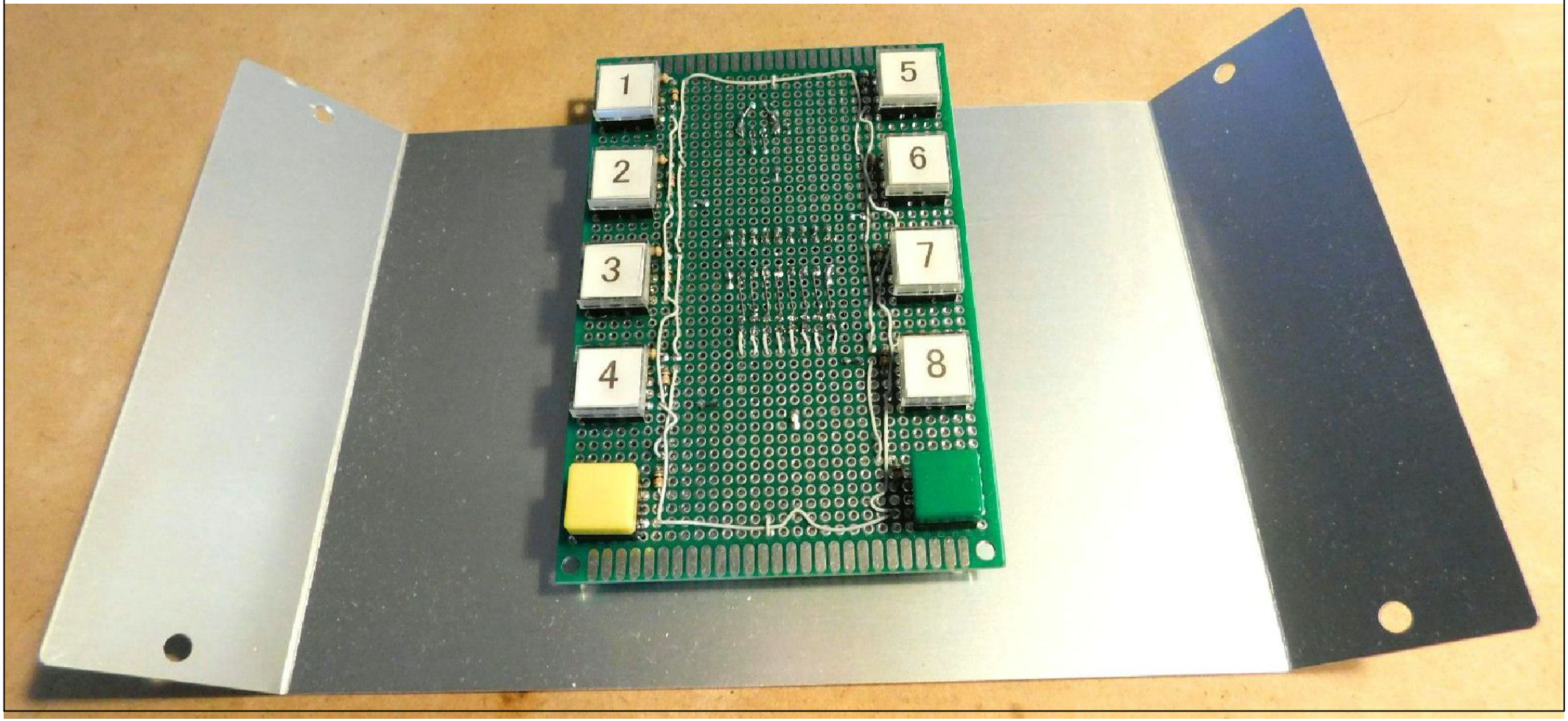
表示操作基板と アルミパネル

下の画像は、24時間タイマーの表示操作基板と 今回 切削加工する アルミパネルです。

このアルミパネルは、タカチの YM-180という アルミケースの天板です。 左右の側板を切削加工の都合で、左右に 広げてます。



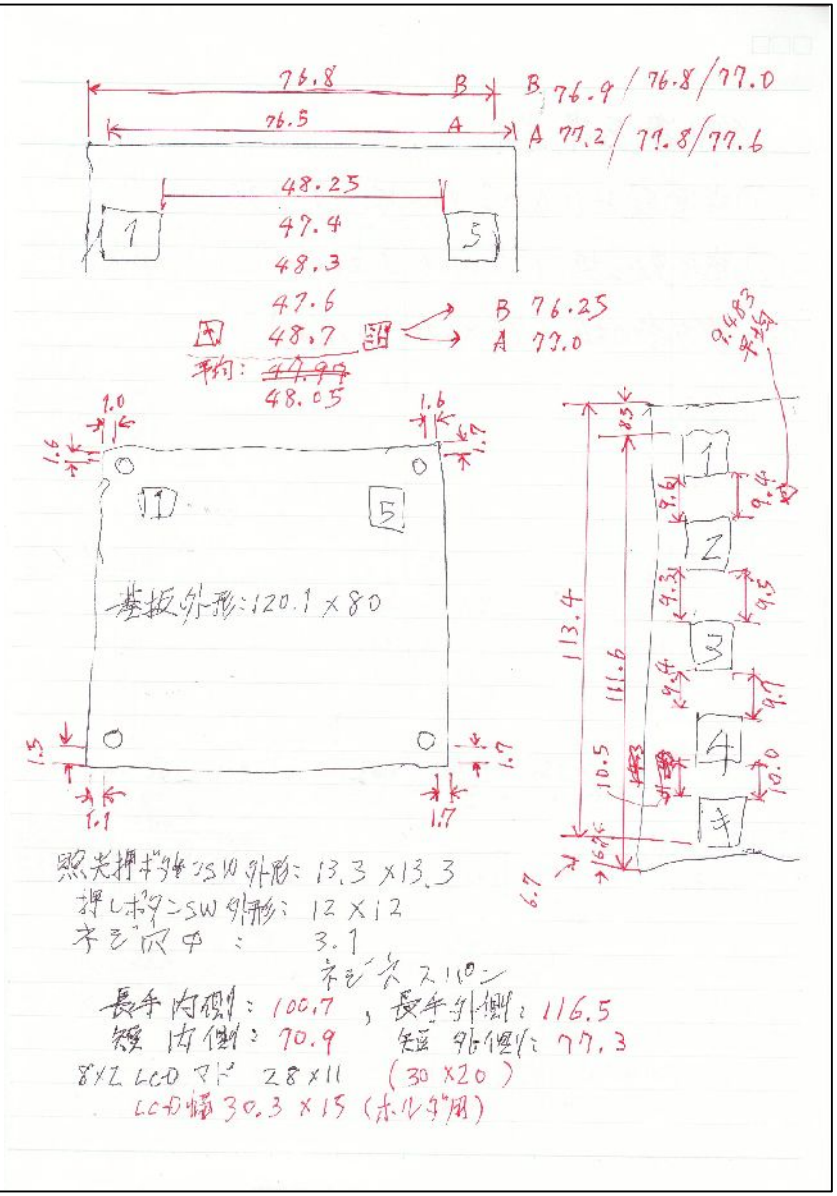
押しボタンスイッチ間の寸法を　書き込んだ紙

右側の 手書きの寸法図は、表示操作基板の　押しボタンスイッチ間の寸法を、出来る限り　ノギスで計り 手書きしたものです。

この図は、表示操作基板を、前面というか 押しボタンスイッチが見える側から描いた図に なります。 押しボタンのスイッチ穴を切削加工する場合は、パネルの裏側から行います。 よって右側の図面全体が、左右反転した形になります。

右の図面には、書いてありませんが、8文字2行の小型LCDを取り付ける予定なので、30x15mm の 液晶窓の 穴も 切削します。 液晶パネルの固定は、木製はたは、樹脂製の ホルダを作り、アルミパネルに　両面テープで貼り付けようと思います。

