ブラウザによるサーボモータ制御

STEP 12

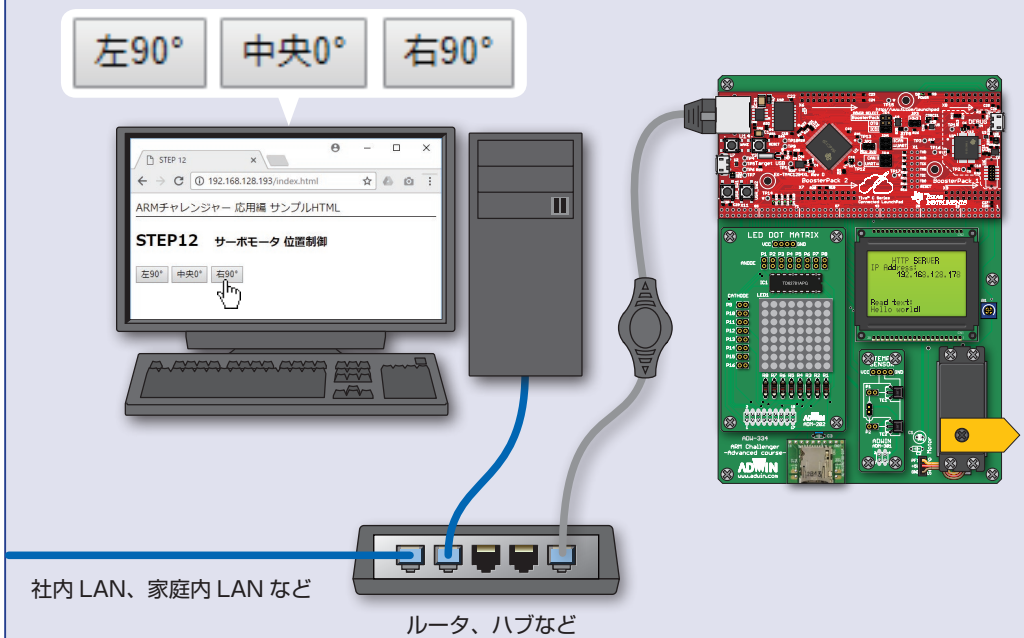
学習内容

パソコンのブラウザからマイコンの I/O 制御を行います。
ブラウザとマイコンのやり取りは STEP10 と同じです。
本 STEP ではサーボモータを制御するため、PWM 出力の方法を学習します。

課題 12

ブラウザ上に「左 90°」「中央 0°」「右 90°」ボタンを表示し、それらから以下のような操作をできるようにしましょう。

「左 90°」ボタンをクリックサーボモータが左 90 度に回転
「中央 0°」ボタンをクリックサーボモータが中央位置に回転
「右 90°」ボタンをクリックサーボモータが右 90 度に回転



⚠ 中央 0° に動かして、矢印板がずれている場合は、矢印版を止めているねじを緩めて調節し止め直してください。

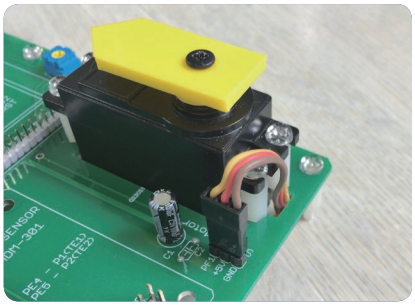
ブラウザによるサーボモータ制御

配線 12

マイコンボードとサーボモータを接続を確認しておきましょう。

マイコンボード（メインボード上のサーボモータ接続ピン）とサーボモータは、以下のように接続します。サーボモータの制御方法について詳しくは、巻末に掲載している取扱説明書をご覧ください。

マイコンボード		サーボモータ	
PF1	⟷	PWM	橙
+5V	⟷	VCC	赤
GND	⟷	GND	茶



⚠ サーボモータの動作中は、電圧が下がって GLCD や LED ドットマトリックスが一時的に暗くなることがあります。

ブラウザによるサーボモータ制御

PWM によるサーボモータの制御

本コースで用いるサーボモータは、PWM 信号のパルス幅で角度を制御します。

パルス幅が 0.7 ～ 2.3 ミリ秒の信号を与えることで、－ 90° から +90° までの 180° 回転させることができます（0° ＝中央位置の場合はパルス幅を 1.5 ミリ秒にします）。また、PWM の周期は 10 ～ 20 ミリ秒である必要があります※1。

