人工知能ミニ四駆 AI ユニット

ソフトウェア開発ガイド

QUANTUM　赤坂清隆

目次

注意事項

本 SDK では以下のオープンソースソフトウェアを静的あるいは動的に利用しています。　ただし本 SDK への組み込みにあたって追加修正等の措置を行っている可能性がある為、オリジナルの挙動を保証するものであはりません。

- FatFs
  - http://elm-chan.org/fsw/ff/00index_j.html

ソフトウェア開発環境概要

ソフトウェアコンポーネント

人工知能ミニ四駆 AI モジュール SDK（以下、単に SDK）は、人工知能ミニ四駆 AI モジュール（以下単に AI モジュール）上で動くソフトウェアの開発を容易化するために、オンボードセンサやタイマー、割り込み、モータードライバなどハードウェア向けデバイスドライバや、ソフトウェア開発のためのフレームワークを提供します。

本 SDK を用いて開発されるソフトウェアは、上記デバイスドライバやフレームワークを含むシステムソフトウェアと、ユーザが定義するユーザランドソフトウェアから構成されます。

デバイスドライバは GPIO、I2C、アナログ - デジタルコンバータ（以下 ADC）、周期タイマ等マイコンのハードウェアに依存する処理を C 言語の関数として定義した関数ライブラリとして提供されます。　通常ユーザランドから呼び出す必要はありませんが、システムソフトウェアはデバイスドライバを直接使用しています。　デバイスドライバは通常関数呼出によって機能を提供しますが、割込処理については C 言語の関数ポインタを用いたコールバックによって割込み発生を通知する機能を有しています。

システムソフトウェアは AI によるミニ四駆制御のフレームワークを提供します。マイコン上で発生した全ての割込は一旦システムソフトウェアがハンドルし、必要な割り込みについてはコールバック関数によってユーザランドに通知します。　AI モジュール上の各デバイスは、ユーザランドの処理が呼ばれるよりも前にシステムソフトウェアによって初期化が行われます。

ユーザランドソフトウェアはエントリーポイント(main 関数）を含む C 言語の関数と言う形でユーザ自身が定義するものです。初期状態の SDK には空のユーザランドソフトウェアが含まれています。　システムソフトウェアが提供する API 関数はユーザランド側のどこから呼び出しても安全に設計されています。

システムソフトウェアとユーザーランド

SDK で用意されているソフトウェアモデルでは、2 つの方法でシステムソフトウェアが提供する機能を使う事が出来ます。1 つは、ユーザランドでユーザが定義した関数からシステムソフトウェア側の API を呼び出す方法です。もう 1 つは、ユーザランド内で、各種イベントに対してコールバック関数を登録することで、システムソフトウェアが検知した各種イベントに対しての処理を記述する方法です。

SDK ディレクトリ構成(TBD)

SDK のディレクトリ構成と各ディレクトリに含まれるファイルの概要を説明します

```
├──docs                    ： ドキュメントが保存されたディレクトリ
└──sdk
    ├──build               ： ビルド成果物が生成される場所
    ├──include             ： ヘッダファイル一式
    ├──lib                 ： 各種ライブラリ
    ├──Samples             ： 各種サンプルコード
    ├──Skeleton            ： スケルトンプロジェクト
    ├──SystemSoftware      ： ドライバ等を含むシステムソフトウェア
```

ベアメタル開発向け参考情報

SDK が提供しているフレームワークはデバイスドライバやシステムソフトウェアと言った抽象化レイヤを多く含み関数呼び出し等のコストが比較的多く必要です。この為 HW の性能をフルに生かす事が構造上難しい場合があります。デバイスドライバやシステムソフトウェアを含む SDK 全体での最適化でも解決できない場合、SDK を使わずにベアメタル開発を行う事も選択肢の 1 つです。SDK に含まれるデバイスドライバに適用されるライセンスは再利用を制限しない為、ドライバの中の必要な部分だけを切り出して利用する事も可能です。

