AP01A AVR MCU

1. Description

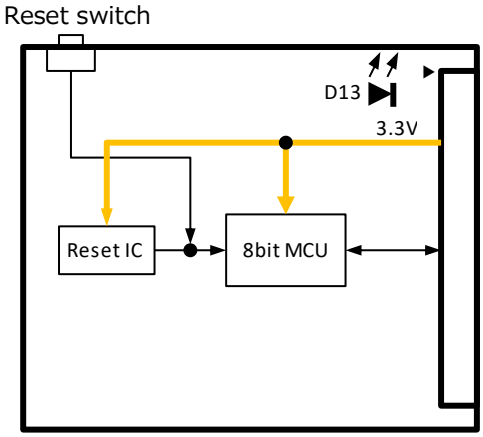
The leaf which is using ATmega328P. It has 14 digital output pins (among them, 6 pins could be PWM output), 6 anlog input pins, 8MHz oscillator, and a reset bottom.

Connect USB Cable when use USB-UART、Connect with Shield when use ICSP

Specify board as Arudino Pro or Pro mini, processor as ATmega328P(3.3V,8MHz) when using Arudino IDE.

2. Leaf specification

2-1. Block diagram



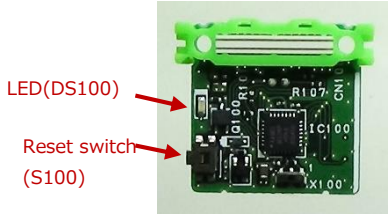
2-2. Power supply specification

Symbol	Parameter	Condition	Min.	Typ.	Max.
Vdd	Power Supply Voltage	–	1.9V	3.3V	5.5V
Idd	Operating current	Active	-	5.2mA	-
		Sleep	-	4.7uA	-

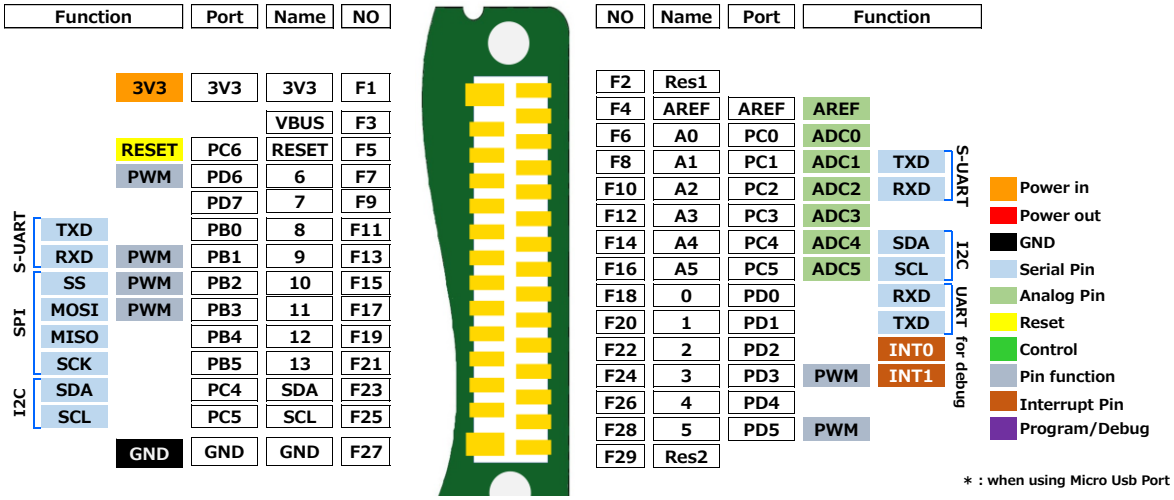
2-3. Main parts

Reference No.	Part name	Part number	Vendor name	note
IC100	AVR MCU	ATmega328P-MMH	Microchip	28pinQFN

