SWR POWER METER F8KGL

Implémentation

V 0.7

F0EOS-F4BJH-11/05/20-Vauréal Amitié Radio

Table des matières

1-INTRODUCTION

L'objectif du présent document est de présenter les choix et solutions retenues pour la réalisation du SWR POWER METER F8KGL, en conformité avec les spécifications retenues (SWR_POWER_METER_F8KGL_specification_V0.7.odt).

1.1-Spécifications

Pour rappel, le SWR POWER METER F8KGL doit répondre aux spécifications suivantes :

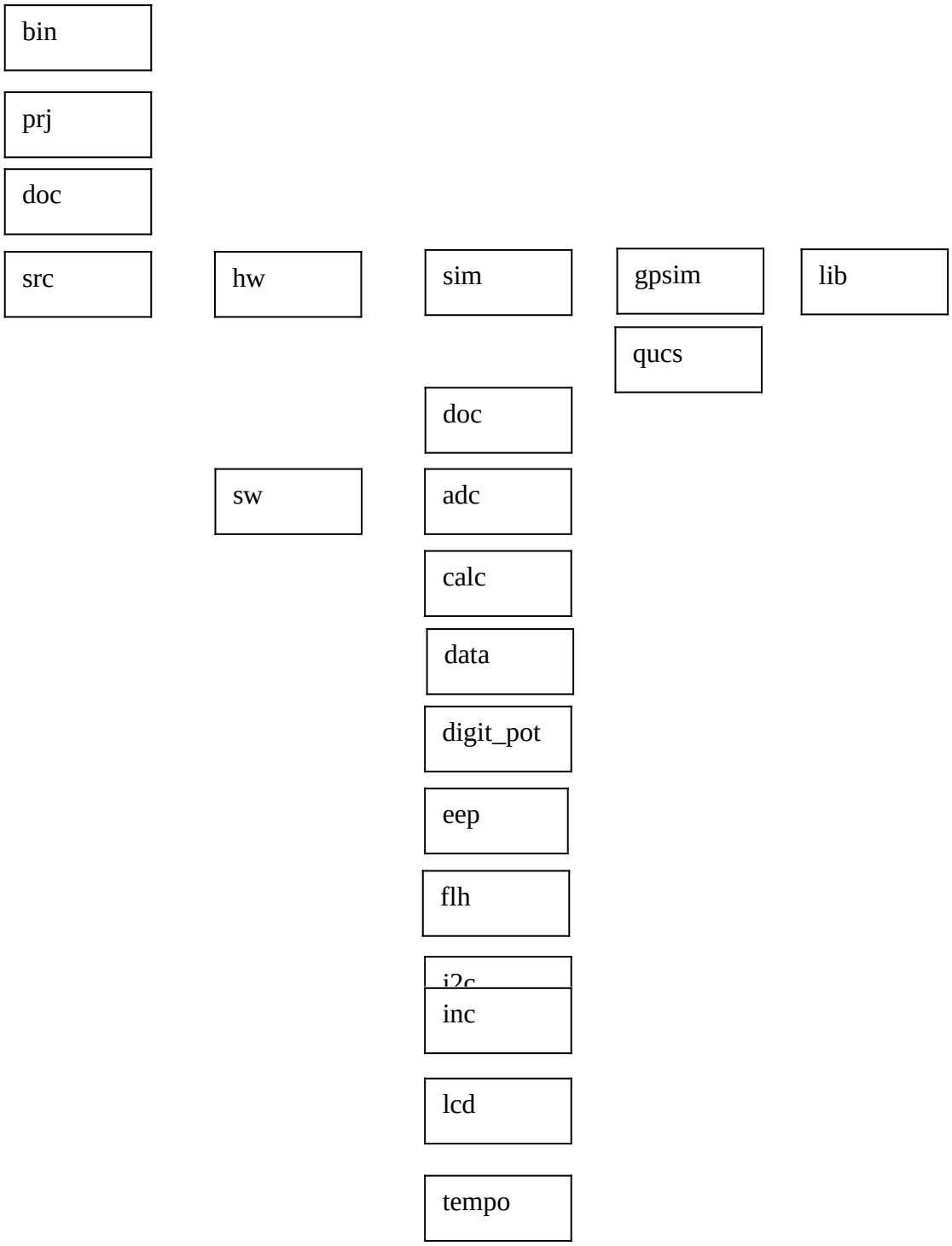
	min	typ	max	Unité
Paramètres radio				
Fréquence de fonctionnement	0		500	MHz
Puissance admissible	1		500	W
Impédance		50		Ω
Pertes en ligne insérées sous 50Ω			1	dB
Alimentation				
Alimentation externe	12	13,8	15	V
Alimentation pack pile	4,5		5,5	V
Autonomie sur pack pile[(1)]	24			h
Consommation (alimentation externe ou pack pile)			100	mA
Mesures				
Précision de la mesure	±10			%
ROS	1,1		∞	
Mécanique				
Dimensions	155x80x100			mm
Poids			1	kg
Connecteurs	N ou PL			