それと、ブザー音が聞こえやすいように、3つほど穴を開けておこうと思います。

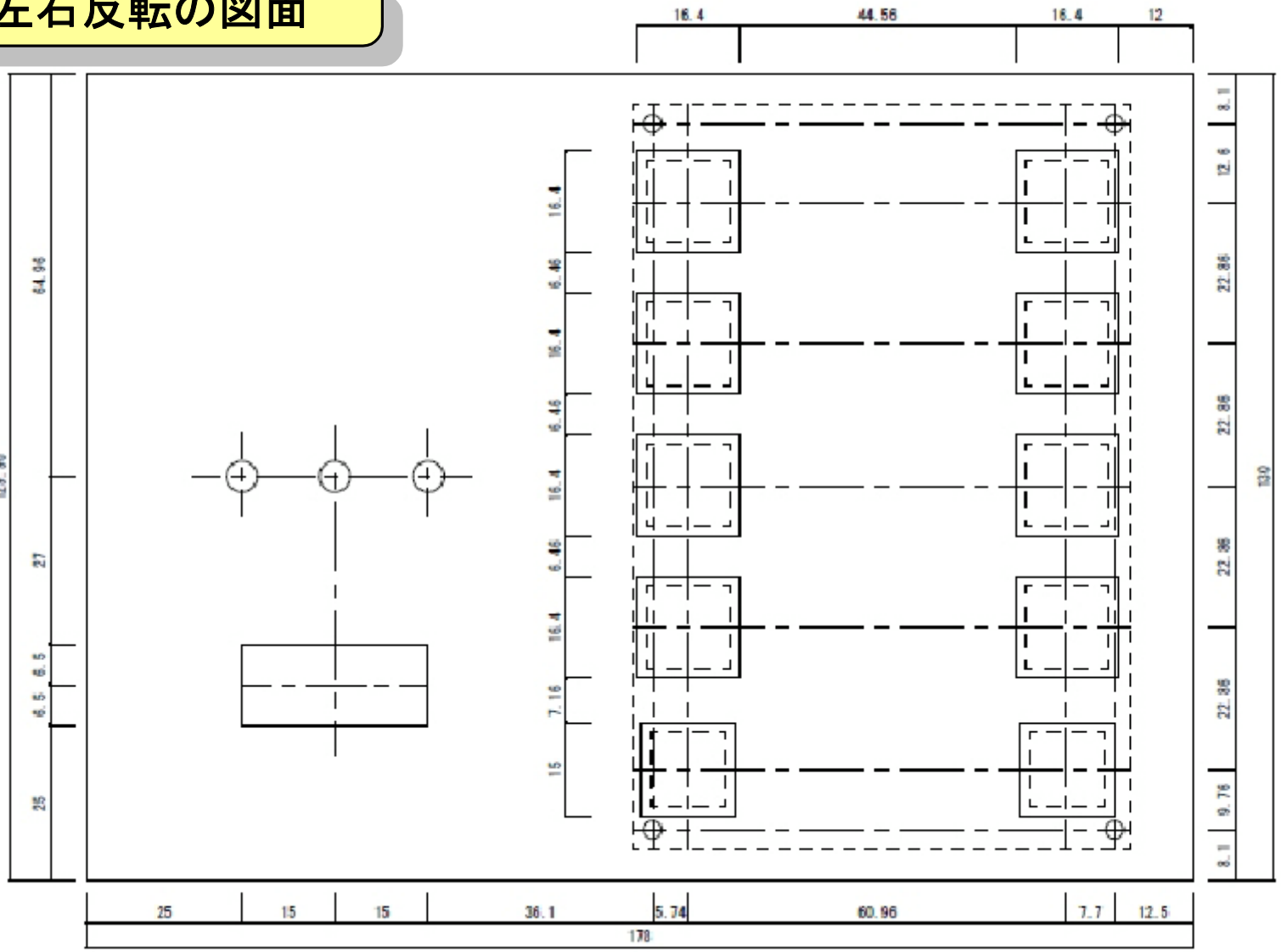


JWCADで作成した左右反転の図面

JWCADの寸法線の数値が、あまりに小さくて、申し訳ありません。

四角い穴を開けるのに、中心座標を指定するやり方で、穴を開ける事にします。

穴の種類は　3種類で、① 照光式押しボタンスイッチ 16.4x16.4
② 緑、黄色の押しボタンスイッチ　15x15
③ LCD表示窓穴
30x13 単位は mmです。



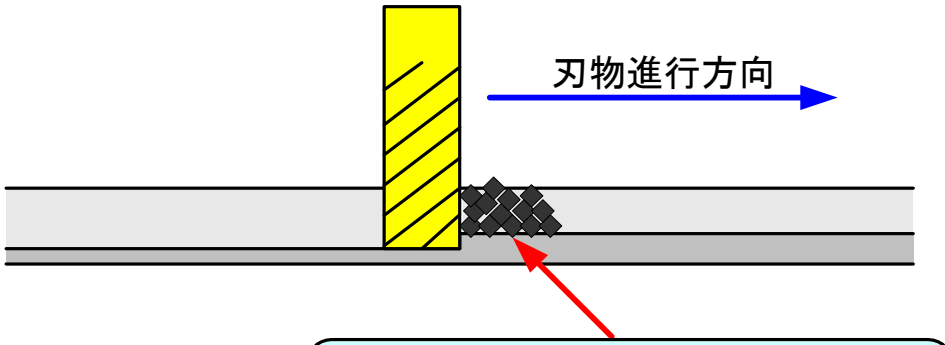
CNC1610での切削加工の手順

まず、先ほどの JWCADの 図面を　一度に切削する事は 加工範囲が　広くて 出来ません。

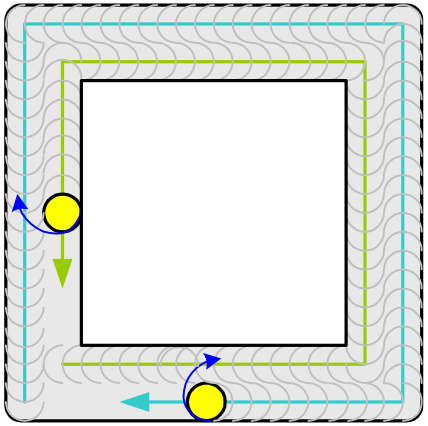
右側の 10個の押しボタンスイッチの穴と左側の　LCDの表示窓穴と、3つの穴を　2回に分けて 切削します。

