SWR POWER METER F8KGL

Description et spécifications

V 0.6

F0EOS-F4BJH-21/12/2017-Vauréal Amitié Radio

Table des matières

[1-INTRODUCTION 4](#__RefHeading__4103_246836813)

[2-FONCTIONNALITES 5](#__RefHeading__4105_246836813)

[2.1-Ligne de mesure 5](#__RefHeading__4107_246836813)

[2.1.1-Coupleur directionnel 6](#__RefHeading__4109_246836813)

[2.1.2-Détecteur HF 7](#__RefHeading__4111_246836813)

[2.1.3-Incertitudes de mesures 7](#__RefHeading__4113_246836813)

[2.2-Dispositif de calcul et d'affichage 8](#__RefHeading__4115_246836813)

[2.2.1-ADC 8](#__RefHeading__6258_63224887)

[2.2.2-Mise en forme 9](#__RefHeading__6260_63224887)

[3-FONCTIONNEMENT 11](#__RefHeading__4117_246836813)

[3.1-Phase de calibration 11](#__RefHeading__6262_63224887)

[3.2-Phase de mesure 11](#__RefHeading__4125_246836813)

[4-DEVELOPPEMENT MATERIEL 12](#__RefHeading__4127_246836813)

[4.1-Description 12](#__RefHeading__4129_246836813)

[4.1.1-LCD 12](#__RefHeading__4131_246836813)

[4.1.2-Ligne de mesure 14](#__RefHeading__4133_246836813)

[4.1.3-ADC 15](#__RefHeading__4135_246836813)

[4.1.4-Alimentation 15](#__RefHeading__4137_246836813)

[4.1.5-Connecteurs 16](#__RefHeading__6264_63224887)

[4.1.6-Boitier 16](#__RefHeading__4139_246836813)

[4.2-Schéma fonctionnel 17](#__RefHeading__4141_246836813)

[4.3-PCB 17](#__RefHeading__5935_1110828432)

[5-DEVELOPPEMENT LOGICIEL 19](#__RefHeading__4143_246836813)

[5.1-Généralités 19](#__RefHeading__4145_246836813)

[5.2-Outils de développement 19](#__RefHeading__4147_246836813)

[5.2.1-Assembleur 19](#__RefHeading__4149_246836813)

[5.2.2-Outils de validation 20](#__RefHeading__4151_246836813)

[5.2.3-Outils de contrôle de version 20](#__RefHeading__5937_1110828432)

[5.3-Arborescence de développement 21](#__RefHeading__4153_246836813)

[5.4-Mode « test » 22](#__RefHeading__6266_63224887)

[5.4-Spécifications SW 23](#__RefHeading__4155_246836813)

[5.4.1-/prj 23](#__RefHeading__4157_246836813)

[5.4.1.1-Makefile 23](#__RefHeading__4159_246836813)

[5.4.1.2-Main.asm 25](#__RefHeading__4161_246836813)

[5.4.2/sw/inc 27](#__RefHeading__4163_246836813)

[5.4.2.1-lcd.inc 27](#__RefHeading__4165_246836813)

[5.4.2.2-eep.inc 27](#__RefHeading__4167_246836813)

[5.4.3-/sw/lcd 27](#__RefHeading__4169_246836813)

[5.4.3.1-driver.asm : 27](#__RefHeading__4171_246836813)

[5.4.3.2-aff.asm : 29](#__RefHeading__4173_246836813)

[5.4.4-/sw/calc 33](#__RefHeading__4175_246836813)

[5.4.4.1-calc.asm 33](#__RefHeading__4545_144346426)

[5.5.5-/sw/readadc 33](#__RefHeading__4181_246836813)

[5.5.5.1-adc\_pic.asm 33](#__RefHeading__4183_246836813)

[5.5.5.2-adc\_maxim.asm 34](#__RefHeading__6268_63224887)

[5.5.5.3-MAX11100.asm 34](#__RefHeading__6270_63224887)

[5.5.5.4-MAX4624.asm 35](#__RefHeading__6272_63224887)

[5.4.6-/sw/eep/ 35](#__RefHeading__4185_246836813)

[5.4.6.1-driver.asm 35](#__RefHeading__4187_246836813)

[5.4.7-/sw/flh/ 35](#__RefHeading__5939_1110828432)

[5.4.7.1-driver.asm 35](#__RefHeading__5941_1110828432)

[5.4.8-/sw/data/ 36](#__RefHeading__5943_1110828432)

[5.4.8.1-swversion.asm 36](#__RefHeading__5945_1110828432)

[5.4.8.2-adc\_theoric\_caltable.asm 36](#__RefHeading__5947_1110828432)

[5.4.8.2-lcdmsg.asm 36](#__RefHeading__5949_1110828432)

[5.5-Plan mémoire 38](#__RefHeading__4760_144346426)

1-INTRODUCTION

Afin d'optimiser la qualité de ses communications, l'OM cherche à transmettre le maximum de puissance 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 (RG402). 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, et le SWR sans unité)

-être solide et robuste pour une utilisation en contest

-être vendable sous la forme de 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 du 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 ».

TRX

Ligne de mesure

ANTENNE

Dispositif de calcul et d’affichage

SWR Power meter F8KGL

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).

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

La mesure de la tension crête du signal transmis (resp. réfléchie) 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 de les atténuer. 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) »

.

