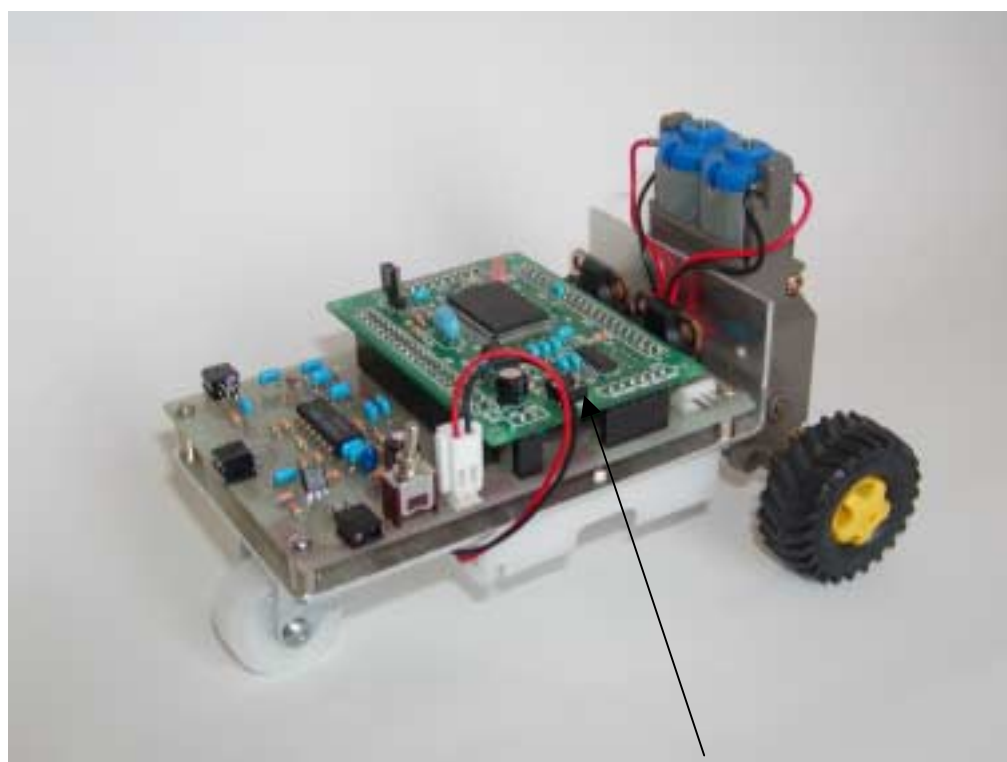


センサーロボット

OAKS16 - SENSOR LABO

ユーザーズマニュアル Ver1.01

TM統合環境 + Nc30wa Ver4.00 Release 2 Entry 版対応



オプション

OAKS16 CPUボード

オークス電子株式会社

目 次

1. はじめに	3
2. OAKS16 - SENSOR LABOの特徴	3
3. OAKS16 - SENSOR LABOの仕様	4
4. DCモータドライブ部	5
(1) DCモータドライブIC「TA8440H」の仕様	5
(2) DCモータドライバ「TA8440H」の使い方	6
(3) CPUボード(OAKS16)との接続	7
5. 光距離センサ部	8
(1) 光距離センサ部の接続	9
(2) 障害物からの距離とDC出力電圧の関係	9
(3) 光距離センサ回路の基本構成	10
6. CPUボード(オプション)	11
(1) パラレル入出力ポート	12
(2) シリアルポート	13
7. OAKS16 - SENSOR LABOのしくみ	14
(1) DCモータの駆動方法	14
(2) PWM制御とは	17
(3) PWMモードの動作	17
(4) DCモータの特性	21
(5) 光距離データとA/D変換器の使い方	22
8. 添付ファイルの構成と使い方	26
(1) DCモータとギヤの使用上の留意点	26
(2) サンプル「Robo_1」の使い方	26
(3) サンプル「Robo_2」の使い方	27
(4) サンプル「Robo_3」の使い方	28
(5) 距離の検出方法	30
9. フラッシュROMへの書き込み方法	31
(1) デバッガを使用する為のフラッシュライターソフトの使い方	32
(2) デバッガの起動方法	34
10. TM統合化開発環境での使い方	36
(1) エントリー版(新バージョン)の仕様について	37
(2) TM統合化開発環境の使い方	38
11. プログラムの作成手順	38
(1) プログラムの実行手順	39
(2) コンパイル方法	40
(3) コンパイル用バッチファイル例	43

(4) I/O ポートの記述例	43
(5) 割り込み処理	44
(6) 割り込みベクタテーブルへの登録	45
12. Flash ROMへの書き込み	47

付 録

付録 1	ドライバボード回路図	49
付録 2	ドライバボード構成図	49
付録 3	OAKS16-16SENSOR LABO外形図	50
付録 4	モータ配線図	50
付録 5	シャシー組立て図	51
付録 6	RS-232C結線図	51
付録 7	部品表	52



安全上の注意

- 1) 乾電池以外の外部電源を接続中に変なにおいがする、発熱するなどの異常状態がみられる場合はすぐには電源を切ってください。感電や火傷に注意しながら速やかにコネクタをはずしてください。また、外部電源は6[V]またはそれ以下で必ずご使用下さい。
- 2) 万一、乾電池値から漏れた液が肌に触れると、火傷の原因になります。破損した電池に触れた場合は、すぐに水で洗い流してください。
- 3) 乾電池は絶対にショートさせないでください。発熱、液漏れ、破裂、発火の原因になります。
- 4) 本製品はメカトロニクス学習用です。幼児、子供の手の届かない所に置いて下さい。



ご使用上の注意

- 1) 使用しているDCモータは、PWM制御によって走行速度の制御を行いますが、乾電池の消耗によっても走行速度が遅くなります。したがって、電池の消耗によって設計通りの動作ができなくなります。十分にご注意下さい。
- 2) 使用されているツインギヤーボックスは、構造上の問題から左右のモータのトルクに差が出ます。設定したPWMデータが左右同じであっても直線上を走行せず、何れかの方向にカーブしますのでご注意ください。

1. はじめに

ロボットブームの今日、アイボ（ホンダ製）に代表される人間型ロボットや工場等での搬送用ロボットや介護用ロボットからホビー用ロボットまで、様々なロボットが次々に紹介されております。このようなロボットブームの21世紀を迎えた今日、ロボットの仕組みを知りたいと言う願望を持つ一般ユーザー人口が急激に増加しております。

センサーロボット「OAKS16 - Sensor Labo」は、前方及び左右の障害物を光距離センサーによって捕らえ、それぞれの障害物を避けて走行する自走ロボットです。

初心者でも経験者でもロボットの製作に取り組む人にとって、その楽しさを体験し、より優秀なロボットの製作が容易に実現する為の参考となります。

2. OAKS16-SENSOR LABO の特徴

最近ではPICマイコンシリーズ、H8マイコンシリーズ、Z80マイコンシリーズ等をCPUポートに採用した学習ロボットが多数紹介されています。しかし、これらのCPUボードは、必ずしもユーザー側から見ると使い易いとは言えないのが現状です。理由は次の点にあります。

パラレルポートやタイマが足りずに増設が必要となる。

CPUの周辺にROMとRAMを必ず接続し、ロムライターを用いてROMに書き込みが必要となる。

デバッガやコンパイラが高価でソフト開発には容易に使用できない。

処理速度が遅く、ロボット制御には向かない。

ROMやRAMの容量が少なく、満足なプログラムが開発できない。

これらの問題を全て解決したロボットがOAKS16 - Sensor Laboです。CPUにはM16C/62シリーズのM30620FCAFP(ルネサステクノロジ社製)と呼ばれる16ビット・ワンチップマイコンを搭載したCPUポート「OAKS16」(オプション)がコネクタを介して接続できる仕様になっています。次のような多くの特徴を持っています。

- オプションのCPUポート「OAKS16」を用いると、プログラム作成に必要なコンパイラ、デバッガ、フラッシュライターソフトが使用できます。
- DCモータ駆動回路にはPWM制御ICを使用し、速度制御を可能にしています。
- 光距離センサは、障害物までの距離に対応したDC電圧が得られます。この電圧をA/Dコンバータに取り込みCPUポートで障害部まで距離を得ます。

3.OAKS16-SENSOR LABO の仕様

OAKS16-SENSOR LABOの仕様は表1で示されるように、CPUには16ビットマイコンを搭載したOAKS16マイコンボード(オプション)がコネクタを介して取り外し可能となっています。

表1 OAKS16-SENSOR LABOの仕様

機 構	内 容
モータ/ギヤー	宮模型 No97 ツインタギヤボック
タイヤ	田宮模型 No101 トラックタイヤセット(2個使用)
モータドライブ	東芝 TA8440H 2個
距離センサ	ロームRPR-220 3個
シャーシ	アルミニウム板一体型、キャスト1個付き
電源	6[V] / 300[mA](電池MU-3×4は付属しません。)
マニュアル	ユーザズマニュアル(PDFファイル形式)
重量サイズ	250g(電池無し) 横：110mm 長さ：145mm 高さ：90mm

また、表2のようなオプションが別価格として用意されています。商品に添付されているCD-RにはオプションのCPUボードで動作するサンプルデモプログラムが入っています。デバッグ用として、汎用のRS-232Cケーブルを使用する場合には、付録 6のRS-232C結線図に従って変換ケーブルを作成して下さい。

表2 オプション仕様

オプション	構 成	価 格
専用RS-232Cケーブル OAKS16-PLC用	9pin-3pinケーブル D-SUB 9 ㄨ/H3P-SHF-AA L=1.8m	¥3,000
CPUボード	OAKS16マイコンボード ルネサステクノロジ社製30620FCAFP使用 フラッシュROM 128KB、RAM 10KB 内臓 ・コンパイラ付き (NC30WA V4.00 Release2 Entry版) ・リモートデバッグKD30付き ・フラッシュライターソフトM16CFIsh付き	¥5,400

4.DCモータドライブ部

モータドライブの構成図を図1に示します。東芝社製のPWM(Pulse Width Modulation)制御によるDCモータ駆動回路ICであるTA8440Hを使用し、OAKS16-SENSOR LABOの前進走行や後進走行を実現します。

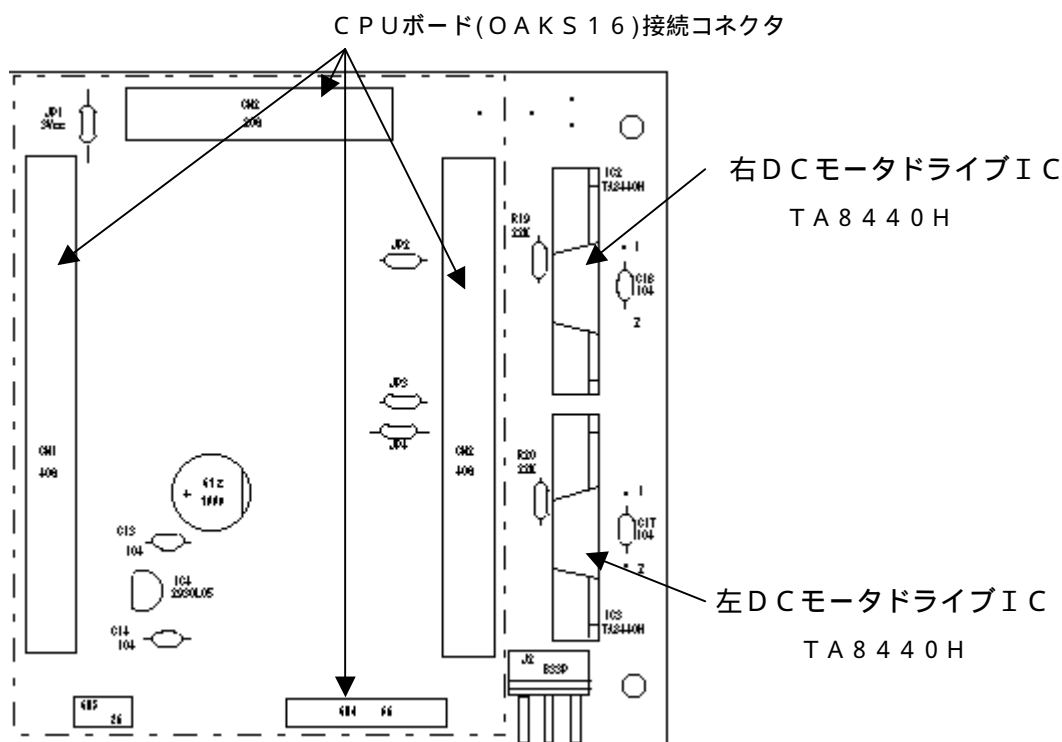


図1 モータドライブ部の構成図

(1)、DCモータドライブIC「TA8440H」の仕様

DCモータドライブICのTA8440H(東芝製)は、ブラシ付きモータの正・逆転切り替え用のフルブリッジドライバで、正転、逆転、ストップ、ブレーキの4モードがコントロールできます。また、モータ駆動部とコントロール部は独立しており、ステッピングモータドライバとしても使用できます。

図2は、TA8440HとDCモータの接続を示したものです。OAKS16-SENSOR LABOでは、1番端子(Enable)を6番端子(制御系電源)と接続し、常にHレベルを与えています。1番端子(Enable)と2番端子(回転方向制御)はIC内部でプルダウン抵抗(100K)が内蔵されています。これらの端子をマイコンポートのI/Oポート端子と接続した場合、プログラムによってI/Oポートをプルアップ(Pull Up)に設定すると、各ポートのプルアップ抵抗は50K(平均値)となりTA8440H側のプルダウン機能が働かずプルアップされ突然モータが回りだすなどの誤動作をする可能性があります。そこで、1番端子(Enable)は常にHレベルを与えています。

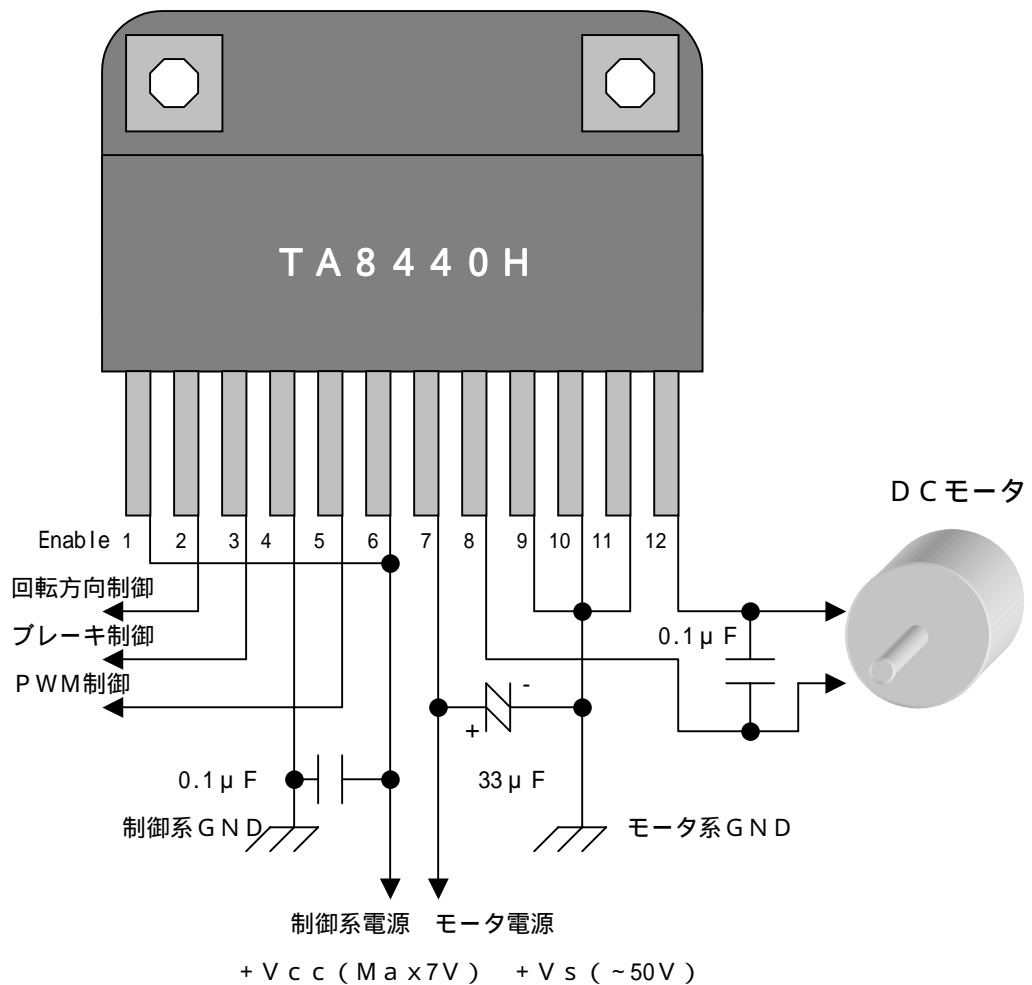


図2 TA8440Hとモータの接続

(2)、DCモータドライバ「TA8440H」の使い方

ENABLE端子(1番端子)が常に‘1’の状態を表3で示されるように、回転方向制御端子(2番端子)に‘0’、ブレーキ制御端子(3番端子)に‘1’を与え、PWM制御端子(5番端子)が‘0’になるとDCモータは正転します。また、回転方向制御端子(2番端子)に‘1’、ブレーキ制御端子(3番端子)に‘1’を与え、PWM制御端子(5番端子)が‘0’になるとDCモータは逆転します。

