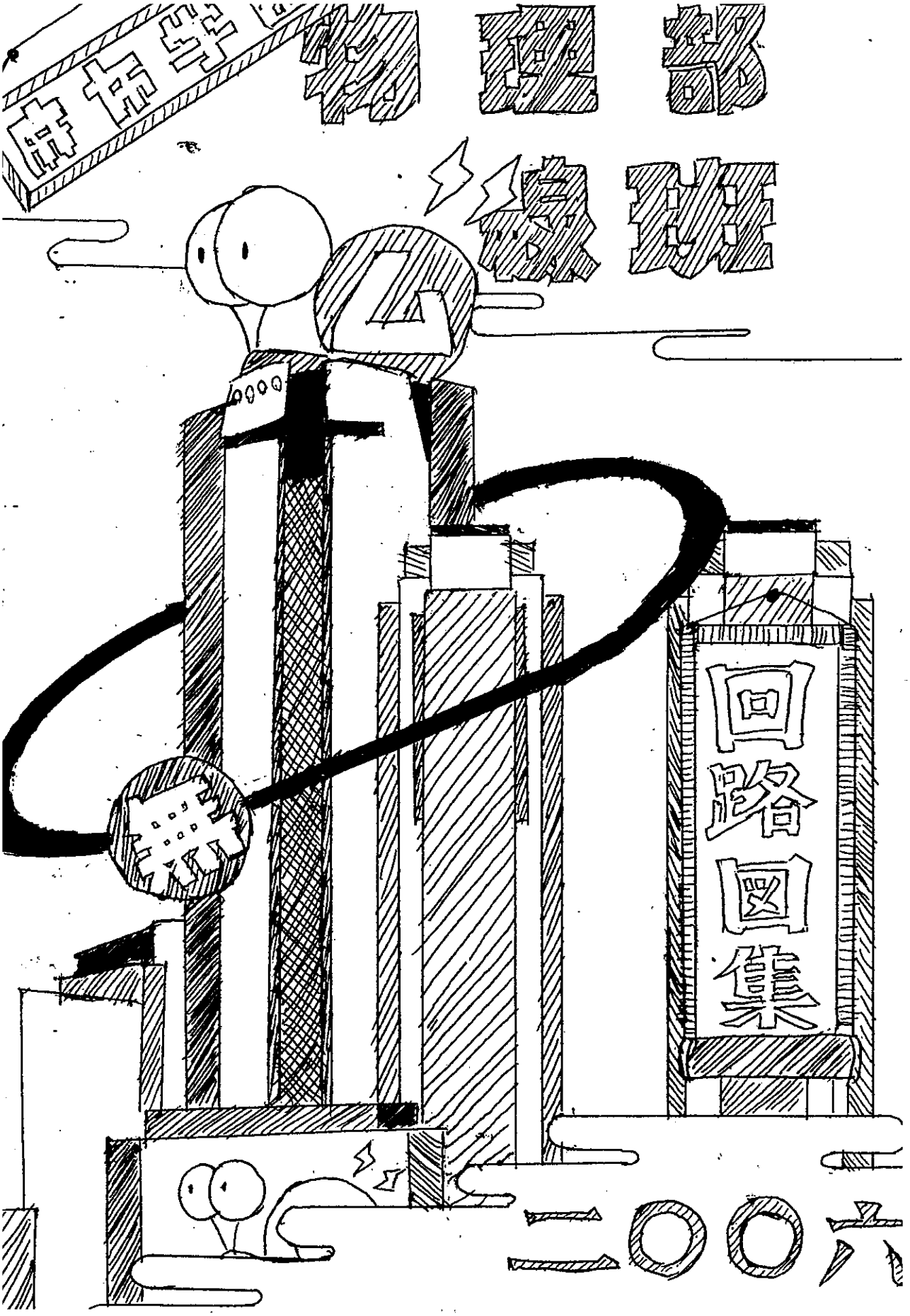

回路図集

はじめに

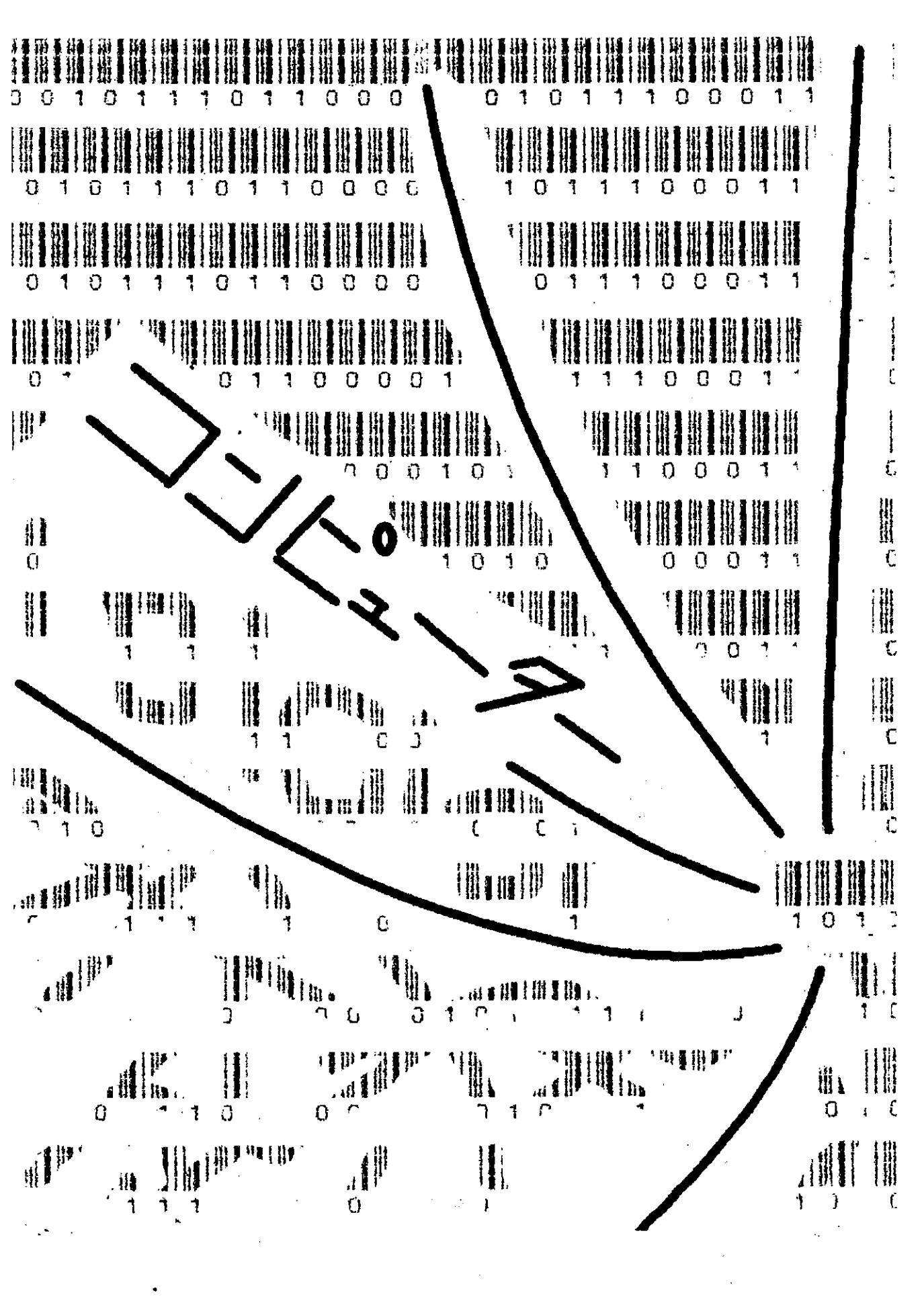
『後少しで4GHzにも達しそうな勢い…』と去年の回路図集にありましたがCore Duoの登場でクロックは3GHzを切って…って下がってますな。

この回路図集には麻布学園物理部無線班の部員が一年間研究、製作したものが記されています。未完成の物もあるためそのまま作ってもうまくいかない場合がありますが当方は一切責任を負いかねますのでご了承ください。

H2会計　伊藤　唯

mokuji モクジ

∵後書き



フェイカーゼロワン

贋作者01

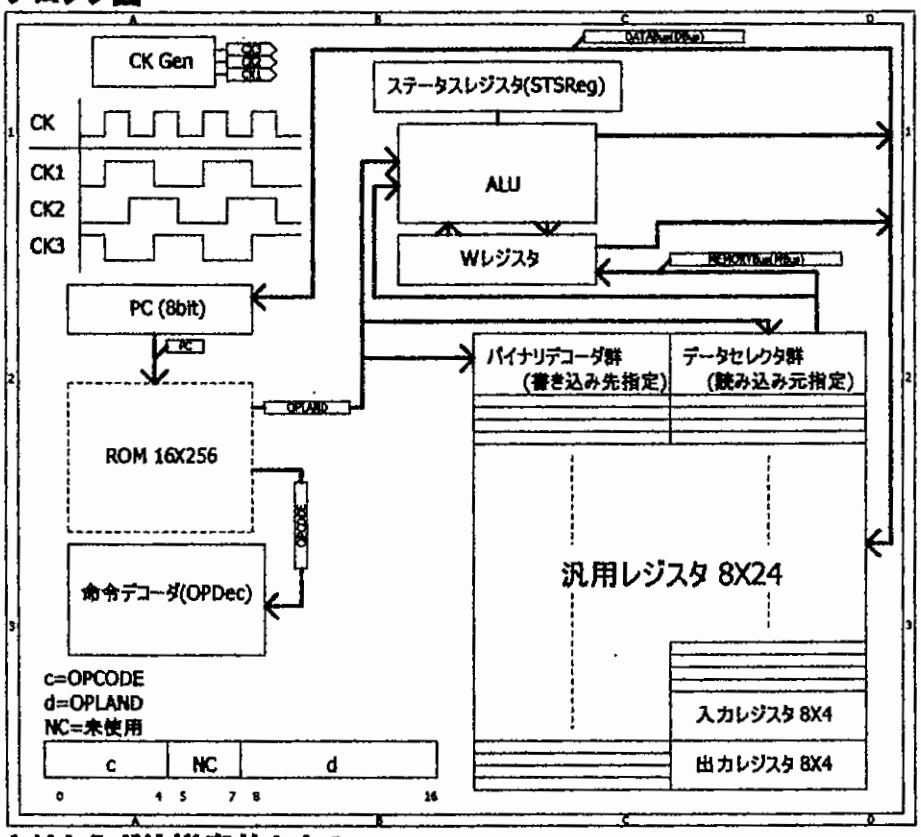
回路設計・製作
中3　高崎尚人
製作協力
物部の先輩方、同輩達

※ここに載っている情報は 4 月 08 日現在のものであり、文化祭当日には変更されている可能性があります。

概要

この作品は、74シリーズの汎用ロジックICと二個のEEPROMを用いて米Microchip社製マイコンのPICシリーズの機能を再現することを目的としたものです。が、自分の無駄なこだわり、頭のメモリ不足、知識不足等の理由によりタイマ、割り込み、スタックメモリが実装されていない、レジスタ、プログラムメモリが半分以下というPICには遠く及ばないものとなってしまいました。

ブロック図



とりあえず仕様表的なもの

アーキテクチャ	RISC、ハーバード型・・・？
I/O アドレッシング方式	メモリマップド I/O
汎用レジスタ	24 バイト(192 ビット)
入出力	それぞれ 32 ビットずつ
最高動作速度	不明(10KHz 出れば十分だと思う)
命令形式	16 ビット固定長(実質 13 ビット使用)

命令セット

命令セットはごく単純かつ必要最低限な RISC 型のもので、全部で 28 命令あります。
以下に簡単な各命令の説明を載せます。ここでは、W レジスタのことを W と呼ぶこととします。
※ビット演算または 1 オペランド演算の場合、11100B を指定すると W レジスタを操作する。

種類	命令名	表記(_は TAB)	説明
ビット演算命令	BS0	BS0_レジスタ_ビット	レジスタの指定されたビットを 0 にする。
	BS1	BS1_レジスタ_ビット_	レジスタの指定されたビットを 1 にする。
1 オペランド演算命令	DECF	DECF_レジスタ	レジスタから 1 減算する。
	INCF	INCF_レジスタ	レジスタに 1 加算する。
	RSHIFT	RSHIFT_レジスタ	レジスタを右シフトする。
	LSHIFT	LSHIFT_レジスタ	レジスタを左シフトする
2 オペランド演算命令	ADDLW	ADDLW_リテラルデータ	W とリテラルデータを加算する。
	SUBLW	SUBLW_リテラルデータ	W とリテラルデータを減算する。
	ANDLW	ANDLW_リテラルデータ	W とリテラルデータを論理積する。
	ORLW	ORLW_リテラルデータ	W とリテラルデータを論理和する。
	XORLW	XORLW_リテラルデータ	W とリテラルデータを排他的論理和する。
	ADDWF	ADDWF_リテラルデータ	W とレジスタを加算する。
	SUBWF	SUBWF_リテラルデータ	W とレジスタを減算する。
	ANDWF	ANDWF_リテラルデータ	W とレジスタを論理積する。
	ORWF	ORWF_リテラルデータ	W とレジスタを論理和する。
	XORWF	XORWF_リテラルデータ	W とレジスタを排他的論理和する。
データ移動命令	MOVELW	MOVELW_リテラルデータ	リテラルデータを W に格納する。
	MOVEWF	MOVEWF_レジスタ	W の値をレジスタにコピーする。
	MOVEFW	MOVEFW_レジスタ	レジスタの値を W にコピーする。
条件分岐命令ジャンプ命令	GOTO	GOTO_アドレス	指定されたアドレスにジャンプする。
	GOTOC	GOTOC_アドレス	C フラグが 1 なら指定されたアドレスにジャンプする。
	GOTOZ	GOTOZ_アドレス	Z フラグが 1 なら指定されたアドレスにジャンプする。
	GOTOA<B	GOTOA<B_アドレス	A<B フラグが 1 なら指定されたアドレスにジャンプする。
	GOTOA=B	GOTOA=B_アドレス	A=B フラグが 1 なら指定されたアドレスにジャンプする。
	GOTOA>B	GOTOA>B_アドレス	A>B フラグが 1 なら指定されたアドレスにジャンプする。
その他	COMPLW	COMPLW_リテラルデータ	リテラルデータと W の値を比較し、ステータスレジスタに格納する。ちなみに A が W である。
	COMPWF	COMPWF_レジスタ	レジスタの値と W の値を比較し、ステータスレジスタに格納する。

主なハードウェアの構成

・ALU の中身

ALU とは Arithmetic and Logic Unit（算術・論理装置）の略で、転送されてきたデータ元に実際に演算を実行する部分です。
Pentium4 等の高度なものになると FPU など他の種類の演算回路も搭載されているものですが、属作者 01 の ALU はきわめて単純で、8 ビットの整数しか扱えません。（手作業で、しかも一人で配線するとこれが精一杯という話も・・・16 ビットにするとバスだけで単純計算 2 倍なので。浮動小数点とか絶対無理。）
加減算と論理積論理和、排他的論理和は 74F181 によって実現しています。この 74F181 というのは、昔 PC がマイクロコンピュータと呼ばれていた時代に演算の ALU としてよく用いられたもので、キャリーをつないで並列動作させることで 4×n ビット、32 種類の演算を実行することができます。ただし、ここでは命令セットを複雑にしないために 5 種類しか使用していません。
BS0、BS1 は 74F537(BCDto10 デマルチプレクサ)からの出力とオペランドを論理積、及び論理和しています。入力の 3 ビット目と出力の 8,9 ビット目が余りますが、入力 3 ビット目を L 固定しておけば問題なし。
SHIFT はただ単に配線を 1 ビット分ずらしただけという手抜き仕様。でも、キャリーはちゃんと発生します。その後、命令コードに従ってデータセレクタで演算した箇所の出力を選べば演算結果となるわけです。（少なくともこの CPU では。）

・内部バスの構成
内部のバスは 1 オペランド演算のときにレジスタから読み出して結果をレジスタへロードするためにレジスタ→ALU 専用の Mbus(メモリバス)、結果の転送には Dbus(データバス)を使うという力業で実現しています。ちゃんとした CPU ならパイプライン処理及びステージが実装されていたり、ブロックダイヤグラムがきちんと考えられていたりするのでひとつのバスを共有することもできるのでしょうが・・・

・汎用レジスタ及び入出力
ブロック図を見てもらえばわかりますが、データの読み出しにデータセレクタを使用しています。
せっかく 74F574(3 ステート D-FF*8)を使っているのだから、読み出すところ以外はイネーブルを H にして無効化すれば良かったのに、何考えていたんだ俺は・・・と言いたいところですが、設計初期によく考えずに実装してしまった部分なので仕方ないのです(殴
しかもオペランドが固定 8 ビットという仕様の中で BS0 及び BS1 を実装しようとしたのでビット指定に 3 ビットとられる計算になり、5 ビット分つまり 32 バイト分のレジスタしか設定できませんでした。
ここらへんはオペランド長を可変にする機構を思いつけなかった俺の知識不足です。
8 ビット分・・・いや、せめて 6 ビット分レジスタ指定ビットが取れれば PIC と同様に特殊レジスタ実装してバンク切り替えという方式をとっても良かったのですが・・・
あ、未使用の 3 ビットをバンク指定に使えば良かったのか・・・今気付いた orz
でも、そうすると回路規模が現実的でなくなる(単純計算 2 倍、セレクタをなくして 3 ステートを使っても一部の配線密度がすごいことになる)ので、賢い人は素直にメモリチップ(SRAM)を使うべきだと思います。
特殊レジスタ実装するなら特定のオペランドのときだけロード信号を横取りすればいいわけで。
そして、入出力についてです。
PIC の場合は入出力切り替えですが、厳作者 01 は役割が固定です。
つまり、切り替えができません。というか、そこまで機構を考える知識がありません(殴
そのかわりというか、ただ単に管理が楽だからですが、それぞれ 32 ビットずつという無駄に大きなポートを持っています。まあ、タイマがないのでシリアル通信ができないという。まあ、エッジトリガでタイミングをはかるパラレル通信なら何とかならないこともないですね。超低速になりますけど。

・デコーダ
たった 5 ビットの OPCODE から各所の制御信号をワイヤードロジックにより作り出す場所です。
中身はとにかくゲートの嵐で、合計 22 個の 74LS00、02、04、08、32 と 2 個の 74LS20 をほぼフルに使っています。こんな単純な CPU もどきでも、けっこう大きな物が必要なのです。
おそらく、命令から各所の制御信号を考えて真理値表をつくり、論理式化して論理反転により整理したこの部分が一番設計に手間かかっていると思います。
実は、GOTO 系命令の条件判断が一番大きな場所をとっていたりします。

・W レジスタ
まあ、独立した項目作らなくてもいいんですけどね。
ブロック図上は別のユニットになっているのでとりあえず。
W レジスタは演算結果を保存しておく場所で、2 オペランド演算の時には強制的に W レジスタの上に演算結果が上書きされます。つまり、W レジスタの中身を残しておきたければ MOVEWF しろという意味です。
2 オペランド演算のときは読み出し元指定で既に 5 ビット分取られてますし、保存先指定ビットを設ければよいという話もデコーダが複雑になるので断念しました。
ちなみに、1 オペランド演算のときは W レジスタを直接示すアドレスがないため入力レジスタのアドレスを指定すると W レジスタが演算結果を横取りするという力技で解決しました。
入力レジスタはクロックに合わせて入力を読み込むだけなので内部からの書き込みはできないからです。
(書き込みアドレス指定のバイナリデコーダの入力レジスタにあたる出力は GND に落としてある)
普通の CPU ならこんなこと絶対しません。多分。

その他細かい事
一部の演算の種類わけに 2 オペランド演算という言葉を使っていますが、これは正確ではありません。
本来、オペランドというものは命令コードに付随するデータのことです。しかし、ここではほかに名称が思いつかなかったのであえて 2 オペランド命令と呼んでいます。
ただの中三が CPU を作ろうとしてもこんなものですかね。少なくとも数年に一回は設計を一新している Intel や AMD のアーキテクト達はすごいですね。偉人です。
そんなわけで、ところどころ口調が変わっていたり、わかりにくい部分があったかもしれませんが、最後まで読んでいただいてありがとうございました。

ROBO
ROBO
ROBO
ROBO

Rapit

..My Dream,and My Result.

By H2-7 Masataro Asai

回路図集お買い上げありがとうございました。
今回の私の製作物は、自立**2**足歩行ロボットです。

Spec:性能

CPU：**Microchip　PIC16F819**マイクロコンピュータ
腰関節自由度***3**
膝第一関節自由度**1**
膝第二関節自由度**1**
足首関節自由度**2**
サーボモータ：**S03T 2BB**サーボモータ
Torque:トルク******　**7.2kg/cm(4.8V)　8.0kg(6.0V)**
Speed:角速度*******　**0.33s/60°(4.8V)**
0.27s/60°(6.0V)

*****自由度とは：関節ごとのモーターの数、ぐらいに考えておけばいいと思う。
******トルクとは：**1cm**の棒をモーターの軸に固定して回そうとした時、棒の先をどこかに引っ掛けるとする。そのときに棒の先から引っかかったところにかかる力、と考えればいい。てこの原理で、**10cm**の棒の先にかかる力はトルクの**1/10**の力。
*******角速度とは：一秒でサーボモータが回る角度。

身長：**40cm**ぐらい
体重：**900g**程度

Walking with two leg : 2足歩行の話

人間の脳はすごくて、体中のあらゆるセンサーを無意識のうちに処理し考えることができるため、いとも簡単に「倒れる前に次の足を出す」ということをやってのけてくれる。しかし、この**CPU**は**PIC16F819**。そんな処理能力など到底持ち合わせてはいない。よって、なるべく工夫したプログラミングを行う必要があった。

Level1：静歩行

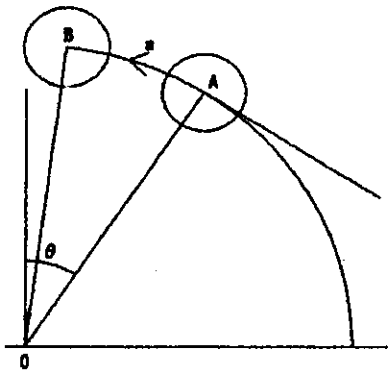
これは、重心が常に支持多角形の内側にあるような歩き方である。支持多角形とは、すべての接地面を凹にならないように囲んだ多角形のことである。重心の移動の仕方は直線的であり、初歩の初歩の歩き方である。その歩き方は、人間でいえば「忍び足」に似たものだ。

全重心の座標は、部品ごとに小分けされた重心の座標の**X,Y,Z**座標にそれぞれの部品の重量を掛け合わせたものの和を、総重量で割って算出される。これが、足の裏の座標の中に入っているかが重要である。足の裏の面積は**57*95**で、横方向が狭くなっている。

Level2：動歩行

動歩行とは、静歩行とは違い、重心が必ずしも支持多角形の中にあるというわけではないという特徴がある。重心の移動がスムーズなのにくわえ、そのことによって体の慣性をうまく使うことができ、より効率的である。
下の図を見てほしい。**OA**という棒があり、**A**に重さ**m**の粘土がついているとする。**O**を固定して、**a**の角速度で棒を垂直方向へ押し出すと、**B**でいったん速度がゼロになり、そして倒れてしまう。
ここで、**A**から**a**の速度で押し出したら、何秒後にどこで静止するか(つまり**B**)が計算できる。このように、与えた力と時間から、移動結果を予想できるやり方を、動力学と言う。逆に、今どこにいて、いつにどこまで移動させたいかから、与えるべき力を予想

することを「逆動力学」という。
このことから、重心を支持多角形からはずして体を左右にゆすることは次のプログラムで可能だ。



①加速度センサによって重心の移動を検知・計算し、**B**で加速度がゼロになるのを計算して遊脚（接地していない方の足）を体の反対側に踏み出す。

②遊脚は地面に接触して固定脚になる。新しい固定脚にかかる衝撃を計算して膝を曲げながら吸収し、また左右への勢いを殺さないように足首と腰の**Roll**軸を調整する。上体は常に水平を保つようにする。

③古い固定脚の足を伸ばし、地面をける。重心は新たな静止点に向かって上昇する。

この動作をはじめるためには、まず片足立ちの静止状態になり、そこから重心を意図的に接地面外に動かすように「わざと」バランスを崩す動作が必要になってくる。
また、これに前後方向の推進を追加するためには、両足を前後に動かす動作が必要だ。前後方向に重心をずらし、そこから得られる加速を保つよう制御するプログラムである。

①先ほどの体を揺らす状態に保つ。
②重心が左右方向に静止した時点で、必要な歩幅・速度から計算される角度にしたがって股を前後に開き始める。
③接地する瞬間の前後方向の加速度を計測し、新たな遊脚の空中での前後運動を制御する。

ただし、この歩行はまったく平らな面での歩行を考えているものであり、また足の裏が、地面を離れるまでは地面に吸着してい

ると仮定しなければならない。

Level3:外力・段差による影響からのフィードバック

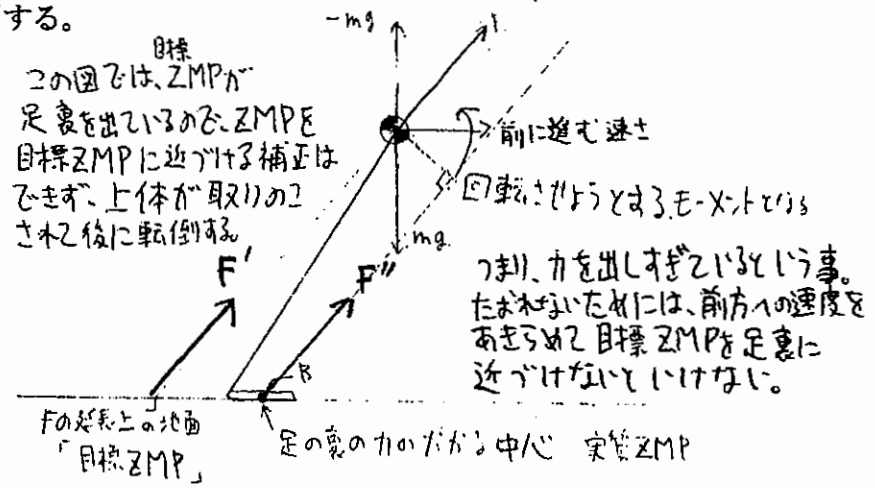
ホンダの**ASIMO**や、その前駆体において研究されてきたのがこの内容である。ホンダはこのことに関してかなりの特許を所有しているが、学生なので使用料は関係ない。

外力による影響とは、たとえばロボットは「押されて倒れてしまうもの」と「押されたら後ろに足を一歩踏み出すもの」という点で二つに分かれている。では、後者のロボットは前者と比べ具体的に何が違うのだろうか?