詳細は、サーボモータの取扱説明書（p.275）をご覧ください。

PWM の設定

TI-RTOS には PWM を扱うためのドライバが備わっています。ただし、サンプルプログラムの初期設定で PWM ピンとして使うことができるのは PF0 ピン※2 のみです。ここではまず、PF1 ピンを PWM ピンに追加してきましょう。これらの設定は、STEP 10 での GPIO ピンの追加と同様に、プロジェクトディレクトリ直下の「EK_TM4C1294XL.c」等を編集することで行います。

コーディング Board.h 12

まずは、ピン名の指定も兼ねて「Board.h」でピン名の定義を行っておきましょう。

以下は、Board.h への追記例です。■は STEP 10 の Board.h 10 から追加した部分です。ここでは、ピン名を「Servo_PWM」とすることにします。

Board.h | EK_TM4C1294.h | EK_TM4C1294.c | main.c

これ以前の行に変更はありません

```
85  #define Board_PWM0                  EK_TM4C1294XL_PWM0
86  #define Board_PWM1                  EK_TM4C1294XL_PWM0
87  #define Servo_PWM                   Servo_PWM
88
89  #define Board_SDSPI0                EK_TM4C1294XL_SDSPI0
90  #define Board_SDSPI1                EK_TM4C1294XL_SDSPI1
```

これ以降の行に変更はありません

※1　角度の制御は、PWM の周期にかかわらず、パルス幅 0.7 ～ 2.3 ミリ秒で行います。

※2　PF0 ピンはマイコンボード上で LED(D4)に接続されています。ただし､本コースでは PF0 ピンおよび D4 はイーサネットの監視用に使用済みです。

ブラウザによるサーボモータ制御

コーディング EK_TM4C1294.h 12

EK_TM4C1294.h に、先ほど定義したピン名を追加しましょう。

以下は、EK_TM4C1294 .h への追記例です。■は STEP 10 の EK_TM4C1294 .h 10 から追加した部分です。

Board.h | EK_TM4C1294.h | EK_TM4C1294.c | main.c

これ以前の行に変更はありません

```
107 /*!
108  *  @def    EK_TM4C1294XL_PWMName
109  *  @brief  Enum of PWM names on the EK_TM4C1294XL dev board
110  */
111 typedef enum EK_TM4C1294XL_PWMName {
112     EK_TM4C1294XL_PWM0 = 0,
113     Servo_PWM,
114
115     EK_TM4C1294XL_PWMCOUNT
116 } EK_TM4C1294XL_PWMName;
```

これ以降の行に変更はありません

コーディング EK_TM4C1294.c 12

EK_TM4C1294.c に、サーボモータ用 PWM ピン（PF1/MOPWM1）の設定を追加しましょう。設定の追加は、既に設定されている PF0/MOPWM0 の設定をコピー・修正すると良いでしょう。

以下は、EK_TM4C1294 .c への追記例です。■は STEP 10 の EK_TM4C1294 .c 10 から追加した部分です。

Board.h | EK_TM4C1294.h | EK_TM4C1294.c | main.c

これ以前の行に変更はありません

```
425 /*
426  *  ============================= PWM ==============================
427  */
428 /* Place into subsections to allow the TI linker to remove items properly */
429 #if defined(__TI_COMPILER_VERSION__)
430 #pragma DATA_SECTION(PWM_config, ".const:PWM_config")
431 #pragma DATA_SECTION(pwmTivaHWAttrs, ".const:pwmTivaHWAttrs")
432 #endif
433
434 #include <ti/drivers/PWM.h>
435 #include <ti/drivers/pwm/PWMTiva.h>
436
437 PWMTiva_Object pwmTivaObjects[EK_TM4C1294XL_PWMCOUNT];
438
439 const PWMTiva_HWAttrs pwmTivaHWAttrs[EK_TM4C1294XL_PWMCOUNT] = {
440     {
441         .baseAddr = PWM0_BASE,
442         .pwmOutput = PWM_OUT_0,
443         .pwmGenOpts = PWM_GEN_MODE_DOWN | PWM_GEN_MODE_DBG_RUN
444     },
445     // Servo_PWM 用
446     {
447         .baseAddr = PWM0_BASE,
448         .pwmOutput = PWM_OUT_1,
449         .pwmGenOpts = PWM_GEN_MODE_DOWN | PWM_GEN_MODE_DBG_RUN
450     }
451 };
452
453 const PWM_Config PWM_config[] = {
454     {
455         .fxnTablePtr = &PWMTiva_fxnTable,
456         .object = &pwmTivaObjects[0],
457         .hwAttrs = &pwmTivaHWAttrs[0]
458     },
459     // Servo_PWM 用
460     {
461         .fxnTablePtr = &PWMTiva_fxnTable,
462         .object = &pwmTivaObjects[1],
```

