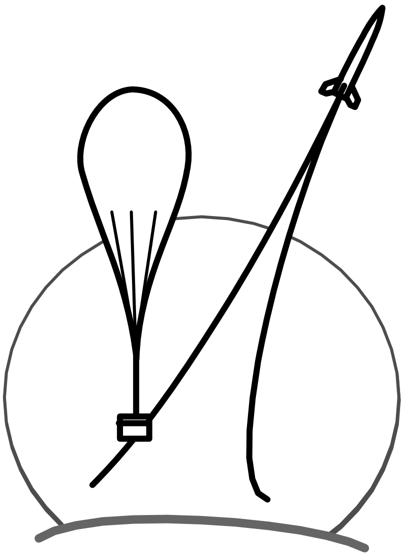


Kiwi "Millenium"
Telemetry System
Comprehensive Datasheet

Version 1.0 - April 2008

Updates

Version	Date	Author	Purpose
2.0	2003	Nicolas Verdier	Kiwi Millenium – Guide de l'utilisateur
0.1	Mar 27, 2007	Staszek Ostoja Starzewski	First translation work
0.5	May 27, 2007	Christophe Scicluna	Translation and add-ons for advanced user
0.6	June 06, 2007	Gabe Arnold	Editing
0.7	June 10, 2007	Christophe Scicluna	Clarifications about external modulation
0.8	Jan 13, 2008	Nicolas Verdier	Review for CNES
0.9	Feb 21, 2008	Frederic Bouchar, Leo Come, Nicolas Courounneau	Updates on data acquisition, demodulators
1.0	April 26, 2008	Christophe Scicluna	Final completion

FOREWORD AND HISTORY

KIWI Telemetry System has been designed by CNES (French Space Agency), Planète Sciences (Space Division) and Tenum in order to allow young amateurs to perform onboard radio data transmission for both rocket and weather balloons projects.

Developing its own transmitter is a challenge that few clubs or schools manage to complete, due to lack of knowledge, of time and test equipment. The purpose of the KIWI telemetry system is to provide a reliable, effective and easy to use transmitter to achieve wireless data transmission over a frequency allocated to CNES.

This document is an excerpt of the original Telemetry System User's Guide. It is intended to the foreign university clubs who take part in Planète Sciences' activities in France.

The original document -in French- provides basic but useful information for teachers and college/high school students. This translation mainly focuses on the device as a third party component, featuring performances, basic usage and advanced usage. The reader is assumed to have a basic knowledge of electronics, wiring, and system integration.

Several versions of Kiwi have been produced.
The original version was developed for groups of young students working on weather balloon projects. By design, only analog data could be transmitted using an internal modulation.
The second version, called Kiwi Millenium, has been designed for a wider range of users, who wish to transmit data of another type. Analog or digital data can be transmitted with the internal or an external modulator. But digital data transmission requires an external modulator, to be designed by the club.

The name *Kiwi* is not related to the fruit but to the bird.
The tradition by Planète Sciences is that transmitters are given names of exotic birds (Colibri, Ibis) and rocket engines are given names of antelopes, ie. wild mammals with horns (Dick-Dick, Eland, Koudu, Cariacou, Wapiti, Isard, Chamois, Barasinga…)

GENERAL DESCRIPTION

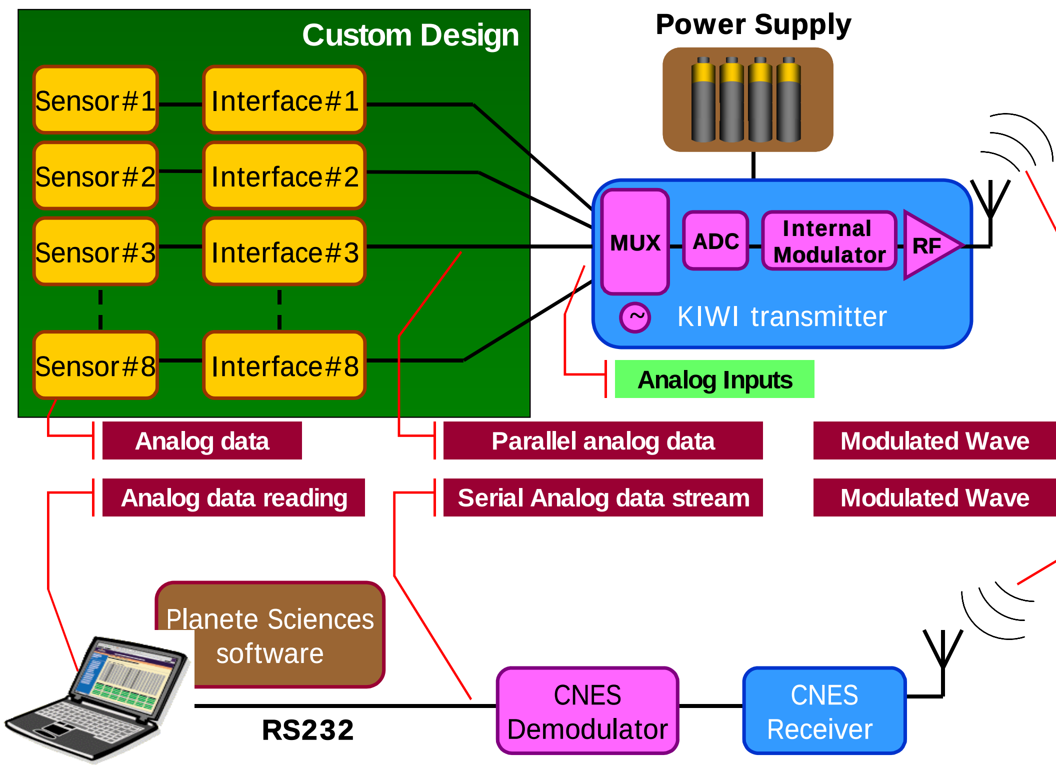
Kiwi is a standalone device, with input connectors for power supply, analog data, programming interface for experimented users, and output for antenna. It allows radio transmission over a long distance (more than 200km). It can transmit 8 analog channels with its internal modulator. **Digital data can be transmitted as well,** but this requires either a user-designed and manufactured external modulator, or to re-program the internal microcontroller [1]. This external modulation allows transmitting, digital binary data (GPS data or microcontroller debugging data as for example). External modulation is described in the chapters of this document.

Transmission frequency can be selected among the two frequencies allocated to CNES: 137.950 MHz and 138.500 MHz.

Kiwi is based on a PIC: it makes it small, light, low consumption and easy to program.

Data are sent over a whip antenna. After reception by a tuned receiver, a demodulator is required to decode the data in order to be input in a RS232 port of a computer for recording.

A global synopsis of a standard analog signals telemetry system is presented below:



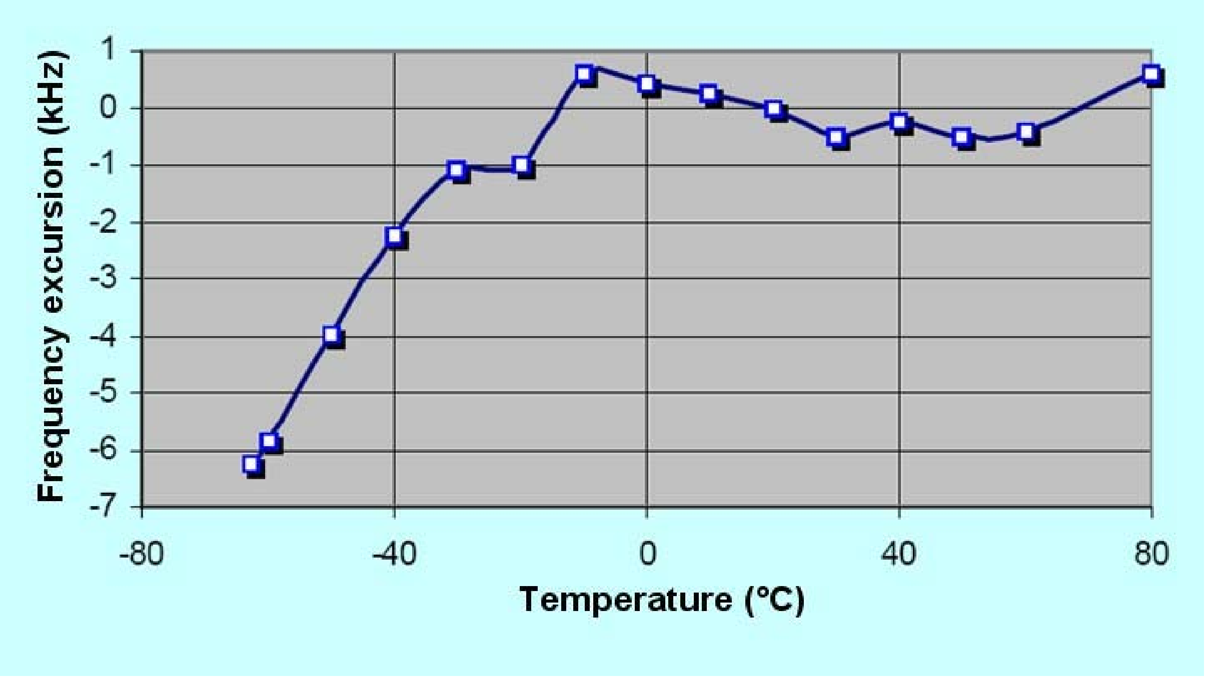
[1]Original Software code is provided in the last section of this document

