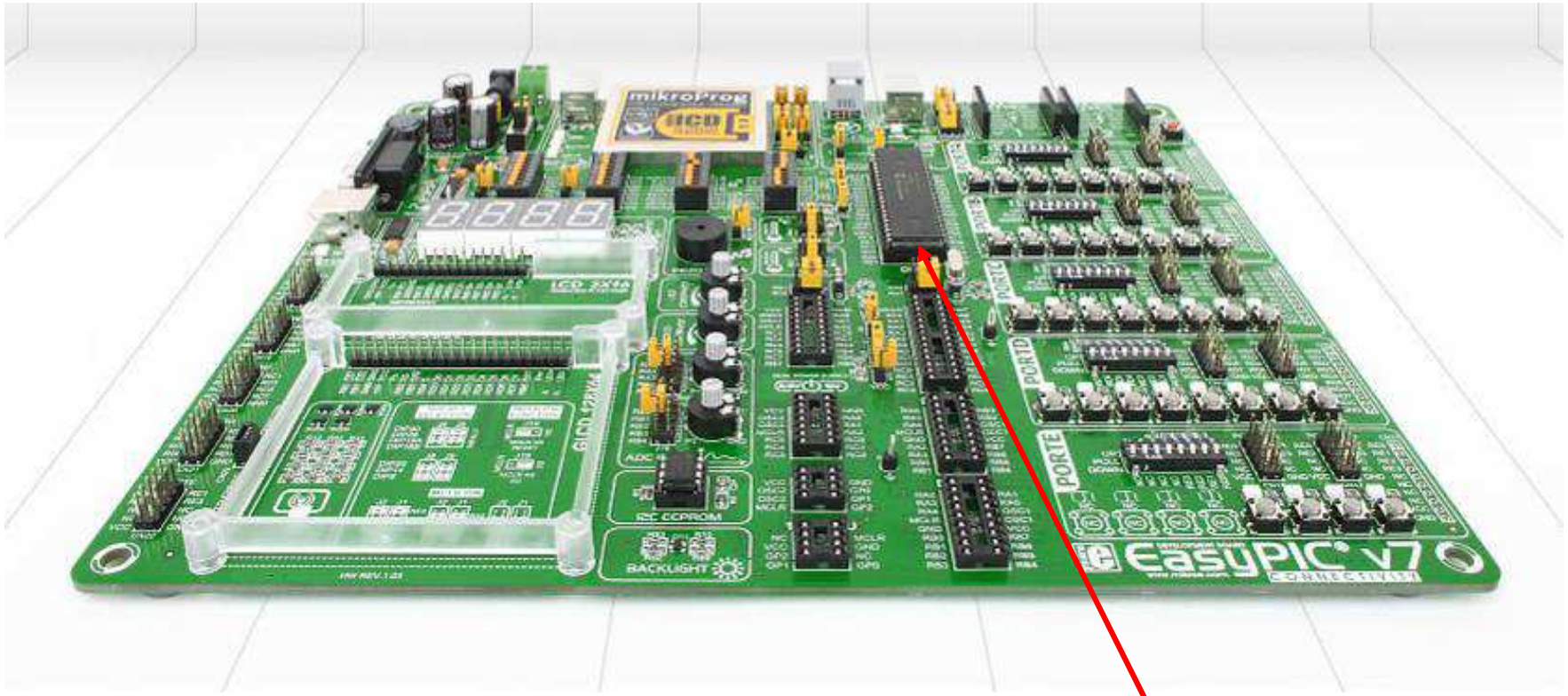
Informatique industrielle



Carte de développement EASYPIC 7

µcontrôleur PIC18F45K22

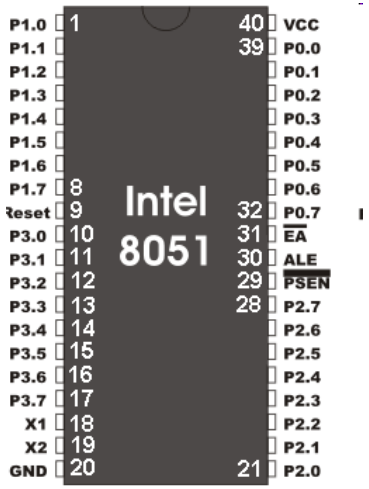
Plan

- Introduction
- Quelques Rappels
  - Les différentes bases de numération (bin, déc, hex)
  - Représentation des nombres (entiers, flottants)
  - Logique combinatoire et séquentielle
- Les différents types de composant
  - Les circuits Logiques Programmables (PLD) – Langage de description VHDL
  - Les DSP
  - Les µContrôleurs (assembleur, basic, C, C++)
  - Les cartes mini-PC type BeagleBone black (OS Linux)
- Les µcontrôleurs et en particulier les PICs
- Le µcontrôleur PIC 18F45K22
- La platine de développement EasyPic7 de chez Lextronics
  - La carte et ses différents modules
  - L'environnement de développement
- Langage C embarqué sur µcontrôleur

Les différentes composants programmables

Les microcontrôleurs

Un microcontrôleur est un circuit intégré rassemblant dans un même boitier un microprocesseur (généralement peu puissant), plusieurs types de mémoires et des périphériques de communication (Entrées-Sorties).



Les microcontrôleurs représentent la plus grosse partie des ventes dans le marché des microprocesseur. En effet, un foyer moyen d'un pays développé est susceptible d'être équipé de seulement un ou deux microprocesseurs généralistes (ordinateurs), mais d'une ou deux douzaines de microcontrôleurs (appareils électroménagers).

Les microcontrôleurs sont des composants microprogrammés. Plusieurs langage sont utilisés : assembleur (bas niveau), Basic, langage C et plus récemment C++

Les microcontrôleurs

Un microcontrôleur (μC) se présente sous la forme d'un circuit intégré réunissant tous les éléments d'une structure à base de microprocesseur. Voici généralement ce que l'on trouve à l'intérieur d'un tel composant :

- un microprocesseur (C.P.U.), avec une largeur du chemin de données allant de 4 bits pour les modèles les plus basiques à 32 ou 64 bits pour les modèles les plus évolués,
- de la mémoire vive (RAM) pour stocker les données et variables,
- de la mémoire non volatile (ROM) pour stocker les programmes, différentes technologies EPROM, EEPROM, flash,
- souvent un oscillateur pour le cadencement. Il peut être réalisé avec un quartz, un circuit RC ou encore une PLL,
- des périphériques, capables d'effectuer des tâches spécifiques.

Les microcontrôleurs

Parmi les périphériques, on peut mentionner entre autre :

- les convertisseurs analogiques-numériques (CAN) (donnent un nombre binaire à partir d'une tension électrique),
- les convertisseurs numériques-analogiques (CNA) (effectuent l'opération inverse),
- les générateurs de signaux à modulation de largeur d'impulsion (MLI, ou en anglais, PWM pour Pulse Width Modulation),
- les timers/compteurs (compteurs d'impulsions d'horloge interne ou d'événements externes),
- les chiens de garde (watchdog),
- les comparateurs (comparent deux tensions électriques),
- les contrôleurs de bus de communication (UART, I^2C, SSP, CAN, FlexRay, USB, Ethernet, etc.).

Les microcontrôleurs

Un microcontrôleur est donc une unité de traitement de l'information de type microprocesseur à laquelle on a ajouté des périphériques internes permettant de réaliser des montages sans nécessiter l'ajout de composants externes.

Plusieurs Constructeurs se partagent le marché des microcontrôleurs, citons INTEL, MOTOROLA, SEAGATE-THOMSON, AMTEL, ZILOG, PHILIPS et enfin MICROCHIP avec ses PICs très populaires qui nous intéressent ici dans ce cours.

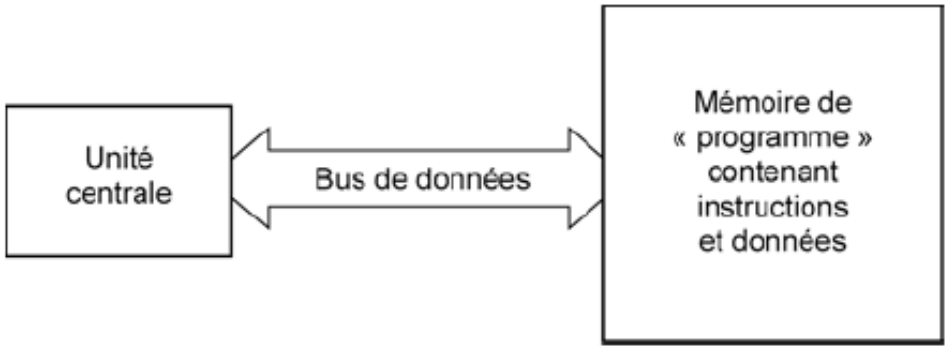
Les µC sont basés sur 2 architectures possibles, Von Neumann (INTEL80XX, motorola HC05, HC08 et HC11, ou ZILOG Z80) ou Harvard (PIC).

Référence	Fabricant	Vitesse	RAM	ROM / EPROM / FLASH	EEPROM	E / S Logiques	Timer	Entrées Analogiques	Particularité
8051	Intel	12 Mhz	128 o	4 Ko	X	32	2	0	
16C71	Microchip	20 Mhz	36 o	1Kx14	X	13	1	4	RISC
6805 S2	Motorola	4 MHz	64	1 Ko	X	16	2	8	
68HC11 A1	Motorola	8 MHz	256 o	X	512	22	1	8	Etendu
AT90S 8515	Atmel	20 MHz	512 o	4 Ko	512	32	3	8	RISC
ST 6265	Thomson	8 MHz	128 o	4 Ko	64 o	21	2	13	

EXEMPLES DE MICROCONTRÔLEURS

Les microcontrôleurs

De nombreux microprocesseurs et microcontrôleurs actuels utilisent une architecture interne dite de Von Neumann, c'est-à-dire en fait une architecture commune à celle que l'on rencontre habituellement dans les micro-ordinateurs. La mémoire, appelée improprement de programme, contient en fait des instructions et des données placées à la suite les unes des autres et l'on ne dispose que d'un bus, appelé bus de données, pour véhiculer tour à tour les codes des instructions et les données qui leur sont associées comme le montre la suivante.



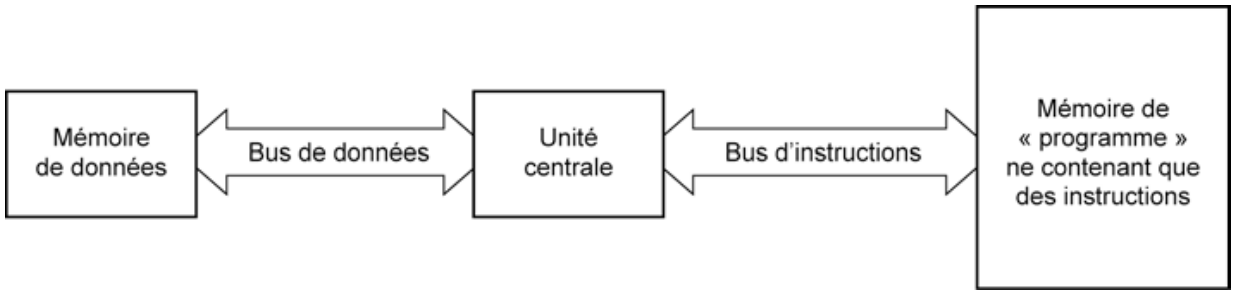
Les microcontrôleurs

Si cette architecture donne toute satisfaction comme vous en avez la preuve éclatante chaque jour, elle pose quelques problèmes dès que l'on veut faire fonctionner l'ensemble rapidement. En effet, l'exécution d'une seule instruction nécessite plusieurs échanges de données sur le seul et unique bus dévolu à cet usage puisqu'il faut tout d'abord aller chercher le code de l'instruction puis le ou les données qu'elle doit manipuler.

Il est alors préférable de faire appel à une architecture dite Harvard dans laquelle les instructions et les données sont clairement différenciées et sont véhiculées sur des bus différents. Vu de l'utilisateur, cela ne change rien bien sûr et les circuits de ce type s'utilisent exactement comme les autres. En revanche, les résultats obtenus, en termes de vitesse d'exécution des programmes, peuvent être impressionnants.

Les microcontrôleurs

En effet, l'exécution d'une instruction ne fait plus appel qu'à un seul cycle machine puisque l'on peut simultanément, grâce aux deux bus, rechercher le code de l'instruction et la ou les données qu'elle manipule.



Rompant avec une tradition bien établie, les microcontrôleurs PIC de Microchip, toutes familles confondues, utilisent une architecture Harvard mais ce n'est pas tout…

Les microcontrôleurs

MICROPROCESSEURS RISC ET CISC

Les microprocesseurs **CISC** (Complex Instruction Set Computing) sont dotés d'un jeu étendu d'instructions complexes. Ces instructions sont relativement lentes. Les microprocesseurs CISC privilégient la puissance de traitement au détriment de la rapidité.

Les microprocesseurs **RISC** (Reduced Instruction Set Computing) sont munis d'un jeu réduit d'instructions simples (75 pour les PIC 18F). Ces instructions sont adaptées et sont très rapides. Les PICs sont des μP de type RISC.

$\Rightarrow$ PIC : architecture Harvard + RISC

Les microcontrôleurs

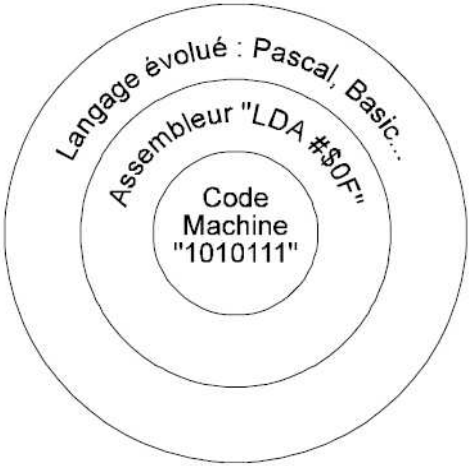
Langage de programmation des µP - Les programmes peuvent être écrits à trois niveaux différents :

- Langages évolués

  Pascal, Basic, Langage C, etc

- Assembleur (mnémonique et opérande),

- Langage machine (binaire ou hexadécimal).



Les microcontrôleurs

Langage de programmation des µP - Exemples

Langage C

```
// Fonction principale

void main(void)
{
    PORTB_en_sortie();        // fonction écrite par le programmeur
    while(VRAI)               // Répéter toujours
                              // voir § La structure alternative ou sélection page 5
    {
       Allumer_LED_PORTB();   // fonction écrite par le programmeur
       DelayMs(250) ;         // fonction prédéfinie
                              // Note prof : Le fichier delay.c doit être
                              // ajouté au projet MPLAB en plus du delay.h
       Eteindre_LED_PORTB() ;// fonction écrite par le programmeur
       DelayMs(250) ;         // fonction prédéfinie
    }
}
```

Assembleur

```
;*****************************************************************
;                PROGRAMME PRINCIPAL                             *
;*****************************************************************
start
        bsf     PORTA,2                 ; allumer la LED
        call      tempo                 ; appeler la tempo de 0.5s
        bcf     PORTA,2                 ; éteindre LED
        call      tempo                 ; appeler la tempor de 0.5s
        goto      start                 ; boucler
        END                             ; directive fin de programme
```

Hexadécimal

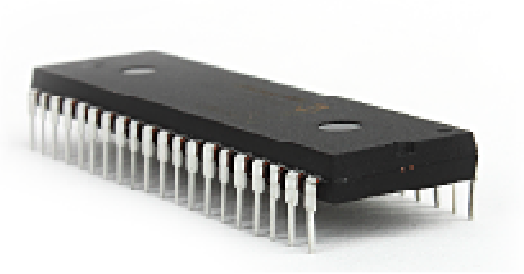
```
:020000040000FA
:020000000428D2
:08000800FF30A100FF30A2004F
:10001000FF30A300FF30A4000F30A50021108A118B
:100020000A12512005308A110A12282026148A113A
:100030000A124B2005308A110A12282026108A1134
:100040000A124B208A110A1212288A110A12252834
:10005000A7002708A8006430A9000A308A110A12F4
:100060003128AA00FF30AB002908AD002808AC00F9
:1000700077308A110A122B0703183B288A110A12BB
:10008000AC0B38288A110A12AD0B36288A110A12D5
:10009000AA0B342808008A110A124E282608850067
:0E00A00008008A110A12542821086500080081
:02400E00F23F7F
:00000001FF
```

Les microcontrôleurs

Les µP PICs

Les microcontrôleurs PIC (ou PICmicro dans la terminologie du fabricant) forment une famille de microcontrôleurs de la société Microchip. Le nom PIC n'est pas officiellement un acronyme, bien que la traduction en « Peripheral Interface Controller » (contrôleur d'interface périphérique) soit généralement admise.

