AP01A AVR MCU

1. 概要

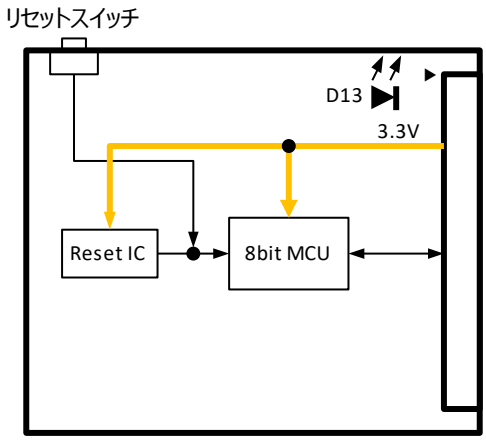
ATmega328P を使用したリーフ。14 個のデジタル入出力ピン（6 個は PWM 出力として使用可能）、6 個のアナログ入力ピン、8MHz 振動子、およびリセットボタンを備えている。

USB 接続する場合は USB を接続、ICSP を使用する場合は Shield を接続する。

Arduino IDE 使用時は、ボードを Arduino Pro or Pro Mini、プロセッサを ATmega328P(3.3V,8MHz)選択。

2. リーフ仕様

2-1. ブロック図



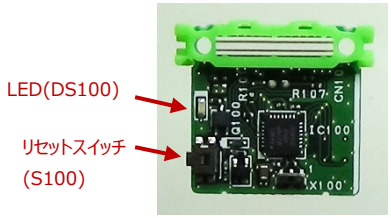
2-2. 電源仕様

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	-

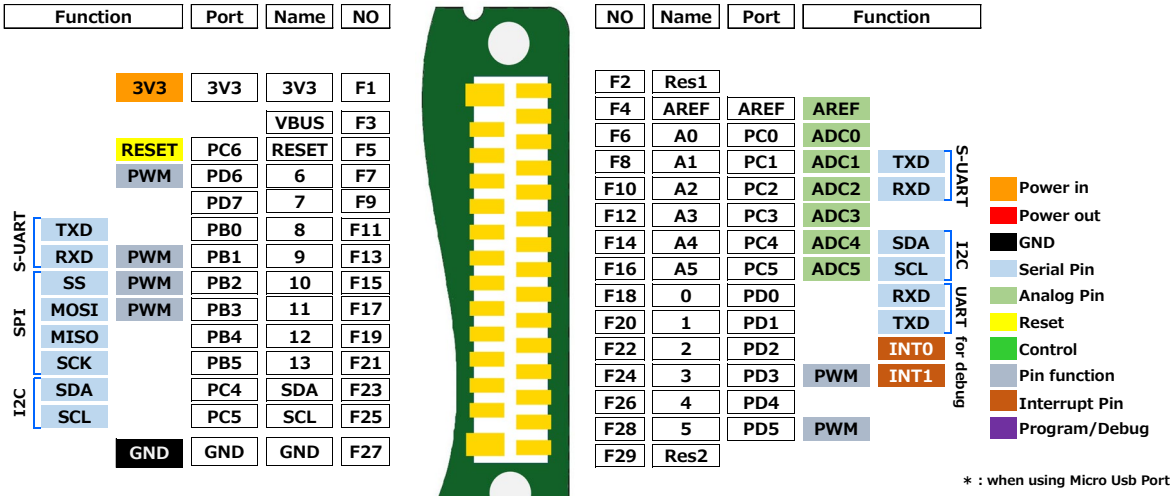
2-3. 主要部品

部品番号	部品名	型番	ベンダー名	備考
IC100	AVR MCU	ATmega328P-MMH	Microchip	28pinQFN

2-4. 外観

表面	裏面
	

2-5. ピンアウト



2-6. LED/スイッチ

項目	部品番号	内容
LED	DS100	pin 13 により LED 制御する(Arduino UNO と同じ) 抵抗 R105(1kΩ)を外すことにより点灯しないように出来る
リセットスイッチ	S100	マイコン、および他のデバイスをリセットする

