転ばない

本誌2007年5月号付属V850マイコン基板を活用

二足歩行ロボットの製作（中編）

岡田 紀雄

本誌2007年5月号付属V850マイコン基板を使用して，二足歩行ロボットを製作した．前編（2008年3月号，pp.131-140）では，きょう体を製作する上での工夫やサーボ・モータの制御方法を説明した．今回は，レート・ジャイロ・センサでバランスをとるA-D変換の処理方法や静歩行による歩行方法などを紹介する．なお，本記事で紹介するアプリケーションのソース・コードなどは本誌のWebサイト（http://www.cqpub.co.jp/interface/）からダウンロードできる．（編集部）

関連データ

1. ロボットのバランスをとる

● 2種類のセンサでバランスを制御する

ロボットにとってのセンサは，人間の五感に相当します．一般的にロボットのバランス制御用センサとして，加速度センサやレート・ジャイロ・センサが使われています．加速度センサは重力方向や異常な加速度を検知し，レート・ジャイロ・センサは急激な回転系の速度を検知します．これらのセンサは，バランスを崩して上体を起こそうとする場合に威力を発揮します．

加速度センサやレート・ジャイロ・センサの信号には，アナログ電圧値を扱うものが多く存在します．マイコンのA-Dコンバータ端子に接続して電圧値を読むことでセンサ情報を得ることができます．センサ出力の電圧値を正確に読み取るためには，センサのハードウェア・マニュアルを参照する必要があります．今回は，秋月電子通商で購入したKXM52-1050モジュール（Kionix社製）を使い，メーカ推奨のコンデンサを外付けしました．レート・ジャイロ・センサには，ラジコン・ヘリコプタで実績の多いENC-03Mシリーズ（村田製作所製）を使う予定でしたが，サイズの小さいENC-03Rシリーズを使い，温度特性比較を行いました（**図1**）．すると，室内の温度変化に対して出力オフセットのズレがない特性が得られました．このレート・ジャイロ・センサは，メーカがサンプル品を有償で直販しています．

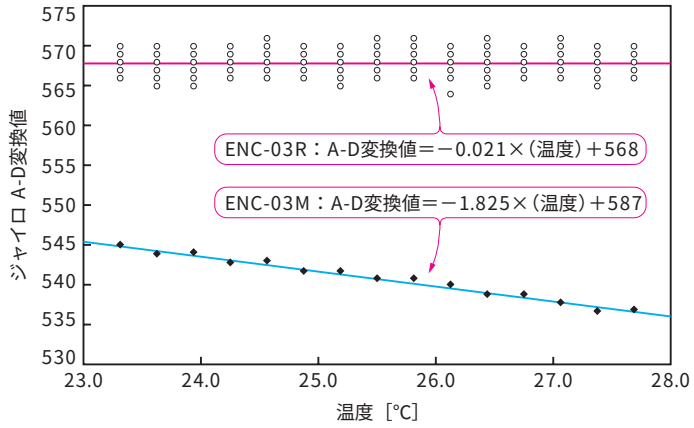

図1　ジャイロ・センサの温度依存性

● A-D変換処理の最適化を考える

一般に，アナログ値の変換に際しては，A-D変換時間と，動作モード（アナログ端子を含む）の設定が必要です．V850の場合，総A-D変換時間は，安定時間（セットアップ時間）の1.3μs～2.5μsと，実際の変換時間2.6μs～10.4μs，そしてウェイト時間（ウェイトなし，または変換時間と同じ）を加算した時間になります[3]．二足歩行ロボットとしては，非常に高速な変換が可能です．

ソフトウェアで扱える動作モードは，1端子ずつユーザが端子指定を行うセレクト・モードと，ANI0端子からANI*n*（*n*=0～11の任意の値）端子まで連続して変換を実施するスキャン・モードの二つです．これらに対して，変換終了後に続けて変換を開始する連続指定と，1回のみの変換で終わるワンショット指定を組み合わせることができます．つまり，4種類の設定が可能となっており，連続セレクト・モード，連続スキャン・モード，ワンショット・セレクト・モード，ワンショット・スキャン・モードからA-D変換の方法を選ぶことになります．

ハードウェア設計時にANI0端子から順番に使用している場合，スキャン・モードでの変換を実施することで変換終了後の割り込み処理などの負荷が減ると期待できます．一方，不連続なアナログ端子を使用している場合は，セレクト・モードで一つずつ変換を行うことが順当な手段と思われます．しかし，変換時間に対して十分に余裕がある場合（それほど高速変換を求めていない場合）でも，スキャ

KEYWORD ── レート・ジャイロ・センサ，加速度センサ，A-D変換，CSI，SPI，CS端子，ACK信号，LPF，D-Aコンバータ，wavファイル，サーボ・モータ，PWM信号，等速直線運動

ン・モードで変換するとよいでしょう．この場合は先に述べたようにアナログ変換時のノイズ低減のため，A-D 変換を行わなくてよい端子は静電防止保護回路を除き，未接続であることが推奨されます．