CNC1610では、あまり太いエンドミルは使えないと思いますので、φ1mmのエンドミルで切削します。　φ1mmのエンドミルは、折れやすいので、深さを　0.1mmずつ 下げて 最後は1.1mmまで、下げて、11回のパスで 削ろうと思います。

あと、念のため、深さの1ステップ毎に本来の寸法で切削した後に、0.9mm内側を、逆回りで削ろうかと思います。　切り子の詰まりを防ぐためです。　溝が深くなると切り子の詰まりによって、エンドミルが折れる恐れがあるからです。



切削溝が深くなると、左右に壁が出来るため　切り子の逃げ場が、なくなり、刃物の進行方向前に　切り子が、固まってたまる傾向があると思われます。



左の図は、外側と内側の二重に削った場合の刃物が通った軌跡をイメージした図です。　刃物は上から見て時計回りに回転します。
よって　青の矢印のように切り子が横に逃げます。　前に溜まる事は、ありません。

CNC1610　NCファイルの定型コード群

過去の動画 030の資料を
一部引用します。

★　NCファイル先頭の 定型的コード群

```
G90　（ 絶対座標指令 ）
G1 Z3 F200　（ 刃先を Work上面より3mm上に
　　上げる ）
　　（ 切削移動速度 200mm/Minを設定 ）
M03 S1000　（ 主軸正転、主軸速度:最大 ）
G0 X... Y...（ 切削開始位置上に移動 ）
G1 Z-0.1　（ 刃物先端を Workに降ろす ）
　　（ この場合、刃先を 0.1mm ）
　　（ 食い込ませる ）
```

★　NCファイル最後の 定型的コード群

```
G1 Z3　（ 刃先を Work上面より 3mm上に
　　上げる ）
G0 X0.000 Y0.000　（ 原点上に 移動する ）

M5　（ 主軸停止 ）
M2　（ プログラム終了 ）
```

（ ）内の緑色のコメントには、漢字を書きましたが、実際のNCコードでは、ASCII文字だけに して下さい。

★ ブロック間移動時の 定型的コード群

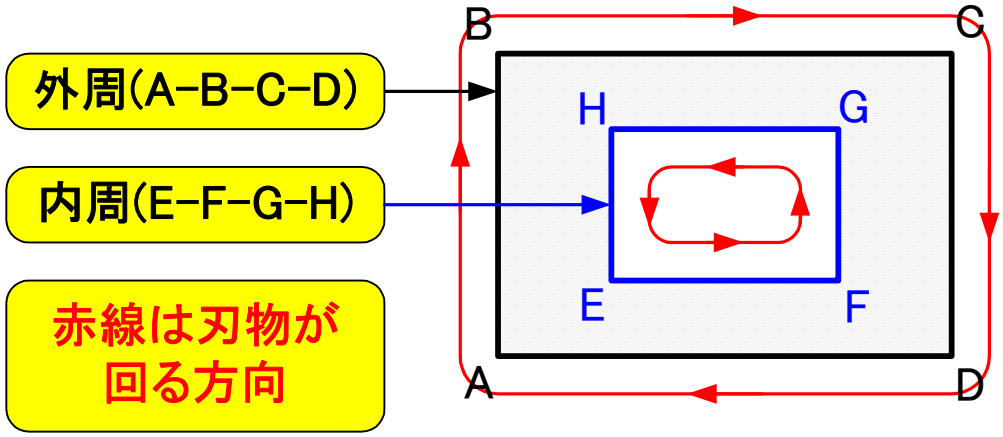
G1 Z3　　（ 刃先を Work上面より 3mm上に 上げる ）
G0 X... Y...　（ 次の切削開始位置上に 移動する ）
G1Z-0.1　（ 刃先先端を Workに降ろす ）
　　（ この場合、Workに 刃先を 0.1mm ）
　　（ 食い込ませる ）

（ ）内の緑色のコメントには、漢字を書きましたが、実際のNCコードでは、ASCII文字だけに して下さい。

閉じた図形、外側は時計回り
内側は反時計回り

プリント基板のパターンデータの時も 今回の iPhoneF200.nc でも、外周を回る切削データは時計回りで、内周を回る切削データは、反時計回りでした。

何らかの約束事があるのだと思いますが、Web検索で、的を得た回答は見つけられませんでした。



この外周は時計回り、内周は反時計回りの約束事に従う事にします。

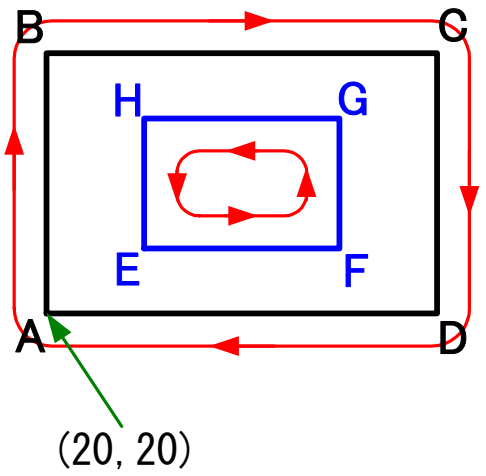
人間は 四角形A-B-C-Dと、四角形E-F-G-Hを見れば、A-B-C-Dが外周で、E-F-G-Hが内周とすぐ分かります。