Famille de PICS : La famille des PICs est subdivisée à l'heure actuelle en 3 grandes familles : La famille *Base-Line*, qui utilise des mots d'instructions de 12 bits, la famille *Mid-Range*, qui utilise des mots de 14 bits (et dont font partie les 16F877A et 18F45K22), et la famille *High-End*, qui utilise des mots de 16 bits.



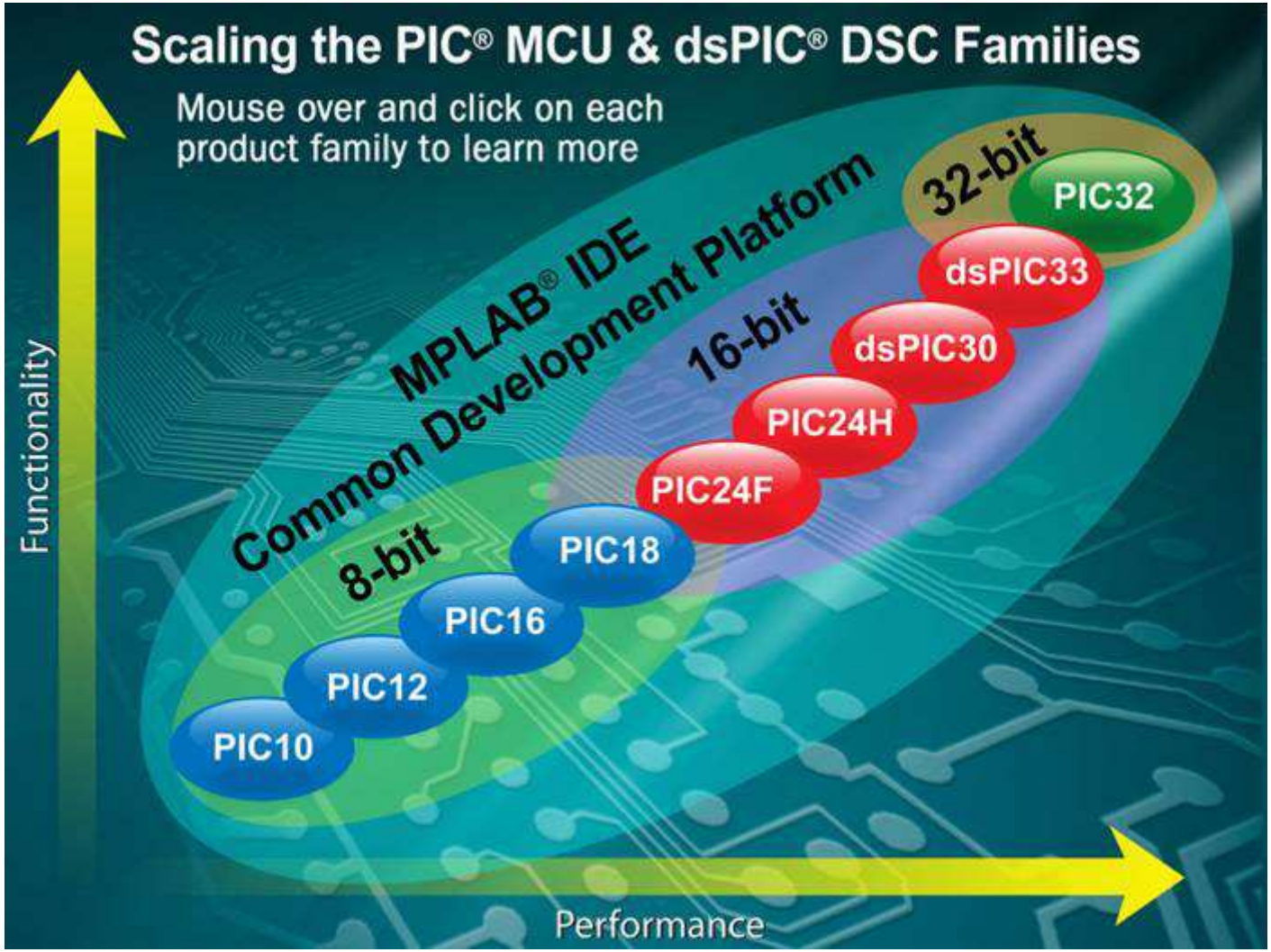
Pic 18F45K22 équipant la carte de test Easypic7

Les microcontrôleurs

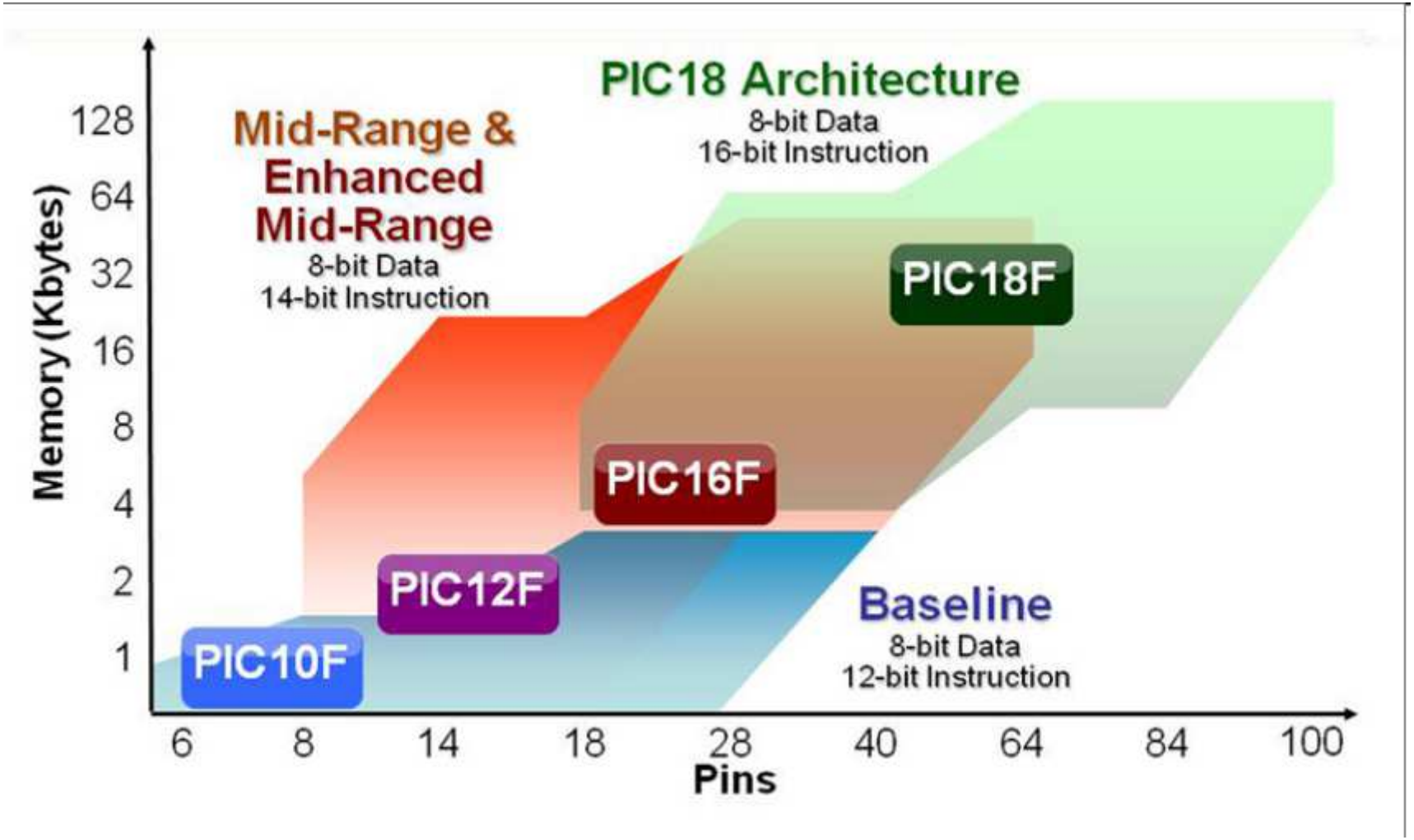
- **2 chiffres** : famille du PIC (10, 12, 16, 17, 18, 24, 30, 32, 33) — le PIC14 existe, c'est le PIC14000.
- **1 lettre** : type de mémoire de programme (C ou F). Le F indique en général qu'il s'agit d'une mémoire *flash* et donc effaçable électroniquement. La lettre C indique en général que la mémoire ne peut être effacée que par exposition aux *ultra-violets* (exception pour le PIC16C84 qui utilise une mémoire *EEPROM* donc effaçable électriquement). Un L peut être ajouté devant pour indiquer qu'il s'agit d'un modèle basse tension (exemple : 2 V à 5,5 V si LF — 4,2 V à 5,5 V si F).
- **un nombre de 2 à 4 chiffres** : modèle du PIC au sein de la famille. Toutefois il y a maintenant des exceptions : PIC18F45K22 par exemple.
- **un groupe de lettres** pour indiquer le boîtier et la gamme de température.

Par exemple, le PIC18LF4682-I/P est un microcontrôleur de la famille PIC18, basse tension (L), à mémoire flash (F), modèle 4682, gamme de température industrielle (I) et boîtier DIL40.

Les microcontrôleurs



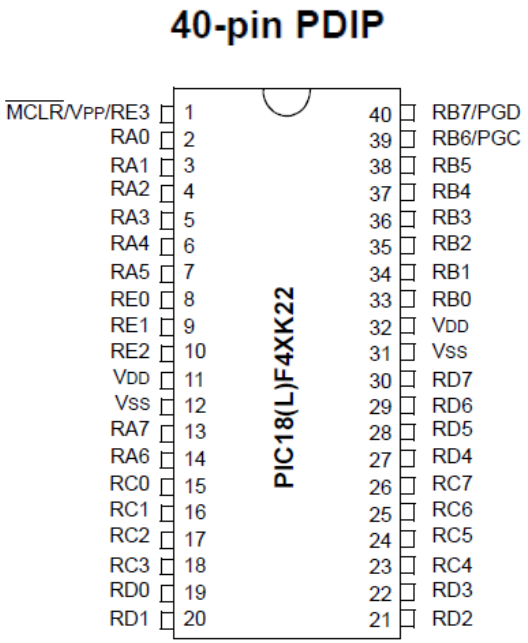
Les microcontrôleurs



Les microcontrôleurs

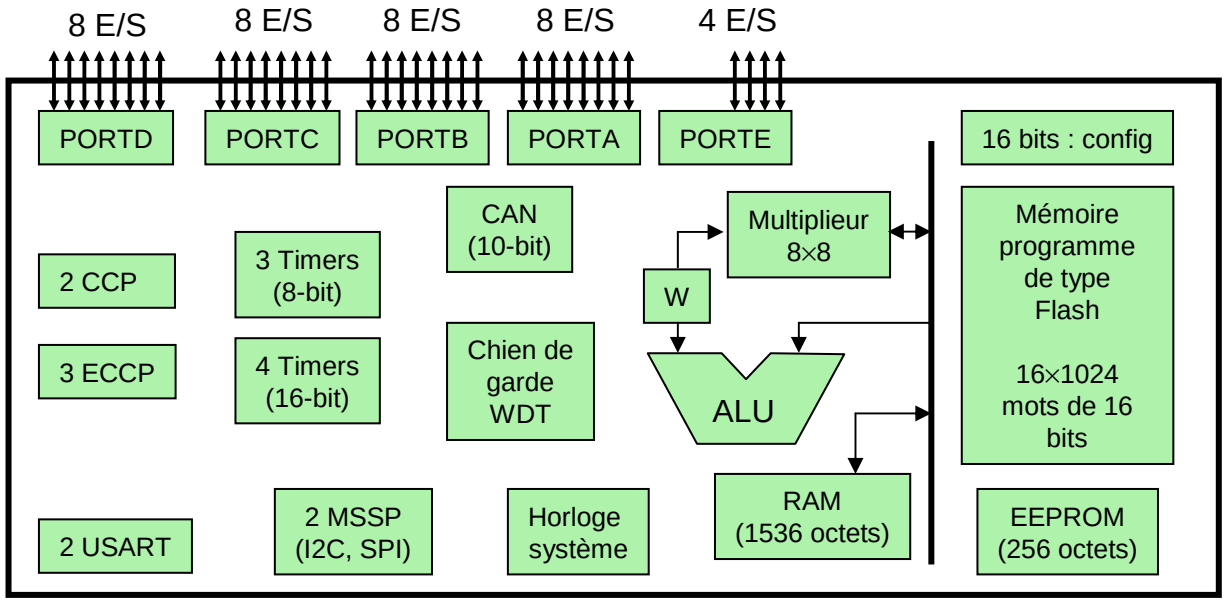
Le PIC 18F45K22 (équipant le carte de test Easypic7)

Parameter Name	Value
Program Memory Type	Flash
Program Memory (KB)	32
CPU Speed (MIPS)	16
RAM Bytes	1,536
Data EEPROM (bytes)	256
Digital Communication Peripherals	2-A/E/USART, 2-MSSP(SPI/I2C)
Capture/Compare/PWM Peripherals	2 CCP, 3 ECCP
Timers	3 x 8-bit, 4 x 16-bit
ADC	28 ch, 10-bit
Comparators	2
Temperature Range (C)	-40 to 125
Operating Voltage Range (V)	1.8 to 5.5
Pin Count	40
XLP	Yes
Cap Touch Channels	28



