auftreten der Unterbrechung war.

Standardmäßig haben alle Peripheriegeräte die Prioritätsstufe 0. Es ist möglich, eine Interrupt-Anforderung der Stufe 1 (hohe Priorität) zuzuordnen, indem Sie ihre Interrupt-Vektornummer in das `CPUINT.LVL1VEC`-Register schreibt. Diese Interrupt-Anforderung hat dann eine höhere Priorität als die anderen (normal priorisierten) Interrupt-Anforderungen.

Priorität	Level	Quelle
Höchste	*Non Maskable Interrupt* (NMI)	für 4809 nur CRC check
Hohe	Level 1	
Niedrige	Level 0	

Interrupts werden entsprechend ihrer Prioritätsstufe UND ihrer Interruptvektoradresse (vgl. Datenblatt Seite 66) priorisiert. Interrupts der Prioritätsstufe 1 unterbrechen Interrupthandler der Prioritätsstufe 0. Bei Interrupts der Prioritätsstufe 0 wird die Priorität anhand der Interruptvektoradresse ermittelt, wobei die niedrigste Interruptvektoradresse die höchste Interruptpriorität hat.

Interruptsystem des 4809 [Microchip4809] Seite 116

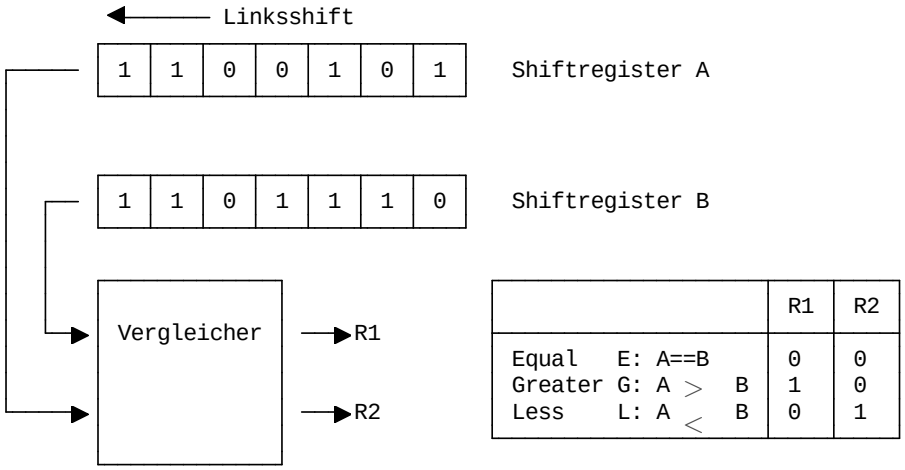
Optional kann für Interrupts der Prioritätsstufe 0 ein Round-Robin-Schema aktiviert werden. Dadurch wird sichergestellt, dass alle Interrupts innerhalb einer bestimmten Zeitspanne bearbeitet werden.

Konfigurierbare Logik

Die konfigurierbare benutzerdefinierte Logik (CCL) ist eine programmierbare Logik-Peripherie, die mit den Geräte Pins an Ereignisse oder an andere interne Peripherie angeschlossen werden kann. Die CCL kann als "Klebelogik" zwischen der Geräteperipherie und externen Geräten dienen.

Exkurs - Schaltwerke

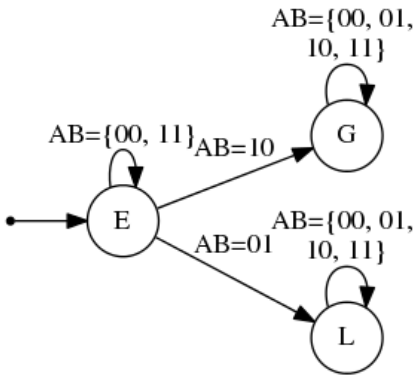
Sequentieller Binärzahlenvergleicher - zwei Zahlenwerte werden sequenziell entsprechend ihren Stellen durch den Vergleicher bewegt und verglichen. Das Schaltwerk speichert das Resultat sobald ein Wert größer als der andere .