2-4. Appearance

Top Side	Back Side
	

2-5. Pinout



2-6. LED/スイッチ

Item	Parts number	Description
LED	DS100	Control LED by using pin 13 (Same as Arduino UNO) Disable the lighting function by removing R105 (1kΩ)
Reset switch	S100	Reset microcontroller or other devices

3. 8bit MCU(ATmega382P-MMH) Specifications

3-1. Description

ItemItem	Description
Microcontroller	ATmega328P,28pin QFN
Operating Voltage	3.3V
Input Voltage	1.5-5 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
Flash Memory	32 KB
SRAM	2 KB
EEPROM	1 KB
Clock Speed	8 MHz
LED_BUILTIN	13
Compatibility	Arduino Pro/Pro mini ／ATmega328P（3.3V 8MHZ）

3-2. Electrical characteristics

3-2-1. Absolute Maximum Ratings

Parameter	Value
Operating Temperature	-55℃ to +125℃
Maximum Operation Voltage	6.0V

3-2-2. Rating (WDT＝Watch Dog Timer)

Symbol	Parameter	Condition	Min.	Typ.	Max.
Vdd	Power Supply Voltage	–	1.8V		5.5V
Idd	Active	1MHz, Vcc=2V		0.3mA	0.5mA
		4MHz, Vcc=3V		1.7mA	2.5mA
		8MHz, Vcc=5V		5.2mA	9mA
	Power-save	32KHz,Vcc=1.8V		0.8uA	
		32KHz,Vcc=3V		0.9uA	
	Power-down	WDT enabled, Vcc=3V		4.2uA	8uA
		WDT disabled, Vcc=3V		0.1uA	2uA

3-3. Link destination of data sheet

https://www.microchip.com/wwwproducts/en/atmega328p

3-4. Main functions and libraries

3-4-1. Digital input/output

Definition	Description
pinMode(pin ,mode)	Set pin as input or output pin. 【parameter】 pin: the pin number to be set mode: INPUT (internal pull-up resistor is ineffective) 、INPUT_PULLUP (internal pull-up resistor is effective)、OUTPUT 【return value】 null
digitalWrite(pin, value)	Send high or low to the specified pin. 【parameter】 pin: the pin number value: HIGH (3.3V) or LOW (0V) 【return value】 null
digitalRead(pin)	Read the value of the specified pin. 【parameter】 pin: the pin number to be read 【return value】 HIGH or LOW

3-4-2. Analog input

Definition	Description
analogRead(pin)	Read the value of the specified analog pin. 【parameter】 pin: the pin number to be read The assigned number is integer and the valid number is one of 0~5. 【return value】 0~1024 integer value

3-4-3. External interruption

Definition	Description
attachInterrupt (interrupt, function, mode)	Assign function which is goin to excute when interrupt. Interruption number and the corresponding pin number are showen as below pin2(int.0) pin3(int.1) 【parameter】 interrupt: interruption number function: the function called when interrupt mode: trigger of interruption LOW interrupt when pin is LOW CHANGE interrupt when pin changes RISING interrupt when pin is changing from low to high FALLING interrupt when pin is changing from high to low 【return value】 null
detachInterrupt (interrupt)	Disable interruption 【parameter】 null 【return value】 null

3-4-4. UART communication (USB-Serial transformation)

Definition	Description
Serial.begin(speed)	Assign the data transfer rate of Serial communication. Set to '115200' when connect to Arudino IDE 【parameter】 speed: transfer rate (int) 【return value】 null
Serial.end()	Terminate serial communication. 【parameter】 null 【return value】 null