モータを停止するには、回転方向制御端子(2番端子)の状態に関係なく、ブレーキ制御端子(3番端子)に‘1’を与え、PWM制御端子(5番端子)を‘1’にするとDCモータは停止します。

表3 TA8440Hの真理値表

入 力			出 力		
回転方向制御	ブレーキ制御	PWM制御	OUTA	OUT \bar{A}	モータ動作
0	1	0	0	1	正転
1	1	0	1	0	逆転
0 / 1	1	1			ストップ
0 / 1	0	0 / 1	0	0	ブレーキ
0	1	1	0		CHOP
1	1	1		0	CHOP

(注) : ハイインピーダンス

(3)、CPUポート(OAKS16)との接続

図3はCPUポート「OAKS16」とDCモータドライバ「TA8440H」の接続を示したものです。右モータの回転方向制御は、CPUポートのI/OポートP10とP11を図3に示されるように接続します。また、左モータの回転方向制御は、CPUポートのI/OポートP30とP31を図3に示されるように接続します。

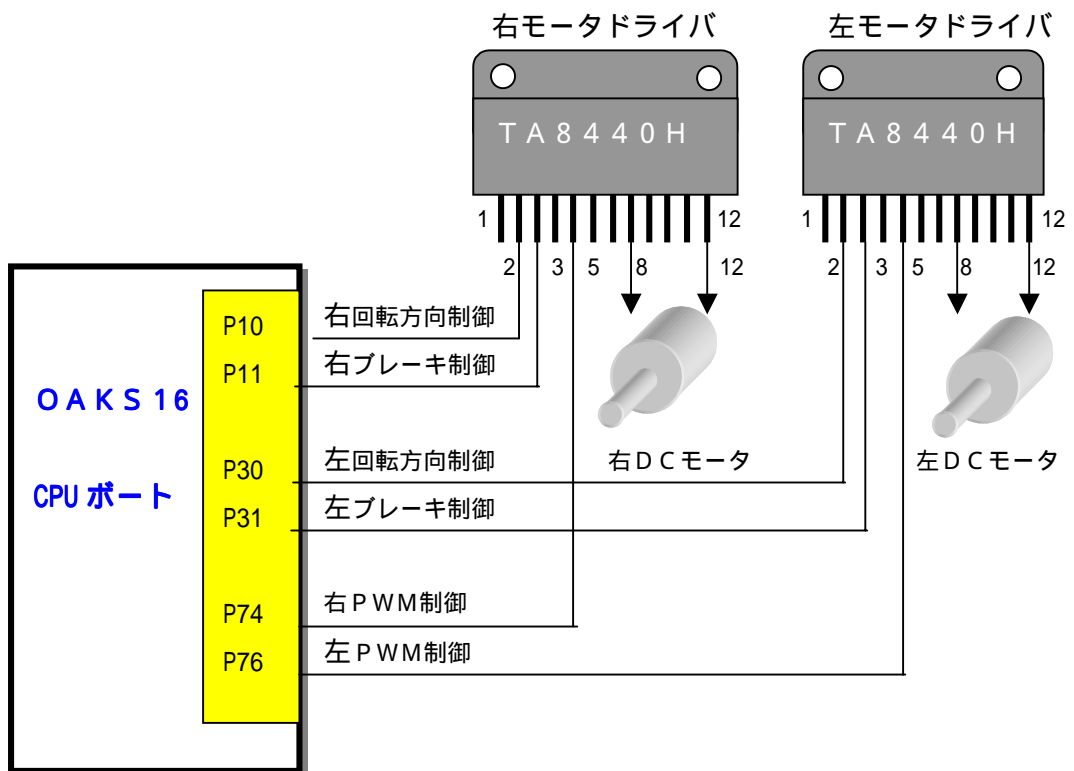


図3 CPUポートとTA8440Hの接続

右モータ用のPWM制御は、I/OポートのP74端子がPWM出力端子(TA2out)と兼用端子で、プログラムによって使用を切替えます。また、左モータ用のPWM制御は、I/OポートのP76端子がPWM出力端子(TA3out)と兼用端子で、プログラムによって使用を切替えます。

5. 光距離センサ部

光距離センサ部の構成図を図4に示します。光距離センサ部は図1で示されるDCモータドライブ部と一体化した基板になっており、取り外しが可能です。回路図は付録3でご覧下さい。光距離センサーにはローム社のフォトレフレクタRPR-220を3個使用します。交流方式により、障害部からの反射出力をOPアンプによる増幅し、この出力をダイオードで検波(整流)します。得られた直流電圧をマイコンのA/Dコンバータから取り込み障害物からの距離を得ています。

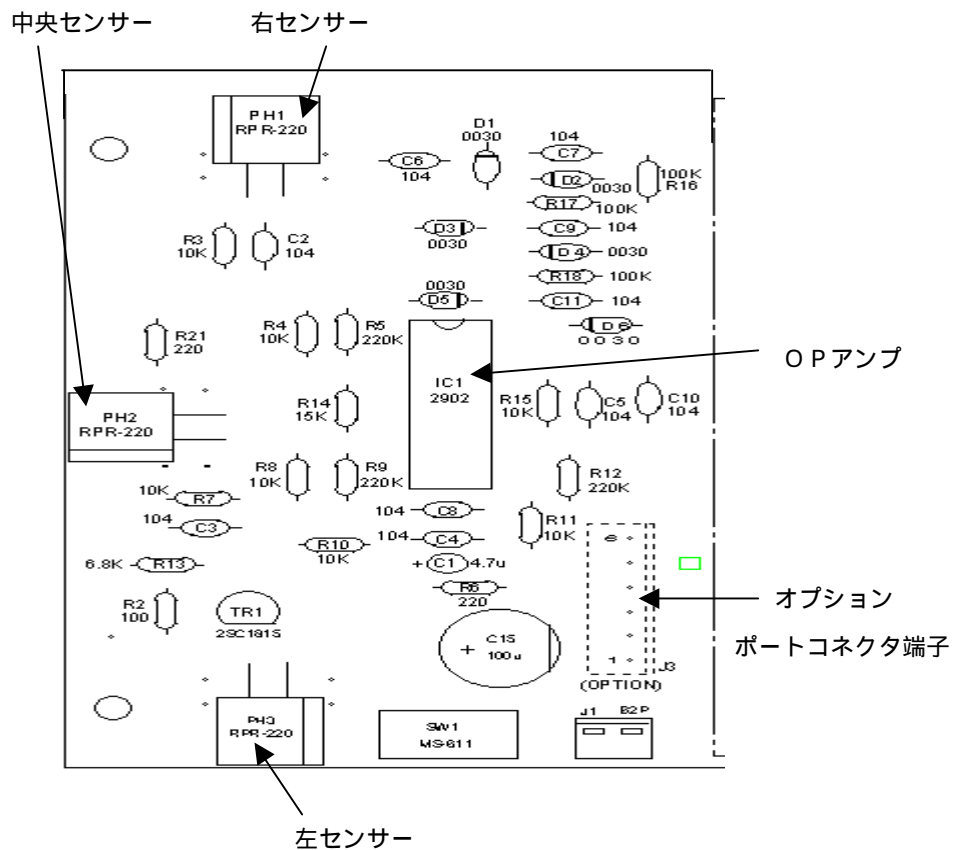


図4 光距離センサ部の構成図

(1) 光距離センサー部の接続

光距離センサー部に取り付けられている3個のフォトレフレクタからの距離情報は、図5に示されるように、OPアンプと検波回路を介してCPUボードのA/D変換器端子AN0、AN1、AN2にそれぞれ入力します。

また、フォトレフレクタを距離センサとして使用する為に、I/OポートP04端子をトランジスタのベースに接続します。この端子からプログラムによってソフト的に800Hzのパルスを作りフォトレフレクタRPR-220に与えています。

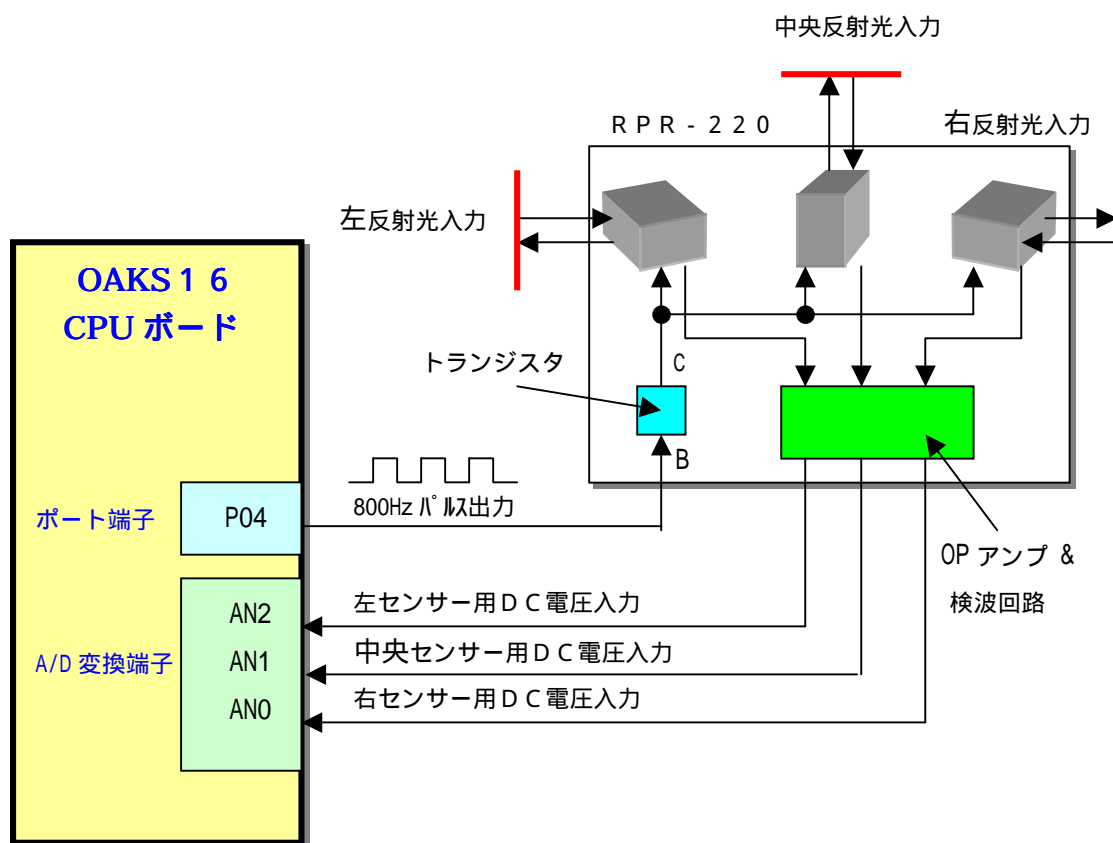


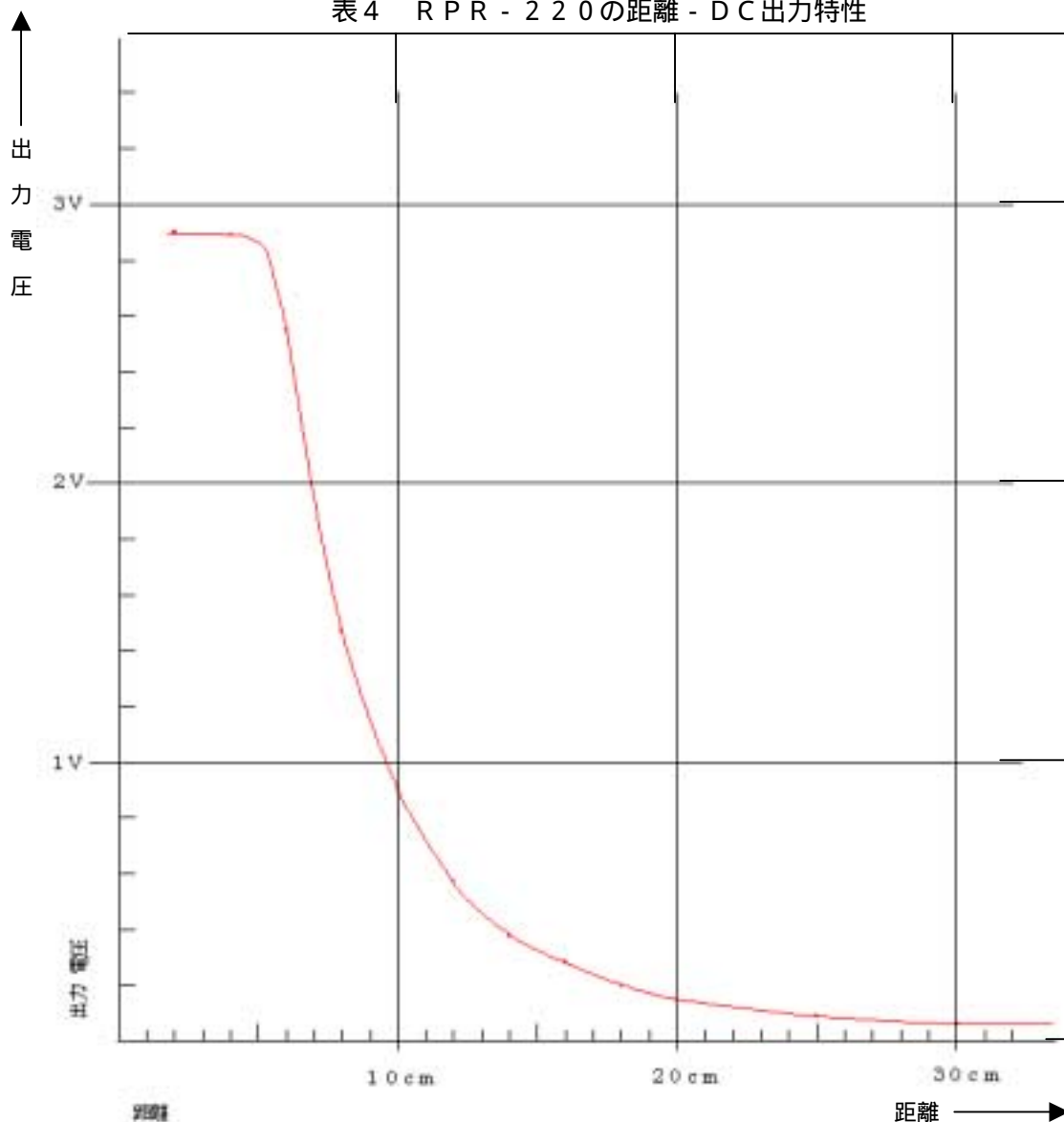
図5 光距離センサー部の接続

(2)、障害物からの距離とDC出力電圧の関係

表4は、フォトレフレクタRPR-220を距離センサーとして使用した場合の障害物までの距離とその時に得られるDC電圧出力の関係を表した特性図です。

連続して正しい距離を検出するには、距離 - DC出力電圧特性からできる限り直線部分を使用します。表4では、距離が6cmから12cmの間を使用するとその距離に比例したDC出力電圧が得られます。

表4 RPR - 220の距離 - DC出力特性



(3)、光距離センサー回路の基本構成

図6は光距離センサー部の基本構成回路です。フォトレフレクタRPR - 220からの発射光は、800Hzのパルス波として発射されます。このパルス波は、I/OポートP04端子からトランジスタ2SC185のベースに与えられます。コレクタ側には発光ダイオードのカソードに接続すると、トランジスタのON/OFFに対応して光が発射されます。800Hzのパルス列は、プログラムによってソフト的に作り、I/OポートP04端子から出力します。

障害物からの反射光は、フォトトランジスタから発光ダイオードのパルス波に同期して受信されます。このパルス波をOPアンプNJM2902で増幅し、ダイオードD1とD

2及びコンデンサ0.1μと抵抗100Kで構成される検波回路で検波し、直流出力電圧を得ています。

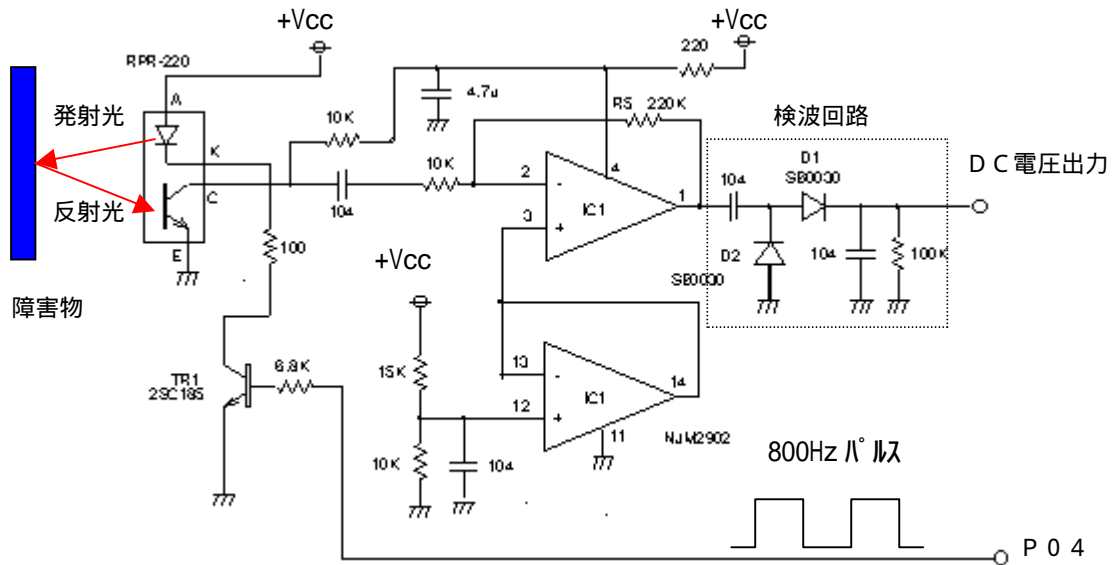


図6 光距離センサーの基本回路構成

6. CPUボード(オプション)

図7はM16C/62シリーズのM30620FCAFPを搭載したOAKS16マイコンボードです。このボードには128KBのフラッシュROMと10KBのRAMがCPUに内蔵されています。

