

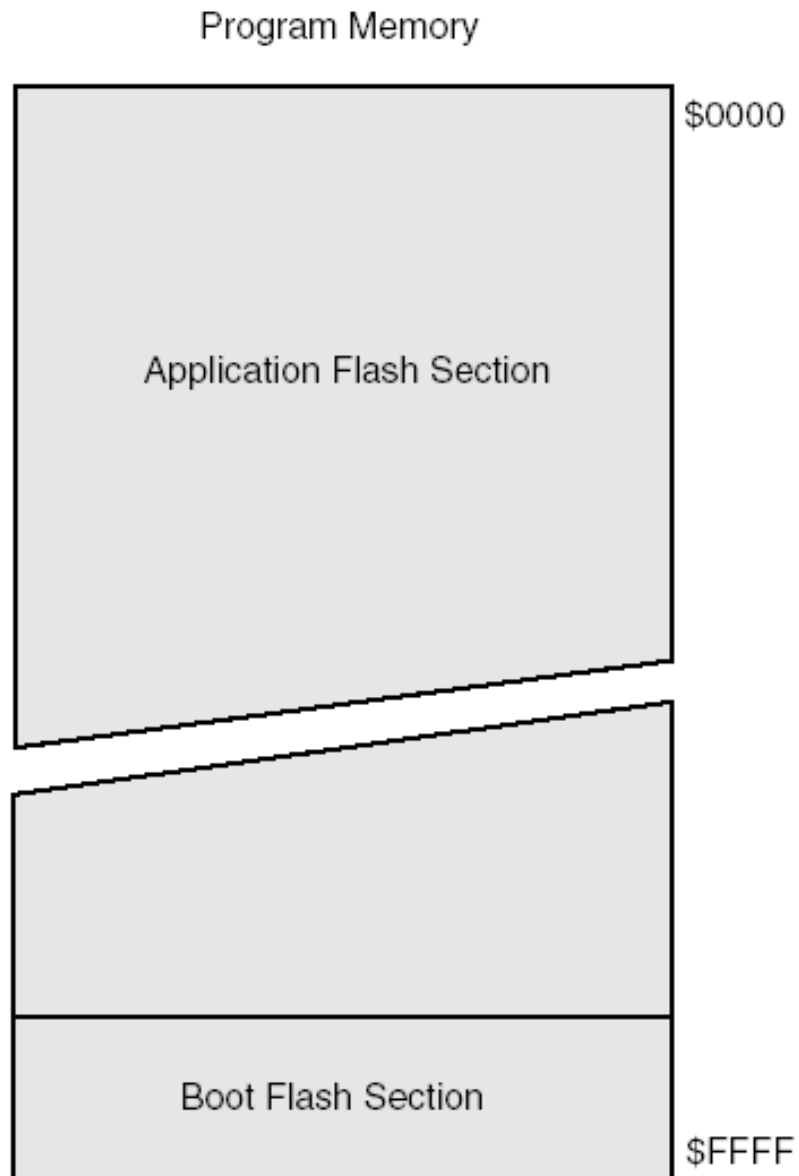
ATmega128 Memory

**(Flash memory, SRAM,
EEPROM & I/O memory)**

ATmega128 Memory

- **AVR architecture has two main memory spaces**
 - **Program memory : Flash memory**
 - **Data memory : SRAM, EEPROM**
- **In-System Reprogrammable Flash Program Memory**
 - **128 Kbytes on-chip flash memory (ATmega128)**
 - **Organized as 64K x 16 (since all instructions are 16 or 32 bits wide)**
 - **Divided into two sections**
 - **Boot Loader Program section**
 - **Application Program section**

Program Memory Map



Boot Loader program (text p.275)

The Boot Loader Support provides a real Read-While-Write Self-Programming mechanism for downloading and uploading program code by the MCU itself. This feature allows flexible application software updates controlled by the MCU using a Flash-resident Boot Loader program. The Boot Loader program can use any available data interface and associated protocol to read code and write (program) that code into the Flash memory, or read the code from the program memory. The program code within the Boot Loader section has the capability to write into the entire Flash, including the Boot Loader memory. The Boot Loader can thus even modify itself, and it can also erase itself from the code if the feature is not needed anymore. The size of the Boot Loader memory is configurable with fuses and the Boot Loader has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

