Timer 1

Mod	Frecvență	Duty
normal	$f_{OVF} = \frac{f_{cpu}}{PS \cdot (1 + MAX)}$	$D_{OCxy} = 50\%$
FPWM	$f_{OCR} = \frac{f_{cpu}}{PS \cdot (1 + TOP)}$	$D_{OCxy} = \frac{1 + OCRxy}{1 + TOP}$ (non-inverting)
PWM/PC PWM/PFC	$f_{OCR} = \frac{f_{cpu}}{2 \cdot PS \cdot (1 + TOP)}$	$D_{OCxy} = \frac{1 + OCRxy}{1 + TOP}$ (non-inverting)

Comune Timere

Câmp	Descriere
COM	Controlează outputul pe canalul PWM
WGM	Modul de lucru al timerului
CS	Prescalerul timerului
OCIExA	Întrerupere de match pe OCRxA
OCIExB	Întrerupere de match pe OCRxB
TOIEx	Întrerupere de overflow

TCNTx	Registru contor (16 biți pentru timer 1)
OCRxA OCRxB ICRx	Registru prag (16 biți pentru timer 1)

TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0			WGM11	WGM10
TCCR1B				WGM13	WGM12	CS12	CS11	CS10
TIMSK1						OCIE1B	OCIE1A	TOIE1

CS12..0	PS
000	stop
001	1
010	8
011	64
100	256
101	1024

COM1A1..0 COM1B1..0	Normal mode	FastPWM	PWM/PC
00	-	-	-
01	Toggle OC1x	Mod 14/15 – toggle OC1A	Mod 9/11 – toggle OC1A
10	Clear on comp	Clear on comp Set on bottom	Clear when ↑ Set when ↓
11	Set on comp	Set on comp Clear on bottom	Clear when ↓ Set when ↑

WGM13..0	type	TOP	OVF
0000	normal	0xFFFF	MAX
0001	PWM PC	0x00FF	BOTTOM
0010	PWM PC	0x01FF	BOTTOM
0011	PWM PC	0x03FF	BOTTOM
0100	CTC	OCR1A	MAX
0101	FPWM	0x00FF	TOP
0110	FPWM	0x01FF	TOP
0111	FPWM	0x03FF	TOP
1000	PWM PFC	ICR1	BOTTOM
1001	PWM PFC	OCR1A	BOTTOM
1010	PWM PC	ICR1	BOTTOM
1011	PWM PC	OCR1A	BOTTOM
1100	CTC	ICR1	MAX
1101	-	-	-
1110	FPWM	ICR1	TOP
1111	FPWM	OCR1A	TOP

Timer 0

TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0			WGM01	WGM00
TCCR0B					WGM02	CS02	CS01	CS00
TIMSK0						OCIE0B	OCIE0A	TOIE0

CS02..0 la fel ca CS12..0

COM0A1..0 COM0B1..0	Normal mode	FastPWM	PWM/PC
00	-	-	-
01	Toggle OC0x	Mod 7 – toggle OC0A	Mod 5 – toggle OC0A
10	Clear on comp	Clear on comp Set on bottom	Clear when ↑ Set when ↓
11	Set on comp	Set on comp Clear on bottom	Clear when ↓ Set when ↑

WGM02..0	type	TOP	OVF
000	normal	0xFF	MAX
001	PWM PC	0xFF	BOTTOM
010	CTC	OCR0A	MAX
011	FPWM	0xFF	MAX
100	-	-	-
101	PWM/PC	OCR0A	BOTTOM
110	-	-	-
111	FPWM	OCR0A	TOP

Timer 2

TCCR2A	COM2A1	COM2A0	COM2B1	COM2B0			WGM21	WGM20
TCCR2B					WGM22	CS22	CS21	CS20
TIMSK2						OCIE2B	OCIE2A	TOIE2

CS22..0	PS
000	stop
001	1
010	8
011	32
100	64
101	128
110	256
111	1024

WGM22..0 la fel ca WGM02..0
COM2A1..0 la fel ca COM0A1..0
COM2B1..0 la fel ca COM0B1..0

