

Table of Contents

● Release History	2
● Background	2
● What's MassDuino	2
● MassDuino Development Process	3
● MD-8088 and MD-328D specification.....	4
● MassDuino UNO family selection guide	5
● How to use.....	5
● My first MassDuino sketch.....	6
● About the analogRead speed for MassDuino MD-328D	9
● How to use MD-328D to replaced ATmega328P in Arduino system	9
● MD-3248P --- new chip released on 2018	11
1. digitalToggle()	11
2. fastioMode()	12
3. sysClock ()	13
4. getBootResetFlag ()	14
5. wdt_enable ()	15
6. analogRead ()	16
7. analogReference ()	18
8. pwmMode ()	19
9. MsTimer1.setMicros ()	22
10. DAC usage	24
11. PMU (Power Management Unit)	25
12. Virtual USB HID sample.....	27

● Release History

- 2014-7-23 : **V1.0** , UM-MASSDUINO-V01-EN
First released
- 2015-11-20 : **V2.5** , UM-MASSDUINO-V2.5-EN
1, Add MD-328D Support
2, Update MassDuino HSP
- 2016-04-11: **V3.0**, UM-MASSDUINO-V3.0-EN
Update new version Arduino IDE Support
- 2018-08-23: **V4.6**, UM-MASSDUINO-V4.6-EN
New chip support **MD-3248P**
- 2019-09-27: **V5.0**, UM-MASSDUINO-V50-EN
New chip support **Mega2560-CORE**

● Background

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It is a very useful and applied tools. We can find many code and sharing information from internet. Arduino development board has good scalability/ compatibility, and a wide range for application. So we can extend from the Arduino board of the modules all type what we need. It's intended for artists, designers, hobbyists and anyone interested in creating interactive objects or environments.

Arduino can be very easy to implement prototypes for the original design verification, but if you want a large-scale commercial applications, the price of Arduino is still too high, so there very little Arduino-based commercial products on the market.

So we released **MassDuino**, a new solution that is easy to use Arduino platform advantages, combined with low manufacturing costs , making the products which is developed on Arduino platform can be mass-produced immediately, quickly turn ideas into products.

● What's MassDuino

Massduino is a new product line, which combines the Arduino platform peripheral -rich, convenient and quick development, low-cost and easy to manufacture large-scale production advantages. Almost all of the Arduino code can be applied to MassDuino without modification (or very small modification), users do not need to learn any new knowledge, you can immediately begin using MassDuino to commercial product development.

MassDuino use a special custom MCU MD-8088 and MD-328D, those chips has a very unique and new design, ensuring high operating efficiency while providing a low cost of applications.

INHAOS upcoming a series of application modules which is based MassDuino. The application modules can be developed in the Arduino environment, and then direct used to commercial products, creative implementation and production time reduced to a minimum.

Stage



● MassDuino Development Process

This chapter describe how to development a product with MassDuino, We assume that you have some basic knowledge of electronic technology, and familiar with the Arduino development environment.

S1, Understand what is MassDuino, the MassDuino series products are highly compatible with standard Arduino UNO R3, So you can use MassDuino like the Arduino UNO R3.

S2, Download MassDuino HSP (Arduinio 3rd-party Hardware Support Package) from www.inhaos.com , At present the latest version of the HSP is V4.5, this version support chips is as follows:

MD-8088: 8KB Flash , 1KB SRAM, 10bit ADC

MD-328D: 32KB Flash , 2KB SRAM, 10/12/16 bit ADC

MD-3248P: 32K Flash, 2KB SRAM, 10/12/16bit ADC, 48Pin --- 2 Apr.2018 Add

MD-328D is highest compatible with the ATmega328P, we recommended to use MD-328D, in most case the Arduino UNO sketch can be used for MD-328P without any modification.

S3, Get one UNO development board , write sketch , run and debug .
Currently we have below products:

MD-8088: MassDuino UNO R4

MD-328D: MassDuino UNO LC , MassDuino UNO LC Lite

MD-3248P: MassDuino UNO Pro, MassDuino UNO LC Pro, MassDuino UNO Core Pro --- 2 Apr.2018 Add

S4, After prototype verified, do the mass production design, we provided below services:

- 1, We can help you do the whole product design and fabrication
- 2, Or you can do the design , we will help you make the prototype / pilot run / and mass production
- 3, Or you can buy chip from us , we can pre-programmed the chip or leave blank chip to you , with very good price

Need above service , please contact: support@inhaos.com

● MD-8088 and MD-328D specification

MD-8088

- Hi performance, Low power consumption 8bit RISC MCU
- 8K bytes of in-system programmable FLASH
- 1K bytes SRAM on-chip
- 504 bytes of data FLASH, support byte read (simulate E2PROM)
- Can be edited in the Arduino environment, concise and easy to use
- Programmable synchronous / asynchronous USART
- Can work in master / slave mode SPI Serial Interface
- Up to 30 programmable I / O
- High-performance, low -power and low-cost
- I2C -compatible two-wire serial communication interface protocols , supporting master and slave device mode



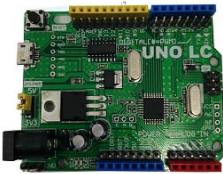
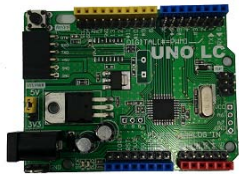
MD-328D

- Hi performance, Low power consumption 8bit RISC MCU
- **32K** bytes of in-system programmable FLASH
- **2K** bytes SRAM on-chip
- **1K** bytes of data FLASH, support byte read (simulate E2PROM)
- Can be edited in the Arduino environment, concise and easy to use
- Programmable synchronous / asynchronous USART
- Can work in master / slave mode SPI Serial Interface
- Up to 30 programmable I / O
- High-performance, low -power and low-cost
- I2C -compatible two-wire serial communication interface protocols , supporting master and slave device mode
- 8-CH **10bit** 250Ksps ADC
- 1.8 to **5.5V**

MD-3248P

- Hi performance, Low power consumption 8bit RISC MCU
- **32K** bytes of in-system programmable FLASH
- **2K** bytes SRAM on-chip
- **1K** bytes of data FLASH, support byte read (simulate E2PROM)
- Can be edited in the Arduino environment, concise and easy to use
- Programmable synchronous / asynchronous USART
- Can work in master / slave mode SPI Serial Interface
- High-performance, low -power and low-cost
- I2C -compatible two-wire serial communication interface protocols , supporting master and slave device mode
- 1.8 to **5.5V**
- **TQFP-48**

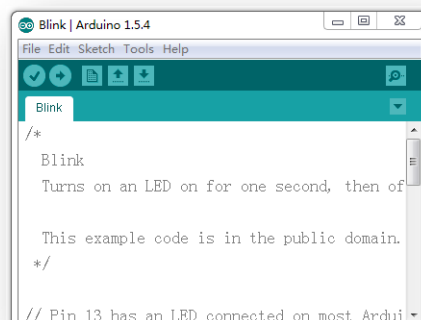
● MassDuino UNO family selection guide

No.	Items	BUONO UNO LC	MassDuino UNO R4	MassDuino UNO LC	MassDuino UNO LC Lite
1	Microcontroller	ATmega328P	MD-8088	MD328D	MD328D
2	Operation Voltage	3.3V or 5V	3.3V or 5V	3.3V or 5V	3.3V or 5V
3	Input Voltage (recommended)	7-24V	7-24V	7-24V	7-24V
4	Digital I/O Pins	14 (of which 6 provide PWM output)	14 (of which 6 provide PWM output)	14 (of which 6 provide PWM output)	14 (of which 6 provide PWM output)
5	Analog Input Pins	8 (A6/A7 in Extend)	8 (A6/A7 in Extend)	8 (A6/A7 in Extend)	8 (A6/A7 in Extend)
6	ADC Resolutions	10 bit	10 bit	10/12/16 bit	10/12/16 bit
7	Flash Memory	32 KB of which 0.5 KB used by bootloader	8 KB of which 1 KB used by bootloader	32 KB of which 1 KB used by bootloader	32 KB of which 1 KB used by bootloader
8	SRAM	2KB	1KB	2KB	2KB
9	EEPROM	1KB	504B	1KB Share with Flash Memory	1KB Share with Flash Memory
10	Clock Speed	16MHz	16MHz	16MHz	16MHz
11	Interface	Micro USB (CH341)	6Pin UART Serial Light	Micro USB (CH341)	6Pin UART Serial Light
12	Picture				
13	Main Advantage	USB UART interface ATmega328P Chipset Good cost performance Line Regulator, low noise	6Pin USB2Serial Light MD-8088 Chipset Very Good cost performance Line Regulator, low noise WILL BE PHASE OUT SOON, NOT RECOMMEND!	USB UART interface MD328D Chipset 10/12/16 bit ADC Very Good cost performance USB UART interface Ready for mass production	6Pin USB2Serial Light MD328D Chipset 10/12/16 bit ADC Very Good cost performance USB UART interface Ready for mass production

