ENSIAME

Ingéniérie Microcontrôleur

Un système incluant un microcontrôleur fait interagir « du logiciel » sur « du matériel » (le µC) en vue d'un objectif codé dans le logiciel. Les éléments constitutifs d'un tel système sont : le µC et ses périphériques, le programme, les données d'entrées / sorties et les actions. L' ingénierie microcontrôleur consiste à organiser ces systèmes et ordonner leur développement spécifique.

L'objectif de ce polycopié est de présenter les concepts associés à la mise en œuvre des microcontrôleurs : donner les pré requis permettant d'analyser et de concevoir des éléments matériels (structuration, organisation) et logiciels (programmation) de systèmes bâtis autour des microcontrôleurs. Les exemples d'illustration se basent sur le noyau du 8051.

Ce document à considérer comme un aide mémoire vous sera utile pour l'organisation des travaux pratiques et pour développer les codes qui vous seront demandés. Son organisation est donnée ci-dessous :

I. Quelques éléments de méthodologie :

I.1 Analyser le système :

Analyser chaque fonction du système en liaison avec le cahier des charges : lister les paramètres du système (valeurs extrêmes, résolution, …).

- Tracer le diagramme fonctionnel du système.

Associer à chaque fonction un sous-système (capteur, actionneur, …) et lister les différentes options technologiques envisageables comme autant de choix possibles, choisir en fonction du cahier des charges.

- Tracer le diagramme matériel.

I.2 Définir les paramètres d'entrée et de sortie du micro contrôleur :

Analyser les signaux qui relient la logique câblée à la logique programmée (internes au µC) à partir des diagrammes fonctionnels ou structurels : faire un diagramme représentant les structures matérielles situées à la frontière entre la logique câblée et la logique programmée (Ex.: TIMER, PORT E/S, IRQ, RESET, etc ...). NB : pour chaque signal ou ensemble de signaux (BUS) caractériser leurs paramètres électriques: U, I, Ze, Zs, etc... Eventuellement leurs représentations temporelles.

- Etablir la liste des paramètres d'entrées, celle des paramètres sorties : information de type logique / analogique ou ensemble d'informations (liaisons séries et liaisons parallèles). Pour chaque paramètre définir : son nom, sa taille (bits, octets), son type (Entier, Réel, Caractère, Chaîne, ...), son mode de représentation (Binaire, BCD, Hexadécimal, ASCII ...). Ex. : bouton poussoir « monter », logique, 1 bit, binaire, 0 / 3 V, ! rebonds / 1 ms, …

I.3 Organiser les programmes :

- Tracer l'organigramme du programme principal représentant l'enchaînement de toutes les actions réalisées.

Remarques : Les actions définies dans l'organigramme sont les actions principales réalisées par le système, elles doivent être traduites par des verbes d'actions (Avancer, Calculer, Positionner, Tester, Sauvegarder .....). Si le système répond a plusieurs événements (i.e. : interruptions), il faut faire un organigramme par événement.

I.4 Définir les variables globales :

En plus des paramètres d'entrée et sortie (I.2), il faut définir des variables nécessaires au déroulement des actions, les définir à partir des organigrammes. Pour chaque variable définir : son nom (Identificateur), sa taille (Nombre de bits ou d'octets), son type (Entier, Réel, Caractère, Chaîne de caractères, Tableau...), son mode de représentation (Binaire, BCD, Hexadécimal, ASCII ...).

I.5 Tableau des fonctions logicielles:

L'organigramme (du programme principal) est structuré en actions, le programme principal sera lui structuré en fonctions logicielles (ou sous-programmes) qui sont déterminés par rapport aux différentes actions. Lister les

appels des différentes fonctions logicielles, les échanges d'informations (variables globales et locales, paramètres d'entrées/sorties). Faire de même pour les autres organigrammes.

Faire un tableau de toutes les fonctions logicielles, pour chaque fonction indiquer : le nom, toutes les opérations réalisées par celle-ci, les paramètres d'entrées et de sorties, les variables.