Coupleur directionnel

TRX

ANTENNE

signal

transmis

signal

réfléchi

Détecteur HF

crête du signal transmis

crête du signal réfléchi

Ligne de mesure

2.1.1-Coupleur directionnel



1

2

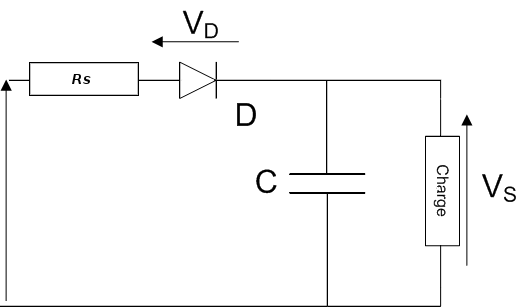
3

4

(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

Le dispositif de calcul et d'affichage est un dispositif numérique. Sa fonction principale est d'afficher sur un écran LCD (2 lignes de 16 caractères) le résultat de la mesure.

ADC

crête du signal transmis

crête du signal réfléchi

*Tension mesurée*

*numérique*

Mise en forme (calcul et affichage)

écran

*Commandes*

Dispositif de calcul et d'affichage

D'un point de vue fonctionnel, le dispositif de calcul et d'affichage permet :

-Mesurer les tensions crête du signal transmis et réfléchis, et les mettre dans un format binaire (au sens informatique du terme). Cette fonction est assurée par un « ADC » (Analog to Digital Converter)

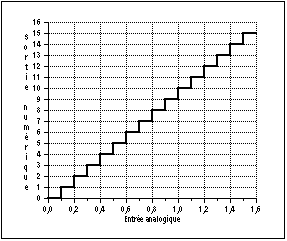
-Convertir la mesure numérique en W, et calculer le SWR

-Envoyer ces mesures de puissance et le calcul du SWR au LCD, dans le standard qui lui est propre. Ces 2 fonctions sont assurées par le dispositif de « mise en forme (calcul et affichage)

2.2.1-ADC

Un ADC (Analog to Digital Converter) est un dispositif électronique permettant de convertir une grandeur analogique (par exemple, une tension) en une valeur numérique, qui prend la forme d'un nombre binaire.

Cette valeur numérique peut être codée sur plusieurs bits, et est proportionnelle à la grandeur analogique d'entrée.



(source : [http://www.bedwani.ch/electro/ch21/index.htm#C01](http://www.bedwani.ch/electro/ch21/index.htm" \l "C01))

La sortie numérique décrite dans ce graphique est affichée en base 10.



2.2.2-Mise en forme

L'ADC ayant fourni une valeur numérique au format binaire, elle peut être facilement traitée par une dispositif informatique.

Le traitement que ce dispositif effectue s’appelle un programme informatique. Le programme est une succession d'instruction, dont le but est de produire un résultat en fonction de la valeur de données d'entrée.

Ces données d'entrées sont stockées dans une mémoire au format binaire. Les résultats produits dépendent des instructions programmées dans une autre mémoire. Dans le cas du « SWR Power Meter F8KGL », le résultat final est un affichage de la puissance transmise et réfléchie.

Le dispositif de mise en forme choisie est un microcontrôleur PIC, pour sa simplicité d'utilisation, sa faible consommation en énergie, sa fiabilité, et sa robustesse. De plus, la simplicité (apparente en première lecture) du programme à développer pour ce dispositif, appelle tout naturellement un composant simple et efficace.

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 phase de « calibration », ou en phase de « mesure ».

3.1-Phase de calibration

La phase de « calibration» correspond au mode de fonctionnement qui permet la calibration de l'appareil de mesure.

*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-Phase de mesure

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

*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

4.1-Description

Le dispositif de calcul et d'affichage repose sur l'emploi d'un microcontrôleur PIC 18F1320. Son choix a été guidé par les principales caractéristiques suivantes :

|  |  |
| --- | --- |
| Taille de la flash | 8K (@0x0) |
| Taille de l'EEPROM | 256 octets (@0xf00000) |
| Taille de la RAM | 256 octets (@0x80) – seuls 127 octets sont exploités |
| ADC | 10 bits |
| Nb de canaux ADC | 7 |
| Oscillateur interne | 31 kHz, 125 kHz, 250 kHz, 500 kHz, 1 MHz, 2 MHz, 4 MHz, 8 MHz |
| Alimentation | Comprise entre 4,2V et 5,5V |
| GPIO | 2x8 GPIO disponibles (multiplexés avec les entrées analogiques) |

Les versions matérielles suivent les versions logicielles. Le prototype HW sera fondé sur le tag V0.6.

*4.1.1-LCD*

*(Source Wikipedia)*

*Le LCD utilisé dispose d'un contrôleur* HD44780. C'est un contrôleur standard permettant de piloter un dispositif d'affichage par cristaux liquides.

Un module HD44780 comporte 16 bornes (dont les 2 dernières sont optionnelles si l'écran piloté ne dispose pas d'un rétroéclairage).

En « mode 4 bits », on n’utilise que les broches D4 à D7 (les broches D0 à D3 doivent être connectées à la masse.