[(1)]Avec un pack pile d'une capacité d'au moins 2500mAh

2-IMPLEMENTATION LOGICIELLES

2.1-Généralités

2.2-Arborescence de développement



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, simulations)
swr_power_meter/src/hw/sim/gpsim	Contient le fichier netlist pour gpsim
swr_power_meter/src/hw/sim/gpsim/lib	Contient la librairie et les modules pour gpsim
swr_power_meter/src/hw/doc	Contient les doc HW des composants
swr_power_meter/src/sw/adc	Composant logiciel ADC • routines de haut niveau de lecture de l'ADC
swr_power_meter/src/sw/calc	Composant logiciel CALC • routines de calcul du ROS
swr_power_meter/src/sw/data	Composant logiciel data : contient les données constantes : table de calibration, version logicielle, constantes
swr_power_meter/src/sw/digit_pot	Composant logiciel DIGIT_POT • routines de haut niveau de contrôle du gain du module AOP
swr_power_meter/src/sw/eep	Composant EEP : Driver.asm : driver bas niveau
swr_power_meter/src/sw/flh	Composant d'accès à la flash du PIC : Driver.asm : driver bas niveau
swr_power_meter/sw/i2c	Composant logiciel i2C: driver bas niveau du bus i2c
swr_power_meter/src/sw/inc	Include
swr_power_meter/src/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/src/sw/tempo	Routines de temporisation

2.3-Firmware

2.3.1-Mode « test »

2.3.2-Mode « calibration »

tbd

2.3.3-Mode « opérationnel »

tbd

2.4-Implémentation SW

2.4.1-/prj

2.4.1.1-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* *Sous linux :* Répertoire pour le linker : /usr/share/gptuils/lkr (à exporter) *Sous Windows :* Répertoire pour le linker : ./ (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 (génère un fichier .map, génère un fichier objet) Flags pour l'assembleur : -c -e ON (génère un fichier objet) -D<Version du firmware » Flags de debug : active une correction référence par un numéro de fiche sur github flag pour la génération du firmware de test par l'assembleur : TEST flag pour la génération du firmware de calibration par l'assembleur : *CALIBRATION*
Composants	Composants : tempo,lcd, i2c, eep, flh, adc, calc, digit_pot
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
```

2.4.1.2-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

2.4.2/sw/inc

2.4.2.1-lcd.inc
Contient les define du LCD.

2.4.2.2-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*

2.4.3-/sw/lcd

2.4.3.1-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

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 2. 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	

2.4.3.2-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 3. Positionner le curseur sur la ligne 2, 4ème case : ○ W=0x10 ○ Positionner le curseur du LCD (f_lcd_setposcursor) 4. v_charpos = 0x00

	5. Afficher le message de boot ligne 2 Tant que W≠0 o Récupérer 1 caractère du message de boot ligne 2 (c_bootmsgL2) o Afficher 1 caractère sur le LCD (f_lcd_affchar) o Incrementer v_charpos 6. Positionner le curseur sur la ligne 2, 10ème case : o W=0x1C o Positionner le curseur du LCD (f_lcd_setposcursor) 7. v_charpos = 0x00 8. afficher la version Tant que W≠0 o W ← v_charpos o Récupérer le caractère en EEPROM (f_eep_readbyte) dans W o Afficher 1 caractère sur le LCD (f_lcd_affchar) o Incrementer v_charpos 9. 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	<table><tr><td>F</td><td>W</td><td>D</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>R</td><td>E</td><td>F</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>
Traitements	1. v_charpos = 0x00 2. Afficher le message de calibration ligne 1 Tant que W≠0 o Récupérer 1 caractère du message de calibration ligne 1 (c_testmsgL1) dans W o Afficher 1 caractère sur le LCD (f_lcd_affchar) o Incrementer v_charpos 3. Positionner le curseur sur la ligne 2 : o W=0x10 o Positionner le curseur du LCD (f_lcd_setposcursor) 4. v_charpos = 0x00 5. Afficher le message de calibration ligne 2 Tant que W≠0 o Récupérer 1 caractère du message de calibration ligne 2 (c_testmsgL2) o Afficher 1 caractère sur le LCD (f_lcd_affchar) o 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	<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></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></tr></table>
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	<table><tr><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>y</td><td>y</td><td>y</td><td>y</td><td>m</td><td>V</td></tr></table>
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	

2.4.4-/sw/calc

2.4.4.1-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*	• $SWR = \frac{ADCfwd + ADCref}{ADCfwd - ADCref}$

2.4.5-/sw/readadc

Mettre le flag ADC_PIC dans le makefile

2.4.5.1-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

2.4.5.1-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

2.4.5.2-adc_maxim.asm

Driver MAX11100 ET MAX4624

2.4.6-/sw/eep/

2.4.6.1-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

2.4.7-/sw/flh/

2.4.7.1-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

2.4.8-/sw/data/

2.4.8.1-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.

2.4.8.2-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

2.4.8.2-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

2.5-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

3-Outils Logiciels

3.1-GPUTILS

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
```