Serial.read()	Read received data. 【parameter】 null 【return value】 Return the first 1 byte of received data Return '-1' when there is no received data
Serial.flush()	Wait for data transformation to complete. 【parameter】 null 【return value】 null
Serial.print(data, format)	Output data by using text format to serial port. To assign format by second parameter. 【statement】 Serial.print(data) Serial.print(data, format) 【parameter】 data: Output value. Every format is corresponded. format: radix or effective digit (floting number) 【return value】 transmitted number of bytes (byte)
Serial.println(data, format)	Attach 'CR' and 'LF' to the end of data and send it. Same format as Serial.print(). Refer to the description of Serial.print(). 【parameter】 data: integer type or string type format: Specify the way changing data (could be omitted) 【return value】 the number of bytes sent (byte)
Serial.write(val)	Send binary data to serial port. 【statement】 Serial.write(val) Serial.write(str) Serial.write(buf, len) 【parameter】 val: sending value (1 byte) str: character string (multiple byte) buf: Multiple byte defined as array len: length of array 【return value】 the number of bytes sent (byte)

3-4-5. Sofeware serial communication

include file:SoftwareSerial.h (Arduino IDE Standard Libraries)

Definition	Description

SoftwareSerial name(rxPin, txPin)	Enable Software Serial. The name of object must be gived. It is necessary for running SoftwareSerial.begin(). Multiple port can open at the same time but there is only one who can receive data. 【parameter】 rxPin: Pin receiving data txPin: Pin sending data
name.begin(speed)	Set communicating speed (baudrate) of serial. typical value is 9600. 【parameter】 speed: baudrate (long) 【return value】 null
name.read()	Return the received word. Only one SoftwareSerial is active at once. Use listen() to choose which one you need. 【parameter】 null 【return value】 The received word (-1 if there is no data)
name.print(data)	Output data to SoftwareSerial port. The same function as print(). 【parameter】 Refer to the description of Serial.print(). 【return value】 the number of bytes to send (byte)
name.println(data)	Output data to SoftwareSerial port. The same function as Serial.println(). 【parameter】 Refer to the description of Serial.println(). 【return value】 the number of bytes sent (byte)
name.listen()	Listen the assigned SoftwareSerial port. Only one port is active at once. 【parameter】 null 【return value】 null
name.write(val)	Output data to SoftwareSerial port. The same function as Serial.write() 【parameter】 Refer to the description of Serial.write(). 【return value】 the number of bytes sent (byte)

3-4-6. I2C communication

include file : Wire.h (Arduino IDE Standard Libraries)

Definition	Description
Wire.begin(address)	Initialize Wire library and connect to I2C bus in the role of master or slave. 【parameter】 address: I2C slave address (7 bytes). Connect in the role of master if omit it. 【return value】 null

Wire.requestFrom(address, count)	Request data from other devices. Get data by using read() function. 【parameter】 address: adress of the device requesting data (7 bits) quantity: byte number of requesting data stop (could be omitted): send stop message after request if the value is 'true'. send restart message after request if the value is 'false'. 【return value】 Return actually received bytes number.
Wire.beginTransmission(address)	Start transmission to specified I2C slave's address. 【parameter】 address: address of transmission target (7 bits) 【return value】 null
Wire.endTransmission()	End transmission to slave device. 【parameter】 stop (could be omitted): send stop message after request if the value is 'true'. (default) send restart message after request if the value is 'false'. 【return value】 transmission result (byte) 0: succesful 1: sending data size is grater than buffer size 2: Send slave's address and receive NACK 3: Send data byte and receive NACK 4: other error
Wire.write(value)	Send data. Excute between beginTransmission() and endTransmission() 【statement】 Wire.write(value) Wire.write(string) Wire.write(data, length) 【parameter】 value: 1 byte data to send (byte) string: character string (char *) data: array (byte *) length: the number of bytes to send (byte) 【return value】 the number of bytes sent (byte)
Wire.read()	Receive data. Read Data from slave device after master device executed requestFrom(). 【parameter】 null 【return value】 received data (byte)

3-4-7. Watchdog timer

include file : avr/wdt.h (Arduino IDE Standard Libraries)

