Projet de trafic haut débit (THD)

de Victor F1BIU

Un projet de trafic packet haut débit est en cours dans les régions Île de France et Alsace.

Le THD n'est pas nouveau, les OMs américains trafiquent depuis longtemps en haut débit. Les allemands avec des TX interlinks à 38 K.
Plus près de chez nous, en Slovénie, l'utilisateur final trafique en 38 Kb/s depuis 5 ans, les links internodes étant passés à un débit de 1.2 Mb/s. Les nouveaux satellites amateurs démarrent le packet à 38 Kb/s et plus. (ex/ le nouveau satellite amateur TMSAT-1, le projet RUDAK sur P3D à 156 Kb/s, le projet TNUM du satellite français Maëlle à 250 Kb/s et d'autres à venir..)

Pourquoi le trafic haut débit n'a pas encore démarré en France? Je donne ici mon avis, mais comme je débarque dans le monde des packetteurs, il y a probablement d'autres raisons qui m'échappent.

La 1ère raison est d'origine réglementaire. Il faut se rappeler que l'ancienne réglementation ne permettait pas d'occuper des largeurs de canaux supérieures à 25Khz pour faire du packet. La nouvelle réglementation n'impose plus rien au niveau des répartitions de modes et canaux, ça devient une affaire interne aux OMs. Il existe de la place surtout à partir de 1.2Ghz où on ne dérange pas les autres utilisateurs. La 2ème raison est que le principal frein au développement du trafic haut débit est dû aux voies radio qui sont inadaptées pour les grandes vitesses. Le comble est que finalement c'est la partie radio que ne sait pas traiter le radioamateur (hi). Par contre, pour le réseau ou les modems, nous avons pas mal d'OMs initiés. L'amateur packet ne peut même pas se procurer des tx/rx large bande chez les fabricants, ces appareils n'existent pas. C'est pour cela que dans la partie technique de cet article je développe seulement la partie TX/RX.

Pourquoi faire du haut débit?

-Pour pouvoir résoudre le problème de l'engorgement du trafic 1200b dans les grandes agglomérations. Plus le débit est élevé, plus vite est transmis un message et donc moins de temps d'occupation hertzienne.

-Pour recevoir les très belles images haute définition envoyées par les nouveaux satellites amateurs avec une montée à 9600b et une descente à 38.4 Kb en UHF (TMSAT-1)

-Pour pouvoir s'échanger des fichiers volumineux, à 38 Kb/s on peut se permettre d'expédier à son correspondant un fichier de 1 Moctets en moins de 4 minutes. A ce propos, les OMs slovènes se sont très vite habitués aux possibilités de transfert de gros

fichiers, à tel point qu'ils commencent à être encombrés à 38 Kb/s, ils envisagent de monter en débit. Il est vrai qu'aujourd'hui nous sommes demandeurs de fichiers contenant du texte, des séquences sonores et des images, comme pour les pages WEB d'Internet.

-Pour concurrencer gratuitement Internet qui est un gros danger pour le packet amateur, surtout avec les nouvelles baisses de tarifications des télécoms. Comment peut-on encourager un jeune à se lancer dans le packet, quand il a la possibilité de communiquer avec le monde entier pour quelques francs par mois via Internet? Avec le packet haut débit on ira plus vite qu'internet.

-Pour expérimenter, tout simplement. Il y a beaucoup d'inconnu dans le THD, surtout en milieu urbain perturbé, la propagation 1.2 Ghz en ville, ses rotations de phase, le taux d'erreurs dans le trés haut débit, les conséquences des chemins multiples sur des signaux numériques rapides etc... C'est parce qu'il y aura des difficultés et de l'inconnu que çà devient intéressant.

Le projet parisien:

Construction d'un réseau THD à 1.2 Ghz en full duplex à fréquences décalées. Le full duplex est l'idéal pour faire disparaître les txdelays qui ont une durée trop longue, ralentissent le trafic et font perdre tout l'intérêt du haut débit. Le VCO du TX doit être maintenu à sa vitesse de croisière pour le haut débit, son temps de réponse est trop long, si on le commute en tx/rx son inertie ira à l'encontre des vitesses souhaitées. D'autre part, les montées et descentes en puissance des PA sont d'une durée non négligeable pour des paquets haut débit. Avec des émissions sans commutation, le problème ne se pose plus. Le débit mini sera de 38 Kb/s et le maxi de 500 Kb/s. Comme convenu lors de la répartition des fréquences 1.2 Ghz, 4 canaux seront utilisés à 1.240 Ghz et 4 canaux à 1.299 Ghz, le shift pour le duplex sera de 59 Mhz. L'espacement des canaux sera de 250 Khz. La bande passante des canaux sera de 250 Khz, permettant un débit max à 38 Kb/s en FM Manchester, 150 Kb/s en CPFSK type G3RUH, 250 Kb/s en BPSK et 500 Kb/s en QPSK.

Modems utilisés:

Dans un premier temps nous utiliserons un modem manchester à 38Kb/s qui est de réalisation facile avec des composants de fond de tiroir. Parallèlement des expériences de nouvelles modulations (QPSK) seront tentées, le modem QPSK sera logiciel et intégré dans un dsp. Nous sommes en contact avec des fournisseurs, notamment Infracom qui dispose déjà de modems haut débit pouvant monter à 1.2Mb/s. Ces fournisseurs intéresseront certainement les OMs packetteurs mais pas bidouilleurs.

Cartes d'interfaces et logiciels pour PC:

Les cartes SCC et USCC actuelles avec les softs conviennent pour du 38 Kb. Au delà on préférera utiliser les interfaces et softs des slovènes ou des allemands qui gèrent le trafic jusqu'à 1.2 Mb/s. Je pense que dans un premier temps le débit plus grand que

38K ne concerne que les links internodes. A 38 Kb/s l'OM utilisateur peut utiliser ses cartes SCC, USCC, RMNC, etc., il pourra utiliser son vieux PC 386.

Protocoles utilisés:

N'entrons pas dans la guerre qui oppose les flexnistes aux fpacistes, la couche matérielle sera transparente au réseau et à son protocole. On pourra donc faire de l'AX25 sous fpac ou flexnet, du TCP/IP, du pacsat ou de nouveaux protocoles à inventer mieux adaptés au temps réel Son et Vidéo.

Antennes 1.2Ghz:

Le choix est porté sur des antennes hélices, polarisation circulaire droit. L'avantage de l'antenne hélice est son "immunité" aux distorsions de phase. Elle est peu sensible aux rotations de polarisations qui sont très fréquentes en propagation urbaine.
Elle est de réalisation facile et peu coûteuse. Son défaut est une bande passante trop large, dépassant les 50Mhz, il faudra soigner la sélectivité des préamplis des RX. Un autre petit défaut, son faible gain, qui n'est pas gênant dans le trafic de proximité inférieur à 10Km. Nous accueillons volontiers tous conseils de spécialistes antennes qui pourraient nous proposer des plans d'antennes mieux adaptées aux contraintes de phase et de bande passante.