ブラウザによるサーボモータ制御

コーディング EK_TM4C1294.c 12

```
463         .hwAttrs = &pwmTivaHWAttrs[1]
464     },
465     {NULL, NULL, NULL}
466 };
467
468 /*
469  *  ======== EK_TM4C1294XL_initPWM ========
470  */
471 void EK_TM4C1294XL_initPWM(void)
472 {
473     /* Enable PWM peripherals */
474     SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);
475
476     /*
477      * Enable PWM output on GPIO pins.  PWM output is connected to an Ethernet
478      * LED on the development board (D4).  The PWM configuration
479      * below will disable Ethernet functionality.
480      */
481     GPIOPinConfigure(GPIO_PF0_M0PWM0);
482     GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_0);
483     // Servo_PWM 用
484     GPIOPinConfigure(GPIO_PF1_M0PWM1);
485     GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_1);
486
487     PWM_init();
488 }
```

これ以降の行に変更はありません

なお、pwmTivaHWAttrs[] 中の .pwmGenOpts では PWM ジェネレータのオプションが設定されています。ここでの設定値は以下のとおりです。

- PWM_GEN_MODE_DOWN　　：ダウンカウントモードで使用
- PWM_GEN_MODE_DBG_RUN：デバッグモードでも PWM を継続

コーディング index.htm 12

以下は、index.htm 11-2 を元にしたコーディング例です。ここでは、「control.cgi」に POST メソッドの本文でコマンド「servo_position= 角度」を送ってサーボモータの角度を制御することにしています（角度は中心が 0 度で時計回りを正としています）。■は index.htm 11-2 から追加した部分です（■は課題名の変更です）。

