お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

H8/300L SLP シリーズ

DC モータの精密制御 (DCmotor)

要旨

このアプリケーションノートでは, H8/38024 SLP MCU を使用してブラシ DC サーボモータを制御する方法を紹介します。H8/38024 SLP MCU を使用する利点は，PWM，シリアルコミュニケーションインタフェース (SCI)，タイマなどの豊富な内蔵周辺機能です。このサーボモータシステムはプリンタ，プロッタ，スキャナなどの位置制御に応用できます。

動作確認デバイス

H8/38024

目次

1. はじめに

単一電源ブラシ DC モータには次の 3 種類があります。

直流直巻 (単一方向)

- 直流分巻 (接続変更による双方向)
- 永久磁石 (逆電流による双方向)

このアプリケーションノートでは，永久磁石型 DC モータを使用します。モータのステータ (固定子) は 2 つ以上の永久磁石で作られるいくつかの極から成り，ロータ (電機子) は機械的なコミュテータ (整流子) に接続された巻き線から成ります。励起された異極の巻き線とステータの磁石は引き付け合い，ロータとステータが直線に並ぶまで回転します。ロータとステータが直線に並んだ時点で，ブラシがコミュータの接点を動かして次の巻き線を励起します。

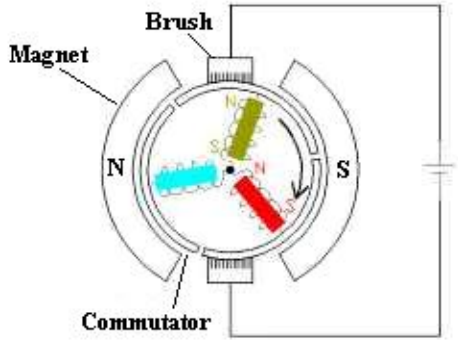


図 1　永久磁石型 DC モータのブロック図

DC モータの長所と短所を表 1 に示します。

表 1　DC モータの長所と短所

長所	短所
制御が容易	メンテナンスが必要
効率	危険な環境で使用不可
同精度の誘導モータより小型	誘導モータより音が大きい
速度制御用の部品が少ない	低馬力 (HP)

2. システム概要

図2にこのアプリケーションノートで説明するサーボモータシステムのブロック図を示します。システムは以下の部品から構成されます。

- H8/38024 SLP MCU
- RS-232C インタフェース
- フルブリッジ PWM モータドライバ
- Faulhaber 社製ブラシ DC モータ (オプティカルエンコーダ付き)

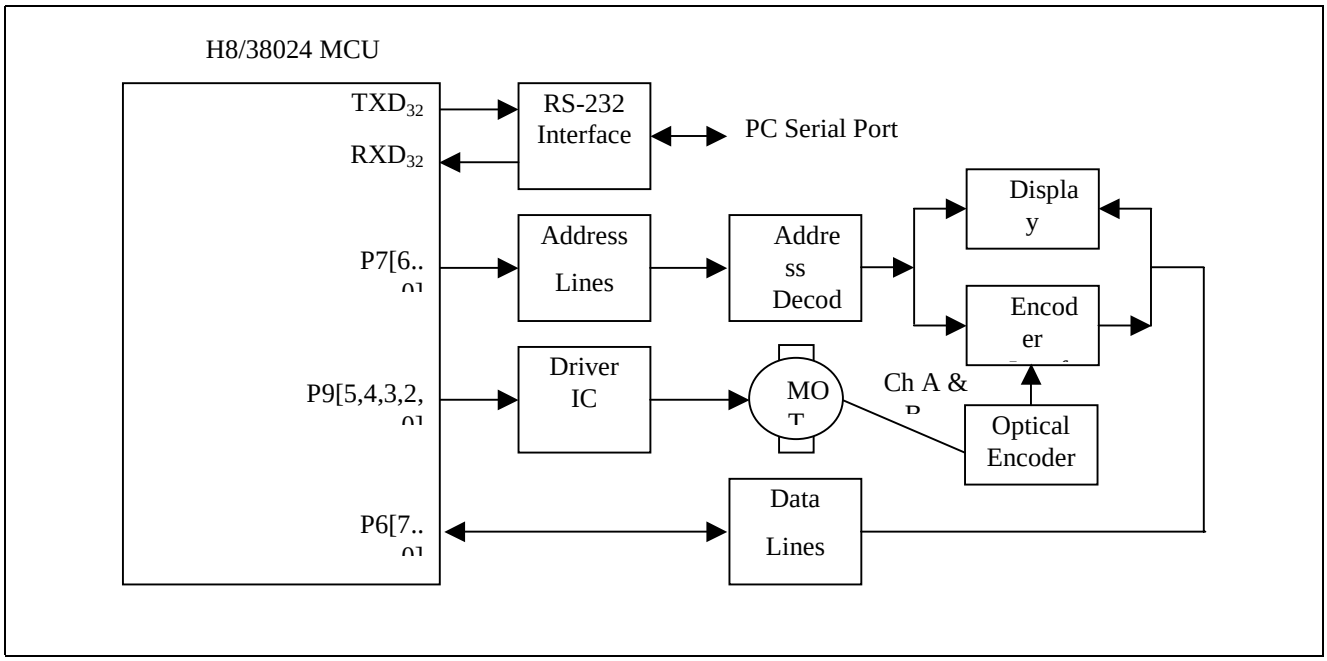


図 2　システムブロック図

このアプリケーションノートの応用例で使用する MCU の機能を以下に示します。

- モータ位置の測定
- 速度特性の生成
- 補償アルゴリズムの計算
- PWM モータドライバを駆動する信号の生成
- RS-232C インタフェースを介して所望する速度と実際の速度データの PC への転送

使用可能な 2 チャネルの 10 ビット分解能の PWM (Pulse Width Modulation) モジュールのうち，1 チャネルを使用して，モータ駆動信号を生成します。MCU の動作周波数が 10MHz のとき，PWM 周波数は 20kHz です。PWM 信号のデューティ比によってモータに印加するトルクが決まります。その PWM 信号は，1.3A を供給できる H-ブリッジドライバ IC に接続されています。

表 2 の特性をもつ Faulhaber 社製ブラシ DC モータ (オプティカルエンコーダ付き) を使用します。

表 2　モータ特性

パラメータ	値	単位
公称電圧	6	V
無負荷速度	9300	rpm
トルク定数	6.10	mNm/A

3. ハードウェアの適用

表 3 に周辺機能の使用方法を示します。

表 3　周辺機能の説明

周辺機能	説明
ポート 6[7..0]	**データバス**
ポート 7[6..0]	**アドレスバス**
P77	WRITE/READ_N
SCI (TXD_{32} **と** RXD_{32})	**ホスト** PC **との通信**
P90/PWM1	*PHASE
P92	*REF
P93	*ENABLE_N
P94	*MODE
P95	*BRAKE_N
タイマ F	**サーボ更新のタイムベース**

* PWM モータドライバ (A3953SB) に接続。詳細は，3.6 章を参照してください。

3.1 電源

このアプリケーション例では，3 つの別々の電源が必要です。

- DC モータ用に 6V
- 74HCT138，レベルシフタ，英数字ディスプレイ，直交エンコーダインタフェース用 5V
- MCU, RS-232C トランシーバ，レベルシフタ, PWM モータドライバ用 3.3V

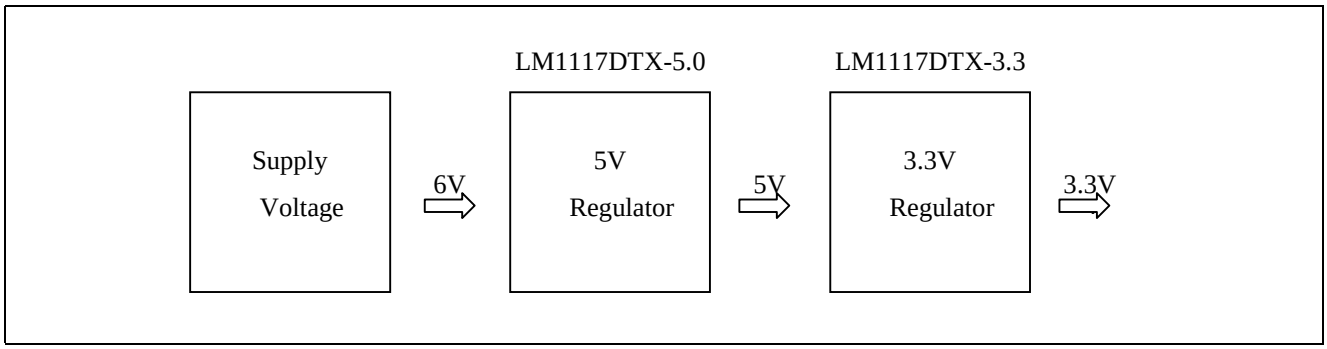


図 3　電源

3.2 アドレスバスとデータバス

図 4 で示すように，H8/38024 SLP MCU をメモリマップされた外付けデバイス，メモリ，周辺機器にアクセスするために，図 4 に示す汎用 I/O ポートを使用して，制御信号の WRITE/READ_N を伴った，アドレスバスおよびデータバスをそれぞれ作成する必要があります。MCU は 3.3V の電源で動作しますが，他のいくつかのデバイスは 5V で動作します。レベルシフタで，MCU を 5V のデバイスにインタフェースします。

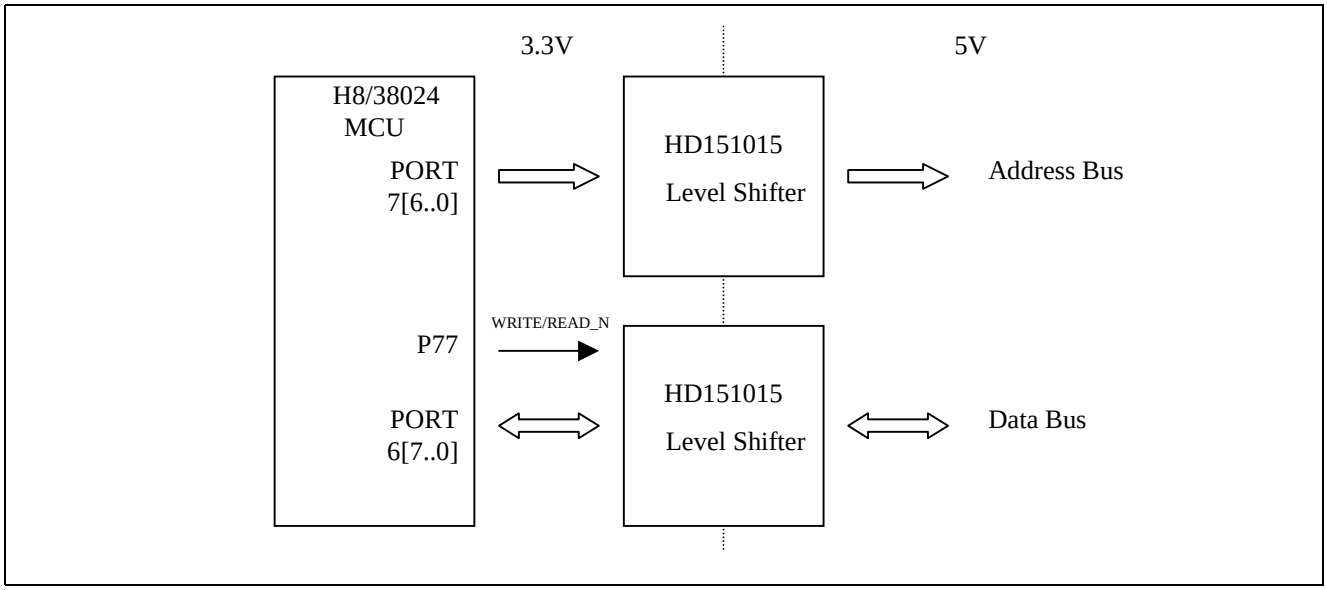


図 4　アドレスバスとデータバス

3.3　アドレスデコーダ

3-8 ラインデコーダ 74HCT138 3 を使用し，メモリマッピングされた英数字のインテリジェントディスプレイ(DLR1414) と直交デコーダを選択します。表 4 にアドレスマップ一覧を示します。