DATASHEET

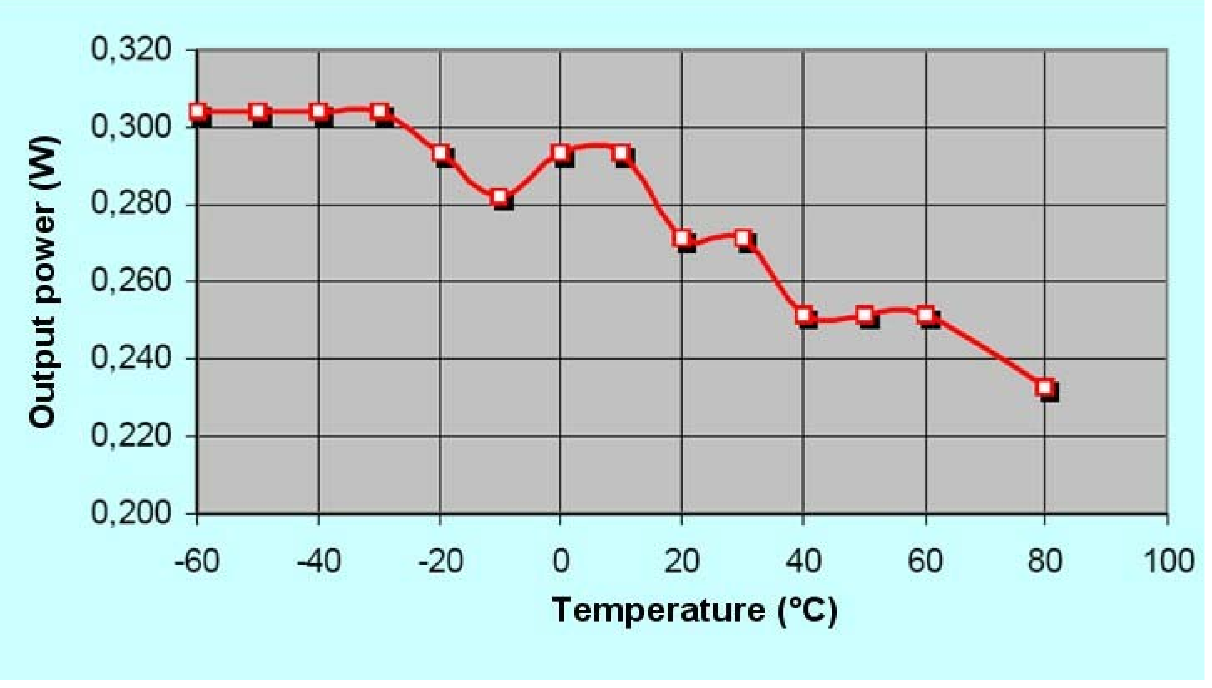
Parameters	Min	Typ	Max	Unit
Transmission frequency 1 Transmission frequency 2	-	137.95 138.50	-	MHz MHz
Output power	173 22.4	251 24	329 25.2	mW dBm
Analog inputs				
Number of channels Input voltage Resolution Data rate	- 0 - 0.5	- - 20 -	8 5 5 2	- V mV frame/sec
Modulation input				
Amplitude Bandwidth	0.1 0.5	- -	5 50	Vpp[2] kHz
Number of input/output (on PIC)[3]	-	-	5	-
General				
Supply voltage	7	9	12	V
Current consumption	170	190	210	mA
Operating temperature	-60	20	85	°C
Weight	50			g
Length (excluding BNC connector)	86			mm
Width	57			mm
Thickness	17			mm
Recommended Length of antenna @ frequency	137.95 MHz 138.50 MHz	~54		cm

[2] Peak to Peak

[3] Inputs and outputs are available for reprogramming PIC with custom firmware. It is recommended for advanced users only. For digital transmission, use external modulation mode as explained further.

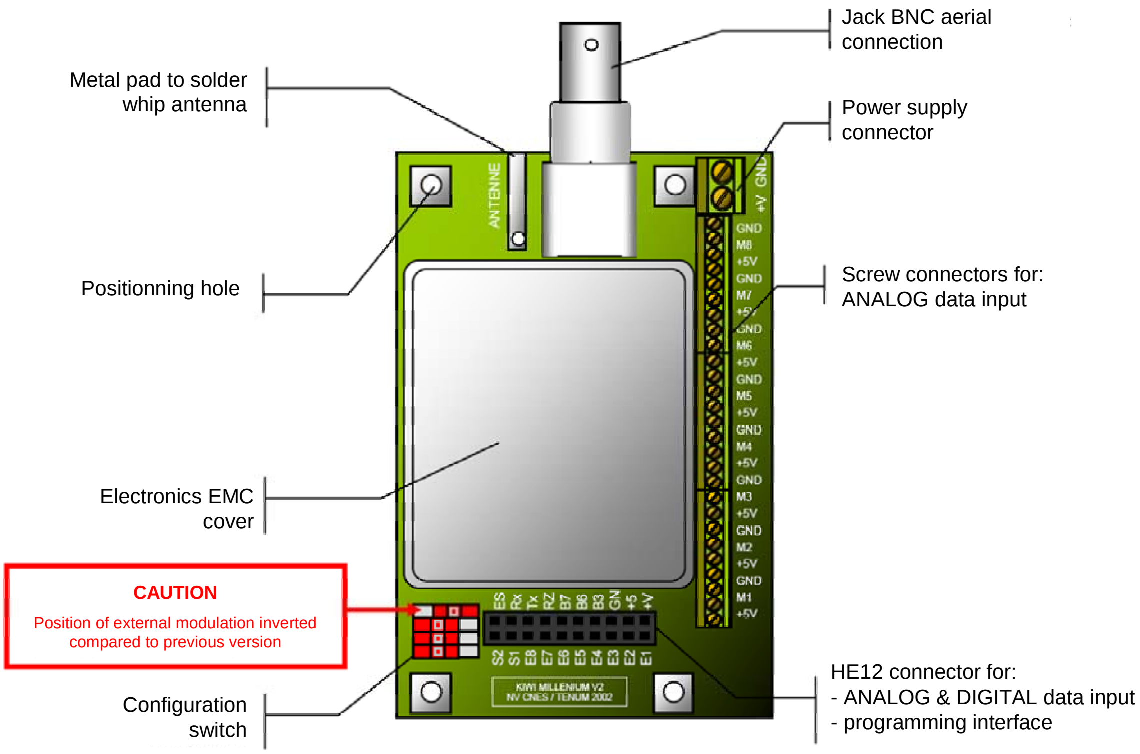


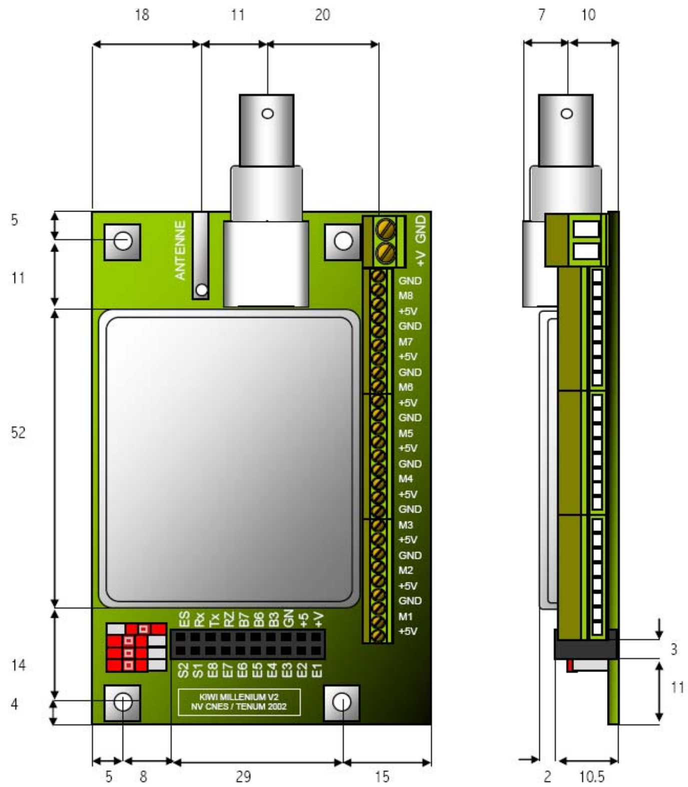
Frequency excursion vs Temperature

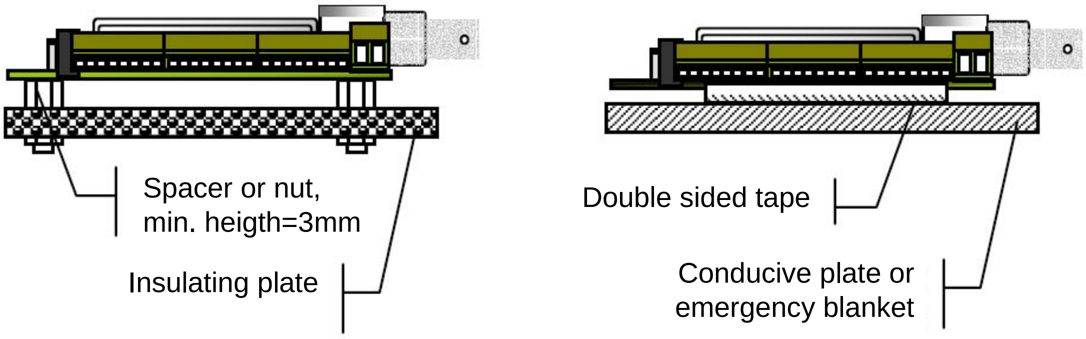


Output Power vs Temperature

MECHANICAL DESCRIPTION







Advices for mechanical integration

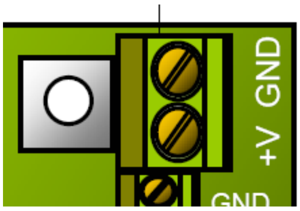
POWER SUPPLY

The Kiwi power supply is DC ~7V to ~12V, delivered by external batteries, rechargeable or not. An internal 5V regulator will supply the internal circuits of the whole transmitter as well as the analog channels.
It is however strongly recommended to use 9V batteries as higher voltage would generate excess power dissipation and increase the temperature of the device.

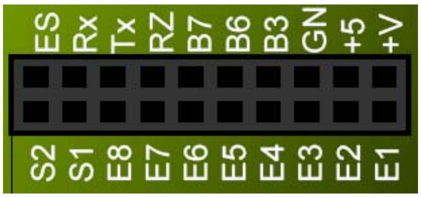
In case of rocket project,
a separate power supply is required for the transmitter

In case of weather balloon project, as the flight may last several hours, long lasting batteries will be used.