	R1	R2
Equal E: A==B	0	0
Greater G: A > B	1	0
Less L: A < B	0	1

1. Schritt: Aufgabenspezifikation, Erstellen eines Zustandsdiagramms

Für die Aufgabe ergibt sich folgender Graph:



Im Beispiel liegt ein Medwedew-Automat vor. Die Zustände werden direkt auf den Ausgang abgebildet.

2. Schritt: Erstellen der Zustandstabelle

Hier wäre eine Zustandstabelle denkbar, die alle Eingangskombinationen mit allen Zuständen zeilenweise verknüpft.

Zustand	A	B	Folgezustand
E	0	0	E
E	0	1	L
E	1	0	G
...			

Eine kompaktere Darstellung fasst die Kombinationen der Eingänge zusammen und ordnet sie den Folgezuständen zu.

aktueller Zustand	AB==00	AB==01	AB==10	AB==11
E	E	L	G	E
G	G	G	G	G
L	L	L	L	L

Schritt 3: Auswahl einer binären Zustandskodierung und Generierung einer binären Zustandstabelle

Insgesamt sind 3 Zustände zu kodieren, entsprechend werden wiederum 2 Flip-Flops benötigt. Dabei wird die Kodierung wie folgt vorgenommen:

Zustand	X	Y
E	0	0
G	0	1
L	1	0

Damit ergibt sich folgende Binäre Zustandstabelle

aktueller Zustand	AB==00	AB==01	AB==10	AB==11
00	00	10	01	00
01	01	01	01	01
10	10	10	10	10

In der traditionellen Darstellung zeigt sich diese wie folgt:

X_t	Y_t	A_t	B_t	X_{t+1}	Y_{t+1}
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	1	0	0
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	D	D
1	1	0	1	D	D
1	1	1	0	D	D
1	1	1	1	D	D

Schritt 4: Auswahl eines Flip-Flop Typs und Ermittlung der für jeden Zustandsübergang benötigten Flip-Flop Ansteuerungen

Wir entscheiden uns für einen D Flip-Flop für die Realisierung. Die entsprechende invertierte Wahrheitstafel haben Sie zwischenzeitlich im Kopf:

$Q(t)$	$Q(t+1)$	D
0	0	0
0	1	1
1	0	0
1	1	1

Damit lässt sich die Zustandsübergangstabelle entsprechend einfach um die zugehörige Eingangsbelegung ergänzen. Für die D-Flip-Flops ist dies einfach eine Kopie der Zustandsspalten.

X_t	Y_t	A_t	B_t	X_{t+1}	Y_{t+1}	DX
0	0	0	0	0	0	0
0	0	0	1	1	0	1
0	0	1	0	0	1	0
0	0	1	1	0	0	0
0	1	0	0	0	1	0
0	1	0	1	0	1	0
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	1	0	1
1	0	0	1	1	0	1
1	0	1	0	1	0	1
1	0	1	1	1	0	1
1	1	0	0	D	D	D
1	1	0	1	D	D	D
1	1	1	0	D	D	D
1	1	1	1	D	D	D

Aufgabe: Lesen Sie die minimale Funktion für `DF` und `DG` ab!

$$DX = X + \overline{Y}\,\overline{A}\,B$$
$$DY = Y + \overline{X}\,A\,\overline{B}$$

Ein weiteres Einführungsbeispiel finden Sie unter Link

Realisierung

Eine Umsetzungsmöglichkeit für Schaltnetze sind die sogenannten PALs (Programmable Array Logic) die bereits aus der Vorlesung Eingebettete Systeme bekannt sind und dort nocheinmal nachgelesen werden können.

Auf der verlinkten Folie wurden die 2 stufigen Schaltfunktionen mit einem programmierbaren `AND` Array vorgestellt.