その他開発者向け情報

ハードウェア情報

開発基板に搭載されている部品の中で、ソフトウェアでの制御が必要になる CPU 及びモーションセンサの情報です。

- Atmel Xmega A4U Series (オンボード CPU)
  - 製品情報
    - http://www.microchip.com/wwwproducts/en/ATxmega128A4U
  - ハードウェアマニュアル
    - http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8331-8-and-16-bit-AVR-Microcontroller-XMEGA-AU_Manual.pdf
- LSM6DS3H (6 軸モーションセンサ)
  - 製品情報
    - http://www.st.com/ja/mems-and-sensors/lsm6ds3h.html
  - データシート
    - http://www.st.com/content/ccc/resource/technical/document/datasheet/group0/38/96/79/af/ab/62/42/c3/DM00229854/files/DM00229854.pdf/jcr:content/translations/en.DM00229854.pdf

FATFS

SDK ではファイルシステムとして ChaN 氏制作の FatFs を利用しています。FatFs に関する各種情報は本ドキュメントの範囲外としています。FatFs の API に関する情報等は公式 Web ページに記載されている API リファレンスを参照してください。

- FatFs 公式 Web ページ
  - http://elm-chan.org/fsw/ff/00index_j.html
- FatFs アプリケーションノート
  - http://elm-chan.org/fsw/ff/ja/appnote.html

開発手順

ここでは SDK を用意した状態から、実際にバイナリファイルをビルトし、制御基板に書き込むまでの手順を説明します。

開発環境の準備

本 SDK は開発環境として AVR Studio 7 を想定しています。リリース時点で動作確認を行っているバージョンは 7.0.1006 です。Microchip の公式 Web サイト (http://www.microchip.com/development-tools/atmel-studio-7)からダウンロードが可能です。

サンプルのビルド方法

SDK ディレクトリの中にあるサンプルプロジェクト(TBD ディレクトリパスを後で決める)を AVR Studio 7 で開きます。メニューバーの[Build] → [Build Solution] を選択します。

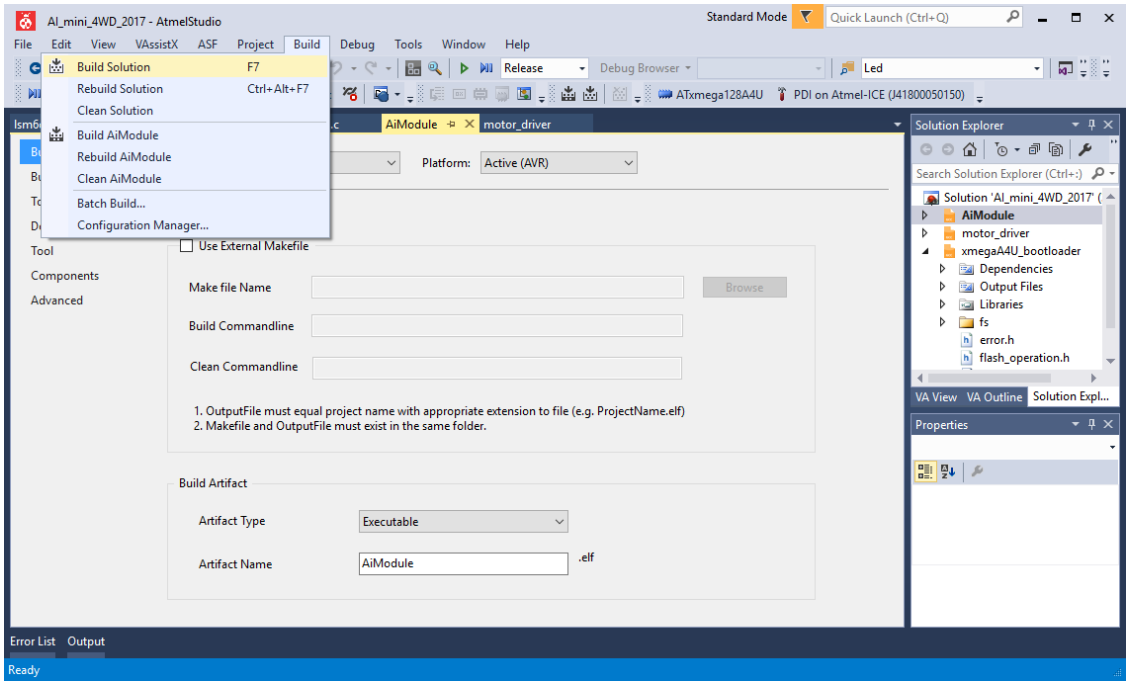

図 1 AVR Studio でのビルド方法

Output ウィンドに Build Successed. と表示されていればビルドは成功です。ビルド成果物は、$(SDK が展開されているディレクトリ)/soft/AI_mini_4WD_2017/build に MINI4WD.AUP という名前で生成されます。