An alternative connector, for +V power supply input, is available on the programming interface connector.

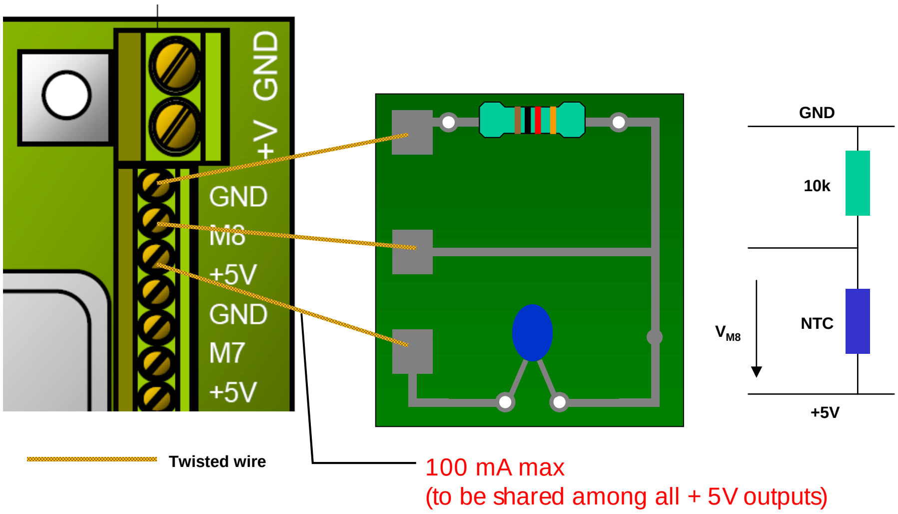


[9V to 14V] main power supply connector



[9V to 14V] alternative connector

The Kiwi is designed to deliver 5V DC to each analog channel, so that the user only needs to design a passive electronics circuit for each sensor. This is especially useful for young scientists willing to measure temperature, pressure, light or another physical parameter from a simple resistive circuit. In that case, a total of 100mA maximum can be delivered by the +5V output, thus to be shared among all the resistive circuits.



Example of simple circuit for temperature sensing using NTC

Note that the M1 to M8 inputs are also available on the programming connector under the label E1 to E8. It allows to connect the experiments via a flat ribbon cable.

AERIAL/ANTENNA

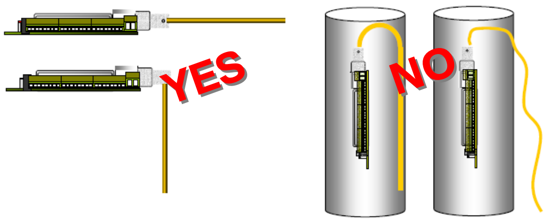
Two aerial connections are available on Kiwi:

- whip antenna soldering pad
- BNC antenna connector

In the case the club has not much experience in radio frequency transmission, the whip antenna is strongly recommended.
The whip antenna consists in a 1/4th of wavelength metal rod, 50 Ohms impedance. This is a general propagation rule of thumb to have an antenna length equal to 1/4th of wavelength.
The relation between frequency and wavelength is reminded below:

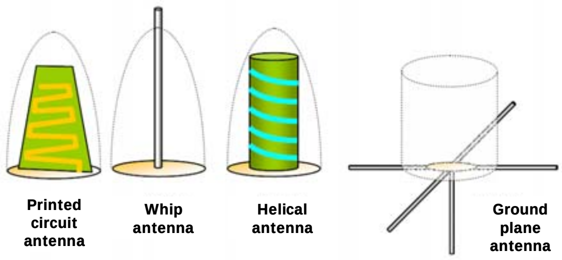
$$Frequency[Hz] = \frac{celerity(speed\ \ of\ \ light)}{wavelength} = \frac{c[m/s]}{\lambda[m]}$$

Thus, for the 137.950 MHz frequency, the wavelength is about 2.17 meters. 1/4th of this length, that is to say 0.54 meters, is enough to provide an efficient transmission.



Whip antenna configurations to prefer (left) and to avoid (right)

The whip antenna from Planète Sciences is almost 1/4th wave and is provided to the club upon request. It has to be carefully and strongly soldered on the pad, in order to achieve good performance…and not to drop during the launch of the rocket!
BNC Antenna is mostly preferred when the integration requires specific antenna shape. An omnidirectional antenna is required in the case of rocket or balloon projects, as the antenna cannot be directed permanently towards the receiver.
The antenna must have a 50 Ohms impedance in order to transmit a maximum of power from the transmitter and to avoid overheating. (SWR: **S**tanding **W**ave **R**atio close to 1, ie a minimum of reflection along the antenna)



Several types of antenna for rockets and balloons

How to build the antenna ?

- Whip antenna: get a copper or brass rod, cut it at the length corresponding to ¼ of you frequency equivalent wavelength. This is a very simple process, sufficient for almost every rocket and balloon launched by Planète Sciences.

- An alternative solution consists in purchasing a BNC antenna for a handheld wide frequency receiver or transmitter.

How to test your transmission with KIWI?

The use of the 137.950/138.50 MHz frequencies is illegal in most countries!
This frequency is allocated to CNES,
and there are strict conditions to make use of it.

Simply terminate the antenna BNC connector with a 60dB attenuator; you can easily find some 20dB attenuator in electronics shop. Get 3 to reach 60dB; get also a 50 Ohms termination load to terminate the circuit.

For example see http://www.jyebao.com.tw/
and browse the Attenuators and Termination sections.

Select items with a max frequency of 1GHz, and a power less than 1W.

In such condition the output level of Kiwi can be calculated as follows:

- Kiwi Max output: Pout=329mW -> 10Log[Pout/1mW]=25.17 dBm
- Attenuation: 60 dB
- Output power with attenuator: 25.17 dBm-60dB=-34.82 dBm, i.e. 330nW

uW	dBm
1	-30.0
2	-27.0
3	-25.2
4	-24.0
5	-23.0
6	-22.2
7	-21.5
8	-21.0
9	-20.5
10	-20.0
20	-17.0
30	-15.2
40	-14.0
50	-13.0
60	-12.2
70	-11.5
80	-11.0
90	-10.5
100	-10.0
200	-7.0
300	-5.2
400	-4.0
500	-3.0
600	-2.2
700	-1.5
800	-1.0
900	-0.5
1000	0.0

mW	dBm
1	0.0
2	3.0
3	4.8
4	6.0
5	7.0
6	7.8
7	8.5
8	9.0
9	9.5
10	10.0
20	13.0
30	14.8
40	16.0
50	17.0
60	17.8
70	18.5
80	19.0
90	19.5
100	20.0
200	23.0
300	24.8
400	26.0
500	27.0
600	27.8
700	28.5
800	29.0
900	29.5
1000	30.0

W	dBm
1	30.0
2	33.0
3	34.8
4	36.0
5	37.0
6	37.8
7	38.5
8	39.0
9	39.5
10	40.0
20	43.0
30	44.8
40	46.0
50	47.0
60	47.8
70	48.5
80	49.0
90	49.5
100	50.0
200	53.0
300	54.8
400	56.0
500	57.0
600	57.8
700	58.5
800	59.0
900	59.5
1000	60.0

Watt to dBm conversion table is provided here for information.

You may find a similar conversion chart available at the address:
http://www.moseleysb.com/mb/mv2dbm.html

330 nW radiated is extremely low, so you won't propagate your signal over more than 3 to 5 meters, that is to say the signal will remain within your laboratory and you won't jam anyone outside. Therefore you remain within the law and this distance is enough for you to perform your tests.

INTERNAL MODULATION FOR ANALOG SIGNALS

The Kiwi's internal modulation (running the original microcontroller code) is **exclusively** for the transmission of the 8 analog data channels. It cannot be used for digital signals.
The following description is given for information only. When using the analog inputs, the user does not have to worry about the modulation.

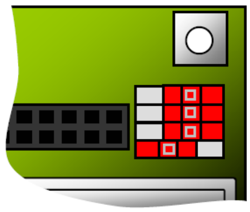
Internal Modulation description

- The 8 channels [M1 to M8], with a dynamic range of 5V, are connected to a 8:1 multiplexer that first selects the channel for which the signal will be digitized by the PIC.
- Once all the channels are digitized, a digital frame is processed and clocked at 600 bits/s.
- PIC then generates 4-bit resolution sine wave converted by a Digital to Analog Converter into a FSK signal (Frequency Shift Keying)
- An ADC then shapes the signals to modulate the high frequency oscillator.
- The oscillator is included within a phase-lock loop to achieve high stability.
- A HF amplifier provides the required power.

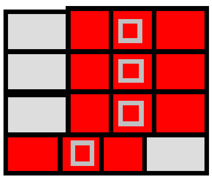
Configuration switches

4 switches are available on Kiwi, which allow the user to:

- Modify the frequency
- Select the source for modulation
- Select the number of frames
- Select the transmission rate of the frames



Frames transmitted once
Frames transmitted continuously (2Hz)
Frequency: 138.500 MHz
Internal modulation



Frames transmitted 3 times
Frames transmitted e very 2 sec (0.5Hz)
Frequency: 137.950 MHz
External Modulation

Default switches positions

In the case of a weather balloon project, it is very useful to transmit the same frame 3 times in a row because the transmitted signal can be weakened over the distance: the repetition of the frames is increasing the chance to receive at least one valid frame.

In the case of a rocket project, more data have to be transmitted because the parameters vary quickly: switches will be adjusted so that the frames are not repeated and are transmitted continuously.

The maximum rate is 2 frames/sec with internal modulation mode! If you need a higher data rate, consider using external modulation.

Note that with the "frames transmitted every 2 sec" configuration, the oscillator is shut down between the transmissions in order no to alter the measurements.

EXTERNAL MODULATION FOR DIGITAL SIGNALS

An **external modulator** is **necessary** for the transmission of **digital data** signals. The user may also choose to design its own modulator for analog inputs.