Les antennes et le full duplex: Les 2 possibilités sont retenues:
1) Une seule antenne équipée d'un duplexeur avec écart TX/RX de 59Mhz. Là nous cherchons un om qui sait fabriquer des duplexeurs, avis aux amateurs.
2) 2 antennes séparées (l'idéal) une pour le TX, l'autre pour le RX.

Emplacement des TX/RX:

Pour les grandes longueurs entre l'antenne et la station il sera conseillé de placer les TX/RX sur le mât, les pertes de coaxial sont trop grandes en 1.2Ghz. Nous envisageons des interfaces pour que dans un câble coaxial d'antenne (câble ordinaire 75ohms par exemple) nous fassions transiter l'alim, les données packet dans les 2 sens et les données de contrôle (changement de fréquence, ptt, etc.). L'autre solution est de monter un câble multibrins.

Je suis en contact avec Jean Matthieu F5RCT, il commence l'étude de réalisation d'un kit tx/rx spécial THD, avec peu de réglages, avec des FOS (filtres à onde de surface) et des circuits intégrés modernes qui regroupent les fonctions de mélangeurs, fi et démodulateurs FM. Ses kits seront prévus aussi pour descendre à 9600b/s. Si d'autres OMs s'intéressent à la conception de tx/rx spécial haut débit, je cite les caractéristiques requises:

TX:

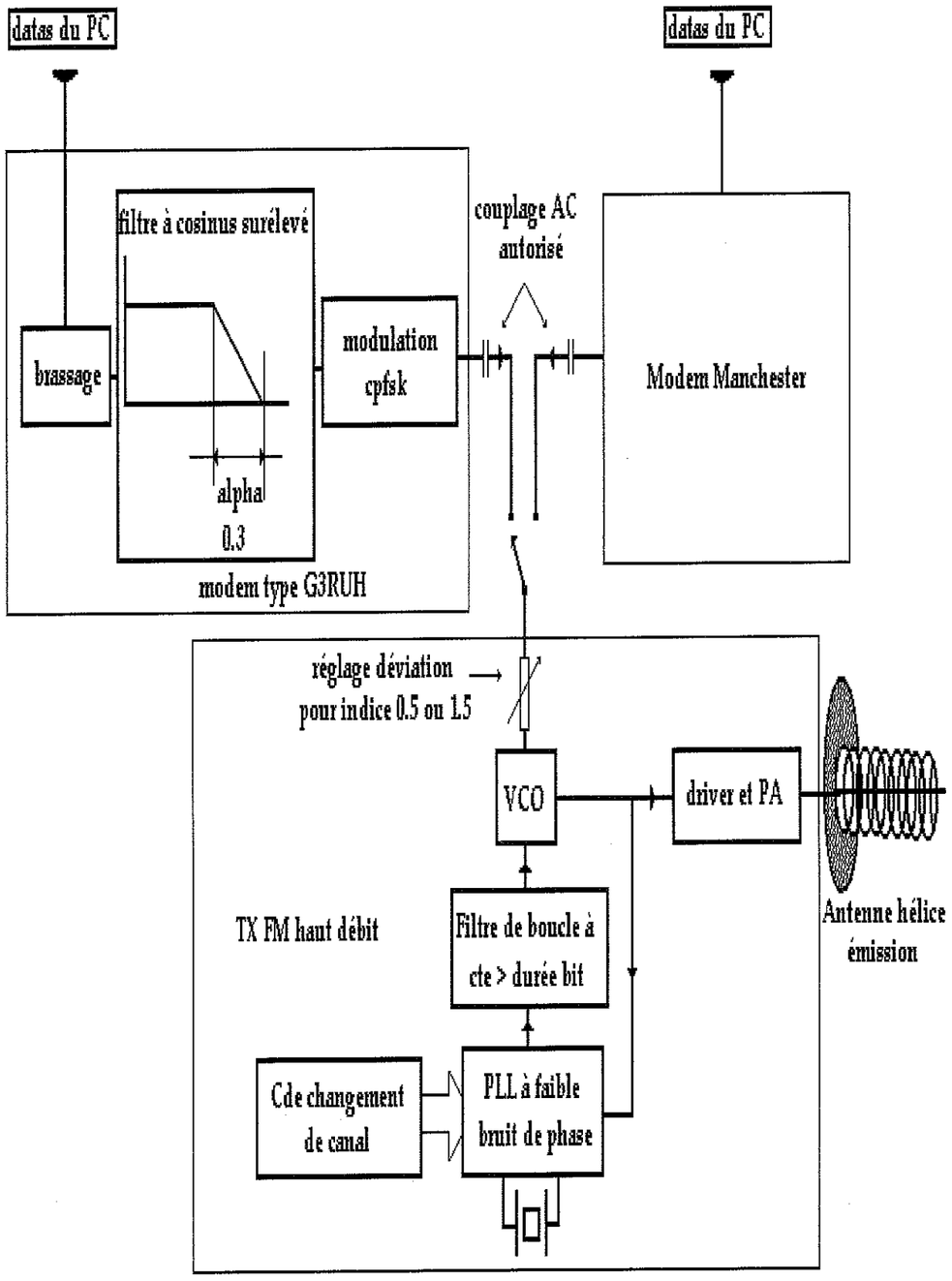
-Synthé avec canaux espacés d'au moins 6 x vitesse en bauds pour du bi-phase

manchester (canaux de 250Khz pour du 38.4Kb)
Synthé avec canaux espacés de 2 x vitesse en bauds pour du cpfsk G3RUH avec filtre en cosinus surélevé, coeff alpha de 0.3. (canaux de 80Khz pour du 38.4K), cpfsk = continue phase fsk.

-Indice de modulation FM de 1.5 pour du manchester.
-Indice de modulation FM de 0.5 pour du cpfsk type G3RUH.
(indice de modulation = déviation fm/vitesse max bauds)
Concrètement pour du 38K Manchester, prévoir une déviation de +/- 57 kHz. car 57K/38K donne bien 1.5.

-Constante de temps du filtre de boucle du pll adapté au débit binaire.
-Bruit de phase du pll le plus petit possible.
-Le bilan de liaison TX/RX doit fournir au moins 15db au dessus du bruit pour un taux d'erreurs de 10e-6. Prévoir en conséquence la puissance du PA, le gain des 2 antennes des TX et RX, la sensibilité du RX, la distance entre tx et RX et la fréquence choisie (UHF, THF, SHF).