表 4　アドレスマッピング

アドレス (16 進数)	デバイス
44	エンコーダ (上位バイト)
45	エンコーダ (下位バイト)
C0	ディスプレイ (1 桁目)
C1	ディスプレイ (2 桁目)
C2	ディスプレイ (3 桁目)
C3	ディスプレイ (4 桁目)
C8	リセットエンコーダ

3.4　RS-232C トランシーバ

シリアルコミュニケーションインタフェース (SCI) の TXD32 ピンと RXD32 ピンを Sipex 社製 RS-232C トランシーバ SP3232 に接続します。それにより MCU はホスト PC と通信できます。

3.5　オプティカルエンコーダ付き DC マイクロモータ

Faulhaber 社製オプティカルエンコーダ (1 回転当たり 180 パルスの 09BP シリーズ) と DC マイクロモータ(2230 U 006 S シリーズ) を使用して位置決めや軸速度と回転方向を表示，制御します。図 5 に示すように，2 つの LED から発せられた光は，いくつかの光送信用のすき間がある金属円盤を通過して光を送信して 2 つのチャネル A, B の位相を 90 度の位相差をもつ信号を発生させます。時計回りの場合は，チャネル B はチャネル A より位相が先行します。同様に，反時計回りの場合は，チャネル B はチャネル A より先行します。

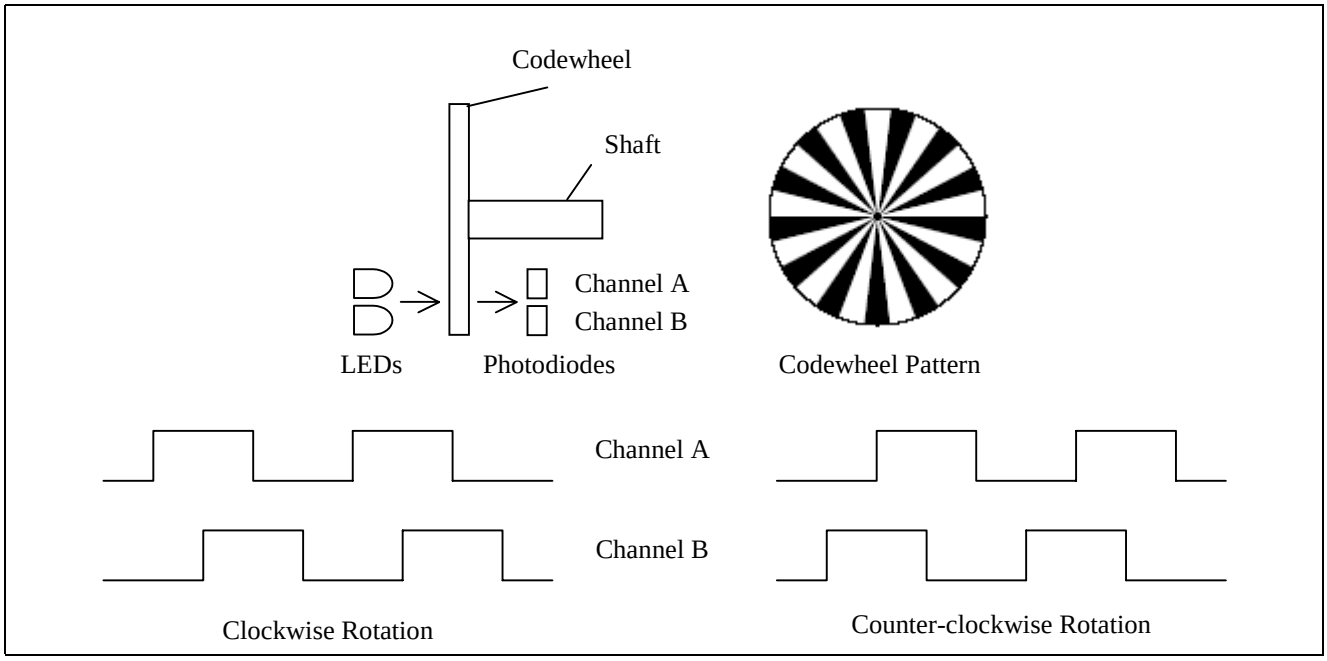


図 5　オプティカルエンコーダの動作

図 6 に示すように，モータやエンコーダ用の電源，および 2 チャネルの出力信号を，150nm のリボンケーブルと 10 ピンコネクタでインタフェースします。

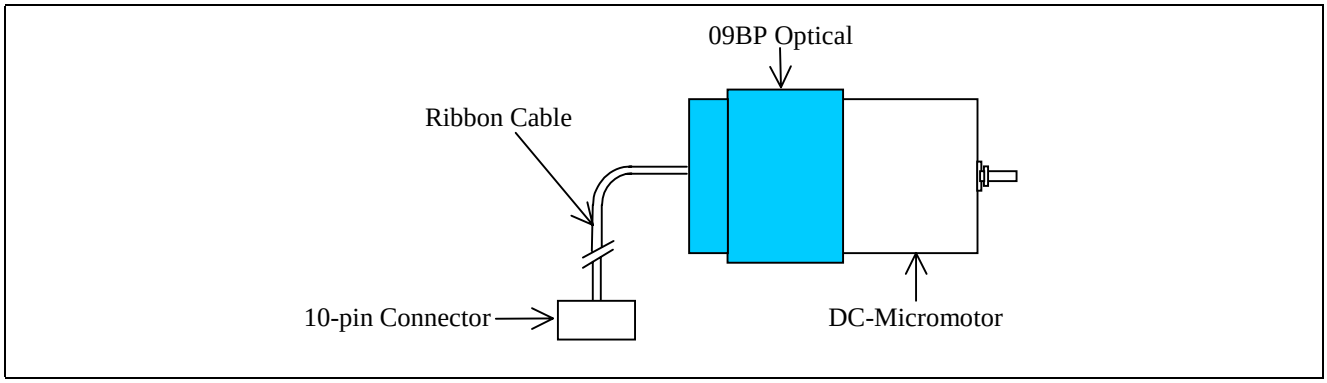


図 6　DC マイクロモータとオプティカルエンコーダ

エンコーダ (チャネル A と B) からの出力は，4x の直交デコーダ，バイナリの 16 ビットアップ/ダウンカウンタ, およびバスインタフェース機能で構成される Agilent 社製の直交デコーダ HCTL-2016 に接続します。シュミットトリガの CMOS 入力と入力ノイズフィルタを使用すると，ノイズの多い環境でも信頼性の高い動作が可能になります。

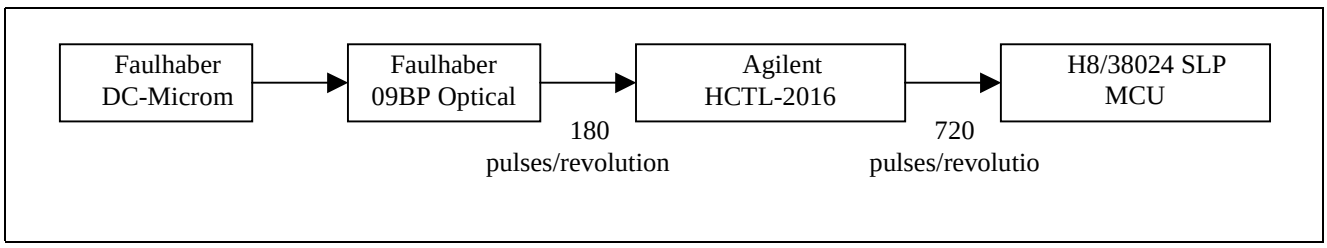


図 7　エンコーダの分解能

3.6 PWM モータドライバ

Allegro 社製 A3953SB のフルブリッジ PWM モータドライバであって，誘導負荷を双方向のパルス幅変調電流で制御するように設計されています。±1.3A の電流を連続して出力して，最大 50V の電圧で動作できます。ピーク負荷電流の上限は，ユーザが選択した入力電圧基準電圧と外付けのセンス抵抗によって決まります。固定したオフ時間のパルス期間は，ユーザが選択した外付け RC タイミングネットワークにより設定されます。内部回路保護はヒステリシス特性をもつサーマルシャットダウン，過渡電圧抑制ダイオード，および切り替え電流保護を含みます。

ENABLE_N 入力を Low レベルに保った状態で，PHASE 入力が，ソースとシンクドライバの対を適切に選択して負荷電流の極性を制御します。BRAKE 入力をロジック Low にすると，ブレーク機能がイネーブルになります。これによって，ENABLE 信号と PHASE 信号に関係なく，両方のソースドライバがオフになり，両方のシンクドライバがオンになります。このブレーク機能はブラシ DC モータを動的にブレーキをかけます。

PWM デューティ比 50%のとき，モータトルクは 0 です。0%または 100%のデューディ比は，それぞれ，逆方向，正方向の最大トルクを生成します。

4. ソフトウェアの実行

ソースプログラムは，適用が容易な C 言語で書かれており，HEW バージョン 2.2 (リリース 15) 用の無償の H8 Tiny/SLP ツールチェーン (バージョン 5.0.0) を使用してコンパイルします。

ソースプログラムの機能を以下に示します。

- モータの位置の測定
- 補正用アルゴリズムの計算
- プロファイルの生成
- RS-232C 通信

図 8 に，I/O ポート，SCI (38400bps，1 ストップビット，パリティディセーブル) ，タイマ，PWM モジュールを初期化する main 関数のフローチャートを示します。タイマ F のアウトプットコンペア割り込みサービスルーチンは 1.5ms ごとに発生し，エンコーダの値を読み出します。最新の軌道値と PID (proportional integration derivative) 値を計算して PWM 出力を設定します。所望の速度と実際の速度を PC に送信します。