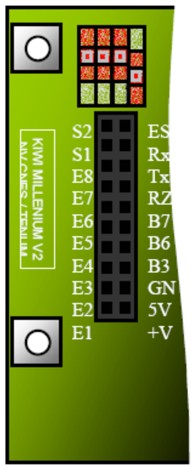
The following description provides general information on how to interface the Kiwi with an external modulator, but doesn't describe how to design or build an external modulator, what is under the responsibility of the user. However, Planète Sciences has released some documents that will help the clubs to design their own modulator. See further section.

External Modulation explanations

The kiwi internal modulator is capable of transmitting digital frames at a 600 bits/sec rate. To achieve higher data rate, especially for experiments onboard rockets, it is possible to use an external modulator to make a wider use of the transmitter bandwidth. Planète Sciences recommends the use of SNR modulation scheme (a FSK modulation with 0xFF as synchronization value) but others can be chosen. Just keep in mind that a few people will probably master an exotic transmission protocol and that volunteers within Planète Sciences will probably not be able to assist you in debugging any modulation related problem.

There is no limit in the number of parameters to transmit. The frame generated only depends on the user. However, for a given data-rate, when the number of parameters increases, the frame gets longer, and fewer samples of data can be transmitted.
The frame is generated ahead of the modulator. See further section.

The Digital & Analog data and programming connector



The 20 holes HE-14 connector has to be used for external modulation. It provides with the following connections:

- DC Power supply input (+V)
- Output of the DC +5V regulated voltage (5V)
- Ground (GND)
- Analog channels input E1 to E8 (same as M1 to M8)
- External modulation input (ES)
- Programming inputs/outputs (B3, B6, B7, RZ)
- Digital user input/outputs (S1, S2, TxD, RxD)

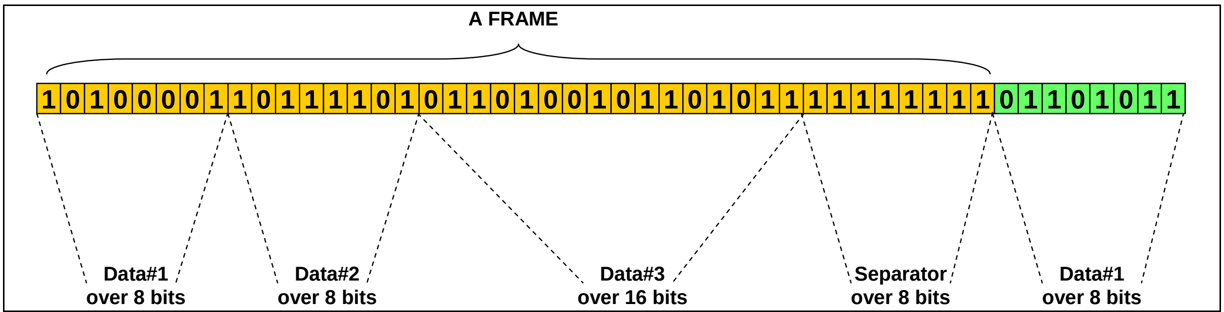
As mentioned in the foreword, the Kiwi allows to transmit the modulated data stream. ES input will be used for that purpose while TxD and RxD are used to re-program the PIC.

Framer

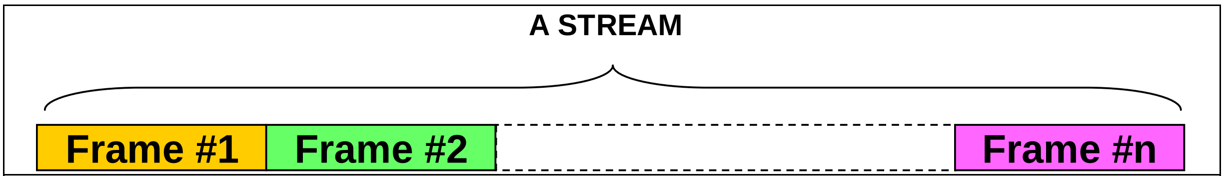
The digital data have to be serialized prior to the modulation: that is the purpose of the framer. It will act as a multiplexer and generates:

1. a serial frame of the digital data
2. a serial stream of serial frames.

Each frame is separated from the previous one by a “separator”, that is to say a special value that will be recognized by the de-framing software to identify the beginning of the frame; the data within a frame are separated based on their bit length (int, long…).



Example of content of a frame of 3 data with different lengths



Content of a stream

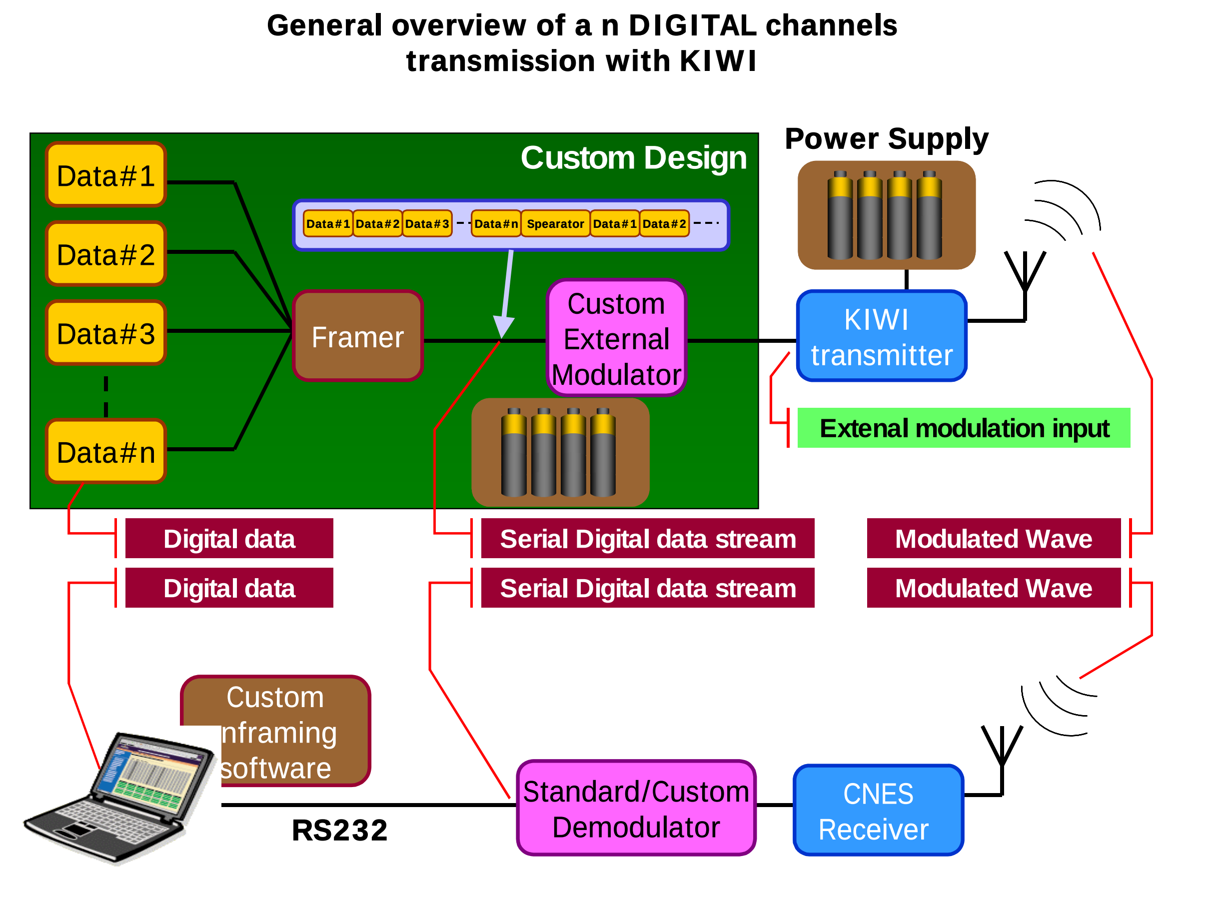
The most simple way to design a framer is to include dedicated code within the program of your control board processor.

Modulator

The modulator will slightly modify the shape of the analogue signal at the input of the transmitted, in order to embed within it the digital information (0 and 1). Modulation is neither a conversion nor a transformation, it is the modification of information by another, but both information remain valid and can be interpreted separately.

The modulator frequency will be chosen depending on the data rate to be obtained. Maximum modulation frequency is about 15 KHz.
The quality of the signal will depend on the trade-off between the modulation frequency and the peak to peak amplitude of the signal. Low amplitude will force to reduce the bandwidth, conversely a large bandwidth (higher data rate) will force the amplitude to be increased. Balance is then to be determined by the user between the range of the transmission and the data rate of the information to transmit. A best practice is to be able to tune the peak-to-peak amplitude from 0.1V to 1V with an adjustable resistor.

Synopsys of a digital transmission



How to design a modulator (and a demodulator):

It has to be understood that the modulator for Kiwi is not available "off the shelf" and has to be designed according to the user's specific criteria.
Planète Sciences recommends the use of the XR2211 and XR2206 chips.
See:
http://www.Planète-sciences.org/espace/publications/fichiers/vco_xr2206.pdf

To be compatible with the FSK demodulators that Planète Sciences uses you should use a FSK modulation scheme, such as listed in the "Receiver & Demodulator" section.

Moreover, it is suggested that you use a RS232 serial data stream to encapsulate your data. You should also send the data at 4800 bit/s, as standard demodulators from Planète Sciences are compatible with this bit rate.

RECEIVER & DEMODULATOR

Planète Sciences equipment includes all the necessary devices to receive signals transmitted by Kiwi, and to demodulate an SNR or FSK modulated stream. You are welcome to make use of this equipment, but you can also design and manufacture your own, what is a challenge that clubs prefer not to focus on. The experiments are, before all, contained within the rocket or the balloon payload!

Receiver

The user, for test purpose, may purchase a wideband receiver, such as Yaesu VR120D ; Maycom AR108 cannot be used in wideband mode.
The receiver must be tunable to receive data at the same frequencies as KIWI.