ZMP:Zero Moment Point

ZMPとは?

みなさん、走り出すときに足の裏の「どこに」一番力をかけていますか?人間は無意識にこれを算出し、歩行時の姿勢の修正に使用する。



同じく、最先端のロボットにおいてもこれは重要となっている。上図を見てほしい。ロボットは「倒れずに前にだけ進みたい。」だから、重心に「前に進む力」と「倒れないための重力**mg**とは逆の方向の力」をかけなければいけない。これらの合力**F**が実際にかかるのは、地面の点**B**である。この**B**は、実質**ZMP**(床反

力作用点 とも言う）と呼ばれ、**B**にかかる力のことは床反力といわれる。

もし力がかかる位置が　　からずれると、下図のように重心に回転を生じさせてしまう。だから、ビリヤードの玉は中心をうまく打たないといけないように、重心を力の方向に延長した上の点において力を発生させなければいけない。この点が「目標**ZMP**」である。**ZMP**の意味は、そこに力をかけても平行移動するだけで、回転を意味するモーメントは生じない場所という意味だ。モーターによって発生する力の合計を、目標**ZMP**に合わせ、ちょうど**F**と同じ方向･大きさに調整すれば、元の理想の「倒れずに前にだけ進みたい。」が実現できる。

また、目標**ZMP**は意図的に変えることができる。足首の**Pitch**軸を帰ることによって、すぐに重心の位置も変わるので、目標**ZMP**も変化するのだ。（上図）これにより歩くスピードも変わり、また歩行中の安定も図ることができる。ただし目標ZMPは支持多角形内。

最先端のロボットは、目標**ZMP**と実質**ZMP**が、意図的ではなく何らかの外力によって変化させられたとき、それを足の裏の力センサによって検知して、姿勢を安定にしようとする。**ASIMO**の場合、床反力制御、目標**ZMP**制御、着地位置制御の３つを使って姿勢を安定させる。

① 床反力制御 **(**図①**)**

床反力制御は、床の凹凸を吸収しながらも、倒れそうになったときに足の裏で踏ん張る制御。例えば、ロボットがつま先で石を踏んだ場合、実質**ZMP**は石を踏んだ点となる。このときつま先を持ち上げる（かかとを落とす）ことで目標**ZMP**まで戻す。また例えば、外力･床の傾斜ででロボットが前傾してしまったときには、つま先を下げて踏ん張ることで、実質**ZMP**を目標**ZMP**より前にずらし、後ろに傾きを戻そうとする。ただし、実質**ZMP**は支持多角形の範囲から越えないので、姿勢復元力には限界があり、ロボットが大きく傾いた場合には転倒してしまう。

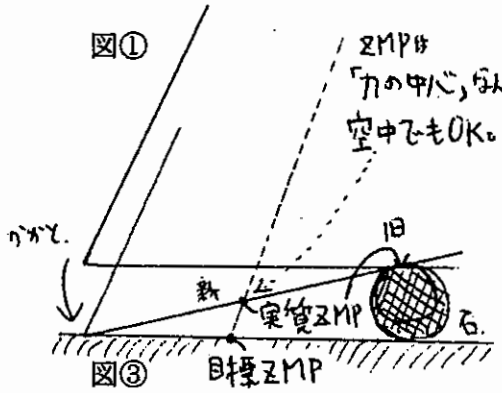
②目標**ZMP**制御**(**図②**)**

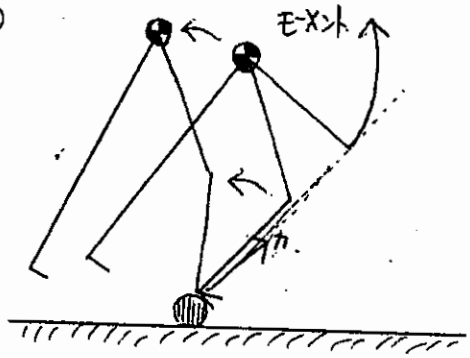
ロボットが大きく傾いた場合には、さらに目標**ZMP**制御が働いて

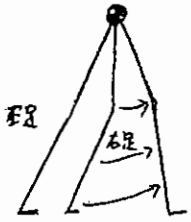
転倒を防ぐ。具体的には、図のように前方 に倒れそうな場合には、ロボットの上体を理想の歩行パターンよりも一層強く前方に加速させます。この結果、体が勢いあまって動くことで、理論上では目標**ZMP**が実質**ZMP**よりも 後方に移動し、後ろに倒そうとする力が働き、姿勢の傾きが戻る。

③着地位置制御

「目標**ZMP**制御」が働くと、目標としていた上体の位置が、より強く加速した方向にズレてしまう。このとき、いつもと同じ歩幅（理想の歩幅）で次の足 を出すと、体は前に出ているのに足はいつもの位置なのでおかしい。「着地位置制御」は、歩幅を適切に修正して、上体と足の理想的な位置関係を取り戻す。



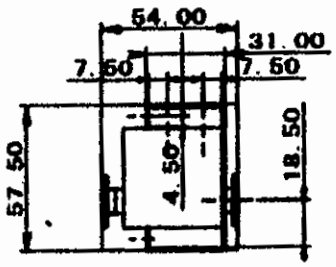




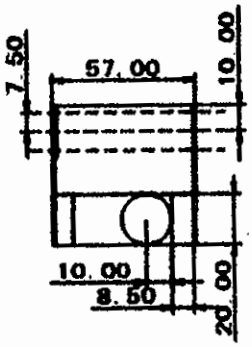
By the Mechanical side... : 本体製作の話

サーボモーターが**14**個もついているこの本体には、位置からの切り出しのため多大なる苦労を要した。

Pitch軸・**Yaw**軸サーボブラケット

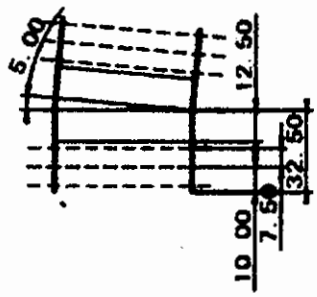


Roll軸サーボブラケット



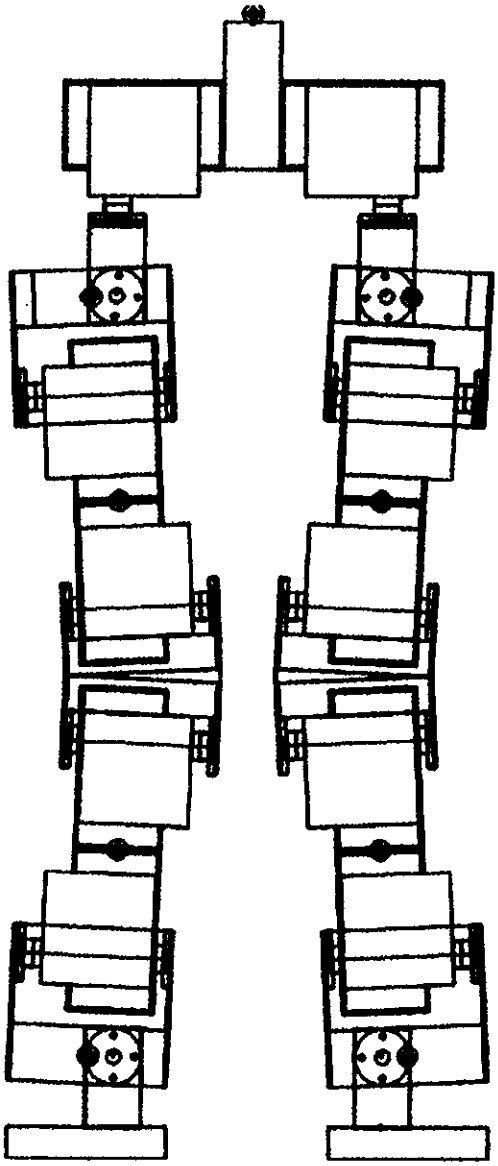
全アルミ製で、曲げ加工を使用している。穴は**3mm**(サーボ固定用)、**2mm**(サーボブラケット接続用)、**5mm**(サーボホーン-サーボモータ固定用)、**6mm**(サーボホーン-サーボブラケット接続用)。固定はすべてミリねじ。また、人間に近い骨格の形を実現するために、大腿骨とすねの骨のあいだが内側に曲がっているのを再現して、膝に**5**度の傾斜角を設けた。一番上の**Yaw**軸のサーボブラケットは、基盤固定用にアルミのロの字棒を固定し、反対側のサーボと接続した。

膝のサーボブラケット



サーボとサーボブラケットの接続は、個人で考えた特殊なものである。まず、ブラケットを構成する**1mm**のアルミの板には、前もって直径**6mm**の穴をあけておく。そして、サーボホーン（余っていた物）のサーボと接続する側に、**6mm**のプラスチックのスペーサーを突っ込んでおく。サーボから飛び出ている部分を削り、**1mm**分だけサーボホーンから飛び出すようにすれば、そこをブラケットの穴にうまくはめて、サーボブラケット同士よく回転するようになる。

ほとんどのサーボブラケットは**1mm**厚のアルミ板を**20mm**の帯にし、穴あけ課程をすべて済ませた後曲げ加工で生成している。**yaw**軸サーボと**roll**軸サーボとの間もしくは**roll**軸サーボと足の裏部分との接続部のみ、**1.5mm**厚で作っている。



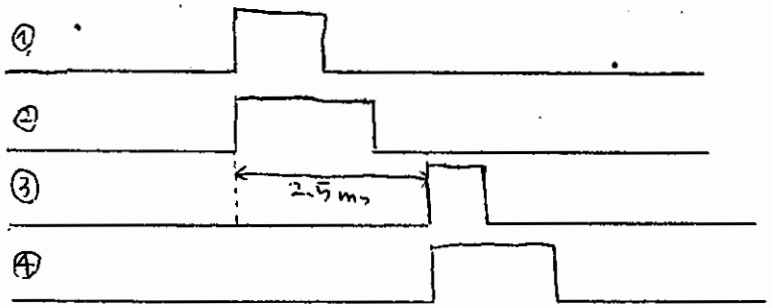
全体図

丸に十字が入っている記号は、重心の記号。

Controlling of servo motor:サーボ制御の機構

サーボ制御は**PIC16f819**で行っている。どこかの気違いがタイマー**IC**を使えというが気にはしない。

サーボの信号の図



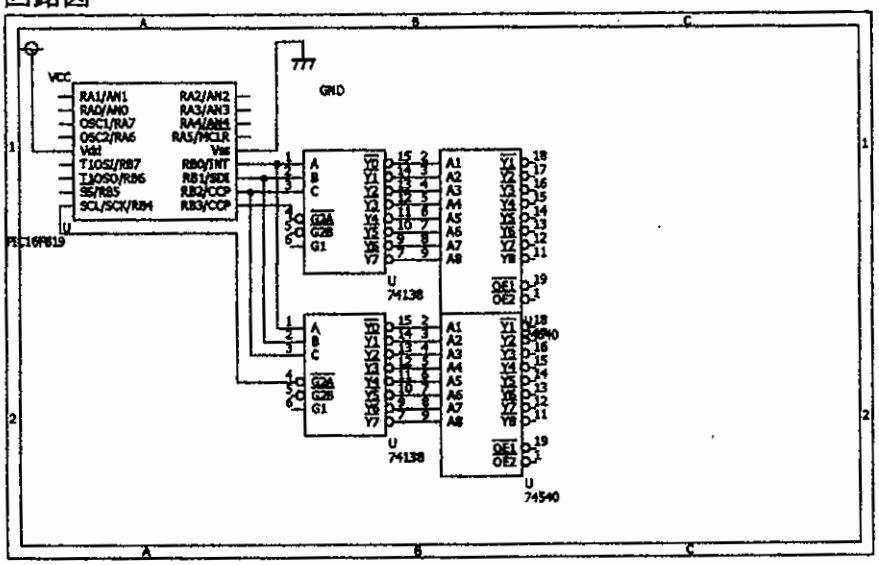
前年度舟渡氏のサーボモータは**20ms**を**8**分割していたが、今回はその分割をソフトウェアでさらにあげていこうというやり方をしている。

Timer1が**CCP1**と同じ値になると起こる割り込みを使用する。以下は割り込み部分のプログラム。

```
#INT_CCP1
                CCP1_isr(){
                if(begin)
                {
                                signed long s;
                                output_B(num);
                                set_timer1(1);
                                s = duty[0][num] - duty[1][num];
                                ir = (s>0);
                //duty[0][num]>duty[1][num]
                                yet = (s!=0);
                //duty[0][num]!=duty[1][num]
                                CCP_1 = duty[ir][num];
                                begin = 0;
                }
                else{
                                if(yet){
                                                                //追加計測
                                                if(ir)
                                                {
```

```
                                        output_high(PIN_B3);
                                        CCP_1 = duty[0][num];
                                }
                                else
                                {
                                        output_high(PIN_B4);
                                        CCP_1 = duty[1][num];
                                }
                                yet = 0;
                                        //追加計測完了
                        }
                        else{
                                        //計測終了処理
                                output_high(PIN_B3);
                                output_high(PIN_B4);
                                CCP_1 = 5090;
                                num = ++num & 0b00000111;
                                yet = 1;
                                begin = 1;
                        }
                                        //計測終了処理完了
                }
}
```

回路図



流れとしては、**ABC**でサーボを選択し、はじめは右サーボ･左サーボともに**H**、デューティの短いほうのみを**L**にしてもう一度割り込みを待ち、両方のサーボの信号を**L**にする。**PIC**は**16bit**の演算は代入だけでも遅いので、なるべく計算量を少なくするために

工夫した。両方を**H**にしてから**2.5ms**後、**ABC**の値をカウントアップする。

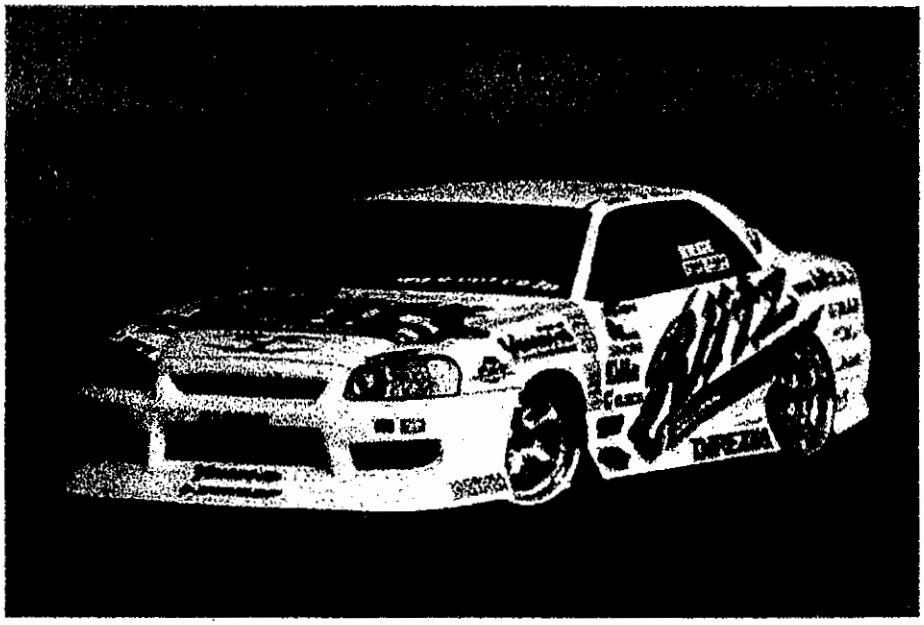
Finally：最後に

最後の一年でかなりの完成度の物を作れて満足だとは思っているが、しかし動いているというわけではない。**3D**の**CAD**を使っているわけではないので、自前で、動く関節のデータをもとにした重心の位置を計算する式を**excel**で書いている途中だ。この先**PC**と**PIC**を通信させて重心計算をやらせる**PC**用のプログラムや、それ用に新たに**16f877**をつけたりと多少の回路変更はありそうだ。

そしてお前は横を向く。

製作　高２　伊藤　高１　井上
協力　物無の人たち。

↓完成予想図（ボディーの乗せればこれぐらいにはなる、ｶﾝ、きっと）



1,　～製作物のセツメイ～

えー。名前に特に意味は無いのですがハッキリ言ってしまえばラジコンカーです。フル４WD のラジコンカーで赤外線による遠隔操作ができるように作っています。

また、いろいろ理由がありましてドリフトができるようになるということを目指して製作しました。今現在（４月１０日）においての製作状況としては本体を共同制作者である井上が、回路系はすべて僕が作っています。井上くんの音声認識回路を載せてリアルミニ四駆を実現しようとしたのですが・・・・後を見ればわかります。

2, ～製作までのいきさつ～

去年の夏あたりから僕はかなり車にハマりはじめ、今年度の製作物を作り始めるときも頭の中は車でいっぱいでした。そして様々な車情報誌を読んでいてドリフトラジコンという物が流行っているということを発見したのです。初めて記事を見たときにインスピレーション沸きまくりですぐに製作が決定しました。

3, ～ドリフトとは何か。そしてラジコンドリフトとは～

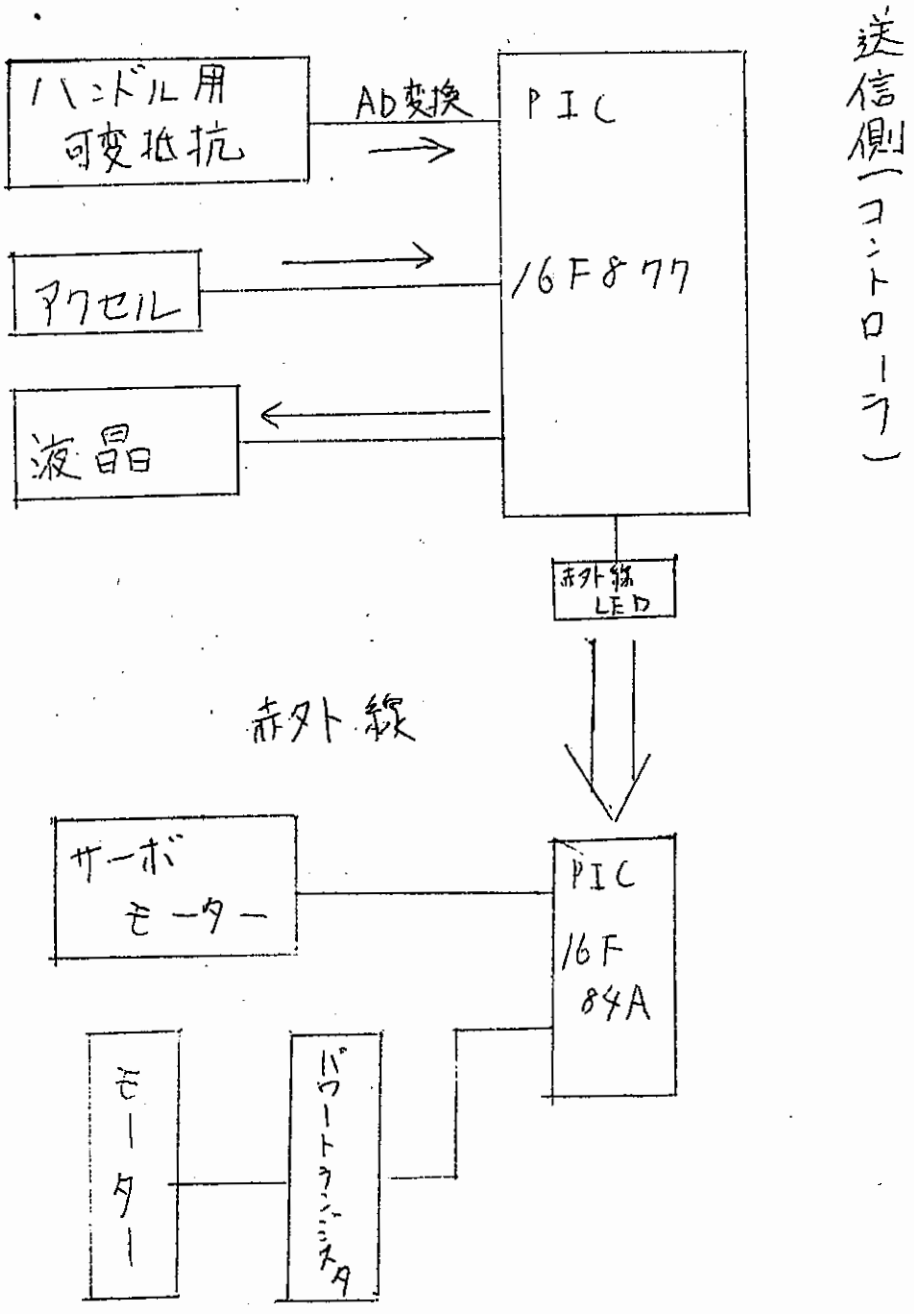
ニュースで一度はドリフト族と呼ばれる人達のニュースを見たことがあるでしょうか。ものすごい音を立てて埠頭などを暴走する様子が映されていたと思います。まぁ、本物のドリフトを見たことが無い人でもドリフトという単語は知っていることでしょう。かの有名な任○堂社の某ゲーム機のマ○オカートですらドリフトとういう物が存在したのです。要するにドリフトという単語は万国共通なんですね（ぇ。

