

SWR POWER METER F8KGL

Description et spécifications
V 0.5

Table des matières

| | |
|--|----|
| 1-INTRODUCTION..... | 3 |
| 2-FONCTIONNALITES..... | 4 |
| 2.1-Ligne de mesure..... | 4 |
| 2.1.1-Coupleur directionnel..... | 5 |
| 2.1.2-Détecteur HF..... | 6 |
| 2.1.3-Incertitudes de mesures..... | 6 |
| 2.2-Dispositif de calcul et d'affichage..... | 7 |
| 3-FONCTIONNEMENT..... | 7 |
| 3.1-Mode « test »..... | 7 |
| 3.2-Mode « opérationnel »..... | 8 |
| 3.2.1-Phase « calibration »..... | 8 |
| 3.2.2-Phase « mesure »..... | 8 |
| 4-DEVELOPPEMENT MATERIEL..... | 10 |
| 2.1-Description..... | 10 |
| 2.1.1-LCD..... | 10 |
| 2.1.2-Ligne de mesure..... | 10 |
| 2.1.3-ADC..... | 11 |
| 2.1.3-Alimentation..... | 11 |
| 2.1.3-Boitier..... | 11 |
| 2.2-Schéma fonctionnel..... | 12 |
| 5-DEVELOPPEMENT LOGICIEL..... | 13 |
| 5.1-Généralités..... | 13 |
| 5.2-Outils de développement..... | 13 |
| 5.2.1-Assembleur..... | 13 |
| 5.2.2-Outils de validation..... | 14 |
| 5.3-Environnement de développement..... | 15 |
| 5.4-Spécifications SW..... | 16 |
| 5.4.1-/prj..... | 16 |
| 5.4.1.1-Makefile..... | 16 |
| 5.4.1.2-Main.asm..... | 17 |
| 5.4.1.3-cal.asm..... | 18 |
| 5.4.2/sw/inc..... | 19 |
| 5.4.2.1-lcd.inc..... | 19 |
| 5.4.2.2-eep.inc..... | 19 |
| 5.4.3-/sw/lcd..... | 19 |
| 5.4.3.1-driver.asm :..... | 19 |
| 5.4.3.2-aff.asm :..... | 21 |
| 5.4.4-/sw/calc..... | 26 |
| 5.4.4.1-calc_adc_mV.asm..... | 26 |
| 5.4.4.2-calc_swr.asm..... | 27 |
| 5.4.4.3-calc_power.asm..... | 27 |
| 5.5-/sw/readadc..... | 27 |
| 5.5.1-adc.asm..... | 27 |
| 5.4.6-/sw/EEP/..... | 28 |
| 5.4.6.1-eep.asm..... | 28 |
| 5.5-Plan mémoire..... | 29 |

1-INTRODUCTION

Afin d'optimiser la qualité de ses communications, l'OM doit rechercher le maximum de puissance transmise à l'antenne. Ce maximum est atteint lorsque les impédances sont dites « adaptées ». On obtient ce point de fonctionnement lorsque la puissance réfléchie (voir §2) est minimale (idéalement nulle), ou lorsque le « SWR » (ou ROS), pour « Standing Wave Ratio » (ou Rapport d'Onde Stationnaire) est proche de (idéalement égal à) 1.

Le « SWR Power Meter F8KGL » (ou Wattmètre/ROSmètre F8KGL) est un dispositif permettant de mesurer la puissance transmise à l'antenne, la puissance réfléchie, et le « SWR ». Il donne ainsi la mesure de la qualité de la chaîne de transmission TRX/Antenne.

L'originalité du « SWR Power Meter F8KGL » vient de la conception de la ligne de mesure. Celle-ci a été pensée en câble coaxial rigide en cuivre. De plus, il est prévu de mettre ce dispositif en vente en KIT pour les OM's désireux de s'équiper en moyens de mesure home-made.

Le « SWR Power Meter F8KGL » doit répondre aux besoins suivants :

- mesurer une puissance de 1W à 500W, avec une précision de 10 %
- mesurer une puissance dans les 3 bandes radioamateurs HF, VHF, UHF
- être alimenté par une source extérieure en 13,8V, ou par un pack batterie 4x1,5V
- atténuer le moins possible le signal à transmettre
- afficher le résultat de la mesure sur un écran LCD (puissance en W, sans unité pour le SWR)
- être solide et robuste pour une utilisation en contest
- être prévu pour être vendu en kit

Ce dispositif a été conçu par les OM du club radioamateur « Vauréal Amitié Radio », situé à Vauréal (95), sous l'indicatif F8KGL.

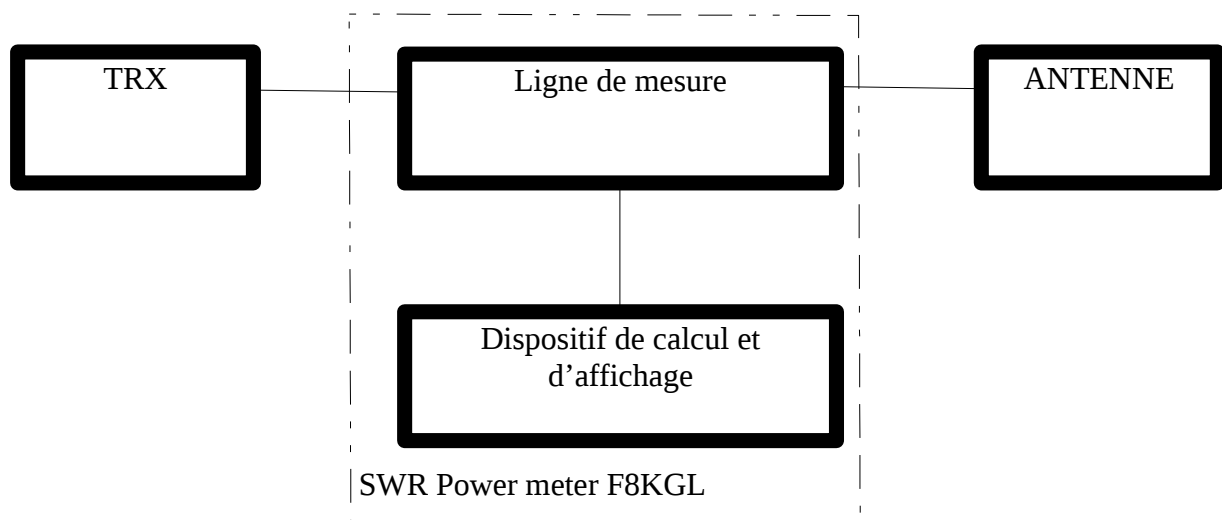
Le projet a été développé par André F0EOS, et Fabrice F4BJH. Portons également à l'attention du lecteur, que l'idée initiale vient de Pierre F1FDD.