Demodulator

To be compatible with the FSK modulators that Planète Sciences uses, you should use a FSK modulation scheme such that:

- Low state '0" is coded by a 9 kHz frequency signal
- High state '1' is coded by a 15 kHz frequency signal

In a first step, it is not necessary to design and build a demodulator. As long as the frames/stream is mastered and the modulation is performed properly with the frequencies mentioned above, the user can have strong confidence in the signal received. The equipment made available by Planète Sciences will confirm it.
Note that most of the clubs who prepare a project to be launched/released by Planète Sciences rely on this method.

The table blow lists the demodulators available from CNES/Planète Sciences:

Bit rate	Keying Frequency	Low/High state for FSK modulation
600 bit/s	900 Hz	Low
600 bit/s	1500 Hz	High
1200 bit/s	1200 Hz	Low
1200 bit/s	2200 Hz	High
4800 bit/s	9000 Hz	Low
4800 bit/s	15000 Hz	High
9600 bit/s	14400 Hz	Low
9600 bit/s	24000 Hz	High

A best practice consists in tuning frequency keying with adjustable resistors.

DATA ACQUISITION

Data acquisition is performed on a computer: data are received from the RS232 or USB ports in order:

- to perform the stream/frames truncation
- to process and display data according to the calibration of the sensors

Planète Sciences volunteers have developed several tools, which can sort out data collected from the output othe demodulator, through R232 port:

- Kicapt, for Kiwi configured with internal modulation mode (analog inputs)
- K-Com (any frame starting with 0xFF)
- K-easy (custom frame, GPS coordinates)

This tool can be downloaded

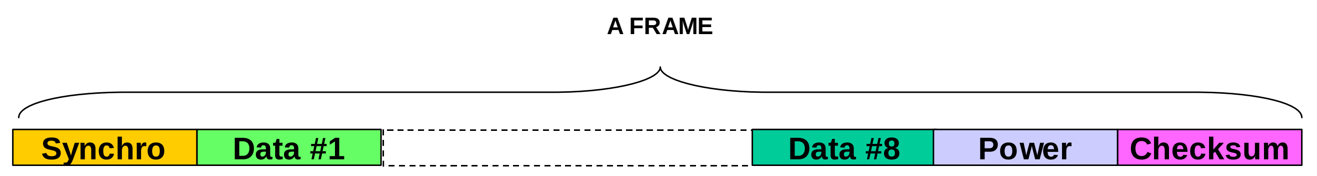
These tools are available here:

http://www.planete-sciences.org/espace/basedoc/public/index.php/Logiciels_de_r%C3%A9ception_des_t%C3%A9l%C3%A9mesure

KICAPT

Kicapt is intended for use by clubs which are still novice with data acquisition, or who don't have time to develop their own tool.

It handles the reception frames when KIWI is set for full telemetry, that is to say a data rate of 600 bits/s and a frame format as below:



Data frame for reception with Kicapt

Kicapt assumes that one frame consists of data coded over 8 bits (one byte) transmitted in the following order:

1. Synchronization byte ("1111111" or $FF)
2. Data #1
3. Data #2
4. Data #3
5. Data #4
6. Data #5
7. Data #6
8. Data #7
9. Data #8
10. Power supply level
11. Checksum for data validation

Moreover, 4 transmission modes can be achieved, depending on the switches configuration (see above chapter):

Data acquisition frequency	Redundancy	Transmission type	Explanation
0.5 Hz	3	Sequential	frames are sent 3 times in a row, every 2 seconds; no transmission in between.
0.5 Hz	1	Sequential	Data frame is sent 1 time, every 2 seconds; no transmission in between
1.5 Hz	3	Permanent	Data frames are sent 3 times in a row, continuously; permanent transmission
2 Hz	1	Permanent	Data is sent 1 time, continuously; permanent transmission

- With a sequential modulation mode, KIWI transmission is continuous, but modulation is turned off after transmission of the data.

- Data are saved sequentially in a text file with a CSV format, which can be loaded easily in a spreadsheet for exploitation.

- When Kicapt rejects inconsistent frames, they are not saved in the file.

Data	Time	M1	M2	M3	M4	M5	M6	M7	M8	Alim	Checksum
08/10/2007;	16:17:23;	3,51;	2,92;	3,48;	2,67;	3,43;	1,28;	3,38;	4,68;	2,54;	OK

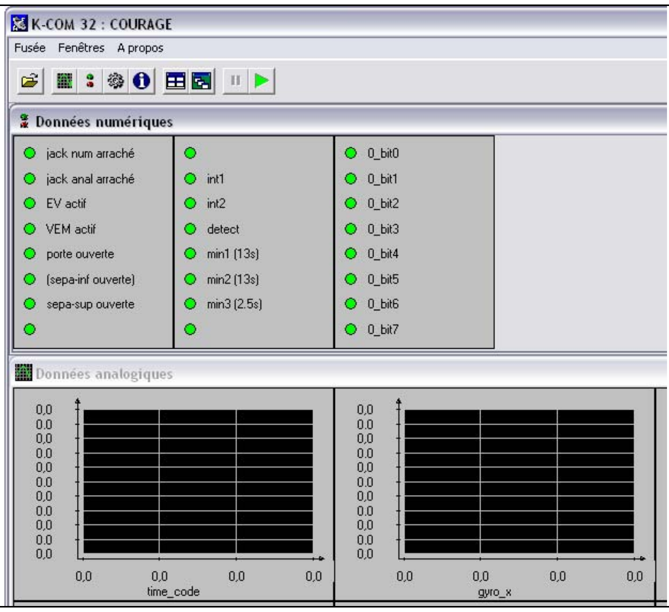
Example of data frame saved by Kicapt in the CSV output file

- Kicapt processes the synchronization byte, but it is not written in the output file.
- Power supply level delivers information about the KIWI internal 5V power supply: analog voltage is divided by 3, with the maximum value $FE=4.98V : 4,98 x 3 = 14,94V ; so when the reading is $CC, the actual voltage is 12V.
- Checksum is used to check the validity of the data frame: it is equal to the sum of all the bytes of the frame, modulo 255, and then divided by two. Kicapt will perform the checksum calculation upon reception of the frame and will compare it with the checksum received. If they are inconsistent, the smiley on the screen will turn red and the data frame is not stored in the output file.
- With a redundancy transmission mode, Kicapt compares the checksum with the data frames and stores the first one with matching checksums.

K-COM and Keasy softwares presented below, are intended for clubs using "external modulation" telemetry, with specific bit rate and data frame.

K-COM

The software lets the user describe each byte of your frame, starting with 0xFF, in a .cnf text file.
It can plot each analog byte, and shows 8 logical states for digital byte.
Data are saved in a .dat text file that can be imported in a spreadsheet (CSV with 0-255;0-1).
From there, the user can post-process the data in order to get back to physical parameters, and to analyze the experimental results.



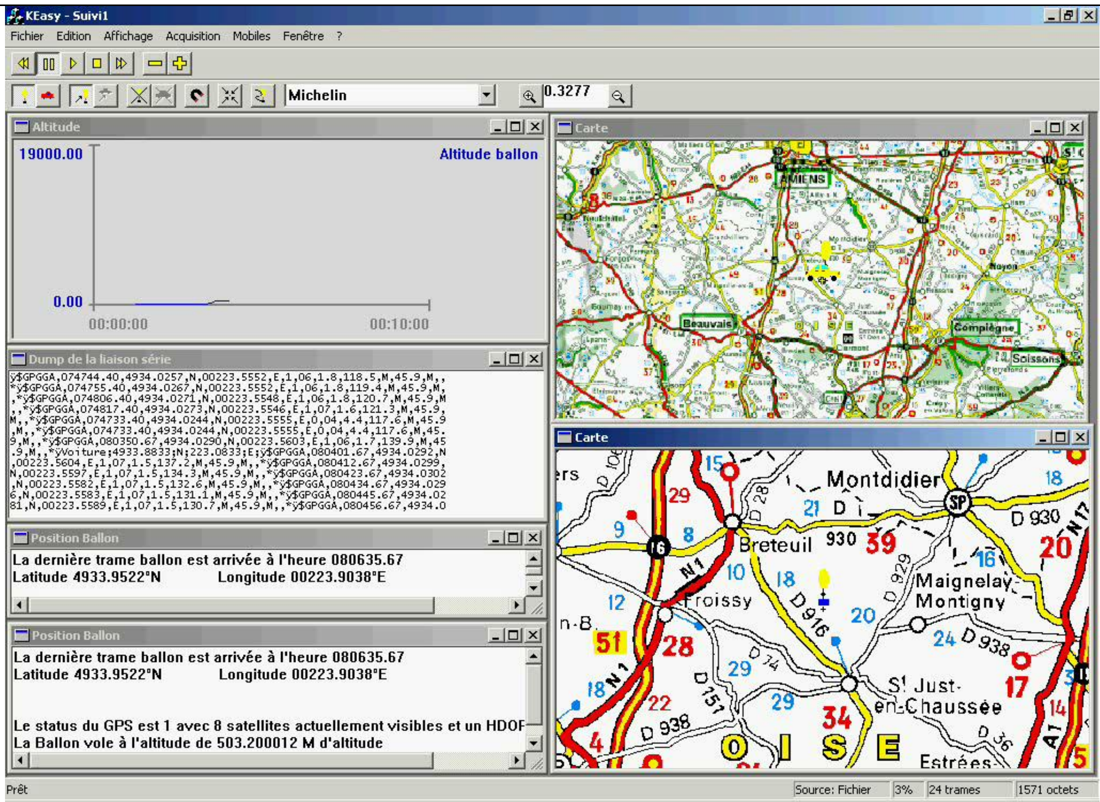
K-COM screenshot

KEasy

The software invites the user to describe frame and plots in some .ini text files.
K-easy is much customizable (data type = int8, float …), but is more difficult to configure than previous software described.
Functions available include:

