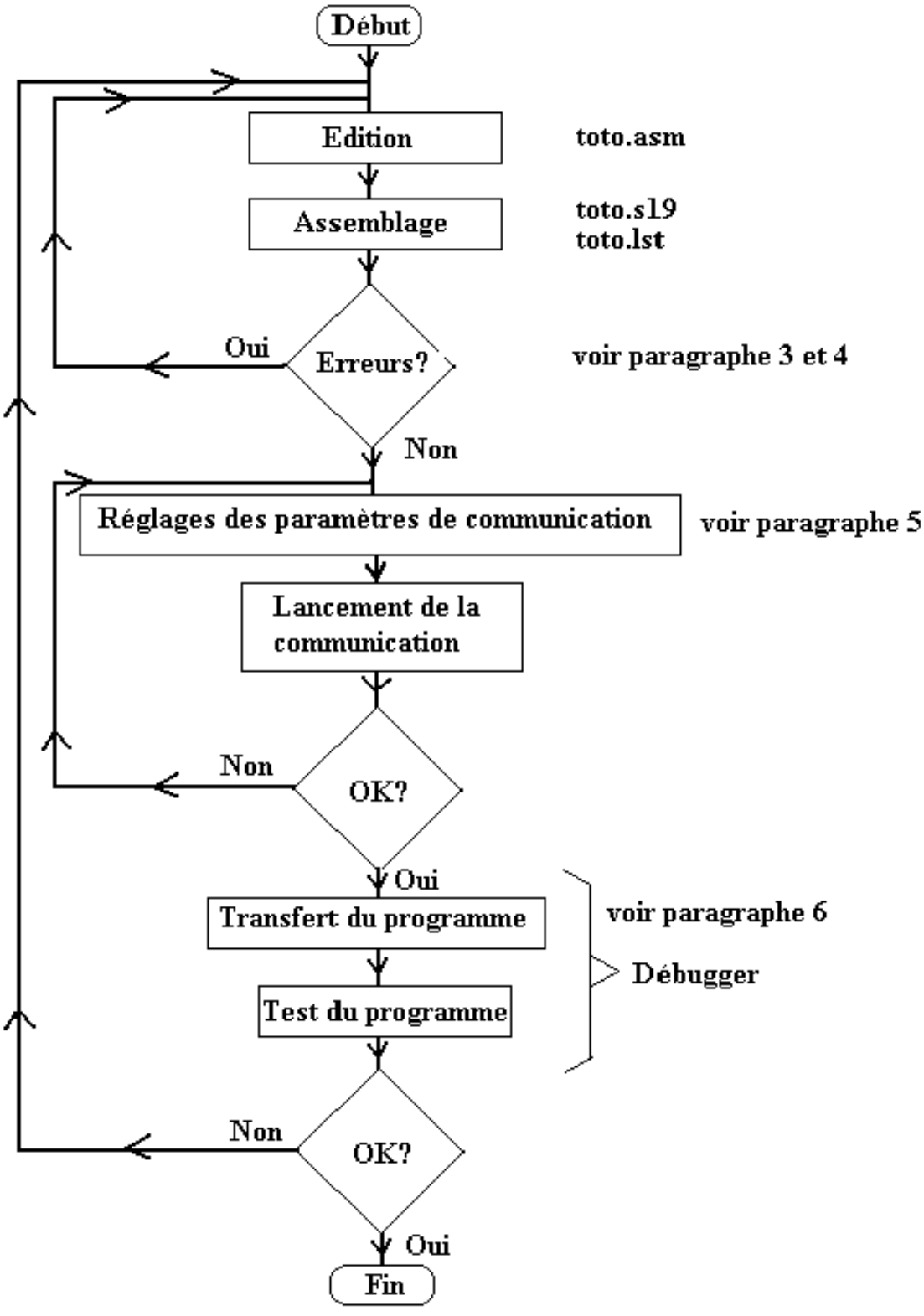
SOMMAIRE

1) DÉFINITION

Pour réaliser et utiliser un programme, il est nécessaire de suivre l'algorithme suivant

ALGORIGRAMME D'UTILISATION DU LOGICIEL WINHC11



2) L'ÉDITION

Elle consiste à écrire un programme (suite d'instructions) à l'aide d'un éditeur de texte.
Elle utilise le langage de l'assembleur X68C11et la syntaxe du µC68HC11.