I.6 Programmation, logiciels :

Il existe de nombreuses solutions pour programmer les fonctions qui ont été développées en organigrammes afin de les traduire dans un langage informatique. Nous utiliserons Keil, l'une des références dans ce domaine qui permet de programmer en assembleur et en C (d'autres utilisent aussi le basic). Tester chaque fonction indépendamment puis assembler les différentes fonctions jusqu'à la réalisation complète de l'organigramme du programme principal. Assurer l'implémentation en fonction des ressources du microcontrôleur (mémoire, …).

II. Principales caractéristiques du 8051 :

II.1 Les interruptions :

Il y a 5 sources d'interruption dans un noyau 8051, (le 8052 en comporte 6 et les 80C535/7, 12).

- Table des sauts vectorisés :

Source d'interruption	Indicateur	Adresse du vecteur
INT0 ; P3.2	IE0	0003H
TIMER0 ; P3.4	TF0	000BH
INT1 ; P3.3	IE1	0013H
TIMER1 ; P3.5	TF1	001BH
PORT SERIE	RI+TI	0023H
TIMER2 (8052)	TF2+EXF2	002BH

Et le Reset (adresse 00H) : il n'y a que 3 octets pour le reset, soit la taille pour mettre une instruction du type LJMP. Les interruptions sont configurables à l'aides des registres IE, IP, et TCON.

- Registre IE, Interrupt Enable, (A8H) : autorisation des interruptions [EA – ET2 ES ET1 EX1 ET0 EX0]
Pour autoriser une interruptions : mettre à 1 le bit correspondant du registre IE.

position	symbole	fonction
IE.7	EA	Inhibition de toutes les interruptions
IE.6	---	
IE.5	ET2	interruption TIMER2 (8052)
IE.4	ES	interruption PORT SERIE
IE.3	ET1	interruption TIMER1
IE.2	EX1	interruption EXTERNE (INT1)
IE.1	ET0	interruption TIMER0
IE.0	EX0	interruption EXTERNE (INT0)

- Registre IP, Interrupt Priority, (B8H) : priorité des interruptions [- - PT2 PS PT1 PX1 PT0 PX0]

position	symbole	fonction
IP.7	---	
IP.6	---	
IP.5	PT2	priorité TIMER2 (8052)
IP.4	PS	priorité PORT SERIE
IP.3	PT1	priorité TIMER1
IP.2	PX1	priorité interruption externe INT1
IP.1	PT0	priorité TIMER0
IP.0	PX0	priorité interruption externe INT0

Il y a une hiérarchie naturelle des interruptions imposée par la séquence de test soit : IE0 ; TF0 ; IE1 ; TF1 ; RI+TI ; (et TF2+EXF2 pour le 8052). Il n'est pas nécessaire de configurer IP pour obtenir la hiérarchie naturelle.

- Registre TCON, Timer CONtrol, (88H) : [TF1 TR1 TF0 TR0 IE1 IT1 IE0 IT0]

symbole	fonction
IE1	Mis à 1 lors d'une transition sur la broche INT1. Remis à 0 pendant l'interruption
IT1	Si IT1 = 0 interruption si niveau 0 sur la patte INT1 Si IT1 = 1 interruption lors d'un front descendant sur la patte INT1
IE0	même fonctionnement que l'indicateur IE1 pour INT0
IT0	même fonctionnement que le bit de configuration IT1 pour INT0

Quatre bits concernent les interruptions Timer (bits grisés, cf. paragraphe suivant).

II.2 Les « Timers » :

Le 8051 est doté de deux timers 16 bits qui se présentent chacun sous forme de deux octets : TH0, TL0 et TH1, TL1. Un timer peut assurer deux fonctions : compteur d'évènements (incrémenté à partir d'évènements extérieurs sur les broches P3.4 et P3.5) ou compteur d'unité de temps (incrémenté à partir d'un signal d'horloge). Les 2 timers sont configurables à l'aide des deux registres TMOD et TCON, Timer MODe et Timer CONtrol.