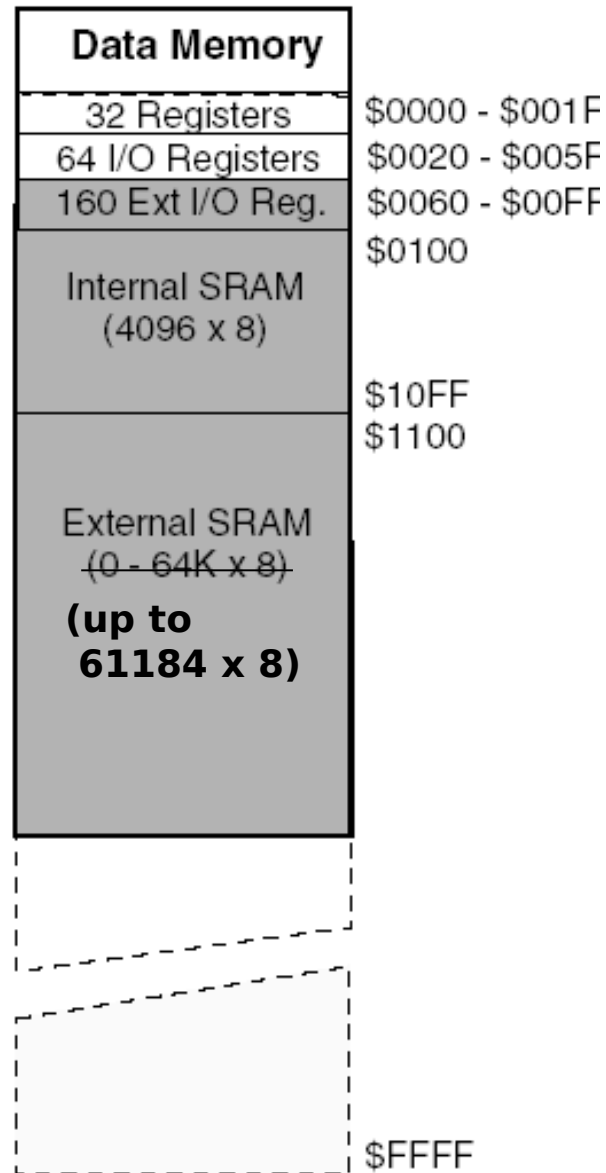
Flash Program Memory (cont'd)

- **Endurance : 10,000 write/erase cycles**
- **Boot Program section is protected by **Boot Lock bits** (refer to p.277)**
- **Flash memory programming is supported by **SPI, JTAG, or Parallel Programming mode****
- **Constant tables can be allocated within the entire program memory address space by using **LPM** (Load Program Memory) and **ELPM** (Extended Load Program Memory) instructions**

SRAM Data Memory

- **Size : 4096 bytes (internal)
up to 64Kbytes (external - optional)**
- **Internal Memory : 4352 locations**
 - **First 32 locations for register file**
 - **Next 64 locations for standard I/O memory**
(can be accessed by I/O specific instructions or memory access instructions)
 - **Next 160 locations for extended I/O memory**
(can be accessed by only memory access instructions -- LD/LDS/LDD, ST/STS/STD instructions)

Data Memory Map



SRAM Data Memory (cont'd)

