Firmware Manual

32

RZ/T1 グループ

速度プロファイル生成ライブラリ

ユーザーズマニュアル

ルネサスマイクロコンピュータ

RZファミリ RZ/T1シリーズ

Rev.1.02 2019.07

ルネサス エレクトロニクス
www.renesas.com

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準：　コンピュータ、OA機器、通信機器、計測機器、AV機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注1.　本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注2.　本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

(Rev.4.0-1　2017.11)

本社所在地

〒135-0061 東京都江東区豊洲3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS製品の取り扱いの際は静電気防止を心がけてください。CMOS製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS製品の入力がノイズなどに起因して、V_{IL}（Max.）からV_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）からV_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

このマニュアルの使い方

1. 目的と対象者

このマニュアルは、速度プロファイル生成ライブラリの機能および使用方法をユーザに理解していただくためのマニュアルです。本ライブラリを使用して、RZ/T1 ソリューションキットの応用例に基づいてモーション制御システムのアプリケーションを設計するユーザを対象にしています。このマニュアルを使用するには、C 言語およびアプリケーションプロジェクトのライブラリの使用に関する基本的な知識が必要です。

本マイコンは、注意事項を十分確認の上、使用してください。注意事項は、各章の本文中、各章の最後、注意事項の章に記載しています。

改訂記録は旧版の記載内容に対して訂正または追加した主な箇所をまとめたものです。改訂内容すべてを記録したものではありません。詳細は、このマニュアルの本文でご確認ください。

2. 略語および略称の説明

略語／略称	英語名	日本語名
ACIA	Asynchronous Communications Interface Adapter	調歩同期式通信アダプタ
bps	bits per second	転送速度を表す単位、ビット/秒
CRC	Cyclic Redundancy Check	巡回冗長検査
DMA	Direct Memory Access	CPU の命令を介さずに直接データ転送を行う方式
DMAC	Direct Memory Access Controller	DMA を行うコントローラ
GSM	Global System for Mobile Communications	FDD-TDMA の第二世代携帯電話の方式
Hi-Z	High Impedance	回路が電気的に接続されていない状態
IEBus	Inter Equipment bus	－
I/O	Input/Output	入出力
IrDA	Infrared Data Association	赤外線通信の業界団体または規格
LSB	Least Significant Bit	最下位ビット
MSB	Most Significant Bit	最上位ビット
NC	Non-Connection	未接続
PLL	Phase Locked Loop	位相同期回路
PWM	Pulse Width Modulation	パルス幅変調
SFR	Special Function Registers	周辺機能を制御するためのレジスタ
SIM	Subscriber Identity Module	ISO/IEC 7816 規定の接触型 IC カード
UART	Universal Asynchronous Receiver/Transmitter	調歩同期式シリアルインタフェース
VCO	Voltage Controlled Oscillator	電圧制御発振器

目次

1. はじめに

1.1 概要

速度プロファイル生成（VPG）ライブラリは、モーションコントロールアプリケーションの構築を支援する関数の集合体で、ルネサスの RZ/T1 ソリューションキットに付属のモーションコントロールファームウェアのリファレンスコードの関数を補完するものです。本ファームウェアは、デュアルチャネルの産業用サーボコントローラの応用例として設計されています。

1.2 機能

VPG ライブラリは、ある位置から別の位置への動作を実行するモータの目標位置、速度、加速度を表す一連のデータポイントを生成する関数を提供します。VPG 関数によって生成されたデータは、位置制御ループの設定値として利用されます。設定した目標位置と現在位置との差である位置誤差は、適切な補正出力を算出し、最終的に VPG の出力に応じた制御動作を生成する位置制御ループで使用されます。