RAM領域は、400h~02BFFhまでの10KB空間に割り当てられます。ただし、後部の1.8KB空間はデバッガのMonitor Programとして使用されるのでユーザは使用できません。また、フラッシュROM領域は、E0000h~FFFFFFhまでの128KB空間に割り当てられます。ただし、ユーザ領域はE0000h~FBE00h(111.5KB)空間に限定されます。FBE00h以降はデバッガが使用するモニタ領域、割り込みの為にベクタエリアとして使用されています。

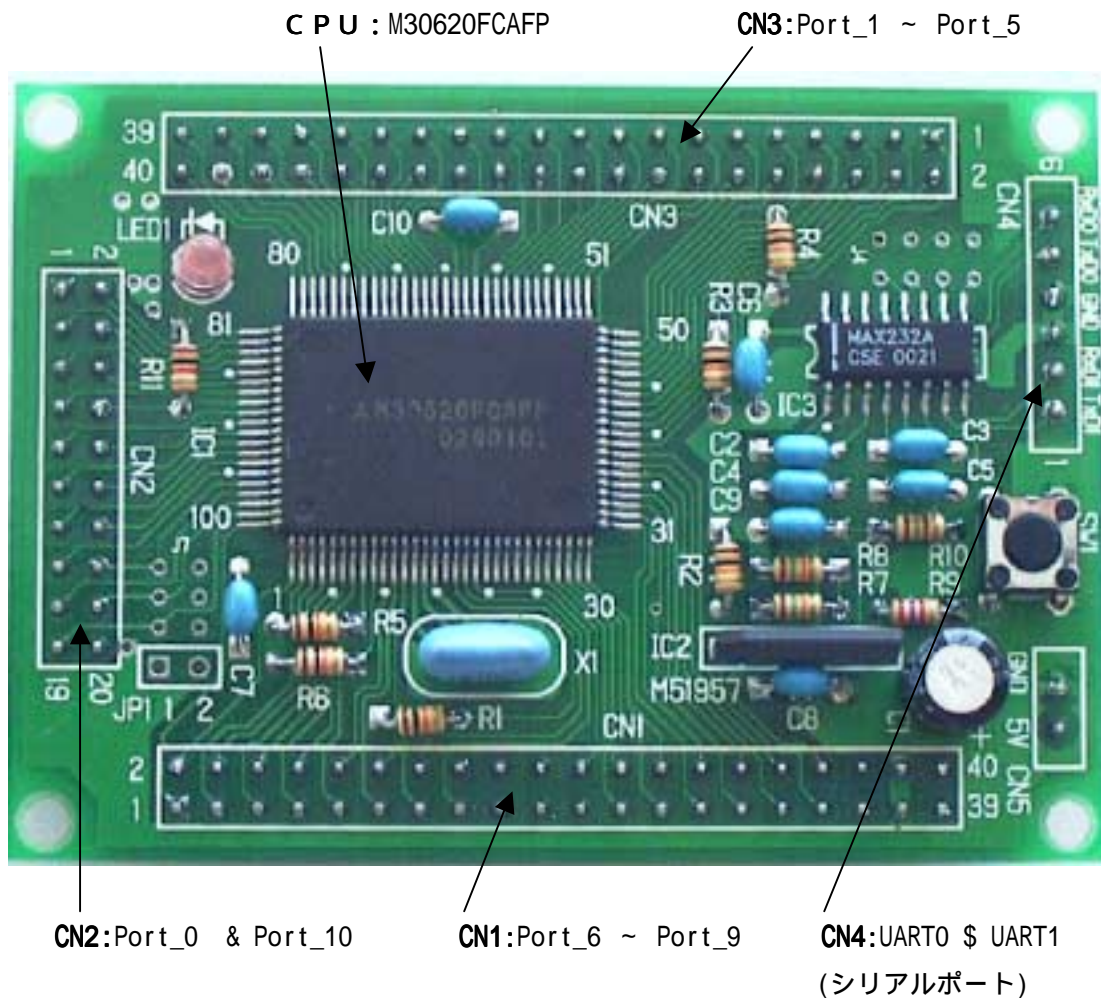


図7 OAKS16 CPUボード

(1)、パラレル入出力ポート

パラレル入出力ポートは、最大で11ポート(88bit)使用できます。実際は、ポート端子がシリアルポート、A/D変換器の入力端子、D/A変換器の出力端子などが併用されている端子は、その使い方で使用できない場合があります。

表5はOAKS16 - SENSOR LABOが使用するパラレル入出力ポートの割り当てを一覧表で表したものです。

ポート1(Port_1)のビットP10とビットP11は、右側モータの駆動用として使用します。また、ポート3(Port_3)のビットP30とビットP31は、左側モータの駆動用として使用します。

ポート7(Port_7)のビットP74はTA2out端子として使用し、右モータ用のPWMを出力します。ビットP76はTA4out端子として使用し、左モータ用のPWMを出力します。

また、ポート0 (Port_0)のビットP04は、800Hzのパルス波の出力用として使用します。

表5 入出力ポートの割り当て

入出力ポート名		割り当て機能
Port_1	P10	右モータ回転方向制御
	P11	右モータブレーキ制御
Port_3	P10	左モータ回転方向制御
	P11	左モータブレーキ制御
Port_0	P04	800Hzパルス波出力
Port_10	AN0	右光距離センサDC電圧入力
	AN1	中央光距離センサDC電圧入力
	AN2	左光距離センサDC電圧入力
Port_7	P74	右モータPWM出力
	P76	左モータPWM出力

(2)、シリアルポート

シリアルポートは、2チャンネル(RS232Cドライバ実装済み)が標準装備されています。チャンネル1 (Tx1, Rx1) は、リモートデバッガKD30とフラッシュライターソフトによるフラッシュROMへの書き込みで使用します。また、チャンネル0 (Tx0, Rx0) は、未使用チャンネルです。オプションのRS232C無線ユニットを使用すると無線による遠隔操が可能です。

チャンネル1 (Tx1, Rx1) は、図8で示されるようにJ2 (ポストサイド型)は日圧(BS3P-SHF-1AA)コネクタが取り付けられています。パソコンとの接続は、オプションのOAKS16-PLC用9pin-3pinケーブル(¥3,000)を使用するか、付録-6で示される接続ケーブルを自作してご使用下さい。

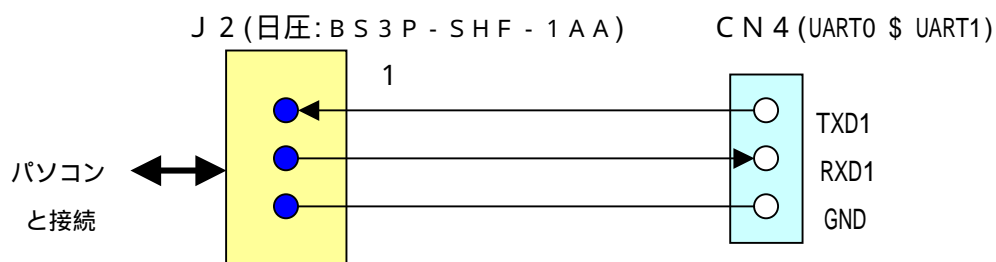


図8 RS-232Cの接続

7.OAKS16 - SENSOR LABOのしくみ

OAKS16 - SENSOR LABOには2個のDCモータを使用しています。DCモータは、DCモータドライバTA8440Hを用いてPWM制御によりモータの速度制御が簡単に行えます。また、光距離センサーによって障害物まで距離を検出し、障害物を避けながら走行できます。

(1)、DCモータの駆動方法

図10は、CPUボードとDCモータドライバ及びDCモータの接続を表したものです。右モータは、I/Oポート1(Port_1)のビットP10で回転方向制御、ビットP11でブレーキ制御を行います。また、I/Oポート7(Port_7)のビットP74をTA2out使用でPWMを出力し、速度制御を行います。

左モータは、I/Oポート3(Port_3)のビットP30で回転方向制御、ビットP31でブレーキ制御を行います。また、I/Oポート7(Port_7)のビットP76をTA3out使用でPWMを出力し、速度制御を行います。

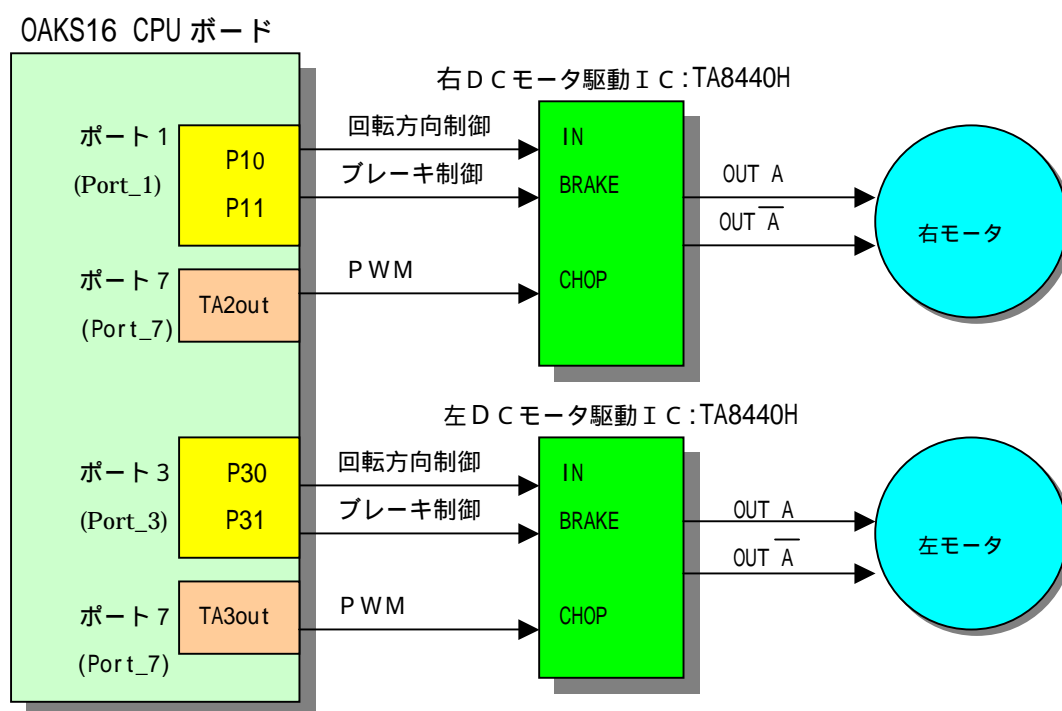


図10 CPUボードとDCモータドライバとDCモータの接続

表6は、OAKS16 - SENSOR LABOを前進走行する為の真理値表です。左右のモータはお互いに向かい合って取り付けられています。従って、表6に示されるように、右モータを正転させ、左モータを逆転させる事により前進走行が可能になります。

表6 前進走行真理値表

	入 力			モータ動作
	ブレーキ	回転方向	16進数	
右モータ	P11	P10	02h	正転
	1	0		
左モータ	P31	P30	03h	逆転
	1	1		

表7は、OAKS16-SENSORLABOを後進走行する為の真理値表です。表7に示されるように、右モータを逆転させ、左モータを正転させる事により後進走行が可能になります。

表7 後進走行真理値表

	入 力			モータ動作
	ブレーキ	回転方向	16進数	
右モータ	P11	P10	03h	逆転
	1	1		
左モータ	P31	P30	02h	正転
	1	0		

CPUポートのポート1 (Port_1) のP10から‘0’、P11から‘1’を出力します。また、ポート3 (Port_3) のP30から‘1’、P31から‘1’を出力すると、ロボットは前進走行します。図11は、前進走行のプログラムを記述したものです。前進走行の動作を記述すると次のようになります。

```
port_1 = 0x02;
port_3 = 0x03;
```

```

前進走行の記述
/*****
 *   前進走行の設定
 *****/
void  for_run(void)
{
    port_1 = 0x02;      /* 右モータ正転動作 */
    port_3 = 0x03;      /* 左モータ逆転動作 */
    ta2 = 0xac3e;       /* 右モータ用PWMデータ */
    ta3 = 0xac3e;       /* 左モータ用PWMデータ */
    tabsr = 0x0c;       /* 左右PWM出力開始 */
}
PWMによる速度設定記述

```

図 1 1 前進走行の記述例

また、CPUポートのポート1 (Port_1) のP10から‘1’、P11から‘1’を出力します。また、ポート3 (Port_3) のP30から‘0’、P31から‘1’を出力すると、ロボットは後進走行します。図12は、後進走行のプログラムを記述したものです。後進走行の動作を記述すると次のようになります。

```

port_1 = 0x03;
port_3 = 0x02;

```

```

後進走行の記述
/*****
 *   後進走行の設定
 *****/
void  bak_run(void)
{
    port_1 = 0x03;      /* 右モータ逆転動作 */
    port_3 = 0x02;      /* 左モータ正転動作 */
    ta2 = 0xa83e;       /* 右モータ用PWMデータ */
    ta3 = 0xa83e;       /* 左モータ用PWMデータ */
    tabsr = 0x0c;       /* 左右PWM出力開始 */
}

```

図 1 2 後進走行の記述例

(2)、PWM制御とは

PWM (Pulse Width Modulation: パルス幅変調) の原理を説明します。PWM信号は図13で示されるような一定周期の信号となっていますが、(a)図と(b)図を比べると、それぞれの信号の‘H’レベルの幅と‘L’レベルの幅が異なります。

例えば、‘H’レベルの区間でモータが回転し、‘L’レベルの区間でモータが停止するとすれば、平均の回転する為のエネルギーは、‘H’レベル区間の長い(デューティの大きい)(b)図のほうが大きくなります。

この信号の繰り返し周期が、モータの回転数より十分早い速さ(高い周波数)であれば、モータの回転エネルギーとしては平均電力で考えると(b)図のほうが高速回転することになります。このような事を利用して、‘H’レベル区間の幅を制御することで、平均電力を制御してモータの速度を制御することができます。

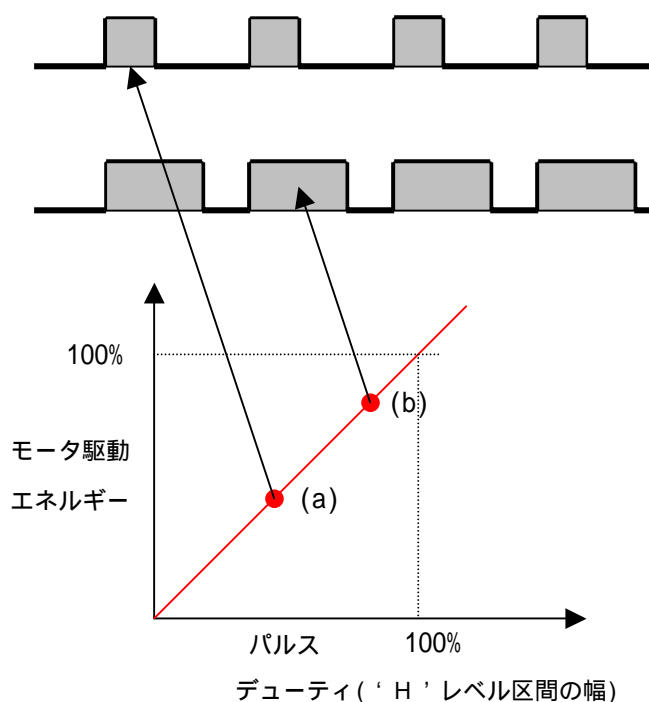
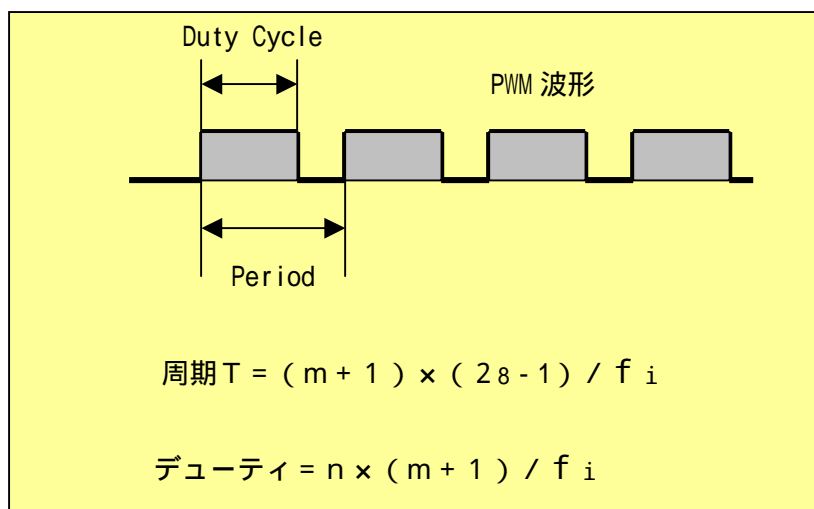