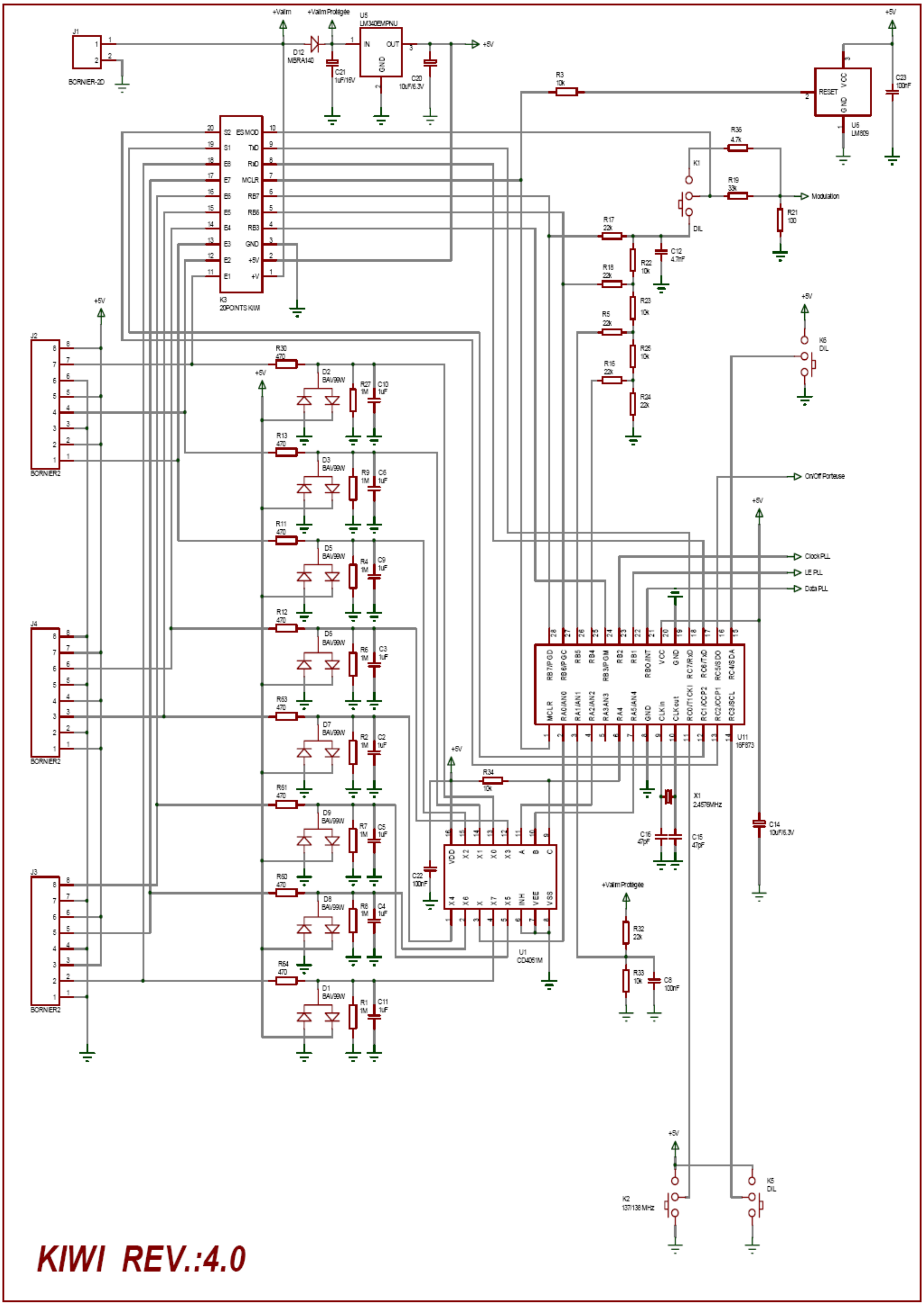
- Hexa or ASCII dump output,
- plots or textual view of data with mathematical formula,
- plot on a map of the position extracted from GPS data stream,
- replay of a recorded sequence.

With current version, only the binary data stream is saved as a .bin file.