PAL Schema

PAL16L8 [AMD]

Diese erweitern wir nun um die Speicherglieder und deren Rückkopplung. Beachten Sie die Ergänzung auf der Ausgangsseite und die zusätzliche Clock-Leitung.

PAL16L8 [AMD]

[AMD] Datenblatt PAL16R8 Family, Advanced Micro Devices, link, 1996

Integration im 4809

Konfigurierbare Logikbausteile des 4809 [Microchip4809] Seite 116

Die CCL-Peripherie bietet eine Reihe von Look-up Tables (LUTs). Jede LUT besteht aus drei Eingängen, einer Wahrheitstabelle, einem Synchronisator/Filter und einem Flankendetektor. Jede LUT kann einen Ausgang als anwenderprogrammierbaren logischen Ausdruck erzeugen mit drei Eingängen. Der Ausgang wird aus den Eingängen mit Hilfe der kombinatorischen Logik generiert und kann gefiltert werden, um Spikes zu entfernen. Der CCL kann so konfiguriert werden, dass er bei Änderungen der LUT-Ausgänge eine Interrupt-Anforderung erzeugt. Benachbarte LUTs können kombiniert werden, um bestimmte Operationen durchzuführen.

Eventsystem

Das Eventsystem erlaubt eine direkte Peripherie-zu-Peripherie-Signalisierung, Peripheriegeräte können direkt Peripherieereignisse erzeugen, verwenden und darauf reagieren. Dabei werden entsprechen kurze Reaktionszeit garantiert. Auf dem 4809 sind 8 parallele Ereigniskanäle verfügbar, wobei jeder Kanal wird von einem Ereignisgenerator gesteuert wird und mehrere Ereignisbenutzer haben kann. Ereignisse sind dabei die Zustände der meisten Peripheriegeräten oder manuelle, aus der Software gesendet Signale. Das Ereignissystem funktioniert in den Modi "Aktiv", "Leerlauf" und "Ruhezustand".

Event-Channel Konzept am Beispiel einer Timer / ADC Kombination [Microchip4809] Seite 124

Abfolge der Konfiguration:

1. ... Konfiguration eines Peripheriegerät als Quelle: Wenn es sich bei der erzeugenden Peripherie z. B. um einen Timer handelt, wird die Vorskalierung, das Vergleichsregister usw. so eingestellt, dass das gewünschte Ereignis erzeugt wird.
2. ... Konfiguration eines Peripheriegerät als ereignisverarbeitende(n) Senke(m): Wenn z. B. der ADC der Ereignisbenutzer ist, wird der ADC Prescaler, die Auflösung, die Wandlungszeit usw. wie gewünscht eingestellt und die ADC-Wandlung so konfiguriert, dass sie beim Empfang eines Ereignisses startet eines Ereignisses startet.
3. ... Konfiguration des Ereignissystems: Im genannten Fall leitet der Timer/Compare seine Events z. B. über Kanal 0, was durch Schreiben in `EVSYS.CHANNEL0` erreicht wird. Der ADC wird so konfiguriert, dass er auf diesen Kanal hört.

Event-Channel Konzept [Microchip4809] Seite 123

Konfigurationsregister	Bedeutung
`CHANNELn`	Definition des singulären Triggers auf dem Kanal
`STROBEx`	Vektor der Zieldevices

Welche Inhalte können mit welchen Ausgaben verknüpft werden finden Sie im Handbuch ab Seite 125.

Events könnnen auch in Software ausgelöst werden.

Software

siehe Beispielfälle im Coderepository!

Aufgaben

- [ ] Installieren Sie sich das AVR Studio für die Arbeit mit dem 4809 Controller. Testen Sie den Aufbau anhand von einfachen Programmen.
- [ ] Implementieren Sie die Beispielumsetzung des Eventsystems mit Ihrem Controller als Einstiegsaufgabe nach. Den Taster dazu finden Sie im Bastelset.