AP02A ARM M0

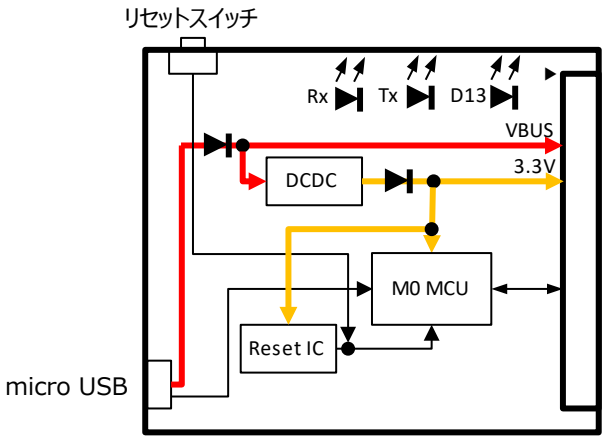
1. 概要

AVR MCU の 32bit 拡張版のリーフである。このボードは Microchip の SAMD21 MCU を搭載し、32 ビット ARMCortex®M0 コアを搭載している。USB の電源を供給して、VBUS(5V)電源として使用可能。

Arduino IDE 使用時は、ボードを Arduino M0 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	3.63V
Idd	Operating current	Active	-	3.4mA	-
		Sleep	-	4.6uA	-

USB 接続時

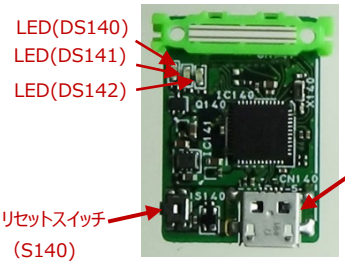
Vout	Output Voltage	–	3.234V	3.3V	3.366V
Iout(max)	MaximumOutput current	–	500mA	-	-
Ilim	Current Limit	–	1.3A	1.5A	2.5A

2-3. 主要部品

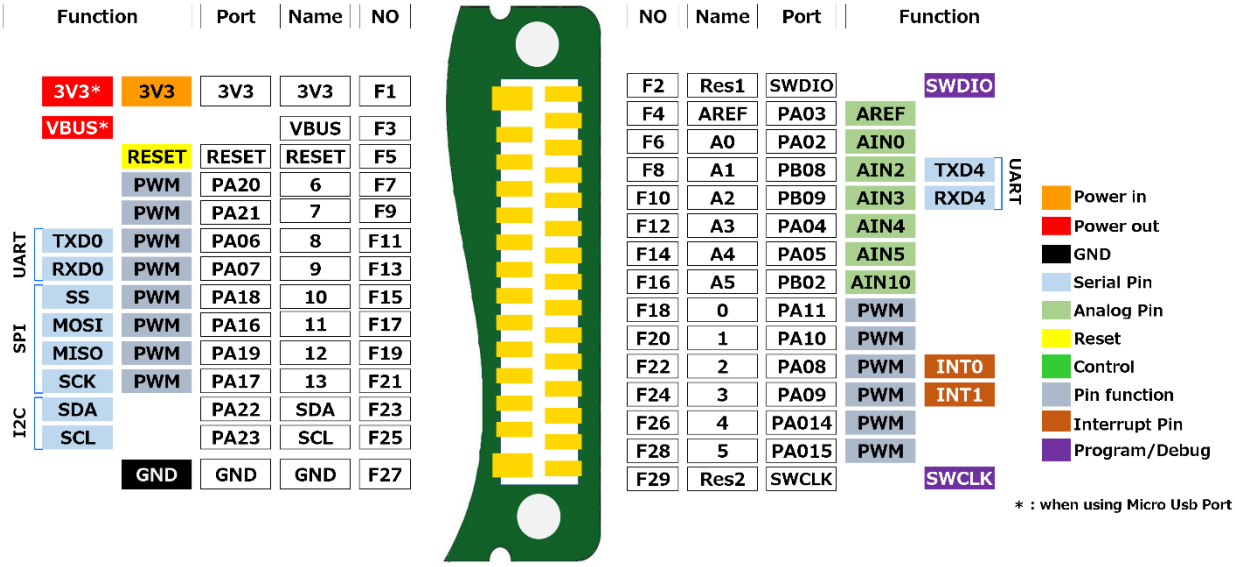
部品番号	部品名	型番	ベンダー名	備考
IC140	ARM M0 MCU	ATSAMD21G18A-MUT	Microchip	–
IC141	降圧電源 IC	XCL222B331ER	TOREX	VBUS→3.3V

