AVR Microcontroller

AtMega162

Lê Chí Thông

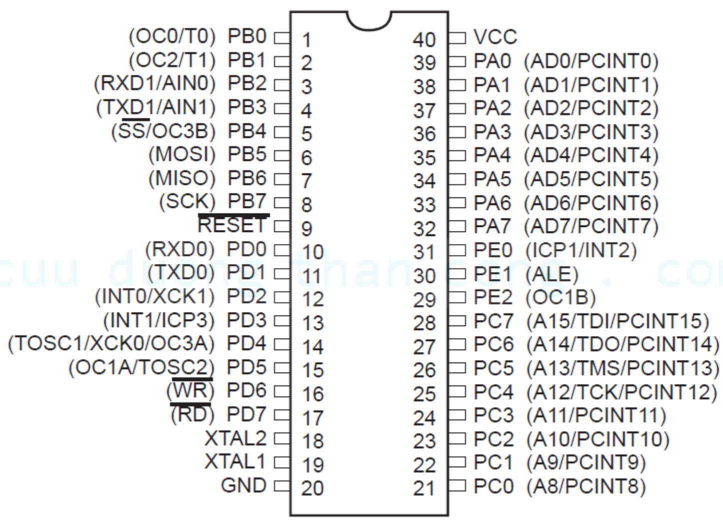
chithong@hcmut.edu.vn

sites.google.com/site/chithong

Ho Chi Minh City University of Technology

Pins

PDIP



Status Register

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
	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	

Status Register

- **Bit 7 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

- **Bit 6 – T: Bit Copy Storage**

The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T bit as source or destination for the operated bit. A bit from a register in the Register File can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

- **Bit 5 – H: Half Carry Flag**

The Half Carry Flag H indicates a half carry in some arithmetic operations. Half Carry is useful in BCD arithmetic. See the “Instruction Set Description” for detailed information.

Status Register

- **Bit 4 – S: Sign Bit, S = N ⊕ V**

The S-bit is always an exclusive or between the Negative Flag N and the Two's Complement Overflow Flag V. See the "Instruction Set Description" for detailed information.

- **Bit 3 – V: Two's Complement Overflow Flag**

The Two's Complement Overflow Flag V supports two's complement arithmetics. See the "Instruction Set Description" for detailed information.

- **Bit 2 – N: Negative Flag**

The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.

- **Bit 1 – Z: Zero Flag**

The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.

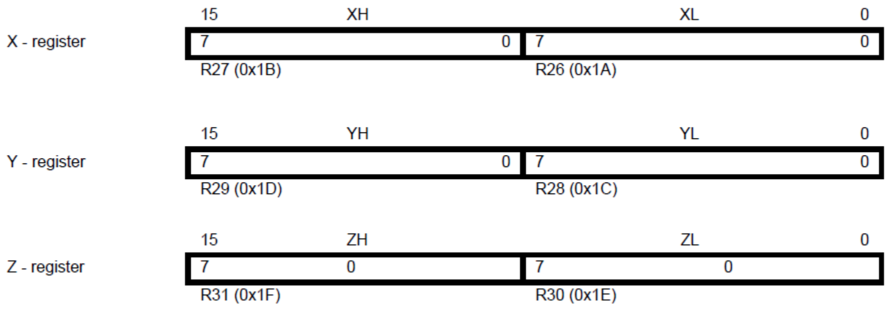
- **Bit 0 – C: Carry Flag**

The Carry Flag C indicates a carry in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.

General Purpose Register

	7 … 0	Addr.	
	R0	0x00	
	R1	0x01	
	R2	0x02	
	…		
	R13	0x0D	
General	R14	0x0E	
Purpose	R15	0x0F	
Working	R16	0x10	
Registers	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

X-register, Y-register, Z-register



Lê Chí Thông 7

Interrupt Vectors

Vector No.	Program Address[2]	Source	Interrupt Definition
1	0x000[1]	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	0x002	INT0	External Interrupt Request 0
3	0x004	INT1	External Interrupt Request 1
4	0x006	INT2	External Interrupt Request 2
5	0x008	PCINT0	Pin Change Interrupt Request 0
6	0x00A	PCINT1	Pin Change Interrupt Request 1
7	0x00C	TIMER3 CAPT	Timer/Counter3 Capture Event
8	0x00E	TIMER3 COMPA	Timer/Counter3 Compare Match A
9	0x010	TIMER3 COMPB	Timer/Counter3 Compare Match B
10	0x012	TIMER3 OVF	Timer/Counter3 Overflow