基板へのソフトウェア書込の方法

基板へのソフトウェア書込みには 2 つの方法があります。

ハードウェアプログラマを利用した書込み方法

1 つ目は AVR 用のプログラマを用いた方法です。SDK リリース時点で動作確認が取れているプログラマは AVRISP mkII と JTACICE mkII の 2 機種です。何れの機種でも PDI 接続で接続します。接続時の基板側のピン配置は、「ミニ四駆制御基板ハードウェアマニュアル」ドキュメントを参照してください。この方法で書き込むには以下の操作を行います。

1. 基板とプログラマを接続
2. 「**AiModule**」というプロジェクトを右クリックし「Set as StartUp Project」というメニューを選択

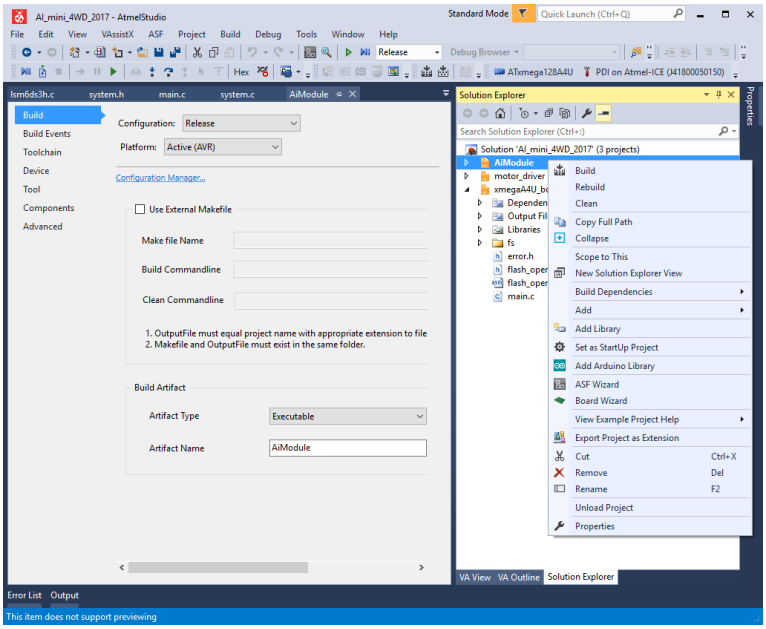

図 2 Set as StartUp Project メニューを選択

3. AVR Studio の Solution Explorer からプロジェクトのプロパティを選択

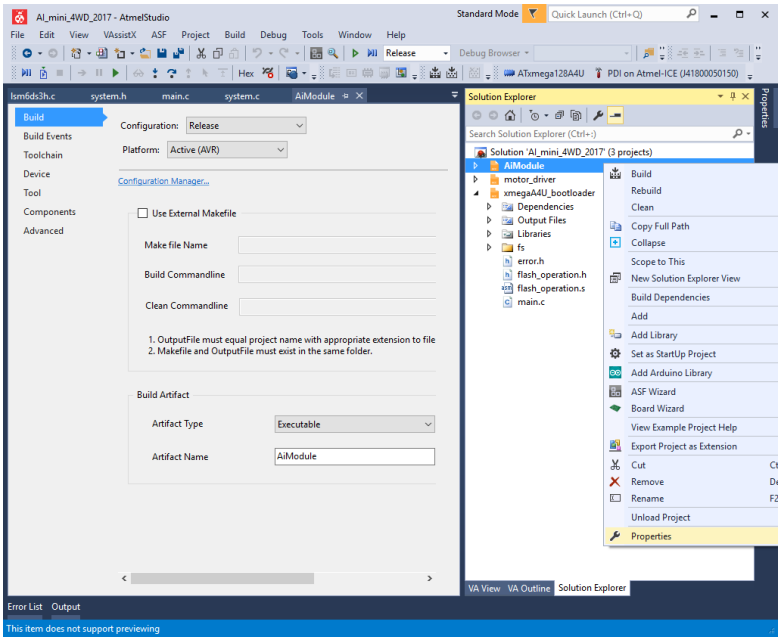

図 3 Solution Explorer でプロジェクトのプロパティを選択

4. プロパティの Tool タブの Selected debugger/programmer ので使用するデバイスを選択

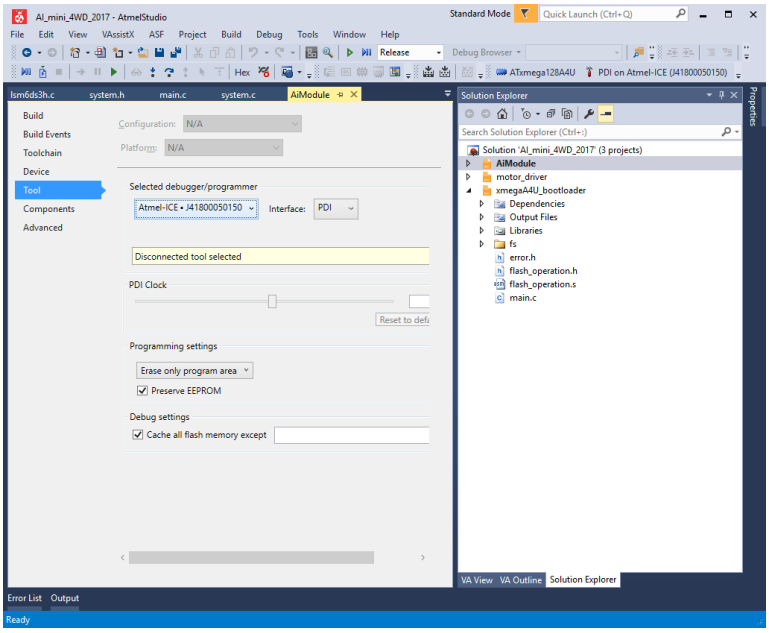

図 4 プロパティの Tool タブで使用するツールを選択

5. 制御基板の電源を投入

6. AVR Studio のメニューバー［Debug]→［Start Debugging and Break］又は、[Debug]→[Start Without Debugging］を選択

なお、ハードウェアプログラマの使用方法等については Atmel の公式 Web ページ又は製品付属のマニュアルにより詳しい記載があるので合わせてご参照ください。