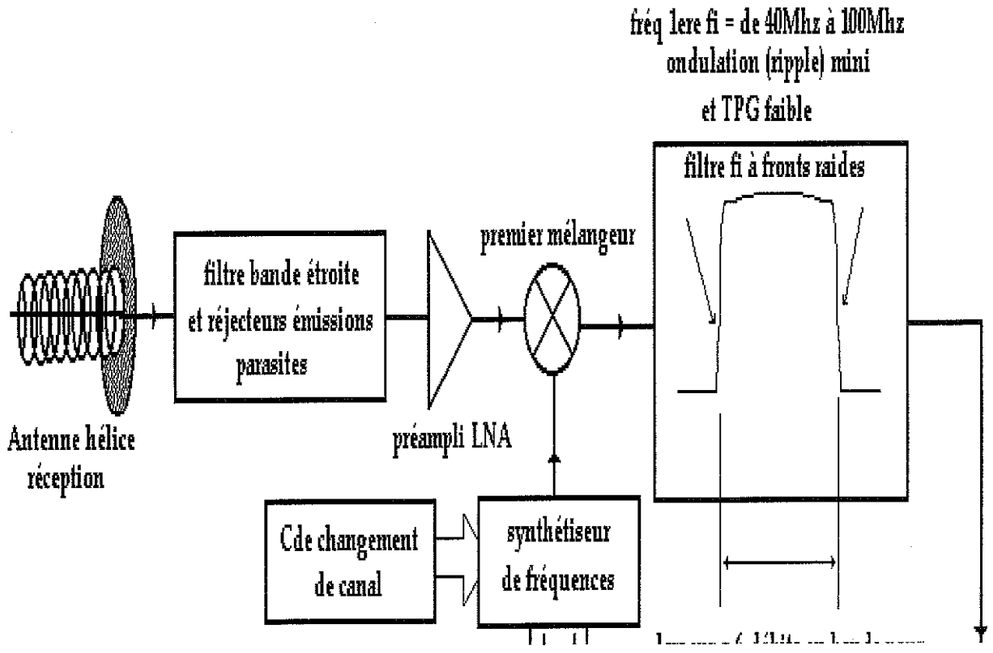
CHAINE D'EMISSION TRAFIC HAUT DEBIT

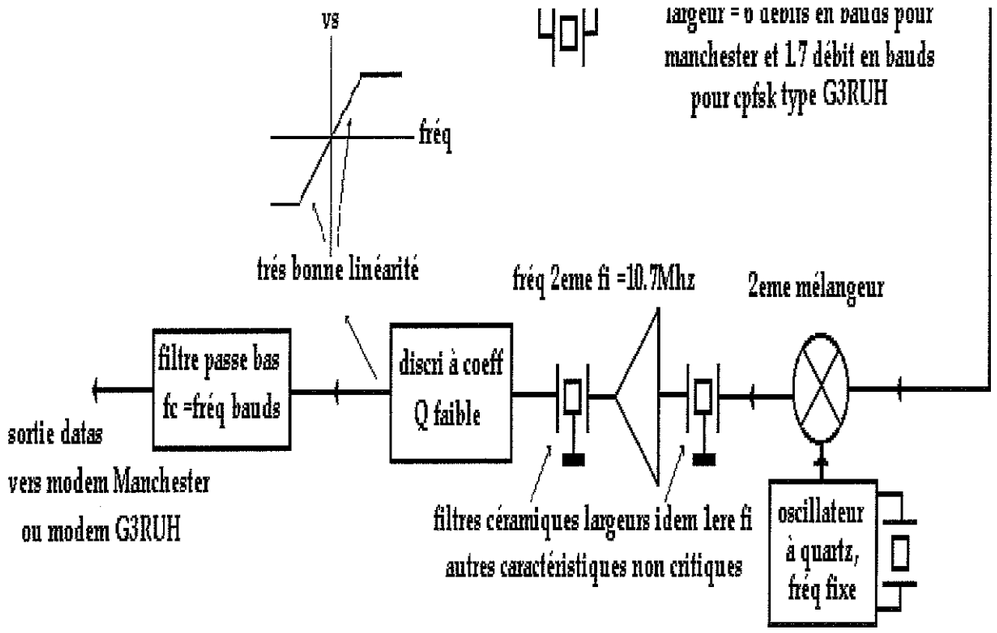


RX:

-RX à minimum double changement de fréquence.
Filtres à bande passante réduite en entrée antenne(20Mhz pour du 23 cm), plus circuits réjecteurs contre les émissions pro parasites. (pour du 23 cm = portables GSM à 950Mhz, radars, etc...)
-Préampli LNA (low noise amplifier) à faible bruit, comme son nom l'indique.
-1er filtre fi autour de 70Mhz (après 1er mélangeur) à flancs raides, très faible distorsion de phase, plateau bande passante le plus plat possible, minimum d'ondulations (ripples), TPG faible, bande passante large égale à 6 vitesse baud pour du manchester et 1.7 vitesse baud pour du cpfsk G3RUH et satellites 38.4Kb.
Ce 1er filtre est l'élément essentiel d'un RX packet haut débit.
-2ème filtre fi (après 2eme mélangeur) à 10.7Mhz, moins critique que le 1er fi, bande passante idem 1er fi.
-Discri FM à coefficient Q faible (courbe du S plate) de manière à conserver une bonne linéarité pour toutes les fréquences traversant le discri.
-Filtre de sortie discri avec bande passante égale à la vitesse bauds. (38Khz pour du 38Kbauds).

CHAINE RECEPTION TRAFIC HAUT DEBIT




vs
fréq
très bonne linéarité
largeur = 9 devis en bauds pour
manchester et 1.7 débit en bauds
pour cpfsk type G3RUH
fréq 2eme fi =10.7Mhz
2eme mélangeur
filtre passe bas
fc =fréq bauds
discri à coeff
Q faible
sortie datas
vers modem Manchester
ou modem G3RUH
filtres céramiques largeurs idem 1ere fi
autres caractéristiques non critiques
oscillateur
à quartz,
fréq fixe

Voilà pour les spécifications nécessaires aux tx/rx THD, sans parler des temps de commutation tx/rx les plus courts possibles. (Le full duplex est l'idéal mais coûte cher). Pour les prix d'un ensemble kit tx et RX séparés, il faut prévoir de ne pas dépasser les 1000F pour les 2, le prix augmentant d'une manière exponentielle avec la montée en puissance des PA en 1.2Ghz et SHF.