2-4. 外観

リーフ外観と実装部品は次の通り。

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	DS140	pin 13 により LED 制御する
LED	DS141	UART TX 通信によるプログラム書き込み時/デバッグ時に点滅する
LED	DS142	UART RX 通信によるプログラム書き込み時/デバッグ時に点滅する
リセットスイッチ	SW140	マイコン、および他のデバイスをリセットする

3. ARM M0 MCU(ATSAMD21G18A-MUT)仕様

3-1. 概要

項目	内容
Microcontroller	ATSAMD21G18, 48pins QFN
Architecture	ARM Cortex-M0+
Operating Voltage	3.3V
Input Voltage	1.5-5 V
Digital I/O Pins	14, with 12 PWM and UART

Analog I/O Pins	6 +1 DAC
Flash memory	256 KB
SRAM	32KB
Clock Speed	48 MHz
LED_BUILTIN	13
Compatibility	Arduino M0

3-2. 電気的特性

3-2-1. 最大定格

Parameter	Value
Operating Temperature	-40℃ to +85℃
Maximum Operation Voltage	3.8V

3-2-2. 定格

Symbol	Parameter	Condition	Min.	Typ.	Max.
Vdd	Power Supply Voltage	–	1.62V	3.3V	3.63V
Idd	Active	25℃	3.11mA	3.37mA	3.64mA
		CoreMark 25℃	5.78mA	6.32mA	6.80mA
	IDLE0	25℃	1.89mA	2.04mA	2.20mA
	IDLE1	25℃	1.34mA	1.46mA	1.58mA
	IDLE2	25℃	1.07mA	1.17mA	1.28mA
	STANDBY	XOSC32K running RTC running at 1kHz 25℃	-	4.06uA	12.8uA
		XOSC32K and RTC stopped 25℃	-	2.70uA	12.2uA

3-3. データシートリンク先

https://www.microchip.com/wwwproducts/en/ATSAMD21G18

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(digitalP inToInterrupt(pin), 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 対応するピン番号は下記の通り。 all digital pins, except 4 【パラメータ】 digitalPinToInterrupt(pin):PIN 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 HIGH ピンが HIGH のとき発生【戻り値】 なし
detachInterrupt(Pin)	割り込みを無効にする。 【パラメータ】 Pin 番号 【戻り値】 なし

3-4-4. UART 通信

関数	概要
Serial1.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
Serial1.end()	シリアル通信を終了。 【パラメータ】 なし 【戻り値】 なし
Serial1.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
Serial1.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
Serial1.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】 Serial1.print(data) Serial1.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
Serial1.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

Serial1.write(val)	シリアルポートにバイナリデータを出力。 【構文】 Serial1.write(val) Serial1.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

3-4-5. USB 通信(デバッグ)

関数	概要
SerialUSB.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
SerialUSB.end()	シリアル通信を終了。 【パラメータ】 なし 【戻り値】 なし
Serial1.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
SerialUSB.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
SerialUSB.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】 SerialUSB.print(data) SerialUSB.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
SerialUSB.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)
SerialUSB.write(val)	シリアルポートにバイナリデータを出力。 【構文】 SerialUSB.write(val) SerialUSB.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)

3-4-6. 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. スリープモード

include file : ArduinoLowPower.h (RTC Library for Arduino)
https://github.com/arduino-libraries/ArduinoLowPower
include file : ZeroTimer.h (RTC Library for Arduino)
https://github.com/EHbtj/ZeroTimer

関数	概要
LowPower.sleep(time)	Sleep モードに移行します 【パラメータ】 スリープする時間(ms)　外部割り込みで起きる場合は時間設定は行わない 【戻り値】 なし

3-4-8. タイマー割り込み

include file : ZeroTimer.h
https://github.com/EHbtj/ZeroTimer

関数	概要
TC.startTimer(unsigned long period, void (*f)())	TC3 タイマーを使用したタイマーカウンタを生成 【パラメータ】 オーバーフローする時間(ms) オーバーフローした時呼ばれる関数 【戻り値】 なし
TCC.startTimer (unsigned long period, void (*f)())	TCC0 タイマーを使用したタイマーカウンタを生成します。f は引数なしの void 型として宣言してください。 【パラメータ】 オーバーフローする時間(ms) オーバーフローした時呼ばれる関数 【戻り値】 なし

3-5. 省電力制御

ARM M0 をタイマーで Sleep モードと Active モードの移行を行う場合は WDT ではなく、ARM M0 内蔵の RTC を使用する。