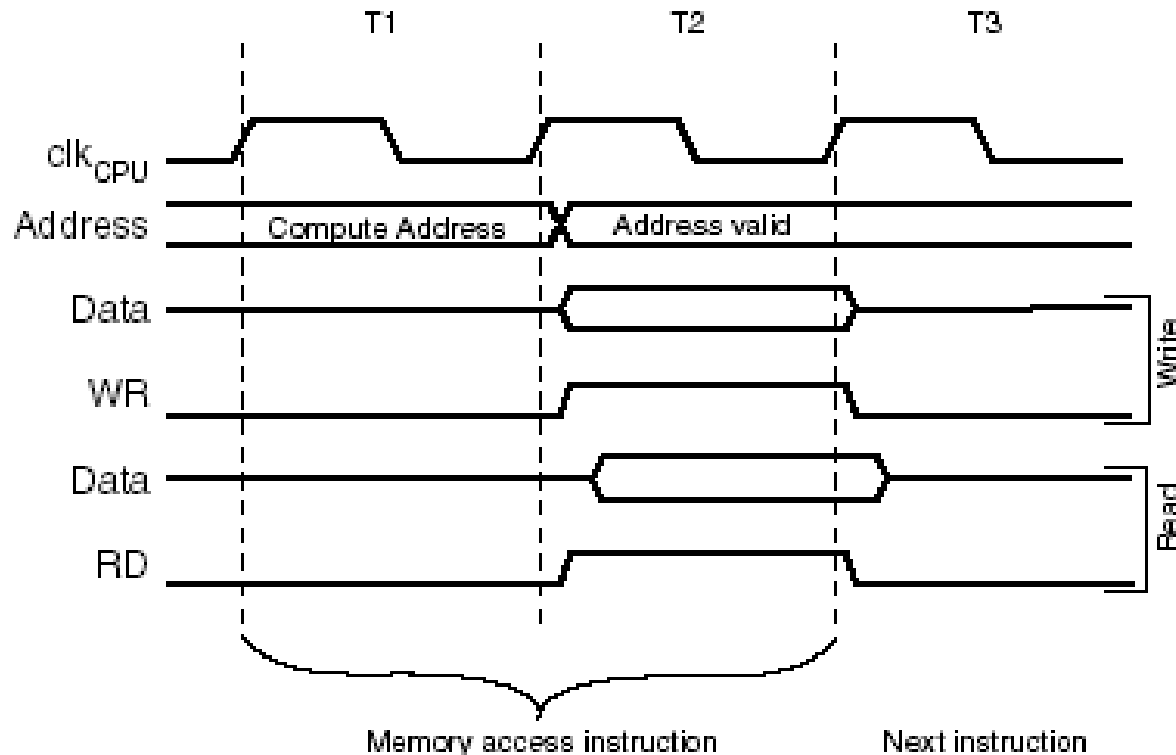
- **Optional external SRAM**
 - **Size : up to 61184 bytes**
(0x1100~0xFFFF : occupies the remaining address locations in the 64K address space) \approx 60KBytes
 - **Access time takes one additional clock cycle per byte compared to access of the internal SRAM**

SRAM Data Memory (cont'd)

- **Addressing modes for data memory :**
 - **Direct** (can reaches the entire data space)
 - **Indirect with displacement**
 - **Indirect**
 - **Indirect with pre-decrement**
 - **Indirect with post-increment**
(X, Y, Z registers are used for automatic pre-decrement and post-increment)

SRAM Data Memory (cont'd)

- **On-chip data memory access time : two CPU clock cycles**



EEPROM Data Memory

- **Size : 4K bytes**
- **Address space : separate from SRAM**
- **Endurance : 100,000 write/erase cycles**
- **EEPROM programming (write) time : 8.5 ms**
- **Access(read/write) operation can be done by CPU through the following registers :**
 - **EEPROM Address Register**
 - **EEPROM Data Register**

EEPROM Data Memory (cont'd)

- **EEPROM Address Register (EEAR)**

- **BIT 15..12 : reserved bits**
- **BIT 11..0 : EEPROM address (4096 bytes)**

Bit	15	14	13	12	11	10	9	8
	–	–	–	–	EEAR11	EEAR10	EEAR9	EEAR8
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0
	7	6	5	4	3	2	1	0
Read/Write	R	R	R	R	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	0	0	0	0	X	X	X	X
	X	X	X	X	X	X	X	X

EEARH: \$1F(\$3F)

EEARL: \$1E(\$3E)

EEPROM Data Memory (cont'd)

- **EEPROM Data Register (EEDR)**
 - **BIT 7..0 : EEPROM data**

Bit	7	6	5	4	3	2	1	0
	MSB							LSB
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

EEDR : \$1D(\$3D)

