認知反応型ロボットの開発

指導教員　全 卓樹　助教授

高知工科大学知能機械システム工学
浜 口　　和 洋

目次

第１章　序章

1.1 はじめに

ロボットに作業を行わせるためにロボットに対してその目的となる作業を教え込ませる必要がある。つまりロボットに対し命を与える行為である。

２０世紀初期においてロボットという概念が生まれ。1980年以降、社会は大量消費を前提とした大量生産を目的とする社会構造へと変化する。それに伴い産業用ロボットの需要はますます増加の一途をたどることとなる。今日の日本においては世界中のロボットの約６割が設置され社会と共に歩み人と共に仕事をしている。また，それに伴いロボットによる作業を実現させる目的において，ロボットの自律化を目指した様々な研究がなされてきた。しかしながら，ロボットの自律化は遅々として進まず，現状では作業をいかに効率よく教示するか，あるいはプログラミングをいかに簡易に実現するかがロボットの応用分野や作業能力に多大な影響を与えている。

ロボット自体を単なる機械と判断するか、もしくは機械としての物ではなく人間のパートナーとして認識するか。この認識自体においても今日の高齢化問題・多品種少量生産化にはじまる社会変化において大きな変化をしようとしている。

ここ数年において，アミューズメントロボットが次々に発表・発売されている。HONDAの2足歩行ロボットP2の発表とも相まって広く大衆にロボットが認められるようになりはじめた。テレビ番組や雑誌にもアミューズメントロボットがしばしば取り上げられている。しかしながら大衆にブームとして到来したこのアミューズメントロボットという位置づけは一つ方向性を間違うと一時の流行で終わってしまい社会に対して何も残らず終わってしまう危険も十分あると考えられる。
なぜ、ブームとなったのか、それはまさに人間との協調を図った新しい分野であり、人間にとって魅力的であり好奇心を駆り立てる存在であるからだと考えられる。つまりまさにここ数年ににおいてはロボット自体への認識が変化しようしている。またその認識の変化につれて教育面においても大きな変化が訪れようとしている。「創造性教育」「ものづくり教育」などを筆頭にロボット製作を通じて自らが物作りを行い、自らがロボットに命を与える。その過程を通じて人は問題解決能力を養いロボットを単なる機械と認識するだけでなく人間のパートナーとして認識することができるようになる。ロボットを作ることによってロボット・物と機械との調和を保ち新しい物を生み出す。また、それにより人間の進化が物作りから発展したことを理解することができる。

新しい時代をどう切り開いていくか、それは人の感性力・創造力に託されている。アミューズメントロボットというの新しい分野において、21世紀のロボット技術が、真に人間社会に寄与できるかどうかは、人間の大きな進化の過程であると考えられる。

１．２ 本研究の意義

今回制作したロボット「ＵＦＯ」は次世代ホーム型ロボットを目指したものである。つまり人間との協調をテーマとし人と共に生活を営むこと目指したものである。

本ロボットであるＵＦＯは駆動輪に車輪を使うのではなくボール型アクチュエーターを採用することにより、全方向移動を可能としている。つまり、人間の居住空間・病院等などの複雑なフィールドにおいても容易に移動を可能することができる。制御系にはより一層、人間を大切にすることを踏まえ３ＣＣＤカメラを搭載し複雑なフィールドを画像としてロボット自信が認識・判断し行動に移すことを実現している。また、超音波センサー・赤外線センサーを駆使することによって人や物との接触を極力避け回避することを可能としている。
その他にロボット自らの行動を人が理解できるように音声機能を追加しより一層アミューズメント性を高めたロボットとなっている。

本ロボットであるＵＦＯは一見単なる玩具として見られがちであるが人が親しみを持って接してくれることを期待できる。また今後、ボール型アクチュエーターの制御技術は介護ロボット等のホーム型ロボットの移動技術として大いに期待できる。

１.３ 本論文の構成

本論文の構成について簡単に述べる。

第２章では、制作したロボット「ＵＦＯ」のメカ構造を述べる。主に駆動系としてボール型アクチュエーターの全方向移動機能についての概要を述べる。

第３章には画像処理・ロボット制御と２つに分別を行い電気・制御システムについて、主にロボットを制御している制御ボード、制御アルゴリズムを含んだプログラムと平行した形でこのシステムの概要を述べる。

第４章は、結章として本ロボットの今後の展開を述べる。

参考文献

[1] 日本ロボット学会誌Vol.16, No.3

[2] ロボットの未来　別冊宝島481

第２章　認知反応型ロボットのメカ構造

本ロボット「ＵＦＯ」の概観を図１に示す。ボディーは強化ＦＲＰ（繊維強化プラスチック）によって作られているために非常に頑丈である。図２はボディーを除いたロボット正面から見た写真である。また、図３はロボットを真横から見た写真である。

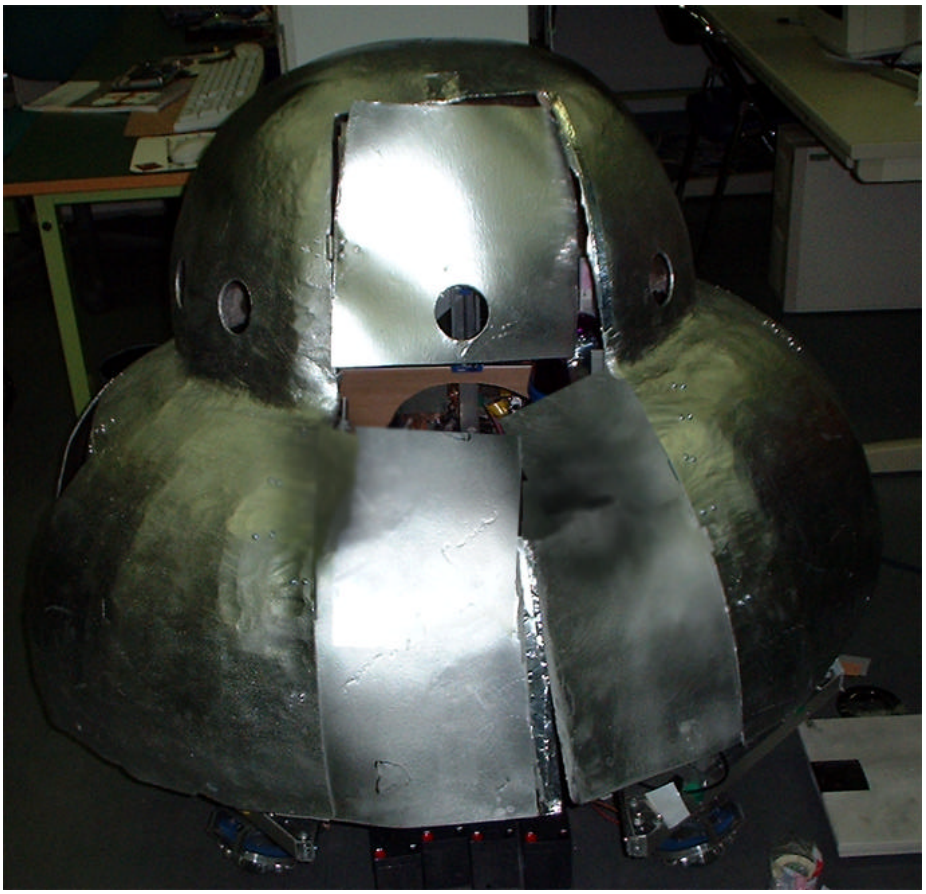


図１

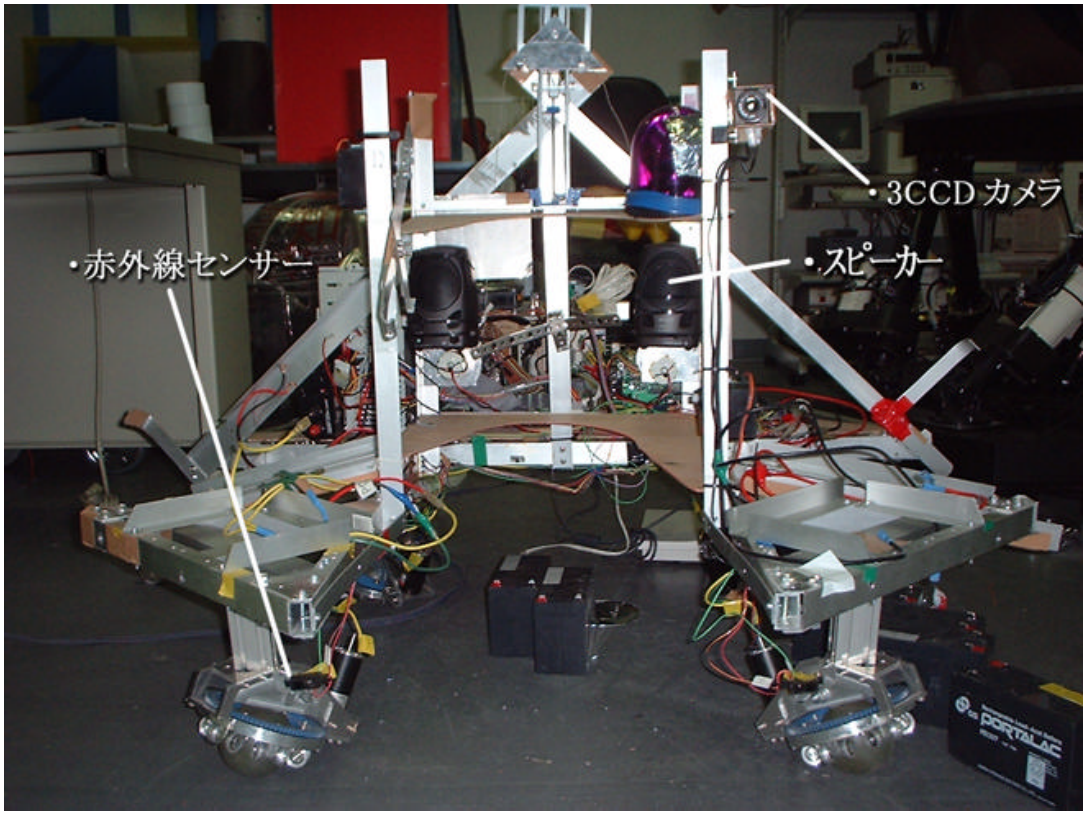


図２

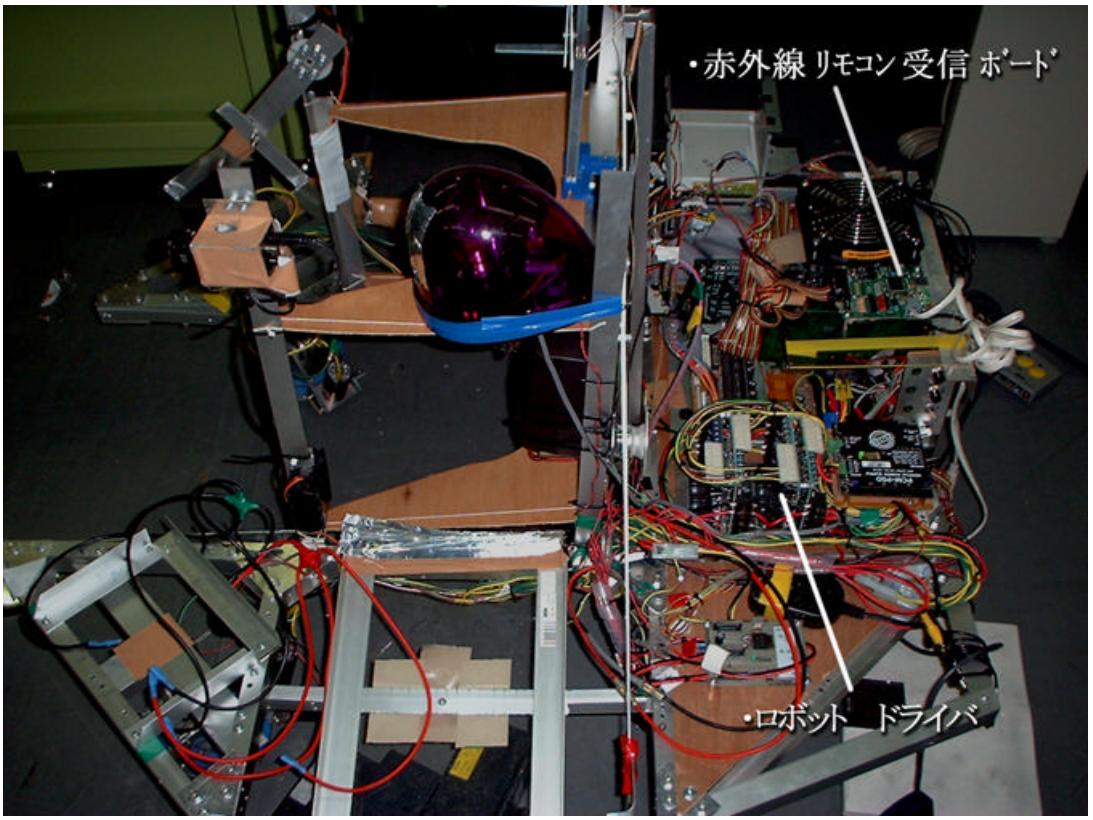
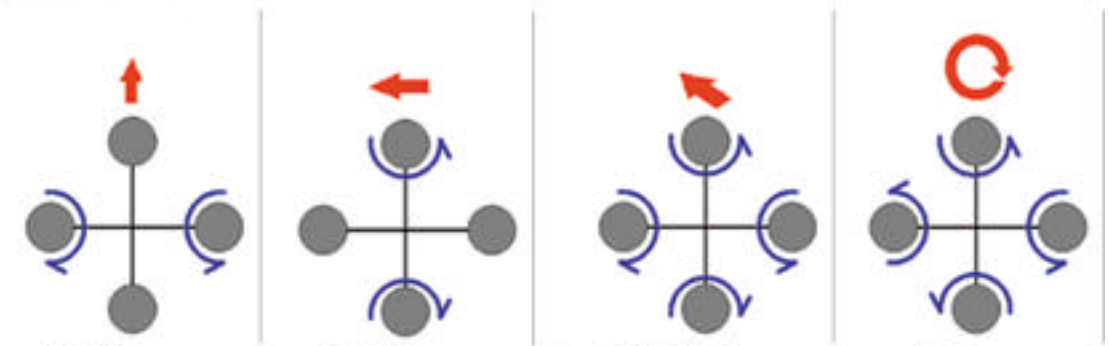


図３

２．１　全方向移動機能について

全方向移動機能とは上部の姿勢を維持したまま、３６０度どの方向にも移動が可能となる機能である。図４に示すように、本ロボットはロボット４隅に対して、駆動輪に車輪を使うのではなく、ボール型アクチュエーターを採用することにより、全方向移動を可能としている。走行の仕組みとして下記にまとめる。