Definition	Description
wdt_enable (value)	Enable watchdog timer. 15ms~8s to reset. 【parameter】 The time spent to reset 【return value】 null
wdt_reset()	Reset watchdog timer. 【parameter】 null 【return value】 null
wdt_disable ()	Disable watchdog timer. 【parameter】 null 【return value】 null

3-4-8. Sleep mode

include file : avr/sleep.h (Arduino IDE Standard Libraries)

Definition	Description
set_sleep_mode (parameter)	Set sleep mode. 【parameter】 parameter : SLEEP_MODE_PWR_DOWN SLEEP_MODE_PWR_SAVE SLEEP_MODE_STANDBY SLEEP_MODE_IDLE SLEEP_MODE_EXT_STANDBY 【return value】 null
sleep_enable()	Enable
sleep_mode()	Start sleep mode

3-4-9. Timer interruption

include file : MsTimer2.h (Contributed Libraries)

http://playground.arduino.cc/Main/MsTimer2

Definition	Description
MsTimer2::set (unsigned long ms, void (*f)())	Assign time (ms) of timer. Call function f every time when time is up. Decalre f as void type argument null. 【parameter】 time of overflow (ms) 【return value】 null

MsTimer2::start()	Enable timer interruption 【parameter】 null 【return value】 null
MsTimer2::stop()	Disable timer interruption 【parameter】 null 【return value】 null

3-5. Power saving control

3-5-1. Active mode and Sleep mode

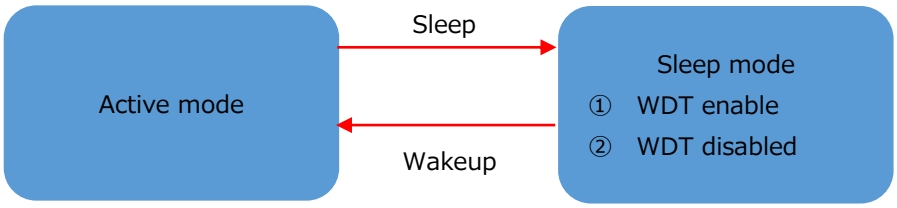
Active mode: Operating state. It usually has large current flow.

Sleep mode: Saving power state. Also called standby mode.

Wakeup : Convert from sleep mode to active mode. Wakeup has 2 types.

1)WDT (Watchdog timer) : MCU's timer will wakeup regularly (e.g. every few seconds). Set WDT 'enabled' before switch to sleep mode to make this function work.

2) External interruption: Wakeup if external interruption like the value of sensor changes or realtime clock, etc happens. Make MCU more saving power than WDT.



3-5-2. Sleep mode･WDT sample switch

The sample sketch which wakeup ever defined period of time by using library group of sleep function and WDT function. Sleep time is 8 seconds. Reset when return caused by WDT after using avr/wdt.h of wdt_enable() function. Write as below into register of ATmega328P when prohibit reset when wakeup.

```
#include <avr/wdt.h>
#include <avr/sleep.h>
void WDT_setup(){
        cli();                                  // Prohibit interruption
        wdt_reset();                            // Reset WDT timer counter
        MCUSR &= ~(1 << WDRF);                  // Reset watchDog system Reset Flag(WDRF)
        WDTCSR |= 1 << WDCE | 1 << WDE;         // Changing WDT is valid (By setting both WDCE and WDE as 1)
        WDTCSR = 1 << WDIE | 0 << WDE | 1 << WDP3 | 0 << WDP2 | 0 << WDP1 | 1 << WDP0; //WDT setting
                                                // WDE=0,WDIE=1 : interruption by WDT overflow
                                                // WDP3=1,WDP2=0,WDP1=0,WDP0=1: 8s
        sei();                                  //Permit interruption
}
void sleepMode(){
        Serial.println("Sleep Mode");
        delay(10);
        WDT_setup();
        set_sleep_mode(SLEEP_MODE_PWR_DOWN);        // Set sleep mode
        sleep_bod_disable();                        // Prohibit Low voltage detector (BOD)
        sleep_mode();                               // Switch to SLEEP
        __asm__("nop¥n¥t");
    Serial.println("WakeUp");
        digitalWrite(13,HIGH); delay(10);
        digitalWrite(13,LOW);
}
void setup(){
  Serial.begin(115200);
}
void loop() {
        sleepMode();
}
ISR(WDT_vect) {                                 // WDT will be run when time up
        wdt_disable();
}
```

3-5-3. Sleep mode

ATmega328P has Power-down mode, whitch is the most power saving sleep mode. How to write Power-down mode related sketch is explained as below.