VPG へのデータは、実行する動作の内容に応じてユーザが入力します。データは以下の通りです。

- 目標位置
- 最大速度
- 最大加速度
- 最大減速度

出力データは定期的に生成されます。位置制御ループが毎回実行するコマンドパラメータのセットが生成されます。データは以下の通りです。

- 指令位置
- 指令速度
- 指令加速度
- ステータス情報（加速、等速、減速、動作終了）

1.3　ソフトウェアの構成

VPG ライブラリは、RZ/T1 ソリューションキットファームウェアの一部となっています。以下に、VPG ライブラリの配置を示します。

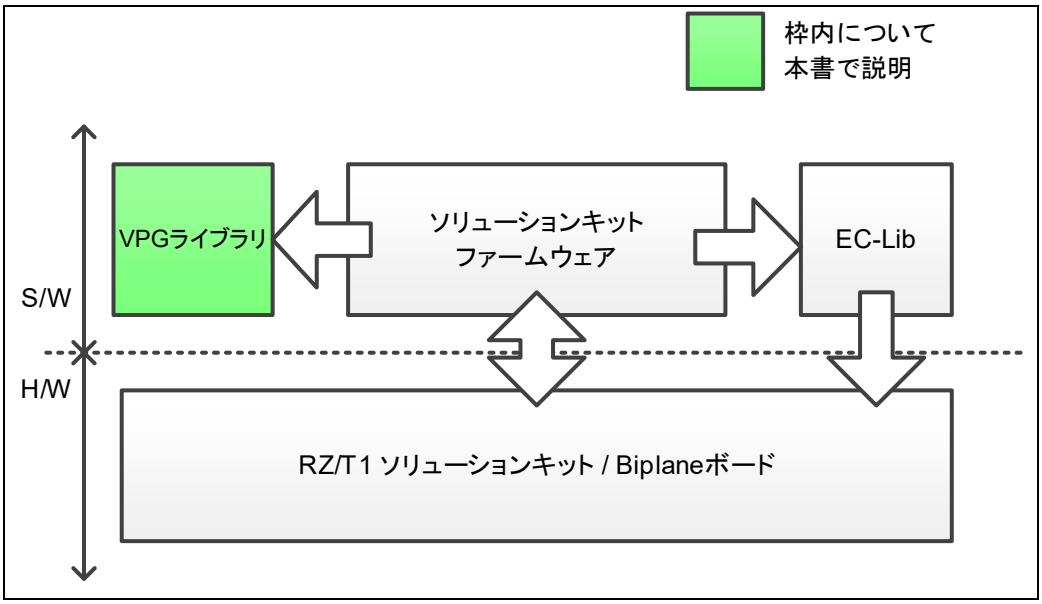


図1.1　VPGライブラリの配置

2. 動作環境

本マニュアルのライブラリは、下記の環境を想定しています。

表 2.1　動作環境

項目	内容
使用マイコン	RZ/T1 (Cortex®-R4)
動作周波数	600/450 MHz
動作電圧	供給電圧（入出力）：3.3V 供給電圧（内部）：1.2V
総合開発環境	ルネサスエレクトロニクス製 e^2 studio version 6.3.0 IAR システムズ製 Embedded Workbench® for Arm version 8.22.2

3. ファイル構成

以下に、VPG ライブラリのファイル構成を示します。

表 3.1　ファイル構成

ファイル	内容
inc\apl\m_common.h	ソリューションキットファームウェアのヘッダファイル
lib\gcc\libVPG.a	gcc 用 VPG ライブラリ
lib\iar\r_vpg.a	IAR 用 VPG ライブラリ

4. VPG ライブラリの API 関数

4.1 API 関数一覧

API 関数名	内容	ページ
vpg_spline_start	スプラインプロファイルを使用した VPG の初期化	15
vpg_spline_next	スプラインプロファイルを使用した 1 地点の生成	16
vpg_bezier_start	ベジェプロファイルを使用した VPG の初期化	18
vpg_bezier_next	ベジェプロファイルを使用した 1 地点の生成	20
vpg_pvt_start	位置・速度・時間（PVT）プロファイルを使用した VPG の初期化	22
vpg_pvt_next	PVT プロファイルを使用した 1 地点の生成	23

4.2 データ型

VPG ライブラリは、RZ/T1 ソリューションキットファームウェアの m_common.h ヘッダファイルを共有しています。ライブラリ関数は、16、32 ビット整数、倍精度の浮動小数点データ変数等の標準型を使用しています。

VPG ライブラリで使用する、新たに定義されたデータ型を以下に示します。

- t_vpg_state：速度プロファイルジェネレータの各状態の定数名の列挙型

4.3 データ構造

VPG 関数で使用されるデータ構造を以下に示します。

- TPosVel
- TMotionProfile
- TMotionParams
- t_motor

上記のデータ構造は **m_common.h** ヘッダファイルで定義されています。**t_motor** データ構造には多数のデータメンバがあります。VPG 関数の演算に関連するものを以下に示します。

TPosVel データメンバの説明

型	データメンバ	内容
long	position	エンコーダカウントの 32 ビットの位置値を格納する変数
long	velocity	固定小数点 (16.16) の 32 ビットの速度値を格納する変数

TMotionProfile データメンバの説明

型	データメンバ	内容
long	position	エンコーダカウントの 32 ビットの位置値を格納する変数
long	velocity	固定小数点 (16.16) の 32 ビットの速度値を格納する変数
long	acceleration	固定小数点 (16.16) の 32 ビットの加速度値を格納する変数
long	stop_dist	現在速度と指定最大減速度での算出された停止距離を格納する変数
t_vpg_state	vpg_state	速度プロファイルジェネレータの現在のステータス

TMotionParams データメンバの説明

型	データメンバ	内容
long	position	エンコーダカウントの 32 ビットの位置値を格納する変数
long	position_rel	現在位置に対する目標位置が指定されているときの、エンコーダカウントの 32 ビットの位置値を格納する変数
long	velocity	固定小数点 (16.16) の 32 ビットの速度値を格納する変数
long	acceleration	固定小数点 (16.16) の 32 ビットの加速度値を格納する変数
long	deceleration	固定小数点 (16.16) の 32 ビットの減速度値を格納する変数
long	accel_jerk	0～1000 までの係数で表される 32 ビットの加速度変化率を格納する変数
long	decel_jerk	0～1000 までの係数で表される 32 ビットの減速度変化率を格納する変数
short	dir_sign	実行する動作の方向を定義する変数
short	profile_mode	速度プロファイルジェネレータの使用する特定モードを定義するコード

PVT ベースの速度プロファイルジェネレータが使用する **t_motor** データメンバの説明

型	データメンバ	内容
TPosVel	pvt_start	最後のストリーミングリクエストの開始地点（位置と速度のセット）。PVT 指定動作を開始するときの位置（および速度）に設定されます。
short	pvt_points	バッファの PVT ポイント数
TPosVel *	pvt_pointsFIFO	ストリーム PVT ポイントを格納するバッファ
TPosVel *	pvt_push	次のストリーム PVT ポイントが格納されるバッファ位置。ホストコマンドハンドラ（ASCII またはパケット）によって更新されます。
TPosVel *	pvt_end_point	現在の補間周期で使用する目標位置／速度
TPosVel *	pvt_start_point	現在の補間周期で使用する開始位置／速度
double	f_vel0, f_vel1, f_time	補間アルゴリズムが使用するデータを一時的に保存する変数
short	pvt_step	現在の補間周期での現行ステップのカウンタ
unsigned short	pvt_period	1 補間周期の補間ステップ数を格納
unsigned short	pvt_watermark	FIFO バッファの容量不足を示す閾値を定義