SD カード経由での書込み

制御基板側にブートローダー及びアップデート機能に対応したシステムソフトウェアが書込み済みであれば、SD カードを介してのアップデートが可能です。SDK に含まれるシステムソフトウェアはユーザ側が意図的に機能を削除しない限り、この機能に対応した状態でリリースされています。

SD カードでの書き込みを行うには以下の操作を行います。

1. SD カードの直下に MINI4WD.AUP ファイルを置きます
2. 制御基板の電源を OFF にする
3. SW0 を押下します
4. SW0 を押下したまま、電源を ON にします
5. ステータス LED（青色 LED）が 3 つとも点灯するまで待ちます
6. SW0 を放します
7. 書込み終了

USB インタフェースを用いた書き込み

制御基板側のシステムソフトウェアで USB デバイス機能と、ブートローダーが有効になっている場合、USB インタフェースを用いた書き込みが可能です。USB デバイス機能の有効化については、aiMini4WdSystemInitialize() を参照してください。

USB インタフェースを通じた書き込みは以下の手順で可能です。

1. Micro SD カードを制御基板に挿入し、制御基板の電源を入れる

2. USB ケーブルを制御基板に接続する
3. OS 側で、USB Mass Storage が認識されるのを待つ。
4. 認識された USB Mass Storage のルートディレクトリに、書き込みたいファームウェアを MINI4WD.AUP という名前で書きこむ
5. USB ケーブルを取り外す。

システムソフトウェア

システムソフトウェアとは

システムソフトウェアはミニ四駆制御を行う為のフレームワークををユーザランド(ユーザが自身で開発する領域）に提供する事を目的として実装されているソフトウェアセットです。