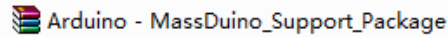
● How to use

We released an Arduino 3rd-party hardware package for MassDuino , so you can download it from website (www.inhaos.com) before using it , and put it in the appropriate location , then you can use it like to use any other Arduino board. The installation process is as follows.

step1: Download the Arduino software from the official website and then install it on the computer.
 (Surport Arduino 1.5.X and Arduino 1.6.x)



step2: Download the MassDuino surport package



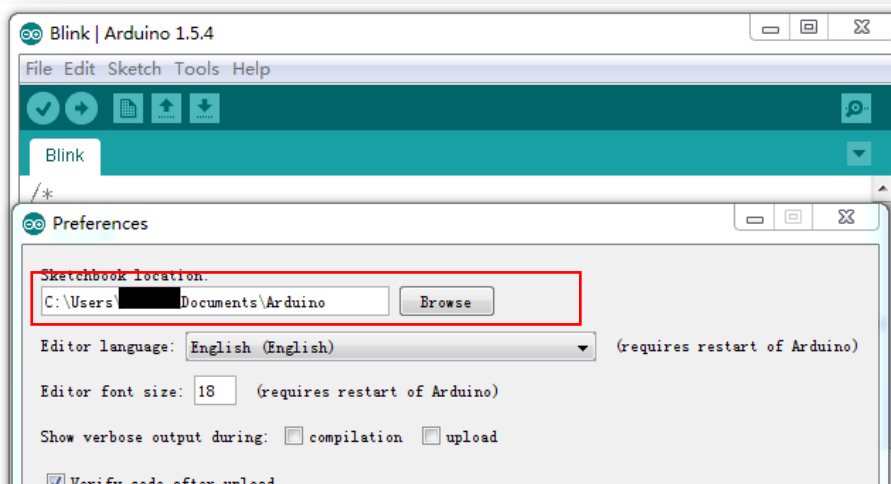
step3: Install MassDuino support package to Arduino IDE

PS: Before do this, please make sure your Arduino IDE is closed.

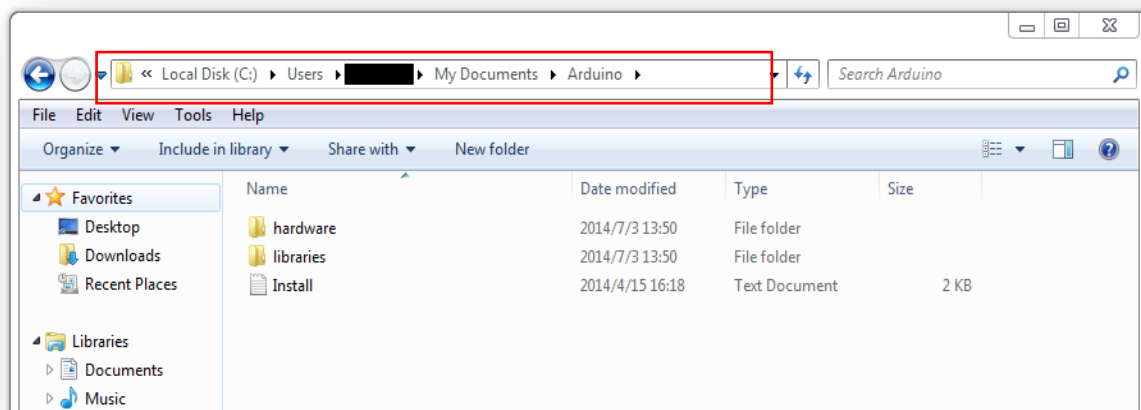
Unzip the support package file, and move the two Folders (libraries and hardware) to:

C:\Users\<USERNAME>\Documents\Arduino

You can check Arduino->File->Preferences to find your support file installation directory.

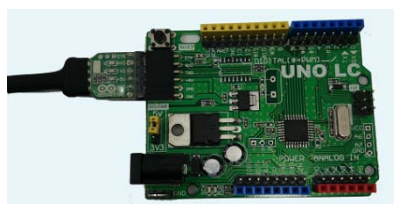


In my computer, the support package Installed here.



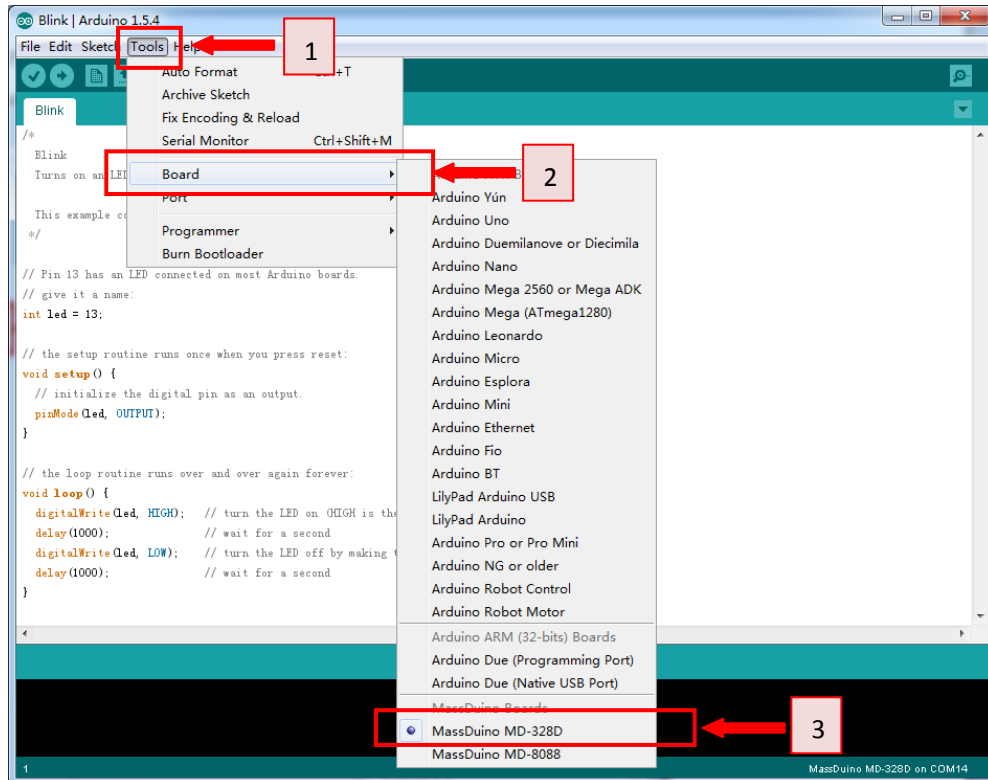
step4: Connect MassDuino board to your computer with a USB-Serial adapter and USB cable.

Used standard Arduino USB2Serial Light cable to upload sketch and communication.

