

UTILISATION DE LOGIPIC

L'AFFICHEUR LCD

LES VARIABLES

Faire clignoter des Leds permet de prendre en main et de tester la programmation visuelle des pics de façon simple avec Logipic.

Les microcontrôleurs permettent cependant de réaliser des applications beaucoup plus sérieuses nécessitant l'utilisation d'un afficheur LCD et des variables.

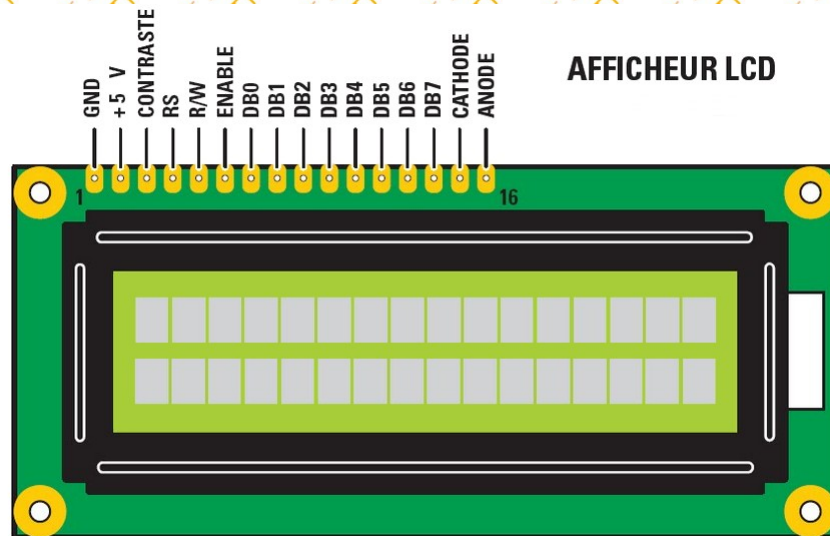
Ce tutoriel permet de mettre en oeuvre un afficheur LCD.

L'Afficheur LCD

L'afficheur LCD que l'on utilisera est un module très répandu qui utilise un contrôleur HD44780 qui gère l'affichage sur 2 lignes de 16 caractères.

Le contrôleur HD44780 est commandé par une liaison série qui peut fonctionner en mode 8 bits (DB1 à DB8) ou en mode 4 bits (DB5 à DB8), les commandes E, RS et RW permettent de gérer l'envoi des données séries.

L'afficheur LCD est complété par ses broches d'alimentation (GND et +5V), par un réglage de contraste et un rétro-éclairage (Cathode et Anode).



BROCHE	DESCRIPTION
1	masse module
2	alimentation module 5V
3	contraste
4	sél. registres
	0 instruction
	1 données
5	Read-write (lis-écrit)
	1 read
	0 write
6	enable
7-14	bus data DB0 - DB7
15	rétro éclairage cathode -
16	rétro éclairage anode +

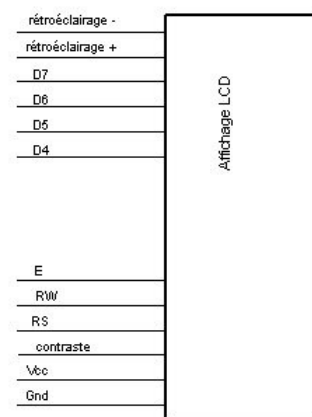
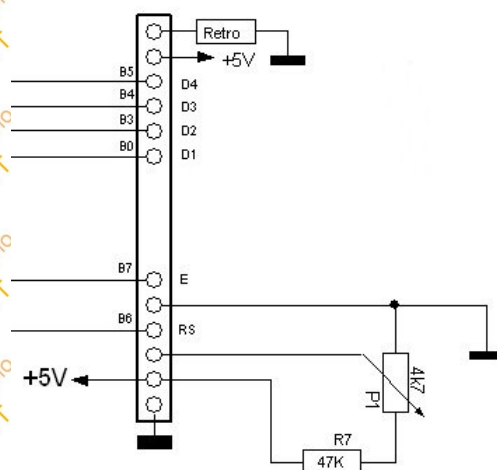
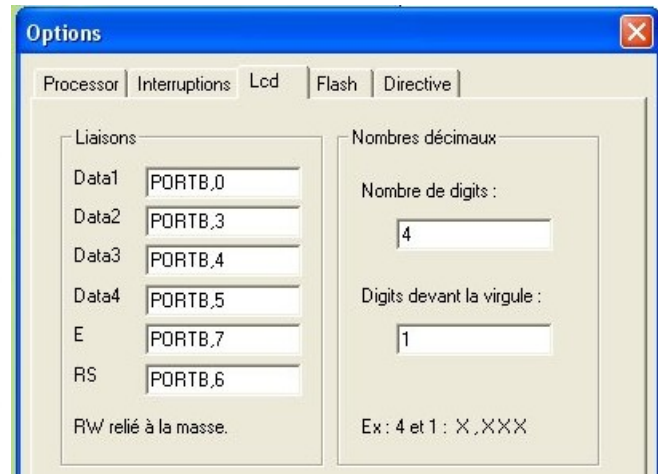
Connexion de l'afficheur LCD à la platine de test

La liaison RW est raccordée à la masse (cette option n'est pas utilisée par Logipic)

Logipic gère le contrôleur en mode 4 bits et il est capable d'allouer n'importe quelle ligne du pic à l'afficheur LCD. La commande de l'afficheur LCD nécessite donc 6 lignes du pic.

Il est indispensable de positionner les liaisons de l'afficheur LCD en fonction du schéma structurel. Tout se passe dans l'onglet LCD des options de Logipic :

Attention : il existe un décalage entre le numéro des liaisons Data et le brochage de l'afficheur LCD. C'est pourquoi Data7 de l'afficheur LCD correspond à Data4 de Logipic (PORTB,5).



La liaison du module LCD avec la platine de test s'effectue par un simple enfichage.