ユーザーランドソフトウェアとの関係性

ユーザランドソフトウェアとシステムソフトウェアは､ユーザランドソフトウェアがシステムソフトウェアが提供する API を呼び出す、または API を通じて登録されたイベントコールバック関数がシステムソフトウェアによって実行されるという関係になります。

システム側から何らかの情報を取得する為に、API 関数が用意されています。API 関数の詳細については、「システムソフトウェア API リファレンス」を参照してください。

システムソフトウェアが提供する機能

システムソフトウェアが API 関数として提供する機能の一覧を示します。個々の機能については「システムソフトウェア API リファレンス」を参照ください。

- 制御基板用バッテリの電圧取得
- ミニ四駆駆動用バッテリ（単 3 乾電池 2 本）の電圧取得（モータードライバから読み取り）
- ミニ四駆用モーターの PWM 制御（正転・逆転）
- デバッグ用 LED の点灯・消灯
- AI モジュール上のスイッチ押下、解放時のイベント登録
- オンボードセンサの更新イベント登録
- オンボードセンサの値の取得
- 5ms、100ms 毎のタイマーイベント登録
- Micro SD 上の FAT ファイルシステム

システムソフトウェア　定義リファレンス

システムソフトウェア側で定義されている、構造体や列挙型について解説します。

AiMini4WdHidsSw

種別

列挙型

概要

この列挙型は、AI モジュールに搭載されているスイッチを表現するための列挙型である。

定義

Name	Value	Description
cAiMini4WdHidsSw0	0x00	スイッチ 0
cAiMini4WdHidsSw1	0x01	スイッチ 1
cAiMini4WdHidsSw2	0x02	スイッチ 2

定義場所

hids.h

AiMini4WdHidsSwCbType

種別

列挙型

概要

この列挙型は、AI モジュールに搭載されているスイッチに対して、押下時、解放時、押されている間定期的に発生するイベントを指定するための列挙型である。

定義

Name	Value	Description
cAiMini4WdHidsSwCbOnPress		スイッチ押下時
cAiMini4WdHidsSwCbOnRepeat		スイッチ押下中
cAiMini4WdHidsSwCbOnRelease		スイッチ解放時

定義場所

hids.h

AiMini4WdMotorDriverMode

種別

列挙型

概要

この列挙型はモータードライバの動作モード（正転、逆転、ブレーキ、解放）を表現するための列挙型である。

定義

Name	Value	Description
cAiMini4WdModeFoward	0	正転モード
cAiMini4WdModeBack	1	逆転モード
cAiMini4WdModeBreak	2	回生ブレーキ
cAiMini4WdModeFree	3	自由回転（解放）

定義場所

motor_driver.h

AiMini4WdSensorsImuData

種別

構造体

概要

オンボード IMU の値を扱うための構造体である。

定義

Field Name	Unit	Description
pitch	Int16_t	X 軸周りの回転
roll	Int16_t	Y 軸周りの回転
yaw	Int16_t	Z 軸周りの回転
ax	Int16_t	X 軸方向加速度
ay	Int16_t	Y 軸方向加速度
az	Int16_t	Z 軸方向加速度

注意事項

得られる値は物理量ではなく、IMU から得られた生データです。物理量に変換するには、以下の変換式を適用する必要があります。

角速度（pitch/roll/yaw)の場合

物理量［dps］ = 0.070 x Value

加速度の場合

物理量[g] = 0.000488 x Value

定義場所

sensors.h

AiMini4WdSensorsMouseData

種別

構造体

概要

マウスモジュールから得られた距離データを扱うための構造体である。

定義

Field Name	Unit	Description
delta_x	Int16_t	X 軸方向の移動距離
delta_y	Int16_t	Y 軸方向の移動距離
reliability	Uint8_t	距離情報の信頼性（値が大きいほうが信頼性が高い）
motion	Uint8_t	

注意事項

光学マウスモジュールは、その特性上路面状況やセンサと路面との距離によってデータの信頼性が大きく変化します。reliability の値が著しく小さい場合（おおむね 50 以下）、移動距離は信頼することが出来ない情報となります。

