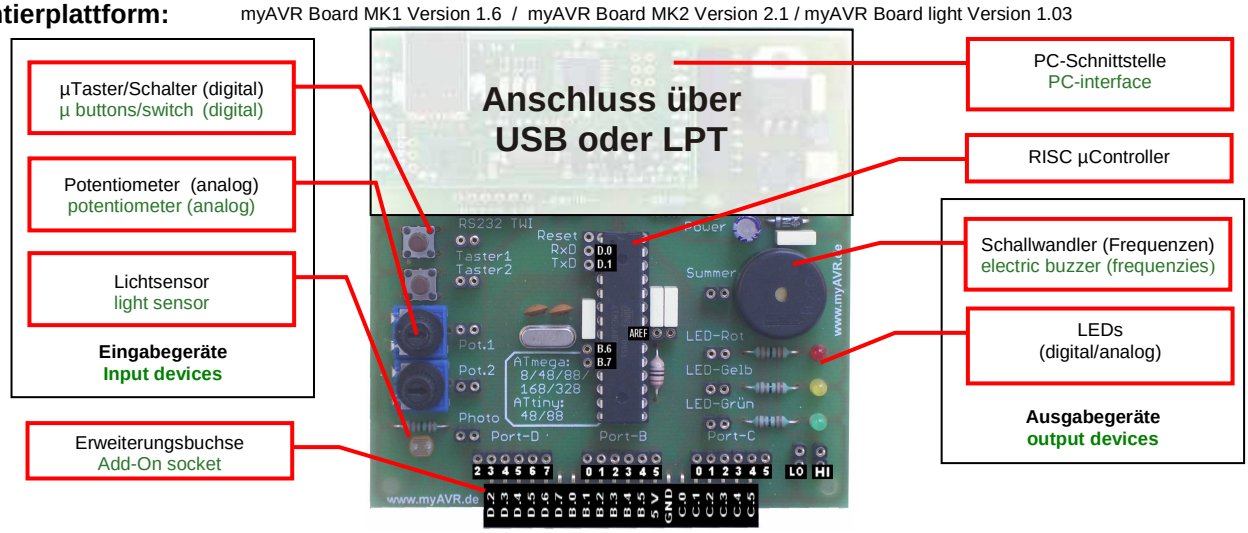
Experimentierplattform:

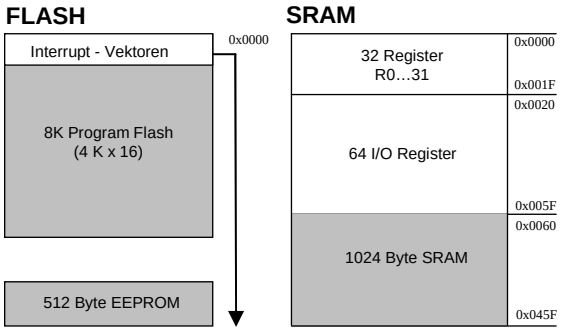
myAVR Board MK1 Version 1.6 / myAVR Board MK2 Version 2.1 / myAVR Board light Version 1.03