スプライン曲線およびベジェ曲線ベースの速度プロファイルジェネレータが使用する **t_motor** データメンバの説明

型	データメンバ	内容
long	start_pos_accel	加速期間の開始位置
long	start_pos_decel	減速期間の開始位置
short	decel_started	減速し始めたどうかを示すフラグ
double	_fA0, _fA1, _fA2, _fA3, _fD1, _fD2, _fD3, _fD4, _fD5	加速および減速スプライン係数
double	max_vel_calc	指定されたモーションパラメータと目標位置に基づいて算出された最大速度
double	max_acc_calc	指定されたモーションパラメータと目標位置に基づいて算出された最大加速度
double	max_dec_calc	指定されたモーションパラメータと目標位置に基づいて算出された最大減速度
long	time_accel	加速期間の時間
long	time_decel	減速期間の時間
long	time_plateau	等速時間
long	time_current	タイムカウンタ
long	time_total	合計動作時間
long	dist_accel	加速期間の距離
long	dist_plateau	等速期間の距離
long	dist_decel	減速期間の距離

生成された位置および速度の指令値を格納するために速度プロファイルジェネレータが使用する **t_motor** データメンバの説明

型	データメンバ	内容
TMotionProfile *	prfl_push	新たに生成されたモーションパラメータへのポインタ
TMotionProfile *	prfl_push_1	最後に生成されたモーションパラメータへのポインタ
TMotionProfile *	profileFIFO	生成されたモーションパラメータを格納するバッファ
volatile short	prfl_points	profileFIFO バッファに格納されているポイントのカウンタ

4.4　マクロ／定数

以下の定数は、common.h ファイルで定義されています。変更する場合、ライブラリをリコンパイルする必要があります。

名称	値	内容
PVT_BUFF_SIZE	100	ストリーム PVT ポイントを格納するバッファのサイズを定義

4.5　エラーコード

特になし

5. APIリファレンス

5.1 APIリファレンスの読み方

API 名称	分類
機能概要	同期／非同期関数
形式	関数呼び出し形式を示します。#include “ヘッダファイル”で示すヘッダファイルは、この関数の実行に必要な標準ヘッダファイルであるため、必ずインクルードしてください。 I, O, IO は、それぞれ入力データ、出力データ、入出力データを意味します。
戻り値	関数の戻り値を示します。戻り値の後のコロンに続いて、値を返す条件などの説明を記述します。
説明	関数の仕様について説明します。
注意	注意事項があればここに示します。
使用例	関数の使用例があればここに示します。

5.2 vpg_spline_start

vpg_spline_start	分類
スプライン関数を使用した VPG の初期化	同期関数

形式

```
#include "m_common.h"
t_vpg_state  vpg_spline_start(t_motor *pm)
```

pm	モータ設定データ構造へのポインタ。入力パラメータは、参照先の t_motor 構造のデータメンバから取り込まれます。 **TMotionParams** trgtMotion：任意の速度プロファイルパラメータ（位置、速度、加速度、減速度）を保持します。 速度および加速度のデータ形式については、上記 **TMotionParams** の定義を参照してください。

戻り値

VPG_Acceleration	加速期間が開始します。本関数からの、通常の予期される応答です。
VPG_MotionCompleted	モーションプラニングを終了します。この値は、目標位置が現在位置と同じか非常に近い値のときにのみ返されます。

説明

スプラインベースの速度プロファイルの生成を初期化します。本関数は、スプライン曲線の係数を算出するためにデータメンバ trgtMotion を使用します。その後、vpg_spline_next()関数による速度プロファイル生成の時間（タイムスライス）ごとに評価されます。

本関数は、動作開始を要求する前に一度実行する必要があります。

計算結果は VPG 関数によって FIFO バッファに格納されます。これによって、位置制御ループからの VPG の非同期動作が可能になります。vpg_spline_start 関数は、FIFO バッファに対するプッシュ／プルポインタを初期化することで、その後それらをファームウェアが使用できるようにします。FIFO バッファの各要素は、上記のタイプ TMotionProfile のデータ構造です。FIFO バッファとそのプッシュ／プルポインタは、以下に示す t_motor データ構造のデータメンバです。

`profileFIFO[40]`：FIFO バッファ要素の記憶領域
`prfl_push_1`：最後に生成された設定値へのポインタ（最後の呼び出しで FIFO バッファにプッシュ）
`prfl_push`：現在生成されている設定値へのポインタ（現在 FIFO バッファにプッシュ）
`prfl_pull`：FIFO バッファからプルされる設定値へのポインタ
`prfl_points`：FIFO バッファに格納されているポイント数へのカウンタ

注意

本関数は、プロファイルの値の計算を行うための準備として、t_motor データ構造の以下のデータメンバを使用します。

```
time_current, time_plateau, time_total, time_decel
decel_started, direction, start_pos_accel, delta_pos,
max_vel_calc, max_dec_calc, pos_calc
_fA1, _fA2, _fA3, _dD3, _dD4, _dD5
```