Brochage

Les microcontrôleurs



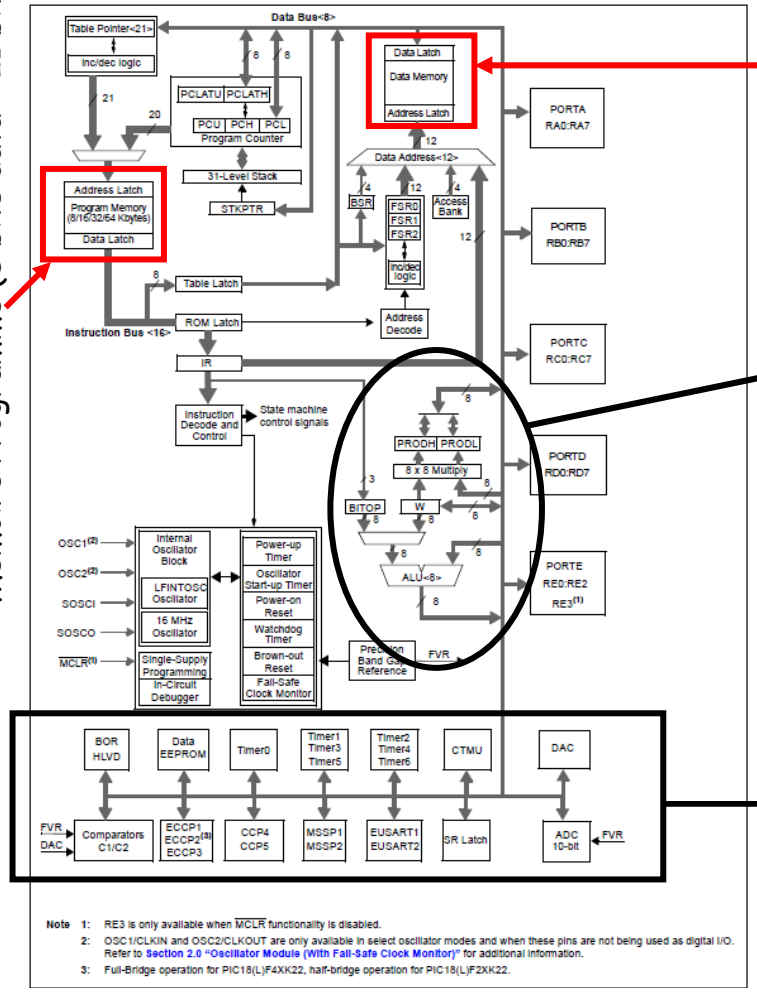
Bloc diagramme du PIC 18F45K22

- TIMERS : Microchip appelle TIMER des compteurs
- WDT : Watch Dog Timer (Chien de garde)
- (E)CCP : (Enhance)Capture/Compare/PWM
    - Capture : permet la mesure de temps
    - Compare : permet la production de signaux rectangulaires
- MSSP : Master Synchronous Serial Port (Port Série Synchrone Maître)
    - SPI : communications séries synchrones sans protocole logiciel
    - I2C : standard Philips, communications séries synchrones avec protocole logiciel
- A/E/USART : (Addressable/Enhanced) Universal Synchronous Asynchronous Receiver Transmitter (RS232 – RS485)
- CAN : convertisseur analogique numérique sur 10 bits à 28 entrées multiplexées

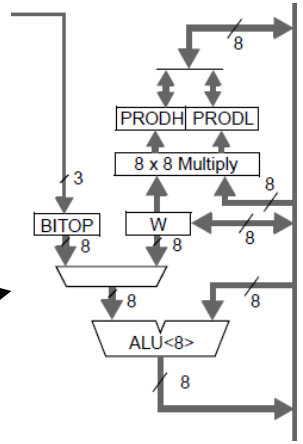
Les microcontrôleurs

Bloc diagramme du PIC 18F45K2 (équipant le carte de test Easypic7)

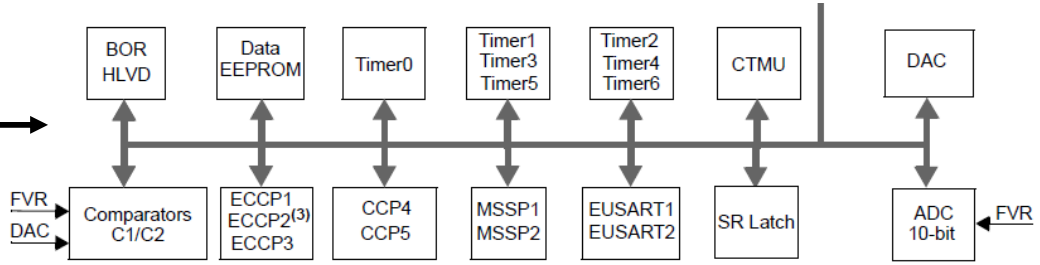
Mémoire Programme (8 bits data – 21 bits @)



Mémoire Data
(8 bits data – 12 bits @)



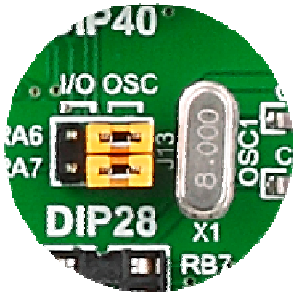
Le PIC18 possède une multiplication 8x8 matérielle, extrêmement rapide (100nS à 1uS) ce qui lui confère des possibilités de DSP particulièrement utiles pour le traitement numérique du signal.



Les microcontrôleurs

40-PDIP	40-UQFN	44-TQFP	44-QFN	I/O	Analog	Comparator	CTMU	SR Latch	Reference	(E)CCP	EUSART	MSSP	Timers	Interrupts	Pull-up	Basic
2	17	19	19	RA0	AN0	C12IN0-										
3	18	20	20	RA1	AN1	C12IN1-										
4	19	21	21	RA2	AN2	C2IN+			VREF- DACOUT							
5	20	22	22	RA3	AN3	C1IN+			VREF+							
6	21	23	23	RA4		C1OUT		SRQ					T0CKI			
7	22	24	24	RA5	AN4	C2OUT		SRNQ	HLVDIN			$\overline{SS}$1				
14	29	31	33	RA6												OSC2 CLKO
13	28	30	32	RA7												OSC1 CLKI
33	8	8	9	RB0	AN12			SRI		FLT0				INT0	Y	
34	9	9	10	RB1	AN10	C12IN3-								INT1	Y	
35	10	10	11	RB2	AN8		CTED1							INT2	Y	
36	11	11	12	RB3	AN9	C12IN2-	CTED2			CCP2 P2A(1)					Y	
37	12	14	14	RB4	AN11								T5G	IOC	Y	
38	13	15	15	RB5	AN13					CCP3 P3A(3)			T1G T3CKI(2)	IOC	Y	
39	14	16	16	RB6										IOC	Y	PGC
40	15	17	17	RB7										IOC	Y	PGD
15	30	32	34	RC0						P2B(4)			SOSCO T1CKI T3CKI(2) T3G			

Horloges externes



Note 1: CCP2 multiplexed in fuses.
2: T3CKI multiplexed in fuses.
3: CCP3/P3A multiplexed in fuses.
4: P2B multiplexed in fuses.

Les microcontrôleurs

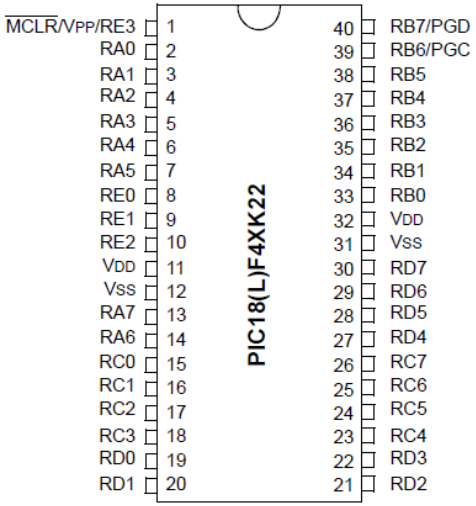
40-PDIP	40-UQFN	44-TQFP	44-QFN	I/O	Analog	Comparator	CTMU	SR Latch	Reference	(E)CCP	EUSART	MSSP	Timers	Interrupts	Pull-up	Basic
16	31	35	35	RC1						CCP2(1) P2A			SOSCI			
17	32	36	36	RC2	AN14		CTPLS			CCP1 P1A			T5CKI			
18	33	37	37	RC3	AN15							SCK1 SCL1				
23	38	42	42	RC4	AN16							SDI1 SDA1				
24	39	43	43	RC5	AN17							SDO1				
25	40	44	44	RC6	AN18						TX1 CK1					
26	1	1	1	RC7	AN19						RX1 DT1					
19	34	38	38	RD0	AN20							SCK2 SCL2				
20	35	39	39	RD1	AN21					CCP4		SDI2 SDA2				
21	36	40	40	RD2	AN22					P2B(4)						
22	37	41	41	RD3	AN23					P2C		$\overline{SS}$2				
27	2	2	2	RD4	AN24					P2D		SD02				
28	3	3	3	RD5	AN25					P1B						
29	4	4	4	RD6	AN26					P1C	TX2 CK2					
30	5	5	5	RD7	AN27					P1D	RX2 DT2					
8	23	25	25	RE0	AN5					CCP3 P3A(3)						

Note 1: CCP2 multiplexed in fuses.
2: T3CKI multiplexed in fuses.
3: CCP3/P3A multiplexed in fuses.
4: P2B multiplexed in fuses.

Les microcontrôleurs

TABLE 2: PIC18(L)F4XK22 PIN SUMMARY (CONTINUED)

40-PDIP	40-UQFN	44-TQFP	44-QFN	I/O	Analog	Comparator	CTMU	SR Latch	Reference	(E)CCP	EUSART	MSSP	Timers	Interrupts	Pull-up	Basic
9	24	26	26	RE1	AN6					P3B						
10	25	27	27	RE2	AN7					CCP5						
1	16	18	18	RE3											Y	$\overline{MCLR}$ V_{PP}
11 32	7, 26	7 28	7,8 28, 29													V_{DD}
12 31	6, 27	6 29	6 30, 31													V_{SS}
—	—	12, 13 33, 34	13	NC												



V_{DD} : Positive supply for logic and I/O pins.

V_{SS} : Ground reference for logic and I/O pins.

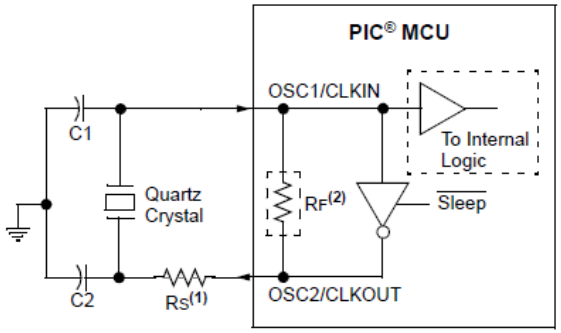
Les microcontrôleurs

Les éléments de base du PIC 18F45K22

L'horloge :

Il existe différents types d'horloge (on compte jusqu'à 12 types sur certains PIC). Elle peut être soit interne, soit externe.

Avec une horloge externe et un oscillateur à quartz, on peut avoir des fréquences allant jusqu'à 64 MHz. Le filtre passe bas (R_s, C_1, C_2) limite les harmoniques dus à l'écrêtage et réduit l'amplitude de l'oscillation, il n'est pas obligatoire.



Quelque soit l'oscillateur utilisé, l'horloge système dite aussi horloge instruction est obtenue **en divisant la fréquence par 4**. Dans la suite de ce document on utilisera le terme Fosc/4 pour désigner l'horloge système. Avec un quartz de 4 MHz, on obtient une horloge instruction de 1 MHz, soit le temps pour exécuter une instruction de 1µs.

La configuration de l'horloge s'effectue via plusieurs registres de configuration : CONFIG1H, OSCON, OSCTUNE etc.

Les microcontrôleurs

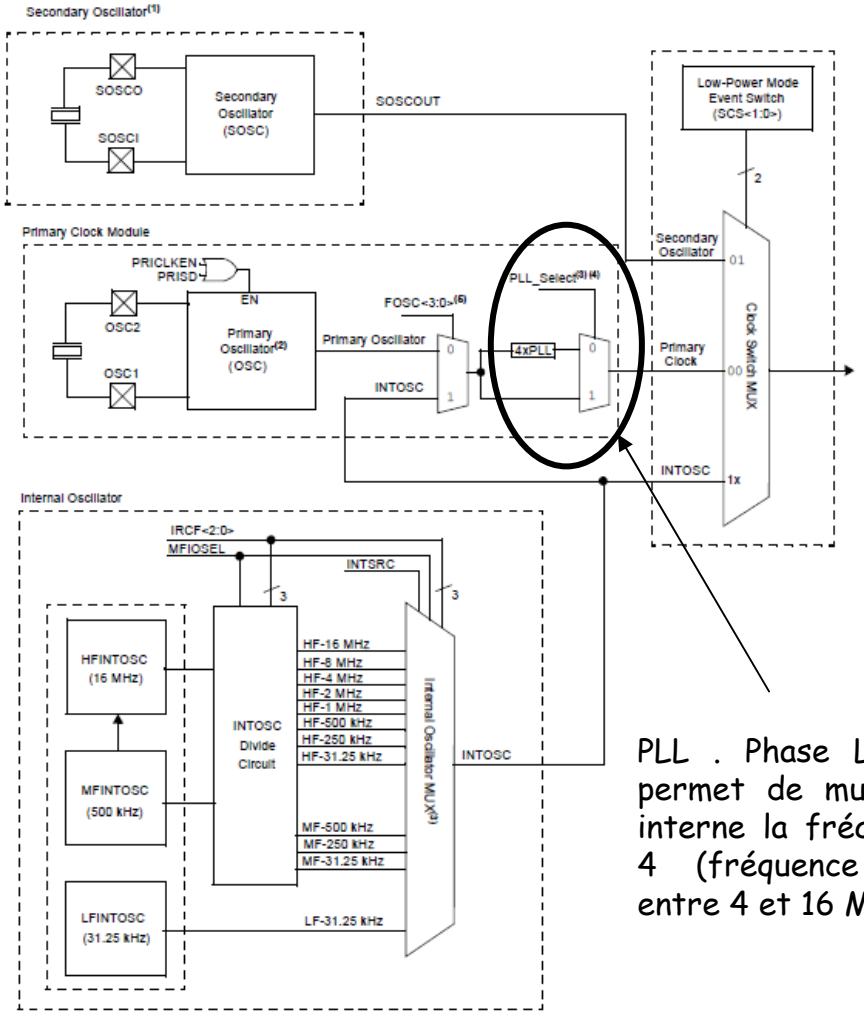
Les éléments de base du PIC 18F45K22

Horloge externe secondaire →

Horloge externe primaire (quartz jusqu'à 16 MHz) →

Horloge interne primaire 16 MHz (jusqu'à 64 MHz si × par 4 via PLL) →

Horloge interne secondaire 31.25 kHz →



PLL . Phase Lock Loop, permet de multiplier en interne la fréquence par 4 (fréquence d'entrée entre 4 et 16 MHz)

Les microcontrôleurs

L'ALU et l'accumulateur W

L'ALU est une Unité Arithmétique et Logique 8 Bits qui réalise les opérations arithmétiques et logiques de base. L'accumulateur W est un registre de travail 8 bits, toutes les opérations à deux opérandes passe par lui. On peut avoir :

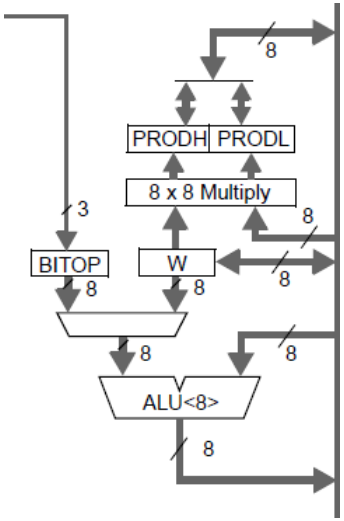
- Une instruction sur un seul opérande qui est en général un registre situé dans la RAM
- Une instruction sur 2 opérandes. Dans ce cas, l'un des deux opérandes est toujours l'accumulateur W, l'autre peut être soit un registre soit une constante.