2-FONCTIONNALITES

D'un point de vue fonctionnel, le « SWR Power Meter F8KGL » permet :

- de mesurer la puissance transmise de l'émetteur (TRX) vers la charge (Antenne), de mesurer la puissance réfléchie, et de calculer le SWR. Cette fonctionnalité porte le nom de « fonctionnalité de mesure » dans la suite de ce document, et est assurée par la « ligne de mesure ».

- d'afficher le résultat des ces 2 mesures en W, et le résultat du calcul de SWR. Cette fonctionnalité porte le nom de « fonctionnalité de calcul et d'affichage » dans la suite de ce document, et est assurée par le « dispositif de calcul et d'affichage ».



2.1-Ligne de mesure

La mesure de la puissance d'un signal radioélectrique revient à mesurer la tension crête de ce signal (cf SWR_POWER_METER_F8KGL_Etude_du_pont_de_mesure.pdf).

$$Pch = K \times |V|^2$$

De plus, le « SWR » (ou ROS) est donné par cette formule :

$$SWR = \frac{P_{transmise} + P_{réfléchie}}{P_{transmise} - P_{réfléchie}}$$

La mesure de la tension crête du signal transmis (resp. réfléchi) donnera donc, à un facteur près (à déterminer) la mesure de la puissance transmise (resp. réfléchie). La mesure de ces 2 grandeurs permet dès lors de calculer le SWR.

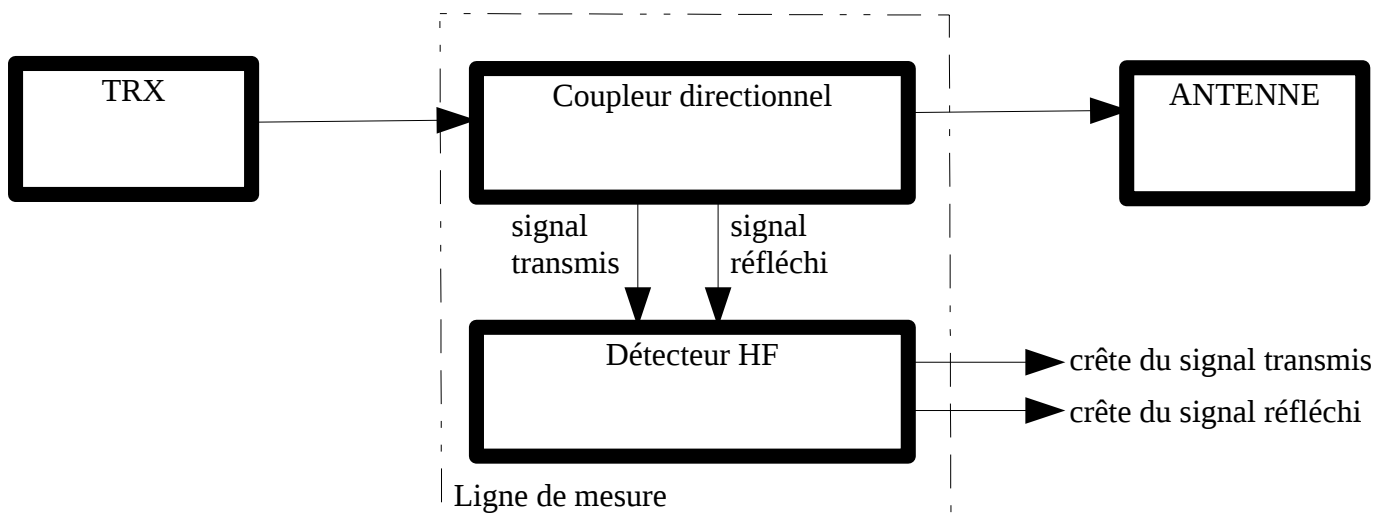
La mesure d'une tension crête d'un signal peut facilement être réalisé à l'aide d'un circuit dit « détecteur d'enveloppe ».

Or les diodes n'acceptent que des signaux relativement faibles. C'est pourquoi un dispositif d'atténuation doit être placé dans la chaîne de mesure, sans que celui-ci n'atténue le signal utile à transmettre à l'antenne.

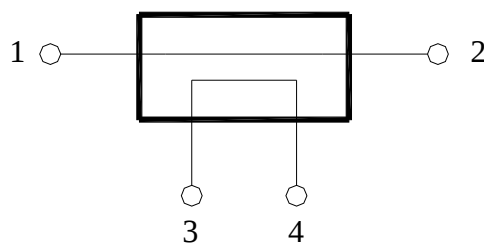
De plus, il faut également prévoir un dispositif capable de séparer le signal transmis, et le signal réfléchi.

D'un point de vue fonctionnel, la ligne de mesure permet :

- séparer le signal transmis du signal réfléchi, et d'adapter le niveau sur chacun de ces 2 signaux en vue d'être redressée par une diode. Cette fonctionnalité porte le nom de « fonctionnalité de couplage », et est assuré par le « coupleur directionnel ».
- d'extraire la crête des signaux transmis et réfléchis. Cette fonctionnalité porte le nom de « fonctionnalité de détection d'enveloppe », et est assuré par le « détecteur d'enveloppe (ou détecteur HF) »



2.1.1-Coupleur directionnel

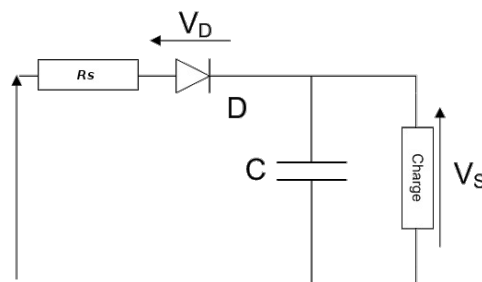


(Source F5ZV)

Un coupleur est constitué d'un tronçon de ligne de même impédance que celle sur laquelle il sera utilisé, par exemple 50 ou 75 ohms. Cette ligne peut être une ligne sur circuit imprimé, un guide d'onde, un câble coaxial... Pour une ligne coaxiale on peut utiliser une petite longueur de câble (de quelques centimètres à quelques décimètres). Parallèlement à l'âme de la ligne est placée à quelques millimètres de celle-ci une ligne de mesure. Le courant qui circule du port P1 au port P2 dans la ligne principale induit un courant dans la ligne de mesure et provoque l'apparition d'une tension entre les deux armatures du condensateur que forment les deux lignes. Dans un coupleur parfait les signaux générés par ces deux phénomènes s'additionnent dans le sens direct et s'annulent dans le sens inverse.

Une des extrémités de la ligne de mesure (port P4) est reliée au blindage de la ligne principale au travers d'une charge purement résistive d'une valeur qui dépend des dimensions de cette ligne de mesure et qui peut être différente de l'impédance de la ligne principale. Lorsqu'un courant circule dans la ligne principale du coupleur, une fraction (un échantillon) de ce courant se retrouve à l'autre extrémité (port P3) de la ligne de mesure.