I/O Register

Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SREG	I	T	H	S	V	N	Z	C
SPH	–	–	–	–	–	SP10	SP9	SP8
SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
Reserved								
GICR	INT1	INT0	–	–	–	–	IVSEL	IVCE
GIFR	INTF1	INTF0	–	–	–	–	–	–
TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0
TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	–	TOV0
SPMCR	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN
TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
MCUCR	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00
MCUCSR	–	–	–	–	WDRF	BORF	EXTRF	PORF
TCCR0	–	–	–	–	–	CS02	CS01	CS00
TCNT0	Timer/Counter0 (8 Bits)							
OSCCAL	Oscillator Calibration Register							
SFIOR	–	–	–	–	ACME	PUD	PSR2	PSR10
TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10
TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10
TCNT1H	Timer/Counter1 – Counter Register High byte							
TCNT1L	Timer/Counter1 – Counter Register Low byte							
OCR1AH	Timer/Counter1 – Output Compare Register A High byte							
OCR1AL	Timer/Counter1 – Output Compare Register A Low byte							
OCR1BH	Timer/Counter1 – Output Compare Register B High byte							
OCR1BL	Timer/Counter1 – Output Compare Register B Low byte							
ICR1H	Timer/Counter1 – Input Capture Register High byte							
ICR1L	Timer/Counter1 – Input Capture Register Low byte							
TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20
TCNT2	Timer/Counter2 (8 Bits)							
OCR2	Timer/Counter2 Output Compare Register							
ASSR	-	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB
WDTCR	–	–	–	WDCE	WDE	WDP2	WDP1	WDP0
UBRRH	URSEL	–	–	–	UBRR[11:8]			
UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
EEARH	–	–	–	–	–	–	–	EEAR8
EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0
EEDR	EEPROM Data Register							
EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE
PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
PORTC	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
DDRC	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0
PINC	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0
PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
SPDR	SPI Data Register							
SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X
SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
UDR	USART I/O Data Register							
UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM
UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
UBRRL	USART Baud Rate Register Low byte							
ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0
ADMUX	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0
ADCSRA	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0
ADCH	ADC Data Register High byte							
ADCL	ADC Data Register Low byte							
TWDR	Two-wire Serial Interface Data Register							
TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0
TWBR	Two-wire Serial Interface Bit Rate Register 168							

Speicheraufbau / memory structure



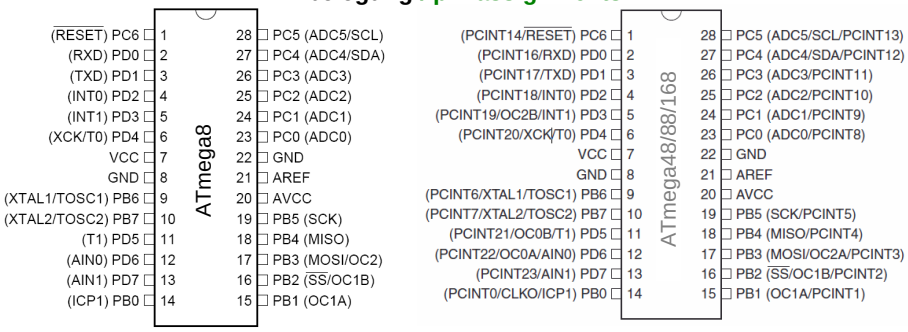
Programmstruktur / structure of program

```
'-----------------------------------------------------------------------
' Titel          : Lauflicht für  myAVR Board
'-----------------------------------------------------------------------
' Funktion       : Lauflicht
' Schaltung      : PD5-PD7 an LED`s
'-----------------------------------------------------------------------
' Prozessor      : ATmega8  3,6864 MHz
' Sprache        : BASCOM-AVR
' Version        : 1.2
'-----------------------------------------------------------------------
$regfile    = "m8def.dat"                ' Prozessortyp ATmega8
$crystal    = 3686400                    ' Taktrate
$hwstack    = 40                         ' 40 Byte Hardware-Stack
$swstack    = 32                         ' 32 Byte Software-Stack
$framesize  = 60                         ' 60 Byte Frame

'---------DEKLARATION-----------------------------------------------------
Dim Mybyte as Byte                       ' Globale Variable ein Byte
Declare Sub init()                       ' Unterfunktion

'---------INITIALISIERUNG-------------------------------------------------
Call init()

'---------MAIN------------------------------------------------------------
Do                                       ' Beginn Mainloop
     PORTD = Mybyte                      ' Ausgabe
     Waitms 100                          ' Warte kurz
     Rotate Mybyte, Left                 ' Bit laufen lassen (Rotation)
Loop                                     ' Ende Mainloop

End                                      ' Programmende

'---------UNTERFUNKTIONEN-------------------------------------------------
Sub init()
      Mybyte = 1
      DDRD  = &B11100000                 ' PD 5-PD 7 auf Ausgang
      PORTD = &B00000000                 ' alle LEDs off
End Sub
```