・前後方向
機体前方向に対し左側の球体を時計回り、右側の球体を反時計回りに回せば前進する。
後進時はこれの逆の動作を行う。

・左右方向
機体上側の球体を反時計回り、下側の球体を時計回りに回転させると左横に進む。
右横移動時はこれの逆の動作を行う。

・左斜め方向
上図の様に球体を回転させることにより左斜め前に移動する。
全ての球体をこの逆に回せば右斜め後ろに移動する。

・旋回
上図の様に球体を回転させる事によりその場で旋回する事ができる。
全ての球体をこの逆に回せば逆回転する事ができる。

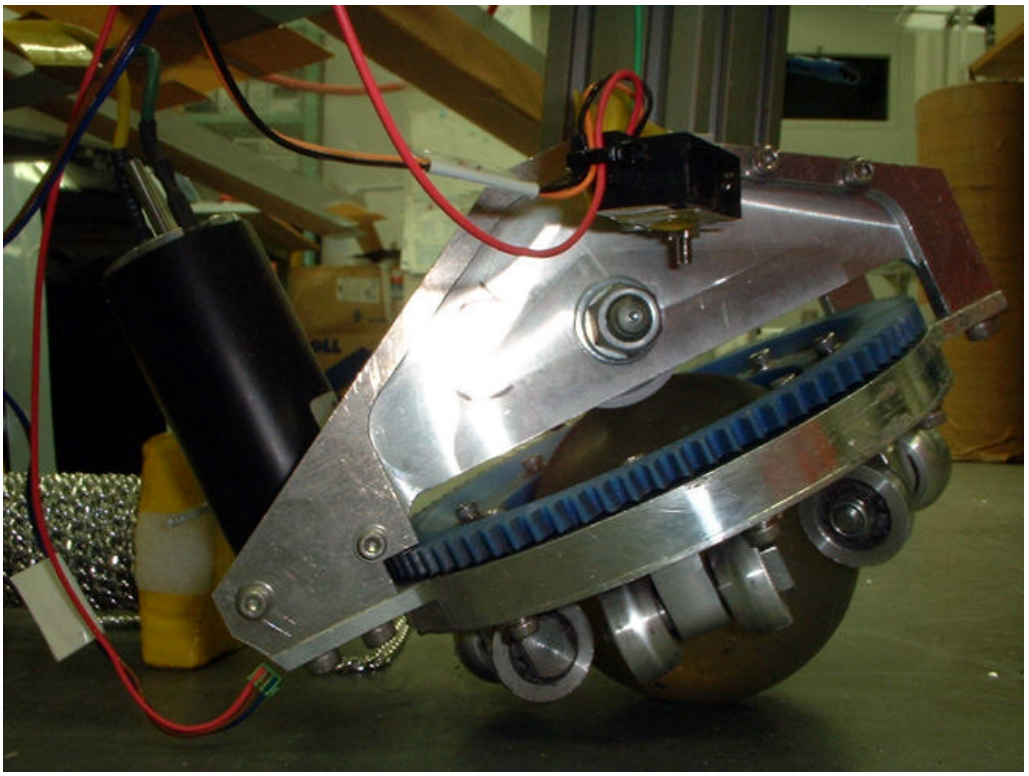


図４

図５に示すように本体底部に配置した駆動ボールを（矢印①）方向に回転させる事により移動が可能となる。また、（矢印②）方向はモータの動きに関係なく自由に動くことができる。この事により本体を一定の方向に向けたまま全方向移動が可能となる。

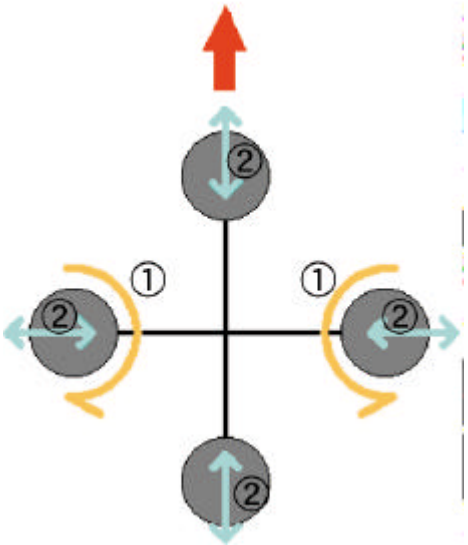


図５

図６はボール型アクチュエーターのより詳細な構造図である。

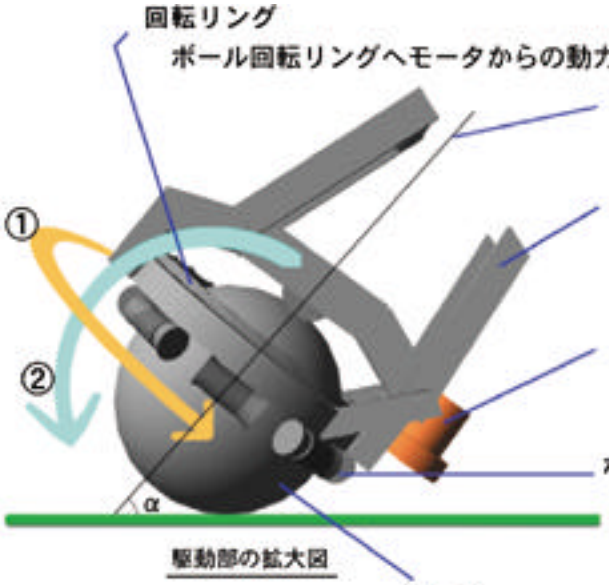


図６

第３章　認知反応型ロボットの機能

３.１　電気・制御システム

図７と図２・図３を合わせて考えると３ＣＤＤカメラは図７ではトラッキングボードへとつながり、スピーカーはサウンドカード、赤外線リモコンボードと赤外線センサーはロボットインターフェイスボードのＵＰＰ部、Ａ/Ｄ部にそれぞれ接続されている。図８に示すようにロボットの制御を可能としている各制御ボードはマザーボードによってすべて管理されている。具体的にはボードの図８と図７を見比べることによりおのおののボードがどういった接続をなしているか認識することができる。

３.１.１ システム構成について

本ロボットは画像処理を中心とおいた処理方法を円滑に行うために図８に示すようにパーソナルコンピューター用マザーボードを採用した。つまり、従来型１０４バス仕様の制御ボードより格段の処理能

力を発揮できプログラミング面においてもより多くのメモリを使える環境下で開発が可能となる。また、より一層マルチメディア面に力を注ぐためにＯＳにはWinodwsを採用する。これは今後、将来的な開発面を想定した場合、誰にでも親しみやすいＯＳ、強いては私自信でなくて別の人間が開発に当たった場合においても順応できるためにこれを選択をした。また、Windowsに準してコンパイラにはＶｉｓｕａｌ Ｃ＋＋を使用する。ロボットの仕様を下記にしめす。

ロボットの仕様	
・画像処理ボード	FUJITSU製PCIﾄﾗｯｷﾝｸﾞﾎﾞｰﾄﾞ
3CCD ｶﾒﾗ	RF_SYSTEMlab Pro9
・ロボット用ｲﾝﾀﾌｪｰｽﾎﾞｰﾄﾞ	FUJITSU製　A/D⇔D/A UPPボード
・赤外線リモコン受信ボード	ﾀﾞｲｾﾝ電子工業
・赤外線センサー	ベストテクノロジー
・超音波センサー	ベストテクノロジー
・サウンドカード	Creative SoundBlaster128
・MotherBord	GIGABYTE GA-6VA7
CPU	Intel Cleron 633Mhz
Memory	DIMM 128M * 2

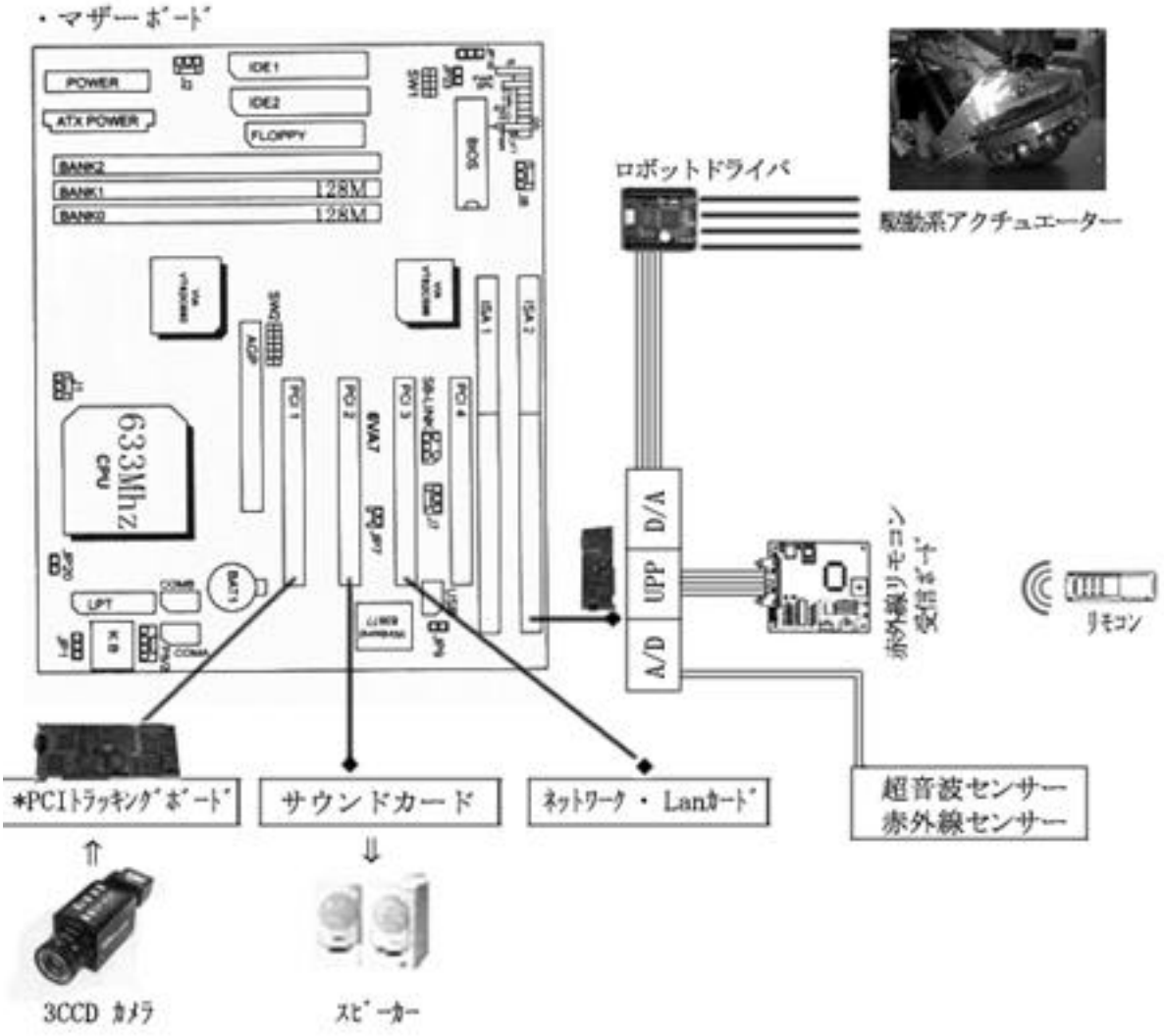

・マザーボード
POWER
ATX POWER
IDE1
IDE2
FLOPPY
BANK2
BANK1 128M
BANK0 128M
BIOS
633Mhz CPU
ロボットドライバ
駆動系アクチュエーター
D/A
UPP
A/D
赤外線リモコン受信ボード
リモコン
*PCIトラッキングボード
サウンドカード
ネットワーク・Lanカード
超音波センサー
赤外線センサー
3CCD カメラ
スピーカー

図７

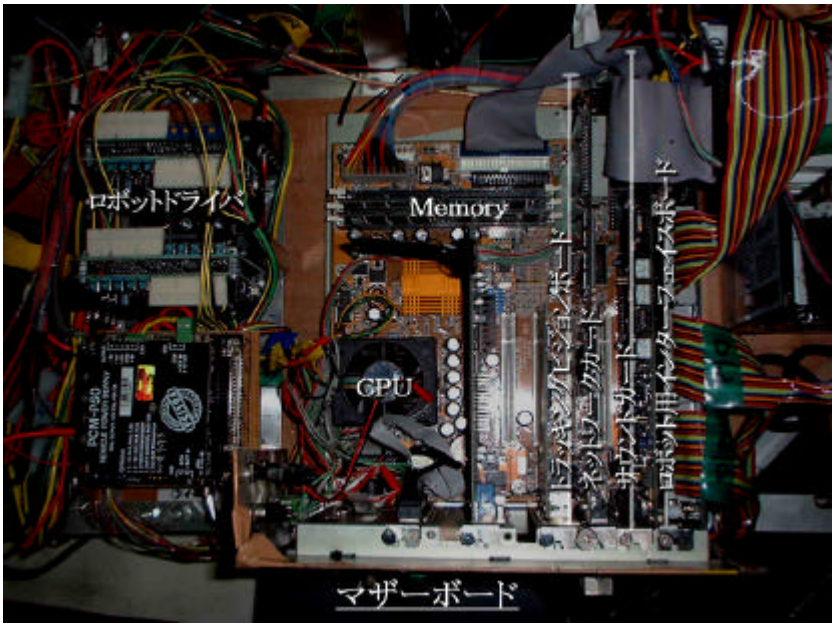

ロボットドライバ
Memory
CPU
トラッキングビジョンボード
ネットワークカード
サウンドカード
ロボット用インターフェイスボード
マザーボード

図８

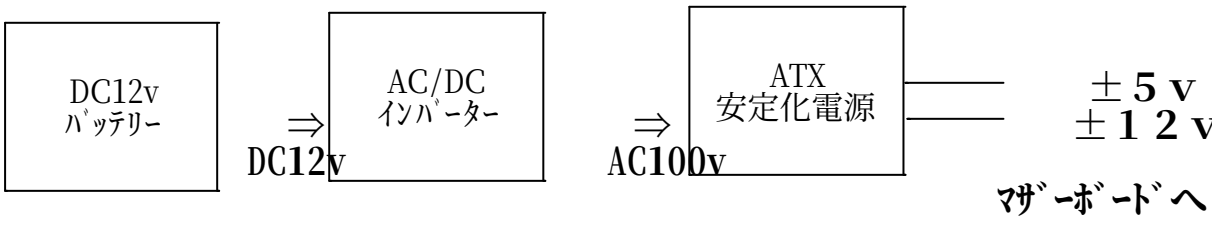
３．１．２ 制御電源について

図７はＰＣ用マザーボードを中心に見取り図を示したものである。各機能について述べる前にマザーボードへのパワーユニット（通電）部について述べる。

このＧＩＧＡＢＹＴＥ＿ＧＡ－６マザーボードは従来型ＡＴ電源コネクタと現行ＡＴＸ電源コネクタの２つを有する構成となっている。従って、ＡＴ電源コネクタに対して±５ｖ・±１２ｖの通電を行うことにより、このマザーボードは機能する。

制作当初、ＣＯＳＥＬ社のＤＣ⇒ＤＣコンバーターを駆使しＤＣ１２ｖバッテリーから±５ｖ・±１２ｖを作り通電を行った。しかしながら、トラッキンビジョンボード・ロボット用インタフェースボード・サウンドカードの３つを順序良くＷＩＮＯＤＷＳに導入をした結果。サウンドカードが認識をした場合、トラッキングビジョンボードが認識しなくなり。トラッキングビジョンボードが認識すると今度はサウンドカードが認識になくなる状態に陥った。上記症状において考えられる原因としてＰＣのＩＲＱの競合であると私は認識をした。そこでカード順を入れ替え再度、ＯＳ上での認識を試みた。しかしながら依然として症状の回復には至らなかった。次に考えられる対策としてはＰＣＩバス用サウンドカードを諦めＩＳＡバス用サウンドカードに入れ替えを行った。なぜなら、ＩＳＡバスは並列に存在しているバスなのでカードの場所に依存しないものであるからである。この対策によって症状は回復しＯＳ上にて３つのカードはどれも認識する結果となる。しかし、

電源部を１２ｖバッテリーから通電を行うのではなく安定化電源（ＡＴＸ電源）からの通電を行うとＰＣＩサウンドカードを含む全てのカードは素直に認識をし。動作もまったく問題がないという結果になった。ではいったいどこに問題があるのか。テスターにてＣＯＳＥＬ社のＤＣ⇒ＤＣコンバーターのチェックも行ったがそれらしき問題は見当たらない。ちゃんと電極の先からは±５ｖ・±１２ｖの通電が行われている。これ以上、問題が解らないと判断した私は、ＩＳＡバス用サウンドカードの使用へと切り替え１２ｖバッテリーからの動作を確立させた。
その後、ＯＳ上にて認識をしているボードへの通電テスト行う。しかしながらＯＳは非常に不安定状態に陥り結果としてトラッキングビジョンボードのアクセスモジューラを破損することとになる。次にＤＣ⇒ＤＣコンバーターを使用することを諦めＤＣ１２ｖからＤＣ⇒ＡＣインバーターを使い。ＡＣ１００ｖに変換した後にそのＡＣ１００ｖを安定化電源（ＡＴＸ電源）へ入電することにより安定した電源を確保するという対策をとった。解りやすく下記に示す。