L’octet de données est envoyé (ou lu) en 2 fois :

* d’abord les 4 bits de poids fort, par une première validation sur la broche E.
* puis les 4 bits de poids faible, par une seconde validation sur la broche E

|  |  |  |  |
| --- | --- | --- | --- |
| Borne | Symbole | Type | Fonction |
| 1 | Vss ou V0 | Alim | Masse 0V |
| 2 | Vcc ou Vdd | Alim | Alimentation générale 5V |
| 3 | Vee | Alim | Alimentation du panneau LCD (Contraste des caractères : Vee = 0 → Caractères invisibles, Vee = Vcc → Contraste maximum ) |
| 4 | RS | Entrée | RS = 1 → Sélection du registre de données  RS = 0 et R/W = 0 → Sélection du registre d’instruction  RS = 0 et R/W = 1 → Sélection du drapeau BUSY et du compteur d’adresse |
| 5 | R/W | Entrée | R/W = 0 → Mode écriture  R/W = 1 → Mode lecture |
| 6 | E | Entrée | Entrée de validation  Les entrées RS et R/W sont lues sur le front montant, et le bus de données est lu sur le front descendant. |
| 7 | D0 | Entrée/Sortie | Bus de données, bit n°0 (LSB) |
| 8 | D1 | Entrée/Sortie | Bus de données, bit n°1 |
| 9 | D2 | Entrée/Sortie | Bus de données, bit n°2 |
| 10 | D3 | Entrée/Sortie | Bus de données, bit n°3 |
| 11 | D4 | Entrée/Sortie | Bus de données, bit n°4 |
| 12 | D5 | Entrée/Sortie | Bus de données, bit n°5 |
| 13 | D6 | Entrée/Sortie | Bus de données, bit n°6 |
| 14 | D7 | Entrée/Sortie | Bus de données, bit n°7 (MSB) |
| 15 | A | Alim | Anode du système de rétro-éclairage (à alimenter en 5V à travers une résistance de 50 à 100Ω pour limiter le courant à 100mA) |
| 16 | K | Alim | Cathode du système de rétro-éclairage (masse) |

*4.1.2-Ligne de mesure*

*Dans la ligne de mesure, le coupleur directionnel fera l'objet d'une étude particulière. En effet, un comparatif entre ces plusieurs solutions sera envisagées :*

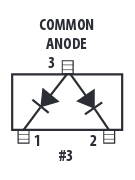
*-coupleur en cable coaxial rigie*

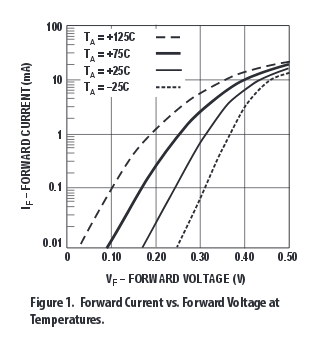
*-ligne imprimée sur circuit*

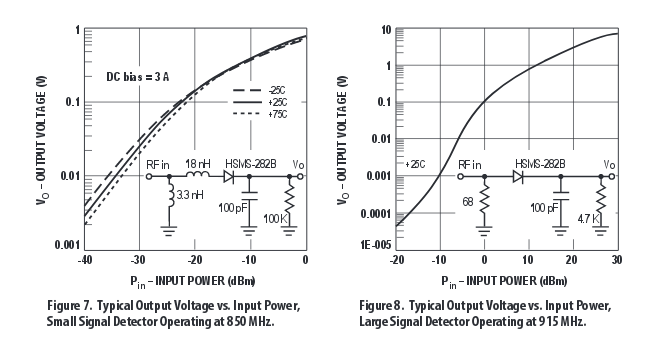
*-coupleur du commerce, issu d'une épave de TOS-mètre*

*-coupleur du commerce issu d'un Wattmètre du monde professionnel (type Bird, ou power sensor 4B250 de Thruline)*

*En revanche, dans la ligne de mesure, le détecteur d'enveloppe est défini par une facilité d’approvisionnement en diode Schottky : HSMS-2822 (marquage C2E, boitier SOT-23).*

**

**

**

*4.1.3-ADC*

La résolution de l'ADC du PIC choisi est de 10 bits. Le tableau ci-dessous donne les valeurs minimales et maximales (tenion, valeur hexdécimale).



La valeur minimale détectable par l'ADC du PIC est de 4,9mV.

La période d’échantillonnage n'a pas une contrainte forte pour le « SWR Power Meter F8KGL » car les puissances crêtes varient peu à l'échelle du temps du PIC (4MHz).. Le logiciel s'attachera à faire du polling sur les valeurs retournées par les ADC, aussi vite que possible.

La technologie de numérisation utilisée par Microchip est l'échantilloneur-bloqueur.

*4.1.4-Alimentation*

Le « SWR Power Meter F8KGL » possède une double alimentation :

* 4 piles de 1,5V type LR6
* prise alimentation 13,8V

*Description des connecteurs :*

4.1.5-Connecteurs

Les connecteurs utilisés pour le LCD, les détecteurs HF, le programmateur de PIC seront des connecteurs HE10.

*4.1.6-Boitier*

*Dimensions ?*

*Matière*

*crosbar*

4.2-Schéma fonctionnel

Vdd (5V)

7805

Connecteur

alim

18F1320

Connecteur LCD

D[7:4]

4

Connecteur

Vfwd

Vref

RS

RW

E

Strap calibration/mesure

Connecteur

Programmateur PIC

4.3-PCB

Les outils de CAO utilisés seront la suite logicielle gratuite KiCAD.

Tous les composants seront en CMS.

Le circuit imprimé sera en epoxy FR4, d'épaisseur 1,6mm, simple face.

