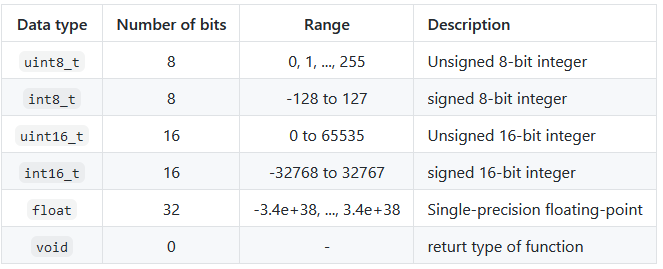
1.



#include <avr/io.h>

// Function declaration (prototype)

*uint16\_t* calculate(*uint8\_t*, *uint8\_t*);

int main(void)

{

*uint8\_t* a = 156;

*uint8\_t* b = 14;

*uint16\_t* c;

// Function call

c = calculate(a, b);

while (1)

{

}

return 0;

}

// Function definition (body)

*uint16\_t* calculate(*uint8\_t* x, *uint8\_t* y)

{

*uint16\_t* result; // result = x^2 + 2xy + y^2

result = x\*x;

result += 2\*x\*y;

result += y\*y;

return result;

}

2.

gpio.c

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* GPIO library for AVR-GCC.

\* ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2

\*

\* Copyright (c) 2019-2020 Tomas Fryza

\* Dept. of Radio Electronics, Brno University of Technology, Czechia

\* This work is licensed under the terms of the MIT license.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Includes ----------------------------------------------------------\*/

#include "gpio.h"

/\* Function definitions ----------------------------------------------\*/

void GPIO\_config\_output(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num)

{

\*reg\_name = \*reg\_name | (1<<pin\_num);

}

/\*--------------------------------------------------------------------\*/

/\* GPIO\_config\_input\_nopull \*/

void GPIO\_config\_input\_nopull(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num)

{

\*reg\_name = \*reg\_name & ~(1<<pin\_num); //set pin as input

\*reg\_name++;

\*reg\_name = \*reg\_name & ~(1<<pin\_num); //turn off internal pull-up

}

/\*--------------------------------------------------------------------\*/

void GPIO\_config\_input\_pullup(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num)

{

\*reg\_name = \*reg\_name & ~(1<<pin\_num); // Data Direction Register

\*reg\_name++; // Change pointer to Data Register

\*reg\_name = \*reg\_name | (1<<pin\_num); // Data Register

}

/\*--------------------------------------------------------------------\*/

void GPIO\_write\_low(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num)

{

\*reg\_name = \*reg\_name & ~(1<<pin\_num);

}

/\*--------------------------------------------------------------------\*/

/\* GPIO\_write\_high \*/

void GPIO\_write\_high(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num)

{

\*reg\_name |= (1<<pin\_num);

}

/\*--------------------------------------------------------------------\*/

/\* GPIO\_toggle \*/

*uint8\_t* GPIO\_toggle(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num)

{

\*reg\_name ^= (1<<pin\_num);

}

/\*--------------------------------------------------------------------\*/

/\* GPIO\_read \*/

*uint8\_t* GPIO\_read(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num)

{

return(bit\_is\_set(\*reg\_name, pin\_num));

}

main.c

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Alternately toggle two LEDs when a push button is pressed. Use

\* functions from GPIO library.

\* ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2

\*

\* Copyright (c) 2019-2020 Tomas Fryza

\* Dept. of Radio Electronics, Brno University of Technology, Czechia

\* This work is licensed under the terms of the MIT license.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Defines -----------------------------------------------------------\*/

#define LED\_GREEN PB5 // AVR pin where green LED is connected

#define LED\_RED PC0 // AVR pin where red LED is connected

#define BTN PD0 // AVR pin where button is connected

#define BLINK\_DELAY 500

#ifndef *F\_CPU*

#define *F\_CPU* 16000000 // CPU frequency in Hz required for delay

#endif

/\* Includes ----------------------------------------------------------\*/

#include <util/delay.h> // Functions for busy-wait delay loops

#include <avr/io.h> // AVR device-specific IO definitions

#include "gpio.h" // GPIO library for AVR-GCC

/\* Function definitions ----------------------------------------------\*/

/\*\*

\* Main function where the program execution begins. Toggle two LEDs

\* when a push button is pressed. Functions from user-defined GPIO

\* library is used instead of low-level logic operations.

\*/

int main(void)

{

/\* GREEN LED \*/

GPIO\_config\_output(&DDRB, LED\_GREEN);

GPIO\_write\_low(&PORTB, LED\_GREEN);

/\* second LED \*/

GPIO\_config\_output(&DDRC, LED\_RED);

GPIO\_write\_low(&PORTB, LED\_GREEN);

/\* push button \*/

GPIO\_config\_input\_pullup(&DDRD, BTN);

// Infinite loop

while (1)

{

// Pause several milliseconds

*\_delay\_ms*(BLINK\_DELAY);

if(GPIO\_read(&PIND, BTN) == 0)

{

GPIO\_toggle(&PORTB, LED\_GREEN);

GPIO\_toggle(&PORTC, LED\_RED);

}

}

// Will never reach this

return 0;

}

declaration

It is done in .h file (or in main.c if project is very short). Consist of return type variable, name of function and input name and type of variable(s).

Example

void GPIO\_write\_low(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num);

Definition  
It is done in .c file. The line is same as in .h file and in brackets - As name suggest is definition of function ( the algorithm is written here and return variable is specified – if not void).

Example

void GPIO\_write\_low(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num)

{

\*reg\_name = \*reg\_name & ~(1<<pin\_num);

}