使用例

```
pm->trgtMotion.position = 1000;
pm->trgtMotion.velocity = 600;
pm->trgtMotion.acceleration = 20;
pm->trgtMotion.deceleration = 20;
pm->vpg_state = vpg_spline_start(pm);
```

5.3 vpg_spline_next

vpg_spline_next	分類
スプラインプロファイルを使用した 1 地点の生成	同期関数

形式

```
#include "m_common.h"
t_vpg_state  vpg_spline_next(t_motor *pm, TMotionProfile *crntPars, TMotionProfile *cmdPars)
```

pm	モータ設定データ構造へのポインタ
crntPars	現行モーションプロファイルパラメータ（入力データ）へのポインタ
cmdPars	新規モーションプロファイルパラメータ（出力結果）へのポインタ

戻り値

なし	本関数は値を返しません。計算結果は、cmdPars 引数で指定された変数に格納されます。

計算結果がパラメータ cmdPars で指定されたデータ構造に格納されます。

cmdPars->position：設定される指令位置
cmdPars->velocity：設定される指令速度
cmdPars->acceleration：設定される指令加速度
cmdPars->vpg_state：新規 VPG ステータス（以下の定数のいずれか）

VPG_MotionCompleted	モーションプラニングの終了（結果は速度プロファイルの最後の地点）
VPG_Acceleration	速度プロファイルの加速実行期間
VPG_Deceleration	速度プロファイルの減速実行期間
VPG_Plateau	速度プロファイルの等速実行期間
VPG_Streaming	VPGはホストPCでストリーミングしたPVT設定値を使用

説明

本関数は、速度プロファイルの新規設定値を生成するために定期的に呼び出されます。プロファイルの生成はプロファイルの使用と切り離して行うため、結果は FIFO バッファに格納されます。このように、プロファイルは位置制御ループの動作に対して非同期に生成されます。

注意

新規速度プロファイルを生成する前に、最初に vpg_spline_start を呼び出す必要があります。そうしないと、vpg_spline_next は位置制御ループのタイムスライスごとに定期的に呼び出されません。

VPG_MotionCompleted を返すと、vpg_spline_next は呼び出されなくなります。

本関数は、中間変数の記憶領域として t_motor データ構造の以下のデータメンバを使用します。

```
time_current
pos_calc
stat_pos_decel
time_decel
time_total
decel_started
```

使用例

```
void vpg_update(t_motor *pm)
{
/* Generate one velocity profile point */
    vpg_spline_next(pm, pm->prfl_push_1, pm->prfl_push);
/* Move the FIFO pointer */
    pm->prfl_push_1 = pm->prfl_push;
    pm->prfl_push++;
    if (pm->prfl_push == &pm->profileFIFO[VPG_BUFF_SIZE])
        pm->prfl_push = &pm->profileFIFO[0];
    pm->prfl_points++;
}

int pos_loop_update()
{
    /* Implement velocity Profile Consumption */
    if (pm->vpg_state != VPG_MotionCompleted)
    {
        /* Check if new velocity / position setpoint is available */
        if (pm->prfl_points != 0)
        {
            pm->vpg_state = pm->prfl_pull->vpg_state;
            pm->cmd_pos = pm->prfl_pull->position;
            pm->cmd_vel = pm->prfl_pull->velocity;
            pm->cmd_acc = pm->prfl_pull->acceleration;

            pm->prfl_pull++;
            if (pm->prfl_pull == &pm->profileFIFO[VPG_BUFF_SIZE])
                pm->prfl_pull = &pm->profileFIFO[0];
            pm->prfl_points--;
        }
        return 1; /* FIFO starvation - handle error */
    }
    /* Invoke position loop PID regulator with the new point */
    pm->output_q = pid_calc(pm, pm->cmd_pos - pm->crnt_pos);
    return 0;
}
```

5.4 vpg_bezier_start

vpg_bezier_start		分類
ベジェ関数を使用した VPG の初期化		同期関数
形式	`#include "m_common.h"` `t_vpg_state  vpg_bezier_start(t_motor *pm)`	
	`pm`	モータ設定データ構造へのポインタ。入力パラメータは、参照先の t_motor 構造のデータメンバから取り込まれます。 **TMotionParams** trgtMotion：任意の速度プロファイルパラメータ（位置、速度、加速度、減速度）を保持します。 速度および加速度のデータ形式については、上記 **TMotionParams** の定義を参照してください。
戻り値	VPG_Acceleration	加速期間が開始されます。本関数からの、通常の予期される応答です。
	VPG_MotionCompleted	モーションプラニングを終了します。この値は、目標位置が現在位置と同じか非常に近い値のときにのみ返されます。
説明	ベジェ曲線ベースの速度プロファイルの生成を初期化します。本関数は、ベジェ曲線の係数を算出するデータメンバ trgtMotion を使用します。その後、vpg_bezier_next()関数による速度プロファイル生成の時間（タイムスライス）ごとに評価されます。 本関数は、動作開始を要求する前に一度実行する必要があります。 計算結果は VPG 関数によって FIFO バッファに格納されます。これによって、位置制御ループからの VPG の非同期動作が可能になります。vpg_bezier_start 関数は、FIFO バッファに対するプッシュ／プルポインタを初期化することで、その後それらをファームウェアが使用できるようにします。FIFO バッファの各要素は、上記のタイプ TMotionProfile のデータ構造です。FIFO バッファとそのプッシュ／プルポインタは、以下に示す t_motor データ構造のデータメンバです。 `profileFIFO[40]`：FIFO バッファ要素の記憶領域 `prfl_push_1`：最後に生成された設定値へのポインタ（最後の呼び出しで FIFO バッファにプッシュ） `prfl_push`：現在生成されている設定値へのポインタ（現在 FIFO バッファにプッシュ） `prfl_pull`：FIFO バッファからプルされる設定値へのポインタ `prfl_points`：FIFO バッファに格納されているポイント数のカウンタ	