ただ誤解して欲しくないのはレースでドリフトをすると速いということは決してなく、ドリフトとはあくまでもパフォーマンスのためにあるということです。ラリーなどの例外はありますが。

実車においてはドリフトは基本的に後輪駆動車でやるものですがラジコンでは前に書いたように四輪駆動でやります。なぜかといいますとホイールベース（前輪と後輪の間の距離）があまりにも短すぎるためにすぐにスピンしてしまうからです。ラジコンにおけるドリフトは後輪を滑らせるのではなくすべてのタイヤを滑りにくい物に変えてすべてのタイヤを滑らせる四輪ドリフトなのです。ですからラジコンドリフトの動きはＦＲのドリフトと違って角度のついたカウンター（逆にハンドルを切ること）のあまり無いものになります。

4, ～回路説明～

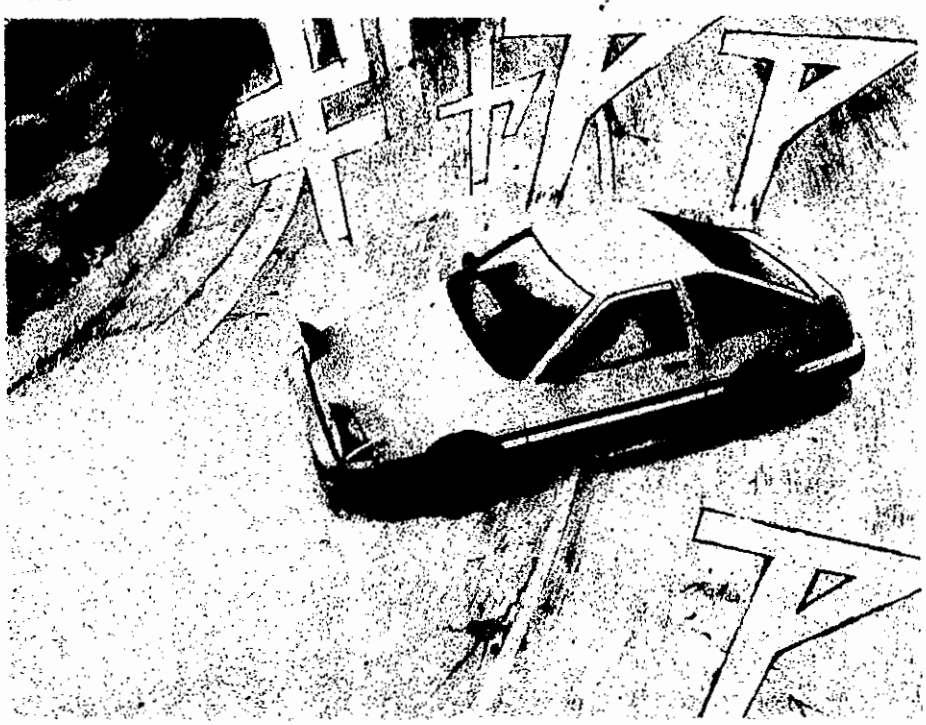
基本的には複雑な回路では無く、プログラムが少しめんどくさいかなと言ったところです。



5, ～プログラミング～

赤外線に二つの情報を入れる為に 8bit をハンドルの角度、1bit をｱｸｾﾙの ON OFF として９ビットのデータをシリアル通信しています。
コントローラに PIC16F877 をひとつ、受信側（本体）にも PIC16F84A をひとつ乗せています。
送信側でやることはＡＤ変換、液晶表示、ｱｸｾﾙ入力、そして入力されたデータからの赤外線送信データ作成です。
受信側では赤外線を受けて８bit のハンドル角度データと 1bit のｱｸｾﾙ開閉データをわけてサーボモータの制御、及び走行用のモーターの制御を行います。

↓動作している図

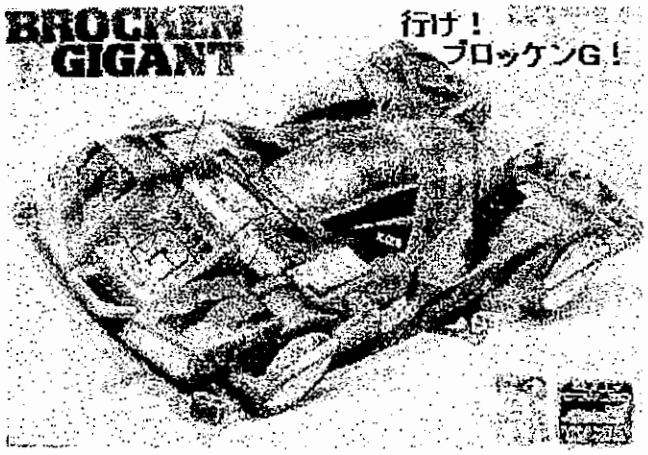


•音声認識

音声認識をする際の手順としては以下の方法が挙げられます。

①まず、マイク等の外部入力から音声データを入力する。
②入力されたデータから元の音(a, o, ki 等)を読み取る。
③②で読み取ったデータから発声された大まかな単語を推測する。
④③で推測した単語と前後の単語から何を発声したのかを推測する。

音声認識をする際に一番の問題となってくるのが④の作業で、これはあらかじめ登録されている「辞書」の中から適する物を探すことになる訳ですが、例えば「壁」という単語は登録されていて「雨」という単語が登録されていなかった場合、「雨が降っている」ということを言ったつもりでも「壁が降っている」と認識してしまうために話が通じなくなります。これを改善する方法として、認識させる文法を設定してそれに合わない発言は無視する(ルールグラマ)と音声認識をする場所によってその単語の発生率を設定し、使う場所で切り替える(言語モデル)というものがあります。
今回は「まえ」、「うしろ」等の比較的簡単な語句を入力するのに使おうとしているので辞書の問題はそこまで考えなくて済みます。



①フロントタイヤはハンドルをきれるようにしないといけないためにユニバーサルジョイントという物を使用しています。これは右図のようにコの字の物体を組み合わせたような物であり、これによってある程度の角度まではメインのシャフトからタイヤを曲げた状態でも回転を続けられることになります。また、①の部分にはサスペンションが取り付けられているのですがこれについては後述します。

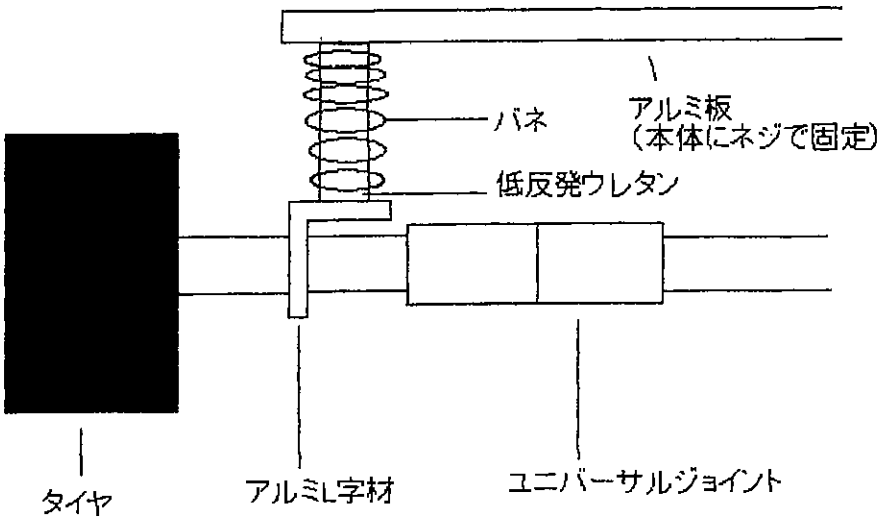
②、④これらは基本的に同じ構造で中央を通るシャフトとタイヤを付ける横向きのシャフトとを結ぶためのギヤです。2つの金属ギヤを使っているだけのもので、②と④の違いはギヤの左右の向きです。これは向きを同じにすると前後のタイヤとも内側ならボディの内側に向かってしか動かなくなり進めなくなるためです。

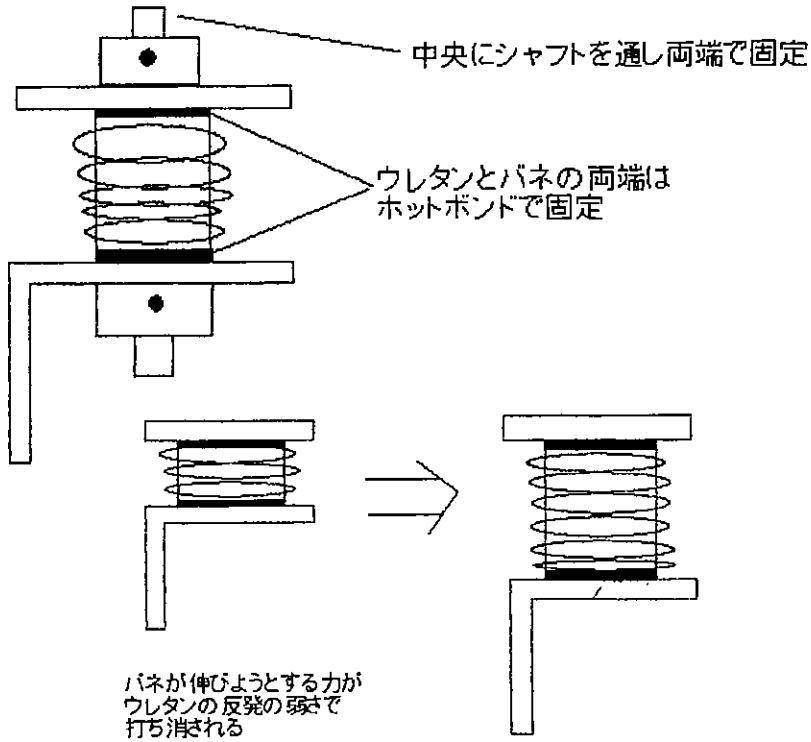
③この部分ではモーターを取り付けて実際にシャフトを回しています。中身は図のような配置で左から金属ギヤ(小)、金属ギヤ(大)、モーターが並んでいます。金属ギヤ(大)はギヤ比というよりも、モーターからシャフトへの橋渡しの意味が大きいです。ちなみにモーターはラジコンヘリコプターについていた物を流用しています。

⑤ここでは特に仕掛けをつける必要はないのでシャフトに直にタイヤをつけているだけです。

サスペンションについて

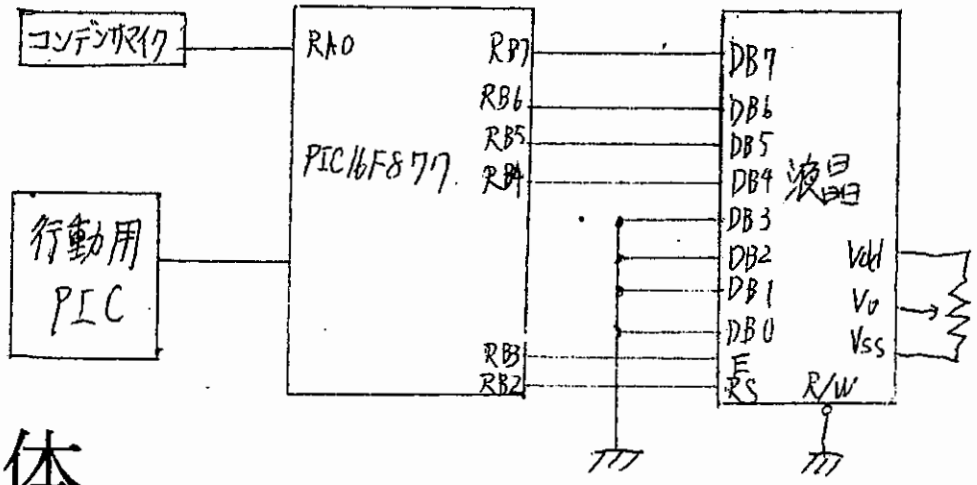
大抵の自動車にはサスペンションというものがついており、これによってスムーズな走行をすることができるようになります。このサスペンションはメインはバネで、でこぼこなところを走りタイヤが持ち上がった時はバネが縮み、乗組員は特に衝撃を感じずに走ることができます。しかし、バネだけではその加わっている力が無くなった時にバネの伸びる力によって車体がはねてしまいます。そこで使うのがショックアブソーバーというもので、これは簡単に言えばバネの伸びる力を抑えるものです。実車では油を使ったものが使われますが、ここでは低反発ウレタンをしようしてその代わりをさせています。





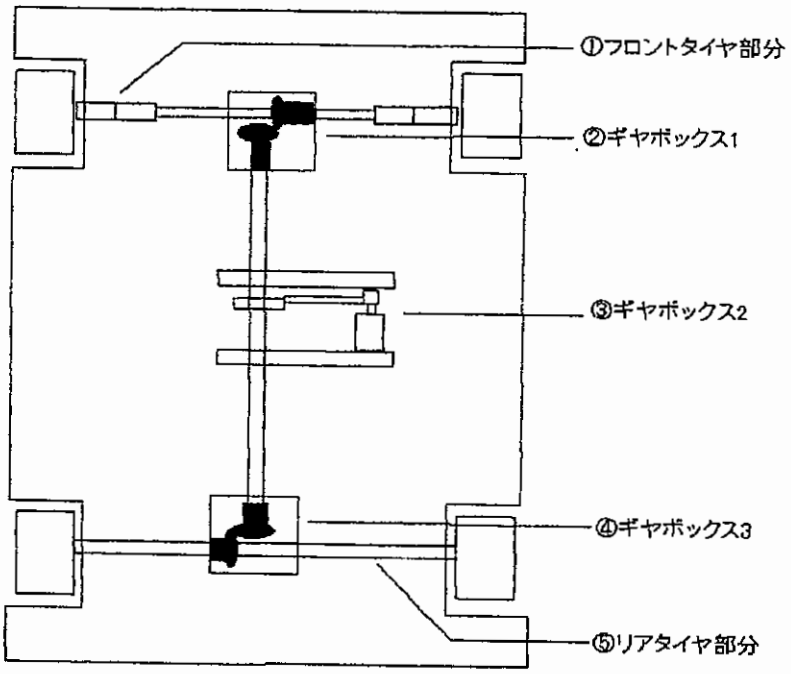
また、タイヤ部分はサーボモーターによってハンドルを切れるようになっています。

回路的にはコンデンサマイクから入力したデータをPIC16F877のA/D変換で変換して、それをそのままPIC内で認識して動作させるという方式で行っています。



・本体

前

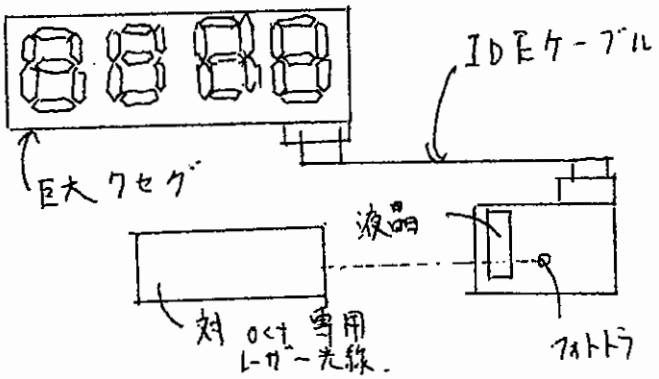


後

入場者かうんた～～～

H２　伊藤

↓完成予想図ってか完成図



1、説明

ハイ、題名からわかるように文化祭における入場者数をカウントするためのものです。基本構成はレーザー送信用の送信機、受信用の回路、受信回路からつながれた７セグメントＬＥＤ（数字を表示する）。

動作としてはまず送信機と回路を向かい合わせにして送信機からのレーザーが回路のフォトトランジスタにあたるようにします。そしてレーザーの通っているところを人が通り抜けるとレーザーが一瞬さえぎられ、またレーザーを受信したときにカウントされるようになっています。それではレーザーの所で立ち止まって足が行き来したりするとたくさんカウントするじゃないかと思うかもしれませんがＰＩＣのプログラムによってたくさんカウントしないようにしてあります。

ちなみにレーザーは安全基準を満たしたものを使用しているので万が一子供がレーザーを少し覗き込んだりしても特に害はありません。だからといって覗いてもよい物でもありませんが。

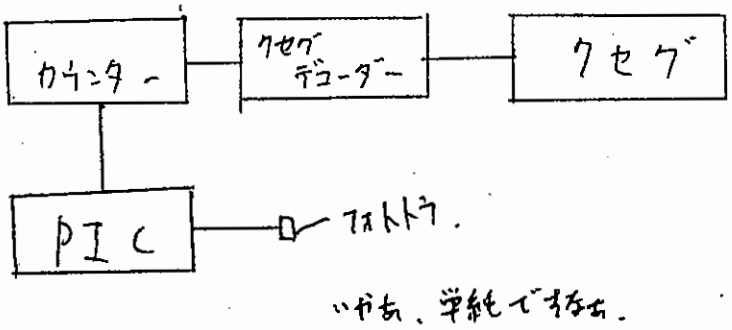
表示用の７セグは秋葉原の鈴商においてもっとも大きいものを選択し、一つ 1800 円の４ケタ分で 7200 円もします。また部員が手元で人数を確認できるようにカウンターなどの回路側にも液晶画面を搭載して人数を表示させています。

2、回路

回路はとても単純でフォトトランジスタの受信した波形の形を治してからＰＩＣに入れて解析し、カウントアップであればカウンタにクロックを入れるという簡単なものです。ただ７セグが特殊で１２Ｖを使うものなので電源は１２Ｖと５Ｖの両方取れるパソコン用ＡＴＸ電源からとってきています。

7セグメントデコーダーである 74LS47 からの出力は特に増幅など無しに７セグに入れてＯＫだったのでヨシとしました。

また、ロータリースイッチ（？）と呼ばれる０から９までのダイヤルがついていて指定した数を２進数で出力する物をカウンターのロードピンにつないで指定した人数から開始できるようにしています。



3、プログラム

プログラムはフォトトランジスタからの入力があれば 0.5 秒間入力を受け付けなくし

た後に一人分カウントアップするようにしています。また手元の液晶画面の表示のた

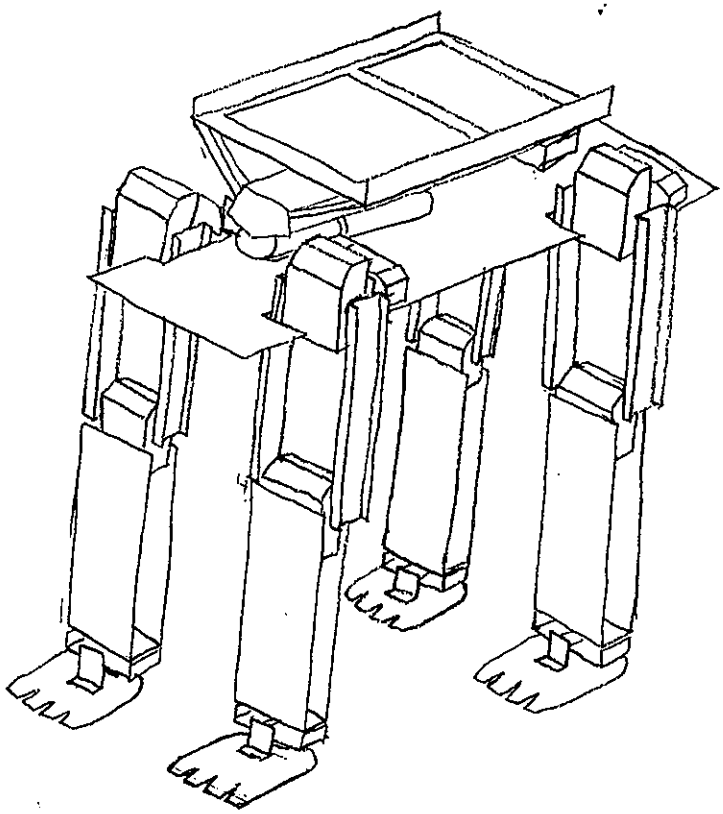
めに汎用ライブラリを使って関数による液晶表示を行っています。
↓以下プログラム（出力ピンなど様々な定義は抜き）

```
//// メイン関数
long data;
void main()
{
    set_tris_a(0b00000010);
    set_tris_b(0b00000000);
    data = 0;                                          //カウンタ初
    期クリア
    lcd_init();                                        //LCD初期化
    lcd_clear();                                       //LCD全消去
    printf(lcd_data,"ﾖｳｺｿﾌﾞﾌﾗﾌﾞﾍ!!!!");                //初期メッセージ
    while(1)                                           //永久ループ
    {
     lcd_cmd(0xC0);  //2行目の先頭へ
     output_bit(PIN_A2, 1);
     printf(lcd_data,"ﾆｭｳｼﾞｮｳｼｬ=%lu",data);
     if(input(PIN_A1) == 0)
     {
                data++;
                output_bit(PIN_A2, 0);
                delay_ms(500);
                output_bit(PIN_A2, 1);
                while(input(PIN_A1) == 0)
                {
                        delay_ms(1);
                }
     }
    }
```

☆

星になった象

製作 H1 安部
M3 竹島
協力 物理部の皆さん



製作の動機

兎に角ロボットが造りたかったのです。で、「ロボットと言えば２足！」とも思ったのですが、流石に重心云々とかの知識も無くぶつかっていっても玉砕しそうだったので、今年はその布石として、重心とかを意識してロボットを作ろうと思い、象型ロボットという結論に至ったのです。何故、重心を意識すると象かと言うと、象はペース歩行という多少面倒な歩行法をしているからです。詳しくは次の「ペース歩行について」をご覧下さい。

ペース歩行について

足の運びは図１のような物で、つまり右側の足２本ずつ、左側の足２本ずつを同時に動かします。しかし、この足の運びだと、何も考えずに足を動かしても、横転するのは火を見るより明らかですね。感覚的に見てもペース歩行はバランスが悪い訳ですが、もう少し科学的に考えてみましょう。

図２を見てください。これは安定性に富んだトロット歩行という歩行法の足の運びです。◓のマークは重心の位置を示します。どの瞬間にも重心は接地部分の間を結んだ斜線部分の中に有るのが分かります。この歩行法だと平面上を外部からの衝撃も無しに歩いている限りは転倒する事は有りません。マク○ナルドのハッピー○ット等について来るチープな玩具も足の設置面積を反則的に大きくする事によって２足歩行を実現しています。

図１

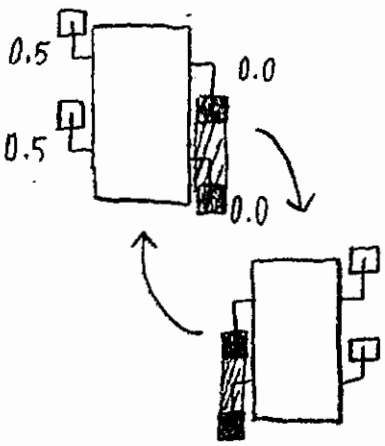
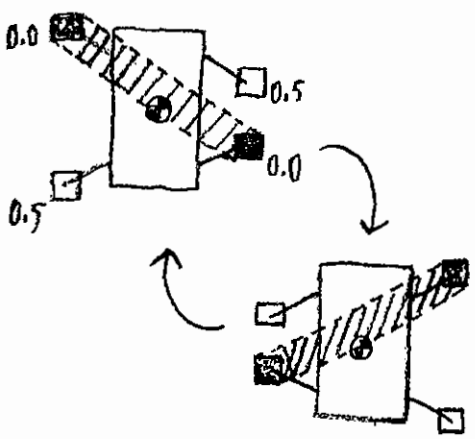