３.２ 画像処理について

3.2.1カラートラッキングビジョンＰＣＩ

画像処理は富士通株式会社のカラートラッキングビジョンＰＣＩを用いて行う。図９に示すこのボードは、特定のターゲットをビデオレートで追跡することができる画像処理システムである。このボードは専用ハードウェアを用いることにより、ＲＧＢカラーでの相関演算を３３ｍｓの間に５００回以上計算することで複数のターゲットを追跡することが可能となり、１テンプレート最大１２８×１２８画素による複数のテンプレートとの比較を可能としている。トラッキングビジョンの柔軟なメモリアクセス機能は、画像メモリには２つの画像メモリ間での高速ＤＭＡ転送機能によってすばやくデータのやり取りを行えるようになっており、動きベクトル・オプティカルフローの計測や、パターンマッチング処理をビデオレートで実現している。

アミューズメントロボット「ＵＦＯ」にこの動きベクトルの測定機能を用いることによってロボットは任意の作業を画像によって判断することが可能となる。動きベクトルとは、特定のターゲットが、どちらにどれがけ動いたかを表す量である。例えば図１０のようにフレームｆでp^fの位置にあったターゲットが次のフレームにおいてp^{f+1}の位置に移動したとすると、p^fとp^{f+1}を結ぶベクトルの動きベクトルがｖとなる。

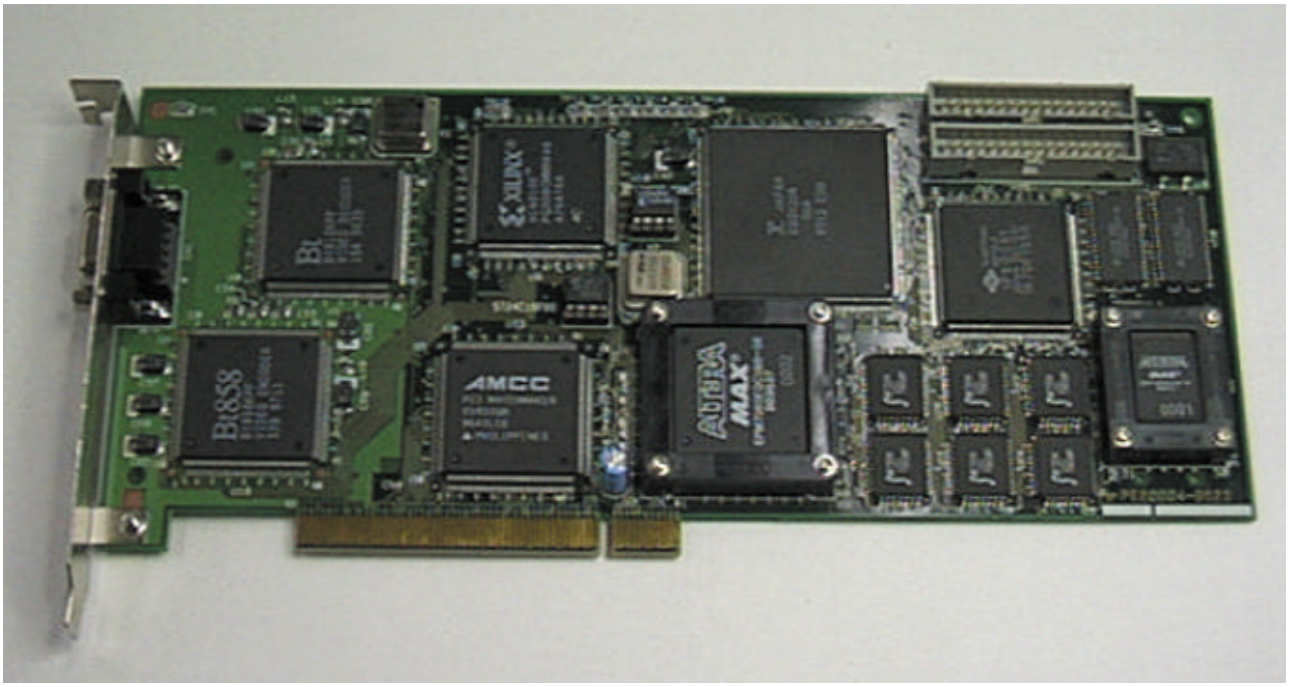


図9

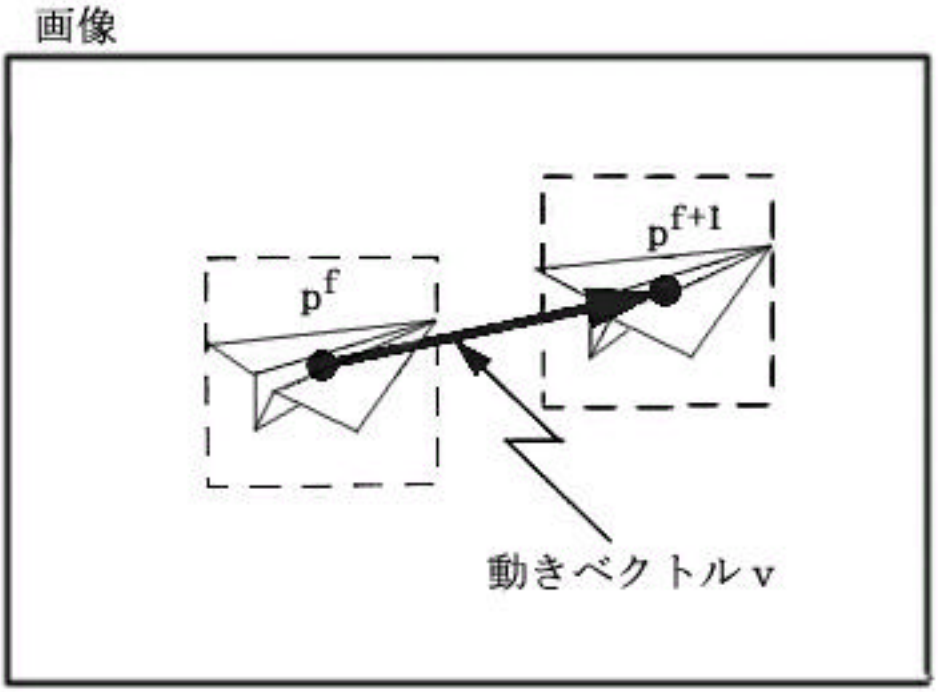


図１０

カラートラッキングビジョンはカメラからの入力画像を時間を追って画像メモリに記憶し、高速ＤＭＡにより相互比較することで、特定のターゲット領域が次の瞬間（時間）どこへ移動したのかを判断することが可能である。具体的にターゲットとなる物のポイントを囲む矩形領域を考え、この画像ブロックに対して述べる相関演算と呼ぶ処理を実行し動きの対応を求めるものである。相関演算とは図１１に示すように追跡対象とする矩形領域を参照ブロック（テンプレート）として抽出した上で、次のフレームの候補ブロックとの間で、画像の相関値を計算するものである。候補ブロックを２次元的にスキャンしながらこれを繰り返し、相関が一番高い位置を求める。これによって特定の目標が次のフレームでどこに動いたかを示す動きベクトルを知ることができる。尚、候補ブロックがスキャンする領域全体を候補ブロックと呼ぶ。具体的に相関演算では、以下の式で示すブロック内の画像ごとの濃度差の絶対値による総和を求めている。

$$D(u, v) = \sum_{x,y=0}^{7} \left| R(x, y) - S(x + u + 8, y + v + 8) \right|$$

Ｒ（ｘ，ｙ）は参照ブロック、Ｓ（ｘ、ｙ）は探索ブロックの画像データである。Ｄ（ｕ、ｖ）を動きベクトルに対応するディストーションと呼び、ディストーション値がもっとも小さくなる位置をもっとも相関の高い位置とみなし、動きベクトルを求めている。カラートラッ

キングビジョンでは、参照ブロックサイズは８×８画素を基本としており、動きベクトルの探索範囲（ｕ、ｖの値）は、－８～＋７画素に固定される。その結果、探索ブロックサイズは２３×２３となっている。参照ブロックの位置や探索ブロックの位置は、パラメータとして指定することが可能である。また、複数のターゲットに対する動きベクトルの追跡・測定は１フレーム（３３ｍｓ）の間に、異なる参照・探索ブロックに対して複数回の相関演算を行うことによって可能となる。カラー画像を参照した追跡処理・大型テンプレートでの追跡処理を可能とするため、カラートラッキングビジョンでは、累積相関と呼ぶ方式を用いている。

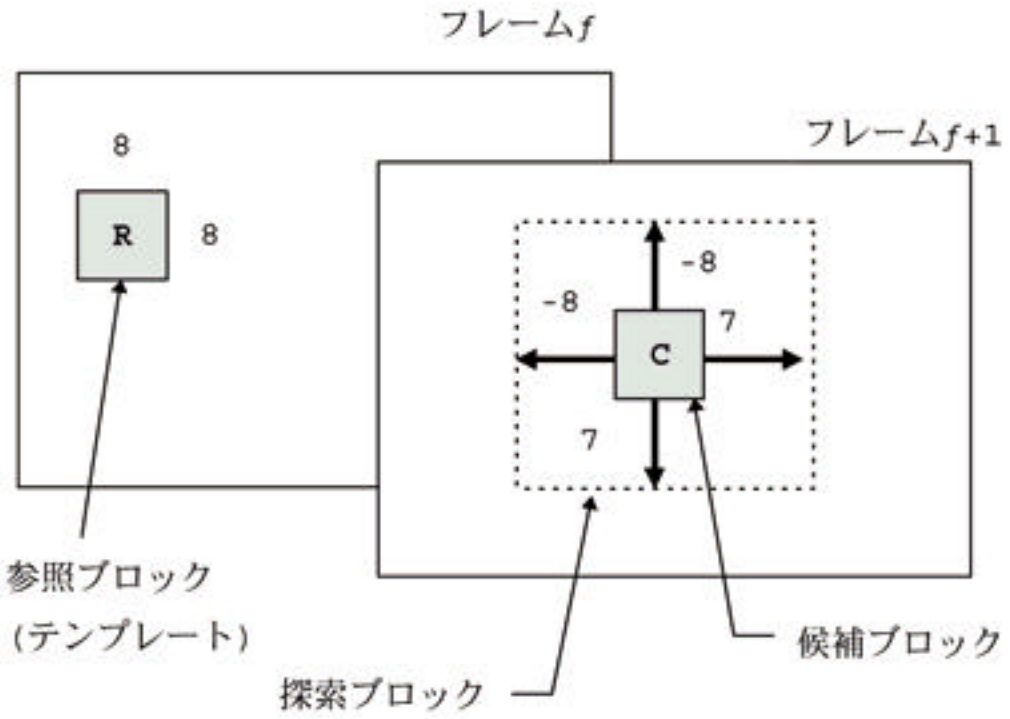


図１１

累積相関とは下記に示す式のように、直前の相関値$D_{k-1}(u,v)$に今回の相関演算結果を加算することで、新しい相関値$D_k(u,v)$を検出できる演算である。

$$D_k(u,v) = D_{k-1}(u,v) + \sum_{x,y=0}^{7} | R(x,y) - S(x+u+8, y+v+8) |$$

この演算を実行するためにカラートラッキングビジョンでは図１２に記するように相関演算回路の後段に２ポートの画像メモリを搭載している。

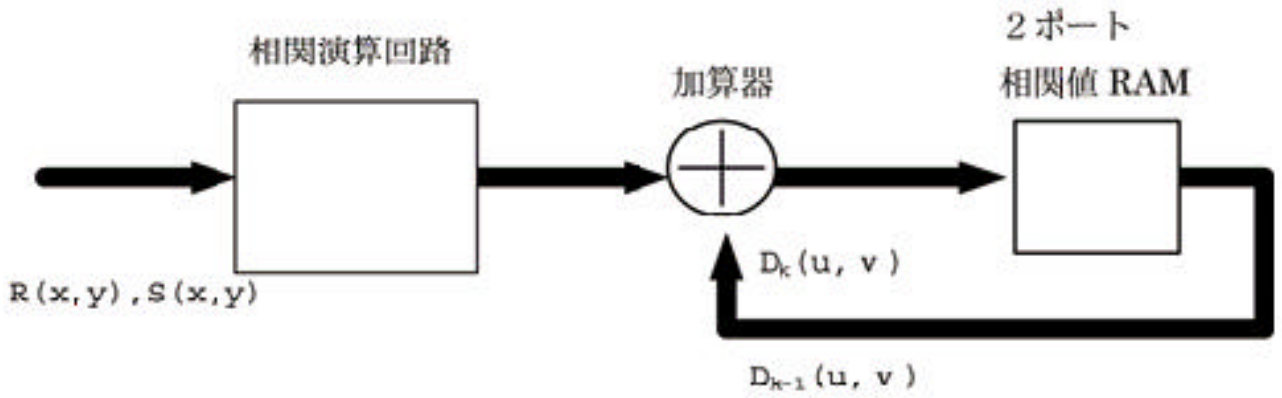


図１２

累積相関を用いることによって、図１３に示すようにＲＧＢ各プレーンごとに相関演算を行わない結果を累積することで、カラー画像を用いた追跡処理が可能となる。また、複数の部分テンプレートでの演算結果を累積することによって大型テンプレートを等価的に実現している。例えば、図１４に示すように１６×１６画素のテンプレートで相関演算を行いたい場合、参照ブロックを４つの８×８テンプレートに分割し順次相関演算を行う。この際、探索ブロックの位置も図のように順次ずらす。そして、４回の演算で得たデイストーションを全て累積することにより等価的に１６×１６画素テンプレートに対するディストーションを得ることが可能となる。しかしながらこの方法においてはＲＧＢ３つのテンプレートに対する処理を行うことになるので１つのプレーンで処理を比較した場合、３倍の処理時間を必要とすることになる。また、１６×１６画素のテンプレートに対する処理は８×８画素のテンプレートと比較した場合、４倍の処理時間を必要とすることになる。プログラミングを行う際、機能と処理速度のトレードオフを行い、用途に応じた適切な処理を選択しなければならない。