Pour les instructions dont un des opérandes est un registre, le résultat peut être récupéré soit dans l'accumulateur, soit dans le registre lui-même.

Opérandes = paramètre (variable ou constante) d'une opération

TABLE 8-1: PERFORMANCE COMPARISON FOR VARIOUS MULTIPLY OPERATIONS

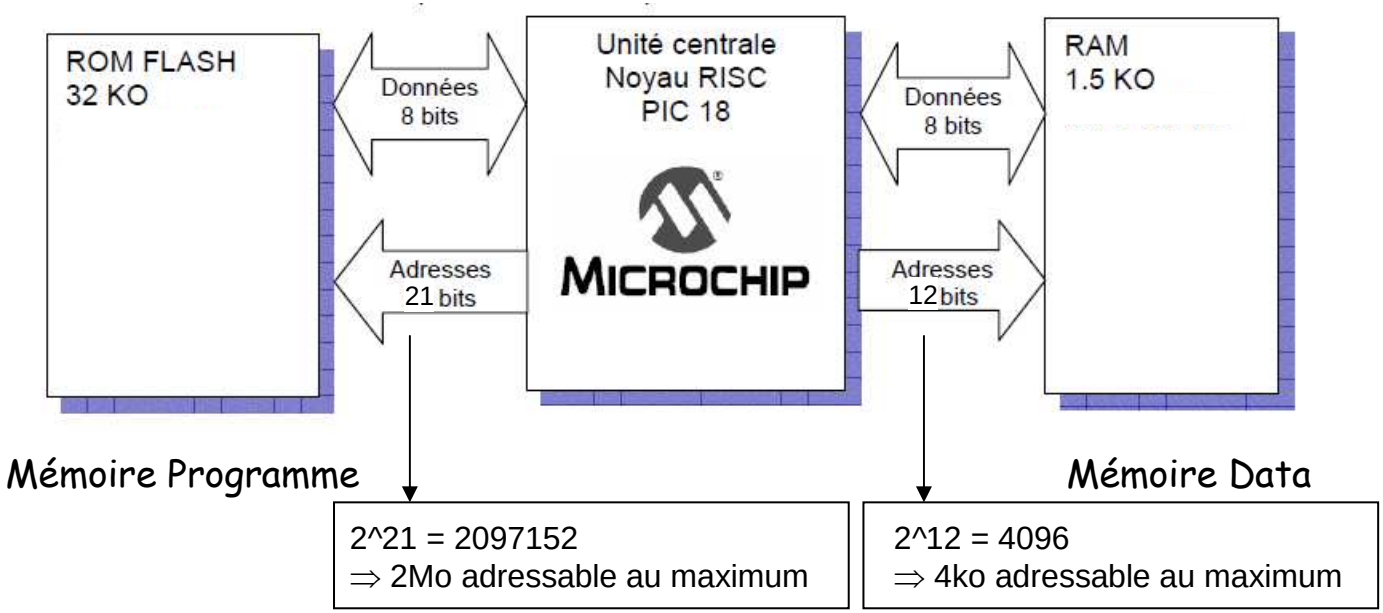
Routine	Multiply Method	Program Memory (Words)	Cycles (Max)	Time			
				@ 64 MHz	@ 40 MHz	@ 10 MHz	@ 4 MHz
8 x 8 unsigned	Without hardware multiply	13	69	4.3 µs	6.9 µs	27.6 µs	69 µs
	Hardware multiply	1	1	62.5 ns	100 ns	400 ns	1 µs
8 x 8 signed	Without hardware multiply	33	91	5.7 µs	9.1 µs	36.4 µs	91 µs
	Hardware multiply	6	6	375 ns	600 ns	2.4 µs	6 µs
16 x 16 unsigned	Without hardware multiply	21	242	15.1 µs	24.2 µs	96.8 µs	242 µs
	Hardware multiply	28	28	1.8 µs	2.8 µs	11.2 µs	28 µs
16 x 16 signed	Without hardware multiply	52	254	15.9 µs	25.4 µs	102.6 µs	254 µs
	Hardware multiply	35	40	2.5 µs	4.0 µs	16.0 µs	40 µs



Les microcontrôleurs

Organisation Mémoire

Les PIC18 sont à architecture HAVARD. Les espaces mémoires programmes et données (appelés registres, les registres des périphériques sont appelés registres spéciaux) sont distincts. Ceci implique la création d'instructions et de processus différents pour l'accès données en ROM et en RAM.

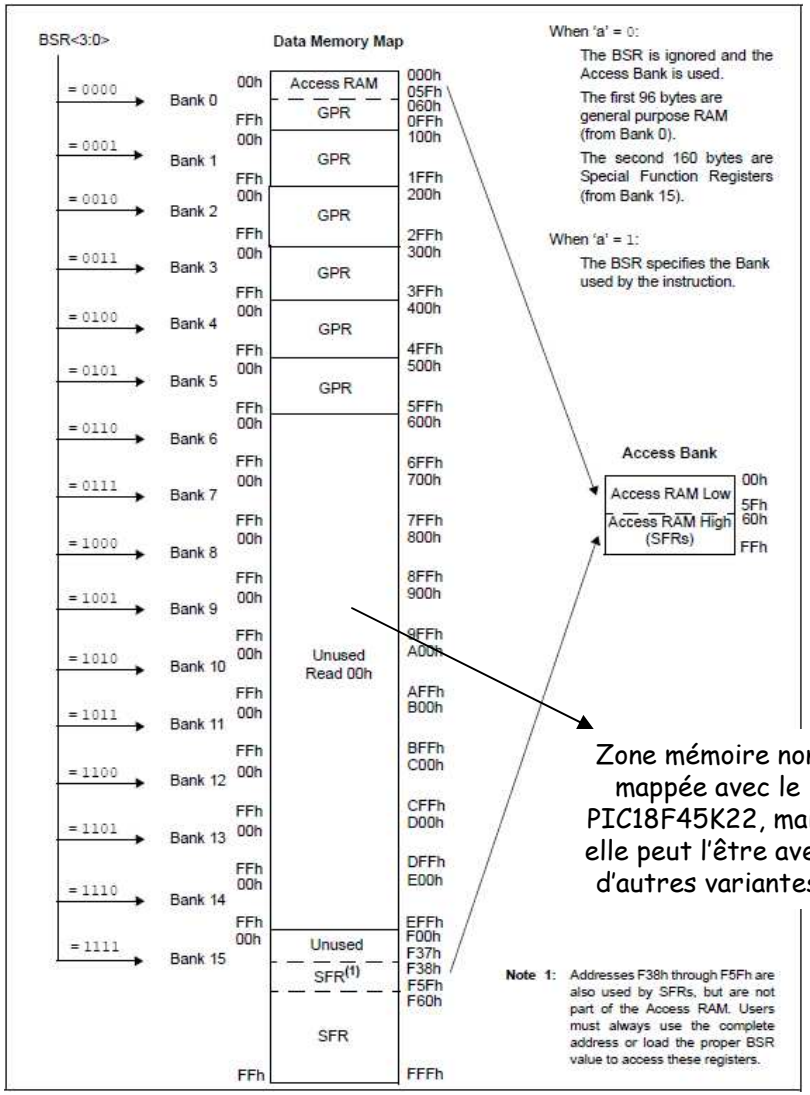


Les microcontrôleurs

Organisation de la mémoire RAM (Data)

L'espace mémoire RAM est divisée artificiellement en 16 pages ou banques (banks), d'une taille fixe de 256 octets chacune (de 00_h à FF_h).

- Les registres **SFR** (Special Function Registers) qui sont les registres de configuration du PIC se trouvent en partie haute de la mémoire, en page 15 (de $F38_h$ à FFF_h).
- Les positions restantes (en partie basse) constituent les registres **GPR** (General Propose Registers) ou RAM utilisateur.

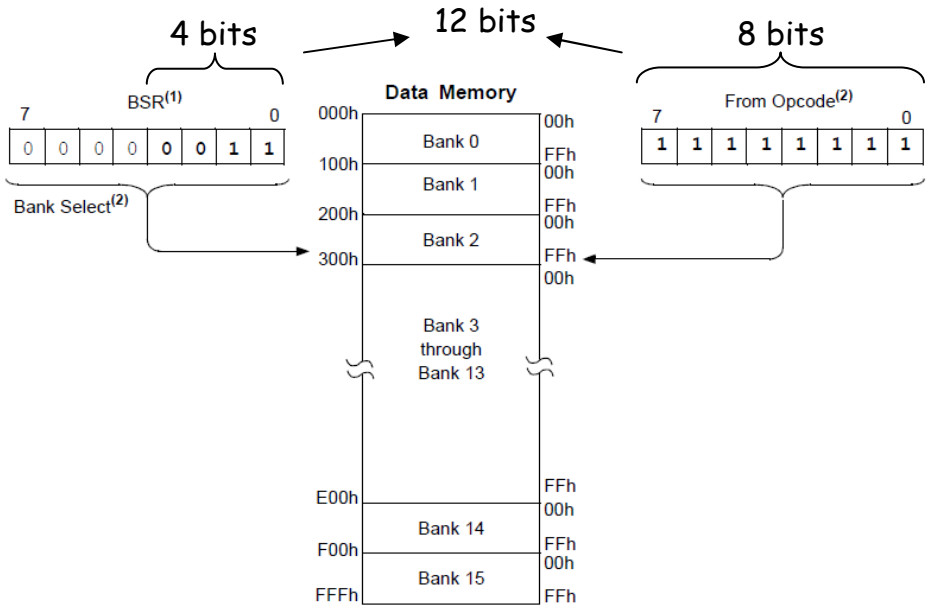


Les microcontrôleurs

Organisation de la mémoire RAM

Compte tenu de la taille d'une page, qui est de 256 octets, il suffit donc de 8 bits pour adresser n'importe quel élément au sein de cette dernière. Comme le montre la figure suivante, ces 8 bits font toujours partie intégrante du code de l'instruction.

Pour savoir quelle adresse complète sera réellement adressée par telle ou telle instruction, il faut donc connaître également le numéro de page mémoire utilisée, numéro qui est codé dans un registre spécifique appelé BSR (Bank Select Register).



Les microcontrôleurs

Registres de configuration (SFR) avec leurs adresses

TABLE 5-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F2X/4XK22 DEVICES

Address	Name	Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FD7h	TMR0H	FAFh	SPBRG1	F87h	—(2)	F5Fh	CCPR3H
FFEh	TOSH	FD6h	TMR0L	FAEh	RCREG1	F86h	—(2)	F5Eh	CCPR3L
FFDh	TOSL	FD5h	T0CON	FADh	TXREG1	F85h	—(2)	F5Dh	CCP3CON
FFCh	STKPTR	FD4h	—(2)	FACh	TXSTA1	F84h	PORTE	F5Ch	PWM3CON
FFBh	PCLATU	FD3h	OSCCON	FABh	RCSTA1	F83h	PORTD(3)	F5Bh	ECCP3AS
FFAh	PCLATH	FD2h	OSCCON2	FAAh	EEADRH(4)	F82h	PORTC	F5Ah	PSTR3CON
FF9h	PCL	FD1h	WDTCON	FA9h	EEADR	F81h	PORTB	F59h	CCPR4H
FF8h	TBLPTRU	FD0h	RCON	FA8h	EEDATA	F80h	PORTA	F58h	CCPR4L
FF7h	TBLPTRH	FCFh	TMR1H	FA7h	EECON2(1)	F7Fh	IPR5	F57h	CCP4CON
FF6h	TBLPTRL	FCEh	TMR1L	FA6h	EECON1	F7Eh	PIR5	F56h	CCPR5H
FF5h	TABLAT	FCDh	T1CON	FA5h	IPR3	F7Dh	PIE5	F55h	CCPR5L
FF4h	PRODH	FCCh	T1GCON	FA4h	PIR3	F7Ch	IPR4	F54h	CCP5CON
FF3h	PRODL	FCBh	SSP1CON3	FA3h	PIE3	F7Bh	PIR4	F53h	TMR4
FF2h	INTCON	FCAh	SSP1MSK	FA2h	IPR2	F7Ah	PIE4	F52h	PR4
FF1h	INTCON2	FC9h	SSP1BUF	FA1h	PIR2	F79h	CM1CON0	F51h	T4CON
FF0h	INTCON3	FC8h	SSP1ADD	FA0h	PIE2	F78h	CM2CON0	F50h	TMR5H
FEFh	INDF0(1)	FC7h	SSP1STAT	F9Fh	IPR1	F77h	CM2CON1	F4Fh	TMR5L
FEEh	POSTINC0(1)	FC6h	SSP1CON1	F9Eh	PIR1	F76h	SPBRGH2	F4Eh	T5CON
FEDh	POSTDEC0(1)	FC5h	SSP1CON2	F9Dh	PIE1	F75h	SPBRG2	F4Dh	T5GCON
FECh	PREINC0(1)	FC4h	ADRESH	F9Ch	HLVDCON	F74h	RCREG2	F4Ch	TMR6
FEBh	PLUSW0(1)	FC3h	ADRESL	F9Bh	OSCTUNE	F73h	TXREG2	F4Bh	PR6
FEAh	FSR0H	FC2h	ADCON0	F9Ah	—(2)	F72h	TXSTA2	F4Ah	T6CON

Les microcontrôleurs

Registres de configuration (SFR) avec leurs adresses (suite)

TABLE 5-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F2X/4XK22 DEVICES

Address	Name	Address	Name	Address	Name	Address	Name	Address	Name
FE9h	FSR0L	FC1h	ADCON1	F99h	—(2)	F71h	RCSTA2	F49h	CCPTMRS0
FE8h	WREG	FC0h	ADCON2	F98h	—(2)	F70h	BAUDCON2	F48h	CCPTMRS1
FE7h	INDF1(1)	FBFh	CCPR1H	F97h	—(2)	F6Fh	SSP2BUF	F47h	SRCON0
FE6h	POSTINC1(1)	FBEh	CCPR1L	F96h	TRISE	F6Eh	SSP2ADD	F46h	SRCON1
FE5h	POSTDEC1(1)	FBDh	CCP1CON	F95h	TRISD(3)	F6Dh	SSP2STAT	F45h	CTMUCONH
FE4h	PREINC1(1)	FBCh	TMR2	F94h	TRISC	F6Ch	SSP2CON1	F44h	CTMUCONL
FE3h	PLUSW1(1)	FBBh	PR2	F93h	TRISB	F6Bh	SSP2CON2	F43h	CTMUICON
FE2h	FSR1H	FBAh	T2CON	F92h	TRISA	F6Ah	SSP2MSK	F42h	VREFCON0
FE1h	FSR1L	FB9h	PSTR1CON	F91h	—(2)	F69h	SSP2CON3	F41h	VREFCON1
FE0h	BSR	FB8h	BAUDCON1	F90h	—(2)	F68h	CCPR2H	F40h	VREFCON2
FDFh	INDF2(1)	FB7h	PWM1CON	F8Fh	—(2)	F67h	CCPR2L	F3Fh	PMD0
FDEh	POSTINC2(1)	FB6h	ECCP1AS	F8Eh	—(2)	F66h	CCP2CON	F3Eh	PMD1
FDDh	POSTDEC2(1)	FB5h	—(2)	F8Dh	LATE(3)	F65h	PWM2CON	F3Dh	PMD2
FDCh	PREINC2(1)	FB4h	T3GCON	F8Ch	LATD(3)	F64h	ECCP2AS	F3Ch	ANSELE
FDBh	PLUSW2(1)	FB3h	TMR3H	F8Bh	LATC	F63h	PSTR2CON	F3Bh	ANSELD
FDAh	FSR2H	FB2h	TMR3L	F8Ah	LATB	F62h	IOCB	F3Ah	ANSELC
FD9h	FSR2L	FB1h	T3CON	F89h	LATA	F61h	WPUB	F39h	ANSELB
FD8h	STATUS	FB0h	SPBRGH1	F88h	—(2)	F60h	SLRCON	F38h	ANSELA

Note 1: This is not a physical register.
2: Unimplemented registers are read as '0'.
3: PIC18(L)F4XK22 devices only.
4: PIC18(L)F26K22 and PIC18(L)F46K22 devices only.