Si beaucoup d'OMs sont intéressés par le THD, j'organiserai une conférence sur le sujet pour le début Janvier 98 avec démo à l'appui.
Pour toutes demandes d'infos supplémentaires contactez moi par courrier à mon club F5KTR, par packet à F6RAC et par internet à mon

E-mail : f1biu@wanadoo.fr

A remercier Bernard F6BVP, qui est à l'origine du projet THD pour Paris.

Mes 73 de Victor F1BIU.

page composée par Bernard Pidoux, F6BVP

23 cm Data-transceiver *(V2.5)*

last update january 16, 1999

Introduction

This project describes a 23cm Wide-Band FM data-transceiver intended for high-speed packet-radio. It is currently used at the 38k4 LAP's at PI1RNI in Utrecht (JO22NB), PI1HVS in Hilversum (JO22OF) and at the BBS PI8GCB in Bussum (JO22OG). Also at other nodes in PA this transceiver might be used in the near future. Antennas used are the panel antenne published in CQ-TV 182 by G8GML, which is working very well at PI1RNI. At PI8GCB we use an "8" (quad hybrid), which does however not perform very well. At PI1HVS a slot antenna is used. All are horizontally polarized.

Abstract

This data-transceiver is build on one eurocard size doublesided pcb, one side etched and the other side left all copper. It is meant to be used split-frequency with an offset of 54 MHz. This fits nicely in the bandplan at PA where high-speed digital experiments are allowed at 1241.XXX and 1295.XXX. It is however possible to work simplex but the PLL has to be reprogrammed every time one switches over from TX to RX or visa versa. Also it is possible to change the 54 MHz IF to another frequency between ~40 and ~70 MHz.

This article is referring to version 2.3 of the design and the pcb. The software in the Atmel is version 3.0.

Description

In 23schema.gif the schematic diagram is shown (Orcad Capture for Windows V7.00). The central part is a VCO around a BFR91A, oscillating directly on the TX-frequency. The IF is chosen to be 54 MHz, which happens to be equal to the shift between TX- and RX-frequency. So there is no need to reprogram the PLL when switching over from RX to TX and visa-versa. Currently the users transmit at 1241.XXX and receive on 1295.250, where the node transmits. So, for the user the PLL is programmed to i.e. 1241.250, and for the node the PLL locks on 1295.250. Of course, it is possible to reprogram the PLL every time when switching over, but this takes time...

The PLL is a SDA3302 which is used in TV-tuners. Interfacing to the Atmel 89C2051, a member of the large 8051-family, takes place via I2C. The loop-filter is build around the NPN-transistor and 2 C's and a R and is optimised for quick response with beta ~ 1.

The output of the VCO is buffered/amplified by a mar-8 and than fed to the PLL, the receiver-mixer and the transmit-path. Coupling to the RX and TX-parts is achieved by coupled striplines. Transmitting is very simple: the power is applied to a TX-buffer/amplifier consisting of a mar-8 giving 10 mW which then is amplified by the final amplifier to 2-3 Watts (Mitshubishi module). *In version 2.5 is the coupling to the PLL achieved using a microstrip coupler in stead of directly via a capacitor.*

The receiver-chain starts with a mar-8 used as preamp, having a gain of about 20 dB and a

noisefigure of around 3.5 dB. After passing through a striplinefilter the 23-signal is fed to the mixer consisting of a GaAs-fet (3SK183, CF300, 3SK97 or equivalent). The first IF is as already mentioned 54 MHz. This signal is first filtered and than amplified by a BF980 dual-gate mosfet. The IF processing further takes place in a MC3356. Using an external crystal-oscillator on 64.7 MHz the IF signal is mixed to 10.7 MHz, filtered in a Murata broadband filter (I use 280 kHz, but depending on the baudrate a narrower filter can be used). Finally the 10.7 MHz is demodulated using a quadrature detector. During transmitting the power of the receiver-chain is switched off, exept the 5V of the MC3356.

The microcontroller 89C2051 has 2k flash rom on-chip, and 128 bytes ram. 2 ports (P1 and P3) are partially available. Although in this set-up the VCO is never reprogrammed, the PTT is also fed into the atmel and case be used by the software to change frequencies. Two sets of jumpers are placed on the print, of which one set is currently used for frequency-selection and high/low or low/high receiving and transmitting.

Construction

In 23pcb.gif the printed circuit board is shown. The pcb is designed with Protel for Windows V1.5. *In 23parts.gif the componentlayout if shown (the componentvalues are not visible).*

After drilling the holes in the pcb (0.7-0.8mm), remove the copper around the holes at the other side. Do not free the holes which are marked red in 23pcb.gif, since these are used to contact through to earth. Drill a hole of the right size on the spot for the BFR91A (5 mm), 3SK183 (3.5 mm) and the BF980 (5 mm). Then solder the pcb all around in a fitting tin box (I use standard size 16x10x3 cm with 2 lids). Take care that you position the pcb with the module-end in a bend corner and not a corner which has to be soldered. Of course, all holes in the tin box have to be drilled before the pcb is fit into it. All signals can be fed through 1 nF feedtrough C's apart from the 23cm signal. This is best led to the outer via a piece of PFTE-coax to the N-connector at the housing.

Use a short piece of wire cut of a resistor and fold it to a "U"-form to make an earth contact for the mar's. Put the "U" in from the copperside, solder it to earth and cut the edges at the other side down to ~1 mm and bend them away from the mar. By the way, leave the mar's out as long as possible, they can be destroyed by oscillations during tuning up. I use a socket only for the Atmel. Cut pin 10 of the socket short, place it in posistion and solder this pin to earth. (Use a pointed soldering iron for this). The earth-pins of the other IC's can be cut short also and directly soldered to earth.

The VCO gets some special attention. First, bend a small piece of copperfoil in an "U"-shape and shorten the earth-side of the pcb with the earthspot next to the VCO. It might be nessecary to file the hole a bit sqare at that side. Then put the transistor in. All leads have to be as short as possible, and do not use a lot of solder. The 120 pF capacitor from BB405 to collector serves as oscillating coil: leave the lead on one side approx. 1 cm, and the other end short. Solder the short end to the varicap (C flat on the pcb), and bend the other lead in an "L" shape and solder the end halfway between the BFR and the mar. The 1p C's I use are SMD-type, but this might not be nessecary, just cut the leads of standard small C's short and solder them in at the etched side.

The cans of the coils 5049 and 5056 (Neosid) and the houses of the crystals have to be soldered to earth.

Take care that the capacitors used are good at these high frequencies. Do NOT use 10 or 100nF surplus from computer boards or so!. This holds true for all 10 or 22nF C's all over the pcb, and specially at the RX-chain. Decoupling is the key to succes here.