Interruptvektoren / interrupt vectors

Vector No.	Source	Interrupt Definition
1	RESET	External Pin,Power-on Reset,Brown-out Reset, and Watchdog Reset
2	INT0	External Interrupt Request 0
3	INT1	External Interrupt Request 1
4	TIMER2 COMP	Timer/Counter2 Compare Match
5	TIMER2 OVF	Timer/Counter2 Overflow
6	TIMER1 CAPT	Timer/Counter1 Capture Event
7	TIMER1 COMPA	Timer/Counter1 Compare MatchA
8	TIMER1 COMPB	Timer/Counter1 Compare MatchB
9	TIMER1 OVF	Timer/Counter1 Overflow
10	TIMER0 OVF	Timer/Counter0 Overflow
11	SPI; STC	Serial Transfer Complete
12	USART , RXC	USART, Rx Complete
13	USART , UDRE	USART, Data Register Empty
14	USART , TXC	USART, TX Complete
15	ADC	ADC Conversion Complete
16	EE_RDY	EEPROM Ready
17	ANA_COMP	Analog Comperator
18	TWI	Two-wire serial Interface
19	SPM_RDY	Store Program Memory Ready

Pinbelegung / pin assignments



Befehlssatz / instruction set

Instruction	Parameter	Description
COMPILER SETTING		
$CRYSTAL	$CRYSTAL = var	override the crystal frequency options setting
$REGFILE	$REGFILE = "name"	use specified register file instead selected dat file
MEMORY		
$HWSTACK	$HWSTACK = var	Sets the available space for the Hardware stack.
$SWSTACK	$SWSTACK = var	Sets the available space for the software stack.
$FRAMESIZE	$FRAMESIZE = var	Sets the available space for the frame.
DIM	DIM var AS [XRAM/SRAM/ERAM]type [AT location/variable] [OVERLAY]	Dimension a variable.
CONST	CONST symbol = numconst | stringconst | expression	Declares a symbolic constant.
ALIAS	new var ALIAS oldvar	variable can be referenced with another name
WRITEEEPROM	WRITEEEPROM var , address	write a variables content
READEEPROM	READEEPROM var , address	reads the content and stores it into a variable
$EEPROM	$EEPROM	store data in DATA lines in an EEP file
$DATA	$DATA	store data in DATA lines in code memory
RESTORE	RESTORE label	reread values in specified DATA statements
READ	READ var	reads values and assigns them to variables
LOOKUP	var = LOOKUP(value, label)	Returns a value from a lable.
LOOKDOWN	var = LOOKDOWN(value, label, entries)	Returns the index of a series of data.
CONFIG PORT	CONFIG PORTx = state CONFIG PINx.y = state	sets the port /port pin to the right data direction
INTERRUPT HANDLING		
CONFIG INTx	CONFIG INTx = state (X can be 0,1 and 4 to 7)	interrupts 0,1 & 4-7 will be triggered, in MEGA chips
ON interrupt	ON interrupt label [NOSAVE]	execute subroutine when specified interrupt occurs
ENABLE	ENABLE interrupt [, prio]	Enable specified interrupt.
DISABLE	DISABLE interrupt [device]	Disable specified interrupt.
TIMER		
CONFIG TIMERn	CONFIG TIMERn = COUNTER | TIMER | PWM, EDGE=RISING/FALLING, CLEAR TIMER = 1|0 , PRESCALE= 1|8|64|256|1024 NOISE CANCEL=0 |1, CAPTURE EDGE = RISING | FALLING COMPARE = CLEAR | SET | TOGGLE I DISCONNECT	Configure TIMER n
MCU CONTROL INSTRUCTIONS		
POWER MODE	POWER mode	put processor in power reserving modes
START	START device	Start the specified device.
STOP	STOP [device]	Stop the specified device. Or stop the program
BRANCH INSTRUCTIONS		
DO-LOOP	DO statements LOOP [UNTIL expression]	repeat block of statements until condition is true
IF-THEN-ELSE-END IF	IF expression THEN	allows conditional execution or branching
ELSE	ELSE	Executed if the IF-THEN expression is false.
FOR-NEXT	FOR var = start TO end [STEP value]	Execute a block of statements a number of times.
WHILE-WEND	WHILE condition statements WEND	Executes a series of statements in a loop, as long as condition is true.
SELECT-CASE-END SELECT	SELECT CASE var CASE test1 : statements [CASE test2 : statements] CASE ELSE : statements END SELECT	executes one of several statement blocks
GOTO	GOTO label	Jump to the specified label.
ON VALUE	ON var [GOTO] [GOSUB] label1 [, label2] [,CHECK]	branch to one of several specified labels
EXIT	EXIT FOR | DO | WHILE | SUB | FUNCTION	exit a ... FUNCTION
END	END	Terminate program execution.
SUBROUTINES/FUNCTIONS		
DECLARE SUB	DECLARE SUB TEST[([BYREF/BYVAL] var as type)]	Declares a subroutine.
DECLARE FUNCTION	DECLARE FUNCTION TEST[([BYREF/BYVAL] var as type)] As type	Declares a user function.
CALL	CALL Test [(var1, var-n)]	Call and execute a subroutine.
GOSUB	GOSUB label	Branch to and execute subroutine.
SUB	SUB Name[(var1 , …)]	Defines a Sub procedure.
BYVAL	Sub Test(BYVAL var)	Specifies that a variable will be passed by value.
RETURN	RETURN	Return from a subroutine.
WAITING INSTRUCTIONS		
WAIT	WAIT seconds	Suspends program execution for a given time.
WAITMS	WAITMS mS	Suspends program execution for a given time.
WAITUS	WAITUS uS	Suspends program execution for a given time.
BITWAIT	BITWAIT x , SET/RESET	Wait until a bit is set or reset.
ARITHMETIC AND LOGIC INSTRUCTIONS		
INCR	INCR var	Increments a variable by one.
DECR	DECR var	Decrements a variable by one.
ROUND	var = ROUND(x)	Returns a value rounded to the nearest value.
RND	var = RND(limit)	Returns a random number.
INT	var = INT(source)	Returns the integer part of a single or double.