Keasy screenshot

KIWI V4 partial schematics



ORIGINAL PIC PROGRAM

```
;*********************************************************
;*                                                       *
;* Programme de gestion de l'émetteur KIWI-ME REVISION 4 *
;* ---------------------------------------------------   *
;*                Mars 2002 / Septembre 2004             *
;*                                                       *
;*                       Version 1.1                     *
;*                         PIC16F873                     *
;*                                                       *
;* N. VERDIER / E. GARDELLE / G. BOSCH                   *
;*********************************************************

          LIST p=16F873
          include "p16f873.inc"

;*********************************************************
;*                                                       *
;*                DECLARATION DES CONSTANTES             *
;*                                                       *
;*********************************************************

;REGISTRES

OPTIO     EQU 0x01
CARRY     EQU 0x00
ZERO      EQU 0x02
TFLAG     EQU 0x02
RTS       EQU 0x05
PSA       EQU 0x03
TOIE      EQU 0x05
TOIF      EQU 0x02

;BITS
F_ANA     EQU 0
F_NUM     EQU 1
F_TIM     EQU 2
CS        EQU 0
CLK       EQU 1
DI        EQU 2
DO        EQU 3
CLOCK     EQU 2 ;
DAT       EQU 0
LE        EQU 1
COD_P     EQU 0x0AA
BASE_TAB  EQU 0x041
TRAME EQU 0x043
FET       EQU 5

;VARIABLES
count     equ H'2E'
pointeur  equ H'2F'
compteur  equ H'30'
config    equ H'31'
result    equ H'32'
mux       equ H'33'
c_ser     equ H'34'
c_eeprom  equ H'35'
Fsr_mem0  equ H'36'
Fsr_mem1  equ H'37'
tempo     equ H'38'
etat      equ H'39'
point     equ H'3A'
c_octet   equ H'3B'
com_bit   equ H'3C'
tampon    equ H'3D'
Rboot0    equ H'3E'
Rboot1    equ H'3F'
R0        equ H'40'
R1        equ H'41'
R2        equ H'42'
R3        equ H'43'
R4        equ H'44'
R5        equ H'45'
R7        equ H'47'
```

```
R8        equ H'48'
R9        equ H'49'
R10       equ H'4A'
alim      equ H'4B'
chksum    equ H'4C'
tim0      equ H'4D'
tim1      equ H'4E'
COUNT     EQU H'50'
TRANSIT   EQU H'51'
BUFF1     equ H'52'
BUFF2     equ H'53'
BUFF3     equ H'54'
TSFER     equ H'55'

          org 0x0000
          goto Debut

          Org 0X0040
          DATA " K I W I "
          ORG 0X0048
          DATA " R E V . 4 "
          ORG 0X0050
          DATA " 1 5 / 0 9 / 04"

;***********************************************************
;*                                                         *
;*                  MAIN PROGRAM                           *
;*                                                         *
;***********************************************************

          org 0xA0

Debut     call init ; Appel de la routine d'initialisation
          Btfsc PORTC,0 ; \
          call PLL137 ; suivant la valeur de RC0
          btfss PORTC,0 ; Chargement de la synthese 137 ou 138
          call PLL138 ; /
          call init2 ; Initialisation des variables
          btfss PORTC,1 ; \
          goto Sauve2 ; suivant la valeur de RC0
Sauve     bcf STATUS,6 ;
          bsf STATUS,5            ;Page 1
          movlw B'11011001'       ; RC0,RC3,RC4,RC6,RC7 en entrées
          movwf TRISC             ;RC1,RC2,RC5 en sortie
          bcf STATUS,5            ; page 0
          movlw 0x0FD             ;
          movwf Rboot0            ; Mot de synchro 6 bits a 1 + FF
          movlw 0x0FF             ;
          movwf Rboot1            ;
          call Ana                ; Conversion A/N des 8 voies et stockage en RAM
          call satur              ;
          call c_check            ; Calcul de la checksum
          bsf PORTC,2             ; flag indiquant l'émission
          bcf PORTC,FET           ; commande du FET fermeture
          movlw 0x0F              ; tempo 25ms
          movwf tim0              ; pour stabiliser le PA
          call tempo1             ; /
          CLRWDT
          movlw 0x0F              ; tempo 25ms
          movwf tim0              ; pour stabiliser le PA
          call tempo1             ; /
          CLRWDT
          movlw 0x0F              ; tempo 25ms
          movwf tim0              ; pour stabiliser le PA
          call tempo1             ; /
          CLRWDT
          movlw 0x0F              ; tempo 25ms
          movwf tim0              ; pour stabiliser le PA
          call tempo1             ; /
          CLRWDT
          call Emission           ; Appel de la routine d'emission
          bcf PORTC,2             ; flag indiquant la fin de l'émission
          btfss PORTC,3           ; si l'entrée nbtrame est à 0
          goto tst_rate           ; alors on teste le delai entre les trames
          movlw 0x14              ; sinon on lance 3 trames
          movwf tim0              ;
          call tempo1             ;
          call Envoi              ; Appel de la routine d'emission
          movlw 0x14              ;
          movwf tim0              ;
          call tempo1             ;
          call Envoi              ; Appel de la routine d'emission
```

```
tst_rate  btfss PORTC,4 ; si RC4 vaut 0
          goto Sauve             ; alors trame continue
                                 ; sinon,
          bsf PORTC,FET          ; commande du FET ouverture
          movwf tim0             ; tempo de 2s
          call tempo1            ;
          movlw 0xFF             ;
          movwf tim0             ;
          call tempo1            ;
          movlw 0xD4             ;
          movwf tim0             ;
          call tempo1            ;
          goto Sauve             ;
          Sauve2
          bcf STATUS,6           ;
          bsf STATUS,5           ;Page 1
          movlw B'11011001'      ;RC0,RC3,RC4,RC6,RC7 en entrées
          movwf TRISC            ;RC1,RC2,RC5 en sortie
          bcf STATUS,5           ; page 0
          bcf PORTC,FET          ; commande du FET fermeture
          bsf PORTC,2            ; flag indiquant l'émission
          movlw 0x0FD            ;
          movwf Rboot0           ; Mot de synchro 6 bits a 1 + FF
          movlw 0x0FF            ;
          movwf Rboot1
          call Ana               ; Conversion A/N des 8 voies et stockage en RAM
          call satur             ; ????
          call c_check           ; Calcul de la checksum
          call Emission          ; Appel de la routine d'emission
          btfss PORTC,3          ; si l'entrée nbtrame est à 0
          goto tst_rate2         ; alors on teste le delai entre les trames
          movlw 0x14             ; sinon on lance 3 trames
          movwf tim0             ;
          call tempo1            ;
          call Envoi             ; Appel de la routine d'emission
          movlw 0x14             ;
          movwf tim0             ;
          call tempo1            ;
          call Envoi             ; Appel de la routine d'emission

tst_rate2 btfss PORTC,4          ; si RC4 vaut 0
          goto Sauve2            ; alors trame continue
                                 ; sinon,
          bcf PORTC,FET          ; commande du FET ouverture
          movwf tim0             ; tempo de 2s
          call tempo1            ;
          movlw 0xFF             ;
          movwf tim0             ;
          call tempo1            ;
          movlw 0xD4             ;
          movwf tim0             ;
          call tempo1            ;
          goto Sauve2            ;

;************************************************************
;*                                                          *
;*                TIMER SUB-PROGRAM                         *
;*                                                          *
;************************************************************

tempo1    movlw 0xFF
          movwf tim1
          decfsz tim0,1
          goto tempo2
          return
          tempo2 clrwdt
          decfsz tim1,1
          goto tempo2
          goto tempo1

;************************************************************
;*                                                          *
;*                SATURATION SUB-PROGRAM                    *
;*                                                          *
;************************************************************

satur     movlw 0x09
          movwf compteur
          movlw TRAME
          movwf FSR
          satur1 movf INDF,0
```

```
          movwf tempo
          movlw 0x0FF
          xorwf tempo,0
          btfsc STATUS,2
          decf tempo,1
          movf tempo,0
          movwf INDF
          incf FSR,1
          decfsz compteur,1
          goto satur1
          return

;************************************************************
;*                                                          *
;*                  CHECKSUM SUB-PROGRAM                    *
;*                                                          *
;************************************************************

c_check   movlw 0x08              ;chargement du compteur d'octet
          movwf compteur          ;a sommer avec 8 entree + alim
          movlw TRAME             ;chargement de l'adresse de R2
          movwf FSR               ;chargement dans FSR
          movf INDF,0             ;chargement dans W du contenu de R2
          movwf chksum            ;sauvegarde dans chksum
          incf FSR,1              ;registre suivant
          movf INDF,0             ;chargement du registre suivant dans W
          addwf chksum,1          ;addition du registre a chksum
          decfsz compteur,1       ;decremente compteur
          goto suivant            ;si compteur #0 alors registre suivant
          bcf STATUS,CARRY
          rrf chksum,1            ;division par 2 de la chksum (#FF)
          return                  ;sinon retour

;************************************************************
;*                                                          *
;*          ANALOG/DIGITAL CONVERSION SUB-PROGRAM           *
;*                                                          *
;************************************************************

Ana       bsf PORTC,1             ; Indique le début des acquisitions
          movlw TRAME
          movwf FSR
          movlw 0x08              ;
          movwf compteur          ; Chargement du compteurde voie
Ana_next  movf compteur,0
          movwf mux
          decf mux,1
          bcf STATUS,CARRY
          rlf mux,1               ;Decalage de mux pour le
          bcf STATUS,CARRY        ;multiplexage des entrees
          rlf mux,1               ;
          btfsc mux,3             ;si le bit 3 est à 1
          bsf mux,5               ;alors on place le bit 5 à 1
          btfss mux,3             ;sinon si le bit 3 est à 0
          bcf mux,5               ;alors on place le bit 5 à 0
          bcf mux,3               ;le bit 3 est forcé à 0
          movf mux,w              ;
          movwf PORTA             ; Commande du multiplexeur
          movlw 0x0A              ;
          movwf tim0              ;
          call tempo1             ;Attente de stabilisation niveau
          bsf ADCON0,2            ; Debut de conversion
Ana_wait btfsc ADCON0,2           ; Attente de fin de conversion
          goto Ana_wait           ;
          movf ADRESH,w           ;
          movwf INDF              ;
          incf FSR,1              ;
          movlw 0x02              ;
          movwf tim0              ;
          call tempo1             ;Attente de stabilisation niveau
          decfsz compteur,1       ;
          goto Ana_next           ;
          bsf ADCON0,3            ; Acquisition de la tension d'alimentation
          bcf ADCON0,4            ; selectionne la voie RA1
          bcf ADCON0,5            ; /
          bsf ADCON0,2            ; Lance la conversion
Ana_wai1  btfsc ADCON0,2          ; Attente de fin de conversion
          goto Ana_wai1           ;
          bcf ADCON0,3            ; Repositionnement entree donnees (RA0)
          movf ADRESH,w           ;
          movwf INDF              ;
          bcf PORTC,1             ;indique la fin des acquisitions
```

```
          return

;**********************************************************
;*                                                        *
;*        INIT SUB-PROGRAM                                *
;*                                                        *
;**********************************************************
init      clrwdt                  ;Mise a zero du chien de garde
          bsf STATUS,5            ;
          bcf STATUS,6            ;Page 1
          clrf OPTIO              ;
          movlw B'00000100'       ;Justification à gauche
          movwf ADCON1            ;RA0, RA1 et RA3 sont des entrees ana
                                  ;RA2 et RA4 sont des entrees numériques
          movlw B'001011'         ;RA0,RA1 et RA3 sont des entrees
          movwf TRISA             ;RA2, RA4 sont des sorties
          movlw B'00001000'       ;RB0,RB1,RB2,RB4,RB5,RB6 et RB7
                                  ;en sortie
          movwf TRISB             ;RB3 en entree
          movlw B'11011011'       ;RC0,RC1,RC3,RC4,RC6,RC7 en entrées
          movwf TRISC             ;RC2,RC5 en sortie
          bcf STATUS,5            ;Retour page 0
          bcf ADCON0,6            ;Selection de l'horloge
          bsf ADCON0,7            ;fosc/32
          bcf ADCON0,5            ;
          bcf ADCON0,4            ;Positionnement du multiplexeur
          bcf ADCON0,3            ;RA0
          bsf ADCON0,0            ;Le convertisseur est operationnel
          bcf INTCON,7            ;Inhibition des interruptions
          clrf PIE1               ;
          clrf PIR1               ;
