Tabelle1

Module	Function / Macro / Constant	#define in File “config.h“
main_general.h	setup()	none
	loop()	
	HIGH / LOW	
	true / false	
	boolean	
	string	
	char(d)	
	byte(d)	
	int()	
	word(a)	
	wordConcat(hb,lb)	
	long(c)	
	float(d)	
	min(a,b)	
	max(a,b)	
	abs(a)	
	constrain(x, low, high)	
	map(x,inMin,inMax,outMin,outMax)	
	pow(x,y)	
	sqrt(x)	
	sin(a)	
	cos(a)	
	tan(a)	
	isAlphaNumeric(a)	
	isAlpha(a)	
	isAscii(a)	
	isWhitespace(a)	

Tabelle1

misc.h (auto loaded)	isControl(a)	none
	isDigit(a)	
	isGraph(a)	
	isLowerCase(a)	
	isPrintable(a)	
	isPunct(a)	
	isSpace(a)	
	isUpperCase(a)	
	isHexadecimalDigit(a)	
	randomSeed(d)	
	random()	
	lowByte(x)	
	highByte	
	bitRead(byte, bit)	
	bitWrite(byte, bit, value)	
	bitSet(byte, bit)	
	bitClear(byte, bit)	
	bitToggle(byte, bit)	
	bit(n)	
	interrupts()	
	noInterrupts()	
	B00000000 – B11111111	
	round(x)	
	ceil(x)	
	floor(x)	
	toASCII(c)	
	toUpperCase(c)	
	toLowerCase(c)	

Tabelle1

	log2(d)	
	floatToString(buf, value, digits)	USE_FTOA
gpio (auto loaded)	pinMode(port, pin, mode)	none
	pinSet(port, pin)	
	pinRead(port, pin)	
	portSet(port)	
	portRead(port)	
	attachInterruptPort(portAddr, fctName, edge)	USE_PORT_ISR
	detachInterruptPort(portAddr)	
	attachInterruptPin(fctName, edge)	USE_TLI_ISR
	detachInterruptPin()	
sw_delay (auto loaded)	sw_delay(uint32_t N)	none
	sw_delayMicroseconds(uint16_t N)	
	sw_delayNOP(uint8_t N)	
stm8as (auto loaded)	ASM(mnem)	none
	NOP	
	WAIT_FOR_INTERRUPT	
	ENTER_HALT	
timer4 (auto loaded)	uint32_t millis()	none
	uint32_t micros()	
	flagMilli()	
	clearFlagMilli()	
	resetTime()	
	attachInterruptMillis(fct)	USE_MILLI_ISR
	detachInterruptMillis()	

Tabelle1

uart1_blocking	UART1_begin(baudrate)	none
	UART1_end()	
	UART1_listen()	
	UART1_write(data)	
	UART1_writeBytes(num, buf);	
	UART1_available()	
	UART1_read()	
putchar	putcharAttach(fct)	none
	putcharDetach()	
tone (requires option byte change)	tone(uint16_t Hz, uint16_t millis)	none
	noTone()	

Tabelle1

Example
setup()
loop()
LED = HIGH;
if (a==true)
boolean a;
string s[20];
c = char(d);
b = char(d);
d = int(c);
w = word(a);
w = wordConcat(hb, lb);
d = long(c);
f = float(d);
a = min(b,c);
a = max(b,c);
a = abs(a);
a = constrain(a, 10, 100);
b = map(a, 0,1024, 0,100);
y = pow(x, 0.3)
y = sqrt(x)
y = sin(x);
y = cos(x);
y = tan(x);
if (isAlphaNumeric(a))
if (isAlpha(a))
if (isAscii(a))
if (isWhitespace(a))

Tabelle1

if (isControl(a))
if (isDigit(a))
if (isGraph(a))
if (isLowerCase(a))
if (isPrintable(a))
if (isPunct(a))
if (isSpace(a))
if (isUpperCase(a))
if (isHexadecimalDigit(a))
randomSeed(10);
a = random();
LB = lowByte(x);
HB = highByte(x);
a = bitRead(b, 4)
bitWrite(a, 3, 1);
bitSet(a, 3);
bitClear(a, 3);
bitToggle(a, 3);
a = bit(3);
interrupts();
noInterrupts()
value = B10100000;
a = round(a);
a = ceil(a);
a = floor(a);
c = toASCII(c);
c = toUpperCase(c);
c = toLowerCase(c);

Tabelle1

n = log2(d)
printf("%s\n", floatToString(str,x,3));
pinMode(PORT_H, pin3, OUTPUT);
pinSet(PORT_H, pin3) = state;
state = pinRead(PORT_D, pin7);
portSet(PORT_H) = portState;
portState = portRead(PORT_H);
attachInterruptPort(&PORT_E, fct, FALLING);
detachInterruptPort(&PORT_E);
attachInterruptPin(fct, FALLING);
detachInterruptPin();
sw_delay(10);
delayMicroseconds(10);
sw_delayNOP(100);
ASM("trap");
NOP;
WAIT_FOR_INTERRUPT;
ENTER_HALT;
time_ms = millis();
time_us = micros();
if (flagMilli())
clearFlagMilli();
resetTime();
attachInterruptMillis(fct);
detachInterruptMillis();

Tabelle1

UART1_begin(19200);
UART1_end();
UART1_listen();
UART1_write(c);
UART1_writeBytes(num, buf);
if (UART1_available())
Rx = UART1_read();
putcharAttach(UART1_write);
putcharDetach();
beep(2000, 500);
noTone()