EEPROM Data Memory (cont'd)

- EEPROM Control Register (EECR)**

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	EERIE	EEMWE	EEWE	EERE	EECR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	: \$1C(\$3C)
Initial Value	0	0	0	0	0	0	X	0	

- **BIT 7..4 : reserved**
- **BIT 3 : EERIE (EEPROM Ready Interrupt Enable)**
- **BIT 2 : EEMWE (EEPROM Master Write Enable)**
- **BIT 1 : EEWE (EEPROM Write Enable)**
- **BIT 0 : EERE (EEPROM Read Enable)**

EEPROM Data Memory (cont'd)

- **Procedure for EEPROM write operation**
 - a. **Wait until EEWE becomes 0 (previous write is complete)**
 - b. **Wait until SPEN in SPMCSR becomes 0**
 - c. **Write new EEPROM address to EEAR**
 - d. **Write new EEPROM data to EEDR**
 - e. **Write a logical 1 to EEMWE bit while writing 0 to EEWE in EECR**
 - f. **Write a logical 1 to EEWE bit (within 4 clock cycles after setting EEMWE)**

Assembly code for EEPROM write

Assembly Code Example

```
EEPROM_write:
    ; Wait for completion of previous write
    sbic EECR,EWE
    rjmp EEPROM_write
    ; Set up address (r18:r17) in address register
    out EEARH, r18
    out EEARL, r17
    ; Write data (r16) to data register
    out EEDR,r16
    ; Write logical one to EEMWE
    sbi EECR,EEMWE
    ; Start eeprom write by setting EWE
    sbi EECR,EWE
    ret
```


C code for EEPROM write

C Code Example

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
    /* Wait for completion of previous write */
    while(EECR & (1<<EEMWE))
        ;
    /* Set up address and data registers */
    EEAR = uiAddress;
    EEDR = ucData;
    /* Write logical one to EEMWE */
    EECR |= (1<<EEMWE);
    /* Start eeprom write by setting EEWE */
    EECR |= (1<<EEWE);
}
```

Assembly code for EEPROM read

Assembly Code Example

```
EEPROM_read:
    ; Wait for completion of previous write
    sbic EECR,EEWE
    rjmp EEPROM_read
    ; Set up address (r18:r17) in address register
    out EEARH, r18
    out EEARL, r17
    ; Start eeprom read by writing EERE
    sbi EECR,EERE
    ; Read data from data register
    in  r16,EEDR
    ret
```

C code for EEPROM read

C Code Example