The regulators do not need to be screwed to the box. They do not get hot, at least not when you use this transceiver for packet. This holds also for the module: beside the tin box itself no extra cooling is needed but of course doesn't harm.

The "6cm coil" in the schematic diagram is a length of 0.3 mm coated wire (yes, indeed, 6 cm long) which is wound on a 1.5 mm drill. It is then streched to fit at the proper place.

Tuning

Of course a programed 89C2051 is needed before one can start tuning up the transceiver The assembler file is link30.asm (version 3.0). The assembler used is ASEM-51 V1.1, a family cross assembler running at the PC. The schematics and latest software release for the programmer can be found at http://sistudio.com .

First thing to do is to get the VCO oscillating at the proper frequency and get the PLL in lock. Only fit the mar-8 in the oscillator-buffer for this. It might take you some time to get the PLL in lock. When the control voltage keeps low (0V), it might help to put a resistor of 1 k between the base of the BC547 and +5V. (so between pen 18 and 14 of the SDA3302 at the printed side.) Try also shortening the "coil" formed by the 120pF capacitor. When nothing helps, remove all parts of the VCO, clean the pcb and start again.

First, tune the RX-chain. Start with the 64.7 MHz oscillator. When tuning the 5049 the oscillator abrubtly switches on and off. Check the output of the oscillator with an ordinary oscilloscoop at pen 3 of the MC3356. When removing the scoop, you might have to readjust a bit because of the influence of the scoop. Then set the 3 sky trimmers to minimum and feed 54 MHz (-30 dBm) into the antenna, or at the testpoint at the input of the IF-filter. Adjust the IF coils (5049) for maximum reading on the S-meter. Connect the scoop to the RxM output and adjust the IF quadrature detector (5056) for proper output. Better is to set the scoop to DC and measure at pin 13 of the MC3356 (direct output). Center the line or noise you see in the middle of the minimum and maximum detector output. When you don't see anything or strange "blobs" when tuning coils or touching parts in the RX-chain, something is oscillating. Try locate this first and solve this problem. Best is to use a (tunable) frequency generator at 54 MHz first and later at the Rx-frequency with output adjustable from -120 to 0 dBm.

When the IF works, adjust the stripline filter for maximum sensitivity at 23cm. The sky-trimmer the closest to the GaAs-fet is at minimum capacity, so this stripline is slightly too long. One can shorten it by 1 mm at the cold end, but this is not nessecary.

The transmitter should work immediatly. No tuning is required except for the deviation. Output of the module is ~2-3 Watts. At the input of the module ~10 mW is measured. The oscillator-buffer should do ~5 mW power.

Results so far

A direct TCP/IP connection with 38k4 using JNOS gives a throughput of ~2.5 kByte/sec. On 76k8 this is around 3.5 kByte/sec.When working via PI1RNI, with two stations at 38k4, the throughput is 1.2-1.3 kByte/sec. The TxDelay is 10-15 ms.

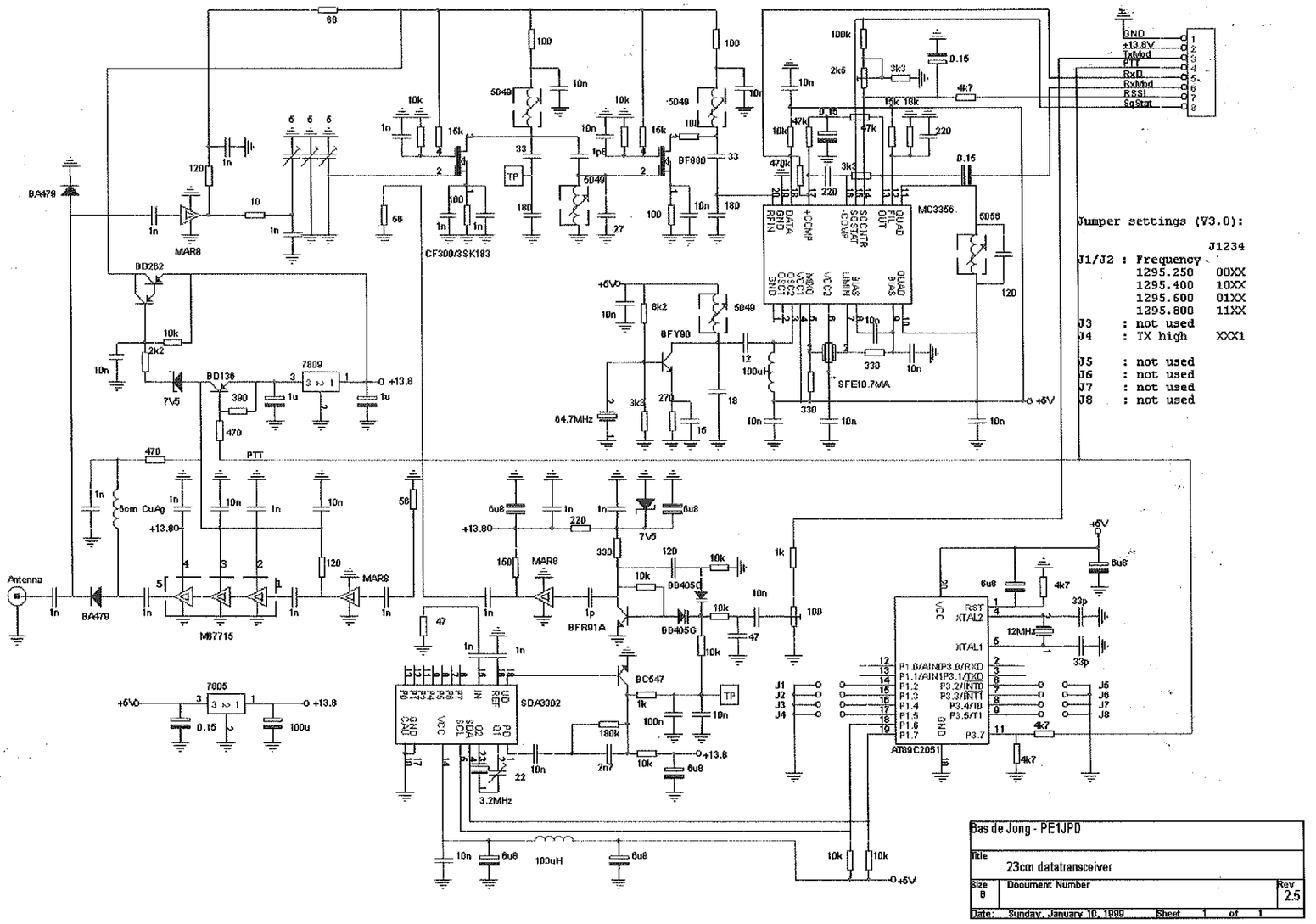
The setup used is a standard PC (386, 486) with a SCC-card designed by PE1PET. Since no DMA is used on this card, the speed is limited to about 100 kbps I guess. The modem used up till now is a simple manchester modem designed by S53MV. In stead of the original design, a sampler is added which decreases the Bit Error Rate significantly (manmodnl.htm only in Dutch, sorry). Also, a 4th order lowpass is used between the transceiver and the modem (rxfilter.htm). This improves the signal-to-noise ratio by approx. 10 dB.