Tabelle1

Short Description / Remark
user initialization routine. Called once after start of program
user loop routine. Called continuously
constants for 1 / 0, e.g. for pinSet()
constants for 1 / 0, e.g. for if
Boolean variable. Same as uint8_t
Character array. Same as char*
Converts a value to the char data type. Same as ((char) d)
Converts a value to the byte data type. Same as ((uint8_t) a)
Converts a value to the int data type.
Convert a value to the word data type.
Convert a word from two bytes.
Converts a value to the long data type.
Converts a value to the float data type.
minimum of 2 numbers; do not use as function argument
maximum of 2 numbers; do not use as function argument
absolute value of a number; do not use as function argument
clip value to range [low;high]; do not use as function argument
re-map a number from one range to another
Calculates the value of a number raised to a power.
Calculates the square root of a number.
Calculates the sine of an angle (in radians). The result is in [-1;1].
Calculates the cosine of an angle (in radians). The result is in [-1;1].
Calculates the tangent of an angle (in radians). The result is in [-inf;inf]
Analyse if a char is alphanumeric.
Analyse if a char is is alpha.
Analyse if a char is ASCII.
Analyse if a char is a white space.

Tabelle1

Analyse if a char is a control character.	
Analyse if a char is a digit.	
Analyse if a char is a printable character.	
Analyse if a char is a lower case character.	
Analyse if a char is a printable character.	
Analyse if a char is punctuation character.	
Analyse if a char is a space character.	
Analyse if a char is a upper case character.	
Analyse if a char is a valid hexadecimal digit.	
seed the random number generator used by the random()	
generate a pseudo random number within [0;INT16_MAX]	
Extracts the low-order (rightmost) byte of a variable (e.g. a word)	Change for compatibil
Extracts the high-order (leftmost) byte of a word (or the second lowest byte of a larger data type).	
read single bit position in byte	Change for compatibil
set single bit value in byte to value	Change for compatibil
set single bit in data to '1'	Change for compatibil
clear single bit in data to '0'	Change for compatibil
toggle single bit state in byte	Change for compatibil
calculate bit value of bit n	Change for compatibil
Globally enable interrupts	
Globally disable interrupts	
Binary number literals	change from bxxxxx f
round x to the nearest integer	
round x upwards to the nearest integer	
round x downwards to the nearest integer	
return lower 7 bits of 1B argument (ASCII range)	
converts an alpha to upper case letter	
converts an alpha to lower case letter	

Tabelle1

Integer calculation of (rough) log2(x), i.e. determine binary power to reach number	
convert float to string for printing floats. No scientific notation. Is rather large → only include if required	new
Set pin direction and optional features. Pin modes are INPUT, INPUT_INTERRUPT, INPUT_PULLUP, INPUT_PULLUP_INTERRUPT OUTPUT, OUTPUT_OPENDRAIN	
Set pin state	
Read pin state	
Set port state (8 pins)	
Read port state (8 pins)	
Attach user routine to port interrupt (=EXINTx). Edges are LOW, CHANGE, RISING, FALLING, PREV_SETTING Enable pin interrupt via pinMode()	
Detach user routine from port interrupt (=EXINTx). Disable pin interrupt via pinMode()	
Attach user routine to pin D7 interrupt (=TLI). Edges are LOW, CHANGE, RISING, FALLING, PREV_SETTING Enable pin interrupt via pinMode()	
Detach user routine from pin D7 interrupt (=TLI). Disable pin interrupt via pinMode()	
Delay code for approximately N milliseconds without timer. Timing depends on interrupt load (inline blocking) For compiler / optimization dependent latency see sw_delay.h	fix re-entrance bug &
Delay code for approximately N microseconds without timer. Timing depends on interrupt load (inline blocking) For compiler / optimization dependent latency see sw_delay.h	calibrate timing for de
Delay code for Nx NOP() (inline blocking) For compiler / optimization dependent latency see sw_delay.h	
Inline STM8 assembler	
NOP operation (1 CPU cycle)	change from _NOP_ f
Halt core with clock running. Resume execution, e.g. by timer interrupt	
Halt core and clock. Resume execution e.g. by auto-wakeup, see “awu“	
Milliseconds since start of program	
Microseconds since start of program with 4µs resolution	
Check if 1ms has passed. Reset by clearFlagMilli()	
Reset flagMilli() flag for 1ms	
Reset millis and micros to 0	
Attach user routine to 1ms interrupt (=TIM4UPD)	
Detach user routine from 1ms interrupt (=TIM4UPD)	

Tabelle1

initialize UART1 baudrate and enable sender & receiver	new
disable sender & receiver	
enable sender & receiver. Retain previous settings	
send 1 byte via UART1	new
send N bytes via UART1	new
check if byte received via UART1	new
read byte from UART1 receive buffer. Non-blocking	new
set send routine (1B) for stdio putchar / printf; For printing floats, use below float2str() helper routine	new
detach send routine from stdio putchar / printf	new
play tone via beeper module with given frequency in Hz (<500 off) and duration in millis (0=forever)	change from beep for
switch off tone started with tone() and duration=0 (see above)	new

lity with Arduino

lity with Arduino

lity with Arduino

lity with Arduino

lity with Arduino

lity with Arduino

lity with Arduino

or compatibility with Arduino

calibrate timing for debug/optimize

bug/optimize

or readability

compatibility with Arduino. Added flexibility