図13 PWM制御の原理

(3)、PWMモードの動作

OAKS16 CPUボードには、タイマA動作の8ビットPWMモード選択ができます。このモードでは、図14で示されるように‘H’レベル区間の長い(デューティサイクル: Duty Cycle)と繰り返し周期(Period)は、次式で表されます。



上式より、mはプリスケラー値(分周値)で00h～FEhの範囲で設定が可能です。また、nはタイマ値で00h～FEhの範囲で設定が可能です。また、 f_i はカウントソースの周波数を表し、CPUボードのクロック周波数16MHzとなります。

OAKS16-SENSOR LABOでは周期Tを1msに設定し、上式からmの値を計算します。

$$m = \left(\frac{f_i \times T}{256} - 1 \right) - 1$$

$$62 (= 3Eh)$$

また、nの値は、プログラムによってモータの速度を見ながらその値を決定します。OAKS16-SENSOR LABOでは、A0hからE0hの範囲でモータの速度を見ながら選びます。例えばn=B0h(=176)を用いた場合のデューティは上式より

$$\text{デューティ} = \frac{176 \times (62 + 1)}{16 \times 106}$$

$$0.7ms$$

となり、ここでLレベル区間の幅を計算すると次のようになります。

$$\begin{aligned} \text{Lレベル区間の幅} &= \text{周期 } T - \text{デューティ} \\ &= 1 - 0.7 \\ &= 0.3ms \end{aligned}$$

DCモータドライバTA8440Hでは、PWMの‘L’レベル区間の幅でモータが回転します。従って、‘L’レベル区間の幅が大きくなるとモータの速度が速くなります。表8は、OAKS16-SENSOR LABOが走行可能なn(タイマ値)に対するデューティと‘L’レベル区間の幅を上記の計算式から計算した一覧表です。光距離センサが補足する障害物までの距離設定が近すぎる場合、速度をあまり早くすると、障害物に衝突する可能性があります。原因は、使用されているモータとギヤの構造上の問題や慣性の影響等により急停止ができません。ユーザプログラムを作成する場合に十分にご注意下さい。

表8 PWMのn値とデューティの一覧表 (T = 1 m s)

n (タイマ値)	デューティ	L' レベル区間の幅	モータ速度
90h(144)	0.567ms	0.433ms	高速回転
a0h(160)	0.63ms	0.37ms	↑ ↓
b0h(176)	0.693ms	0.307ms	
c0h(192)	0.756ms	0.244ms	
d0h(208)	0.819ms	0.181ms	低速回転

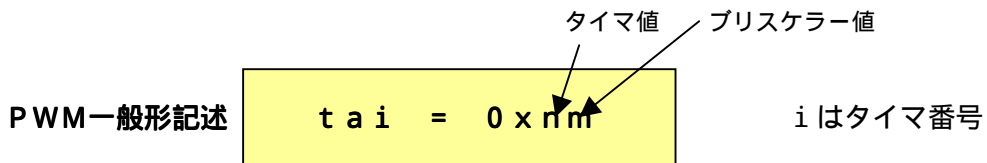


図11と図12は、以上の計算数値を用いてPWMによる走行プログラムです。PWM制御は、ポートP74端子をタイマAのTA2outで使用し、右モータの速度制御用です。また、ポートP76端子をタイマAのTA3outで使用し、左モータ速度制御用です。PWMの初期化プログラムを図14に示します。次の手順でプログラムを記述します。

```

*****
* PWMの初期化 *
*****/
void pwminit(void)
{
    ta2mr = 0x27; /* A1: 8ビットPWMモード fi=f 使用 */
    ta3mr = 0x27; /* A4: 8ビットPWMモード fi=f 使用 */
    ta2 = 62; /* A0 初期値 n=0, m=3Eh T=1ms */
    ta3 = 62; /* A1 初期値 n=0, m=3Eh T=1ms */

    tabsr = 0x00; /* A2 & A3 かつ開始フラグ 停止 */
}

```

図14 PWMの初期化記述例

PWMモードの選択および各機能の選択

タイマAのモードレジスタで右モータはta2mr (= 398h) 左モータはta3mr (= 399h) が対応します。これらのレジスタの各ビット図15で示される意味を持ちます。

が出力されます。また、ビットを0に設定するとPWM出力は停止します。

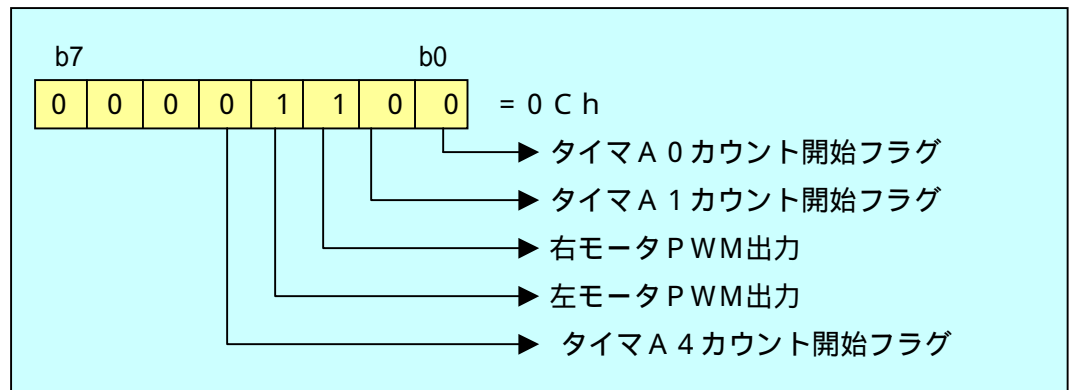


図17 カウント開始フラグの設定

(4)、DCモータの特性

DCモータを使って何らかの制御を行う場合、DCモータの等価回路を考慮する必要があります。図(a)はモータと電源を接続した回路例です。この回路を等価回路で置き換えると図(b)のようになります。図(b)より下記の関係式が成り立ちます。

$$E_b = R_a \times I_a + V_B + E_c$$

上式より E_b : 電源電圧 [V]、 R_a : 電機子抵抗 [Ω]、 I_a : モータ電流 [A]、 V_B : ブラシ - 整流子間の接触電圧 [V]、 E_c : モータの誘導電圧 [V] を表します。

ここで、ブラシ - 整流子間に発生する接触電圧 V_B は $E_b - V_B$ 、 $E_c - V_B$ から、これを無視すると上式は

$$E_b = R_a \times I_a + E_c$$

のようになります。従って、DCモータは、一定の回転数で回転している場合、図(c)のような等価回路で表すことができます。

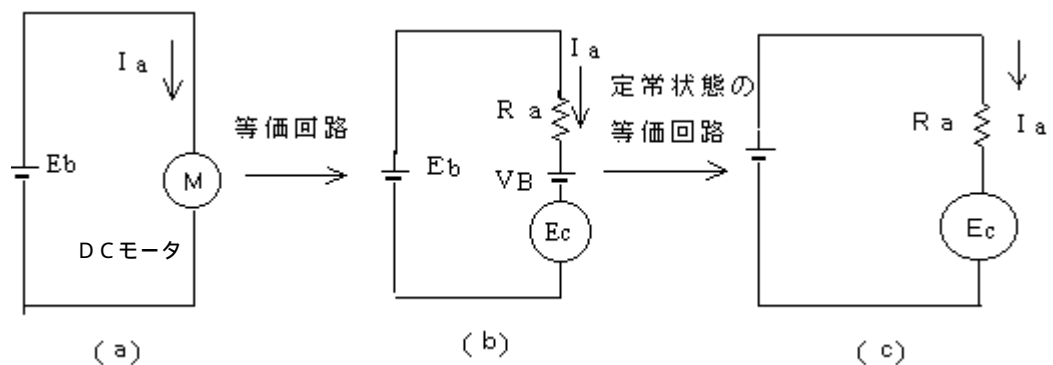


図18 DCモータの等価回路

これらの関係から下図のようなDCモータの特性が得られます。図19より電圧を上げるとこれに比例して回転数が速くなります。モータのトルクTが大きくなり、より大きな力が得られます。また、電圧を下げると回転数が遅くなり、これに比例してトルクTも小さくなります。

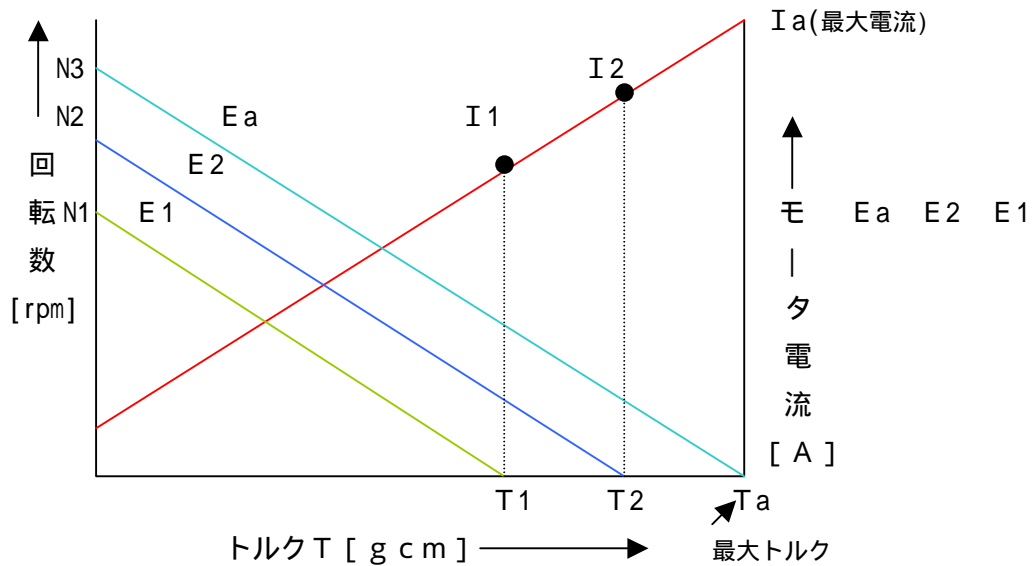


図19 DCモータの特性

OAKS16 - SENSOR LABOでもPWMの制御によって自走速度を自由に変わりますが、PWMの設定値を変えてモータの速度を遅くした場合、ある設定値以下になると、車体を持ち上げたときにタイヤが回転してもモータのトルクと車体の重量により走行できなくなります。ご注意ください。

(5)、光距離データとA/D変換器の使い方

右側光距離センサのDC電圧出力は、A/D変換器のアナログ入力AN0端子から取り込みます。中央右側光距離センサのDC電圧出力は、A/D変換器のアナログ入力AN1端子から取り込みます。また、左側光距離センサのDC電圧出力は、A/D変換器のアナログ入力AN2端子から取り込みます。図20は、光距離センサ回路とA/D変換器のアナログ入力端子の接続を示したものです。

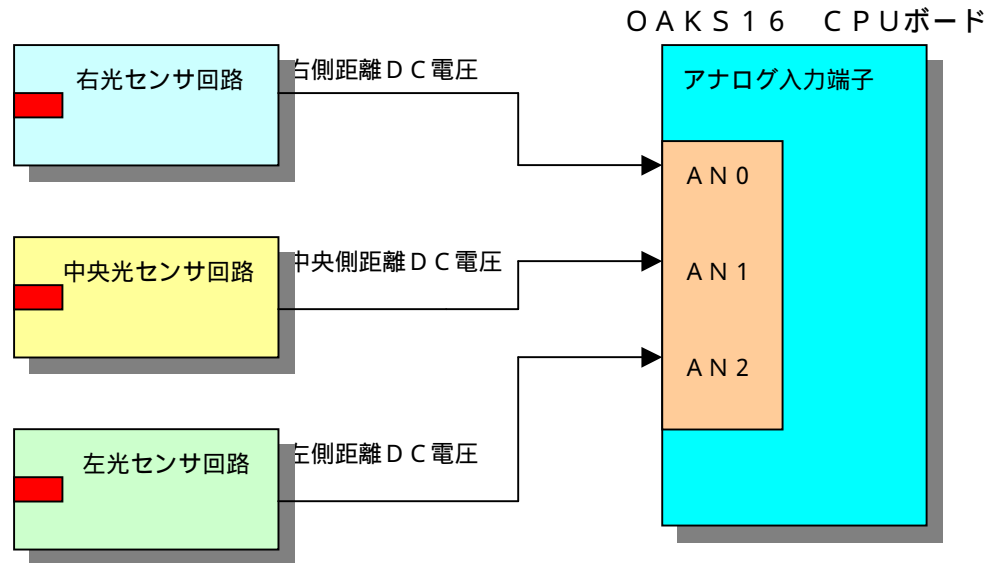


図 2 0 光距離センサ回路と A / D 変換器の接続

A / D 変換器の初期設定について順を追って説明します。図 2 1 は、A / D 変換器の初期設定を記述したプログラム例です。

```

/*****
*   A / D 変換器の初期設定
*****/

void adinit(void)
{
    adcon2 = 0x01;    /* サンプルホールド有り */
    adcon0 = 0x80;    /* fAd/2, 単発モード, AN0 端子使用 */
    adcon1 = 0x28;    /* EXIT OP AMP NO, fAD/2, 10bit mode, Vref */
    adcon2 = 0x01;    /* サンプルホールド有り */
    adcon0 = 0x81;    /* fAd/2, 単発モード, AN1 端子使用 */
    adcon1 = 0x28;    /* EXIT OP AMP NO, fAD/2, 10bit mode, Vref */
    adcon2 = 0x01;    /* サンプルホールド有り */
    adcon0 = 0x82;    /* fAd/2, 単発モード, AN2 端子使用 */
    adcon1 = 0x28;    /* EXIT OP AMP NO, fAD/2, 10bit mode, Vref */

    adic = 0x03;     /* ad 割り込みレベル 3 の設定 */
}

```

図 2 1 A / D 変換器の初期設定プログラム例

サンプル&ホールドの選択

サンプル&ホールドは、A - D制御レジスタ2「ADCON2」(3D4h)のビットb0に1を書き込むと使用が可能となります。図22は、サンプル&ホールドの選択をありに設定した例です。

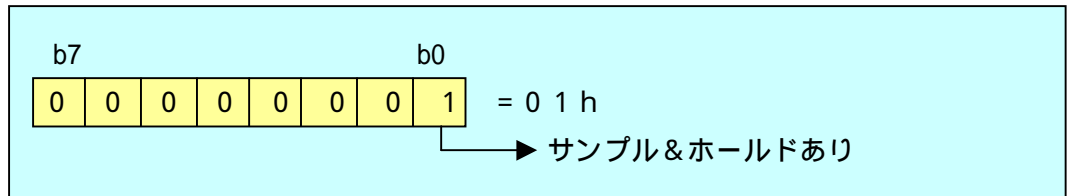


図22 サンプル&ホールドの選択

A - D制御レジスタ0の設定

A - D制御レジスタ0は、図23で示されるように、A - D制御レジスタ0「ADCON0」(3D4h)の各ビット毎にA / D変換器の使用方法を決めます。

A / D変換器は単発モードでします。また、A / D変換器の動作クロックを $f_{AD}/2$ で使用します。

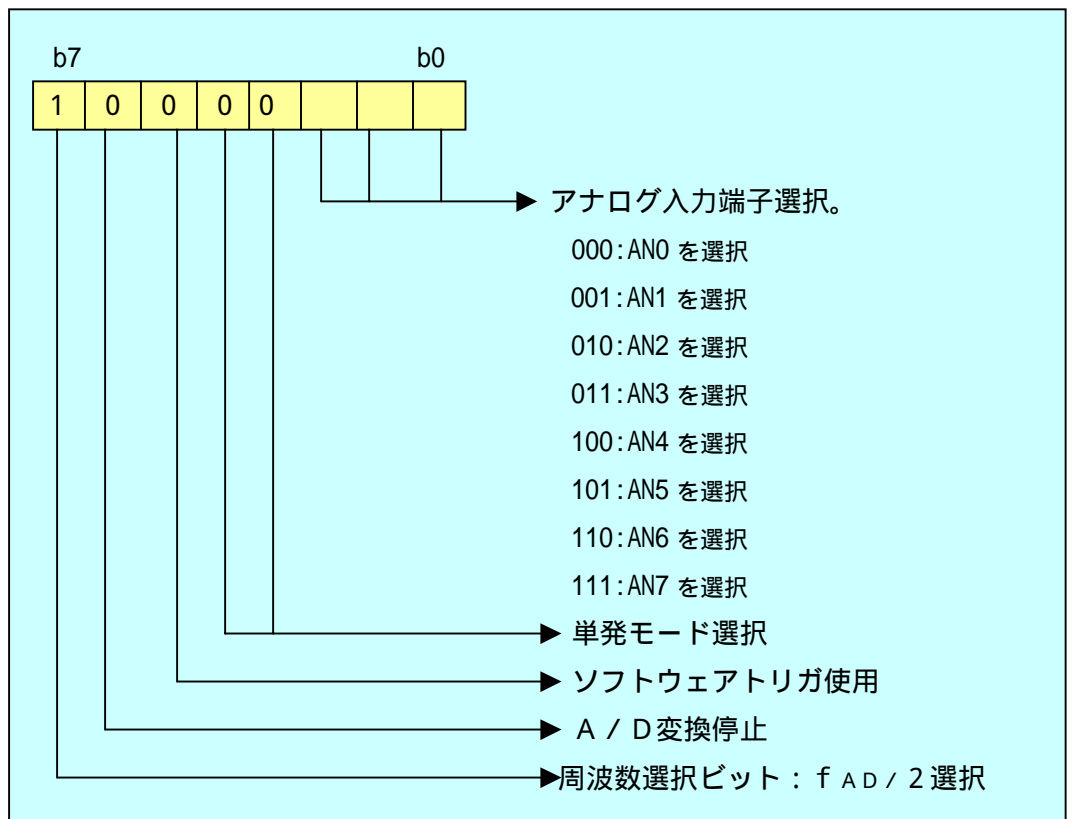


図23 A - D制御レジスタ0の設定

A - D制御レジスタ1の設定

A - D制御レジスタ1は、図24で示されるように、A - D制御レジスタ1「ADCON1」(3D7h)の各ビット毎にA / D変換器の使用方法を決めます。ここでは、10ビットA / D変換器として使用します。