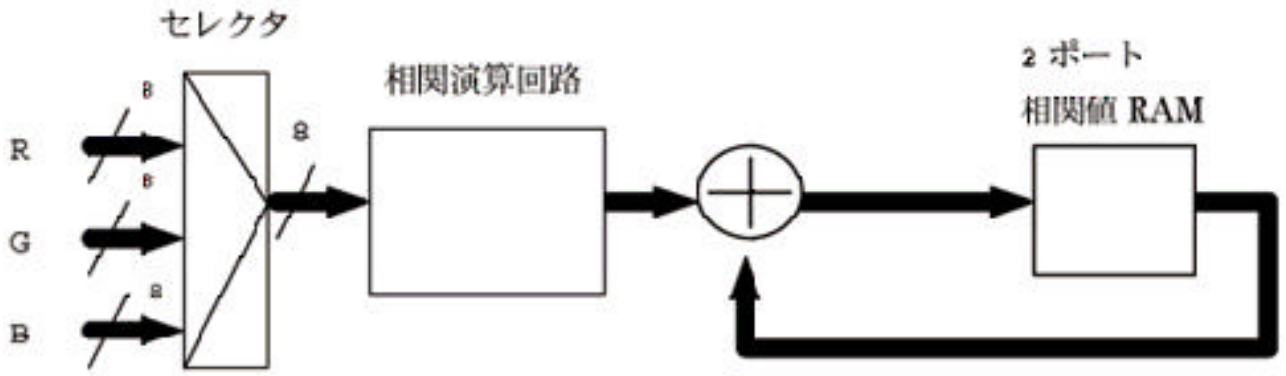
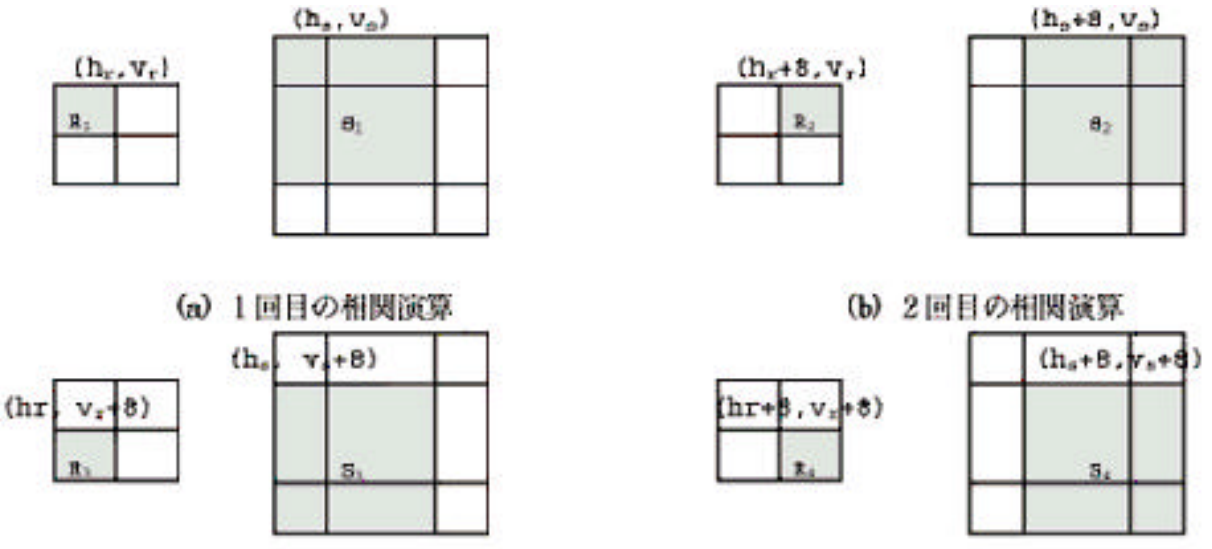


図１３



(8×8 ブロックの相関演算を 4 回繰り返し実行し、16×16 ブロックの累積相関演算を行なう例)

図１４

３.２.２カラートラッキングビジョンＰＣＩの応用

以下のプログラムUFO()は１個のウインドウに対して追跡処理を実行する最も基本的なプログラムである。追跡はカラー画像を参照し、カラーでの相関演算を繰り返し行いながら実行する。追跡ポイントは画像の中央に固定されている。また、追跡に必要なテンプレート画像を取得するために、最初に１枚静止画を取り込んでから、追跡処理を開始する。

```
void UFO(){

    int bsize,mag,roff,soff,h0,v0,h,v,rh,rv,sh,sv,I,I2=0,I3
    int dh,dv,mdist,dhz=0,dvz=0,mdistZ=0,devn=0,H=0,V=0;
    int X=0,Y=0,z,Z=0;
    double   volt,a,b,c,d,e,f,g,i,j,k,S1,S2,S3,S4,S5,S6;

    /*初期パラメータ*/
    f=300;                  //追跡時間をフレーム数で指定
    bsize=32;               //参照ブロックサイズ
    mag=2;                  //拡大率（間引き率）
    h0=320;                 //参照ブロックの中心座標（H）
    v0=240;                 //参照ブロックの中心座標（V）
    h=h0;                       //探索ブロック初期中心座標（H）
    v=v0;                       //探索ブロック初期中心座標（V）

////参照・探索ブロックの中心から左上座標を計算するオフセット
    roff = bsize*mag/2;//32
    soff = roff+8*mag; //48

/////////////////初期/////////////
TrvInut();
TrvSetSyncMode(EXTERNAL);
TrvVideoThrough();

TrvSetSample(devn,mag-1,mag-1,mag-1,mag-1);//間引き率指定
TrvFetchImage(devn,0);                     //参照画面取り込み
TrvVideoInput(devn,DOUBLE_BUF,0);          //画像入力開始
```

```
    for(;;){
            TrvWaitNextFrame(0);

            //////////カラー相関演算///////
            rh = h0 - roff;     //320-32 = 288
            rv = v0 - roff;     //240-32 = 208
            sh = h - soff; //320-48 = 272
            sv = v - soff; //240-48 = 192

        TrvColorCorAny(
            devn,
            bsize,//参照ブロック水平サイズ（８の倍数を指定）
            bsize,//参照ブロック垂直サイズ（８の倍数を指定）
            VRAM_REF,//参照ブロックを取り出すVRAM番号
            VRAM_SER,//探索ブロックを取り出しVRAM番号
            rh,//298参照ブロック水平アドレス（左上角）
            rv,//208参照ブロック垂直アドレス（左上角）
                sh,//探索ブロック水平アドレス（左上角）
            sv,//探索ブロック垂直アドレス（左上角）
            0,//累積指定０：累積しない１：累積する
            0,//参照ブロックマスクデータＩＤ番号
            0,//探索ブロックマスクデータＩＤ番号
            &dh,//（出力）動き水平ベクトル
            &dv,//（出力）動き垂直ベクトル
            &mdist//（出力）最小相関値
            );
        TrvDrawBox( h-roff, v-roff,h+roff, v+roff, 0);

        h += (dh * mag);    //新しいカーソルを書く
        v += (dv * mag);
        TrvDrawBox( h-roff, v-roff,h+roff, v+roff, OVL_RED
    }
        TrvVideoStop(0);    //画像入力停止
}
```

UFO()では、最初に幾つかのパラメーター設定行う。f=300は３００フレームの間追跡を行うことを指定している。bsizze=32、mag=2は３２×３２画素の参照ブロックを用い、拡大率２で追跡を行うことを指定している。拡大率を２としているので見かけ上６４×６４画素のテンプレートで追跡をしているように見える。h0=320、v0=240は、参照

ブロック（テンプレート）の中心座標であり、ｈ、ｖは探索ブロックの中心座標である。このプログラムは最初に取り込んだテンプレートを固定的に使うのでｈ０、ｖ０は一定である。一方、ｈ、ｖはターゲットの動きに応じて順次更新されることになる。尚、参照ブロックと探索ブロックの実際の指定は、中心座標でなく左墨の座標で行うことになっている。このために中心から左隅を算出するための定数roffとsoffを下記の用に事前計算する必要がある。

```
roff = bsize *  mag / 2;
soff = roff + 8 * mag;
```

カラートラッキングビジョンはTrvInit()に始まる３つの関数、TrvSetSybcMode()、TrvVideoThrough()によっては初期化される。初期化後、TrvSetSample(0,mag-1,mag-1,mag-1,mag-1)によって間引き率をセットする。TrvSample()を一旦実行すると、それ以降の全ての相関演算でこの間引き率に従った処理が実行される。尚、TrvSetSample()で指定する間引き率は、０が間引き無しを意味するものであるので拡大意率ｍａｇから１を引いて与える必要がある。また、間引き率は、参照ブロックと探索ブロックで独立に指定でき、かつ水平方向と垂直方向の間引き率を独立に指定することができる。

次にTrvFetchImage()を実行し、図１５に示すように画像メモリ#0に静止画を取り込む。カラートラッキングビジョンを用いて追跡処理等を実行するためにはビデオ画像の入出力を正しく制御し、これに同期して処理を行う必要がある。

具体的に,TrvVideoInput(int devn, int mode, int mem)を実行することによって、図１５に示すようにカメラからの動画像が順次画像メモリに保存される。この時、引数modeとmemで指定したモードに従って、メモリローテーションが行われる。ローテーションには２つのモードがあり、ここでは２画面切り替えモード(DOUBLE_BUF)を用いる。また、画像入力中はカメラ画像がモニタへスルー表示される。

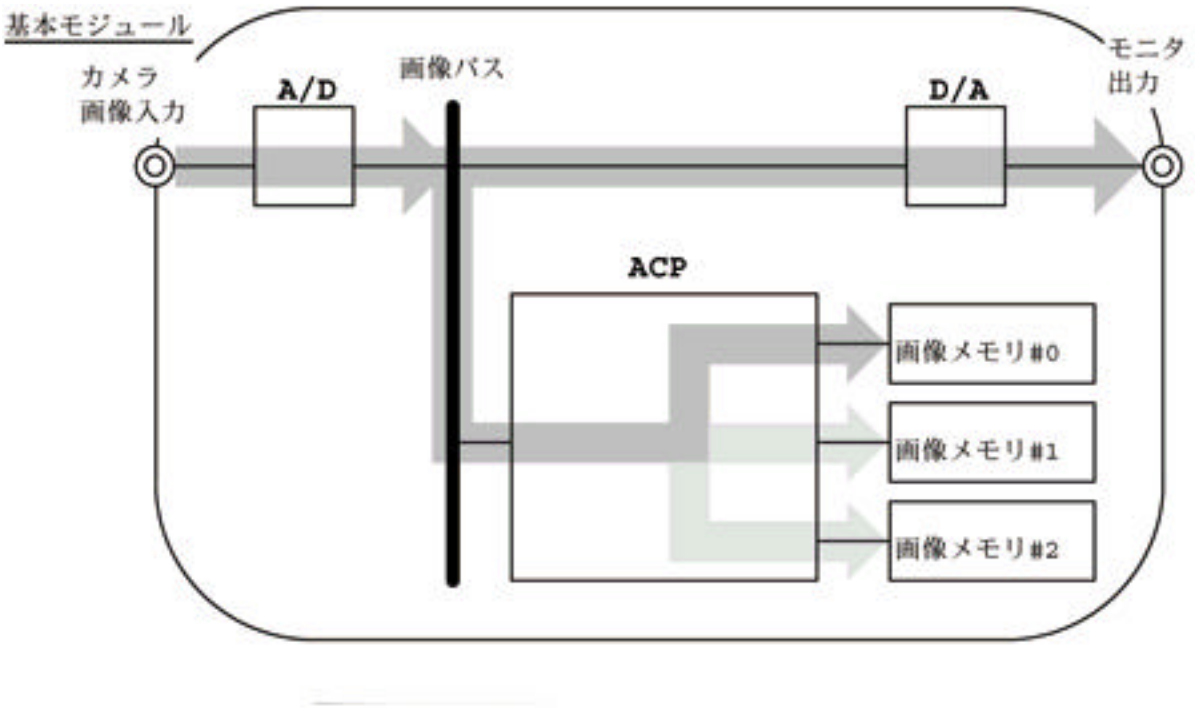


図１５

画像入力を開始した後、for文の中で、毎フレームごとに行う処理を記述する。

```
for(;;){
        TrvWaitNextFrame(0);
        ……
        ……
        /*追跡処理*/
        ……
}
```

TrvWaitNextFrame()はフレームの開始地点を待ち合わせる関数である。この関数により、画像信号に同期しながら、処理を繰り返すことができる。一連の追跡処理は原則として１フレーム（３３ｍｓ）の間に終了するように設計されなければならない。尚、２フレーム毎（６６ｍｓ毎）に処理したり、逆に１/２フレーム毎に処理したりというふうにフレーム周期を変えることも可能である。

for文の中では実際以下の関数が相関演算を行っている。

```
TrvColorCorAny(    0,
                bsize,
                bsize,
              VRAM_REF,
              VRAM_SER,
                    rh,
                    rv,
                    sh,
                    sv,
                     0,
                     0,
                     0,
                  &dh,
                  &dv,
                  &mdist
);
```

TrvColorCorAny()は、任意の参照ブロックサイズでカラー相関を実行する関数であり、ＲＧＢ３つのプレーンについて順次相関演算を行いその結果を累積した後、動きベクトルを求める。相関演算の結果、ｄｈ、ｄｖに動きベクトル（ｄｈ、ｄｖ＝－８～７）がｍｄｉｓｔに最小ディストーション値がセットされる。

TrvDrawBox()は、オーバーレイプレーンにブロックカーソル（四角の枠）を描画する関数である。前フレームで描画したカーソルを消すために、値０のカーソルを描画し、次に新しい座標でカーソルを描画する。

最後のTrvVideoStop()は、動画像の入力を停止するためのものである必ずしも実行する必要はないが省エネのためにアイドリング状態に戻す方が好ましいと考えられる。

下記の式において、具体的な制御パラメーターｈ、ｖに着目して考えると。

```
h += (dh * mag);
v += (dv * mag);
```

ｍａｇは拡大率であり。ｄｈ、ｄｖはTrvColorCorAny()の関数によって、求められた対象物の動きベクトルである。故にｈ、ｖはリアルタイムに更新されたモニターに映る対象物の現座標である。この座標値を元に制御を行う。

３.３ ロボット制御

３.３.１　ロボット用インタフェースボード

本ロボットのモーター系つまり動力系への命令伝達はすべて図１７が示すロボット用インタフェースボード（以下、ＲＩＦ－０１）が担っている。このボードは、図１６に示すようにＩＳＡバス規格のハーフサイズ・ボードに、アナログ入力３６チャンネル、アナログ出力１６チャンネル（－１０ｖ～＋１０）、デジタル入出力３２チャンネルを搭載したロボット用インタフェース・ボードである。

入出力制御チップ部としてD/A変換器１２ビット/８チャンネルのＭＰ７６１３（ＥＸＡＲ社製）を２個搭載している。A/D変換器は１２ビット/８チャンネルのＡＤ７８９１（アナログ・デバイス製）を２個搭載している。さらに、日立製ＵｎｉｖｅｒｓａｌＰｌｕｓｅＰｒｏｃｅｓｓｏｒ以下ＵＰＰも搭載しいる。

インタフェース部としてベース・アドレスはジャンパにて０００Ｈ～０７Ｆ８Ｈの範囲でＩＳＡバスのアドレス空間のうち、わずか８バイト分の領域のみを占有し任意に設定できる。