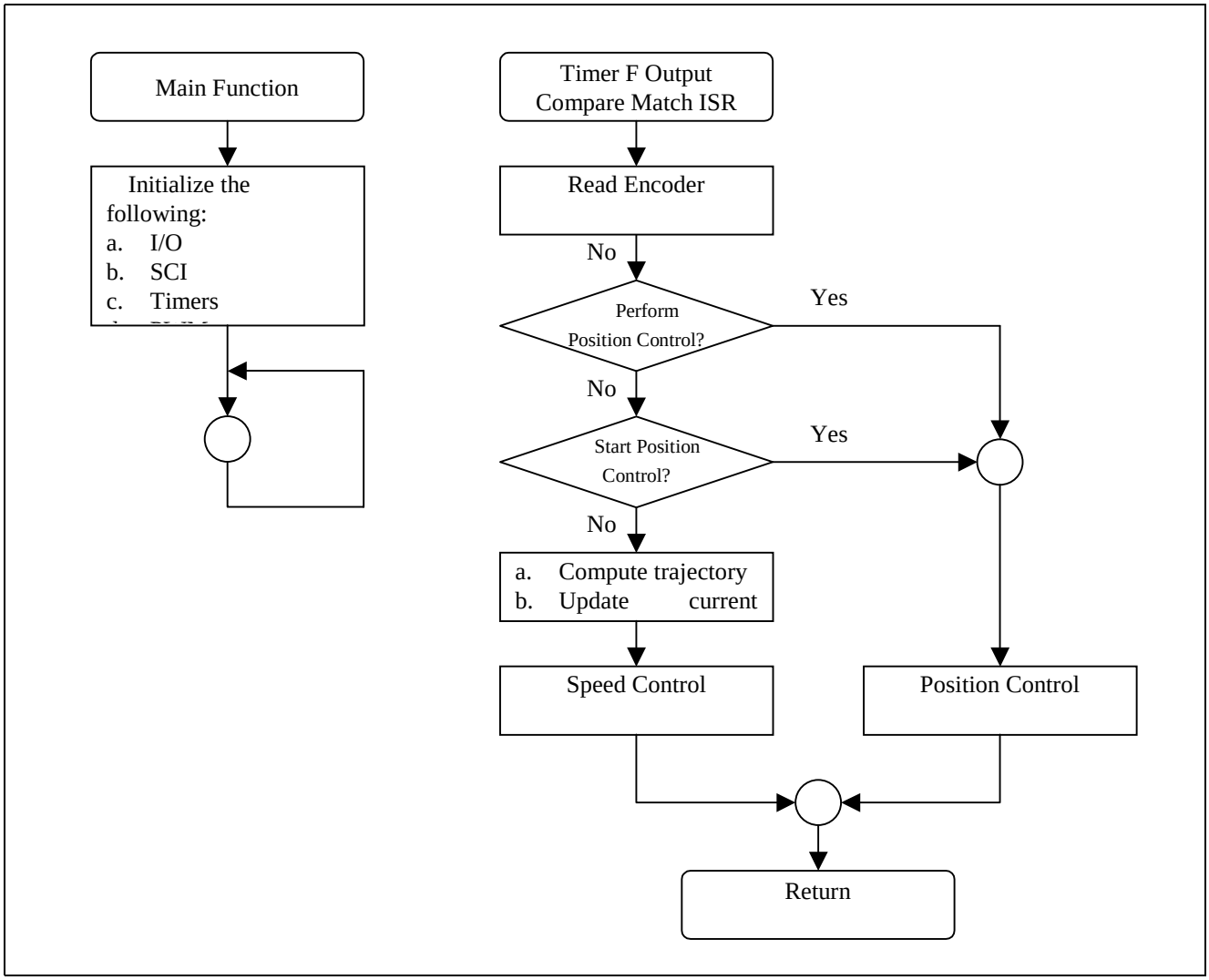


図 8 フローチャート

4.1 エンコーダのフィードバック

モータの軸に取り付けたオプティカルエンコーダから，速度と位置のフィードバックの情報を得ることができます。オプティカルエンコーダから出力された信号は HCTL-2016 4x 直交エンコーダで処理され, 16 ビットのアップ/ダウンカウンタに入力されます。16 ビットのアップ/ダウンカウンタの内容は 8 ビットバスインタフェースを介して連続した 2 バイトで読み出されます。まず，上位バイト (ビット 15-8，SEL = 0) が読み出され，次に下位バイト (ビット 7-0, SEL = 1) が読み出されます。正の値は時計回りの回転を表し，負の値は反時計回りを表します。したがって，1 回転当たり 4 x 180 = 720 パルスを生成します。

サーボのサンプリング周期は 1.5ms に設定しています。つまり, 16 ビットのアップ/ダウンカウンタは 1.5 ms ごとに読み出されます。読み出されたデータは計算され，値が PWM ドライバに出力されます。以下の表に示すように，回転速度と回転方向は，現在と前回の 16 ビットのアップ/ダウンカウンタの値を比較することによって，得られます。

エンコーダ数		速度 (数/サンプリング周期)	回転方向
現在	前回		
1000	900	100	**時計回り**
-1095	-1000	-95	**反時計回り**

4.2 PID コントローラ

システム設計者が直面する問題の 1 つは，モータの負荷が変動することです。老朽化，較正，環境などの他の要素も，モータの動作に影響を与えます。したがって，通常，フィードバック機構はモータの速度制御に用いられます。一般的に，PID (Proportional Integral Derivative) コントローラが使用されます。この例では，基準信号 r(n) が所望のモータ速度です。実際のモータ速度 (負荷によって変化する) はモータに取り付けられたオプティカルエンコーダで測定します。

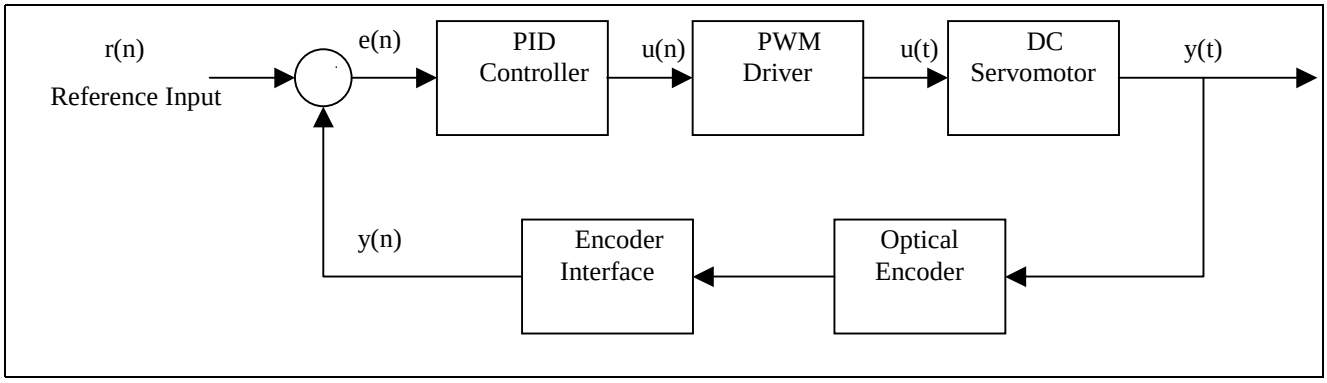


図 9 PID コントローラ

PID コントローラは以下の式によって求められます。

$$u(t) = K_p \cdot e(t) + K_i \cdot e_i T + K_d \cdot (de/dt)$$

このとき

- K_p は比例利得
- K_i は積分の利得
- K_d は微分利得
- T はサンプリング周期
- e(t) はエラー信号で，所望の速度と測定された速度との差に等しい

比例項は，現在の基準値と実際の測定速度の差だけを考えます。この比例項は，特定の制御されたモータの経験を利用して得られた利得係数 K_p で重みづけされます。積分項は，これまでに測定されたすべてのエラーの合計を取り込みます。積分項は，システムをダンピングして発振を取り除く働きをします。たとえば，測定速度が所望の速度よりやや遅い場合，比例項は印加する電力を増加してモータの速度をスピードアップします。最後のいくつかの測定により，モータ速度が速すぎる場合，積分項は電力を減少するか，それ以上の電力の供給をやめます。さらに，積分項は蓄積されたエラーを削除します。比例項によってもたらされたエネルギー量はエラーに比例するので (所望のモータ速度に対する実際のモータ速度の偏差) 所望のモータ速度に近いとき，その影響はきわめて少なくなります。PID ループを何度も繰り返している間，ごくわずかなエラーがある場合，積分項は蓄積され，そのエラーを是正するための制御量を変えます。

微分項はエラーが変化している変化の割合を正しく取り込み，設定時間を短縮させます。実際のモータ速度が目標とする速度よりはるかに遅ければ，速度を加速させます。モータの加速が一定であれば，エラーの変化速度も一定にします (微分項による)。このように微分項を増加させて特定の定数値にします。目標速度に近づくと，行き過ぎを防ぐためモータの加速度を減少させます。つまり，微分項を 0 に低減します。

対策は，積分項は各サンプリング周期で測定されたエラー項目の合計として，離散的に表されます。

$$e_i \cdot T = \sum_{j=0}^{i} \left(R_j - Y_j \right)$$

同様に，微分項も以下のように，離散的に概算できます。

$\frac{de}{dt} = \frac{\left(e_n - e_{n-1}\right)}{T}$ ここで，e_n と e_{n-1} は現在と前回の測定から計算されたエラー信号です。

以上のことから，PID を以下のように離散的に概算できます。

$$u(n) = K_p \cdot e(n) + K_i \cdot \sum e_i \cdot T + K_d \cdot \frac{\left[e(n) - e(n-1) \right]}{T}$$

4.3 プロファイルの生成

軌道の作成は，動き制御には必須です。この応用例では，直線的に区分された速度を作成します。ある特定の最大速度に到達するまで，速度を定数値で加速します。全体の 70%の回転量が完了するまで，この最大速度を維持します。次に，最小速度に到達するまで，速度を同じ定数値で減速します。モータが必要な回転数に達すると，位置制御が行われ，モータを一定の位置に保ちます (ブレーキ機能)。

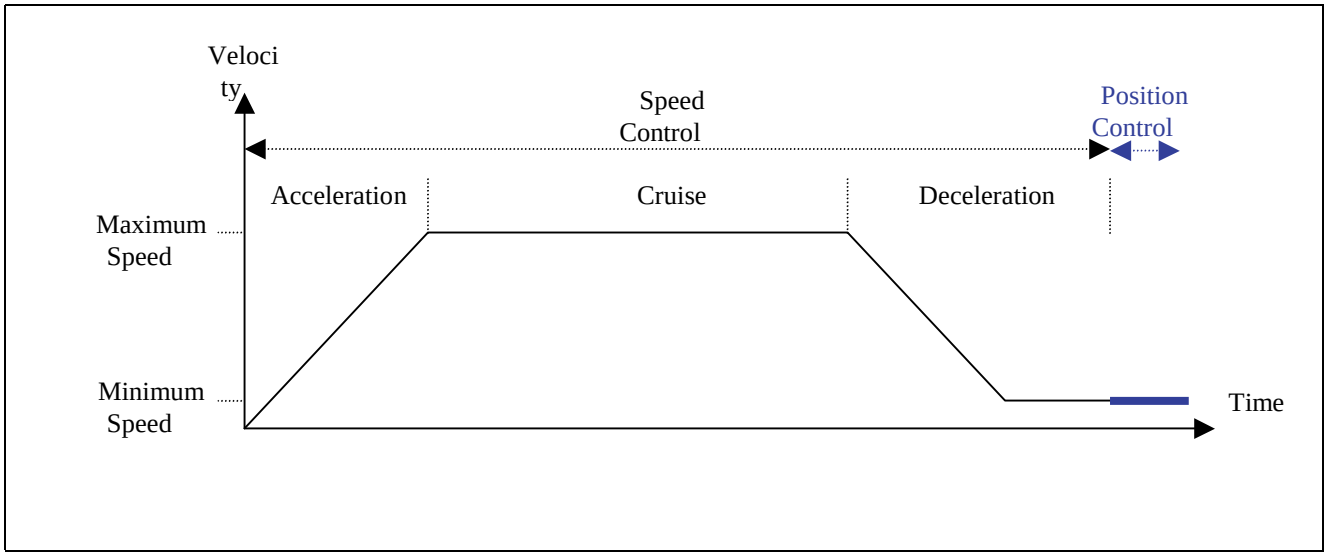


図 10　PID コントローラ

4.4 RS-232C 通信

RS-232C 通信の機能は，所望の速度と実際の速度 (回数，サンプリング周期) を PC に送信することです。Tera Term Pro のようなターミナルエミュレーションソフトを使用することで，これらの値は EXCEL に送られ，図 11 に示すような線が描かれます。さまざまな利得を変更して結果を描くことで，システムを微調整できます。

