試作検討 2A

プラレールの PID 制御

創造設計第二 TA：春山 弘貴, 岩淵 教郎

平成 20 年 10 月 16 日, 23 日

1. はじめに

今回の試作検討ではモータ全般の駆動の仕方，プラレールモータの PID 制御を行う．モータを駆動，制御する際にはタイマを利用するので，タイマに関する設定の復習も行う．

2. タイマの設定

タイマ A には「タイマモード」「イベントカウンタモード」「ワンショットタイマモード」「パルス幅変調モード」という 4 種類のモードがあります．前回までの試作検討では「タイマモード」を用い外部割り込みと併用することでタイマ割り込みを実現していました．

今回はプラレールモータの制御を行うため「イベントカウンタモード」と「パルス幅変調モード」を使用します．

1.「パルス幅変調モード」: 任意の幅のパルスを連続して出力するモード
   $\Longrightarrow$ モータへの入力となる PWM 出力を生成する．
2.「イベントカウンタモード」: 外部からのパルスをカウントするモード
   $\Longrightarrow$ エンコーダのパルス信号をカウントし，モータの回転角を測定する．

補足

＜その他のモードについて＞

- 「タイマモード」: 内部カウントソースをカウントするモード
- 「ワンショットタイマモード」: カウント値が 0x0000 になるまでの間，1 度だけパルス出力をするモード

＜タイマ B について＞

タイマ B には「タイマモード」「イベントカウンタモード」「パルス幅測定モード，パルス周期測定モード」という 3 種類のモードがある．

「パルス幅変調モード」「ワンショットタイマモード」が存在しない．そのため，PWM 出力をタイマ B からは作ることができない．「タイマモード」「イベントカウンタモード」に関してもタイマ A に比べ機能が削減されたものになっている．

2.1 PWM 出力

PWM とは Pulse Width Modulation（パルス幅変調）の略で一定周期においてパルスが High を出力する時間（デューティ比）を調節することで出力する電力を調節する方法である．モータを制御する場合にはほぼ PWM 制御が用いられる．

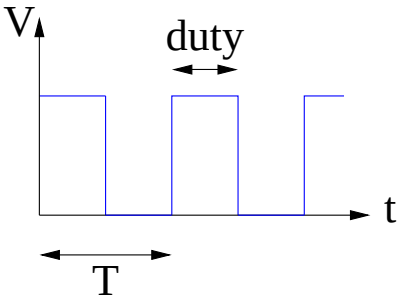

Fig. 1: PWM 信号

Fig. 1 のような形で信号が出力される．入力としてデューティ比を調節する．一見，ON/OFF の連続でモータがカクカク動いてしまうように思われるが周期 T を十分速くすることでなめらかに制御することが可能になる．

> **注意** 今回は 16bit の PWM モードを用いているが，16bit モードでは周期（**Fig.** 1 の T）を変更することができない．8bit モードでは周期を変更できる．詳細はハードウェアマニュアルを参照してください．

2.1.1 設定

PWM 出力を得るために必要な設定項目は次のようになっている．

1. pclkr レジスタでカウントソースの周波数を設定
   （pclkr レジスタの書き込みを許可する必要がある）
2. タイマ割り込みを起さない設定をする．
3. タイマのモード等を設定する．
4. タイマの初期化
5. カウント開始

2.1.2 設定例

以下はタイマ A2 で PWM 出力をするための設定例である．PWM 出力は 16bit でやっている．詳細に関してはハードウェアマニュアルの「12. タイマ」を参照すること．

```
1:      prc0 = 1; // PCLKR レジスタへの書き込み許可
        // タイマのカウントソースを変更
        pclkr = pclkr & ~0x01; //bit0 を 0 にする
        // bit0: 1=f1    0=f2
        // bit1: 1=f1SIO 0=f2SIO

        // タイマ A2 の初期化
2:      ta2ic = 0x00; // TA2 割り込みを使用しない
3:      ta2mr = 0x0f;
        // bit1,0:  11: PWM モード
        // bit2:     1: PWM では 1
        // bit4,3:  01: ゲート機能無し（TAiIN 端子は通常のポート端子）
        // bit5:     0: 16bit モード
        // bit7,6:  00: カウントソースの周波数 = f1 or f2
4:      ta2 = 0; // タイマ値の初期化

5:      tabsr = 0x04; // カウント開始フラグ
```

