ARDF 用
ID ジェネレータの試作
(試作 1 号機)

※記憶を頼りに書いてみました(笑)

2015 年 6 月 20 日

JR5HJJ

ホームページ　http://wwwb.pikara.ne.jp/potter-2005/

電子メール　jr5hjj@jarl.com

■事の始まり

ARDF 用の受信機を数年前から試作してきたが、実は、TX を持っていなかった。試作した受信機をテストするのは練習会を利用したり TX を持っている局長さんに TX を用意してもらったり・・・。

自宅で試作受信機をテストするには固定の 144MHz のトランシーバから SSB で試験電波を出したり実験用の自作オシレータのモレ電波を使っていたが、あまりにも効率が悪かった。固定のトランシーバから出る SSB の電波はパワーを絞っても強すぎるし受信機のテストには使いづらく、自作オシレータは送信機ではないので当然パワーは小さすぎるし、チャチな発振回路で周波数変動もあり試作受信機の具合を確認することは難しかった。

練習会で試作受信機を使おうとすると自宅で調整した周波数からずれてしまっているような(^^;)、毎度練習会の現場で試作受信機を解体し局部発振周波数を調整をすることになった。やはり、受信機はちゃんとした送信機でテストしないといけないなあ・・・と思った。

推定であるが・・・、自作のオシレータと試作受信機の局部発振回路は基本的に同じ回路を使っていて、温度が変われば実験用の自作オシレータも試作受信機も同じように周波数が変動していたようで、季節の違いによる気温の変化のような長期の変動に気がつかなかった。

そこでネットでも時々見かける ARDF 用の ID ジェネレータを作って、手持ちのハンディトランシーバへつないで試作受信機の動作チェックができるよう考えてみた。

■どんな ID ジェネレータを作るか

①ハンディトランシーバへ簡単につなぐ

一般にハンディトランシーバといえば電波形式が FM のハンディトランシーバで、ID ジェネレータをつなぐと当然 FM の電波が出るが、平成 26 年の ARDF 競技の改正で 144MHz の競技実施方法からは A2A だけとなったので、競技には使えない。

しかし、受信機で音が聞けて、ARDF の TX を持っていない人が受信機のテストをするなり、ちょっと ARDF の宣伝をするなり(笑)、ARDF を知らない人がちょっと体験するくらいなら、小型で簡単にハンディトランシーバにつないで使えるものがいいと思った。

②乾電池で長時間動く

ハンディトランシーバから ID ジェネレータの電源を取ることは簡単ではないし、ちょっと ARDF を体験、という時にはどこででも手に入る乾電池で長時間動く方が望ましいだろうし、将来、AM 方式の送信機で TX を作るとすればベースとなる部分として使いたいと思ったので、乾電池で長時間動くモノが良いと思った。

③TX1~5、ゴールビーコンとして符号を出せる

設定によって TX1~TX5、ゴールビーコンの符号を出せるようにしておけば電波形式が FM とはいえ実際の競技に近い模擬練習も可能だと思うので同じ ID ジェネレータを 6 個用意し、それぞれをハンディトランシーバにつないで使うことも想定した。

④複数の ID ジェネレータを同期スタートできる

複数の ID ジェネレータをコードでつないで 1 個の ID ジェネレータをスタートすれば、他の ID ジェネレータも同期スタートできるように考えた。具体的には・・・単に電源を並列接続して ON するだけ。(笑)

⑤操作が簡単であること

基本的な機能に絞った。電源 ON 後、直ちにスタートし、スタート後は再スタート無しに何番目の TX に設定するか 1 個のプッシュスイッチで自由に選択できるようにした。

プッシュスイッチを押したまま電源を入れるとゴールビーコンとして動作するようにした。

ストラップの設定で即送信開始とスタート後 40 分経過してから送信開始の選択ができるようにした。

■最初の壁

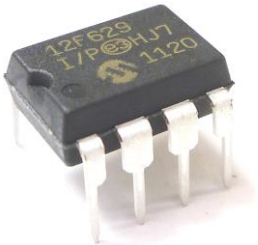
ID ジェネレータはネットの記事でもマイコンを使っているモノが多い。パーツを扱ってる店では PIC とか H8 とか SH と Arduino とか AVR とかいろいろ見かけるが、自分はそれらをよく知らなくて 30 年ほど前にザイログの Z80 やインテルの 8085 をかじったくらいだった。

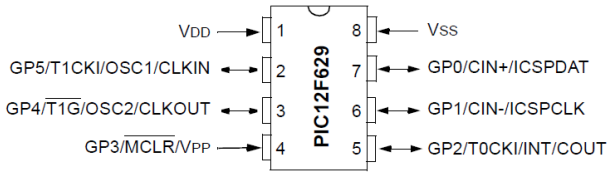
しかし、ID ジェネレータをマイコン無しで作ると小型化は無理だろう。汎用ロジック等のハードウェアだけで動くようなのは完成予想のイメージすらできない。そこで以前から興味のあった PIC を使ってみようと思った。

PIC を使うと言っても、どのチップを使うか、PIC の機能も命令もよくわからない状態からのスタートであるが、ID ジェネレータはトランシーバの送受信を ON／OFF できることとモールスの音を ON／OFF できること、TX やゴールビーコンの設定をスイッチでできることを考えると、出力 2 ピン、入力 1 ピンの 3 ピンあれば何とかなると思った。電源のプラスマイナスの 2 ピンを足すと 5 ピンになるので、8 ピンの PIC でやってみようかな、と始めたのであるが・・・、8 ピンの PIC といってもいくつもあるし、何が何やら・・・(苦笑)

まず、PIC を扱ったことの無い僕は何回もプログラムを書き直しすることになるだろう、という予想からフラッシュタイプ、ネットで参考になりそうな製作記事がいくつか見つかった PIC12F629 を使うことにした。

■PIC12F629 とは





・８ピンフラッシュベース←何回も電気的に消去書き込みができる
・８ビットＣＭＯＳマイクロコントローラ
＝主な仕様＝
・プログラムメモリ：１０２４ワード←たぶん足りるだろう(笑)
・ＳＲＡＭ：６４バイト←今回は充分足りる
・ＥＥＰＲＯＭ：１２８バイト←今回は使わないが将来使ってもいい
・Ｉ／Ｏ：６←今回は少なくても５以上あれば足りる
・コンパレータ：１←今回は使わない
・タイマー：８ビットｘ１、１６ビットｘ１←クロックを分周して時刻の処理をするのに使う
※パッケージ：ＤＩＰ８ピン
と秋月電子通商のネット広告には書いてあった。(笑)
ちなみに平成 27 年 3 月 29 日現在、1 個 90 円

■正確なタイミングが必要だ

最初勘違いしていたのであるが、PIC の内蔵オシレータは水晶振動子を使っていると思っていた。(笑)

プログラムで発振モードを切り替えるのは、必要な処理速度を確保しながら消費電力を抑えるのに、どのモードが良いか選択する、ということか？、と誤解していた。実際には RC 発振回路らしいが・・・。

ARDF では 1 分毎に各 TX が 1 番から 5 番まで電波を発射して行くが、TX の送信の切替時刻は誤差 5 秒以内(JARL の ARDF ハンドブック、平成 20 年 7 月 1 日　第 2 版)と決められていて、1 年中 3 時間程度の間、内蔵オシレータで誤差 5 秒以内に押さえられるか、という点で不安があったので外付けで水晶振動子を使うことにした。

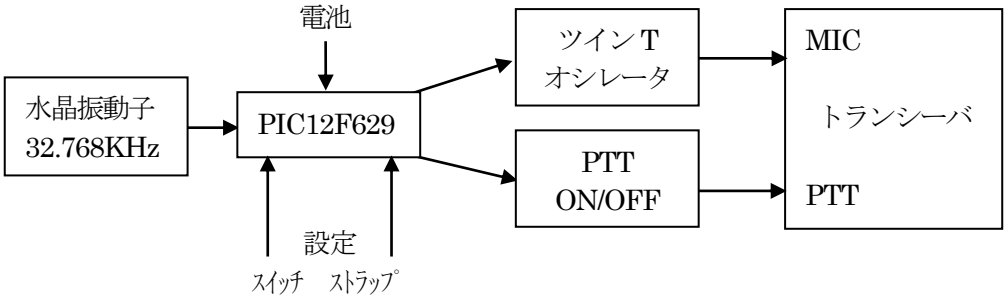
時計用の水晶振動子で 32.768KHz の安い部品があったのでそれを使うことにしたが、12F629 に水晶振動子をつけるためピンが 2 つ必要なので、先に書いた 5 ピンに 2 ピン足して 7 ピン。8 ピンのチップなので、あと 1 ピン何かに使える。(^^)

※使ってなかった最後のピンは 40 分タイマの使用／不使用の設定用に使った。

■ID ジェネレータの回路イメージをする

PIC でどこまで何をするか・・・小型軽量で作るためにソフトウェアでできる部分はソフトウェアで作るが、モールス信号のトーン(音)はハードウェアで作りたいという気持ちがあった。
ツイン T オシレータという発振回路を使うと正弦波に近い波形が出せるような記事を読んだし、PIC12F629 で正弦波を出すにはソフトウェアだけでは無理、いずれにしても何かハードウェアの助けが必要と考えたから。

