Embedded Systems / Eingebettete Systeme

Studiengang Informatik
Campus Minden

Matthias König

FH Bielefeld
University of
Applied Sciences

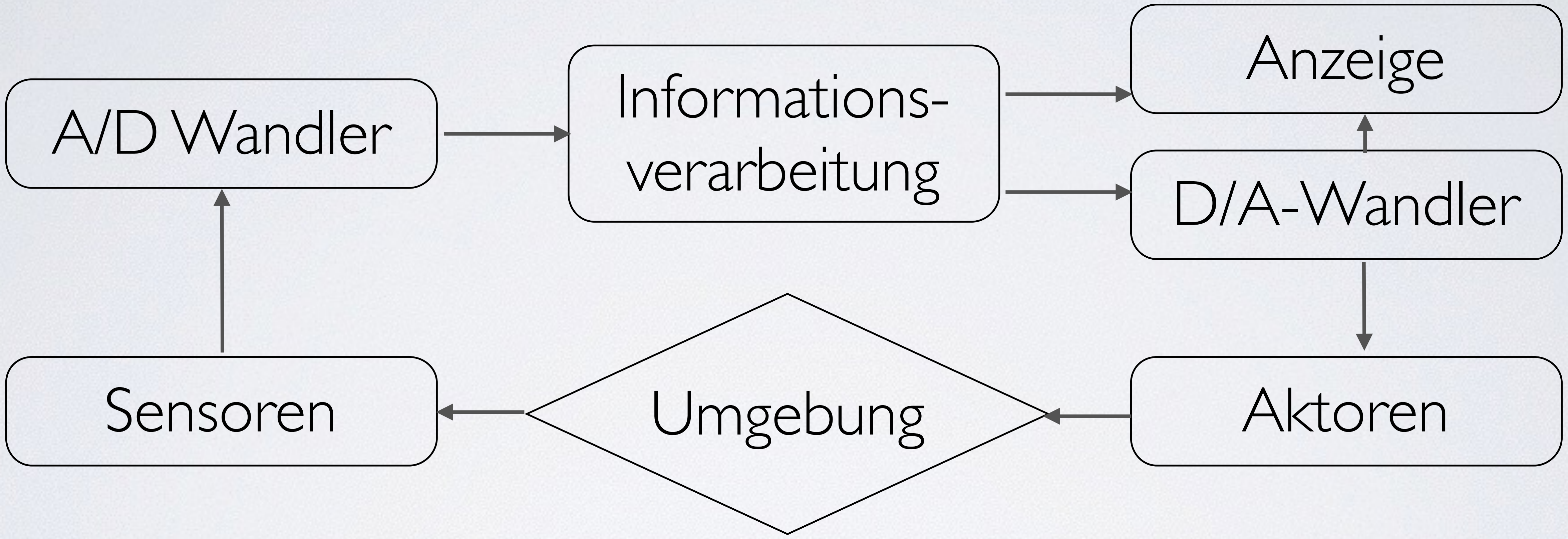
Beispiel einer Anwendung: Blumentopf

- 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]

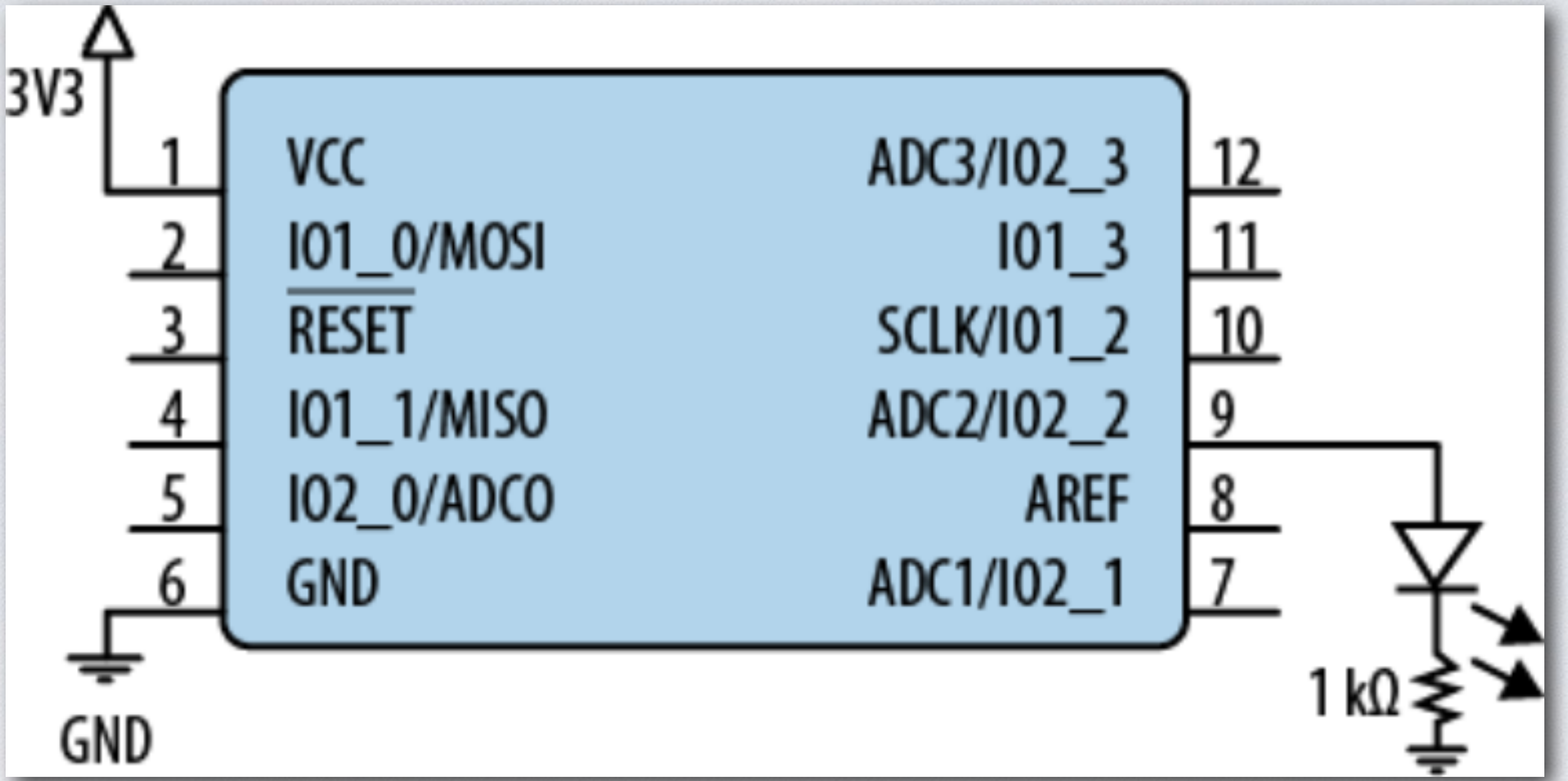
WIEDERHOLUNG

Hardware in a loop



[Quelle: Marwedel, Eingebettete Systeme]

Beispiel: Einschalten der LED



[Quelle: White, Making Embedded Systems]