定義場所

sensors.h

AiMini4WdSensorsData

種別

構造体

概要

AI モジュールに接続されているセンサの情報をひとまとめにして扱うための構造体である。

定義

Field Name	Unit	Description
imu_data	AiMini4WdSensorsImuData	IMU データ
mouse_data	AiMini4WdSensorsMouseData	マウスデータ

注意事項

定義場所

sensors.h

システムソフトウェア API リファレンス

システムソフトウェアが提供する API 関数について解説します。ファイルシステム（FatFS）関連の API 関数については、FatFs の公式 Web ページをご参照ください。

aiMini4WdSystemInitialize

概要

システムソフトウェアを初期化します。この関数を呼ぶことで、CPU 内のモジュールや AI モジュールのオンボードセンサなどが使えるようになります。有効にする機能を引数で選ぶことが可能です。

この関数を呼ばれた後は CPU の割り込みが有効になります。各種イベントハンドラ（コールバック関数）が登録されている場合には、イベントの発生に合わせてイベントハンドラが実行されます。

引数

uint32_t function_bitmap

有効にする機能をスイッチするための引数です。いかに定義されている値に従って使いたい機能を有効化してください。

定義名	値	内容
AI_SYSTEM_FUNCTION_BASE	0x00000001	AI モジュールの基本機能
AI_SYSTEM_FUNCTION_FS	0x00000002	ファイルシステム
AI_SYSTEM_FUNCTION_ADNS9800	0x00000004	マウスモジュール
AI_SYSTEM_FUNCTION_USB	0x00000008	USB 機能

戻り値

int8_t エラーコード

この戻り値はシステムソフトウェアの初期化時に発生した各種エラーをユーザランドに通知することに利用されます。戻り値が AI_OK 以外の値だった場合システムソフトウェアは正常に動作しないことが予想されます。

注意事項

この関数を呼び出さない状態だと、ほかのあらゆるシステムソフトウェア API の動作は未定義です。

使い方

system.h をインクルードしてください。

aiMini4WdSystemGetBatteryVoltage

概要

AI モジュールに接続されているバッテリの電圧を取得します。

引数

なし

戻り値

uint16_t AI モジュール用バッテリの電圧

この戻り値は、AI モジュールに搭載されているバッテリの電圧を mv 単位で返します。

注意事項

使い方

system.h をインクルードしてください。

aiMini4WdSystemRegisterExtTriggerCallback

概要

AI モジュールに接続されている外部トリガーがアクティブになったときに発行されるコールバックを登録します。

引数

AiMini4WdSystemExtTriggerCallback cb

cb は、**typedef void (*AiMini4WdSystemExtTriggerCallback)(void);** で定義されるコールバック関数です。登録されたコールバックは、外部トリガー端子が Lo レベルに落ちたときに実行されます。

戻り値

int8_t エラーコード

関数実行時に発生したエラーを返します。

注意事項

使い方

system.h をインクルードしてください。

aiMini4WdSystemChangeLogOutput

概要

aiMini4WdSystemLog() で残したログの出力先を、標準の UART からファイルへの出力に切り替える為に使用します。AI モジュールは UART 端子を持ちますが走行中にアクセスすることは難しいため、走行時のログはファイルに残すことをお勧めします。

引数

FIL *file

FIL 構造体のポインタを渡します。この file は事前に f_open などで開いておく必要があります。

戻り値

int8_t エラーコード

関数実行時に発生したエラーを返します。

注意事項

この機能を使うには、ファイルシステムを有効化しておく必要があります。

使い方

system.h をインクルードしてください。

aiMini4WdSystemLog

概要

文字列でログを出力します。標準では UART に出力されますが、aiMini4WdSystemChangeLogOutput()を使う事で、FILE にログを書きだすことも可能です。

引数

const char *format, ...