Les microcontrôleurs

TABLE 5-2: REGISTER FILE SUMMARY FOR PIC18(L)F2X/4XK22 DEVICES (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
FA2h	IPR2	OSCFIP	C1IP	C2IP	EEIP	BCL1IP	HLVDIP	TMR3IP	CCP2IP	1111 1111
FA1h	PIR2	OSCFIF	C1IF	C2IF	EEIF	BCL1IF	HLVDIF	TMR3IF	CCP2IF	0000 0000
FA0h	PIE2	OSCFIE	C1IE	C2IE	EEIE	BCL1IE	HLVDIE	TMR3IE	CCP2IE	0000 0000
F9Fh	IPR1	—	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP	-111 1111
F9Eh	PIR1	—	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	-000 0000
F9Dh	PIE1	—	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE	-000 0000
F9Ch	HLVDCON	VDIRMAG	BGVST	IRVST	HLVDEN	HLVDL<3:0>				0000 0000
F9Bh	OSCTUNE	INTSRC	PLLEN	TUN<5:0>						00xx xxxx
F96h	TRISE	WPUE3	—	—	—	—	TRISE2(1)	TRISE1(1)	TRISE0(1)	1--- -111
F95h	TRISD(1)	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0	1111 1111
F94h	TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	1111 1111
F93h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111
F92h	TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1111 1111
F8Dh	LATE(1)	—	—	—	—	—	LATE2	LATE1	LATE0	---- -xxx
F8Ch	LATD(1)	LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0	xxxx xxxx
F8Bh	LATC	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0	xxxx xxxx
F8Ah	LATB	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	xxxx xxxx
F89h	LATA	LATA7	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0	xxxx xxxx
F84h	PORTE(2)	—	—	—	—	RE3	—	—	—	---- x---
	PORTE(1)	—	—	—	—	RE3	RE2	RE1	RE0	---- x000
F83h	PORTD(1)	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	0000 0000
F82h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	0000 00xx
F81h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxx0 0000
F80h	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xx0x 0000

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note
1: PIC18(L)F4XK22 devices only.
2: PIC18(L)F2XK22 devices only.
3: PIC18(L)F23/24K22 and PIC18(L)F43/44K22 devices only.
4: PIC18(L)F26K22 and PIC18(L)F46K22 devices only.

Détail de quelques registres SFR et leurs états au démarrage

Les microcontrôleurs

EXEMPLE : Le registre STATUS ou registre d'état (@ FD8h).

Ce registre, présent sur tous les µcontrôleur avec parfois des noms légèrement différents selon les fabricant, a pour principal vocation d'indiquer sous forme codée le « résultat » de la dernière opération réalisée par l'Unité Centrale. Son contenu est donc principalement utilisé par les instructions de test.

Chaque bit de ce registre a une signification particulière :

REGISTER 5-2: STATUS: STATUS REGISTER

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC(1)	C(1)
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Les microcontrôleurs

Le registre STATUS

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC(1)	C(1)
bit 7							bit 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **N:** Negative bit
This bit is used for signed arithmetic (two's complement). It indicates whether the result was negative (ALU MSB = 1).
1 = Result was negative
0 = Result was positive

bit 3 **OV:** Overflow bit
This bit is used for signed arithmetic (two's complement). It indicates an overflow of the 7-bit magnitude which causes the sign bit (bit 7 of the result) to change state.
1 = Overflow occurred for signed arithmetic (in this arithmetic operation)
0 = No overflow occurred

bit 2 **Z:** Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC:** Digit Carry/$\overline{\text{Borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)(1)
1 = A carry-out from the 4th low-order bit of the result occurred
0 = No carry-out from the 4th low-order bit of the result

bit 0 **C:** Carry/$\overline{\text{Borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)(1)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Les microcontrôleurs

La mémoire EEPROM de données (256 Octets)

Le PIC 18F45K22 dispose de 256 octets (00_h à FF_h) de mémoire EEPROM de donnée (mémoire non volatile). Cette mémoire permet la lecture et l'écriture de données par l'intermédiaire de 4 ou 5 registres particuliers :

- EECON1 : registre de control 1
- EECON2 : registre de control 2
- EEDATA : registre de donnée
- EEADR : registre d'adresse (adressage sur 8 bits $\Rightarrow$ 256 octets)
- EEADRH : registre d'adresse (extensions à 1024 octets par ajout de 2 bits supplémentaires)

La procédure de lecture/écriture dans l'EEPROM est détaillée dans la datasheet du PIC, nous n'en parlerons pas ici. Si nous devions programmer en assembleur, il nous faudrait cependant connaître le déroulement des opérations.

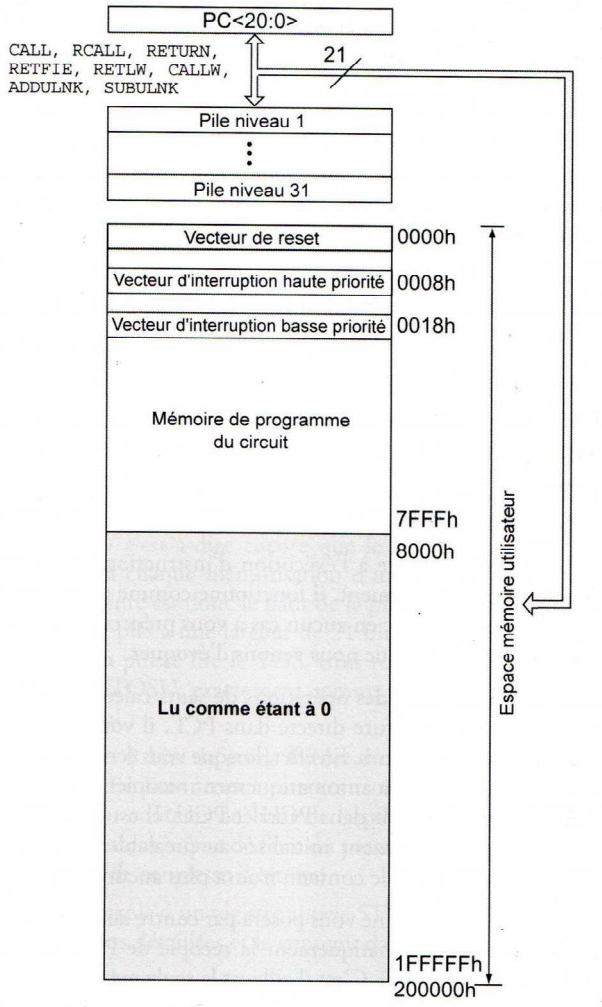
256 octets est une très petite mémoire pour stocker des données. Dans certaines cas, cela peut être suffisant mais très souvent on rajoute une mémoire de stockage externe (composant EEPROM ou carte Flash). Par exemple : datalogger.

Les microcontrôleurs

La mémoire Programme ou mémoire flash (32 ko)

Cette mémoire de 16 x 1024 mots de 16 bits sert à stocker le programme, mais elle est accessible par programme et peut donc être utilisée comme une extension de la mémoire EEPROM de données. Elle est non volatile (flash) et reprogrammable à souhait (enfin , presque…).

La mémoire programme s'étend linéairement de l'adresse 0000_h à l'adresse la plus élevées permise compte tenu de la taille mémoire du circuit choisi. Dans le cas du Pic 18F45K22, il s'agit de l'adresse $7FFF_h$. Au total, ce sont donc 32768 vecteurs de 8 bits que l'on peut adresser (32 ko). Mais comme chaque instruction est au minimum codée sur 2 octets, le compteur ordinal ou PC, évolue de 2 en 2.

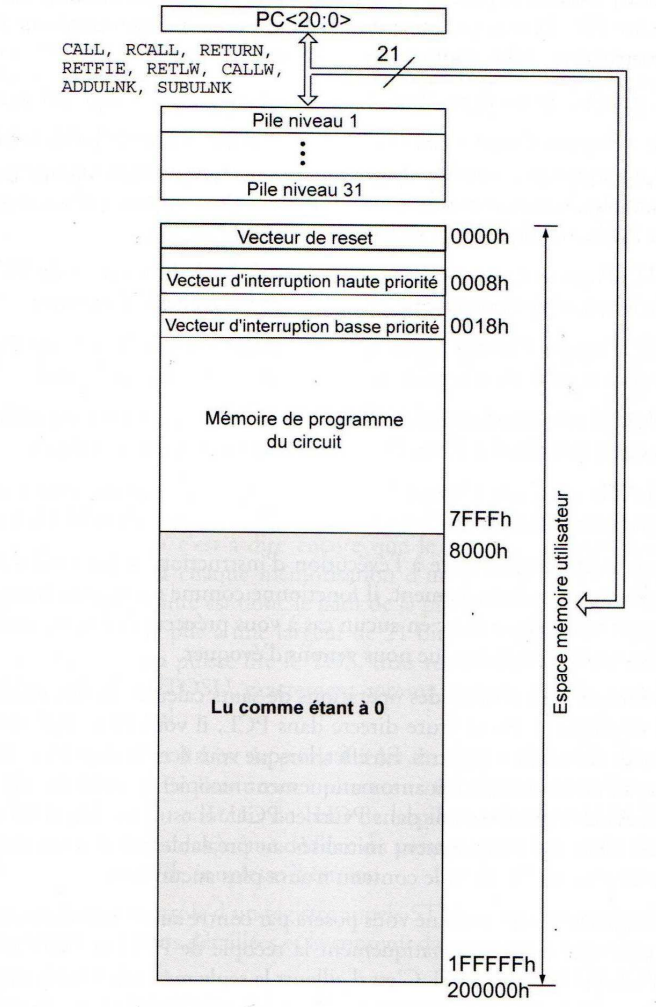


Les microcontrôleurs

La mémoire Programme ou mémoire flash (32 ko)

Chaque position de 16 bits contient une instruction. L'emplacement du programme peut se situer à n'importe quel endroit de la mémoire. Cependant il faut savoir que suite à un RESET ou lors de la mise sous tension, le PIC commence l'exécution à l'adresse 0000_h. De plus, lorsqu'il y a une interruption, le PIC va à l'adresse 0008_h (haute priorité) ou 0016_h (basse priorité) . Lorsque l'on programme en bas niveau (assembleur), il est donc nécessaire de bien organiser le programme si celui-ci utilise des interruptions.

Le programme exécutable par le PIC est implanté dans la mémoire flash à l'aide d'un programmateur sur lequel nous reviendrons en TP lorsque nous utiliserons la carte easypic7.



Les microcontrôleurs

Les ports d'E/S

Le PIC 18F45K22 dispose de 5 ports parallèles (de A à E) pour un total de 36 lignes d'Entrée/Sortie (E/S). Chaque port à ses particularités. Quasiment toutes les lignes d'E/S sont multiplexées avec d'autres ressources internes, les périphériques (UART etc.). En général, quand un périphérique est activé, les connections concernées ne peuvent plus utilisées comme E/S générale.

Chaque port parallèle est commandé par cinq registres (x=A, B, C, D ou E) :

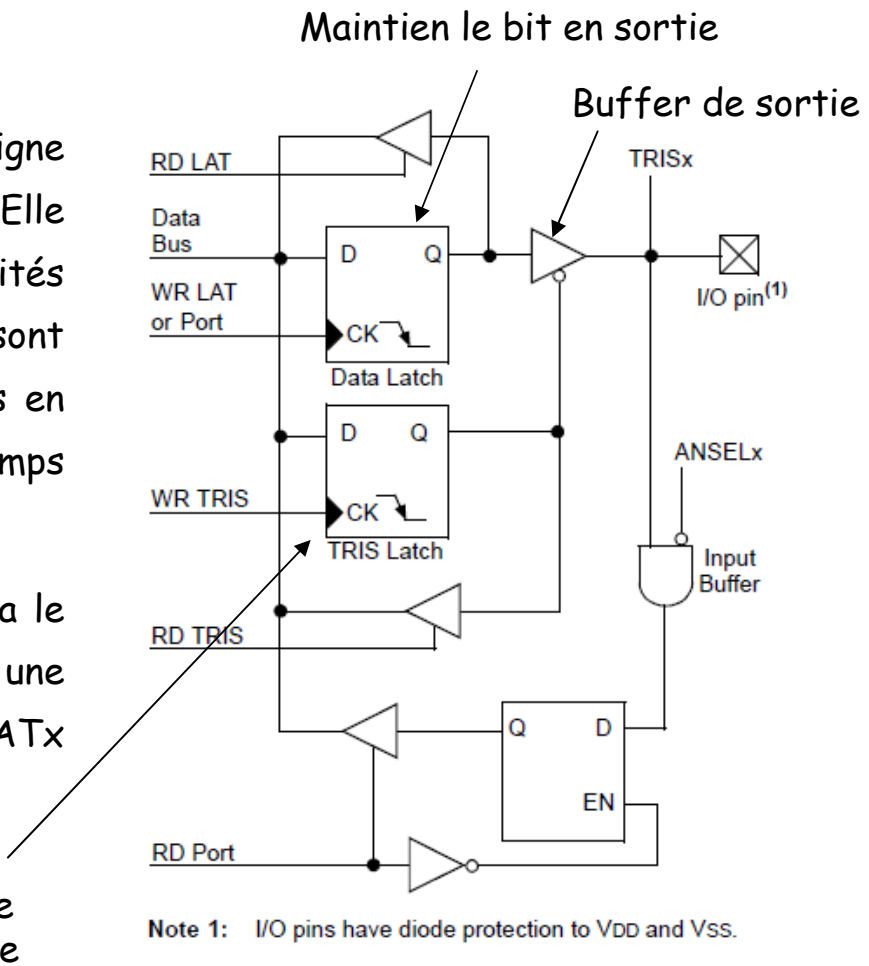
- un registre appelé PORTx, ou registre de données, fonctionnant en entrée/sortie logique
- un registre appelé LATx ou registre tampon (Latch en anglais), ne fonctionnant qu'en sortie
- un registre appelé TRISx, ou registre de sens de fonctionnement du port (entrée ou sortie)
- un registre appelé ANSELx pour le contrôle des entrées analogiques
- un registre appelé SLRCON permettant le contrôle du Slew Rate (vitesse de balayage, permet de limiter les problème d'IEM, Interférence ElectroMagnétique)

Les microcontrôleurs

Les ports d'E/S

La figure suivante précise la structure de chaque ligne de port au moyen d'un schéma de principe général. Elle permet de comprendre l'essentiel des fonctionnalités des ces lignes, à savoir que les données de sortie sont « latchées » (mémorisées), alors que les données en entrées ne le sont pas et sont donc lues en « temps réel ».

Pour lire une valeur logique en entrée, on utilisera le registre PORTx (PORTA par exemple). Pour écrire une valeur logique en sortie, on utilisera le registre LATx (LATA par exemple).