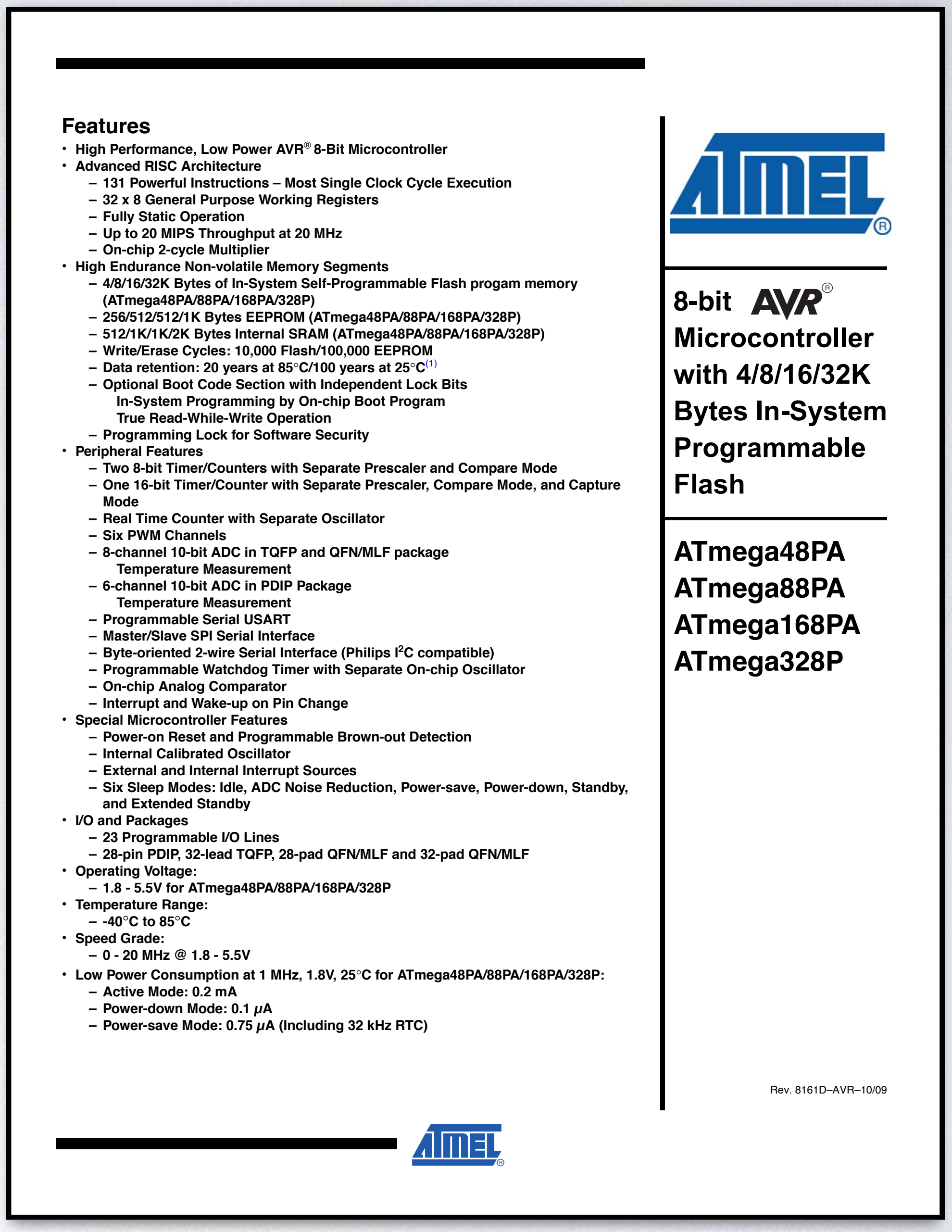
```
P2DIR |= (1 << 2); // set to output

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

MIKROCONTROLLER-PROGRAMMIERUNG AM BEISPIEL DES ATMEGA328P

Am Beispiel des Mikrocontrollers Atmega

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

in aller Kürze



Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32K Bytes of In-System Self-Programmable Flash progam memory (ATmega48PA/88PA/168PA/328P)
  - 256/512/512/1K Bytes EEPROM (ATmega48PA/88PA/168PA/328P)
  - 512/1K/1K/2K Bytes Internal SRAM (ATmega48PA/88PA/168PA/328P)
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I^{2}C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V for ATmega48PA/88PA/168PA/328P
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - 0 - 20 MHz @ 1.8 - 5.5V
- Low Power Consumption at 1 MHz, 1.8V, 25°C for ATmega48PA/88PA/168PA/328P:
  - Active Mode: 0.2 mA
  - Power-down Mode: 0.1 μA
  - Power-save Mode: 0.75 μA (Including 32 kHz RTC)

8-bit AVR® Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash

ATmega48PA
ATmega88PA
ATmega168PA
ATmega328P

Rev. 8161D–AVR–10/09

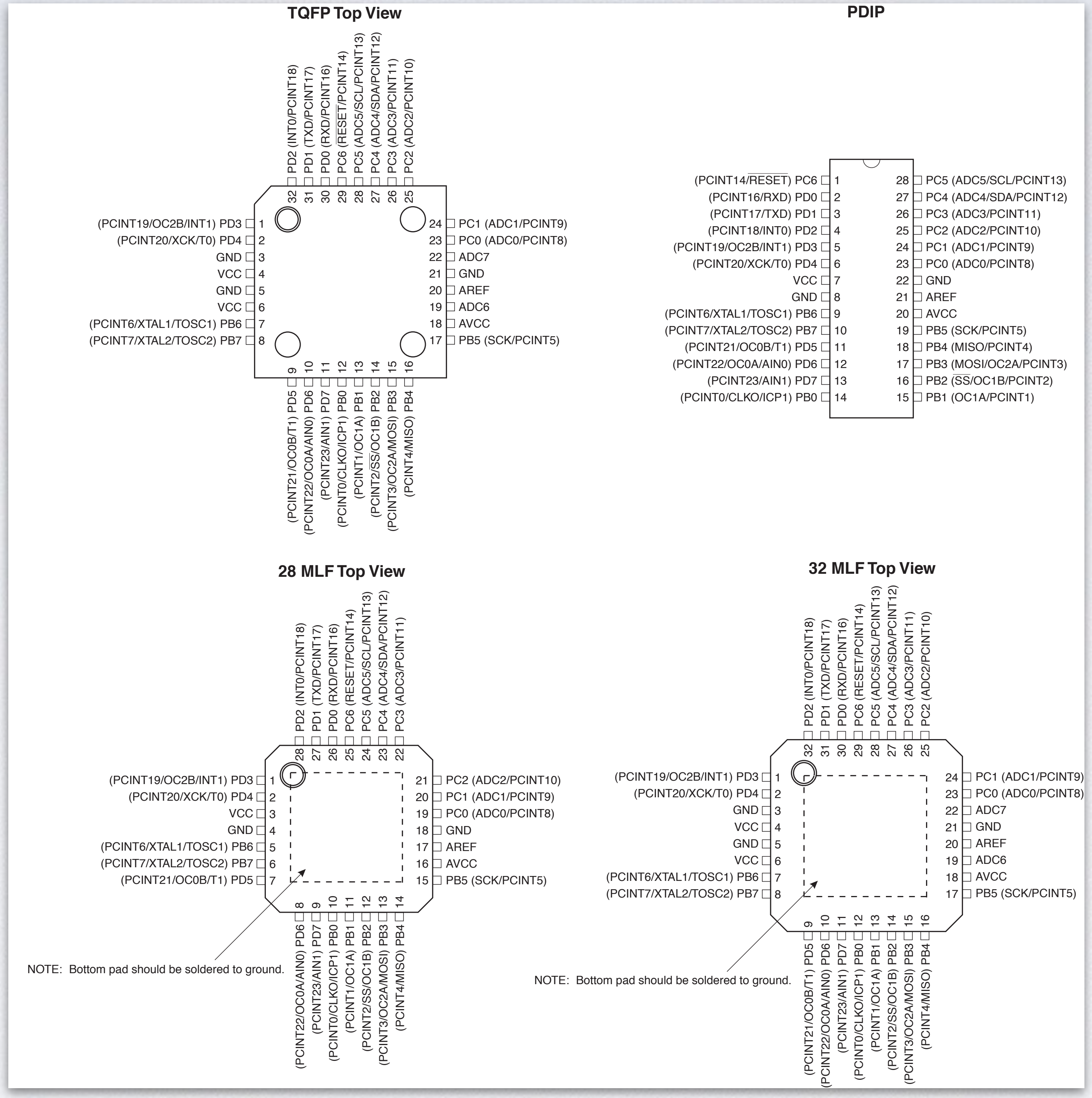
Datenblatt: 448 Seiten

[Quelle: Atmel, 8-bit AVR Microcontroller]

Datenblatt Atmega: Inhalt

Pin Configurations

Figure 1-1. Pinout ATmega48PA/88PA/168PA/328P

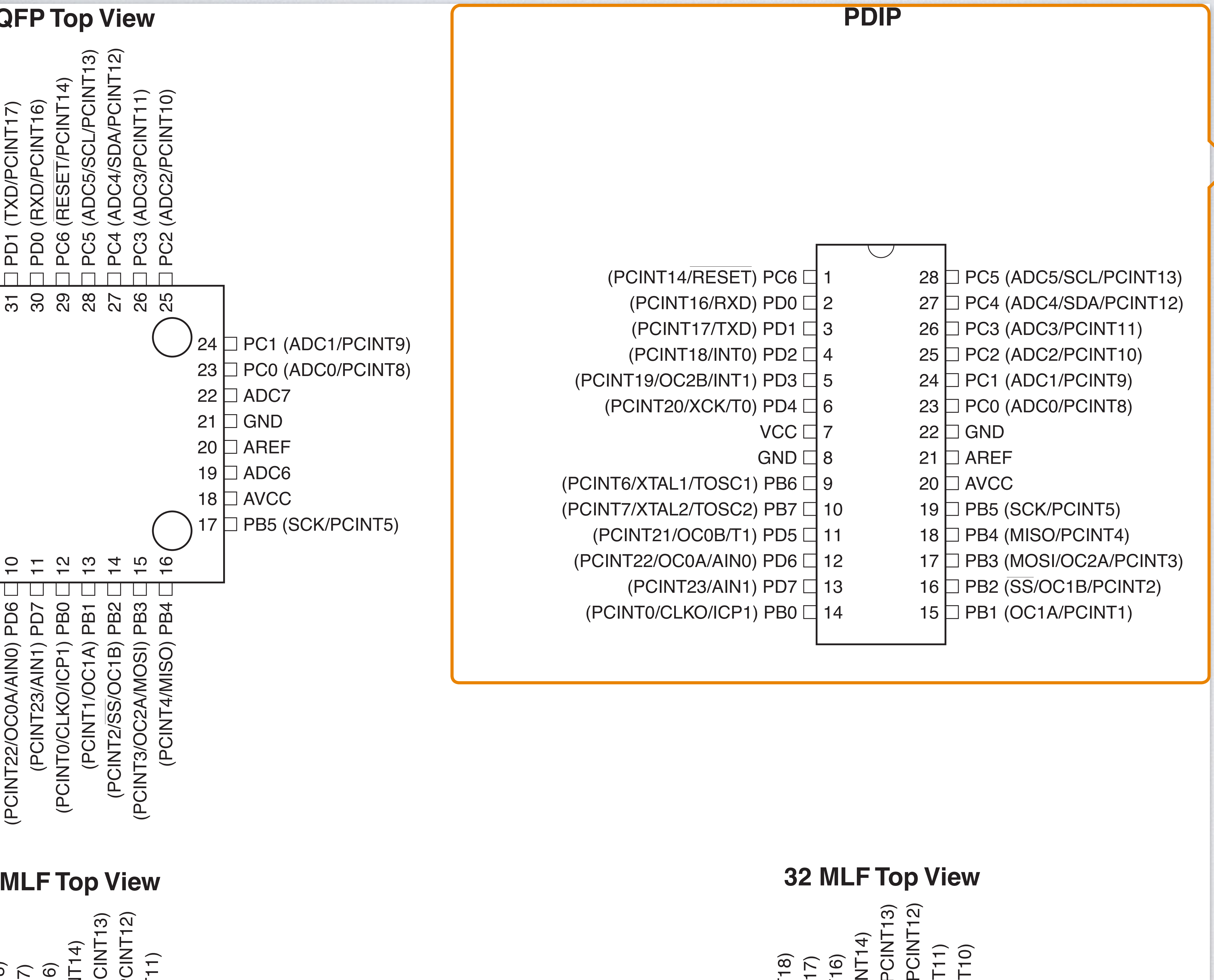


Datenblatt
Seite 2

[Quelle: Atmel, 8-bit AVR Microcontroller]

PA/88PA/168PA/328P

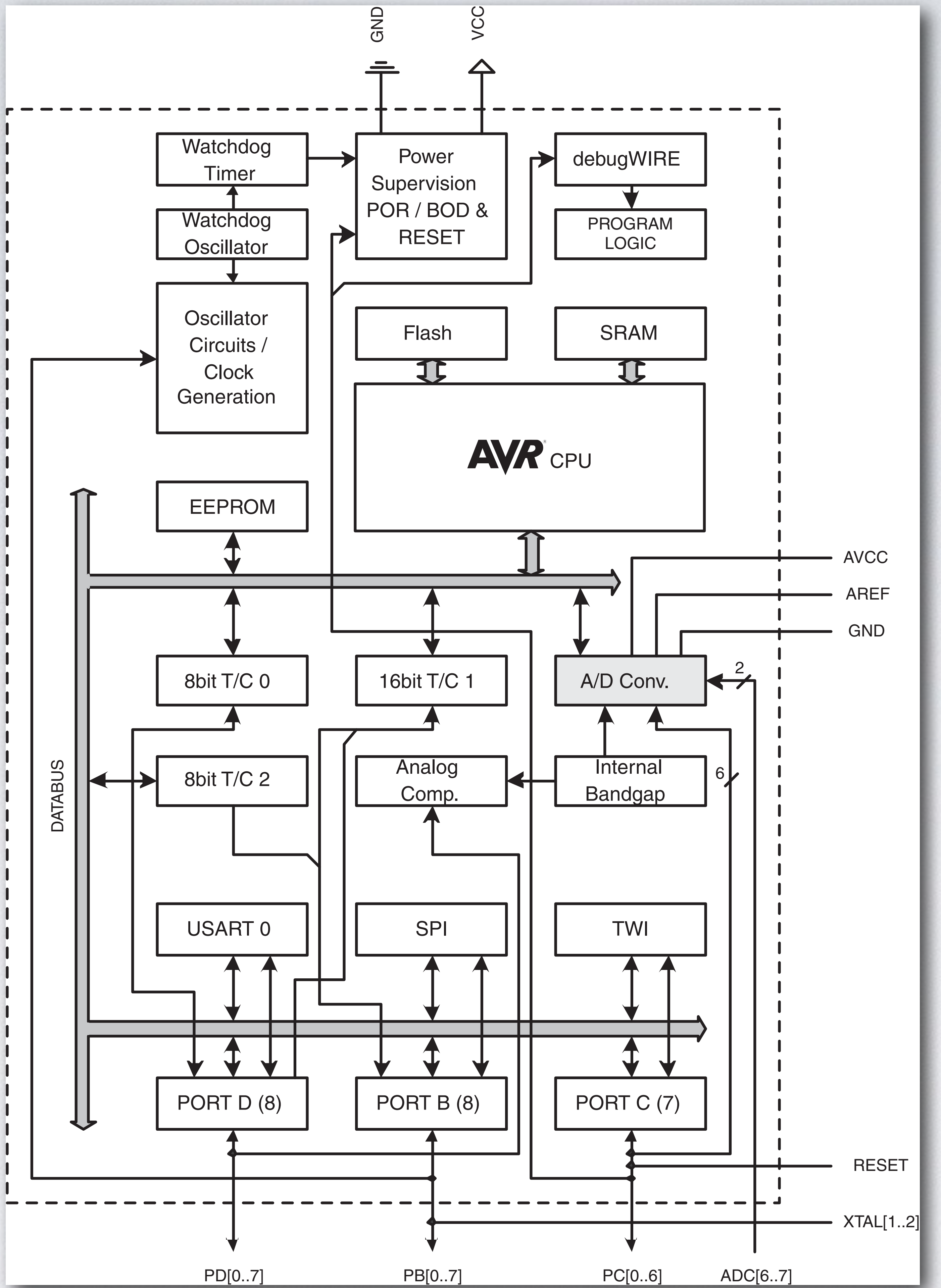
Pin Configurations



Arduino nutzt Atmega mit DIP (dual in-line package)

Datenblatt Seite 2

[Quelle: Atmel, 8-bit AVR Microcontroller]



Blockdiagramm eines ATmegas

Datenblatt
Seite 5

[Quelle: Atmel, 8-bit AVR Microcontroller]

Speichergröße

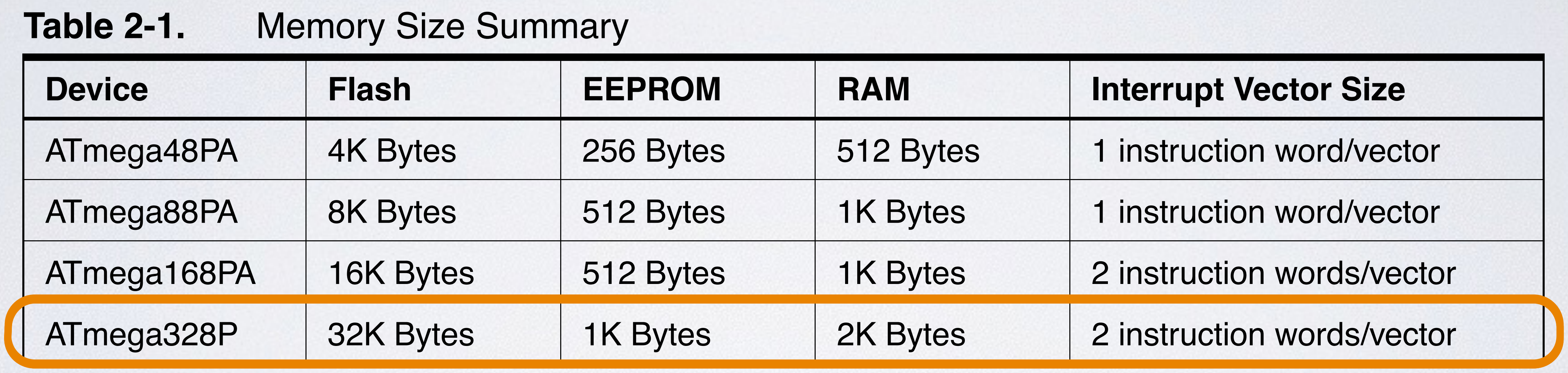
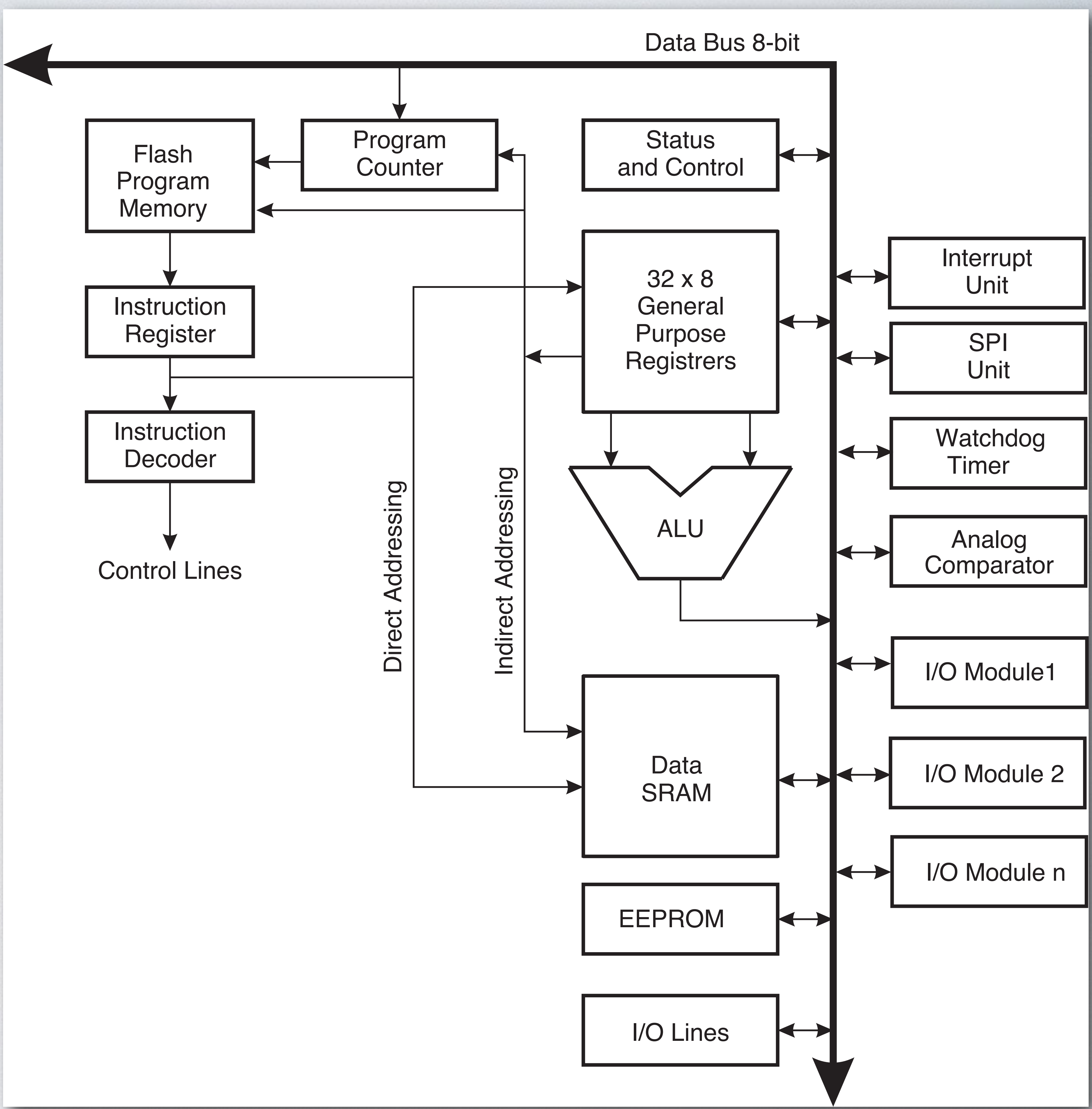


Table 2-1. Memory Size Summary

Device	Flash	EEPROM	RAM	Interrupt Vector Size
ATmega48PA	4K Bytes	256 Bytes	512 Bytes	1 instruction word/vector
ATmega88PA	8K Bytes	512 Bytes	1K Bytes	1 instruction word/vector
ATmega168PA	16K Bytes	512 Bytes	1K Bytes	2 instruction words/vector
ATmega328P	32K Bytes	1K Bytes	2K Bytes	2 instruction words/vector

Datenblatt
Seite 6

[Quelle: Atmel, 8-bit AVR Microcontroller]



ATmega Kern

Datenblatt
Seite 8

[Quelle: Atmel, 8-bit AVR Microcontroller]

“Wichtigste” Register

- Program Counter PC
- Status Register SREG
- General Purpose Register R0 bis R31
- Address Register X, Y, Z (R26-R31)
- Stack Pointer SPH und SPL
- MCU Control Register
- Watchdog Timer Control Register

Datenblatt
Seiten 8-13

[Quelle: Atmel, 8-bit AVR Microcontroller]

General Purpose Register

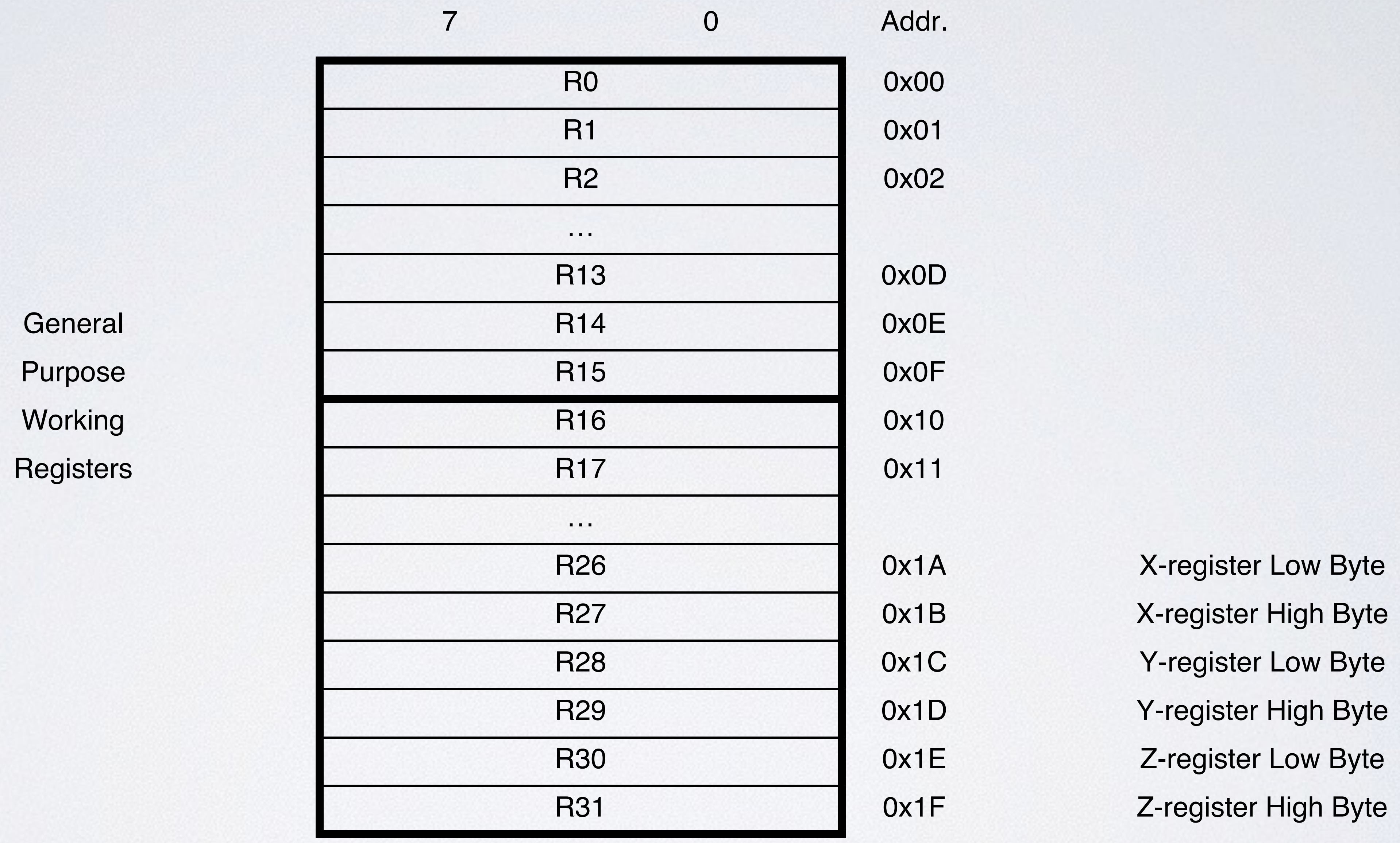


Figure 6-2. AVR CPU General Purpose Working Registers

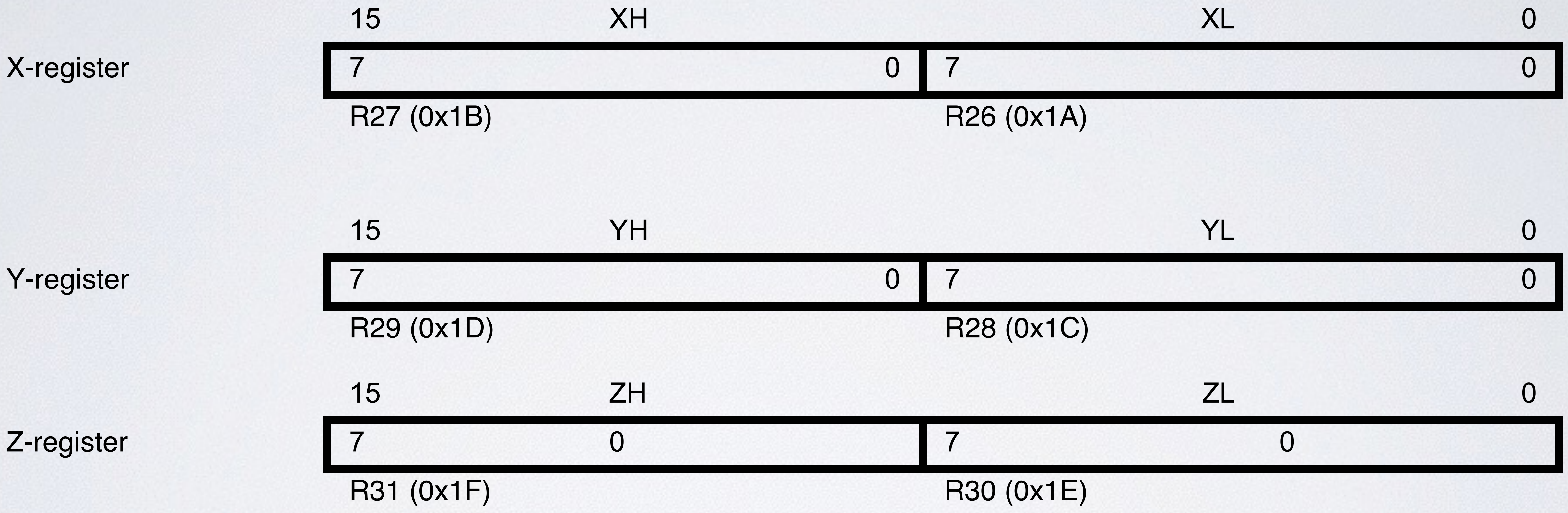
7 … 0	Addr.	
R0	0x00	
R1	0x01	
R2	0x02	
…		
R13	0x0D	
R14	0x0E	
R15	0x0F	
R16	0x10	
R17	0x11	
…		
R26	0x1A	X-register Low Byte
R27	0x1B	X-register High Byte
R28	0x1C	Y-register Low Byte
R29	0x1D	Y-register High Byte
R30	0x1E	Z-register Low Byte