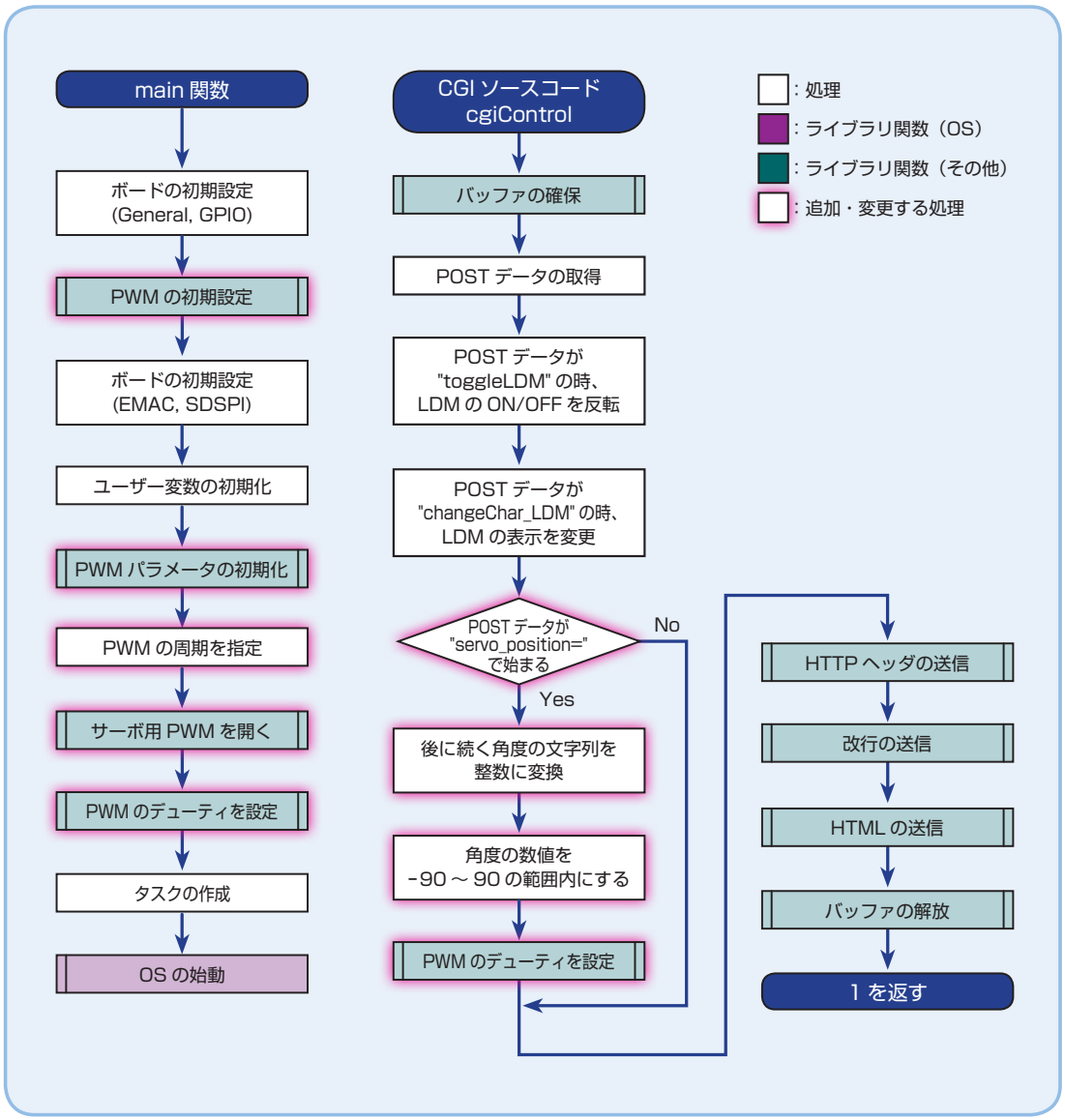
index.htm

```
<!DOCTYPE html>
<html>
<head>
<meta charset='Shift_JIS'>
<title>STEP 12</title>
   .
   .
   .
<body onload="do_periodic()">
        <h1>ARM チャレンジャー 応用編 サンプル HTML</h1>

        <h2> STEP10-3 <span class="exp"> ドットマトリックス LED 文字 A/B 変更 </span> </h2>
        <input type="button" value=" トグル LED" onclick='control_onclick("toggle_LDM")'>
        <input type="button" value=" 文字変更 " onclick='control_onclick("changeChar_LDM")'>

        <h2> TEP11-1 <span class="exp"> スイッチ 入力取得 </span> </h2>
        <input id="text_sw" type="text">

        <h2> STEP11-2 <span class="exp"> スイッチ 入力取得 </span> </h2>
        <span id="sw1" class="sw">SW1</span>
        <span id="sw2" class="sw">SW2</span>

        <h2> TEP12 <span class="exp"> サーボモータ 位置制御 </span> </h2>
        <input type="button" value=" 左 90° " onclick='control_onclick("servo_position=-90")'>
        <input type="button" value=" 中央 0° " onclick='control_onclick("servo_position=0")'>
        <input type="button" value=" 右 90° " onclick='control_onclick("servo_position=90")'>
</body>

</html>
```

ブラウザによるサーボモータ制御

フローチャート 12

以下は、課題 12 を実現するためのフローチャート例です。PWM の初期設定は、イーサネットの監視用の使っている PF4 ピン（D4）の機能を PWM 用に上書きしてしまうので、イーサネット監視用としての用途を優先する場合は PWM の初期設定の後に EMAC の初期設定を行う必要があります※。また、サーボモータの動作範囲は－90 度～ 90 度なので、この範囲を超える値が来た場合には範囲内に収めるようにしています。その他の関数・タスクは「フローチャート 11-2」と同じです。



※ PF4 ピンの機能が PWM 用に上書きされても、監視用の D4 が点灯しなくなるだけでイーサネットの動作自体には問題はありません。

インクルードファイル 12

課題 12 で使用するインクルードファイルを解説します。

なお、インクルードされているファイルは、テキストカーソルをファイル名部分に合わせて「F3」キーを押すことで開くことができます。

標準ヘッダファイル　stdlib.h

```
#include <stdlib.h>
```

C 言語の標準ヘッダファイルのひとつです。本課題で使用する関数のうち、以下のものが定義されています。

- atoi(const char *_st)