ＲＩＦ－０１　構成図を下記に示す。

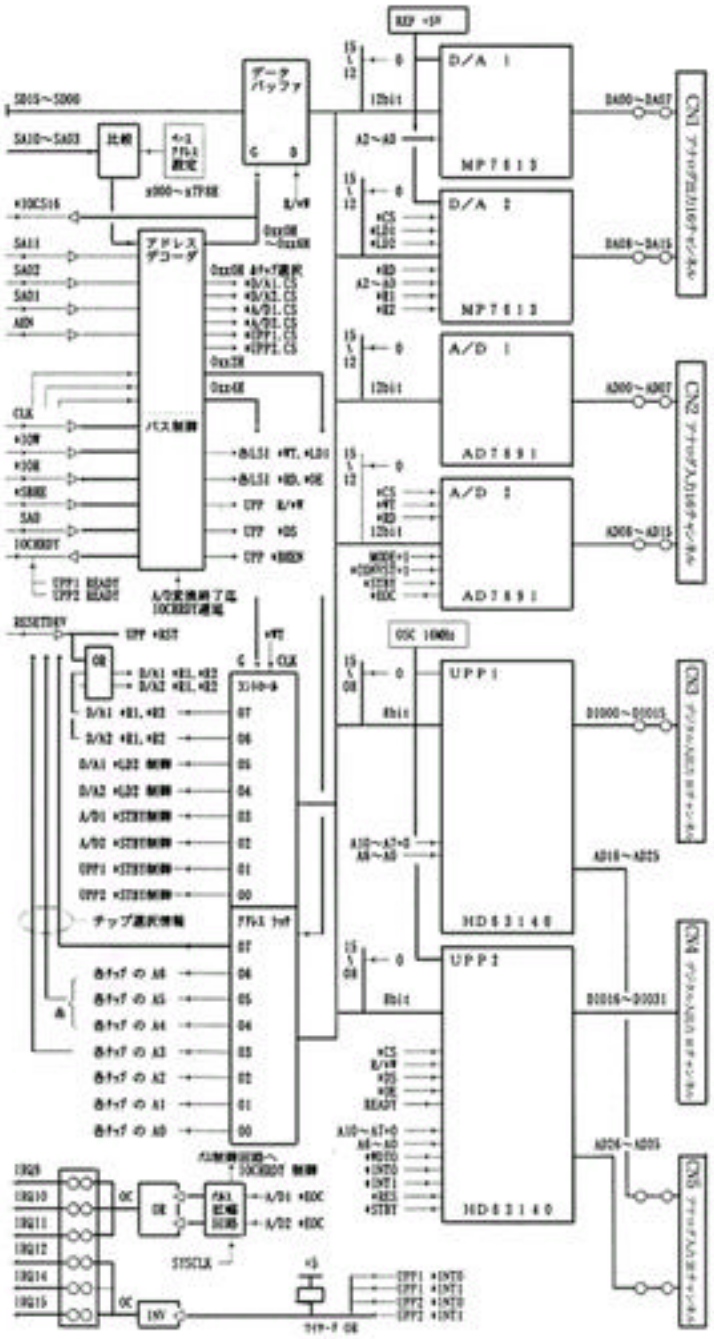


図１６

ＲＩＦ－０１を下記に示す。

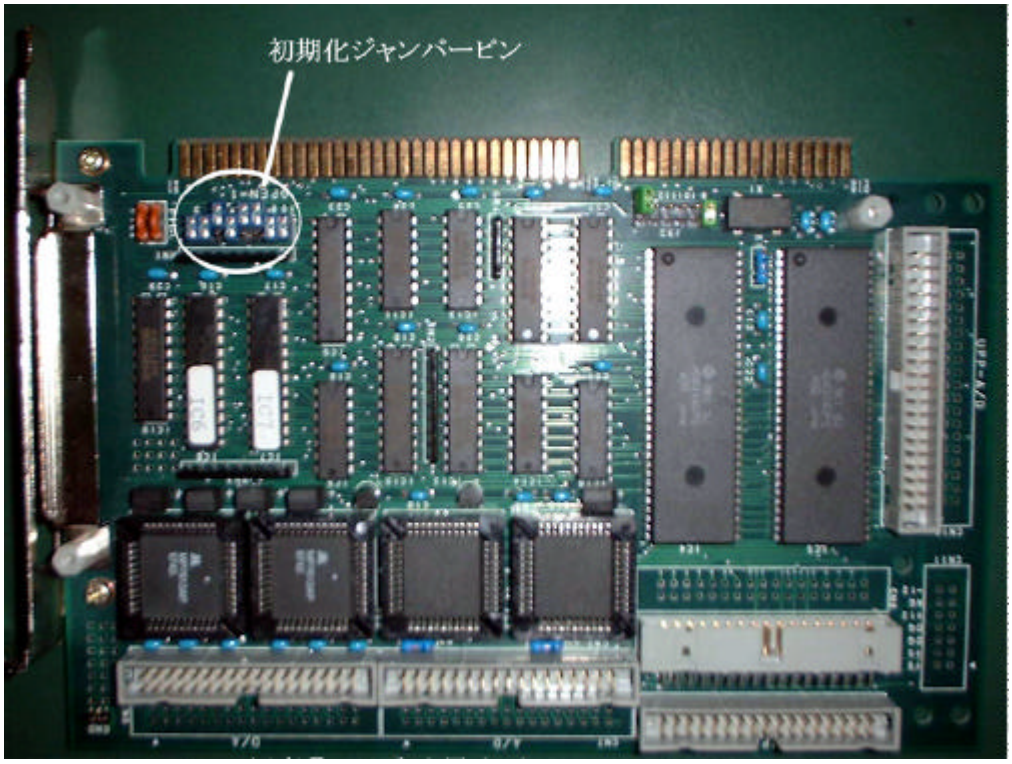


図１７

本ロボットは図７が示すようにＲＩＦ－０１ロボット用インタフェースのD/A・A/D・ＵＰＰの機能を使用する。D/Aは駆動用モーターを直接制御してるロボットドライバへと命令伝達を行う。A/Dについては外部センサーからのアナログ電圧信号の読み取りを行っている。ＵＰＰについては赤外線リモコンからのパルス信号を解析している。基本的にＲＩＦ－０１は電圧・パル

ス信号の読み取り、測定を行う。測定を行い具体的なデータの形で読み取りを行うために下記に示す４つのＣ言語サブルーチンが必要である。

```
////////////ボードの初期化/////////////////////
int regst( unsigned baseadr)
{
     reg_data=0xcf;
     _outpw(baseadr+4, reg_data); //ｼﾞｬﾝﾊﾟｰとの認証

}


int regstoutpw(unsigned baseadr ,unsigned int regstno)
{
     reg_data=regstno;
     _outpw( baseadr, reg_data);
     return 0;
}


/////////  D/A出力  ////////////////////////////
double daoutpw(unsigned baseadr,int portno,double value
{
     int da_data;
     if( (portno >= 0)&&(portno <= 15) ){
       da_data = (value + 10.0) / 20.0 * 4096.0; //2048

       if (da_data <    0) da_data =    0;
       if (da_data > 4095) da_data = 4095;
       _outpw(baseadr+2, 0xf0+portno);
       _outpw(baseadr, (unsigned)da_data);
     }
     else{
       printf("チャンネルEra\n");
     }
}
```

```
//////A/D ｲﾝﾌﾟｯﾄ ////////////////
double adinpw( unsigned int baseadr ,unsigned int portn
{
     double        volt;
     int       ad_read_data;
     unsigned int   ad_write_data;
     ad_write_data = portno*8+5;

     if( (portno >= 0)&&(portno <= 15) ){

        _outpw(baseadr+2, 0x70+portno);
        _outpw(baseadr  , portno*8+5);
        for(i=0;i<10; i++) ad_write_data=i;
        ad_read_data = (signed)_inpw( baseadr );

        if(ad_read_data & 0x0800) ad_read_data |= 0xf00

        volt = (double)ad_read_data / 2048 * 10;
        return(volt);
     }
     else{
        erron=(1);
        return(erron);
     }
}
```

・ボード初期化

ＲＩＦ－０１ボードの初期化をするにいたってregst(base_adr)のサブルーチンを使用する。_outpw(baseadr+4, reg_data)によってあらかじめ図１７に示すようにＩＳＡバス上のジャンパーピンの設定状態とreg_data=0xcf=base_adrという状態を一致させる。この命令をメイン文に記載することによってこのボードを待機状態に維持することができる。尚、今から下記に述べる機能は全てボードの初期化を必要とする。

・D/A

このボードのＤ/Ａ部の機能を使用する場合、上記の初期化手順によってこのボードへのアクセスを行う。その後、例えば図１６に示すようにＣＮ１アナログ出力のチャンネル１に対して＋５ｖの出力を行う場合、daoutpw(base_adr,1,5.0)となる。また、チャンネル５に対して-７ｖの出力を行う場合、daoutpw(base_adr,1,-7.0)となる。

・A/D

Ａ/Ｄを使用するに至ってadinpw()のサブルーチンを使用する。Ａ/Ｄ部はアナログ電圧の入力であり、Ｄ/Ａ部はアナログ電圧の出力である。故にＣ言語上で読み取った電圧値を測定するためにサブルーチンの最後で下記の命令を実行する必要がある。

```
volt = (double)ad_read_data / 2048 * 10;
return(volt);
```

このサブルーチンの戻り値であるvoltは０ｖ入力の場合、２０４８という値がad_read_dataということになる。つまり、ad_read_dataは－１０～＋１０間の測定に対して０～４０９６の変動がある。その変動を電圧の変動値としてわかりやすく表示を行うために上記の計算を行っている。具体的にv=adinpw(base_adr,0)と指定する事により図１６に示すＣＮ２アナログ入力チャンネル１の電圧信号を読み取ることができる。

・**ＵＰＰ**

ＵＰＰの使用にあたって下記の特別な初期化設定が必要である。

```
////////////UPP初期化/////////////////
_outp(base_adr+2,0x11);                    //////←1
_outp(base_adr,0xff);                      //////←2

_outpw(base_adr+2,0x03);                   //////←3
upp= _inpw(base_adr)-32768;                //////←4
```

図１６に示すＣＮ３ＵＰＰ１６チャンネルのデジタル入出力端子の１～８チャンネルを下位ビット、８～１６チャンネルを上位ビットと定める。上記ソース１よって下位ビット８に対して、ＭＰＵ（直接入力）モードに設定を行う。次に、２によりＭＰＵポートに指定する端子を１６ビットの形で指定を行う。この場合、ｆｆという指定により全ての端子がＭＰＵ端子として設定されることとなる。各チャンネルのモード設定を行った後、上記ソース３により上位ビットからの信号を判断するのか下位ビットからの信号を判断するのかを指定する。つまりＵＰＰはマイクロコンピューターである故にA/Dなどのような入力信号を直接受け渡すのではなく。そのまま使用できるデータの形に変換する処理を施すことができる。故にここでは下位８ビットを指定することにより入力されたチャンネルをビット変換した形を上位ソース４のような命令により取得することができる。

３.３.２　ロボット用インタフェースボードの応用

本ロボットでＲＩＦ－０１ボードを使用した具体的な制御方法について述べる。図７ようにＲＩＦ－０１ボードを介して電圧信号のやり取りを行っている機能はD/A・A/D・ＵＰＰであり、それぞれ具体的な接続部として図１６に示すＣＮ１・ＣＮ２・ＣＮ３がそれに相当する。図７に示すように、D/Aの接続先はロボットドライバであり、A/Dへの入力にはセンサー類である赤外線センサー・超音波センサーがあり、ＵＰＰは赤外線リモコンボードからの信号が入力されている。

・D/A⇒ロボットドライバ

本ロボットの駆動用モーター電圧を直接制御しているロボットドライバを図１８に示す。また、その構成図を図１９に示す。このロボットドライバは図１９に示す１１、１２（ＣOM、ＲＥＦ）にD/Aからの信号を入力することによりＶｄｄ、０に接続されたＤＣバッテリーをエネルギー源として駆動用モーターの制御を行う。ロボットドライバ自体の制御モードとして「速度制御モード」・「電流制御モード」があり、図１９に示すＳＷ１の切り替えによりモードを選択することが可能ととなる。本ロボットは速度制御モードを利用してモーター制御を行う。

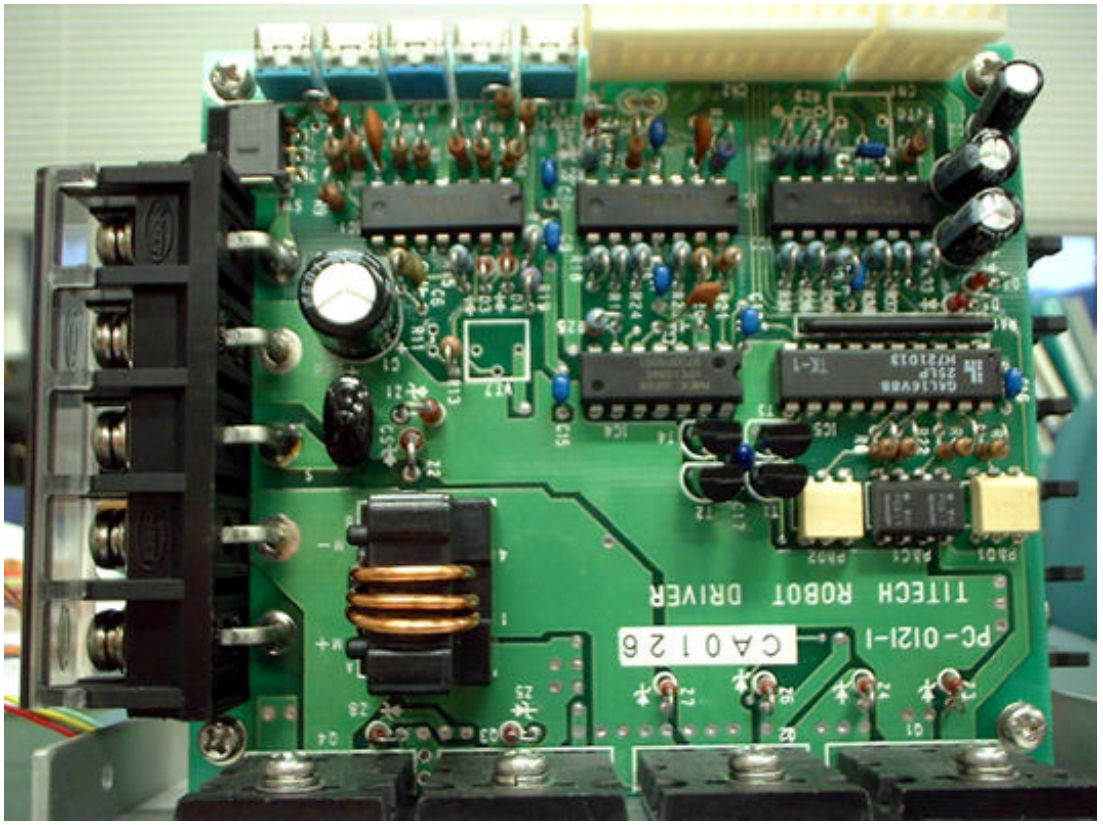

TITECH ROBOT DRIVER
PC-0121-I
CA0126

図１８

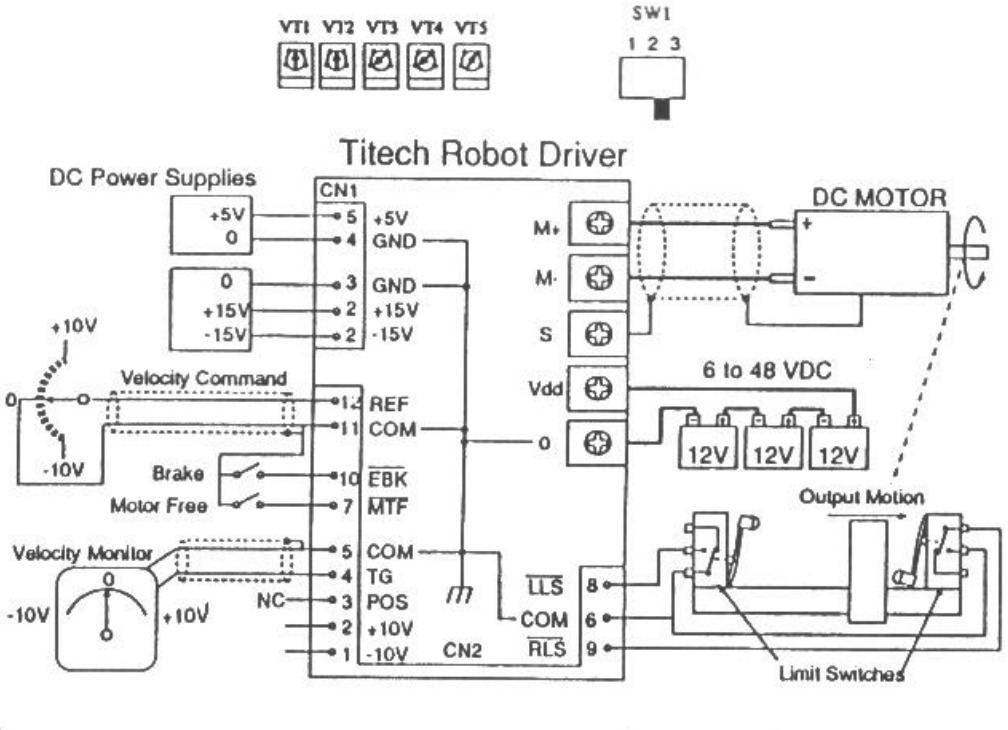

VT1 VT2 VT3 VT4 VT5
SW1
1 2 3
Titech Robot Driver
DC Power Supplies
+5V
0
0
+15V
-15V
CN1
5 +5V
4 GND
3 GND
2 +15V
2 -15V
+10V
0
-10V
Velocity Command
12 REF
11 COM
Brake
10 EBK
Motor Free
7 MTF
Velocity Monitor
0
-10V
+10V
NC
5 COM
4 TG
3 POS
2 +10V
1 -10V
CN2
M+
M-
S
Vdd
0
LLS 8
COM 6
RLS 9
DC MOTOR
+
-
6 to 48 VDC
12V
12V
12V
Output Motion
Limit Switches