2.1.2-Détecteur HF



(source Wikipedia)

Un circuit détecteur d'enveloppe est constitué d'une diode en série reliée à une charge constituée d'un condensateur et d'une résistance.

Son signal d'entrée est une fréquence porteuse dont on veut extraire la tension crête. C'est donc un courant alternatif, présentant une tension tantôt positive, tantôt négative.

Quand la tension d'entrée est positive, la diode conduit et le condensateur se charge. Quand la tension d'entrée est négative, la diode se bloque, le condensateur se décharge dans la charge.

Si la résistance présente dans le circuit lors de la charge de la capacité est faible, celle-ci est beaucoup plus rapide que la décharge dans la résistance. Alors, si la constante de temps du circuit résistance-condensateur est correctement choisie, sa tension reste *à peu près* constante entre deux crêtes de la porteuse.

2.1.3-Incertitudes de mesures

2.2-Dispositif de calcul et d'affichage

Dispositif « numérique »

ADC

calcul

affichage

3-FONCTIONNEMENT

A la mise sous tension, le « SWR-POWER METER F8KGL », affiche le message suivant pendant 5s :

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|--|
| S | W | R | - | P | O | W | E | R | | m | e | t | e | r | |
| | | | | F | 8 | K | G | L | | V | n | . | m | | |

Vn.m correspond à la version du logiciel chargée dans la mémoire du microcontrôleur.

Au bout des 5 secondes, le « SWR-POWER METER F8KGL » détermine s'il est en mode « test », ou en mode « opérationnel »

3.1-Mode « test »

Le mode « test » est un mode de fonctionnement de validation du « SWR POWER METER F8KGL ».

Il permettra de valider le fonctionnement des ADC (linéarité, précision, stabilité), du détecteur HF (stabilité, et précision), et la ligne de mesure.

En mode test, il affiche le message suivant :

| | | | | | | | | | | | | | | | |
|---|---|---|--|---|---|---|---|---|---|---|---|---|---|---|---|
| F | W | D | | u | u | u | u | h | - | v | v | v | v | m | V |
| R | E | F | | x | x | x | x | h | - | y | y | y | y | m | V |

« FWD » : chaîne de caractère fixe, indiquant que la ligne 1 du LCD est dédié au port FWD

uuuu : correspond à la valeur de l'ADC du port FWD en hexadécimal

vvvv : correspond à la tension calculée à partir de la valeur de l'ADC par le PIC sur le port FWD en mV

« REF » : chaîne de caractère fixe, indiquant que la ligne 1 du LCD est dédié au port REF

xxxx : correspond à la valeur de l'ADC du port REF en hexadécimal

yyyy : correspond à la tension calculée à partir de la valeur de l'ADC par le PIC sur le port REF en mV

« h » : caractère symbolisant l'unité de la mesure de l'ADC (hexadécimal)

« mV » : chaîne de caractère indiquant l'unité de la mesure de la tension (mV)

La conversion « valeur hexadécimal de l'ADC » vers « tension calculée de l'ADC en mV » se fera à l'aide d'une table de calibration, placée en mémoire flash.

| ADC(hexa) sur 10 bits | Tension en mV | Valeur de la tension en mV stockée flash |
|-----------------------|---------------|--|
| 0x0000 | 0 | 0x0000 |
| 0x0001 | 5 | 0x0005 |
| 0x0002 | 10 | 0x000A |
| ... | ... | ... |
| 0x3FD | 4985 | 0x1379 |
| 0x3FE | 4990 | 0x137E |
| 0x3FF | 4995 | 0x1383 |

3.2-Mode « opérationnel »

Le mode « opérationnel » correspond au mode de fonctionnement conventionnel du « SWR POWER METER F8KGL ». C'est ce mode de fonctionnement qui permet la calibration et la mesure des puissances transmises, réfléchie, et du ROS.

Le mode « opérationnel » se décline en 2 phases :

- phase « calibration »
- phase « mesure »

3.2.1-Phase « calibration »

En phase « calibration », le « SWR-POWER METER F8KGL » affiche :

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|--|---|---|
| | | C | A | L | I | B | R | A | T | I | O | N | | | |
| x | x | W | - | y | y | y | M | h | z | | | | | O | K |

3.2.2-Phase « mesure »

En phase « mesure », le « SWR-POWER METER F8KGL » affiche :

| | | | | | | | | | | | | | | | |
|---|---|---|---|--|---|---|---|---|--|---|---|---|---|---|---|
| F | W | D | | | R | E | F | | | S | W | R | | | |
| a | a | a | W | | b | b | b | W | | c | . | c | c | ! | ! |

« aaa » correspond à la mesure de la puissance transmise en W

« bbb » correspond à la mesure de la puissance réfléchie en W

« c.cc » correspond à la mesure du ROS. 2 points d'exclamation clignotant s'affichent « !! » si le $ROS > 2$

4-DEVELOPPEMENT MATERIEL

2.1-Description

Le coeur du dispositif repose sur l'emploi d'un microcontrôleur PIC 16F88. Son choix a été guidé par ses principales caractéristiques suivantes :

| | |
|--------------------|----------------------|
| | |
| | |
| Taille de la flash | 4K (mot de 14 bits) |
| Taille de l'EEPROM | 256 octets (@0x2100) |
| Taille de la RAM | 368 octets |
| ADC | 10 bits |
| | |
| | |
| | |
| | |

PIC (+sa programmation) 4MHz

LCD

<http://digitaldiy.io/articles/mcu-programming/assembly/55-assembly-example/114-mpasm-tutorial-liquid-crystal-display-lcd#.Wi5383mDO9I>

Ligne de mesure

ADC

Boitier

Alimentation

2.1.1-LCD

2.1.2-Ligne de mesure

HSMS tartempion

2.1.3-ADC

Résolution 10 bits

2.1.3-Alimentation

Pile + connecteur

2.1.3-Boîtier

Dimensions ?

Matière

crosbar

2.2-Schéma fonctionnel

5-DEVELOPPEMENT LOGICIEL

5.1-Généralités

L'architecture hardware étant fondée sur le PIC 16F88 de chez Microchip, le langage de développement du logiciel sera l'assembleur.

Le logiciel sera développé sous Linux (Ubuntu 16.04 ou Debian 8).

3 firmwares seront générés :

-Firmware de test, correspondant au mode de fonctionnement « test » du « SWR Power Meter F8KGL.

Nom : « swr_power_meter_f8kgl-Vn.m.TEST.hex »

-Firmware de calibration, correspondant au mode de fonctionnement « calibration » du « SWR Power Meter F8KGL ».