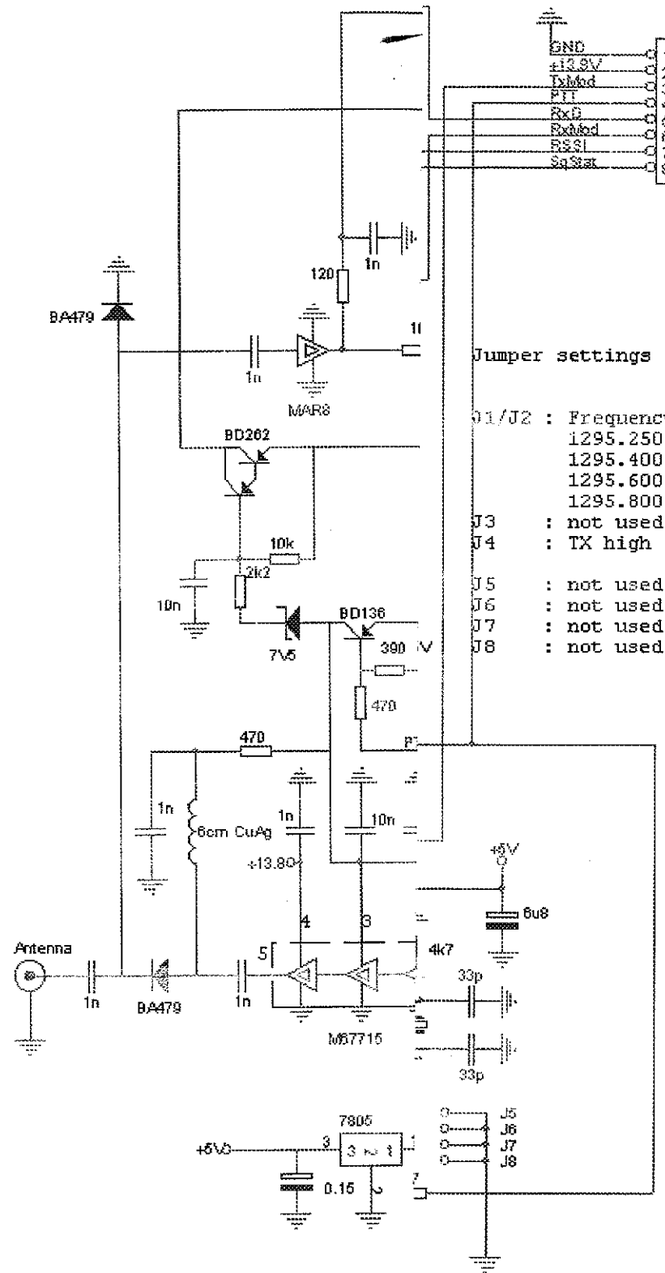
Up till now, about 20 to 30 transceivers are working correctly.

Acknowledgements

Special thanks to Hans, PE1CKK who helped a lot with the design, the prototypes and the tuning.

Back to homepage

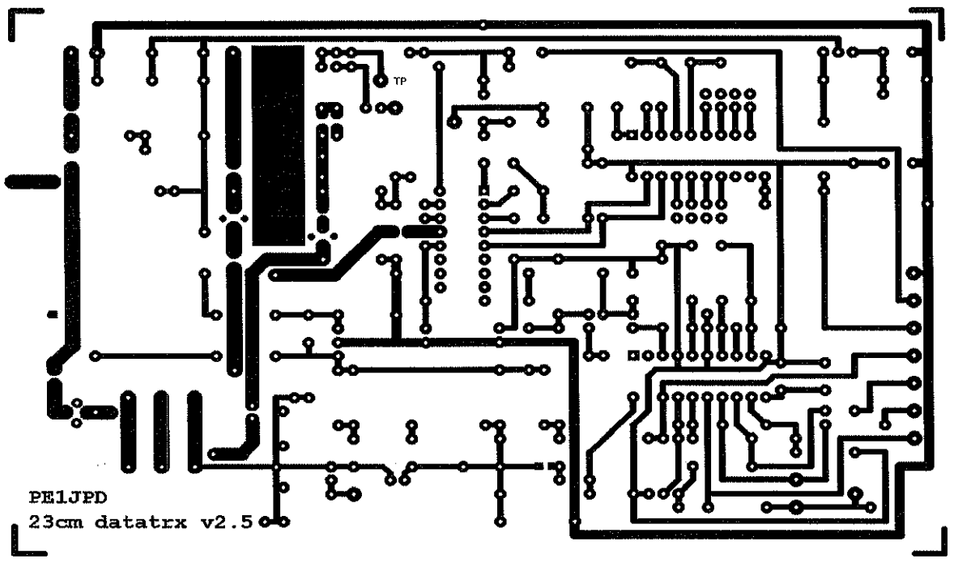

Jumper settings (V3.0):
J1234
J1/J2 : Frequency
1295.250 00XX
1295.400 10XX
1295.600 01XX
1295.800 11XX
J3 : not used
J4 : TX high XXX1
J5 : not used
J6 : not used
J7 : not used
J8 : not used
GND
+13.8V
TxMod
PTT
RxD
RxMod
RSSI
SqStat
MC3356
SFE10.7MA
CF300/3SK183
BF980
BFY90
BB405G
BFR91A
BC547
BA479
BD262
BD138
MAR8
M67715
SDA3302
AT89C2051
12MHz
9.2MHz
64.7MHz
Antenna
PTT
Bas de Jong - PE1JPD
23cm datatransceiver
Document Number
Rev 2.5
Sunday, January 10, 1999
Sheet 1 of 1