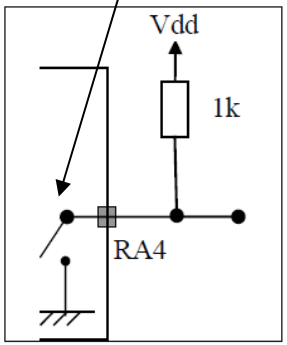


Les microcontrôleurs

Interrupteur = Transistor

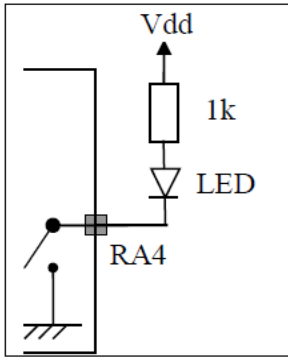
Remarque : les résistances de tirage (pull-up / pull-down)

Lorsqu'une sortie est à drain ouvert, pour l'utiliser comme sortie logique, il faut ajouter une résistance de pull-up externe. Le schéma de droite illustre le principe d'une sortie drain ouvert (ou collecteur ouvert) : si le port est positionnée à 0, l'interrupteur est fermé, la sortie est reliée à la masse, c'est un niveau bas. Si le port est placée à 1, l'interrupteur est ouvert, la sortie serait déconnectée s'il n'y avait pas la résistance externe qui place la sortie au niveau haut.



Contrairement à une sortie classique, le port en collecteur ouvert permet d'alimenter une charge avec une tension différente de celle du µcontrôleur (Vdd = 12V par exemple).

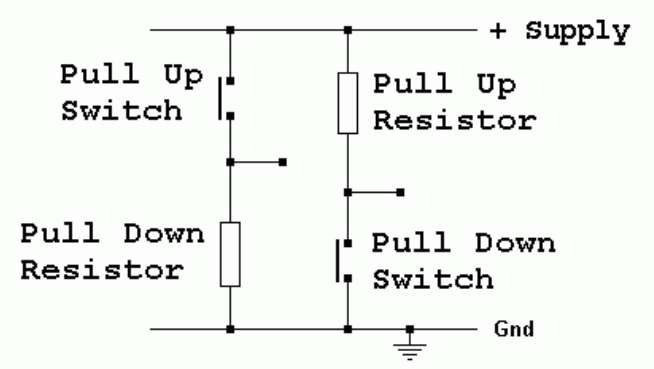
Si on veut utiliser ce type de port pour allumer une LED, on peut utiliser le schéma ci-dessous. Il faut juste remarquer que la logique est inversée, si on envoie 0 sur le port, l'interrupteur se ferme et la LED s'allume. Si on envoie 1, l'interrupteur s'ouvre et la LED s'éteint.



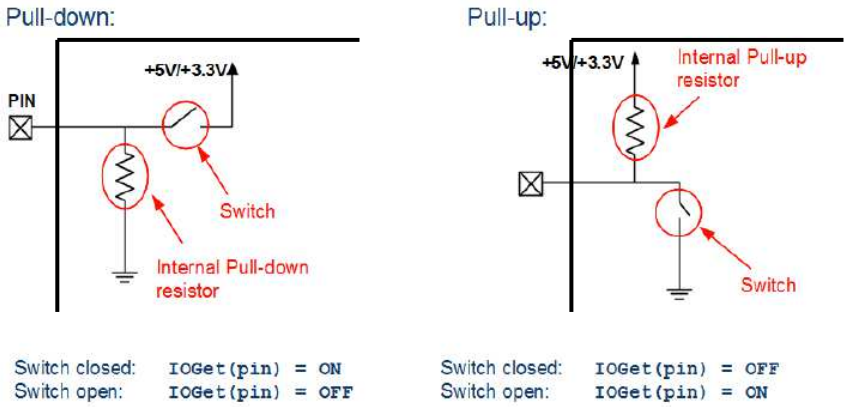
Les microcontrôleurs

Remarque : résistances de pull-up et pull-down

On utilise une résistance de pull-up lorsque l'interrupteur est connecté à la masse. S'il est connecté à l'alimentation, on utilisera une résistance de pull-down.



Dans certains cas, les résistances de pull-up ou pull-down sont directement intégrées dans le composant (par exemple, pull-up sur le PortB du PIC 18F45K22)



Les microcontrôleurs

A - Le port d' E/S PORTA

Le port A désigné par PORTA est un port bidirectionnel de 8 bits (RA0 à RA7). La configuration de direction se fait à l'aide du registre TRISA, positionner un bit de TRISA à 1 configure la broche correspondante de PORTA en entrée et inversement. Au départ toutes les broches sont configurées en entrée.

Les broches RA6 et RA7 peuvent être utilisées en E/S ou comme lignes d'entrée de l'oscillateur principal.

La ligne RA4 est quant à elle multiplexée avec l'entrée d'horloge externe du Timer 0 (T0CKL) et l'une des sorties du comparateur 1 (C1OUT). Elle est parfois nommée RA4/T0CKL/C1OUT.

Les autres lignes RA0 à RA3 et RA5 sont partagées avec les entrées du convertisseur analogiques/numérique (CAN), les entrées $V_{ref}+$ et $V_{ref}-$, la tension de référence de sortie du comparateur etc.

Les lignes RA0 à RA5 peuvent aussi être utilisées comme entrées ou sorties du comparateur.

Les microcontrôleurs

TABLE 10-1: PORTA I/O SUMMARY

Pin Name	Function	TRIS Setting	ANSEL Setting	Pin Type	Buffer Type	Description
RA0/C12IN0-/AN0	RA0	0	0	O	DIG	LATA<0> data output; not affected by analog input.
		1	0	I	TTL	PORTA<0> data input; disabled when analog input enabled.
	C12IN0-	1	1	I	AN	Comparators C1 and C2 inverting input.
	AN0	1	1	I	AN	Analog input 0.
RA1/C12IN1-/AN1	RA1	0	0	O	DIG	LATA<1> data output; not affected by analog input.
		1	0	I	TTL	PORTA<1> data input; disabled when analog input enabled.
	C12IN1-	1	1	I	AN	Comparators C1 and C2 inverting input.
	AN1	1	1	I	AN	Analog input 1.
RA2/C2IN+/AN2/ DACOUT/VREF-	RA2	0	0	O	DIG	LATA<2> data output; not affected by analog input; disabled when DACOUT enabled.
		1	0	I	TTL	PORTA<2> data input; disabled when analog input enabled; disabled when DACOUT enabled.
	C2IN+	1	1	I	AN	Comparator C2 non-inverting input.
	AN2	1	1	I	AN	Analog output 2.
	DACOUT	x	1	O	AN	DAC Reference output.
	VREF-	1	1	I	AN	A/D reference voltage (low) input.
RA3/C1IN+/AN3/ VREF+	RA3	0		O	DIG	LATA<3> data output; not affected by analog input.
		1	0	I	TTL	PORTA<3> data input; disabled when analog input enabled.
	C1IN+	1	1	I	AN	Comparator C1 non-inverting input.
	AN3	1	1	I	AN	Analog input 3.
	VREF+	1	1	I	AN	A/D reference voltage (high) input.
RA4/CCP5/ C1OUT/SRQ/ T0CKI	RA4	0	—	O	DIG	LATA<4> data output.
		1	—	I	ST	PORTA<4> data input; default configuration on POR.
	CCP5	0	—	O	DIG	CCP5 Compare output/PWM output, takes priority over RA4 output.
		1	—	I	ST	Capture 5 input/Compare 5 output/ PWM 5 output.
	C1OUT	0	—	O	DIG	Comparator C1 output.
	SRQ	0	—	O	DIG	SR latch Q output; take priority over CCP 5 output.
	T0CKI	1	—	I	ST	Timer0 external clock input.
RA5/C2OUT/ SRNQ/SS1/ HLVDIN/AN4	RA5	0	0	O	DIG	LATA<5> data output; not affected by analog input.
		1	0	I	TTL	PORTA<5> data input; disabled when analog input enabled.
	C2OUT	0	0	O	DIG	Comparator C2 output.
	SRNQ	0	0	O	DIG	SR latch $\overline{Q}$ output.
	$\overline{SS1}$	1	0	I	TTL	SPI slave select input (MSSP1).
	HLVDIN	1	1	I	AN	High/Low-Voltage Detect input.
	AN4	1	1	I	AN	A/D input 4.

Legend: AN = Analog input or output; TTL = TTL compatible input; HV = High Voltage; OD = Open Drain; XTAL = Crystal; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I^2C^{TM} = Schmitt Trigger input with I^2C.

TABLE 10-1: PORTA I/O SUMMARY (CONTINUED)

Pin Name	Function	TRIS Setting	ANSEL Setting	Pin Type	Buffer Type	Description
RA6/CLKO/OSC2	RA6	0	—	O	DIG	LATA<6> data output; enabled in INTOSC modes when CLKO is not enabled.
		1	—	I	TTL	PORTA<6> data input; enabled in INTOSC modes when CLKO is not enabled.
	CLKO	x	—	O	DIG	In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
	OSC2	x	—	O	XTAL	Oscillator crystal output; connects to crystal or resonator in Crystal Oscillator mode.
RA7/CLKI/OSC1	RA7	0	—	O	DIG	LATA<7> data output; disabled in external oscillator modes.
		1	—	I	TTL	PORTA<7> data input; disabled in external oscillator modes.
	CLKI	x	—	I	AN	External clock source input; always associated with pin function OSC1.
	OSC1	x	—	I	XTAL	Oscillator crystal input or external clock source input ST buffer when configured in RC mode; CMOS otherwise.

Legend: AN = Analog input or output; TTL = TTL compatible input; HV = High Voltage; OD = Open Drain; XTAL = Crystal; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I^2C^{TM} = Schmitt Trigger input with I^2C.

TABLE 10-2: REGISTERS ASSOCIATED WITH PORTA

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
ANSELA	—	—	ANSA5	—	ANSA3	ANSA2	ANSA1	ANSA0	154
CM1CON0	C1ON	C1OUT	C1OE	C1POL	C1SP	C1R	C1CH<1:0>		317
CM2CON0	C2ON	C2OUT	C2OE	C2POL	C2SP	C2R	C2CH<1:0>		317
LATA	LATA7	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0	158
VREFCON1	DACEN	DACLPS	DACOE	—	DACPSS<1:0>		—	DACNSS	347
VREFCON2	—	—	—	DACR<4:0>					348
HLVDCON	VDIRMAG	BGVST	IRVST	HLVDEN	HLVDL<3:0>				349
PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	153
SLRCON	—	—	—	SLRE	SLRD	SLRC	SLRB	SLRA	158
SRCON0	SRLEN	SRCLK<2:0>			SRQEN	SRNQEN	SRPS	SRPR	340
SSP1CON1	WCOL	SSPOV	SSPEN	CKP	SSPM<3:0>				260
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS<2:0>			159
TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	156

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for PORTA.

TABLE 10-3: CONFIGURATION REGISTERS ASSOCIATED WITH PORTA

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
CONFIG1H	IESO	FCMEN	PRICLKEN	PLLCFG	FOSC<3:0>				357

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for PORTA.

Les microcontrôleurs

B - Le port d' E/S PORTB

- Le port B désigné par PORTB est un port bidirectionnel de 8 bits (RB0 à RB7).
- Présence, sur toutes les lignes de ce port, de résistances de pull-up (résistance de tirage de niveau haut) de forte valeur. Elle peuvent être invalidées simultanément par la mise au niveau 1 du bit $\overline{RPBU}$ du registre INTCON2. Par contre, lorsqu'elles sont validées par la mise à 0 de ce même bit, cette validation n'est effective que pour les lignes du port B qui sont placées en entrée.
- La configuration de direction se fait à l'aide du registre TRISB, positionner un bit de TRISB à 1 configure la broche correspondante de PORTB en entrée et inversement. Au départ toutes les broches sont configurées en entrée.
- En entrée, une quelconque des lignes RB4 à RB7 peut déclencher, sur changement d'état, une interruption (scrutation d'un clavier par exemple ; le µcontrôleur reste en veille tant qu'aucune touche n'est activée). Cette fonction peut être validée au moyen du bit RBIF du registre INTCON.
- Comme tous les autres ports, le port B peut être partagée avec une ou plusieurs autres ressources internes.

Les microcontrôleurs

C - Les ports d' E/S C et D

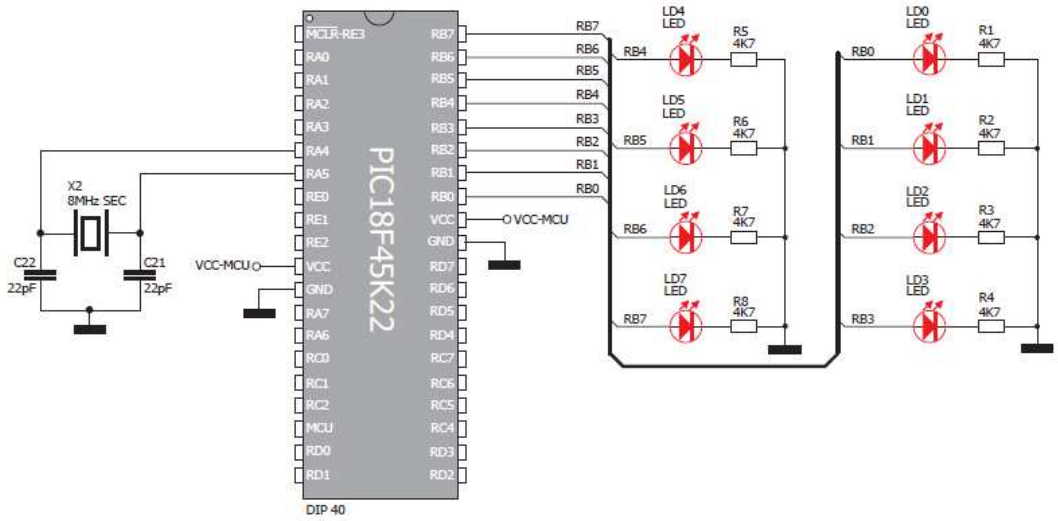
- Les ports C et D sont des ports bidirectionnel de 8 bits (RC0 à RC7).
- La configuration de direction se fait à l'aide des registres TRISC et TRISD, positionner un bit de TRISC (TRISD) à 1 configure la broche correspondante de PORTC (PORTD) en entrée et inversement. Au départ toutes les broches sont configurées en entrée.
- Toutes les broches du port C (port D) peuvent être utilisées soit comme E/S normales soit comme broches d'accès à différents modules comme les timers, les modules de comparaison et de capture CCP1/2, le port I2C ou encore le port série (EUSART).
- Pour l'utilisation d'une broche du port C ou D comme E/S normale, il faut s'assurer qu'elle n'a pas été affectée à un de ces modules.

Les microcontrôleurs

E - Le port d' E/S PORTE

- PORTE contient seulement 3 bits RE0, RE1 et RE2. Les 3 sont configurables en entrée ou en sortie à l'aide des bits 0, 1 ou 2 du registre TRISE.

Un exemple simple : clignotement des LED du port B (mode pull-up désactivé)



Connexions sur la carte EasyPic

```
void main() {
  // set PORTB to be digital output
  TRISB = 0;

  // Turn OFF LEDs on PORTB
  LATB = 0;

  while(1) {
    // Toggle LEDs on PORTB
    LATB = ~LATB;

    // Delay 1000 ms
    Delay_ms(1000);
  }
}
```

Programme C

Les microcontrôleurs

Les Interruptions :

Une interruption(*) provoque l'arrêt du programme principal pour aller exécuter une procédure d'interruption. A la fin de cette procédure, le microcontrôleur reprend le programme principal à l'endroit où il l'a laissé.