注意	本関数は、プロファイルの値の計算を行うための準備として、t_motor データ構造の以下のデータメンバを使用します。
	`time_current`
	`decel_started, direction, delta_pos`
	`start_pos_accel, start_pos_decel`
	`max_vel_calc, max_dec_calc, pos_calc`
	`dist_accel, dist_decel`
	`_fA1, _fA2, _fA3, _dD1, _dD2, _dD3`	

使用例	pm->trgtMotion.position = 1000;
	pm->trgtMotion.velocity = 600;
	pm->trgtMotion.acceleration = 20;
	pm->trgtMotion.deceleration = 20;
	pm->trgtMotion.acc_jerk = 800;
	pm->trgtMotion.dec_jerk = 200;
	pm->vpg_state = vpg_bezier_start(pm);

5.5 vpg_bezier_next

vpg_bezier_next

分類

ベジェ曲線プロファイルを使用した１地点の生成　　同期関数

形式

```
#include “m_common.h”
t_vpg_state  vpg_bezier_next(t_motor *pm, TMotionProfile *crntPars, TMotionProfile *cmdPars)
```

pm	モータ設定データ構造へのポインタ
crntPars	現行モーションプロファイルパラメータ（入力データ）へのポインタ
cmdPars	新規モーションプロファイルパラメータ（出力結果）へのポインタ

戻り値

なし	本関数は値を返しません。計算結果は、cmdPars 引数で指定された変数に格納されます。

計算結果がパラメータ cmdPars で指定されたデータ構造に格納されます。
cmdPars->position：設定される指令位置
cmdPars->velocity：設定される指令速度
cmdPars->acceleration：設定される指令加速度
cmdPars->vpg_state：新規 VPG ステータス（以下の定数のいずれか）

VPG_MotionCompleted	モーションプラニングの終了（結果は速度プロファイルの最後の地点）
VPG_Acceleration	速度プロファイルの加速実行期間
VPG_Deceleration	速度プロファイルの減速実行期間
VPG_Plateau	速度プロファイルの等速実行期間
VPG_Streaming	VPGはホストPCでストリーミングしたPVT設定値を使用

説明

本関数は、速度プロファイルの新規設定値を生成するために定期的に呼び出されます。プロファイルの生成はプロファイルの使用と切り離して行うため、結果は FIFO バッファに格納されます。このように、プロファイルは位置制御ループの動作に対して非同期に生成されます。

注意

新規速度プロファイルを生成する前に、最初に vpg_bezier_start を呼び出す必要があります。そうしないと、vpg_bezier_next は位置制御ループのタイムスライスごとに定期的に呼び出されません。

VPG_MotionCompleted を返すと、vpg_bezier_next は呼び出されなくなります。

本関数は、中間変数の記憶領域として t_motor データ構造の以下のデータメンバを使用します。

```
time_current
pos_calc
stat_pos_decel
time_decel
time_total
decel_started
```

使用例

```
void vpg_update(t_motor *pm)
{
/* Generate one velocity profile point */
    vpg_bezier_next(pm, pm->prfl_push_1, pm->prfl_push);
/* Move the FIFO pointer */
    pm->prfl_push_1 = pm->prfl_push;
    pm->prfl_push++;
    if (pm->prfl_push == &pm->profileFIFO[VPG_BUFF_SIZE])
        pm->prfl_push = &pm->profileFIFO[0];
    pm->prfl_points++;
}

int pos_loop_update()
{
    /* Implement velocity Profile Consumption */
    if (pm->vpg_state != VPG_MotionCompleted)
    {
        /* Check if new velocity / position setpoint is available */
        if (pm->prfl_points != 0)
        {
            pm->vpg_state = pm->prfl_pull->vpg_state;
            pm->cmd_pos = pm->prfl_pull->position;
            pm->cmd_vel = pm->prfl_pull->velocity;
            pm->cmd_acc = pm->prfl_pull->acceleration;

            pm->prfl_pull++;
            if (pm->prfl_pull == &pm->profileFIFO[VPG_BUFF_SIZE])
                pm->prfl_pull = &pm->profileFIFO[0];
            pm->prfl_points--;
        }
        return 1; /* FIFO starvation - handle error */
    }
    /* Invoke position loop PID regulator with the new point */
    pm->output_q = pid_calc(pm, pm->cmd_pos - pm->crnt_pos);
    return 0;
}
```