Instruction	Parameter	Description
BIT AND STRING INSTRUCTIONS		
SET	SET bit / SET var.x / SET var	Set a bit to the value one.
RESET	RESET bit / RESET var.x / RESET var	Reset a bit to zero.
TOGGLE	TOGGLE pin / TOGGLE var	inverts state of output pin or bit variable
ROTATE	ROTATE var , LEFT/RIGHT[, shifts]	Rotate all bits one place to the left or right.
SHIFT	…	
STR	var = STR(x)	Returns a string representation of a number.
CHR	PRINT CHR(var) s = CHR(var)	numeric variable/constant to a string
VAL	var = VAL(s)	string representation of a number into a number
LEN	var = LEN(string)	Returns the length of a string.
LEFT	var = LEFT(var1 , n)	Set the contrast of a TEXT LCD.
RIGHT	var = RIGHT(var1 ,n)	return a specified number of rightmost characters
MAKEINT	varn = MAKEINT(LSB , MSB)	Compact two bytes into a word or integer.
MAKEBCD	var1 = MAKEBCD(var2)	Convert a variable into its BCD value.
HIGH	var = HIGH(s)	Retrieves the most significant byte of a variable.
LOW	var = LOW(s)	Retrieves the least significant byte of a variable.
SPLIT	count = SPLIT (source, array, search)	Split a string into a number of array elements.
FUSING	target = FUSING(source, "mask")	returns a formatted string of a single value
Time/Date		
CONFIG CLOCK	CONFIG CLOCK = soft | USER [, GOSUB = SECTIC]	configures timer used for TIME$ & DATE$
CONFIG DATE	CONFIG DATE = DMY , Separator = char	configure the Format of Date String
CONFIG DCF77	CONFIG DCF77 = pin , timer = timer [INVERTED=inv, CHECK=check, UPDATE=upd, UPDATETIME=updtime , TIMER1SEC=tmr1sec, SWITCHPOWER=swpwr, POWERPIN=pin, POWERLEVEL = pwrlvl , SECONDTICKS=sectick ,DEBUG=dbg , GOSUB = Sectic]	instruct the compiler to use DCF-77 radio signal
DATE$	DATE$ = "mm/dd/yy" var = DATE$	Internal variable that holds the date.
TIME$	TIME$ = "hh:mm:ss" var = TIME$	Internal variable that holds the time.
DAYOFWEEK	Target = DayOfWeek() | DayOfWeek(bDayMonthYear) |DayOfWeek(strDate) | DayOfWeek(wSysDay) | DayOfWeek(lSysSec)	Returns the Day of the Week of a Date.
LCD		
CONFIG LCD	CONFIG LCD = LCDtype , CHIPSET=KS077 | Dogm163v5 | DOG163V3 | DOG162V5 | DOG162V3 [,CONTRAST=value]	configure LCD display
CONFIG LCDBUS	CONFIG LCDBUS = constant	configures the LCD data bus
CONFIG LCDMODE	CONFIG LCDMODE = type	configures the LCD operation mode