Le PIC 18F45K22 possède plusieurs sources d'interruptions (primaires ou périphériques) qui peuvent être configurées selon un niveau de priorité haute (@ 0008_h) ou basse (@ 0018h), grâce au registre IPRx. Seule l'interruption INT0 ne possède pas de niveau de priorité (toujours élevée).

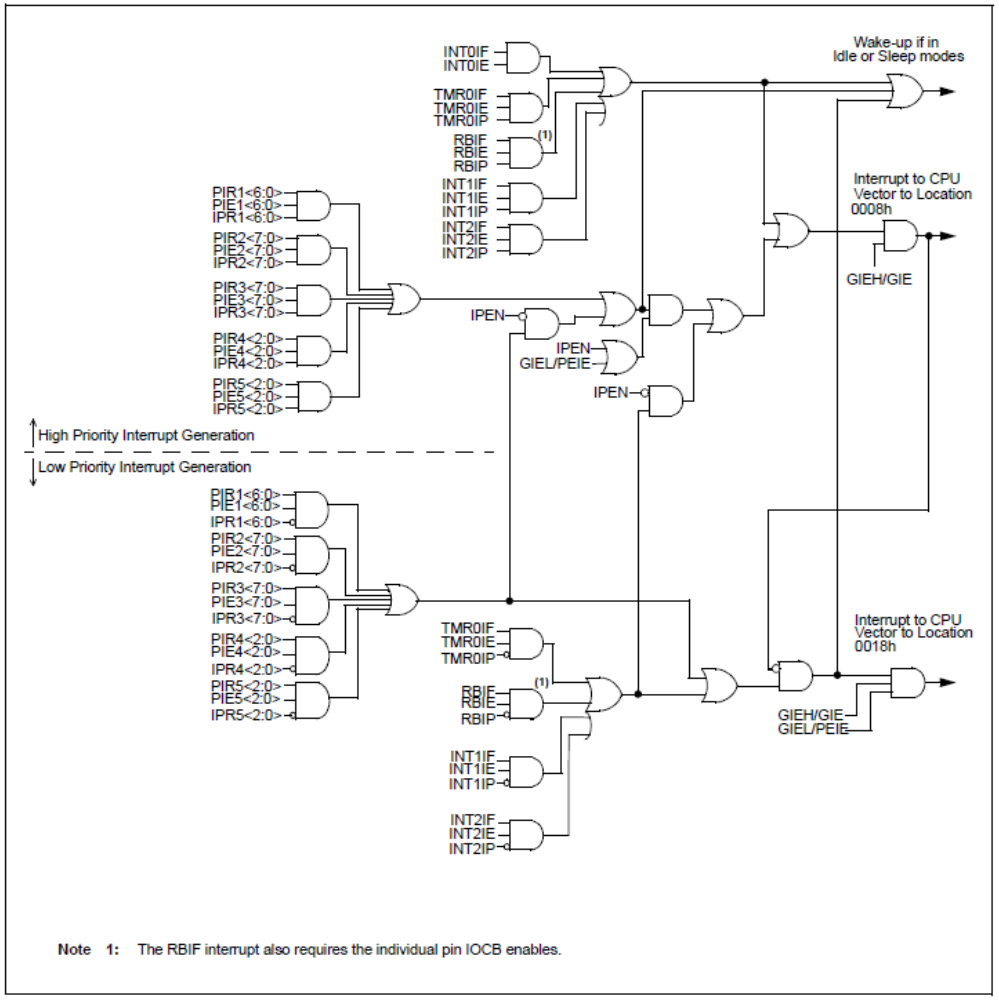
En général, à chaque interruption sont associés trois bits, un bit de validation (PIE), un bit drapeau (PIR) un bit de priorité (IPR). Le premier permet d'autoriser ou non l'interruption, le second permet au programmeur de savoir de quelle interruption il s'agit et le troisième donne le niveau de priorité.

(*) Evènement provoqué par exemple à : la fin d'une conversion A/N, la fin de programmation d'un octet dans l'EEPROM, la réception d'une information, la détection d'un front, etc

Les microcontrôleurs

Schéma de principe de la circuiterie de prise en compte des interruptions

FIGURE 9-1: PIC18 INTERRUPT LOGIC



Note 1: The RBIF interrupt also requires the individual pin IOCB enables.

Les microcontrôleurs

Les Interruptions :

Il existe 2 niveaux de fonctionnement (bit 7 du registre RCON - IPEN):

- en mode non hiérarchisé IPEN.RCON = 0, dans ce cas toutes les interruptions ont la même priorité (haute)
- en mode hiérarchisé IPEN.RCON = 1, dans ce cas les 2 niveaux de priorité sont activés, par défaut en mode haute priorité. Il vous appartient donc de définir individuellement les interruptions de basse priorité, registre IPRx. Une interruption de priorité haute peut interrompre une interruption de niveau bas.

Par ailleurs, il est possible d'activer toutes les interruptions ou seulement les interruptions primaires (si IPEN=0) ou de niveau bas (si IPEN=1) en jouant sur les bits 6 et 7 du registre INTCON.

Les microcontrôleurs

Les Interruptions : registres associés

Gestion globale + interruptions primaires

INTCON : INTerrupt CONtrol

IPR : Interrupt Priority Register

PIE : Peripheral Interrupt Enable

PIR : Peripheral Interrupt Register

Gestion des interruptions périphériques

RCON : Reset Control Register →

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ANSELB	—	—	ANSB5	ANSB4	ANSB3	ANSB2	ANSB1	ANSB0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF
IOCB	IOCB7	IOCB6	IOCB5	IOCB4	—	—	—	—
IPR1	—	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCL1IP	HLVDIP	TMR3IP	CCP2IP
IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	CTMUIP	TMR5GIP	TMR3GIP	TMR1GIP
IPR4	—	—	—	—	—	CCP5IP	CCP4IP	CCP3IP
IPR5	—	—	—	—	—	TMR6IP	TMR5IP	TMR4IP
PIE1	—	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCL1IE	HLVDIE	TMR3IE	CCP2IE
PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	CTMUIE	TMR5GIE	TMR3GIE	TMR1GIE
PIE4	—	—	—	—	—	CCP5IE	CCP4IE	CCP3IE
PIE5	—	—	—	—	—	TMR6IE	TMR5IE	TMR4IE
PIR1	—	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCL1IF	HLVDIF	TMR3IF	CCP2IF
PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	CTMUIF	TMR5GIF	TMR3GIF	TMR1GIF
PIR4	—	—	—	—	—	CCP5IF	CCP4IF	CCP3IF
PIR5	—	—	—	—	—	TMR6IF	TMR5IF	TMR4IF
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
RCON	IPEN	SBOREN	—	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for Interrupts.

Les microcontrôleurs

Le registre INTCON

REGISTER 9-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

Lorsque la hiérarchisation des interruptions n'est pas autorisée (bit IPEN=0), le bit7 du registre INTCON s'appelle GIE et le bit6 s'appelle PEIE

- Toutes les interruptions peuvent être validées/interdites par le bit INTCON.GIE
- Toutes les interruptions périphériques peuvent être validées/interdites par le bit INTCON.PEIE
- Chaque interruption peut être validée/interdite par son bit de validation individuel

En résumé, pour valider une interruption périphérique (par exemple) en mode non hiérarchisé, il faut positionner 4 bits:

- IPEN à 0
- GIE, PEIE et le bit individuel de l'interruption (PIE) à 1

Les microcontrôleurs

Le registre INTCON

REGISTER 9-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

Lorsque la hiérarchisation des interruptions est autorisée (bit IPEN=1), le bit7 du registre INTCON s'appelle GIEH et le bit6 s'appelle GIEL.

- Toutes les interruptions peuvent être validées/interdites par le bit INTCON.GIEH
- Toutes les interruptions de priorité basse peuvent être validées/interdites par le bit INTCON.GIEL
- Chaque interruption peut être validée/interdite par son bit de validation individuel

En résumé, pour valider une interruption périphérique (par exemple) en basse priorité, il faut positionner 5 bits:

- IPEN à 1
- Le niveau de priorité de l'interruption à 0 (basse priorité)
- GIEH, GIEL et le bit individuel de l'interruption à 1

Les microcontrôleurs

Les Timers

Un timer est un compteur prépositionnable qui compte ou décompte au rythme d'une horloge interne ou externe. Les PICs 18F45K22 disposent de 7 timers, 3 de 8 bits et 4 de 16 bits.

- Le timer 0 (8 ou 16 bits): il peut être incrémenté par des impulsions extérieures via la broche (TOCKI/ RA4) ou par l'horloge système (Fosc/4).

- Les timers 1/3/5 (16 bits): ils peuvent être incrémentés soit par l'horloge interne(Fosc), par l'horloge système (Fosc/4), par des impulsions sur les broches TxCKI (RC0/RB5/RC2) ou encore par un oscillateur (RC ou quartz) connecté sur les broches SOSC0/RC0 et SOSC1/RC1.

- Le timers 2/4/6 (8 bits) : ils sont incrémentés par l'horloge système (Fosc/4), celle-ci peut être prédivisée par 4 ou 16.

Tous ces timers peuvent déclencher une interruption interne.

Les microcontrôleurs

Le TIMER0 :

C'est le plus ancien des timers implantés dans les PICs, son ancienne appellation était RTC, pour Real Time Clock (horloge temps réelle). C'est aussi le plus simple car il est autonome, c'est-à-dire qu'il n'est pas associé à d'autres ressources internes.

On peut s'en servir pour générer des événements périodiques, comme le clignotement de LEDs ou l'incrémentation de variables (secondes, minutes …).

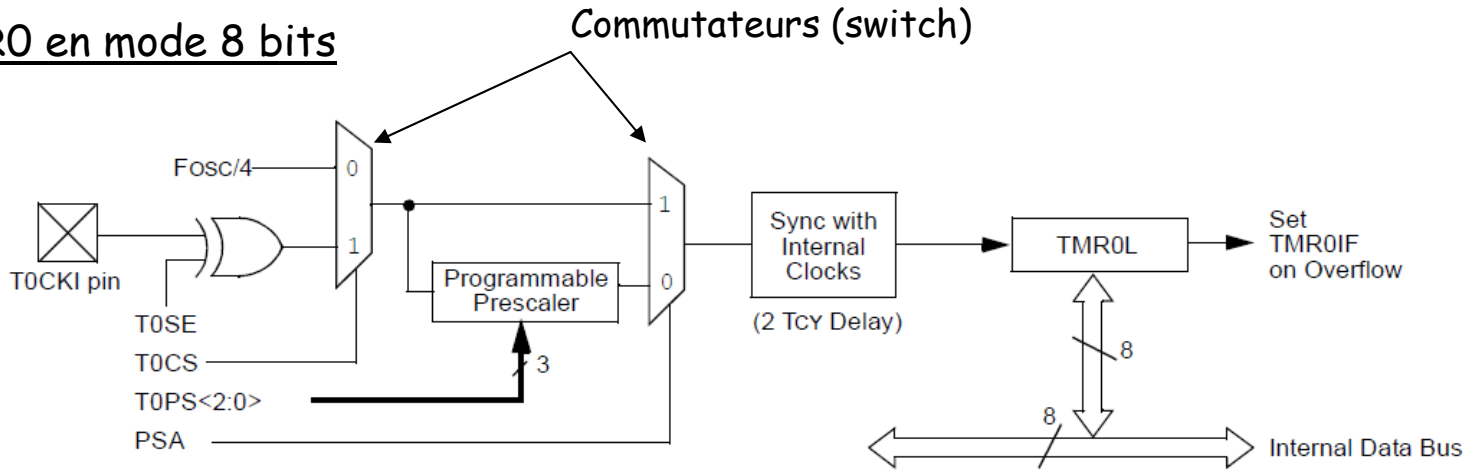
Il peut fonctionner en mode 8 bits (comptage de 00 à FF) ou en mode 16 bits (comptage de 0000 à FFFF).

Il est incrémenté soit par l'horloge système (Fosc/4), soit par un front montant ou descendant appliqué sur la broche TOCKI/RA4.

Il est configuré par les registres T0CON, TMR0L, TMR0H et les registres INTCON, INTCON2 (interruptions) et TRISA (broche RA4).

Les microcontrôleurs

Le TIMER0 en mode 8 bits



- Un premier commutateur, commandé par le bit T0CS, permet de sélectionner l'origine de la source (horloge système ou front sur la broche T0CKL).
- Le ou exclusif, via le bit T0SE, permet de sélectionner le front montant ou descendant sur la broche T0CKL (lorsque cette source est activée).
- Le module « Programmable Prescaler » permet de fixer le taux de division de l'horloge. Il est commandé par 3 bits, T0PS<2:0>.
- Le second commutateur, commandé par le bit PSA, permet de sélectionner ou non le prédiviseur.
- Le registre TMR0L (sur 8 bits) est accessible en lecture/écriture. Il contient la valeur courante du comptage et doit au préalable être chargé par une valeur initiale.

Les microcontrôleurs

Les registres associés au TIMER0

TABLE 11-1: REGISTERS ASSOCIATED WITH TIMER0

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	116
INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	117
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS<2:0>			159
TMR0H	Timer0 Register, High Byte								—
TMR0L	Timer0 Register, Low Byte								—
TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	156

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used by Timer0.

TMR0L seul : si compteur 8 bits

TMR0L + TMR0H : si compteur 16 bits

Principe de fonctionnement : quand le contenu du timer 0 passe de FF à 00 (si mode 8 bits, registre TMR0L), le bit TMR0IF passe à 1 pour signaler un débordement ; si le bit TMR0IE est à 1, alors une interruption timer 0 est déclenchée.

Les microcontrôleurs

Le registre T0CON

REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS<2:0>		
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

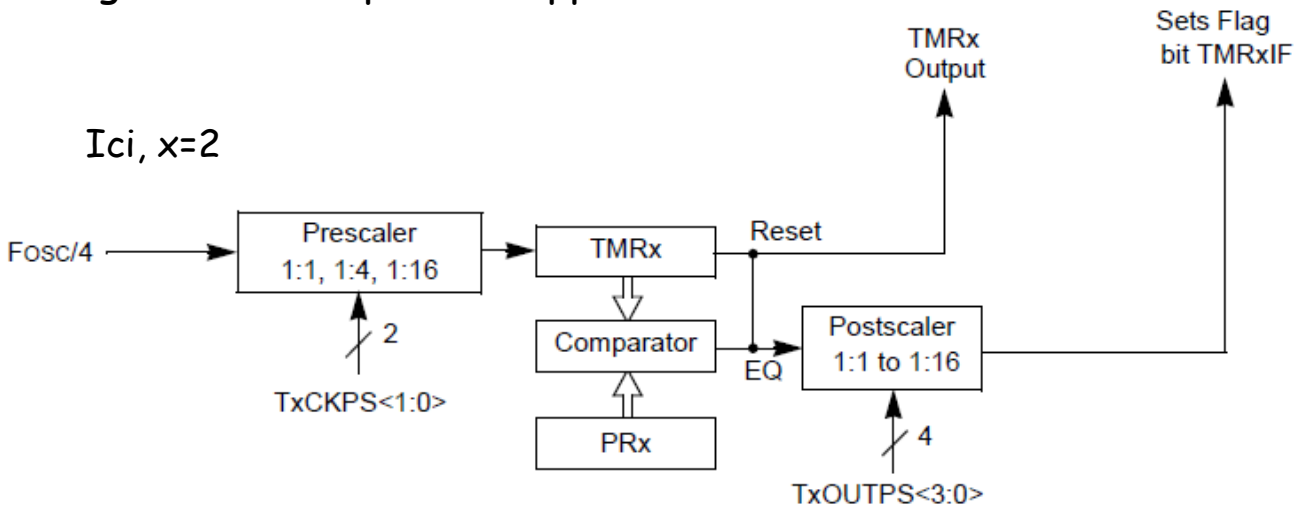
bit 2-0 **T0PS<2:0>**: Timer0 Prescaler Select bits
111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value

Valeurs du prédiviseur
De 2 à 256

Les microcontrôleurs

Le TIMER2 (/4/6)

C'est un timer 8 bits, son horloge ne peut être que l'horloge système (Fosc/4). Il est composé d'un registre 8 bits appelé TMR2, associé à un prédiviseur, à un postdiviseur ainsi qu'à un registre dit de période appelé PR2.