2.2　エンコーダ

今回はモータの回転角度を測定するのにエンコーダを用いる．エンコーダとは一定角回転するとパルスを出力するようなもので，そのパルス数をマイコンのタイマでカウントすることでモータの回転角を求める．

今回用いるエンコーダは**Fig.** 2 のような２相出力のものである．これは１つのエンコーダから位相のずれた２つのパルスが出てくるもので，これにより回転方向の判別ができる．今回用いるマイコンには２相出力のパルスをカウントするモードがあるので，そのモードを利用する．

また，カウントする際には４逓倍モードでカウントする．これは A 相，B 相のパルスの立ち上がりと立ち下がりを１カウントとすることで通常に比べて４倍細かくカウントできる機能である．今回利用するエンコーダの１回転当りのパルス数は 100 である．これを４逓倍でカウントしているので１回転当りのパルス数は「400」である．

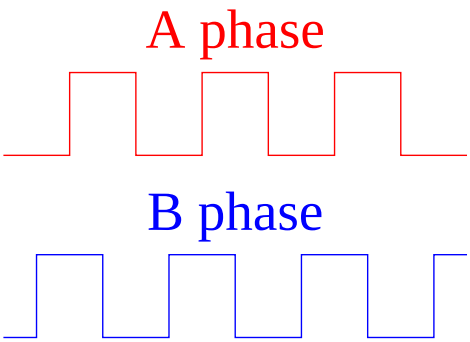

Fig. 2:　２相出力信号

2.2.1　設定

エンコーダを用いるために必要な設定項目は次のようになっている．

1. udf レジスタで用いるタイマの２相パルス信号処理設定を行う．
2. タイマ割り込みを起さない設定をする．
3. タイマのモード等を設定する（４逓倍の設定もここで行う）．
4. タイマの初期化
5. カウント開始

2.2.2　設定例

以下はタイマ A3 でエンコーダ波形のカウントをする際の設定例である．カウントは２相パルス信号処理でやっている．詳細に関してはハードウェアマニュアルの「12. タイマ」を参照すること．

```
1:    udf = 0x40; // アップダウンフラグ
      // 0:禁止, 1:許可
      // bit5: TA2 ２相パルス処理選択ビット
      // bit6: TA3 ２相パルス処理選択ビット
      // bit7: TA4 ２相パルス処理選択ビット

      // タイマ A3 の初期化
2:    ta3ic = 0x00; // 割り込みを使用しない
3:    ta3mr = 0b11010001;
      // bit1,0:         01: イベントカウンタモード
      // bit5,4,3,2:  0100: 2 相パルス信号処理をする場合
      // bit6:             1: フリーラン
      // bit7:             1: 4 逓倍
4:    ta3 = 0; // タイマ値の初期化

5:    tabsr = 0x08; // カウント開始フラグ
```