          clrf PIE2               ;
          clrf PIR2               ;
          return
init2     clrf R0                 ; initialisation des registres de donnees
          clrf R1                 ;
          clrf R2                 ;
          clrf R3                 ;
          clrf R4                 ;
          clrf R5                 ;
          clrf R6                 ;
          clrf R7                 ;
          clrf R8                 ;
          clrf R9                 ;
          clrf R10                ;
          clrf alim               ;
          clrf chksum             ;
          clrf compteur           ; initialisation du compteur … 8
          bsf compteur,3          ;
          clrf config             ; initialisation config
          clrf result             ; initialisation resultat
          return
;**********************************************************
;*                                                        *
;*        FRAME TRANSMISSION SUB-PROGRAM                  *
;*                                                        *
;**********************************************************
Emission: movlw 0x08
          movwf c_octet
          clrf pointeur
          movlw 0x0B
          movwf count
          clrf com_bit
          bsf com_bit,3
          clrf compteur           ; initialisation du compteur
          clrf config             ; initialisation config
          clrf result             ; initialisation resultat
          ;
;|||||||||||||| Pre-processing |||||||||||||||||||
;
;         movf c_octet,0
;         movwf count
          movlw TRAME
          movwf Fsr_mem0          ; Fsr_mem0 <= TRAME (0x43)
          movwf FSR
          movf INDF,0
          movwf tampon
          decf FSR,1
          decf FSR,1
          decf FSR,1
          movf FSR,0
          movwf Fsr_mem1
```

```
Debut_T:  call Inc_tab
          bcf STATUS,CARRY
          call Set_bit
Nouveau:  call Inc_tab
          rrf tampon,1
          call Set_bit
          decfsz c_octet,1
          goto Nouveau
          call Inc_tab
          bsf STATUS,CARRY
          call Set_bit
          movf FSR,0
          movwf Fsr_mem1
          movf Fsr_mem0,0
          movwf FSR
          incf FSR,1
          movf INDF,0
          movwf tampon
          movf FSR,0
          movwf Fsr_mem0
          movf Fsr_mem1,0
          movwf FSR
          bsf c_octet,3
          decfsz count,1
          goto Debut_T
          bsf chksum,0
          bsf chksum,1
          bsf chksum,2
          bsf chksum,3
          goto Envoi
Inc_tab:  incf pointeur,1
          btfss pointeur,3
          return
          btfss pointeur,0
          return
          clrf pointeur
          bsf pointeur,0
          incf FSR,1
          return
Set_bit:  movf INDF,0
          movwf tempo
          movf STATUS,0
          movwf etat
          movf pointeur,0
          movwf point
New_rot:  rlf tempo,1
          decfsz pointeur,1
          goto New_rot
          btfss etat,CARRY
          bcf STATUS,CARRY
          btfsc etat,CARRY
          bsf STATUS,CARRY
          movf point,0
          movwf pointeur
New_rot1: rrf tempo,1
          decfsz pointeur,1
          goto New_rot1
          movf tempo,0
          movwf INDF
          movf point,0
          movwf pointeur
          movf etat,0
          mo vwf STATUS
          return
;""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
Envoi:    clrf pointeur
          clrf count
          bsf count,5
          clrf com_bit
          bsf com_bit,3
          movlw 0x0F
          movwf c_octet
          movlw TRAME-5
          movwf FSR
          movf INDF,0
          movwf config
          Retour1: nop
          nop
          nop
          nop
          nop
          nop
```

```
Retour2:  nop
          nop
          nop
          nop
          nop
Retour3:  clrwdt ; Clear Watchdog Timer
Begin:    movlw 0x06
          addwf pointeur,1
          movlw 0x04
          btfsc config,7
          addwf pointeur,1
          rrf pointeur,0
          movwf tampon
          rrf tampon,1
          movf tampon,0
          andlw 1F
          call Table
          m ovwf PORTB
          decfsz count,1
          goto Retour1
          decfsz com_bit,1
          goto Suite
          bsf com_bit,3
          decfsz c_octet,1
          goto Suite1
          return
Suite1:   incf FSR,1
          movf INDF,0
          movwf config
          bsf count,5
          goto Retour3
Suite:    rlf config,1
          bsf count,5
          goto Retour2

;************************************************************
;*                                                          *
;*        137.950 FREQUENCY SYNTHESIS CODING SUB-PROGRAM    *
;*                                                          *
;************************************************************
;
; LA FREQUENCE AUTORISEE EST DE 137.950 MHz
; LA FREQUENCE DU QUARTZ EST DE 13.000 MHz
; LA FREQUENCE DE REFERENCE EST DE 25.000 KHz
; fvco = [(P x B) + A] x fosc / R
;         P = 8
;         N = fvco / fref = 5518 = 158E h
;         R = fosc / fref = 520= 208 h
;         B = div(N/P) = 689 = 2B1 h
;         A = N - B x P = 6 = 6h

PLL137    CLRF BUFF1             ;effacement de BUFF1
          CLRF BUFF2             ;effacement de BUFF2
          CLRF BUFF3             ;effacement de BUFF3
          CLRF PORTB
; 1) FUNCTION AND INITIALIZATION LATCHES
          movlw H'00'            ; \
          movwf BUFF1            ; \
          movlw H'04'            ; chargement des valeurs d'init
          movwf BUFF2            ; dans des variables
          movlw H'10'            ; /
          movwf BUFF3            ; /
          movf BUFF1,0           ; \
          movwf TSFER            ; envoi du premier octet
          call TFRoctet          ; /
          movf BUFF2,0           ; \
          movwf TSFER            ; envoi du deuxieme octet
          call TFRoctet          ; /
          movf BUFF3,0           ; \
          movwf TSFER            ; envoi des 5 derniers bits
          call TFR_5bits         ; /
          bcf PORTB,DAT          ; efface le bit de données
          bsf PORTB,LE           ; Validation des données
          bcf PORTB,LE           ; /
; 2) PROGRAMMABLE REFERENCE DIVIDER (R COUNTER)
          movlw H'00'            ; \
          movwf BUFF1            ; \
          movlw H'41'            ; chargement de la valeur de R
          movwf BUFF2            ; dans des variables
          movlw H'00'            ; /
          movwf BUFF3            ; /
          movf BUFF1,0           ; \
```

```
         movwf TSFER            ; envoi du premier octet
         call TFRoctet          ; /
         movf BUFF2,0           ; \
         movwf TSFER            ; envoi du deuxieme octet
         call TFRoctet          ; /
         movf BUFF3,0           ; \
         movwf TSFER            ; envoi des 5 derniers bits
         call TFR_5bits         ; /
         bcf PORTB,DAT          ; efface le bit de données
         bsf PORTB,LE           ; Validation des données
         bcf PORTB,LE           ; /
; 3) PROGRAMMABLE DIVIDER (N COUNTER = A et B)
         movlw H'0A'            ; \
         movwf BUFF1            ; \
         movlw H'C4'            ; chargement des valeurs de A et B
         movwf BUFF2            ; dans des variables
         movlw H'C8'            ; /
         movwf BUFF3            ; /
         movf BUFF1,0           ; \
         movwf TSFER            ; envoi du premier octet
         call TFRoctet          ; /
         movf BUFF2,0           ; \
         movwf TSFER            ; envoi du deuxieme octet
         call TFRoctet          ; /
         movf BUFF3,0           ; \
         movwf TSFER            ; envoi des 5 derniers bits
         call TFR_5bits         ; /
         bcf PORTB,DAT          ; efface le bit de données
         bsf PORTB,LE           ; Validation des données
         bcf PORTB,LE           ; /
         RETURN

;************************************************************
;*                                                          *
;*       138.50 FREQUENCY SYNTHESIS CODING SUB-PROGRAM      *
;*                                                          *
;************************************************************;
;LA FREQUENCE AUTORISEE EST DE 138.5000 MHz (fvco)
;LA FREQUENCE DU QUARTZ EST DE 13.000 MHz (fosc)
;LA FREQUENCE DE REFERENCE EST DE 25.000 KHz (fref)
; fvco = [(P x B) + A] x fosc / R
;        P = 8
;        N = fvco / fref = 5540 = 15A4 h
;        R = fosc / fref = 520 = 208 h
;        B = div(N/P) = 692 = 2B4 h
;        A = N - B x P = 4 = 4h
PLL138
         CLRF BUFF1             ;effacement de BUFF1
         CLRF BUFF2             ;effacement de BUFF2
         CLRF BUFF3             ;effacement de BUFF3
         CLRF PORTB
; 1) FUNCTION AND INITIALIZATION LATCHES
         movlw H'00'            ; \
         movwf BUFF1            ; \
         movlw H'04'            ; chargement des valeurs d'init
         movwf BUFF2            ; dans des variables
         movlw H'10'            ; /
         movwf BUFF3            ; /
         movf BUFF1,0           ; \
         movwf TSFER            ; envoi du premier octet
         call TFRoctet          ; /
         movf BUFF2,0           ; \
         movwf TSFER            ; envoi du deuxieme octet
         call TFRoctet          ; /
         movf BUFF3,0           ; \
         movwf TSFER            ; envoi des 5 derniers bits
         call TFR_5bits         ; /
         bcf PORTB,DAT          ; efface le bit de données
         bsf PORTB,LE           ; Validation des données
         bcf PORTB,LE ;          /
; 2) PROGRAMMABLE REFERENCE DIVIDER (R COUNTER)
         movlw H'00'            ; \
         movwf BUFF1            ; \
         movlw H'41'            ; chargement de la valeur de R
         movwf BUFF2            ; dans des variables
         movlw H'00'            ; /
         movwf BUFF3            ; /
         movf BUFF1,0           ; \
         movwf TSFER            ; envoi du premier octet
         call TFRoctet          ; /
         movf BUFF2,0           ; \
         movwf TSFER            ; envoi du deuxieme octet
```

```
         call TFRoctet          ; /
         movf BUFF3,0           ; \
         movwf TSFER            ; envoi des 5 derniers bits
         call TFR_5bits         ; /
         bcf PORTB,DAT          ; efface le bit de données
         bsf PORTB,LE           ; Validation des données
         bcf PORTB,LE           ; /
; 3) PROGRAMMABLE DIVIDER (N COUNTER = A et B)
         movlw H'0A'            ; \
         movwf BUFF1            ; \
         movlw H'D0'            ; chargement des valeurs de A et B
         movwf BUFF2            ; dans des variables
         movlw H'88'            ; /
         movwf BUFF3            ; /
         movf BUFF1,0           ; \
         movwf TSFER            ; envoi du premier octet
         call TFRoctet          ; /
         movf BUFF2,0           ; \
         movwf TSFER            ; envoi du deuxieme octet
         call TFRoctet          ; /
         movf BUFF3,0           ; \
         movwf TSFER            ; envoi des 5 derniers bits
         call TFR_5bits         ; /
         bcf PORTB,DAT          ; efface le bit de données
         bsf PORTB,LE           ; Validation des données
         bcf PORTB,LE           ; /
         RETURN
; ---------------------------------------------------------------------------------
; Transmission d'un octet au synthetiseur
TFRoctet: MOVLW 8               ; \
         MOVWF COUNT            ; chargement du nombre de bits à transferer
         CLRF STATUS
         RLF TSFER,1            ; rotation pour faire rentrer le msb dans la
                                ;carry
T_DATA1  RLF TSFER,1            ; rotation pour récupérer la carry dans le lsb
         MOVF TSFER,0           ; copie de TSFER dans w
         ANDLW H'01'            ; ET logique pour isoler RB0
         MOVWF PORTB            ; envoi de DAT sur le port

         BSF PORTB,2            ; clock d'horloge pour enregistrer les
                                ;différents bits dans le LMX2603

         BCF PORTB,2            ; /
         DECF COUNT,1           ; decrementation de count
         BNZ T_DATA1            ; test si count est arrivé à 0
         RETURN
; ---------------------------------------------------------------------------------
; Transmission de 5 bits synthetiseur
TFR_5bits: MOVLW 5              ; \
          MOVWF COUNT           ; chargement du nombre de bits à transferer
          CLRF STATUS
          RLF TSFER,1           ; rotation pour faire rentrer le msb dans la
                                ;carry
T_DATA2 RLF TSFER,1             ; rotation pour récupérer la carry
                                ;dans le lsb
         MOVF TSFER,0           ; copie de TSFER dans w
         ANDLW H'01'            ; ET logique pour isoler RB0
         MOVWF PORTB            ; envoi de DAT sur le port
         BSF PORTB,2            ; clock d'horloge pour enregistrer les
                                ;différents bits dans le LMX2603
         BCF PORTB,2            ; /
         DECF COUNT,1           ; decrementation de count
         BNZ T_DATA2            ; test si count est arrivé à 0
         RETURN
         org 0x10
Table:   addwf PCL,1
         retlw 0x84
         retlw 0x94
         retlw 0xB4
         retlw 0xC4
         retlw 0xD4
         retlw 0xE4
         retlw 0xE4
         retlw 0xF4
         retlw 0xF4
         retlw 0xF4
         retlw 0xE4
         retlw 0xE4
         retlw 0xD4
         retlw 0xC4
         retlw 0xB4
         retlw 0x94
```

```
retlw 0x84
retlw 0x74
retlw 0x54
retlw 0x44
retlw 0x34
retlw 0x24
retlw 0x24
retlw 0x14
retlw 0x14
retlw 0x14
retlw 0x24
retlw 0x24
retlw 0x34
retlw 0x44
retlw 0x54
retlw 0x74
END
```