PWM – pulse width modulation
FPWM – Fast PWM
PWM/PC – PWM phase correct
PWM/PFC – PWM phase and frequency correct
TOP – până la cât numără un timer
MAX – maximul până la cât poate număra un timer

ADC

ADMUX	REFS1	REFS0	ADLAR			MUX2	MUX1	MUX0
ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
ADCSRB						ADTS2	ADTS1	ADTS0

ADC Registru pe 16 biți (10 biți aliniați după ADLAR, default dreapta)

Câmp	Descriere
REFS	Referință tensiune
ADLAR	Rezultat ajustat stânga
MUX	Indexul canalului
ADEN	Enable ADC
ADSC	Start conversie
ADATE	Enable auto-triggering
ADIF	1 în timpul conversiei
ADIE	Întrerupere ADC complete
ADPS	Prescaler pentru conversie
ADTS	Selecție eveniment auto-trigger

ADPS	PS
000	2
001	2
010	4
011	8
100	16
101	32
110	64
111	128

ADTS	PS
000	Free-running
001	
010	INT0
011	Timer0 comp
100	Timer0 OVF
101	Timer1 COMPB
110	Timer1 OVF

REFS1..0	parity
00	AREF
01	AVCC
10	1.11V
11	2.56V

USART

UCSR0A	RXC0		UDRE0				U2X0	
UCSR0B	RXCIE0		UDRIE0	RXEN0	TXEN0	UCSZ02		
UCSR0C			UPM01	UPM00	USBS0	UCSZ01	UCSZ00	

UDR0 Registru de transmisie/recepție, citirea se face din alt buffer față de scriere

UBRR0 Registru pe 16 biți pentru configurarea baud rate

Câmp	Descriere
RXC0	1 când a fost primit un caracter
UDRE0	1 când bufferul de TX e gol
U2X0	Dublare baud rate
RXCIE0	Întrerupere receive
UDRIE0	Întrerupere buffer TX gol
RXEN0	Enable recepție
TXEN0	Enable transmisie
UCSZ	Dimensiune Pachete
UPM	Selecție paritate
USBS	Selecție biți de stop

UCSZ02..0	size
000	5-bit
001	6-bit
010	7-bit
011	8-bit

USBS0	stop
0	1
1	2

UPM01..0	parity
00	disabled
01	-
10	even
11	odd

U2X0==0 $BAUD = \frac{f_{CPU}}{16 \cdot (1 + UBRR)}$

U2X0==1 $BAUD = \frac{f_{CPU}}{8 \cdot (1 + UBRR)}$

SPI

SPCR	SPIE	SPE	DORD	MSTRL			SPR1	SPR0
SPSR	SPIF							SPI2X
SPDR	Registru buffer							

SPR1..0	prescaler
00	4
01	16
10	64
11	128

Câmp	Descriere
SPIE	Întrerupere transfer complet
SPE	Enable SPI
DORD	1 dacă transmitem LSB first
MSTR	1 dacă este master
SPR	prescaler
SPIF	1 când un transfer este complet
SPI2X	1 pentru dublarea frecvenței

PCINT

PCICR					PCIE3	PCIE2	PCIE1	PCIE0
PCMSK3	PCINT31	PCINT30	PCINT29	PCINT28	PCINT27	PCINT26	PCINT25	PCINT24
PCMSK2	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16
PCMSK1	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT09	PCINT08
PCMSK0	PCINT07	PCINT06	PCINT05	PCINT04	PCINT03	PCINT02	PCINT01	PCINT00

Pentru Pin Change Interrupt (întrerupere pe orice schimbare de nivel pe un pin), activăm bitul respectiv din PCMSKx și apoi PCIEx corespunzător PCMSK-ului