そのようなことを考えて、下の回路(ブロック図)をイメージした。



ネットの記事の例ではツイン T オシレータを 1.5V の電池 1 本で消費電流が 1mA 未満で動作させている例があった。そんなに小さいパワーで動いている例があるなら、PIC のピン出力をそのままツイン T オシレータの電源にしてしまおう、と考えた。
PIC の本を読むと 1 ピンあたり 25mA を超えないよう電流を抑える必要はあるようで・・・オシレータだけで 25mA はまず超えることはないから大丈夫だろうと思う。
動作表示用の LED もツイン T オシレータと並列接続して PIC で点灯させるが、僕の製作では LED に流す電流を毎回抑えている(必要以上に流さない)ので 1mA 前後になり、これも含めて、まず大丈夫だろう。

ちなみに LED に流す電流は実際に電流を流して点灯させて明るさを確認して決めていることが多い。LED に流す電流を机上計算して抵抗値を決めて点灯させると期待の明るさと違うことがあり、仮に半固定抵抗器で電流を流しながら調整し、必要な明るさになったら、その抵抗値をテスタで確認して抵抗の定数を決めている。

既製品の TX は何番の TX にするか等、DIP-SW で設定してからスタートしていたような・・・これをやろうとすると、スイッチがたくさん必要になるが、基本的な機能だけに絞って ID ジェネレータのプログラムがスタートした後に何番の TX として使うか設定できるようにすれば、例えばプッシュスイッチで押すたびに切替できれば、そのスイッチは 1 個で良い。
ビーコンとして使いたいときには前述の切替スイッチを押したまま ID ジェネレータをスタートした時、ビーコンとして動作するようプログラムすればビーコンとして動作させるためのスイッチを別途用意しなくて良い。

受信機のテストや簡単な練習では、TX 設置後、ただちに電波を発射した方が良いと思うし、実際の競技大会では 40 分経過後送信を開始するよう設定していることが多いと思うので、今回試作の ID ジェネレータではストラップで即送信と 40 分後送信の選択が可能なように考えた。

毎分 55 秒から 60 秒までの 5 秒間はコールサインを入れることにした。電波法上必要だと思ったし、144MHz の FM でコールサイン無しで MO、MOE、MOI、MOS、MOH、MO5 のモールス符号だけ聞こえると「なんじゃこりゃ？」と総合通信局へ通報されるのは間違いないと思う。(苦笑)

■ブレッドボードで PIC の動作確認

ネットの記事を参考に、手持ちのパソコンに MPLAB　X を入れて、アセンブラを使って PIC にプログラムを書き込む練習をしばらくやってみた。

ネットの記事を見ながらパソコンで打ったプログラムが PIC に書き込まれて、期待の動作をしているのを見て、この後は目的の ID ジェネレータのプログラムを作ることができれば、何とかなるか・・・と思った。

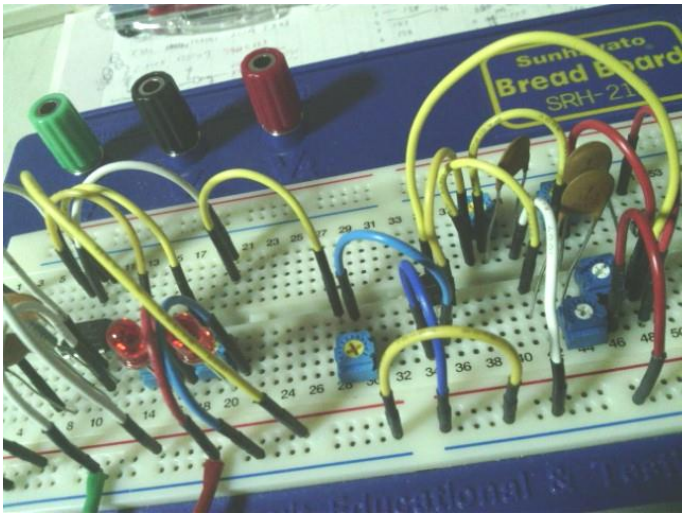
まず、スイッチの ON／OFF の状態取り込みと取り込んだ情報を元に LED を点灯させたり消灯させるプログラムなどを試してみた。

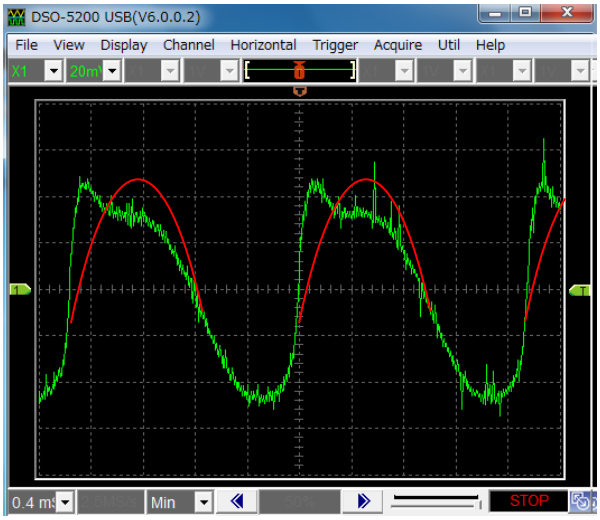
次に時刻に関するプログラムを考えた。TX の ID を 1 分毎に切り替えないといけない。マイコンは処理が速いので割り込み処理を使わなくても作れると思ったが、コンピュータの仕事をしている友人から、割り込み処理を使うべきだと聞いて、今回の ID ジェネレータの処理に使ってみた。
クロック周波数は 32.768KHz なので、これを内蔵カウンタで分周して 1 秒ごとに割り込み処理を行う。1 秒ごとに LED を点滅させるなど、PIC に入ってるタイマーを使う練習も兼ねて簡単なテストをやってから、実際に ID ジェネレータで使うプログラム作りを始めた。

何回もプログラムを書いてはテストし、修正してまたテストの繰り返しだったが、何とか PIC 単体で思うように動作するようになった。

■PIC 周辺の回路をブレッドボードに追加

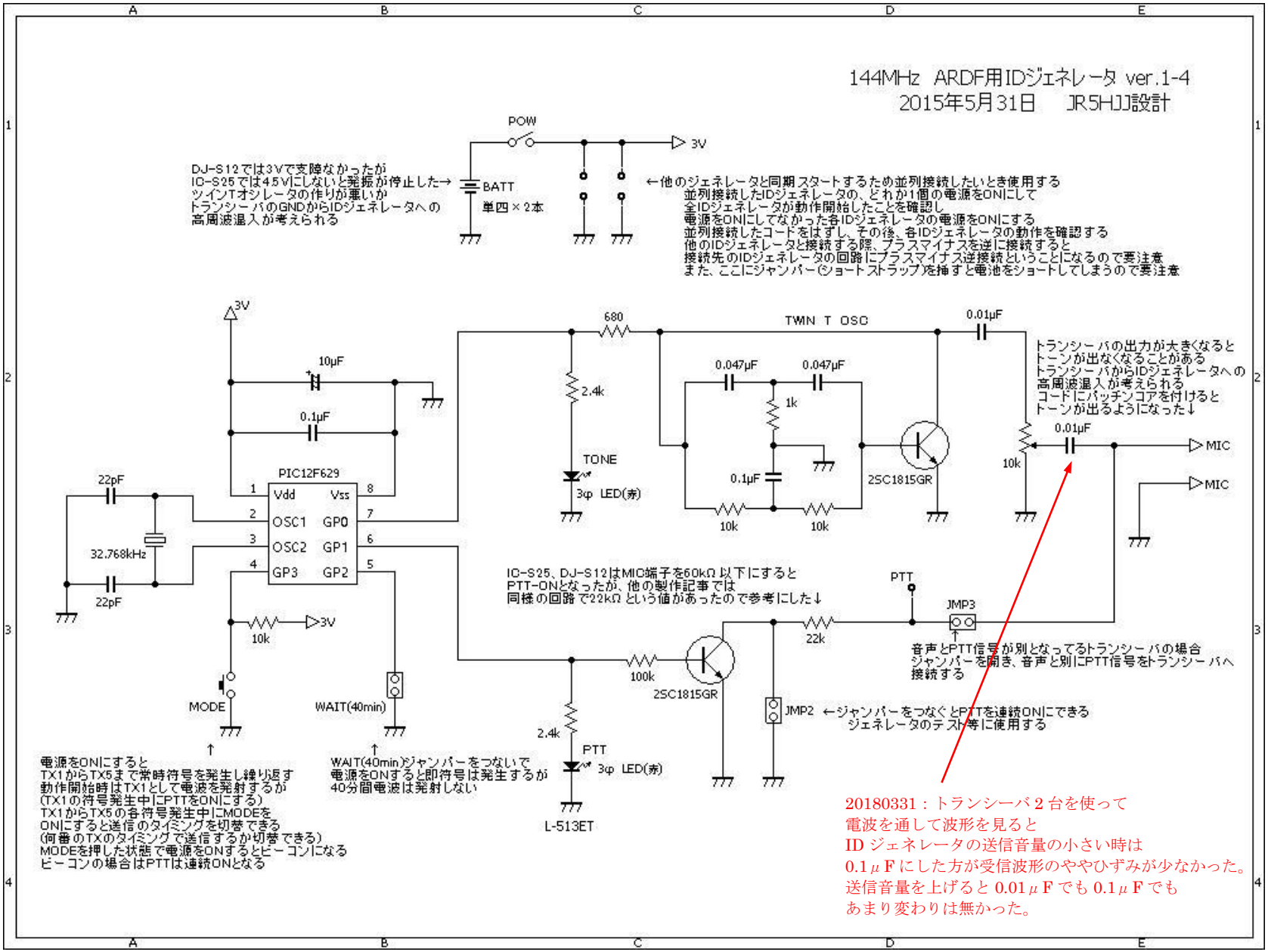
PIC 単体で動作確認をした後、PIC にモールス信号の音を作るツイン T 発振回路とトランシーバの送信を制御する回路を追加し、トランシーバと接続して、動作テストをしてみた。