The Required function when switch to sleep mode

set_sleep_mode(parameter) : Set Sleep mode (parameter)

sleep_mode() : Switch to Sleep mode

Call library

#include <avr/sleep.h>

Sleep mode parameter	Processing details
SLEEP_MODE_PWR_DOWN	Set as Power-Down mode

Example of switch

```
#include <avr/sleep.h>
void setup() { }
void sleep() {
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);
  ADCSRA &= ~(1 << ADEN);                                  //Stop ADC
  sleep_enable();
  MCUCR |= (1 << BODSE) | (1 << BODS);                     // Set MCUCR BODSand BODSE
  MCUCR = (MCUCR & ~(1 << BODSE)) | (1 << BODS);   // Set BODSSE as 0, BODS as 1 immidiately (no grater
than 4 clock)
  asm("sleep");                                            // Sleep no grater than 3 clock
}
```

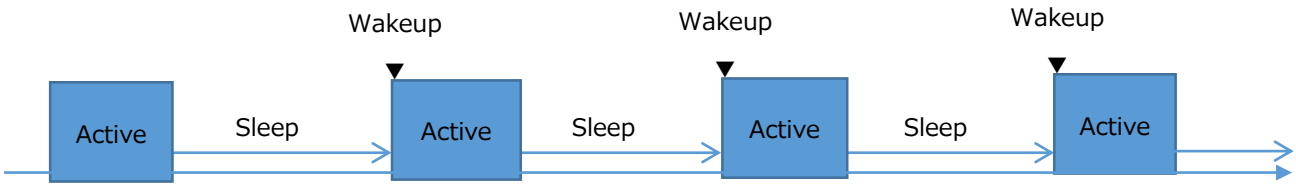
Before setting sleep mode, WDT must be set. Enable WDT when return to sleep mode by way of WDT and start setting WDT. Power-down mode (WDT disabled) is active if WDT is disable. Stop ADC and BOD when sleep mode for lowering current.

3-5-4. Wakeup

There are 2 ways for wakeup. Sketch relating to Power-down mode is described.

・The way using WDT (automatically return at a certain interval) maximum: 8 secs

・The way using external interrupt



1) The way using WDT

The sleep time of WDT is up to 8 seconds. If you want to sleep more than that, you can get a sleep time that is an integral multiple of 8 seconds by repeating Sleep multiple times。

Maximum sleep time of WDT is 8 second. Sleep repeatedly make the time over that.

The required function for WDT control

wdt_reset()	: Reset WDT function
wdt_enable(value)	: Enable WDT function
	value: time, constant
	Reset also occurs when WDT overflows
wdt_disable()	: Disable WDT function

Call library

```
#include <avr/wdt.h>
```

2) The way using external interruption

External interrupt when the response value of sensor is over the threshold value or switch is changing between ON/OFF during the program is running.

Interruption of 8bit-MCU Leaf is assigned to pin 2 or pin 3. Interrupt when switch between High and Low.

External interrupt processing function

attachInterrupt(interruption number, function name, interruption mode)

interruption number

external interruption　pin 2: interruption number =0

pin 3: interruption number =1

function name

function called when interrupt

interruption mode

LOW: occur when pin is low

CHANGE: occur when the state of pin is changing

RISING: occur when the state of pin changes from low to high

FALLING: occur when the state of pin changes from high to low