Nom : « swr_power_meter_f8kgl-Vn.m.CALIBRATION.hex »

-Firmware opérationnel, correspondant au mode de fonctionnement « opérationnel » du « SWR Power Meter F8KGL ».

Nom : « swr_power_meter_f8kgl-Vn.m.hex »

n : correspond à une version majeure du « SWR Power Meter F8KGL ».

m : correspond à une version mineure du « SWR Power Meter F8KGL ».

V0.5 : mode « test » implémenté, validé en simulation et sur prototypes.

V0.6 : phase de calibration implémenté, validé en simulation et sur cible matériel

V1.0 : phase de mesure implémenté, validé en simulation et sur cible matériel.

5.2-Outils de développement

5.2.1-Assembleur

Sous linux, la suite « GPUTILS », permet la compilation d'un projet développé en assembleur pour PIC.

GPUTILS est une collection d'outil pour les microcontrôleurs PIC. Elle inclut :

- Gpasm : compilateur assembleur
- Gplib : compilateur assembleur permettant la génération d'une librairie
- Gplink : éditeur de lien symbolique

Installation :

1. Désinstaller la version courante de la distribution
2. Télécharger la version 1.5.0-1 en suivant ce lien :

<https://sourceforge.net/projects/gputils/files/gputils/1.5.0/gputils-1.5.0-1.tar.gz/download>

3. Installation

```
$ tar -xvzf gputils-1.5.0-1.tar.gz
$ cd gputils-1.5.0-1.tar.gz
$ ./configure
$ make
$ sudo make install
```

5.2.2-Outils de validation

Sous linux, la suite « gpsim » permet la simulation d'un code compilé par GPUTILS

Installation :

1. Désinstaller la version courante de la distribution
2. Télécharger la version 0.30.0 en suivant ce lien :
<https://sourceforge.net/projects/gpsim/files/gpsim/0.30.0/gpsim-0.30.0.tar.gz/download>
3. Installation

```
$ tar -xvzf gpsim-0.30.0.tar.gz
$ cd gpsim-0.30.0.tar.gz
$ ./configure
$ make
$ sudo make install
```

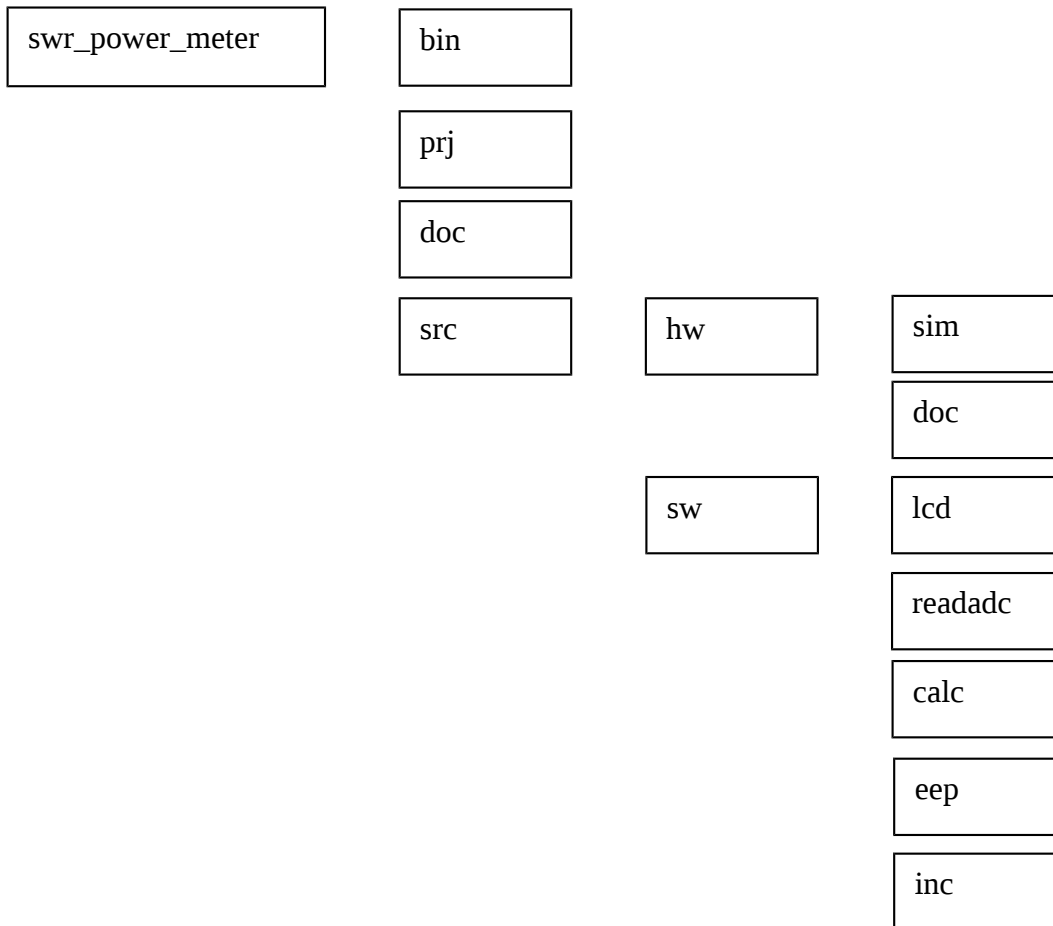
Utilisation :

1.

```
$ gpsim -s nom_du_fichier.cod
```
2. Aller dans File->Open et choisir le fichier .stc
3. Par défaut, la fréquence est fixée à 20MHz. Il faut fixer la fréquence de travail à 4MHz :

```
**gpsim> frequency 4000000
**gpsim> frequency
Clock frequency: 4 MHz.
```

5.3-Environnement de développement



| | |
|----------------------------|---|
| swr_power_meter/bin | Contient l'ensemble des binaires produits : <ul style="list-style-type: none"> • *.a : librairie associée à un composant sw • *.cod : simulation • *.hex : binaire à flasher dans le PIC • *.map : mapping mémoire • *.cof : fichier objet résultat de la compilation • *.lst : ? |
| swr_power_meter/prj | Contient le Makefile du projet, et le point d'entrée sw (main.asm), la table de calibration |
| swr_power_meter/doc | Documentation du projet |
| swr_power_meter/src | Contient les sources du projet |
| swr_power_meter/src/hw | Contient les sources HW du projet |
| swr_power_meter/hw/sim | Contient le fichier netlist pour gpsim |
| swr_power_meter/hw/ | Contient les schéma et le PCB |
| swr_power_meter/sw/lcd | Composant LCD <ul style="list-style-type: none"> • Driver.asm : driver bas niveau du LCD • Aff.asm : routines haut niveau d'affichage des messages • Makefile : make de la librairie LCD |
| swr_power_meter/sw/readadc | Composant ADC |
| swr_power_meter/sw/calc | Composant CALC |

| | |
|------------------------|---------------------------------------|
| swr_power_meter/sw/eep | Composant EEP : fonctionnalité EEPROM |
| Ros_metre/sw/inc | Include |