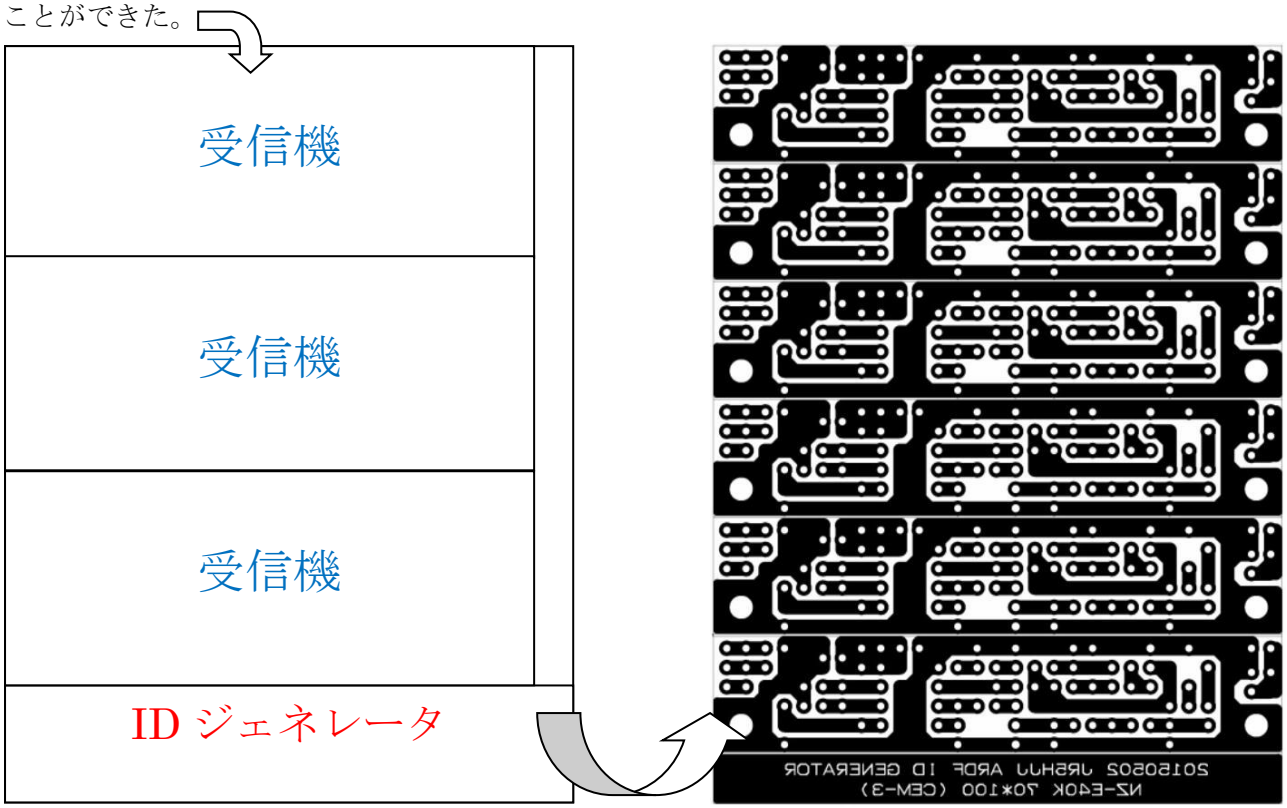
動いている!(喜)。しかし、音が(-_-)ツイン T オシレータ・・・の音がなんだか・・・正弦波ではない。
いろいろ定数を変えてみた。少しきれいな音に近づいたが、これは・・・まだ正弦波ではない、たぶん。
でも支障のある音でもない。オシロスコープで波形を見てみた。やっぱり歪んでいる。ツイン T オシレータ、正弦波が出せると書いてあったが、他の回路とつながっているし、ある程度やむをえないか・・・。
波形にヒゲがあるのは PIC からの雑音か、すぐ近くから無線機が送信している影響か？
定数をいろいろ変えて、ここら辺で(笑)支障なしというところで、定数を決めた。

■回路


144MHz ARDF用IDジェネレータ ver.1-4
2015年5月31日 JR5HJJ設計
POW
3V
BATT
単四×2本
DJ-S12では3Vで支障なかったが
IC-S25では4.5Vにしないと発振が停止した→
ツインTオシレータの作りが悪いか
トランシーバのGNDからIDジェネレータへの
高周波混入が考えられる
←他のジェネレータと同期スタートするため並列接続したいとき使用する
並列接続したIDジェネレータの、どれか1個の電源をONにして
全IDジェネレータが動作開始したことを確認し
電源をONにしてなかった各IDジェネレータの電源をONにする
並列接続したコードをはずし、その後、各IDジェネレータの動作を確認する
他のIDジェネレータと接続する際、プラスマイナスを逆に接続すると
接続先のIDジェネレータの回路にプラスマイナス逆接続ということになるので要注意
また、ここにジャンパー(ショートストラップ)を挿すと電池をショートしてしまうので要注意
3V
10μF
0.1μF
PIC12F629
Vdd
Vss
OSC1
GP0
OSC2
GP1
GP3
GP2
22pF
32.768kHz
22pF
10k
3V
MODE
WAIT(40min)
680
TWIN T OSC
0.01μF
2.4k
TONE
3φ LED(赤)
0.047μF
0.047μF
1k
0.1μF
10k
10k
2SC1815GR
トランシーバの出力が大きくなると
トーンが出なくなることがある
トランシーバからIDジェネレータへの
高周波混入が考えられる
コードにバッチンコアを付けると
トーンが出るようになった↓
0.01μF
10k
MIC
MIC
IC-S25、DJ-S12はMIC端子を60kΩ以下にすると
PTT-ONとなったが、他の製作記事では
同様の回路で22kΩという値があったので参考にした↓
PTT
JMP3
22k
音声とPTT信号が別となってるトランシーバの場合
ジャンパーを開き、音声と別にPTT信号をトランシーバへ
接続する
100k
2SC1815GR
JMP2 ←ジャンパーをつなぐとPTTを連続ONにできる
ジェネレータのテスト等に使用する
2.4k
PTT
3φ LED(赤)
L-513ET
電源をONにすると
TX1からTX5まで常時符号を発生し繰り返す
動作開始時はTX1として電波を発射するが
(TX1の符号発生中にPTTをONにする)
TX1からTX5の各符号発生中にMODEを
ONにすると送信のタイミングを切替できる
(何番のTXのタイミングで送信するか切替できる)
MODEを押した状態で電源をONするとビーコンになる
ビーコンの場合はPTTは連続ONとなる
WAIT(40min)ジャンパーをつないで
電源をONすると即符号は発生するが
40分間電波は発射しない
20180331：トランシーバ2台を使って
電波を通して波形を見ると
IDジェネレータの送信音量の小さい時は
0.1μFにした方が受信波形のややひずみが少なかった。
送信音量を上げると0.01μFでも0.1μFでも
あまり変わりは無かった。

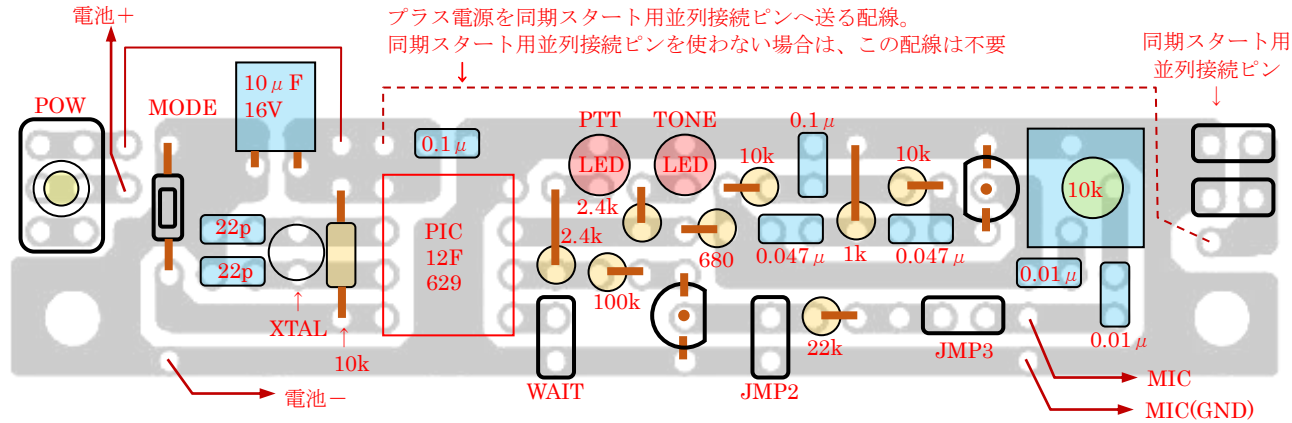
■プリント基板

サンハヤトの感光基板 NZ-E40K で ARDF の試作受信機の基板を作った際、基板 1 枚のスペースに受信機 3 台分とあまりが出た。そのあまりのスペースに ID ジェネレータ 1 個のプリント基板を作りこむことができた。

