CH-1092 Belmont, Switzerland
Tel +41 21 728-6156, Fax 728-6157
Email info@didel.com, www.didel.com

Logiciels Wellbot

Du point de vue matériel, le Wellbot est un processeur avec des ports d'entrée et de sortie. Le logiciel cache ce niveau physique en déclarant des macros et des routines.

Les entrées logicielles sont :
- **Poussoir** Une routine compte les actions sur le poussoir et permet de choisir ce que doit faire le robot.
- **RS232** Les ordres Hexo permettent une interaction facile avec un clavier de PC ou un programme écrit dans un langave qui permet de communiques avec un port RS323 ou USB/adaptateur série. On peut (prévu) télécharger des petits programmes.
- **Emir** La télécommande infrarouge Emir/Bimotel est décodée et les paramètre sont envoyés au moteur.
- **Distance IR** Les deux capteurs donnent toutes les 20ms des paramètres qui permettent de connaître la dostance à un obstacle
- **Sons** Le microphone est lu par une électronique simplifiés qui ne donne qu'un niveau. Sera amélioré ou plus probablement supprimé.

Les sorties logicielles sont
- **Moteurs** La vitesse des deux moteurs est assignée, et on a en retour la distance parcourue et l'angle
- **Buzzer** Permet quelques sons, et en supprimant les interruptions, on peut programmer des notes et mélodies
- **Led** La fréquence de clignotement dépends d'une variable.
- **Bico64** Extension avec 64 diodes bicolores et un processeur propre

Les interruptions sont de deux types :
- RS232 Les routines Hexo sont activées par Enable:#bHexoOn , ce qui permet d'écrire ses propres programmes sans interruption
- Interruptions du Timer1 pour le clignotement de la LED et les timers, avec en plus
  + L'avance des moteurs si Enable:#bMotOn est activé
  + La mesure de distance des capteurs IR si Enable:#bDistOn et #bEclOn sont activés
  + La télécommande Emir si Enable:#bEmirOn est activé
  + Les bips du buzzer si Enable:#bBuzzOn est activé

Le programmes de test We0x.asm, expliqués plus loin, permettent de comprendre les primitives à disposition.

Documentation

La notice de montage explique le schéma et la liaison à un PC via RS232.

RS232 Pas encore faite spécifiquement pour le WellBot.
Voir www.didel.com/doc/interf/Doc232.pdf et
Avec le processeur à 4 MHz, la vitesse est limitée à 19600 b/s. La vitesse par défaut dans les programmes est de 9600 b/s. Le programme terminal recommandé est Ttrempro
Le chargeur de fichiers .hex est limité à 2400 b/s.

Logiciel

La dernière mise à jour du logiciel se trouve sur www.didel.com/wbot/WbSoft.zip
Le processeur est programmé avec le programme We11.asm, qui est le programme complet prévu pour être utilisé avec un programme sur PC interagissant avec les ordres Hexo. Voir WbHexo.pdf + WbResume.pdf

Tests
We01.asm Test initial robot clignote, active tous les bits
We02.asm Test poussoir compte sur portD si 877, LedOn si impair
We03.asm Idem avec fichier inséré WePous.asi clignote le nombre de fois et se bloque
We04.asm Accélère les 2 moteurs. Test l'interruption et les tâches moteur
We05.asm Copie le bitmap vers Bico64. Permet de se familiariser
avec le format sur l'écran en modifiant la
table Motif: (voir www.didel.com/??/Bico64
We06.asm Test RS232 echo Hexo taper un nombre hexa et terminateur
We07.asm Test Bico64 via un clavier série (a finir)
We08.asm Test chargeur (a finir)
We09.asm Test timers, clignotement et buzzer
We0A.asm Test capteurs de distance (a finir)
We0B.asm Test commande infrarouge Emir et démos de mouvement

Pour comprendre le logiciel, le mieux est de charger et modifier ces 11 programmes qui ont servi à la mise au point des modules de logiciel.