Lê Chí Thông 8

Interrupt Vectors

11	0x014	TIMER2 COMP	Timer/Counter2 Compare Match
12	0x016	TIMER2 OVF	Timer/Counter2 Overflow
13	0x018	TIMER1 CAPT	Timer/Counter1 Capture Event
14	0x01A	TIMER1 COMPA	Timer/Counter1 Compare Match A
15	0x01C	TIMER1 COMPB	Timer/Counter1 Compare Match B
16	0x01E	TIMER1 OVF	Timer/Counter1 Overflow
17	0x020	TIMER0 COMP	Timer/Counter0 Compare Match
18	0x022	TIMER0 OVF	Timer/Counter0 Overflow
19	0x024	SPI, STC	Serial Transfer Complete
20	0x026	USART0, RXC	USART0, Rx Complete

Lê Chí Thông 9

Interrupt Vectors

21	0x028	USART1, RXC	USART1, Rx Complete
22	0x02A	USART0, UDRE	USART0 Data Register Empty
23	0x02C	USART1, UDRE	USART1 Data Register Empty
24	0x02E	USART0, TXC	USART0, Tx Complete
25	0x030	USART1, TXC	USART1, Tx Complete
26	0x032	EE_RDY	EEPROM Ready
27	0x034	ANA_COMP	Analog Comparator
28	0x036	SPM_RDY	Store Program Memory Ready

Lê Chí Thông 10

I/O Port

Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. In addition, the Pull-up Disable – PUD bit in SFIOR disables the pull-up function for all pins in all ports when set.

Port Pin Configurations

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn.

The DDxn bit in the DDRx Register selects the direction of this pin. If DDxn is written logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.

If PORTxn is written logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logic zero or the pin has to be configured as an output pin. The port pins are tri-stated when a reset condition becomes active, even if no clocks are running.

If PORTxn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

Port Pin Configurations

When switching between tri-state ({DDxn, PORTxn} = 0b00) and output high ({DDxn, PORTxn} = 0b11), an intermediate state with either pull-up enabled ({DDxn, PORTxn} = 0b01) or output low ({DDxn, PORTxn} = 0b10) must occur. Normally, the pull-up enabled state is fully acceptable, as a high-impedant environment will not notice the difference between a strong high driver and a pull-up. If this is not the case, the PUD bit in the SFIOR Register can be set to disable all pull-ups in all ports.

Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ({DDxn, PORTxn} = 0b00) or the output high state ({DDxn, PORTxn} = 0b11) as an intermediate step.

Port Pin Configurations

7	6	5	4	3	2	1	0	
PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

7	6	5	4	3	2	1	0	
DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

7	6	5	4	3	2	1	0	
PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
R	R	R	R	R	R	R	R	
N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Port Pin Configurations

DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

C Code Example

```
unsigned char i;
  ...
  /* Define pull-ups and set outputs high */
  /* Define directions for port pins */
  PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
  DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
  /* Insert nop for synchronization*/
  _NOP();
  /* Read port pins */
  i = PINB;
  ...
```

Alternate Functions of Port A

Port Pin	Alternate Function
PA7	AD7 (External memory interface address and data bit 7) PCINT7 (Pin Change INTerrupt 7)
PA6	AD6 (External memory interface address and data bit 6) PCINT6 (Pin Change INTerrupt 6)
PA5	AD5 (External memory interface address and data bit 5) PCINT5 (Pin Change INTerrupt 5)
PA4	AD4 (External memory interface address and data bit 4) PCINT4 (Pin Change INTerrupt 4)
PA3	AD3 (External memory interface address and data bit 3) PCINT3 (Pin Change INTerrupt 3)
PA2	AD2 (External memory interface address and data bit 2) PCINT2 (Pin Change INTerrupt 2)
PA1	AD1 (External memory interface address and data bit 1) PCINT1 (Pin Change INTerrupt 1)
PA0	AD0 (External memory interface address and data bit 0) PCINT0 (Pin Change INTerrupt 0)