3. モータの接続

3.1 プラレールモータ

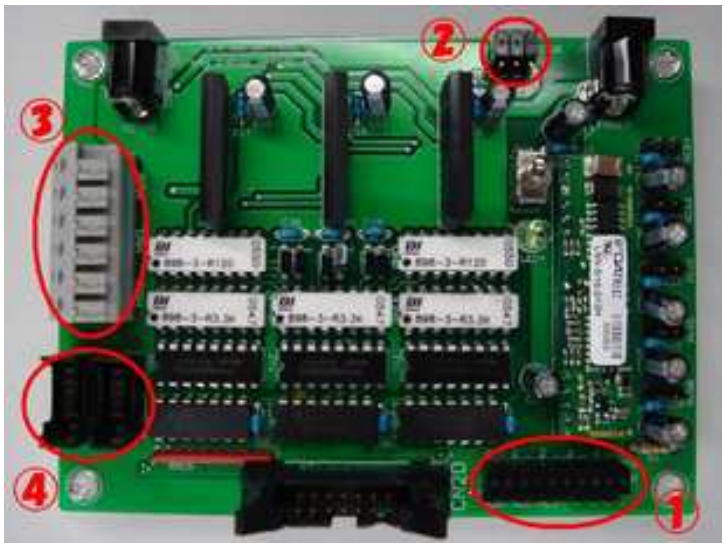

Fig. 3: Motor Driver Board

プラレールモータは **Fig.** 3 のモータドライバボードに接続する．その時にいくつか注意があるのでここで列挙しておく．

3.1.1 DC モータ／サーボモータ切り替えジャンパ

Fig. 4: モータドライブ回路／サーボモータ切り替えジャンパ

Fig. 4 はモータドライブ回路／サーボモータ切り替えジャンパである．上につけるとサーボモータ，下につけるとモータドライブ回路につながる．サーボモータは左から 1 つおきに RCS0~ RCS4，モータドライブ回路は左から 2 つごとに HB0~ HB2．ジャンパを動かすときは誤動作を防ぐために必ず 2 個セットで動かすこと．

警告	**Fig.** 4 では HB0 のジャンパは下になっているが，HB0 は回路の設計不備により使用不可である．RCS0（ジャンパを上に付けた場合）の方は使用可能である．また，モータの方につないでおくとタグ読み取りモジュール（￥12000）を破壊してしまう恐れがある．よって，ここのジャンパは常に「上」に付けておくこと．

3.1.2　DC モータ電源供給源切り替えジャンパ

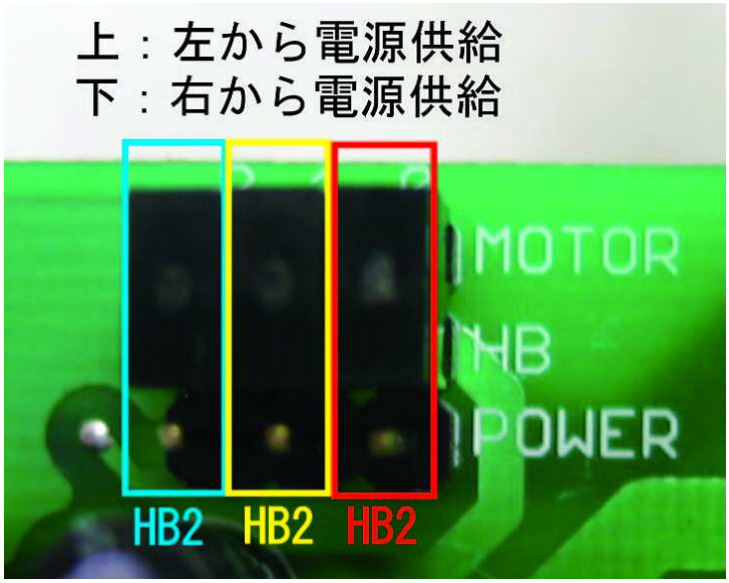

Fig. 5: モータドライブ回路電源供給源切り替えジャンパ

Fig. 5 はモータドライブ回路電源供給源を切り替えるジャンパである．「右」の電源端子から電源を供給するときはジャンパを「下」に，「左」の電源端子から電源を供給するときはジャンパを「上」にして使用すること．

3.1.3　DC モータドライブ回路出力

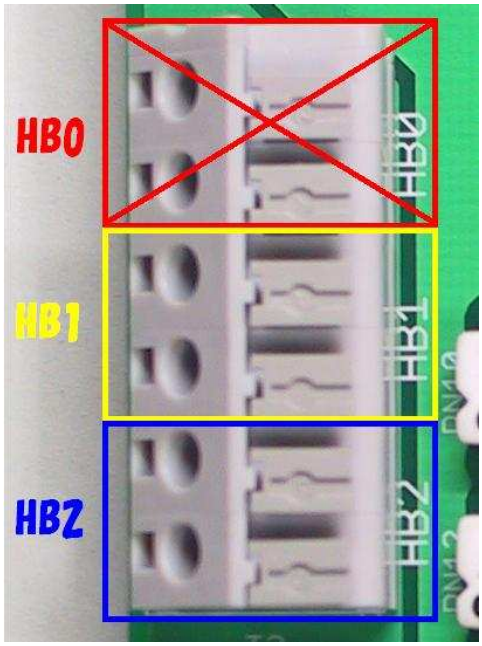

Fig. 6: モータドライブ回路出力

Fig. 6 はモータドライブ回路の出力端子台である．上から 2 つセットで HB0~ HB2 である．

注意	HB0 は使用不可.