(Source Wikipedia)

|  |  |
| --- | --- |
| Propriété | Valeur |
| Constante diélectrique | 4,70 max, 4,35 à 500 MHz, 4,34 à 1 GHz |
| [Facteur de pertes](https://fr.wikipedia.org/w/index.php?title=Facteur_de_pertes&action=edit&redlink=1) | 0,02 à 1 MHz, 0,01 à 1 GHz |
| [Rigidité diélectrique](https://fr.wikipedia.org/wiki/Rigidité_diélectrique) | 20 kV/mm |
| Résistivité de surface (min) | 2 × 105 |
| Résistivité volumique (min) | 8×107 MΩ·cm |
| Épaisseur typique | 1,25 à 2,54 mm |
| Rigidité | 17 GPa |
| Coefficient de dilatation thermique | 11 ppm/K (dans la direction des fibres) |
| Coefficient de dilatation thermique | 15 ppm/K (dans la direction perpendiculaire aux fibres) |
| [Conductivité thermique](https://fr.wikipedia.org/wiki/Conductivité_thermique) | 0,3 W·m-1·K-1 (dans la direction des fibres) |
| [Capacité calorifique](https://fr.wikipedia.org/wiki/Capacité_thermique) | 800 J·kg-1·K-1 |
| [Densité](https://fr.wikipedia.org/wiki/Densité) | de 1,80 à 1,90 kg·L-1[1](https://fr.wikipedia.org/wiki/FR-4" \l "cite_note-1) |

5-DEVELOPPEMENT LOGICIEL

5.1-Généralités

L’architecture hardware étant fondée sur le PIC 18F1320 de chez Microchip, le langage de développement du logiciel sera l’assembleur.

Le logiciel sera développé sous Linux (Debian 8).

3 firmwares seront généres :

-Firmware de test, correspondant au mode de fonctionnement « test » du « SWR Power Meter F8KGL. Ce firmware est généré en position le flag de compilation TEST.

Nom : « swr\_power\_meter\_f8kgl-Vn.m.TEST.hex »

-Firmware de calibration, correspondant à la fonctionnalité de la phase de « calibration » seule du « SWR Power Meter F8KGL ». Ce firmware est généré en position le flag de compilation CALIBRATION.

Nom : « swr\_power\_meter\_f8kgl-Vn.m.CALIBRATION.hex »

-Firmware opérationnel, correspondant au mode de fonctionnement opérationnel du « SWR Power Meter F8KGL » telle que décrite dans le §3.

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 : V0.5 avec nouvel ADC (MAX11100, MAX4624), et LCD 4 lignes 16 caractères

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

V2.0 : 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 microcontroleurs PIC. Elle inclut :

* Gpasm : compilateur assembleur
* Gplib : compilateur assembleur permettant la génération d’une librairie
* Gplink : éditeur de lien symboliquez

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>

1. Installation

$ tar –xvzf gpsim-0.30.0.tar.gz

$ cd gpsim-0.30.0.tar.gz

$ ./configure

$make

$sudo make install

Utilisation :

$ gpsim –s nom\_du\_fichier.cod

1. Aller dans File->Open et choisir le fichier .stc
2. 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.

5.2.3-Outils de contrôle de version

Les versions de logiciel, schémas, BOM et circuit imprimés sont contrôlées par un serveur SVN à l'adresse :

https://svn.riouxsvn.com/swr\_power\_meter

5.3-Arborescence de développement

bin

swr\_power\_meter

prj

doc

sim

hw

src

doc

sw

calc

data

eep

flh

inc

lcd

readadc

|  |  |
| --- | --- |
| swr\_power\_meter/bin | Contient l’ensemble des binaires produits :   * \*.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) |
| 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 (schéma, PCB) |
| swr\_power\_meter/src/hw/sim | Contient le fichier netlist pour gpsim |
| swr\_power\_meter/src/hw/doc | Contient les doc HW des composants |
| swr\_power\_meter/sw/lcd | Composant LCD   * 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 :  Driver.asm : driver bas niveau |
| swr\_power\_meter/sw/flh | Composant d’accès à la flash du PIC :  Driver.asm : driver bas niveau |
| swr\_power\_meter/sw/data | Composant data : contient les données : table de calibration, version logicielle, constantes |
| swr\_power\_meter/sw/inc | Include |

5.4-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 » : caracère 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. Elle portera le nom de « table de calibration théorique de l’ADC ».

|  |  |  |
| --- | --- | --- |
| 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 |