```
unsigned char EEPROM_read(unsigned int uiAddress)
{
    /* Wait for completion of previous write */
    while(EECR & (1<<EWE))
        ;
    /* Set up address register */
    EEAR = uiAddress;
    /* Start eeprom read by writing EERE */
    EECR |= (1<<EERE);
    /* Return data from data register */
    return EEDR;
}
```

I/O Memory

- All I/Os and peripherals are placed in the I/O space
- All I/O locations are accessed by LD/LDS/LDD and ST/STS/STD instructions : transfer data between 32 working registers and I/O registers (in this case, **addresses are added by \$20 → \$20~\$FF: memory-mapped I/O**)
- I/O registers within the address range **\$00 ~ \$3F** can be accessed by using IN and OUT instructions (**Isolated I/O**)
- I/O registers within the address range **\$00 ~ \$1F**
 - directly bit-accessible using SBI and CBI instructions
 - value of single bit can be checked by using SBIS and SBIC instructions
- Extended I/O space (**\$60 ~ \$FF**) can be

I/O Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
(\$61)	DDRF	DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0
(\$60)	Reserved	–	–	–	–	–	–	–	–
\$3F (\$5F)	SREG	I	T	H	S	V	N	Z	C
\$3E (\$5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8
\$3D (\$5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
\$3C (\$5C)	XDIV	XDIVEN	XDIV6	XDIV5	XDIV4	XDIV3	XDIV2	XDIV1	XDIV0
\$3B (\$5B)	RAMPZ	–	–	–	–	–	–	–	RAMPZ0
\$3A (\$5A)	EICRB	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40
\$39 (\$59)	EIMSK	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
\$38 (\$58)	EIFR	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0
\$37 (\$57)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
\$36 (\$56)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0
\$35 (\$55)	MCUCR	SRE	SRW10	SE	SM1	SM0	SM2	IVSEL	IVCE
\$34 (\$54)	MCUCSR	JTD	–	–	JTRF	WDRF	BORF	EXTRF	PORF
\$33 (\$53)	TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
\$32 (\$52)	TCNT0	Timer/Counter0 (8 Bit)							
\$31 (\$51)	OCR0	Timer/Counter0 Output Compare Register							
\$30 (\$50)	ASSR	–	–	–	–	AS0	TCN0UB	OCR0UB	TCR0UB
\$2F (\$4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10
\$2E (\$4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10
\$2D (\$4D)	TCNT1H	Timer/Counter1 – Counter Register High Byte							
\$2C (\$4C)	TCNT1L	Timer/Counter1 – Counter Register Low Byte							
\$2B (\$4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High Byte							
\$2A (\$4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low Byte							
\$29 (\$49)	OCR1BH	Timer/Counter1 – Output Compare Register B High Byte							
\$28 (\$48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low Byte							