5.4-Spécifications SW

5.4.1-/prj

5.4.1.1-Makefile

| | |
|-----------------------|--|
| Variables | <p>Nom du projet : swr-power-meter_f8kgl-</p> <p>Processeur : 16F88 (à exporter)</p> <p>Version : Vn.m (à exporter)</p> <p>Nom du firmware de test : <Nom du projet><Version>.TEST.hex</p> <p><i>Nom du firmware de calibration : <Nom du projet><Version>.CALIBRATION.hex</i></p> <p><i>Nom du firmware opérationnel: <Nom du projet><Version>.hex</i></p> <p>Répertoire pour le linker : /usr/share/gptuils/lkr (à exporter)</p> <p>Script du PIC pour le linker : <Répertoire pour le linker><Processeur>.lkr (à exporter)</p> <p>Répertoire des Include : -I../src/sw/inc</p> <p>Adresse de la table de calibration : 0x1000</p> |
| Outils | <p>AS : gpasm (assembleur)</p> <p>LD : gplink (linker)</p> |
| Flags | <p>Flags pour le linker : --map -c -s (génère un fichier .map, génère un fichier objet, spécifier le fichier script pour le linker)</p> <p>flag pour la génération du firmware de test par l'assembleur : TEST</p> <p>flag pour la génération du firmware de calibration par l'assembleur : <i>CALIBRATION (pour le firmware de calibration)</i></p> |
| Composants | Comps : lcd, eep, readadc |
| Librairies | <p>Librairies de TEST : <Nom du composant>.TEST.a</p> <p>Librairies de CALIBRATION: <Nom du composant>.CALIBRATION.a</p> <p>Librairies OPERATIONNELLES : <Nom du composant>.a</p> |
| Règles de compilation | <p>All : règles du firmware de test, <i>calibration et opérationnel</i></p> <p>effacer les fichiers *.a</p> <p>Règle du firmware de test : règles de main.o , règle de cal.o règles de la Librairie de Test</p> <p>linker avec les flags du linker, le script du linker, main.o, les librairies de test, vers le fichier de sortie en ../bin/</p> <p>règle de la Librairie de TEST : faire le make de ../src/sw/<Nom du composant associé à la librairie> avec le flag du firmware de test pour l'assembleur (-DTEST),</p> |

| | |
|--|---|
| | <p><i>règle de la Librairie de CALIBRATION : faire le make de ../src/sw/<Nom du composant associé à la librairie> avec le flag du mode CALIBRATION pour l'assembleur, et le flags -C (génère un fichier objet)</i></p> <p><i>règle de la Librairie de MESURE : faire le make de ../src/sw/<Nom du composant associé à la librairie> avec le flag du mode MESURE pour l'assembleur, et le flags -C (génère un fichier objet)</i></p> <p>Règle de main.o : assembler main.asm en main.o</p> <p>Règle de la table de calibration : assembler cal.asm en cal.o</p> <p>Règle de clean : efface tous les fichiers de ../bin</p> |
|--|---|

```

$ cd prj
//Génération du firmware en mode TEST
$ make rule_test
//Génération du firmware en mode CALIBRATION
$ make rule_calibration
//Génération du firmware en mode MESURE
$ make rule_mesure
//Génération de tous les firmwares
$ make all

```

5.4.1.2-Main.asm

| | |
|--------------------|--|
| Fonctions | Fonction principale, point d'entrée du logiciel |
| Nom | Init |
| Paramètres entrée | |
| Paramètres sorties | |
| Traitements | <ul style="list-style-type: none"> • Initialisation <ul style="list-style-type: none"> ◦ PIC : effectuer l'initialisation du PIC ◦ LCD : Effectuer l'initialisation du LCD (lcd_init) ◦ ADC : Effecteur l'initialisation de l'ADC ◦ • Afficher le message de boot (lcd_affbootmsg) • Tempo de 5s <ul style="list-style-type: none"> ◦ temporisation de 2,5s (tempo_boot) ◦ temporisation de 2,5s (tempo_boot) • Effacer le LCD (lcd_clear) • Positionner le curseur du LCD sur la ligne 1 <p>#Le code ci-dessous est assemblé uniquement dans le firmware de test</p> <ul style="list-style-type: none"> • afficher le message du mode test (lcd_aff_fwd_and_ref) • Dans une boucle infinie |

| | |
|--|---|
| | <ul style="list-style-type: none"> ▪ lire les registres ADCfwd et ADCref (adc_readAN0, adc_readAN1) ▪ afficher la mesure des ADC en mode test (lcd_affadc) ▪ Convertir la mesure des ADC en mV (calc_adcmV) ▪ Affichage de la mesure en tension des ADC en mode test (lcd_affadcmV) <p><i>#Le code ci-dessous est assemblé uniquement dans le firmware de calibration</i></p> <ul style="list-style-type: none"> • Tester la phase (calibration ou mesure) • Si le boîtier est en phase « calibration » <ul style="list-style-type: none"> ◦ afficher le message de calibration (lcd_affcalib) ◦ Dans une boucle infinie <ul style="list-style-type: none"> ▪ <p><i>#Le code ci-dessous est assemblé uniquement dans le firmware opérationnel</i></p> <ul style="list-style-type: none"> • Sinon <ul style="list-style-type: none"> ◦ Dans une boucle infinie : <ul style="list-style-type: none"> ▪ ▪ ▪ ▪ |
|--|---|

| | |
|--------------------|-----------------------------------|
| Fonctions | Temporisation de 2,5 secondes |
| Nom | tempo_boot |
| Paramètres entrée | |
| Paramètres sorties | |
| Traitements | Appeler 10 fois un délai de 250ms |

5.4.1.3-cal.asm

| | | |
|--------------------|--|--------|
| Fonctions | Table de calibration du détecteur HF | |
| Nom | c_calibration_table | |
| Paramètres entrée | | |
| Paramètres sorties | | |
| Traitements | Zone de mémoire dédiée au stockage de la calibration du détecteur HF. Zone en flash à l'adresse défini dans le makefile | |
| | <i>#Le code ci-dessous est assemblé uniquement dans le firmware de test</i> | |
| | 0x0000 | 0x0005 |
| | 0x0001 | 0x0010 |
| | ... | ... |

| | | |
|--|-------|--------|
| | 0x3FE | 0x137E |
| | 0x3FF | 0x1383 |

5.4.2/sw/inc

5.4.2.1-lcd.inc

Contient les define du LCD.