5.4-Spécifications SW

5.4.1-/prj

5.4.1.1-Makefile

|  |  |
| --- | --- |
| Variables | Nom du projet : swr-power-meter\_f8kgl-  Processeur : 18F1320 (à exporter)  Version : Vn.m (à exporter)  Nom du firmware de test :<Nom du projet><Version>.TEST.hex  *Nom du firmware de calibration : <Nom du projet><Version>.CALIBRATION.hex*  *Nom du firmware opérationnel: <Nom du projet><Version>.hex*  Répertoire pour le linker : /usr/share/gptuils/lkr (à exporter)  Script du PIC pour le linker : <Répertoire pour le linker><Processeur>.lkr (à exporter)  Répertoire des Include : -I../src/sw/inc |
| Outils | AS : gpasm (assembleur)  LD : gplink (linker) |
| Flags | Flags pour le linker : --map -c -s (génère un fichier .map, génère un fichier objet, spécifier le fichier script pour le linker)  Flags pour l'assembleur : -c (génère un fichier objet) -D<Version du firmware »  flag pour la génération du firmware de test par l’assembleur : TEST  flag pour la génération du firmware de calibration par l’assembleur : *CALIBRATION*  Si ADC\_PIC=Yes, alors le flag de choix de l'ADC HW vaut HW\_ADC\_PIC. Sinon, ilest indéfini |
| Composants | Composants : lcd, eep, readadc, calc, flh |
| Fichiers sources | Fichiers sources communs à tous les firmware : main.asm, ../src/sw/data/swversion.asm  Fichiers sources du firmware de test : ../src/sw/data/adc\_theoric\_cal.asm |
| Objets | objets communs en mode test : [pour chacun des fichiers sources communs à tous les firmwares : <nom du fichier source sans l’extension .asm>.TEST.o]  objets du firmware de test [pour chacun des fichiers sources du firmware de test : <nom du fichier source sans l’extension .asm>.o],  objets de tests : les objets communs en mode test, les objets du firmware de test |
| Librairies | librairies de test : [pour chacun des composants : libtest<Nom du composant>.a]  librairies de calibration : [pour chacun des composants : libcalib<Nom du composant>.a  Librairies opérationnelles : [pour chacun des composants : <Nom de chaque composant>.a] |
| Règles de compilation | All :  -applique les règles du firmware de test, *calibration et opérationnel*  -rule\_operationnel  -appliquer les règles du firmware opérationnel  *rule\_calibration :*  *-appliquer les règles du firmware de calibration*  rule\_test :  -appliquer les règles du firmware de test  *Règle du firmware opérationnelles :*  *-appliquer les règles des objets opérationnels, les règles de la librairie opérationnelles*  *-linker*  *-effacer*  *-effacer*  Règle du firmware de test :  -appliquer les règles des objets de test, les règles de la librairie de test  -linker avec les flags du linker, avec le script du linker, les objets de test, les librairies de test, vers le firmware de test en ../bin/<Nom du firmware de test>  -effacer les fichier objets de tests, les librairies de test  -effacer tous les fichiers \*.lst  *règle de la librairie opérationnelle*  *-faire le make, avec le flag -C, avec le flag du choix de l'ADC HW, de la librairie opérationnelle de ../src/sw/<Nom du composant associé à la librairie>*    règle de la librairie de test :  -faire le make, avec le flag -C, avec le flag du choix de l'ADC HW, de la librairie de test de ../src/sw/<Nom du composant associé à la librairie>  Règle d’un objet de test commun issu des sources communes à tous les firmware  assembler avec les flags de l'assembleur, le flag du firmware de test, pour le processeur, avec le répertoire des Includes, le fichier source commun à tous les firmware associé à l’objet, en un objet commun à tous les firmware en mode test  *Règle d'un objet du firmware opérationnel*  *assembler avec les flags de l'assembleur, pour le processeur, avec le répertoire des Includes, le fichier source spécifique au mode de test, en un objet du firmware opérationnel*  Règle d'un objet du firmware de test  assembler avec les flags de l'assembleur, le flag du firmware de test, pour le processeur, avec le répertoire des Includes, le fichier source spécifique au mode de test, en un objet du firmware de test  Règle de clean : efface tous les fichiers de ../bin |

$ cd prj

//Génération du firmware en mode TEST

$ make rule\_test

//Génération du firmware en mode CALIBRATION

$ make rule\_calibration

//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 | * CONFIG :   OSC = INTIO2 ; Internal Osc with FOSC/4 -RA6 and RA7 = I/O  FSCM = OFF ; Fail-Safe Clock Monitor disabled  IESO = OFF ; Internal External Switch Over mode disabled  PWRT = OFF ; Power up timer disabled  BOR = OFF ; Brown out reset disabled  WDT = OFF ; Watch dog timer off  MCLRE = OFF ; MCLRE off (pin available for input)  LVP = OFF ; Low voltage programming disabled  DEBUG = OFF ; Background debugger off CONFIG   * Initialisation PIC   OSCCON = 4MHz  #Si FLAG=PIC\_ADC  TRISA = RA0, RA1 input  TRISB = PortB Outputs  #SINON  #FIN  INTCON = disable all interrupts  INTCON2 = disable all interrupts - PORTB pull-up disable  INTCON3 = disable all interrupts  IPR1, IPR2 = clear, no priority is used  PIE1, PIE2 = Individualy disable interrupts  RCON = Disable priority levels  EECON1 = clear EEPROM control register  WDTCON = stop watchdog  CCP1CON = Capture/Compare/PWM of   * Initialisation ADC * Initialisation LCD * Afficher le message de boot (f\_lcd\_affboot) * Tempo de 5s   + temporisation de 2,5s (f\_tempo\_boot)   + temporisation de 2,5s (f\_tempo\_boot) * Effacer le LCD (f\_lcd\_clear) * Positionner le curseur du LCD sur la ligne 1 (f\_lcd\_setposcursor)   #Le code ci-dessous est assemblé uniquement dans le firmware de test   * afficher le message du mode test (lcd\_aff\_fwd\_and\_ref) * Dans une boucle infinie   + - lire les registres ADCfwd et ADCref (f\_adc\_readAN0, f\_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)   #FIN  *#Le code ci-dessous n'est pas assemblé dans le firmware de test*  *#Le code ci-dessous n'est pas assemblé dans le firmware de calibration*   * *Tester la phase (calibration ou mesure)* * *Si le boîtier est en phase « calibration »*   *#FIN*   * + *afficher le message de calibration (lcd\_affcalib)*   + *Dans une boucle infinie*   *#Le code ci-dessous n'est pas assemblé pour le firmware de calibration*   * *Sinon*   + *Dans une boucle infinie  :*   *#FIN* |

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

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* | *\_\_OFFSET\_CAL\_TABLE* | *Offset de calibration* |