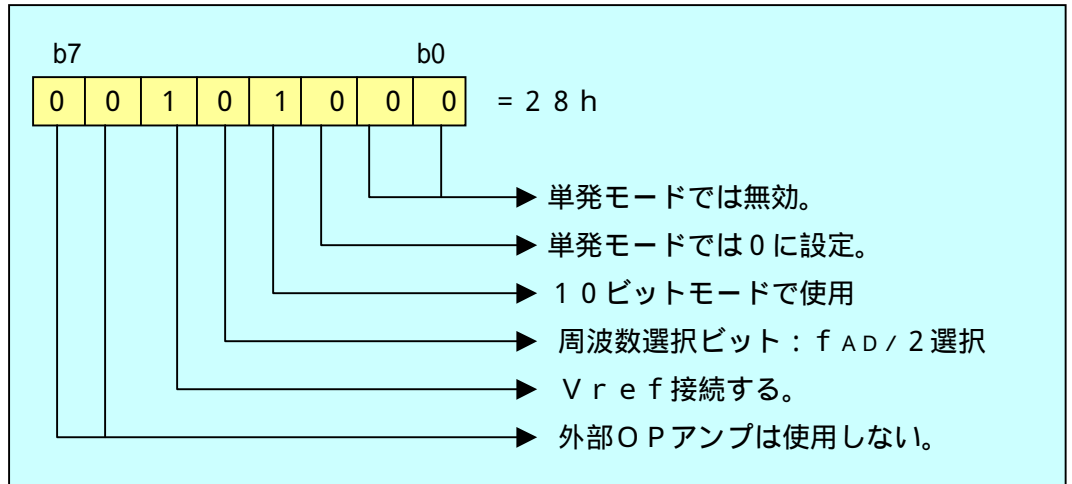


図24 A - D制御レジスタ1の設定

A / D変換開始

A - D制御レジスタ0は、図25で示されるように、A - D制御レジスタ0「ADCON0」(3D4h)のビット6を1に設定するとA / D変換器を開始します。

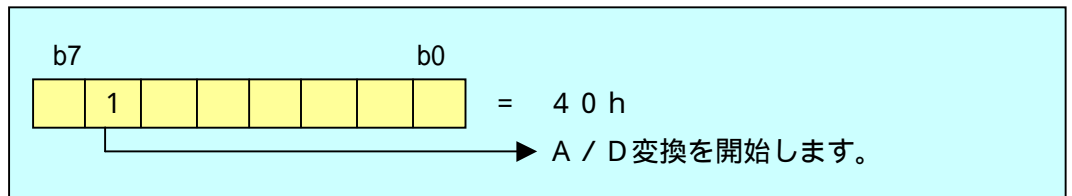


図25 A - D変換の開始

変換結果の読み出し

図26で示されるA / D変換レジスタad0(右光距離データ)、ad1(中央光距離データ)、ad2(左光距離データ)に10ビットのデジタルデータとして読み出されます。

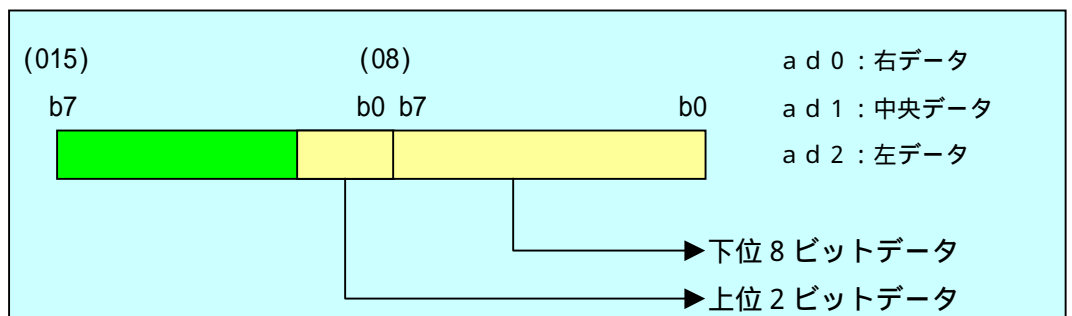


図26 変換結果の読み出し

8. 添付ファイルの構成と使い方

センサーロボット「OAKS16 - SENSOR LABO」には表9で示されるファイルがCD-Rに入っています。目的に応じて自由に取り出してご使用下さい。ただし、添付されている全てのC言語プログラムには著作権が存在しますので商用として使用する事はできません。

表9 OAKS16 - SENSOR LABO添付ファイル一覧表

ディレクトリ名	ファイル名	内 容
Robo_1	Robo_1.c Robo_1.x30 Robo_1.mot	前進走行 - 後進走行の繰り返しソースプログラム デバッグファイル。 フラッシュROM書き込みファイル。
Robo_2	Robo_2.c Robo_2.x30 Robo_2.mot	前進 - Uターン走行の繰り返しソースプログラム。 デバッグファイル。 フラッシュROM書き込みファイル。
Robo_3	Robo_3.c Robo_3.x30 Robo_3.mot	障害物からの距離を検出し障害物を避けるソースプログラム。 デバッグファイル。 フラッシュROM書き込みファイル。

(1) DCモータとギヤ - の使用上の留意点

センサーロボット「OAKS16 - SENSOR LABO」には、田宮模型のツインモータギヤボックスを使用しております。添付されているサンプルデモプログラムやユーザプログラムで走行させる場合、次の点につきましては十分に留意されてからご使用下さい。

使用されているDCモータは、PWM制御によって走行速度の制御を行いますが、乾電池の消耗によっても走行速度が遅くなります。したがって、電池の消耗によって設計通りの動作ができなくなります。十分にご注意下さい。

使用されているツインギヤボックスは、構造上の問題から左右のモータのトルクに差が出ます。設定したPWMデータが左右同じであっても直線上を走行せず、何れかの方向にカーブしますのでご注意ください。

(2) サンプル「Robo_1」の使い方

サンプルデモプログラム「Robo_1」は、電源スイッチをONにすると、一定時間の間、前進走行を行い一旦停止します。次に、同じ一定時間で後進走行で戻ってきます。停止

後に前進走行を繰り返します。図27は、前進走行と後進走行を繰り返すmain()関数のプログラムです。一定時間は、ソフトタイマwait(short k)関数で行います。定数kの値で時間を変えられます。

```
void    main()
{
    pwm_init();          /* PWMの初期化 */
    port_init();        /* ポートの初期化 */
    port_0 = 0x00;      /* LEDの点灯 */
    while(1)
    {
        for_run();      /* 前進走行 */
        wait(600);      /* 走行距離の設定 */
        stp_run();      /* 両モータ停止 */
        wait(200);
        bak_run();      /* 後進走行 */
        wait(600);      /* 走行距離の設定 */
        stp_run();      /* 両モータ停止 */
        wait(200);
    }
}
```

図27 「Robo_1」のmain()関数記述例

(3) サンプル「Robo_2」の使い方

サンプルデモプログラム「Robo_2」は、電源スイッチをONにすると、一定時間の間、前進走行を行い一旦停止します。次に、Uターンを行って一旦停止します。次に、同じ一定時間で前進走行で戻ってきます。

図28はUターン動作を行う関数「u_run()」の記述例です。Uターン動作を行う場合、何れか一方のモータを停止し、他方を一定時間回転させるとターンをしますが、この方法では、同じコース上に戻ってきません。同じコース上に戻すには、左右のモータを一定時間逆転させます。

図28のように、右モータをport_1=0x02記述によって正転させます。また、左モータをport_3=0x02記述で正転させます。左右のモータはお互いに向かい合っているため、お互いに反対方向に回転してUターンを行います。

```

void    u_run(void)
{
    port_1 = 0x02;          /* 右ターン走行    */
    port_3 = 0x02;          /* 左ターン走行    */
    ta2 = 0xb03e;          /* 右モータ用PWMデータ */
    ta3 = 0xb03e;          /* 左モータ用PWMデータ */
    tabsr = 0x0c;
}

```

図 2 8 Uターン関数の記述例

(4) サンプル「 R o b o _ 3 」の使い方

サンプルデモプログラム「 R o b o _ 3 」は、電源スイッチをONにすると、前方、左右に取り付けられている光距離センサーによって障害物までの距離をA / D変換器で計測し、予めプログラムされている距離まで接近すると、次の回避動作を行います。障害物が黒色の場合には、光が反射しませんので障害物を検地できません。できる限り白色の障害物をご用意下さい。

中央センサーが障害物に接近した場合

中央センサーが障害物に予めプログラムされている距離まで接近すると、一定時間で後進動作を行い、次にUターン動作を行います。一旦停止後に前進走行を行います。

右側センサーが障害物に接近した場合

右側センサーが障害物に予めプログラムされている距離まで接近すると、一旦停止後、約90度左にターンし、障害物を避けます。次に、一旦停止後に前進走行を行います。

左側センサーが障害物に接近した場合

左側センサーが障害物に予めプログラムされている距離まで接近すると、一旦停止後、約90度右にターンし、障害物を避けます。次に、一旦停止後に前進走行を行います。

フォトレフレクタ RPR - 220 を距離センサーとして使用する為に、I/Oポート P04 端子からトランジスタ 2SC185 のベースに 800 Hz のパルス波を与えます。

図 29 は、ソフトで 800 Hz のパルス波を作るプログラムの記述例です。

```

*****
*      タイマ割込み関数 : T A 0
*****/
void ta0int(void)
{
    time400Cnt--;                               /* 400Hz カウンタ-1      */
    if(time400Cnt == 0){
        port_0 ^= 0x10;                          /* P04 のオン <-> オフ  */
        time400Cnt = TA01SCNT; /* 400Hz カウンタ初期化(8) */
    }
    ta0 = TA0CNT;                               /* タイマ値の初期化    */
}

```

図 29 800 Hz のパルス波出力プログラム例

周波数 800 Hz の周期は 1.25 ms となります。ソフトで半周期 (0.625 ms) の間隔でポート P04 端子から H レベルと L レベルを交互に出力すれば、800 Hz のパルス波が得られます。

従って、タイマ割込みによって半周期毎に割込み関数でポート P04 端子を ON と OFF を繰り返します。タイマ設定値はポートのクロック周期が 2 μs になっているので次の数式から算出します。

$$\text{タイマ設定値} = \frac{0.625 \times 10^{-3}}{2 \times 10^{-6}} = 312.5$$

ここでタイマ A0 を使用します。このタイマは 16 ビットカウンタなので上記のタイマ値を $31 \times 10 = 310$ として 2 つの変数に分け、プログラムの中で最初に定義しておきます。

```

#define TA0CNT 31-1 /* A0:31:タイマカウンタ値 */
#define TA01SCNT 10 /* A0:秒カウンタ値 31*10=310 */

int time400Cnt; /* 400Hz 用カウンタ */

```

(5) 距離の検出方法

光距離センサで得られたDC電圧をA/D変換器に入力し、距離を検出するサンプルプログラムを図30に示します。アナログデータは補正値を用いて次のようにA/D変換し、デジタル変換データを変数an_0に格納します。

```
an_0 = (ad0 & 0x03ff) / 5 ;
```

A/D変換器は10ビットモードで使用するので、変換結果の読み出しレジスタad0の値を上位5ビットをマスクするために0X03ffとのANDをとります。

距離に対するDC出力電圧特性は、表4で表されている通り反比例します。従って、定数100から減算して比例して増加するように変更します。この値を障害物までの距離データとして使用します。

```
an_0 = 100 - an_0;
```

```
void    ad0_int(void)          /* A/D変換割り込み */
{
    unsigned _far int  an_0, an_1, an_2;
    _asm( "%tFCLR      l");    /* 割り込み禁止 */

    adcon0= 0x40;
    an_0 = (ad0 & 0x03ff) / 5 ;    /* A/D ch0 データの取得: 補正値= 5 */
    an_0 = 100 - an_0;

    adcon0= 0x41;              /* A/D ch1 変換開始 */
    an_1 = (ad1 & 0x03ff) / 5 ;    /* A/D-1 データの取得: 補正値= 5 */
    an_1 = 100 - an_1;

    adcon0= 0x42;              /* A/D ch2 変換開始 */
    an_2 = (ad2 & 0x03ff) / 5 ;    /* A/D-2 データの取得: 補正値= 5 */
    an_2 = 100 - an_2;
```

図30 距離の検出プログラム例

9. フラッシュROMへの書き込み方法

ファイルの拡張子が「.c」はC言語ソースプログラムです。「.x30」はデバッガが使用するオブジェクトファイルです。「.mot」は「.x30」ファイルをモトローラ2形式に変換したファイルで、フラッシュROMに書き込むファイルです。

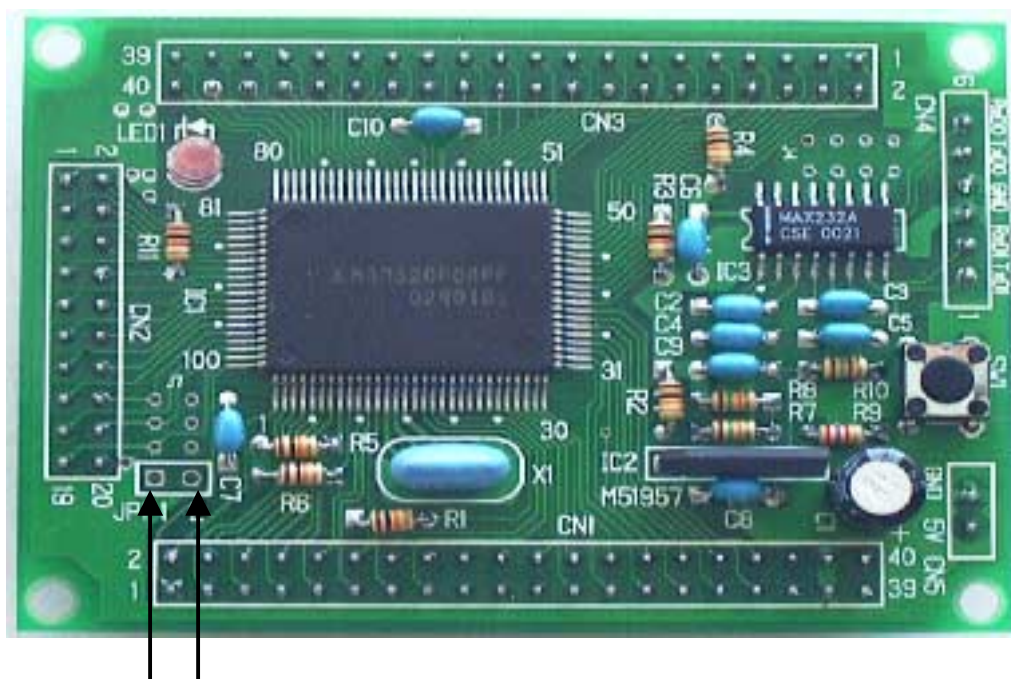
センサーロボット「OAKS16 - SENSOR LABO」には、OAKS16 CPUボードは付属していません。お持ちでない場合には、別途ご購入下さい。

商品に添付されているCD-Rに入っているサンプルデモプログラムをフラッシュROMに書き込んで実行する場合とリモートデバッガKD30からサンプルデモプログラムを実行する方法をご説明します。

(1)、デバッガを使用する為のフラッシュライターソフトの使い方

デバッガ「KD30」を用いてデモプログラムを実行するには、予めフラッシュROMにモニタファイル「Mon_uart.mot」ファイルを書き込まなければなりません。次の手順で書き込んで下さい。

センサーロボット「OAKS16 - SENSOR LABO」とパソコン間をRS232Cケーブルで接続します。また、図31に示されるCPUボード「OAKS16」のJP1端子12間を何らかのショート端子を用いてショートして下さい。



JP1の1 2間をショート端子でショートします。

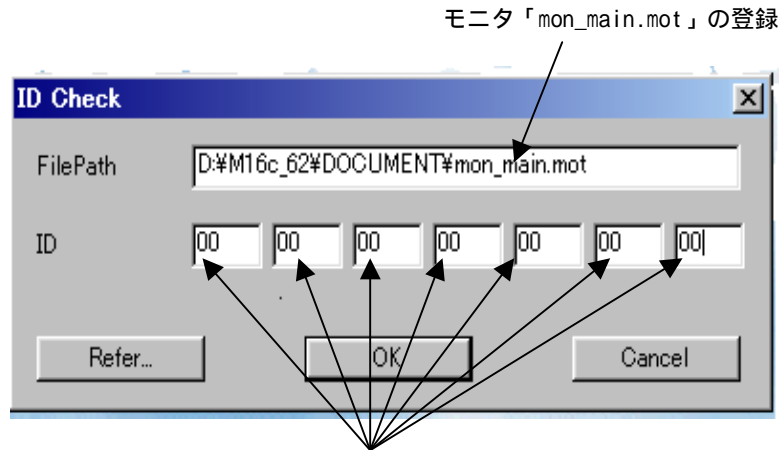
図31 CPUボード「OAKS16」のJP1端子

OAKS16 - SENSOR LABOの電源をオンにし、フラッシュライターソフト「M16CFlash」アイコンをクリックすると図32で示されるウインドウが表示します。