Alternate Functions of Port B

Port Pin	Alternate Functions
PB7	SCK (SPI Bus Serial Clock)
PB6	MISO (SPI Bus Master Input/Slave Output)
PB5	MOSI (SPI Bus Master Output/Slave Input)
PB4	$\overline{SS}$ (SPI Slave Select Input) OC3B (Timer/Counter3 Output Compare Match Output)
PB3	AIN1 (Analog Comparator Negative Input) TXD1 (USART1 Output Pin)
PB2	AIN0 (Analog Comparator Positive Input) RXD1 (USART1 Input Pin)
PB1	T1 (Timer/Counter1 External Counter Input) OC2 (Timer/Counter2 Output Compare Match Output)
PB0	T0 (Timer/Counter0 External Counter Input) OC0 (Timer/Counter0 Output Compare Match Output) $clk_{I/O}$ (Divided System Clock)

Alternate Functions of Port C

Port Pin	Alternate Function
PC7	A15 (External memory interface address bit 15) TDI (JTAG Test Data Input) PCINT15 (Pin Change INTerrupt 15)
PC6	A14 (External memory interface address bit 14) TDO (JTAG Test Data Output) PCINT14 (Pin Change INTerrupt 14)
PC5	A13 (External memory interface address bit 13) TMS (JTAG Test Mode Select) PCINT13 (Pin Change INTerrupt 13)
PC4	A12 (External memory interface address bit 12) TCK (JTAG Test Clock) PCINT12 (Pin Change INTerrupt 12)
PC3	A11 (External memory interface address bit 11) PCINT11 (Pin Change INTerrupt 11)
PC2	A10 (External memory interface address bit 10) PCINT10 (Pin Change INTerrupt 10)
PC1	A9 (External memory interface address bit 9) PCINT9 (Pin Change INTerrupt 9)
PC0	A8 (External memory interface address bit 8) PCINT8 (Pin Change INTerrupt 8)

Alternate Functions of Port D

Port Pin	Alternate Function
PD7	$\overline{\text{RD}}$ (Read strobe to external memory)
PD6	$\overline{\text{WR}}$ (Write strobe to external memory)
PD5	TOSC2 (Timer Oscillator Pin 2) OC1A (Timer/Counter1 Output Compare A Match Output)
PD4	TOSC1 (Timer Oscillator Pin 1) XCK0 (USART0 External Clock Input/Output) OC3A (Timer/Counter3 Output Compare A Match Output)
PD3	INT1 (External Interrupt 1 Input) ICP3 (Timer/Counter3 Input Capture Pin)
PD2	INT0 (External Interrupt 0 Input) XCK1 (USART1 External Clock Input/Output)
PD1	TXD0 (USART0 Output Pin)
PD0	RXD0 (USART0 Input Pin)

Alternate Functions of Port E

Port Pin	Alternate Function
PE2	OC1B (Timer/Counter1 Output CompareB Match Output)
PE1	ALE (Address Latch Enable to external memory)
PE0	ICP1 (Timer/Counter1 Input Capture Pin) INT2 (External Interrupt 2 Input)

C Code Example

```
void main(void)
{
   // Port initialization
   PORTA=0x00; // Pull up registers
   DDRA=0xff;  // output
   PORTB=0x00; // Pull up registers
   DDRB=0xff;  // output
   PORTA=0x6f; // code of number 9
   PORTB=0xc0; // 0b11000000
   // Loop
   while(1)
   {
   }
}
```

8-bit Timer/Counter 0 with PWM

Timer/Counter0 is a general purpose, single channel, 8-bit Timer/Counter module. The main features are:

- **Single Channel Counter**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch-free, Phase Correct Pulse Width Modulator (PWM)**
- **Frequency Generator**
- **External Event Counter**
- **10-bit Clock Prescaler**
- **Overflow and Compare Match Interrupt Sources (TOV0 and OCF0)**

Timer0 Registers

The Timer/Counter (TCNT0) and Output Compare Register (OCR0) are 8-bit registers. Interrupt request (abbreviated to Int.Req. in the figure) signals are all visible in the Timer Interrupt Flag Register (TIFR). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK). TIFR and TIMSK are not shown in the figure since these registers are shared by other timer units.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T0 pin. The Clock Select logic block controls which clock source and edge the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the clock select logic is referred to as the timer clock (clk_T0).

- Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the Clock Select (CS02:0) bits

Timer0 Registers

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	0	0	0	0	0	0	0	0	

Timer/Counter Control Register

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Lê Chí Thông 25

Timer0 Registers

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	0	0	0	0	0	0	0	0	

Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	0	0	0	0	0	0	0	0	

Output Compare Register

Bit	7	6	5	4	3	2	1	0	
	TOIE1	OCIE1A	OCIE1B	OCIE2	TICIE1	TOIE2	TOIE0	OCIE0	TIMSK
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	0	0	0	0	0	0	0	0	

Timer/Counter Interrupt Mask Register

Lê Chí Thông 26

16-bit Timer/Counter1 (Timer/Counter3)

The 16-bit Timer/Counter unit allows accurate program execution timing (event management), wave generation, and signal timing measurement. The main features are:

- **True 16-bit Design (i.e., allows 16-bit PWM)**
- **Two Independent Output Compare Units**
- **Double Buffered Output Compare Registers**
- **One Input Capture Unit**
- **Input Capture Noise Canceler**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch-free, Phase Correct Pulse Width Modulator (PWM)**
- **Variable PWM Period**
- **Frequency Generator**
- **External Event Counter**
- **Eight Independent Interrupt Sources (TOV1, OCF1A, OCF1B, ICF1, TOV3, OCF3A, OCF3B, and ICF3)**

8-bit Timer/Counter2 with PWM and Asynchronous operation

Timer/Counter2 is a general purpose, single channel, 8-bit Timer/Counter module. The main features are:

- **Single Channel Counter**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch-free, Phase Correct Pulse Width Modulator (PWM)**
- **Frequency Generator**
- **10-bit Clock Prescaler**
- **Overflow and Compare Match Interrupt Sources (TOV2 and OCF2)**
- **Allows Clocking from External 32 kHz Watch Crystal Independent of the I/O Clock**

The Timer/Counter (TCNT2) and Output Compare Register (OCR2) are 8-bit registers. Interrupt request (shorten as Int.Req.) signals are all visible in the Timer Interrupt Flag Register (TIFR). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK). TIFR and TIMSK are not shown in the figure since these registers are shared by other timer units.

C Code Example

```
unsigned int i;
...
/* Set TCNTn to 0x01FF */
TCNTn = 0x1FF;
/* Read TCNTn into i */
i = TCNTn;
...
```

Lê Chí Thông 29

C Code Example

```
unsigned int i;
...
/* Set TCNTn to 0x01FF */
TCNTn = 0x1FF;
/* Read TCNTn into i */
i = TCNTn;
...
```


Lê Chí Thông 30

Example 1: Delay 200 us

```
//============================================================
//Define Function
//============================================================
//                    Required Delay
// Timer Count = --------------------------  - 1
//                    Clock Time Period
// clock interrupt 11.0592/8 = 1.3824 Mhz
// 1 clock = 0.72337 us
// t = 200us => Timer count =  200us/0.72337us -1 =  276-1 = 275
// TCNT1= 65535-276= 65529 ; or -275

#define  RELOAD_TIMER1 -275
```


Example 1: Delay 200 us

```
//============================================================
//Define Function
//============================================================
//                    Required Delay
// Timer Count = --------------------------  - 1
//                    Clock Time Period
// clock interrupt 11.0592/8 = 1.3824 Mhz
// 1 clock = 0.72337 us
// t = 200us => Timer count =  200us/0.72337us -1 =  276-1 = 275
// TCNT1= 65535-276= 65529 ; or -275

#define  RELOAD_TIMER1 -275
```

Example 1: Delay 200 us

```
void Init_Timer1(void)
{
   // Clock prescaler is set to divided by 8
   TCCR1B = (0<<CS12)|(1<<CS11)|(0<<CS10);

    // initialize counter
   TCNT1 = 0 ; //16 bit, TCNT1H = 0; TCNT1L = 0; 8bit

   TCNT1 = RELOAD_TIMER1;    //Loading Timer/Counter0
   TIMSK = (1<<TOIE1);       //Enable  Timer Overflow
   #asm("sei")               //Enable global interrupt
}
```


Example 1: Delay 200 us

```
//==============================================================
// Timer1 overflow interrupt service routine
// called whenever TCNT0 overflows
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
   LED = 0; // LED TURN ON
   TIMSK &= ~(1<<TOIE1);
   //TCNT1 = RELOAD_TIMER1;   //Loading Timer/Counter0
    }
```