5.4.3-/sw/lcd

5.4.3.1-driver.asm :

|  |  |
| --- | --- |
| 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](http://digitaldiy.io/articles/mcu-programming/assembly/55-assembly-example/114-mpasm-tutorial-liquid-crystal-display-lcd" \l ".Wi5383mDO9I) |

|  |  |
| --- | --- |
| Fonction | Envoi d’une commande au LCD |
| Nom | \_f\_lcd\_sendcmd |
| Paramètres entrée | W(1 byte) : contient la commande |
| Paramètres sorties |  |
| Traitements |  |

|  |  |
| --- | --- |
| Fonction | Positionner le curseur du LCD |
| Nom | f\_lcd\_setposcursor |
| Paramètres entrée | W(1 byte) : contient la position du curseur  0-15 : 1ère ligne  16-31 : 2ème ligne |
| Paramètres sorties |  |
| Traitements | 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   1. Envoi de la commande au LCD (lcd\_sendcmd) |

|  |  |
| --- | --- |
| Fonction | Efface le LCD |
| Nom | f\_lcd\_clear |
| Paramètres entrée |  |
| Paramètres sorties |  |
| Traitement | 1. W=0x01 2. Envoi de la commande au LCD (lcd\_sendcmd) |

|  |  |
| --- | --- |
| Fonction | Positionne le curseur sur la 2ème ligne |
| Nom | f\_lcd\_setposL2 |
| Paramètres entrée |  |
| Paramètres sorties |  |
| Traitements | 1. W=0xC0 2. Envoi de la commande au LCD (lcd\_sendcmd) |

|  |  |
| --- | --- |
| Fonction | Conversion hexa-ASCII |
| Nom | f\_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 | Conversion hexa-BCD |
| Nom | f\_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> |

|  |  |
| --- | --- |
| Fonction | Initialisation du LCD |
| Nom | f\_lcd\_init |
| Paramètres entrée |  |
| Paramètres sorties |  |
| Traitements | * Configurer le LCD en mode 4 bits * effacer le RAM du LCD * allumer le curseur * allumer le LCD * effacer le LCD |

|  |  |
| --- | --- |
| Fonction | Affichage d’un caractère |
| Nom | f\_lcd\_affchar |
| Paramètres entrée | W(1 byte) : contient le caractère à afficher à la position courante du curseur |
| Paramètres sorties |  |
| Traitements |  |

5.4.3.2-aff.asm :

|  |  |
| --- | --- |
| Fonction | Affichage du message de boot |
| Nom | f\_lcd\_affboot |
| Paramètres entrée | -c\_bootmsgL1 : zone mémoire (15 bytes) contenant le message de boot ligne 1  -c\_bootmsgL2 : zone mémoire (5 bytes) contenant le message de boot ligne 2  -c\_data\_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 | 1. v\_charpos  = 0x00 2. Afficher le message de boot ligne 1   Tant que W≠0   * + Récupérer 1 caractère du message de boot ligne 1 (c\_bootmsgL1) dans W   + Afficher 1 caractère sur le LCD (f\_lcd\_affchar)   + Incrementer v\_charpos  1. Positionner le curseur sur la ligne 2, 4ème case :    * W=0x10    * Positionner le curseur du LCD (f\_lcd\_setposcursor) 2. v\_charpos  = 0x00 3. Afficher le message de boot ligne 2   Tant que W≠0   * + Récupérer 1 caractère du message de boot ligne 2 (c\_bootmsgL2)   + Afficher 1 caractère sur le LCD (f\_lcd\_affchar)   + Incrementer v\_charpos  1. Positionner le curseur sur la ligne 2, 10ème case :    * W=0x1C    * Positionner le curseur du LCD (f\_lcd\_setposcursor) 2. v\_charpos  = 0x00 3. afficher la version   Tant que W≠0   * + W ← v\_charpos   + Récupérer le caractère en EEPROM (f\_eep\_readbyte) dans W   + Afficher 1 caractère sur le LCD (f\_lcd\_affchar)   + Incrementer v\_charpos  1. FIN |

|  |  |
| --- | --- |
| Fonction | Affichage du message du mode test  #Le code ci-dessous est assemblé uniquement dans le firmware de test |
| Nom | f\_lcd\_aff\_fwd\_and\_ref |
| Paramètres entrée | -c\_testmsgL1 : zone mémoire (7bytes) contenant le message de calibration L1  -c\_testmsgL2 : zone mémoire (7bytes) contenant le message de calibration L2 |
| Paramètres sorties | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | F | W | D |  |  |  |  |  |  |  |  |  |  |  |  |  | | R | E | F |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| Traitements | 1. v\_charpos  = 0x00 2. Afficher le message de calibration ligne 1   Tant que W≠0   * + Récupérer 1 caractère du message de calibration ligne 1 (c\_testmsgL1) dans W   + Afficher 1 caractère sur le LCD (f\_lcd\_affchar)   + Incrementer v\_charpos  1. Positionner le curseur sur la ligne 2 :    * W=0x10    * Positionner le curseur du LCD (f\_lcd\_setposcursor) 2. v\_charpos  = 0x00 3. Afficher le message de calibration ligne 2   Tant que W≠0   * + Récupérer 1 caractère du message de calibration ligne 2 (c\_testmsgL2)   + Afficher 1 caractère sur le LCD (f\_lcd\_affchar)   + Incrementer v\_charpos |

|  |  |
| --- | --- |
| Fonction | Affichage d'1 octet en hexa sur le LCD |
| Nom | f\_lcd\_affhexa |
| Paramètres entrée | W : contient l'octet en hexa à afficher |
| Paramètres sorties |  |
| Traitements | 1. v\_tmp = W  2. swapper les quartets de v\_tmp, et mettre le résultat dans W  3. Appliquer un masque sur les bits de poids fort sur W  4. Convertir le quartet de poids faible en ASCII (f\_lcd\_convtoascii)  5. Afficher 1 caractère sur le LCD (f\_lcd\_affchar)  6. W=v\_temp&0F  7. Convertir le quartet de poids faible en ASCII (f\_lcd\_convtoascii)  8. Afficher 1 caractère sur le LCD (f\_lcd\_affchar) |