今回の製作事例では，接続端子が不連続なアナログ端子を使用しています(**表 1**)．ここでは，可能性のある 3 種類の方法を紹介します．一つ目の方法は，ハードウェア接続に忠実かつ連続的に A-D 変換を行う方法で，ワンショット・セレクト・モードを用います．割り込み処理発生ごとに，次の A-D コンバータ端子を指定してから A-D 変換を開始するという手順です．二つ目の方法は，A-D コンバータをほかのソフトウェアとは非同期で連続的に変換して取り込む手法で，連続スキャン・モードを用います．この方法の特徴は，一つ目の方法よりも割り込み回数を減らした手法になります．不連続でアナログ端子を接続している場合は，無駄な A-D 変換処理で待たされることになります．

そこで今回は，A-D 変換後の割り込みを発生させないで，ほかのイベント(例えば 16 ビット・タイマ TMM)により A-D 変換を開始(ADA0CE ＝ 1 に設定)する方法を採用し(**リスト 1**)，ワンショット・スキャン・モードで変換を行います．16 ビット・タイマ TMM により指定時間ごとに変換する場合，A-D コンバータの変換処理時間よりも長い周期の割り込みである必要がありますが，その一方で割り込み処理を多様化することを避けられるというメリットがあります．手法として，タイマ・トリガ・モードによる変換開始の指示も可能ですが，16 ビット・タイマの TMP2 固定になっていることから，今回は使用しませんでした．

表 1　アナログ機器と A-D コンバータ端子

加速度センサの *xy* 軸と本製作事例の *xy* 軸は逆である．正負方向を含めて，製作時の取り付け方向を確認して，プログラミングする必要がある．

A-D コンバータ端子	アナログ機器
ANI0	バッテリの 1/20 電圧値(OP アンプの出力)
ANI1	加速度センサ *z* 軸(KXM52-1050 OutZ 14)
ANI2	(非接続)
ANI3	加速度センサ *x* 軸(KXM52-1050 OutY 13)
ANI4	(非接続)
ANI5	加速度センサ *y* 軸(KXM52-1050 OutX 2)
ANI6	(非接続)
ANI7	(非接続)
ANI8	(非接続)
ANI9	レート・ジャイロ(ピッチ軸；*y* 軸回転)
ANI10	温度測定(LM35)
ANI11	レート・ジャイロ(ロール軸；*x* 軸回転)

● 無線コントローラを選ぶ

今回，受信機インターフェースが CSI(Clocked Serial Interface ； SPI とも呼ぶ)規格の PlayStation2 用ワイヤレ

リスト 1　A-D 変換の割り込みなし＆ワンショット・スキャン・モードの例

```
#include    "typedef.h"
/*
 *      割り込み処理関数(ベクタ)設定
 */
#pragma interrupt INTTM0EQ0 intc_TMM0EQ0
/*
 *      グローバル変数定義
 */
uint16  adc_val[12];    // アナログ変換値
/*
 *      割り込み処理ルーチン(TMM0)
 *          引き数：割り込み処理のため引き数はなし
 *          返り値：割り込み処理のため返り値はなし
 */
__interrupt
void    intc_TMM0EQ0( void )
{
    /*
     *      割り込み処理ルーチン(ADC)
     *      ・・・ワンショット・スキャン・モードの場合，
     *            ANI0-ANI11 変換終了して割り込み発生．
     */
    adc_val[ 0] = ADA0CR0  >> 6;    // 上位 10 ビット
    adc_val[ 1] = ADA0CR1  >> 6;    // 上位 10 ビット
    adc_val[ 3] = ADA0CR3  >> 6;    // 上位 10 ビット
    adc_val[ 5] = ADA0CR5  >> 6;    // 上位 10 ビット
    adc_val[ 9] = ADA0CR9  >> 6;    // 上位 10 ビット
    adc_val[10] = ADA0CR10 >> 6;    // 上位 10 ビット
    adc_val[11] = ADA0CR11 >> 6;    // 上位 10 ビット

    ADA0CE  = 1;                    // A-D 変換開始
}
/*
 *      ADC 初期化
 *          引き数：なし
 *          返り値：なし
 */
void    init_ADC( void )
{
    int     i;

    for( i = 0; i < 12; i++ ) {
        adc_val[i] = 0;  // adc_val[]の 0 クリア
    }
    ADA0CE = 0;          // A-D 変換 動作ストップ
    ADMK   = 1;          // A-D 変換 割り込み禁止
    ADIF   = 0;          // A-D 変換 割り込み状態フラグ・クリア

    ADA0S  = 0b00001011; // ANI0-ANI11 チャネル
    ADA0M0 = 0b00110000; // ワンショット・スキャン・モード
    ADA0M1 = 0b10000010; // 高速変換(計 48.75 μ s)
    ADA0CE = 1;          // A-D 変換動作スタート
}
/*
 *      TMM0 割り込み処理の初期化
 *          フリー・ランニング・モード
 *          引き数：なし
 *          返り値：なし
 */
void        init_TMM0( void )
{
    TM0CE    = 0;           // TMM0 タイマ動作禁止
    TM0EQIC0 = 0b01000111;  // 割り込み優先レベル:7
    TM0CTL0  = 0b00000010;  // カウント・クロック:fxx/4
    TM0CMP0  = (50*20/4)-1; // インターバル時間:50 μ s
    TM0EQMK0 = 0;           // TMM0 割り込み許可
    TM0CE    = 1;           // TMM0 タイマ動作許可
}
```