- Registre TMOD (89H) : contrôle du mode [GATE C/T M1 M0 GATE C/T M1 M0]

Le quartet de poids fort de ce registre s'adresse au Timer 1 et celui de poids faible au Timer 0.
GATE = 1 & INTx = 1 & TRx = 1 ----> TIMERx validé.
GATE = 0 & TRx = 1 ----------> TIMERx validé.
C/T = 0 ----------> fonction compteur de temps (incrémenté à chaque cycle machine)
C/T = 1 ----------> fonction Compteur à partir des évènements sur broche Tx (front descendant)

Choix du mode de fonctionnement :

M1	M0	Mode
0	0	0 : compteur 13 bits
0	1	1 : compteur 16 bits
1	0	2 : compteur 8 bits à rechargement auto. Si TLx passe à 0, il est rechargé avec THx
1	1	3 : TIMER0 : TL0 compteur 8 bits contrôlé par les bits de contrôle de TIMER0 et TH0 devient temporisateur 8 bits contrôlé par les bits de contrôle de TIMER1

1	1	3 : TIMER1 est alors à l'arrêt.

- Registre TCON (88H) : [TF1 TR1 TF0 TR0 IE1 IT1 IE0 IT0]

TF1 : indicateur de débordement du TIMER1. Il est mis à 1 lorsque le compteur passe à 0. Si l'interruption lui correspondant est validée, il est remis à 0 automatiquement lorsque le sous-programme d'int. est exécuté.
TR1 : bit de déclenchement du TIMER1. Le mettre à « 1 » pour valider le TIMER1 (setb TR1).
TF0 : indicateur de débordement du TIMER0. Même fonctionnement que TF1.
TR0 : bit de déclenchement du TIMER0. Même fonctionnement que TR1.

II.3 Transmission port Série :

Le port série est un port full-duplex, c'est à dire qu'il peut fonctionner en émission et réception simultanées. Il possède un tampon de réception d'un octet. Le registre de réception et d'émission est le même : SBUF (099H), une écriture à cette adresse provoque une émission et une lecture, une réception. Le registre SCON permet le contrôle de l'interface de communication série. Connexions : broche P3.0 pour RxD et broche P3.1 pour TxD.

Il y a 4 modes de fonctionnement :

- **mode 0** : Les données séries sont véhiculées via la ligne RxD. L'horloge est sortie par TxD. la transmission (I ou O) se fait en 8 bits en commençant par celui de poids le plus faible. La vitesse (baud rate) est de 1/12 de la fréquence d'oscillateur.
- **mode 1** : 10 bits sont transmis via TxD ou reçus via RxD:1 bit de start, 8 bits de données, 1 bit d'arrêt. A la réception le stop bit est envoyé dans le registre RB8 de SCON. La vitesse est variable.
- **mode 2** : 11 bits sont transmis ou reçus, c'est à dire qu'en plus du cas précédent il y a un 9ème bit de donnée programmable (TB8 de SCON) qui peut être un bit de parité (le P de PSW) à la transmission, à la réception ce bit est mis en RB8 de SCON et le bit de stop est ignoré. La vitesse est 1/32 ou 1/64 de la fréquence de base.
- **mode 3** : Identique au mode 2, mais la vitesse est programmable.

Dans tous les modes la réception est initiée par REN=1, en mode 0 il faut en plus RI=0.

- Registre SCON (098H) : [SM0 SM1 SM2 REN TB8 RB8 TI RI]

SM0	SM1	Mode	Description	fréquence de communication
0	0	0	registre à décalage	fosc / 12
0	1	1	UART 8 bits	variable
1	0	2	UART 9 bits	fosc/64 ou fosc/32
1	1	3	UART 9 bits	variable

SM2 : En mode 2 ou 3, si SM2=1, alors RI activé que si le 9e bit est à 1. En mode 1, si SM2=1, alors RI ne sera activé que si le bit de stop est valide. En mode 0, SM2 doit être à 0.
REN : autorisation de la réception série (valide à 1).
TB8 : 9e bit à transmettre en mode 2 ou 3.
RB8 : 9e bit reçu en mode 2 ou 3. En mode 1, correspond au bit de stop si SM2=0
TI : indicateur d'émission. Passe à 1 à la fin d'une trame. Doit être remis à 0 par le programme.
RI : indicateur de réception. Passe à 1 à la fin d'une trame. Doit être remis à 0 par le programme.
N.B. : UART pour Universal Asynchronous Receiver/Transmitter

Communication multiprocesseurs :

Les modes 2 et 3 sont spécifiquement conçus pour le fonctionnement multiprocesseur via le 9ème bit. En effet quand le processeur maître veut transmettre à l'un de ses esclaves, il commence par envoyer une adresse qui définit le récepteur. Le 9ème bit est alors utile : il sera mis à 1 pour une adresse et à 0 pour une donnée, ceci permet la différenciation entre donnée et adresse.