図１９

具体的に制御対象となるモーターは４つ存在するのでロボットドライバは各々４つ必要となる。故にD/Aチャンネルは CH１～４チャンネルを使用する。各チャンネルの信号端子とグラウンド端子を図１９に示す１１、１２（COM、REF）に接続し、下記に示すC言語プログラムによりロボットドライバへの命令を行う。

```
daoutpw(base_adr,1,0.0);  //出力 CH1⇒ロボットドライバ１
daoutpw(base_adr,2,0.0);  //出力 CH2⇒ロボットドライバ２
daoutpw(base_adr,3,0.0);  //出力 CH3⇒ロボットドライバ３
daoutpw(base_adr,4,0.0);  //出力 CH4⇒ロボットドライバ４
```

上記Cソースはロボットドライバに対して０ｖを入力している。故にドライバに接続されているモーターは停止状態を維持する。速度、制御モードを用いることにより４つのドライバに対して例えば、下記のようなプログラムを実行すると

```
daoutpw(base_adr,1,10.0);  //出力 CH1⇒ロボットドライバ１
daoutpw(base_adr,2,10.0);  //出力 CH2⇒ロボットドライバ２
daoutpw(base_adr,3,10.0);  //出力 CH3⇒ロボットドライバ３
daoutpw(base_adr,4,10.0);  //出力 CH4⇒ロボットドライバ４
```

ロボットドライバは－１０ｖ～＋１０ｖまでの勾配により接続されているDCバッテリーにより速度制御を行うので、例えば接続されているDCバッテリーがDC２４ｖだとするとMAXの２４ｖをモーターに流し制御を行う。また、D/Aからの値がこれと逆の－１０だとするとドライバは－２４ｖをモーターに供給し制御を行うことになる。つまり、モーターはD/Aからの電圧信号が０ｖだと停止、＋信号を正回転、－信号を逆回転となる。

また、４つのモーター全て、例えば＋４ｖというＤ/Ａからの信号が入力されると。図１９に示すＶＴ１～ＶＴ５を調節する事によりほど同回転での制御が実現され。ロボットは直進や斜め移動を簡単に実現することが可能である。しかしながら、このロボットドライバの使用に至って細心の注意を払わないといけないことは図１９に示すVdd、０に対して大容量のアンペアを持つＤＣ電源を使用する際、図１９に示す１１、１２（ＣＯＭ、ＲＥＦ）に対してＤ/Ａからの入力電圧がまったくない場合、先にVdd、０にＤＣ電源を接続してしまった場合、ボードの暴走を引き起こしてしまう可能性が十分にあるということである。この点を守って使用すればボード、モーターの破損はまずないと考えられる。実際、本ロボットに積んだ場合。図１９に示すVdd、０にスイッチを取り付けることによって上記の危険性から回避することができた。しなしながら最終的にそのスイッチを押すのはプログラムではなく人間であるので、Ｄ/Ａ入力電圧を確認した後に入電するよう心がけなければならない。

・超音波センサー　⇒　A/D

図２０に示すのは、ベストテクノジー製８方向超音波センサーボードである。また、図２１にその構成図を示す。このセンサーボードは基本的にＡ/Ｄからの入力によって対象物を検知を行うことができる。しかしながら８チャンネルのチャンネル切り替え時においてＤ/Ａからの５ｖ信号を必要とする。

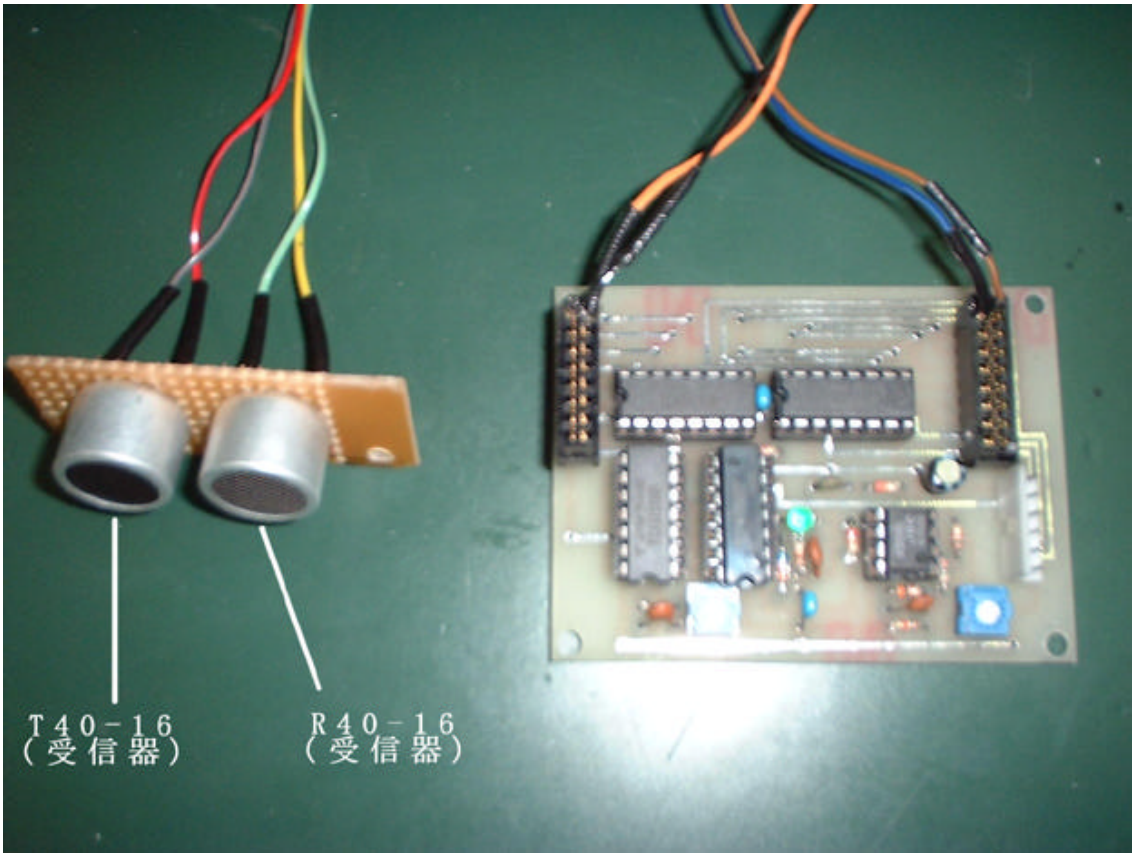

T40-16
（受信器）
R40-16
（受信器）

図２０

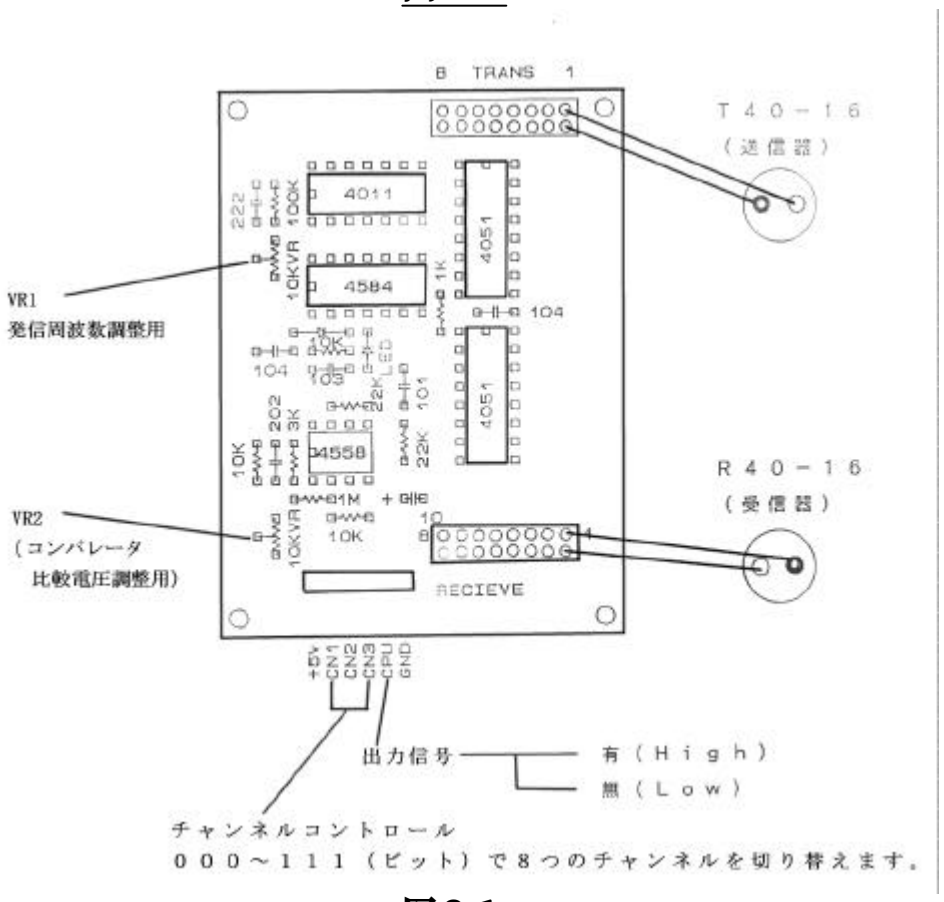

8 TRANS 1
T40-16
（送信器）
VR1
発信周波数調整用
VR2
（コンパレータ
比較電圧調整用）
R40-16
（受信器）
4011
4584
4051
4051
4558
RECIEVE
+5v CN1 CN2 CN3 CPU GND
出力信号
有（High）
無（Low）
チャンネルコントロール
０００～１１１（ビット）で８つのチャンネルを切り替えます。

図２１

具体的に、図２１に示す「チャンネルコントロール」部においてＣＮ１～３に対して５ｖ信号を入力すると接続されている超音波８チャンネルが切り替わる構成になっている。例えば、１チャンネルに接続されている超音波センサーを使用したい場合０ｖ・０ｖ・５ｖをＣＮ３・ＣＮ２・ＣＮ１に対してA/Dから入電を行う。これによりボード回路構成がチャンネル１に対する処理を開始する。その構成を下記の表に示す。

	CN3	CN2	CN1
1ch	0v	0v	0v
2ch	0v	0v	5v
3ch	0v	5v	5v
4ch	5v	0v	0v

・
・
・

通常超音波センサーは赤外線センサーと違って対象物までの距離をリアルタイムに測定することができる。しかし、このセンサーボードに関しては予め設定した距離まで近づくと応答がある。つまり、その距離に対して対象物が有るか無いかのみの判断がとなっている。詰まるところ、赤外線センサーと同じ役割しかできない。本ロボット製作段階において当初このセンサーボードを使用した対象物検知を行っていたが、現在は装備していない。

・赤外線センサー　⇒　A/D

図２２・２３に示すのはベストテクノジー製赤外線センサーである。また、図２４にその構成図を示す。

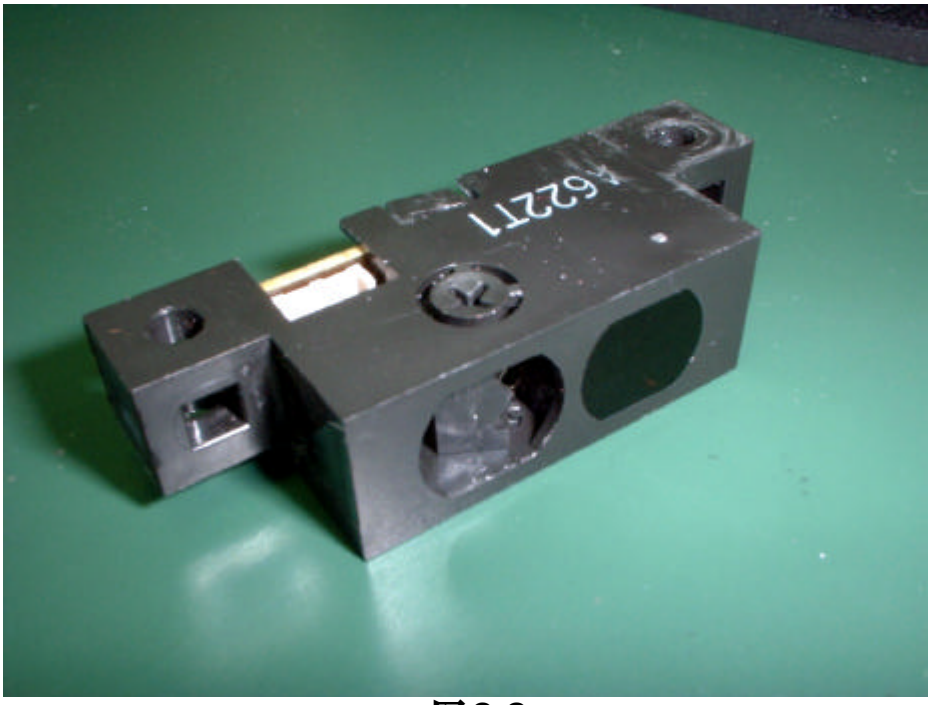


図２２

図２３

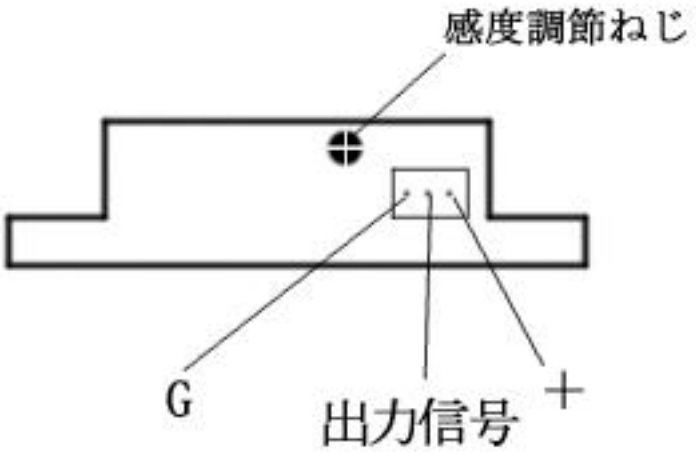


図２４

このセンサーはA/Dへの入力において使用することができる。図２４に示す構成図より右より＋・出力信号・Gとなっている。＋、Gに対してＤＣ５ｖを入電することによりこのセンサーは作動し、図２２に見られる２つレンズ、発光レンズと受信レンズにより対象物を判断し、図２４に示す出力信号より、その判断結果を電圧信号により出力を行う。出力された電圧信号は図７に示すように、ＲＩＦ－０１のA/D部へ接続により読み取ることが可能となる。本ロボットはこのセンサーを８方向に取りつけることによりロボットの周囲に対物があるかないかを判断できる。具体的な使用として下記のＣ言語ソースにより、Ｓ１～Ｓ８にセンサー出力値が代入される。

```
S1=adinpw(base_adr,8);          //入力 CH__8
S2=adinpw(base_adr,9);          //入力 CH__9

S3=adinpw(base_adr,10);         //入力 CH__10
S4=adinpw(base_adr,11);         //入力 CH__11
```

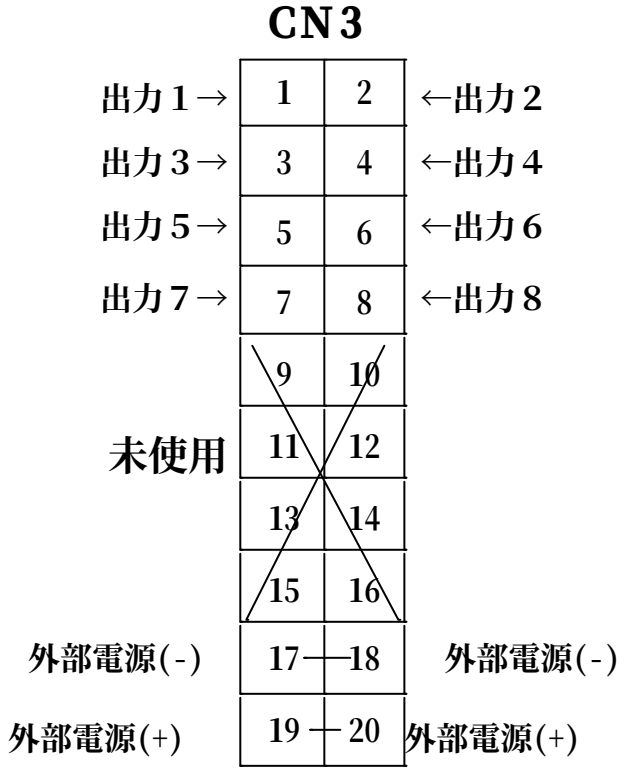
```
S5=adinpw(base_adr,12);          //入力 CH__12
S6=adinpw(base_adr,13);          //入力 CH__13

S7=adinpw(base_adr,14);          //入力 CH__14
S8=adinpw(base_adr,15);          //入力 CH__15
```

