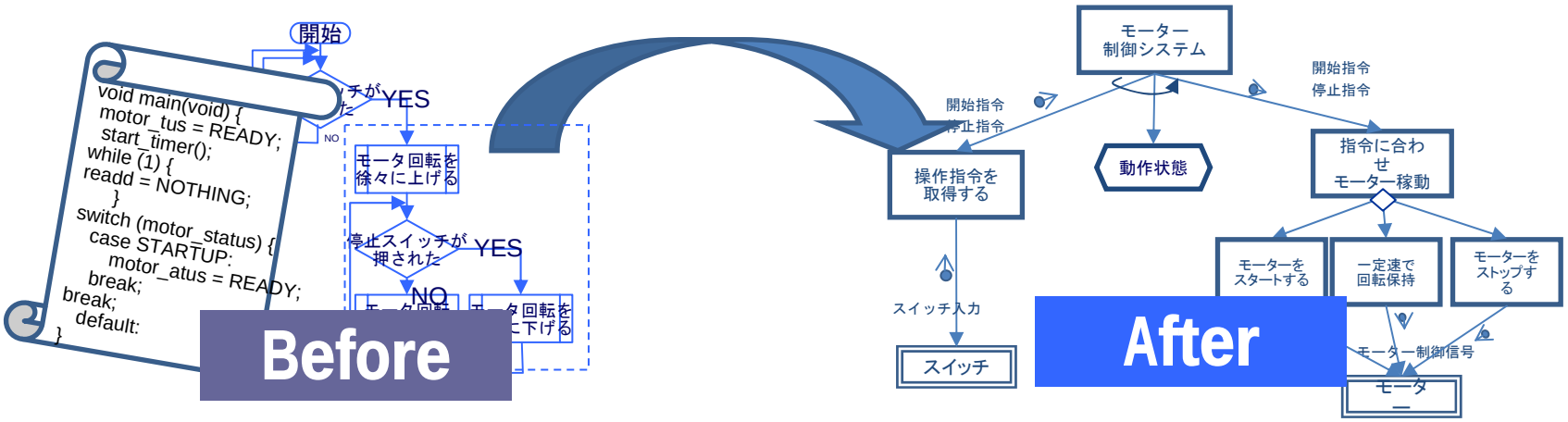
BACKSLASH DESIGN

『設計力でカイゼン Before/After』

ソースコードの迷宮、
作りこんでいませんか？

本講座の概要

ハードウェアの制御仕様そのままに作成した実験用
プログラミングコード、どのように製品開発として利用しますか？
作られたモジュールを改変する際、コードだけを見て手を
入れるよりも、設計図を書いて機能の本質を見極めながら
改変することで、品質・効率両面での改善が図れます。
本セッションでは、実際のコードを用いて、**設計によるカイゼン**の
例をご紹介します。