On procède de la façon suivante : le 9ème bit va dans RB8. Il est possible de programmer tout port de réception de telle sorte que l'interruption correspondante ne soit activée que si RB8 = 1 (on peut le définir ainsi en plaçant dans RB8 le contenu du bit SM2 qui appartient à SCON et est à 1 en mode 2 ou 3). Quand le processeur maître enverra un bloc adresse, tous les ports en réception sont actifs et vont donc pouvoir identifier le bloc reçu et le comparer avec leur propre adresse, tous ceux pour lesquels cette comparaison donnera un résultat de non identité resteront inchangés tandis que le vrai destinataire passera son bit SM2 à 0 ce qui alors autorise l'entrée des blocs de données qui vont suivre dans le buffer de ce seul processeur esclave.

Les TIMERS et le port série :

Le TIMER 1 peut être utilisé comme horloge. Il doit être configuré en compteur 8 bits à rechargement automatique. La fréquence de communication (FC) est alors :

FC = (K * fosc) / (32 * 12 * (256 - TH1)) ; avec fosc = fréquence de l'oscillateur du micro et K = 1 si SMOD = 0 ou K = 2 si SMOD = 1. NB : SMOD est un bit du registre PCON[1], adressable directement, par défaut à 0.

Tableau des fréquences de communication :

Fréquence de communication	Fréquence de l'oscillateur	SMOD	TH1
19200	11.059 MHz	1	FDH
9600	11.059 MHz	0	FDH
4800	11.059 MHz	0	FAH
2400	11.059 MHz	0	F4H
1200	11.059 MHz	0	E8H
300	6 MHz	0	CCH
110	6 MHz	0	72H
31250 (MIDI)	12 MHz	0	FFH

II.4 Les registres spéciaux (SFRs) :

1) accumulateur : Il est référencé par le mnémonique ACC sur le tableau, cependant dans les instructions de programmation on le désigne par A
2) registre B : Il est utilisé tout particulièrement pour l'exécution des multiplications et divisions. Mais il peut être considéré aussi comme un registre quelconque.
3) PSW : program status word.
4) stack pointer : Le pointeur de pile est incrémenté avant l'exécution d'un PUSH ou d'un CALL. Bien qu'on puisse a priori mettre la pile n'importe où, lors d'un reset le pointeur est initialisé à 07H, et par conséquent la pile débute à l'adresse 08H.
5) data pointer : Le pointeur de données est un registre en deux parties DPH pour la partie haute et DPL pour la partie basse de l'adresse. On peut manipuler séparément ses deux moitiés, ou les considérer comme un ensemble 16 bits.
6) ports 0 à 3 : P0, P1, P2 et P3 sont les latches des ports correspondants.

7) buffer de données série : Le buffer de données série est lui aussi un double registre l'un pour la transmission, l'autre pour la réception, mais c'est transparent car il n'y a qu'une adresse : SBUF.
8) registres des timers : les registres TLx et THx (avec x variant de 0 à 2) constitue les registres de comptages sous 16 bits des timers correspondants.
9) registres de capture : La paire de registres RCAP2 sont réservés au timer 2 et permettent sous certaines conditions une recopie de sauvegarde de TH2 et TL2. Ils n'existent pas systématiquement.
10) registres de contrôle : Les registres spécialisés IP, IE, TMOD, TCON, T2CON, SCON et PCON contiennent les mots de statut pour le système d'interruption, les timers et le port série.

Tableau des registres du 8051 : la colonne d'extrême gauche et celle de droite indiquent les limites d'adresses du tableau, ainsi le registre SP est à l'adresse 81h et PCON à l'adresse 87h.