R31	0x1F	Z-register High Byte

Datenblatt
Seite 11

[Quelle: Atmel, 8-bit AVR Microcontroller]

Address-Register

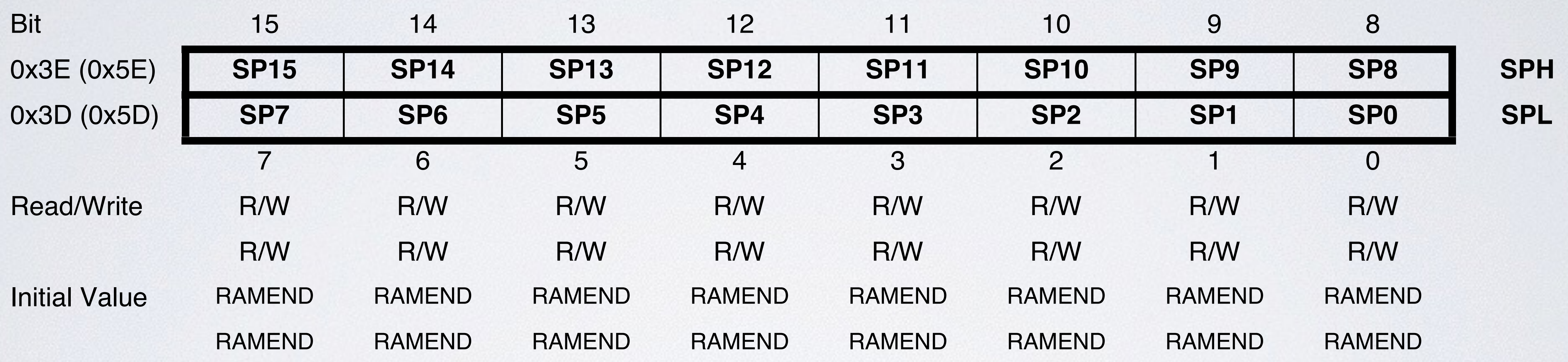
Figure 6-3. The X-, Y-, and Z-registers



Datenblatt
Seite 12

[Quelle: Atmel, 8-bit AVR Microcontroller]

Stack Pointer



Bit	15	14	13	12	11	10	9	8	
0x3E (0x5E)	**SP15**	**SP14**	**SP13**	**SP12**	**SP11**	**SP10**	**SP9**	**SP8**	**SPH**
0x3D (0x5D)	**SP7**	**SP6**	**SP5**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**	**SPL**
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	
	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	

Datenblatt
Seite 13

[Quelle: Atmel, 8-bit AVR Microcontroller]

Status-Register

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**	**SREG**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

I: Global Interrupt Enable

T: Bit Copy Storage

H: Half Carry Flag

S: Sign Bit

V: Two's Complement Overflow Flag

N: Negative Flag

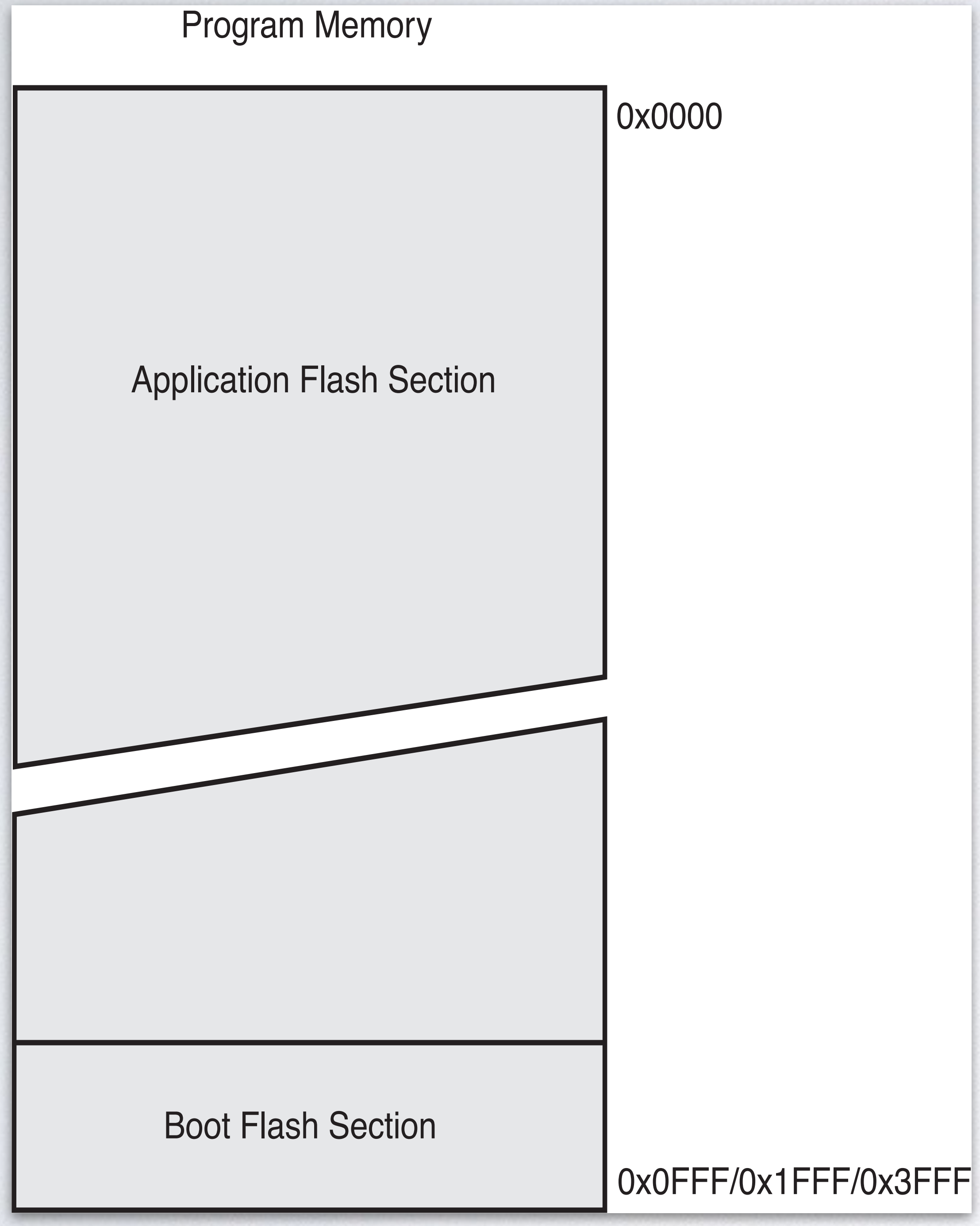
Z: Zero Flag

C: Carry Flag

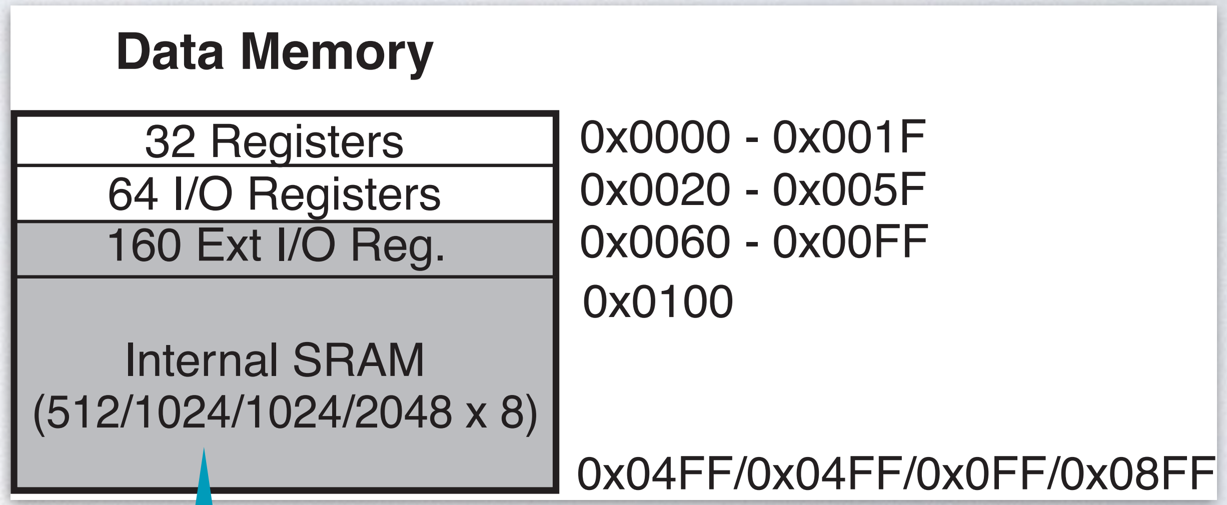
Datenblatt
Seite 10

[Quelle: Atmel, 8-bit AVR Microcontroller]

Speicherstruktur



Data Memory Map



.data (init. data)
.bss (uninit. data)
heap ↓
stack ↑

Datenblatt
Seiten 17-18

[Quelle: Atmel, 8-bit AVR Microcontroller]

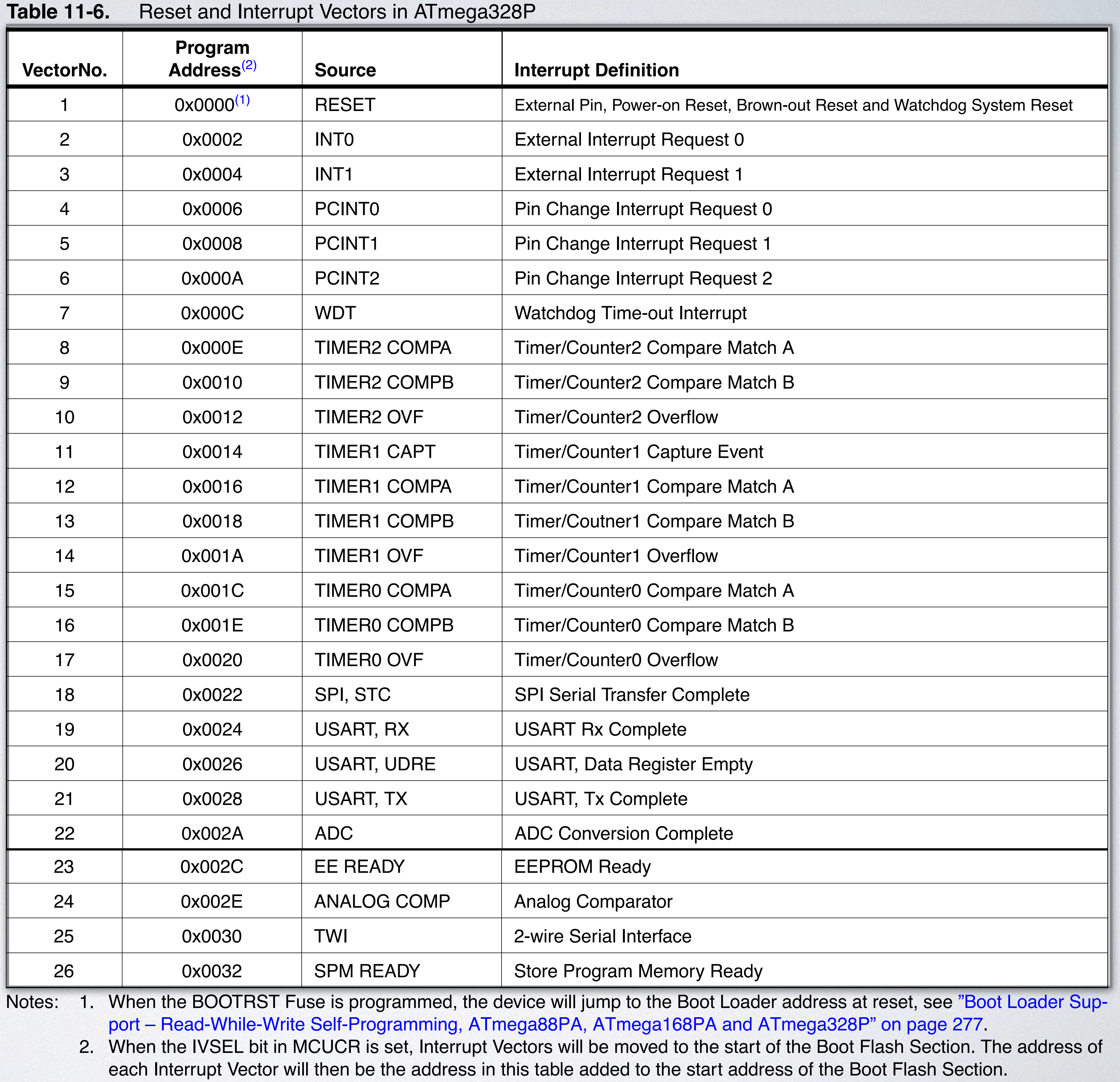


Table 11-6. Reset and Interrupt Vectors in ATmega328P

VectorNo.	Program Address[(2)]	Source	Interrupt Definition
1	0x0000[(1)]	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 1
4	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x000A	PCINT2	Pin Change Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 Compare Match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 Compare Match B
10	0x0012	TIMER2 OVF	Timer/Counter2 Overflow
11	0x0014	TIMER1 CAPT	Timer/Counter1 Capture Event
12	0x0016	TIMER1 COMPA	Timer/Counter1 Compare Match A
13	0x0018	TIMER1 COMPB	Timer/Coutner1 Compare Match B
14	0x001A	TIMER1 OVF	Timer/Counter1 Overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 Compare Match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 Compare Match B
17	0x0020	TIMER0 OVF	Timer/Counter0 Overflow
18	0x0022	SPI, STC	SPI Serial Transfer Complete
19	0x0024	USART, RX	USART Rx Complete
20	0x0026	USART, UDRE	USART, Data Register Empty
21	0x0028	USART, TX	USART, Tx Complete
22	0x002A	ADC	ADC Conversion Complete
23	0x002C	EE READY	EEPROM Ready
24	0x002E	ANALOG COMP	Analog Comparator
25	0x0030	TWI	2-wire Serial Interface
26	0x0032	SPM READY	Store Program Memory Ready

Notes: 1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see "Boot Loader Support – Read-While-Write Self-Programming, ATmega88PA, ATmega168PA and ATmega328P" on page 277.
2. When the IVSEL bit in MCUCR is set, Interrupt Vectors will be moved to the start of the Boot Flash Section. The address of each Interrupt Vector will then be the address in this table added to the start address of the Boot Flash Section.

Interrupt Vectors

Datenblatt
Seiten 65-66

[Quelle: Atmel, 8-bit AVR Microcontroller]