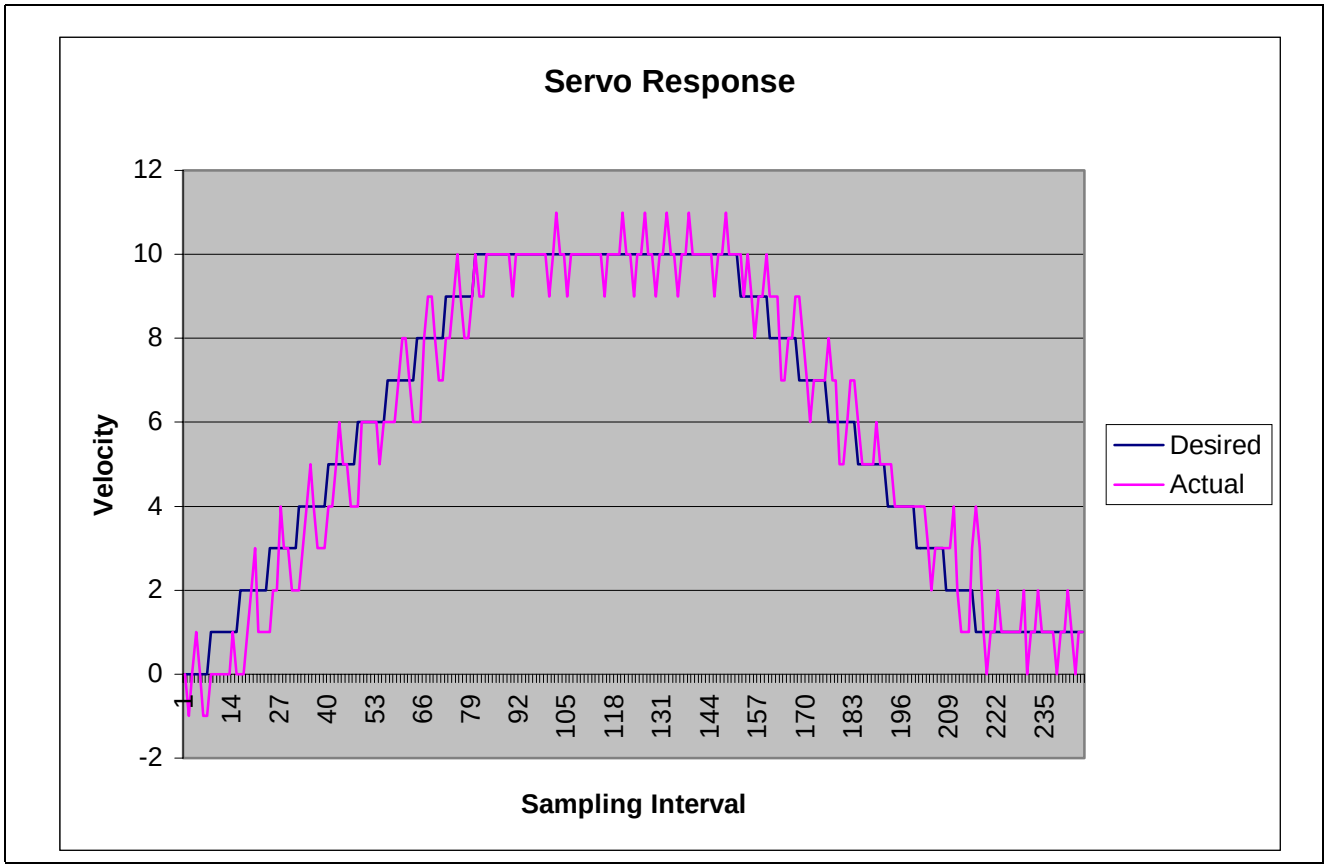


図 11　サーボ結果

4.5 PID コントローラの調整

与えられたシステム向けに PID コントローラを設計する際，所望の結果を得るために，以下の手順に従ってください。

- PID コントローラを比例モードだけに設定してください。つまり，K_i と K_d を 0 に設定してください。
- 比例利得 (K_p) を小さい値に設定して，出力を制限することによって，サーボモータシステムが初期段階から制御不能にならないようにしてください。モータをなにかの機械を駆動するために使用する場合などは特に，安全対策を行ってください。
- 基準値に小さな値を設定して，制御している速度やモータの位置など，可変物のレスポンスを観察してください。
- 迅速にシステムをオーバーシュートなく基準値になるように K_p を設定してください。
- 定常誤差をなくすために積分利得 (K_i) を含めてください。
- 過渡的なレスポンスを向上させるため，微分利得 (K_d) を含めてください。
- 所望のレスポンスが得られるまで利得 (K_p, K_i K_d) を調整してください。

4.6 ソースプログラム

ソースプログラムはおもに以下のファイルにあります。

- Dcmotor.c
  — main 関数を含む
  — I/O ポート，SCI，タイマ，PWM チャネルの初期化を行う
  — 軌道の生成，エンコーダカウントの表示，PID 値，速度，位置制御の計算を行う
- int_prg.c
  — タイマ F アウトプットコンペア割り込みサービスルーチンを含む
  — 位置カウントを更新する