ドライバヘッダファイル　ti/drivers/PWM.h

```
#include <ti/drivers/PWM.h>
```

TI-RTOS のドライバヘッダファイルで、PWM を扱うための関数等が定義されています。本課題で使用する型（構造体）、関数のうち、以下のものが定義されています。

- PWM_Handle
- PWM_Params
- PWM_Params_init()
- PWM_open()
- PWM_setDuty()

ブラウザによるサーボモータ制御

型・構造体 12

課題 12 で使用する型・構造体を解説します。

なお、テキストカーソルをソースコード中の型や構造体の部分に合わせると、その型・構造体に関する情報がポップアップされます。さらに「F3」キーを押すと、その型・構造体が定義されているファイルを開くことができます。

PWM 関係の型・構造体

PWM 関係の型・構造体は「ti/drivers/PWM.h」で定義されています。

詳細は CCS の「Help」→「Help Contents」から「TI-RTOS for TivaC *」→「Documentation Links」→「Drivers Documentation」→「TI-RTOS Drivers Runtime APIs (doxygen)」→「PWH.h」をご覧ください。

PWM のハンドラ　PWM_Handle

PWM 出力を扱うためのハンドラです。

PWM のパラメータ　PWM_Params

PWM の初期設定を行うための構造体です。主なメンバ変数は以下のとおりです。

- period　：PWM の周期を指定（マイクロ秒で指定）
- duty　：デューティの設定モードを指定
- polarity　：PWM の極性を指定

ライブラリ関数 12

課題 12 で使用する関数を解説します。

なお、テキストカーソルをソースコード中の関数の部分に合わせると、その関数に関する情報がポップアップされます。さらに「F3」キーを押すと、その関数が定義されているファイルを開くことができます。

C 言語標準

文字列の比較　int strncmp(const char *string1, const char *string2, size_t n)

string.h※で定義されている C 言語標準の関数で、文字列 string1 と文字列 string2 の最初の n 文字を比較し、同じ文字列の場合は 0 を、そうでない場合は正または負の値を返します。以下は、文字列 buffer が「"servo_position="」で始まる場合に処理を実行する例です。

```
if(strncmp(buffer,"servo_position=", 15)==0) { /* 処理 */ }
```

文字列の整数への変換　int atoi(const char *_st)

stdlib.h で定義されている C 言語標準の関数で、文字列 _st で表されている数値を整数に変換します。以下は、文字列 buffer の 15 文字目から数値が始まっている場合に、その数値を整数に変換し整数型変数 degree に代入する例です。

```
int degree = atoi(buffer+15);
```

ボード関係

PWM の初期設定　void Board_initPWM(void)

PWM の初期設定を行います。実体は EK_TM4C1294XL.c にある EK_TM4C1294XL_initPWM() です。

※　ヘッダファイル string.h は ti/ndk/inc/netmain.h をインクルードすることで自動的にインクルードされます。

ブラウザによるサーボモータ制御

ライブラリ関数 12

PWM 関係

PWM 関係の関数は「ti/drivers/PWM.h」で定義されています。

各関数の詳細は CCS の「Help」→「Help Contents」から「TI-RTOS for TivaC *」→「Documentation Links」→「Drivers Documentation」→「TI-RTOS Drivers Runtime APIs (doxygen)」→「PWH.h」をご覧ください。

PWM のパラメータの初期化　void PWM_Params_init(PWM_Params *params)

PWM の初期設定パラメータを初期化します。設定される初期値は以下の３つです。

- period = 10000（10000 マイクロ秒 =10 ミリ秒）
- dutyMode = PWM_DUTY_TIME（デューティをマイクロ秒で指定）
- polarity = PWM_POL_ACTIVE_HIGH（極性をアクティブ High に指定）

```
PWM_Params_init(&pwmParams);
```

引数の設定例

- &pwmParams：初期設定パラメータへのポインタ

PWM を開く　PWM_Handle PWM_open(unsigned int index, PWM_Params *params)