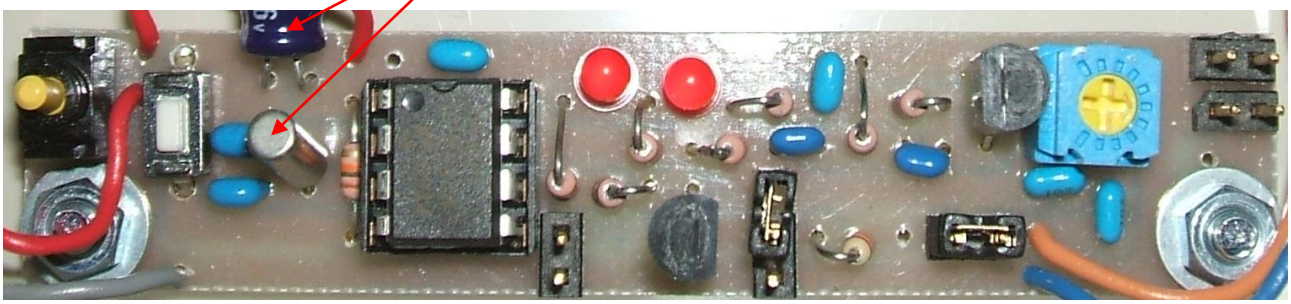


NZ-E40K に ID ジェネレータだけ作りこむと、こんな感じ。↑
NZ-E40K　1 枚で ID ジェネレータが 6 個分取れるので
6 個作ると TX5 個とビーコン 1 個でちょうど良いかも。(笑)

■部品の取り付け

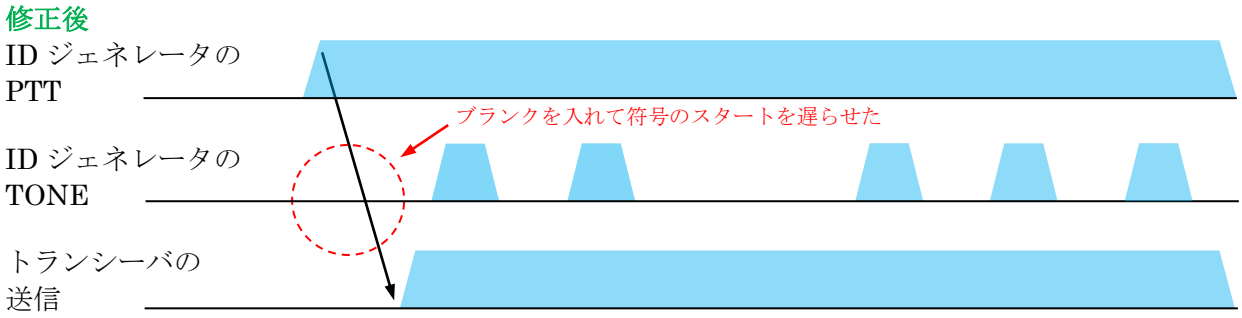


ケースの高さが低い時は、背の高い部品を横に倒してつけたり
斜めにつけたりしてフタが閉まるか事前に確認してください。



■手直し、その１

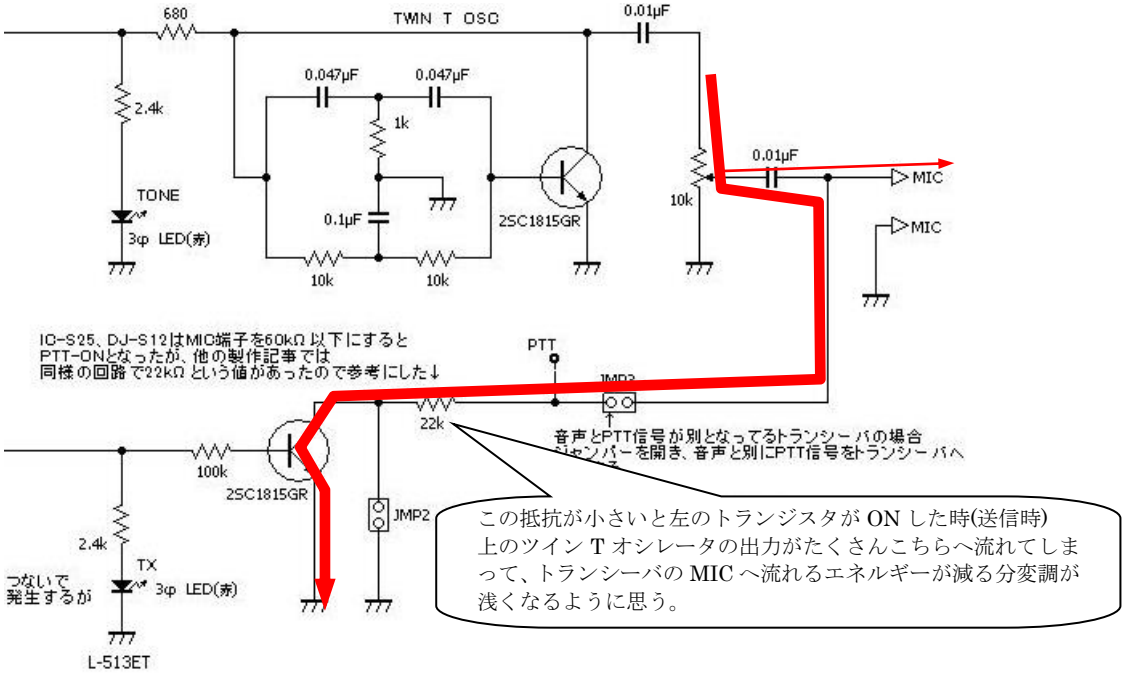
トランシーバを送信(PTT を ON)にした直後に符号を発生(トーンの発生)させると符号の頭切れが起きる(電波が安定して出る前に符号が出始める)のでトーン発生を PTT を ON した後、わずかに遅らせるようプログラムを修正。実際には符号の ON／OFF のパターンの始めにブランクをいれただけ。(笑)