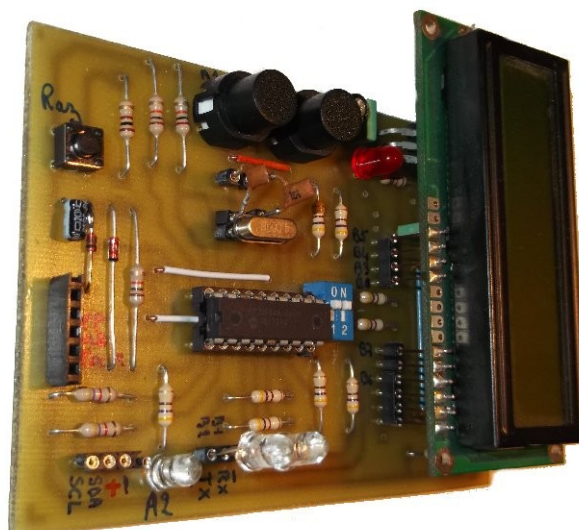
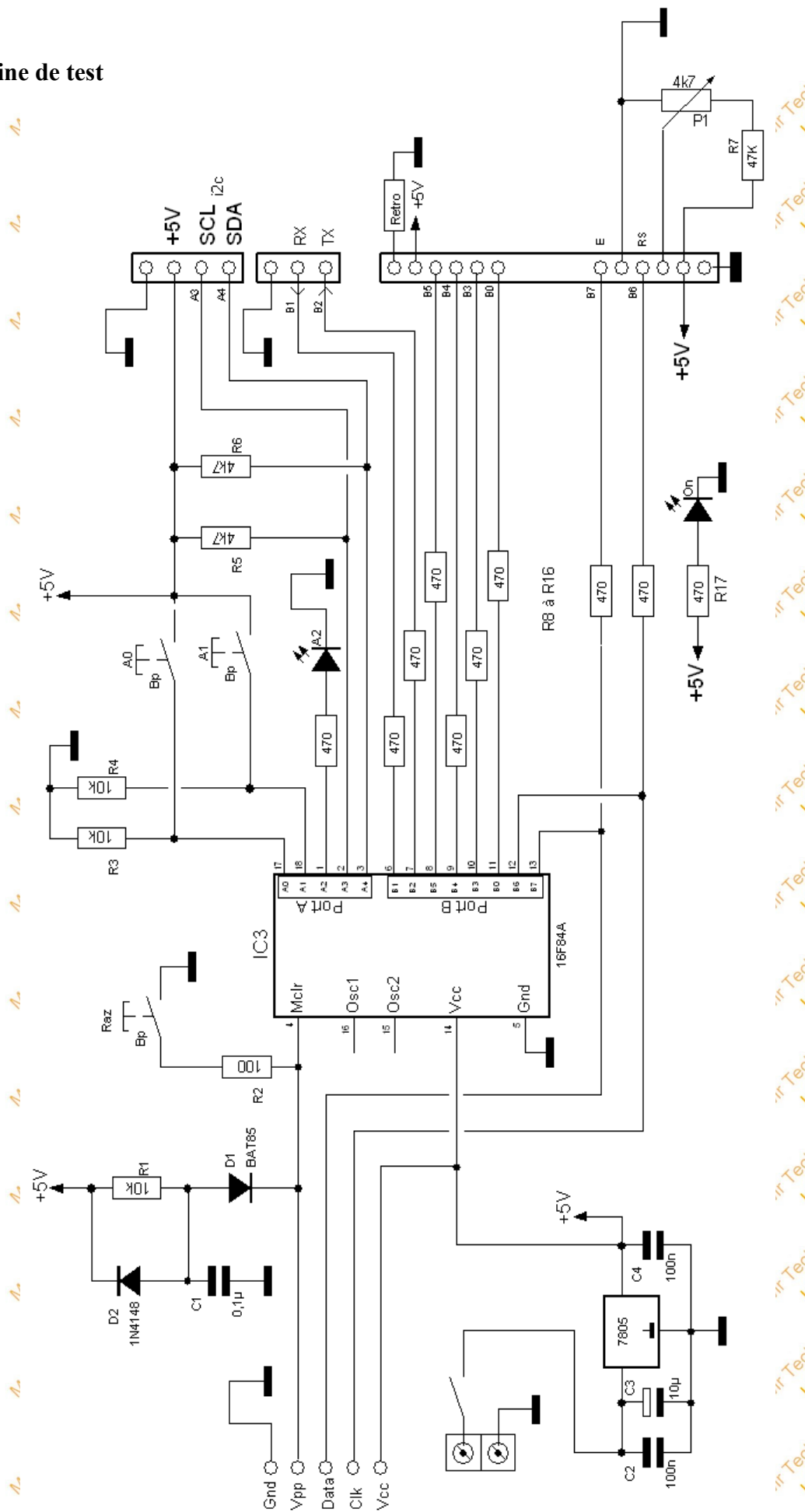


Schéma de la platine de test



Utilisation de l'afficheur LCD avec Logipic

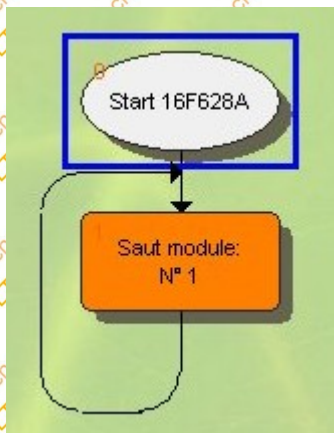
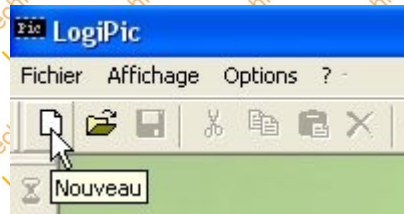
Ce programme consistera à afficher un message simple.

La platine de test utilise un pic 16F628A cadencé à l'aide d'un quartz à 4MHz.

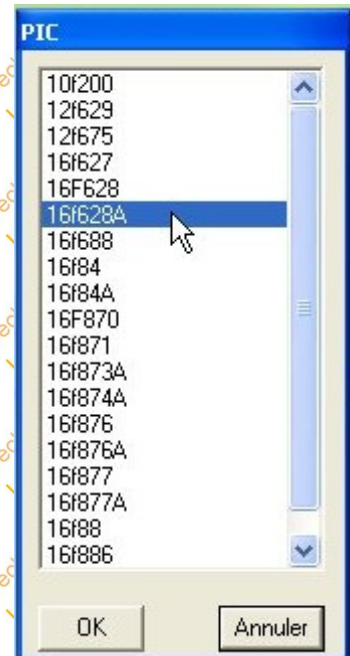
Lancer Logipic214



Demander Nouveau



Choisir le pic :



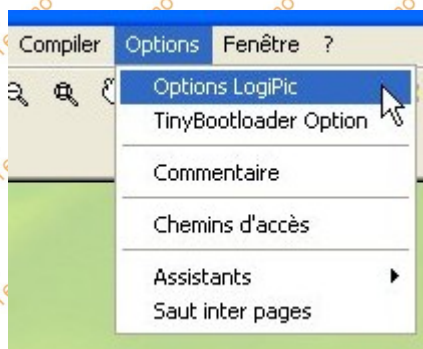
Je vous conseille d'enregistrer votre programme dans un dossier caractéristique, par exemple 'soft pic' nom du fichier : lcd1.prj

Attention longueur maximum de nom : 8 caractères

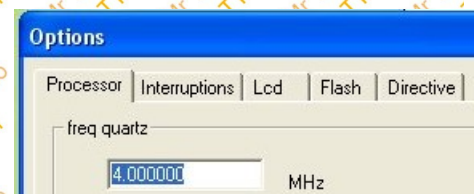
Dans un premier temps, il est très important de configurer les options de Logipic.

Sans cette phase, le pic ne fonctionnera pas

Demander Options > Options Logipic :



Freq quartz :
4MHz



Directive :



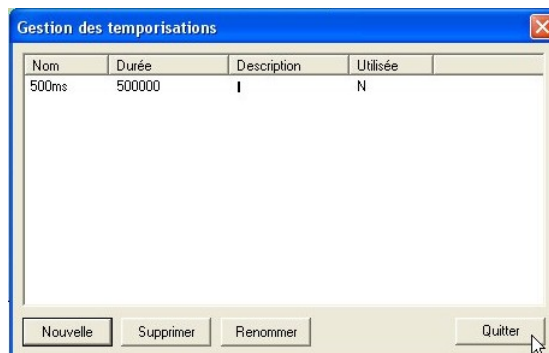
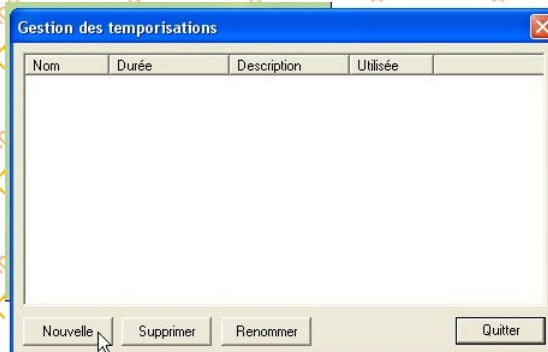
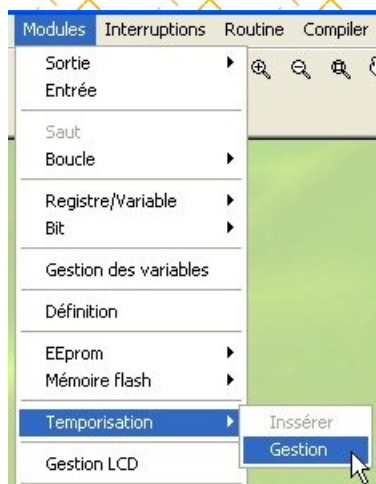
Options LCD :



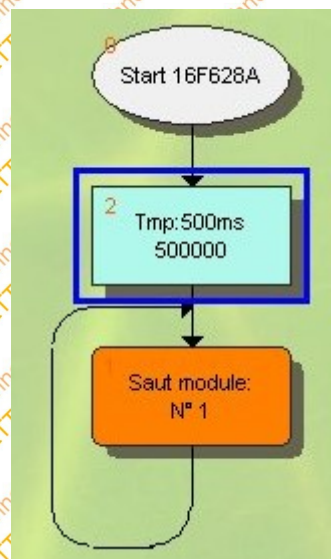
Thierry Lancelot

Logipic2

Dans un premier temps, positionner un temporisation de 500ms :



Dans un premier temps, positionner une temporisation en début de programme, celle ci est non indispensable mais permet à l'afficheur LCD de se configurer correctement.