5.4.2.2-eep.inc

Contient le plan mémoire de l'EEPROM

| | | |
|--------------------|-----------------------|-----------------------|
| __EEPROM_START | __SW_VERSION_EEP_ADDR | Version du logiciel |
| __EEPROM_START + 5 | __CALIBRATION_OFFSET | Offset de calibration |

5.4.3-/sw/lcd

5.4.3.1-driver.asm :

| | |
|--------------------|-----------------------|
| Fonction | Initialisation du LCD |
| Nom | lcd_init |
| Paramètres entrée | |
| Paramètres sorties | |
| Traitements | |

| | |
|--------------------|--|
| Fonction | Affichage d'un caractère |
| Nom | lcd_affchar |
| Paramètres entrée | W(1 byte) : contient le caractère à afficher à la position courante du curseur |
| Paramètres sorties | |
| Traitements | |

| | |
|-------------------|---|
| Fonction | Envoi d'une commande au LCD |
| Nom | lcd_sendcmd |
| Paramètres entrée | W(1 byte) : contient la commande 0x28 Set Interface Length 0x10 Turn Off Display 0x01 Clear Display RAM 0x06 Set Cursor Movement 0x0C Turn on Display/Cursor 0x01 Clear display 0xc0 move to 2 nd row, first column |

| | |
|--------------------|--|
| Paramètres sorties | |
| Traitements | |

| | |
|--------------------|--|
| Fonction | Positionner le curseur du LCD |
| Nom | lcd_setposcursor |
| Paramètres entrée | W(1 byte) : contient la position du curseur 0-15 : 1 ^{ère} ligne 16-31 : 2 ^{ème} ligne |
| Paramètres sorties | |
| Traitements | <ol style="list-style-type: none"> 1. Si le curseur doit être positionné sur la première ligne : W = W + 0x80 Si le curseur doit être positionné sur la deuxième ligne : W = W + 0xC0 2. Envoi de la commande au LCD (lcd_sendcmd) |

| | |
|--------------------|---|
| Fonction | Efface le LCD |
| Nom | lcd_clear |
| Paramètres entrée | |
| Paramètres sorties | |
| Traitement | <ol style="list-style-type: none"> 1. W=0x01 2. Envoi de la commande au LCD (lcd_sendcmd) |

| | |
|--------------------|---|
| Fonction | Positionne le curseur sur la 2 ^{ème} ligne |
| Nom | lcd_setposL2 |
| Paramètres entrée | |
| Paramètres sorties | |
| Traitements | <ol style="list-style-type: none"> 1. W=0xC0 2. Envoi de la commande au LCD (lcd_sendcmd) |

| | |
|--------------------|---|
| Fonction | Conversion hexa-ASCII |
| Nom | lcd_convtoascii |
| Paramètres entrée | W (1 quartet) : contient le quartet de poids faible à convertir |
| Paramètres sorties | W (1 byte) : contient l'octet converti |
| Traitements | W=W + 0x30 |

| | |
|-------------------|------------------------------------|
| Fonction | Routines de temporisation et pulse |
| Nom | |
| Paramètres entrée | |

| | |
|--------------------|---|
| Paramètres sorties | |
| Traitements | http://digitaldiy.io/articles/mcu-programming/assembly/55-assembly-example/114-mpasm-tutorial-liquid-crystal-display-lcd#.Wi5383mDO9I |

| | |
|--------------------|---|
| Fonction | Conversion hexa-BCD |
| Nom | lcd_convtobcd |
| Paramètres entrée | v_hexa_to_conv (2 bytes) : 2 octets à convertir en BCD |
| Paramètres sorties | v_bcd (2 bytes) : 2 octets convertis en BCD |
| Traitements | http://www.microchip.com/forums/m322713.aspx v_bcd = R1 v_bcd + 1 = R2 |

5.4.3.2-aff.asm :

| | |
|--------------------|--|
| Fonction | Message de version courante du logiciel |
| Nom | c_swversion |
| Paramètres entrée | |
| Paramètres sorties | |
| Traitements | Zone mémoire (5 bytes) dédiée au stockage de la version du logiciel « Vn.m »,0x00 Cette zone de mémoire est placée au début de l'EEPROM (0x2100). Cette zone mémoire doit se terminer par l'octet 0x00. Cette zone mémoire est remplie par le compilateur. |

| | |
|--------------------|--|
| Fonctions | Message de boot ligne 1 du LCD |
| Nom | bootmsgL1 |
| Paramètres entrée | v_charpos : position du caractère à retourner |
| Paramètres sorties | W (1 byte) : contient le caractère ou 0x00 si pas de caractère |
| Traitements | Zone mémoire dédiée au stockage du message de boot (ligne 1 du LCD) contenant la chaîne suivante : « SWR-POWER meter » <ul style="list-style-type: none"> Additionner le pointeur de programme avec v_charpos Retourner le caractère contenu en mémoire à cette position dans W Fin de chaîne = retourner 0x00 dans W |

| | |
|--------------------|---|
| Fonctions | Message de boot ligne 2 du LCD |
| Nom | bootmsgL2 |
| Paramètres entrée | v_charpos : position du caractère à retourner |
| Paramètres sorties | W (1 byte) : contient le caractère ou 0x00 si pas de caractère |
| Traitements | <p>Zone mémoire dédiée au stockage du message de boot (ligne 2 du LCD) contenant la chaîne suivante : « F8KGL »</p> <ul style="list-style-type: none"> • Additionner le pointeur de programme avec v_charpos • Retourner le caractère contenu en mémoire à cette position dans W • Fin de chaîne = retourner 0x00 dans W |

| | |
|--------------------|--|
| Fonctions | <p>Message du mode test L1 du LCD</p> <p>#Le code ci-dessous est assemblé uniquement dans le firmware de test</p> |
| Nom | testmsgL1 |
| Paramètres entrée | v_charpos : position du caractère à retourner |
| Paramètres sorties | W (1 byte) : contient le caractère ou 0x00 si pas de caractère |
| Traitements | <p>Zone mémoire dédiée au stockage du message du mode test (ligne 1 du LCD) contenant la chaîne suivante : « FWD » avec un espace à la fin</p> <ul style="list-style-type: none"> • Additionner le pointeur de programme avec v_charpos • Retourner le caractère contenu en mémoire à cette position dans W • Fin de chaîne = retourner 0x00 dans W |

| | |
|--------------------|---|
| Fonctions | <p>Message du mode test L2 du LCD</p> <p>#Le code ci-dessous est assemblé uniquement dans le firmware de test</p> |
| Nom | testmsgL2 |
| Paramètres entrée | v_charpos : position du caractère à retourner |
| Paramètres sorties | W (1 byte) : contient le caractère ou 0x00 si pas de caractère |
| Traitements | <p>Zone mémoire dédiée au stockage du message de boot (ligne 2 du LCD) contenant la chaîne suivante : « REF » avec un espace à la fin</p> <ul style="list-style-type: none"> • Additionner le pointeur de programme avec v_charpos • Retourner le caractère contenu en mémoire à cette position dans W • Fin de chaîne = retourner 0x00 dans W |