Example 2: 5 KHz Square Wave

```
void main(void)
{
   // Port B initialization
   PORTB=0xff;
   DDRB=0xff; // OUTPUT

   Init_Timer1();

   // Loop
   while(1)
   {
   }
}
```


Example 2: 5 KHz Square Wave

```
#define  RELOAD_TIMER1 -137

void Init_Timer1(void)
{
   // Clock prescaler is set to divided by 8
   TCCR1B = (0<<CS12)|(1<<CS11)|(0<<CS10);

    // initialize counter
   TCNT1 = 0 ; //16 bit, TCNT1H = 0; TCNT1L = 0; 8bit

   TCNT1 = RELOAD_TIMER1;    //Loading Timer/Counter0
   TIMSK = (1<<TOIE1);       //Enable  Timer OverFlow
   #asm("sei")               //Enable gobal interrupt
}
```

Example 2: 5 KHz Square Wave

```
//==============================================================
// Timer1 overflow interrupt service routine
// called whenever TCNT0 overflows
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
   PORTB ^= 0x01; // P1.0
   TCNT1 = RELOAD_TIMER1;   //Loading Timer
     }
```


USART

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly flexible serial communication device. The main features are:

- **Full Duplex Operation (Independent Serial Receive and Transmit Registers)**
- **Asynchronous or Synchronous Operation**
- **Master or Slave Clocked Synchronous Operation**
- **High Resolution Baud Rate Generator**
- **Supports Serial Frames with 5, 6, 7, 8, or 9 Data Bits and 1 or 2 Stop Bits**
- **Odd or Even Parity Generation and Parity Check Supported by Hardware**
- **Data OverRun Detection**
- **Framing Error Detection**
- **Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter**
- **Three Separate Interrupts on TX Complete, TX Data Register Empty and RX Complete**
- **Multi-processor Communication Mode**
- **Double Speed Asynchronous Communication Mode**

The ATmega162 has two USARTs, USART0 and USART1.

USART

The USART is fully compatible with the AVR UART regarding:

- Bit locations inside all USART Registers
- Baud Rate Generation
- Transmitter Operation
- Transmit Buffer Functionality
- Receiver Operation

Clock Generation

The Clock Generation logic generates the base clock for the Transmitter and Receiver. The USART supports four modes of clock operation: Normal asynchronous, Double Speed asynchronous, Master synchronous and Slave synchronous mode. The UMSEL bit in USART Control and Status Register C (UCSRC) selects between asynchronous and synchronous operation. Double Speed (asynchronous mode only) is controlled by the U2X found in the UCSRA Register. When using synchronous mode (UMSEL = 1), the Data Direction Register for the XCK pin (DDR_XCK) controls whether the clock source is internal (Master mode) or external (Slave mode). The XCK pin is only active when using synchronous mode.

Baud Rate

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal Mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(\mathbf{UBRR}+1)}$	$\mathbf{UBRR} = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed Mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(\mathbf{UBRR}+1)}$	$\mathbf{UBRR} = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master Mode	$BAUD = \frac{f_{OSC}}{2(\mathbf{UBRR}+1)}$	$\mathbf{UBRR} = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps).

BAUD Baud rate (in bits per second, bps)

f_{OSC} System Oscillator clock frequency

UBRR Contents of the UBRRH and UBRRL Registers, (0 - 4095)

USART Initialization

The USART has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting frame format and enabling the Transmitter or the Receiver depending on the usage. For interrupt driven USART operation, the Global Interrupt Flag should be cleared (and interrupts globally disabled) when doing the initialization.

Before doing a re-initialization with changed baud rate or frame format, be sure that there are no ongoing transmissions during the period the registers are changed. The TXC Flag can be used to check that the Transmitter has completed all transfers, and the RXC Flag can be used to check that there are no unread data in the receive buffer. Note that the TXC Flag must be cleared before each transmission (before UDR is written) if it is used for this purpose.

USART Initialization

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume asynchronous operation using polling (no interrupts enabled) and a fixed frame format. The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 Registers. When the function writes to the UCSRC Register, the URSEL bit (MSB) must be set due to the sharing of I/O location by UBRRH and UCSRC.

The frame format used by the USART is set by the UCSZ2:0, UPM1:0 and USBS bits in UCSRB and UCSRC. The Receiver and Transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

The USART Character SiZe (UCSZ2:0) bits select the number of data bits in the frame. The USART Parity mode (UPM1:0) bits enable and set the type of parity bit. The selection between one or two stop bits is done by the USART Stop Bit Select (USBS) bit. The receiver ignores the second stop bit. An FE (Frame Error) will therefore only be detected in the cases where the first stop bit is zero.

