PCL6046 で円弧補間を実行

Raspberry Pi PICO からの制御

ソースファイル解説

NPM 顧客「満足」から「感動」へ。
日本パルスモーター株式会社

目次

1. はじめに

弊社製 LSI の使用事例をご覧頂き、ありがとうございます。

本書は、PCL6046 による円弧補間動作を行うためのソースコードの解説をしたものです。

PCL6046 はパラレルバス制御の製品です。近年の CPU は、パラレルバスを持ったものは減りつつあり、比較的高価な製品に限られつつあるようです。

そのような中で、Raspberry Pi PICO にはパラレルバスを実装する機能があります。

ご存じの通り Raspberry Pi PICO は非常に安価な RP2040 マイコンで構成されています。

この安価な CPU から、PCL6046 を制御し、高精度な制御を行う事例を参考にしていただくよう、お願いいたします。

1.1 本書の取扱い

① 本書の全部または一部を無断で転載することは、著作権法によって禁止されています。

② 本書の内容については、性能や品質の向上に伴い、将来予告なく変更することがあります。

③ 本書の内容については、万全を期しておりますが、万一不可解な点や誤り、ならびに記載もれ等お気付きの点がありましたら、弊社営業担当へご連絡をお願いいたします。

1.1.1 記号説明

1.1.1.1 負傷レベル

本書では、次のように負傷レベルを定義します。

- 重傷

  失明、けが、火傷、感電、骨折、中毒等後遺症が残るもの、および治療に入院や長期の通院を要するもの。

- 軽傷

  治療に入院や長期の通院が必要ないもの。(上記「重傷」以外)

1.1.1.2 危険レベル

本製品は、運用者の安全を第一に考え、設計されています。しかし、製品の性質上、どうしても取除けないリスクが存在します。本書では、それらのリスクの重大性および危険性のレベルを、「危険」、「警告」および「注意」事項の 3 段階に分けて表示しています。表示項目をよく読み十分に理解してから、本製品の操作および保守作業を行ってください。

「危険」、「警告」および「注意」事項の表示は、危険性に関する重大性の順(危険＞警告＞注意)で、その内容を下記に説明します。

⚠ 危　険

「危険」項目は、本製品の運用中に、作業者が死亡または重傷に至る切迫した危険性のある場合について記述しています。

⚠ 警　告

「警告」項目は、本製品の運用中に、作業者が死亡または重傷を負う可能性のある場合について記述しています。

⚠ 注　意
「注意」項目は、本製品の運用中に、作業者が軽傷を負う可能性のある場合について記述しています。

注　意
⚠ (警告記号)のない「注意」項目は、作業者が負傷する恐れはないが、本製品、設備、機器等に損害や故障を引き起すことが予想される場合について記述しています。