5.6 vpg_pvt_start

vpg_pvt_start		分類
PVT 補間関数による VPG の初期化		同期関数
形式	`#include "m_common.h"` `t_vpg_state  vpg_pvt_start(t_motor *pm)`	
	pm	モータ設定データ構造へのポインタ。入力パラメータは、参照先の t_motor 構造のデータメンバから取り込まれます。 **TMotionParams** trgtMotion：任意の速度プロファイルパラメータ（位置、速度、加速度、減速度）を保持します。 速度および加速度のデータ形式については、上記 **TMotionParams** の定義を参照してください。
戻り値	VPG_Acceleration	加速期間が開始されます。本関数からの、通常の予期される応答です。
	VPG_MotionCompleted	モーションプラニングを終了します。この値は、目標位置が現在位置と同じか非常に近い値のときにのみ返されます。
説明	ホスト定義の速度プロファイルの生成を初期化します。ホストプロファイルは、一定時間間隔の一連の位置・速度セットとしてストリームされます。 本関数はデータメンバを使用します。 本関数は、動作開始を要求する前に一度実行する必要があります。 計算結果は VPG 関数によって FIFO バッファに格納されます。これによって、位置制御ループからの VPG の非同期動作が可能になります。vpg_pvt_start 関数は、FIFO バッファに対するプッシュ／プルポインタを初期化することで、その後それらをファームウェアが使用できるようにします。FIFO バッファの各要素は、上記のタイプ TMotionProfile のデータ構造です。FIFO バッファとそのプッシュ／プルポインタは、以下に示す t_motor データ構造のデータメンバです。 `profileFIFO[40]`：FIFO バッファ要素の記憶領域 `prfl_push_1`：最後に生成された設定値へのポインタ（最後の呼び出しで FIFO バッファにプッシュ） `prfl_push`：現在生成されている設定値へのポインタ（現在 FIFO バッファにプッシュ） `prfl_pull`：FIFO バッファからプルされる設定値へのポインタ `prfl_points`：FIFO バッファに格納されているポイント数のカウンタ	

注意	本関数は、プロファイルの値の計算を行うための準備として、t_motor データ構造の以下のデータメンバを使用します。
	`time_current`
	`_fA1, _fA2, _fA3, _dD1, _dD2, _dD3`	

使用例	`pm->vpg_state = vpg_pvt_start(pm);`	

5.7 vpg_pvt_next

vpg_pvt_next	分類
PVT 設定値を補間する 1 地点の生成	同期関数

形式

```
#include "m_common.h"
t_vpg_state  vpg_pvt_next(t_motor *pm, TMotionProfile
*crntPars, TMotionProfile *cmdPars)
```

pm	モータ設定データ構造へのポインタ
crntPars	現行モーションプロファイルパラメータ（入力データ）へのポインタ
cmdPars	新規モーションプロファイルパラメータ（出力結果）へのポインタ

戻り値

なし	本関数は値を返しません。計算結果は、cmdPars 引数で指定された変数に格納されます。

計算結果がパラメータ cmdPars で指定されたデータ構造に格納されます。

cmdPars->position：設定される指令位置
cmdPars->velocity ：設定される指令速度
cmdPars->vpg_state：新規 VPG ステータス（以下の定数のいずれか）

VPG_MotionCompleted	モーションプラニングの終了（結果は速度プロファイルの最後の地点）
VPG_Streaming	VPGはホストPCでストリーミングしたPVT設定値を使用

説明

本関数は、速度プロファイルの新規設定値を生成するために定期的に呼び出されます。プロファイルの生成はプロファイルの使用と切り離して行うため、結果は FIFO バッファに格納されます。このように、プロファイルは位置制御ループの動作に対して非同期に生成されます。

注意

新規速度プロファイルを生成する前に、最初に **vpg_pvt_start** を呼び出す必要があります。そうしないと、**vpg_pvt_next** は位置制御ループのタイムスライスごとに定期的に呼び出されません。

VPG_MotionCompleted を返すと、vpg_pvt_next は呼び出されなくなります。

本関数は、中間変数の記憶領域として t_motor データ構造の以下のデータメンバを使用します。

```
time_current
pos_calc
```

使用例

```
void vpg_update(t_motor *pm)
{
/* Generate one velocity profile point */
    vpg_pvt_next(pm, pm->prfl_push_1, pm->prfl_push);
/* Move the FIFO pointer */
    pm->prfl_push_1 = pm->prfl_push;
    pm->prfl_push++;
    if (pm->prfl_push == &pm->profileFIFO[VPG_BUFF_SIZE])
        pm->prfl_push = &pm->profileFIFO[0];
    pm->prfl_points++;
}

int pos_loop_update()
{
    /* Implement velocity Profile Consumption */
    if (pm->vpg_state != VPG_MotionCompleted)
    {
        /* Check if new velocity / position setpoint is available */
        if (pm->prfl_points != 0)
        {
            pm->vpg_state = pm->prfl_pull->vpg_state;
            pm->cmd_pos = pm->prfl_pull->position;
            pm->cmd_vel = pm->prfl_pull->velocity;
            pm->cmd_acc = pm->prfl_pull->acceleration;

            pm->prfl_pull++;
            if (pm->prfl_pull == &pm->profileFIFO[VPG_BUFF_SIZE])
                pm->prfl_pull = &pm->profileFIFO[0];
            pm->prfl_points--;
        }
        return 1; /* FIFO starvation - handle error */
    }
    /* Invoke position loop PID regulator with the new point */
    pm->output_q = pid_calc(pm, pm->cmd_pos - pm->crnt_pos);
    return 0;
}
```