C 標準のフォーマット付き書式を指定できます。

戻り値

なし

注意事項

使い方

system.h をインクルードしてください。

aiMini4WdAllocateMemory

概要

メモリプールからメモリを確保します。

引数

size_t size 確保したいサイズ

戻り値

void *　確保されたメモリ

確保に成功した場合、非 NULL のアドレスが得られます。確保に失敗した場合、NULL が返ります。

注意事項

引数で指定したサイズは、メモリを開放する際に必要になるため、確保する側（ユーザーランド側）で管理しておく必要があります。

使い方

memory_manager.h をインクルードしてください。

aiMini4WdDestroyMemory

概要

メモリプールから確保したメモリをメモリプールに返します。

引数

void *ptr	確保したメモリへのポインタ
size_t size	確保したサイズ

戻り値

なし

注意事項

確保したときと異なるサイズを Destory した場合、メモリ開放がうまくいかないか、うまくいったとしても、その後使用できない領域が発生する場合があります。
aiMini4WdAllocateMemory で指定したサイズを使って Destroy するように心がけてください。

使い方

memory_manager.h をインクルードしてください。

aiMini4WdHidsSetLedValue

概要

デバッグ用 LED の点灯・消灯を行います

引数

uint8_t　val

val には、点灯または消灯させたい LED のパタンを設定することが出来ます。各 LED は下に示す 3 つの値として定義されています。論理和を用いて複数の LED を指定することが出来ます。

指定された LED については点灯、指定されなかった LED については消灯します。すべての LED を消灯するには 0 を設定してください。

Definition	Value	Description
AI_MINI_4WD_LED0	0x01	デバッグ LED0
AI_MINI_4WD_LED1	0x02	デバッグ LED1
AI_MINI_4WD_LED2	0x03	デバッグ LED2

戻り値

なし

注意事項

使い方

hids.h をインクルードしてください。

aiMini4WdHidsSetLed

概要

LED を点灯させます

引数

uint8_t bitmap

この引数で指定されたビットマップに従って、LED を点灯させます。各 LED は下に示す 3 つの値として定義されています。論理和を用いて複数の LED を指定することが出来ます。

Definition	Value	Description
AI_MINI_4WD_LED0	0x01	デバッグ LED0
AI_MINI_4WD_LED1	0x02	デバッグ LED1
AI_MINI_4WD_LED2	0x03	デバッグ LED2

戻り値

なし

注意事項

使い方

hids.h をインクルードしてください。

aiMini4WdHidsClearLed

概要

LED を消灯させます

引数

uint8_t bitmap

この引数で指定されたビットマップに従って、LED を消灯させます。各 LED は下に示す 3 つの値として定義されています。論理和を用いて複数の LED を指定することが出来ます。

Definition	Value	Description
AI_MINI_4WD_LED0	0x01	デバッグ LED0
AI_MINI_4WD_LED1	0x02	デバッグ LED1
AI_MINI_4WD_LED2	0x03	デバッグ LED2

戻り値

なし

注意事項

使い方

hids.h をインクルードしてください。

aiMini4WdHidsToggleLed

概要

LED を点灯・消灯を切り替えます

引数

uint8_t bitmap

この引数で指定されたビットマップに従って、LED の点灯・消灯を切り替えます。各 LED は下に示す 3 つの値として定義されています。論理和を用いて複数の LED を指定することが出来ます。

Definition	Value	Description
AI_MINI_4WD_LED0	0x01	デバッグ LED0
AI_MINI_4WD_LED1	0x02	デバッグ LED1
AI_MINI_4WD_LED2	0x03	デバッグ LED2

戻り値

なし

注意事項

使い方

hids.h をインクルードしてください。

aiMini4WdHidsGetSw

概要

AI モジュールのオンボードスイッチの状態を取得します

引数

AiMini4WdHidsSw sw

取得したいスイッチを指定します。

戻り値

uint16_t タイマー割り込みの頻度（Hz）

タイマー割り込みの発生頻度です。

注意事項

タイマー割り込みはマイコンの周期タイマーを用いて発生させています。また、システムソフトウェアはこのタイマーを基準にして様々な処理を走らせます。このため、タイマー割り込みの発生間隔が一定になる保証はありません。　確実に一定間隔の処理が必要な場合には、センサーデータ取得割り込みを基準にすることをお勧めします。

使い方

hids.h をインクルードしてください。

aiMini4WdHidsRegisterSwCallback

概要

AI モジュールのオンボードスイッチの押下時、押下中繰り返し、解放時に発生するイベントに対するコールバック関数を登録します。

引数

AiMini4WdHidsSwCbType　type

AiMini4WdHidsSw　sw

AiMini4WdHidsSwCb　cb

type はイベントタイプ（押下時、押下中繰り返し、解放時）を指定します。sw はどのスイッチを対象としてコールバック関数を設定するかを指定します。cb は、

```
typedef void (*AiMini4WdHidsSwCb)(void);
```