| | | | | | | | | | | | | | | | | |
|--------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|
| Fonction | Affichage du message de boot | | | | | | | | | | | | | | | |
| Nom | lcd_affboot | | | | | | | | | | | | | | | |
| Paramètres entrée | -bootmsgL1 : zone mémoire (15 bytes) contenant le message de boot ligne 1 -bootmsgL2 : zone mémoire (5 bytes) contenant le message de boot ligne 2 -c_swversion : zone EEPROM (5bytes) contenant la version courante du logicielle | | | | | | | | | | | | | | | |
| Paramètres sorties | S | W | R | - | P | O | W | E | R | | m | e | t | e | r | |
| | | | | F | 8 | K | G | L | | V | 0 | . | 5 | | | |
| Traitements | <div>1. v_charpos = 0x00</div> <div>2. Afficher le message de boot ligne 1</div> <div>Tant que W≠0</div> <div>o Récupérer 1 caractère du message de boot ligne 1 (bootmsgL1) dans W</div> <div>o Afficher 1 caractère sur le LCD (lcd_affchar)</div> <div>o Incrementer v_charpos</div> <div>3. Positionner le curseur sur la ligne 2, 4ème case :</div> <div>o W=0x13</div> <div>o Positionner le curseur du LCD (lcd_setposcursor)</div> <div>4. v_charpos = 0x00</div> <div>5. Afficher le message de boot ligne 2</div> <div>Tant que W≠0</div> <div>o Récupérer 1 caractère du message de boot ligne 2 (bootmsgL2)</div> <div>o Afficher 1 caractère sur le LCD (lcd_affchar)</div> <div>o Incrementer v_charpos</div> <div>6. Positionner le curseur sur la ligne 2, 10ème case :</div> <div>o W=0x1C</div> <div>o Positionner le curseur du LCD (lcd_setposcursor)</div> <div>7. v_charpos = 0x00</div> <div>8. afficher la version</div> <div>Tant que W≠0</div> <div>o Récupérer le caractère en EEPROM</div> <div>W ← v_charpos</div> <div>EEADDR ← W</div> <div>Lire un octet en EEPROM (eep_readbyte)</div> <div>o Afficher 1 caractère sur le LCD (lcd_affchar)</div> <div>o Incrementer v_charpos</div> <div>9. FIN</div> | | | | | | | | | | | | | | | |

| | |
|-------------------|--|
| Fonction | Affichage du message du mode test #Le code ci-dessous est assemblé uniquement dans le firmware de test |
| Nom | lcd_aff_fwd_and_ref |
| Paramètres entrée | -testmsgL1 : zone mémoire (7bytes) contenant le message de calibration L1 -testmsgL2 : zone mémoire (7bytes) contenant le message de calibration L2 |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------------|---|--|--|---|---|---|---|---|---|---|---|--|--|--|--|--|--|--|--|--|--|--|---|---|---|---|---|---|--|--|--|--|--|--|--|
| Fonction | Affichage de la mesure des ADC en mode test #Le code ci-dessous est assemblé uniquement dans le firmware de test | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Nom | lcd_affadc | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Paramètres entrée | v_adcfwd (2bytes) : résultat de l'ADC AN0 sur 10 bits v_adcref (2bytes) : résultat de l'ADC AN1 sur 10 bits | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Paramètres sorties | <table><tr><td></td><td></td><td></td><td></td><td>u</td><td>u</td><td>u</td><td>u</td><td>h</td><td>-</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td>x</td><td>x</td><td>x</td><td>x</td><td>h</td><td>-</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table> | | | | | u | u | u | u | h | - | | | | | | | | | | | | x | x | x | x | h | - | | | | | | | |
| | | | | u | u | u | u | h | - | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | x | x | x | x | h | - | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Traitements | 1.positionner le curseur sur la ligne 1, 5ème case 2.W=v_adcfwd 3.Afficher un octet en hexa (lcd_affhexa) 4.W =v_adcfwd +1 5.Afficher un octet en hexa (lcd_affhexa) 6. W='h' 7.Afficher 1 caractère sur le LCD (lcd_affchar) 8. W='-' 9.Afficher 1 caractère sur le LCD (lcd_affchar) 10.positionner le curseur sur la ligne 2, 5ème case 11.W=v_adcref 12.Afficher un octet en hexa (lcd_affhexa) 13.W =v_adcref +1 14.Afficher un octet en hexa (lcd_affhexa) 15. W='h' 16.Afficher 1 caractère sur le LCD (lcd_affchar) 17. W='-' 18.Afficher 1 caractère sur le LCD (lcd_affchar) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------------|---|--|--|--|--|--|--|--|--|--|---|---|---|---|---|---|---|---|--|--|--|--|--|--|--|--|--|--|--|---|---|---|---|---|---|
| Fonction | Affichage de la mesure en tension des ADC en mode test #Le code ci-dessous est assemblé uniquement dans le firmware de test | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Nom | lcd_affadcmV | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Paramètres entrée | v_adcfwd_mV (2bytes) : résultat de l'ADC en mV compris entre [0;5000] v_adcref_mV (2bytes) : résultat de l'ADC en mV compris entre [0;5000] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Paramètres sorties | <table><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>v</td><td>v</td><td>v</td><td>v</td><td>m</td><td>V</td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>y</td><td>y</td><td>y</td><td>y</td><td>m</td><td>V</td></tr></table> | | | | | | | | | | | | v | v | v | v | m | V | | | | | | | | | | | | y | y | y | y | m | V |
| | | | | | | | | | | | v | v | v | v | m | V | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | y | y | y | y | m | V | | | | | | | | | | | | | | | | | | | |
| Traitements | 1.positionner le curseur sur la ligne 1, 11ème case 2.v_hexa_to_conv = v_adcfwd_mV 3.v_hexa_to_conv +1 = v_adcfwd_mV +1 4. Conversion hexa-BCD (lcd_convtobcd) 5. W = v_bcd 6. Affichage d'un octet en hexa (lcd_affhexa) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | |
|--|--|
| | 7. $W = v_bcd + 1$ 8. positionner le curseur sur la ligne 2, 11ème case 2. $v_hexa_to_conv = v_adcref_mV$ 3. $v_hexa_to_conv + 1 = v_adcref_mV + 1$ 4. Conversion hexa-BCD (lcd_convtobcd) 5. $W = v_bcd$ 6. Affichage d'un octet en hexa (lcd_affhexa) 7. $W = v_bcd + 1$ |
|--|--|

| | |
|---------------------------|--|
| <i>Fonction</i> | <i>Affichage de la puissance du port FWD</i> |
| <i>Nom</i> | <i>lcd_affpfwd</i> |
| <i>Paramètres entrée</i> | <i>v_pfw</i> |
| <i>Paramètres sorties</i> | |
| <i>Traitements</i> | |

| | |
|---------------------------|--|
| <i>Fonction</i> | <i>Affichage de la puissance du port REF</i> |
| <i>Nom</i> | <i>lcd_affpref</i> |
| <i>Paramètres entrée</i> | <i>v_pref</i> |
| <i>Paramètres sorties</i> | |
| <i>Traitements</i> | |

| | |
|---------------------------|-------------------------|
| <i>Fonction</i> | <i>Affichage du SWR</i> |
| <i>Nom</i> | <i>lcd_affpref</i> |
| <i>Paramètres entrée</i> | <i>v_p_ref</i> |
| <i>Paramètres sorties</i> | |
| <i>Traitements</i> | |