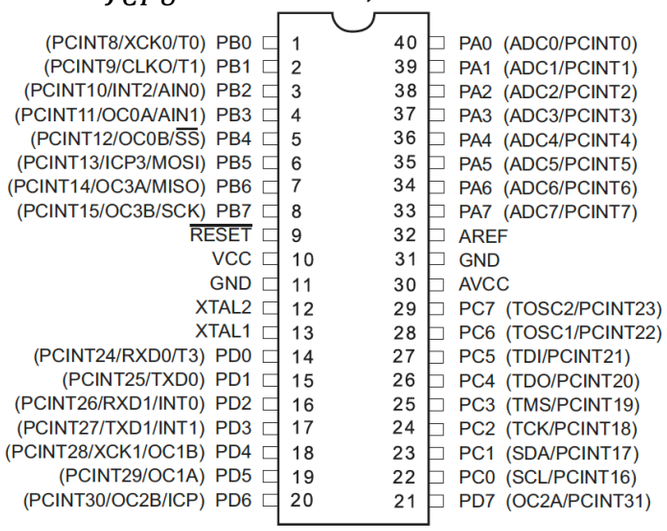
GPIO

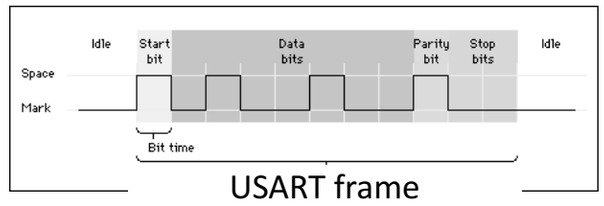
DDRX (A|B|C|D)	1 dacă pinul este output, 0 altfel
PORTX (A|B|C|D)	Pt output reprezintă valoarea, pentru input este activarea rezistențelor de pull-up
PCINTX(A|B|C|D)	Valoarea citită de la fiecare pin din portul respectiv

Proiectarea Cu Microprocesoare

Cheatsheet ATMega324a 2018 rev1

$f_{CPU} = 16MHz, Vcc = 3.3V$





USART frame

API LCD Text

```
void LCD_init(void);                    // Initializare LCD
void LCD_writeInstr(uint8_t instr); // Scrie o instructiune catre LCD.
void LCD_writeData(uint8_t data);    // Scrie date catre LCD.
void LCD_putChar(char c);               // Scrie caracter pe LCD la cursor
void LCD_putCharAt(uint8_t addr, char c); // Scrie caracter la adresa data
void LCD_print(const char* msg); // Afiseaza string la cursor.
void LCD_printAt(uint8_t addr, const char* msg); // Afiseaza string la adresa
```

API LCD Grafic

```
/* Initializeaza Display-ul grafic. */
void ST7735R_Begin();
 /* Deseneaza o linie de la (x0, y0) la (x1, y1),
  * avand culoarea data de parametrii r, g, b. */
void ST7735R_Line(int x0, int y0, int x1, int y1,
                  uint8_t r, uint8_t g, uint8_t b);
 /* Afiseaza un text, incepand de la pozitia (x, y).
  * Textul are culoarea specificata de pereche (r, g, b).
  * Background-ul pe care este afisat textul are culoarea (bgR, bgG, bgB). */
void ST7735R_DrawText(int x, int y, const char *text,
                      uint8_t r, uint8_t g, uint8_t b,
                      uint8_t bgR, uint8_t bgG, uint8_t bgB);
 /* Deseneaza un cerc cu centrul la coordonatele (x, y),
  * de raza 'radius', cu culoarea specificată. */
void ST7735R_Circle(int x, int y, uint8_t radius,
                    uint8_t red, uint8_t green, uint8_t blue);
 /* Deseneaza un cerc cu centrul la coordonatele (x, y) de raza 'radius'
  * și îl umple cu culoarea dată. */
void ST7735R_FilledCircle(int x, int y, uint8_t radius,
                          uint8_t red, uint8_t green, uint8_t blue);
 /* Deseneaza un dreptunghi cu colțul stânga sus dat de (x0, y0) *
  * și colțul dreapta jos dat de (x1, y1). *
  * Dreptunghiul este umplut cu culoarea (r, g, b). */
ST7735R_FillRect(x0, y0, x1, y1, r, g, b);
```

Extras

```
cli(); // dezactiveaza intreruperi
sei(); // activeaza intreruperi
```