USART Initialization

The frame format used by the USART is set by the UCSZ2:0, UPM1:0 and USBS bits in UCSRB and UCSRC. The Receiver and Transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

The USART Character SiZe (UCSZ2:0) bits select the number of data bits in the frame. The USART Parity mode (UPM1:0) bits enable and set the type of parity bit. The selection between one or two stop bits is done by the USART Stop Bit Select (USBS) bit. The receiver ignores the second stop bit. An FE (Frame Error) will therefore only be detected in the cases where the first stop bit is zero.

UDR Register – Data Register

- Transmit: write data to UDR
- Receive: read UDR

UCSR0A Register – Status Data

UCSR0A Bit #	Name	Description
bit 7	**RXC0**	**USART Receive Complete**. Set when data is available and the data register has not be read yet.
bit 6	**TXC0**	**USART Transmit Complete.** Set when all data has transmitted.
bit 5	UDRE0	USART Data Register Empty. Set when the UDR0 register is empty and new data can be transmitted.
bit 4	**FE0**	**Frame Error**. Set when next byte in the UDR0 register has a framing error.
bit 3	**DOR0**	**Data OverRun**. Set when the UDR0 was not read before the next frame arrived.
bit 2	**UPE0**	**USART Parity Error.** Set when next frame in the UDR0 has a parity error.
bit 1	U2X0	USART Double Transmission Speed. When set decreases the bit time by half doubling the speed.
bit 0	MPCM0	Multi-processor Communication Mode. When set incoming data is ignored if no addressing information is provided

UCSR0B Register - Settings

UCSR0B Bit #	Name	Description
bit 7	**RXCIE0**	**RX Complete Interrupt Enable.** Set to allow receive complete interrupts.
bit 6	**TXCIE0**	**TX Complete Interrupt Enable**. Set to allow transmission complete interrupts.
bit 5	UDRIE0	USART Data Register Empty Interrupt Enable. Set to allow data register empty interrupts.
bit 4	**RXEN0**	**Receiver Enable**. Set to enable receiver.
bit 3	**TXEN0**	**Transmitter enable**. Set to enable transmitter.
bit 2	**UCSZ02**	USART Character Size 0. Used together with UCSZ01 and UCSZ00 to set **data frame size.** Available sizes are 5-bit (000), 6-bit (001), 7-bit (010), 8-bit (011) and 9-bit (111).
bit 1	RXB80	Receive Data Bit 8. When using 8 bit transmission the 8th bit received.
bit 0	TXB80	Transmit Data Bit 8. When using 8 bit transmission the 8th bit to be submitted.

Lê Chí Thông 47

UCSR0C Register - Settings

UCSR0C Bit #	Name	Description
bit 7	URSEL0	Select between UCSRC or UBRRH. Set URSEL when writing the UCSRC
bit 6	UMSEL0	0: asynchronous ;1: synchronous
bit 5 bit 4	UPM01 UPM00	USART Parity Mode 1 and 0. UPM01 and UPM00 select the parity. Available modes are none (00), even (10) and odd (11).
bit 3	USBS0	0: 1 stop bit; 1: 2 stop bits.
bit 2 bit 1	**UCSZ01** **UCSZ00**	USART Character Size 1 and 0. Used together with with UCSZ20 to set **data frame size.** Available sizes are 5-bit (000), 6-bit (001), 7-bit (010), 8-bit (011) and 9-bit (111).
bit 0	UCPOL0	USART Clock Polarity. Set to transmit on falling edge and sample on rising edge. Unset to transmit on rising edge and sample on falling edge.