本書では前述の危険レベル分けのほかに、下記の表記も使用しています。

重　要
「重要」項目は、本製品の操作および保守作業上、特に知っておかなければならない情報や内容がある場合に記述します。

備　考
「備考」項目は、本製品の操作および保守作業上、役立つ情報や内容がある場合に記述します。

1.1.1.3 警告図記号

本書では、「危険」、「警告」、「注意」、「重要」の表記に併せて次のようなシンボル記号を付加し、その警告内容をわかりやすく表現しています。

高電圧が印可される場合があることを表します。
安全確認を怠ったり、取扱いを誤ると感電によるショック、火傷、および死に至る危険を警告します。

表面温度が高くなる部品等があることを表します。
取扱いを誤ると、火傷の危険があることを意味します。

取扱いを誤ると、火災を起こす可能性があることを表します。

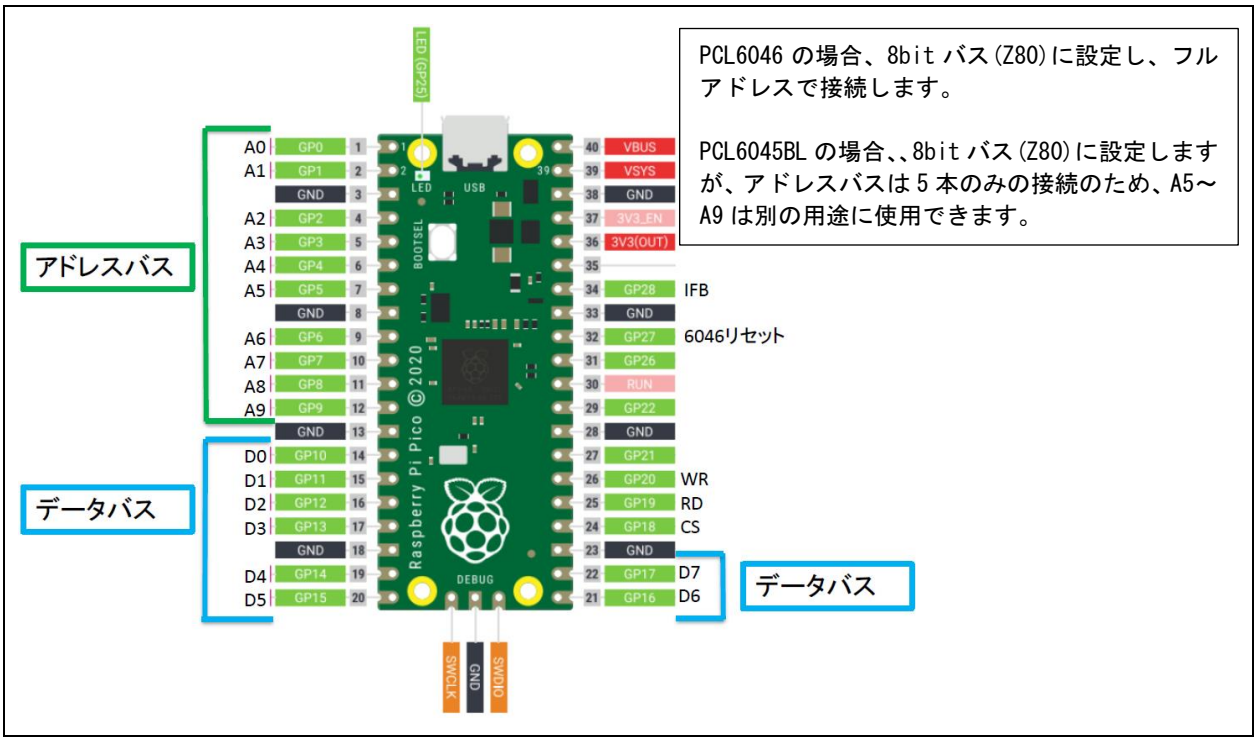
本製品の操作およびメンテナンス作業において、行ってはいけない「禁止」事項を示します。

本製品の操作およびメンテナンス作業において、必ず行っていただく「強制」事項を示します。

2. 概要

2.1 接続情報

Raspberry Pi PICO の端子は次のように使用します。



2.2 事例概要

Raspberry Pi PICO から PCL6046 を制御し、真円を描く動作を 1 回行います。

2.3 開発環境

2.3.1 ソフトウェア類

開発は、OS として Windows10 Pro を使用し、次のソフトウェアを使用しています。

名称	内容
ARM GCC Compiler	コンパイラ
CMake	ビルド自動化ツール
Visual Studio Code	エディタ
Build Tools for Visual Studio	ビルドツール
Python 3.9	Python の実行環境
Git for Windows	GitHub から SDK をダウンロードするツール

各ツール類の入手先やインストール手順、操作方法 当に関しての解説は省略します。

2.3.2 ハードウェア類

使用したハードウェアは次の通りです。

名称	販売元
Raspberry Pi Pico ボード	(秋月電子通商)
AD1431 (ドライバ) (TB67S249FTG ドライバモジュール」でも可)	(日本パルスモーター)
PFC30-24V4 (ステッピングモーター) (バイポーラ駆動なら他社製品でも可)	(日本パルスモーター) (秋月電子通商)
PCL6046 ピッチ変換基板 (PCL6045BL ピッチ変換基板でも可)	(日本パルスモーター／非買品) (日本パルスモーター)