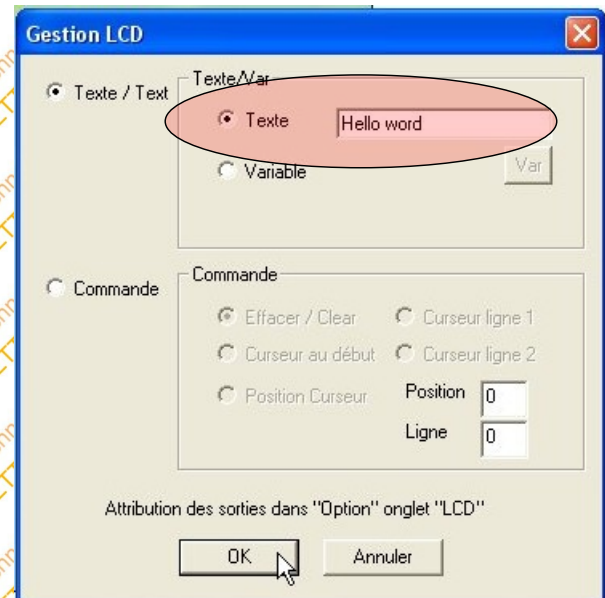
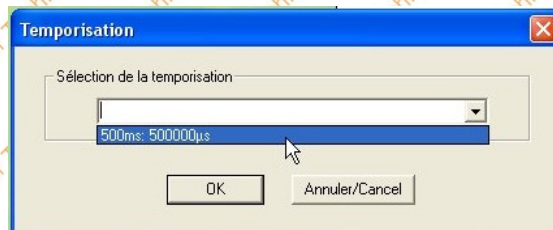


Puis cliquer sur l'icone LCD :



Demander l'affichage du texte : Hello word

Puis mettre une temporisation.

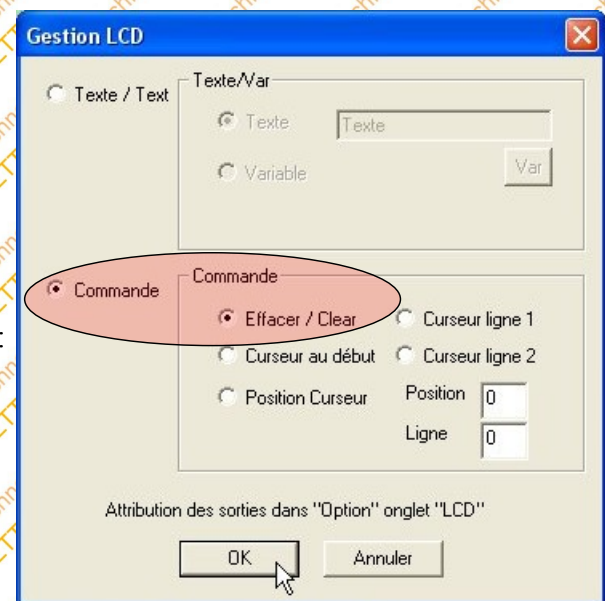


Cela permet d'afficher le texte pendant cette temporisation

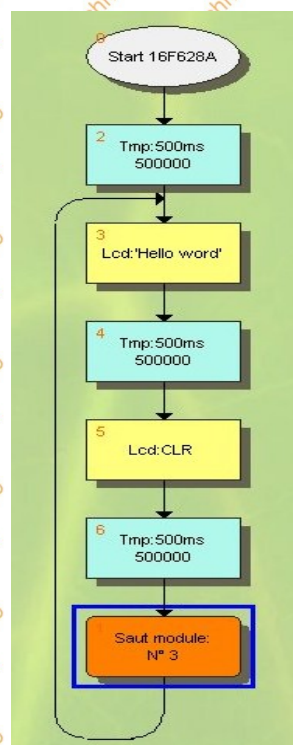
Recliquer sur LCD :



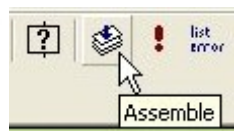
Demander une commande : Effacer/Clear :



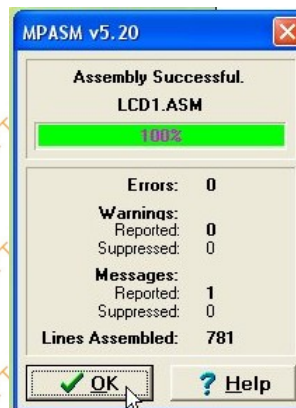
Terminer l'organigramme avec une temporisation effectuer un bouclage sur la commande LCD



Effectuer l'assemblage :



puis la compilation :



Et enfin, demander le transfert du fichier 'hex' créé :



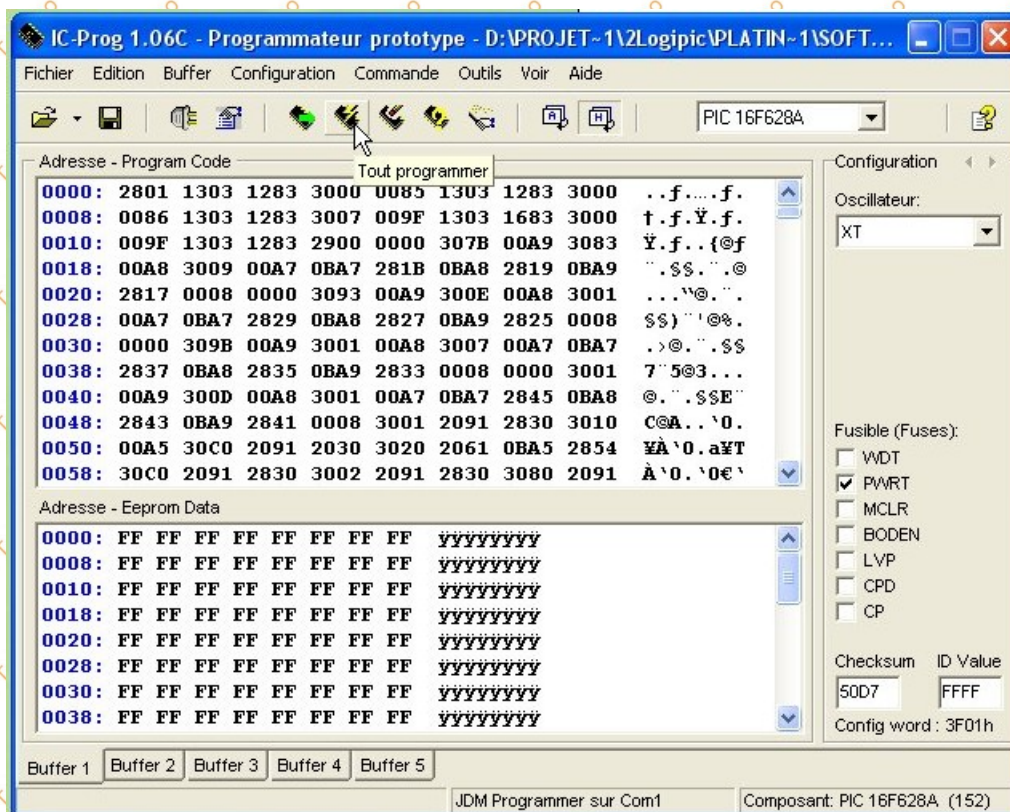
Remarque ; si le pic sélectionné par IcProg n'est pas le bon, il est nécessaire de sélectionner le pic

16F628A, de fermer IcProg et de relancer la commande



Connecter le programmeur icsp à la platine de test, mettre sous tension la platine de test.