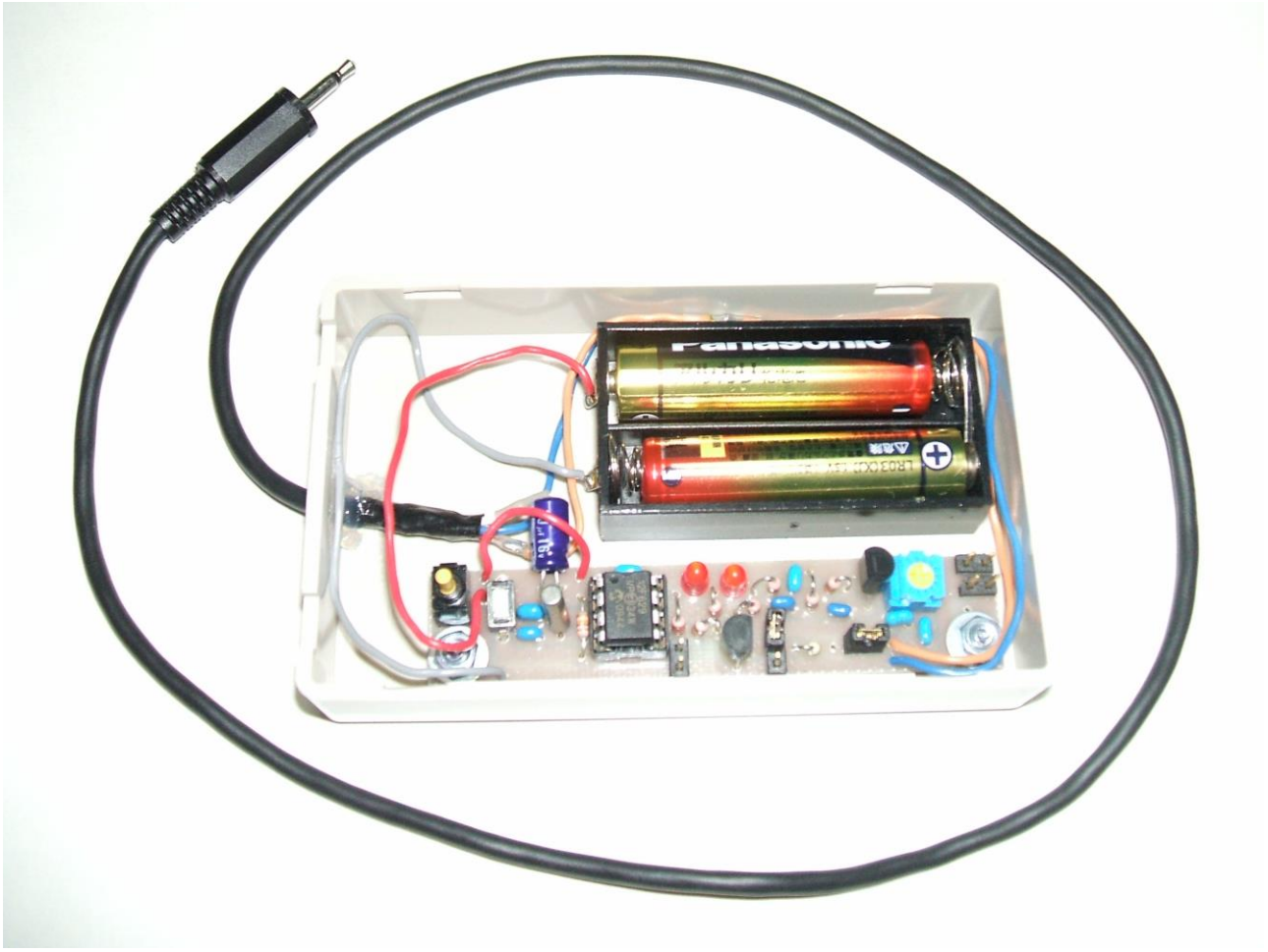
■手直し、その２

トランシーバを受信から送信に切り替える時、トランシーバの MIC と GND の間を何Ωの抵抗でつなぐか､については 1kΩや 22kΩ等メーカ機種によって違いがあって使用するトランシーバに合わせる必要がある。

最初 1kΩで試作した際、変調を浅く感じたので 22kΩとした。抵抗を小さくしすぎると、ツイン T オシレータの出力のうち MIC に行かずに PTT　ON／OFF 用のトランジスタを通してアースに流れてしまうエネルギー量が多くなるようだ。確実に PTT のコントロールができる範囲で、なるべく大きい値を採用した方が良いと思う。



■内部写真

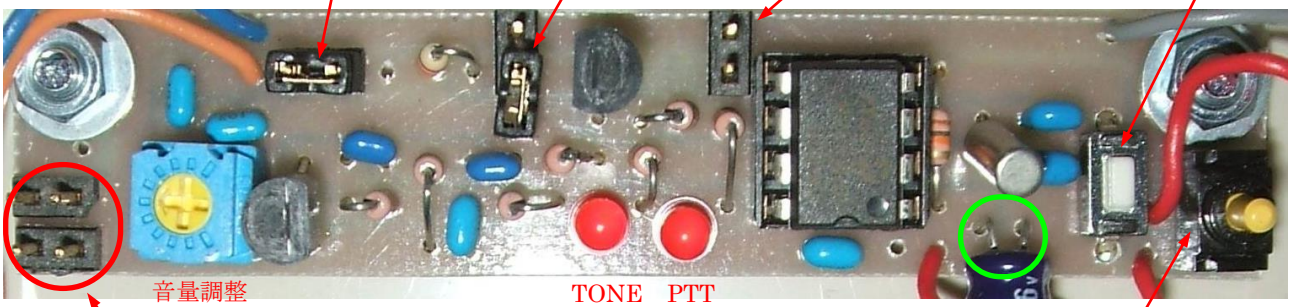


JMP3 ストラップ
ストラップを開放すると
MIC 端子への
PTT　ON／OFF 信号を
切ることができる

JMP2 ストラップ
ストラップを
ショートすると
ビーコンでなくても
無条件に PTT を
ON にできる

WAIT ストラップ
ストラップを
ショートすると
電源 ON 後４０分間
PTT が ON に
ならない

MODE 切替 SW
TX1～5 の切替と
押したまま電源を
ON するとゴール
ビーコンとして動作する



同期スタート用並列接続ピン(現在使っていない)

電源 SW

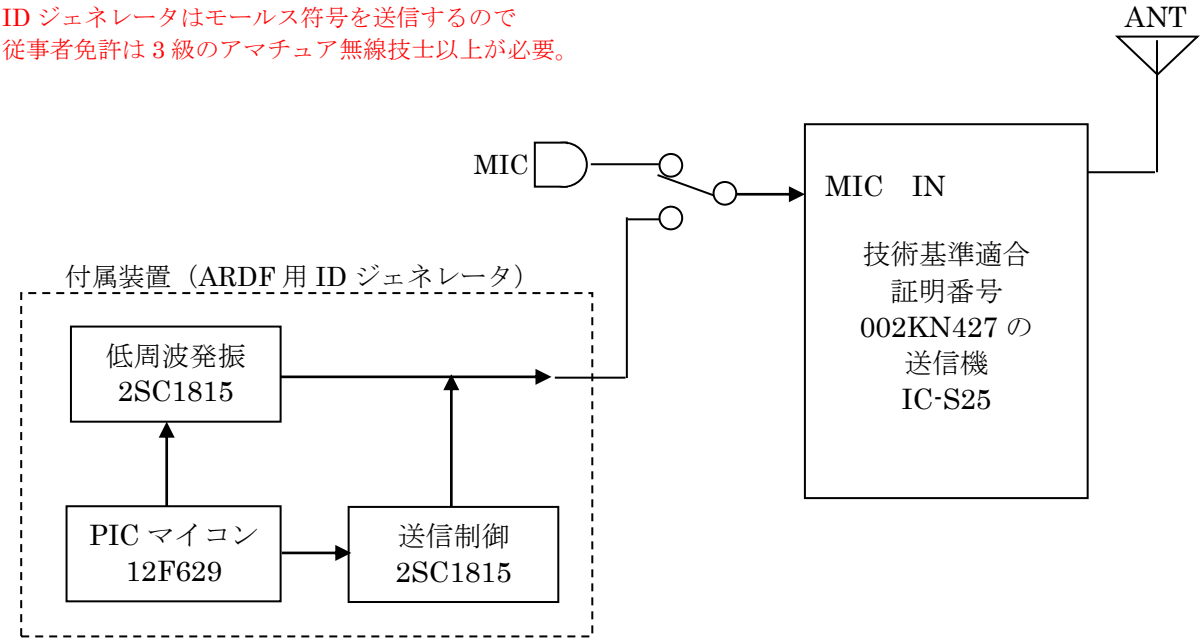
同期スタート用並列接続ピン○は複数の ID ジェネレータを同期スタートしたいときに使用するためのものであるが、たぶん個人では使わない(と思う)ので上の写真では回路のプラス電源は接続していない。
このピンが無くても、PIC の右下側にある電解コンデンサの足○を他の ID ジェネレータのコンデンサと並列接続すれば同期スタートできる。バッテリー上がりの自動車を救済するのと同様である。
同期スタート用並列接続ピン○へリード線でプラス電源を接続した場合は、同期スタート用並列接続ピンに余ったショートストラップを挿入等すると電源を短絡する(回路図参照)ので○の部分にストラップを挿入してはいけない。
回路を良く理解したうえで並列接続に使う配線をプラスマイナスで色分けするなど注意して使用する必要がある。
この記事を参考に製作する場合は同期スタート用並列接続ピン○を取付けしないことを推奨する。

■変更申請（届）

既に 144MHz は電波の型式 3VA で免許があって、ID ジェネレータを使うときに発射される F2A の電波は 3VA に含まれるので免許状の指定事項には変更は無いが、ID ジェネレータを免許を受けているトランシーバにつないで使う場合は付属装置をつける変更届が必要になった。
前にパケット通信用に付属装置の申請を出したことはあるが、ID ジェネレータは初めてだ。

下のような送信機系統図を書いてみた。(下の図は IC-S25)

※ID ジェネレータはモールス符号を送信するので
従事者免許は 3 級のアマチュア無線技士以上が必要。



付属装置（ARDF 用 ID ジェネレータ）は
PIC マイコンにより低周波発振回路の ON／OFF を行いトーン信号を使用したモールス符号を発生させます。
送信の ON／OFF は PIC マイコンから送信制御回路を通して MIC　IN へ PTT の ON／OFF 信号を送ります。
（トーン信号を使用するモールス符号の送信）

付属装置の諸元として下のような表を書いてみた。

ID ジェネレータの低周波発振器から出る周波数を測ると 550Hz だった。

装置の名称または種類	方式・規格など		使用する送信機
ARDF 用 ID ジェネレータ	副搬送周波数	550Hz	第 1 送信機 第 2 送信機
	送信符号	欧文モールス符号	
	符号の内容	MO MOE MOI MOS MOH MO5 呼出符号	

呼出符号は JR5HJJ

持っている 2 台の 144MHz のハンディ機が第 1 送信機と第 2 送信機になるのでこう書いた。

返信用封筒を同封して変更届を出してから約 3 週間後、総合通信局から封書が届いて、今回は無線設備の一部変更であり免許状や処分通知書・証票の発給が無いので返信封筒は返しますよ、という内容が書かれていた。
ちなみにネット情報では、今回同様に免許状の記載事項が変わらず処分通知書・証票も無い場合は、日付だけ更新された免許状を返信する総合通信局もあるらしいし、届出者が新しい日付の免許状は不要と指定して届けを出すケースもあるらしい。