We01.asm|Test initial robot
\b;Clignote les Led et tourne les moteurs (simplifié)
\b;Pulse le transistor et le HP
\b;Envoie un + sur le série à 9600 bits/s et fait l'echo des car reçus
; - ! Définir T870=0 et T882=1 si vous avez un 16F882 ! -
Ce programme permet de voir comment on définit et initialise les ports. Il importe les 4 fichiers que l'on retrouvera partout:
WeSet.asi les définitions des constantes de tous les programmes
WeDef.asi les définitions des entrée-sorties du 16F870/882
WeVar.asi les variables de tous les programmes
WeInit.asi l'initialisation et des routines simples
Le programme clignote la LED, envoie un caractère, fait tourner les moteurs. Il faut comprendre cette programmation avant d'aller plus loin. Les boucles d'attente, les tables sont expliquées dans différents documents :
http://www.didel.com/dev877/ et http://www.didel.com/Dauphin3.pdf

Si les fichiers insérés vous troublent, chargez la version **We01a** (ou We01a8842 si vous avez un 882**)**, selon le processeur que vous avez. Ces programmes contiennent le minimum de déclarations et variables. Faites-vous des variantes de ce programme en rajoutant des clignotements astucieux dans un programme, un moteur qui va dans les deux sens dans un autre programme, etc.
Si vous revenez à la version We01, vous comprenez mieux l'avantage des .If (le même programme pour le 877 et le 882) et la meilleure lisiblilté en insérant les fichiers sur lesquels il n'y a plus besoin de réfléchir.

Routines dans WeInit.asi
Cligno \in ; W nombre de clignotement
Sirene pas de paramètre
SndSer \out ; W caractère envoyé
GetSer \in; W caractère reçu du clavier

We03.asm| Test lecture poussoir, comptage des impulsions
; Affiche comptage sur portD, 877 seulement
; Invese la Led à chaque pression rapprochée
; Clignote le nb de fois et recommence
Ce programme teste la lecture des impulsions sur le poussoir. C'est un peu plus complexe car le poussoir partage le même bit que la LED. Si on presse sur le poussoir, on allume nécessairement la LED. Le logiciel rétablit la valeur voulue au relâchement.

Routines dans WePous.asi
GetPous \out: CntPous

Attend un nombre d'actions sur le poussoir. Retour au programme appelant par manque d'action pendant 1 sec
Le Led est allumée pour les valeurs impaires si on a mis FreqLed=0
En général au retour de la routine (1 seconde sans pression), on décompte CntPous pour s'aiguiller (qq pressions) ou on passe par une table

[pour un spécialiste : La LED est multiplexée avec le poussoir. Si on agit sur le poussoir, la LED s'allume nécessairement. Si on n'agit pas sur le poussor, la LED dépends de PortC:#bLed. Les macros LedOn LedOff sont utilisées de préférence.
La lecture du poussoir est eu peu plus compliquée. Il faut
- couper les interruptions (agissent aussi sur le portC)
- sauver l'état de la LED (allumée ou éteinte)
- mettre la sortie à zéro pour avoir un bon zéro au cas ou le poussoir n'es pas activé
- commuter le port en entrée
- lire si c'est un 1 (pressé) ou zéro (relâché)
- remettre le port en sortie
- rétablir l'état de la LED
- rétablir l'état des interruptions
Les macros SkipIfPousOn et SkipIfPousOff (dans WeDef.asi) font tout ce travail (18 instructions, donc 18 us).