Before/Afterコード 開発環境

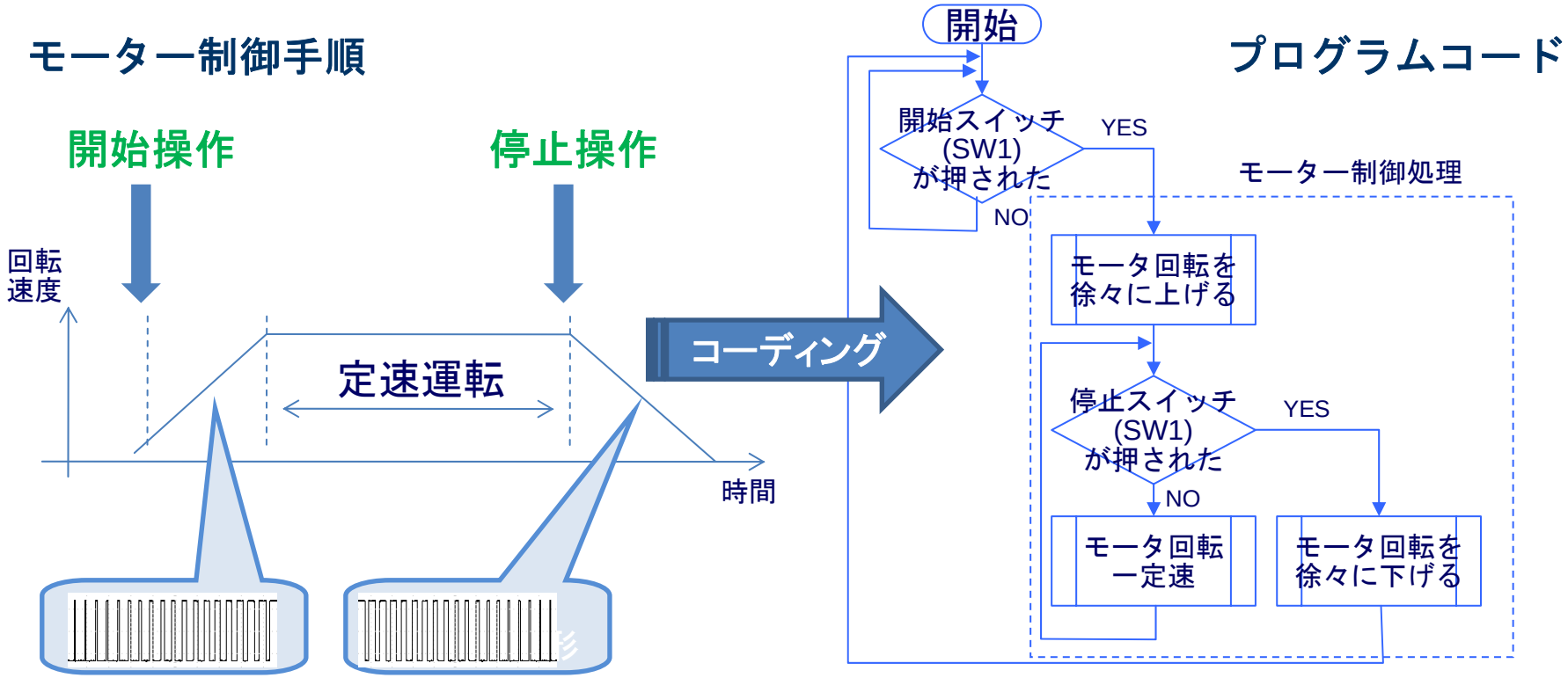
- ターゲットボード
  - ➢日経エレクトロニクス オリジナル
    『組込みシステム入門 』 付録ボード
    CPU：16ビット・マイコンR8C/25
    (エミュレータ付き)
- 開発環境ソフトウェア
  - ➢ルネサス社製 統合開発環境
    - ◆統合開発環境 HEW
    - ◆コンパイラ NC30

Before/After サンプル システム仕様

- モーターを制御して、ファンの回転を制御する
- ファンの制御方法
  - ➢ 駆動にはDCモーターを使用
  - ➢ 起動・停止時は、
    徐々にモータ回転を加速／減速する
    - ◆モーターへのPWM信号のデューティ変化で、
      モータ回転速度を制御する
    - ◆加速・減速は、台形駆動でUP／DOWN
  - ➢ 起動指令、停止指令は同一スイッチで入力
    - ◆トグル制御 OFF → ON → OFF

Beforeコード　解説

- 仕様書に示された制御手順でプログラム作成



ここで、ちょっと仕様変更

- 不具合の修正
  1. モーター加速中は、停止指令が受け付けられない

- 仕様追加
  2. 緊急停止ボタンが押されたら、即座にモーター停止
     （スイープ停止ではなく、瞬断停止）
     その後はエラーランプを点灯し、電源再投入するまで
     開始指令は無効とする

Beforeコードの修正

```
/* モータ駆動信号を出力する */
void    motor_control(void)
{
    unsigned char    pwm_step, hold_cou

    /* 回転をを徐々に上げる */
    for (pwm_step = 0; pwm_step < PWM_FULL_RANGE; pwm_step+
        MOTOR_OUT_PORT = ON;
        for (hold_count = 0; hold_count < pwm_step; hold_count++) {
            wait_5us();
        }
        MOTOR_OUT_PORT = OFF;
        for (hold_count = 0; hold_count < ~pwm_step; hold_count++) {
            wait_5us();
        }
    }

    /* 停止指令があるまで、定速回転を保つ */
    for ( ;; ) {
        MOTOR_OUT_PORT = ON;
        if (get_switch(SW1_STATUS) == SW_PUSHED) {
            break;
        }
    }
```