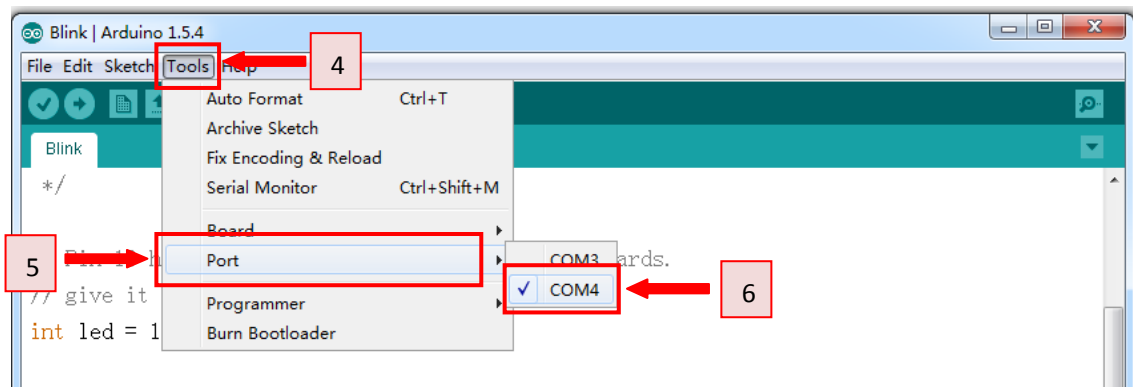


● My first MassDuino sketch

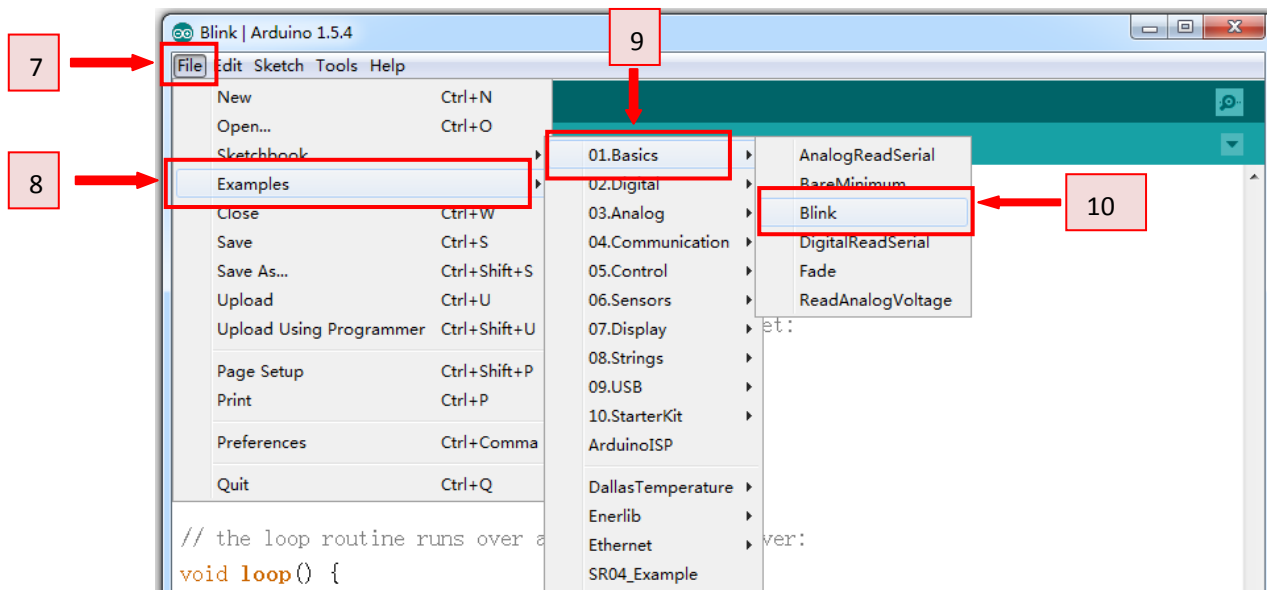
- 1) Open the Arduino IDE. Select the board: Click Tools -> Board -> MassDuino UNO R4.0



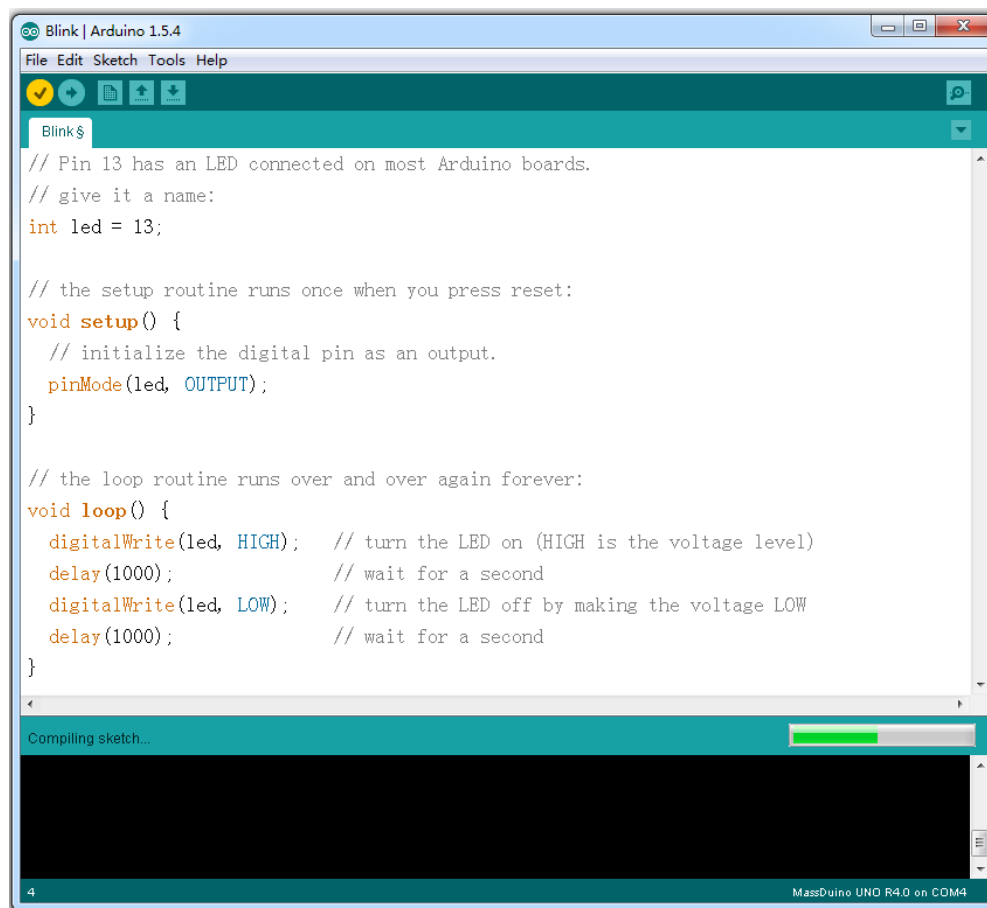
- 2) Select the COM: Click Tools -> Serial Port -> COM4(which connected with MassDuino.)



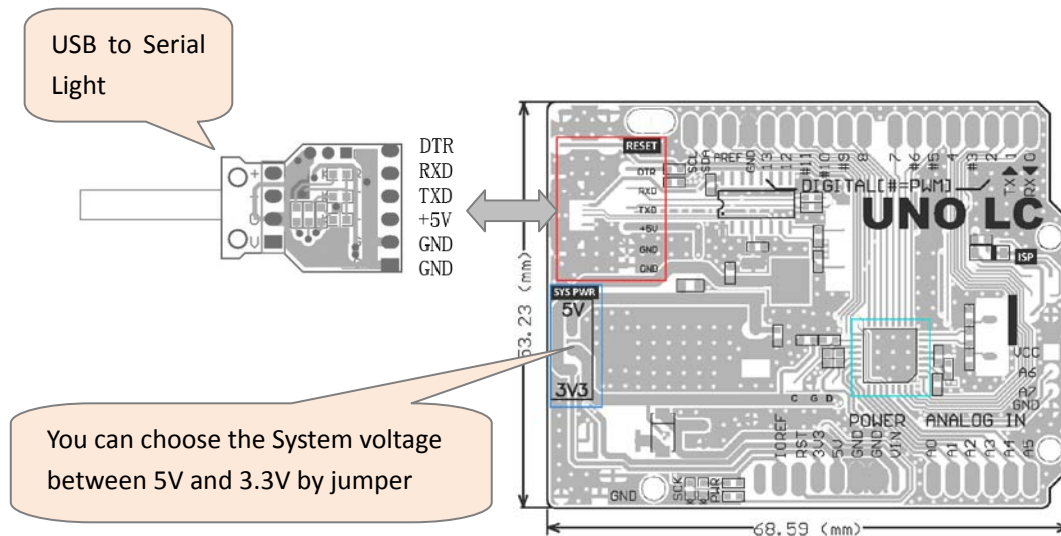
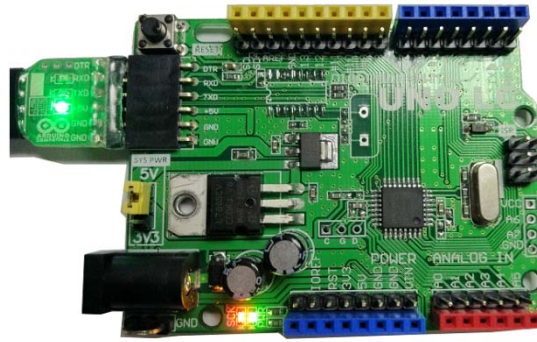
- 3) step6: An example of program: Click File -> Examples -> which you want.