しかし機械は外側と内側の判定が、難しい場合があるので、多角形の面積計算で得られる法線ベクトルを用いて外周、内周の判断をしているのではと 思われます。

法線ベクトルは、時計回りの点列の多角形であれば 面積計算の答えが（−）です。
反時計回りの点列の多角形であれば、面積計算の答えが（＋）です。
面積がほしい場合は、絶対値を取ります。

簡単なテストデータの作成

前ページの二重の四角のデータに座標値を付けて NCデータを作成します。



点名	X座標	Y座標
A	20	20
B	20	60
C	80	60
D	80	20
E	35	30
F	65	30
G	65	50
H	35	50

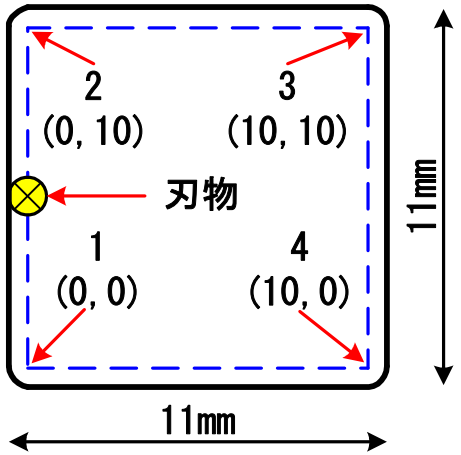
凡そ、CNC1610の NCコードのイメージは、掴めましたでしょうか。 今回は 使用しないので円弧のGコードは、説明していません。円弧のGコードは、030の動画を 参照して下さい。

```
G90
G1 Z3 F200
M03 S1000
G0 X20 Y20  ( Point A )
G1 Z-0.1
G1 X20 Y60  ( Point B )
G1 X80 Y60  ( Point C )
G1 X80 Y20  ( Point D )
G1 X20 Y20  ( Point A )
G1 Z3
G0 X35 Y30  ( Point E )
G1 Z-0.1
G1 X65 Y30  ( Point F )
G1 X65 Y50  ( Point G )
G1 X35 Y50  ( Point H )
G1 X35 Y30  ( Point E )
G1 Z3
G0 X0 Y0
M05
M02
```

切削刃物のオフセットを 考慮する

フライス盤とか使いなれている方にとっては当たり前の事ですが、初心者にとっては、切削刃物のオフセット って、何。？ という方も おられると思いますので簡単に説明しておきます。

NCコードというか、Gコードで指定する座標値は、基本的に、主軸の回転中心の XY座標となります。 これは、そのまま 切削刃物の回転中心の、XY座標です。 且つ、刃物には 刃先径が、あります。 今回使用する刃物は、φ1mmです。 この刃物で、左下座標が、(0, 0)で、10mm の 四角い穴を 開けようとすると 単純に考えると、座標の移動は、(0, 0)–(0, 10)–(10, 10)–(10, 0)–(0, 0) となりますが、切削した結果は、刃物の半径分 外側を、削ってしまう事に なります。