6. リソース

6.1 ハードウェア

VPG ライブラリはハードウェアリソースを必要としません。

6.2 オペレーティングシステム

VPG ライブラリはオペレーティングシステムに依存しません。

6.3 メモリ

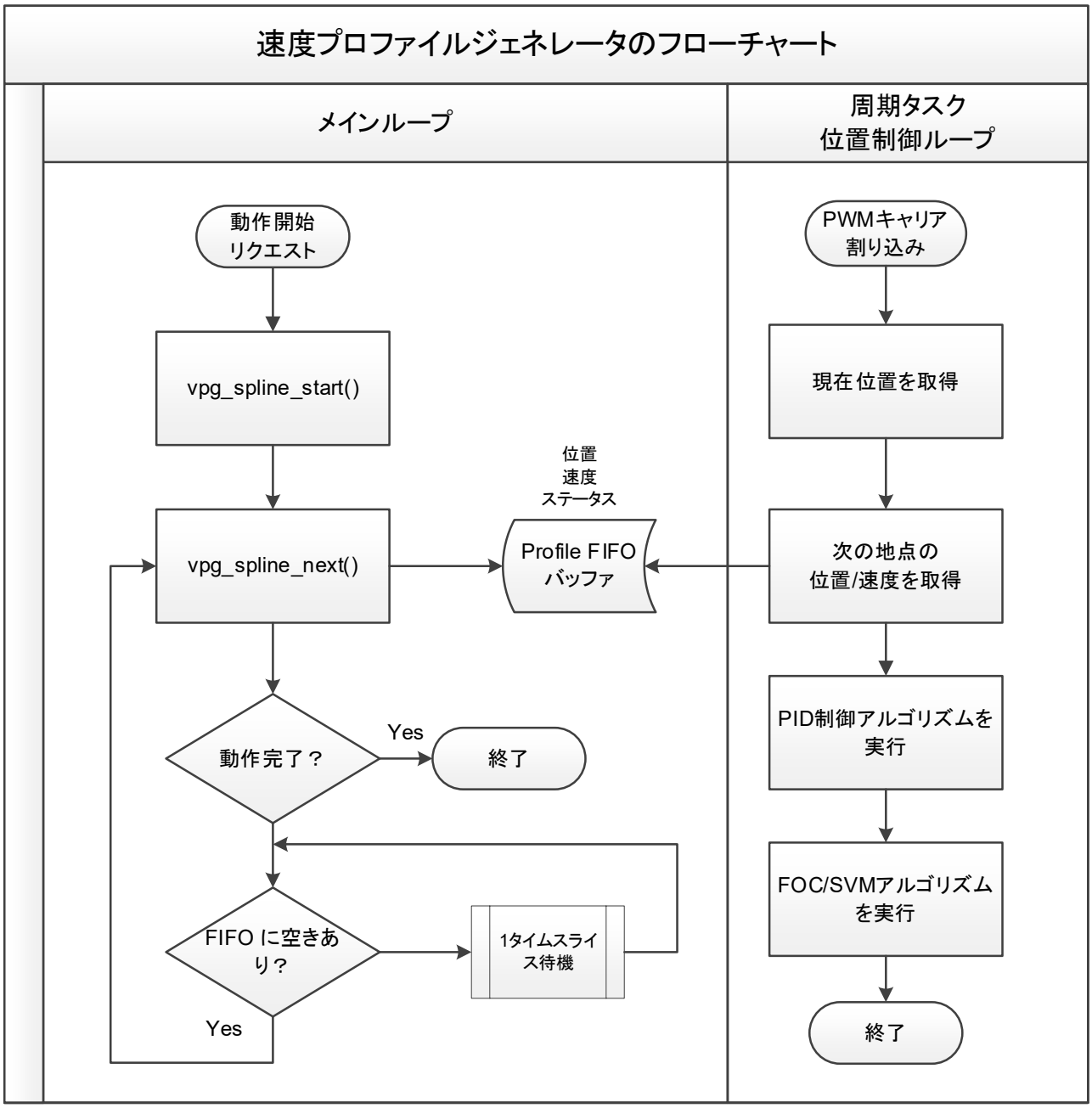
以下に、VPG ライブラリのメモリ要件を示します。

コードサイズ：3282 バイト

データサイズ：ホストアプリケーションは、FIFO バッファと VPG 関連変数にデータメモリを割り当てる必要があります。大半のアプリケーションにとって十分なデータメモリの標準的な容量は、1 軸あたり約 2600 バイトです。