図２



しかしペース歩行は重心が接地部分の間を結んだ直線内には収まりません。その為片側の足を上げると転倒するのです。これを解決するにはどうするかと言うと、重心を移動させるしか有りません。という事でこのロボットは背中を左右に揺すって重心を移動させます。しかし、それだけでは不安なので、別の策も講じる事にします。足自体を体の内側に持っていくのです。そうすると重心と接地部分の距離は短くなります。これと背中の揺すりを合わせれば、重心を接地部分同士を結んだ枠内にギリギリ収める事が出来るかどうか、くらいまでには持っていく事が出来ます。

何故、「収まるかどうか位」で妥協するかと言いますと、ここまでの話では勢い、つまり慣性の法則というものを無視しているからです。慣性の法則は有名過ぎて今更事細かに説明する必要は無いでしょうが、走ってる車が止まれずに、ボールを取ろうと飛び出してきた子供を撥ねちゃうアレです。兎に角それを計算に入れるとＺＭＰ（ゼロ・モーメント・ポイント）という物が出ます。実の所は、重心は接地部分外に有っても、このＺＭＰという物さえ接地部分内に収まれば良いのです。もっと言えば実際の動物は走る時、どの足も地面につかない瞬間まで有る訳なので、別段不思議な話では有りませんね。

ちょっとした技

基本的に私は本体製作と重心云々が担当なので、わざわざ載せるべき回路も無いので、ちょっと重心計算の仕方を紹介します。条件は均一な材料で出来たパーツとします。ちなみに計算は紙等に部品の三面図を描き、そこに重さを書き込んでいく感じで行います。

①先ずは重さを量ります
②次にパーツの全ての部分を直方体やら円柱やら兎に角キレイな形（重心が分かる形）ごとに区分けします（図４）
③区分けしたブロックごとに寸法を求め、体積を出します
④①と③からブロックごとの重量を出します
⑤ブロックの重心部にそのブロックの重量が在ると考え、重心同士の距離を重さの比で分けた点に、両ブロックの合計重量が在る事になる（図５）
⑥⑤を繰り返し重心を一点にまとめて行く

大体こんな作業を延々と続ければ全体の重心も出ます。ただし、モーターや可変抵抗のような部品ともなると、正確な重心は出ないので、そこら辺はアバウトになってしまいます。

図4

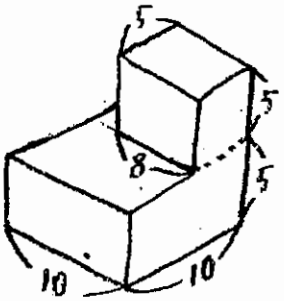
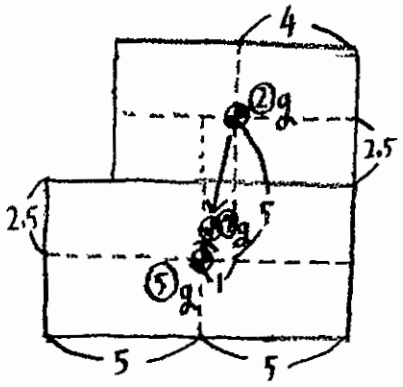


図5（正面図）

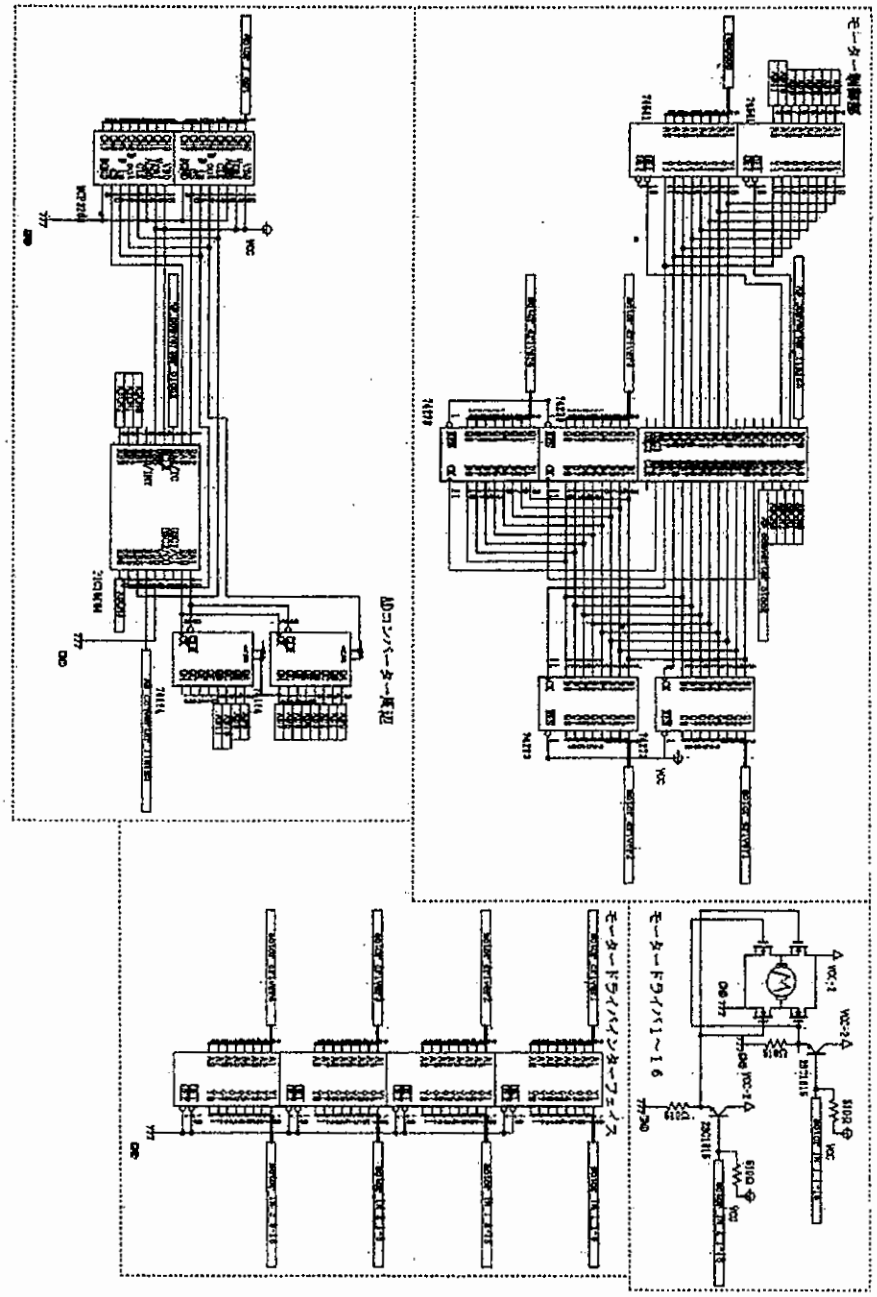


その他感想など

本当はちゃんとＺＭＰを出したかったんですが、それに必要な加速度センサーという物が何故か手に入らず、（かつて売っていた店が置かなくなったせいなんですが）しかも重心を計算してみたら「実はこれ、ＺＭＰ求めるまでも無く立つんじゃねぇ？」という結果が出てしまった為に加速度センサーは使わなさそうです。２足への布石のはずが・・・（泣）そうは言ってもまだちゃんと歩いた訳でもなんでも無いので、さっさと動かしたいですね。何せこのロボには確か４万円弱の金が投資されてますからね。責任重大です。

ここからは共同制作者の竹島がお送りします。

→　回路図



- 回路説明
  きわめて醜い回路ですね(w。あとVSSは全部5VにVDDはGNDにつながっていて、16F84Aの発振子は4Mhz,16F877は20Mhzになってます。まずはラベルの説明から。
- ラベル説明
  - ADコンバーター関連
    - AD0～11
    - 12bitADコンバーターの出力12bitです。ただシリアルなので74LS164でパラレルに変換してあります。また、この回路を作るにあたって「持続可能な回路」というどこぞの国連で採択された協定のようなことをモットーとして汎用性の高い基盤を目指したため、別基盤になっているわけですね。
    - ADCH0～3・AD_converter_clock・AD_converter_finish
      言うまでもなくAD_converter_clockに立ち上がりが入るとADCHで指定したチャンネルのAD変換を開始して終了するとAD_converter_finishをHighにします。
  - モータードライバ関連
    - motor_driver_0～3
      モーターのオン・オフを切り替えます。motor_driver_0・1が1ビット目、motor_driver_2・3が2ビット目になってます。
    - motor_IN_1_1～16・motor_IN_2_1～16
      motor_driver_0～3をただバッファに通しただけです。先にはそれぞれもタードライバ１～１６が計十六個付いてます。74LS541には3ステートの役目すらしてもらってません。motor_driver_0～3を出してる74LS273にし慣れちゃ困るからはさんでるだけです。またここも「持続可能な回路」ということでモータードライバ１～１６とともに別基盤になってます。
  - etc
    - remocon
      まだついてませんリモコンがつく予定です。
    - motor_r_out
      モーターの軸に取り付けられた可変抵抗の出力電圧です。
- 回路説明
  - ADコンバーター
    さて本格的な回路の説明ですが、このADコンバーター部分はMicrochip社のシリアル12bit8chADコンバータMCP3208を使

用しています。こいつは12bit8chにしてはシリアル通信なため秋月電商で400円と比較的安く、さらに誤差は最大で最大電圧の $1/2^{12}$ つまり最大電圧5Vの場合

$5.00[V]/2^{12}=244\times10^{-5}[V]$ 角度にして、

$1.5PI[rad]/2^{12}=13\times0.6PI^{-2}\times10^{-1}[rad]$ しかないため使用しました。(ちなみに同じく8chであるが分解能が8bitなADC0808はマルツ電波で2000円程度。店の違いもあるんだろうが何なんだろうねこの差は・・・) この回路はAD_converter_clockが立ち上がると、ADCHで指定されたチャンネルをPICがMCP3208に送り、その後MCP3208と16F84Aに交互クロックを送りシリアルをパラレルに変更、そしてAD_converter_finishをhighにする。という動作をします。

- モータードライバ

さてここがこの回路で一番時間を食った部分であり、一番完成度が高い部分です。この回路はHブリッジと呼ばれモーター制御の世界では一般的な回路です。さて僕はこの回路を親の本棚にあった「DCモーターの制御回路設計」(CQ出版、コアブックスシリーズ　今は新版が出てるらしいですが)で知ったので最初はバイポーラトランジスタを使おうとしていたわけです。「人に頼んで買ってきてもらっても希望の物がくる可能性は低い！」と思った僕は自分で秋葉原に行きひたすらトランジスタとにらめっこしたわけですね。バイポーラトランジスタでHブリッジを作る場合NPN型とPNP型両方2つづつ必要なわけです。しかし、PNP型はいいのがあるのですがNPN型はなかなかよいのが見つからず最後品揃えはいいが全体的に値段の高いマルツ電波に行くと2SK1086という耐圧60V耐電流20AのFETが10個入りで1500円ぐらいの値段で売ってました。FETなら1種類を4つでよいので、結局それを80個ほど購入。そして、モータードライバを16個作ったわけです。そしてモータードライバの製作はひとまずおいておき、その他の部分の製作に移ったわけです。

その後モータードライバICを作っていた川崎くんがFETの使用を決意し同じ回路を作ったわけですが、じつはLSICの出力ではこのFETは感知してくれないどころか過電流でLSICを殺してくれるとの情報が入ったので急遽モータードライバインターフェイス部分 (ちなみに16個書くのが面倒くさいため書いていないがを製作motor_INが入っている2SC1815二つもインターフェイス部分)を製作。まあもちろん2SC1815は電流増幅用、プルダウン抵抗510Ωは実験の結果これが一番だっ

たためこの抵抗値。プルアップ抵抗は510Ωが余ってたから。よって1Kとかでもイイっぽい。FETの間に秋月で購入したペルチェ用熱伝導両面テープで貼り付けたアルミ版をはさみこみ、その上にPentiumMMX用のヒートシンクをつけています。(ペルチェ使いたかったな・・・)付けないと煙噴いて再起不能ぐらいにはなってくれます。現に川崎くんが36個殺して実証してくれています。あと、Hブリッジの仕組みは説明すると長いんでhttp://google.co.jpで「モータードライバ」って検索すれば出てきます。

- プログラム説明

これもADコンバーターと本体に分けます。コンパイラはCCSコンパイラです。

- ADコンバーター

こっちが簡単なんでこっちを先に。

① ADコンバーターを両方オフにするために、CS1,CS2を両方highにしておきます。
② ad_converter_clockが立ち上がると割り込み。
③ ADCHの4ビット目を読み、1だったらCS1、2だったらCS2をlowにする。
④ ADコンバーターにhighを出力する。
⑤ ADCHの3～0ビット目をADコンバーターに出力する。
⑥ 2回クロックを送ってNULLビットをやり過ごす。
⑦ ADコンバーターと74LS164に交互にクロックを計12回送る。
⑧ ad_converter_finishをhighにする。
⑨ ①に戻る。

- 本体

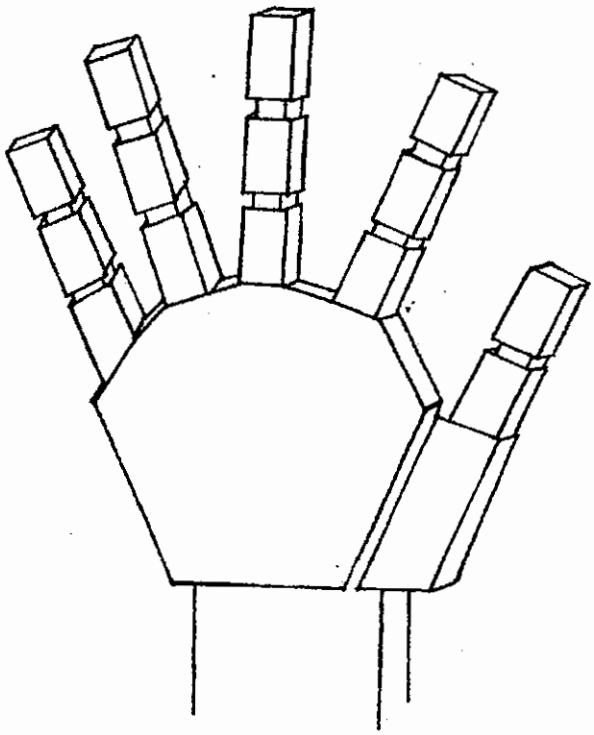
① remoconの値を読む。
② 進行方向決定
③ ADコンバータの値を読む。
④ 目的角度と比較する。
⑤ 全間接においてそれを行う。
⑥ 一歩進んだら①に戻る。

これで終わりです。やっと終わりました。
トランジスタはいい物です。
本もいいです。
特にCQ出版のコアブックシリーズはとてもいいです。
2006/4/9 竹島　啓純

アルミ腕アトム

製作者 M3 川崎敦史
協力者 物理部無線班の皆様、OB 様

～概観図～



～説明～

スイッチで様々な手の動作をさせることができます。
動作箇所は9箇所でそれぞれの指を曲げることと、親指は折り曲げ、人差し指、薬指、小指を左右に動かすことができます。

～構造～

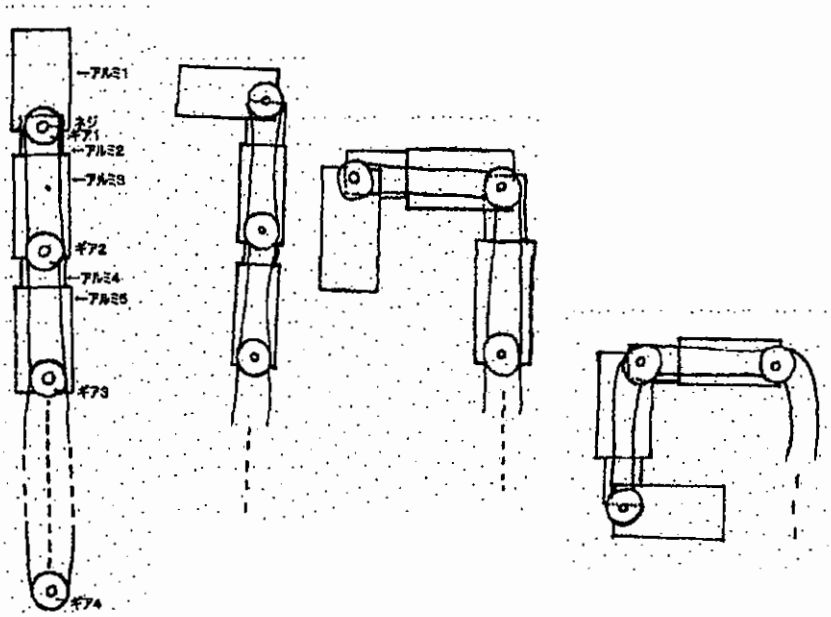


図1　図2　図3　図4

ギア1をアルミ1とネジを、アルミ2とアルミ3を、アルミ4とアルミ5を固定する。そして、ギア2、ギア3は自由に回転できるようにし、ギア4はモーターに固定し1つのチェーンに通す。

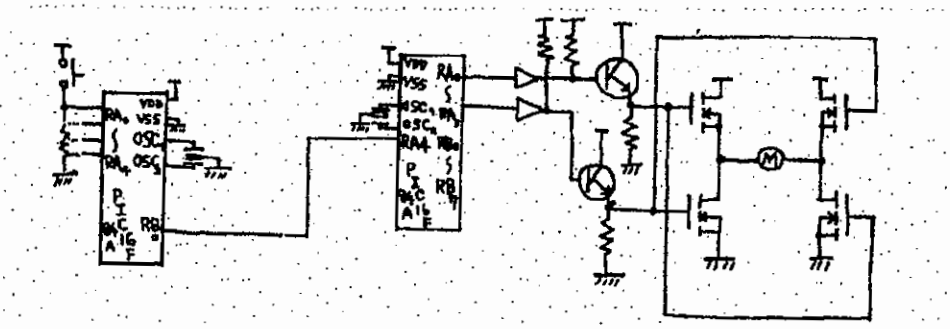
＊手を閉めるとき

図1からモーターを左に回すと同時にチェーンも回り、最初に固定されているギア1が回転し始める。図2のようにある一定の場所で止まると、次に力は内側に働きそれにあわせてギア2ギア3を使い図3、図4のようになり図4の場所で止まるようにする。

＊手を開くとき

原理は先ほどと同じでモーターを右に回すとギア1の部分が始めに動き一定の場所で止まると、今度は外側に力が働きギア2、ギア3を使い指は開き図1の場所で止まるようにする。

～回路図～



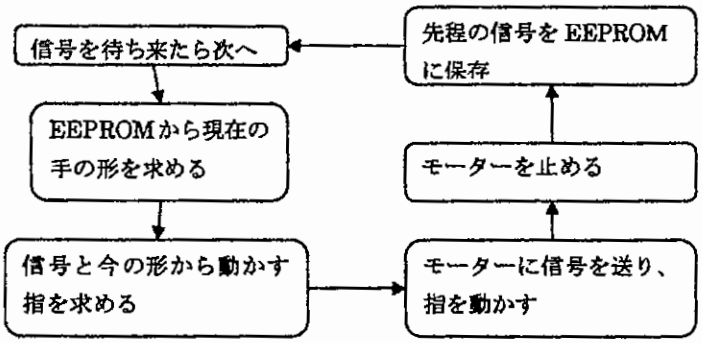
スイッチ用回路　　制御回路　　モーター制御用回路

＊仕組み

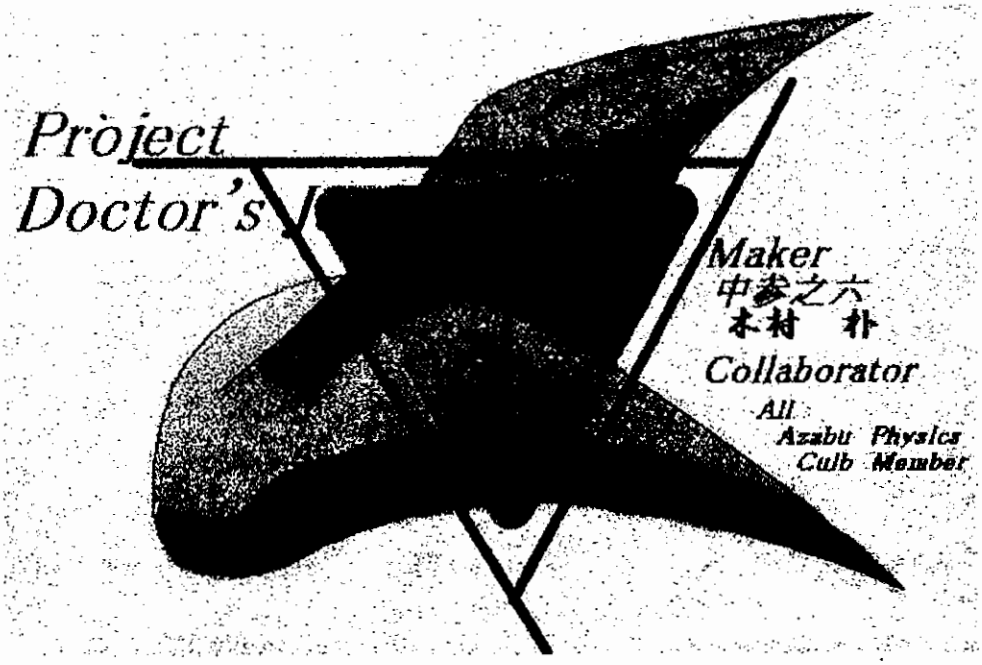
スイッチ用回路の PIC からモーター用回路の PIC に信号を送り RA0～RA3,RB0～RB,7 の2PIN ずつ使い出力させる。出力はバッファ、トランジスタ（Ｃ１８１５）を通し電流を増幅させる。増幅させた出力は FET (2SK2661) を上の回路図通りに配線する。L, L をモーター制御回路に入力すると(H,H だと FET が死んでしまいます)、モーターは止まる。H,L を入力すると H を入力された斜めの FET 同士のドレインとソースがつながりモーターは回る。L,H を入力すると同じように H を入力された FET のドレインとソースがつながり今度は逆回転するようになっている。（これは4月前半のときの回路図なので変更される場合があります。

～プログラム～

今現在プログラムがまだ定まっていないのでフローチャートを書きます。すみません。

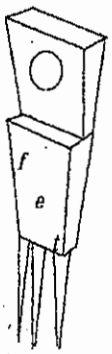


医者の息子が
よく持っている**バイアグラ**と印刷された三色ボールペン



-caution- この制作物は体に開いている穴でくわえるためのものでは断じて御座いません。

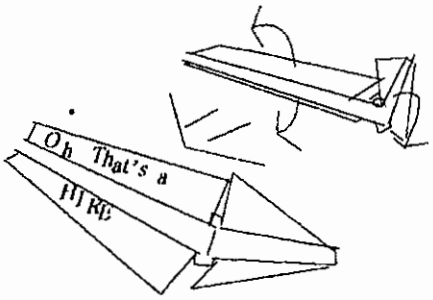
この製作物の基本コンセプトは、二つの鰭によって水を掻く事によって水中を自在に泳ぐというというもので、鰭の仕様は企画段階から縦に帯状の鰭を動かし、推進力を得るものから、横に掻いて推進力を得るものとなり、さらに其れを大型し、改良を加えるなど、数度モデルチェンジを経ています。