3.2-GPSIM

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

3.2.1-Installation:

1. Désinstaller la version courante de la distribution
2.
```
$sudo apt-get remove gpsim
```
3. Télécharger la version 0.31.0 en suivant ce lien : https://sourceforge.net/projects/gpsim/files/gpsim/0.31.0/
4. Installation

```
$tar -xvzf gpsim-0.31.0.tar.gz
$cd gpsim-0.31.0/
$./configure
$make
$sudo make install
$sudo /sbin/ldconfig
```

3.2.2-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.
```

3.2.3-Librairie et module

Les modules AOP et ADC ne sont pas nativement inclus dans la suite gpsim. Une librairie a été développée à cette fin, que l'on peut retrouver dans src/hw/sim/gpsim/lib.

Pour compiler gpsim avec cette librairie :

1. Création de lien symboliques vers les sources de la librairie, des modules, et le Makefile

```
$cd .../gpsim0.31/modules
$ ln -s ~/devel/f8kgl/swr_power_meter/src/hw/sim/gpsim/lib/Makefile.am
$ ln -s ~/devel/f8kgl/swr_power_meter/src/hw/sim/gpsim/lib/ltc2305.cc
$ ln -s ~/devel/f8kgl/swr_power_meter/src/hw/sim/gpsim/lib/ltc2305.h
$ ln -s ~/devel/f8kgl/swr_power_meter/src/hw/sim/gpsim/lib/ad5175.cc
$ ln -s ~/devel/f8kgl/swr_power_meter/src/hw/sim/gpsim/lib/ad5175.h
$ ln -s ~/devel/f8kgl/swr_power_meter/src/hw/sim/gpsim/lib/swrpowermeterf8kgl.cc
```

2. compiler les fichiers ajoutés

```
$cd .../gpsim0.31/
$ autoreconf
$./configure
$make
$sudo make install
$sudo /sbin/ldconfig
```

3.3-GitHub Desktop

L'ensemble du projet a été placé sous contrôle de version sur un serveur git :

https://github.com/f8kgl/swr_power_meter

Les développeur ont utilisé :

- Atom sous liux (Debian 10) pour la partie logicielle
- GitHub Desktop sous Windos pour la partie matérielle

Etiqueter une release :

```
$ git tag -a v0.6 -m "Tag V0.6"
$ git tag
$ git show v0.6
$ git push origin v0.6
```

3.4-Kicad

À faire (petit tuto)

3.5-Qucs

Qucs est une suite logicielle permettant de simuler un circuit électronique. Il se concentre que les contraintes et aspects radio des siganux.
Cependant, les résultats de la simulation du détecteut HF par exemple n'ont pas montré de les même résultats que les mesures pratiques. En particulier, les modèles de diode présentes dans les librairies de Qucs ne sont pas conformes aux spécifications constructeurs.

Son utilisation a dès lors été abadonnée.