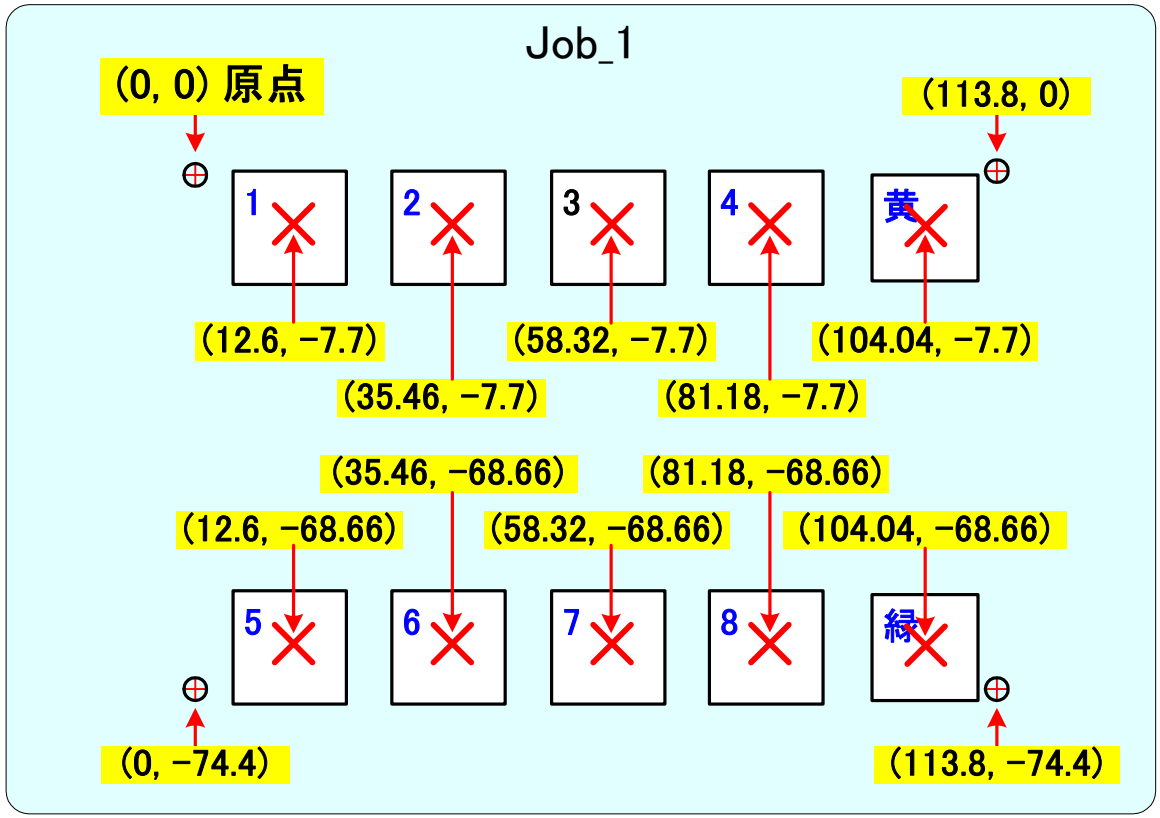
左の図は、刃物の半径分 外側を、削ってしまう図ですが、イメージが掴めましたでしょうか。？