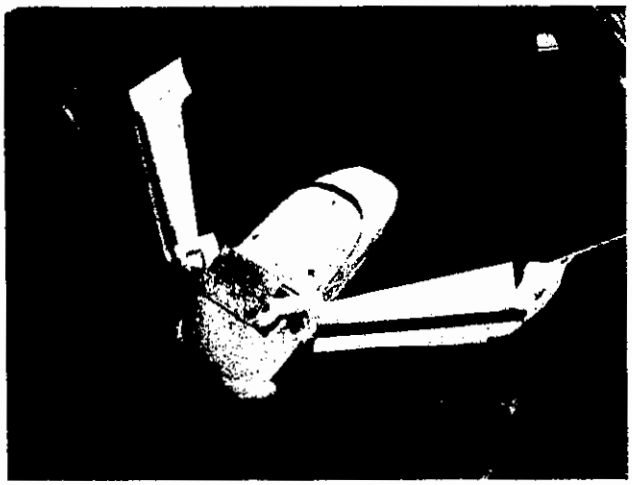


主な動力源はギアボックスによってギア比を極度に上げたマブチモーターの FA-140 で、Pic16f84a と FET(左図)によるモータードライバによってモーターの正回転、逆回転を制御します。そして、正回転、逆回転を交互に繰り返すことにより、鰭を動かし、水を掻いて前に進みます。Fet とはゲート、GND 間にかける電位差の大小によってドレイン、ソース間の電流を制御するものです。基本的な動作

はトランジスタと似ているのですが、構造がまったく違うために、大きく違う部分もあります。まずPNP型のトランジスタは、ベース、GND間に入れた電位差以上の電位差はエミッタに出力することは出来ません。
Fet は電圧の大小のみで出力電流を制御できるため、トランジスタに比べ、大変効率が良いといえます。

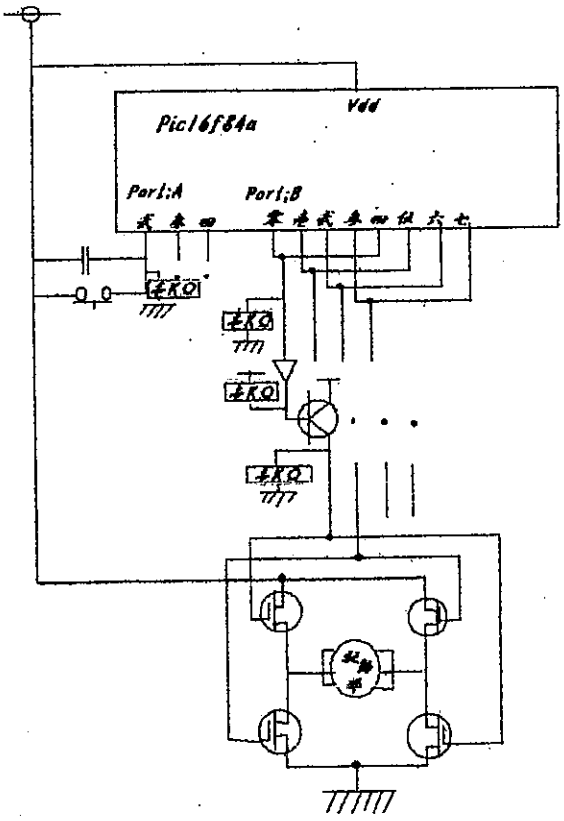
鰭の機構はどのようになっているか説明すると、前に鰭を押し出す時には
水の粘度により。水の抵抗を抑える状態へと変形し、水を掻くときには、水の粘度により鰭がおされ、変形し、より多くの水を掻ける状態へとなるように設計されています。変形機構は、針金と、螺子によって作られています。





概観は極力水の抵抗を下げつつ、揚力を得る形になるように設計したものですが、実際に揚力を得ることができているかどうかは不明です。
一般的に最も水の抵抗が下がっている形というのは、半球のようなものが前で先に行くほどすぼまってゆく円錐がその後ろに付いている物とされています。横についている突起は、鰭を縦に動かして泳がそうとしていたときの名残であります。之は、バキュームフォームという技法やパテ細工やアルミの金属加工などを用いて製作されています。
その他、全体的に突起を少なくしようと努力しているのは、(あからさまに先端のほうに邪魔な名残があるが)無駄な突起があるとそこに水の流れの渦が出来てしまい、其れが前に進む時に大きな抵抗となってしまう為であります。
(展示時の形状と、写真とは大きく異なっている場合が御座います。予め御了承ください。)

回路図は右図のようになっております。
Pic16f84a のソフトウェアを変更するだけで大きく動作を改変できるように、極力、回路を簡略化しました。
鰭を正回転、逆回転で制御するようにしたのもソフトの変更により、一度クランクで作ってしまった場合には改変できない部分も改変できるようにしてある為であります。あと、態々ポートBの零ビットと四ビットが繋がっているの様なものは 84a の出力が壱ピン 20mmA までの為、弐つ繋げる事により、40mm まで出せるようにする為であります。(実際に展示して在る物は大きく回路が改変されている恐れがあります。また、この通りに作ったとしても、動作する保障は御座いません予め御了承ください。)



尚、現在は実装されていないのですが、無線でコントロールされているとすれば、去年の回路図集と、四年前の回路図集を参照していただければ、FM によるラジコン制御と言う形で載っているので参照してください。(おそらく実装されてないでしょうが)

あとは製作開始段階においてちょろっとやった流体力学についてでも、

壱、飛行機は流体力学によって成り立っているッッ！
一般的に飛行機の翼は上の面が長く、下の面が短いことによって、
気圧の差が生まれ揚力を得るとされているが、実は其れだけではないのだぁ
詳しくは専門書で

弐、　傘を開いた上体で傘を盾のようにして猛突進したとき、抵抗を感じるのは
傘の前にある空気のせいではないッッ！
実は傘のふちにおいて空気の渦ができて其れが傘をうしろへ押し戻そうとしているのです。よって、その渦さえなければ傘を開いて突進したほうが、傘を閉じて突進するより抵抗が少なくなるのです。（音速を超えるとはなしは別。音速を超えると逆に先が尖っていた方が空気の抵抗が下がります。）
例;競輪のヘルメット

参、実は流体力学は大学物理の範囲なのだぁtjkl;fsd！
はい、厨ヴォウに理解できるわけがありません

というわけで今回は泳ぐロボット（鰭で）

という事でしたが、当日は元気に泳いでいるのでしょうか

実際、現在、鰭の稼動部の防水で困っています。

うはwwww、動いてるといいなwwwww（現実逃避）

Twenty-six

April eight

At midnight

BUTURIBU MUSENHAN

設計・製作　M3中野晃平
協力　物理部無線班の方々

TITLE: Monocycle2006

概要・構造。

この製作物は、今までの趣向とは一味違う物を目指して作られたロボット?です。

とある大学で作られていた一輪車ロボットを簡素化し、タイヤの中に回路を詰めたような造りになっています。また、内部に加速度センサーを取り付け、それによって重力加速度を計測し、傾いただけ水平に戻すように錘を移動させて水平にさせる機構を搭載し、そのモーターを遠隔操作で意図的に動かすことで傾きを利用して曲がることが出来るようにもなっています。

では基本的な構造をここで説明していきたいと思います。回路図や部品については後ろのほうで説明していきますのでそちらを参照してください。

基本的な構成として、この製作物は概観のように、外殻の中に一本の固定された軸を通しそこに歯車をつけ、モーター側に付いた歯車とチェーンを噛ませ、固定されていない回路とモーターを内側で回すようにします。ここで内側の回路が重たくなれば、回路が内側で空回りせずに、外側が回り、地面を転がる構造となっています。

回路についてですが、この回路には主に加速度センサー、A-D コンバータ、PIC16F84A を使っています。

二軸加速度センサーは本来、重力加速度を計測し、どのくらいの速度で走っているのかをアナログ信号で出力するものですが、この場合は地球の重力を計測し、そのときに出力される値が角度によって違うことを利用しています。

PIC16F84A は A-D コンバータを内蔵していないので、別の A-D コンバータを使用します。今回使用したのは ADC0804 という型番のものです。これに加速度センサー(以下、ADXL311 と呼ぶ)から出力される値を入力してそれを 8 ビットのデジタル信号に変換して出力します。ちなみに今回の回路では左右の傾きしか計測しないので、使用するのは X 軸の値のみです。

ADXL311 は水平にして何もしない状態での出力は約 2.5 ボルトですが、傾けることによって 2～3 ボルトの信号が出ます。このため ADC0804 の基準電圧は 2 ボルトに設定しています。出力ピンは 8 ビットですが、今回は細かい測定を必要としないので、XXXXXXXX といった出力のうち 4 ピン目と 5 ピン目について測定すればいいのでそれを PIC に入れます。そしてプログラムを通してモータードライバに電流を流してモーターを回して傾きを補正します。

4月7日現在の進行状況

現在、プログラムは完全に動いており、コントローラ、赤外線部分、傾き感知部分も完全に動いています。残っているのはモーター制御部分、つまりモータードライバとモーターのノイズ消しという課題が残っています。それが終わり次第外側の車輪部分の微調整に移り、時間があれば塗装して作業終了です。モータードライバが最大の難関でしょう。その後に出るノイズを消さないとA-Dコンバータが壊れてしまうのでそこも注意が必要です。

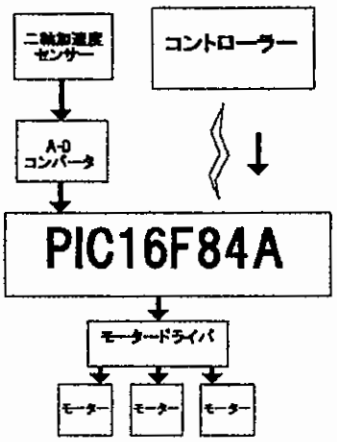
プログラム

プログラムについては大幅な流れを説明します。A-Dコンバータの出力、赤外線受光モジュールの出力をPICの入力につなぎ、出力をモータードライバの入力に繋ぎます

```
    ORG 0
    GOTO SETUP
    ORG 4
    GOTO 割り込みプログラム
**********************************
```

A-Dコンバータの出力を調べる。
何が出ているかによってどの出力PINをHIにするか決める。

プロック図



コントローラーは赤外線センサーを使用しており、本体のほうはPICの割り込みのプログラムで動かしています。これには傾きを計測する必要もあまり無いので回路は単純なものです。

製作物の説明はざっとこんな感じです。ここからは回路図とプログラムを書いていきたいと思います。

注意

人の神経の影響を受けやすい一種車型の製作物です。大事な試験機器には4,000Vもの高圧のストレスが容易に蓄積され検知されないまま放電されます。この製品は作者独自の理論かせる術を内蔵してはいますが、みなさまが高すぎるほどのストレスを掛けた場合、主役人体の損傷を生じる可能性があります。したがって、ストレスがたまるのを防止するため、割引に対する適切な予防措置を講じることをお勧めします

プログラム続き

A-D コンバータは前述したとおり PIN を二つしか使っていない。このためプログラムは BTFSS や BTFSC といった簡単な命令で済ませることが出来る。

```
 BTFSS PORTB,1
 GOTO MAEMIGI
 GOTO USIROHIDARI

MAEMIGI
 BTFSS PORTB,2
 GOTO MAE
 GOTO USIRO

HIDARIUSIRO
 BTFSS PORTB,2
 GOTO USIRO
 GOTO HIDARI

MAE
 BSF PORTA,0
 GOTO WAIT0.5sec
 GOTO MAIN

MIGI
 BSF PORTA,0
 BSF PORTA,2
 GOTO WAIT0.5sec
 GOTO MAIN
```

以下同様

基本的な部分はこれで完成。

PORTA,0 は前進させるモーターを表し、PORTA,2 は右側へ傾けるモーターへの出力を表している。次は割り込み部分のプログラムを説明します

****

割り込みのプログラムは赤外線が PORTB,0 に入った時に使用します。INT 割り込みを今回は使用しました。

このプログラムに関してはほとんど説明は要らないので簡単に言葉で済ませてしまおうと思います。同時にコントローラー側のプログラムも説明したいと思います。

まずは赤外線の送信についてですが赤外線の受光モジュールは一定の周波数の信号にしか反応しません。今回はその周期に合わせて前進時には HHLL という信号を送り、同様に後退、左折、右折時の信号も別々のものにして送ります。

受信側はこれを受けてモータードライバ側に信号を送ります。割り込み時には現在作業中の値を W レジスタから別のレジスタに移し、割り込みの作業を行います。

割り込みについての説明は以上です。

ちなみに WAIT プログラムは

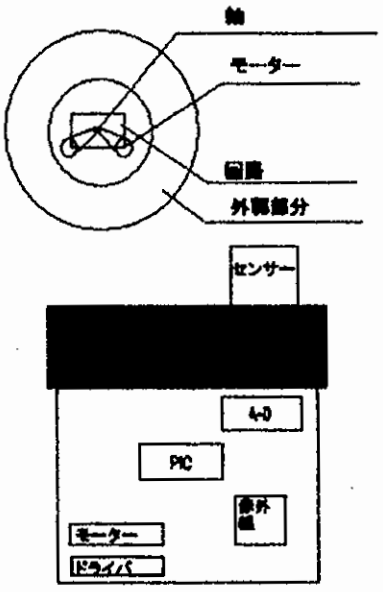
600 マイクロセカンド、300 マイクロセカンド

0.5 セカンドの WAIT をつくりました。

なんか

長くなってしまってすみません。

概要をちょこっと書きます。



だーつ

製作・回路設計
M3　大塚　雅人

協力者　　物置の方々

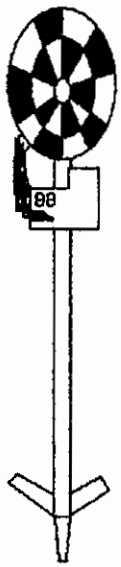
・ダーツとは…

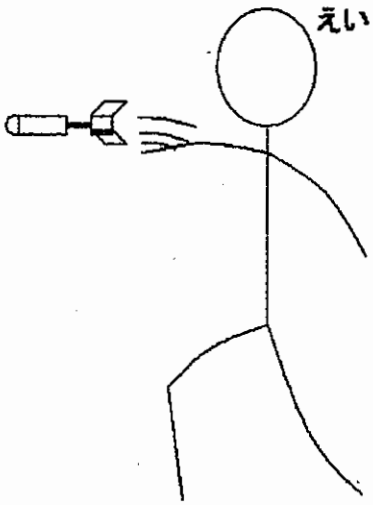
まあ知らない人はほとんどいないと思います。矢を投げて的に当てるあのかっこいいゲームです。

・このダーツは…

よく見かけるダーツは的に矢がささるのですが、このダーツでは安全性を考慮してささるのではなく、当たるようにしています（その結果かっこ悪くなりましたけど…）。

・外見…





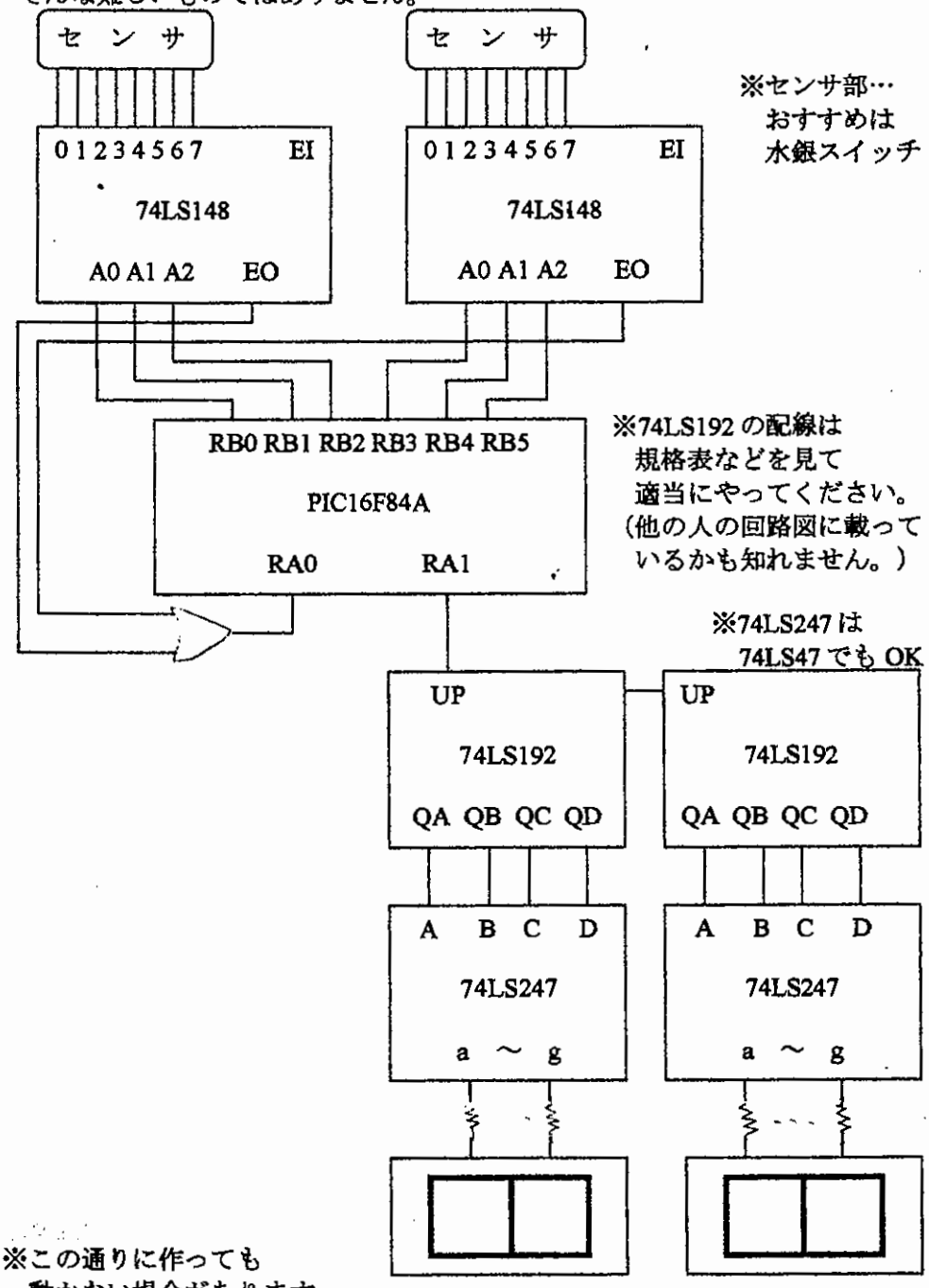
結構場所とりましたけど、まあこんな感じですかね。

・仕様…

このダーツではもっとも一般的な『カウント・アップ』というゲームで作ってあります。ルールは簡単。当たった部分の点数が加算され、その点数を競うものです。（もしこのようにダーツを作るとしたら、ほかの種類のゲームも作ってもいいと思います。）

・ 回路図…

そんな難しいものではありません。



※センサ部…
おすすめは
水銀スイッチ

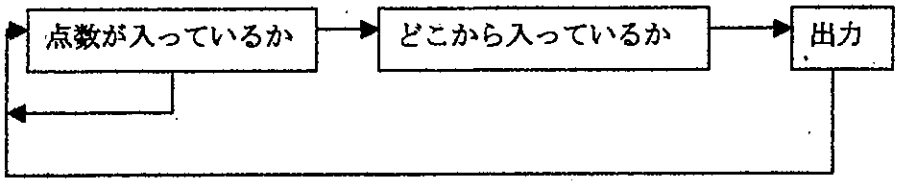
※74LS192の配線は
規格表などを見て
適当にやってください。
(他の人の回路図に載って
いるかも知れません。)

※74LS247は
74LS47でもOK

※この通りに作っても
動かない場合があります。

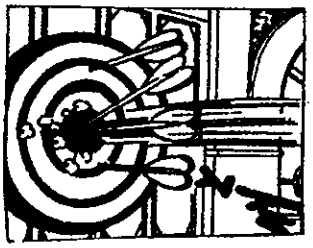
・ プログラム…

とても
長いため（８００行くらいあるかも…）フローチャートで。



・ 製作してみて…

思ったことは、ダーツは作るものではなく、遊んで楽しむものだということです。皆さんもダーツを見かけたらやってみてください。



回路の中の大迷宮

製作者　M－３　佐野祐士
協力者　物無の皆様

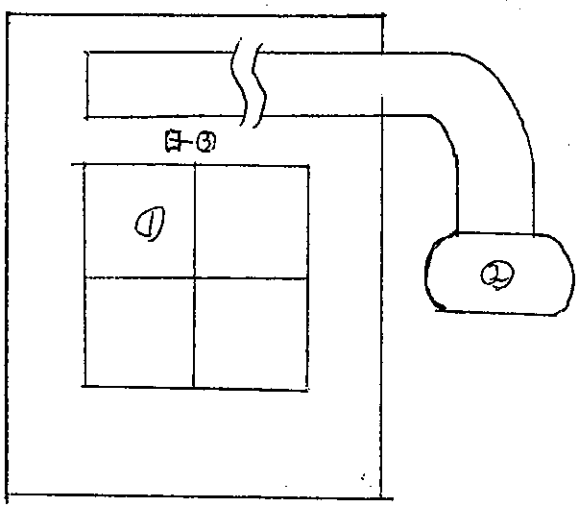
ストーリーと思われしもの

とあるところにピンク色の丸い物体（以下、アレ）がぐうたら平和に暮らしていました。ところが、いきなり空から長剣を持ち黒い仮面をかぶった剣士（以下、ソレ）が現れ、アレはソレに４つに切り裂かれてしまいました（ピンク、赤、黄色、緑の四色セット）。アレ達は元に戻るためソレが帰っていった回路の中の世界にワープするスターで顔面から突っ込んでいきました。らしいよ。

