Function

A function is a type of procedure.

A function is a self contained block of statements that perform a consistent task of some kind.

"Every program is collection of functions.

When the program passes control to a function, the function performs that task & returns control to the instruction following the calling instruction.

" The function may perform a stand alone task, such as initializing a set of variables or it may accepts values from the calling instruction, process them & pass the results back when finished.

Advantage of Using Functions.

- We can divide c program in smaller modules.
- We can call module whenever require. for example suppose we have written calculator program then we can write 4 modules (i.e. add, Sub, multiply, divide)
- Modular programming makes program more readable.
- Modules once created, can be reused in other programs
- Individual fn. can be easily built, tested.
- program development become easy
- frequently used function can be put togather in the customized library.

A function can call other function & also itself.

Types of function

1. System defined
   or
   Built-in functions
   or
   library function

2. User defined function

* There are two required functions in an Arduino sketch,

```
setup()
loop()
```

System defined or built in is a function type, under functions are built in with C compiler which you can call & use

Standard library of C function in Arduino IDE.

Arduino IDE comes with set of standard libraries for commonly used functionality.

Standard libraries are pre installed in the libraries folder of the Arduino install.

Digital I/O functions :

digitalRead ( )

Reads the value from a specified digital pin, either High or LOW.

Syntax —

digitalRead (pin)

parameters: pin — the Arduino pin number you want to read.

returns — HIGH or LOW.

Example —

Sets pin 13 to the same value as pin 7, declared as an input

```
int ledPin = 13; // LED connected to digital pin 13
int inPin = 7; // pushbutton connected to digital pin 7
int val = 0; // variable to store the read value.

void setup()
{
pinMode(ledPin, OUTPUT); // sets the digital pin 13 as output

pinMode(inPin, INPUT); // sets the digital pin 7 as input

}

void loop()
{
val = digitalRead(inPin); // (read the input pin)

digitalWrite(ledPin, val); // sets the LED to the buttons value
```

digitalWrite( ) —

Write a High or Low value to a digital Pin

If the pin has been configured as an OUTPUT with pinMode( ), the voltage will be set to the corresponding value.

If the pin is configured as an INPUT, digitalWrite( ) will enable (HIGH) or disable (LOW) the internal pull up on the input pin.

It is recommended to set the pinMode( ) to INPUT_PULLUP to enable the internal pull up register.

Syntax :—

```
digitalWrite( pin, Value)
```

Parameters –

Pin – the Arduino pin number

Value – HIGH or LOW

Returns – nothing.

Example :

```
void setup( )
 {
  pinMode( 13, OUTPUT); // Set the digital pin 13 as
                           output
  }
void loop( )
   {
    digitalWrite(13, HIGH); // Sets the digital pin 13 on
    delay (1000);    // waits for a second.
    digitalWrite (13, LOW); // Sets the digital pin 13 off
    delay (1000);    // waits for a second.

   }.
```

pinMode( ) :→

Configures the specified pin to behave either as an input or an output.

Syntax :

pinMode(pin, mode)

parameters :

pin – the Arduino pin number to the set the mode of.

mode : INPUT, OUTPUT or INPUT_PULLUP

INPUT, OUTPUT or INPUT_PULLUP.

Return – nothing.

Ex—

```
void setup( )
{
  pinMode(13, OUTPUT);
}
void loop( )
  {
   digitalWrite(13, High);
   delay(1000);
   digitalWrite(13, LOW);
   delay(1000);
  }
```

Analog input pins can be used as digital pins.

Analog functions

analogRead()

This function reads the value from the specified analog pin.

In the lower right part of the Arduino Board, you will see six pins marked "Analog In".

By using analogRead() function, we can read the voltage applied to one of the pins

analogRead() function returns a number between 0 & 1023, which represents voltage between 0 & 5 volts.

Ex: if there is voltage of 2.5V applied to pin number 0, analogRead(0) returns (512)

Syntax:

```
analogRead(pin);
```

pin – The number of the analog input pin to read from (A0 to A5 or A0 to A7 mini board) (A0 to A11)

Ex :—

```
int analogPin =A3;
// potentiometer wiper (middle terminal) connected to analog pin 3

int val=0;

void setup()
  {
    Serial.begin(9600); // Setup serial.
  }

void loop()
  {
    val= analogRead(analogPin); // read the input Pin
    Serial.println(val); // debug value.
  }
```

analogWrite() :—

Writes an analog value (PWM wave) to a pin.