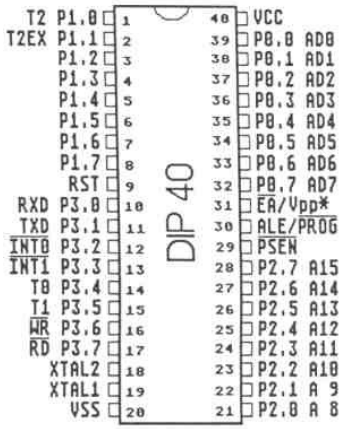
F8								FF
F0	B							F7
E8								EF
E0	ACC							E7
D8								DF
D0	PSW							D7
C8	T2CON		RCAP2L	RCAP2H	TL2	TH2		CF
C0								C7
B8	IP							BF
B0	P3							B7
A8	IE							AF
A0	P2							A7
98	SCON	SBUF						9F
90	P1							97
88	TCON	TMOD	TL0	TL1	TH0	TH1		8F
80	P0	SP	DPL	DPH			PCON	87

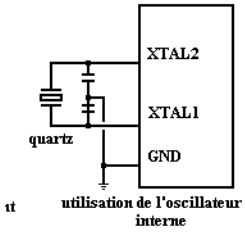
II. 5 Les ports particuliers :

Les fonctions des ports particuliers ne sont activées que si les ports sont activés en lecture (le bit correspondant dans le latch du port concerné est activé à 1 par le programme).

port-pin	fonction
P1.0	timer 2 entrée externe T2 (dans 8052)
P1.1	timer 2 trigger de capture T2EX (dans 8052)
P3.0	entrée du port série RXD
P3.1	sortie du port série TXD
P3.2	interruption externe INT0
P3.3	interruption externe INT1
P3.4	entrée externe du timer 0 T0
P3.5	entrée externe du timer 1 T1
P3.6	write strobe pour la mémoire externe WR
P3.7	signal read pour la mémoire externe RD

II.6 Le brochage :





Les broches aux fonctions particulières :

- Entrée /EA : (**E**xternal **A**ccess) si EA=0, les instructions sont recherchées dans la mémoire programme externe.
- RST : Entrée d'initialisation. Un état haut pendant deux cycles machines sur cette broche entraîne une initialisions du microcontrôleur.
- Sortie /PSEN : (**P**rogramm **S**tore **EN**able) passe à 0 lorsque le micro va rechercher une instruction en mémoire programme externe.
- Sortie ALE : (**A**dress **L**atch **E**nable) prévue pour commander le démultiplexage du port P0. Si ALE est à 1, P0 présente la partie A0 à A7 du bus d'adresse et si ALE est à 0, P0 sert de bus de donnée. Pour mieux comprendre, se reporter à l'organisation du bus.
- XTAL1 et XTAL2 : Placer le quartz entre ces deux broches avec deux condensateurs de 22pF entre ces deux broches et la masse.
- P0.0 à P0.7 : 8 lignes du port P0 du type "à drain ouvert". Si ces lignes sont utilisées en sortie, il est nécessaire de le doter de résistances de rappel.
- P1.0 à P1.7 : Port bidirectionnel avec résistances de rappel au +5V intégrées.
- P2.0 à P2.7 : Idem que port P1 sauf : fonction secondaire du port: adresses de A8 à A15.
- P3.0 à P3.7 : Idem que port P1 sauf : fonctions secondaires.

III. Les instructions du 8051 :

III.1 Symboles utilisés:

Rn	Un des registres actifs R0 à R7
direct	Adresse d'un octet de RAM interne, d'un port, ou SFR
@Ri	Adressage indirect par registre R0 ou R1
#data	Donnée 8 bits
#data16	Donnée 16 bits
bit	Adresse au niveau du bit
rel	Adresse relative au PC en complément à 2 de -128 à +127
addr11	Adresse limitée au bloc de 2Ko dans lequel figure l'instruction
addr16	Adresse sur l'espace de 64Ko

III.2 Jeu d'instructions :

Ecriture / lecture :

MOV : (MOV <dest>, <source>)

Mnémonique	Syntaxe
MOV	A, Rn
MOV	A, direct
MOV	A, @Ri
MOV	A, #data
MOV	Rn, A
MOV	Rn, direct
MOV	Rn, #data
MOV	direct, A
MOV	direct, Rn
MOV	direct, direct
MOV	direct, @Ri
MOV	direct, #data
MOV	@Ri, A
MOV	@Ri, direct
MOV	@Ri, #data
MOV	DPTR, #data16

MOV : (MOV <bitdest>, <bitsrc>) copie le bitsrc dans le bitdest

MOV	bit, C
MOV	C, bit

MOVC : (MOVC A, @A+<base-reg>) permet de lire un octet dans la mémoire pgm.