3. ソースコード説明

ソースコードは次のファイルで構成されています。

ファイル名	内容
main.c	メインとなる動作のシーケンスを記載。
main.h	各種レジスタの定数を記載。
pcl_access.c	パラレルバスアクセス関数を記載。
pcl_access.h	パラレルバスアクセス関数の宣言を記載。 PCL のアドレス、書き込みデータを分割する際に使用する構造体を定義。
parallel_bus_module.pio	パラレルバスアクセスの機能を記載。
CmakeLists.txt	コンパイルの定義。
pico_sdk_import.cmake	コンパイルの定義。
stdafx.c	Raspberry Pi PICO 用のソース。
stdafx.h	Raspberry Pi PICO 用のソース。

3.1 Main.c

3.1.1 インクルードファイル

```
#include "stdafx.h"
#include "main.h"
#include "pcl_access.h"
```

3.1.2 関数

3.1.2.1 main

次の処理を行っています。

以降では、PCL6046 の制御に必要な関数の説明を行ってゆきます。

const PIO pio = pio0; const uint sm = 1;	
stdio_init_all();	USB ポートの初期化(Raspberry Pi PICO 用)。
sleep_ms(3000);	
pcl_bus_init(pio, sm, IFB, PIO_OUT_BASE, PIO_OUT_NUM, PIO_SIDE_BASE, PIO_SIDE_BASE_NUM, PIO_IN_BASE);	パラレルバスの初期化
pcl_reg_set(pio, sm);	各種レジスタ設定(X 軸,Y 軸)
org_run(pio, sm);	X 軸,Y 軸原点位置移動
arc_init_pos(pio, sm);	円弧補間初期位置へ移動
sleep_ms(1000);	
arc_run(pio, sm);	円弧補間スタート

3.1.2.2 pcl_reg_set

引数は次の通りです。

pio	使用する PIO 番号(PIO0 or PIO1)
sm	使用するステートマシン番号(0～3)

PCL6046 のレジスタへ、次のような初期設定を行います。

レジスタ	X 軸	Y 軸	内容
PRFL	32h	32h	初速度。
PRFH	190h	190h	最高速度。
PRUR	5AAh	5AAh	加速度。
PRDR	0h	0h	減速度(ゼロを設定した場合、加速度と同じ値が適用される)。
PRMG	12Bh	12Bh	速度倍率(今回は 1 倍)。
PRUS	0h	0h	S 字加減速動作時の、直線加速領域の定義。
PRDS	0h	0h	S 字加減速動作時の、直線減速領域の定義。
RFA	64h	64h	補助的な速度(円弧補間では未使用)。
PRMV	64h	64h	移動量。
PRIP	96h	0h	円弧補間の中心位置。
PRCI	0h	0h	円弧補間歩進数。
PRDP	0h	0h	スローダウンポイント。
PRMD	1Dh	1Dh	動作モード（マイナス方向への原点サーチ動作)。
RENV1	80000000h	80000000h	環境設定 1
RENV3	100000h	100000h	環境設定 2

レジスタへの初期設定後、X 軸、Y 軸のメインステータスを読み出し、UART へ出力します。

3.1.2.3 org_run

引数は次の通りです。

pio	使用する PIO 番号(PIO0 or PIO1)
sm	使用するステートマシン番号(0～3)

高速スタート 2 を X 軸、Y 軸に書き込むことで、動作を開始します。

直前の関数で ″マイナス方向への原点サーチ動作″ が設定されているため、この動作はセンサー原点位置まで移動となります。

3.1.2.4 arc_init_pos

引数は次の通りです。

pio	使用する PIO 番号(PIO0 or PIO1)
sm	使用するステートマシン番号(0～3)

PCL6046 のレジスタへ、次のような初期設定を行います。

レジスタ	X 軸	Y 軸	内容
PRMD	41h	41h	動作モード（位置決め動作)。
PRMV	32h	FAh	移動量。

レジスタの設定後、X 軸、Y 軸に高速スタート 2 コマンドを書き込みます。

次に、この動作が完了するまで待ちます。

動作完了後、X 軸、Y 軸のカウンター1 をクリアします。

3.1.2.5 arc_run

引数は次の通りです。

pio	使用する PIO 番号(PIO0 or PIO1)
sm	使用するステートマシン番号(0～3)