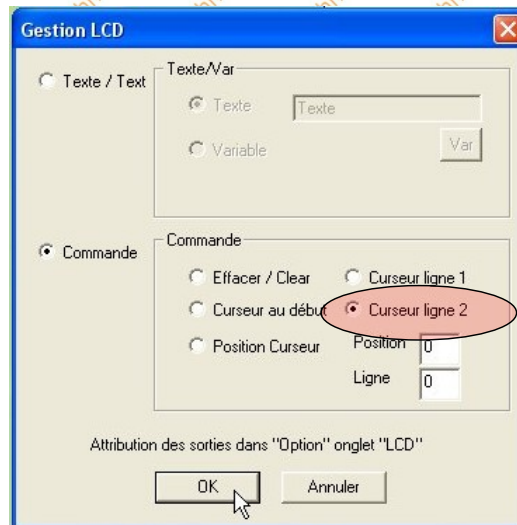
Demander le transfert du programme :



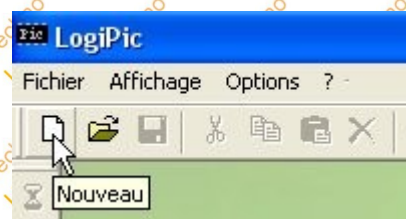
L'afficheur LCD s'anime et affiche « Hello word ».

Gestion de l'afficheur LCD

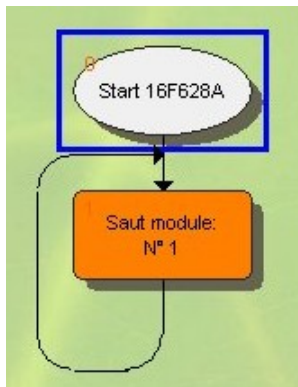
Les commandes LCD permettent de contrôler l'afficheur : le choix de la ligne ou la position du curseur (Position et Ligne).



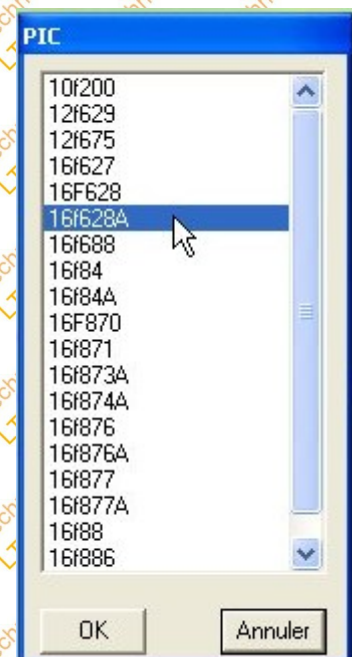
Lancer Logipic214



Demander Nouveau

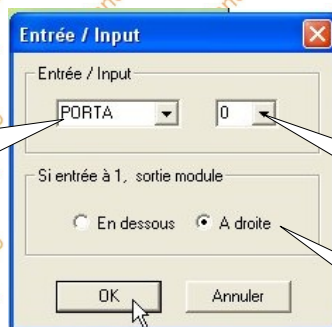


Choisir le pic :



nom du fichier : lcd2.prj

Le test d'une entrée s'effectue avec l'instruction entrée :



Choix du port l'entrée

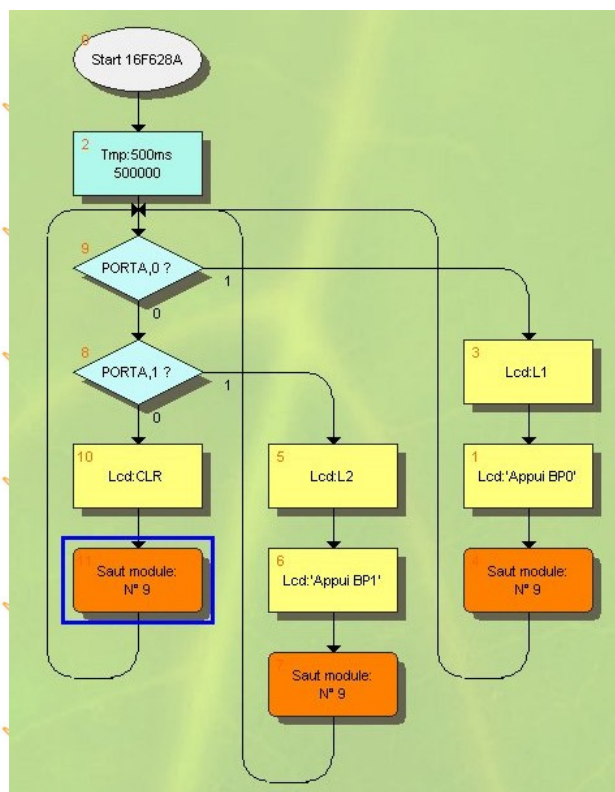
Choix du numéro de port

Choix du sens de sortie



Ecrire l'organigramme suivant :

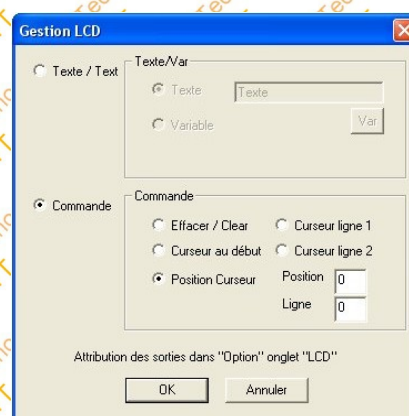
Il permet d'afficher le numéro du bouton-poussoir qui est actionné :



L'Afficheur LCD 4 lignes

Il s'agit d'une information de Michel Leclerc. Un afficheur 4 lignes / 20 caractères peut être utilisé comme un afficheur de 2 lignes / 40 caractères.

Gestion LCD, [commande] / [position du curseur]



- 1ère ligne - Position0 à 19 et Ligne1
- 2ème ligne - Position0 à 19 et Ligne2
- 3ème ligne- Position20 à 39 et Ligne1
- 4ème ligne- Position20 à 39 et Ligne2

Sur les anciennes versions de Logipic, il y a un problème :

logipic n'accepte pas de donner un valeur nulle et en réalité avec une position 1 on écrit le caractère en position 2. (Ce problème n'est pas caractéristique aux afficheurs 4 lignes, c'est exactement pareil pour les afficheurs 1 ligne et 2 lignes).

Dans ce cas, procédez comme suit :

Faire une première programmation en position1, ligne 1 (ou 2) puis accepter en cliquant OK.

Faire un double clic sur la zone jaune juste créée et introduire "0" pour la position et cliquez OK.

Vous allez recevoir 2 messages disant que la valeur doit être comprise entre 1 et 80... cliquez 2X OK puis sur annuler quand il n'y a plus de message d'erreur, et oh surprise, en quittant vous verrez la fenêtre avec x0,y1 ou x0,y2 suivant votre choix.

Affichage des 0 non significatifs

Il s'agit d'une information de Michel Leclerc. Pour afficher l'heure en conservant les 0 non significatifs, par exemple un chrono devrait afficher 00:00 au démarrage. Ainsi, l'affichage de la variable 4 s'effectuera de la façon suivante : 04

La méthode est la suivante:

Soit la variable Var

Pour afficher cette variable il suffit de la diviser par 10 pour isoler les dizaines (Vdiz)

On récupère les unités par la variable système : V_tmp.(Vunit)

Une routine (affichage) affiche sur le LCD sous forme Ascii, il faut donc ajouter 48 à Vdiz et Vunit pour pouvoir obtenir l'affichage correct sur le LCD.

Exemple pour afficher la variable Var de valeur 52 :

1/ isoler les dizaines - $V_{diz} = Var/10$

$V_{diz} = 5$

2/ récupérer les unités $V_{unit} = V_tmp$

$V_{unit} = 2$

3/ Appel de la routine « affichage » utilisant des code assembleur et la routine système LCDchar.