たとえば，モータを反時計回りに 1 回転させるためには，Dcmotor.c のなかで定義した以下の定数を設定します。

パラメータ	値	備考
distance_reference	-720	1° → 2 カウント 1 回転→ 360° → (360 * 2) = 720 カウント 反時計回りは負
maximum_speed_reference	-7	加速/定常値は最大基準速度 反時計回りは負
minimum_speed_reference	-1	減速中は最小基準速度 反時計回りは負
speed_adjustment	-1	基準速度用のステップ調整 反時計回りは負
proportional_gain	6	PID コントローラ用比例利得
integral_gain	2	PID コントローラ用積分利得
derivative_gain	4	PID コントローラ用微分利得

```
/************************************************************************/
/*                                                                      */
/*  FILE       :Dcmotor.c                                               */
/*  DATE       :Mon, Jan 12, 2004                                       */
/*  DESCRIPTION :Main Program                                           */
/*  CPU TYPE   :H8/38024                                                */
/*                                                                      */
/*  This file is generated by Renesas Project Generator (Ver.2.1).      */
/*                                                                      */
/************************************************************************/

#ifdef __cplusplus
extern "C" {
#endif
void abort(void);
#ifdef __cplusplus
}
#endif

#include "iodefine.h"
#include <machine.h>

//-----------------------------------------------------------------------
//Constant Declarations
//-----------------------------------------------------------------------

//Constants for Address Decoder
#define   first_digit                 0xC0
#define   second_digit          0xC1
#define   third_digit                 0xC2
#define   fourth_digit          0xC3

#define   enc_count_high        0x44
#define   enc_count_low         0x45

#define   reset_2016            0xC8

#define   de_select             0x3F

//Constants for Motor Control
#define   distance_reference          -720          //1 degree = 2 encoder counts
#define   maximum_speed_reference     -7            //negative for counter-clockwise
#define   minimum_speed_reference     -1            //negative for counter-clockwise
#define   speed_adjustment            -1            //negative for counter-clockwise
#define   proportional_gain           6
#define   integral_gain         2
#define   derivative_gain       4

#define max_pos_con_val  256

//Others
#define   address_bus                 P_IO.PDR7.BYTE      //Address Bus
#define   data_bus                    P_IO.PDR6.BYTE      //Data Bus
//-----------------------------------------------------------------------
//Function Prototypes
//-----------------------------------------------------------------------
```

```
void init_sci(void);
void char_put(char);
void PutStr(char *);

void init_port(void);

void display_char(unsigned char, unsigned char);

void init_timers(void);

void display_word(unsigned int);

void reset_encoder(void);
int  read_encoder(void);

void init_pwm(void);

void serial_transmit(unsigned int);

void speed_control(void);

void speed_profile(void);

void position_control(void);

//---------------------------------------------------------------------

//16-bit number: +ve for clockwise, -ve for counter-clockwise
int encoder_count, current_encoder_count, previous_encoder_count;

//16-bit number: reference speed: +ve for clockwise, -ve for counter-clockwise
int speed_reference;

int current_speed;
int current_speed_error, previous_speed_error, sum_speed_error,
       diff_speed_error;
int speed_control_output;

int position_reference, position_reference_a;
int current_position_error, previous_position_error, sum_position_error,
       diff_position_error;
int position_control_output;

int speed_increment;    //+ve for clockwise, -ve for counter-clockwise
int max_speed_reference;//+ve for clockwise, -ve for counter-clockwise
int min_speed_reference;//+ve for clockwise, -ve for counter-clockwise

int temp_encoder_count, temp_position_reference;

unsigned int        position_control_count;

static unsigned char reference_step = 0;

int                 kp_gain, ki_gain, kd_gain;

unsigned char           start_position_control;
```

```
//---------------------------------------------------------------------

void main(void)
{
    //-----------------------------------------------------------------

    init_port();

    init_sci();

    init_timers();

    init_pwm();

    reset_encoder();

    //-----------------------------------------------------------------

    speed_reference = 0;

    speed_increment = speed_adjustment;
    max_speed_reference = maximum_speed_reference;
    min_speed_reference = minimum_speed_reference;

    previous_speed_error = 0;
    sum_speed_error = 0;
    diff_speed_error = 0;

    kp_gain = proportional_gain;
    ki_gain = integral_gain;
    kd_gain = derivative_gain;

    position_reference = distance_reference;

    position_reference_a = ((position_reference * 7)/ 10);
    if (position_reference < 0)
    {
        position_reference_a = 0 - position_reference_a;
    }

    current_position_error = 0;
    previous_position_error = 0;
    sum_position_error = 0;

    start_position_control = 0;
    position_control_count = 0;

    //-----------------------------------------------------------------

    P_SYSCR.IENR2.BIT.IENTFH = 1;            //Enable Timer FH interrupt requests

    //if Timer FH interrupt request flag (IRRTFH) is set, clear to 0
    if (P_SYSCR.IRR2.BIT.IRRTFH == 1)
        P_SYSCR.IRR2.BIT.IRRTFH = 0;

    set_imask_ccr(0);                        //Clear IMASK
```

```
    //---------------------------------------------------------------------

    while (1)
    {
    }
}

//---------------------------------------------------------------------

void abort(void)
{

}

//---------------------------------------------------------------------

/*
    init_pwm()
*/

void init_pwm(void)
{
    P_PWM1.PWCR1.BYTE   = 0xFD;              //Conversion period is 1024/phi

    P_PWM1.PWDRL1.BYTE  = 0x00;        //Write to lower 8 bits

    P_PWM1.PWDRU1.BYTE  = 0xFE;        //Write to upper 2 bits

    P_IO.PDR9.BYTE &= 0xF7;                  //Port 9[3] : Clear ENABLE_N
}

//---------------------------------------------------------------------

/*
    reset_encoder(): Reset enocder counts
*/

void reset_encoder(void)
{
    P_IO.PCR6.BYTE  = 0xFF;                  //Set Port 6[7..0] as output

    address_bus &= de_select;

    address_bus = reset_2016;

    address_bus &= de_select;
}

//---------------------------------------------------------------------

/*
    read_encoder(): Read encoder counts
*/

int read_encoder(void)
{
    int          counts;
```

```
    unsigned char low_byte, high_byte;

    P_IO.PCR6.BYTE  = 0x00;                  //Set Port 6[7..0] as input

    address_bus &= de_select;

    address_bus = enc_count_high;            //Address

    high_byte = data_bus;              //Data

    address_bus &= de_select;

    address_bus = enc_count_low;       //Address

    low_byte = data_bus;         //Data

    address_bus &= de_select;

    counts = (int)((high_byte << 8) | low_byte);

    return(counts);
}

//---------------------------------------------------------------------

/*
    init_timers() : Set up Timer F
*/

void init_timers(void)
{
    //Timer Control Register F
    //TOLL = '1': Initial output for TMOFH is high
    //CKSH2 = '0', CKSH1 = '0', CKSH0 = '0': 16-bit mode, counting on TCFL overflow signal
    //Clock input to TCFL at phi/4 i.e., (10MHz/2/4 = 1.25MHz)
    P_TMRF.TCRF.BYTE = 0x86;

    //Timer Control/Status Register F
    //CCLRH = '1': in 16-bit mode, TCF clearing by compare match is enabled
    P_TMRF.TCSRF.BYTE = 0x10;

    //1.5ms -> 0x753
    P_TMRF.OCRF.BYTE.H = 0x07;               //Output Compare Register FH
    P_TMRF.OCRF.BYTE.L = 0x53;               //Output Compare Register FL
}

//---------------------------------------------------------------------

/*
    init_port() : Set up the I/O ports

    a.  Port 6[7..0] -> Data[7..0]
    b.  Port 7[7..0] -> Address[7..0]
        Note that Port 7_7 also functions as the WRITE/READ_N signal
*/

void init_port(void)
```

```
{
    P_IO.PMR3.BYTE   = 0x04;                          //P32 functions as TMOFH

    P_LCD.LPCR.BYTE = 0x00;                           //SEG[32..1] as I/O Port

    P_IO.PCR6.BYTE  = 0xFF;

    P_IO.PCR7.BYTE  = 0xFF;

    P_IO.PDR6.BYTE  = 0xFF;

    P_IO.PDR7.BYTE  = 0xFF;

    P_IO.PMR9.BYTE   = 0xF1;                          //PWM1
}

//---------------------------------------------------------------------

/*
    display_char()

    a.  Port 6[7..0] -> Data[7..0]
    b.  Port 7[7..0] -> Address[7..0]
        Note that Port 7_7 also functions as the WRITE/READ_N signal
*/

void display_char(unsigned char digit_position, unsigned char digit_info)
{
    P_IO.PCR6.BYTE  = 0xFF;                   //Set Port 6[7..0] as output

    address_bus &= de_select;

    data_bus = digit_info;                    //Data

    address_bus = digit_position;             //Address

    address_bus &= de_select;
}

//---------------------------------------------------------------------

/*
    display_word()
*/

void display_word(unsigned int display_data)
{
    unsigned char position, digit_info, digit_position;

    P_IO.PCR6.BYTE  = 0xFF;                           //Set Port 6[7..0] as output

    for (position = 4 ; position != 0 ; position--)
    {
        switch (position)
        {
            case 1:
                digit_position = first_digit;
```

```
            digit_info = (unsigned char)(display_data & 0x000F);
            break;

        case 2:
            digit_position = second_digit;
            digit_info = (unsigned char)((display_data & 0x00F0) >> 4);
            break;

        case 3:
            digit_position = third_digit;
            digit_info = (unsigned char)((display_data & 0x0F00) >> 8);
            break;

        default:
            digit_position = fourth_digit;
            digit_info = (unsigned char)((display_data & 0xF000) >> 12);
            break;
    }

    if ((digit_info >= 0) && (digit_info <= 9))
        digit_info += 0x30;
    else
    {
        if ((digit_info >= 0xA) && (digit_info <= 0xF))
        {
            digit_info -= 0xA;
            digit_info += 0x41;
        }
    }

    address_bus &= de_select;

    data_bus = digit_info;                //Data

    address_bus = digit_position;   //Address

    address_bus &= de_select;
  }
}

//---------------------------------------------------------------------

/*
   serial_transmit()
*/

void serial_transmit(unsigned int display_data)
{
   unsigned char position, digit_info;

   for (position = 4 ; position != 0 ; position--)
   {
      switch (position)
      {
         case 1:
            digit_info = (unsigned char)(display_data & 0x000F);
            break;
```

```
            case 2:
               digit_info = (unsigned char)((display_data & 0x00F0) >> 4);
               break;

            case 3:
               digit_info = (unsigned char)((display_data & 0x0F00) >> 8);
               break;

            default:
               digit_info = (unsigned char)((display_data & 0xF000) >> 12);
               break;
         }

         if ((digit_info >= 0) && (digit_info <= 9))
            digit_info += 0x30;
         else
         {
            if ((digit_info >= 0xA) && (digit_info <= 0xF))
            {
               digit_info -= 0xA;
               digit_info += 0x41;
            }
         }

         char_put(digit_info);
      }
   }


   //---------------------------------------------------------------------

   /*
      init_sci() : Sets up the Serial Communication Interface for debugging purposes.
   */

   void init_sci(void)
   {
      //SCR3 : |TIE|RIE|TE|RE|MPIE|TEIE|CKE1|CKE0|
      //TIE : Transmit interrupt enable
      //RIE : Receive interrupt enable
      //TE  : Transmit enable
      //RE  : Receive enable
      //MPIE : Multiprocessor interrupt enable
      //TEIE : Transmit end interrupt enable
      //CKE1 : Clock enable 1
      //CKE0 : Clock enable 0

      //CKE1 = CKE0 = 0
      //asynchronous mode, internal clock source, SCK32 functions as I/O port
      P_SCI3.SCR3.BYTE &= 0x00;    //clear TE & RE

      //SMR : |COM|CHR|PE|PM|STOP|MP|CKS1|CKS0| : |0|0|0|0|0|0|0|0|
      //COM : Communication Mode  : 0 : asynchronous mode
      //CHR : Character Length    : 0 : character length = 8 bits
      //PE  : Parity Enable       : 0 : parity bit addition and checking disabled
      //PM  : Parity Mode         : 0 : even parity (no effect since parity is already disabled)
```

```
    //STOP: Stop Bit Length    : 0 : 1 stop bit
    //MP  : Multiprocessor Mode : 0 : multiprocessor communication function disabled
    //|CKS1|CKS0| : Clock Select: |0|0| : clock source for baud rate generator = clk
    P_SCI3.SMR.BYTE = 0x00;

    //For clk = 10MHz, bit rate = 38400 bps, n = 0, N = 3
    P_SCI3.BRR = 3;

    //minimum of 1-bit delay = 417ns
    nop();
    nop();
    nop();

    //SPCR : |---|---|SPC32|---|SCINV3|SCINV2|---|---| : |1|1|1|0|0|0|0|0|
    //SPC32 = 1 : P42 functions as TXD32 output pin
    //need to set TE bit in SCR3 after setting this bit to 1
    //SCINV3 = 0 : TXD32 output data is not inverted
    //SCINV2= 0 : RXD32 input data is not inverted
    //Bits 7 and 6 are reserved and always read as 1
    //Bits 4, 1 and 0 are reserved and only 0 can be written to these bits
    P_SCI3.SPCR.BYTE = 0xE0;

    P_SCI3.SCR3.BYTE |= 0x30;   //Set TE & RE
}

//---------------------------------------------------------------------

/*
    char_put() : Transmits a character to the PC for debugging purposes.
*/

void char_put(char OutputChar)           //Serial Port
{
    //SSR  : |TDRE|RDRF|OER|FER|PER|TEND|MPBR|MPBT|
    //TDRE : transmit data register empty
    //RDRF : receive data register full
    //OER  : overrun error
    //FER  : framing error
    //PER  : parity error
    //TEND : transmit end
    //MPBR : Multiprocessor bit receive
    //MPBT : Multiprocessor bit transfer

    while ((P_SCI3.SSR.BIT.TDRE) == 0);      //Wait for TDRE = 1

    P_SCI3.TDR = OutputChar;

    while ((P_SCI3.SSR.BIT.TEND) == 0);      //Wait for TEND = 1

    P_SCI3.SSR.BIT.TEND = 0;
}

//---------------------------------------------------------------------

/*
    PutStr() : Transmits a string of characters to the PC for debugging purposes.
```

```
*/
void PutStr(char *str)
{
    while (*str != 0)
    {
        char_put(*str++);
    }
}

//---------------------------------------------------------------------

/*
    speed_control()

    1. current_speed = current_encoder_count - previous_encoder_count

    2. current_speed_error = speed_reference - current_speed

    3. output = kp_gain * speed_error + ki_gain * sum(speed_error) + kd_gain *
(current_speed_error - previous_speed_error);

    range-limited from -512 to 511
*/

void speed_control(void)
{
    //-------------------------------------------------------------

    previous_encoder_count = current_encoder_count;
    current_encoder_count = encoder_count;

    current_speed = current_encoder_count - previous_encoder_count;

    //-------------------------------------------------------------

    serial_transmit(speed_reference);

    //-------------------------------------------------------------

    previous_speed_error = current_speed_error;

    //-------------------------------------------------------------

    current_speed_error = speed_reference - current_speed;

    PutStr("/");
    serial_transmit(current_speed);
    PutStr("¥r¥n");

    //-------------------------------------------------------------

    sum_speed_error += (current_speed_error);

    diff_speed_error = (current_speed_error - previous_speed_error);

    //-------------------------------------------------------------
```

```
    speed_control_output = (int)((kp_gain * current_speed_error) + (sum_speed_error /
ki_gain) + (diff_speed_error * kd_gain));

    //-------------------------------------------------------------
    //limit speed_control_output from -511 to +511

    if (speed_control_output > 511)
        speed_control_output = 511;
    else
    {
        if (speed_control_output < -511)
            speed_control_output = -511;
    }

    //introduce offset of 512
    speed_control_output += 512;

    P_PWM1.PWDRL1.BYTE  = (unsigned char)(speed_control_output & 0x00FF);
    //Write to lower 8 bits

    P_PWM1.PWDRU1.BYTE  = (unsigned char)((speed_control_output & 0xFF00)>> 8);//Write
to upper 2 bits

    //-------------------------------------------------------------
}

//---------------------------------------------------------------------------

/*
    speed_profile()  - generates the speed profile
*/

void speed_profile(void)
{
    int temp_encoder_count;

    reference_step++;

    temp_encoder_count = encoder_count;

    if (position_reference < 0)
        temp_encoder_count = 0 - encoder_count;

    if (temp_encoder_count > position_reference_a)
    {
        //decelerate
        if (reference_step > 7)
        {
            reference_step = 0;

            //For clockwise rotation, speed_increment is +ve
            //For counter-clockwise rotation, speed_increment is -ve

            //For example
            //a. clockwise, speed_increment = 1, speed_reference = 5 -> speed_reference = 5
- 1 = 4
```

```
            //b. counter-clockwise, speed_increment = -1, speed_reference = -5 ->
speed_reference = -5 - (-1) = -4

            if (speed_reference < 0)    //Counter-clockwise
            {
                speed_reference -= speed_increment;

                if (speed_reference > min_speed_reference)
                    speed_reference = min_speed_reference;
            }
            else                        //Clockwise
            {
                speed_reference -= speed_increment;

                if (speed_reference < min_speed_reference)
                    speed_reference = min_speed_reference;
            }
        }
    }
    else
    {
        //accelerate and cruise
        if (reference_step > 7)
        {
            reference_step = 0;

            //For clockwise rotation, speed_increment is +ve
            //For counter-clockwise rotation, speed_increment is -ve

            //For example
            //a. clockwise, speed_increment = 1, speed_reference = 5 -> speed_reference = 5
+ 1 = 6
            //b. counter-clockwise, speed_increment = -1, speed_reference = -5 ->
speed_reference = -5 + (-1) = -6

            speed_reference += speed_increment;

            if (speed_reference < 0)    //Counter-clockwise
            {
                if (speed_reference < max_speed_reference)
                    speed_reference = max_speed_reference;
            }
            else                        //Clockwise
            {
                if (speed_reference > max_speed_reference)
                    speed_reference = max_speed_reference;
            }
        }
    }
}

//---------------------------------------------------------------------

/*
    position_control()

    1. position_error = reference_encoder_count - current_encoder_count
```