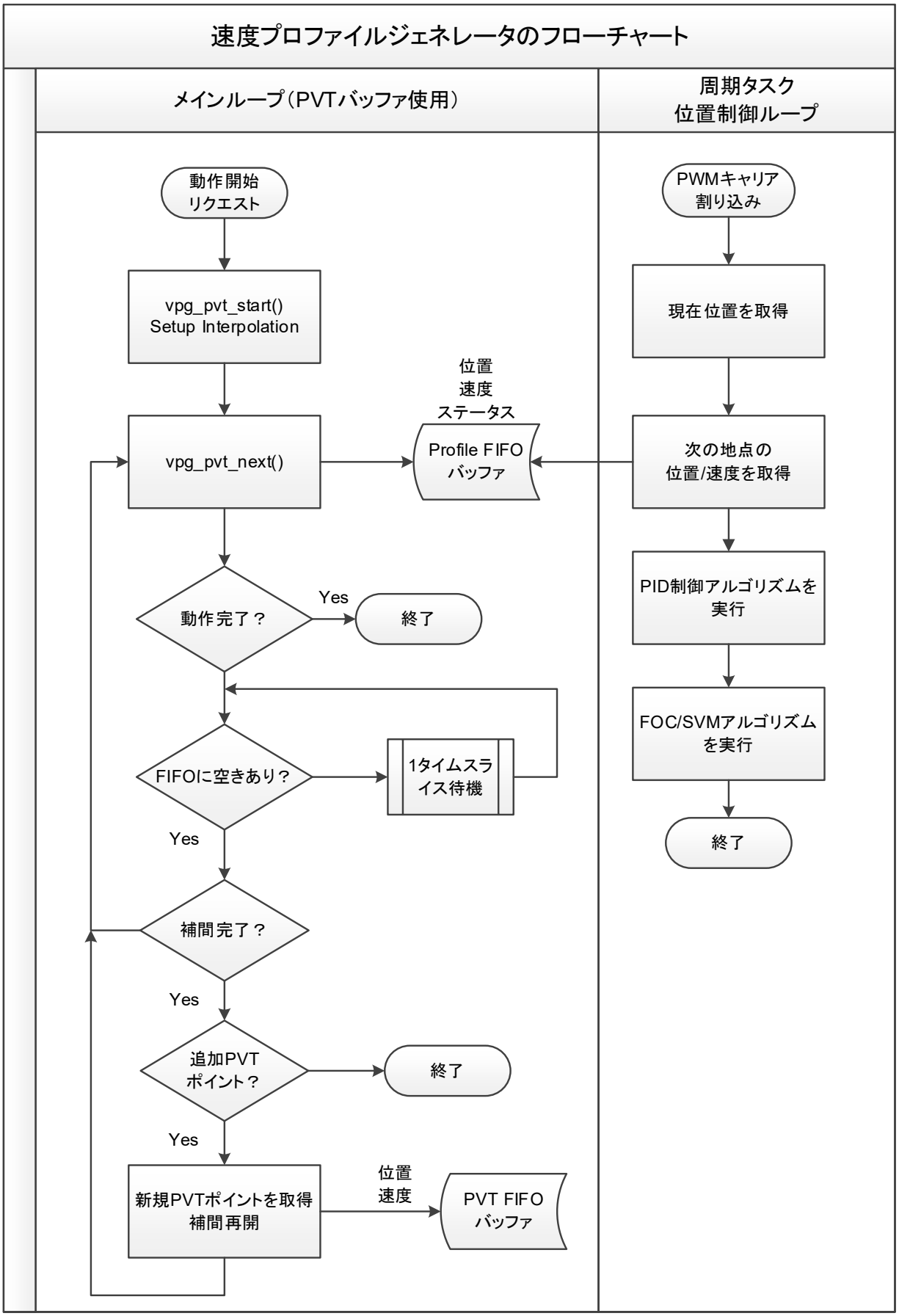
7. フローチャート

VPG ライブラリの一般的な使用方法を、以下のフローチャートに示します。目標位置への制御動作を実行するために、2 つのスレッドを使用します。一つはモーションコマンドに応答して速度プロファイルを生成し、もう一つは生成されたデータを使用して処理を行います。



上記の例では、スプラインベースの VPG 関数を参照しています。速度プロファイル生成の PVT ベースモードを除く、他のタイプの VPG を使用する場合も同じです。

ホストコンピュータから入力された PVT FIFO バッファの使用に必要な追加の手順を、以下のフローチャートに示します。これによって、完全な速度プロファイルが生成されます。


速度プロファイルジェネレータのフローチャート
メインループ（PVTバッファ使用）
周期タスク
位置制御ループ
動作開始
リクエスト
vpg_pvt_start()
Setup Interpolation
vpg_pvt_next()
位置
速度
ステータス
Profile FIFO
バッファ
動作完了？
Yes
終了
FIFOに空きあり？
1タイムスライス待機
Yes
補間完了？
Yes
追加PVT
ポイント？
終了
Yes
新規PVTポイントを取得
補間再開
位置
速度
PVT FIFO
バッファ
PWMキャリア
割り込み
現在位置を取得
次の地点の
位置/速度を取得
PID制御アルゴリズムを
実行
FOC/SVMアルゴリズム
を実行
終了

8. 関連ドキュメント

RZ/T1 グループ ソリューションキットファームウェア アプリケーションノート (R11AN0086JJ)

改訂記録	RZ/T1 グループ 速度プロファイル生成ライブラリ ユーザーズマニュアル

Rev.	発行日	改訂内容	
		ページ	ポイント
1.01	2018.08.21	—	初版発行
1.02	2019.07.31	—	製品ご使用上の注意事項　登録商標に関する文言を追加
		9	表 2.1 動作環境　統合開発環境に e^2 studio を追加、EWARM のバージョンを変更（7.80→8.22.2）
		10	表 3.1 ファイル構成　gcc 用 VPG ライブラリを追加、IAR 用 VPG ライブラリのファイル名を変更
		12, 13	4.3 データ構造　表「TMotionProfile データメンバの説明」：型名を変更（short→t_vpg_state）、表「スプライン曲線およびベジェ曲線ベースの速度プロファイルジェネレータが使用する t_motor データメンバの説明」：データメンバ名を修正

RZ/T1 グループ 速度プロファイル生成ライブラリ
ユーザーズマニュアル

発行年月日　2018年08月21日　Rev.1.01
　　2019年07月31日　Rev.1.02

発行　ルネサス エレクトロニクス株式会社
　　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

RZ/T1グループ

RENESAS

ルネサス エレクトロニクス株式会社

R11UZ0004JJ0102