5.4.4-/sw/calc

5.4.4.1-calc_adc_mV.asm

| | |
|--------------------------|--|
| <i>Fonction</i> | Convertir la mesure des ADC en mV #Le code ci-dessous est assemblé uniquement dans le firmware de test |
| <i>Nom</i> | calc_adcmV |
| <i>Paramètres entrée</i> | v_adcfwd (2bytes) : résultat de l'ADC AN0 sur 10 bits v_adcref (2bytes) : résultat de l'ADC AN1 sur 10 bits |

| | |
|--------------------|---|
| Paramètres sorties | v_adcfwd_mV (2bytes) : résultat de l'ADC en mV en hexa v_adcref_mV (2bytes) : résultat de l'ADC en mV en hexa |
| Traitements | 1. v_flh_offset_addr = v_adcfwd 2. v_flh_offset_addr + 1 = v_adcfwd + 1 3. Lecture d'un octet en flash (flh_cal_table_read) 4. v_adcfwd_mV = v_flh_read 5. v_adcfwd_mV + 1 = v_flh_read + 1 6. v_flh_offset_addr = v_adcfwd 7. v_flh_offset_addr + 1 = v_adcfwd + 1 8. Lecture d'un octet en flash (flh_cal_table_read) 9. v_adcfwd_mV = v_flh_read 10. v_adcfwd_mV + 1 = v_flh_read + 1 |

5.4.4.2-calc_swr.asm

| | |
|----------------------|---|
| • Paramètres entrée | • P_FWD • P_REF |
| • Paramètres sorties | • SWR |
| • Traitements | • $SWR = \frac{ADC_{fwd} + ADC_{ref}}{ADC_{fwd} - ADC_{ref}}$ |

5.4.4.3-calc_power.asm

| | |
|--------------------|---|
| Paramètres entrée | ADCfwd, ADCref table de calibration stockée en mémoire |
| Paramètres sorties | P_FWD P_REF |
| Traitements | Lire la valeurs dans les registres ADC récupérer la valeur correspondante dans la table de calibration |

5.5.5-/sw/readadc

5.5.5.1-adc.asm

| | |
|--------------------|---|
| Fonction | Lire le résultat de la conversion A/N AN0 |
| Nom | adc_readAN0 |
| Paramètres entrée | |
| Paramètres sorties | -v_adcfwd (2bytes) : résultat de l'ADC sur 10 bits |
| Traitements | 1. Selectionner le canal à échantillonner (AN0) ADCON0(CHS2) = b'0' ADCON0(CHS1) = b'0' ADCON0(CHS0) = b'0' 2. Mise en service du convertisseur ADCON(ADON) = b'1' |

| | |
|--|---|
| | 3. Tempo de 20us 4.Lancer la phase de conversion ADCON0(G0)=b'1' 5. Tant ADCON0(G0)≠b'0' 6.v_adcfwd = ADRESH v_adcfwd(+1) = ADRESL |
|--|---|

| | |
|--------------------|---|
| Fonction | Lire le résultat de la conversion A/N AN1 |
| Nom | adc_readAN1 |
| Paramètres entrée | |
| Paramètres sorties | -v_adcref (2bytes) : résultat de l'ADC sur 10 bits |
| Traitements | 1. Selectionner le canal à échantillonner (AN1) ADCON0(CHS2) = b'0' ADCON0(CHS1) = b'0' ADCON0(CHS0) =b'1' 2. Mise en service du convertisseur ADCON(ADON) = b'1' 3. Tempo de 20us 4.Lancer la phase de conversion ADCON0(G0)=b'1' 5. Tant ADCON0(G0)≠b'0' 6.v_adcref = ADRESH v_adcref(+1) = ADRESL |

5.4.6-/sw/EEP/

5.4.6.1-eep.asm

| | |
|--------------------|--|
| Fonction | Lecture d'un octet en EEPROM |
| Nom | eep_readbyte |
| Paramètres entrée | -W : contient l'adresse à lire en EEPROM |
| Paramètres sorties | -W : contient l'octet lu en EEPROM |
| Traitements | 1. Choisir la bank de EEADRH 2. EADDR = W 3. Choisir la bank de EECON1 4. EECON1(EEPGD) = b'0' 5. EECON1(RD) = b'1' 6. Choisir la bank de EEDATA 7. W ← EEDATA |

| | |
|-------------------|--|
| Fonction | Lecture d'un octet en flash |
| Nom | flh_cal_table_read |
| Paramètres entrée | v_flh_offset_addr (2 bytes) : contient l'offset du mot à lire en flash à partir de |

| | |
|--------------------|---|
| | 0x1000 |
| Paramètres sorties | v_flh_read (2 bytes) : contient le mot de 14 bits lu |
| Traitements | 1.Choisir la bank de EEADRH 2. W = v_flh_offset_addr 3. W = W +0x10 3. EEADRH = W 4. W = v_flh_offset_addr +1 5. EEADR = W 6. Choisir la bank de EECON1 7. EECON(EEPGD) = b'1' 8. EECON(RD) = b'1' 9. Pas d'instruction 10. Pas d'instruction 11. Choisir la bank de EEDATA 12. W = EEDATA 13. v_flh_read+1=W 14. W = EEDATAH 15. v_flh_read=W |

5.5-Plan mémoire

| section | Adresse de début – adresse de fin | Taille (octets) | Plan mémoire |
|---------|--------------------------------------|--------------------|--|
| .code | 0x0000-0x1FFF | 8K | 0x0000-0x0FFF : programme 0x1000-0x13FF : table de calibration 0x1400-0x0x1FFF : ne pas utiliser |
| .code | 0x2100-0x21FF | 256 | 0x2100-0x2103 : version 0x2104-0xAAAA : <i>offset calibration</i> |
| .data | 0x20-0x7F (bank 0) | 96 | 0x20-0x7F : variables |