On lance l'édition à l'aide d'une commande du logiciel WinHC11.

Une ligne d'instruction se présente sous la forme suivante :

ETIQUETTE CODE OPERATOIRE OPERANDE ; ; ''commentaire''

2.1) LANGAGE DE L' ASSEMBLEUR X68C11

2.1.1) LA SYNTAXE DE L'ASSEMBLEUR X68C11

2.1.1.1) Label ou étiquette

Distingue les majuscules des minuscules. Il est de règle de les mettre toujours en majuscule.
Un label doit être déclaré dans son ***champ label (colonne 1)*** , sinon les sauts relatifs sur ce label ne pourront pas être calculés et donc forcés à l'adressage étendu.

2.1.1.2)Adressage

Mode	Code opérande	Opérande	*Le code opérande est ...*
Inhérent		**Implicite**ment contenue dans l'instruction	**absent**
Immédiat	**#Label** **#Valeur**	**Valeur :** *sur 8 ou 16 bits suivant l'instruction.*	**l'opérande lui-même**
Etendu	**Label**		**l'adresse de l'opérande**
Direct	**< Label**		**l'adresse de l'opérande** Utilisé lorsque l'opérande est situé en page 0. L'adressage direct peut être automatique en utilisant les directives 'Absolute' et 'relative' lors des réservations de mémoire.
	Label		
Indexé	**Offset,X**		l'index utilisé et le décalage. L'adresse de l'opérande est calculée par : **X + Offset (en non signé)**
	Offset,Y		

2.1.1.3) Types

	Bases	***ex :***	
	Décimal	LDAA #35	
$	Hexadécimal	LDAA #$2A	
@	Octal	LDAA #@25	
%	Binaire	LDAA #%10010111	
'	ASCII	LDAA #'K'	

2.1.1.4) Opérations : le type du résultat doit rester compatible *AVEC L'INSTRUCTION SINON* *erreur.*

+ , -	arithmétique	ldab #$46+5	**& , ^**	ET , OU	ldab #$46&$F0
*** , /**	en non signé	ldab #3562/60	**.XOR.**	Xor	ldab #$46.XOR.$F0
!	complément	ldab #!$45			
******	Exp ; 2**12 = 2^{12}	ldab #73728/2**9			

Les opérations ne travaillent que sur des entiers sans limite de dimension, alors il faut faire les * et + avant les / et - pour garder la précision. Le résultat doit être compatible avec son utilisation.

2.1.2) DIRECTIVES D'ASSEMBLAGE

2.1.2.1) Contrôle de stockage

Directive	***Signification***	***Exemples***
ORG	origine des labels	ORG $B600
END	fin d'assemblage	END
FCB	stocke des octets en mémoire.	MEM1 FCB $55 MEM2 FCB 21,%01101011,$E7 Mess1 FCB 'I am happy !'
FDB	comme FCB sur des mots de 16 bits	MEM3 FDB $55AA
FCC	stocke une chaîne de caractères délimitée par un même caractère.	Mess2 FCC /I'm happy !/ Mess3 FCC *Vitesse (m/s) = *
RMB	réserve des bytes en mémoire	TAB RMB 4

2.1.2.2) Définitions

Directive	*Signification*	*Exemples*
EQU	définit une constante	CONST1 EQU $B600
VAR	comme EQU mais redéfinissable en cours de programme. Utilisé lors des déclarations des variables locales stockées dans la pile.	CPTR VAR 2

2.1.2.3) Mode

Directive	*Signification*	*Exemples*
INCLUDE	permet d'inclure un fichier DOS. Un seul niveau d'inclusion.	INCLUDE FICHIER.EXT
ABSOLUTE **:** **:** **:** **RELATIVE**	Pour encadrer les labels qui doivent être adressés en direct sur la page 0 On peut s 'en passer mais alors l'adressage direct doit être explicite.	ABSOLUTE ORG 0 LAB1 ... ... RELATIVE BRSET <Ad,M,Rel

2.2) INSTRUCTIONS DU MC 68HC11

UN jeu complet des instructions est donné en annexe

2.2.1) ADRESSAGE

Les instructions sont données à l'aide de mnémoniques

Dans les instructions 'ad' peut être :

rien **:** *Inhérent*.
#Valeur **:** *Immédiat*.
Label : *Etendu, ou direct automatique en page 0*.
Offset,X **:** *Indexé* sur X ou Y avec offset constant 8 bits, non signé.
'rel' représente l'adressage relatif exclusivement réservé pour les branchements.