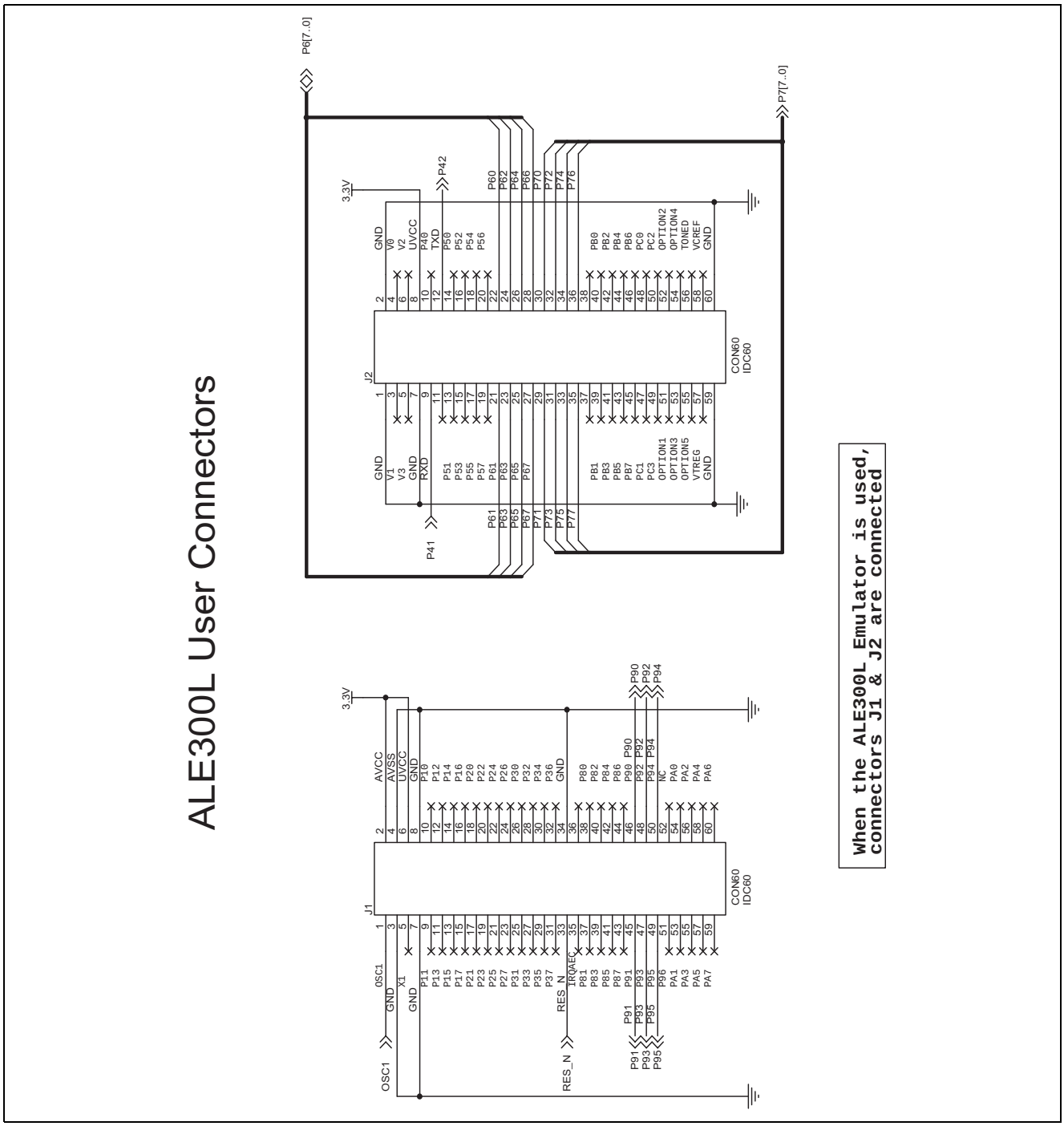
```
    2. output = kp_gain * current_position_error
               + sum(position_error) / ki_gain
               + kd_gain * (position_error - previous_position_error);

    range-limited from -512 to 511
*/

void position_control(void)
{
    kp_gain = 10;
    ki_gain = 2;
    kd_gain = 3;

    position_control_count++;

    //-------------------------------------------------------------

    previous_position_error = current_position_error;

    //-------------------------------------------------------------

    current_position_error = position_reference - encoder_count;

    //-------------------------------------------------------------

    sum_position_error += (current_position_error);

    diff_position_error = (current_position_error - previous_position_error);

    //-------------------------------------------------------------

    position_control_output = (int)((kp_gain * current_position_error) + (kd_gain *
diff_position_error));

    //-------------------------------------------------------------
    //limit position_control_output from -128 to +128

    if (position_control_output > max_pos_con_val)
        position_control_output = max_pos_con_val;
    else
    {
        if (position_control_output < -max_pos_con_val)
            position_control_output = -max_pos_con_val;
    }

    //introduce offset of 512
    position_control_output += 512;

    P_PWM1.PWDRL1.BYTE  = (unsigned char)(position_control_output & 0x00FF);
    //Write to lower 8 bits

    P_PWM1.PWDRU1.BYTE  = (unsigned char)((position_control_output & 0xFF00)>> 8);
    //Write to upper 2 bits

    //-------------------------------------------------------------
}
```

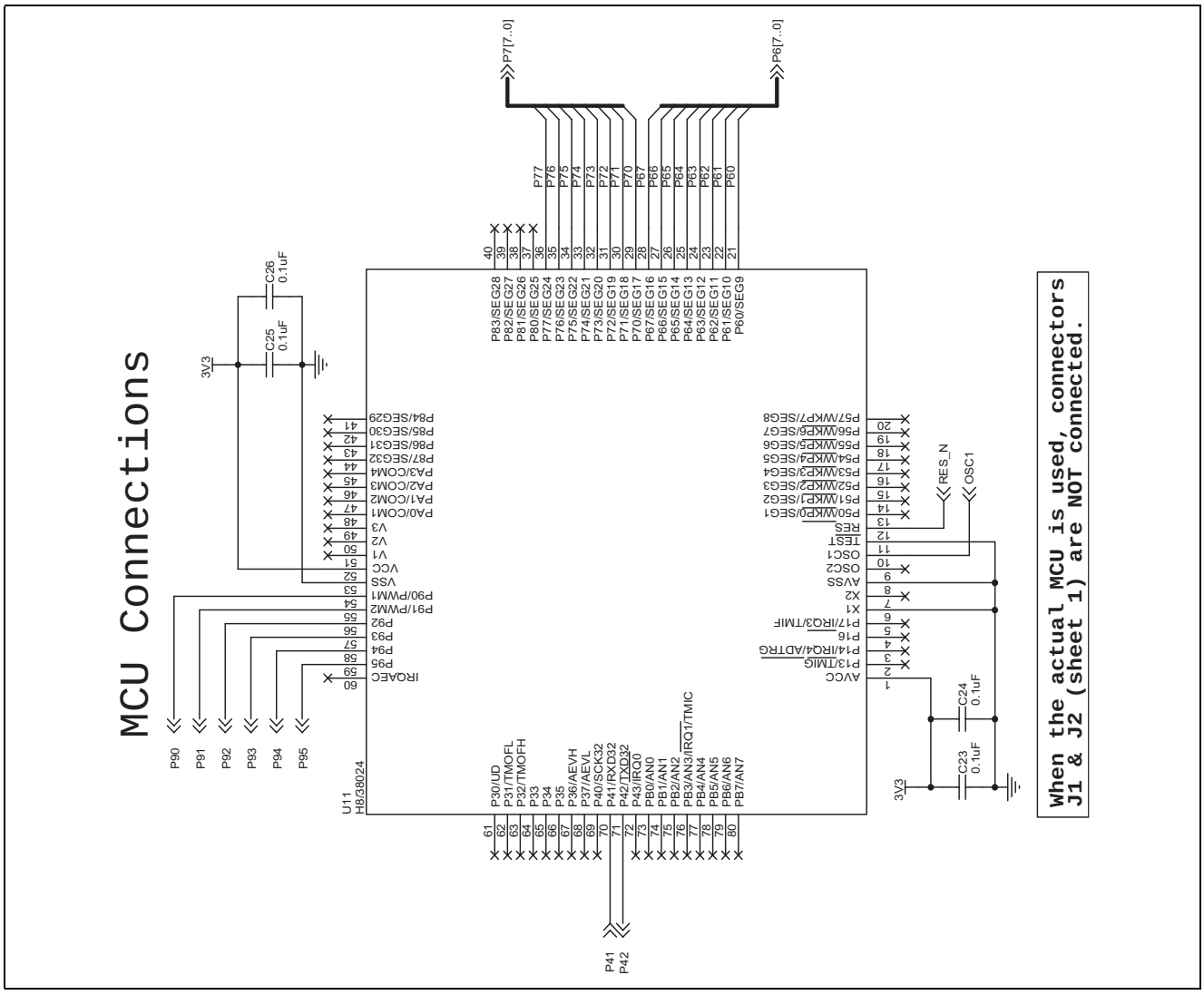
```
//---------------------------------------------------------------------
/*********************************************************************/
/*                                                                   */
/*  FILE       :intprg.c                                             */
/*  DATE       :Mon, Jan 12, 2004                                    */
/*  DESCRIPTION :Interrupt Program                                   */
/*  CPU TYPE   :H8/38024                                             */
/*                                                                   */
/*  This file is generated by Renesas Project Generator (Ver.2.1).   */
/*                                                                   */
/*********************************************************************/

#include   <machine.h>
#include   "iodefine.h"

//---------------------------------------------------------------------
extern void display_word(unsigned int);
extern int  read_encoder(void);
extern void serial_transmit(unsigned int);
extern void speed_control(void);

extern void speed_profile(void);

extern void position_control(void);

extern int             encoder_count, position_reference;
extern int             temp_encoder_count, temp_position_reference;
extern unsigned int position_control_count;
extern unsigned char   start_position_control;

//---------------------------------------------------------------------

#pragma section IntPRG

//  vector 1 Reserved

//  vector 2 Reserved

//  vector 3 Reserved

//  vector 4 IRQ0
__interrupt(vect=4) void INT_IRQ0(void) {/* sleep(); */}

//  vector 5 IRQ1
__interrupt(vect=5) void INT_IRQ1(void) {/* sleep(); */}

//  vector 6 IRQAEC
__interrupt(vect=6) void INT_IRQAEC(void) {/* sleep(); */}

//  vector 7 Reserved

//  vector 8 Reserved


//  vector 9 WKP0_7
__interrupt(vect=9) void INT_WKP0_7(void) {/* sleep(); */}
```