4) step7: Upload the blink example to MassDuino.



5) Now you can see the LED is blinking according to the arduino code.



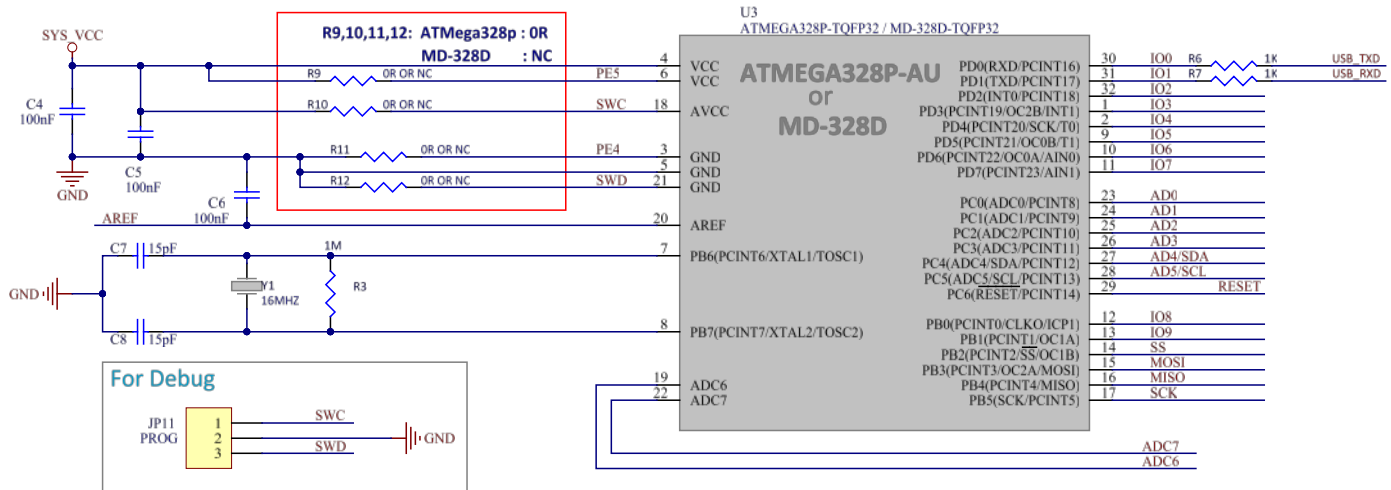
● About the analogRead speed for MassDuino MD-328D

The MassDuino MD-328D support 3 kinds of analogRead mode:

Mode	Resolution	Function	Time takes
1	10	analogRead()	300us
2	12	analogRead_12bits()	768us
3	16	analogRead_16bits()	8ms

So the higher resolution it outputs the longer time it takes, please select the appropriate output resolution according to your specific application.

● How to use MD-328D to replaced ATmega328P in Arduino system



The schematic showed the hardware difference between MD-328D and ATmega328P, here only 4pins need to difference connection:

No.	Pin Number	ATmega328P pin Function	MD-328D pin Function	Remark
1	3	GND	PE4	R11
2	6	VCC	PE5	R9
3	18	AVCC	SWC	R10
4	21	GND	SWD	R12

INHAOS Arduino UNO LC design support both ATMEGA328P and MD-328D , the difference of them is the resistor , R9,R10,R11,R12, if user want use MD-328D to design his own project , just reference the schematic show above.

INHAOS also sell MD-328D with very good price, please contact : support@inhaos.com

- **MD-3248P --- new chip released on 2018**

We release two new chip MD-3248P, below is the new feature of MD-3248P. to use MD-3248P, please go to www.inhaos.com download the newest version of Arduino support package , the MD-3248P will be supported start from V4.0.

Below are some feature function which supported for MD-3248P

1. digitalToggle()

Description

This function is used to toggle the specified pin.

Syntax

digitalToggle(pin)

Parameters

pin: the number

Returns

Nothing

Example Code

```
void setup() {  
    pinMode ( 13 , OUTPUT ) ;  
}  
  
void loop() {  
    digitalToggle(13);           // Toggle the pin status  
    delay(1000);                 // waits for a second  
}
```

2. fastioMode()

fastioToggle()

fastioWrite()

fastioRead()

Description

This function is used to fast operation the specified pin.

Syntax

```
fastioMode( pin , dir )           // like pinMode( pin ) function
fastioToggle ( pin )              // toggle the pin status
fastioWrite ( pin , value )       // like digitalWrite( pin ) function
fastioRead ( pin )                // like digitalRead( pin ) function
```

Parameters

```
fastioMode( pin , dir )           // like pinMode( pin ) function
    pin: the pin number
    dir: INPUT or OUTPUT
fastioToggle( pin )              // toggle the pin status
    pin: the pin number
fastioWrite( pin , value )       // like digitalWrite( pin ) function
    pin: the pin number
    dir: HIGH or LOW
fastioRead( pin )                // like digitalRead( pin ) function
    pin: the pin number
```

Returns

```
fastioMode( pin , dir )           // like pinMode( pin ) function
    nothing
fastioToggle( pin )              // toggle the pin status
    nothing
fastioWrite( pin , value )       // like digitalWrite( pin ) function
    nothing
fastioRead( pin )                // like digitalRead( pin ) function
    HIGH or LOW
```

Example Code

```
void setup() {
    fastioMode ( 13 , OUTPUT ) ;
}

void loop() {
    fastioToggle( 13 );           // Toggle the pin status
}
```

3. sysClock ()

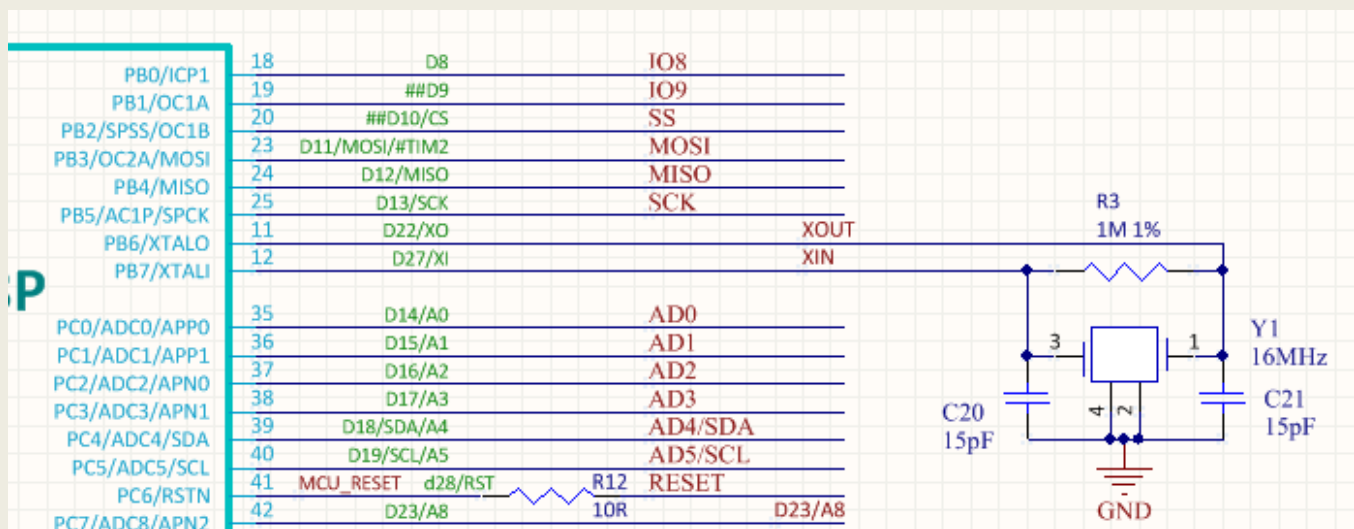
Description

This function is used to switch clock source between internal or external.