2.2.2) CHARGEMENT, MEMORISATION ET TRANSFERT

Fonction		***Mnémonique***
Mise à 0	d'un octet mémoire	CLR ad
	de l'accu A ou B	CLRr
Charge	l'accu A ou B	LDAr ad
	D, X , Y ou S	LDr ad
Mémorise	l'accu A ou B	STAr ad
	D, X, Y ou S	STr ad
Transfert	A dans B ou l'inverse	TAB ou TBA
	A dans CCR ou l'inverse	TAP ou TPA
	SP+1 dans X ou Y	TSr
	X-1 ou Y-1 dans SP	TrS
Echange D avec X ou Y		XGDr
Tire A, B, X ou Y de la pile		PULr
Pousse A, B, X ou Y dans la pile		PSHr

2.2.3) MANIPULATION ET TEST DE BITS

Fonction		***Mnémonique***
Bit Test A ou B avec la mémoire		BITr ad
Bits mis à	Zéro	BCLR ad, msk
	Un	BSET ad, msk
Branche si bit(s) égal	Zéro	BRCLR ad, msk, rel
	Un	BRSET ad, msk, rel

2.2.4) OPERATIONS ARITHMETIQUES

Fonction		*Mnémonique*
Addition	de l'accu B à A , à X ou Y (non signé)	ABr
	d'une mémoire à A ou B (8 bits) à D (16bits)	ADDr ad
	avec retenue à A ou B	ADCr ad
Soustrait	mém. à A ou B sans retenue à D (16bits)	SUBr ad
	mém. à A ou B avec retenue	SBCr ad
	B à A	SBA
Ajustement décimal de A (après une addition, pas une incrémentation)		DAA
Incrémente	un octet mémoire	INC ad
	A ou B	INCr
	S, X ou Y	INr
Décrémente	un octet mémoire	DEC ad
	A ou B	DECr
	S, X ou Y	DEr
Complément à 2	d'un octet mémoire	NEG ad
	de A ou B	NEGr
Compare	A à B	CBA
	A ou B à une mémoire	CMPr ad
	D, X ou Y à la mémoire	CPr ad
Test si zéro ou négatif	d'un octet	TST ad
	de A ou B	TSTr

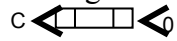
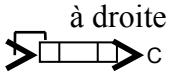
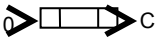
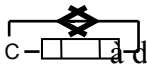
2.2.5) OPERATIONS LOGIQUES

Fonction	*Mnémonique*
mémoire **ET** A ou B	ANDr ad
mémoire **OU** A ou B	ORAr ad
mémoire **OU EX** A ou B	EORr ad
Bit test mémoire **ET** A ou B	BITr ad
Complémente mémoire	COM ad
A ou B	COMr

2.2.6) MULTIPLICATION ET DIVISION

Fonction	*Mnémonique*
Multiplication : A x B => D	MUL
Division fractionnaire : D : X => X ; r => D Num, Dénom = entiers 16bits, Résult et Reste= fractionnaires 16 bits: $.xxxx	FDIV
Division entière : D : X =>X ; r => D Num, Dénom, Résult et Reste = entiers 16bits	IDIV

2.2.7) DECALAGES ET ROTATIONS

Fonction			*Mnémonique*
Décalage	à gauche	d'un octet	ASL ad
Arithmétique		de A, B ou D	ASLr
	à droite	d'un octet	ASR ad
		deA ou B	ASRr
Décalage	à gauche	d'un octet	LSL ad
logique	(≡ ASL)	de A, B ou D	LSLr
	à droite	d'un octet	LSR ad
		de A, B ou D	LSRr
Rotation	à gauche	d'un octet	ROL ad
		de A ou B	ROLr
	à droite	d'un octet	ROR ad
		de A ou B	RORr

2.2.8) STRUCTURE

Fonction	*Mnémonique*
Branche à un sous-programme	BSR rel
Saut à un sous-programme	JSR ad
Retour de sous-programme	RTS
Retour d'interruption	RTI
Interruption programmée	SWI
Attente d'interruption	WAI
Pas d'opération	NOP
Arrête l'horloge	STOP

2.2.9) INSTRUCTIONS SUR LE REGISTRE D'ETAT

Fonction	*Mnémonique*
Mise à zéro du bit C, I, V	CLb
Mise à un du bit C, I, V	SEb