\$27 (\$47)	ICR1H	Timer/Counter1 – Input Capture Register High Byte							
\$26 (\$46)	ICR1L	Timer/Counter1 – Input Capture Register Low Byte							
\$25 (\$45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20
\$24 (\$44)	TCNT2	Timer/Counter2 (8 Bit)							
\$23 (\$43)	OCR2	Timer/Counter2 Output Compare Register							
\$22 (\$42)	OCDR	IDRD/OCDF7	OCDF6	OCDF5	OCDF4	OCDF3	OCDF2	OCDF1	OCDF0
\$21 (\$41)	WDTCSR	–	–	–	WDCE	WDE	WDP2	WDP1	WDP0
\$20 (\$40)	SFIOR	TSM	–	–	–	ACME	PUD	PSR0	PSR321

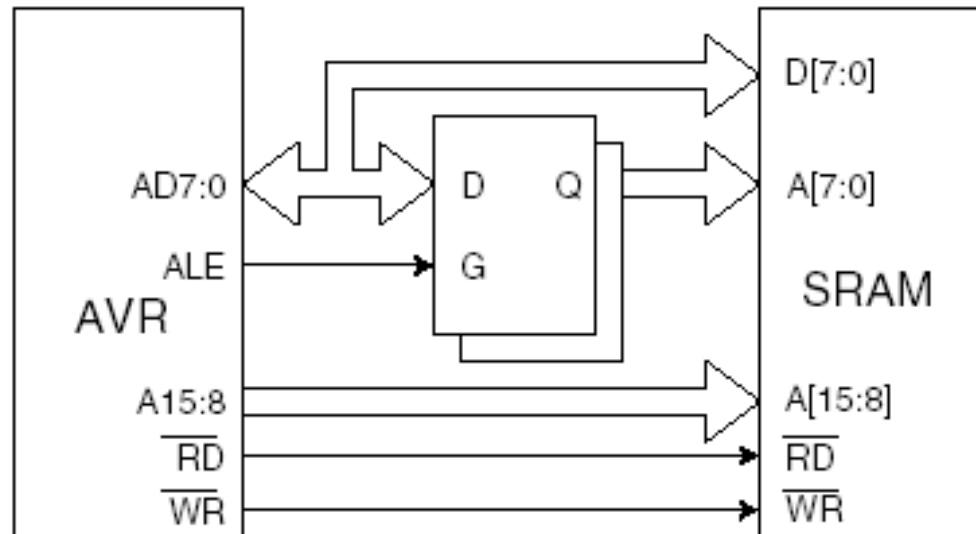
I/O Registers (cont'd)

\$1F (\$3F)	EEARH	–	–	–	–	EEPROM Address Register High			
\$1E (\$3E)	EEARL	EEPROM Address Register Low Byte							
\$1D (\$3D)	EEDR	EEPROM Data Register							
\$1C (\$3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE
\$1B (\$3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0
\$1A (\$3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0
\$19 (\$39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0
\$18 (\$38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
\$17 (\$37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
\$16 (\$36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
\$15 (\$35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
\$14 (\$34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0
\$13 (\$33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
\$12 (\$32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
\$11 (\$31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0
\$10 (\$30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
\$0F (\$2F)	SPDR	SPI Data Register							
\$0E (\$2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X
\$0D (\$2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
\$0C (\$2C)	UDR0	USART0 I/O Data Register							
\$0B (\$2B)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
\$0A (\$2A)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
\$09 (\$29)	UBRR0L	USART0 Baud Rate Register Low							
\$08 (\$28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0
\$07 (\$27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
\$06 (\$26)	ADCSRA	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0
\$05 (\$25)	ADCH	ADC Data Register High Byte							
\$04 (\$24)	ADCL	ADC Data Register Low byte							
\$03 (\$23)	PORTE	PORTE7	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0
\$02 (\$22)	DDRE	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0
\$01 (\$21)	PINE	PINE7	PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0
\$00 (\$20)	PINF	PINF7	PINF6	PINF5	PINF4	PINF3	PINF2	PINF1	PINF0

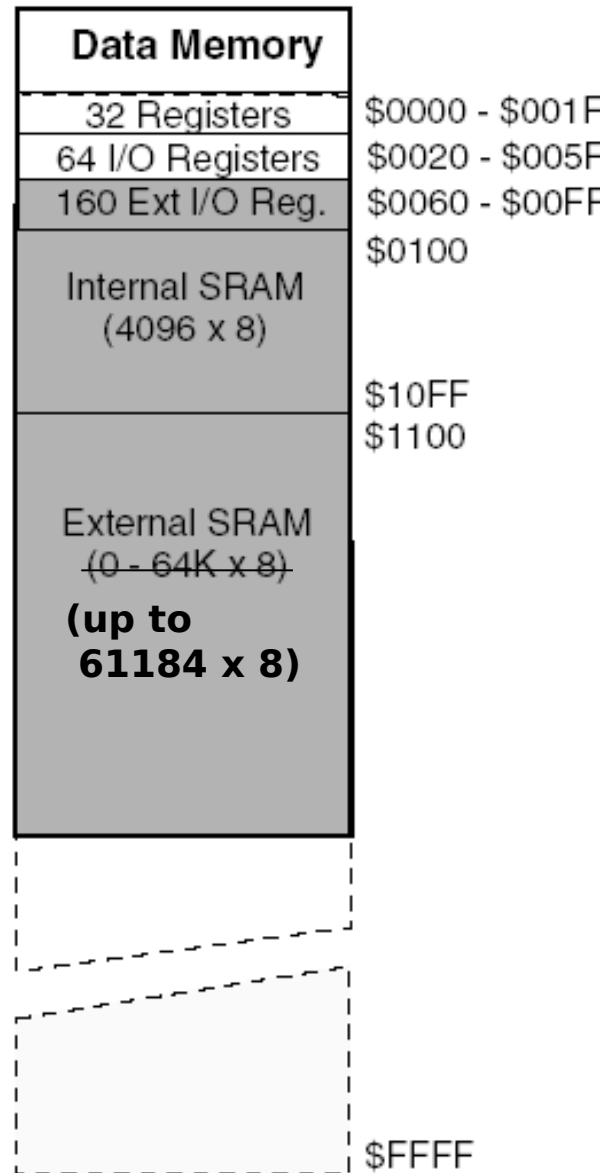