このセンサーは図２４に示す感度調節ねじによりセンサーが反応する対象物までの距離を設定することができる。赤外線センサーはリアルタイムにて対象物までの距離測定は不可能である。しかしながら本ロボットに使用する理由として図２０に示す超音波センサーボードにてこの処理を実現しようとした場合には、多チャンネルのチャンネル変更時にＤ/Ａ数チャンネルを使用してしまうからである。

・赤外線リモコンボード⇒ＵＰＰ

図２５に示すのはダイセン電子工業製赤外線リモコン受信ボードである。このボードの構成図は、図２６に示す。本ボードはＲＩＦ－０１ボードのＵＰＰの機能を利用する事によって使用することが可能となる。リモコン自体は１チャンネルから８チャンネルまであり、このボードはリモコンから発せられた赤外線を受信し。図２６に示すＣＮ３より出力を行う。ＣＮ３の構成として下記の表に示す。



上記、表ＣＮ３の構成図より１７・１８、１９・２０は内部で接続されていて外部電源ＤＣ１２ｖを使用しこのボードの電源として使用する。

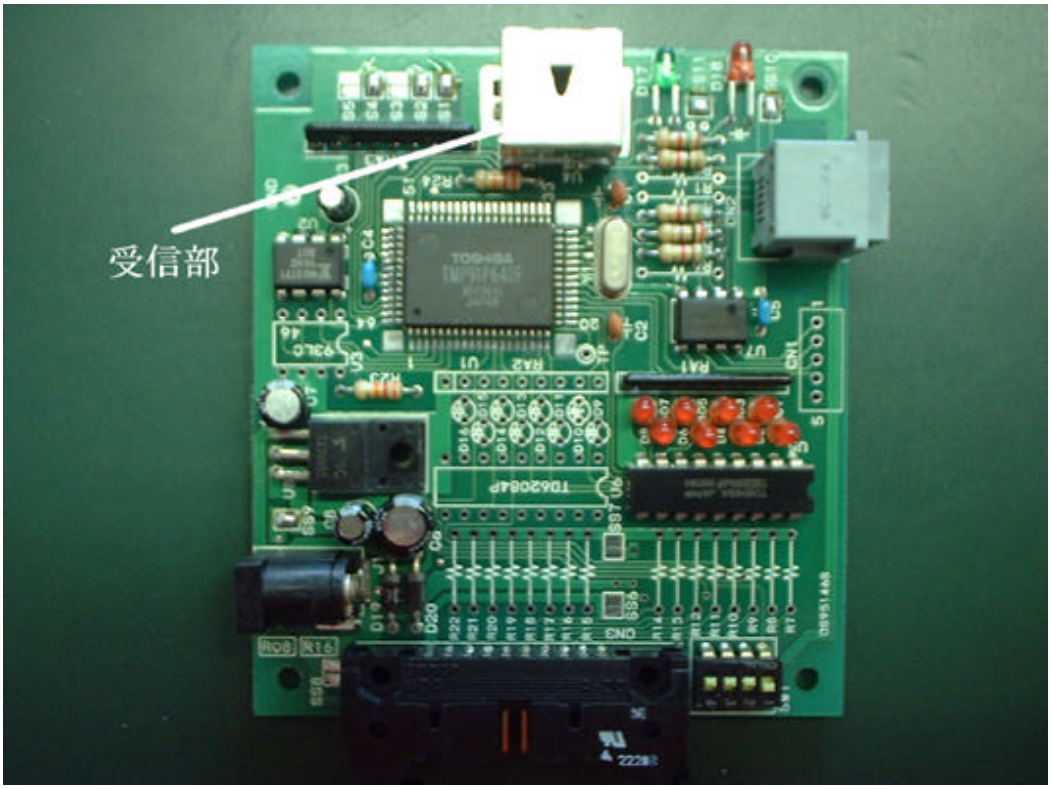

受信部

図２５

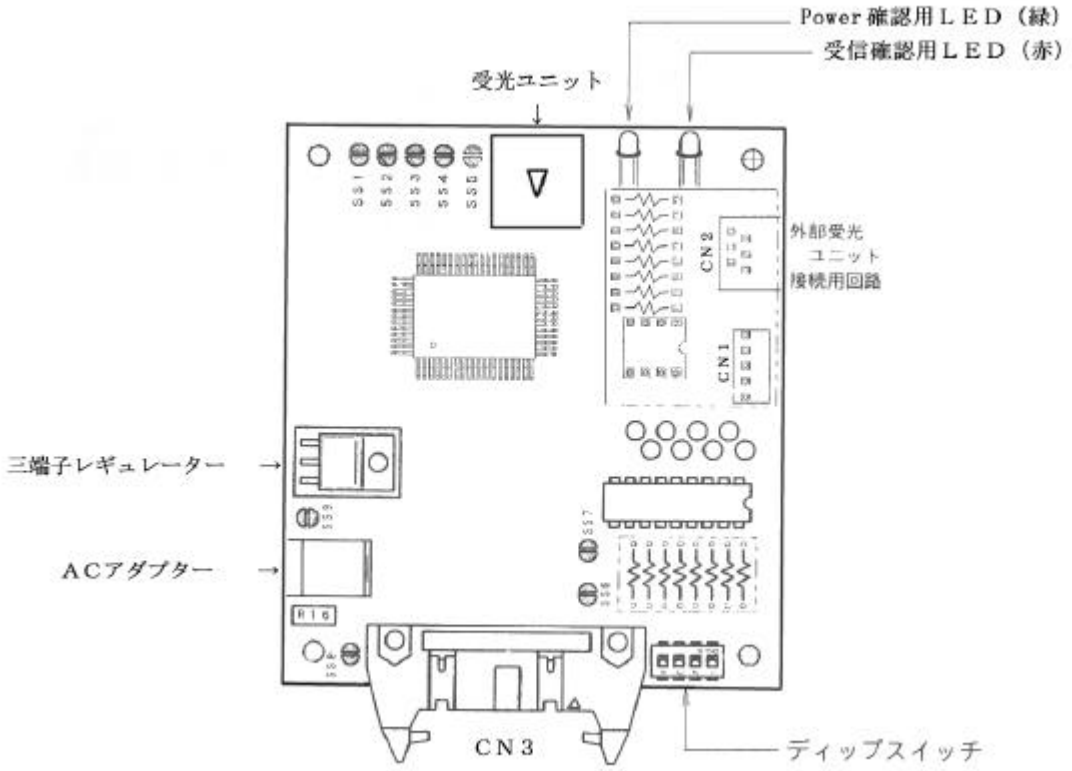

Power 確認用ＬＥＤ（緑）
受信確認用ＬＥＤ（赤）
受光ユニット
外部受光
ユニット
接続用回路
三端子レギュレーター
ACアダプター
CN3
ディップスイッチ

図２６

ＵＰＰと赤外線リモコンボードとの接続として、図１６に示すＣＮ３、ＵＰＰ１６チャンネルのデジタル入出力端子の１～８チャンネル（下位ビットチャンネル）に対して図２６に示す赤外線リモコン受信ボードＣＮ３の出力１～８を接続する。具体的な使用法を下記のＣ言語にそって述べる。

```
////////////UPP初期化////////////////////
_outp(base_adr+2,0x11);                                    /////
_outp(base_adr,0xff);                                            /////
/////////////////////////////////////////

do{
                _outpw(base_adr+2,0x03);
                up = _inpw(base_adr);
                printf("insett\n");
                cls();

}while(up!=1);
```

上記Ｃソースは赤外線リモコン１のボタンを押さない限りｄｏ文から抜けれないアルゴリズムである。この文を利用してロボットを待機状態などを実現することができる。また、下記にリモコンのボタンと実際に返される数値を表にする。

ﾘﾓｺﾝ	ｕｐ
1	1
2	2
3	4
4	16
5	32
6	64
7	128
8	256

３．４　プログラムの流れ

コンパイラとしてＭｉｃｒｏｓｏｆｔＶｉｓｕａｌＣ＋＋Ver.6.
を使用する。本ロボットの全ソースは付属に付け加える。今回製作を行ったＣ言語ソースの流れについて述べる。このソースは赤外線リモコンからの命令を中心としたアルゴリズムで構成している。このソースの簡易略図を図２７に示す。図２７に示す円付き番号は、全て赤外線リモコンの番号である。プログラム起動時、ロボットは図７に示すD/Aから、ロボットドライバへの出力は０ｖ、つまり静止状態を維持している。その後、赤外線リモコンにより各々の動作を行う。例えば、メイン状態から①を押した場合、ロボットの動作はリモコンの指示に全て託される。また、メイン状態から②を押した場合、自動モードとなり、②を押す瞬間に図７に示す３ＣＣＤカメラに写っていた画像を記憶しロボットは単純にその対象物に対して追いかける動作をする。また、その際に対物すると緊急停止をする。

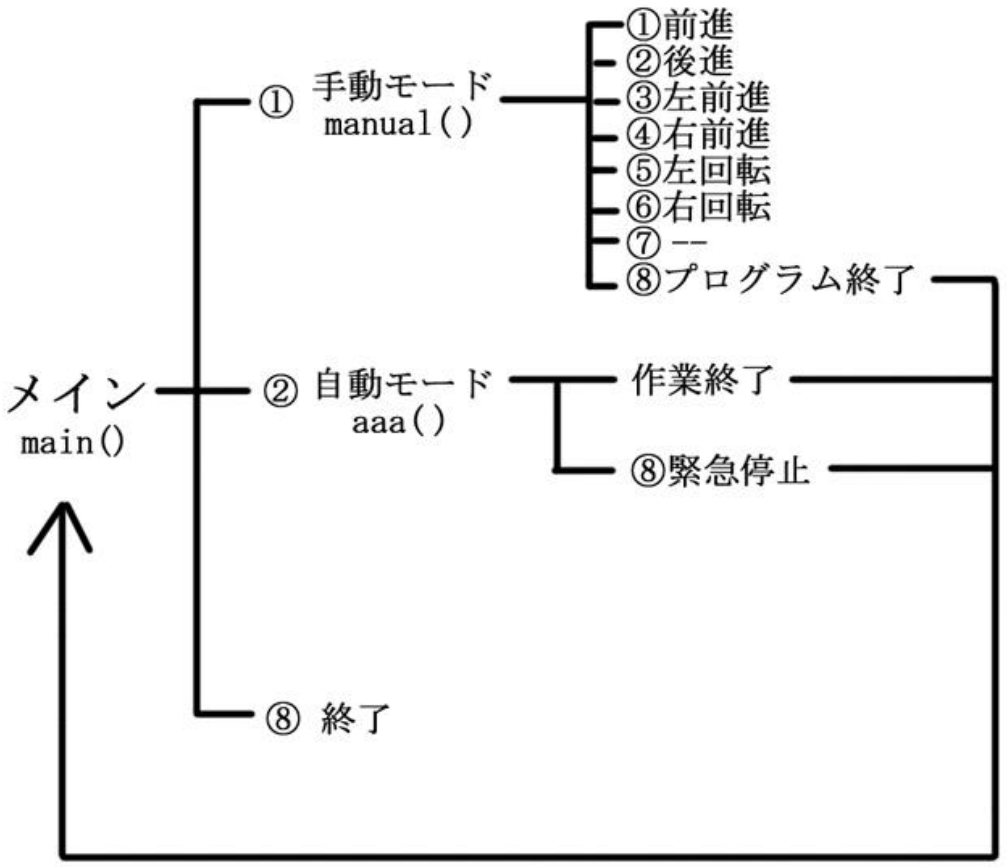


図２７

参考文献

・カラートラッキングビジョンユーザーマニュアル富士通株式会社
・ロボット用インタフェースボード　ＲＩＦ－０１説明書株式会社
・TITECH ROBOT DRIVER 説明書　東京工業大学
・超音波センサーボード・赤外線センサー　説明書ベストテクノロジー

第4章　結　章

本研究では、ロボットの自律制御に成功した。具体的な動きとして、赤外線リモコンの指示により、予め認識させたターゲットを追従することができる。また、その際、ロボット前部に取り付けられた赤外線センサーにより、対象物を認識し、衝突を避ける事が可能である。ホーム型ロボットとしての今後の改良としては、使用する場所に合わせた、最適なプログラミングをすることによって、処理速度の向上が挙げられる。また、現段階において、具体的にどのような作業を行うかは、ユーザーがプログラミングしないといけない。今後はこのプログラミングを学習アルゴリズムによってカバーする形を目指している。