■使い方

先にトランシーバの電源を入れてから ID ジェネレータの電源を入れる。

JMP2 は通常開放して使用する。
ハンディ機に接続して使うときは、ほとんどの場合 JMP3 は短絡して使うことになると思う。

●ゴールビーコンとして使いたいとき

モード切替スイッチ(プッシュスイッチ)を押したまま電源スイッチを入れて
すぐにモード切替スイッチをはなす。

※電源スイッチ ON 後、モード切替スイッチを離すのが遅くなると(約 1 秒)
1 分後に TX2 の符号に切り替わってしまうので電源 ON 後すぐモード切替スイッチをはなす。
1 分後に TX2 の符号に切り替わり PTT　OFF となった場合は再度やり直せば OK。

●通常の TX として使いたいとき

電源 ON で TX1～5 の符合を順に発生する。
電源 ON 後、TX1 として直ちに電波を発射したい時には WAIT ストラップは開放しておく。
電源 ON40 分後から TX1 として電波を発射したい時には WAIT ストラップを短絡しておく。

TX2 として使いたいときは、TX2 の符号発生中にモード切替スイッチを 1 秒以上押す。
TX3 として使いたいときは、TX3 の符号発生中にモード切替スイッチを 1 秒以上押す。
TX4 として使いたいときは、TX4 の符号発生中にモード切替スイッチを 1 秒以上押す。
TX5 として使いたいときは、TX5 の符号発生中にモード切替スイッチを 1 秒以上押す。

モードが切り替わると PTT　LED が点灯し送信状態となる。

※TX1～5 で動作中はゴールビーコンへの切替はできない。
ゴールビーコンとして使いたい時は、一度電源を切り、ゴールビーコンとして使う操作を行う。

●TX1～5 の符号で連続送信させたいとき

JMP2 を短絡するとトランシーバへの PTT 信号が無条件に ON になる。
PTT の LED 表示に関係なく、ID ジェネレータの動作や電源の ON／OFF の状態に関係なく
無条件に PTT が ON となる。

■使った感想等

割りと便利。(笑)

●本来、ハンディトランシーバは人が手で持って使うが、ID ジェネレータをつないでどこかへ置いた場合、人が手で持っていないのでトランシーバのホイップアンテナに対して接地が効いていないような感じがする。
ハンディトランシーバの出力を Lo(0.5W 程度)にして使うと問題ないが、Hi(5W 程度)にすると、電波にモールスの音が乗らず、クリック音が聞こえる。

ハンディトランシーバは通常 1／4λの付属アンテナを使うことが多いと思う。付属アンテナを使って ID ジェネレータを動作させた場合、ID ジェネレータにつながってるコードがアース代わりになって、ID ジェネレータに向かって高周波が流れるようで・・・。
それが原因で ID ジェネレータ内のツイン T オシレータに本来流れる低周波以外の電流が流れて発振が止まってしまうのだろうか・・・。

コードを手で握ったら発振停止が軽減できたので、コードをパッチンコアに巻いてみたらうまく動作するようになった。小さいパッチンコアがあればケース内に入れられるが、あるだろうか・・。
いや、マイクロインダクタでも効くだろうか。
1／4λのアンテナでなければ大丈夫な気がする。ARDF で使うターンスタイルアンテナ等を使えば、パッチンコアは不要かもしれない。

●ID ジェネレータの PTT が ON の状態でハンディトランシーバの電源を ON にすると、トランシーバが送信状態にならない。(DJ-S12、IC-S25 で確認済み)
トランシーバの仕様で、安全対策としてそうなっているようだ。
ID ジェネレータを使うときには先にトランシーバの電源を ON にしておく必要がある。

●ID ジェネレータの遠隔操作を考えてみた。
せっかく ID ジェネレータをトランシーバにつなぐなら、MIC や PTT 以外に SP にもつないで簡単な遠隔操作もできるのでは？と考えた。

例えば簡単にシングルラジコンのように使うとか。
送信以外の時間に連続 5 秒以上のトーンがトランシーバから入ると ID ジェネレータが待機状態になるとか。
待機状態の時、特定の時間帯はトランシーバからのトーンを受信し、何かトーンが入ると動作を再開するとか。

■注意

この文書や当方サイトの情報および情報を元に製作した機器によって、いかなる損害を被られた場合であっても、当方は一切の責任を負いません。

■PIC のプログラム

以下、PIC12F629 に書き込んだ ID ジェネレータのプログラム。
試行錯誤して作って、使わなかった記述や不用な命令があったりコメントが変な部分もあると思うがとりあえず MPLAB　X の画面を下にコピーペーストした。(内容は 2015 年 6 月 20 日現在)