Lê Chí Thông 48

USART Initialization

- **Character size:**

UCSZ02	UCSZ01	UCSZ00	Size
0	0	0	5 bit
0	0	1	6 bit
0	1	0	7 bit
0	1	1	8 bit
1	1	1	9 bit

- **Receiver Enable: Set RXEN0**
- **Transmitter Enable: Set TXEN0**

USART Initialization

- **UBRR = f_{OSC}/16/Baud – 1**

Ex: f_{OSC} = 11.0592 MHz, Baud rate = 2400

→UBRR = 11059.2/16/2400 – 1 = 287 = 11FH

C Code Example

```
UCSR0C = 0
  | (0<<UMSEL01) | (0<<UMSEL00) // Asynchronous USART
  | (0<<UPM01) | (0<<UPM00) // Parity Disabled
  | (0<<USBS0) // 1 stop bit
  | (1<<UCSZ01)
  | (1<<UCSZ00) // 8-bit character size
  | (0<<UCPOL) // Rising TX, falling RX ;
```


Example 1: Sending Data

```
#define FRAMING_ERROR (1<<FE0)
#define PARITY_ERROR (1<<UPE0)
#define DATA_OVERRUN (1<<DOR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)
```

Example 1: Sending Data

```
void main(void)
{
    // Port D initialization
    PORTD=0x00;
    DDRD=0x00;

    // Init UART
    Init_UART();

    // Loop
    while(1)
    {
        UART_Printf("Micro-processing Systems Lab\r");
        delay_ms(10);
    }
}
```


Example 1: Sending Data

// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: Off
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud Rate: 2400

UCSR0A=0x00;

RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM

UCSR0B=0x08;

RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8

UCSR0C=0x86;

URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL

UBRR0H=0x01;
UBRR0L=0x1F;

Example 1: Sending Data

```
char UART_getChar(void)
{
    char status,data;
    while (1)
    {
        while (((status=UCSR0A) & RX_COMPLETE)==0);
        data=UDR;
        if ((status & (FRAMING_ERROR | PARITY_ERROR
  | DATA_OVERRUN))==0)
        return data;
    };
}
```

Lê Chí Thông 55

Example 1: Sending Data

```
void UART_putChar(char c)
{
    while ((UCSR0A & DATA_REGISTER_EMPTY)==0);
    UDR=c;
}
void UART_Printf(char *s)
{
    // loop through entire string
    while (*s)
    {
        UART_putChar(*s);
        delay_ms(20);
        s++;
    }
}
```


Lê Chí Thông 56

USART Initialization

```
#define FOSC 1843200// Clock Speed
#define BAUD 9600
#define MYUBRR FOSC/16/BAUD-1
void main( void )
{
  ...
  USART_Init ( MYUBRR );
  ...
}
void USART_Init( unsigned int ubrr )
{
  /* Set baud rate */
  UBRRH = (unsigned char)(ubrr>>8);
  UBRRL = (unsigned char)ubrr;
  /* Enable receiver and transmitter */
  UCSRB = (1<<RXEN)|(1<<TXEN);
  /* Set frame format: 8data, 2stop bit */
  UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}
```


USART Transmitter – Data Transmission

The USART Transmitter is enabled by setting the *Transmit Enable* (TXEN) bit in the UCSRB Register. When the Transmitter is enabled, the normal port operation of the TxD pin is overridden by the USART and given the function as the transmitter's serial output. The baud rate, mode of operation and frame format must be set up once before doing any transmissions. If synchronous operation is used, the clock on the XCK pin will be overridden and used as transmission clock.

Sending Frames with 5 to 8 Data Bit

A data transmission is initiated by loading the transmit buffer with the data to be transmitted. The CPU can load the transmit buffer by writing to the UDR I/O location. The buffered data in the transmit buffer will be moved to the Shift Register when the Shift Register is ready to send a new frame. The Shift Register is loaded with new data if it is in idle state (no ongoing transmission) or immediately after the last stop bit of the previous frame is transmitted. When the Shift Register is loaded with new data, it will transfer one complete frame at the rate given by the Baud Register, U2X bit or by XCK depending on mode of operation.

The following code examples show a simple USART transmit function based on polling of the *Data Register Empty* (UDRE) Flag. When using frames with less than eight bits, the most significant bits written to the UDR are ignored. The USART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in Register R16

Sending Frames with 5 to 8 Data Bit

```
void USART_Transmit( unsigned char data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE)) )
        ;
  /* Put data into buffer, sends the data */
  UDR = data;
}
```

Sending Frames with 9 Data Bit

If 9-bit characters are used (UCSZ = 7), the ninth bit must be written to the TXB8 bit in UCSRB before the low byte of the character is written to UDR. The following code examples show a transmit function that handles 9-bit characters. For the assembly code, the data to be sent is assumed to be stored in Registers R17:R16.