We04.asm| Accelere le moteur

; accélère jusqu'à VitD=VitG = 63 et recommence
; Test routine Step par interruption timer 1
; LedOn si bit 3 vitesse à 1
; Pulse bS1 = RC5 durée tâche DoStep durée 30-40 us, max 60 us

Deux fichiers sont insérés : WeMotT.asi doit être dans la page des tables et contient quatre tables beaucoup plus astucieuses que les tables du programme We01. Ces tables calculent la distance parcourue et la différence entre les deux moteurs (donc l'angle).
On active le service du moteur dans la routine d'interruption en activant Enable:#bMotOn.
Variables : \out; VitD VitG \in; DistHi DistLow Angle

We05| Test l'envoi d'une image en série sur Bico64

; Ck RA5 Data RA2 Datain RA4
; il faut CkOn >120 us CkOff > 200
; interférence avec les poussoirs de l'affichage

Le programme envoie en permanence (toutes les 20 ms) le contenu de la table Motif mise en mémoire programme. L'exercice est donc de changer cette table et de voir l'effet.
On se référera à la figure dans www.didel.com/WeBico.pdf. Pour tester les ordres et modifier plus facilement, voir We08.asm.

Routines dans WeBico.asi

MotifToBitMap \in; table Motif: \out; bloc variables BitMap
SendBitmap \in; W BitMap envoi en série via RA5 3 2
\out; Data poussoirs de bimo sur bits 3 et 5

We06|Test RS232 Hexo

; 9600 b/s changer dans WeDef.asi
, on tape des chiffres hexa et une lettre G .. Z
; Hexo envoie en retour la valeur SavBin et l'ordre
; La lettre G correspond à la valeur SavOrder= 0

Hexo doit être activé par Enable:#bHexo et il ne faut alors plus utiliser les routines simples SndSer et GetSer.
Hexo est plein de bits de mode pour par exemple passer à la ligne après chaque ordre.
Voir We08 et We0A pour un exemple d'utilisation
Voir aussi www.didel.com/wellbot/Resume.pdf

We07|Test ecran avec Hexo
; Terminal à 9600 bits/s pour envoi ordre
; J efface
; G set bit v0yyyxxx ou 0ryyyxxxG
; H inverse

We08| Chargeur en page 0
pas encore fait – m^me programma que dans Dev877 mais hésitation sur l'emplacement mémoire.
Transfert à 2400 b/s on ne peut pas changer.

We09|Test timers, clignotement et buzzer
; Cligno selon #IniFreq (impulsions 2ms)
; Compte les actions sur le poussoir et lance le buzzer
; Décompte les timers si diff de zéro (pas d'action)
La variable FreqLed fixe la fréquence du clignotement Zero pas de clignotement 5 période de qq secondes, 100 très rapide. Le programme est (sera) expliqué dans Dauphin3.pdf.
Le buzzer joue 7 bips et petites séquences si Enable :#BuzzOn est activé. Il faut charger la valeur 1, 2 , 8 (pas plus) dans TkBuz et si on veut enchaîner, attendre que TkBuz pass à zéro.

We0A.asm|Test capteur de distance IR
; Test Capteur distance, affiche sur PortB si 877
; Clignote selon distance, capteur droite si poussoir actif
Adapté du Didelbot, Pas encore vérifié et ajusté les paramètres

We0B.asm| Demo de mouvement et télecommande par Emir-BimoTel
Le Wellbot se comporte comme le Bimo (télécommande ou demos si actions sur le poussoir).
A l'enclenchement, après la sirène, le Wellbot attend un signal infrarouge ou une action sur le poussoir. Les démos 1 et 3 doivent être interrompues en coupant l'alimentation. A la fin des démos 2 et 4, on se retrouve comme à l'enclenchement.

Fichiers insérés

wedef.asi	2'441
weinit.asi	1'731
weinter.asi	2'455
weirr.asi	4'605
weirt.asi	465
wemotr.asi	762
wemott.asi	1'197
wepous.asi	1'066
weset.asi	2'754
wevar.asi	2'052
pas utilisé dans We0B	
webico.asi	1'752
wedist.asi	649
wetsr.asi	1'274
wetst.asi	269

jdn 090915/091203