External Memory Interface

- When the External Memory(XMEM) is enabled, address space outside the internal SRAM becomes available using the dedicated External Memory pins (**Port A: AD7:0: Port C: A15:8: Port G: A1 F. RD/- WR/**).

Figure 12. External SRAM Connected to the AVR

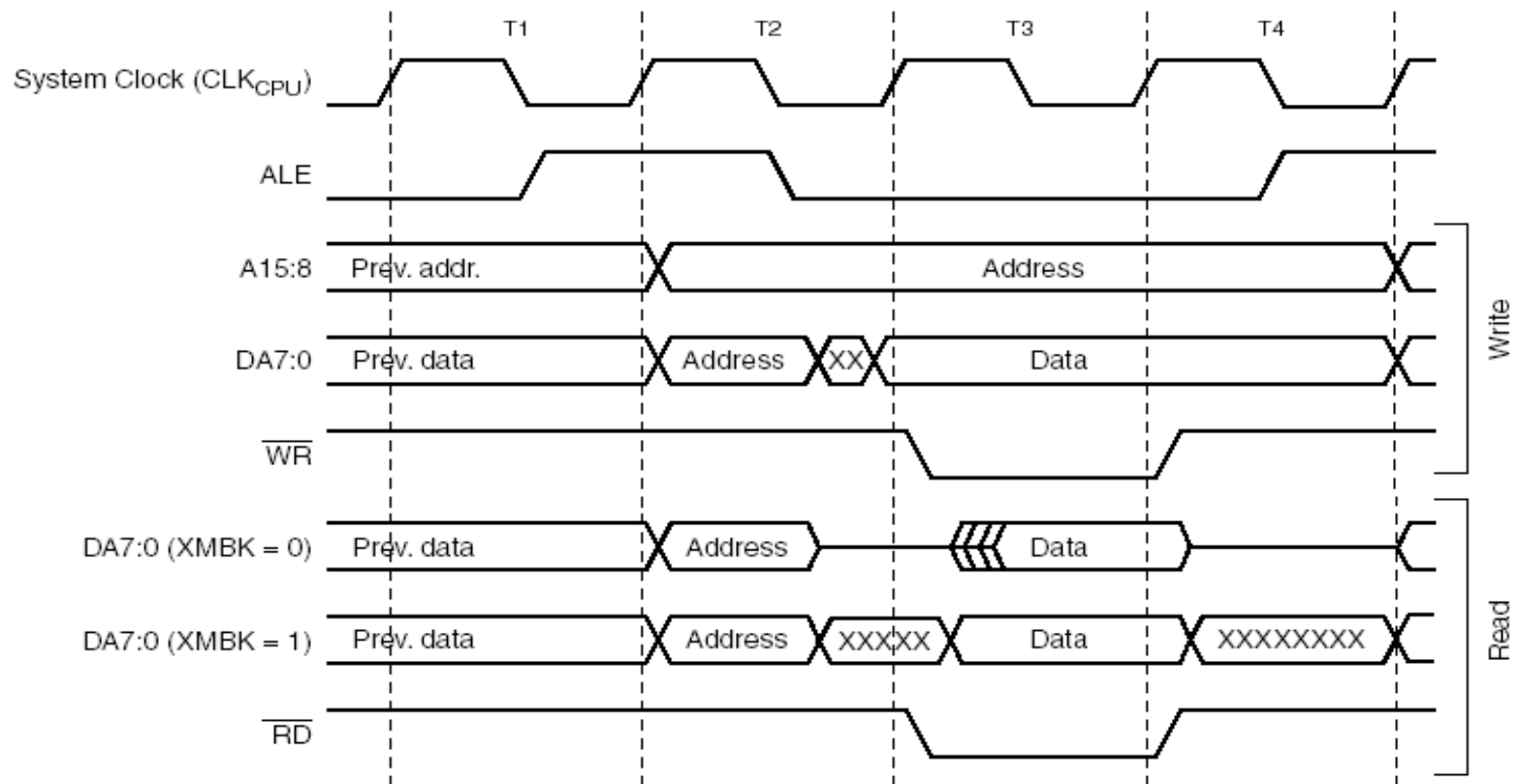


Data Memory Map



Timing diagram

Figure 13. External Data Memory Cycles without Wait-state (SRWn1=0 and SRWn0=0)



Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	$Rd \leftarrow Rd + Rr$	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	$Rd \leftarrow Rd + Rr + C$	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	$Rdh:Rdl \leftarrow Rdh:Rdl + K$	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	$Rd \leftarrow Rd - Rr$	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	$Rd \leftarrow Rd - K$	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	$Rd \leftarrow Rd - Rr - C$	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	$Rd \leftarrow Rd - K - C$	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	$Rdh:Rdl \leftarrow Rdh:Rdl - K$	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	$Rd \leftarrow Rd \bullet Rr$	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	$Rd \leftarrow Rd \bullet K$	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	$Rd \leftarrow Rd \vee Rr$	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	$Rd \leftarrow Rd \vee K$	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	$Rd \leftarrow Rd \oplus Rr$	Z,N,V	1
COM	Rd	One's Complement	$Rd \leftarrow \$FF - Rd$	Z,C,N,V	1
NEG	Rd	Two's Complement	$Rd \leftarrow \$00 - Rd$	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	$Rd \leftarrow Rd \vee K$	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	$Rd \leftarrow Rd \bullet (\$FF - K)$	Z,N,V	1
INC	Rd	Increment	$Rd \leftarrow Rd + 1$	Z,N,V	1
DEC	Rd	Decrement	$Rd \leftarrow Rd - 1$	Z,N,V	1
TST	Rd	Test for Zero or Minus	$Rd \leftarrow Rd \bullet Rd$	Z,N,V	1
CLR	Rd	Clear Register	$Rd \leftarrow Rd \oplus Rd$	Z,N,V	1
SER	Rd	Set Register	$Rd \leftarrow \$FF$	None	1
MUL	Rd, Rr	Multiply Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULS	Rd, Rr	Multiply Signed	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$	Z,C	2

Instruction Set Summary (Continued)

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

Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC \leftarrow PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC \leftarrow PC + k + 1	None	1 / 2
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd \leftarrow Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd \leftarrow Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd \leftarrow K	None	1