3. 8bit MCU(ATmega382P-MMH)仕様

3-1. 概要

項目	内容
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. 電気的特性

3-2-1. 最大定格

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

3-2-2. 定格　(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. データシートリンク先

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

3-4. 主な関数とライブラリ

3-4-1. デジタル入出力

関数	概要
pinMode(pin ,mode)	ピンの動作を入力か出力に設定。 【パラメータ】 pin: 設定したいピンの番号 mode: INPUT(内部プルアップは無効) 、INPUT_PULLUP(内部プルアップ抵抗を有効)、OUTPUT 【戻り値】 なし
digitalWrite(pin, value)	HIGH または LOW を指定したピンに出力。 【パラメータ】 pin: ピン番号 value: HIGH(3.3V)か LOW(0V) 【戻り値】 なし
digitalRead(pin)	指定したピンの値を読み取る。 【パラメータ】 pin: 読みたいピンの番号 【戻り値】 HIGH または LOW

3-4-2. アナログ入力

関数	概要
analogRea d(pin)	指定したアナログピンの値を読み取る。 【パラメータ】 pin: 読みたいピンの番号 読み取りに使いたいピンの番号を整数で指定。0 から 5 が有効な数値。 【戻り値】 0 から 1023 までの整数値

3-4-3. 外部割込

関数	概要
attachInterrupt (interrupt, function, mode)	外部割り込みが発生したときに実行する関数を指定。割り込み番号(int.0~)と、それに対応するピン番号は下記のとおり。 pin2(int.0) pin3(int.1) 【パラメータ】 interrupt: 割り込み番号 function: 割り込み発生時に呼び出す関数 mode: 割り込みを発生させるトリガ LOW ピンが LOW のとき発生 CHANGE ピンの状態が変化したときに発生 RISING ピンの状態が LOW から HIGH に変わったときに発生 FALLING ピンの状態が HIGH から LOW に変わったときに発生 【戻り値】 なし
detachInterrupt (interrupt)	割り込みを無効にする。 【パラメータ】 なし 【戻り値】 なし

3-4-4. UART 通信(USB-シリアル変換)

関数	概要
Serial.begin(speed)	シリアル通信のデータ転送レート(ボーレート)を指定。Arudino IDE と接続する場合は 115200 を設定。 【パラメータ】 speed: 転送レート (int) 【戻り値】 なし
Serial.end()	シリアル通信を終了。 【パラメータ】 なし 【戻り値】 なし
Serial.read()	受信データを読み込み。 【パラメータ】 なし 【戻り値】 読み込み可能なデータの最初の 1 バイトを返す。-1 の場合は、データが存在しない
Serial.flush()	データの送信がすべて完了するまで待つ。 【パラメータ】 なし 【戻り値】 なし

Serial.print(data, format)	テキスト形式でデータをシリアルポートへ出力する。 オプションの第 2 パラメータによって基数(フォーマット)を指定できる。 【構文】 Serial.print(data) Serial.print(data, format) 【パラメータ】 data: 出力する値。すべての型に対応。 format: 基数または有効桁数(浮動小数点数の場合) 【戻り値】 送信したバイト数
Serial.println(data, format)	データの末尾に CR と LF を付けて送信。Serial.print()と同じフォーマットが使える。詳細は Serial.print()の項を参照。 【パラメータ】 data: すべての整数型と String 型 format: data を変換する方法を指定 (省略可) 【戻り値】 送信したバイト数 (byte)
Serial.write(val)	シリアルポートにバイナリデータを出力。 【構文】 Serial.write(val) Serial.write(str) Serial.write(buf, len) 【パラメータ】 val: 送信する値(1 バイト) str: 文字列(複数バイト) buf: 配列として定義された複数のバイト len: 配列の長さ 【戻り値】 送信したバイト数 (byte)