```
Address  Labels Code                      Comments
0x0000          jmp    RESET              ; Reset Handler
0x0002          jmp    EXT_INT0           ; IRQ0 Handler
0x0004          jmp    EXT_INT1           ; IRQ1 Handler
0x0006          jmp    PCINT0             ; PCINT0 Handler
0x0008          jmp    PCINT1             ; PCINT1 Handler
0x000A          jmp    PCINT2             ; PCINT2 Handler
0x000C          jmp    WDT                ; Watchdog Timer Handler
0x000E          jmp    TIM2_COMPA         ; Timer2 Compare A Handler
0x0010          jmp    TIM2_COMPB         ; Timer2 Compare B Handler
0x0012          jmp    TIM2_OVF           ; Timer2 Overflow Handler
0x0014          jmp    TIM1_CAPT          ; Timer1 Capture Handler
0x0016          jmp    TIM1_COMPA         ; Timer1 Compare A Handler
0x0018          jmp    TIM1_COMPB         ; Timer1 Compare B Handler
0x001A          jmp    TIM1_OVF           ; Timer1 Overflow Handler
0x001C          jmp    TIM0_COMPA         ; Timer0 Compare A Handler
0x001E          jmp    TIM0_COMPB         ; Timer0 Compare B Handler
0x0020          jmp    TIM0_OVF           ; Timer0 Overflow Handler
0x0022          jmp    SPI_STC            ; SPI Transfer Complete Handler
0x0024          jmp    USART_RXC          ; USART, RX Complete Handler
0x0026          jmp    USART_UDRE         ; USART, UDR Empty Handler
0x0028          jmp    USART_TXC          ; USART, TX Complete Handler
0x002A          jmp    ADC                ; ADC Conversion Complete Handler
0x002C          jmp    EE_RDY             ; EEPROM Ready Handler
0x002E          jmp    ANA_COMP           ; Analog Comparator Handler
0x0030          jmp    TWI                ; 2-wire Serial Interface Handler
0x0032          jmp    SPM_RDY            ; Store Program Memory Ready Handler
;
0x0033RESET:    ldi    r16, high(RAMEND); Main program start
0x0034          out    SPH,r16            ; Set Stack Pointer to top of RAM
0x0035          ldi    r16, low(RAMEND)
0x0036          out    SPL,r16
0x0037          sei                       ; Enable interrupts
0x0038          <instr>  xxx
  ...     ...     ...  ...
```

Interrupt Vectors

Datenblatt
Seiten 66-67

[Quelle: Atmel, 8-bit AVR Microcontroller]

Reset des Atmega

- Beim Reset werden
  - alle Register auf initiale Werte gesetzt,
  - alle I/O Ports auf initialen Zustand gesetzt,
  - das Programm vom Reset Vector gestartet.
- Resetquellen:
  - Power-On
  - Extern (Pins)
  - Watchdog
  - Brown-Out (mittels Eingangsspannung unter definierten Pegel)

Datenblatt
Seiten 46-50

[Quelle: Atmel, 8-bit AVR Microcontroller]

Reset des Atmega

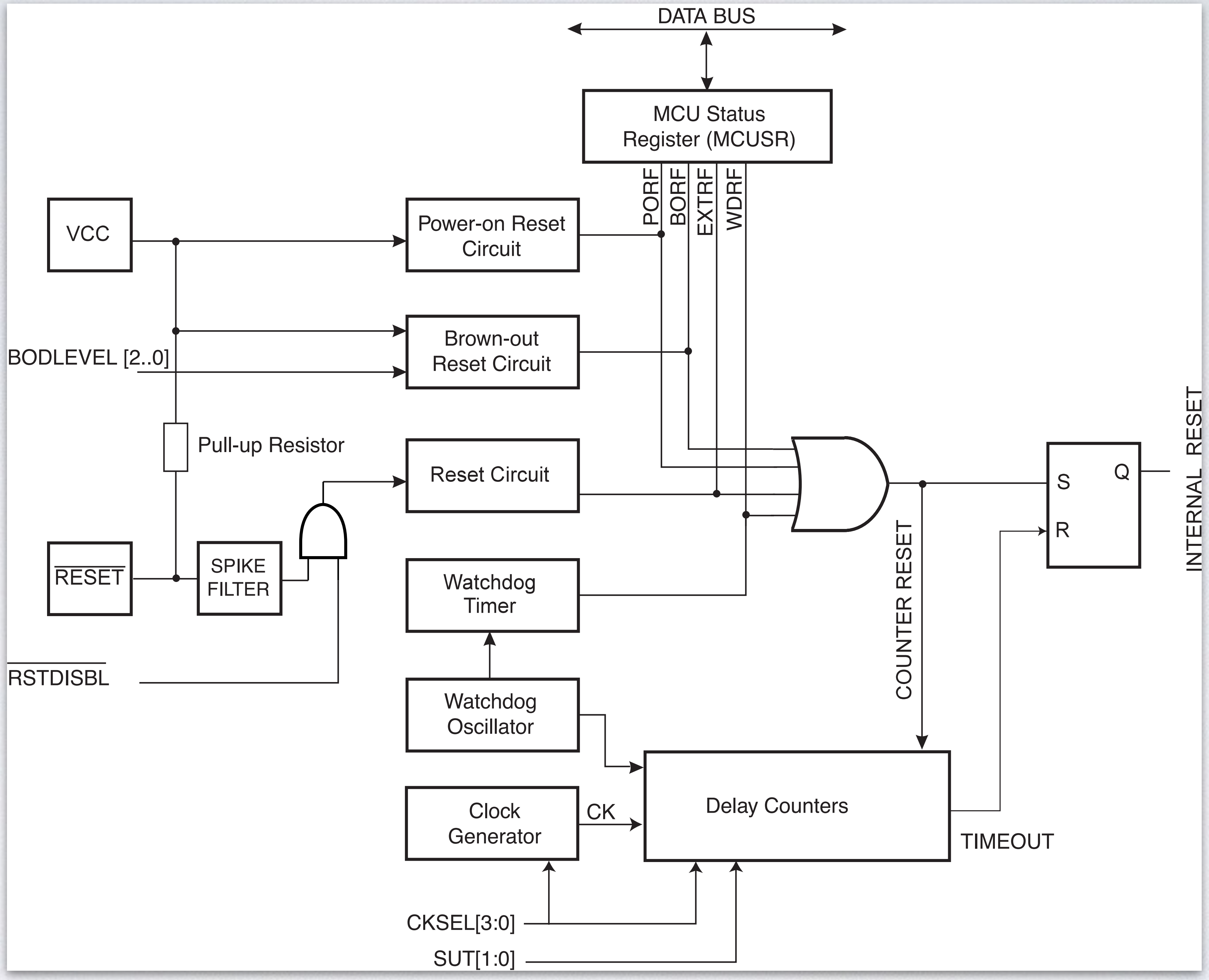
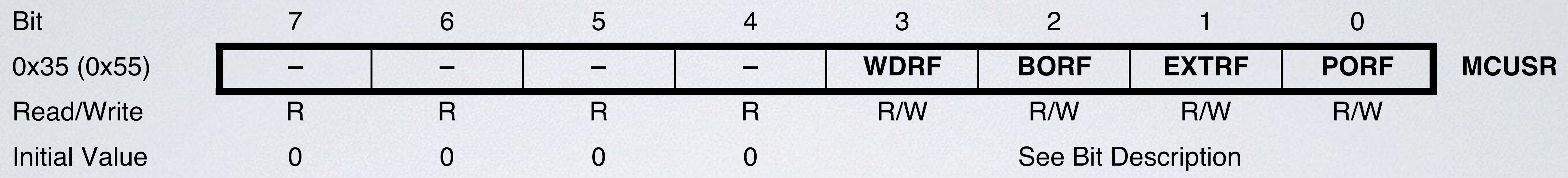


Figure 10-1. Reset Logic

Datenblatt
Seite 47

[Quelle: Atmel, 8-bit AVR Microcontroller]

MCU-Status-Register



Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	–	–	–	–	**WDRF**	**BORF**	**EXTRF**	**PORF**	**MCUSR**
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

WDRF: Watchdog System Reset Flag

BORF: Brown-out Reset Flag

EXTRF: External Reset Flag

PORF: Power-on Reset Flag

Datenblatt
Seite 54

[Quelle: Atmel, 8-bit AVR Microcontroller]

Atmega Watchdog

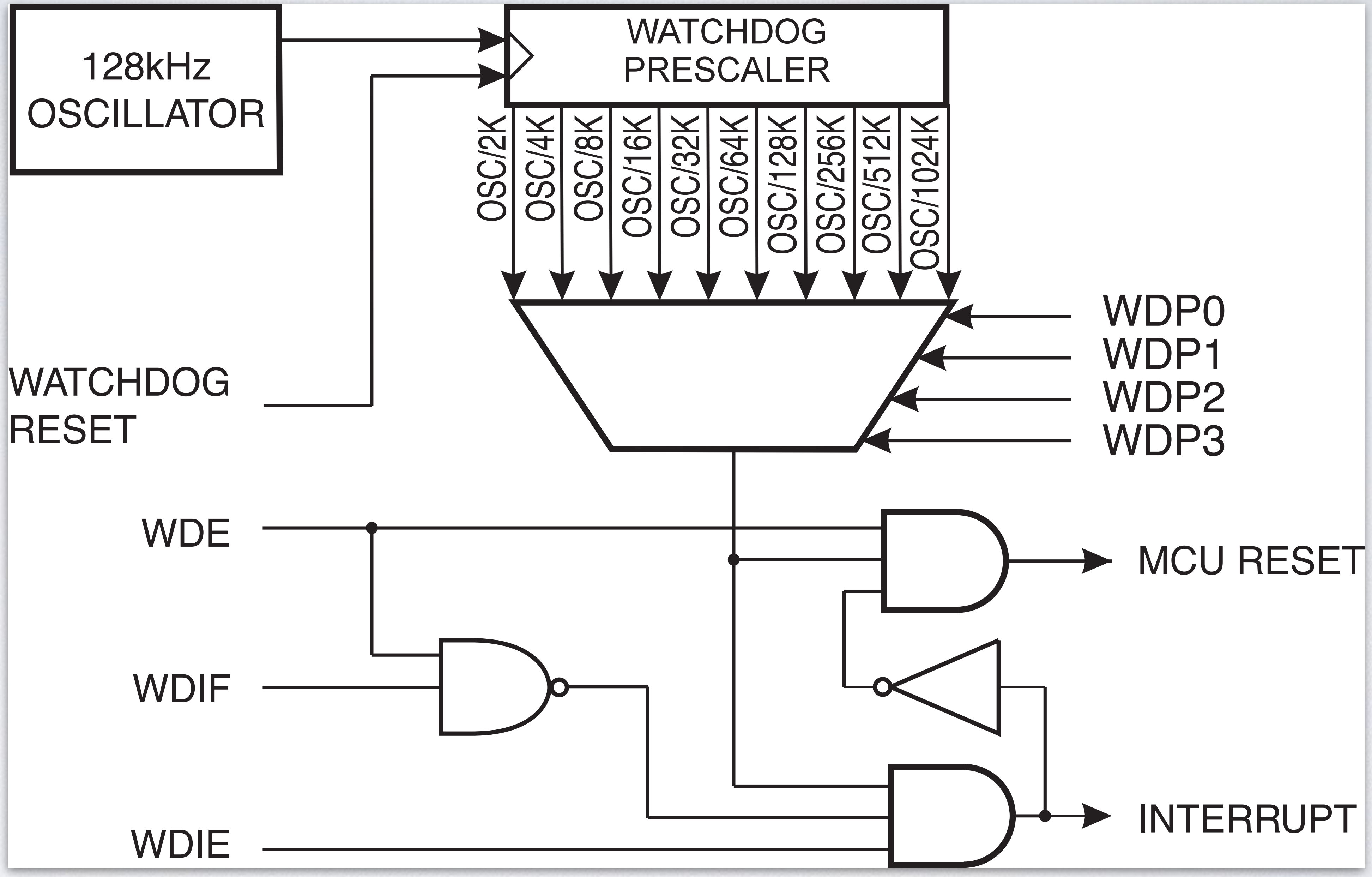


Figure 10-7. Watchdog Timer

Achtung: Genauer Ablauf zum Setzen der Flags im Datenblatt

Datenblatt
Seite 50

[Quelle: Atmel, 8-bit AVR Microcontroller]

Watchdog Timer Control-Register

Bit	7	6	5	4	3	2	1	0	
(0x60)	**WDIF**	**WDIE**	**WDP3**	**WDCE**	**WDE**	**WDP2**	**WDP1**	**WDP0**	**WDTCSR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	X	0	0	0	

WDIF: Watchdog Interrupt Flag

WDIE: Watchdog Interrupt Enable Flag

WDTON[(1)]	**WDE**	**WDIE**	**Mode**	**Action on Time-out**
1	0	0	Stopped	None
1	0	1	Interrupt Mode	Interrupt
1	1	0	System Reset Mode	Reset
1	1	1	Interrupt and System Reset Mode	Interrupt, then go to System Reset Mode
0	x	x	System Reset Mode	Reset

Note: 1. WDTON Fuse set to "0" means programmed and "1" means unprogrammed.

Datenblatt
Seiten 54-56

[Quelle: Atmel, 8-bit AVR Microcontroller]

Watchdog Timer Control-Register

Bit	7	6	5	4	3	2	1	0	
(0x60)	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	WDTCSR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	X	0	0	0	

Table 10-2. Watchdog Timer Prescale Select

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at $V_{CC} = 5.0V$
0	0	0	0	2K (2048) cycles	16 ms
0	0	0	1	4K (4096) cycles	32 ms
0	0	1	0	8K (8192) cycles	64 ms
0	0	1	1	16K (16384) cycles	0.125 s
0	1	0	0	32K (32768) cycles	0.25 s
0	1	0	1	64K (65536) cycles	0.5 s
0	1	1	0	128K (131072) cycles	1.0 s

. . .

Datenblatt
Seiten 54-56

[Quelle: Atmel, 8-bit AVR Microcontroller]

Power Management

- Sechs Sleep Modes
- Ablauf:
  - SE bit in SMCR to logic one
  - SLEEP instruction
- SMCR Register Bits geben Modus an.
- Genaueres s. Datenblatt

Datenblatt
Seite 39

Power Management

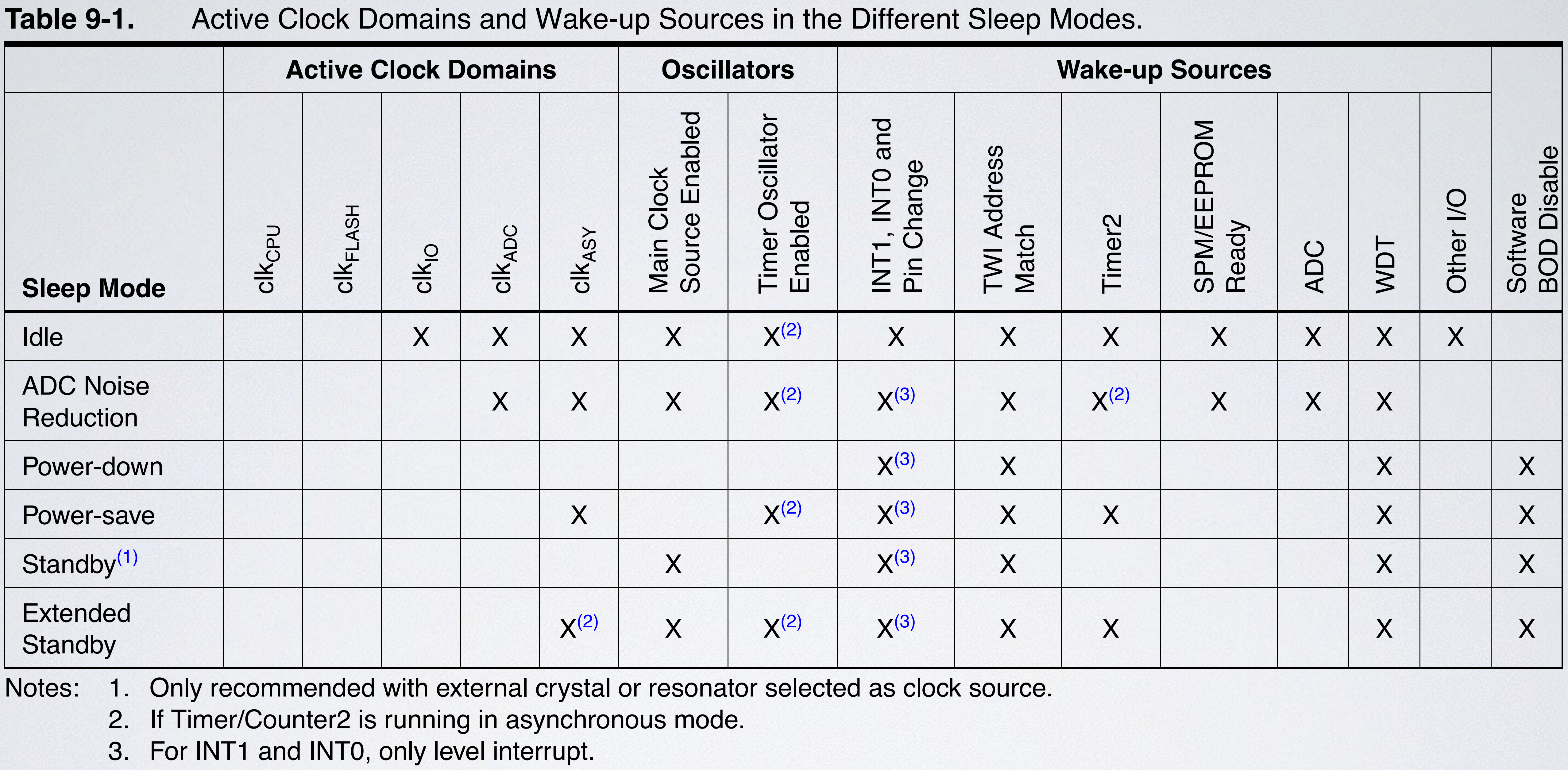


Table 9-1. Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

	Active Clock Domains					Oscillators		Wake-up Sources							
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Oscillator Enabled	INT1, INT0 and Pin Change	TWI Address Match	Timer2	SPM/EEPROM Ready	ADC	WDT	Other I/O	Software BOD Disable
Idle			X	X	X	X	X(2)	X	X	X	X	X	X	X	
ADC Noise Reduction				X	X	X	X(2)	X(3)	X	X(2)	X	X	X		
Power-down								X(3)	X				X		X
Power-save					X		X(2)	X(3)	X	X			X		X
Standby(1)						X		X(3)	X				X		X
Extended Standby					X(2)	X	X(2)	X(3)	X	X			X		X

Notes:
1. Only recommended with external crystal or resonator selected as clock source.
2. If Timer/Counter2 is running in asynchronous mode.
3. For INT1 and INT0, only level interrupt.

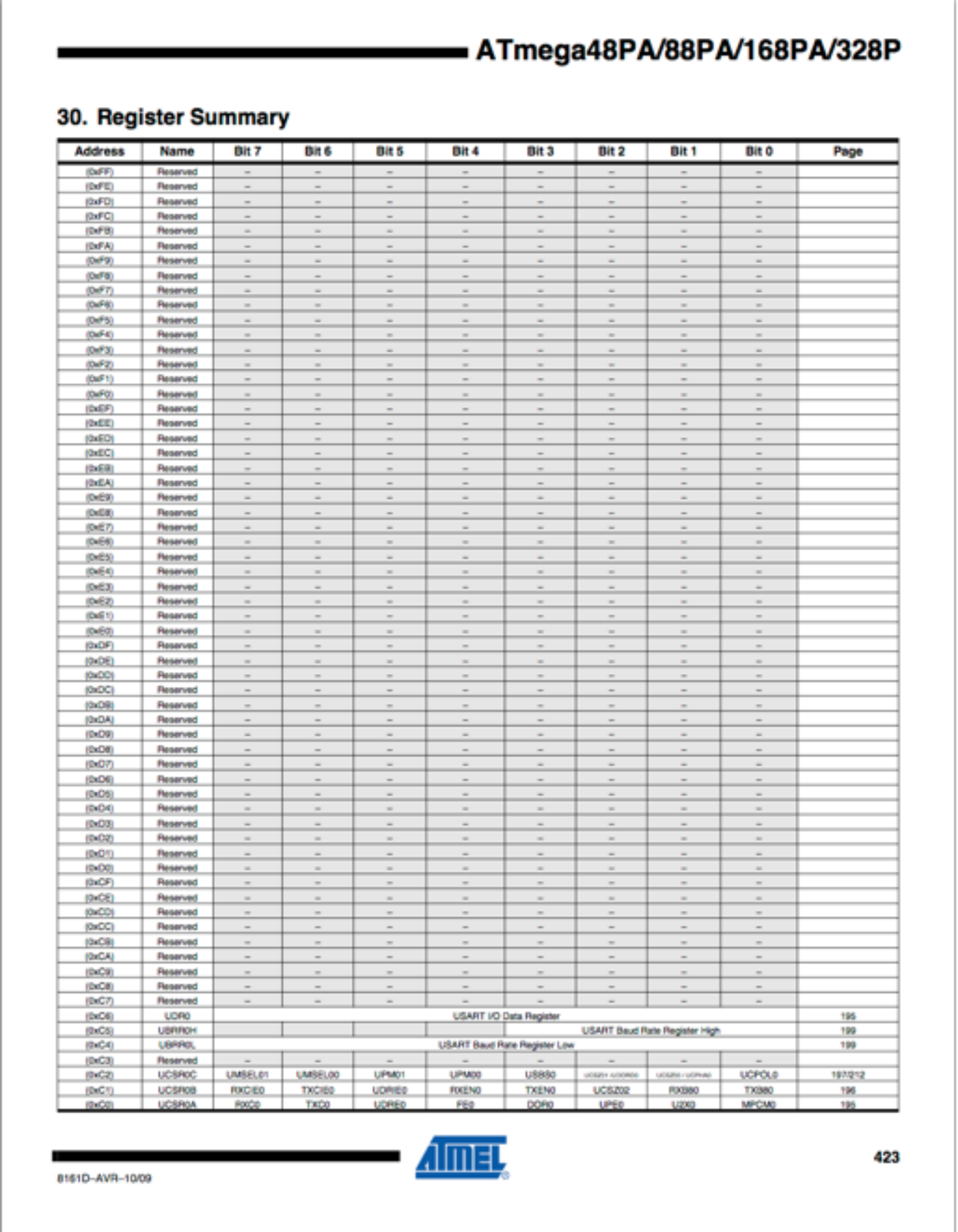
SMCR: Sleep Mode Control Register

MCUCR: MCU Control Register

PRR: Power Reduction Register

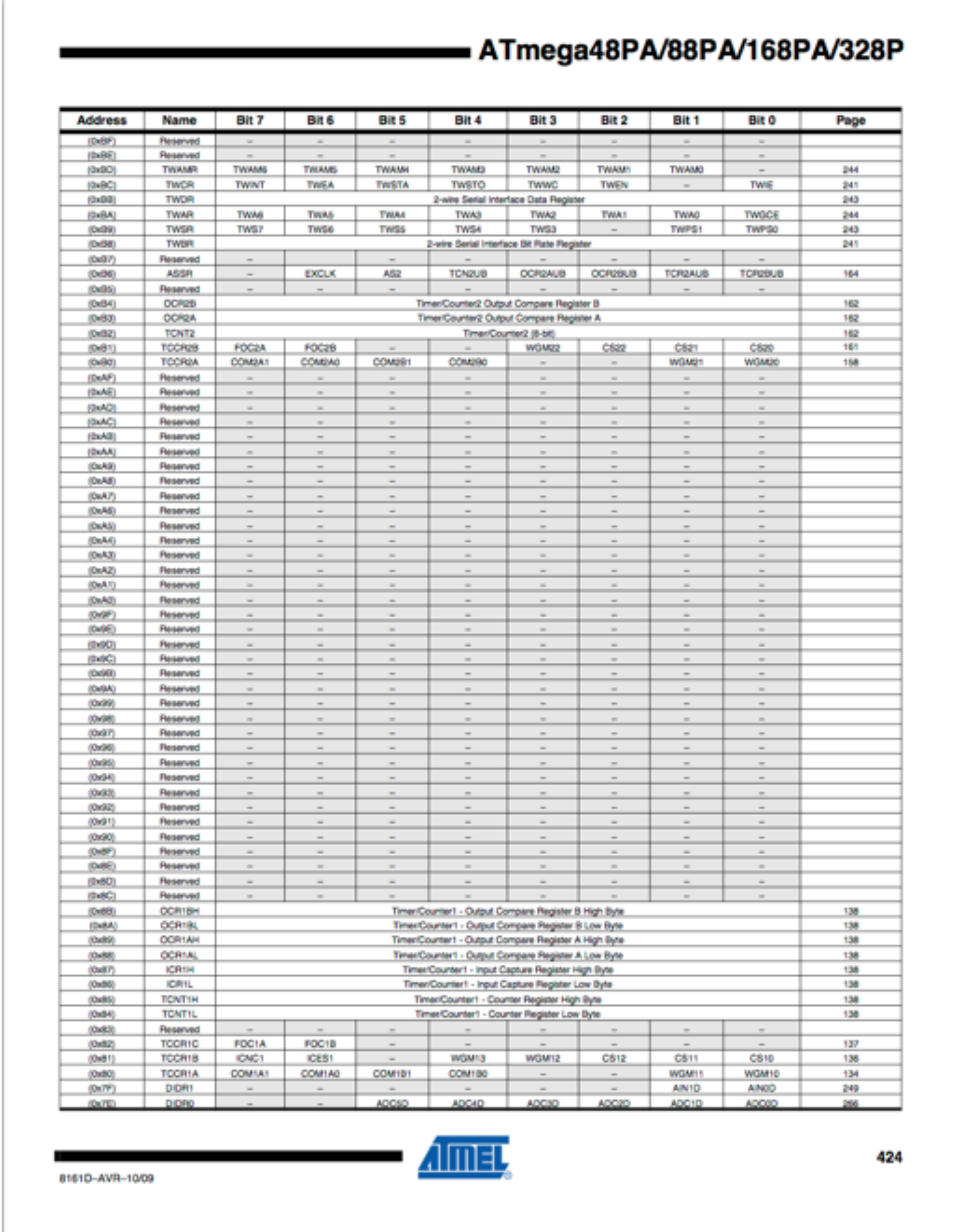
Datenblatt
Seite 39

[Quelle: Atmel, 8-bit AVR Microcontroller]



30. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xFF)	Reserved	–	–	–	–	–	–	–	–	
(0xFE)	Reserved	–	–	–	–	–	–	–	–	
(0xFD)	Reserved	–	–	–	–	–	–	–	–	
(0xFC)	Reserved	–	–	–	–	–	–	–	–	
(0xFB)	Reserved	–	–	–	–	–	–	–	–	
(0xFA)	Reserved	–	–	–	–	–	–	–	–	
(0xF9)	Reserved	–	–	–	–	–	–	–	–	
(0xF8)	Reserved	–	–	–	–	–	–	–	–	
(0xF7)	Reserved	–	–	–	–	–	–	–	–	
(0xF6)	Reserved	–	–	–	–	–	–	–	–	
(0xF5)	Reserved	–	–	–	–	–	–	–	–	
(0xF4)	Reserved	–	–	–	–	–	–	–	–	
(0xF3)	Reserved	–	–	–	–	–	–	–	–	
(0xF2)	Reserved	–	–	–	–	–	–	–	–	
(0xF1)	Reserved	–	–	–	–	–	–	–	–	
(0xF0)	Reserved	–	–	–	–	–	–	–	–	
(0xEF)	Reserved	–	–	–	–	–	–	–	–	
(0xEE)	Reserved	–	–	–	–	–	–	–	–	
(0xED)	Reserved	–	–	–	–	–	–	–	–	
(0xEC)	Reserved	–	–	–	–	–	–	–	–	
(0xEB)	Reserved	–	–	–	–	–	–	–	–	
(0xEA)	Reserved	–	–	–	–	–	–	–	–	
(0xE9)	Reserved	–	–	–	–	–	–	–	–	
(0xE8)	Reserved	–	–	–	–	–	–	–	–	
(0xE7)	Reserved	–	–	–	–	–	–	–	–	
(0xE6)	Reserved	–	–	–	–	–	–	–	–	
(0xE5)	Reserved	–	–	–	–	–	–	–	–	