複数の条件文の追加が必要

STOP操作のbreakが必要

```
    /* 回転をを徐々に下げる */
    for (pwm_step = PWM_FULL_RANGE; pwm_step > 0; pw
        MOTOR_OUT_PORT = ON;
        for (hold_count = 0; hold_count < pwm_step; hold_cou
            wait_5us();
        }
        MOTOR_OUT_PORT = OFF;
        for (hold_count = 0; hold_count < ~pwm_step; hold_co
            wait_5us();
        }
    }
}

/* 指定のスイッチ状態を取得する */
unsigned char get_switch(unsigned char sw_status)
{
    if (sw_status == 1) {
        return SW1_IN_PORT;
    }
    else if (sw_status == 2) {
        return SW2_IN_PORT;
    }
    return SW_FREE;
}
```

緊急停止操作のチェックを各ループに追加

緊急停止操作の対応追加

設計図を作りましょう！

- コードだけを見た修正
  - ➢視野が狭く、局所的な変更に陥りがち
  - ➢修正を重ねるうちに、スパゲティ化の恐れも・・・

- コードを俯瞰した図 ＝ 設計図　を作りましょう
  - ➢例：モジュール構造図を作る
    - ◆コードの各部分の意味合いがハッキリ
    - ◆仕様変更の影響範囲もクッキリ

↓

他者が見てもわかりやすい　＝　保守向上
テストもしやすい　＝　品質も向上

設計図の紹介

- フローチャート

  問題の定義、分析または解放の図的表現であって、データ流れ図、プログラム流れ図及びシステム流れ図とする。

  - プログラム流れ図

    プログラム中における**一連の演算を表わす**。

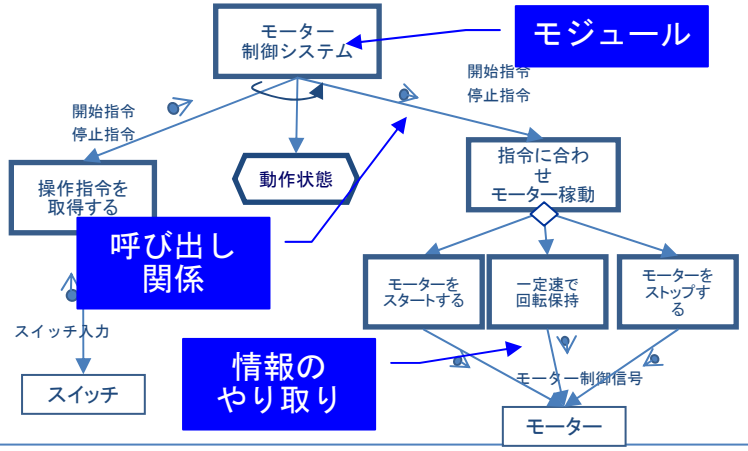
    プログラムの流れを理解し、かつ作成するのに便宜を図る特殊記号。

  JISX0121より

- 構造図(ストラクチャーチャート)