謝　辞

本研究は高知工科大学知能機械システム工学へ在学中にまとめた卒業研究である。本研究を進めるにあたり、ご助言を下さった高知工科大学知能機械システム工学科 全卓樹助教授に心より深謝いたします。また、ロボット製作においてお世話になった高知工科大学知能機械システム工学科 王碩玉助教授及び高知工科大学ロボット倶楽部の部員達に深く感謝をしています。

```
#include "conio.h"
#include <stdio.h>
#include "trvlib.h"
#include <stdlib.h>
#include <stdio.H>
#include <windows.h>
#include <process.h>
#include "mmsystem.h"

//////////////////////D/A ボード用//////////////////////
double daoutpw(unsigned,int, double);                    //
double adinpw( unsigned int,unsigned int);               //
int _outp( unsigned short, int);                         //
unsigned short _inpw( unsigned short );                  //
unsigned int reg_data,baseadr1,baseadr2;                 //
char erron,erroff;                                       //
long int i,jikan;                                        //
long double jikan1;                                      //
unsigned int data;                                       //
unsigned int base_adr;                                   //
#define VecToDaddrs (dh, dv) (((dv)+8 * 16 + ((dh) + 8))//
#define     ND    4                                      //
#define base_adr 0x1a0                                   //
//////////////////////////////////////////////////////////

unsigned a,n;
int sel_da1;
int nPortNum=1;

void main( )
{
      int         a,h,i,j,k,Z,l,m;
      unsigned int m_upp;

      //★★★★★★★初期化★★★★★★★★★★★★★★★★★★★★★//
      //////////D/A A/Dボードの初期化////////////////////////
      regst(base_adr);         //ﾏﾆｭｱﾙ  10P 参照

      /////////トラッキングビジョン初期化////////////////////
      TrvModuleAllocate(1,0);
      TrvPciModuleSetup(0,0,"0515");
      TrvInit();
      TrvDrawBox( 320-32, 240-32,320+32, 240+32, 8);//ﾀｰｹﾞｯﾄ表示

      sndPlaySound( "Magic.wav",1);

      printf("Hello Robot Club!!!\n\n");




      ////////////UPP初期化///////////////
      _outp(base_adr+2,0x11);                 /////1
      _outp(base_adr,0xff);                   /////1
      _outp(base_adr+2,0x01);                 /////2
      _outp(base_adr,0x00);                   /////2
```

```
	do{

		TrvDrawBox( 320-32, 240-32,320+32, 240+32, 9);//ﾀｰｹﾞｯﾄ確定
		_outpw(base_adr+2,0x03);

		m_upp= _inpw(base_adr);
		////////////////////////////////////
		daoutpw(base_adr,1,0.0);		//出力 CH1
		daoutpw(base_adr,2,0.0);		//出力 CH2
		daoutpw(base_adr,3,0.0);		//出力 CH3
		daoutpw(base_adr,4,0.0);		//出力 CH4

		daoutpw(base_adr,8,0.0);		//L Motor
		daoutpw(base_adr,9,0.0);		//-L Motor
		daoutpw(base_adr,10,0.0);	//R Motor
		daoutpw(base_adr,13,0.0);	//-R Motor



		if(m_upp==1)manual();	//ch1
		if(m_upp==2)aaa();		//ch2
		}while(m_upp!=128)	//ch8



//☆☆☆☆☆☆☆☆☆終了処理☆☆☆☆☆☆☆☆☆☆☆☆//
	printf("Power Down\n");
	PlaySound( "finish.wav", NULL, SND_FILENAME);
	daoutpw(base_adr,5,0.0);
	daoutpw(base_adr,6,0.0);
	daoutpw(base_adr,7,0.0);
	/*★★★★ビジョンの終了★★★★*/
	TrvVideoStop(0);
	TrvPciExit();
	TrvDrawBox( 320-32, 240-32,320+32, 240+32, 0);
	return 0;
}
```

```
//☆☆☆☆☆☆CH1 UFO手動制御モード☆☆☆☆☆☆☆☆//
void manual(){

	int port,i,Z,I;
	double	volt,a,b,c,d,e,f,g,h,j;
	double	a1=0.0,b1=0.0,c1=0.0,d1=0.0,e1=0.0,f1=0.0,g1=0.0;
	unsigned int p_upp;

	printf("-----------------------------------------------------\n");
	printf("手動モード\n");
	printf("-----------------------------------------------------\n");

		TrvDrawBox( 320-32, 240-32,320+32, 240+32, 9);

			daoutpw(base_adr,1,0);		//出力 CH1
			daoutpw(base_adr,2,0);		//出力 CH2
			daoutpw(base_adr,3,0);		//出力 CH3
			daoutpw(base_adr,4,0);		//出力 CH4

			sndPlaySound( "joy.wav",1);
			printf("Please ENJOY\n");

			////////////UPP初期化//////////
			_outp(base_adr+2,0x11);		/////1
			_outp(base_adr,0xff);		/////1

			_outp(base_adr+2,0x01);		/////2
			_outp(base_adr,0x00);		/////2

	for(;;){
		_outpw(base_adr+2,0x03);
		p_upp=_inpw(base_adr)

		if(p_upp >= 1 && p_upp <= 128){

			if(p_upp == 1){		//////↑
				//printf("チャンネル反応↑\n");
				daoutpw(base_adr,1,-7.0);
				daoutpw(base_adr,2,7.0);
				daoutpw(base_adr,3,-7.0);
				daoutpw(base_adr,4,7.0);
			}

			if(p_upp == 2){		//////↓
				//printf("チャンネル反応↓\n");
				daoutpw(base_adr,1,7.0);
				daoutpw(base_adr,2,-7.0);
				daoutpw(base_adr,3,7.0);
				daoutpw(base_adr,4,-7.0);
			}
			if(p_upp == 4){		//////←
				//printf("チャンネル反応←\n");
				daoutpw(base_adr,1,7.0);
				daoutpw(base_adr,2,7.0);
				daoutpw(base_adr,3,-7.0);
				daoutpw(base_adr,4,-7.0);
			}
```

```
                if(p_upp == 8){       //////→
                      //printf("チャンネル反応→\n");
                      daoutpw(base_adr,1,-7.0);
                      daoutpw(base_adr,2,-7.0);
                      daoutpw(base_adr,3,7.0);
                      daoutpw(base_adr,4,7.0);
                }
                if(p_upp == 16){      //////→ 左回り
                      //printf("チャンネル反応  左回り\n");
                      daoutpw(base_adr,1,7.0);
                      daoutpw(base_adr,2,7.0);
                      daoutpw(base_adr,3,7.0);
                      daoutpw(base_adr,4,7.0);
                }
                if(p_upp == 32){      //////← 右回り
                      //printf("チャンネル反応  右回り\n");
                      daoutpw(base_adr,1,-7.0);
                      daoutpw(base_adr,2,-7.0);
                      daoutpw(base_adr,3,-7.0);
                      daoutpw(base_adr,4,-7.0);
                }
                if(p_upp == 128){       //////CH8脱出
                      daoutpw(base_adr,1,0.0);            //出力 CH1
                      daoutpw(base_adr,2,0.0);            //出力 CH2
                      daoutpw(base_adr,3,0.0);            //出力 CH3
                      daoutpw(base_adr,4,0.0);            //出力 CH4

                      for(Z=0;Z<=10000;Z++){printf("R");}
                      cls();
                      PlaySound( "you2.wav", NULL, SND_FILENAME);
                      printf("手動モード終了!!!\n\n");
                      break;

                }
          }

          else{
                      daoutpw(base_adr,1,0.0);            //出力 CH1
                      daoutpw(base_adr,2,0.0);            //出力 CH2
                      daoutpw(base_adr,3,0.0);            //出力 CH3
                      daoutpw(base_adr,4,0.0);            //出力 CH4

          }

     }

          for(Z=0;Z<=10000;Z++){printf("R");}
}
```

```
//☆☆☆☆☆☆☆☆☆CH2 自動制御モード☆☆UFO駆動系開発☆☆☆☆☆☆☆☆//
void aaa(){

	/*初期化変数*/
	int	bsize,mag,roff,soff,h0,v0,h,v,rh,rv,sh,sv,I,I2=0,I3=0,
	int	dh,dv,mdist,dhz=0,dvz=0,mdistZ=0,devn = 0,H=0,V=0,X=0,Y=0,z,Z=0;
	double		volt,a,b,c,d,e,f,g,i,j,k,S1,S2,S3,S4,S5,S6;
	int s3,s4;
	unsigned int up;
	int Q;
	/*パラメータ*/
	f=300;				//追跡時間をフレーム数で指定
	bsize=32;			//参照ブロックサイズ
	mag=2;				//拡大率（間引き率）
	h0=320;				//参照ブロックの中心座標（H）
	v0=240;				//参照ブロックの中心座標（V）
	h=h0;				//探索ブロック初期中心座標（H）
	v=v0;				//探索ブロック初期中心座標（V）


	PlaySound( "002.wav", NULL, SND_FILENAME);

	for(Q=0;Q<=800;Q++)printf("R");
	cls();
	printf("-----------------------------------------------------\n");
	printf(" UFO AUTO\n");
	printf("-----------------------------------------------------\n");

	daoutpw(base_adr,0,3.0);	//出力 CH0

	////////////////モーターの初期化////////////////////////////
	daoutpw(base_adr,1,0);		//出力 CH1 足に出力　ＤＡ
	daoutpw(base_adr,2,0);		//出力 CH2
	daoutpw(base_adr,3,0);		//出力 CH3
	daoutpw(base_adr,4,0);		//出力 CH4

	////////////////////リモコン入力待ち//////////////////////////
	do{

		_outpw(base_adr+2,0x03);
		up = _inpw(base_adr)-32768;
		printf("Insett\n");
		cls();
		if(up==2)daoutpw(base_adr,15,5.0);  //15リフト上げています
		if(up==128)daoutpw(base_adr,15,0.0);	//リフト止まる

		//printf("%d\n",up);

	}while(up!=1);
	/////////////////////////////////////////////

	printf("ＵＦＯ始動！\n");
	sndPlaySound( "ufo.wav",1);		//音再生
```

```
//////////モーターの初期化////////////////////////////
daoutpw(base_adr,1,0);          //出力 CH1
daoutpw(base_adr,2,0);          //出力 CH2
daoutpw(base_adr,3,0);          //出力 CH3
daoutpw(base_adr,4,0);          //出力 CH4

daoutpw(base_adr,8,0.0);              //L Motor
daoutpw(base_adr,9,0.0);              //-L Motor
daoutpw(base_adr,10,0.0);       //R Motor
daoutpw(base_adr,13,0.0);       //-R Motor
daoutpw(base_adr,14,0.0);       //リフト
daoutpw(base_adr,15,0.0);


//参照・探索ブロックの中心から左上座標を計算するオフセット
roff = bsize*mag/2;       //32
soff = roff+8*mag;        //48
TrvSetSyncMode(EXTERNAL);
TrvVideoThrough();
TrvSetSample(devn,mag-1,mag-1,mag-1,mag-1);//間引き率指定
TrvFetchImage(devn,0);                          //参照画面取り込み
TrvFrameAddressMode(NTSC_EVEN);
TrvVideoInput(devn,DOUBLE_BUF,0);    //画像入力開始


for(;;){

        daoutpw(base_adr,0,2.0);        //出力 CH0
        TrvWaitNextFrame(0);
        //カラー相関演算
        rh = h0 - roff;    //320-32 = 288
        rv = v0 - roff;    //240-32 = 208
        sh = h - soff;     //320-48 = 272
        sv = v - soff;     //240-48 = 192


          TrvColorCorAny(
                devn,
                   bsize,//参照ブロック水平サイズ（８の倍数を指定）
                   bsize,//参照ブロック垂直サイズ（８の倍数を指定）
                   VRAM_REF,//参照ブロックを取り出すVRAM番号
              VRAM_SER,//探索ブロックを取り出しVRAM番号
                   rh,//298参照ブロック水平アドレス（左上角）
                   rv,//208参照ブロック垂直アドレス（左上角）
                   sh,//探索ブロック水平アドレス（左上角）
                   sv,//探索ブロック垂直アドレス（左上角）
                   0,//累積指定０：累積しない１：累積する
                   0,//参照ブロックマスクデータＩＤ番号
                   0,//探索ブロックマスクデータＩＤ番号
                   &dh,//（出力）動き水平ベクトル
                   &dv,//（出力）動き垂直ベクトル
                   &mdist//（出力）最小相関値
          );

         TrvDrawBox( h-roff, v-roff,h+roff, v+roff, 0);
        h += (dh * mag);  //新しいカーソルを書く
        v += (dv * mag);
```

```
        ////////超音波センサー入力////////////////
        S1=adinpw(base_adr,10);         //入力 CH10  赤外線入力  1ch
        S2=adinpw(base_adr,11);         //入力 CH11  赤外線入力  2ch

        S3=adinpw(base_adr,12);         //入力 CH12  赤外線入力  3ch 前
        S4=adinpw(base_adr,13);         //入力 CH13  赤外線入力  4ch

        S5=adinpw(base_adr,14);         //入力 CH14  赤外線入力  5ch
        S6=adinpw(base_adr,15);         //入力 CH15  赤外線入力  6ch


    //////////////よける作業/////////////////////////////////////

        if(S3<1.0){////←

              daoutpw(base_adr,1,7.0);
              daoutpw(base_adr,2,-7.0);
              daoutpw(base_adr,3,7.0);
              daoutpw(base_adr,4,-7.0);
              for(z=0;z<900;z++)printf("よけています↓");



              daoutpw(base_adr,1,7.0);
              daoutpw(base_adr,2,7.0);
              daoutpw(base_adr,3,-7.0);
              daoutpw(base_adr,4,-7.0);
              for(z=0;z<1000;z++)printf("よけています←");

        }


        if(S4<1.0){////→

              daoutpw(base_adr,1,7.0);
              daoutpw(base_adr,2,-7.0);
              daoutpw(base_adr,3,7.0);
              daoutpw(base_adr,4,-7.0);
              for(z=0;z<900;z++)printf("よけています↓");

              daoutpw(base_adr,1,-7.0);
              daoutpw(base_adr,2,-7.0);
              daoutpw(base_adr,3,7.0);
              daoutpw(base_adr,4,7.0);
              for(z=0;z<1000;z++)printf("よけています→");
        }


        ////////////運動部/////////////////////////////////////

        if(h<452 && h>188){ //前進します。

              daoutpw(base_adr,1,-7.0);
              daoutpw(base_adr,2,7.0);
              daoutpw(base_adr,3,-7.0);
              daoutpw(base_adr,4,7.0);
              }
```

```
else if(h>452 && h<640){
            daoutpw(base_adr,1,-8.0);

      daoutpw(base_adr,2,-8.0);
      daoutpw(base_adr,3,-8.0);
      daoutpw(base_adr,4,-8.0);
}

else if(h>0 && h<188){
      daoutpw(base_adr,1,8.0);
      daoutpw(base_adr,2,8.0);
      daoutpw(base_adr,3,8.0);
      daoutpw(base_adr,4,8.0);

}

///////////////ﾄﾗｯｷﾝｸﾞｴﾗｰ処理//////////////////////////
if(h<soff){
      h=soff;
      s3=1;s4=1;
      daoutpw(base_adr,1,-10.0);
      daoutpw(base_adr,2,10.0);
      daoutpw(base_adr,3,-10.0);
      daoutpw(base_adr,4,10.0);
      for(z=0;z<3000;z++)printf("直進しています↑");
}
if(v<soff){
      v=soff;
      s3=1;s4=1;
      daoutpw(base_adr,1,-10.0);
      daoutpw(base_adr,2,10.0);
      daoutpw(base_adr,3,-10.0);
      daoutpw(base_adr,4,10.0);
      for(z=0;z<3000;z++)printf("直進しています↑");
}


g=adinpw(base_adr,0)-160;           //入力 CH0

if(g >= 2.0){           //////G 衝突時停止用
      printf("対物感知！！！！\n");
      daoutpw(base_adr,1,0.0);
      daoutpw(base_adr,2,0.0);
      daoutpw(base_adr,3,0.0);
      daoutpw(base_adr,4,0.0);
      sndPlaySound( "hatti.wav",1);       //音再生
      break;
}

_outpw(base_adr+2,0x03);
up = _inpw(base_adr);
```

```
            if(up == 128){          ////JOYSTICK 非常停止用
                        printf("活動停止！！！\n");
                        daoutpw(base_adr,1,0.0);
                        daoutpw(base_adr,2,0.0);
                        daoutpw(base_adr,3,0.0);
                        daoutpw(base_adr,4,0.0);
                        break;
            }

              TrvDrawBox( h-roff, v-roff,h+roff, v+roff, OVL_RED);
      }

      printf("自動モード終了します\n");
      TrvOVLClear();
      TrvVideoStop(devn);
      daoutpw(base_adr,6,0.0);
      daoutpw(base_adr,7,0.0);
      //cls();
      sndPlaySound( "you3.wav",1);
}
```

```
////////D/A///A/D//ドライバ/////////////////
int regst( unsigned baseadr)
{
      reg_data=0xcf;                                  //ﾏﾆｭｱﾙ 12P 参照
      _outpw(baseadr+4, reg_data);  //ｼﾞｬﾝﾊﾟｰとの認証　使用機能の召還
      erroff=(baseadr+4);
      return(erroff);
}

int regstoutpw(unsigned baseadr ,unsigned int regstno)
{
      reg_data=regstno;
      _outpw( baseadr, reg_data);
      return 0;
}

/////////  D/A出力  ////////////////////////////
double daoutpw(unsigned baseadr ,int portno, double value )
{
      int da_data;

      if( (portno >= 0)&&(portno <= 15) ){
        da_data = (value + 10.0) / 20.0 * 4096.0; // 2048 =  0v
        if (da_data <    0) da_data =    0;
        if (da_data > 4095) da_data = 4095;
        _outpw(baseadr+2, 0xf0+portno);
        _outpw(baseadr, (unsigned)da_data);

      }
      else{
            printf("チャンネルEra\n");
      }
}

////A/D ｲﾝﾌﾟｯﾄ ///////////////
double adinpw( unsigned int baseadr ,unsigned int portno )
{
      double            volt;
      int         ad_read_data;
      unsigned int      ad_write_data;
      ad_write_data = portno*8+5;

      if( (portno >= 0)&&(portno <= 15) ){
          _outpw(baseadr+2, 0x70+portno);
          _outpw(baseadr  , portno*8+5);
          for(i=0;i<10; i++) ad_write_data=i;
          ad_read_data = (signed)_inpw( baseadr );

      if ( ad_read_data & 0x0800 ) ad_read_data |= 0xf000;
        volt = (double)ad_read_data / 2048 * 10;
        return(volt);
      }
      else{
        erron=(1);
        return(erron);
      }
}
```