The MD-328D and MD-3248P have build in 16MHz RC clock source , the MCU will be default to use internal clock source , in some usage , user need to switch to extnal clock source , use this function to switch clock source to extnal source.

Notice: the Ext OSC driver using IO PB6 and PB7 , the PB6 is mapping to Arduino digital IO D22 , and PB7 is mapping to Arduino digital IO D27.

Both D22 and D27 only available with INT_OSC, **if you using EXT_OSC , do not operation D22 and D27 , otherwise it will switch the system click source to INT_OSC.**



Syntax

sysClock (clk_Source)

Parameters

clk_Source: INT_OSC or EXT_OSC

Returns

Nothing

Example Code

```
void setup() {
    sysClock(EXT_OSC);
    pinMode (13, OUTPUT);
}

void loop() {
    digitalToggle(13);           // Toggle the pin status
    delay(1000);                 // waits for a second
}
```

4. getBootResetFlag ()

Description

Get system reset source.

Syntax

getBootResetFlag()

Parameters

Nothing

Returns

0x 07 : Power on Reset

0x 08 : Watchdog Reset

notice: when press Hardware reset pin to reset , the MCU will entry bootloader code area and will jump to application code area by WDT reset , so in this case it also will get WDT reset.

Example Code

```
void setup() {  
    Serial.begin(115200);  
    Serial.print("initd,");  
    Serial.print("resetFlag:");  
    Serial.println(getBootResetFlag(), HEX); //Get reset flag.  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
}
```

5. wdt_enable ()

wdt_disable()

wdt_reset ()

Description

In default setting of Arduino , the WDT (Watchdog) was disable , you can enable watchdog to make your code much stable , use this function to enable the WDT.

Notice: you must put wdt_reset in appropriate location to ensure the WDT will not be overflow in normal case. normally we put it in main loop.

Syntax

wdt_enable (timeout) // enable WDT , the max count is define by timeout parmeter

wdt_disable () // disable WDT

wdt_reset () // Reset the WDT count , must add in main loop if you enable the WDT.

Parameters

timeout: unit: mS , **MD-3248P** support below timeout:

WTO_64MS , WTO_128MS , WTO_256MS , WTO_512MS , WTO_1S , WTO_2S , WTO_4S , WTO_8S , WTO_16S , WTO_32S

MD-328D support below timeout:

WTO_1MS , WTO_2MS , WTO_4MS , WTO_8MS , WTO_16MS , WTO_64MS , WTO_128MS , WTO_256MS , WTO_512MS

Returns

Nothing

Example Code

```
#include <WDT.h>

void setup() {
    // put your setup code here, to run once:
    pinMode(13, OUTPUT);
    wdt_enable(WTO_256MS);
}

void loop() {
    // put your main code here, to run repeatedly:
    //wdt_reset();
    //wdt_disable();
    digitalToggle(13);
    delay(10);
}
```

6. analogRead ()

analogRead_11bits ()
analogRead_12bits ()
analogRead_13bits ()
analogRead_14bits ()
analogRead_15bits ()
analogRead_16bits ()

Description

Massduino support 10~16 bit ADC resolutions.

Syntax

```
#include <analogFuncs.h>
analogRead( pin )
```

Parameters

pin: the number of the analog input pin.

MD-3248P have 12ch ADC, the digital pin number and Analog pin number is below:

A0 = D14 , **A1** = D15 , **A2** = D16 , **A3** = D17 , **A4** = D18 , **A5** = D19 , **A6** = D20 , **A7** = D21
A8 = D23 , **A9** = D24 , **A10** = D25 , **A11** = D26

MD-328D have 8ch ADC, the digital pin number and Analog pin number is below:

A0 = D14 , **A1** = D15 , **A2** = D16 , **A3** = D17 , **A4** = D18 , **A5** = D19 , **A6** = D20 , **A7** = D21

Returns

10/12/16 bit ADC value.

Sample Rate:

Mode	Resolution	Function	Time takes	Sample (SPS)
1	10	analogRead()	240us	4.174 Ksps
2	12	analogRead_12bits()	340us	2.941 Ksps
3	16	analogRead_16bits()	3.1ms	322.6 sps

Example Code

```
#include <analogFuncs.h>
unsigned int ADValue = 0 ;
double Voltage = 0 ;

void setup() {
    analogReference ( EXTERNAL ) ; // External reference source = 4.096V
    Serial.begin(115200);
```



```
}

void loop() {
    // 10bit read
    ADValue = analogRead (A0) ;
    Voltage = (double)ADValue * 4 ;
    Serial.print ( Voltage );
    Serial.print ( ", " );

    // 12bit read
    ADValue = analogRead_12bits (A0) ;
    Voltage = (double)ADValue ;
    Serial.print ( Voltage );
    Serial.print ( ", " );

    // 16bit read
    ADValue = analogRead_16bits (A0) ;
    Voltage = (double)ADValue / 16 ;
    Serial.println( Voltage );

    delay(50);
}
```

7. analogReference ()

Description

Configures the reference voltage used for analog input and analog output.

Syntax

analogReference(type)

Parameters

type: which type of reference to use.

MD-3248P support reference type below:

EXTERNAL: the voltage applied to the AREF pin is used as the reference

DEFAULT: the default analog reference is the voltage of AVCC , for Massduino UNO board , the AVCC is follow the VCC_SYS , which can be switch between 3.3V and 5V by a jumper.

INTERNAL2V048: a built-in 2.048V reference.

INTERNAL4V096: a built-in 4.096V reference.

MD-328D support reference type below:

EXTERNAL: the voltage applied to the AREF pin is used as the reference

DEFAULT: the default analog reference is the voltage of AVCC , for Massduino UNO board , the AVCC is follow the VCC_SYS , which can be switch between 3.3V and 5V by a jumper.

INTERNAL2V56: a built-in 2.56V reference.

Returns

nothing.

Example Code

```
double Voltage = 0 ;

void setup() {
    analogReference ( DEFAULT ) ;
    Serial.begin(115200);
}

void loop() {
    // 12bit read
    Voltage = (double)analogRead_12bits (A0) ;
    Serial.println ( Voltage );

    delay(100);
}
```

8. pwmMode ()

pwmResolution ()

pwmFrequency ()

pwmWrite ()

pwmTurnOff ()

Description

The PWM signal is very useful function , PWM pin will generate a steady square wave of the specified duty cycle , Arduino used analogWrite() to write an analog value to a pin , but the resolution and frequency can not be setting . now we provide ways to configure the PWM frequency and resolution.

MD-3248P have 4 timers (MD-328D and ATmega328P have only 3 timers).

Timer0: 8bit , used for general timekeeping (delay() , millis() ... etc)

Timer1: 16bit

Timer2: 8bit , used for some system function (Tone () ... etc)

Timer3: 16bit

The corresponding PWM pin is as follows:

Timer0: **OC0A:** D6,D31 , **OC0B:** D5,D35

Timer1: **OC1A:** D9,D37 , **OC1B:** D10,D36

Timer2: **OC2A:** D11,D38 , **OC2B:** D3,D39

Timer3: **OC3A:** D33 , **OC3B:** D34 , **OC3B:** D35 (Multiplexed with OC0B)

Those function only support to Timer1 and Timer3, There are pins related to D9,D37,D10,D36,D33,D35,D35.

Each timer mapped two digital pins , in the same time , you can only use one of them , *for example:*

OC0A mapped to **D6** and **D31** share **OC0A** ,so you can not use D6 and D31 as PWM at the same time.

We also provide MsTimer1 and MsTimer3 function to used the Timer1 and Timer3 programming.