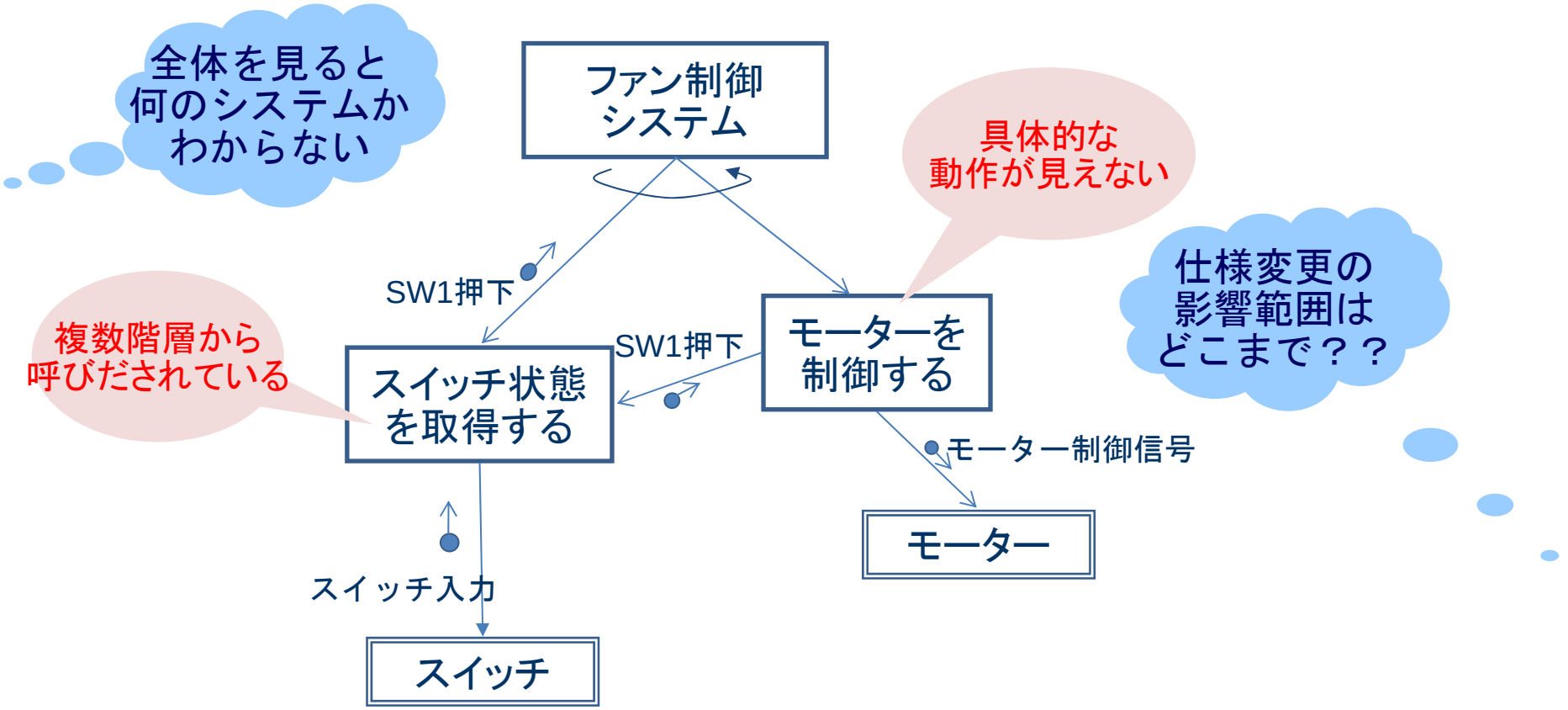
  プログラムの階層構造を表した図。ソフトウェアモジュールやデータ、それらの呼び出し関係、情報のやり取りを示す。