3.1.4　エンコーダ入力

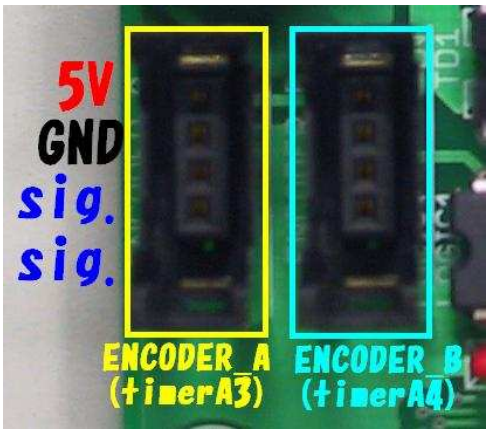

Fig. 7: エンコーダ入力

Fig. 7 はエンコーダの入力端子である．左はタイマ A3，右はタイマ A4 に接続されている．ピン配列は上から

- +5V
- GND
- 信号線
- 信号線

である．

注意	コントローラボードをつないでいるとコントローラボードに付いているエンコーダと干渉してうまくカウントができないことがある．そのときはコントローラボードのエンコーダを適当な位置に動かすとうまくカウントしてくれるようになる．

3.1.5　例

今回の試作検討でのジャンパピンの設定は以下のようになる．

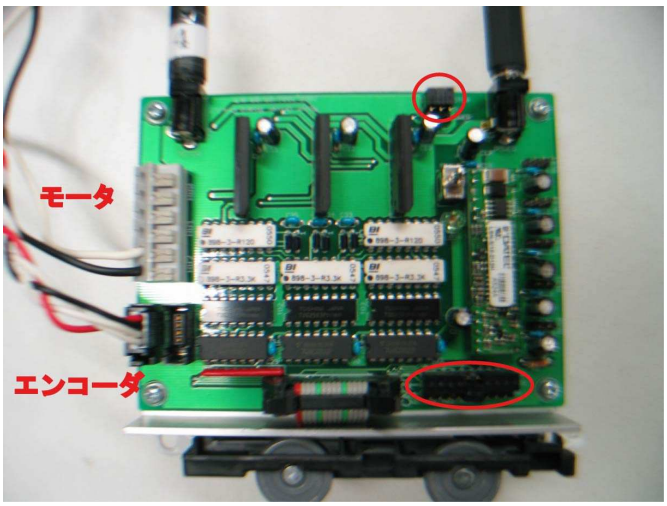

Fig. 8: エンコーダ入力

- モータは HB2 に接続．
- エンコーダは ENCODER_A に接続．

4. 課題１：プラレールモータの駆動

課題１

shisaku02\kadai01 を実行し，PWM のデューティ比を変えるとモータの速度が変わることを確認してください．

- PWM のデューティ比の変更はコントローラボードのボリューム VR_A で行えます．
- モータの回転方向の変更はコントローラボードのトグルスイッチ SW_A で行えます．

モータを MD ボードに接続する際には 3. 節を参照すること．

注意

コントローラボードをつないでいるとエンコーダのモータにつけたエンコーダの値がうまく読み込めないことがある．これは，コントローラボードのエンコーダとモータのエンコーダが同じところにつながっているために干渉してしまうためである．このようなときは，コントロールボードのエンコーダを適当に動かして，モータのエンコーダの値を読み取れるように調節すること．