(0xE4)	Reserved	–	–	–	–	–	–	–	–	
(0xE3)	Reserved	–	–	–	–	–	–	–	–	
(0xE2)	Reserved	–	–	–	–	–	–	–	–	
(0xE1)	Reserved	–	–	–	–	–	–	–	–	
(0xE0)	Reserved	–	–	–	–	–	–	–	–	
(0xDF)	Reserved	–	–	–	–	–	–	–	–	
(0xDE)	Reserved	–	–	–	–	–	–	–	–	
(0xDD)	Reserved	–	–	–	–	–	–	–	–	
(0xDC)	Reserved	–	–	–	–	–	–	–	–	
(0xDB)	Reserved	–	–	–	–	–	–	–	–	
(0xDA)	Reserved	–	–	–	–	–	–	–	–	
(0xD9)	Reserved	–	–	–	–	–	–	–	–	
(0xD8)	Reserved	–	–	–	–	–	–	–	–	
(0xD7)	Reserved	–	–	–	–	–	–	–	–	
(0xD6)	Reserved	–	–	–	–	–	–	–	–	
(0xD5)	Reserved	–	–	–	–	–	–	–	–	
(0xD4)	Reserved	–	–	–	–	–	–	–	–	
(0xD3)	Reserved	–	–	–	–	–	–	–	–	
(0xD2)	Reserved	–	–	–	–	–	–	–	–	
(0xD1)	Reserved	–	–	–	–	–	–	–	–	
(0xD0)	Reserved	–	–	–	–	–	–	–	–	
(0xCF)	Reserved	–	–	–	–	–	–	–	–	
(0xCE)	Reserved	–	–	–	–	–	–	–	–	
(0xCD)	Reserved	–	–	–	–	–	–	–	–	
(0xCC)	Reserved	–	–	–	–	–	–	–	–	
(0xCB)	Reserved	–	–	–	–	–	–	–	–	
(0xCA)	Reserved	–	–	–	–	–	–	–	–	
(0xC9)	Reserved	–	–	–	–	–	–	–	–	
(0xC8)	Reserved	–	–	–	–	–	–	–	–	
(0xC7)	Reserved	–	–	–	–	–	–	–	–	
(0xC6)	UDR0	USART I/O Data Register								195
(0xC5)	UBRR0H					USART Baud Rate Register High				199
(0xC4)	UBRR0L	USART Baud Rate Register Low								199
(0xC3)	Reserved	–	–	–	–	–	–	–	–	
(0xC2)	UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 /UDORD0	UCSZ00 / UCPHA0	UCPOL0	197/212
(0xC1)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	196
(0xC0)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	195

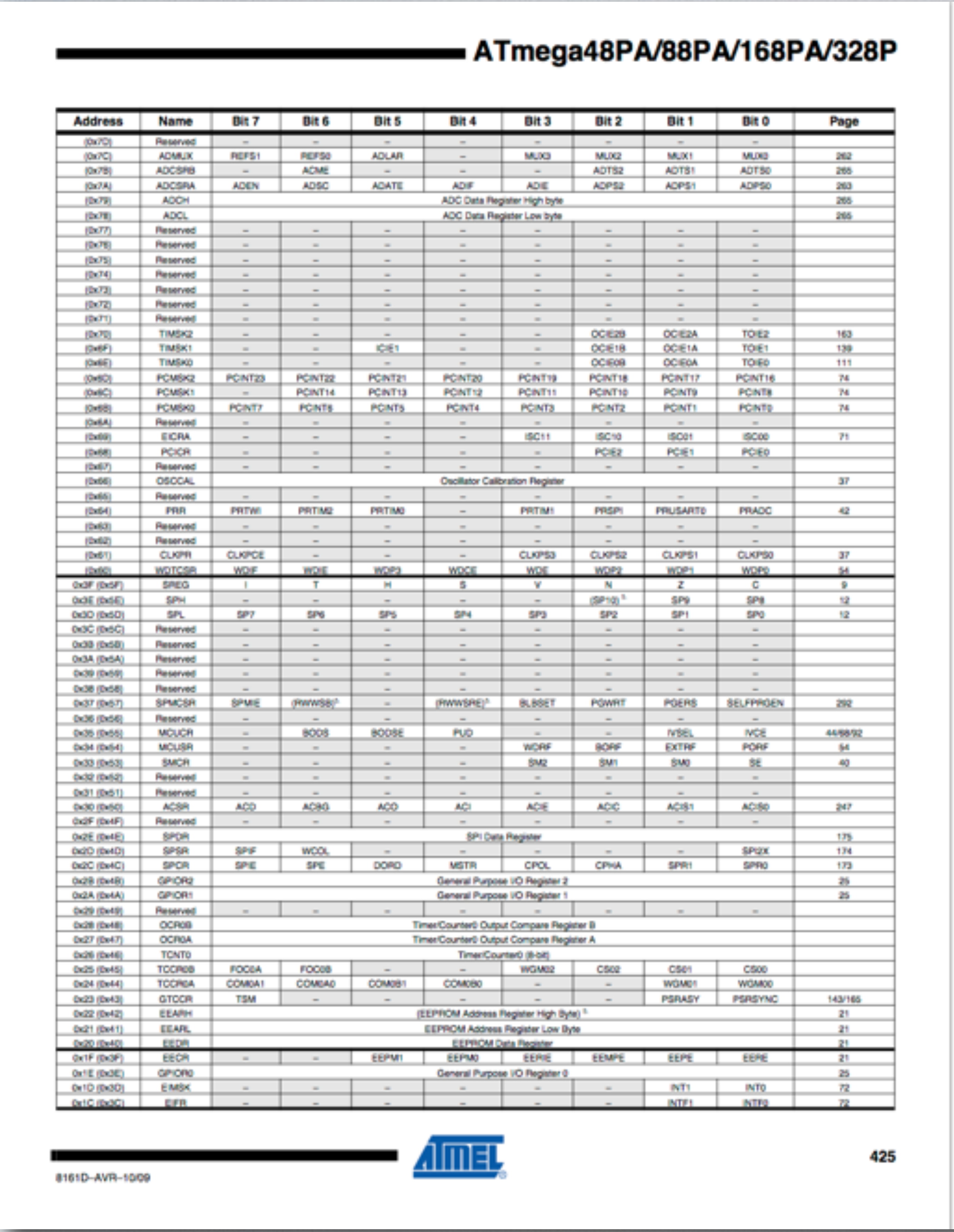


Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xBF)	Reserved	–	–	–	–	–	–	–	–	
(0xBE)	Reserved	–	–	–	–	–	–	–	–	
(0xBD)	TWAMR	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	–	244
(0xBC)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	241
(0xBB)	TWDR	2-wire Serial Interface Data Register								243
(0xBA)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	244
(0xB9)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	243
(0xB8)	TWBR	2-wire Serial Interface Bit Rate Register								241
(0xB7)	Reserved	–		–	–	–	–	–	–	
(0xB6)	ASSR	–	EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB	164
(0xB5)	Reserved	–	–	–	–	–	–	–	–	
(0xB4)	OCR2B	Timer/Counter2 Output Compare Register B								162
(0xB3)	OCR2A	Timer/Counter2 Output Compare Register A								162
(0xB2)	TCNT2	Timer/Counter2 (8-bit)								162
(0xB1)	TCCR2B	FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	161
(0xB0)	TCCR2A	COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	158
(0xAF)	Reserved	–	–	–	–	–	–	–	–	
(0xAE)	Reserved	–	–	–	–	–	–	–	–	
(0xAD)	Reserved	–	–	–	–	–	–	–	–	
(0xAC)	Reserved	–	–	–	–	–	–	–	–	
(0xAB)	Reserved	–	–	–	–	–	–	–	–	
(0xAA)	Reserved	–	–	–	–	–	–	–	–	
(0xA9)	Reserved	–	–	–	–	–	–	–	–	
(0xA8)	Reserved	–	–	–	–	–	–	–	–	
(0xA7)	Reserved	–	–	–	–	–	–	–	–	
(0xA6)	Reserved	–	–	–	–	–	–	–	–	
(0xA5)	Reserved	–	–	–	–	–	–	–	–	
(0xA4)	Reserved	–	–	–	–	–	–	–	–	
(0xA3)	Reserved	–	–	–	–	–	–	–	–	
(0xA2)	Reserved	–	–	–	–	–	–	–	–	
(0xA1)	Reserved	–	–	–	–	–	–	–	–	
(0xA0)	Reserved	–	–	–	–	–	–	–	–	
(0x9F)	Reserved	–	–	–	–	–	–	–	–	
(0x9E)	Reserved	–	–	–	–	–	–	–	–	
(0x9D)	Reserved	–	–	–	–	–	–	–	–	
(0x9C)	Reserved	–	–	–	–	–	–	–	–	
(0x9B)	Reserved	–	–	–	–	–	–	–	–	
(0x9A)	Reserved	–	–	–	–	–	–	–	–	
(0x99)	Reserved	–	–	–	–	–	–	–	–	
(0x98)	Reserved	–	–	–	–	–	–	–	–	
(0x97)	Reserved	–	–	–	–	–	–	–	–	
(0x96)	Reserved	–	–	–	–	–	–	–	–	
(0x95)	Reserved	–	–	–	–	–	–	–	–	
(0x94)	Reserved	–	–	–	–	–	–	–	–	
(0x93)	Reserved	–	–	–	–	–	–	–	–	
(0x92)	Reserved	–	–	–	–	–	–	–	–	
(0x91)	Reserved	–	–	–	–	–	–	–	–	
(0x90)	Reserved	–	–	–	–	–	–	–	–	
(0x8F)	Reserved	–	–	–	–	–	–	–	–	
(0x8E)	Reserved	–	–	–	–	–	–	–	–	
(0x8D)	Reserved	–	–	–	–	–	–	–	–	
(0x8C)	Reserved	–	–	–	–	–	–	–	–	
(0x8B)	OCR1BH	Timer/Counter1 - Output Compare Register B High Byte								138
(0x8A)	OCR1BL	Timer/Counter1 - Output Compare Register B Low Byte								138
(0x89)	OCR1AH	Timer/Counter1 - Output Compare Register A High Byte								138
(0x88)	OCR1AL	Timer/Counter1 - Output Compare Register A Low Byte								138
(0x87)	ICR1H	Timer/Counter1 - Input Capture Register High Byte								138
(0x86)	ICR1L	Timer/Counter1 - Input Capture Register Low Byte								138
(0x85)	TCNT1H	Timer/Counter1 - Counter Register High Byte								138
(0x84)	TCNT1L	Timer/Counter1 - Counter Register Low Byte								138
(0x83)	Reserved	–	–	–	–	–	–	–	–	
(0x82)	TCCR1C	FOC1A	FOC1B	–	–	–	–	–	–	137
(0x81)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	136
(0x80)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	134
(0x7F)	DIDR1	–	–	–	–	–	–	AIN1D	AIN0D	249
(0x7E)	DIDR0	–	–	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	266

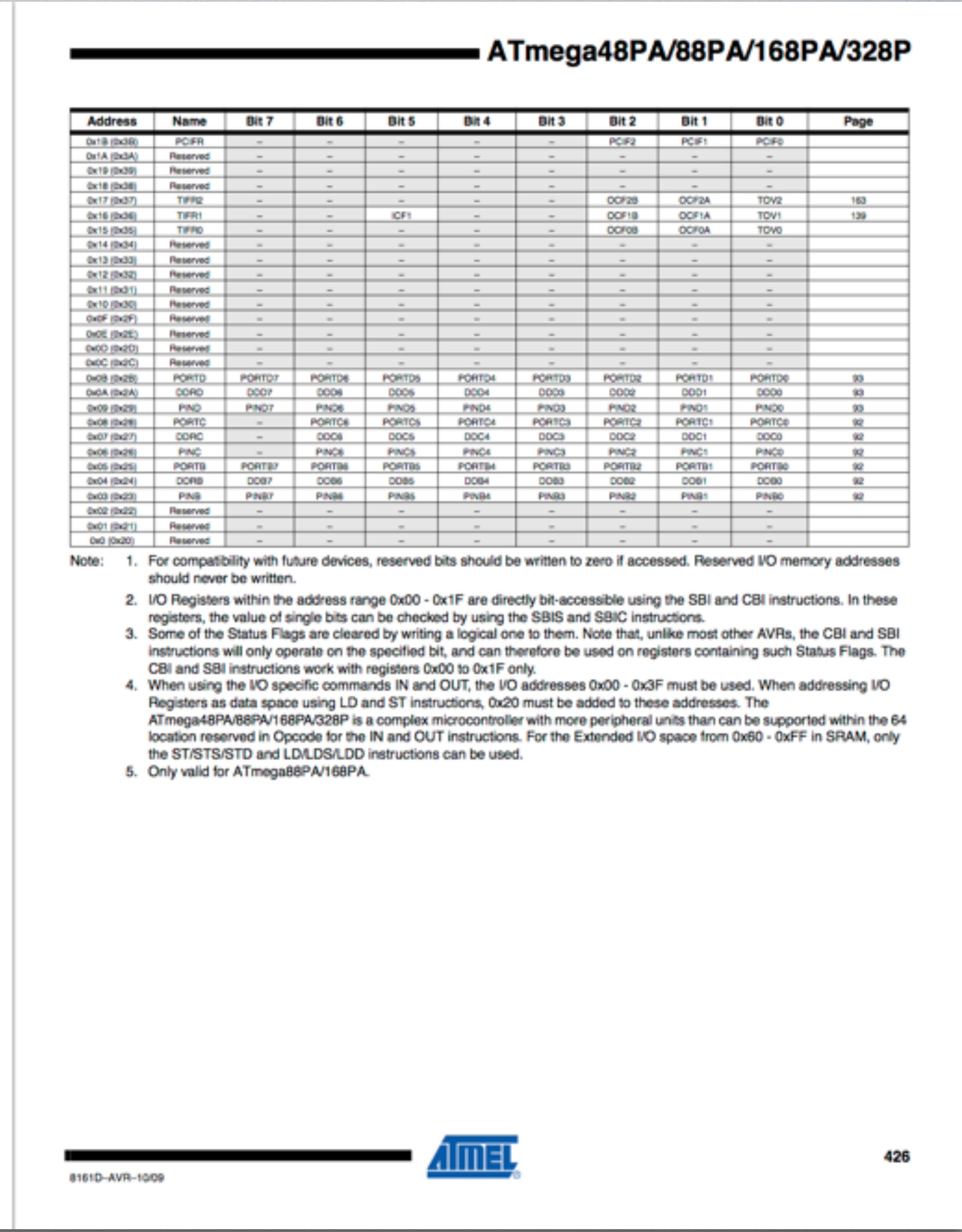
Register-Summary

Datenblatt
Seite 423-425

[Quelle: Atmel, 8-bit AVR Microcontroller]



Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x7D)	Reserved	–	–	–	–	–	–	–	–	
(0x7C)	ADMUX	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	262
(0x7B)	ADCSRB	–	ACME	–	–	–	ADTS2	ADTS1	ADTS0	265
(0x7A)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	263
(0x79)	ADCH	ADC Data Register High byte								265
(0x78)	ADCL	ADC Data Register Low byte								265
(0x77)	Reserved	–	–	–	–	–	–	–	–	
(0x76)	Reserved	–	–	–	–	–	–	–	–	
(0x75)	Reserved	–	–	–	–	–	–	–	–	
(0x74)	Reserved	–	–	–	–	–	–	–	–	
(0x73)	Reserved	–	–	–	–	–	–	–	–	
(0x72)	Reserved	–	–	–	–	–	–	–	–	
(0x71)	Reserved	–	–	–	–	–	–	–	–	
(0x70)	TIMSK2	–	–	–	–	–	OCIE2B	OCIE2A	TOIE2	163
(0x6F)	TIMSK1	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	139
(0x6E)	TIMSK0	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	111
(0x6D)	PCMSK2	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	74
(0x6C)	PCMSK1	–	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	74
(0x6B)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	74
(0x6A)	Reserved	–	–	–	–	–	–	–	–	
(0x69)	EICRA	–	–	–	–	ISC11	ISC10	ISC01	ISC00	71
(0x68)	PCICR	–	–	–	–	–	PCIE2	PCIE1	PCIE0	
(0x67)	Reserved	–	–	–	–	–	–	–	–	
(0x66)	OSCCAL	Oscillator Calibration Register								37
(0x65)	Reserved	–	–	–	–	–	–	–	–	
(0x64)	PRR	PRTWI	PRTIM2	PRTIM0	–	PRTIM1	PRSPI	PRUSART0	PRADC	42
(0x63)	Reserved	–	–	–	–	–	–	–	–	
(0x62)	Reserved	–	–	–	–	–	–	–	–	
(0x61)	CLKPR	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	37
(0x60)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	54
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	9
0x3E (0x5E)	SPH	–	–	–	–	–	(SP10)[5]	SP9	SP8	12
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	12
0x3C (0x5C)	Reserved	–	–	–	–	–	–	–	–	
0x3B (0x5B)	Reserved	–	–	–	–	–	–	–	–	
0x3A (0x5A)	Reserved	–	–	–	–	–	–	–	–	
0x39 (0x59)	Reserved	–	–	–	–	–	–	–	–	
0x38 (0x58)	Reserved	–	–	–	–	–	–	–	–	
0x37 (0x57)	SPMCSR	SPMIE	(RWWSB)[5]	–	(RWWSRE)[5]	BLBSET	PGWRT	PGERS	SELFPRGEN	292
0x36 (0x56)	Reserved	–	–	–	–	–	–	–	–	
0x35 (0x55)	MCUCR	–	BODS	BODSE	PUD	–	–	IVSEL	IVCE	44/68/92
0x34 (0x54)	MCUSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	54
0x33 (0x53)	SMCR	–	–	–	–	SM2	SM1	SM0	SE	40