  ソフトウェア設計の**全体像や、モジュールの依存関係を表す。**



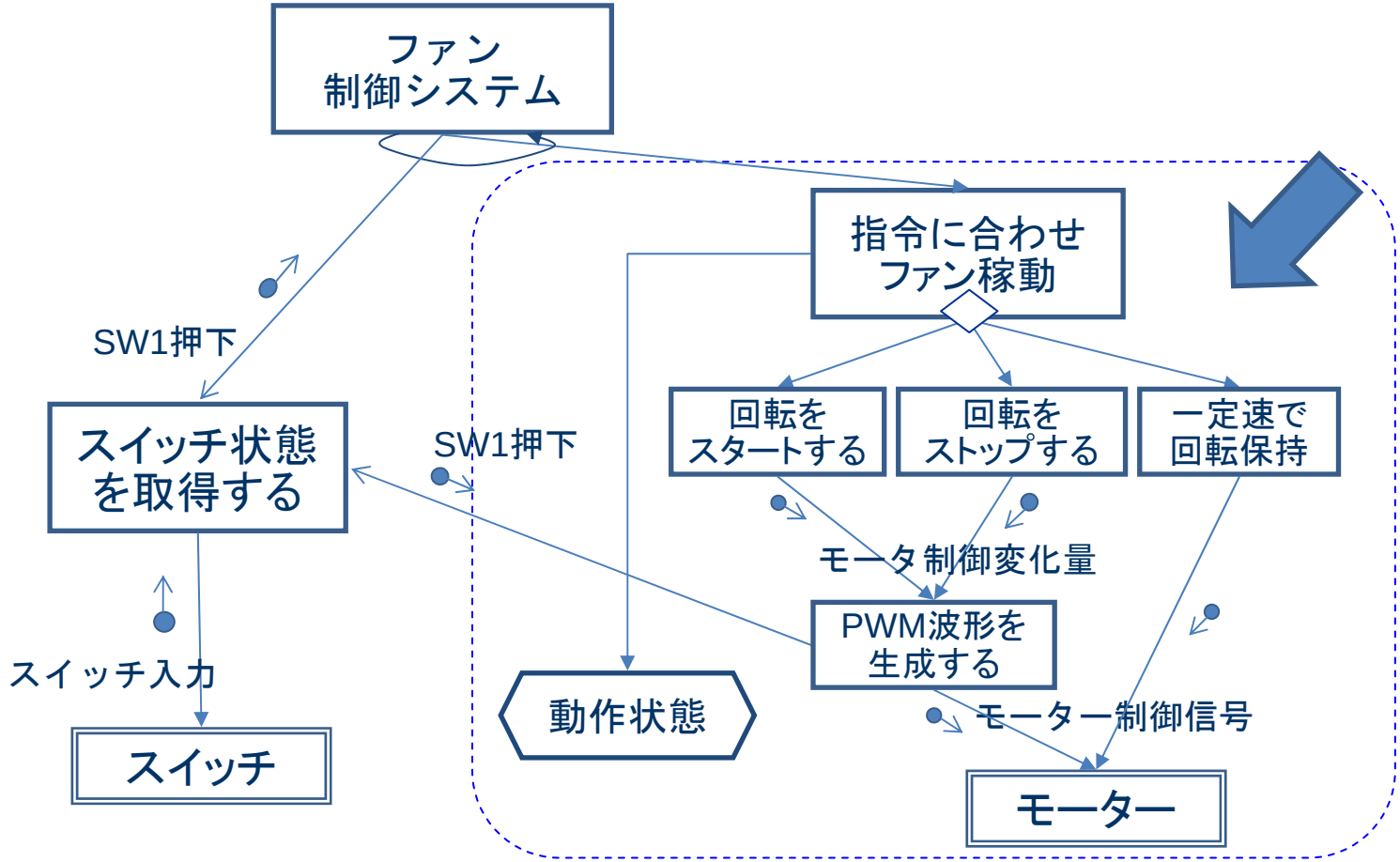
Beforeコードをリバースする