3-4-5. ソフトウエアシリアル通信

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

関数	概要
SoftwareSerial name(rxPin, txPin)	Software Serial を使用可能にする。オブジェクトに名前を付ける必要がある。 SoftwareSerial.begin()を実行することも必要。複数のポートを同時に開くことができるが、受信できるのは 1 度に 1 ポートのみ。 【パラメータ】 rxPin: データを受信するピン txPin: データを送信するピン
name.begin(speed)	シリアル通信のスピード(ボーレート)を設定する。通常は 9600 を設定する。 【パラメータ】 speed: ボーレート (long) 【戻り値】 なし

name.read()	受信した文字を返す。同時に複数の SoftwareSerial で受信することはできない。listen()を使って、ひとつ選択する必要がある。 【パラメータ】 なし 【戻り値】 読みこんだ文字（データがないときは -1）
name.print(data)	ソフトウェアシリアルポートにデータを出力。Serial.print()と同じ機能。 【パラメータ】 Serial.print()の項参照。 【戻り値】 送信するバイト数（byte）
name.println(data)	ソフトウェアシリアルポートにデータを出力。Serial.println()と同じ機能。 【パラメータ】 Serial.println()の項参照。 【戻り値】 送信したバイト数（byte）
name.listen()	指定したソフトウェアシリアルポートを受信状態(listen)する。同時に複数のポートを受信状態にすることはできない。 【パラメータ】 なし 【戻り値】 なし
name.write(val)	ソフトウェアシリアルポートにデータを出力。Serial.write()と同じ機能。 【パラメータ】 Serial.write()の項参照。 【戻り値】 送信したバイト数（byte）

3-4-6. I2C 通信

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

関数	概要
Wire.begin(address)	Wire ライブラリを初期化し、I2C バスにマスタかスレーブとして接続。 【パラメータ】 address: 7 ビットの I2C スレーブアドレス。省略した場合は、マスタとしてバスに接続。 【戻り値】 なし
Wire.requestFrom(address, count)	他のデバイスにデータを要求。データは read()関数を使って取得。 【パラメータ】 address: データを要求するデバイスのアドレス(7 ビット) quantity: 要求するデータのバイト数 stop(省略可): true に設定すると stop メッセージをリクエストのあと送信 false に設定すると restart メッセージをリクエストのあと送信 【戻り値】 実際に受信したバイト数を返す。

Wire.beginTransmission(address)	指定したアドレスの I2C スレーブに対して送信処理を開始。 【パラメータ】 address: 送信対象のアドレス(7 ビット) 【戻り値】 なし
Wire.endTransmission()	スレーブデバイスに対する送信を完了する。 【パラメータ】 stop(省略可): true に設定すると stop メッセージをリクエストのあと送信(デフォルト)。 false に設定すると restart メッセージをリクエストのあと送信 【戻り値】 送信結果 (byte) 0: 成功 1: 送ろうとしたデータが送信バッファのサイズを超えた 2: スレーブアドレスを送信し、NACK を受信した 3: データ・バイトを送信し、NACK を受信した 4: その他のエラー
Wire.write(value)	データを送信。beginTransmission()と endTransmission()の間で実行する。 【構文】 Wire.write(value) Wire.write(string) Wire.write(data, length) 【パラメータ】 value: 送信する 1 バイトのデータ (byte) string: 文字列 (char *) data: 配列 (byte *) length: 送信するバイト数 (byte) 【戻り値】 送信したバイト数 (byte)
Wire.read()	データを受信。マスタデバイスでは、requestFrom()を実行したあと、スレーブから送られてきたデータを読み取るときに使用。 【パラメータ】 なし 【戻り値】 受信データ (byte)

3-4-7. ウォッチドッグタイマー

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

関数	概要
wdt_enable (value)	ウォッチドッグタイマーを有効にする。リセットされるまでの時間は 15ms～8s。 【パラメータ】 リセットがされるまでの時間 【戻り値】 なし

wdt_reset()	ウォッチドッグタイマーをリセットする。 【パラメータ】 なし 【戻り値】 なし
wdt_disable ()	ウォッチドッグタイマーを無効にする。 【パラメータ】 なし 【戻り値】 なし