After a call to analogwrite(), the pin will generate a steady square wave of the specified duty cycle until the next call to analogWrite().

Note: It is not necessary to call pinMode() to set the pin as an output before calling analogwrite()

Syntax : `analogWrite(pin, value)`

pin - the Arduino pin to write to (allowed data types - int

Value - the duty cycle : between 0 (always off) & 255 (always on).

Returns nothing.

Example -

```
int ledPin = 9;  // LED connected to digital Pin 9
int analogPin = 3; // Potentiometer connected to analog Pin 3
int val = 0;  // variable to store the read value.

void setup() {
  pinMode(ledPin, OUTPUT) // sets the pin as Output.
                  }

void loop()
  {
   val = analogRead(analogPin); // read the input pin
   analogWrite(ledPin, val/4); // analogRead values
                               go from 0 to 1023
                               analogwrite values from 0 to 255.
```

* * * *

PWM pin :

PWM pin provide 8-bit PWM output.

PWM stands for PULSE Width Modulation

66 It is used to convert the digital signal into an analog by varying the width of the pulse.

Ex: they are used to set the LED Brightness or to run Stepper or Servo Motor or anything which require analog inputs.

I/o Functions :

analogReference ( )

Configures the reference voltage used for analog input.

Syntax –

analogReference (type);

type : DEFAULT | INTERNAL | INTERNAL 1V1 | EXTERNAL

Time Functions

delay() : pauses the program for the amount of time (in miliseconds) specified as parameter.

Note : There are 1000 miliseconds in a Second)

Syntax : delay (ms)

parameters :

ms : the number of miliseconds to pause.

delayMicroseconds( )

pause the program for the amount of time.

micros( )

It returns the number of microseconds since the Arduino board began running the current program.

millis( )

Returns the number of milisecond passed since the Arduino board began ruming the current program.

Random Number functions :

randomseed( ) The function randomseed (seed) resets Arduino's pseudo random number generator.

Communication function

Serial — Used for communication between the Arduino board & a computer or other devices. All Arduino boards have at least one serial port (also known as UART or USART)

Example

```
if(Serial)
available()
availableForWrite()
begin()
end()
find()
findUntil()
flush()
peek()
print()
println()
read()
write()
SerialEvent()
```

Serial.begin() —

Everything begins with serial.begin() function.

This function in the setup routine is executed only once, i.e when the Arduino is starting.

Serial.print() & Serial.println()

These functions behave almost identically. prints data to the serial port as human readable ASCII text.

But In version also adds a carriage return and a new line.

Serial.Write() :-

writes binary data to the serial port.

This data is sent as a byte or series of bytes.

```
Serial.write(val)
Serial.write(str)
Serial.write(buf, len)
```

Serial.flush()

waits for the transmission of outgoing serial data to complete

Syntax : `Serial.flush();`

PROTOTYPE OF A FUNCTION.

A prototype declares the function name, its parameters and its return type to the rest of the program prior to the function's actual declaration.

A function prototype has the following syntax:

```
return_type function_name( parameter1, ---- n)
        {
          Declaration;
          Statements;
          Return value;
        }
```

return type :—

Return type is Typeof value returned by function.

Return type may be void, if the function is not going to return a value.

The void keyword is used only in function declarations. It indicates that the function expected to return no information to the function.

function name –

It is Unique name that identifies function.

Return Value –

It is value returned by function upon termination.

Formal Parameters & Actual Parameters

You can pass different- different arguments to a function. if the function takes multiple arguments. the arguments listed in the function call are assign to the function parameters in specific order.

The first argument is assign to the first parameter, next to the next.

The parameters that appears in a function call statement are actual arguments.

The parameters that appears in function definition are formal arguments.

i.e. the function creates its own copy of argument values and then uses them.

The formal arguments may be declared by the same or different names in the calling portion of the program.

The argument that is passed is often called an actual argument & while the received copy is called a formal argument or formal parameter.

Example —

```
int sum( int x, int y ); // x & y are formal parameters.
{
  return ( x+y ) ;
}

Void loop ( )
{
   int a = 10 ; b = 20, result = 0;
  result = sum(a, b); // a & b are the actual parameters in this call.
```

Function Call :

Once a function is defined it can be called to perform its proposed operation. To call a function, you simply need to pass the required parameters along with the function name & if the function returns a value, then you can restore the returned value.