|  |  |
| --- | --- |
| Fonction | Affichage de la mesure des ADC en mode test  #Le code ci-dessous est assemblé uniquement dans le firmware de test |
| Nom | f\_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 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  | 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 (f\_lcd\_affhexa)  4.W =v\_adcfwd +1  5.Afficher un octet en hexa (f\_lcd\_affhexa)  6. W=’h’  7.Afficher 1 caractère sur le LCD (f\_lcd\_affchar)  8. W=’-’  9.Afficher 1 caractère sur le LCD (f\_lcd\_affchar)  10.positionner le curseur sur la ligne 2, 5ème case  11.W=v\_adcref  12.Afficher un octet en hexa (f\_lcd\_affhexa)  13.W =v\_adcref +1  14.Afficher un octet en hexa (f\_lcd\_affhexa)  15. W=’h’  16.Afficher 1 caractère sur le LCD (f\_lcd\_affchar)  17. W=’-’  18.Afficher 1 caractère sur le LCD (f\_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 | f\_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 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  |  | 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 (f\_lcd\_convtobcd) ;  5. W = v\_bcd  6. Affichage d'un octet en hexa (\_f\_lcd\_affhexa)  7. W = v\_bcd+1  8. Affichage d'un octet en hexa (\_f\_lcd\_affhexa)  9. W = v\_bcd+2  10. Affichage d'un octet en hexa (\_f\_lcd\_affhexa)  11. Afficher "mV"  12.positionner le curseur sur la ligne 2, 11ème case  13.v\_hexa\_to\_conv = v\_adcref\_mV  14.v\_hexa\_to\_conv +1 = v\_adcref\_mV +1  15. Conversion hexa-BCD (f\_lcd\_convtobcd) ;  16. W = v\_bcd  17. Affichage d'un octet en hexa (\_f\_lcd\_affhexa)  18. W = v\_bcd+1  19. Affichage d'un octet en hexa (\_f\_lcd\_affhexa)  20. W = v\_bcd+2  21. Afficher "mV" |

|  |  |
| --- | --- |
| *Fonction* | *Affichage du message de calibration*  *#le code ci-dessous n'est pas assemblé dans le firmware de test* |
| *Nom* | *f\_lcd\_affcalib* |
| *Paramètres entrée* |  |
| *Paramètres sorties* |  |
| *Traitements* |  |

|  |  |
| --- | --- |
| *Fonction* | *Affichage de la puissance du port FWD* |
| *Nom* | *f\_lcd\_affpfwd* |
| *Paramètres entrée* | *v\_pfwd* |
| *Paramètres sorties* |  |
| *Traitements* |  |

|  |  |
| --- | --- |
| *Fonction* | *Affichage de la puissance du port REF* |
| *Nom* | *f\_lcd\_affpref* |
| *Paramètres entrée* | *v\_pref* |
| *Paramètres sorties* |  |
| *Traitements* |  |

|  |  |
| --- | --- |
| *Fonction* | *Affichage du SWR* |
| *Nom* | *f\_lcd\_affpref* |
| *Paramètres entrée* | *v\_p\_ref* |
| *Paramètres sorties* |  |
| *Traitements* |  |

5.4.4-/sw/calc

*5.4.4.1-calc.asm*

|  |  |
| --- | --- |
| Fonction | Convertir la mesure des ADC en mV  #Le code ci-dessous est assemblé uniquement dans le firmware de test |
| Nom | f\_calc\_adcmV |
| 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 | 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  v\_flh\_offset\_addr + 1 = v\_adcfwd + 1  2. v\_flh\_offset\_addr = 2\*v\_flh\_offset\_addr et propager la retenue  3. Lecture d'un octet en flash (f\_flh\_readword)  4.v\_adcfwd\_mV = v\_flh\_read  5.v\_adcfwd\_mV +1 = v\_flh\_read+1  6. v\_flh\_offset\_addr = v\_adcref  v\_flh\_offset\_addr + 1= v\_adcref + 1  7. v\_flh\_offset\_addr = 2\*v\_flh\_offset\_addr et propager la retenue  8. Lecture d'un octet en flash (f\_flh\_readword)  9.v\_adcref\_mV = v\_flh\_read  10.v\_adcref\_mV +1 = v\_flh\_read+1 |

|  |  |
| --- | --- |
| Fonction |  |
| Nom |  |

|  |  |
| --- | --- |
| * *Paramètres entrée* | * *P\_FWD* * *P\_REF* |
| * *Paramètres sorties* | * *SWR* |
| * *Traitements* |  |

5.5.5-/sw/readadc

*Mettre le flag ADC\_PIC dans le makefile*

5.5.5.1-adc.asm

|  |  |
| --- | --- |
| Fonction | Initialisation des ADC |
| Nom | f\_adc\_init |
| Paramètres entrée | v\_conf\_adc\_hw |
| Paramètres sorties |  |
| Traitements | # Si flag ADC\_PIC  appeler f\_adc\_pic\_init  #SINON  appeler f\_adc\_maxim\_init |

|  |  |
| --- | --- |
| Fonction | Initialisation de l'ADC FWD |
| Nom | f\_adc\_read\_fwd |
| Paramètres entrée |  |
| Paramètres sorties |  |
| Traitements | # Si flag ADC\_PIC  appeler f\_adc\_pic\_readAN0  #SINON  appeler f\_adc\_maxim\_read\_fwd |

|  |  |
| --- | --- |
| Fonction | Initialisation de l'ADC REF |
| Nom | f\_adc\_read\_ref |
| Paramètres entrée |  |
| Paramètres sorties |  |
| Traitements | # Si flag ADC\_PIC  appeler f\_adc\_pic\_readAN1  #SINON  appeler f\_adc\_maxim\_read\_ref |