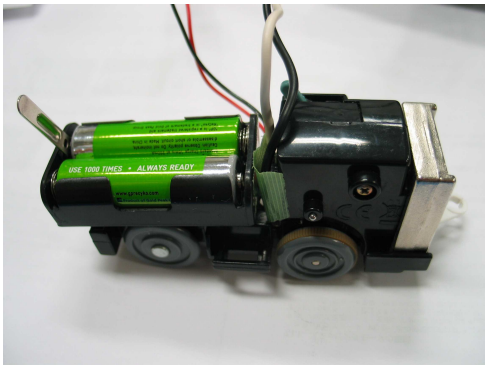

Fig. 9: motor & encoder

Fig. 10: Control Board

5. PID 制御とは

PID 制御とは制御手法の 1 つであり，産業界では主な制御手法として広く用いられている．「PID」は

- 比例：Proportional
- 積分：Integral
- 微分：Derivative

それぞれの頭文字を取ったものであり，その名のとおりこれら 3 つの動作を組合わせた制御手法である．

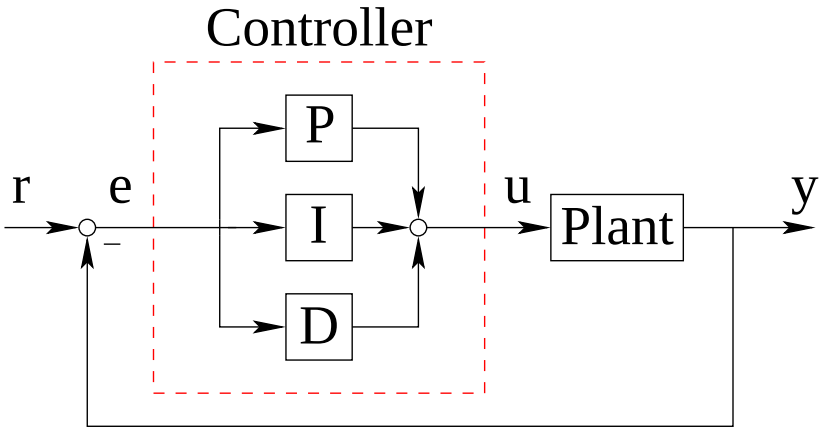

Fig. 11: PID 制御のブロック線図

具体的には **Fig.** 11 のように目標値 r と出力 y との偏差 e そのものに，また偏差の積分，微分にそれぞれ係数を掛け，足し合わせたものが制御入力 u となる．式で表すと以下のようになる．

$$u = K_P e + K_I \int e\,\mathrm{d}t + K_D \dot{e} \tag{1}$$

K_P，K_I，K_D は比例，積分，微分動作に対するゲインでありこれらの値を調節することで所望の出力を実現する．次に，それぞれが出力に対してどのように影響するかを見ていく．

5.1 P 制御

P 制御は最も単純な制御方法である．入力 u は

$$u = K_P e \tag{2}$$

のように表される．すなわち，「出力が目標からずれていれば入力を大きくして，目標に近づいたら入力を小さくする」．しかし，これだけでは必ずしも所望の動作は得られない可能性がある．例えば，

- 定常偏差が残る．
- 収束スピードを早くしようとすると振動的になる．

という問題がある．

そこで，P 制御を基本として積分動作や微分動作を加えて所望の動作の実現を目指す．

5.2 PI 制御

P 制御に積分動作を加えたものである．入力 u は

$$u = K_P e + K_I \int e\,\mathrm{d}t \tag{3}$$

のように表される．P 制御では，今回扱うモータのように摩擦があるような場合には偏差が小さくなり入力が小さくなると入力が摩擦力より小さくなってしまいモータが動かなくなり偏差が残ってしまう（定常偏差）．この時，偏差を積分していれば時間の経過と共に徐々に入力が大きくなり偏差をなくすことができる．このように積分動作には「定常偏差」を取り除く効果がある．

5.3 PD 制御

P 制御に微分動作を加えたものである．入力 u は

$$u = K_P e + K_D \dot{e} \tag{4}$$

のように表される．P 制御に速度の偏差の項が加わっているため，より速く動かそうとする．その結果，応答性が高まり外乱に強くなる．ただし，微分項のゲイン K_D を大きくしすぎるとオーバーシュートが発生しやすくなるので注意が必要である．

5.4 PID 制御

定常目標値に対し PI 制御，PD 制御を行った場合のイメージ図をそれぞれ **Fig.** 12，**Fig.** 13 に載せる．
PID パラメータの設計方法は限界感度法やステップ応答法などが主な方法として広く使われている．