LD	Rd, X	Load Indirect	Rd \leftarrow (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd \leftarrow (X), X \leftarrow X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X \leftarrow X - 1, Rd \leftarrow (X)	None	2
LD	Rd, Y	Load Indirect	Rd \leftarrow (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd \leftarrow (Y), Y \leftarrow Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y \leftarrow Y - 1, Rd \leftarrow (Y)	None	2
LDD	Rd, Y+q	Load Indirect with Displacement	Rd \leftarrow (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd \leftarrow (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd \leftarrow (Z), Z \leftarrow Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z \leftarrow Z - 1, Rd \leftarrow (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd \leftarrow (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd \leftarrow (k)	None	2
ST	X, Rr	Store Indirect	(X) \leftarrow Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) \leftarrow Rr, X \leftarrow X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X \leftarrow X - 1, (X) \leftarrow Rr	None	2
ST	Y, Rr	Store Indirect	(Y) \leftarrow Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) \leftarrow Rr, Y \leftarrow Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y \leftarrow Y - 1, (Y) \leftarrow Rr	None	2
STD	Y+q, Rr	Store Indirect with Displacement	(Y + q) \leftarrow Rr	None	2
ST	Z, Rr	Store Indirect	(Z) \leftarrow Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) \leftarrow Rr, Z \leftarrow Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z \leftarrow Z - 1, (Z) \leftarrow Rr	None	2
STD	Z+q, Rr	Store Indirect with Displacement	(Z + q) \leftarrow Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) \leftarrow Rr	None	2
LPM		Load Program Memory	R0 \leftarrow (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd \leftarrow (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd \leftarrow (Z), Z \leftarrow Z+1	None	3
ELPM		Extended Load Program Memory	R0 \leftarrow (RAMPZ:Z)	None	3
ELPM	Rd, Z	Extended Load Program Memory	Rd \leftarrow (RAMPZ:Z)	None	3
ELPM	Rd, Z+	Extended Load Program Memory and Post-Inc	Rd \leftarrow (RAMPZ:Z), RAMPZ:Z \leftarrow RAMPZ:Z+1	None	3
SPM		Store Program Memory	(Z) \leftarrow R1:R0	None	-
IN	Rd, P	In Port	Rd \leftarrow P	None	1
OUT	P, Rr	Out Port	P \leftarrow Rr	None	1
PUSH	Rr	Push Register on Stack	STACK \leftarrow Rr	None	2
POP	Rd	Pop Register from Stack	Rd \leftarrow STACK	None	2

Instruction Set Summary (Continued)

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