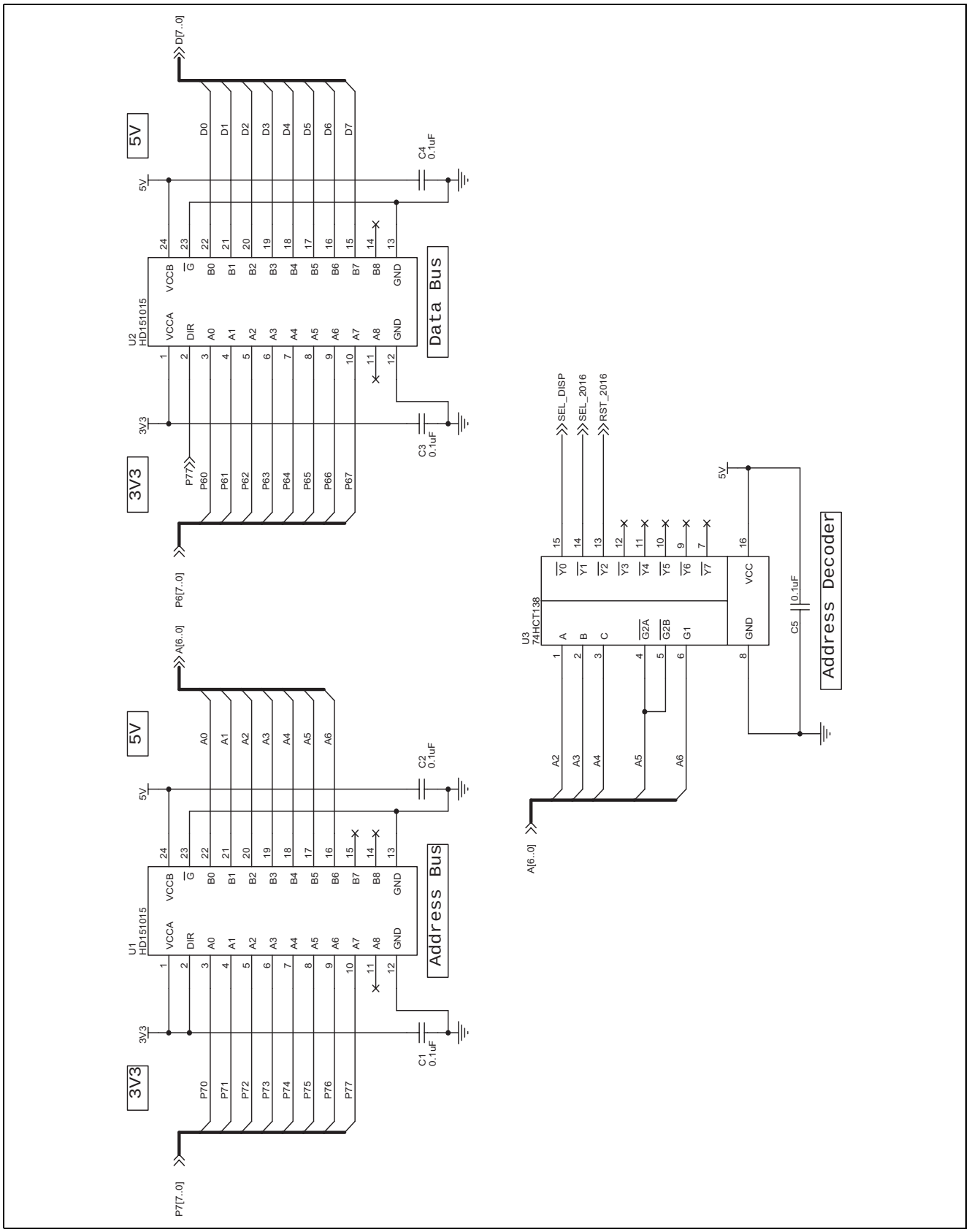
```
//  vector 10 Reserved

//  vector 11 Timer A Overflow
__interrupt(vect=11) void INT_TimerA(void) {/* sleep(); */}

//  vector 12 Counter Overflow
__interrupt(vect=12) void INT_Counter(void) {/* sleep(); */}

//  vector 13 Reserved

//  vector 14 Timer FL Overflow
__interrupt(vect=14) void INT_TimerFL(void) {/* sleep(); */}

//  vector 15 Timer FH Overflow

__interrupt(vect=15) void INT_TimerFH(void)
{
    //if Timer FH interrupt request flag (IRRTFH) is set, clear to 0
    if (P_SYSCR.IRR2.BIT.IRRTFH == 1)
        P_SYSCR.IRR2.BIT.IRRTFH = 0;

    if (P_TMRF.TCSRF.BIT.CMFH == 1)
    {
        P_TMRF.TCSRF.BIT.CMFH = 0;

        encoder_count = read_encoder();

        if (start_position_control)
        {
            if (position_control_count < 300)
                position_control();
            else
                P_IO.PDR9.BYTE &= 0xDF;  //Apply BRAKE_N -> P95
        }
        else
        {
            if (position_reference < 0)
            {
                temp_encoder_count = 0 - encoder_count;
                temp_position_reference = 0 - position_reference;
            }
            else
            {
                temp_encoder_count = encoder_count;
                temp_position_reference = position_reference;
            }

            if (temp_encoder_count > temp_position_reference)
            {
                start_position_control = 1;
                position_control();
            }
            else
            {
                P_IO.PDR9.BYTE ^= 0x02;  //Toggle P91
```

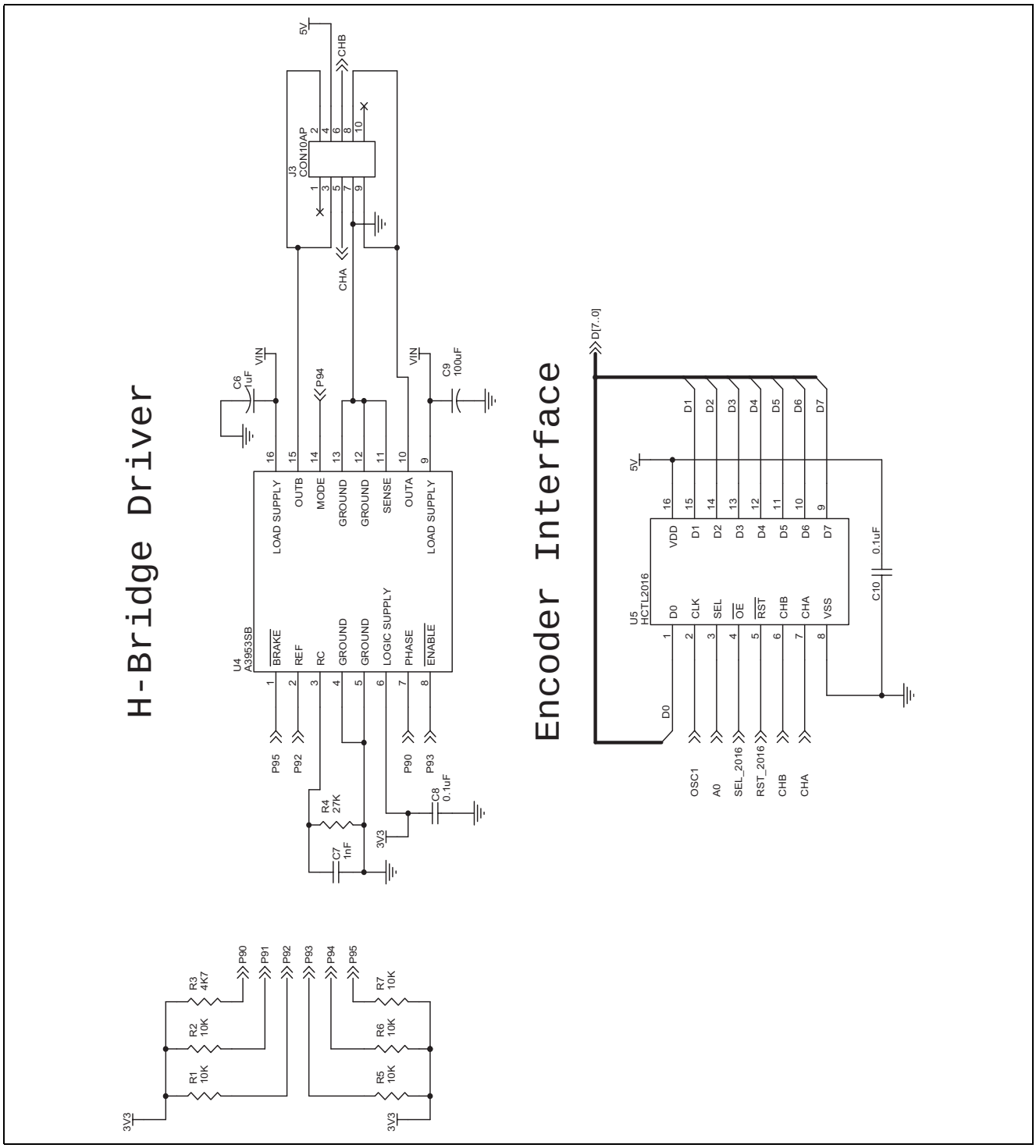
```
                speed_profile();
                speed_control();
                P_IO.PDR9.BYTE ^= 0x02;  //Toggle P91
            }
        }

        display_word(encoder_count);
    }

    /* sleep(); */
}

//  vector 16 Reserved

//  vector 17 Reserved

//  vector 18 SCI3
__interrupt(vect=18) void INT_SCI3(void) {/* sleep(); */}

//  vector 19 ADI
__interrupt(vect=19) void INT_ADI(void) {/* sleep(); */}

//  vector 20 Direct Transition
__interrupt(vect=20) void INT_Direct_Transition(void) {/* sleep(); */}
```