3-4-8. スリープモード

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

関数	概要
set_sleep_mode (parameter)	スリープモードの設定。 【パラメータ】 parameter : SLEEP_MODE_PWR_DOWN SLEEP_MODE_PWR_SAVE SLEEP_MODE_STANDBY SLEEP_MODE_IDLE SLEEP_MODE_EXT_STANDBY 【戻り値】 なし
sleep_enable()	スリープを有効にする。
sleep_mode()	スリープを開始する。

3-4-9. タイマー割り込み

include file : MsTimer2.h (Contributed Libraries)

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

関数	概要
MsTimer2::set (unsigned long ms, void (*f)())	タイマー時間を ms で指定。時間が来るたびに関数 f が呼ばれる。f は引数なしの void 型として宣言。 【パラメータ】 オーバーフローする時間(ms) 【戻り値】 なし
MsTimer2::sta rt()	タイマー割り込みを有効にする。 【パラメータ】 なし 【戻り値】 なし
MsTimer2::sto p()	タイマー割り込みを無効にする。 【パラメータ】 なし 【戻り値】 なし

3-5. 省電力制御

3-5-1. Active モードと Sleep モード

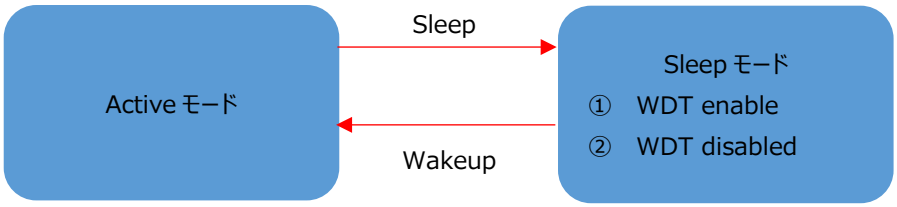
Active モード：動作状態。通常、大きな電源電流が流れる。

Sleep モード：低消費電力状態。スタンバイ・モードともいう。

Wakeup：Sleep モードから Active モードへの移行。Wakeup には、典型的には次の 2 種類がある。

1)WDT（ウォッチドッグタイマー）：MCU が持っているタイマーで定期的（例えば数秒間ごと）に Wakeup する。この機能を生かしたい場合は Sleep モードに入る前に、WDT を enabled に設定する必要がある。

2)外部割り込み：センサー値の変化やリアルタイムクロックなどによる割り込みで Wakeup する。WDT に比べて MCU を低消費電力に出来る。



3-5-2. Sleep モード・WDT サンプルスケッチ

Sleep 関数と WDT 関数のライブラリ群を使って定時間ごとに Wakeup するサンプルスケッチ。Sleep 時間は、8 秒となる。
avr/wdt.h の wdt_enable()関数を使用すると WDT による復帰時にリセットが発生する。Wakup 時にリセットをさせたくない場合は以下の様に WDT の設定を ATmega328P のレジスタに書き込む。

```
#include <avr/wdt.h>
#include <avr/sleep.h>
void WDT_setup(){
        cli();                                          // 割り込み禁止
        wdt_reset();                                    // WDT タイマーカウンタリセット
        MCUSR &= ~(1 << WDRF);                          // WatchDog system Reset Flag(WDRF)リセット
        WDTCSR |= 1 << WDCE | 1 << WDE;     //WDT 変更有効（WDCE と WDE を同時に 1 にセットで WDT 変更許可)
        WDTCSR = 1 << WDIE | 0 << WDE | 1 << WDP3 | 0 << WDP2 | 0 << WDP1 | 1 << WDP0; //WDT 設定
                                                        // WDE=0,WDIE=1 :WDT overflow で割り込み
                                                        // WDP3=1,WDP2=0,WDP1=0,WDP0=1: 8s
        sei();                                          //割り込み許可
}
void sleepMode(){
        Serial.println("Sleep Mode");
        delay(10);
        WDT_setup();
        set_sleep_mode(SLEEP_MODE_PWR_DOWN);        //SLEEP モード設定
        sleep_bod_disable();                            //低電圧検出器(BOD)禁止
        sleep_mode();                                   //SLEEP 移行
        __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 がタイムアップした時に実行される処理
        wdt_disable();
}
```