MOVC	A,@A+DPTR
MOVC	A,@A+PC

MOVX : (MOVX <dest>, <source>) permet la lecture ou l'écriture d'un octet en RAM externe.

MOVX	A,@Ri
MOVX	A,@DPTR
MOVX	@Ri,A
MOVX	@DPTR,A

Opérations arithmétiques / logiques :

SETB : (SETB <bit>) met à 1 un bit

SETB	bit
SETB	C

CLR : (CLR <A/bit>) met A ou un bit à 0

CLR	A
CLR	bit
CLR	C

INC : (INC <byte>) incrémente un octet ou DPTR.

INC	@Ri
INC	A
INC	direct
INC	Rn
INC	DPTR

DEC : (DEC <byte>) décrémente un octet.

DEC	@Ri
DEC	A
DEC	direct
DEC	Rn

ADD : (ADD A, <byte>) additionne un octet et l'accu A, résultat dans A.

ADD	A,Rn
ADD	A, direct
ADD	A, @Ri

ADD	A, #data

ADDC : (ADDC A, <byte>) additionne un octet, l'accu A et la retenue, résultat dans A.

ADDC	A, Rn
ADDC	A, direct
ADDC	A, @Ri
ADDC	A, #data

SUBB : (SUBB A, <byte>) soustrait un octet ainsi que la retenue au contenu de A, résultat dans A.

SUBB	A, Rn
SUBB	A, direct
SUBB	A, @Ri
SUBB	A, #data

MUL : (MUL AB) multiplie l'accu A et le registre B, résultat : octet faible dans A et octet fort dans B.

DIV : (DIV AB) divise le contenu de A par le contenu de B, quotient dans A et reste dans B.

ANL : (ANL <dest>, <source>) réalise un ET logique entre source et dest, résultat dans dest.

ANL	A, #data
ANL	A, @Ri
ANL	A, direct
ANL	A, Rn
ANL	direct, #data
ANL	direct, A
ANL	C, /bit
ANL	C, bit

ORL : (ORL <dest>, <source>) réalise un OU logique entre source et dest, résultat dans dest.

ORL	A, #data
ORL	A,@Ri
ORL	A, direct
ORL	A, Rn
ORL	direct, #data
ORL	direct, A
ORL	C, /bit
ORL	C, bit

XRL : (XRL <dest>, <source>) réalise un OU exlusif logique entre source et dest, résultat dans dest.

XRL	A, #data
XRL	A, @Ri
XRL	A, direct
XRL	A, Rn

XRL	direct, #data
XRL	direct, A

CPL : (CPL <A/bit>) complémente A ou un bit

CPL	A
CPL	bit
CPL	C

DA : (DA A) ajustement décimal de A.

RL : (RL A) rotation vers la gauche du contenu de A

RLC : (RLC A) rotation vers la gauche du contenu de A+retenue

RR : (RR A) rotaion vers la droite du contenu de A

RRC : (RRC A) rotation vers la droite du contenu de A+retenue

SWAP : (SWAP A) échange le quartet de poids faible avec celui de poids fort de A

RL	A	23	1	1
RLC	A	33	1	1
RR	A	03	1	1
RRC	A	13	1	1
SWAP	A	C4	1	1

PUSH et POP permettent respectivement de sauvegarder sur la pile ou d'y récupérer des données.

PUSH	direct	C0	2	2
POP	direct	D0	2	2

XCH : (XCH A, <byte>) échange les données de l'accumulateur A et de l'octet adressé.

XCH	A, Rn	C8+n	1	1
XCH	A, direct	C5	2	1
XCH	A, @Ri	C6+i	1	1

XCHD : (XCHD A, @Ri) échange les quartets de poids faible entre l'accu A et l'octet adressé.