2015 年 6 月 20 日版のプログラムは 5 秒間におよそ 80 ビットの情報を送信するように組んであったが、モールス符号でたくさんのビット数を必要とするコールサイン(数字のゼロや Q や J や Y がたくさん含まれるコールサイン)では 80 ビットでは不足するため 5 秒間でおよそ 112 ビットの情報を送信するよう変更した。
また、MOX の部分とコールサインの部分で符号の速度を切替できるようにした。たくさんのビット数を必要とするコールサインではコールサインの部分だけ符号の速度を上げる等可能となった。
最新版のプログラム(2017 年 11 月 11 日現在)は
http://wwwb.pikara.ne.jp/potter-2005/kousaku/idg20171111.txt

```
;PIC12F629 を使った ARDF 用 ID ジェネレータ
;1 ピン(Vdd)
;2 ピン(GP5)=32.768KHz 水晶振動子
;3 ピン(GP4)=32.768KHz 水晶振動子
;4 ピン(GP3)=モード切替スイッチ SW1
;5 ピン(GP2)=40 分待ちスイッチ SW2
;6 ピン(GP1)=送信制御
;7 ピン(GP0)=トーン制御
;8 ピン(Vss)
      LIST P=12F629
      INCLUDE P12F629.INC
CB  = _CPD_OFF     ;データメモリを保護しない
CB &= _CP_OFF      ;プログラムメモリを保護しない
CB &= _BODEN_ON    ;電源電圧が 2V 以下でリセットをかける
CB &= _MCLRE_OFF ;3 番ピンを外部リセットに使わない
CB &= _PWRTE_ON    ;電源投入後電圧上昇を確認後 72msec 待つ
CB &= _WDT_OFF     ;ウォッチドッグタイマを使わない
CB &= _LP_OSC      ;発振モードは LP(水晶振動子、省電力モード)
      __CONFIG CB
      __IDLOCS B'00000001'
;汎用レジスタの指定
FUGOU0 EQU 0x20;符号 0 は汎用レジスタ 0X20 を使う
FUGOU1 EQU 0x21;符号 1 は汎用レジスタ 0X21 を使う
FUGOU2 EQU 0x22;符号 2 は汎用レジスタ 0X22 を使う
FUGOU3 EQU 0x23;符号 3 は汎用レジスタ 0X23 を使う
FUGOU4 EQU 0x24;符号 4 は汎用レジスタ 0X24 を使う
FUGOU5 EQU 0x25;符号 5 は汎用レジスタ 0X25 を使う
FUGOU6 EQU 0x26;符号 6 は汎用レジスタ 0X26 を使う
FUGOU7 EQU 0x27;符号 7 は汎用レジスタ 0X27 を使う
FUGOU8 EQU 0x28;符号 8 は汎用レジスタ 0X28 を使う
FUGOU9 EQU 0x29;符号 9 は汎用レジスタ 0X29 を使う
TXNO     EQU 0x30;送信時間帯データは汎用レジスタ 0X30 を使う
MODE     EQU 0x31;モードデータは汎用レジスタ 0X31 を使う
SEC1     EQU 0x32;秒は汎用レジスタ 0X32 を使う
MIN1     EQU 0x35;分は汎用レジスタ 0X35 を使う
LOOPW    EQU 0x37;時間待ちループ A は汎用レジスタ 0X37 を使う
MIN40    EQU 0x38;40 分後オール 1 にする
WORDNO EQU 0x39;送信符号の何番目のワード(0X20?0X29)
BITNO    EQU 0x40;送信符号の何番目のビット(10 バイトの 79 ビット)
WTEMP    EQU 0x41;W レジスタの退避場所
STEMP    EQU 0x42;STATUS レジスタの退避場所
DOOUT    EQU 0x43;DO 作業用
CSF      EQU 0x44;CALLSIGN 設定フラグ
;START
      ORG B'00000000'
      GOTO SETUP
;割込発生時はここ
      ORG B'00000100'
;お決まりらしい退避処理
```

```
	MOVWF WTEMP;W レジスタを退避
	MOVF STATUS,W;STATUS レジスタを W レジスタへ代入
	MOVWF STEMP;W レジスタの内容を STEMP へ代入
;
	CALL WATCH
;
;お決まりらしい復帰準備
	MOVF STEMP,W;割込み前の STATUS レジスタ復帰
	MOVWF STATUS
	SWAPF WTEMP,F;割込み前の W レジスタを復帰
	SWAPF WTEMP,W;フラグが変化しない SWAP を使用
	RETFIE
;
;セットアップ
SETUP:
;バンク 0 に切替
	BCF STATUS,RP0
;一度 GPIO を 0 にする
	CLRF GPIO
;コンパレータを不使用にしてデジタル I/O を有効にする
	MOVLW B'00000111'
	MOVWF CMCON
;バンク 1 に切替
	BSF STATUS,RP0
;GP2,3 は入力用に設定
	MOVLW B'00001100'
	MOVWF TRISIO
;GPIO を内部プルアップする
	BCF OPTION_REG,7;
;タイマー0 の設定
;タイマー0 は 1 秒タイマーとして使う
;1/32768 秒×4(TOCS は 0)×32×256(PSA は 0、PS2 は 1、PS1 は 0、PS0 は 0)
	BCF OPTION_REG,T0CS;T0CS を 0 にする
	BCF OPTION_REG,PSA;PSA を 0 にする
	BSF OPTION_REG,PS2;PS2 を 1 にする
	BCF OPTION_REG,PS1;PS1 を 0 にする
	BCF OPTION_REG,PS0;PS0 を 0 にする
;バンク 0 に切替
	BCF STATUS,RP0
;MODE に 00000001 を代入し TX モードとして TX1 の準備
	MOVLW B'00000001'
	MOVWF MODE
;符号ワード番号をリセット
	CLRF WORDNO;WORD 番号を 0 にする
;符号ビット番号を設定
	MOVLW B'10000000';0 を W レジスタに入れる
	MOVWF BITNO;BITNO に W レジスタの内容を入れる
;TXNO に 00000001 を代入し TX1 から送信の準備
	MOVLW B'00000001'
	MOVWF TXNO
;送信コードの準備
	CALL TXCODE
;SW1 が押されていればビーコン用に設定変更
	BTFSS GPIO,GPIO3;GP3 が Hi であれば(SW が押されてなければ)1 命令スキップ
	CALL BEACONMODE;GP3 が Lo であれば(SW が押されていれば)ビーコン設定
```

```
;時刻を 0 分 0 秒にセットする
	CLRF MIN1;分を 0 にする
	CLRF SEC1;秒を 0 にする
;40 分フラグを 0 にする
	CLRF MIN40;MIN40 を 0 にする
;SW2 が ON ならば即送信設定
	BTFSS GPIO,GPIO2;GP2 が Hi であれば(SW が ON ならば)1 命令スキップ
	GOTO DOOUTCODE
	MOVLW B'11111111'
	MOVWF MIN40
DOOUTCODE:;DO 出力コードオールゼロ設定
	CLRF DOOUT
;タイマー0 をリセット
	CLRF TMR0
;GIE(グローバル割込)と T0IE(タイマー0 割込)を有効にする
	MOVLW B'10100000'
	MOVWF INTCON
;コールサインフラグを 1 にする
	CLRF CSF
;
;==========MAIN==========
MAIN:
;40 分経過の確認
	MOVLW B'00100111';W レジスタに 39 を入れる
	SUBWF MIN1,W;MIN1 から W レジスタの内容を引いて結果を W レジスタに入れる
	BTFSC STATUS,C;40 分未満なら C が 0 なのでスキップ
	CALL MIN40ALL1
;TXNO と MODE の AND が 0 でないなら GP0 を Hi にする
TXMODE:
	MOVF TXNO,W;TXNO の内容を W レジスタへ代入する
	ANDWF MODE,W;TXNO と MODE の値の AND を W レジスタへ代入する
	BTFSS STATUS,Z;Z フラグが 1 であれば 1 行スキップ
	GOTO GP0ON
	GOTO GP0OFF
;電波を発射する
GP0ON:
	MOVF DOOUT,W;W レジスタに DOOUT の値をセットし
	IORLW B'00000010';W レジスタと 00000010 の論理和を W レジスタに入れる
	ANDWF MIN40,W;MIN40 と W レジスタと論理積を W レジスタに入れる
	MOVWF DOOUT;W レジスタの値を DOOUT に転記
	MOVWF GPIO;W レジスタの値を GPIO へ出力
	GOTO CODE55
;電波を停波する
GP0OFF:
	MOVF DOOUT,W;W レジスタに DOOUT の値をセットし
	ANDLW B'00000001';W レジスタと 00000001 の論理積を W レジスタに入れる
	ANDWF MIN40,W;MIN40 と W レジスタと論理積を W レジスタに入れる
	MOVWF DOOUT;W レジスタの値を DOOUT に転記
	MOVWF GPIO;W レジスタの値を GPIO へ出力
	GOTO CODE55
;
;コールサインを出すタイミングかチェックする
CODE55:;毎分 55 秒のチェック
	DECFSZ CSF,0;CSF から 1 引いて 0 ならばスキップ
	GOTO TONE0
```

```
        CALL CALLSIGN;CALLSIGN を CALL
;トーン処理
TONE0:
        CALL TONE
;時間待ち
        CALL WAIT
        GOTO MAIN
;==========SUB==========
TXCODE:
;符号を汎用レジスタに設定する
TXMOCODE:
        MOVLW B'00011111'
        MOVWF FUGOU0
        MOVLW B'10011111'
        MOVWF FUGOU1
        MOVLW B'10000001'
        MOVWF FUGOU2
        MOVLW B'11111001'
        MOVWF FUGOU3
        MOVLW B'11111001'
        MOVWF FUGOU4
        MOVLW B'11111000'
        MOVWF FUGOU5
;その時間帯のコードを選ぶ
TXNOCODE:
        BTFSC TXNO,0;TX1 の時間帯でないなら(0 ビットが 0 なら)スキップ
        CALL TX1CODE;TX1 の時間帯なら TX1CODE
        BTFSC TXNO,1;TX2 の時間帯でないなら(1 ビットが 0 なら)スキップ
        CALL TX2CODE;TX2 の時間帯なら TX2CODE
        BTFSC TXNO,2;TX3 の時間帯でないなら(2 ビットが 0 なら)スキップ
        CALL TX3CODE;TX3 の時間帯なら TX3CODE
        BTFSC TXNO,3;TX4 の時間帯でないなら(3 ビットが 0 なら)スキップ
        CALL TX4CODE;TX4 の時間帯なら TX4CODE
        BTFSC TXNO,4;TX5 の時間帯でないなら(4 ビットが 0 なら)スキップ
        CALL TX5CODE;TX5 の時間帯なら TX5CODE
        RETURN
TX1CODE:
        MOVLW B'00011000'
        MOVWF FUGOU6
        MOVLW B'00000000'
        MOVWF FUGOU7
        MOVLW B'00000000'
        MOVWF FUGOU8
        MOVLW B'00000000'
        MOVWF FUGOU9
        RETURN
TX2CODE:
        MOVLW B'00011001'
        MOVWF FUGOU6
        MOVLW B'10000000'
        MOVWF FUGOU7
        MOVLW B'00000000'
        MOVWF FUGOU8
        MOVLW B'00000000'
        MOVWF FUGOU9
        RETURN
```

←左から 1 ビットづつ読み込んで 0 の時は
ツイン T オシレータを OFF、1 の時はツイン T オシレータを ON にする。
最初の 0 三つの 3 ビットはトランシーバを送信にしたあと
3 ビット分の時間、待ってから（ツイン T オシレータを OFF にしてから）
MO（— —　— — —）と送信させるため。

このプログラムでは 80 ビットの情報を約 5 秒で送信する。
手持ちのトランシーバでいろいろ試したところ頭切れの問題は
1 ビット分(0.0625 秒)符号発生を遅らせると充分だったが
毎分 55 秒から発生させるコールサイン(JR5HJJ)の最後と
次の MO の符号の間隔を短く感じたので 0 三つの 3 ビットとした。

```
TX3CODE:
    MOVLW B'00011001'
    MOVWF FUGOU6
    MOVLW B'10011000'
    MOVWF FUGOU7
    MOVLW B'00000000'
    MOVWF FUGOU8
    MOVLW B'00000000'
    MOVWF FUGOU9
    RETURN
TX4CODE:
    MOVLW B'00011001'
    MOVWF FUGOU6
    MOVLW B'10011001'
    MOVWF FUGOU7
    MOVLW B'10000000'
    MOVWF FUGOU8
    MOVLW B'00000000'
    MOVWF FUGOU9
    RETURN
TX5CODE:
    MOVLW B'00011001'
    MOVWF FUGOU6
    MOVLW B'10011001'
    MOVWF FUGOU7
    MOVLW B'10011000'
    MOVWF FUGOU8
    MOVLW B'00000000'
    MOVWF FUGOU9
    RETURN
CALLSIGN:;JR5HJJ
    MOVLW B'10111011'
    MOVWF FUGOU0
    MOVLW B'10111000'
    MOVWF FUGOU1
    MOVLW B'10111010'
    MOVWF FUGOU2
    MOVLW B'00101010'
    MOVWF FUGOU3
    MOVLW B'10100010'
    MOVWF FUGOU4
    MOVLW B'10101000'
    MOVWF FUGOU5
    MOVLW B'10111011'
    MOVWF FUGOU6
    MOVLW B'10111000'
    MOVWF FUGOU7
    MOVLW B'10111011'
    MOVWF FUGOU8
    MOVLW B'10111000'
    MOVWF FUGOU9
;符号ワード番号をリセット
    CLRF WORDNO;WORD 番号を 0 にする
;符号ビット番号を設定
    MOVLW B'10000000';0 を W レジスタに入れる
    MOVWF BITNO;BITNO に W レジスタの内容を入れる
;コールサインをセットしたら CSF の 0 ビットをを 0 にする
```