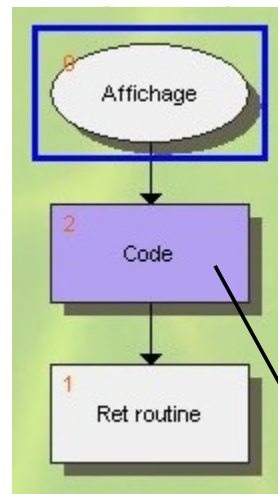
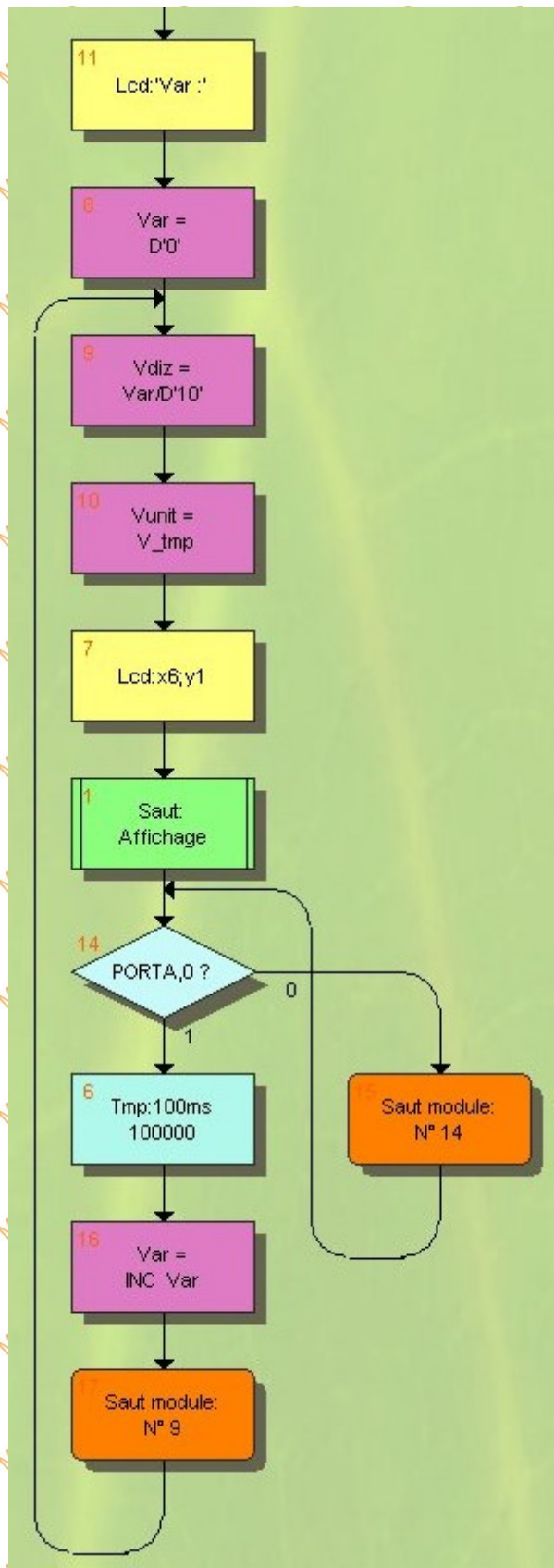
Cette routine ajoute 48 aux 2 variables afin d'obtenir le code Ascii (Le code Ascii 48 à 57 correspondent aux chiffres 0 à 9). L'affichage de 5 s'effectue avec le code Ascii 53.

Cette méthode est valable jusqu'à 99. en effet, au delà de cette valeur il y a un problème puisque $v_{diz} = 10$, ce qui affiche le code Ascii « : » (c'est le code Ascii de 58).

Tableau Ascii :

048d	30h	0	064d	40h	@	080d	50h	P	096d	60h	^	112d	70h	p
049d	31h	1	065d	41h	A	081d	51h	Q	097d	61h	a	113d	71h	q
050d	32h	2	066d	42h	B	082d	52h	R	098d	62h	b	114d	72h	r
051d	33h	3	067d	43h	C	083d	53h	S	099d	63h	c	115d	73h	s
052d	34h	4	068d	44h	D	084d	54h	T	100d	64h	d	116d	74h	t
053d	35h	5	069d	45h	E	085d	55h	U	101d	65h	e	117d	75h	u
054d	36h	6	070d	46h	F	086d	56h	V	102d	66h	f	118d	76h	v
055d	37h	7	071d	47h	G	087d	57h	W	103d	67h	g	119d	77h	w
056d	38h	8	072d	48h	H	088d	58h	X	104d	68h	h	120d	78h	x
057d	39h	9	073d	49h	I	089d	59h	Y	105d	69h	i	121d	79h	y
058d	3Ah	:	074d	4Ah	J	090d	5Ah	Z	106d	6Ah	j	122d	7Ah	z
059d	3Bh	;	075d	4Bh	K	091d	5Bh	[107d	6Bh	k	123d	7Bh	{
060d	3Ch	<	076d	4Ch	L	092d	5Ch	\	108d	6Ch	l	124d	7Ch	
061d	3Dh	=	077d	4Dh	M	093d	5Dh]	109d	6Dh	m	125d	7Dh	}
062d	3Eh	>	078d	4Eh	N	094d	5Eh	^	110d	6Eh	n	126d	7Eh	~
063d	3Fh	?	079d	4Fh	O	095d	5Fh	_	111d	6Fh	o	127d	7Fh	☐

Affichage de 00 à 99



Code ASM

```

MOVWF Vdiz,W
ADDLW D'48'
CALL LCDchar
MOVWF Vunit,W
ADDLW D'48'
CALL LCDchar
  
```

Chaque appui de la touche A0 incrémente la variable. Affichage de 00 à 99.

Le code ASM de l'affichage doit prendre en compte l'affichage des centaines Vcent



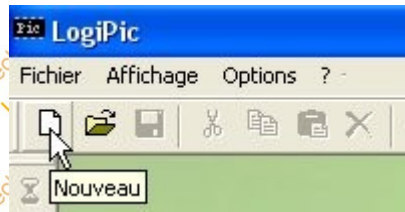
Utilisation des variables avec Logipic

En informatique, une variable permet d'associer un nom à une valeur et est utilisée afin de stocker temporairement des données lorsque le programme est en exécution.

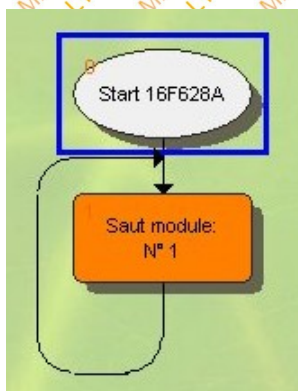
Logipic est prévu pour utiliser des variables de plusieurs types : 8 bits, 16bits etc...

Cependant, il semble que Logipic se limite à l'utilisation des variables 8 bits.

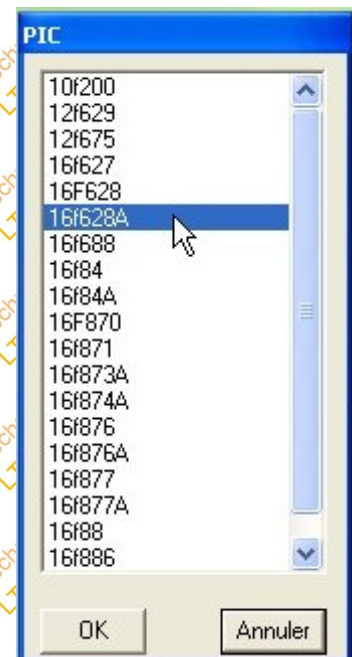
Lancer Logipic214



Demander Nouveau



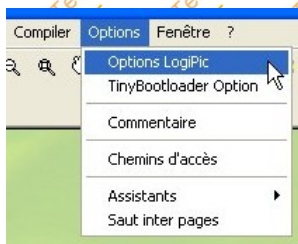
Choisir le pic :



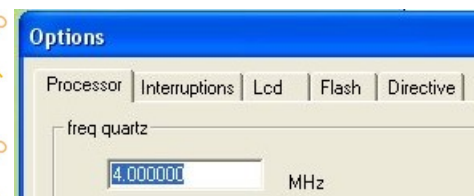
Dans un premier temps, il est très important de configurer les options de Logipic.

Sans cette phase, le pic ne fonctionnera pas

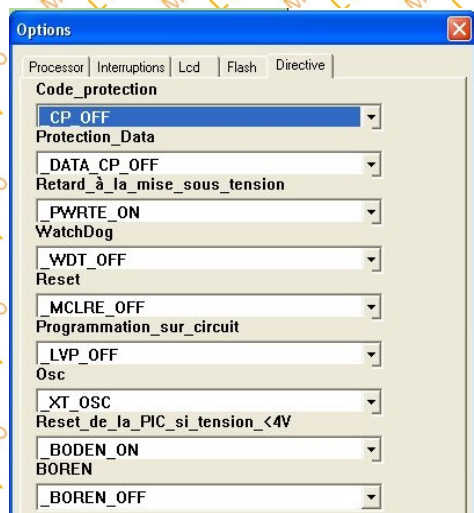
Demander Options > Options Logipic :



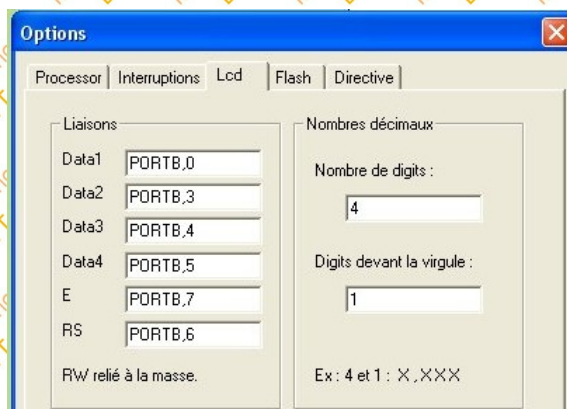
Freq quartz :
4MHz



Directive :



Options LCD :



Positionner un temporisation de 500ms (voir page 5).