Sending Frames with 9 Data Bit

```
void USART_Transmit( unsigned int data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE)) )
       ;
  /* Copy 9th bit to TXB8 */
  UCSRB &= ~(1<<TXB8);
  if ( data & 0x0100 )
    UCSRB |= (1<<TXB8);
  /* Put data into buffer, sends the data */
  UDR = data;
}
```

Transmitter Flags and Interrupts

The USART Transmitter has two flags that indicate its state: USART Data Register Empty (UDRE) and Transmit Complete (TXC). Both flags can be used for generating interrupts.

The Data Register Empty (UDRE) Flag indicates whether the transmit buffer is ready to receive new data. This bit is set when the transmit buffer is empty, and cleared when the transmit buffer contains data to be transmitted that has not yet been moved into the Shift Register. For compatibility with future devices, always write this bit to zero when writing the UCSRA Register.

When the Data Register Empty Interrupt Enable (UDRIE) bit in UCSRB is written to one, the USART Data Register Empty Interrupt will be executed as long as UDRE is set (provided that global interrupts are enabled). UDRE is cleared by writing UDR. When interrupt-driven data transmission is used, the Data Register Empty Interrupt routine must either write new data to UDR in order to clear UDRE or disable the Data Register Empty Interrupt, otherwise a new interrupt will occur once the interrupt routine terminates.

USART Receiver – Data Reception

The USART Receiver is enabled by writing the Receive Enable (RXEN) bit in the UCSRB Register to one. When the receiver is enabled, the normal pin operation of the RxD pin is overridden by the USART and given the function as the receiver's serial input. The baud rate, mode of operation and frame format must be set up once before any serial reception can be done. If synchronous operation is used, the clock on the XCK pin will be used as transfer clock.

Receiving Frames with 5 to 8 Data Bit

The Receiver starts data reception when it detects a valid start bit. Each bit that follows the start bit will be sampled at the baud rate or XCK clock, and shifted into the Receive Shift Register until the first stop bit of a frame is received. A second stop bit will be ignored by the Receiver. When the first stop bit is received, i.e., a complete serial frame is present in the Receive Shift Register, the contents of the Shift Register will be moved into the receive buffer. The receive buffer can then be read by reading the UDR I/O location.

The following code example shows a simple USART receive function based on polling of the Receive Complete (RXC) Flag. When using frames with less than eight bits the most significant bits of the data read from the UDR will be masked to zero. The USART has to be initialized before the function can be used.

Receiving Frames with 5 to 8 Data Bit

```
unsigned char USART_Receive( void )
{
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) )
        ;
  /* Get and return received data from buffer */
  return UDR;
}
```

Receiving Frames with 9 Data Bit

If 9-bit characters are used (UCSZ=7) the ninth bit must be read from the RXB8 bit in UCSRB **before** reading the low bits from the UDR. This rule applies to the FE, DOR and UPE Status Flags as well. Read status from UCSRA, then data from UDR. Reading the UDR I/O location will change the state of the receive buffer FIFO and consequently the TXB8, FE, DOR and UPE bits, which all are stored in the FIFO, will change.

The following code example shows a simple USART receive function that handles both nine bit characters and the status bits.

Receiving Frames with 9 Data Bit

```
unsigned int USART_Receive( void )
{
  unsigned char status, resh, resl;
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) )
        ;
  /* Get status and 9th bit, then data */
  /* from buffer */
  status = UCSRA;
  resh = UCSRB;
  resl = UDR;
  /* If error, return -1 */
  if ( status & (1<<FE)|(1<<DOR)|(1<<UPE) )
    return -1;
  /* Filter the 9th bit, then return */
  resh = (resh >> 1) & 0x01;
  return ((resh << 8) | resl);
}
```

Receive Complete Flag and Interrupt

The USART Receiver has one flag that indicates the receiver state.

The Receive Complete (RXC) Flag indicates if there are unread data present in the receive buffer. This flag is one when unread data exist in the receive buffer, and zero when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled (RXEN = 0), the receive buffer will be flushed and consequently the RXC bit will become zero.

When the Receive Complete Interrupt Enable (RXCIE) in UCSRB is set, the USART Receive Complete Interrupt will be executed as long as the RXC Flag is set (provided that global interrupts are enabled). When interrupt-driven data reception is used, the receive complete routine must read the received data from UDR in order to clear the RXC Flag, otherwise a new interrupt will occur once the interrupt routine terminates.