「Refer」ボタンをクリックし、モニタプログラム「mon_main.mot」を登録します。

空欄になっている七つのIDウインドウに「00」を書き込んで下さい。

「OK」ボタンをクリックします。



全てのIDウインドに「00」を入れる

図32 「ID Check」ウインドウ

図33で示される「M16C flash Write」ウインドウが表示します。

: ブランクチェック、書込み、ベリファイ

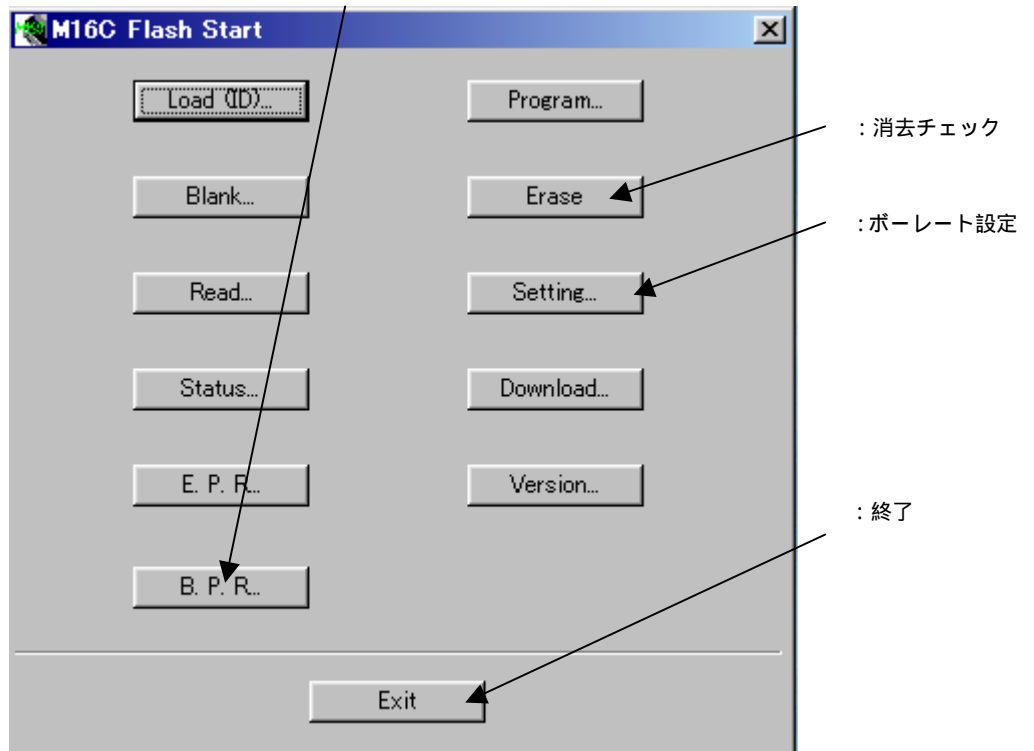


図33 「M16C flash Write」ウインドウ

「Setting」ボタンを押します。図34で示されるウインドウ内の「Baud rate」設定プルダウンメニューからボーレートを図のように57600を選んで下さい。設定後、「OK」ボタンを押して下さい。

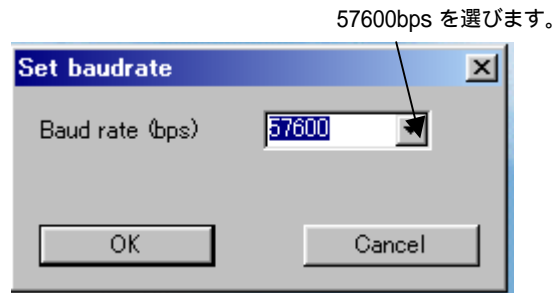


図34 「Set baudrate」ウインドウ

「Erase」ボタンを押します。図35(a)で示されるウインドウの「OK」ボタンを押すと、(b)図で示される「Erase OK」ウインドウが表示します。

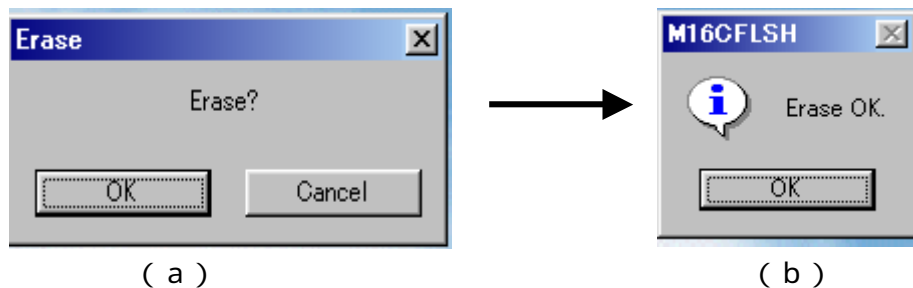


図35 「Erase」チェックウインドウ

「B.P.R」ボタンを押します。このボタンは、ブランクチェック、プログラムの書込み、書込み確認を一連で実行します。図36(a)で示されるプログラムのメモリ範囲が表示されます。確認後、「OK」ボタンを押します。図(b)のウインドウが開きブランクチェックを開始します。

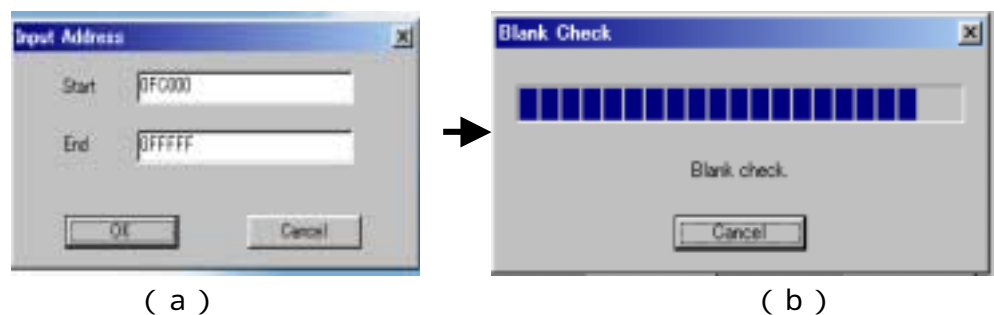


図36 「Blank Check」ウインドウ

ブランクチェックが正常に終了すると、図37(a)に示されるウインドウが表示されます。「OK」ボタンを押すと図(b)のウインドウが開き、書き込みを開始します。

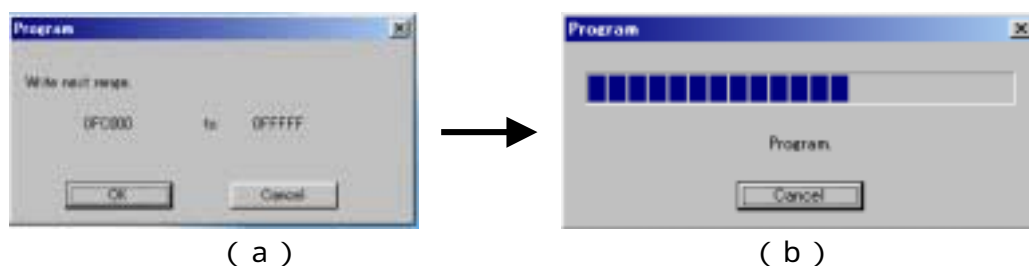


図37 「Program」ウインドウ

プログラムが正常に書き込まれると、図38(a)に示されるウインドウが表示され、書き込みの確認を行います。正常に書き込まれていると図(b)のウインドウが開き、書き込み確認が終了します。「OK」ボタンを押して下さい。

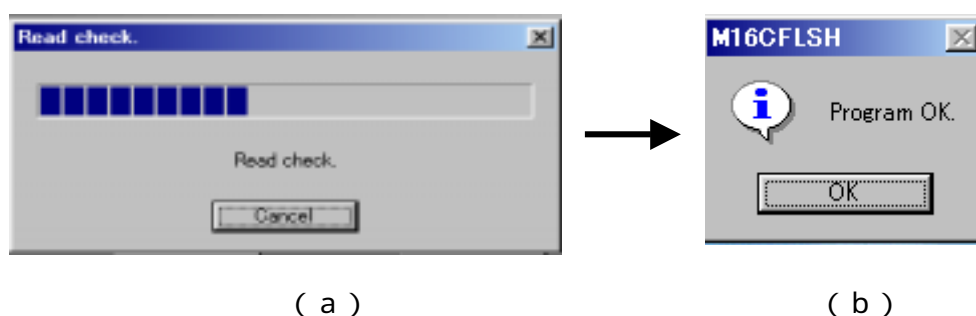


図38 「Read check」ウインドウ

最後に、図33で示される「M16C flash Write」ウインドウから「Exit」ボタンを押して下さい。

(2)、デバッガの起動方法

デバッガ「KD30」を起動するには、OAKS16-SENSOR LABOのシリアルポートからチャンネル1とパソコン間をRS-232Cケーブルで接続します。

デバッガ「KD30」アイコンをクリックすると図39に示される起動ウインドウが表示されるので、パソコン側のシリアル端子はCOM1に固定されています。

ロボットの電源をONにして下さい。また、RS-232Cケーブルの接続を確認後、「OK」ボタンを押します。もし、エラーウインドウが表示される場合には、ケーブルの接続や電源を再確認します。また、予めフラッシュROMにモニタファイル「Mon_main.mot」ファイルが書き込まれていなければなりません。

図39で示される起動ウインドウの詳しい使用方法は「KD30」に添付されているマニュアルをご覧下さい。

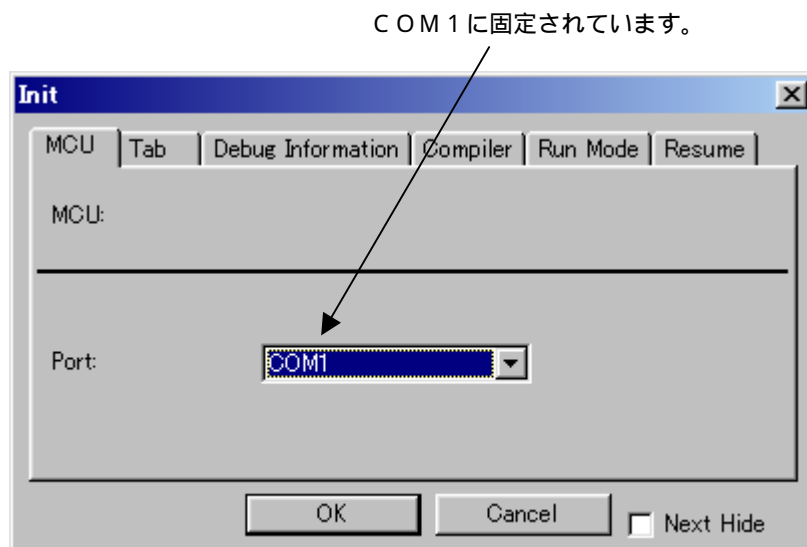


図 3 9 デバッガ起動ウインドウ

「OK」ボタンを押すと、図 40 で示されるデバッグウインドウが表示されます。

ファイルのロード方法

ファイルをロードするには、メニューから左隅の「File」をクリックします。プルダウンメニューから「Download」を選びます。更にこの中の「Load Module」を選ぶとファイルロードウインドウが開きます。ロード可能なファイルは、拡張子に「.x30」です。デモ用のファイルから「Robo__1.x30」と「Robo__2.x30」と「Robo__3.x30」がロード可能です。

プログラムの実行方法

プログラムを実行するには、通常プログラムカウンタにロードしたユーザプログラムの先頭番地を予め設定する必要があります。しかし、デモプログラムの場合には、ロードの段階で自動的にプログラムカウンタに先頭番地が設定されます。


プログラムの実行は、左隅の実行アイコン「」をクリックするとプログラムが“RUN”します。

図 40 で示されるデバッグウインドウの詳しい使用方法はリモートデバッガ「KD30」に添付されているマニュアルをご覧ください。

(1) エントリー版 (新バージョン) の仕様について

プログラムの記述について

旧バージョンのプログラムをエントリー版でコンパイルする際、表10で示される語の前にアンダースコア (_) を付加して下さい。

表10 新旧バージョンの記述相違一覧

KNC30 旧バージョン版	NC30 エントリー版
<code>inline</code>	<code>__inline</code>
<code>near</code>	<code>__near</code>
<code>far</code>	<code>__far</code>
<code>asm()</code>	<code>__asm()</code>

TM統合開発環境「TMV3.11」について

- ・ インスペクタ機能はご使用になれません。
- ・ ブラリプロジェクトは作成できません。

ソフトウェアおよびユーティリティについて

下記に示すソフトウェアおよびユーティリティはご使用になれません。

表11 使用できないソフトウェアおよびユーティリティー一覧

ソフトウェア	STK ビューワ、MPA ビューワ、アセンブルオプティマイザ (aopt30)、ライブラリアン (lb30)、構造化記述アセンブラ (pre30)、] 標準関数ライブラリソースファイル
ユーティリティ	utl30 (SBDATA 宣言&SPECIAL ページ宣言ユーティリティ)

NC30とAS30のオプションについて

マニュアルに記載されている下記のオプションはご使用になれません。

表 1 2 使用できないCコンパイラオプション一覧

Cコンパイラ (N C 3 0)	
デバッグ用オプション	genter -gno_reg
最適化オプション	-O[1-5], -OR, -OS, -Oconst(-OC), -Ono_bit(-ONB), -Ono_break_source_debug(-ONBSD), -Ono_float_const_fold(-ONFCF), -Osp_adjust(-OSA), -Ostack_frame_align(-OSFA), Oloop_unroll(-OLU), Ono_asmopt(-ONA), -Ono_logical_or_combine(-ONLOG), -Ocompare_byte_to_word(-OCBTW), -Ono_stdlib
生成コード変更オプション	-finfo, -fuse_DIV(-fUD), -fanshi, -fnear_ROM(-fNROM), -fsmall_array(-fSA), -fno_align(-fNA)
アセンブル、リンクオプション	-as30, -ln30
その他のオプション	-dsource(-dS), -dsource_in_list(-dSL)

表 1 3 使用できないアセンブラオプション一覧

アセンブラ (A S 3 0)	
オプション	-finfo, -P, -M

(2) T M統合化開発環境の使い方

「 T M 」を起動すると細長いプロジェクトバーが表示されます。マウスによって右端を縮めると図 4 1 で示されるフローティング状態のプロジェクトバーが表示されます。

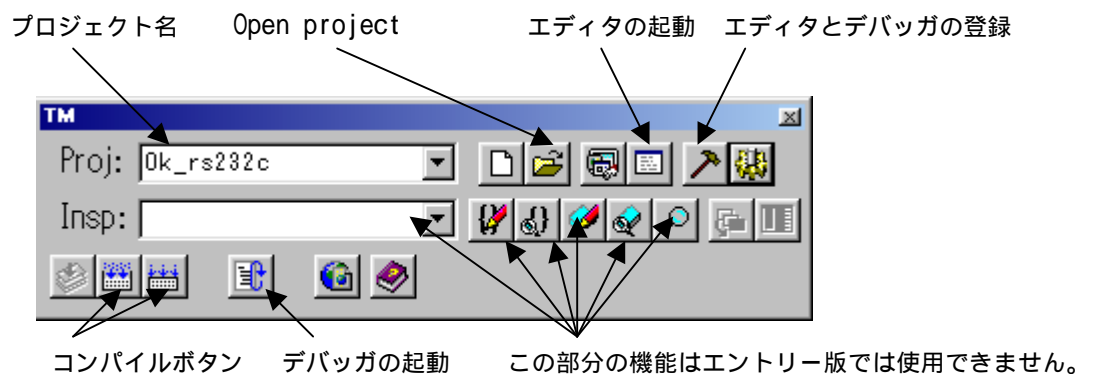





図 4 1 プロジェクトバー



エディタとデバッガの登録

最初に (ツールの登録) ボタンをクリックし、使い慣れたエディタとリモートデバッガ「KD30」を登録します。


ここで登録しないと、 (エディタの起動) ボタンと (デバッガの起動) ボタンが使用できません。

プロジェクトファイルのロード



TM統合化開発環境「TM」を用いてWindows上でコンパイルを行うには、プロジェクトファイルを作成しなければなりません。新規のユーザプログラムからプロジェクトファイルを作成しコンパイルする方法は、付属のユーザズマニュアル(PDFファイル)をご覧ください。本製品のデモサンプルプログラムにはプロジェクトファイル「Ok_rs232c.tmk」が入っています。

プロジェクトファイルをロードするには (プロジェクトを開く) ボタンをクリックします。開くウインドウから付属のCD-Rからコピーしたドライブとディレクトリを選び「Robo_3.tmk」をロードします。 で示されるプロジェクトウインドウにファイル名が表示されます。