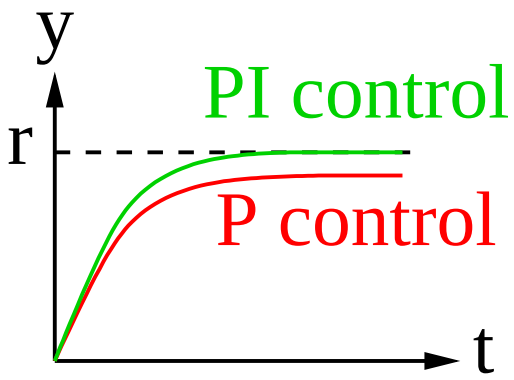

Fig. 12: PI 制御

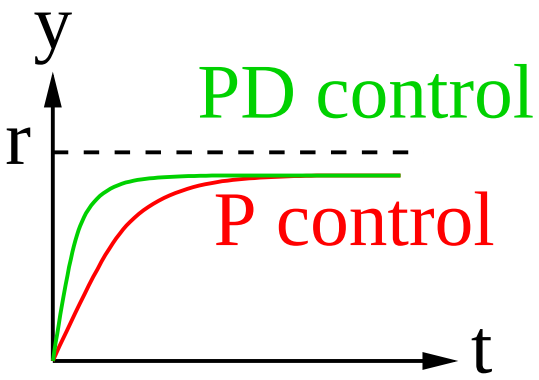

Fig. 13: PD 制御

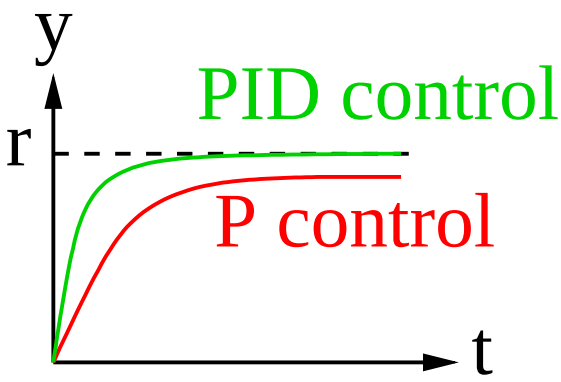

Fig. 14: PID 制御

5.5 目標軌道

PID 制御はある目標値に追従させるための制御方法である．十分早い時間で目標値に追従させることができるパラメータが見つかった場合，目標値にダイナミクスを加えた目標軌道に追従させることができる．ここでは目標軌道として次のようなものを考える．

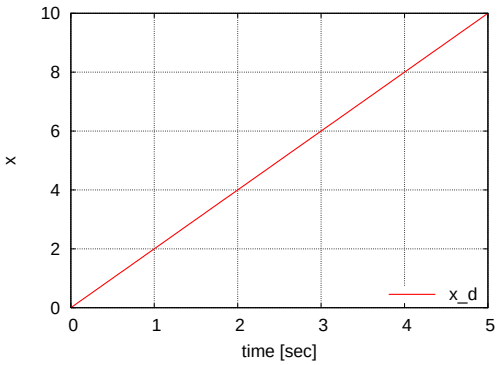

Fig. 15: 目標位置

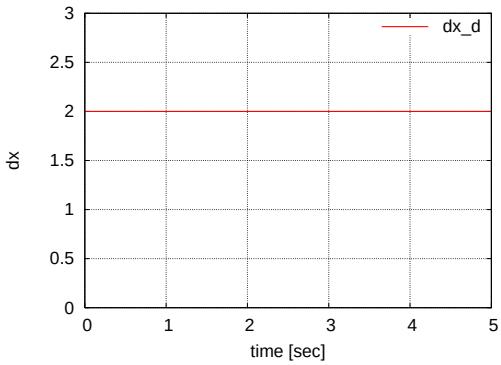

Fig. 16: 目標速度

Fig. 15，**Fig.** 16 は目標速度 $v = 2$ の場合のものである．目標軌道は次の式で求めている．

$$x_d = vt \tag{5}$$

$$\dot{x}_d = v \tag{6}$$

すなわち，目標位置軌道は目標速度を係数とした一次関数になっている．

ある位置に何秒で到達したいかによって目標速度が決まる．タイムトライアルでできるだけ早く到達したい場合などでは次の点に注意しなければならない．

注意	目標速度を大きくしていき，目標速度軌道がモータの出せる最大速度を越えると目標軌道に追従できなくなるので注意が必要である．

6. 課題２

課題２．１

\shisaku02\kadai02 のプログラム全体を一読して，タイマ ta3 をイベント カウントモードとして用いて (ハードウェアマニュアル 12 章参照)，エンコーダの値を読み込み，5，6 の目標軌道に追従する PID 制御を実現するプログラムを完成せよ．