よって、刃物の半径分 内側に引っ込めた座標値で 刃物センターを指定して、最終的に目標値の 10 カケル 10mm で削る事にします。

この、刃物の半径分 内側に引っ込めた座標値の事を 刃物のオフセット、を考慮した座標値という事になります。

CAD図面の寸法値から、座標値を生成する、押しボタン穴 側

JWCADの図面から、CNC1610のNCコードで
指定する座標値を生成します。



左上 ネジ穴位置を、原点にします。
よって、第4象限に配置される事になります。

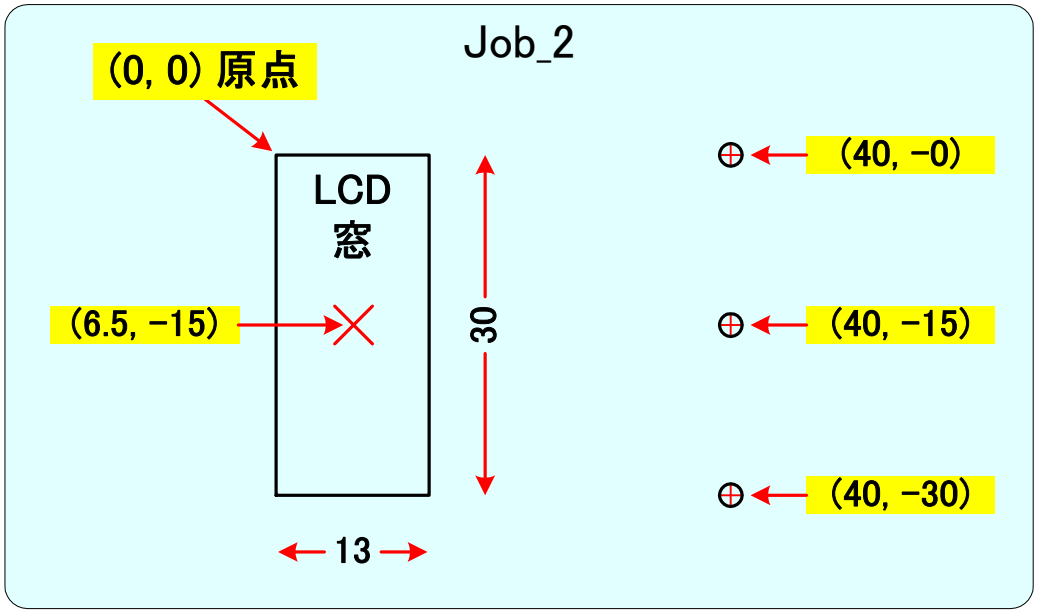
ボタン位置は、中心座標で指定します。

照光式押しボタンスイッチの 穴寸法
15.4 × 15.4

黄色、緑の押しボタンスイッチの 穴寸法
14 × 14

ネジ穴の寸法 3mm
CNC1610では、センタ穴 φ1mmだけ
開けます。

CAD図面の寸法値から、座標値を生成する、LCD窓穴 側



NCファイルの名前は、前ページの Job_1 とこの、ページの Job_2 という名前で、2本出力します。

LCD窓の 左上角を、原点にします。
よって、第4象限に配置される事になります。 ネジ穴の寸法 3mm
CNC1610では、センタ穴 φ1mmだけ開けます。

CNC1610用の Gコードファイルを Delphiで 作成

```
//*********************************
//**  開始処理                   **
//*********************************
procedure TForm1.nc_start;
begin
    Memo1.Lines.Add( 'G90' );           // 初期化 NCコード 1
    Memo1.Lines.Add( 'G1 Z5 F200' );    // 初期化 NCコード 2
    Memo1.Lines.Add( 'G0 X0.0 Y0.0' );  // 初期化 NCコード 3
    Memo1.Lines.Add( 'M03 S1000' );     // 初期化 NCコード 4
    Memo1.Lines.Add( '' );  // 空白挿入
end;

//*********************************
//**  終了処理                   **
//*********************************
procedure TForm1.nc_end;
begin
    Memo1.Lines.Add( '' );  // 空白挿入
    Memo1.Lines.Add( 'G1 Z3' );         // 終了処理  NCコード 1
    Memo1.Lines.Add( 'G0 X0.0 Y0.0' );  // 終了処理  NCコード 2
    Memo1.Lines.Add( 'M5' );            // 終了処理  NCコード 3
    Memo1.Lines.Add( 'M2' );            // 終了処理  NCコード 4
end;
```

Delphiは、Windowsアプリ開発用の統合開発環境です。 私は、慣れが あって、Delphiが 早く確実に作れますので、Delphiを 使用します。

NCコード文字列を 出力するのは、Delphiの メモ帳コンポーネントです。

一通り、NCコード文字列を 出力し終わったら、メモコンポーネントの機能で、テキストファイルとして、出力出来ます。

左の、プログラムは 開始処理と、終了処理の 手続きです。
(C言語でいうところの関数です。)
開始処理呼び出しは、nc_start; になります。
終了処理呼び出しは、nc_end; に なります。

```
//************************************
//**  センター穴あけ 1.1mm開ける     **
//** --------------------------- **
//**  xc, yc : 穴の中心座標          **
//************************************
procedure TForm1.nc_wr_hole( xc, yc: Double );
var
i:  Integer;
zd: Double;
tx: String;
begin
    Memo1.Lines.Add( 'G1 Z3' );     // 一旦、刃物を上に上げる
    tx := Format('G0 X%6.2f Y%6.2f', [xc, yc]);
    Memo1.Lines.Add( tx );              // NCコード出力
    zd := -0.1;
    for i:= 1 to 11  do
    begin
        tx := Format('G1 Z%5.2f', [zd]);
        Memo1.Lines.Add( tx );          // NCコード出力
        Memo1.Lines.Add( 'G1 Z3' );     // 刃物を上に上げる
        zd := zd - 0.1;             // 刃物深さ更新
    end;
    Memo1.Lines.Add( 'G1 Z3' );     // 刃物を上に上げる
end;
```

左のソースは、ネジ穴のセンタ穴明け処理 nc_wr_hole です。

引数として、穴中心の X座標、Y座標を 渡します。

for 文で 11回、回してますが、0.1mm単位で、穴を開ける切り子を出すために、都度 刃物を 上に上げて、次の深さに、刃物を下げます。 1.1mmまで、穴を開けたら 終りに なります。

```
//************************************
//**  四角穴の切削 NCコマンド出力    **
//**  xc, yc :  四角形の中心座標     **
//**  xl, yl :  四角穴の 横幅、高さ **
//************************************
procedure TForm1.nc_wr_rect( xc, yc, xl, yl: Double );
var
    i:  Integer;
    tx: String;
    sw: Byte;
begin
    Nr.xc := xc;
    Nr.yc := yc;
    Nr.xl := xl;
    Nr.xl := yl;
    Nr.xho := (xl / 2) - Em_Rad; // 外周長さ生成
    Nr.yho := (yl / 2) - Em_Rad;
    Nr.xhi := Nr.xho - 1;          // 内周長さ生成
    Nr.yhi := Nr.yho - 1;
    tx := Format('( Center  X=%6.2f, Y=%6.2f )', [Nr.xc, Nr.yc]);
    Memo1.Lines.Add( tx );  // 中心座標のコメント
    sw := 0;
    Nr.zd := -0.1;          // 0.1mm刃物を食い込ませる
```