PCL6046 のレジスタへ、次のような初期設定を行います。

レジスタ	X 軸	Y 軸	内容
PRMD	64h	64h	動作モード（CW 方向の円弧補間動作)。
PRMV	0h	0h	移動量。

レジスタの設定後、X 軸、Y 軸に高速スタート 2 コマンドを書き込みます。

スタートにより動作が開始され、真円を描くような円弧補間が行われます。

3.1.2.6 msts_chk

引数は次の通りです。

pio	使用する PIO 番号(PIO0 or PIO1)
sm	使用するステートマシン番号(0～3)

X 軸、Y 軸のメインステータスを読み出す操作を、2 つの軸が動作を完了するまで繰り返します。

3.2 Main.h

3.2.1 マクロ定義

PIO_OUT_BASE	PCL アドレスバス A0 に接続する GPIO の番号を指定。
PIO_OUT_NUM	PCL アドレスバスとデータバスに使用する本数を指定。
PIO_SIDE_BASE	CS,WR,RD を割り当てるピンのベース番号を指定。
PIO_SIDE_BASE_NUM	CS,WR,RD に使用するピンの本数を指定。
PIO_IN_BASE	PCL データバス D0 に接続する GPIO の番号を指定。
IFB	PCL_IFB 信号読み込み端子を指定。

3.2.2 列挙型

Axis	PCL6046 の軸選択を定義。
Register	PCL6046 用のレジスタアドレス値を定義。
Command	PCL6046 用のコマンドの値を定義。

3.3 pcl_access.c

3.3.1 インクルードファイル

```
#include "stdafx.h"
#include "pcl_access.h"
#include "parallel_bus_module.pio.h"
```

3.3.2 関数

3.3.2.1 pcl_bus_dir_change

パラレルバスの入出力方向を切り替えます。

引数は次の通りです。

pio	使用する PIO 番号(PIO0 or PIO1) 。
sm	使用するステートマシン番号(0～3) 。

3.3.2.2 address_byte

アドレス値を 1byte ごとに分割します。

Raspberry Pi PICO でパラレルアクセスさせるための前処理です。

引数は次の通りです。

address	PCL6046 アクセス用のアドレス値。
pcl_address_bytes[]	バイト単位に分割されたアドレス値を格納する配列。

3.3.2.3 data_byte

書き込みデータを 1byte ごとに分割します。

Raspberry Pi PICO でパラレルアクセスさせるための前処理です。

引数は次の通りです。

data	PCL6046 書き込み用のデータ。
reg_data_bytes[]	バイト単位に分割されたデータを格納する配列。

3.3.2.4 write_tx_fifo

パラレルバスインターフェースのバッファにデータを書き込みます。

Raspberry Pi PICO でパラレルアクセスさせるための前処理です。

引数は次の通りです。

pio	使用する PIO 番号(PIO0 or PIO1) 。
sm	使用するステートマシン番号(0～3) 。
write_data	バッファに書き込むデータ。

3.3.2.5 read_rx_fifo

パラレルバスインターフェースのバッファからデータを読み出します。

Raspberry Pi PICO でパラレルアクセスさせた後の処理です。

引数は次の通りです。

pio	使用する PIO 番号(PIO0 or PIO1) 。
sm	使用するステートマシン番号(0～3) 。
write_data	ダミー(ゼロを指定) 。

戻り値は次の通りです。

uint32_t	読み出されたデータ。

3.3.2.6 pcl_bus_init

パラレルバスインターフェースを初期化します。

Raspberry Pi PICO でパラレルアクセスさせるための前処理です。

引数は次の通りです。

pio	使用する PIO 番号(PIO0 or PIO1) 。
sm	使用するステートマシン番号(0～3) 。
ifb	PCL6046 の IFB 信号読み込み端子。
pio_out_base	PCL アドレスバス A0 に接続する GPIO の番号。
pio_out_num	PCL アドレスバスとデータバスに使用する本数。
pio_side_base	CS,WR,RD を割り当てるピンの番号。
pio_side_num	CS,WR,RD に使用するピンの本数。
pio_in_base	PCL データバス D0 に接続する GPIO の番号。

3.3.2.7 write_reg_pcl

PCL6046 へレジスタデータを書き込みます。

引数は次の通りです。

pio	使用する PIO 番号(PIO0 or PIO1) 。
sm	使用するステートマシン番号(0～3) 。
ifb	PCL6046 の IFB 信号読み込み端子。
axs	書き込む軸。
address	書き込みアドレス。
data	書き込みデータ。