0x32 (0x52)	Reserved	–	–	–	–	–	–	–	–	
0x31 (0x51)	Reserved	–	–	–	–	–	–	–	–	
0x30 (0x50)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	247
0x2F (0x4F)	Reserved	–	–	–	–	–	–	–	–	
0x2E (0x4E)	SPDR	SPI Data Register								175
0x2D (0x4D)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	174
0x2C (0x4C)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	173
0x2B (0x4B)	GPIOR2	General Purpose I/O Register 2								25
0x2A (0x4A)	GPIOR1	General Purpose I/O Register 1								25
0x29 (0x49)	Reserved	–	–	–	–	–	–	–	–	
0x28 (0x48)	OCR0B	Timer/Counter0 Output Compare Register B								
0x27 (0x47)	OCR0A	Timer/Counter0 Output Compare Register A								
0x26 (0x46)	TCNT0	Timer/Counter0 (8-bit)								
0x25 (0x45)	TCCR0B	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	
0x24 (0x44)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	
0x23 (0x43)	GTCCR	TSM	–	–	–	–	–	PSRASY	PSRSYNC	143/165
0x22 (0x42)	EEARH	(EEPROM Address Register High Byte)[5]								21
0x21 (0x41)	EEARL	EEPROM Address Register Low Byte								21
0x20 (0x40)	EEDR	EEPROM Data Register								21
0x1F (0x3F)	EECR	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	21
0x1E (0x3E)	GPIOR0	General Purpose I/O Register 0								25
0x1D (0x3D)	EIMSK	–	–	–	–	–	–	INT1	INT0	72
0x1C (0x3C)	EIFR	–	–	–	–	–	–	INTF1	INTF0	72



Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x1B (0x3B)	PCIFR	–	–	–	–	–	PCIF2	PCIF1	PCIF0	
0x1A (0x3A)	Reserved	–	–	–	–	–	–	–	–	
0x19 (0x39)	Reserved	–	–	–	–	–	–	–	–	
0x18 (0x38)	Reserved	–	–	–	–	–	–	–	–	
0x17 (0x37)	TIFR2	–	–	–	–	–	OCF2B	OCF2A	TOV2	163
0x16 (0x36)	TIFR1	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	139
0x15 (0x35)	TIFR0	–	–	–	–	–	OCF0B	OCF0A	TOV0	
0x14 (0x34)	Reserved	–	–	–	–	–	–	–	–	
0x13 (0x33)	Reserved	–	–	–	–	–	–	–	–	
0x12 (0x32)	Reserved	–	–	–	–	–	–	–	–	
0x11 (0x31)	Reserved	–	–	–	–	–	–	–	–	
0x10 (0x30)	Reserved	–	–	–	–	–	–	–	–	
0x0F (0x2F)	Reserved	–	–	–	–	–	–	–	–	
0x0E (0x2E)	Reserved	–	–	–	–	–	–	–	–	
0x0D (0x2D)	Reserved	–	–	–	–	–	–	–	–	
0x0C (0x2C)	Reserved	–	–	–	–	–	–	–	–	
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	93
0x0A (0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	93
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	93
0x08 (0x28)	PORTC	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	92
0x07 (0x27)	DDRC	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	92
0x06 (0x26)	PINC	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	92
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	92
0x04 (0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	92
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	92
0x02 (0x22)	Reserved	–	–	–	–	–	–	–	–	
0x01 (0x21)	Reserved	–	–	–	–	–	–	–	–	
0x0 (0x20)	Reserved	–	–	–	–	–	–	–	–	

Note: 1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

2. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.

3. Some of the Status Flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such Status Flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

4. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega48PA/88PA/168PA/328P is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

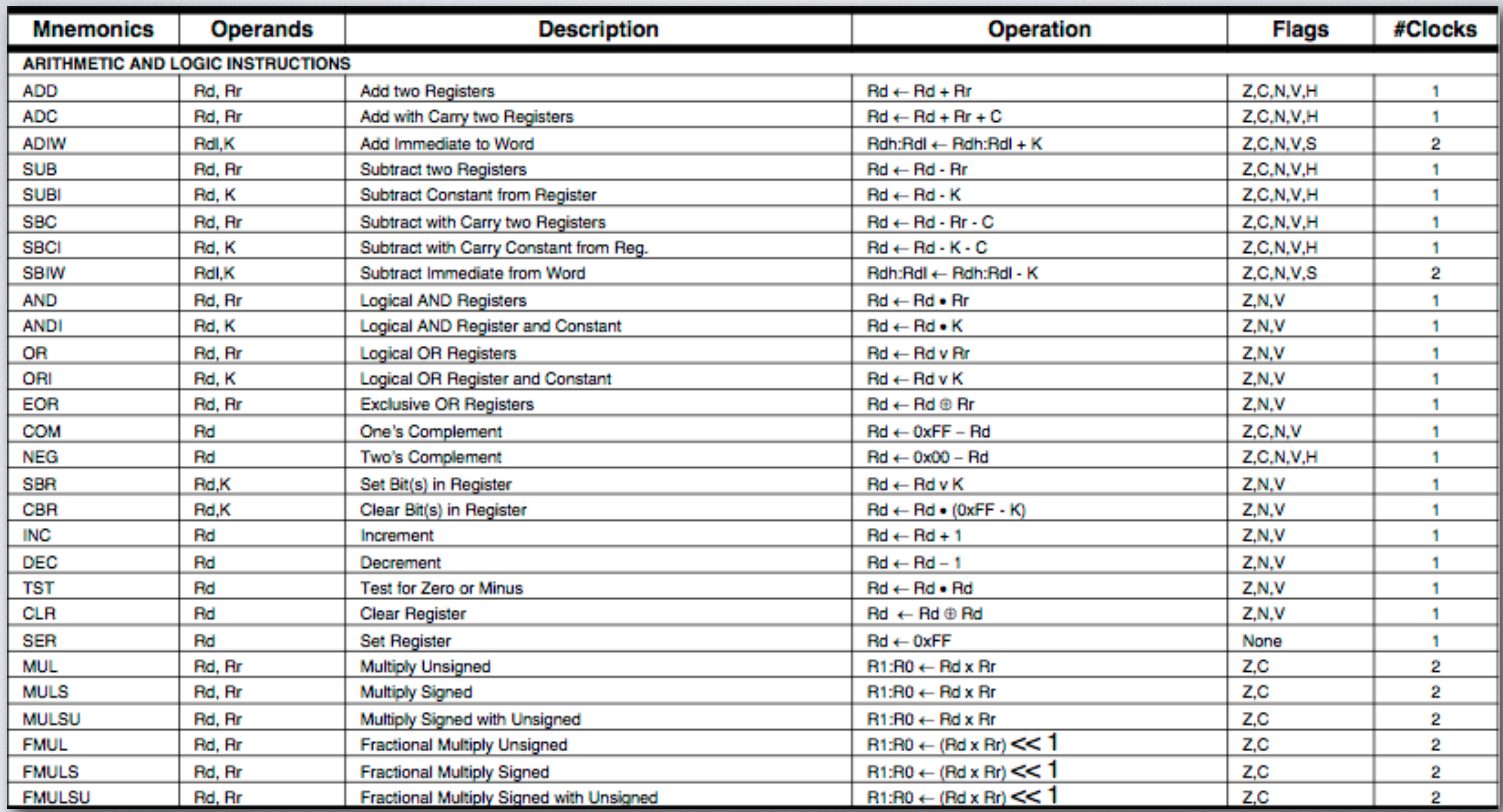
5. Only valid for ATmega88PA/168PA.

Register-Summary

Datenblatt
Seite 423-425

[Quelle: Atmel, 8-bit AVR Microcontroller]

Befehlsübersicht

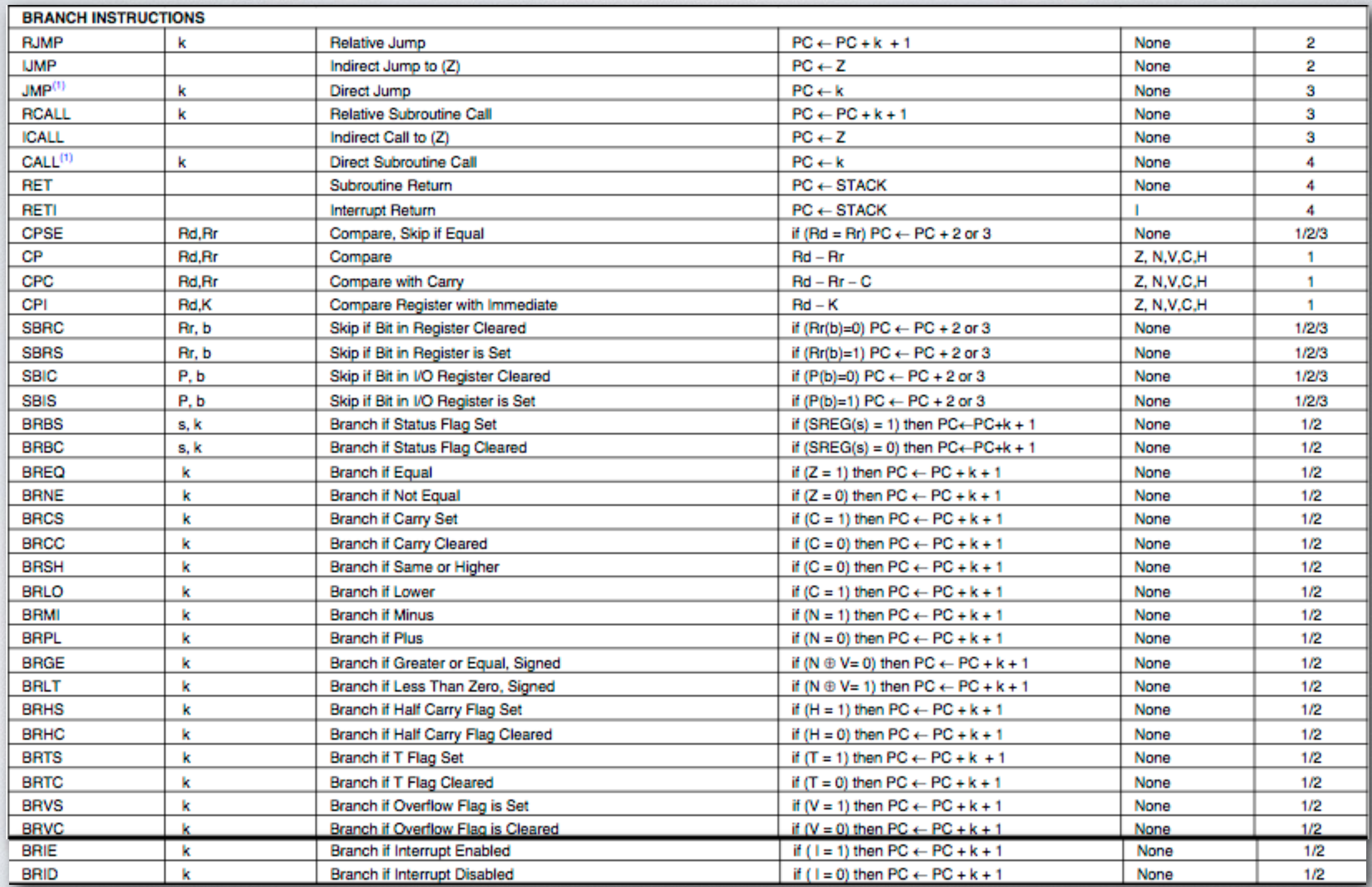


Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← 0xFF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • (0xFF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← 0xFF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2

Datenblatt
Seite 427

[Quelle: Atmel, 8-bit AVR Microcontroller]

Befehlsübersicht

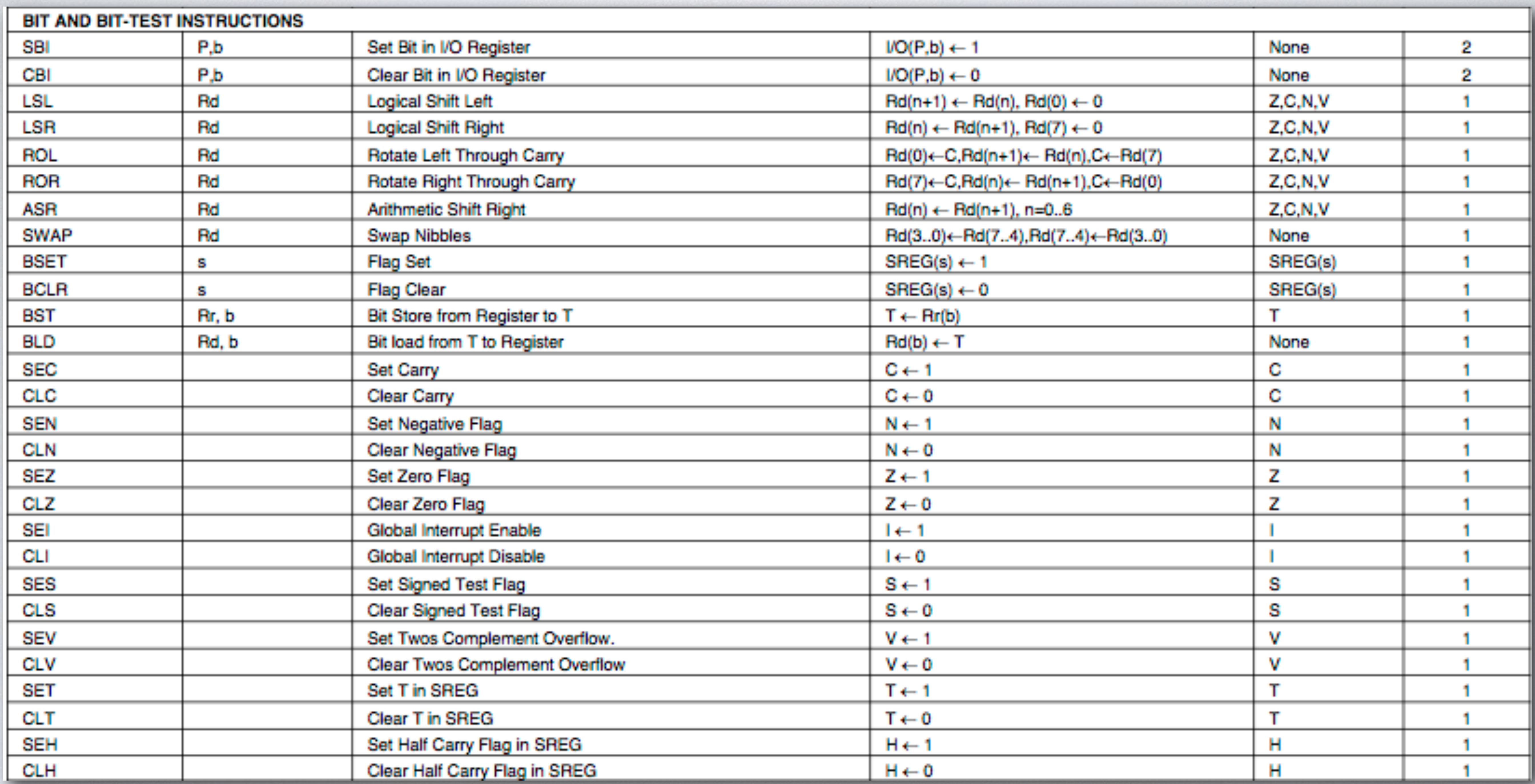


BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP[1]	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL[1]	k	Direct Subroutine Call	PC ← k	None	4
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2

Datenblatt Seite 427

[Quelle: Atmel, 8-bit AVR Microcontroller]

Befehlsübersicht

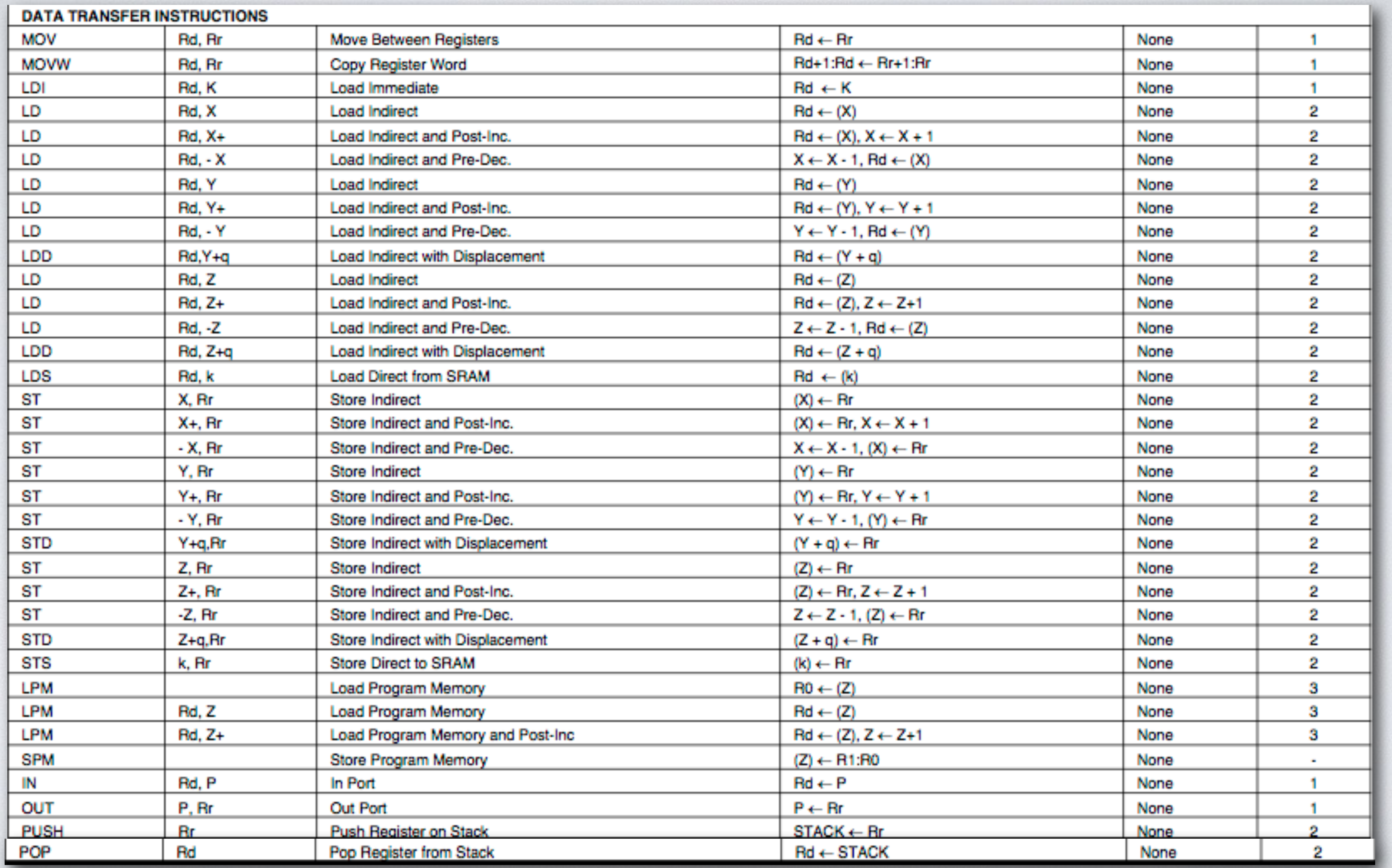


BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1

Datenblatt
Seite 428

[Quelle: Atmel, 8-bit AVR Microcontroller]

Befehlsübersicht

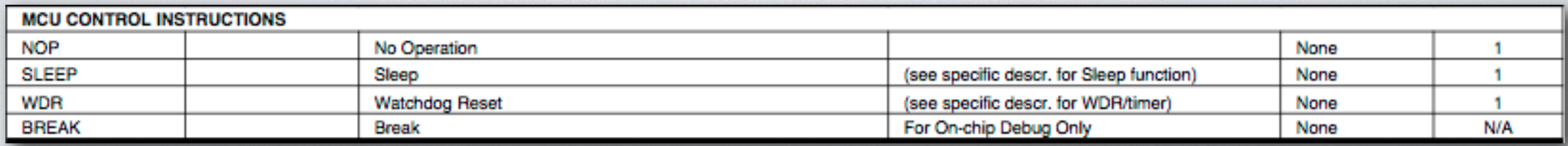


DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2

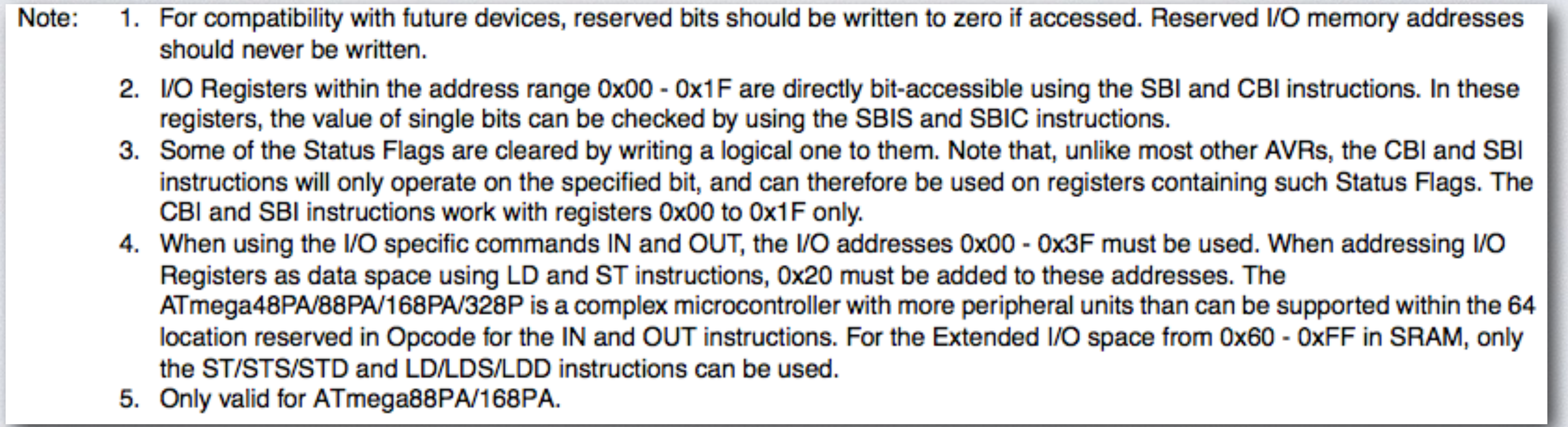
Datenblatt Seite 428-9

[Quelle: Atmel, 8-bit AVR Microcontroller]

Register-Summary



MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-chip Debug Only	None	N/A



Note: 1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

2. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the Status Flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such Status Flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega48PA/88PA/168PA/328P is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.
5. Only valid for ATmega88PA/168PA.

Datenblatt
Seite 428

[Quelle: Atmel, 8-bit AVR Microcontroller]

8-Bit AVR Instruction Set

Registers and Operands

Rd:	Destination (and source) register in the Register File
Rr:	Source register in the Register File
R:	Result after instruction is executed
K:	Constant data
k:	Constant address
b:	Bit in the Register File or I/O Register (3-bit)
s:	Bit in the Status Register (3-bit)
X,Y,Z:	Indirect Address Register
	(X=R27:R26, Y=R29:R28 and Z=R31:R30)
A:	I/O location address
q:	Displacement for direct addressing (6-bit)

Datenblatt
Seite 1

8-Bit AVR Instruction Set

RAMPX, RAMPY, RAMPZ

Registers concatenated with the X-, Y-, and Z-registers enabling indirect addressing of the whole data space on MCUs with more than 64K bytes data space, and constant data fetch on MCUs with more than 64K bytes program space.

RAMPD

Register concatenated with the Z-register enabling direct addressing of the whole data space on MCUs with more than 64K bytes data space.

EIND

Register concatenated with the Z-register enabling indirect jump and call to the whole program space on MCUs with more than 64K words (128K bytes) program space.

Stack

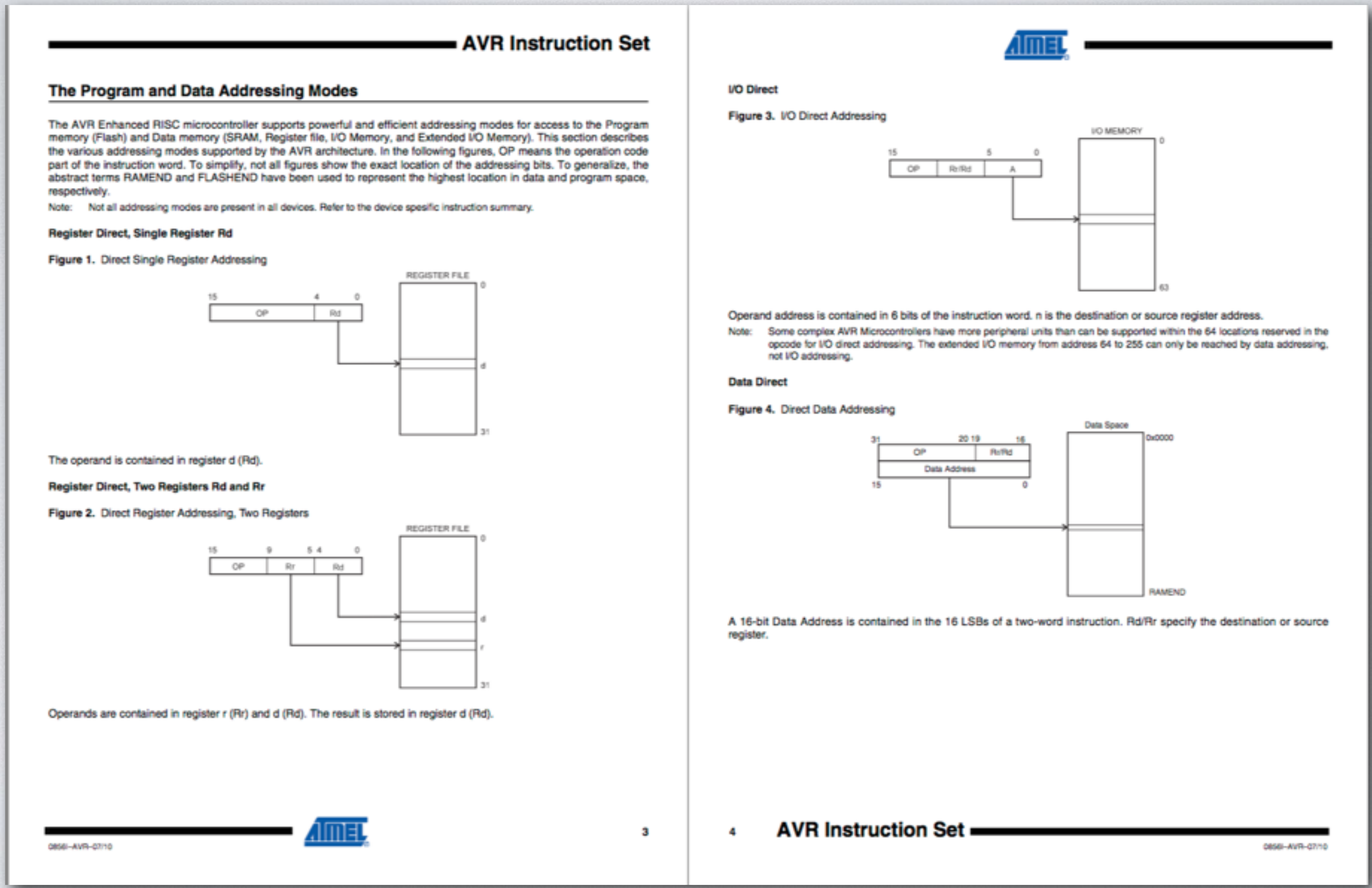
STACK:	Stack for return address and pushed registers
SP:	Stack Pointer to STACK

Flags

⇔:	Flag affected by instruction
0:	Flag cleared by instruction
1:	Flag set by instruction
-:	Flag not affected by instruction

[Quelle: 8-bit AVR Instruction Set]

Addressing Modes



AVR Instruction Set

The Program and Data Addressing Modes

The AVR Enhanced RISC microcontroller supports powerful and efficient addressing modes for access to the Program memory (Flash) and Data memory (SRAM, Register file, I/O Memory, and Extended I/O Memory). This section describes the various addressing modes supported by the AVR architecture. In the following figures, OP means the operation code part of the instruction word. To simplify, not all figures show the exact location of the addressing bits. To generalize, the abstract terms RAMEND and FLASHEND have been used to represent the highest location in data and program space, respectively.