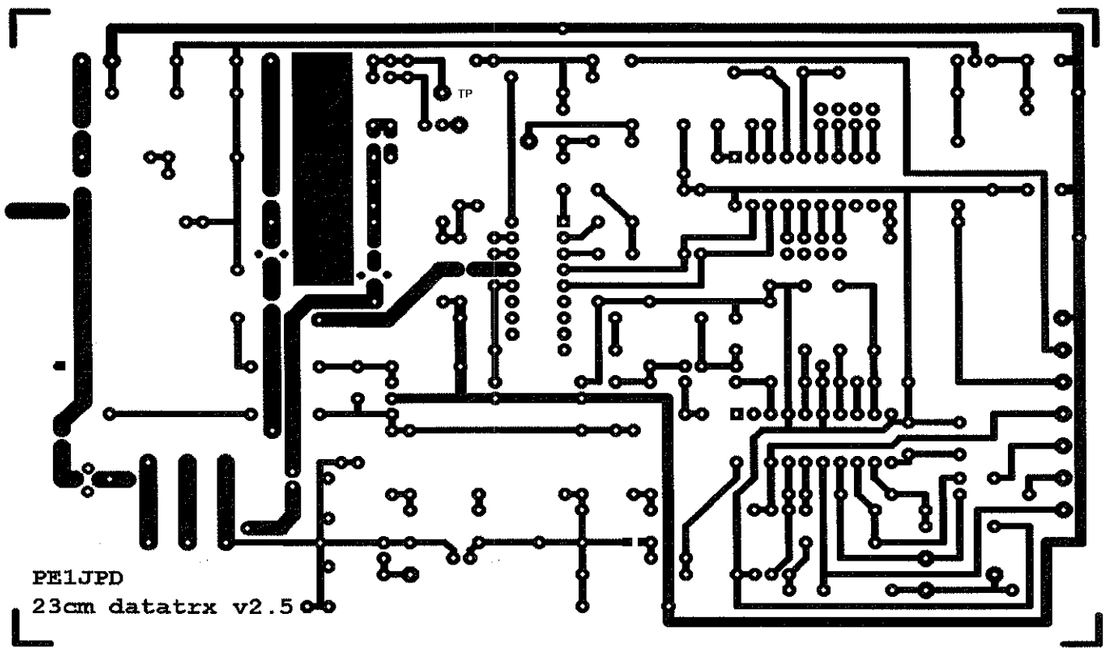


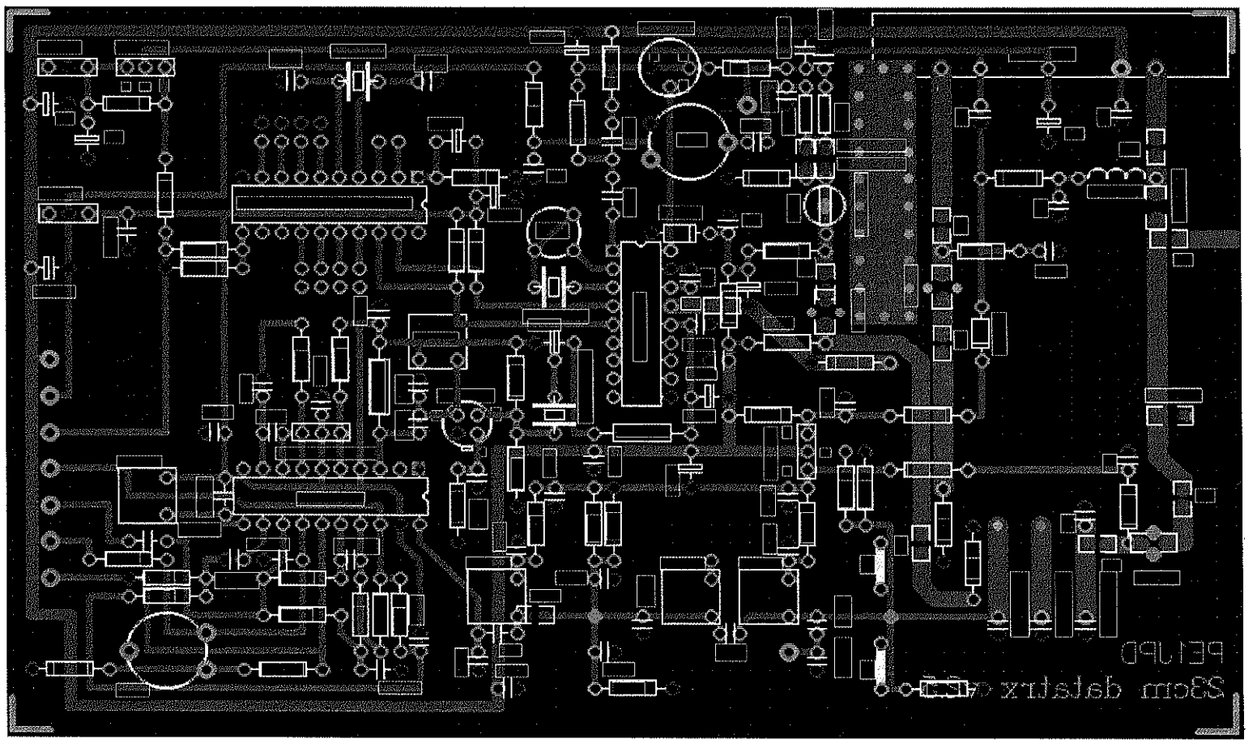
Jumper settings (V3.0):

		J1234
J1/J2	Frequency	
	1295.250	00XX
	1295.400	10XX
	1295.600	01XX
	1295.800	11XX
J3	: not used	
J4	: TX high	XXX1
J5	: not used	
J6	: not used	
J7	: not used	
J8	: not used	