で定義されるコールバック関数を指定します。

戻り値

int8_t エラーコード

注意事項

使い方

hids.h をインクルードしてください。

aiMini4WdMotorDriverSetDuty

概要

モータードライバに対して PWM 制御の制御値（Duty）を設定します。

引数

AiMini4WdMotorDriverMode　direction

uint8_t　duty

direction には、cAiMini4WdModeFoward　または　cAiMini4WdModeBack を指定します。それ以外の値を指定した場合にはエラーとなります。duty は 0～255（0 で停止、255 でフルパワー）を指定します。

戻り値

int8_t エラーコード

注意事項

使い方

motor_driver.h をインクルードしてください。

aiMini4WdMotorDriverGetCurrentSettings

概要

モータードライバの状態を取得します。

引数

AiMini4WdMotorDriverMode *mode

uint8_t *duty

現在の動作モードは mode に、現在の制御指示値は duty に格納されます。

戻り値

int8_t エラーコード

注意事項

使い方

motor_driver.h をインクルードしてください。

aiMini4WdMotorDriverBreak

概要

モータードライバを使って回生ブレーキをかけます

引数

なし。

戻り値

int8_t エラーコード

注意事項

使い方

motor_driver.h をインクルードしてください。

aiMini4WdMotorDriverNeutral

概要

モータードライバをニュートラル状態にします。

引数

なし。

戻り値

int8_t エラーコード

注意事項

使い方

motor_driver.h をインクルードしてください。

aiMini4WdMotorDriverGetBatteryLevel

概要

ミニ四駆駆動用バッテリ（単 3 乾電池 2 直列）の電圧を取得します。

引数

uint16_t *battery_mv

ミニ四駆用バッテリの電圧を mv で取得します。

戻り値

int8_t エラーコード

注意事項

この関数でバッテリ電圧を取得すには、最低 1 回以上 aiMini4WdMotorDriverSetDuty または、aiMini4WdMotorDriverBreak、aiMini4WdMotorDriverNeutral のいずれかを呼び出す必要があります。

使い方

motor_driver.h をインクルードしてください。

aiMini4WdSensorsGetCurrentImuData

概要

IMU から得られた最新のセンサ情報を取得します。

引数

AiMini4WdSensorsImuData *imu_data

IMU のデータを受け取るためのポインタ

戻り値

int8_t エラーコード

注意事項

使い方

sensors.h をインクルードしてください。

aiMini4WdSensorsGetCurrentMouseData

概要

光学マウスから得られた最新のセンサ情報を取得します。

引数

AiMini4WdSensorsMouseData *mouse_data

光学マウスのデータを受け取るためのポインタ

戻り値

int8_t エラーコード

注意事項

使い方

sensors.h をインクルードしてください。

aiMini4WdSensorsRegisterCapturedCallback

概要

オンボード IMU から値を取得するごとに発生するイベントを登録するための関数です。

引数

AiMini4WdSensorsCapturedCb　cb

。cb は、

```
typedef void (*AiMini4WdSensorsCapturedCb)( AiMini4WdSensorsData *sensor_Data);
```

で定義されるコールバック関数を指定します。

戻り値

int8_t エラーコード

注意事項

使い方

sensors.h をインクルードしてください。

aiMini4WdTimerRegister5msCb

概要

5ms 毎に発生するイベントを登録するための関数です。

引数

AiMini4WdTimerCb cb

。cb は、

```
typedef void (*AiMini4WdTimerCb)(void);
```

で定義されるコールバック関数を指定します。

戻り値

int8_t エラーコード

注意事項

使い方

timer.h をインクルードしてください。

aiMini4WdTimerRegister100msCb

概要

100ms 毎に発生するイベントを登録するための関数です。

引数

AiMini4WdTimerCb cb

。cb は、

```
typedef void (*AiMini4WdTimerCb)(void);
```

で定義されるコールバック関数を指定します。

戻り値

int8_t エラーコード

注意事項

使い方

timer.h をインクルードしてください。