Note: Not all addressing modes are present in all devices. Refer to the device spesific instruction summary.

Register Direct, Single Register Rd

Figure 1. Direct Single Register Addressing

The operand is contained in register d (Rd).

Register Direct, Two Registers Rd and Rr

Figure 2. Direct Register Addressing, Two Registers

Operands are contained in register r (Rr) and d (Rd). The result is stored in register d (Rd).

0856I–AVR–07/10 3

I/O Direct

Figure 3. I/O Direct Addressing

Operand address is contained in 6 bits of the instruction word. n is the destination or source register address.

Note: Some complex AVR Microcontrollers have more peripheral units than can be supported within the 64 locations reserved in the opcode for I/O direct addressing. The extended I/O memory from address 64 to 255 can only be reached by data addressing, not I/O addressing.

Data Direct

Figure 4. Direct Data Addressing

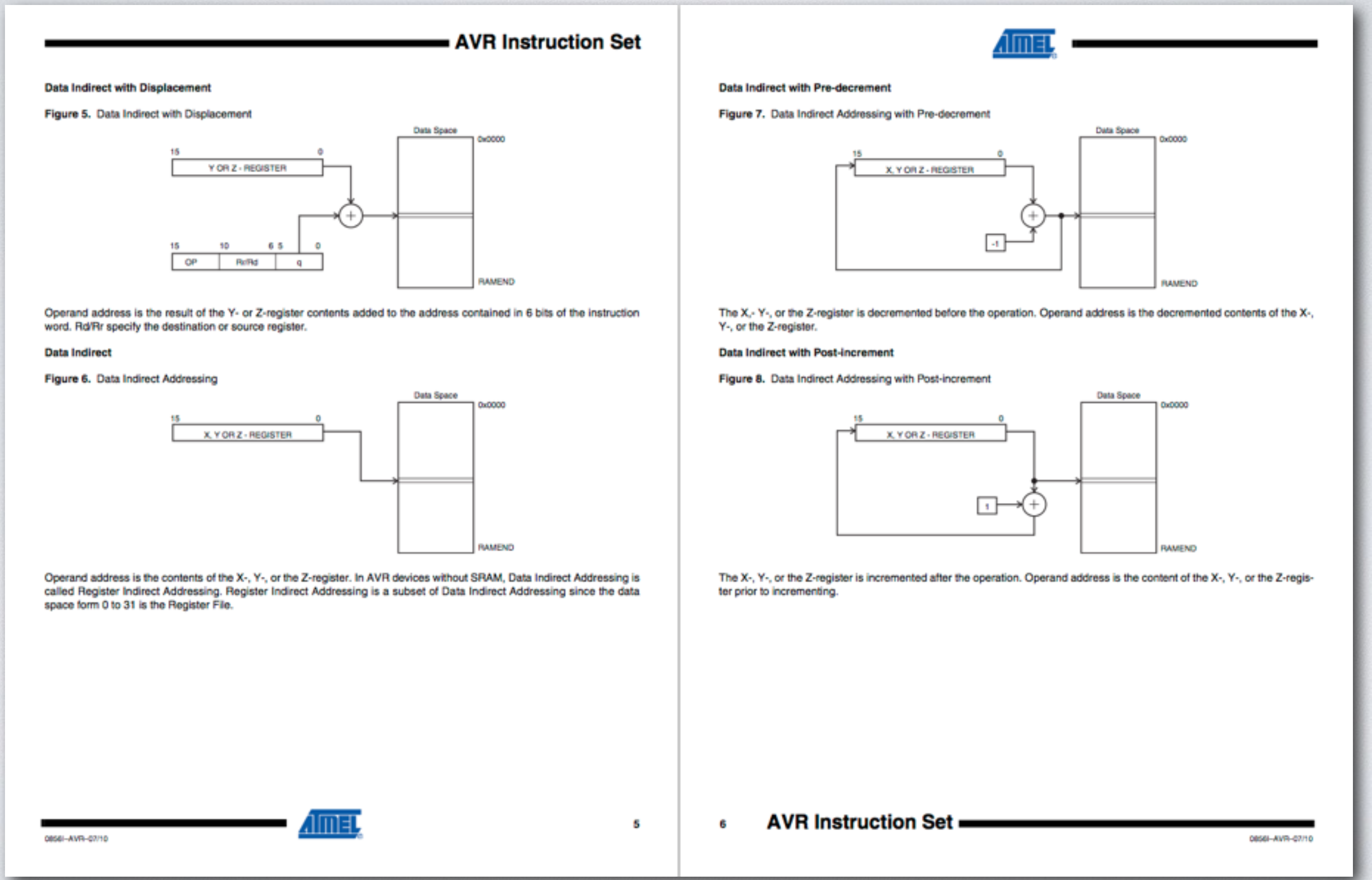
A 16-bit Data Address is contained in the 16 LSBs of a two-word instruction. Rd/Rr specify the destination or source register.

4 AVR Instruction Set 0856I–AVR–07/10

Datenblatt Seite 3-4

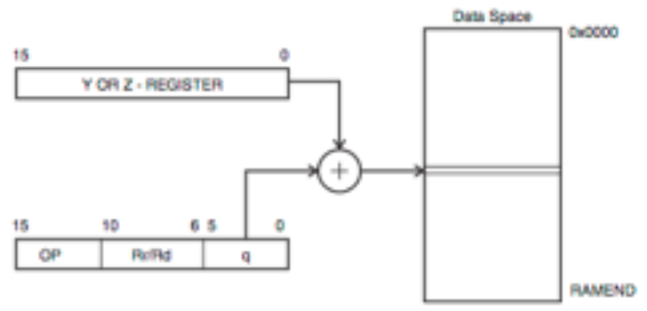
[Quelle: 8-bit AVR Instruction Set]

Addressing Modes



Data Indirect with Displacement

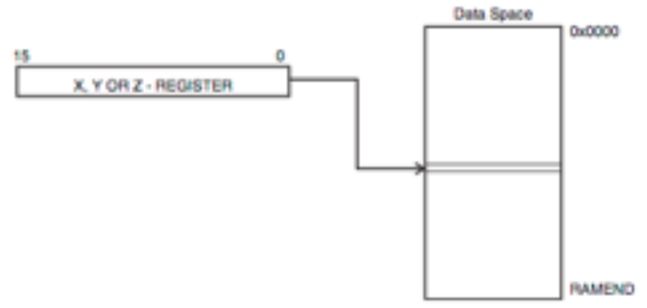
Figure 5. Data Indirect with Displacement



Operand address is the result of the Y- or Z-register contents added to the address contained in 6 bits of the instruction word. Rd/Rr specify the destination or source register.

Data Indirect

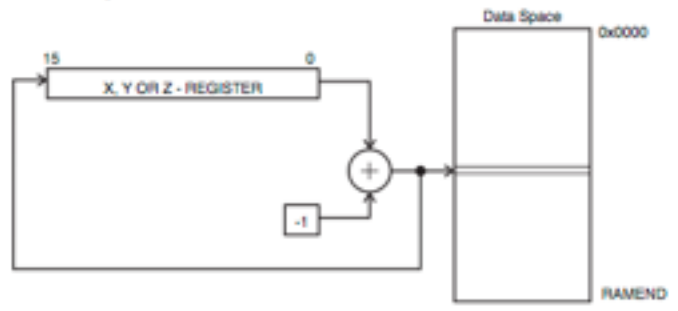
Figure 6. Data Indirect Addressing



Operand address is the contents of the X-, Y-, or the Z-register. In AVR devices without SRAM, Data Indirect Addressing is called Register Indirect Addressing. Register Indirect Addressing is a subset of Data Indirect Addressing since the data space form 0 to 31 is the Register File.

Data Indirect with Pre-decrement

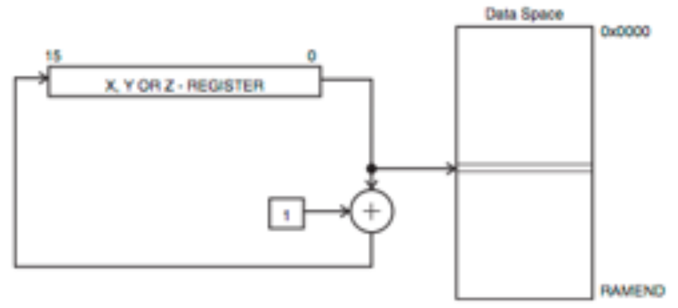
Figure 7. Data Indirect Addressing with Pre-decrement



The X,- Y-, or the Z-register is decremented before the operation. Operand address is the decremented contents of the X-, Y-, or the Z-register.

Data Indirect with Post-increment

Figure 8. Data Indirect Addressing with Post-increment

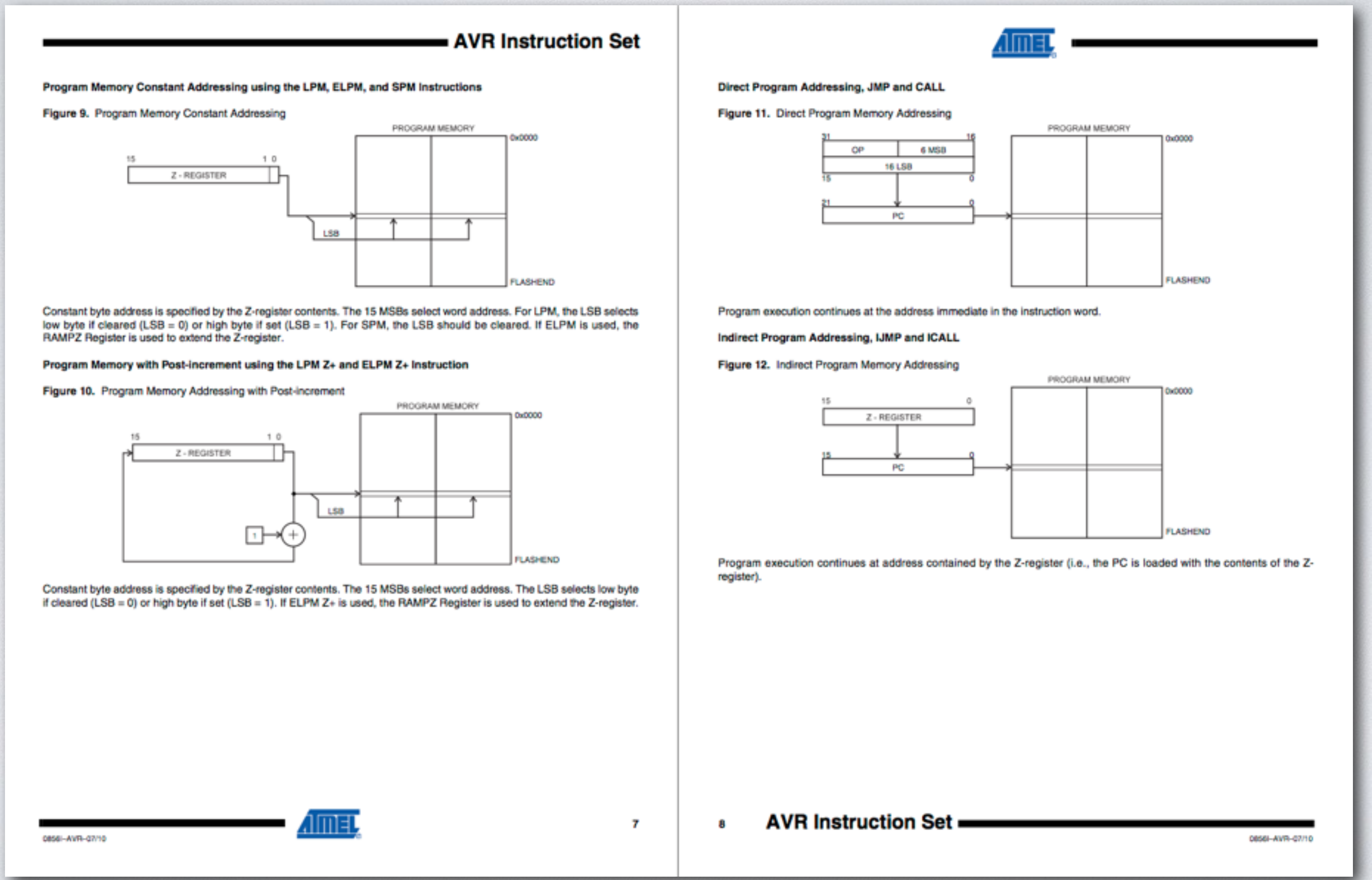


The X-, Y-, or the Z-register is incremented after the operation. Operand address is the content of the X-, Y-, or the Z-register prior to incrementing.

Datenblatt Seite 5-6

[Quelle: 8-bit AVR Instruction Set]

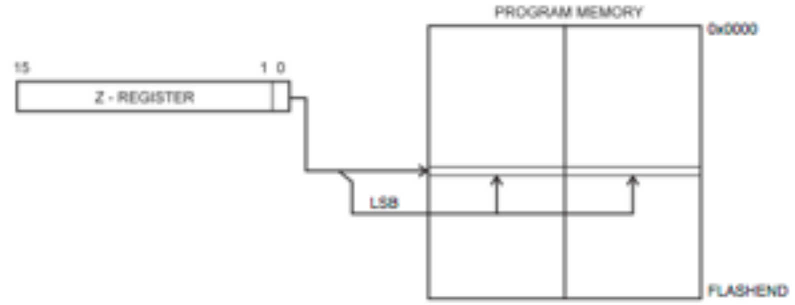
Addressing Modes



AVR Instruction Set

Program Memory Constant Addressing using the LPM, ELPM, and SPM Instructions

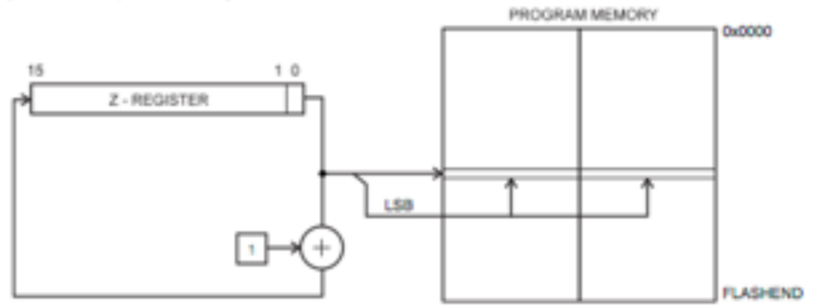
Figure 9. Program Memory Constant Addressing



Constant byte address is specified by the Z-register contents. The 15 MSBs select word address. For LPM, the LSB selects low byte if cleared (LSB = 0) or high byte if set (LSB = 1). For SPM, the LSB should be cleared. If ELPM is used, the RAMPZ Register is used to extend the Z-register.

Program Memory with Post-increment using the LPM Z+ and ELPM Z+ Instruction

Figure 10. Program Memory Addressing with Post-increment

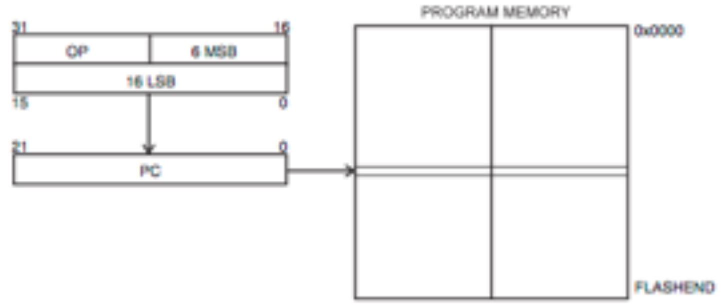


Constant byte address is specified by the Z-register contents. The 15 MSBs select word address. The LSB selects low byte if cleared (LSB = 0) or high byte if set (LSB = 1). If ELPM Z+ is used, the RAMPZ Register is used to extend the Z-register.

0856I–AVR–07/10 7

Direct Program Addressing, JMP and CALL

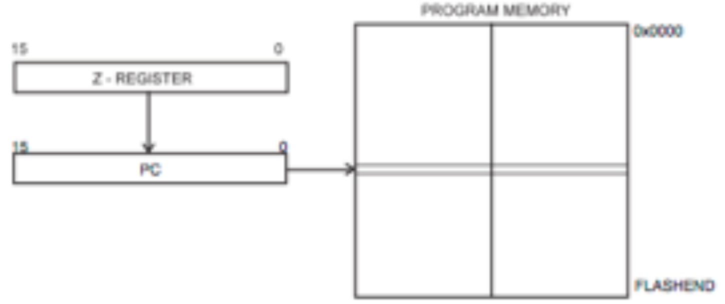
Figure 11. Direct Program Memory Addressing



Program execution continues at the address immediate in the instruction word.

Indirect Program Addressing, IJMP and ICALL

Figure 12. Indirect Program Memory Addressing



Program execution continues at address contained by the Z-register (i.e., the PC is loaded with the contents of the Z-register).

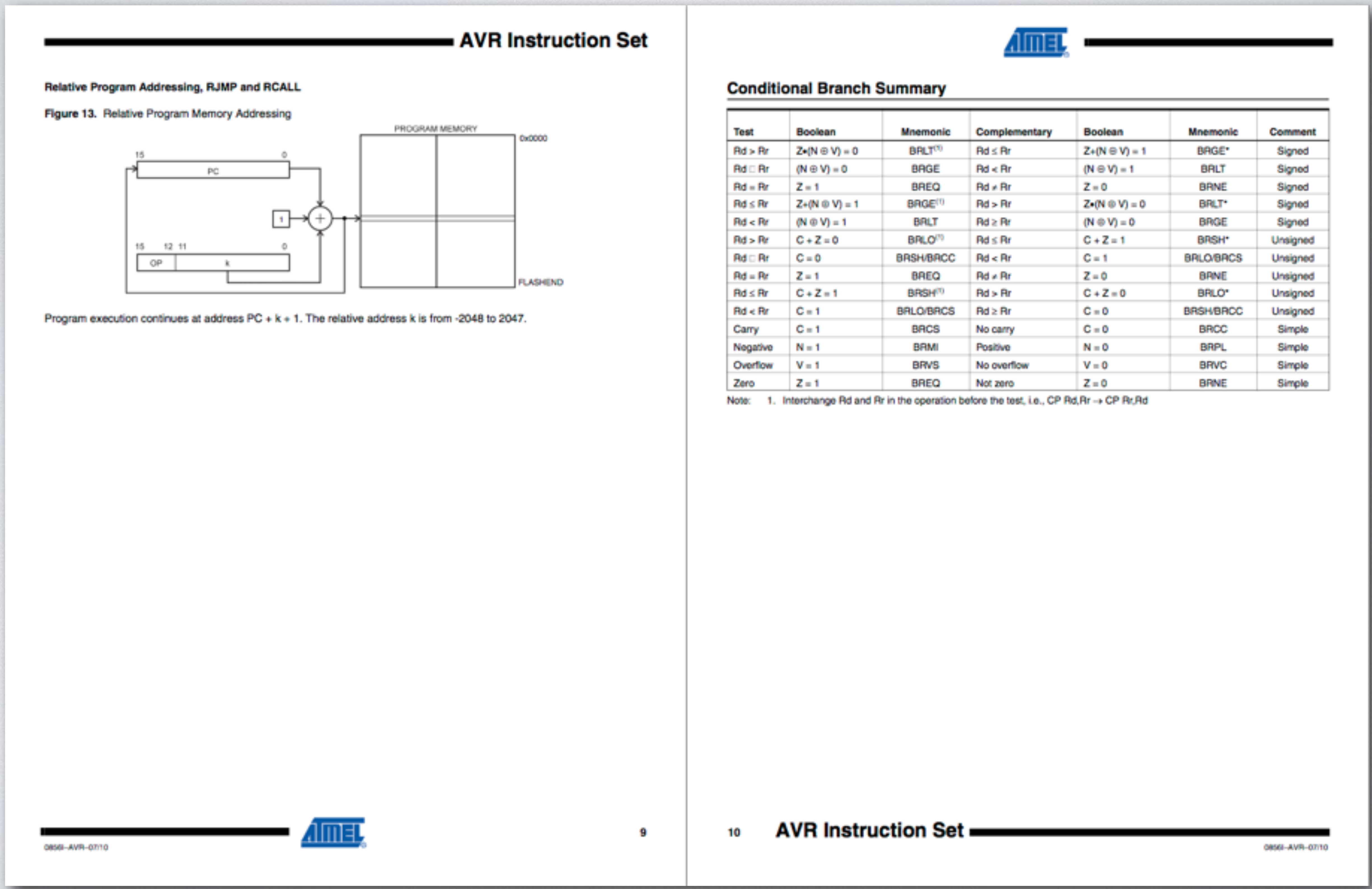
8 **AVR Instruction Set**

0856I–AVR–07/10

Datenblatt Seite 7-8

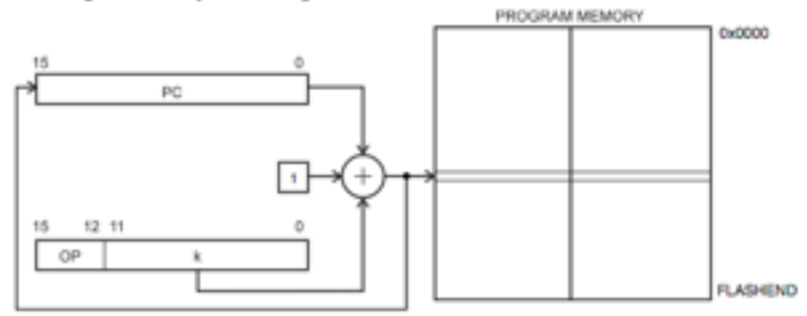
[Quelle: 8-bit AVR Instruction Set]

Addressing Modes und Branch Condition



Relative Program Addressing, RJMP and RCALL

Figure 13. Relative Program Memory Addressing



Program execution continues at address PC + k + 1. The relative address k is from -2048 to 2047.

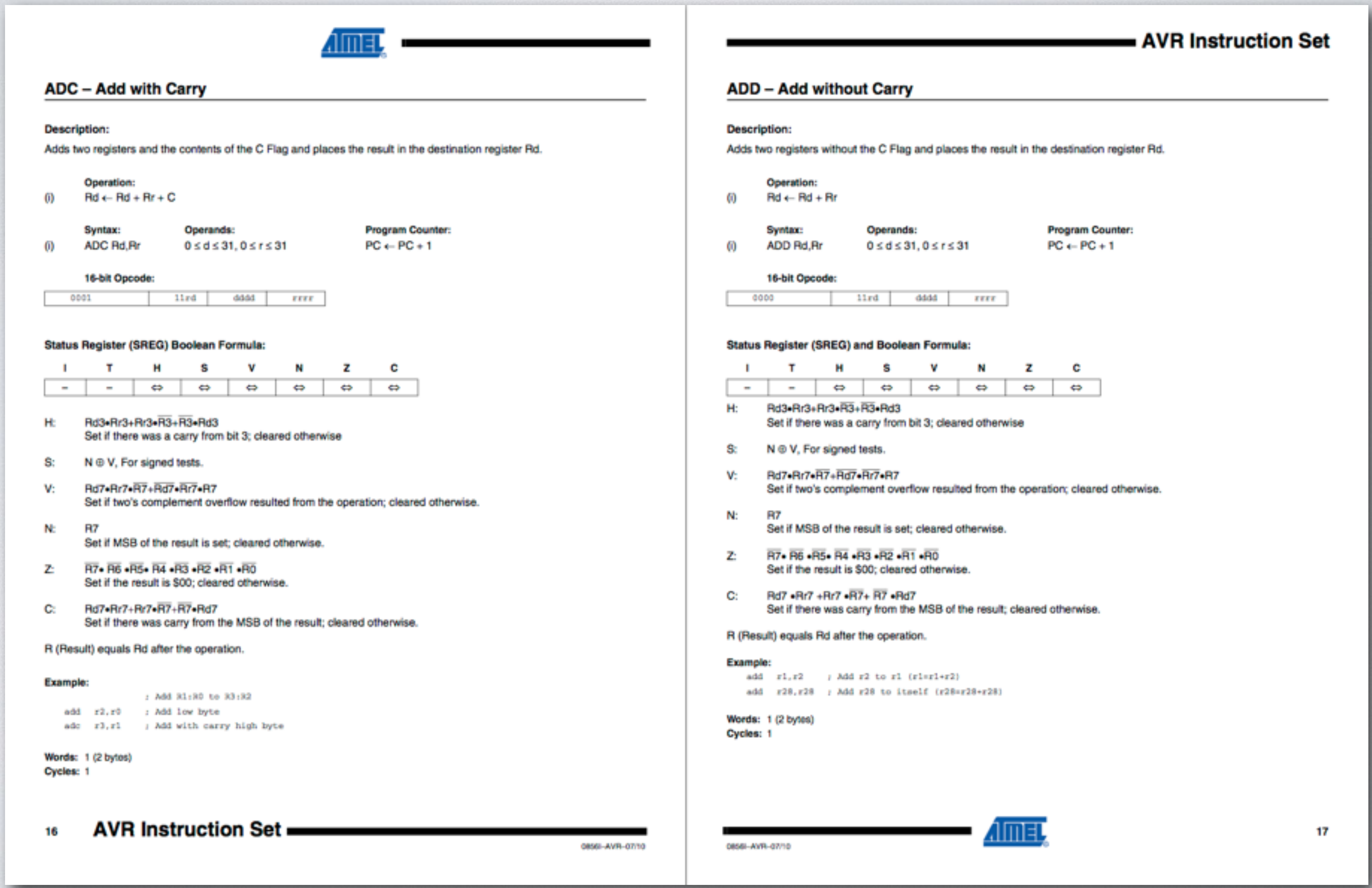
Conditional Branch Summary

Test	Boolean	Mnemonic	Complementary	Boolean	Mnemonic	Comment
$Rd > Rr$	$Z \bullet (N \oplus V) = 0$	BRLT[1]	$Rd \leq Rr$	$Z+(N \oplus V) = 1$	BRGE*	Signed
Rd □ Rr	$(N \oplus V) = 0$	BRGE	$Rd < Rr$	$(N \oplus V) = 1$	BRLT	Signed
$Rd = Rr$	$Z = 1$	BREQ	$Rd \neq Rr$	$Z = 0$	BRNE	Signed
$Rd \leq Rr$	$Z+(N \oplus V) = 1$	BRGE[1]	$Rd > Rr$	$Z \bullet (N \oplus V) = 0$	BRLT*	Signed
$Rd < Rr$	$(N \oplus V) = 1$	BRLT	$Rd \geq Rr$	$(N \oplus V) = 0$	BRGE	Signed
$Rd > Rr$	$C + Z = 0$	BRLO[1]	$Rd \leq Rr$	$C + Z = 1$	BRSH*	Unsigned
Rd □ Rr	$C = 0$	BRSH/BRCC	$Rd < Rr$	$C = 1$	BRLO/BRCS	Unsigned
$Rd = Rr$	$Z = 1$	BREQ	$Rd \neq Rr$	$Z = 0$	BRNE	Unsigned
$Rd \leq Rr$	$C + Z = 1$	BRSH[1]	$Rd > Rr$	$C + Z = 0$	BRLO*	Unsigned
$Rd < Rr$	$C = 1$	BRLO/BRCS	$Rd \geq Rr$	$C = 0$	BRSH/BRCC	Unsigned
Carry	$C = 1$	BRCS	No carry	$C = 0$	BRCC	Simple
Negative	$N = 1$	BRMI	Positive	$N = 0$	BRPL	Simple
Overflow	$V = 1$	BRVS	No overflow	$V = 0$	BRVC	Simple
Zero	$Z = 1$	BREQ	Not zero	$Z = 0$	BRNE	Simple

Note: 1. Interchange Rd and Rr in the operation before the test, i.e., CP Rd,Rr → CP Rr,Rd

Datenblatt Seite 9-10
[Quelle: 8-bit AVR Instruction Set]

AVR-Assembler-Beispielbefehl



AVR Instruction Set

ADC – Add with Carry

Description:

Adds two registers and the contents of the C Flag and places the result in the destination register Rd.

Operation:

(i) Rd ← Rd + Rr + C

	Syntax:	Operands:	Program Counter:
(i)	ADC Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	PC ← PC + 1

16-bit Opcode:

0001	11rd	dddd	rrrr

Status Register (SREG) Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

H: $Rd3 \bullet Rr3 + Rr3 \bullet \overline{R3} + \overline{R3} \bullet Rd3$
Set if there was a carry from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: $Rd7 \bullet Rr7 \bullet \overline{R7} + \overline{Rd7} \bullet \overline{Rr7} \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$00; cleared otherwise.

C: $Rd7 \bullet Rr7 + Rr7 \bullet \overline{R7} + \overline{R7} \bullet Rd7$
Set if there was carry from the MSB of the result; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
                  ; Add R1:R0 to R3:R2
    add   r2,r0   ; Add low byte
    adc   r3,r1   ; Add with carry high byte
```