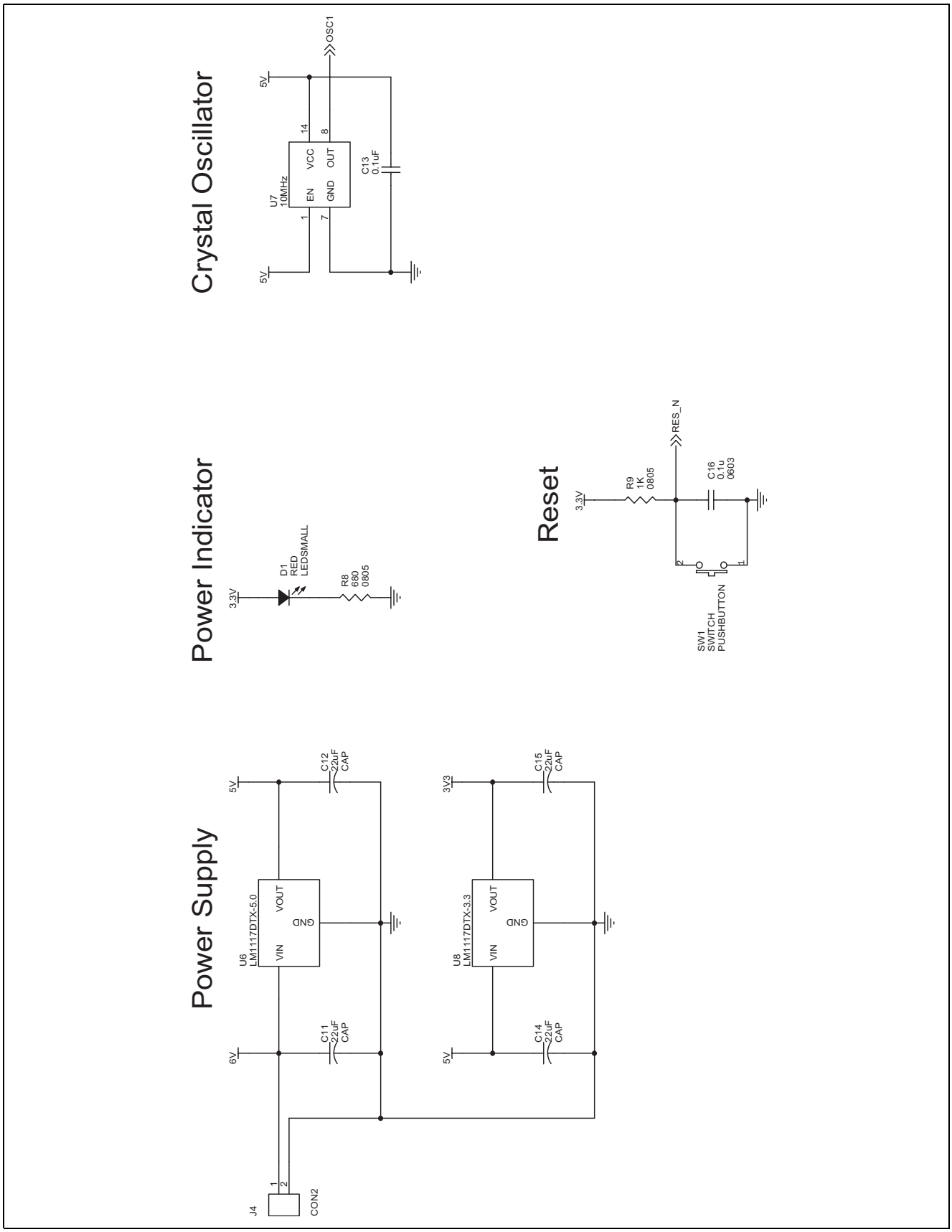
5. ハードウェアの概要

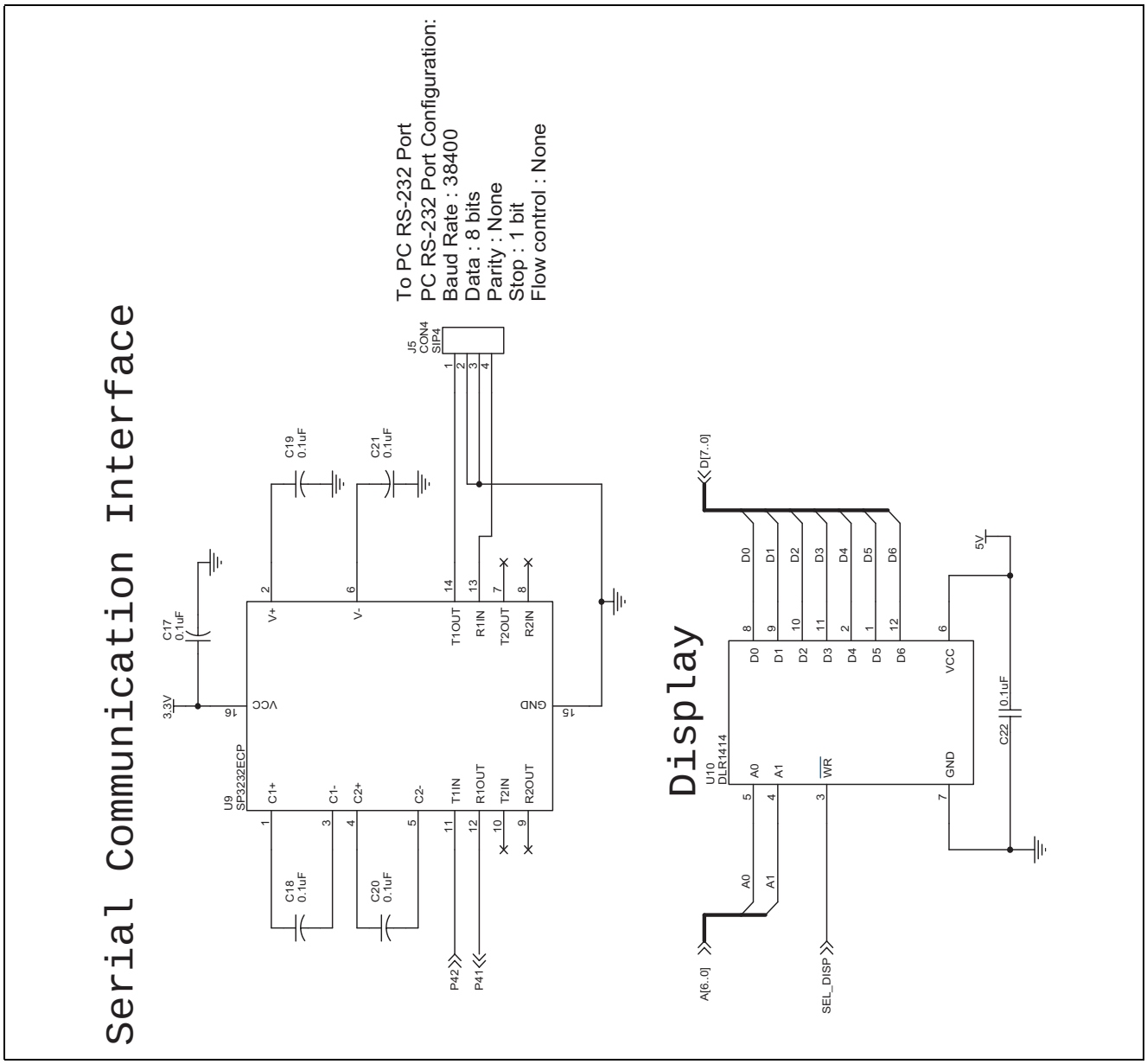

ALE300L User Connectors
When the ALE300L Emulator is used, connectors J1 & J2 are connected
J1
J2
CON60
IDC60
3.3V
GND
OSC1
RES_N
P6[7..0]
P7[7..0]
P41
P42


MCU Connections
When the actual MCU is used, connectors J1 & J2 (sheet 1) are NOT connected.
U11
H8/38024
C25 0.1uF
C26 0.1uF
C23 0.1uF
C24 0.1uF
3V3
RES_N
OSC1
P7[7..0]
P6[7..0]
P90
P91
P92
P93
P94
P95
P41
P42


Address Bus
Data Bus
Address Decoder
U1 HD151015
U2 HD151015
U3 74HCT138
5V
3V3
C1 0.1uF
C2 0.1uF
C3 0.1uF
C4 0.1uF
C5 0.1uF
A[6..0]
P7[7..0]
P6[7..0]
D[7..0]
SEL_DISP
SEL_2016
RST_2016


H-Bridge Driver
Encoder Interface
5V
CHB
CHA
J3
CON10AP
VIN
C6
1uF
C9
100uF
P94
U4
A3953SB
LOAD SUPPLY
OUTB
MODE
GROUND
SENSE
OUTA
BRAKE
REF
RC
LOGIC SUPPLY
PHASE
ENABLE
P95
P92
P90
P93
R4
27K
C7
1nF
C8
0.1uF
3V3
R1
10K
R2
10K
R3
4K7
R5
10K
R6
10K
R7
10K
P90
P91
P92
P93
P94
P95
D[7..0]
U5
HCTL2016
VDD
D0
CLK
SEL
OE
RST
VSS
D1
D2
D3
D4
D5
D6
D7
C10
0.1uF
OSC1
A0
SEL_2016
RST_2016


Crystal Oscillator
OSC1
5V
14
8
VCC
OUT
GND
EN
U7
10MHz
C13
0.1uF
1
7
5V
Power Indicator
3.3V
D1
RED
LEDSMALL
R8
680
0805
Reset
RES_N
3.3V
R9
1K
0805
C16
0.1u
0603
SW1
SWITCH
PUSHBUTTON
Power Supply
5V
C12
22uF
CAP
VOUT
GND
VIN
U6
LM1117DTX-5.0
C11
22uF
CAP
6V
3V3
C15
22uF
CAP
VOUT
GND
VIN
U8
LM1117DTX-3.3
C14
22uF
CAP
5V
J4
1
2
CON2


Serial Communication Interface
To PC RS-232 Port
PC RS-232 Port Configuration:
Baud Rate : 38400
Data : 8 bits
Parity : None
Stop : 1 bit
Flow control : None
J5
CON4
SIP4
C19
0.1uF
C21
0.1uF
C17
0.1uF
3.3V
U9
SP3232ECP
V+
V-
VCC
GND
C1+
C1-
C2+
C2-
T1IN
R1OUT
T2IN
R2OUT
T1OUT
R1IN
T2OUT
R2IN
C18
0.1uF
C20
0.1uF
P42
P41
Display
U10
DLR1414
D[7..0]
A[6..0]
SEL_DISP
A0
A1
WR
D0
D1
D2
D3
D4
D5
D6
VCC
GND
5V
C22
0.1uF

6. 参考文献

1. LM1117/LM1117I 800mA Low-Dropout Linear Regulator, 2002, National Semiconductor Corporation.
2. Quadrature Decoder/Counter Interface ICs, 2002, Agilent Technologies.
3. 3953 Full-bridge PWM Motor Driver, 2002, Allegro MicroSystems, Inc.
4. HD151015 9-bit Level Shifter/Transceiver with 3 State Outputs, 3rd Edition, June 1993, Renesas Technology Corp. (Ref. no.: REJ03D0300-0400, http://renesas.com)
5. DLR1414 4-character 5 × 7 Dot Matrix Alphanumeric Intelligent Display with Memory/Decode/Driver, Infineon Technologies.
6. DC-Micromotor 2230 U-006S with 09BP Optical Encoder, Faulhaber Group (http://www.faulhaber.com/).

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2004.08.06	—	初版発行

安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ(http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。