- Beforeコードから、構造図を作成してみる



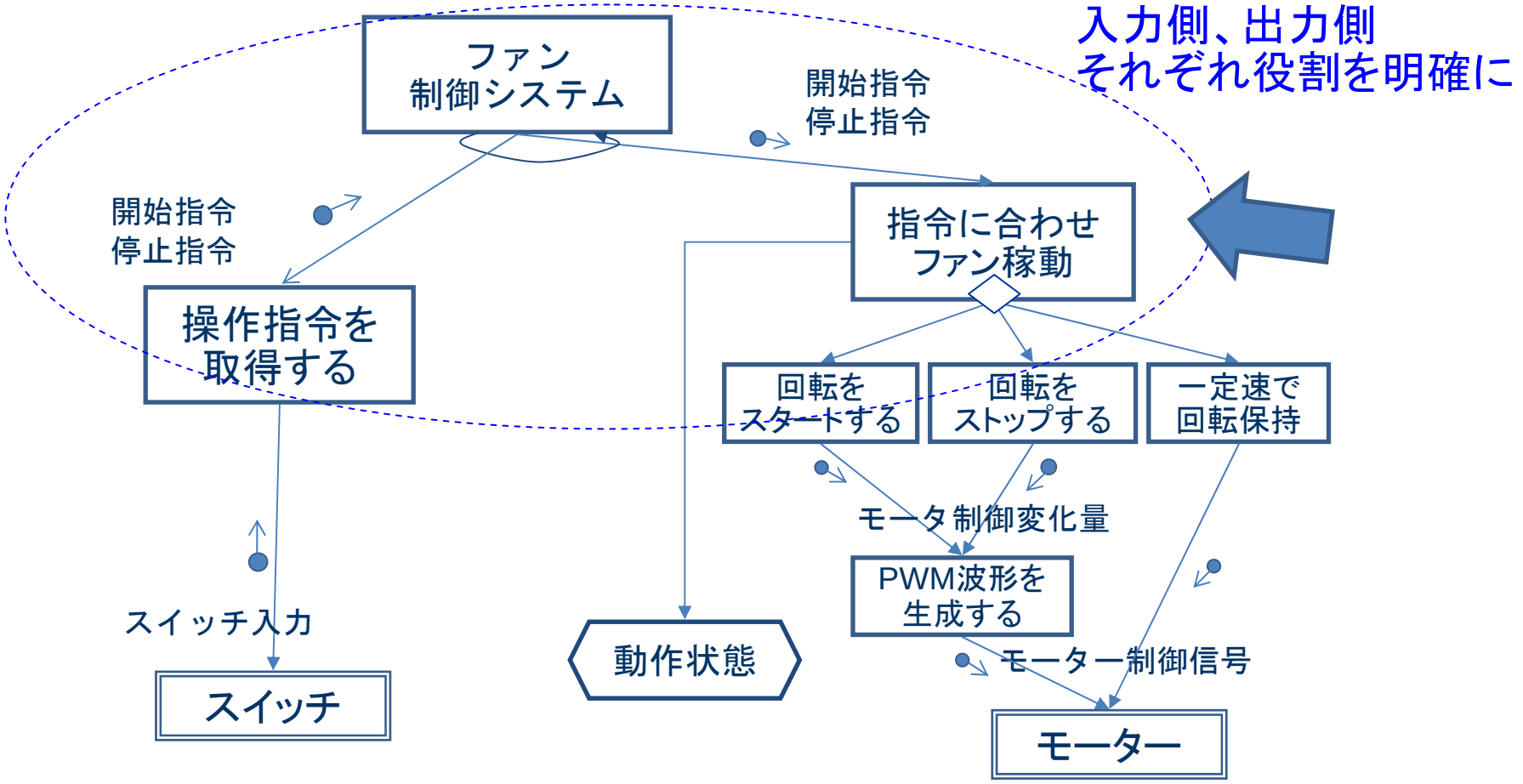
リファクタリング Step1.