Puis cliquer sur l'icone VAR :



Demander Nouvelle :

Définir la variable : var1

Définir une variable

Type: Entier 8 bits
Banque: Banque 0
Nom de la variable: var1
Nb Octet(Buffer): 1
Description: variable simple 8 bits
le nombre d'octet défini la taille du tableau (défaut 1)

OK Annuler

Gestion des variables

Nom	Type	Buffer	Banque	Description	U...
-----	------	--------	--------	-------------	------

Nouvelle Supprimer Renommer Quitter

La variable est créée :

Gestion des variables

Nom	Type	Buffer	Banque	Description	U...
Var1	Int 8	1	0	variable simple 8 bits	N

Nouvelle Supprimer Renommer Quitter

Commencer à écrire le programme :

Mettre une temporisation de 500ms,

puis le texte : « compteur : »

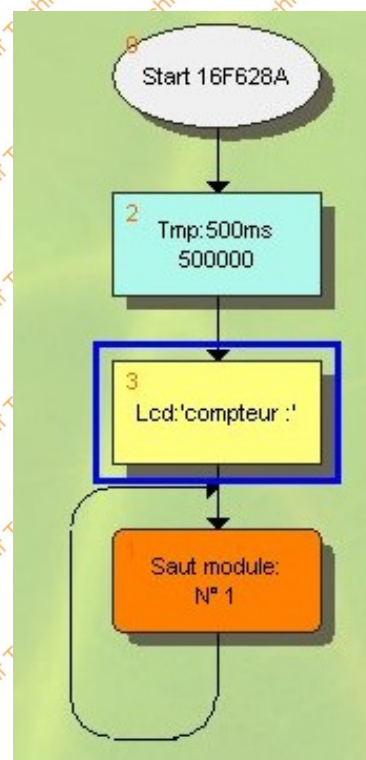
Gestion LCD

☒ Texte / Text
Texte/Var: ☒ Texte compteur : ☐ Variable Var

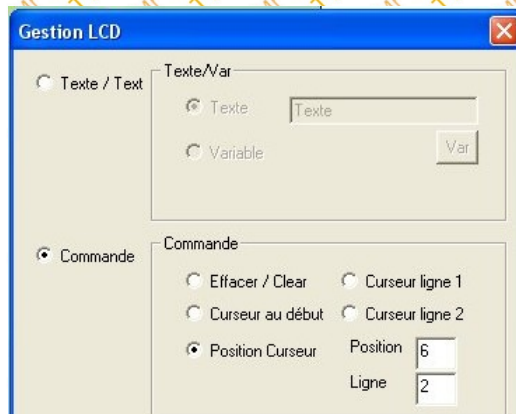
☐ Comande
Comande:
☒ Effacer / Clear ☐ Curseur ligne 1
☐ Curseur au début ☐ Curseur ligne 2
☐ Position Curseur Position: 0
Ligne: 0

Attribution des sorties dans "Option" onglet "LCD"

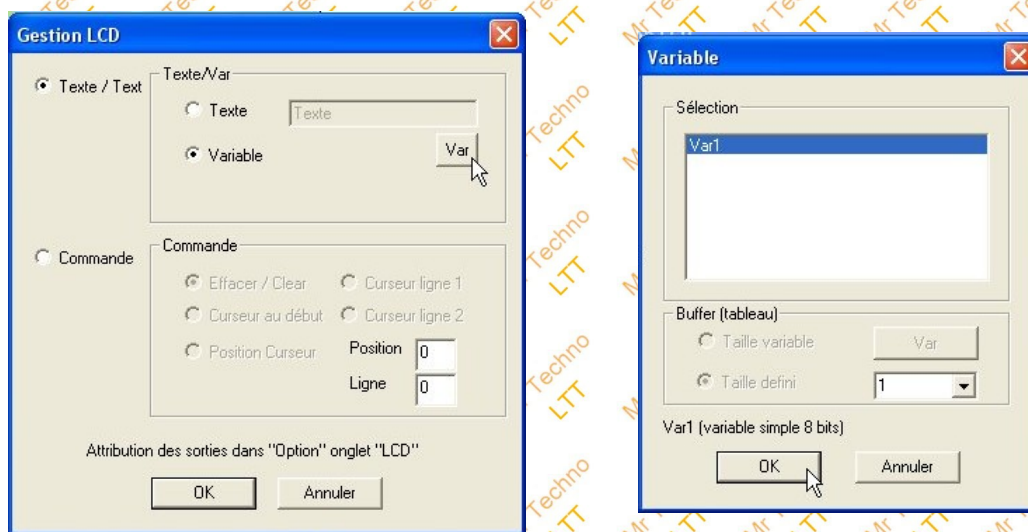
OK Annuler



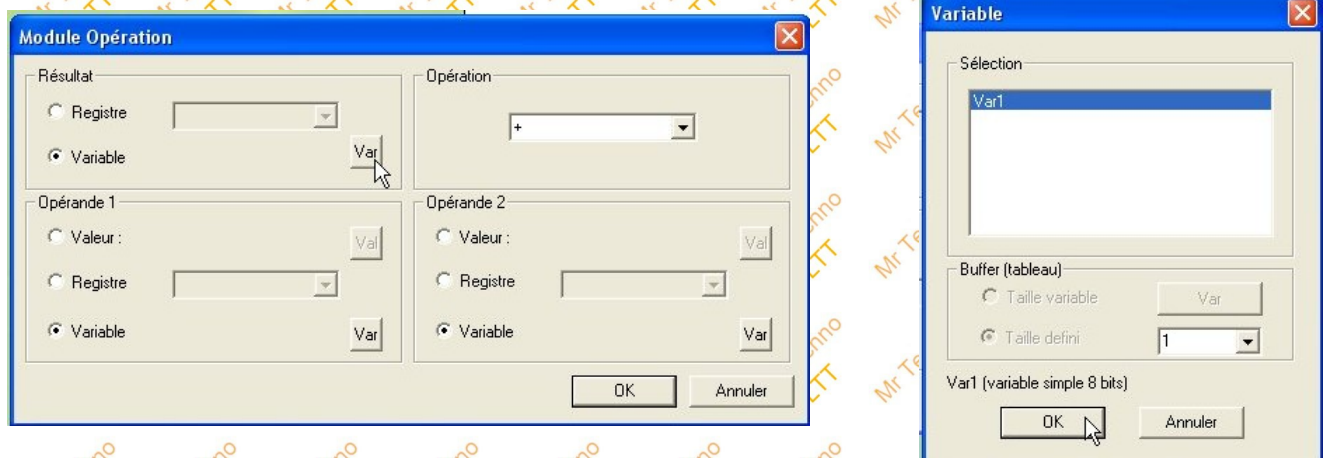
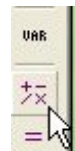
Pour afficher la variable sur la ligne 2, demander une commande Position Curseur