コンパイル方法

プロジェクトで設定したソースプログラム「Robo_3.c」をコンパイルするには、 (ビルドまたはリビルド) ボタンをクリックします。ウインドウにコンパイルメッセージが表示されます。スクロールバーによって前後のメッセージを読むことが可能です。

エディタの起動

 (ツールの登録) ボタンによって前もって使い慣れたエディタを登録しておく、 (エディタの起動) ボタンでエディタが起動します。ソースプログラムをロードしてプログラムを修正できます。

デバッガの起動



 (ツールの登録) ボタンによってリモートデバッガを登録しておく、 (デバッガの起動) ボタンをクリックするとリモートデバッガ「KD30」が起動できます。

図4-1において、2段目のインスペクタとその関連機能は、エントリー版では使用できません。また、TM統合化開発環境「TM」の詳しい使用方法は、OAKS16CPUボードに付属のCD-Rに入っているユーザズマニュアルをご覧ください。

11. プログラムの作成手順

センサーロボット「OAKS16 - SENSOR LABO」には、プログラム開発の為にCコンパイラ、アセンブラ、リモートデバッガ、フラッシュライターソフトが付属していません。別途OAKS16 CPUボードに付属しております。

C言語プログラムは、他社のCPUボードで開発したソースプログラムを、OAKS-LABOで使用するM16C/62シリーズ用プログラムに容易に移植することが可能です。また、アセンブラも熟知する必要がありませんので、C言語でプログラムする事をお勧めします。また、商品に添付しているデモプログラムは全てC言語プログラムとなっています。

(1)、プログラム開発の手順

プログラムを開発するには、図42に示されるように次の手順でプログラムを開発します。

プログラム図法を用いてフローチャート等のアルゴリズムを作成します。

使い慣れたエディタを用いてソースプログラムを記述します。

コーディングが終了したら、コンパイラ「NC30WA」を用いてコンパイルを行います。

エラーメッセージが表示された場合には、 項に戻り、エラーを修正します。

コンパイルが終了したら、リモートデバッガ「KD30」を用いて、ロボットがソフトウェア設計通りに動作するか確かめます。

正常に動作しない場合には、原因を究明し、 項に戻りプログラムを修正します。

正常にロボットが動作したら、フラッシュライターソフト「M16CF1sh」を用いてフラッシュROMにプログラムを書き込みます。

プログラム開発を終了します。

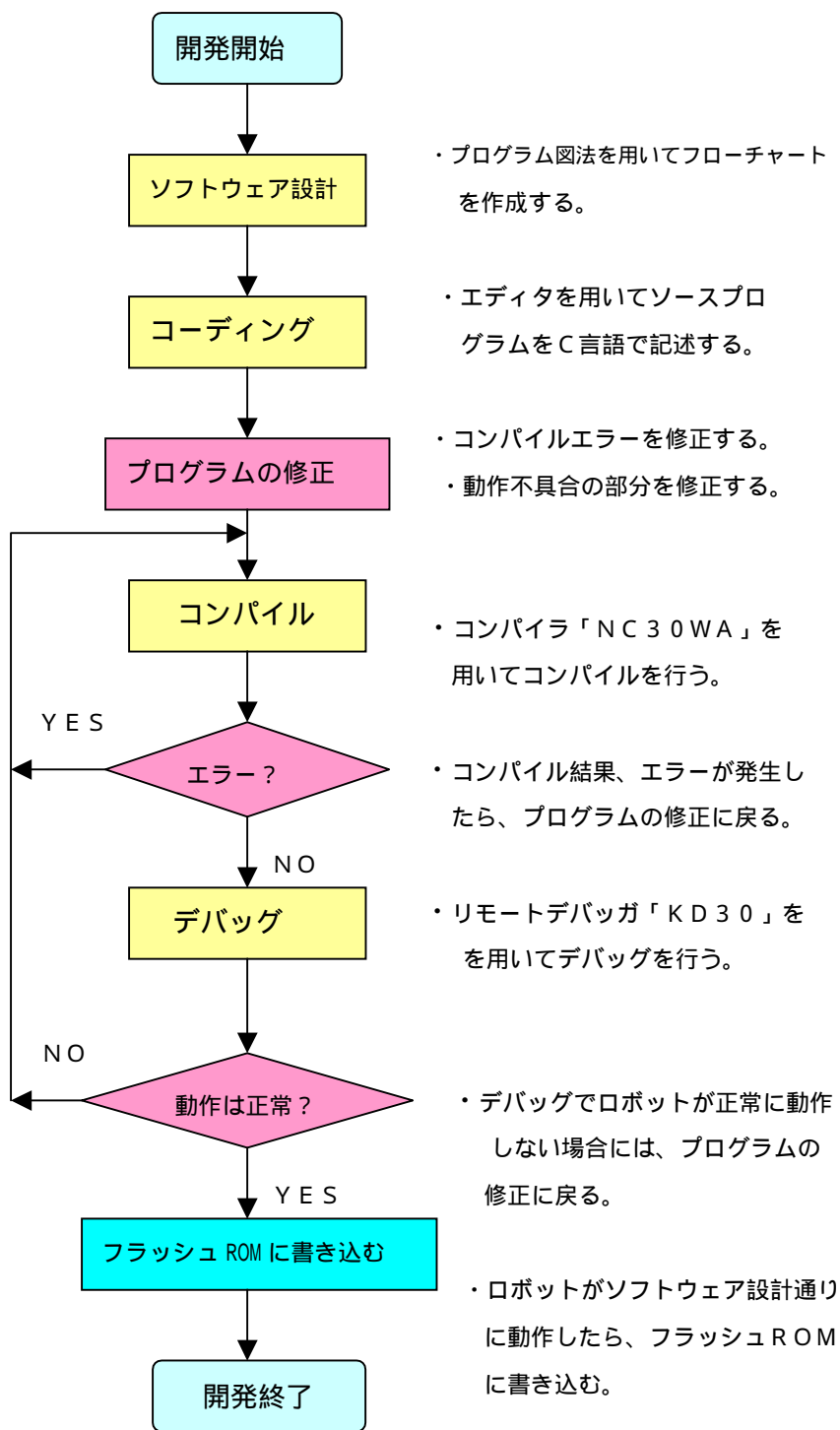


図4-2 プログラム開発の手順

(2)、コンパイル方法

M16C / 60シリーズ専用のコンパイラ「NC30WA」にはCコンパイラとアセンブラが標準で付属しております。図4-3はコンパイル手順を示したフローチャートです。作成したC言語プログラム「Robo_3.c」はコンパイルドライバ「nc30」によってアセンブラプログラ

ム「Robo_3.r30」に変換します。

Cソースプログラムにエラーがなければ、スタックポインタ等を記述したスタートアップルーチン「ncrt0.a30」と割り込みを使用する場合には、インクルードファイル「sect30.inc」の割り込みベクタテーブルを書き換えてアセンブルします。全てのファイルにエラーがなければリンカでリンクします。リンク段階でエラーがなければデバッグ可能な実行ファイルが完成します。次に、リモートデバッガ「KD30」を用いてプログラムが正常に動作する事を確認後、ファイル形式変換ソフト「Lmc30」によってモトローラ2形式のファイル「Robo_3.mot」に変換し、フラッシュライタソフト「M16CF1sh」で書き込み、プログラム作成は終了です。

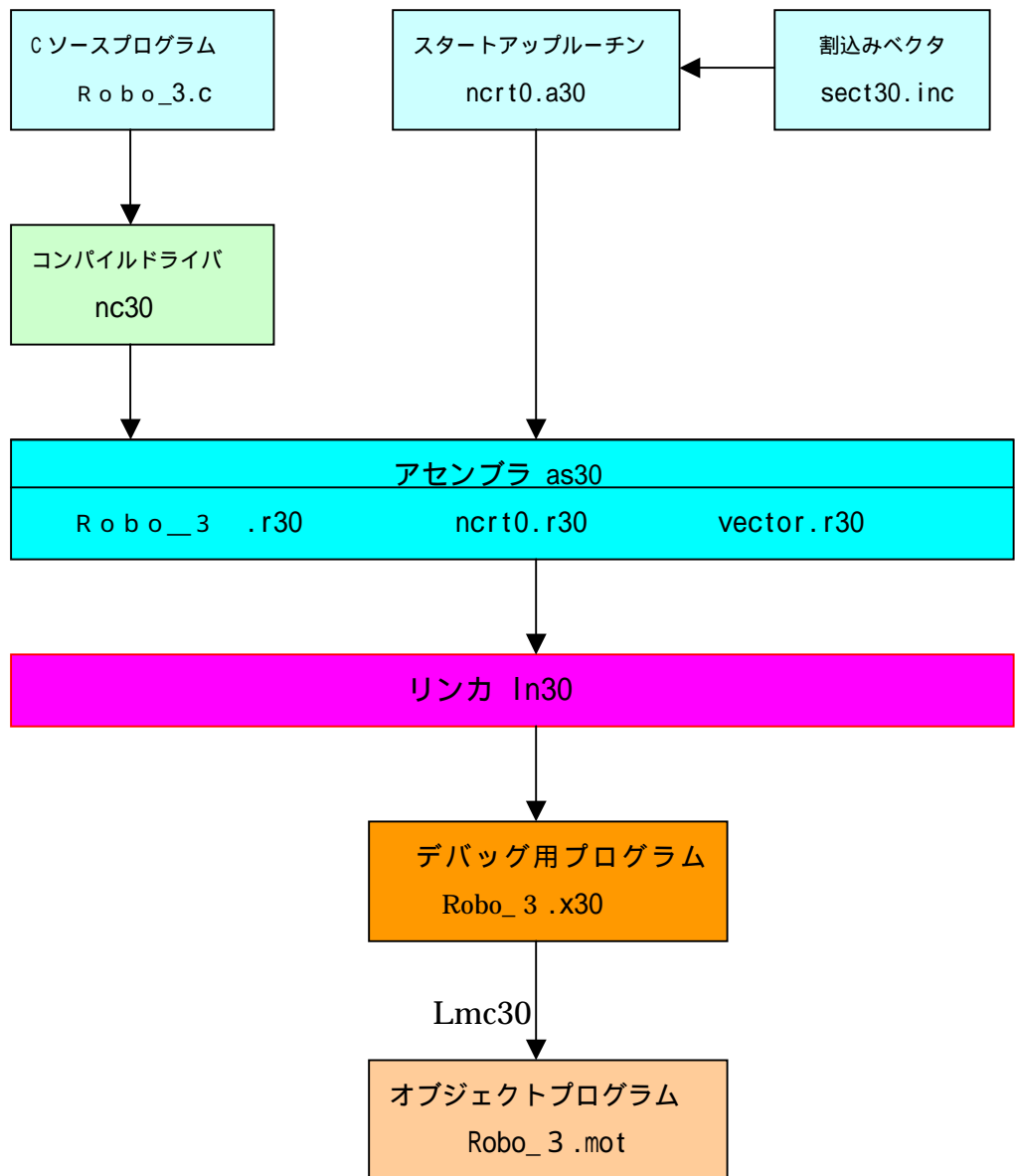


図43 コンパイラ「NC30WA」のコンパイル手順

(3)、コンパイル用バッチファイル例

コンパイラ「NC30WA」は通常、TM統合化開発環境の元でコンパイルすると便利ですが、バッチファイルによってDOS窓上でコンパイルする事ができます。

図44は、バッチファイルに予めコンパイルの手順を記述した記述例です。

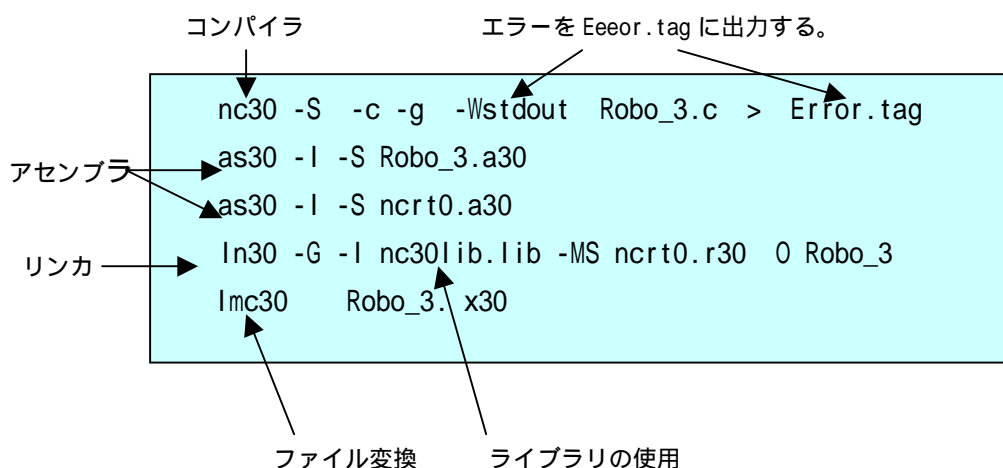


図44 コンパイル用バッチファイル記述例

図44で示されるバッチファイルは、出荷時にフラッシュROMに書き込まれているデモソフト「Robo_3.c」をコンパイルする為のバッチファイルです。

コンパイラのオプションに記述されている「Wstdout」は、エラーリストを出力する指定です。行の最後に「> Error.tag」のように記述すると、エラーリストをError.tagファイルに書き込まれます。このエラーファイルを見ながらソースプログラムを修正すると便利です。

図44に使用されているオプションの詳しい使い方は、コンパイラ「NC30WA」マニュアルをご覧ください。

(4) I/Oポートの記述方法

M16C/62シリーズのCPUは、メモリマップI/O方式で全てのI/Oアドレスを0000番地から0040番地に割り付けます。C言語でI/Oポートを定義するには、図45のように拡張機能による絶対アドレス指定を用いると便利です。「#program ADDRESS」は、関数外で定義された変数および関数内でスタティック宣言された変数のみ有効となります。ポート名(変数名)には、適当な名前を付けて、次のように記述します。

```
# program ADDRESS 変数名 絶対アドレス
```

上記の宣言により、変数名を絶対アドレスに配置します。このような拡張機能による絶対アドレス指定を用いると、メモリ使用量を節約できます。

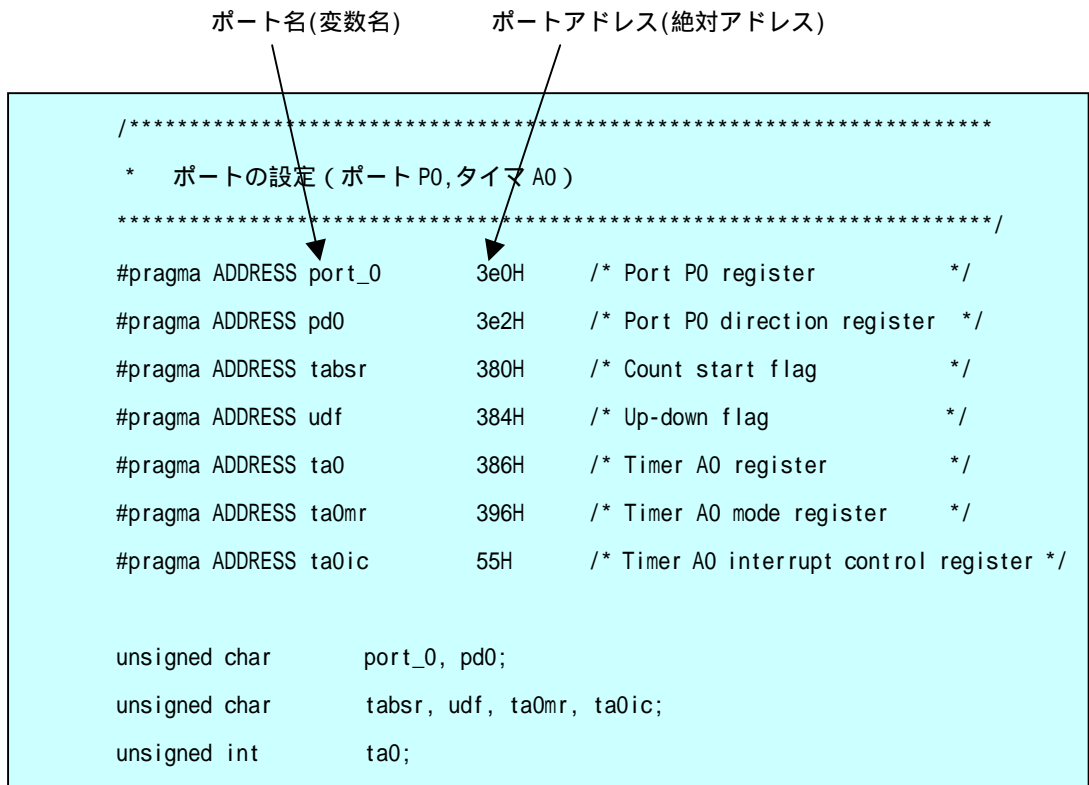


図 4 5 I / Oポートの記述例

(5)、割り込み処理

M16C / 62 シリーズの CPU で割り込みを C 言語関数として記述する手順は次の 2 つです。

割り込み処理関数の定義。

割り込みベクタテーブルの登録。

割り込み処理関数の定義は、割り込み関数名に適当な名前を付けて、次のように記述します。

```
# program INTERRUPT 割り込み関数名
```

上記のように記述すると、指定した関数の入り口と出口において、通常の関数の手続き以外に、全レジスタの退避、復帰と“`reit`”命令を生成します。割り込み処理関数の型は、引数 / 戻り値共に `void` 型のみ有効です。図 4 6 は、タイマ A 0 を割り込みに使用する為の割り込み処理関数の記述例です。