- 処理を細分化 → 機能ごとにモジュール再分割



リファクタリング Step2.

- 全体の役割分担を見直し、モジュール再設計



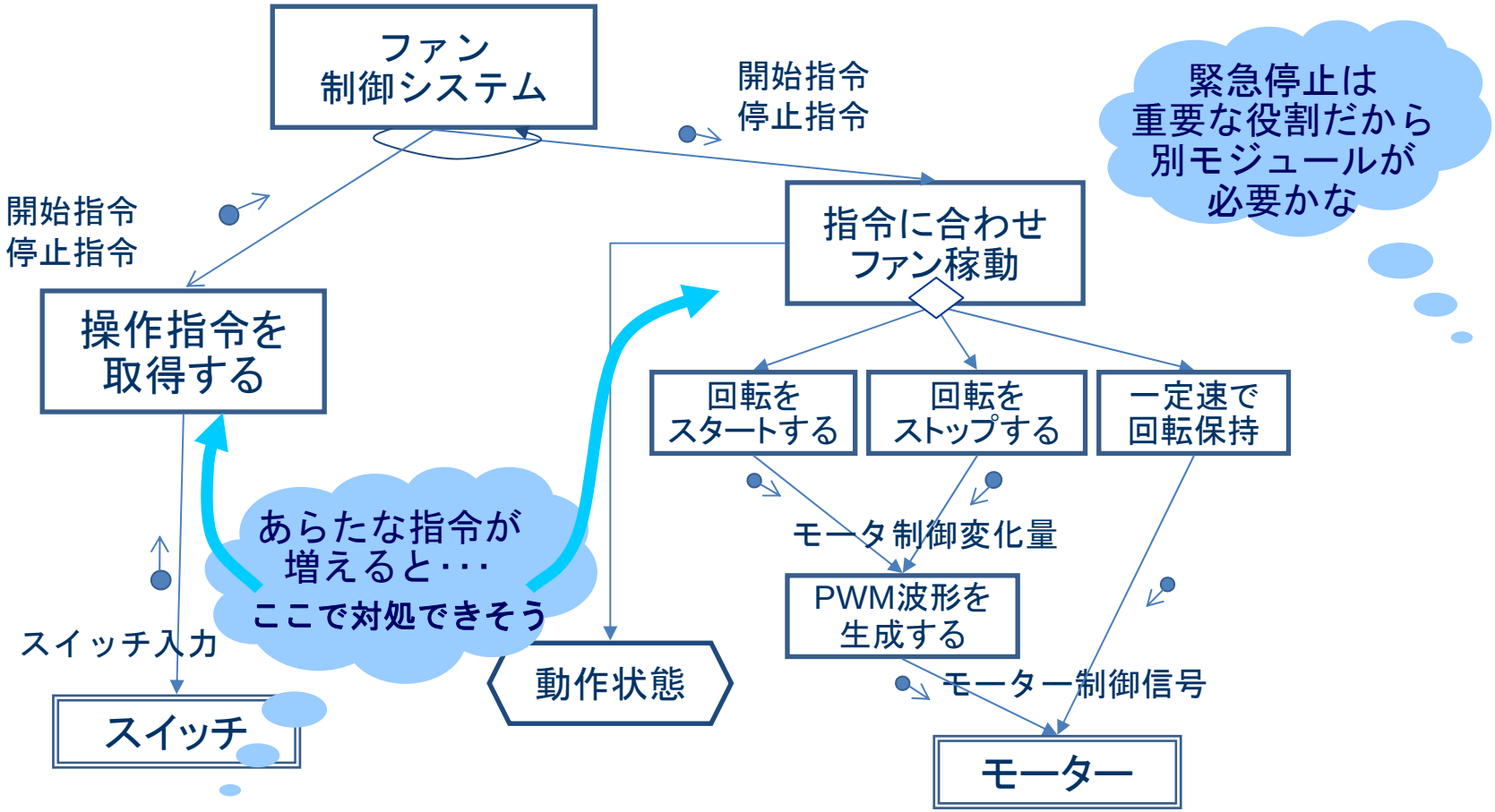
Afterコードの解説

- 入力側、出力側モジュールの役割を明確に
  - 入力部：ユーザからの操作入力
  - 出力部：モーター動作制御部
  - トップのモジュールが、入出力全体を統括
    - モジュール間のインタフェースも見直

- ファン制御部を、役割により分割
  - スタート・定常・ストップ処理を分離
    - イベントにより、各モジュールは起動される
    - これらの処理は一連とは限らない