指定した PWM 出力を有効化し、PWM のハンドラを返します。

```
pwm = PWM_open(Servo_PWM, &pwmParams);
```

引数の設定例

- Servo_PWM　：有効化する PWM 出力のインデックス
- &pwmParams：初期設定パラメータへのポインタ

PWM のデューティ比の設定　void PWM_setDuty(PWM_Handle handle, uint32_t duty)

PWM のデューティ比を、PWM サイクル数（今の場合、マイクロ秒）で設定します。

```
PWM_setDuty(pwm, 1.5*1000);
```

引数の設定例

- PWM　　：PWM_open が返したハンドラ
- 1.5*1000　：ここでは、デューティを 1.5 ミリ秒（1.5 × 1000 マイクロ秒）に設定している

コーディング main.c 12

以下は、main.c 11-2 のソースコードを元にしたコーディング例です。■は main.c 11-2 から追加した部分です。

Board.h | EK_TM4C1294.h | EK_TM4C1294.c | main.c

```
/* XDCtools ヘッダファイル */
#include <xdc/std.h>
#include <xdc/runtime/System.h>

/* BIOS ヘッダファイル */
#include <ti/sysbios/BIOS.h>
#include <ti/sysbios/knl/Task.h>
#include <ti/sysbios/knl/Clock.h>

/* C 言語ヘッダファイル */
#include <stdio.h>
#include <stdlib.h>

/* ドライバヘッダファイル */
#include <ti/drivers/GPIO.h>
#include <ti/drivers/PWM.h>
#include <ti/drivers/SDSPI.h>
```

この間の行に変更はありません

```
/*
 * ユーザー変数
 */
uint32_t IP;       // IP アドレス格納用変数
char lockSD;       // SD カード排他制御用変数
char toggleLDM;    // LDM のトグル
char charLDM;      // LDM に表示する文字
PWM_Handle pwm;    // PWM のハンドラ

/*
 * ユーザー関数
 */

/*
 * CGI ソースコード（control.cgi）
 */
static int cgiControl(SOCKET s, int ContentLength, char *pArgs)
{
    .
    .
    .
```

ブラウザによるサーボモータ制御

コーディング main.c 12

```
112     // POST データが "changeChar_LDM" の時、LDM の表示を変更
113     if (strcmp(buffer, "changeChar_LDM") == 0)
114     {
115         if (charLDM == 'A')
116         {
117             charLDM = 'B';
118         }
119         else
120         {
121             charLDM = 'A';
122         }
123     }
124
125     // POST データが "servo_position= 数値 " の時、指定された数値の角度に設定
126     if (strncmp(buffer, "servo_position=", 15) == 0)
127     {
128         // 角度の文字列を整数に変換
129         int degree = atoi(buffer + 15);
130         // 角度を -90 ～ 90 の範囲内にする
131         if (degree < -90)
132             degree = -90;
133         if (degree > 90)
134             degree = 90;
135         CONSOLE("degree: %d\n", degree);
136         // 角度に応じたデューティ（マイクロ秒）を設定
137         PWM_setDuty(pwm, (1.5 - 0.8 * degree / 90) * 1000);
138     }
139
140     httpSendStatusLine(s, HTTP_OK, CONTENT_TYPE_HTML);
141     httpSendClientStr(s, CRLF);
```

この間の行に変更はありません

```
551 /*
552  * メイン
553  */
554
555 int main(void)
556 {
557     Task_Params taskParams;
558     PWM_Params pwmParams;
559
560     // ボードの初期設定
561     CONSOLE(" ボードの初期設定 \n");
562     Board_initGeneral();
563     Board_initGPIO();
564     Board_initPWM();
565     Board_initEMAC();
566     Board_initSDSPI();
567
```

```
568     // GLCD の初期化
569     GLCD_init();
570
571     // ユーザー変数の初期値の設定
572     IP = 0;
573     lockSD = 0;
574     toggleLDM = 1;
575     charLDM = 'A';
576
577     // PWM の初期設定
578     // : PWM パラメータの初期化
579     PWM_Params_init(&pwmParams);
580     // : PWM の周期を指定（マイクロ秒）
581     pwmParams.period = 20*1000;
582     // : PWM を開く
583     pwm = PWM_open(Servo_PWM, &pwmParams);
584     // : PWM のパルス幅を設定（マイクロ秒）
585     PWM_setDuty(pwm, 1.5*1000);
```

これ以降の行に変更はありません