左のソースは、
四角穴の切削　NCコマンド出力処理
nc_wr_rect;　です。
以下は、引数です。
xc, yc :　四角形の 中心座標
xl, yl :　四角形の 横幅、高さ

begin の 下では、Nr という構造体に四角形作図に関わるパラメータを、計算して入れ込んでます。
Nr は、複数のパラメータを、引数として　並べて 下位の手続きを呼び出すのが、煩わしいので、Nr という 構造体に入れ込んでます。 ここでは、nc_wr_rect 手続きは、中心座標と、幅、高さを与えれば、四角形を切削してくれると解釈して下さい。

```
    for i:=1 to 11  do
    begin
        nc_rect_layer( sw );       // レイヤー １面切削
        NR.zd := Nr.zd - 0.1;    // 0.1mm 刃物を深くする
        sw := 1;
    end;
    Nr.zd := 3;
    nc_z_out( Nr.zd, 1 );                    // 刃物を 3mm上げる
end;
```

左のソースは、四角穴の切削 NCコマンド出力処理 nc_wr_rect; の続きです。
0.1mm 単位のレイヤーを 11回 繰り返します。
最後に、刃物を 3mm 上げます。
ループの中で、
nc_rect_layer(sw); は １レイヤー分の 切削処理 手続きです。
パラメータの大半は、Nr 構造体で渡してます。

```
//*****************************
//**  レイヤー  １面切削      **
//*****************************
procedure TForm1.nc_rect_layer( sw: Byte );
begin
    if sw = 1  then  Memo1.Lines.Add( '' );  // 空白挿入

    nc_xy_out( Nr.xc -Nr.xho, Nr.yc -Nr.yho, 0 );   // 1.外周切削
    nc_z_out( Nr.zd, 1 );                    // 刃物降ろす
    nc_xy_out( Nr.xc -Nr.xho, Nr.yc +Nr.yho, 1 );   // 2.外周切削
    nc_xy_out( Nr.xc +Nr.xho, Nr.yc +Nr.yho, 1 );   // 3.外周切削
    nc_xy_out( Nr.xc +Nr.xho, Nr.yc -Nr.yho, 1 );   // 4.外周切削
    nc_xy_out( Nr.xc -Nr.xho, Nr.yc -Nr.yho, 1 );   // 1.外周切削
    Memo1.Lines.Add( '' );  // 空白挿入
    nc_xy_out( Nr.xc -Nr.xhi, Nr.yc -Nr.yhi, 1 );   // 1.内周切削
    nc_xy_out( Nr.xc -Nr.xhi, Nr.yc +Nr.yhi, 1 );   // 2.内周切削
    nc_xy_out( Nr.xc +Nr.xhi, Nr.yc +Nr.yhi, 1 );   // 3.内周切削
    nc_xy_out( Nr.xc +Nr.xhi, Nr.yc -Nr.yhi, 1 );   // 4.内周切削
    nc_xy_out( Nr.xc -Nr.xhi, Nr.yc -Nr.yhi, 1 );   // 1.内周切削
end;
```

これが、1レイヤーの切削処理で指定した、四角形中心座標を中心に仕上げ寸法となる 外周切削を 点列 １２３４１と 切削して行きます。

そして、切り子対策のための、内周切削を 点列 １２３４１と 行っていきます。

ちなみに、nc_xy_out は、引数 x、y が移動先座標で、3番目のパラメータが 0 の時 G0 、1 の時 G1 で、切削する指定です。 因みに Nr.xho, Nr.yho は、半分の幅、半分の高さの 値です。

Nr.xhi、Nr.yhi は 内周側の半分の幅、半分の高さの 値です。

Nr.xc、Nr.yc は、中心座標です。

```
//*********************************
//**  Job 2  LCD窓穴、穴 3個     **
//*********************************
procedure TForm1.Button6Click(Sender: TObject);
begin
    nc_start;
      nc_wr_rect( 6.5, -15, 13, 30 );     // LCD窓穴
      nc_wr_hole( 40.0,   0.0 );     // ネジ穴 1
      nc_wr_hole( 40.0, -15.0 );     // ネジ穴 2
      nc_wr_hole( 40.0, -30.0 );     // ネジ穴 3
    nc_end;
end;
```

今まで説明してきたプログラムを、呼び出している処理です。 Job_1 側は、プログラム量が、やや多いので、Job_2 を 表示します。

最初に nc_start; （ CNC1610の 初期化処理 ） を 呼び出しています。
nc_wr_rect(6.5, -15, 13, 30); が LCDの 窓穴の四角形加工です。 因みに 4つの引数は、X中心座標、Y中心座標、横幅、高さ です。
nc_wr_hole(40.0, 0.0); は、センター穴開け加工です。 2つの引数は、穴あけ座標値の X と Yです。
最後に、nc_end; にて CNC1610に 終了処理を 通知します。

このように、これらのサブプログラム手続きの引数に 適切な値を 設定する事により、NCコードを 生成する事が 出来ます。