3.3.2.8 read_reg_pcl

PCL6046 からレジスタデータを読み出します。

引数は次の通りです。

pio	使用する PIO 番号(PIO0 or PIO1) 。
sm	使用するステートマシン番号(0～3) 。
ifb	PCL6046 の IFB 信号読み込み端子。
axs	書き込む軸。
address	書き込みアドレス。

戻り値は次の通りです。

uint32_t	読み出されたレジスタ値。

3.3.2.9 write_com_pcl

PCL6046 へコマンドを書き込みます。

引数は次の通りです。

pio	使用する PIO 番号(PIO0 or PIO1) 。
sm	使用するステートマシン番号(0～3) 。
ifb	PCL6046 の IFB 信号読み込み端子。
com_axis	書き込む軸。
address	書き込みアドレス。
com	動作コマンド。

3.3.2.10 read_msts_pcl

PCL6046 からメインステータスを読み出します。

引数は次の通りです。

pio	使用する PIO 番号(PIO0 or PIO1) 。
sm	使用するステートマシン番号(0～3) 。
ifb	PCL6046 の IFB 信号読み込み端子。
axs	書き込む軸。
address	書き込みアドレス。

戻り値は次の通りです。

uint32_t	読み出されたステータス値。

3.3.2.11 read_port_substat_pcl

PCL6046 から汎用ポートとサブステータスを読み出し、UART へ出力します。

引数は次の通りです。

pio	使用する PIO 番号(PIO0 or PIO1) 。
sm	使用するステートマシン番号(0～3) 。
ifb	PCL6046 の IFB 信号読み込み端子。
axs	書き込む軸。
address	書き込みアドレス。

戻り値はありません。

3.3.2.12 write_port_pcl

PCL6046 へ汎用出力ポートに書き込みます。

引数は次の通りです。

pio	使用する PIO 番号(PIO0 or PIO1) 。
sm	使用するステートマシン番号(0～3) 。
ifb	PCL6046 の IFB 信号読み込み端子。
axs	書き込む軸。
address	書き込みアドレス。
data	書き込みデータ。

3.4 pcl_access.h

3.4.1 マクロ定義

なし。

3.4.2 列挙型

address	アドレスデータを 1byte ごとに分割するために使用。
data	書き込みデータを 1byte ごとに分割するために使用。

3.5 parallel_bus_module.pio

3.5.1 関数

3.5.1.1 parallel_bus_module_init

パラレルバスインターフェースを初期化します。

引数は次の通りです。

pio	使用する PIO 番号(PIO0 or PIO1) 。
sm	使用するステートマシン番号(0～3) 。
offset	PIO プログラムの先頭アドレス
out_base	OUT 命令で出力するベースピン番号
out_pin_num	OUT 命令で出力するピンの本数(IN 命令の時はこのうち 8 本が IN 方向となる)
side_base	SIDE-SET 命令で使用するピンのベース番号
side_pin_num	SIDE-SET 命令で使用するピンの本数(CS と WR と RD 用)
in_base	IN 命令で使用するベースピン(PCL 読み出しデータの bit0)

弊社は、弊社ソフトウェアについて著作権を含む一切の知的所有権を保持します。弊社は、弊社ソフトウェアに関するいかなる権利もお客様に譲渡しません。お客様は、弊社の製品を使用する目的でのみ、現状有姿の弊社ソフトウェアを使用することができます。弊社は、弊社ソフトウェアの完全性、正確性、適用性、有用性、第三者知財の非侵害性を含め、明示たると黙示たるとを問わず何らの保証をいたしません。また、弊社ソフトウェアを使用したことで生じる損害（収入または利益の逸失を含む）について、一切の責任を負いません。お客様が、購入国以外で弊社ソフトウェアを使用する場合は、購入国と使用国の輸出管理法や規制を遵守する必要があります。

顧客「満足」から「感動」へ。
日本パルスモーター株式会社

www.pulsemotor.com

お問合せ
www.pulsemotor.com/support

東京 電話 03(3813)8841 FAX 03(3813)8550
大阪 電話 06(6576)8330 FAX 06(6576)8335
お電話受付時間 平日 9:00～17:00

2023 年 12 月発行