Syntax

```
#include <pwmFuncs.h>
pwmMode (pin , wmode ,fmode , dband)
pwmResolution (pin , resBits )
pwmFrequency (pin , fhz)
pwmWrite (pin , value )
pwmTurnOff (pin)
```

Parameters

pwmMode (pin , wmode ,fmode , dband)

pin : the pin to write to.

wmode : PWM_MODE_NORMAL

fmode : **PWM_FREQ_BOOST** : enable frequency boost x4 mode

PWM_FREQ_FAST: freq prescale = 1 (fast mode)

PWM_FREQ_NORMAL: freq prescale = 64 (default)

PWM_FREQ_SLOW : freq prescale = 1024 (slow mode)

dband: 0

`pwmResolution (pin , resBits)`

pin : the pin to write to.

resBits : pwm resolution , value in 1 to 16.

Notice , the difference resolution setting support difference frequency , this function will be return a frequency show what frequency will be used with this resolution.

`pwmFrequency (pin , fhz)`

pin : the pin to write to.

fhz : pwm frequency

Notice , the difference resolution setting support difference frequency , this function will be return a frequency show what resolution will be used with this frequency.

`pwmWrite (pin , value)`

pin : the pin to write to.

value : the duty cycle between 0 to the resolutions.

`pwmTurnOff (pin)`

pin : turn off the PWM function , and then operation as GPIO..

To avoid frequency and resolution setting is not match , so please do not configure the frequency and resolution at the same time.

Returns

`pwmMode (pin , wmode , fmode , dband)`

nothing

`pwmResolution (pin , resBits)`

The frequency which match with the specified resolution.

`pwmFrequency (pin , fhz)`

The resolution which match with the specified frequency.

`pwmWrite (pin , value)`

nothing

`pwmTurnOff (pin)`

nothing

Remark

/* the pwm freq and max duty table: with setting:

`pwmMode(PWM_PIN, PWM_MODE_NORMAL, PWM_FREQ_FAST, 0);`

freq:100, maxDuty:65535	freq:200, maxDuty:65535	freq:300, maxDuty:53333	freq:400, maxDuty:40000
freq:500, maxDuty:32000	freq:600, maxDuty:26666	freq:700, maxDuty:22857	freq:800, maxDuty:20000

```

freq:900, maxDuty:17777   freq:1000, maxDuty:16000   freq:1100, maxDuty:14545   freq:1200, maxDuty:13333
freq:1300, maxDuty:12307   freq:1400, maxDuty:11428   freq:1500, maxDuty:10666   freq:1600, maxDuty:10000
freq:1700, maxDuty:9411    freq:1800, maxDuty:8888    freq:1900, maxDuty:8421    freq:2000, maxDuty:8000
freq:2100, maxDuty:7619    freq:2200, maxDuty:7272    freq:2300, maxDuty:6956    freq:2400, maxDuty:6666
freq:2500, maxDuty:6400    freq:2600, maxDuty:6153    freq:2700, maxDuty:5925    freq:2800, maxDuty:5714
freq:2900, maxDuty:5517    freq:3000, maxDuty:5333    freq:3100, maxDuty:5161    freq:3200, maxDuty:5000
freq:3300, maxDuty:4848    freq:3400, maxDuty:4705    freq:3500, maxDuty:4571    freq:3600, maxDuty:4444
freq:3700, maxDuty:4324    freq:3800, maxDuty:4210    freq:3900, maxDuty:4102    freq:4000, maxDuty:4000
freq:4100, maxDuty:3902    freq:4200, maxDuty:3809    freq:4300, maxDuty:3720    freq:4400, maxDuty:3636
freq:4500, maxDuty:3555    freq:4600, maxDuty:3478    freq:4700, maxDuty:3404    freq:4800, maxDuty:3333
freq:4900, maxDuty:3265    freq:5000, maxDuty:3200    freq:5100, maxDuty:3137    freq:5200, maxDuty:3076
freq:5300, maxDuty:3018    freq:5400, maxDuty:2962    freq:5500, maxDuty:2909    freq:5600, maxDuty:2857
freq:5700, maxDuty:2807    freq:5800, maxDuty:2758    freq:5900, maxDuty:2711    freq:6000, maxDuty:2666
freq:6100, maxDuty:2622    freq:6200, maxDuty:2580    freq:6300, maxDuty:2539    freq:6400, maxDuty:2500
freq:6500, maxDuty:2461    freq:6600, maxDuty:2424    freq:6700, maxDuty:2388    freq:6800, maxDuty:2352
freq:6900, maxDuty:2318    freq:7000, maxDuty:2285    freq:7100, maxDuty:2253    freq:7200, maxDuty:2222
freq:7300, maxDuty:2191    freq:7400, maxDuty:2162    freq:7500, maxDuty:2133    freq:7600, maxDuty:2105
freq:7700, maxDuty:2077    freq:7800, maxDuty:2051    freq:7900, maxDuty:2025    freq:8000, maxDuty:2000
freq:8100, maxDuty:1975    freq:8200, maxDuty:1951    freq:8300, maxDuty:1927    freq:8400, maxDuty:1904
freq:8500, maxDuty:1882    freq:8600, maxDuty:1860    freq:8700, maxDuty:1839    freq:8800, maxDuty:1818
freq:8900, maxDuty:1797    freq:9000, maxDuty:1777    freq:9100, maxDuty:1758    freq:9200, maxDuty:1739
freq:9300, maxDuty:1720    freq:9400, maxDuty:1702    freq:9500, maxDuty:1684    freq:9600, maxDuty:1666
freq:9700, maxDuty:1649    freq:9800, maxDuty:1632    freq:9900, maxDuty:1616    freq:10000, maxDuty:1600
*/

```

Example Code

```

#include <pwmFuncs.h>
#define PWM_PIN      9

void setup() {
    pwmMode(PWM_PIN, PWM_MODE_NORMAL, PWM_FREQ_FAST, 0);
    pwmResolution(PWM_PIN, 16);
}

void loop() {
    for ( unsigned long i = 0 ; i <= 60000u ; i += 1000 ) {
        pwmWrite ( PWM_PIN, i ) ;
        delay (5);
    }
    pwmTurnOff ( PWM_PIN ); // trun off the PWM function
    delay(100);
    digitalWrite ( PWM_PIN , HIGH ) ;// Control as a digital IO
    delay(50);
    digitalWrite ( PWM_PIN , LOW ) ; // Control as a digital IO
    delay(50);
}

```

9. MsTimer1.setMicros ()

MsTimer1.set ()

MsTimer1.start()

MsTimer1.stop()

MsTimer3. setMicros ()

MsTimer3.set ()

MsTimer3.start()

MsTimer3.stop()

Description

MD-3248P have 4 timers (MD-328D and ATmega328P have only 3 timers).

Timer0: 8bit , used for some system function (delay() , millis() ... etc)

Timer1: 16bit

Timer2: 8bit , used for some system function (Tone () ... etc)

Timer3: 16bit

Those function will help user to use Timer1 and Timer3.

Syntax

MsTimer1 setMicros (unsigned long uS, void (*f)())

MsTimer1. set (unsigned long mS, void (*f)())

MsTimer3. setMicros (unsigned long uS, void (*f)())

MsTimer3. set (unsigned long mS, void (*f)())

MsTimer1.start () // start the timer1

MsTimer3.start () // start the timer3

MsTimer1.stop () // stop the timer1

MsTimer3.stop () // stop the timer3

Parameters

type:

unsigned long uS: the time on uS for the overflow

f(): "f" is the function which will be called when each overflow. "f" has to be declared void with no parameters.

Notice: with MsTimer.SetMicros() function , the overflow range must limit to 10uS to 4095uS.

Returns

nothing.

Example Code

```
#include <MsTimer1.h>
#include <MsTimer2.h>
#include <MsTimer3.h>

void D13_flash() {
    fastioToggle(13);
}

void D11_flash() {
    fastioToggle(11);
}

void D10_flash() {
    fastioToggle(10);
}

void setup() {
    fastioMode(13, OUTPUT);
    fastioMode(11, OUTPUT);
    fastioMode(10, OUTPUT);
    fastioMode ( 5 , INPUT );
    fastioWrite ( 5 , HIGH ) ;
    fastioMode ( 7 , INPUT );
    fastioWrite ( 7 , HIGH ) ;
    fastioMode ( 6 , INPUT );
    fastioWrite ( 6 , HIGH ) ;
    fastioMode ( 7 , INPUT );
    fastioWrite ( 7 , HIGH ) ;

    //MsTimer1.set(20, flash); // 20ms period
    MsTimer1.setMicros(30, D13_flash); //30us
    MsTimer2::set(1, D10_flash); //1mS
    MsTimer3.setMicros(20, D11_flash); //20us
    MsTimer1.start();
    MsTimer2::start();
    MsTimer3.start();
}

void loop() {
    if ( ! fastioRead ( 5 )) MsTimer1.stop();
    if ( ! fastioRead ( 6 )) MsTimer2::stop();
    if ( ! fastioRead ( 7 )) MsTimer3.stop();
}
```

10. DAC usage

Description

MD-3248P have 1CH 8bit DAC, to use this DAC , user can be do below step:

S1: set analog reference : `analogReference (DEFAULT);`

S2: Set DAC output pin mode: `pinMode (4 , ANALOG);`

S3: write DAC value by function : `analogWrite (4 , 100);`

Syntax

Parameters

Returns

nothing.