毎分 55 秒にコールサインを送信するための設定部分
左のパターンでは JR5HJJ と送信する。
この記事を参考に製作する場合は
この部分のビットパターンを自分のコールサインに合うよう変更が必要。
コールサインに対するビットのパターン 80 ビットに収まらない場合は
プログラム自体を 5 秒で 90 ビット送信する等に変更する必要あり。

```
	BCF CSF,0
	RETURN
BEACONMODE:;ビーコン動作
;MODE に 00011111 を代入
	MOVLW B'00011111'
	MOVWF MODE;ビーコン動作のための設定
;MO の後をオールゼロする
	MOVLW B'00000000'
	MOVWF FUGOU6
	MOVLW B'00000000'
	MOVWF FUGOU7
	MOVLW B'00000000'
	MOVWF FUGOU8
	MOVLW B'00000000'
	MOVWF FUGOU9
	RETURN
;
;符号発生
TONE:
;目的のビットを検査し目的のビットが 1 ならトーンを ON にする
	MOVLW B'00100000';W レジスタに 0X20 を入れる
	ADDWF WORDNO,W;WORDNO を W レジスタに加算し W レジスタに入れる
	MOVWF FSR;0X20 に WORDNO を足した値を FSR に入れる
	MOVF INDF,W;FSR の示すレジスタの値を W レジスタに入れる
	ANDWF BITNO,W;W レジスタの値と BITNO の値の AND を W レジスタに入れる
	BTFSS STATUS,Z;Z が 1 なら 1 行スキップ
	GOTO TONEON;トーン ON
	GOTO TONEOFF;トーン OFF
TONEON:
	MOVF DOOUT,W;W レジスタに DOOUT の値をセットし
	IORLW B'00000001';W レジスタと 00000001 の論理和を W レジスタに入れる
	MOVWF DOOUT;W レジスタの値を DOOUT に転記
	MOVWF GPIO;W レジスタの値を GPIO へ出力
	GOTO NEXTBIT
TONEOFF:
	MOVF DOOUT,W;W レジスタに DOOUT の値をセットし
	ANDLW B'00000010';W レジスタと 00000010 の論理積を W レジスタに入れる
	MOVWF DOOUT;W レジスタの値を DOOUT に転記
	MOVWF GPIO;W レジスタの値を GPIO へ出力
NEXTBIT:
;次のビット位置を計算
	BCF STATUS,C;STATUS の C ビットを OFF
	RRF BITNO,F;BITNO のビットを右シフト
	BTFSC STATUS,C;C が 0 なら 1 行スキップ
	CALL NEXTWORD;次のワード
	RETURN
NEXTWORD:
	INCF WORDNO,F;WORDNO を 1 加算、結果を WORDNO へ入れる
	MOVLW B'10000000';W レジスタに 10000000 を入れる
	MOVWF BITNO;W レジスタの内容を BITNO に入れる
	MOVLW B'00001010';W レジスタに 10 を入れる
	SUBWF WORDNO,W;WORDNO から W レジスタの内容を引いて結果を W レジスタに入れる
	BTFSC STATUS,Z;10 でないなら Z が 0 なのでスキップ
	CLRF WORDNO;WORDNO を 0 に戻す
	RETURN
```

```
;時計処理
WATCH:
	INCF SEC1,F;秒を 1 加算、結果を SEC1 に入れる
	MOVLW B'00111100';W レジスタに 60 を入れる
	SUBWF SEC1,W;SEC1 から W レジスタの内容を引いて結果を W レジスタに入れる
	BTFSC STATUS,C;60 秒未満なら C が 0 なのでスキップ
	CALL SEC60
	BCF INTCON,T0IF;1 秒経過フラグをクリヤ
;ＳＷ１チェック
	BTFSS GPIO,GPIO3;GP3 が Hi であれば(SW1 が押されてなければ)1 命令スキップ
	CALL MODECHANGE;押されているならモードを変更する
;SEC1 が 55 なら
	MOVLW B'00110111';W レジスタに 55 を入れる
	SUBWF SEC1,W;SEC1 から W レジスタの内容を引いて結果を W レジスタに入れる
	BTFSC STATUS,Z;55 秒でないなら Z が 0 なのでスキップ
	BSF CSF,0
	RETURN
SEC60:
	CLRF SEC1;秒を 0 にする
	INCF MIN1,F;分を加算、結果を MIN1 に入れる。時の処理はしない
TXNOSHIFT:
	BCF STATUS,C;STATUS レジスタの C ビットを OFF
	RLF TXNO,F;TXNO を左シフト
	BTFSC TXNO,5;5 ビットが 0 ならスキップ
	CALL TXNO1
	CALL TXCODE;TXNO に合う送信コードを設定
;BEACON か
	MOVLW B'00011111';W レジスタに 00011111 を入れる
	SUBWF MODE,W;MODE から W レジスタの内容を引いて結果を W レジスタに入れる
	BTFSC STATUS,Z;ビーコンモードでなければ Z が 0 なのでスキップ
	CALL BEACONMODE
;符号ワード番号をリセット
	CLRF WORDNO;WORD 番号を 0 にする
;符号ビット番号を設定
	MOVLW B'10000000';0 を W レジスタに入れる
	MOVWF BITNO;BITNO に W レジスタの内容を入れる
	RETURN
TXNO1:;TXSQ を 1 に戻す
	MOVLW B'00000001';W レジスタに 1 を入れる
	MOVWF TXNO;TXNO に W レジスタの値を入れる
	RETURN
;TX のモードを変更する TX1?TX5
MODECHANGE:
	MOVF TXNO,W;TXNO の内容を W レジスタへ代入する
	MOVWF MODE;W レジスタの内容を MODE へ代入する
	RETURN
;
MIN40ALL1:
	MOVLW B'11111111';W レジスタに 1 を入れる
	MOVWF MIN40;MIN40 に W レジスタの値を入れる
	RETURN
;時間待ち 1
WAIT:
	MOVLW B'01011101';93 にしてみる
```

この 93 と次ページの LOOPA の下の NOP の数で符号の送信速度を調整した。
↓

```
;5 秒 80 ビットとして 1 ビットあたり 0.0625 秒、
      MOVWF LOOPW
LOOPA:
      NOP
      NOP
      DECFSZ LOOPW,F;LOOPW から 1 引いて、引いた結果が 0 なら 1 行スキップ
      GOTO LOOPA
      RETURN
;周波数のキャリブレーション
;      BSF STATUS,RP0;STATUS レジスタの RP0 ビットを 1 にして BANK1 に切替
;      CALL 3FFh;Get the cal value
;      MOVWF OSCCAL;Calibrate
      END
```

↑
ID ジェネレータは
32.768KHz の外付け水晶振動子を使っているので
この部分は使ってない。

【修正メモ】
20150824：ID ジェネレータの使い方を追記
20171111：プログラムバージョンアップに関する追記

上記(最初)のプログラムは速度固定で 5 秒間におよそ 80 ビットの情報を送信するプログラムになっていた。

これは 1 回 60 秒の送信のうち最後の 5 秒間(5 秒と決めなくても良いのであるが・・・)で自分のコールサインを送信しようとすると何ビット必要か、ということも考えたところ JR5HJJ だと
・－－－　・－・　・・・・・　・・・・　・－－－　・－－－
となり、モールス信号の音の出るタイミングだけビットを 1 にして並べると
10111011101110001011101000101010101000101010100010111011101110001011101110111
で 77 ビット必要となる。
実際には TX1～5 の番号を識別する MOE、MOI、MOS、MOH、MO5 などの符号の後にコールサインが続くし、ゴール付近に設置されるビーコンでは連続送信であるためコールサインの後にも MO の符号が続くのでコールサインの前後には無音の時間がわずかでも必要になる。
よって、コールサインの前後数ビットは無音とするための 0 が必要と考えて 80 ビットとした。

ところが
日本のアマチュア無線のコールサインには 5 エリア(四国)以外は必ず－(長点)が含まれる分ビット数が多く必要で、アルファベットに J(・－－－)や Q(－－・－)や Y(－・－－)のように－がたくさんあるコールサインをプログラムしようとすると、5 秒間で 80 ビットでは符号の速度が不足する。
(－がたくさんあると毎分 55 秒から 60 秒の 5 秒間にコールサインが収まらない)
コールサインを 5 秒でなく 6 秒や 7 秒かけて送信しても良いのであるが・・・。

そこで速度最大で 5 秒間に 112 ビットとして、長い符号のコールサインでも対応できるようにバージョンアップした。

細かい話をすれば、毎分 0 秒から 55 秒の MOE、MOI、MOS、MOH、MO5 などの符号を送信する時間帯と毎分 55 秒から 60 秒のコールサインを送信する時間帯で符号の速度を切り替えられるようにした。
切替可能にしたのは短いコールサインでは少し遅く、長いコールサインでは少し早く送信することで 5 秒間の符号の納まり(聴こえ)が良いのでは？と思ったから。

バージョンアップ後のプログラムは
http://wwwb.pikara.ne.jp/potter-2005/kousaku/idg20171111.txt

20180321　ひずみに関する追記(回路図のコンデンサ 2 か所)、部品取り付けに関する追記
20180331　回路図のひずみに関する追記を修正