まあ、アレたちは馬鹿で頭が悪いから迷宮から抜け出せなくなったのです。テキトーに誘導してあげましょう。

え？外見と操作方法が知りたいって？

外見



① ドットマトリクス
② コントローラー
③ リセットスイッチ

操作方法

まず、リセットスイッチを押してゲームを始めてください。

コースは１面から８面まであります。どんどん難しくなっていきます。

移動はすべてコントローラーでやります。

このコントローラーは傾きを察知できるので傾ければ自機がうごきます。

自機は同じ方向に傾け続ければ速くなっていきます。

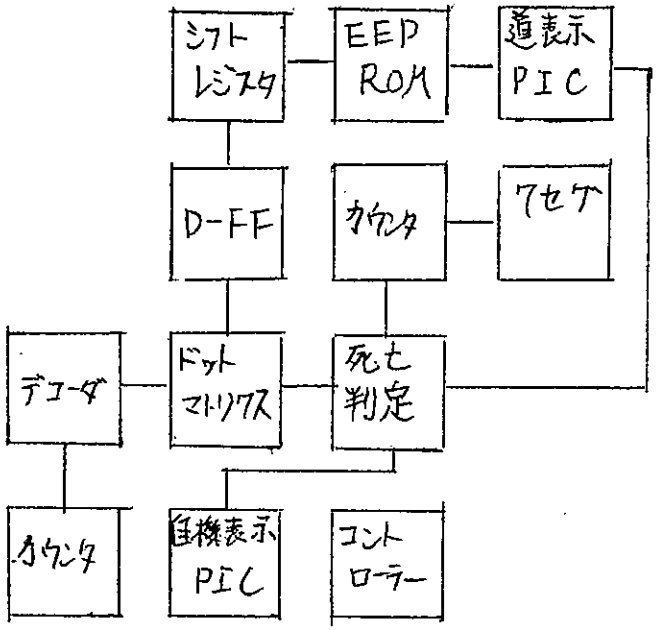
赤く表示されているのは道で、そこをそれると即死です。

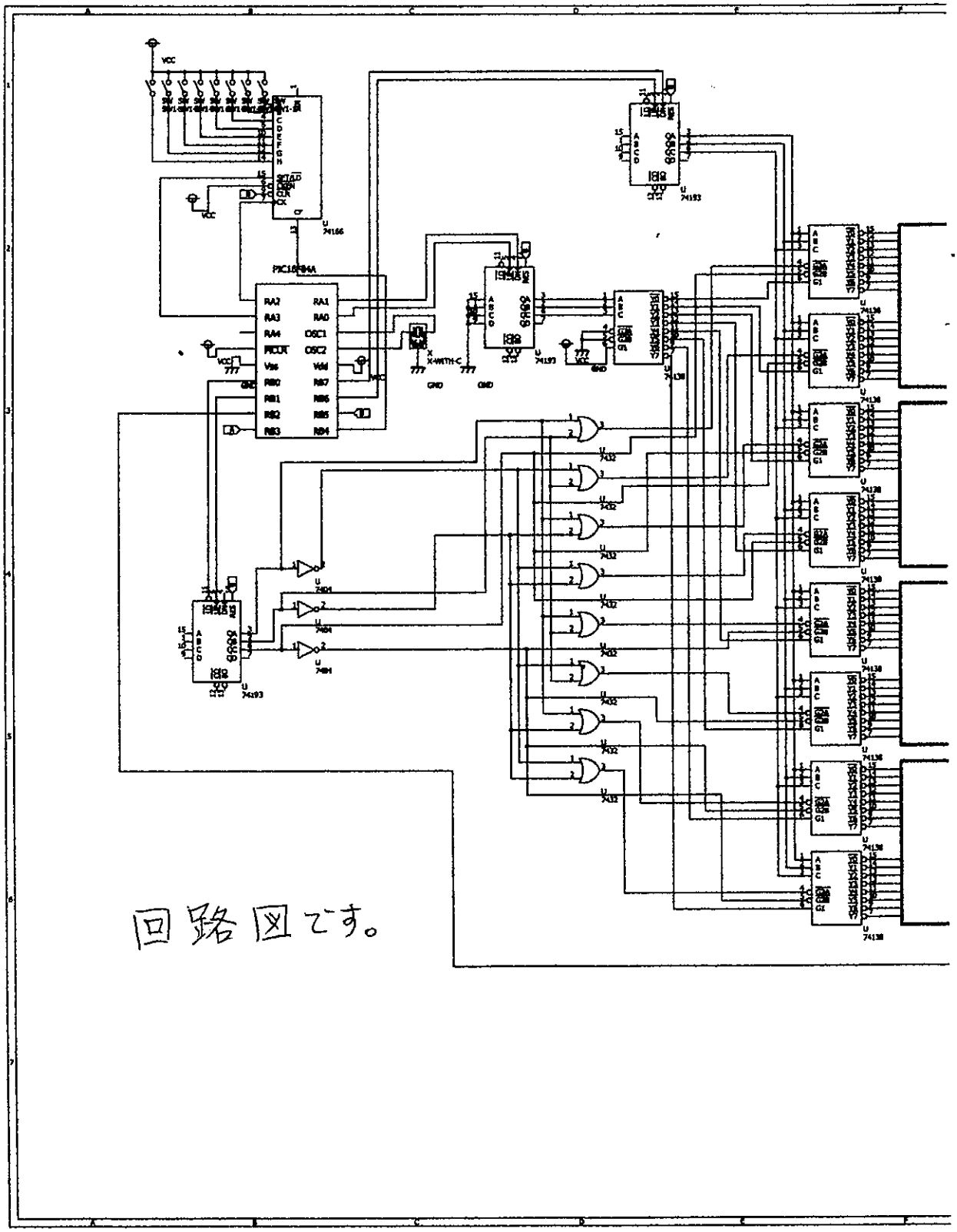
アレは見るも無残に死んでゆきます。

スタートは左下でゴールは右上です。ゴールに着くと次の面へいけます。

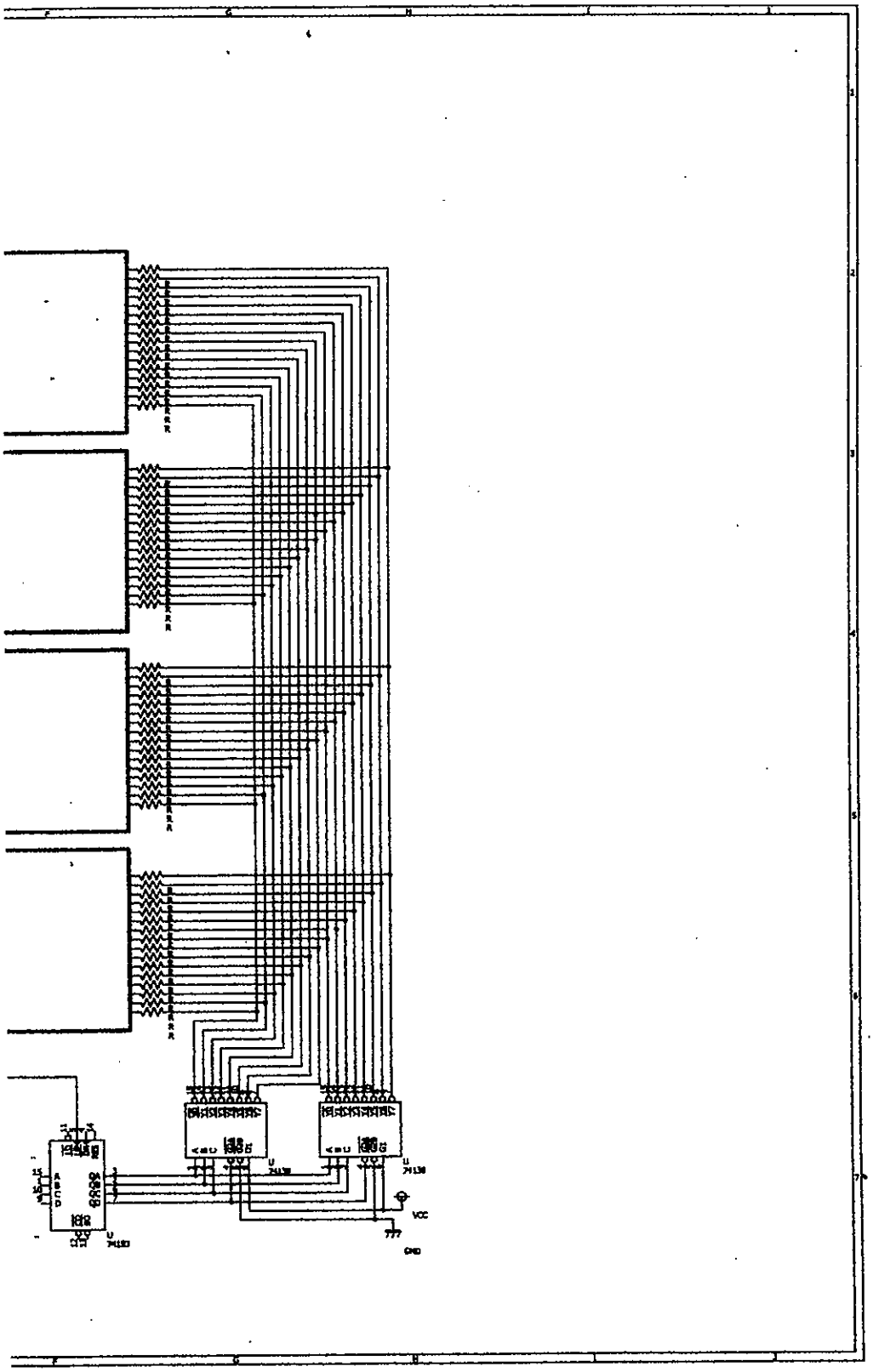
８面クリアしたら飛び上がって喜んでください。

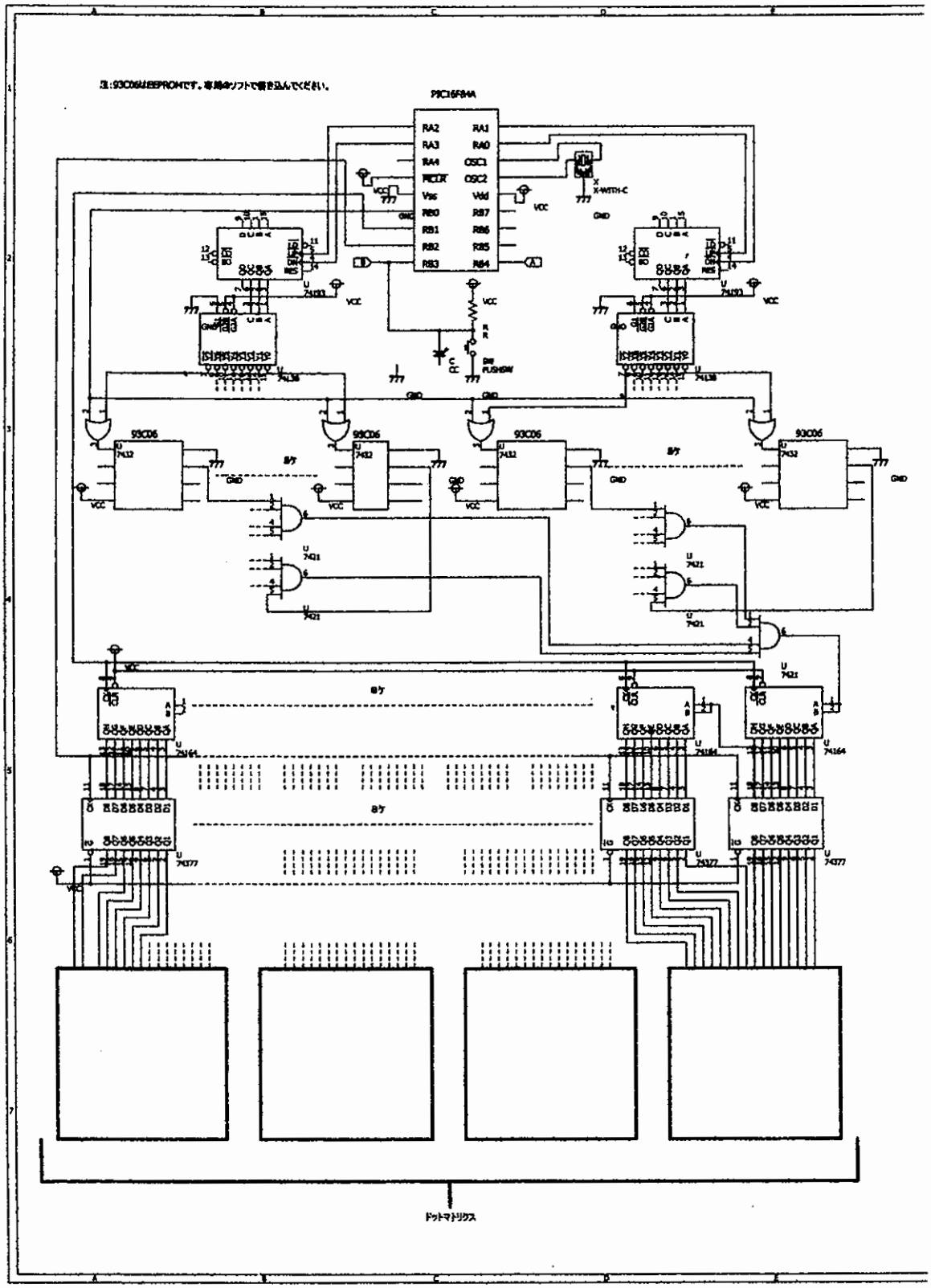
うん。ブロック図だってさ。

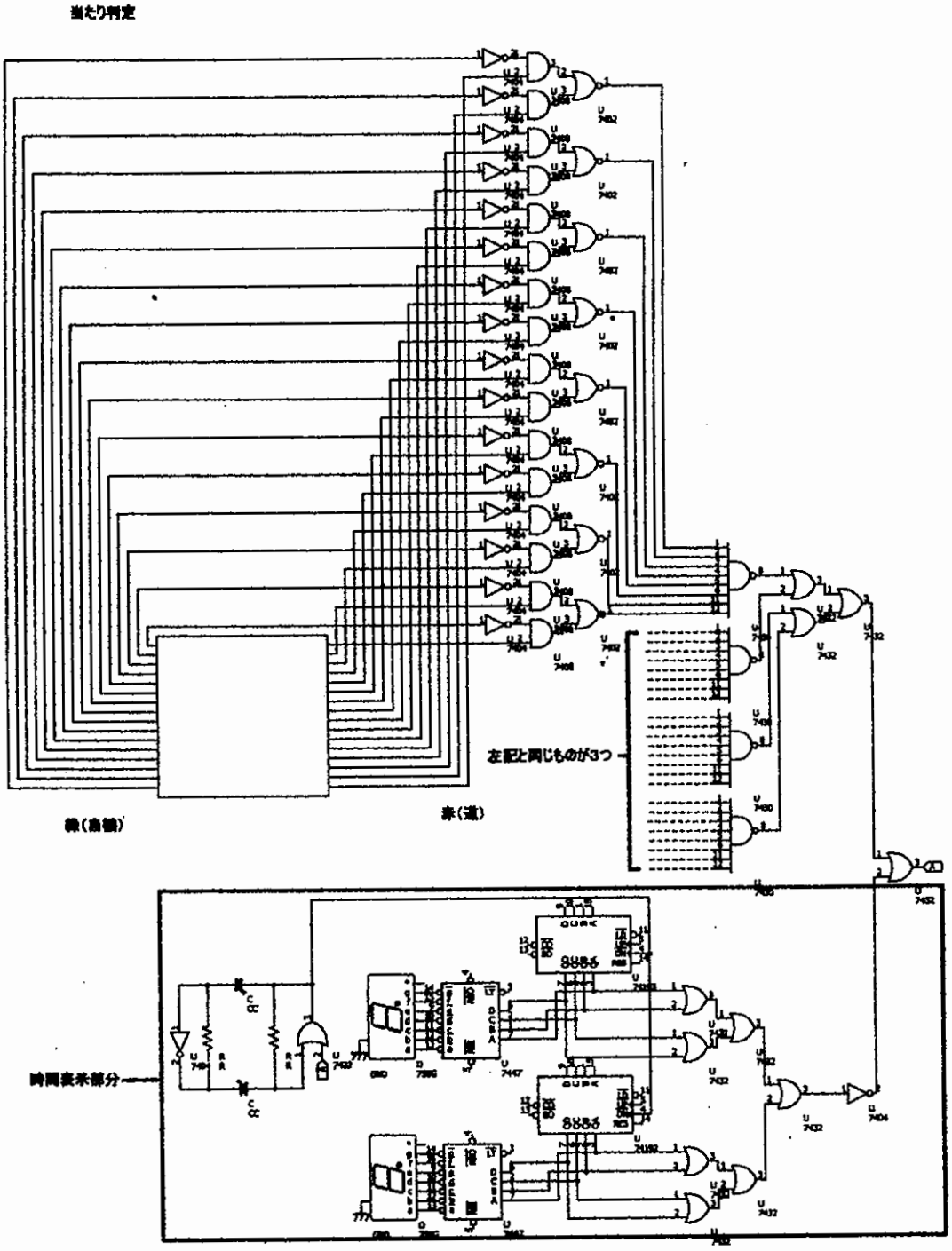



VCC
PIC16F84A
RA2
RA3
RA4
MCLR
Vss
RB0
RB1
RB2
RB3
RA1
RA0
OSC1
OSC2
Vdd
RB7
RB6
RB5
RB4
X-WITH-C
GND
74166
74193
74138
7404
7432

回路図です。




PIC16F84A
RA2
RA3
RA4
MCLR
Vss
RB0
RB1
RB2
RB3
RA1
RA0
OSC1
OSC2
Vdd
RB7
RB6
RB5
RB4
X-WITH-C
VCC
GND
93C06
74132
74138
74164
74377
74221
8ヶ
ドットマトリクス


当たり判定
左記と同じものが3つ
時間表示部分

Teacher 2

～帰ってきた先生～

製作者　M３　中嶋剛大
協力者　物無の方々
OB の方々

— 製作物について —

この製作物は、昨年の製作物である、先生-Teacher-を激しくグレードアップさせたものです。

昨年の製作物は、同輩の木村、川崎とともに作り、赤外線が当たると光って点数が入り、モーターで的が動くと言うものでしたが、残念ながらモーターは文化祭までに動かず、赤外線が当たると光って点数が入ると言うだけのものになってしまいました。

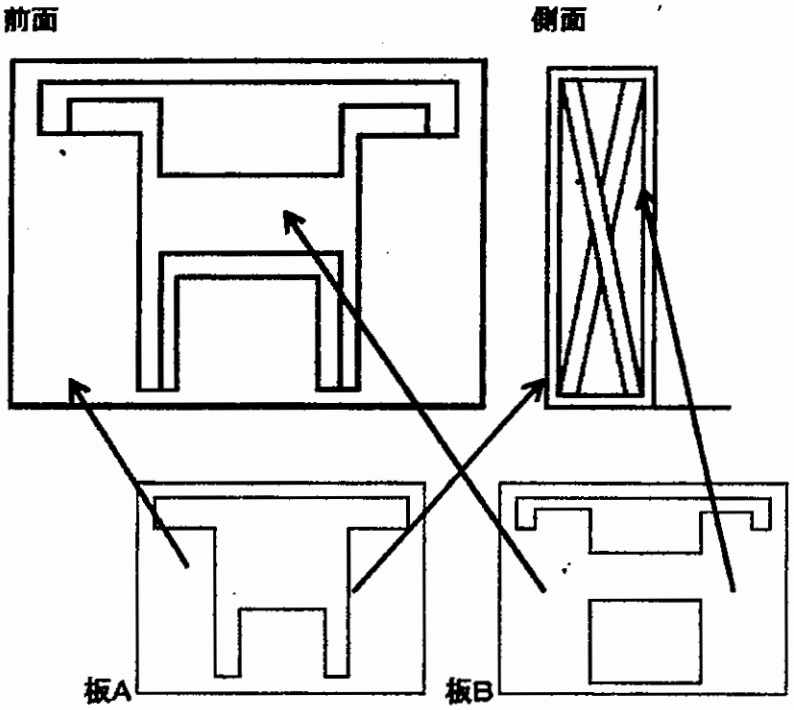
そこで、リベンジ？の意味も兼ねて作ったのが、このTeacher 2です。

ちなみに、タイトルに意味は無く昨年 Teacher 作っていたときに使った雑誌に、「先生！嘘だよな、先生！」という台詞が乗っていたため、というだけです。

ちなみに、4 月 5 日現在,
まだ完動していません。
非常にまずいです。
文化祭までには必ず、完動させます。頑張ります。

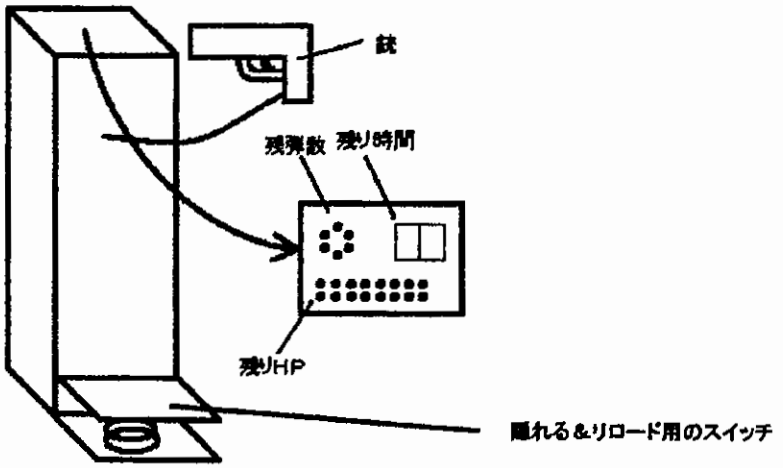
— 外見 —

本体(的側)



板Aが前面、板Bが背面。板Aには的が5つ、板Bには的が3つ付きます。

本体(銃側)

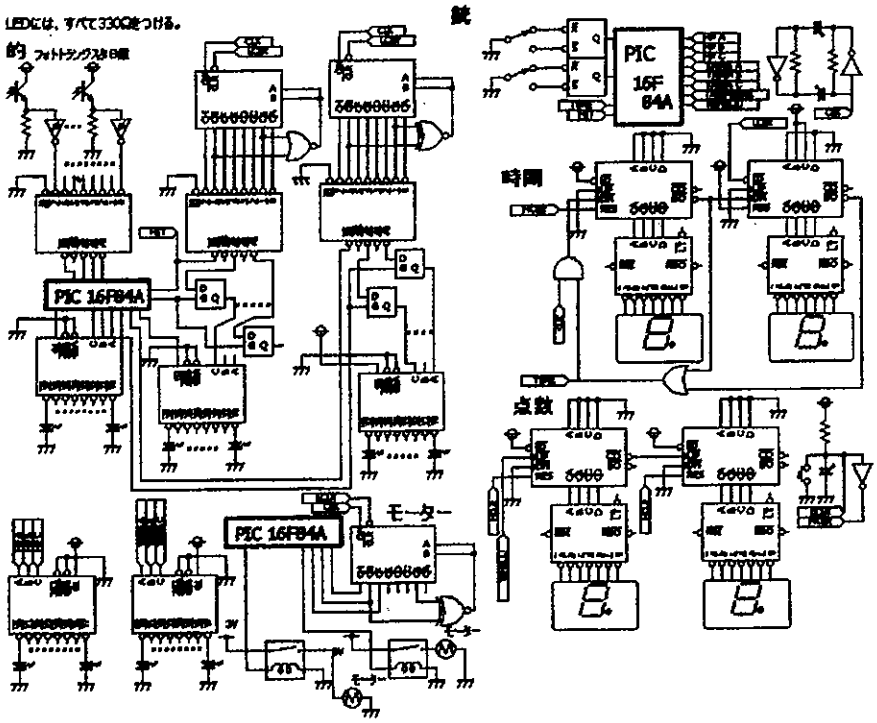


— ルール説明など —

的側の本体に、的が8つついています。それを銃で撃つゲームです。
銃には、レーザーが入っており、それを的のフォトトランジスタに当てることで HIT となります。また、銃の弾がなくなるといくら銃をうとうとしてもレーザーが出ません。その場合は、銃側の本体の下のほうにあるスイッチから足を離すことでリロードできます。
自分だけでなく、的も弾を撃ってきます。敵が弾を撃つ直前に、的の一部分が光ります。それが緑色の場合は自分に当たることはないので隠れる必要はありませんが、赤色の場合は必ず自分に当たるので、隠れないとダメージを受けてしまいます。それを目印にして、弾をよけてください。敵の弾に 8 回あたると、ゲームオーバーとなります。ちなみに、隠れるスイッチは、リロードのスイッチと同じスイッチです。
的に自分の撃った弾が HIT すると、点数が入り、的の外周にある LED が点滅します。
的はモーターで、左右又は、上下にランダムで移動します。移動すると、自分の弾は当たりませんが、敵の弾は当たるので、注意してください。

さぁ,時間が無くなるか、
HP が無くなるまでに
敵の弾を避けながら
いったい何発の弾を
的に当てられるか！！

— 回路図 —



— プログラム —

計画性の無さによって、PIC16F84A が、"銃制御用"，"的制御用"，"モーター制御用"と3つあるわけなのですが、簡単にプログラムの説明をします。

フローチャート書くほどのものでもないんで・・・。

"銃制御用"は

HP が残っているか→制限時間があるか→隠れているか→撃ったか

"的制御用は"

乱数が入る→LED が光る。（割り込み）レーザー受信→点数 UP→的光る

"モーター制御用"

乱数が入る→リレーのスイッチが入る

まぁ、これだけなんですが・・・・。やはり、無駄ですね。

モーター制御用なんて 7 ピンだけですからね・・・。

— スライドクランクについて —

この製作物の本体で使えそうなところはスライドクランクくらいしかないということで
スライドクランクについて少し説明します。
図で表すと図1みたいな感じ。C のところに的をつけるわけです。