Notice: Limited by the cost , the DAC's resolution and linearity is not so good , for high performance request application , user can be use PWM + LPF to get very good DAC function.

Example Code

```
void setup() {  
    analogReference ( EXTERNAL ) ;  
    pinMode( 4 , ANALOG );  
}  
  
void loop() {  
    for ( int i = 0 ; i <= 255 ; i++ ) {  
        analogWrite( 4 , i );  
        delay( 20 );  
    }  
}
```


11. PMU (Power Management Unit)

Description

MD-3248P provided rich power management function , this is very good in low power application (eg. battery powered application.)

To use PMU function , you need to add headed file <PMU.h>

Syntax

PMU.sleep(mode , period);

Parameters

mode:

PM_IDLE: IDLE mode, turn off the core clock, reduced power consumption is very small.

PM_POWERDOWN: turn off all clock , wakeup source : pin change int , external int , WDT overflow , TMR2 overflow

PM_POFFS0: PowerOff mode 0 , wake up source : wakeup source : pin change int , external int , WDT overflow , TMR2 overflow

PM_POFFS1: PowerOff mode 1 , wake up source : wakeup source : pin change int , external int , TMR2 overflow

period:

SLEEP_64MS

SLEEP_64MS = 0

SLEEP_128MS

SLEEP_256MS

SLEEP_512MS

SLEEP_1S

SLEEP_2S

SLEEP_4S

SLEEP_8S

SLEEP_16S

SLEEP_32S

SLEEP_FOREVER

Returns

nothing.

Example Code

```
#include <PMU.h>

volatile uint8_t g_LastState;
volatile uint8_t g_IntFlag;

void setup()
{
    Serial.begin(9600);
    pinMode(2, INPUT);
    digitalWrite(2, HIGH);
    attachInterrupt(0, donothing, FALLING);
    g_LastState = digitalRead(2);
}
```

```
        g_IntFlag = 0;
    }

    void loop()
    {
        Serial.write("hello, I am working!\n");

        // power/down mode
        if (g_LastState == HIGH)
        {
            Serial.println("Change to falling");
            attachInterrupt(0, donothing, FALLING);
        }
        else
        {
            Serial.println("Change to rising");
            attachInterrupt(0, donothing, RISING);
        }

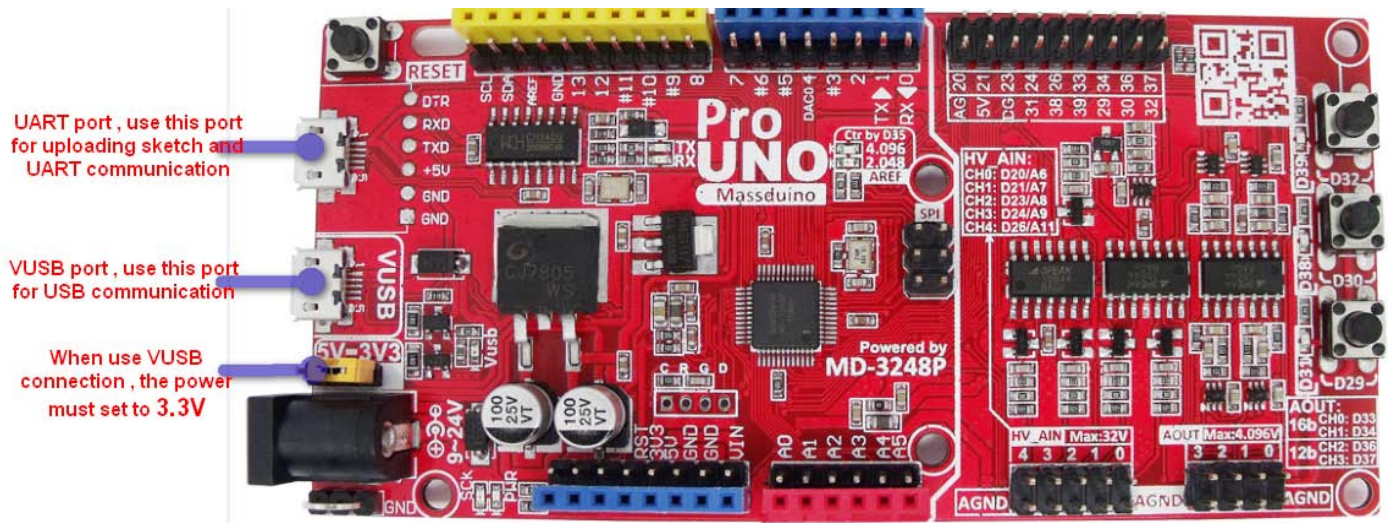
        Serial.flush();

        PMU.sleep(PM_POFFS1, SLEEP_4S);
        g_LastState = digitalRead(2);
    }

    void donothing()
    {
        //g_IntFlag = 1;
    }
```

12. Virtual USB HID sample

UNO Pro have onboard VUSB interface , user can be programming UNO Pro as an USB HID device such as keyboard / mice / etc...



We provide below sample code for fast VUSB coding.

VUsbAllInOne:

Implement USB keyboard / Mouse / HID communication

VUsbDatInOut:

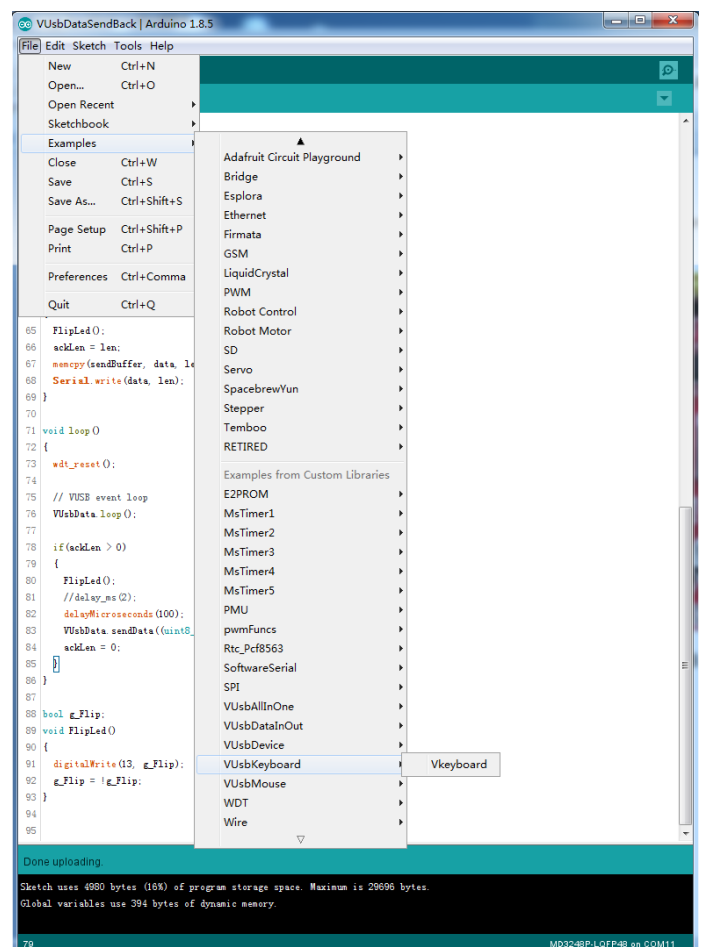
Implement USB HID communication

VUsbKeyboard:

Implement an USB keyboard

VUsbMouse:

Implement a USB mouse

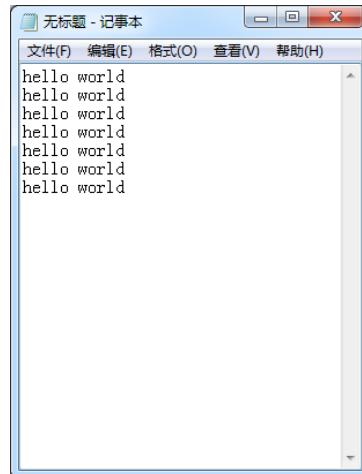


Note , to use VUSB communication , you need to set the power jumper to 3.3V.