CONFIG LCDPIN	CONFIG LCDPIN = PIN , DB4= PN,DB5=PN, DB6=PN, DB7=PN, E=PN, RS=PN [BUSY=PIN] [MODE=mode], PORT=PORTx, E=PN, RS=PN	Override the LCD-PIN select options.
INITLCD	INITLCD	Initializes the LCD display.
LCD	LCD x	Send constant or variable to LCD display.
DISPLAY	DISPLAY ON | OFF [, CURSOR | NOCURSOR , BLINK | NOBLINK]	Turn LCD display ON or OFF.
CURSOR	CURSOR ON / OFF BLINK / NOBLINK	Set the LCD Cursor State.
CLS	CLS [TEXT] [GRAPH] CLS Y, X1 , X2 [, CHAR]	Clears the display/screen
HOME	HOME UPPER | LOWER | THIRD | FOURTH	Place the cursor at the specified line at location
LOCATE	LOCATE y , x	Moves the LCD cursor to the specified position.
*LINE	UPPERLINE | LOWERLINE | THIRDLINE | FOURTHLINE	Reset LCD cursor to the *line.
DEFLCDCHAR	DEFLCDCHAR char,r1,r2,r3,r4,r5,r6,r7,r8	Define a custom LCD character.
DATA TRANSFER INSTRUCTIONS		
$BAUD	$BAUD = var	override the baud rate setting
CONFIG COMx	CONFIG COMx = baud , synchrone=0|1, parity=none|disabled|even|odd, stopbits=1|2,databits=4|6|7|8|9, clockpol=0|1	Configures serial interface.
PRINT	PRINT [#channel ,] var ; " constant"	output to RS-232 port.; writes string to file; data to device
OPEN / CLOSE	OPEN "device" for MODE As #channel CLOSE #channel	open/close serial interface.
CONFIG SDA	CONFIG SDA = pin	overrides the SDA pin assignment
CONFIG SCL	CONFIG SCL = pin	overrides the SCL pin assignment
I2CINIT	I2CINIT I2CINIT #const	Initializes the SCL and SDA pins.
I2CRECEIVE	I2CRECEIVE slave, var [, b2W, b2R]	Receives data from an I2C serial slave device.
I2CSEND	I2CSEND slave, var [, bytes]	Send data to an I2C-device.
I2START,I2CSTOP, I2CRBYTE, I2CWBYTE	I2CSTART I2CSTOP I2CRBYTE var, ack/nack I2CWBYTE var	receives/ sends one byte; generates an condition
SPECIAL FUNCTIONS		
ENCODER	Var = ENCODER(pin1, pin2, LeftLabel, RightLabel , wait)	Reads pulses from a rotary encoder.
SOUND	SOUND pin, duration, pulses	Sends pulses to a port pin.
CONFIG DEBOUNCE	CONFIG DEBOUNCE = time	This statement configures the debounce time.
DEBOUNCE	DEBOUNCE Px.y , state , label [, SUB]	Debounce a port pin connected to a switch.
CONFIG SERVOS	CONFIG SERVOS = X , Servo1 = Portb.0 , Servo2 = Portb.1 , Reload = rl	Configures how much servo's will be controlled.
CONFIG ADC	CONFIG ADC = single, PRESCALER = AUTO, REFERENCE = opt	Configure the analouge to digital converter.
GETADC	var = GETADC(channel)	retrieves analog value from specified channel