- 変数の型に注意．

課題２．２

完成したプログラムを用いて次の手順に沿って PID パラメータの調整をして下さい．モータを動かす際には次の章を読んで間違いのないよう注意して行うこと．

1. $K_P = 400$，$K_I = K_D = 0$ としてプログラムを実行してください．この時，目標と違う方向に動く時はモータの接続方向を反転して下さい．
2. K_P, K_I，K_D を調整して応答を改善してください．

K_P，K_I，K_D を変更する際には「C Watch Window」を利用すること．

注意 コントローラボードが付いているとエンコーダの値がうまく読み取れなかったりするので，コントローラボードをはずして調整を行ってください．

PID パラメータ設定方法

ステップ応答法を用いる。**Fig.** 17 にあるように、この曲線の変曲点において接線を引き、その傾きを R とおく．また、この接線が横軸と交わる時刻を L とし、応答の定常値を K とする．このパラメータ R,L に対して、以下の表に示されるような調整則が知られている．（フィードバック制御より）

コントローラ	K_P	T_I	T_D
P	1/RL	-	-
PI	0.9/RL	L/0.3	-
PID	1.2/RL	2L	0.5L

IC のとなりについているトグルスイッチを２つとも OFF（LED 側に傾ける）にするとステップ応答法のパラメータ取得モードになります。「C Watch Window」で「PWMinput」の値を適当（クラッチが外れない最大出力）にセットして、次のパラメータの値を「C Watch Window」上で取得してください。

T（遅れ時間）　: timeT
L（加速時間）　: timeL
K（ゲイン）　: maxdx

設計する際に、ゲインをかけることを忘れないように。
また、取得データの信頼性があまり高くないため、取得した値を元にパラメータを調整すること．

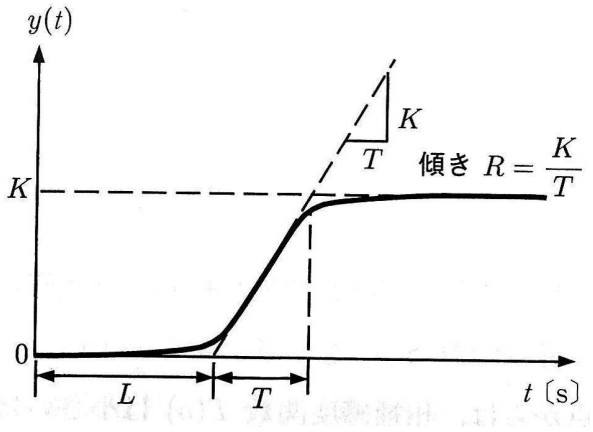

Fig. 17: プロセス反応曲線

主な関数

```
関数名：vel_control
内容　：定速度制御を行う
　　：xT：目標位置、span：目標到達時間、init_flag1：初期化用フラグ
　　：x：位置、dx：速度、Kp,Ki,Kd：PID パラメータ
　　：cwccw_flag：モータ回転方向指定フラグ
　　：input：モータへの PWM 入力
 void vel_control(int xT,  float span, unsigned char _far *init_flag1,
                  int x, float dx,
                  int Kp, int Ki, int Kd,
                  unsigned char _far *cwccw_flag, unsigned int _far *input)
*************************************************************************
関数名：PID_Control
内容　：目標位置、速度に対して PID 制御を行う
　　：x：位置、dx：速度、x_d：目標位置、dx_d：目標速度
　　：init_flag：初期化用フラグ，Kp,Ki,Kd：PID パラメータ
　　：cwccw_flag：回転方向 ，input：PWM 入力
void PID_control(int x, float dx, int x_d, float dx_d,
                  unsigned char init_flag, unsigned char _far *cwccw_flag,
                  unsigned int _far *input, int Kp, int Ki, int Kd)
```

試作検討の内容や，プログラムを書く上で役に立ちそうなソースをライブラリとして試作検討の HP に載せておきます．更新は適宜行い HP 上に更新記録を載せるのでそれを元にライブラリを参考にしてください．
http://www.sc.ctrl.titech.ac.jp/lecture/ss207/index.html