3-5-3. Sleep モード

ATmega328P では Sleep モードの一つに最も低電力状態になる Power-down mode がある。ここでは、Power-down モード関連のスケッチの書き方について説明する。

Sleep モードへの移行に必要な関数

set_sleep_mode(パラメータ)　：Sleep モード(パラメータ)設定

sleep_mode()　：Sleep モードに移行

ライブラリの呼び出し

#include <avr/sleep.h>

Sleep モードパラメータ	処理内容
SLEEP_MODE_PWR_DOWN	Power-Down モードへ設定

スケッチの例

```
#include <avr/sleep.h>
void setup() { }
void sleep() {
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);
  ADCSRA &= ~(1 << ADEN);                          //ADC 停止
  sleep_enable();
  MCUCR |= (1 << BODSE) | (1 << BODS);             // MCUCR の BODS と BODSE に 1 をセット
  MCUCR = (MCUCR & ~(1 << BODSE)) | (1 << BODS);   // すぐに（4 クロック以内）BODSSE を 0, BODS を 1 に設定
  asm("sleep");                                    // 3 クロック以内にスリープ
}
```

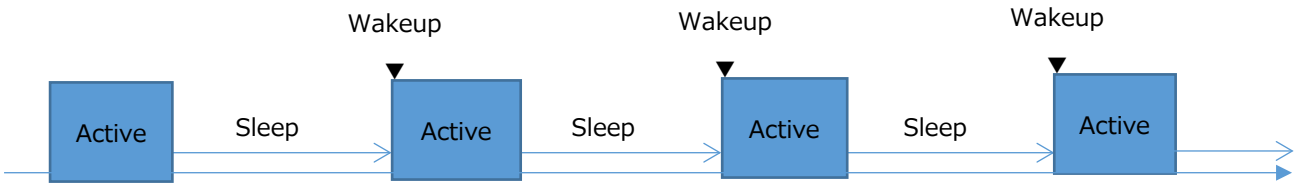
Sleep モード設定前に WDT の設定を行う。WDT による Sleep モード復帰を行う場合は WDT を有効にし、WDT の設定を行い開始する。ここで WDT を無効にすることにより、Power-down mode(WDT disabled)に設定することが可能。
消費電流を下げるためには、Sleep モード設定時に ADC と BOD を停止させる必要がある。

3-5-4. Wakeup

Wakeup には、以下の方法がある。ここでは、Power-down モード関連のスケッチの書き方について説明する。

・WDT を使う方法(一定時間間隔で自動復帰)最大 8 秒

・外部割り込みを使う方法



1)WDT での Wakeup

WDT の Sleep 時間は、最大 8 秒である。それ以上 Sleep させたい場合は、Sleep を複数回繰り返すことにより 8 秒の整数倍の Sleep 時間を得ることが出来る。

WDT の制御に必要な関数

wdt_reset()	： WDT の設定をリセットするための関数
wdt_enable(value)	：WDT を有効にする関数。value に定数を入れて時間を設定する ただし、WDT のオーバーフロー発生時リセットが発生する
wdt_disable()	： WDT を無効にする関数

ライブラリの呼び出し

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

2)外部割り込みでの Wakeup

外部割り込みは、通常のプログラム実行中に、センサーなどの応答値が閾値を超えたり、スイッチが ON/OFF 切り替わった時に割り込みが発生し処理を行う。

8bit-MCU Leaf には、pin 2 または pin 3 に割り込み機能が割り当てられており、High と Low の切り替わりで割り込みが発生する。

割り込み処理関数

attachInterrupt(割り込み番号, 関数名, 割り込みモード)

割り込み番号

外部割り込み pin 2 の場合：割り込み番号=0

pin 3 の場合：割り込み番号=1

関数名

割り込み発生時に呼び出す関数

割り込みモード

LOW ：ピンが Low のとき発生

CHANGE：ピンの状態が変化したときに発生

RISING ：ピンの状態が Low から High に変わったときに発生

FALLING：ピンの状態が High から Low に変わったときに発生