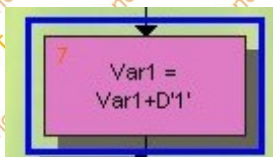
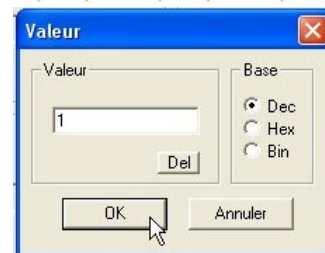
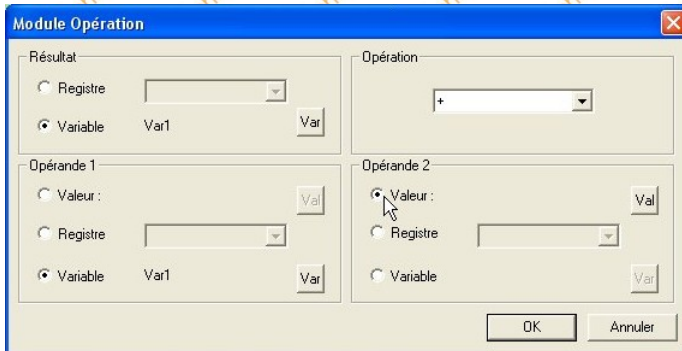
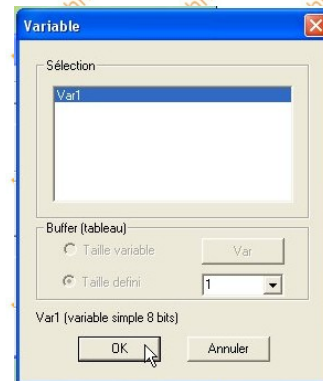
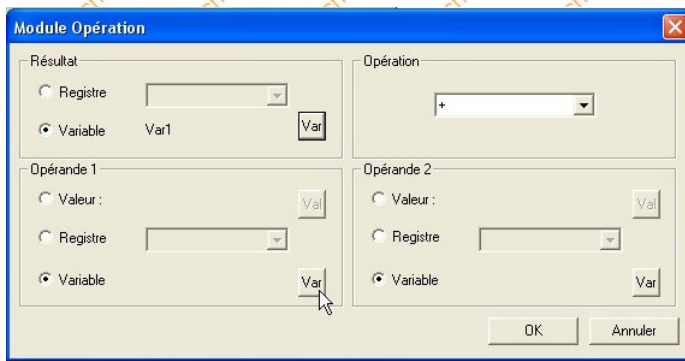


Pour écrire une variable, utiliser la commande Gestion LCD et demander la Variable Var1 :



Pour pouvoir faire évoluer la variable cliquer sur l'icone calcul.





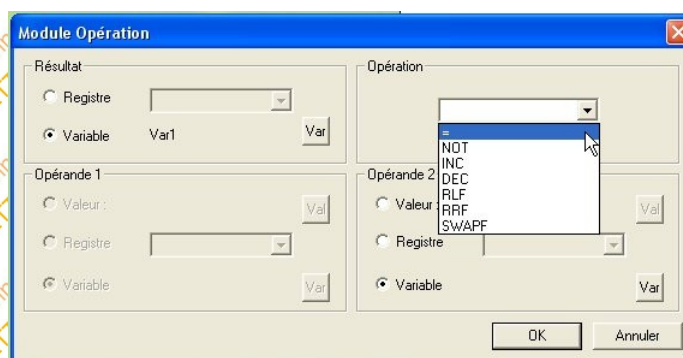
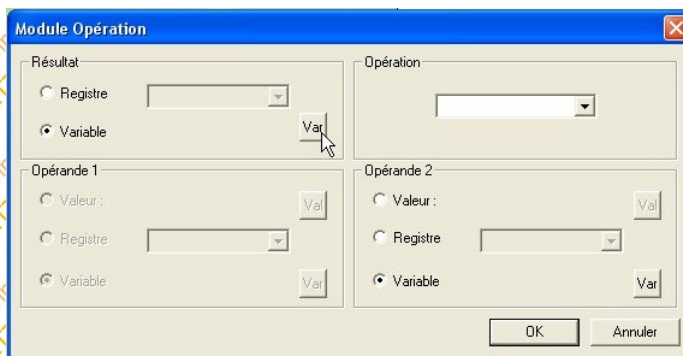
L'opération est écrite : $Var1 = Var1 + D'1'$

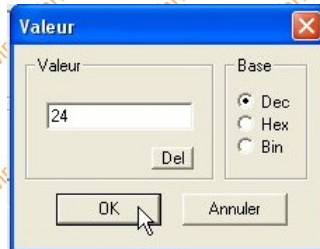
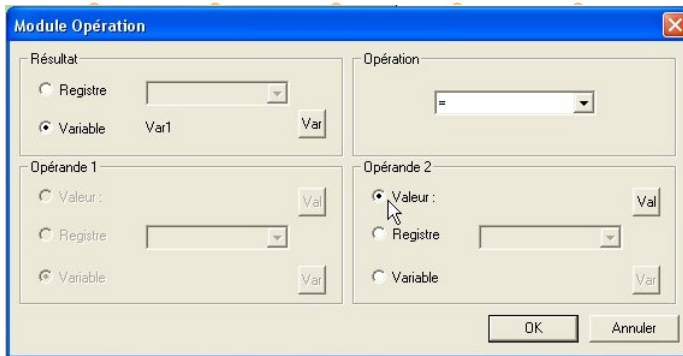
ce qui se comprend par : on additionne 1 à Var1 et on positionne le tout dans la variable Var1.

cela permet d'incrémenter la variable Var1.

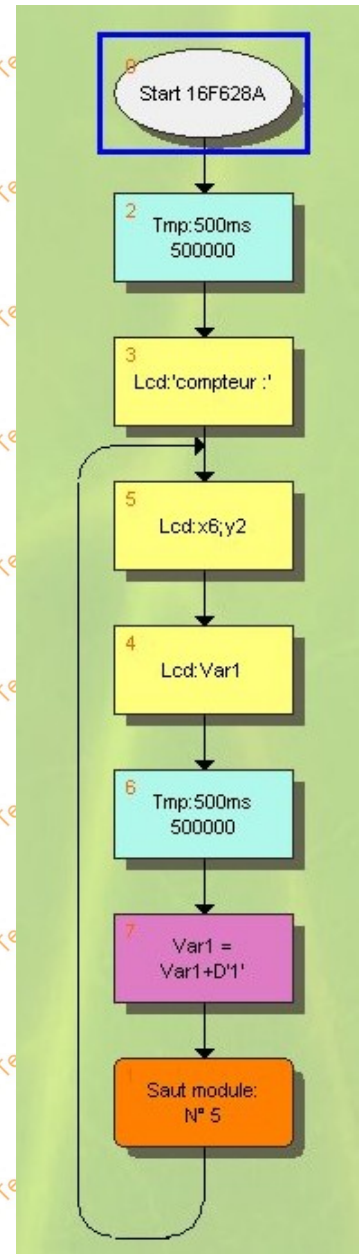
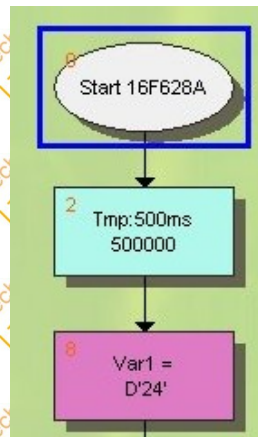
Le programme lcd1.prj permet donc de voir évoluer la variable sur l'afficheur LCD de 0 à 255.

Pour positionner une valeur de départ à la variable Var1, utiliser l'icône :





Cette instruction permet ainsi de positionner la variable Var1 à la valeur décimale 24.



Faire un sablier : compteur 3 minutes

Cahier des charges :

Il s'agit de réaliser un chronomètre de 3 minutes.

Le départ du comptage s'effectue par l'appui du bouton-poussoir branché sur A0.

A la fin des 3 minutes, la sortie A2 est actionnée.

Il est possible d'utiliser cette sortie pour commander un buzzer par exemple.

La lecture de l'organigramme permet de distinguer :

- l'affichage du nom de programme,
- l'attente de l'appui sur A0
- la raz des variables minutes et secondes,
- l'affichage des variables,
- le comptage des secondes et des minutes,
- l'action à réaliser dès la détection du temps 3 minutes

Remarque :

L'utilisation d'une temporisation de 1 seconde est largement suffisante pour un timer 3 minutes.