XCHD	A, @Ri	D6+i	1	1

NOP : pas d'opération

Jonction / sauts :

ACALL : réalise un saut absolu inconditionnel

LJMP : réalise un saut long inconditionnel

SJMP : réalise un saut court par adressage relatif

JMP : réalise un saut indirect

AJMP	addr11	E1	2	2
LJMP	addr16	02	3	2
SJMP	rel	80	2	2
JMP	@A+DPTR	73	1	2

JZ : saut si A=0

JNZ : saut si A<>0

JC : saut si retenue à 1

JNC : saut si retenue à 0

JB : saut si bit à 1

JNB : saut si bit à 0

JBC : saut si le bit est à 1 et mise à zero de celui-ci

JZ	rel	60	2	2
JNZ	rel	70	2	2
JC	rel	40	2	2
JNC	rel	50	2	2
JB	bit, rel	20	3	2
JNB	bit, rel	30	3	2
JBC	bit,rel	10	3	2

CJNE : (CJNE <byte1>, <byte2>, <rel>) saut si byte1 et byte2 sont différents

CJNE	@Ri, #data, rel	B6+i	3	2
CJNE	A, #data, rel	B4	3	2
CJNE	A, direct, rel	B5	3	2
CJNE	Rn, #data, rel	E8+n	3	2

DJNZ : (DJNZ <byte>, rel) décrémente byte et saut si résultat différent de 0

DJNZ	direct, rel	D5	3	2
DJNZ	Rn, rel	D8+n	2	2

IV. Utilisation du logiciel Keil µVision 3 :

Vous trouverez ci-dessous les instructions pour créer un projet à l'aide de ce logiciel, créer un code et le simuler. Sauvegarder dans Mes documents/Mecatro/nom_du_binome

- Créer un nouveau projet (« *Project* », « *new* », « *µvision project* »)

Après le choix du nom de projet et de l'emplacement pour la sauvegarde, l'onglet « *Select device* … » apparaît. Il permet de choisir le type de microcontrôleur qui va être programmé. Pour le projet il convient de sélectionner : « *Atmel* », puis, « *89C51ED2* » et « OK ». Le logiciel suggère ensuite d'inclure un code d'initialisation au projet : cliquer sur NON. Le projet est créée et son nom apparaît en haut à gauche.

- Créer un fichier de programmation (« *File* », « *new* »)

Il faut enregistrer le fichier : (« *File* », « *Save as* »). Par défaut l'emplacement proposé pour le sauvegarde est celui du projet. Donner un nom **en ajoutant l'extension « .a »**. (Taper : nom_ficher.a ; puis « *save* »). Il convient ensuite d'ajouter le fichier qui vient d'être créée au projet. Dans l'onglet « Project *Workspace* » à gauche, cliquer sur le « + » du répertoire « *Target 1* », le répertoire « *Source group 1* » apparaît, un clic droit sur ce répertoire fait apparaître un menu dans lequel on choisit « *Add files to group …* ». (N.B. : plus tard pour la compilation on choisira « *Rebuild all target files* » dans ce menu). On choisit le type de fichiers « *.s, .src, .a* ». On clique sur le nom du fichier que l'on vient de créer, puis « *Add* » et « *Close* ».

- Programmation

Pour le projet, la programmation se fait en assembler pour le 8051. Le compilation reconnaît les instruction du 8051 : les étiquettes, les symboles, les indicateurs, les mnémotechniques des registres particuliers… (ex. : les instructions *mov A,#0Fh ; inc R1* ou *mov IE, #10000001B* , etc…, sont tout à fait reconnues). Se reporter au polycopié du cours « Intégration microcontrôleur » pour des exemples.

Quand le fichier est prêt (ou après chaque modification avant une nouvelle simulation), il faut compiler le fichier, cf. le N.B. ci-dessus.

- Simulation du code

C'est le menu « *Debug* », « *Start stop debug* » qui permet la simulation. Le logiciel vous avertit que la simulation sera faite avec une version bridée du programmateur à 2k de code, cliquer « *OK* ». A l'aide du menu « *Perpherials* » ouvrir les fenêtres de dialogue des ports concernés par votre projet (ex. : « *I/O-ports* », « *Port 3* », pour avoir accès aux broches des interruptions P3.2 et P3.3). Ces fenêtres permettent de vérifier le bon déroulement de votre code et d'intervenir dans l'exécution du programme (ex. : décocher la case devant P3.2 revient à la mettre à zéro et donc déclencher l'interruption 0).

Les commandes « *Run* » ou « *Step* » du menu « *Debug* » permettent l'exécution du code.

Pour modifier le code il faut arrêter le mode de débogage (« *Start stop debug* », changer le code, le recompiler, puis repasser en mode de débogage et exécuter de nouveau le code).