*5.5.5.1-adc\_pic.asm*

|  |  |
| --- | --- |
| Fonction | Initialisation des ADC |
| Nom | f\_adc\_pic\_init |
| Paramètres entrée |  |
| Paramètres sorties |  |
| Traitements | * ADCON0[VCFG] : VREF+=VDD, VREF-=VSS * ADCON1 : ;RA0-RA1 analog channel * ADCON2 : ADFM = right justified – ACQT=16Tad – ADCS = Fosc/16 |

|  |  |
| --- | --- |
| Fonction | Lire le résultat de la conversion A/N AN0 |
| Nom | f\_adc\_pic\_readAN0 |
| Paramètres entrée |  |
| Paramètres sorties | -v\_adcfwd (2bytes) : résultat de l'ADC sur 10 bits |
| Traitements | 1. Selectionner le canal à échantilloner (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 | f\_adc\_readAN1 |
| Paramètres entrée |  |
| Paramètres sorties | -v\_adcref (2bytes) : résultat de l'ADC sur 10 bits |
| Traitements | 1. Selectionner le canal à échantilloner (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 |

5.5.5.2-adc\_maxim.asm

Driver MAX11100 ET MAX4624

5.4.6-/sw/eep/

*5.4.6.1-driver.asm*

|  |  |
| --- | --- |
| Fonction | Lecture d'un octet en EEPROM |
| Nom | f\_eep\_readbyte |
| Paramètres entrée | -W : contient l’offset à partir de \_\_EEPROM\_START de l'adresse à lire en EEPROM |
| Paramètres sorties | -W : contient l'octet lu en EEPROM |
| Traitements | * EADDR = W * EECON1(EEPGD) = b'0’ * EECON1(RD) = b’1’ * W ← EEDATA |

5.4.7-/sw/flh/

*5.4.7.1-driver.asm*

|  |  |
| --- | --- |
| Fonction | Lecture d'un octet en flash |
| Nom | f\_flh\_readword |
| Paramètres entrée | v\_flh\_offset\_addr (2 bytes) : contient l'offset du mot à lire en flash à partir du début de la table |
| Paramètres sorties | v\_flh\_read (2 bytes) : contient le mot de 16 bits lu |
| Traitements | 1. Mettre le poids faible de l'offset dans W  2. Ajouter à W l'adresse absolue de la tahle  3. Propager la retenue dans le poids fort de l'offset  4. Placer l'adresse de poids faible dans TBLPTRL  5.Placer l'adresse de poids fort dans TBLPTRH  6. Mettre 0x00 dans TBLPTRU  7.Vérifier le non dépassement de la table, sinon renvoyer la valeur max de la table  8.Lire la table, et incrémenter le pointeur  9. Transférer le contenu dans v\_flh\_read+1  10.Lire la table, et incrémenter le pointeur  11. Transférer le contenu dans v\_flh\_read |

5.4.8-/sw/data/

*5.4.8.1-swversion.asm*

|  |  |
| --- | --- |
| Fonction | Message de version courante du logiciel |
| Nom | N/A |
| Paramètres entrée | N/A |
| Paramètres sorties | N/A |
| 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 au moment de l’assemblage. |

*5.4.8.2-adc\_theoric\_caltable.asm*

|  |  |
| --- | --- |
| Fonctions | Table de calibration théorique de l’ADC |
| Nom | c\_data\_adc\_theoric\_caltable |
| 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  #Le code ci-dessous est assemblé uniquement dans le firmware de test  #cette table est la conversion brute d'une valeur hexa en mV (§3.1)   |  |  | | --- | --- | | 0x0000 | 0x0000 | | 0x0001 | 0x0005 | | ... | ... | | 0x3FE | 0x137E | | 0x3FF | 0x1383 | |

*5.4.8.2-lcdmsg.asm*

|  |  |
| --- | --- |
| Fonctions | Message de boot ligne 1 du LCD |
| Nom | c\_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 »   * 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 | c\_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 | Zone mémoire dédiée au stockage du message de boot (ligne 2 du LCD) contenant la chaîne suivante :  « F8KGL »   * 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 du mode test L1 du LCD  #Le code ci-dessous est assemblé uniquement dans le firmware de test | | Nom | c\_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 | 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   * 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 du mode test L2 du LCD  #Le code ci-dessous est assemblé uniquement dans le firmware de test |
| Nom | c\_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 | Zone mémoire dédiée au stockage du message de boot (ligne 2 du LCD) contenant la chaîne suivante :  « REF » avec un espace à la fin   * 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 |

5.5-Plan mémoire

|  |  |  |  |
| --- | --- | --- | --- |
| section | Adresse de début – adresse de fin | Taille (octets) | Plan mémoire |
| .code | 0x0000-0x1FFF | 8K | 0x0000-0x13FF : programme + table de calibration |
| .s\_eep | 0xf00000-0xF000FF | 256 | 0x2100-0x2103 : version  0x2104-*0xAAAA*: *offset calibration* |
| .data | 0x80-0xFF (bank 0) | 127 | 0x80-0xFF : variables |