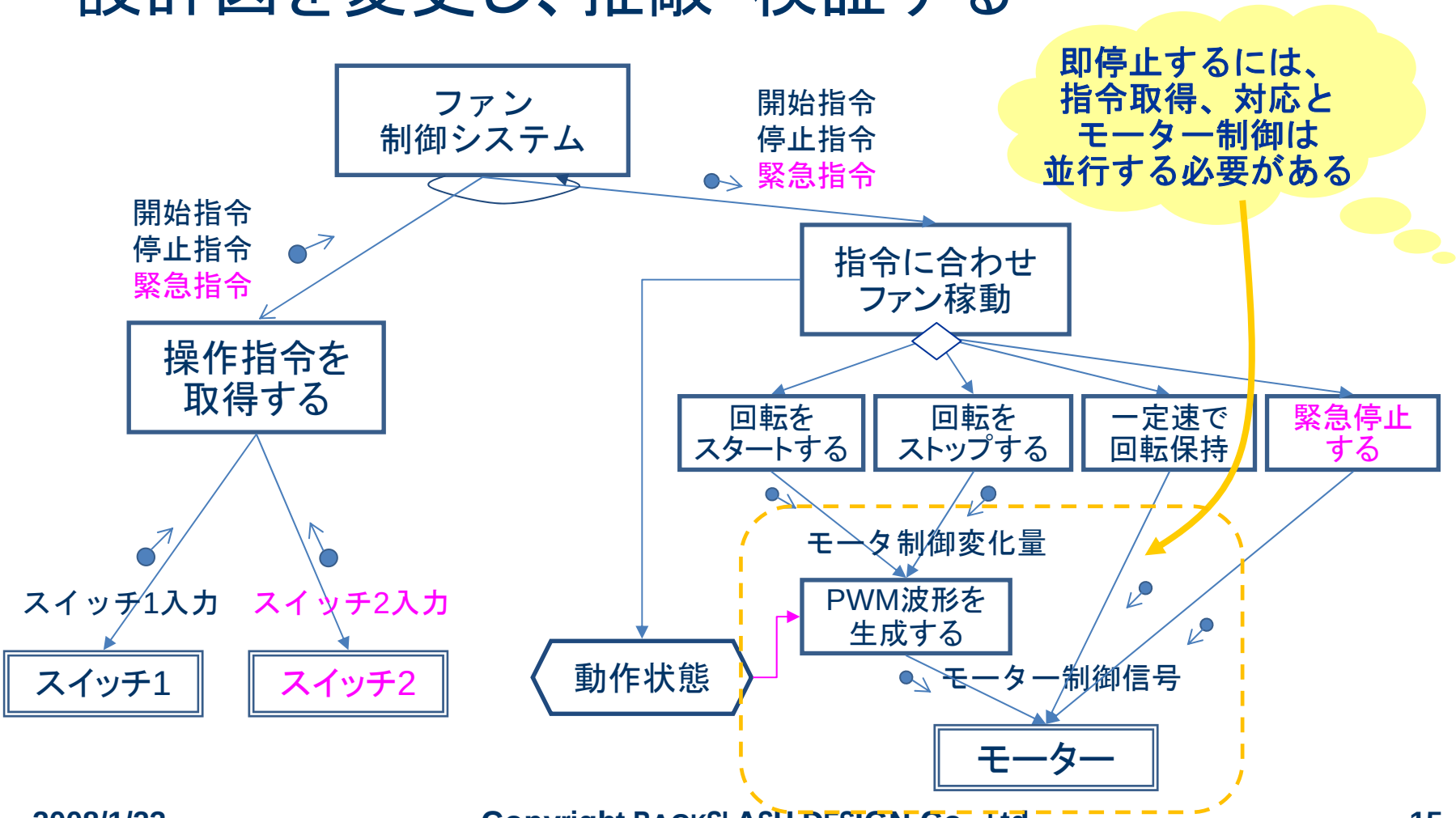
設計図で修正を検討　Step1

- 変更箇所と、その影響範囲を見極める



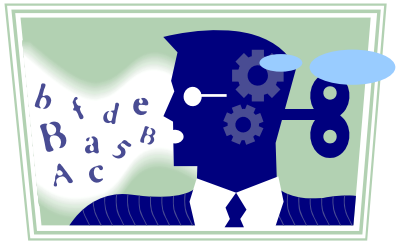
設計図で修正を検討 Step2

- 設計図を変更し、推敲・検証する



Beforeコードを振り返る

- 良い点
  - ➢コード行数が少ない
  - ➢言われたままコードにしたので、早くできた
- 悪い点
  - ➢全体を把握しにくい
    → 変更の都度、コードを解読する？
  - ➢追加・修正時、影響範囲が特定しにくい



実は・・・
設計図はコード作成者の
頭の中だけにある

Afterコードを振り返る

- 良い点
  - ➢コードを可視化した設計図がある
    - ◆広い範囲が見渡せる
    - ◆設計意図がコードにも反映されている
  - ➢変更・拡張しやすい
    - ◆仕様変更時の影響箇所が特定しやすい
- 悪い点
  - ➢コードサイズが大きくなった
  - ➢設計図の作成に手間がかかる

設計者以外にも
扱いやすい
コードに

本当に、Afterコードが良いの？

Before/After それぞれのコードで次の仕様変更の対応方法を検討してみてください

- 更なる仕様変更
  1. イベント入力の変更
     - START操作 → 複数系統(SW＋ハードウェア信号)に
     - STOP操作　→ ハードウェアからの信号
  2. タイプアップ機能の追加
     - STARTから一定時間経過したら、自動的にSTOPする

最後のメッセージ

- 一手間を惜しまないでください
  - “**設計図の作成**”で、この効果は生まれます
  - 「即コード修正」と、「熟考した修正」
    ながら～い目で見て・・・どちらを選択しますか？
- 一人でコードと睨めっこするよりも
  設計図を使ってチームで取り組みましょう！

設計図を作って、
ソフトウェア開発を
より楽しくしましょう

参考資料

- 組込みソフトウェア開発のための構造化モデリング
  - SESSAME WG2 著
  - 翔泳社　　2006年
- CODE COMPLETE　第2版
  ～完全なプログラミングを目指して～
  - スティーブ マコルネ 著
    クイープ 訳
  - 日経BPソフトプレス　　2005年
- 組込みソフトウェア開発のための
  リバースモデリング
  - SESSAME WG2 著
  - 翔泳社　　2007年
- ソフトウェアの複合／構造化設計
  - Glenford J.Myers 著
    國友義久 伊藤武夫訳
  - 近代科学社　1979年
- Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design.
  - Edward Yourdon, Larry L.Constantine
  - Prentice-Hall 1979年
- The Practical Guide to Structured Systems Design.
  - Page-Jones, M.
  - YOURDON Press 1980年