E1JPD		
atatransceiver		
t Number		Rev 2.5
January 10, 1999	Sheet 1 of 1	


TP
PE1JPD
23cm datatrx v2.5


TP
PE1JPD
23cm datatrx v2.5


PETUPD
Z80m dalatrx

```
;*************************************************************************
; version date      change                                              by
; ------ ---------  --------------------------------------------------- ----
; 1.0    22/12/97   first version.                                      JPD
;                   tx:    1241.250 MHz
;                   rx:    1295.250 MHz
;                   shift: 54 MHz
; 2.0    14/08/98   no boot when changing jumpers, and implemented
;                   4 freqs incl. the reversed                         JPD
;*************************************************************************

;*************************************************************************
; The VCO-frequency Fvco is determined as follows:
;
;                3.2 MHz
; Fvco = Fr*N = ------- * 8 * n
;                 512
;
; where Fr the reference frequency which is equal to the crystal frequency
; devided by 512, and N the total division. The SDA3302 devides by 8 fixed
; also, so n equals the values to program the device with.
;
; When 3.2 MHz crystal frequency is used the reference frequency gets
; Fr = 3200 kHz / 512 = 6.25 kHz and the final raster equals
; 8 * Fr = 50 kHz.
;*************************************************************************

                DSEG
                ORG     0008H
ByteCnt:        DS      1
Xmit:           DS      8
PrevP1:         DS      1

                BSEG
NoAck:          DBIT    1

SCLPin          EQU     P1.6                ; I2C serial clock line.
SDAPin          EQU     P1.7                ; I2C serial data line.

                CSEG
                ORG     0000H
                LJMP    Start

                ORG     0100H

Start:          MOV     SP,#40H

                MOV     PrevP1,#0C0H
Loop:           MOV     A,P1                ; get P1 in A
                ANL     A,#03CH             ; leave only relevant bits
                MOV     R1,A                ; save A in R1
                XRL     A,PrevP1            ; any difference with previous P
                JNZ     M1                  ; branch if so
                MOV     PrevP1,R1           ; else, update previous
                SJMP    Loop

M1:             MOV     PrevP1,R1           ; update previous
                MOV     A,R1                ; 4 bits of P1 in A
                MOV     DPTR,#FreqTbl1      ; get start of tbl1
                JB      ACC.5,M2            ; or if reversed,
                MOV     DPTR,#FreqTbl2      ; get start of tbl2

M2:             CPL     A                   ; invert logic
                ANL     A,#0CH              ; now leave only freqbits
                RR      A                   ; shift to match tbl
                MOV     R1,A                ; save A in R1
                MOVC    A,@A+DPTR           ; get freq
                INC     DPTR                ; point to 2nd byte
```

```
                MOV     Xmit,A                  ; store 1st byte
                MOV     A,R1                    ; restore A
                MOVC    A,@A+DPTR               ; get 2nd byte
                MOV     Xmit+1,A                ; store it too
                ACALL   SetFreq                 ; and program ppll
                SJMP    Loop

FreqTbl1:       DB      060H,0F9H               ; 1241.250
                DB      060H,0FCH               ; 1241.400
                DB      061H,000H               ; 1241.600
                DB      061H,004H               ; 1241.800

FreqTbl2:       DB      065H,031H               ; 1295.250
                DB      065H,034H               ; 1295.400
                DB      065H,038H               ; 1295.600
                DB      065H,03CH               ; 1295.800


SetFreq:        MOV     A,#0C0H
                MOV     R0,#Xmit
                MOV     ByteCnt,#2
                ACALL   SendData
                RET


;*************************************************************************
;                               I2C-stuff
;*************************************************************************

BitDly:         NOP
                NOP                     ;NOPs to delay 5 microseconds (minus 4
                RET                     ;machine cycles for CALL and RET).

SCLHigh:        SETB    SCLPin          ;Set SCL from our end.
                JNB     SCLPin,$        ;Wait for pin to actually go high.
                RET

SendStart:      CLR     SDAPin          ;Begin I2C Start.
                ACALL   BitDly
                CLR     SCLPin
                ACALL   BitDly          ;Complete I2C Start.
                CLR     NoAck
                RET

SendStop:       CLR     SDAPin          ;Get SDA ready for stop.
                ACALL   SCLHigh         ;Set clock for stop.
                ACALL   BitDly
                SETB    SDAPin          ;Send I2C stop.
                ACALL   BitDly
                RET                     ;Bus should now be released.


; SendByte - sends one byte of data to an I2C slave device.
; Input         A     = data byte to be sent.
; Affected:     A, R2

SendByte:       MOV     R2,#8           ;Set bit count.
SBLoop:         RLC     A               ;Send one data bit.
                MOV     SDAPin,C        ;Put data bit on pin.
                ACALL   SCLHigh         ;Send clock.
                ACALL   BitDly
                CLR     SCLPin
                ACALL   BitDly
                DJNZ    R2,SBloop       ;Repeat until all bits sent.
                SETB    SDAPin          ;Release data line for acknowledge.
                ACALL   SCLHigh         ;Send clock for acknowledge.
                ACALL   BitDly
                JNB     SDAPin,SBEX     ;Check for valid acknowledge bit.
                SETB    NoAck           ;Set status for no acknowledge.
```

```
SBEX:           CLR     SCLPin          ;Finish acknowledge bit.
                ACALL   BitDly
                RET


; RcvByte - receives one byte of data from an I2C slave device.
; Output:       A     = data byte received
; Affected:     A, R2

RcvByte:        MOV     R2,#8           ;Set bit count.
RBLoop:         ACALL   SCLHigh         ;Read one data bit.
                ACALL   BitDly
                MOV     C,SDAPin        ;Get data bit from pin.
                RLC     A               ;Rotate bit into result byte.
                CLR     SCLPin
                ACALL   BitDly
                DJNZ    R2,RBLoop       ;Repeat until all bits received.
                PUSH    ACC             ;Save accumulator
                MOV     A,ByteCnt
                CJNE    A,#1,RBAck      ;Check for last byte of frame.
                SETB    SDAPin          ;Send no acknowledge on last byte.
                SJMP    RBAClk
RBAck:          CLR     SDAPin          ;Send acknowledge bit.
RBAClk:         ACALL   SCLHigh         ;Send acknowledge clock.
                POP     ACC             ;Restore accumulator
                ACALL   BitDly
                CLR     SCLPin
                SETB    SDAPin          ;Clear acknowledge bit.
                ACALL   BitDly
                RET


; SendData - sends one or more bytes of data to an I2C slave device.
; Input:        ByteCnt = count of bytes to be sent.
;               A       = slave address.
;               @R0     = data to be sent (the first data byte will be the
;                         subaddress, if the I2C device expects one).
; Affected:     A, R0, R2

SendData:       ACALL   SendStart
                ACALL   SendByte        ;Send slave-address
                JB      NoAck,SDEX      ;Check for slave not responding
SDLoop:         MOV     A,@R0           ;Get data byte from buffer.
                ACALL   SendByte        ;Send next data byte.
                INC     R0              ;Advance buffer pointer.
                JB      NoAck,SDEX      ;Check for slave not responding.
                DJNZ    ByteCnt,SDLoop  ;All bytes sent?
SDEX:           ACALL   SendStop        ;Done, send an I2C stop.
                RET

; RcvData - receives one or more bytes of data from an I2C slave device.
; Input:        ByteCnt = count of bytes to be sent.
;               A       = slave address.
; Output:       @R0     = data received.
; Affected:     A, R0, R2
;
; Note: to receive with a subaddress, use SendData to set the subaddress
; first (no provision for repeated Start).

RcvData:        INC     A               ;Set for READ of slave.
                ACALL   SendStart       ;Acquire bus and send slave address.
                ACALL   SendByte
                JB      NoAck,RDEX      ;Check for slave not responding
RDLoop:         ACALL   RcvByte         ;Receive next data byte.
                MOV     @R0,A           ;Save data byte in buffer.
                INC     R0              ;Advance buffer pointer.
                DJNZ    ByteCnt,RDLoop  ;Repeat untill all bytes received.
RDEX:           ACALL   SendStop        ;Done, send an I2C stop.
                RET
```

END