VUsbAllInOne:

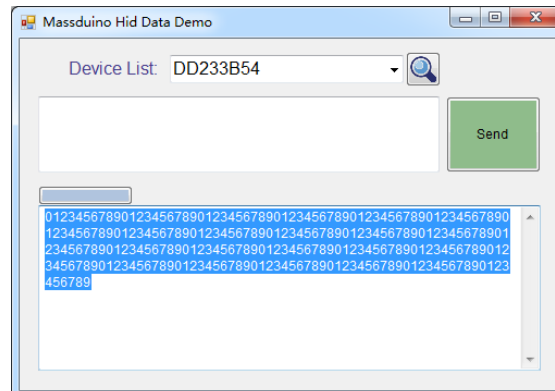
This code implement USB keyboard / Mouse / HID communication, to use this code with below step:

- 1, Connected UART port to PC , and upload the sketch.
- 2, Change USB cable connected to VUSB port.
- 3, Open Notepad or text editor, move cursor to edit area.
- 4, press and hold "D30" button , you will see the text input on the app.

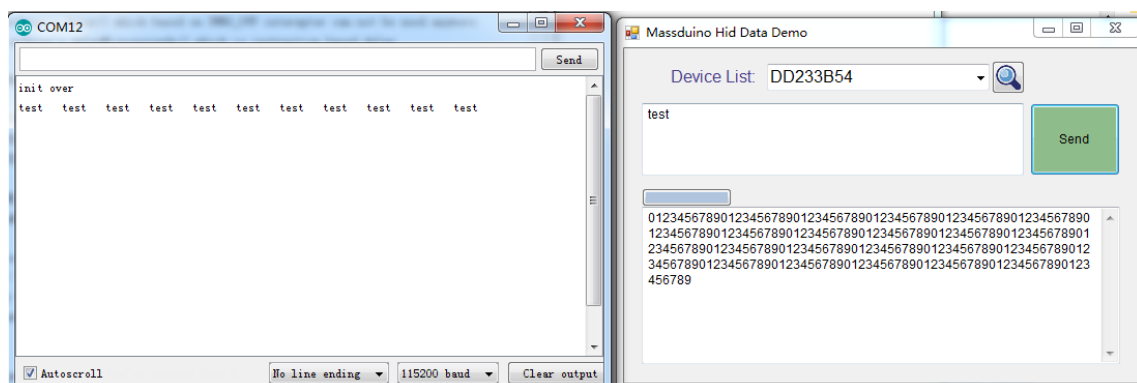


- 5, press and hold “D32” button , you will see the cursor draw circle on the screen.
- 6, Open “MassduinoHidDataDemo.exe” which attached in the Arduino - Massduino_Support_Package_V46r1 .
- 7, press “D29” button , you will see the data which the UNO Pro sent to PC via USB port .

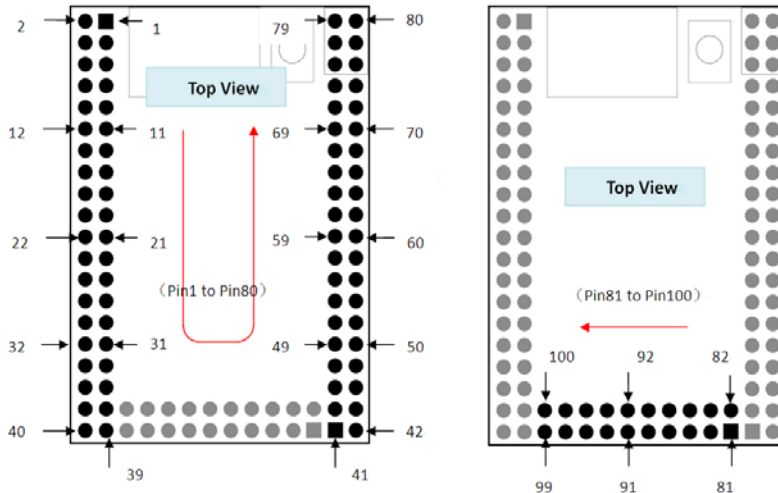
Be notice that HID communication support half-duplex only which means you need to make a protocol to avoid host and device sent data at the same time.



- 8, Connected both UART and VUSB port , open “Serial Monitor” in Arduino IDE , then you send data and it will sent back to PC via UARY port.



● Mega2560-CORE



Mega2560 Pin Mapping

Mega2560-CORE module

(http://www.inhaos.com/product_info.php?products_id=118) has sink all pin to the DIP 2.54mm pin header (except Pin-4 'PE4'), in original Arduino Mega2560 library , not all pin assigned digital pin number , Meg2560-CORE has patched it.

Pin Description Table (Support on Arduino - Massduino_Support_Package_V50)

Pin	Function /MCU	Arduino Mega2560	Pin	Function /MCU	Arduino Mega2560	Pin	Function /MCU	Arduino Mega2560
1	VCC		40	GND		79	PB3	D50
2	VCC		41	AGND		80	VCC	
3	PG5	D4	42	AGND		81	VCC	
4	PE0	RXD	43	ADC0	A0/D54	82	VCC	
5	PE1	TXD	44	VREF		83	PC7	D30
6	PE3	PWM/D5	45	ADC2	A2/D56	84	PC6	D31
7	PE4	PWM/D2	46	ADC1	A1/D55	85	PC5	D32
8	PE5	PWM/D3	47	ADC4	A4/D58	86	PC4	D33
9	PE6	D74	48	ADC3	A3/D57	87	PC3	D34
10	PE7	D75	49	ADC6	A6/D60	88	PC2	D35
11	TXD3	D14	50	ADC5	A5/D59	89	PC1	D36
12	RXD3	D15	51	ADC8	A8/D62	90	PC0	D37
13	TXD2	D16	52	ADC7	A7/D61	91	PG1	D40
14	RXD2	D17	53	ADC10	A10/D64	92	PG0	D41
15	TXD1	D18	54	ADC9	A9/D63	93	PD7	D38
16	RXD1	D19	55	ADC12	A12/D66	94	PD6	D72
17	SDA	D20	56	ADC11	A11/D65	95	PD5	D71
18	SCL	D21	57	ADC14	A14/D68	96	PD4	D70
19	PH2	D78	58	ADC13	A13/D67	97	PG4	D77
20	PH3	PWM/D6	59	PJ2	D80	98	reserv	
21	PH4	PWM/D7	60	ADC15	A15/D69	99	GND	
22	PH5	PWM/D8	61	PJ4	D82	100	GND	
23	PH6	PWM/D9	62	PJ3	D81			
24	PB0	53	63	PJ6	D84			
25	PB4	PWM/D10	64	PJ5	D83			
26	PB5	PWM/D11	65	PA7	D29			
27	PB6	PWM/D12	66	PG2	D39			
28	PB7	PWM/D13	67	PA5	D27			
29	PL0	D49	68	PA6	D28			
30	PL1	D48	69	PA3	D25			
31	PL2	D47	70	PA4	D26			
32	PL3	PWM/D46	71	PA1	D23			
33	PL4	PWM/D45	72	PA2	D24			
34	PL5	PWM/D44	73	PJ7	D85			
35	PL6	43	74	PA0	D22			
36	PL7	42	75	RESET				
37	PH7	D79	76	GND				
38	PG3	D76	77	PB1	D52			
39	GND		78	PB2	D51			

INHAOS Headquarter :

1111 Oakmont Drive #C, San Jose, CA 95117

E-mail : support@inhaos.com

INHAOS China office :

No.6 Building,Songke Estate,Songshan Lake National Hi-tech Industrial
Development Zone,Dongguan,Guangdong Province, 523808,China

E-mail: Support@inhaos.com