図1

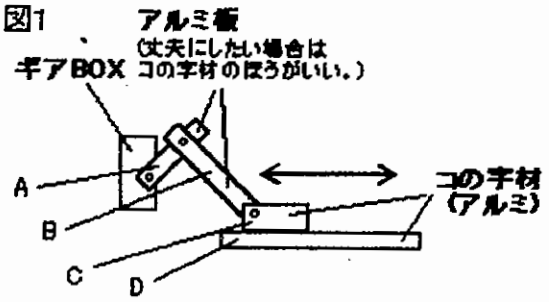


図2

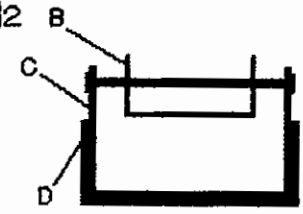
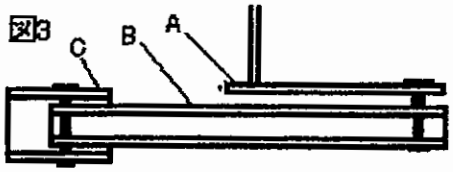


図3

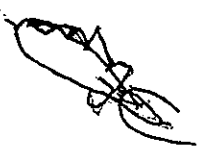


ギアBOXとAは動かないようにしっかり固定。AとB, BとCは、回るように少しゆるめに固定。CはDの中をすべる大きさに。Dは動かないように固定する。
単純だが、結構スムーズに左右に動いてくれます。C と D の位置関係によっては回りにくかったり、回らなかったりするので、注意して下さい。
※上下方向のスライドクランクは、純粋に機構を 90° 回転させるだけでは
Cのコの字材がDのコの字材から、外れてしまうため、外れないように
覆いをつけるなどの工夫をしなければいけないので、注意してください。
ちなみに、モーターを回すのには、1極リレーを使っています。
(一方向にしかまわさないので。)

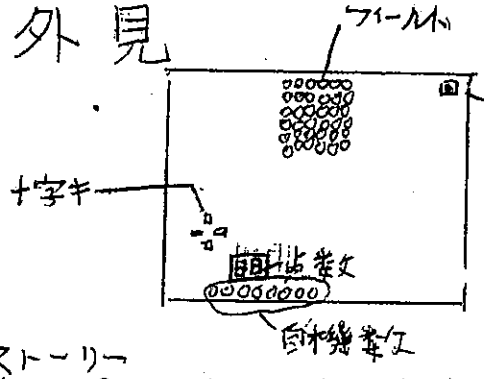
感想・・・ 企画書出したときは結構早めに完動するかなとか考えていたのですが
4月5日現在まだ完動していません。細部の設定をあまり考えていなかったり、ケアレスミスに悩まされたり・・・。意識が足りなかったようです。
なんとか文化祭までには動くよう頑張ります。

ていーぷいんくと

制作者 中二三菅原寛生
回路設計者さん 高二浅井さん
協力者さん 物凄の先輩方
及び同輩達



・外見



だいたい左のような感じになりますかね……？

・ストーリー

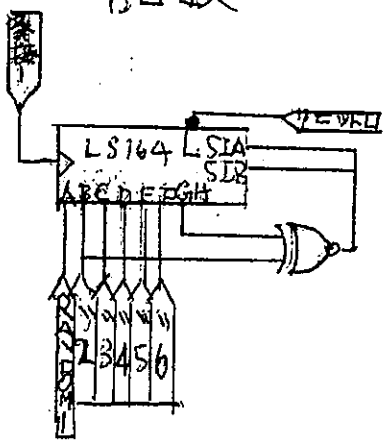
時は1976年ロッキード事件で大いに盛り上がったころ。どっかの星で……(仮にA星とでもしておきます。)なにかのまちがいで巨大隕石群がA星にやってきてしまいました。しかたないがらタマの止まらない機関砲をつんだオンボロ宇宙船が出発した……。

・ゲームのルール

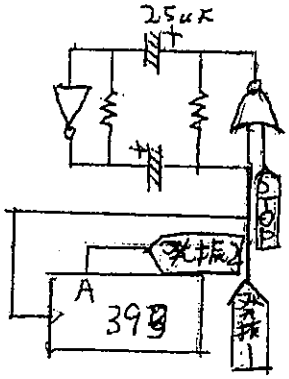
十字キーで自機を動かし自機から出るタマで隕石を破壊します。自機は3機。1コ隕石をこわすと1点入ります。(変更があるかもしれません)

・回路図

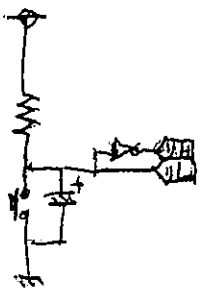
5L数



発振



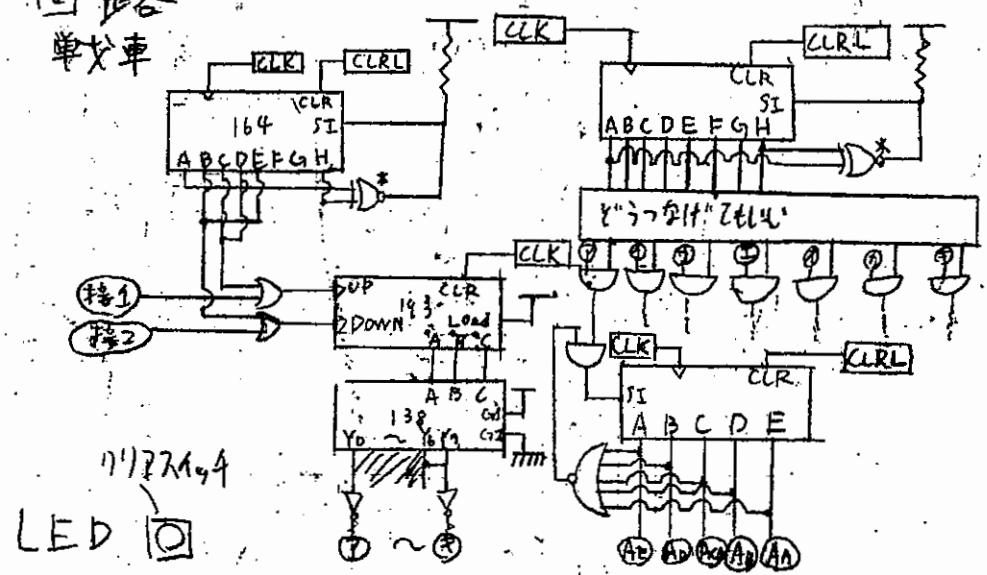
リセット



⑥ 説明

操作は、緑色に光っているLEDを、左右の移動ボタンで動かし、ゴールまで行くというものですが、途中で雷や、戦車(一番下の列の黄色)の弾玉を上げなければいけません。雷は、一番上の列がまず3回点めつし、その後戦車のLEDをのぞき、点めつした列すべてが光ります・また、戦車は木林(自分)のまわりを追ってきて、下から上に向かって玉(LED一つ)を打ってきます・ゴールをすると、「GOAL」と表示されますが、途中で死んだ(雷、玉に当たった時は、リセットボタンを押して下さい。

⑥ 回路



(AA)	(BA)	(CA)	(DA)	(EA)	(FA)	(GA)	(HA)
(AB)	(BB)	(CB)	(DB)	(EB)	(FB)	(GB)	(HB)
2色 (AC)	(BC)	(CC)	(DC)	(EC)	(FC)	(GC)	(HC)
(AD)	(BD)	(CD)	(DD)	(ED)	(FD)	(GD)	(HD)
(AE)	(BE)	(CE)	(DE)	(EE)	(FE)	(GE)	(HE)
戦 (ア)	(イ)	(ウ)	(エ)	(オ)	(カ)	(キ)	

※雷、戦車は、(H)の列には来ません。

※LEDには、330Ωの抵抗が必要ですが、ここでは省略します

←左右移動スイッチ

Fitece

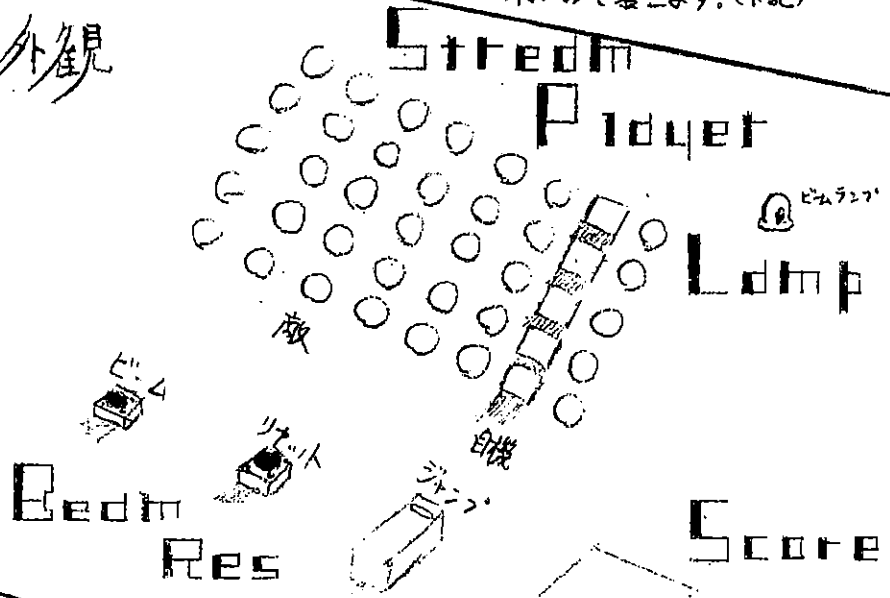
製作：m2 早川 瑠
回路：m2 西脇 知弘さん
協力：m2 浅井 政太郎さん

物語の方々..

解説

ストーリー？~~そんなモノ今度Flashで後期でもしない限り書かないでしょう。~~
というわけで避けゲーであります。
とてもシンプル。でも説明しないわけには
イカンので書きます。(下記)

外観



操作

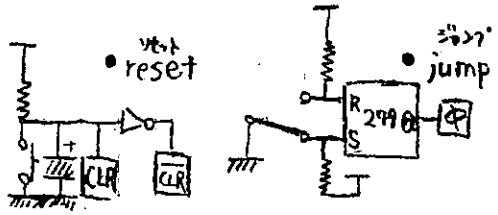
とにかく
ジャンプで避けろ！
スコアが溜まると
ビームランプが光り、ビーム
スイッチを押せば自機
前方の敵を撃滅、
爽快なコトに
なります。

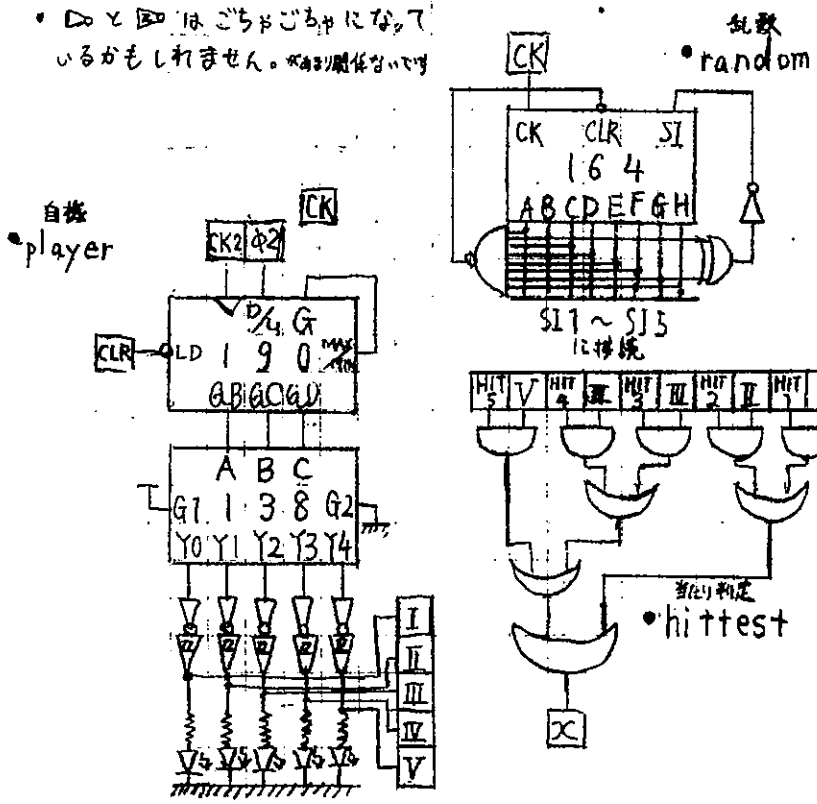
Stteedm

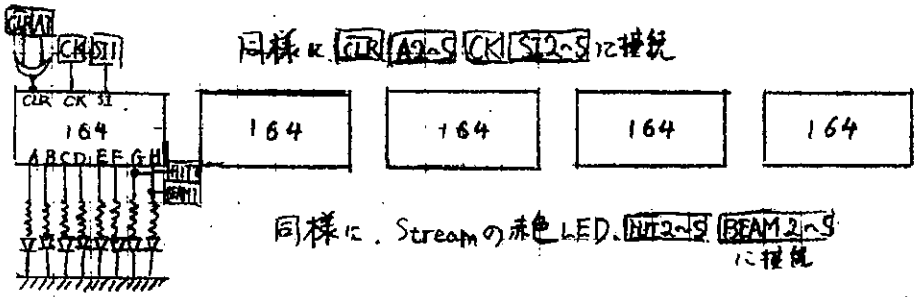
※Flashに関してはこの部と関係ないので気にしないでくださいね(汗

Firece Stream の回路

- 特に表記の無い場合は IC は 74LS シリーズです.
- ノイズ消し等は省略されている場合があります.
- 見にくいですか？(疑問形)
- □ と ▣ はごちゃごちゃになっているかもしれません。気にしないでください



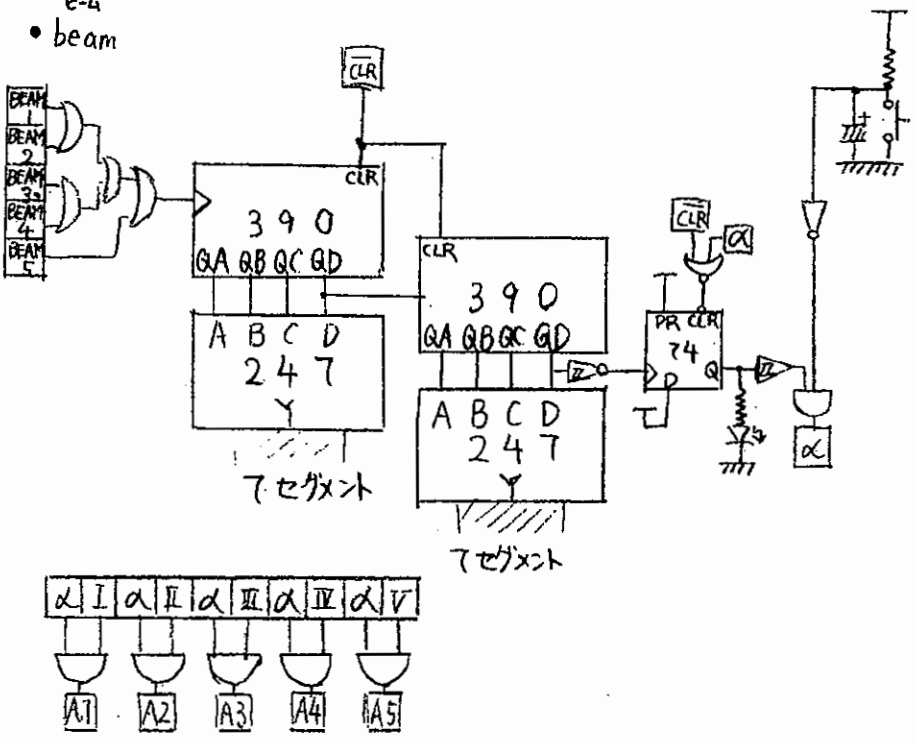




Firece Stream の回路

ビーム

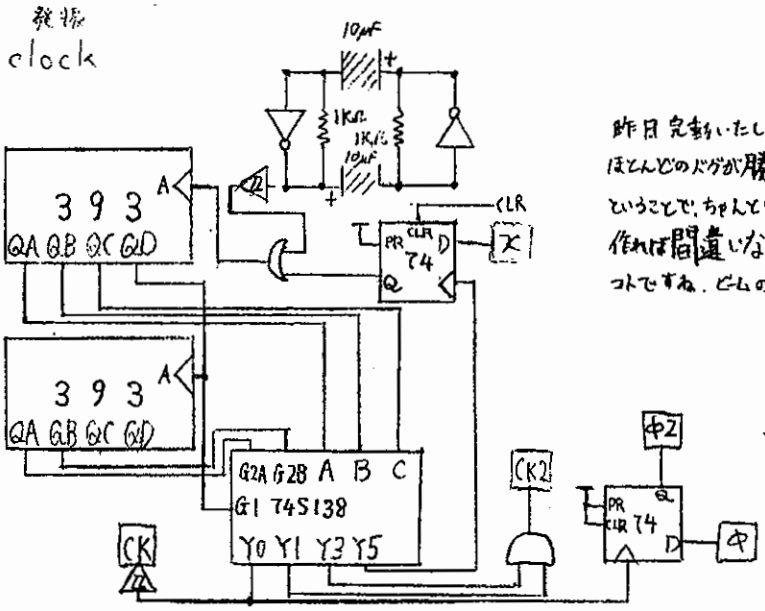
• beam



{2006.4.14 追加}

発振

• clock



昨日完動いたしました。
ほとんどのバグが勝手に直った
ということで、ちゃんと回路図に
作れば間違いない
ってことですね。ビームの存在意義UP!

働ケド働ケド我が暮ラシ楽ニナラザリ…

※題名です

製作 中二 村瀬　回路回設計 高二 伊藤先輩　協力 浅井先輩,井上先輩,他の先輩方

～ストーリー～

ストーリーなんて各々方がめいめいで考えてただければいいものだとは思いますが、書かなければ後のゲーム説明等にも多少不具合が生じるので、僕の考えたものを書いておきます。割と真剣に考えたので読み飛ばして下さい、という訳にもいかないので読んで下さい。

働き者の男がいた。彼は6人家族の主人で、妻と4人の子の口が男の肩にかかっていた。もとよりたいした貯金もなく、彼は日々働きづめである。果たして彼が報われることはあるのだろうか…？

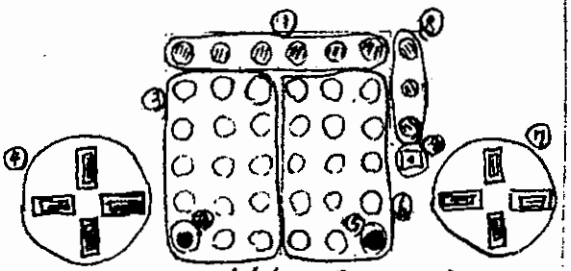
※関係各位

このゲーム、当初は「ボクシング」ゲームという設定だったので意外に思った方も多いと存じますが、そっちの方面の知識に乏しく良い題名が思いつかなかったため、設定変更をしたという次第です。

～ゲーム説明～

〈概観図〉

①緑LED（敵機発生場所）/上司（仕事発生場所）
②左側自機（赤色）/左のわらじ（肉体労働）
③左側自機移動範囲//上から下までが労働時間
④左側自機コントローラー/足を動かして働け



⑤右側自機（赤色）/右のわらじ（デスクワーク）
⑥右側自機移動範囲/上から下までが提出期限
⑦右側自機コントローラー/手を動かして働け
⑧体力ゲージ/身体の無理も三度まで
⑨クリアスイッチ/人生が二度あれば

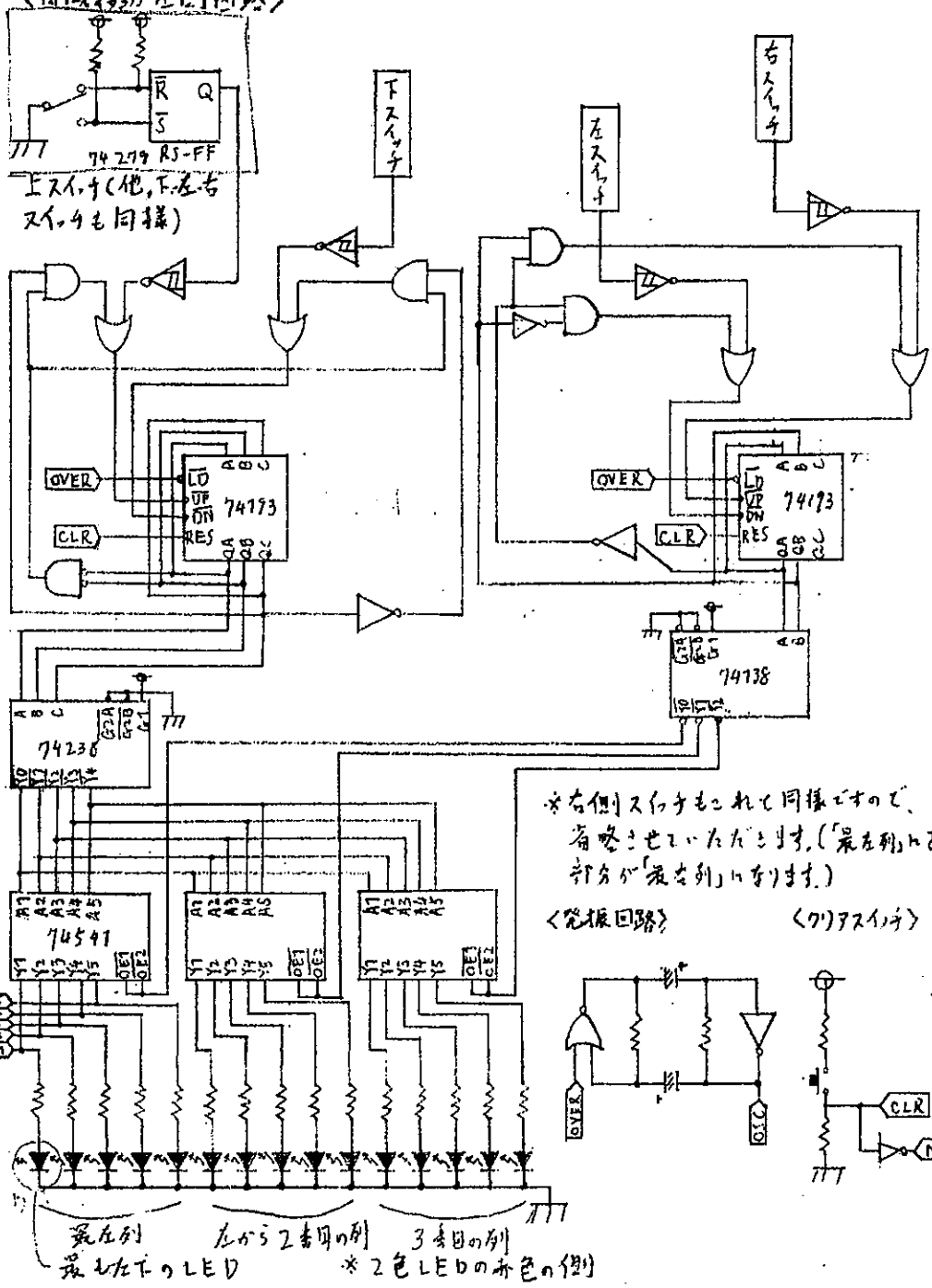
〈ルール〉

上司から命じられた仕事（緑色の敵機）が、上から降ってきます。プレーヤーは、敵機が一番下の段に着くまでに（制限期間中に）、敵機を倒さねば（ノルマをこなさねば）なりません。敵機は、自機を重ねることにより消滅します。敵機が一番下の段からも過ぎ去ってしまった場合、居残り残業で疲労がたまり、体力ゲージのポイントが1つ減ります。　〈続く〉

〈ルール続〉
体力ゲージが0になった場合(疲労が極限に達した場合)、ゲームオーバーとなります(昏睡状態に陥ります)。その場合、クリアスイッチを押せば、人生をなかったことにして、もう一度初めからやり直せます。…ちなみに、自機が左右に2つあるのは、二足の草鞋を履いているからです。