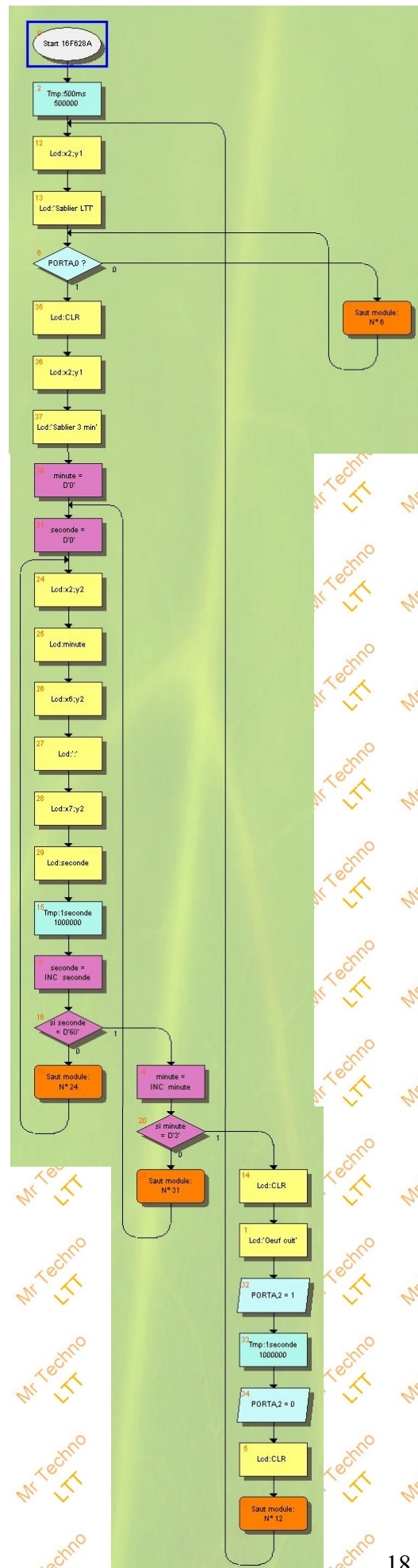
Cependant, les instructions effectuées lors de l'affichage ou des tests introduisent un léger décalage qui ne sont pas supportables pour des applications horaires sérieuses.

Il sera nécessaire d'utiliser un timer.

Sablier1.prj

A partir de cet organigramme, il est possible de :

- de modifier le temps,
- rajouter le clignotement d'une diode led,
- d'effectuer un décomptage des secondes/minutes,
- de modifier l'action à l'issue du comptage,
- de rajouter un bouton-poussoir pour l'arrêt,
- etc...



Ecriture de l'opération Var1 = Inc1

(cette opération est identique à Var1 = Var1 + D'1' mais c'est plus rapide)



Module Opération

Résultat: ☐ Registre ☐ Variable

Opération:

Opérande 1: ☐ Valeur: ☐ Registre: ☐ Variable:

Opérande 2: ☐ Valeur: ☐ Registre: ☐ Variable:

OK Annuler

Variable

Sélection:

Buffer (tableau): ☐ Taille variable ☐ Taille défini:

Var1 (variable simple 8 bits)

OK Annuler

Demander l'opération INC

Module Opération

Résultat: ☐ Registre ☐ Variable

Opération:

Opérande 1: ☐ Valeur: ☐ Registre: ☐ Variable:

Opérande 2: ☐ Valeur: ☐ Registre: ☐ Variable:

OK Annuler

Module Opération

Résultat: ☐ Registre ☐ Variable

Opération:

Opérande 1: ☐ Valeur: ☐ Registre: ☐ Variable:

Opérande 2: ☐ Valeur: ☐ Registre: ☐ Variable:

OK Annuler

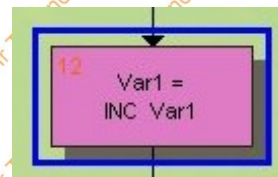
Variable

Sélection:

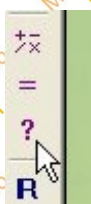
Buffer (tableau): ☐ Taille variable ☐ Taille défini:

Var1 (variable simple 8 bits)

OK Annuler



Ecriture du test de la variable :



Aller chercher Var1

Module test

Registre/Variable 1: ☐ Registre ☐ Variable

Val/Reg/Var 2: ☐ Valeur: ☐ Registre: ☐ Variable:

Test:

Si condition vrai, sortie module: ☐ en dessous ☐ à droite

OK Cancel

Variable

Sélection:

Buffer (tableau): ☐ Taille variable ☐ Taille défini:

Var1 (variable simple 8 bits)

OK Annuler

Demander le test par rapport à la valeur 60

Module test

Registre/Variable 1: ☐ Registre ☐ Variable

Val/Reg/Var 2: ☐ Valeur: ☐ Registre: ☐ Variable:

Test:

Si condition vrai, sortie module: ☐ en dessous ☐ à droite

OK Cancel

Valeur

Valeur:

Base: ☐ Dec ☐ Hex ☐ Bin

Del

OK Annuler

Calculs sur les variables

Une variable est codée sur 8 bits ce qui permet de prendre une valeur décimale de 0 à 255. Le résultat d'une opération est codé sur 8 bits, ce qui provoque parfois des résultats surprenants.

Le dépassement (overflow) s'accompagne de la mise à 1 du bit Carry (retenue) du registre STATUS. Exemples :

Variable1	Variable2	Addition (8 bits)	Carry
250	2	252	0
250	6	0	1
250	8	2	1
250	134	128	1

Il existe une variable particulière gérée par Logpic : V_tmp. Cette variable V_tmp contient le reste d'une division ou le dépassement d'une multiplication. En cas d'opérations successives, il faut bien sûr penser à sauvegarder V_tmp dans une autre variable.

Variable1	Variable2	Multiplication (8 bits)	V_tmp
123	2	246	0
123	3	113	1
123	4	236	1
123	8	216	3
123	9	83	4
123	10	206	4

Explications :
 $123 \times 3 = 369$ soit (V_tmp) $1 \times 256 + 113$
 $123 \times 8 = 984$ soit (V_tmp) $3 \times 256 + 216$
 $123 \times 10 = 1230$ soit (V_tmp) $4 \times 256 + 206$

Variable1	Variable2	Division (8 bits)	V_tmp
123	1	123	0
123	2	61	1
123	4	30	3
123	8	15	3
123	10	12	3
123	124	0	123

Explications :
 $123 / 4 = 30,75$ soit $30 \times 4 + (V_tmp) 3$
 Remarques : lors d'une division par 10, V_tmp contient le reste qui correspond à valeur de l'unité.

Exemples :
 $12 / 10$ V_tmp = 2
 $25 / 10$ V_tmp = 5
 $54 / 10$ V_tmp = 4

Exemple de problème :

Valeur de la variable : 0 -----> 127

Affichage souhaité : 0 -----> 96 ($128 \times 0.75=96$) $\times 0,75$ -----> $x \ 3 / 4$

Le reste de la division par 4 (V_tmp) aura dans ce cas ci une valeur toujours comprise entre 0 et 3. Il faut donc interpréter ce reste comme étant une fraction, cad 1/4, 2/4 ou 3/4

=> V_Source=x (variable de base où x = 0 à 127 mais est valable jusque 255)

=> V_Target=V_Source (On attribue V_Source à la variable de calcul V_Target)

=> V_Résultat=0 (On met la variable de résultat à 0)

=> STATUS,C=0 (On met à 0 le bit Status C)

=> V_Résultat=V_Target/4 (Division par 4)

=> V_Reste=V_tmp (On sauve le reste de la division)

=> V_Reste=V_Reste*3 (Et on le multiplie par 3)

=> V_Reste=V_Reste/4 (Il faut rediviser par 4, V_Reste devient le débordement et V_tmp le nouveau reste)

=> V_Dixieme=V_tmp*25 (On multiplie le nouveau reste par 25 pour avoir la forme décimale)

=> V_Résultat=V_Résultat*3 (On multiplie le résultat par 3)

=> V_Résultat=V_Résultat+V_Reste (Et on ajoute le débordement)

Il suffit alors d'afficher V_Résultat, qui est la partie entière puis V_Dixieme qui est la partie fractionnaire mais sous forme décimale après la virgule.

On pourrait pousser le développement un peu plus loin en effectuant une justification à gauche pour que la partie décimale colle tout juste à la virgule.

Il est souhaitable de créer une variable 8 bits mais à 2 buffer, ce qui revient à avoir une variable 16 bits. Pour la variable <var>, il y a création de « var » et de « var(1) ».