割り込み関数名

```

/*****
*   割り込み関数定義(タイマ A0)
*****/
void far   ta0int();   /* 割り込み関数 */
#pragma INTERRUPT ta0int

```

図 4 6 割り込みの記述例

(6)、割り込みベクタテーブルへの登録

割り込みを正常に使用する為には、割り込み処理関数を定義すると共に割り込みベクタテーブルに登録する必要があります。

割り込みベクタテーブルの変更は、次の手順で行います。

割り込み処理関数名を疑似命令“`.glob`”で外部定義します。

使用する割り込みのダミー関数“`dummy__int`”を、割り込み処理関数名に変更します。

図 4 7 は、スタートアップルーチン「`ncrt0.A30`」に記述されている見出し文「`Interrupt section start`」内に割り込み処理関数を記述したものです。外部定義では次のように必ずアンダーバーを付けて記述して下さい。

```
.glob    _ta0int
```

また、タイマ A 0 を割り込みで使用する場合には、図 4 8 に示されるインクルードファイル「`sect30.inc`」のベクタテーブル見出し名「`variable vector section`」21番のダミー関数“`dummy__int`”を次のように必ずアンダーバーを付けて記述して下さい。

```
.lword   _ta0int
```

割り込み処理関数名の記述

```

;=====
; Interrupt section start
;-----
.insf      start,S,0
.glob     _ta0int
.glob     start
.section   interrupt

start:

```

図 4 7 スタートアップルーチンに割り込み処理関数名を記述する例

割り込み処理関数名の記述

```

.else
.lword    dummy_int      ; vector 0 (BRK)
.org      (VECTOR_ADR +44)
.lword    dummy_int      ; DMA0 (for user)
.lword    dummy_int      ; DMA1 2 (for user)
.lword    dummy_int      ; input key (for user)
.lword    dummy_int      ; AD Convert (for user)
.org      (VECTOR_ADR +68)
.lword    dummy_int      ; uart0 trance (for user)
.lword    dummy_int      ; uart0 receive (for user)
.lword    dummy_int      ; uart1 trance (for user)
.lword    dummy_int      ; uart1 receive (for user)
.lword    _ta0int      ; TIMER A0 (for user) (vector 21)
.lword    dummy_int      ; TIMER A1 (for user) (vector 22)
.lword    dummy_int      ; TIMER A2 タイマ A2(vector 23)
.lword    dummy_int      ; TIMER A3 タイマ A3(vector 24)
.lword    dummy_int      ; TIMER A4 (for user) (vector 25)
.lword    dummy_int      ; TIMER B0 (for user) (vector 26)
.lword    dummy_int      ; TIMER B1 (for user) (vector 27)
.lword    dummy_int      ; TIMER B2 (for user) (vector 28)
.lword    dummy_int      ; INT0 (for user) (vector 29)
.lword    dummy_int      ; INT1 (for user) (vector 30)
.lword    dummy_int      ; INT2 (for user) (vector 31)
.endif

```

図 4 8 「sect30.inc」ベクターテーブルの記述例

12. Flash ROM への書込み

デバッガによってセンサーロボット「OAKS16 - SENSOR LABO」が正常に動作することが確かめられ後、最後にFlash ROMに書き込んで完成です。デバッグで使用したデバッグ用プログラムをコンパイラに添付されているファイル形式変換プログラム「lmc30」を用いてモトローラS2形式のROM化ファイル(Oklabo_4.mot)に変換します。デバック時の状態でRS232Cケーブルはそのまま接続しておきます。また、CPUポート上のJP1をショートするとFlash ROMへの書込みが可能となります。書込みにはフラッシュライターソフト「M16CFlash」を起動します。使用方法は、第10章の(1)項「デバッガを使用する為のフラッシュライターソフトの使い方」をご欄下さい。

表14は、図33で示される「M16C flash Write」ウインドウのメニューボタンの機能を表したものです。

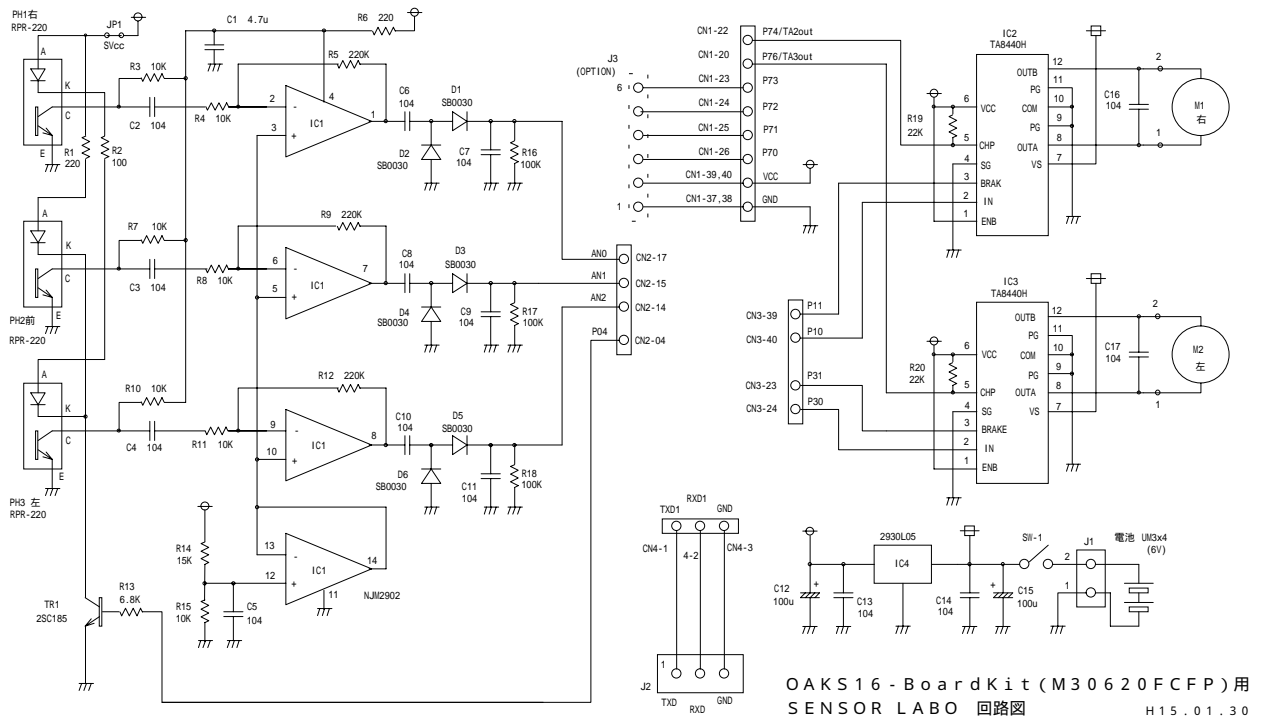
表14 各メニューボタンの機能

メニュー	機能
Load(ID)	ファイルの選択とIDチェックを行います。
Blank	指定した領域に対してブランクチェックを行います。
Read	指定したファイルの内容とを比較します。
Status	マイコンのステータスを表示します。
B.P.R	ブランクチェック、プログラム、ベリファイを順次行います。
Program	指定したファイルをフラッシュROMに書き込みます。
Erase	フラッシュROMの全領域をイレースします。
Setting	通信設定を行います。
Exit	書込みを終了します。

付 録

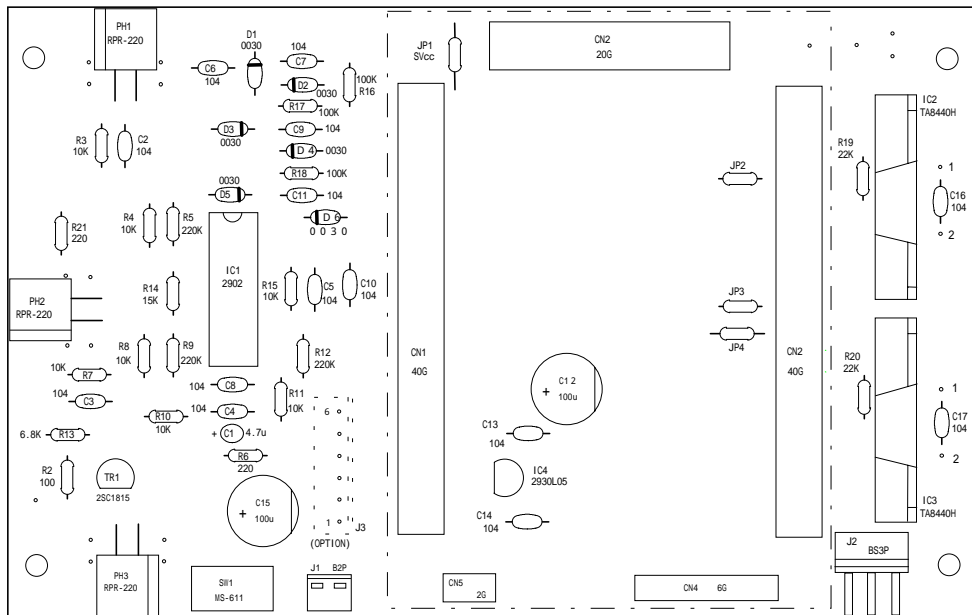
付録 1 ドライバーボード回路図

拡大してご覧下さい。



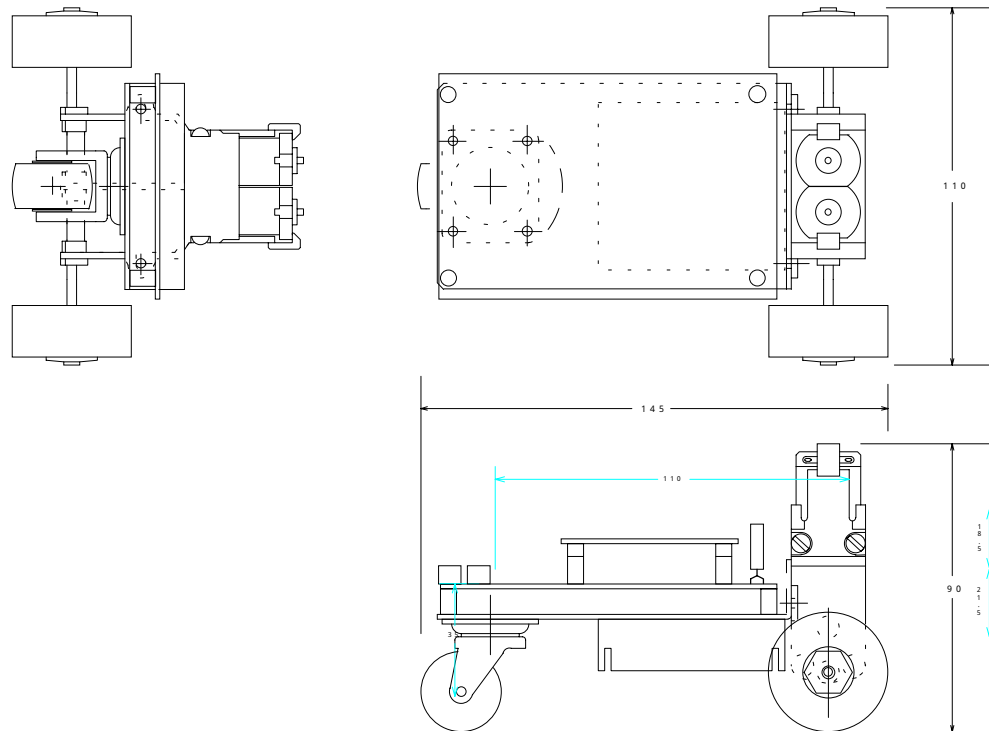
付録 2 ドライバーボード構成図

拡大してご覧下さい。



付録 3 OAKS16 - SENSOR LABO外形図

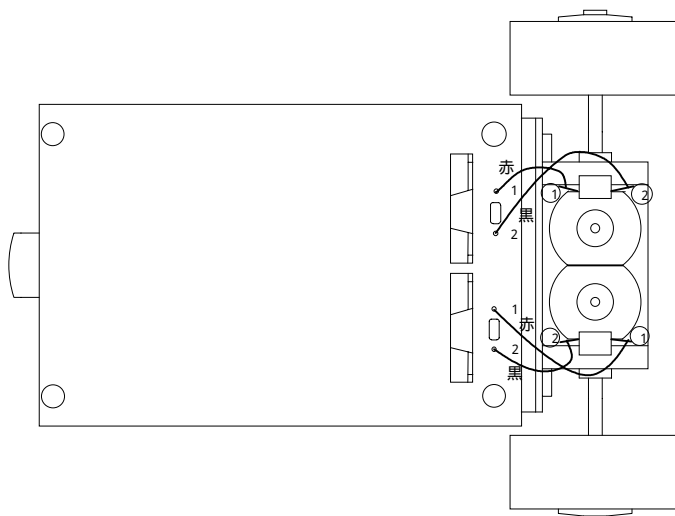
拡大してご覧下さい。



付録 4 モータ配線図


拡大してご覧下さい。

モータ 配線



付録 7 部品表

記号	型番	メーカー	数量	備考
IC1	NJM2902	JRC	1	4ヶ入 OP AMP (相当品)
IC2,3	TA8440H	東芝	2	モーター用 IC
IC4	NJM2930L05	JRC	1	5V 3端子レギュレータ (相当品)
PH1-3	RPR-220	ロム	3	反射型フォトセンサ (相当品)
D1-6	SB0030-04A	サンヨー	6	ショットキリアクタート (相当品)
TR1	2SC1815GR	東芝	1	トランジスタ (相当品)
R1,6	RD16S-220	コーア	2	220 オーム炭素皮膜抵抗 (相当品)
R2	100	"	1	100 オーム炭素皮膜抵抗 (相当品)
R3,4,7,8,10,11,15	10K	"	7	10K オーム炭素皮膜抵抗 (相当品)
R5,9,12,	220K	"	3	220K オーム炭素皮膜抵抗 (相当品)
R13	6.8K	"	1	6.8K オーム炭素皮膜抵抗 (相当品)
R14	15K	"	1	15K オーム炭素皮膜抵抗 (相当品)
R16,17,18	100K	"	3	100K オーム炭素皮膜抵抗 (相当品)
R19,20	22K	"	2	22K オーム炭素皮膜抵抗 (相当品)
JP1-4	Z16	"	4	0 オーム抵抗 (相当品) 又はリフト線
C1	SS1C475M	エルパ	1	4.7u/16v タンタルコンデンサ (相当品)
C2-11,13,14,16,17	RPE132F104Z50	村田	14	0.1u/50v セラミックコンデンサ (相当品)
C12	SRA25VB100M	ニッケミ	1	100u/25V 電解コンデンサ (相当品)
C15	SRM25VB100M	"	1	100u/25V 電解コンデンサ (相当品)
SW1	MS-611A	ミヤマ	1	トグルスイッチ (相当品)
CN1,3	OX-114-DS-40G	OAKS	2	コネクタ (相当品)
CN2	OX-114-DS-20G	"	1	コネクタ (相当品)
CN4	OX-114-SS-6G	"	1	コネクタ (相当品)
CN5	OX-114-SS-2G	"	1	コネクタ (相当品)
電池ホルダ	MC-304-3 (UM3x4)	タカチ	1	電池ホルダ (相当品)
ハウジング	H2P-SHF-AA	日圧	1	電池ホルダ用コネクタ (相当品)
コンタクト	SHF-001T-0.8SS	"	2	電池ホルダ用コンタクト (相当品)
J1 (ポストトップ型)	B2P-SHF-1AA	"	1	コネクタ (相当品)
J2 (ポストサイド型)	BS3P-SHF-1AA	"	1	コネクタ (相当品)
シャーシ	ATK-9146-05		1	
キャスタ	NO.420-G-N25-1	トキキヤ	1	
ツインモーターキヤボックス	No.97	タミヤ	1	
トラックタイヤセット (36mm)	No.101 (4ヶ入)	"	2/4	
PCB	9164		1	

 **ご注意**

- (1) 本書の内容の一部または全部を無断転記することは禁止されています。
- (2) 本書の内容は将来予告なしに変更することがあります。
- (3) 本書は内容について万全を期して作成致しましたが、万一ご不審な点や誤り、記載洩れなどお気づきのことがありましたら、お買い求めの販売代理店にご連絡下さい。
- (4) 運用した結果の影響については(3)項にかかわらず責任を負いかねますのでご了承下さい。

参考文献

- (1) 「マイコン制御ロボットの製作」横山直隆著 シータスク発行
- (2) デバッガ「KD30 ルネサステクノロジ社
- (3) コンパイラ「NC30WA Ver4.00 Release 2 Entry」 ルネサステクノロジ社
- (4) フラッシュライター「M16CF1sh」 ルネサステクノロジ社
- (5) 統合開発環境「TM」 ルネサステクノロジ社

商標

Windows 95 / 98 / ME / 2000 / XPは、米国マイクロソフト社の登録商標です。

OAKS16 - SENSOR LABO ユーザーズマニュアル

2003年3月23日 第一冊発行

2003年4月 1日 第一冊改訂版発行

編集・発行 **オークス電子株式会社**

〒101-0025 東京都千代田区神田佐久間町3 21 3

第一千代田ビル3F

TEL (03)3863-1121 FAX (03)3863-1130

<http://WWW.oaks-ele.com>