～回路図～

〈自機移動・左側回路〉

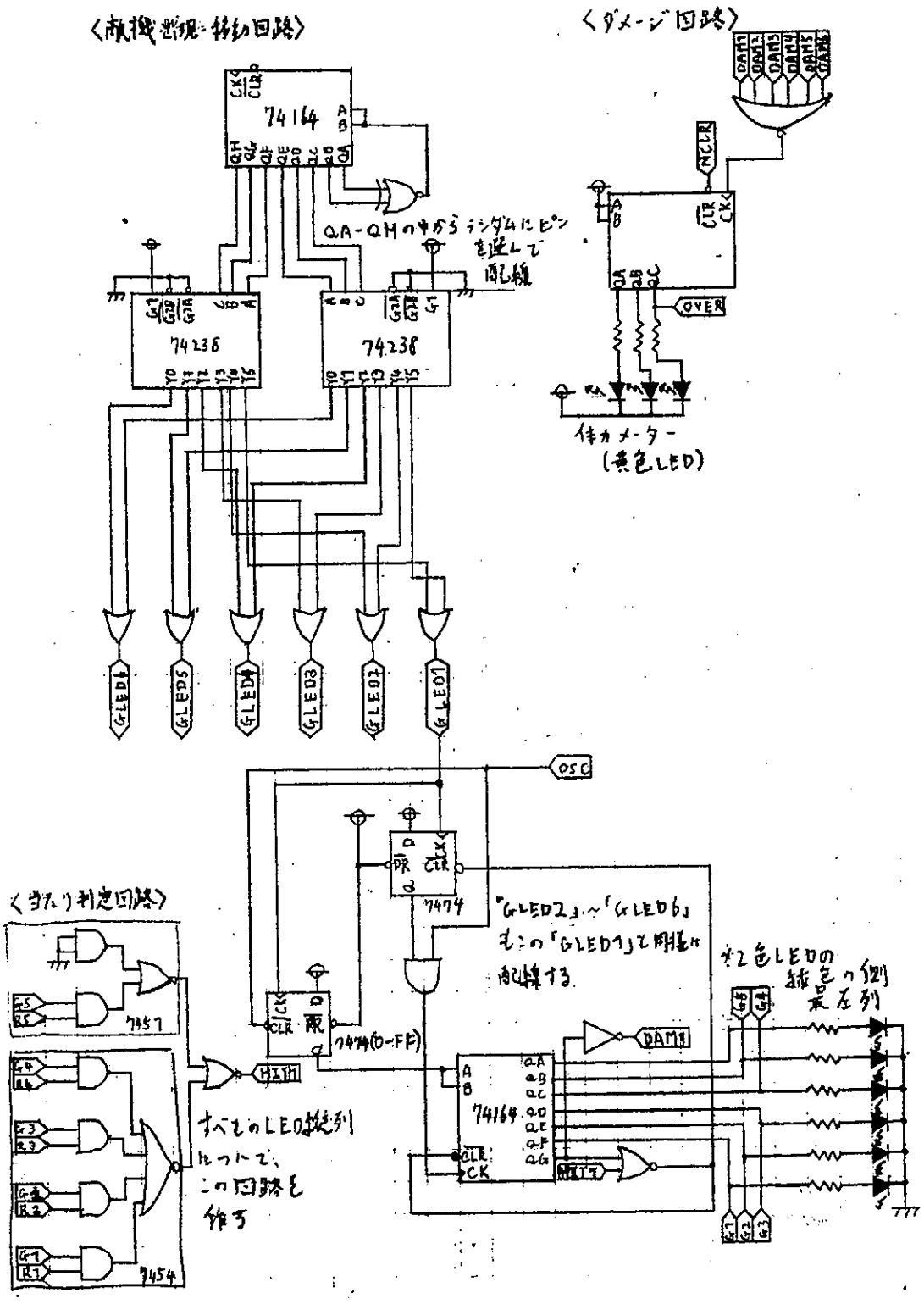


※右側スイッチもこれて同様ですので、省略させていただきます。(「最も列」にあたる部分が「最も列」になります。)

〈発振回路〉

〈クリアスイッチ〉

最左列 最も下のLED　左から2番目の列　3番目の列 ※2色LEDの赤色の側

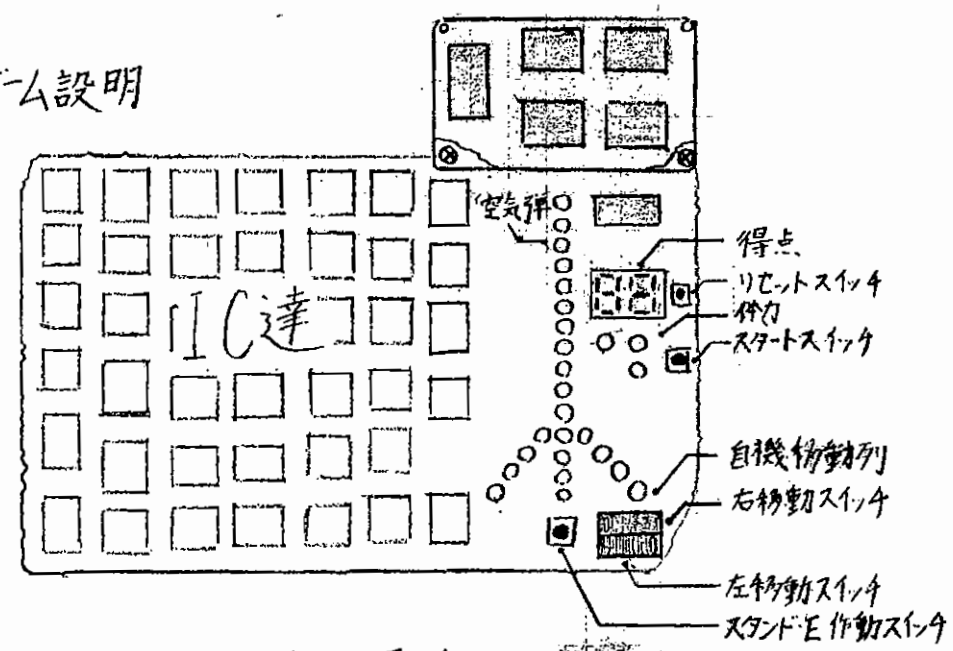

〈敵機出現・移動回路〉
〈ダメージ回路〉
74164
QA-QHの中からランダムにピンを選んで配線
74238
74238
GLED1
GLED5
GLED4
GLED3
GLED2
GLED1
NCLR
OVER
体力メーター
(黄色LED)
OSC
7474
「GLED2」~「GLED6」をこの「GLED1」と同様に配線する
〈当たり判定回路〉
7457
7454
HIT1
すべてのLEDの接列について、この回路を作る
7474(D-FF)
74164
DAM1
HIT1
*2色LEDの緑色の側
最左列

Part4
バトルインリング

製作者：M2 渡邊 翔　　回路設計者：H1 井上さん

協力者：物無の皆さん

① ゲーム説明



図の上から流れてくるボールを最後列で左右に動きタイミング良くキャッチし、得点を入れていくゲームです。くわしくストーリーにそって説明すると、スタートスイッチを押すとバトルが始まり、空気弾が流れてきます。スタンド（自機）は最後列で左右に動いてどこに障害物を置くか決め、空気弾と重なった瞬間、「スタンドE作動スイッチ」を押し、ブロックして下さい。空気弾の通り道は、左・右・真ん中・加速（瞬間移動）があります。成功すると、点数が入り、もう1個空気弾が出てきます。何でもない所でブロックしたり、見逃すと、本体に1ダメージがあります。2ダメージくらうと死亡でゲームオーバーです。

⑵ ストーリー

ある暗い裏通りを歩いている若い少年がいた。この少年の名は城野條成。彼は最近突然「スタンド」というスタンド使いにしか見えない力を手に入れていた。その力で浮浪者、チンピラを殴りながらも、城野はその力にあきてきていた。気がつくと、目の前にガラの悪い男が立っていた。男は何も言わずにスタンドを出し、殴ってきた。城野もスタンドを出し、ぎりぎりふせいだが、男に質問をすることにした。

「おまえもスタンド使いなのか？」

「そうだ。オレはギャング組織の指令でおまえを始末する!!」

しかし、相手は殴りかかりすに、妙なポーズをとった。とすると、空気弾が顔の横をすりぬけていった。男は語った。

「オレのスタンドは周りに流れをつくる能力！死ねい!!」

男は空気に流れをつくり、空気弾を飛ばしているのだ。

さらに、空気弾はスタンドをかわし直接本体へ向かってくる。

「レダクション・ブラック!!」

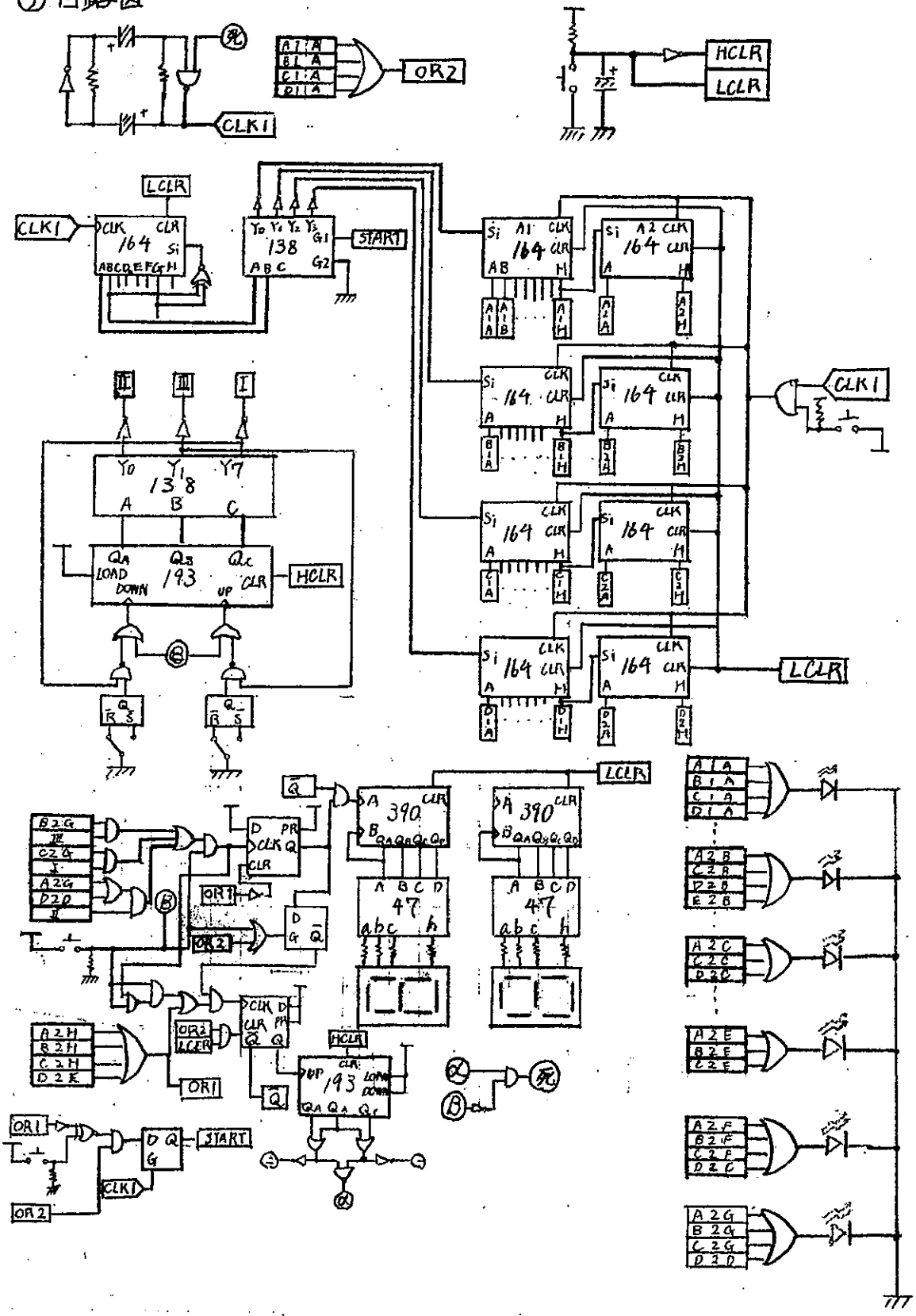
城野もスタンドを出した。このスタンドの能力はものの体積を減らす能力である。

「周りのものの体積を減らし、その急に低くなった気圧差を使ってものを投げこみふせぐしかない!!」

しかも、この空気弾は威力が強いから、一瞬にスタンドエネルギーを送り、より強い力で守る！しかし、この球を2発くらったら確実に死ぬだろう。

さあ君は方向転換する空気弾を止め、本体を守ることはできるのか？

③ 回路図



500yen

売り物

ハンバイブツ

第５０回運動会実行委員会選挙
投票用紙

	○
信任	ぶっぶ 不信任

2006/6/9
選挙管理委員会

第５０回運動会実行委員会選挙
投票用紙

	○
信任	~~不信任~~ 4

2006/6/9
選挙管理委員会

第５０回運動会実行委員会選挙
投票用紙

	○
信任	ぶっぶ 不信任だって

2006/6/9
選挙管理委員会

第５０回運動会実行委員会選挙
投票用紙

	いやまじでする
信任	~~不信任~~ ブンムウー ムセイセン

無効票

2006/6/9
選挙管理委員会

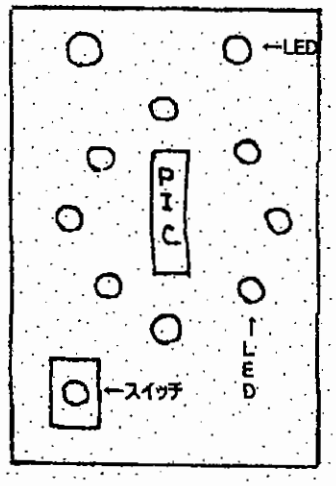
ブ、ルーレット　回るだけ♪

製作者　　M3　川崎敦史
原案　　M3　高崎尚人
協力者　　物無の皆様

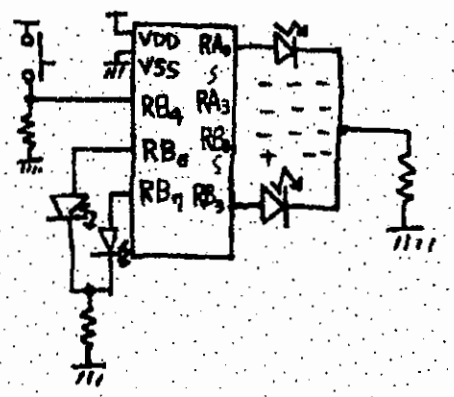
～ルール説明～

名前の通りルーレットゲームですね。まずスイッチを押すと円になっている LED のどれかが点滅します。数回点滅するとある一定の場所の LED が光り右回り、左回りと回転し始めるので、先ほど点滅した場所の LED が光ったときにタイミングよくスイッチを押してください。成功すると上についている LED が光りその場所が点滅します。失敗すると上についているほかの LED が光ります。成功しても失敗してもまたスイッチを押せばまた他の(同じ場合もあります)LED が点滅してゲームがもう一度できます。成功すればどんどん回転のスピードが速くなりますが失敗すると最初のスピードに戻ってしまうので注意してください。

～概観図～



～回路図～



～プログラム～

```
 LIST  P=PIC16F628A
 INCLUDE
P16F628A.INC
 __CONFIG
_CLKOUT_OSC&_WDT_
OFF&_PWRTE_ON&_CP_
OFF
ORG     0
OOTO    ～SETUP～
ORG     4
CLRF    INTCON
GOTO    ～WARI～
～SETUP～
BSF     STATUS,RP0
CLRF    TRISB
CLRF    TRISA
BSF     TRISB,4
BCF     STATUS,C
BCF     STATUS,RP0
CLRF    PORTA
CLRF    PORTB
MOVLW   46H
MOVWF   2DH
～SETUP2～
MOVLW   B'10001000'
MOVWF   INTCON
BSF     20H,0
BSF     21H,0
～KAITEN1～
RLF     21H
CALL    ～WAIT1～
GOTO    ～KAITEN～
～KAITEN2～
```

```
CLRF    21H
BSF     22H,0
CALL    ～WAIT1～
～KAITEN3～
RLF     22H
CALL    ～WAIT1～
BTFSS   22H,3
GOTO    ～KAITEN3～
CLRF    22H
BSF     21H,0
CALL    ～WAIT1～
GOTO    ～KAITEN1～
～WAIT1～
MOVLW   1EH
MOVWF   24H
～WAIT2～
MOVWF   25H
～WAIT3～
DECFSZ  25H
GOTO    ～WAIT3～
DECFSZ  24H
GOTO    ～WAIT2～
BTFSS   21H,3
RETURN
GOTO    ～KAITEN2～
～MAIN～
MOVLW   B'10001000'
MOVWF   INTCON
CLRF    PORTA
CLRF    PORTB
BSF     2BH,0
MOVF    2BH,W
MOVWF   PORTA
CALL    WAIT8
```

```
～KAITEN4～
RLF     2BH
MOVF    2BH,W
MOVWF   PORTA
CALL    ～WAIT8～
BTFSS   2BH,3
GOTO    ～KAITEN4～
BCF     2BH,3
MOVF    2BH,W
MOVWF   PORTA
BSF     2CH,0
MOVF    2CH,W
MOVWF   PORTB
CALL    ～WAIT8～
～KAITEN5～
RLF     2CH
MOVF    2CH,W
MOVWF   PORTB
CALL    ～WAIT8～
BTFSS   2CH,3
GOTO    ～KAITEN5～
BCF     2CH,3
MOVF    2CH,W
MOVWF   PORTB
BSF     2BH,0
MOVF    2BH,W
MOVWF   PORTA
CALL    ～WAIT8～
GOTO    KAITEN4
～WAIT8～
MOVF    2DH,W
MOVWF   2EH
～WAIT9～
MOVWF   2FH
```

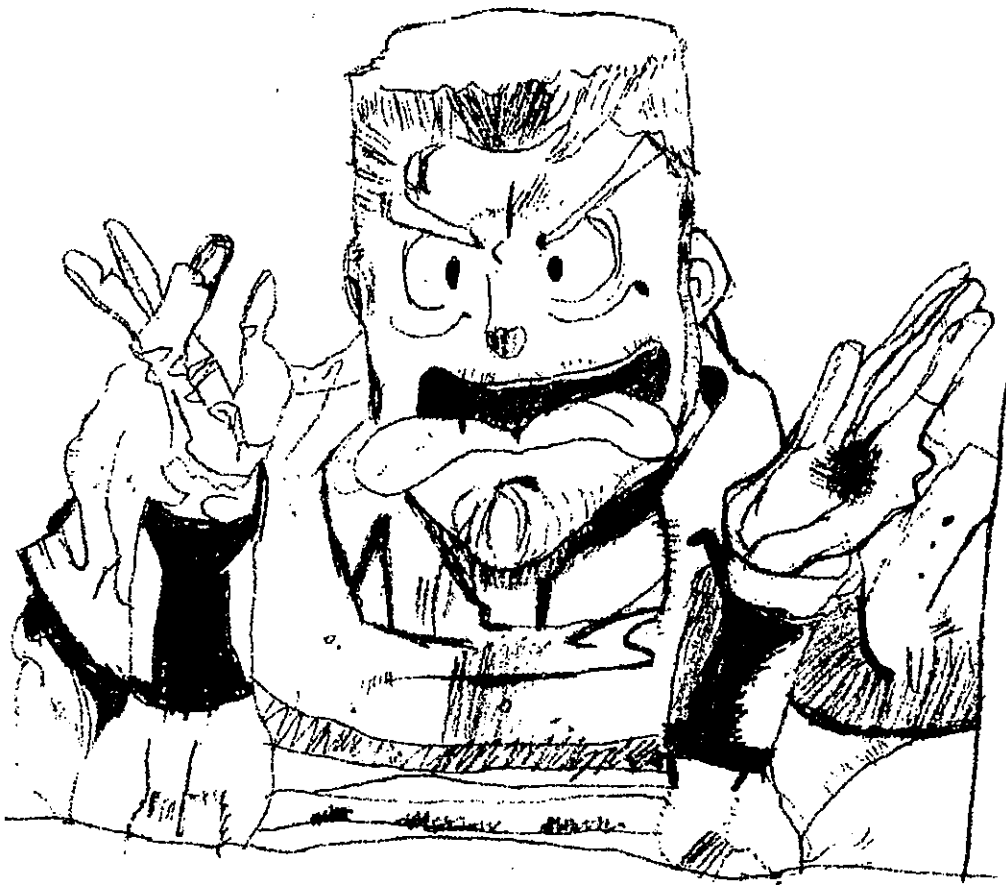
```
~WAIT10~
MOVWF   30H
~WAIT11~
DECFSZ  30H
GOTO    ~WAIT11~
DECFSZ  2FH
GOTO    ~WAIT10~
DECFSZ  2EH
GOTO    ~WAIT9~
RETURN
~WARI~
BTFSC   PORTB,4
GOTO    ~WARI~
BTFSC   20H,0
GOTO    ~SENTAKU~
MOVF    2BH,W
ANDWF   26H,W
MOVWF   31H
BTFSC   31H,0
GOTO    ~ATARI~
BTFSC   31H,1
GOTO    ~ATARI~
BTFSC   31H,2
GOTO    ~ATARI~
BTFSC   31H,3
GOTO    ~ATARI~
MOVF    2CH,W
ANDWF   27H,W
MOVWF   32H
BTFSC   32H,0
GOTO    ~ATARI~
BTFSC   32H,1
GOTO    ~ATARI~
BTFSC   32H,2
GOTO    ~ATARI~
~OUT~
```

```
BSF     PORTB,7
MOVLW   46H
MOVWF   2DH
GOTO    ~SETUP2
~ATARI~
BSF     PORTB,6
MOVLW   08H
MOVWF   33H
~TENMETU2~
MOVLW   B'01000000'
MOVWF   PORTB
CLRF    PORTA
CALL    ~WAIT4~
MOVF    28H,W
MOVWF   PORTA
MOVF    29H,W
MOVWF   PORTB
BSF     PORTB,6
DECFSZ  33H
GOTO    ~TENMETU2
DECFSZ  2DH
DECFSZ  2DH
DECFSZ  2DH
DECFSZ  2DH
DECFSZ  2DH
GOTO    ~SETUP2~
MOVLW   46H
MOVWF   2DH
GOTO    ~SETUP2~
~SENTAKU~
MOVF    21H,W
MOVWF   26H
MOVWF   28H
MOVF    22H,W
MOVWF   27H
MOVWF   29H
```

```
MOVLW   08H
MOVWF   2AH
~TENMETU~
CLRF    PORTA
CLRF    PORTB
CALL    ~WAIT4~
MOVF    28H,W
MOVWF   PORTA
MOVF    29H,W
MOVWF   PORTB
DECFSZ  2AH
GOTO    ~TENMETU~
BCF     20H,0
GOTO    ~MAIN~
~WAIT4~
MOVLW   2EH
MOVWF   34H
~WAIT5~
MOVWF   35H
~WAIT6~
MOVWF   36H
~WAIT7~
DECFSZ  36H
GOTO    ~WAIT7~
DECFSZ  35H
GOTO    ~WAIT6~
DECFSZ  34H
GOTO    ~WAIT5~
RETURN

END
```

剣客



井上雄彦

編集後記

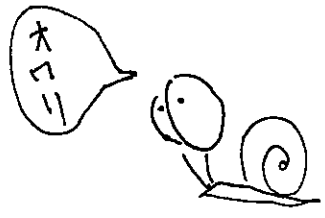
第1版　2006年4月

第2版　2006年6月10日

責任者　総務. 浅井

助手　中嶋
木村
池上

編集　物無員.



2006年度

物理部無線班

回路図集

本書は麻布学園物理部無線班の部員が1年間をかけて研究し創り出してきた様々な作品の説明、研究内容をまとめた1冊です。

定価：100YEN

麻布学園物理部無線班