2.2.10) SAUTS ET BRANCHEMENTS

Fonction			*Mnémonique*	
Saut			JMP ad	
Branche	**toujours**		BRA rel	
	jamais		BRN rel	
	si **non retenue**	(C = 0)	BCC rel	
	retenue	(C = 1)	BCS rel	
	# 0	(Z = 0)	BNE rel	
	= 0	(Z = 1)	BEQ rel	
	≥ 0	(N = 0)	BPL rel	
	< 0	(N = 1)	BMI rel	
	pas de dépassement	(V = 0)	BVC rel	
	dépassement	(V = 1)	BVS rel	
	Après opération de comparaison ou soustraction			
	si ≥	(signé)	BGE rel	
	≥	(non signé)	BHS rel	≡ BCC
	>	(signé)	BGT rel	
	>	(non signé)	BHI rel	
	≤	(signé)	BLE rel	
	≤	(non signé)	BLS rel	
	<	(signé)	BLT rel	
	<	(non signé)	BLO rel	≡ BCS
	si **bits à 0**		BRCLR ad,msk,rel	
	bits à 1		BRSET ad,msk,rel	

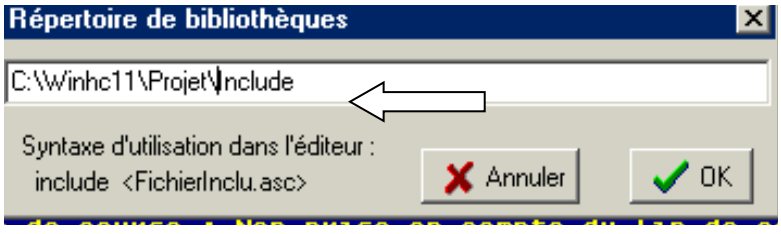
3) INCLUSION DE FICHIER

Pour inclure un fichier mnémonique dans votre programme, par exemple le fichier de configuration des 64 octets du µC68HC11 (registr.equ), il faut ajouter la ligne suivante dans votre programme :

pour la version Winhc11 INCLUDE ../REGISTR.EQU ;table des registres de sortie

pour la version Winhc12 INCLUDE ⟨ REGISTR.EQU ⟩ ;table des registres de sortie

Il faut aussi définir le chemin d'accès au fichier inclus par la commande

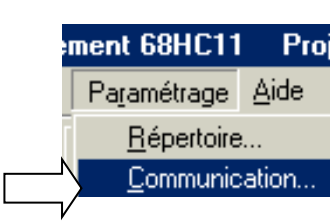


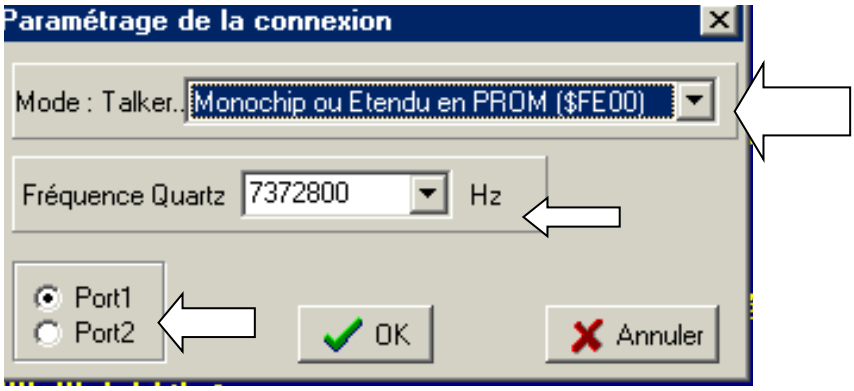
4) ASSEMBLAGE

Il s'effectue à l'aide d'une commande du logiciel WinHC11.
Deux fichiers sont générés d'extensions lst et S19.

5) REGLAGE DES PARAMETRES DE COMMUNICATION

Il s'effectue à l'aide d'une commande du logiciel WinHC11.





* Choix du port série de l'ordinateur (com1 / com2)
* Choix de la fréquence du quartz : 7372800Hz
* Choix du talker : : Bootstrap en PROM $FE00

6) DEBUGGER

Il s'effectue à l'aide d'une commande du logiciel WinHC11.
Une aide est disponible dans le logiciel.