Words: 1 (2 bytes)
Cycles: 1

16 **AVR Instruction Set**

0856I–AVR–07/10

ADD – Add without Carry

Description:

Adds two registers without the C Flag and places the result in the destination register Rd.

Operation:

(i) Rd ← Rd + Rr

	Syntax:	Operands:	Program Counter:
(i)	ADD Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	PC ← PC + 1

16-bit Opcode:

0000	11rd	dddd	rrrr

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

H: $Rd3 \bullet Rr3 + Rr3 \bullet \overline{R3} + \overline{R3} \bullet Rd3$
Set if there was a carry from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: $Rd7 \bullet Rr7 \bullet \overline{R7} + \overline{Rd7} \bullet \overline{Rr7} \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$00; cleared otherwise.

C: $Rd7 \bullet Rr7 + Rr7 \bullet \overline{R7} + \overline{R7} \bullet Rd7$
Set if there was carry from the MSB of the result; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
    add   r1,r2     ; Add r2 to r1 (r1=r1+r2)
    add   r28,r28   ; Add r28 to itself (r28=r28+r28)
```

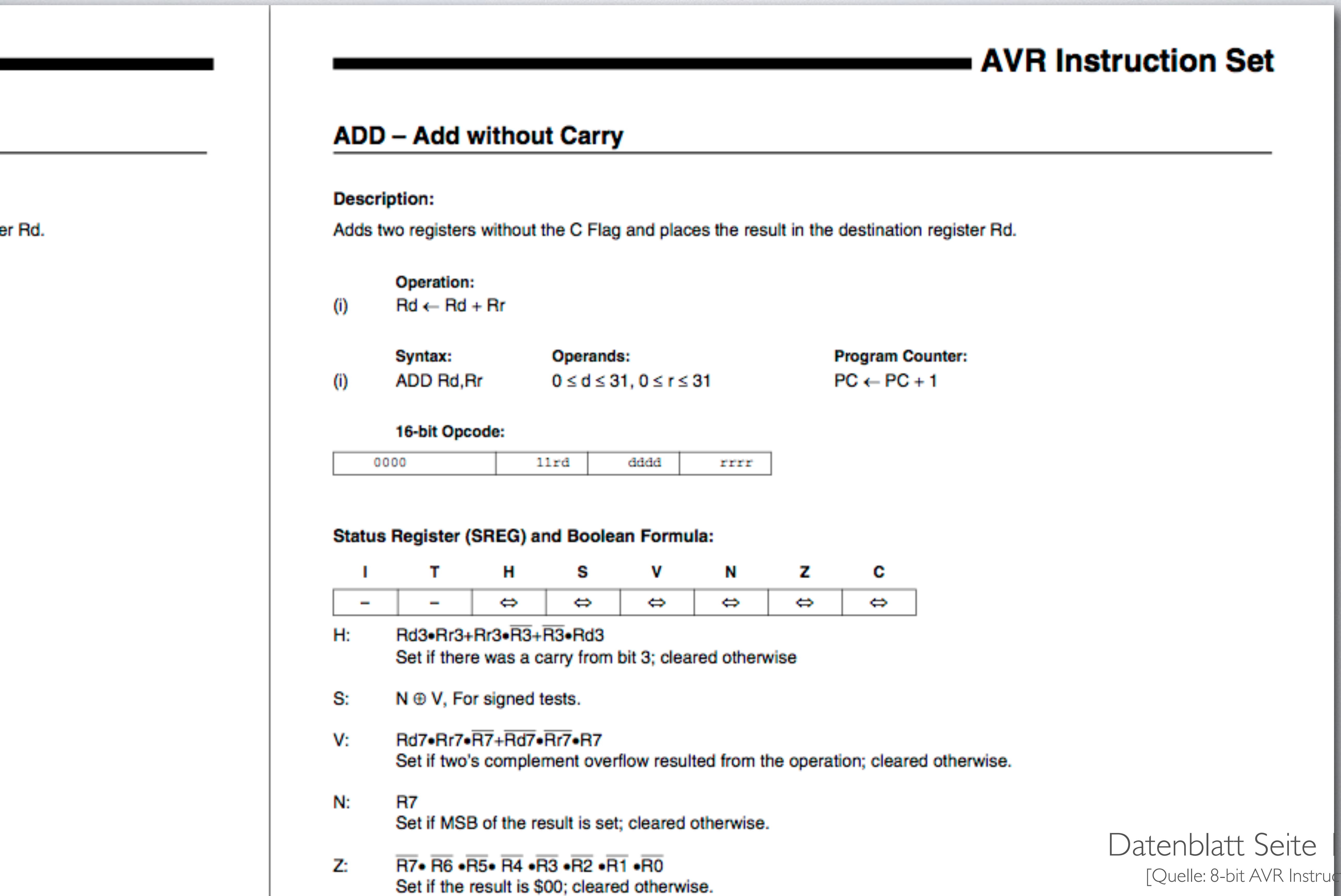
Words: 1 (2 bytes)
Cycles: 1

0856I–AVR–07/10

17

Datenblatt Seite 16-17
[Quelle: 8-bit AVR Instruction Set]

AVR-Assembler-Beispielbefehl



er Rd.

AVR Instruction Set

ADD – Add without Carry

Description:

Adds two registers without the C Flag and places the result in the destination register Rd.

	Operation:
(i)	Rd ← Rd + Rr

	Syntax:	**Operands:**	**Program Counter:**
(i)	ADD Rd,Rr	0 ≤ d ≤ 31, 0 ≤ r ≤ 31	PC ← PC + 1

16-bit Opcode:

0000	11rd	dddd	rrrr

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

H: $Rd3 \bullet Rr3 + Rr3 \bullet \overline{R3} + \overline{R3} \bullet Rd3$
Set if there was a carry from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

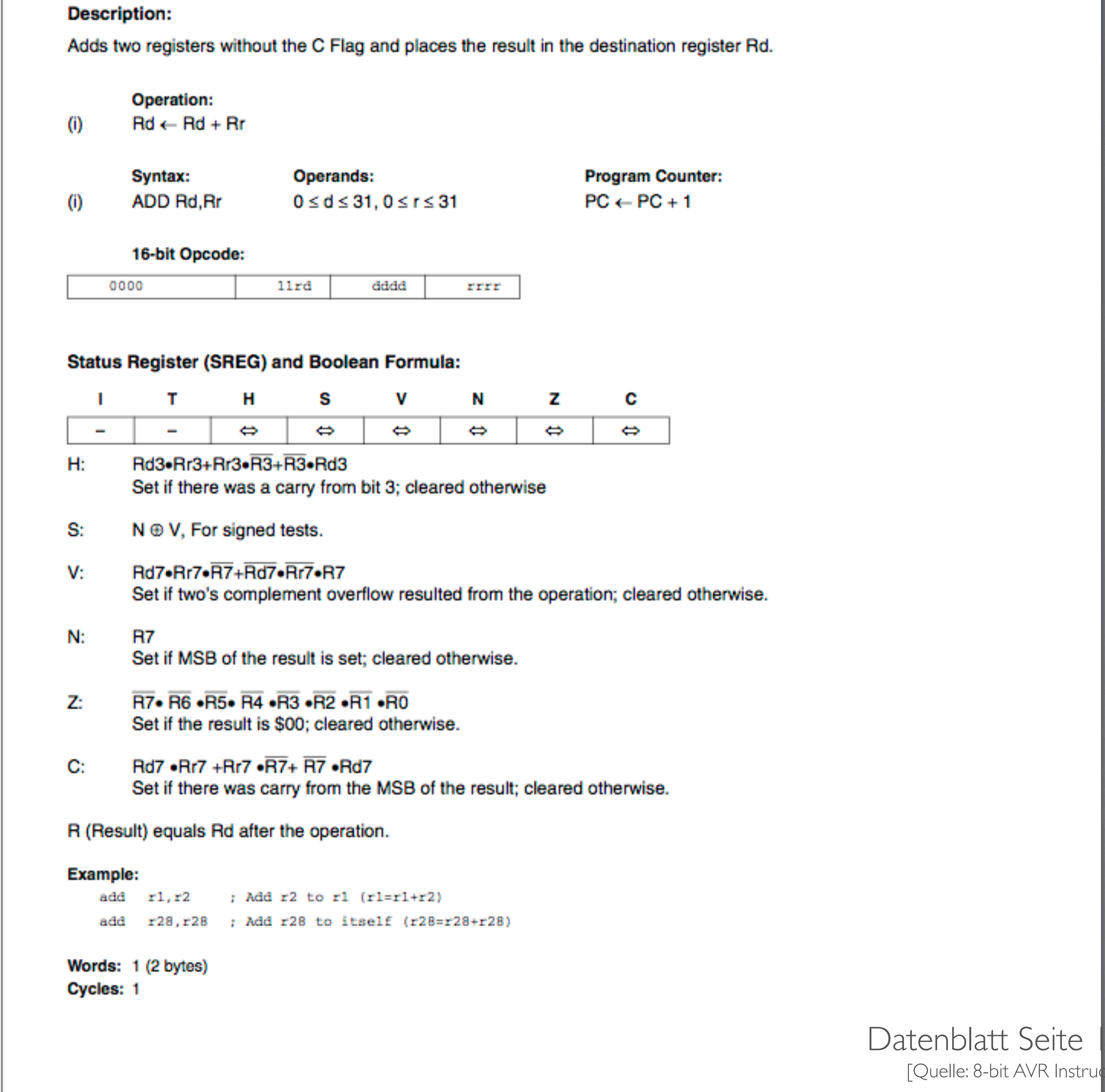
V: $Rd7 \bullet Rr7 \bullet \overline{R7} + \overline{Rd7} \bullet \overline{Rr7} \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$00; cleared otherwise.

Datenblatt Seite 16-17
[Quelle: 8-bit AVR Instruction Set]

er Rd.



Description:

Adds two registers without the C Flag and places the result in the destination register Rd.

	Operation:
(i)	Rd ← Rd + Rr

	Syntax:	**Operands:**	**Program Counter:**
(i)	ADD Rd,Rr	$0 \le d \le 31, 0 \le r \le 31$	PC ← PC + 1

16-bit Opcode:

0000	11rd	dddd	rrrr

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

H: $Rd3 \bullet Rr3 + Rr3 \bullet \overline{R3} + \overline{R3} \bullet Rd3$
Set if there was a carry from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: $Rd7 \bullet Rr7 \bullet \overline{R7} + \overline{Rd7} \bullet \overline{Rr7} \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: $R7$
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$00; cleared otherwise.

C: $Rd7 \bullet Rr7 + Rr7 \bullet \overline{R7} + \overline{R7} \bullet Rd7$
Set if there was carry from the MSB of the result; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
    add   r1,r2     ; Add r2 to r1 (r1=r1+r2)
    add   r28,r28   ; Add r28 to itself (r28=r28+r28)
```

Words: 1 (2 bytes)
Cycles: 1

Datenblatt Seite 16-17
[Quelle: 8-bit AVR Instruction Set]

Arduino - AVR-Assembler

- Hinweise zu Assembler mit der AVR Toolchain:
    - http://www.nongnu.org/avr-libc/user-manual/assembler.html

- Assembler-Programm enthält
    - AVR-Befehle und
    - Anweisungen für Linker, z.B. `.section, global, .end.`

Arduino - Blink in AVR-Assembler

34 Byte

```
    .section .text
    .global .main

main:
    cli                 ; no interrupts
    sbi 0x04, 5         ; set bit 5 of i/o reg 0x04 (portB5 output)
loop:
    sbi 0x05, 5         ; set bit 5 of 0x05 (portB5=led on)
    rcall wait          ; wait
    cbi 0x05, 5         ; clear bit 5 of 0x05 (portB5=led off)
    rcall wait          ; wait
    rjmp loop           ; jump back, endless loop

wait:                   ; just waiting
    clr r26             ; clear r24, r25, r26
waitCount:              ; inner wait loop
    clr r24
    clr r25
waitLoop:
    adiw r24, 1         ; inc word r24,25
    brvs waitNext       ; branch to next if overflow
    rjmp waitLoop       ; otherwise to waitLoop
waitNext:
    inc r26             ; inc r26
    cpi r26, 0x40       ; compare with 0x40
    brlo waitCount      ; branch to waitCount if lower (inner loop)
waitEnd:
    ret                 ; return from sub function

    .end
```

Arduino - Blink in C

368 Byte

gcc mit Option -O1

```
// Arduino Uno PIN 13 is connected to PortB5 of ATMega328P

typedef unsigned char uint8_t; // set type for unsigned int 8

void simpleDelay(unsigned int f_iterations) {
    // loop with nop for delay
    for (unsigned long i = 0; i < f_iterations; ++i) {
        for (volatile unsigned long j = 0; j < 1000; ++j) ;
    }
}

int main() {
    // set direction of DDRB5
    uint8_t *portAdd = (uint8_t*) 0x24; // DDRB address
    *portAdd |= 0x20; // set bit 5

    for (;;) { // infinite loop
        uint8_t *portB = (uint8_t*) 0x25; // PortB address
        *portB |= 0x20; // set bit 5
        simpleDelay(1000);
        *portB &= 0xDF; // clear bit 5
        simpleDelay(1000);
    }
}
```

```
ldi r28, 0x25
ldi r29, 0x00
ld  r24, Y
ori r24, 0x20
st  Y,   r24
```

Arduino - Inline-Assembler

```
// Arduino Uno PIN 13 is connected to PortB5 of ATMega328P

typedef unsigned char uint8_t; // set type for unsigned int 8

void simpleDelay(unsigned int f_iterations) {
    // loop with nop for delay
    for (unsigned long i = 0; i < f_iterations; ++i) {
        for (volatile unsigned long j = 0; j < 1000; ++j) ;
    }
}

int main() {
    // set direction of DDRB5
    uint8_t *portAdd = (uint8_t*) 0x24; // DDRB address
    *portAdd |= 0x20; // set bit 5

    for (;;) { // infinite loop
        uint8_t *portB = (uint8_t*) 0x25; // PortB address
        *portB |= 0x20; // set bit 5
        simpleDelay(1000);
        asm volatile("cbi 0x5, 5\n\t"); // clear bit 5
        simpleDelay(1000);
    }
}
```

Inline Assembler

Zusammenfassung

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

in aller Kürze

Beispiel einer Anwendung: Bratenthermometer

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

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

Literatur / Quellen

- **Atmel**, *8-bit AVR Instruction Set*, 2010
- **Atmel**, *8-bit AVR Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash,* 2009
- 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: 03.03.2014**