Remarque : La sortie du comparateur d'égalité entre le contenu de TMR2 et PR2 peut être utilisé par d'autres ressources internes. En particulier, par le module de modulation de largeur d'impulsions (PWM).

Les microcontrôleurs

Les registres associés au TIMER2 (/4/6)

TABLE 13-1: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER2/4/6

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
CCPTMRS0	C3TSEL<1:0>		—	C2TSEL<1:0>		—	C1TSEL<1:0>		208
CCPTMRS1	—	—	—	—	C5TSEL<1:0>		C4TSEL<1:0>		208
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	116
IPR1	—	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP	128
IPR5	—	—	—	—	—	TMR6IP	TMR5IP	TMR4IP	131
PIE1	—	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE	124
PIE5	—	—	—	—	—	TMR6IE	TMR5IE	TMR4IE	127
PIR1	—	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	119
PIR5	—	—	—	—	—	TMR6IF	TMR5IF	TMR4IF	123
PMD0	UART2MD	UART1MD	TMR6MD	TMR5MD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	55
PR2	Timer2 Period Register								—
PR4	Timer4 Period Register								—
PR6	Timer6 Period Register								—
T2CON	—	T2OUTPS<3:0>				TMR2ON	T2CKPS<1:0>		172
T4CON	—	T4OUTPS<3:0>				TMR4ON	T4CKPS<1:0>		172
T6CON	—	T6OUTPS<3:0>				TMR6ON	T6CKPS<1:0>		172
TMR2	Timer2 Register								—
TMR4	Timer4 Register								—
TMR6	Timer6 Register								—

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used by Timer2/4/6.

Les microcontrôleurs

Le registre TxCON

X=2 ici

REGISTER 13-1: TxCON: TIMER2/TIMER4/TIMER6 CONTROL REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TxOUTPS<3:0>				TMRxON	TxCKPS<1:0>	
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7 **Unimplemented:** Read as '0'

bit 6-3 **TxOUTPS<3:0>:** TimerX Output Postscaler Select bits

Valeurs du postdiviseur

De 1 à 16

0000 = 1:1 Postscaler
0001 = 1:2 Postscaler
0010 = 1:3 Postscaler
0011 = 1:4 Postscaler
0100 = 1:5 Postscaler
0101 = 1:6 Postscaler
0110 = 1:7 Postscaler
0111 = 1:8 Postscaler
1000 = 1:9 Postscaler
1001 = 1:10 Postscaler
1010 = 1:11 Postscaler
1011 = 1:12 Postscaler
1100 = 1:13 Postscaler
1101 = 1:14 Postscaler
1110 = 1:15 Postscaler
1111 = 1:16 Postscaler

bit 2 **TMRxON:** TimerX On bit

1 = TimerX is on
0 = TimerX is off

bit 1-0 **TxCKPS<1:0>:** Timer2-type Clock Prescale Select bits

Valeurs du prédiviseur

1 – 4 - 16

00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

Les microcontrôleurs

Le Watchdog Timer WDT (Chien de garde)

C'est un compteur 8 bits incrémenté en permanence (même si le µC est en mode sleep) par une horloge RC intégrée indépendante de l'horloge système.

Lorsqu'il déborde (WDT TimeOut), deux situations sont possibles :

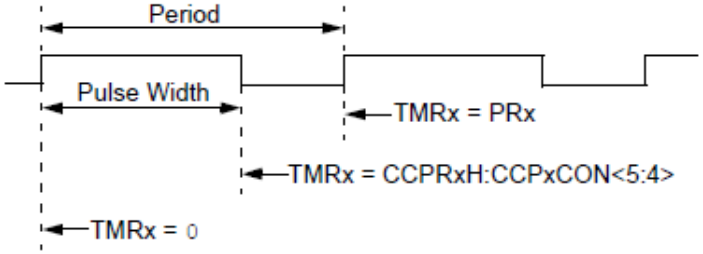
- Si le µC est en fonctionnement normal, le WDT time-out provoque un RESET. Ceci permet d'éviter les situation de blocage du microcontrôleur par un processus indésirable non contrôlé

- Si le µC est en mode SLEEP, le WDT time-out provoque un WAKE-UP, l'exécution du programme continue normalement là où elle s'est arrêtée avant de rentrer en mode SLEEP. Cette situation est souvent exploitée pour réaliser des temporisations

Les microcontrôleurs

La modulation PWM (Pulse Width Modulation)

Principe : un signal PWM est caractérisé par sa période et son rapport cyclique.



Period = Période

Pulse Width = Largeur d'impulsion

Duty Cycle Ratio = Rapport cyclique = Pulse Width/Period

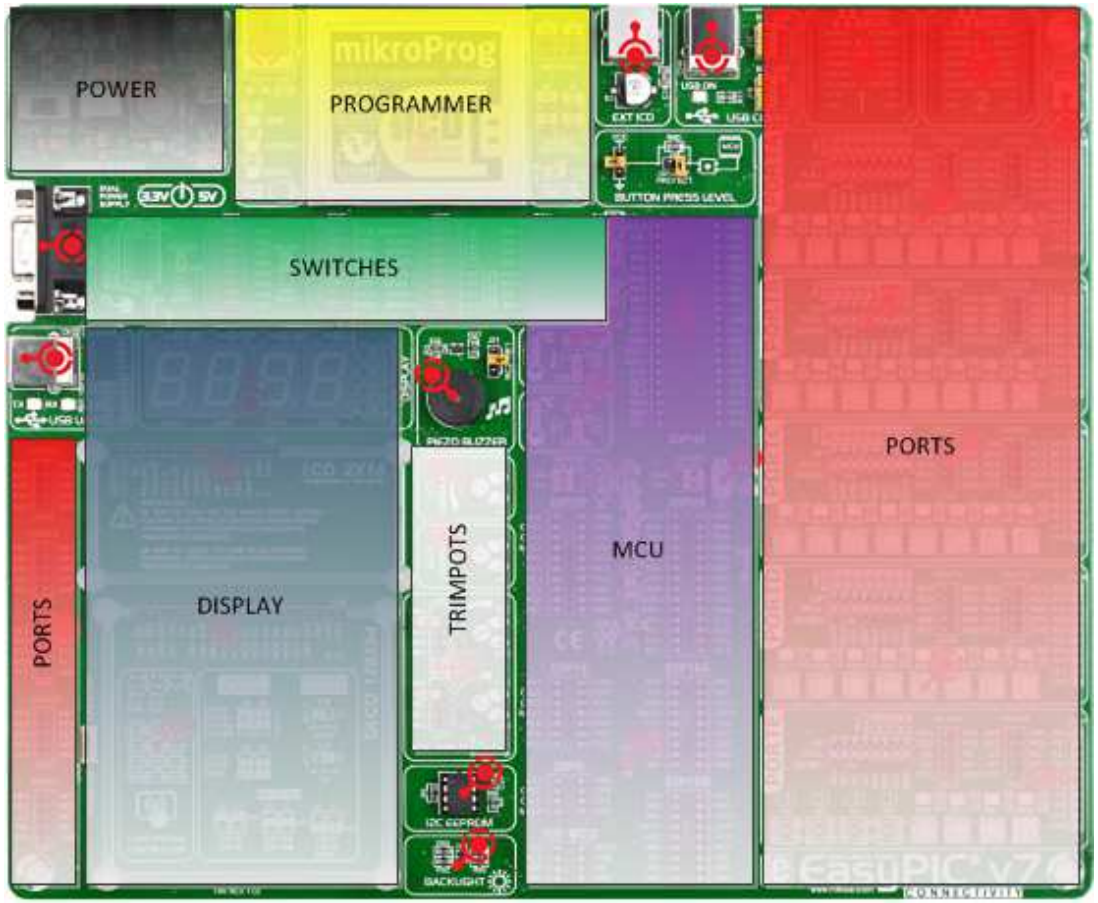
$$\text{Période} = (PRx+1)\cdot 4\cdot T_{OSC}\cdot(\text{Contenu du prédiviseur de TMRx})$$
$$\text{Largeur} = DCx\cdot T_{OSC}\cdot(\text{Contenu du prédiviseur de TMRx})$$
$$\text{Rapport Cyclique} = DCx/[4\cdot(PRx+1)]$$
$$\text{où } DCx = CCPRxH:CCPxCON<5:4>$$

Si utilisation d'un module CCP du MCU

L'une des application classique de cette modulation est le contrôle de moteurs (mode PWM étendu ou EPWM). Mais elle peut également être mise en œuvre pour contrôler le Buzzer sur la carte EasyPic7 : note (période du signal et donc fréquence), volume (rapport cyclique). Deux possibilités pour cela : soit en utilisant l'un des modules CCP du MCU ou en émulant la forme du signal voulu sur l'une des broches du MCU.

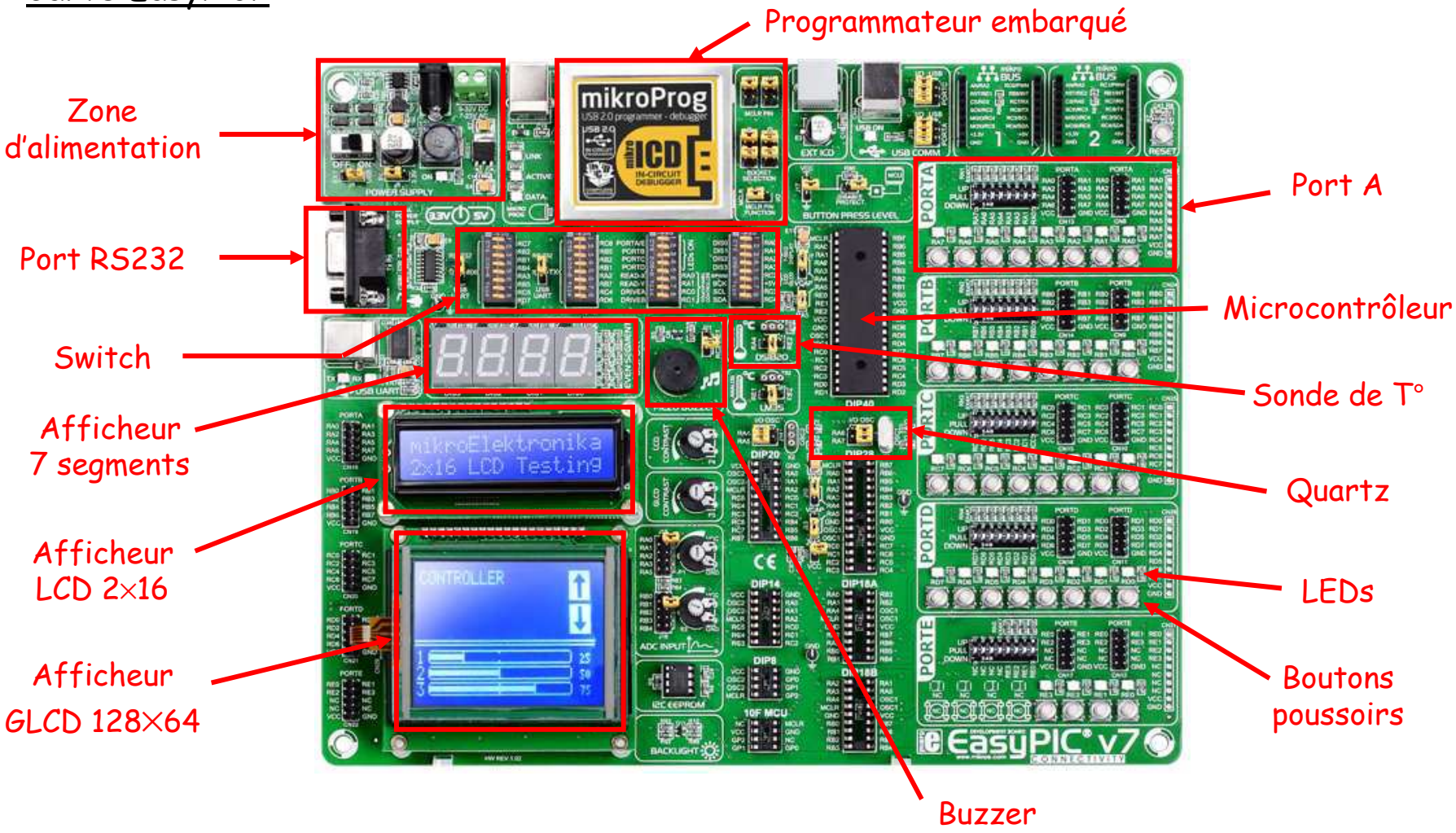
Les microcontrôleurs

Carte EasyPic7 : durant les TP, vous mettrez en œuvre la carte EasyPic7 de MikroElektronika sur laquelle est installé un *µC* PIC 18F45K22.



Les microcontrôleurs

Carte EasyPic7

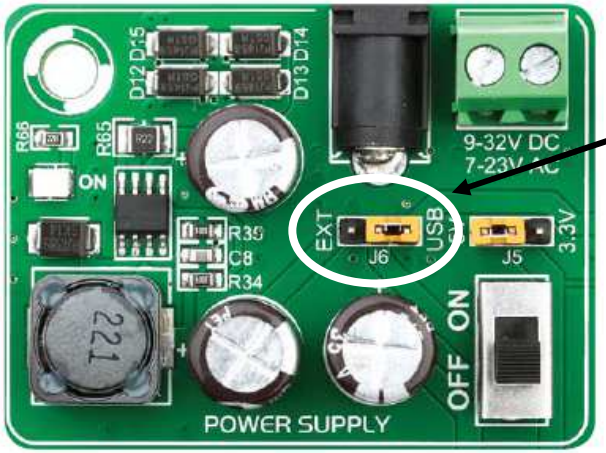


Les microcontrôleurs

Alimentation

2 possibilités pendant les TP :

- par le câble USB
- grâce au bloc d'alimentation externe



How to power the board?

1. With USB cable

Set J6 jumper to USB position

To power the board with USB cable, place jumper **J6** in USB position and place jumper **J5** in 5V or 3.3V position. You can then plug in the USB cable as shown on images 1 and 2, and turn the power switch ON.

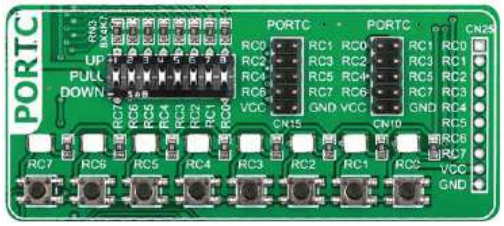
2. Using adapter

Set J6 jumper to EXT position

To power the board via adapter connector, place jumper **J6** in EXT position, and place jumper **J5** in 5V or 3.3V position. You can then plug in the adapter cable as shown on images 3 and 4, and turn the power switch ON.

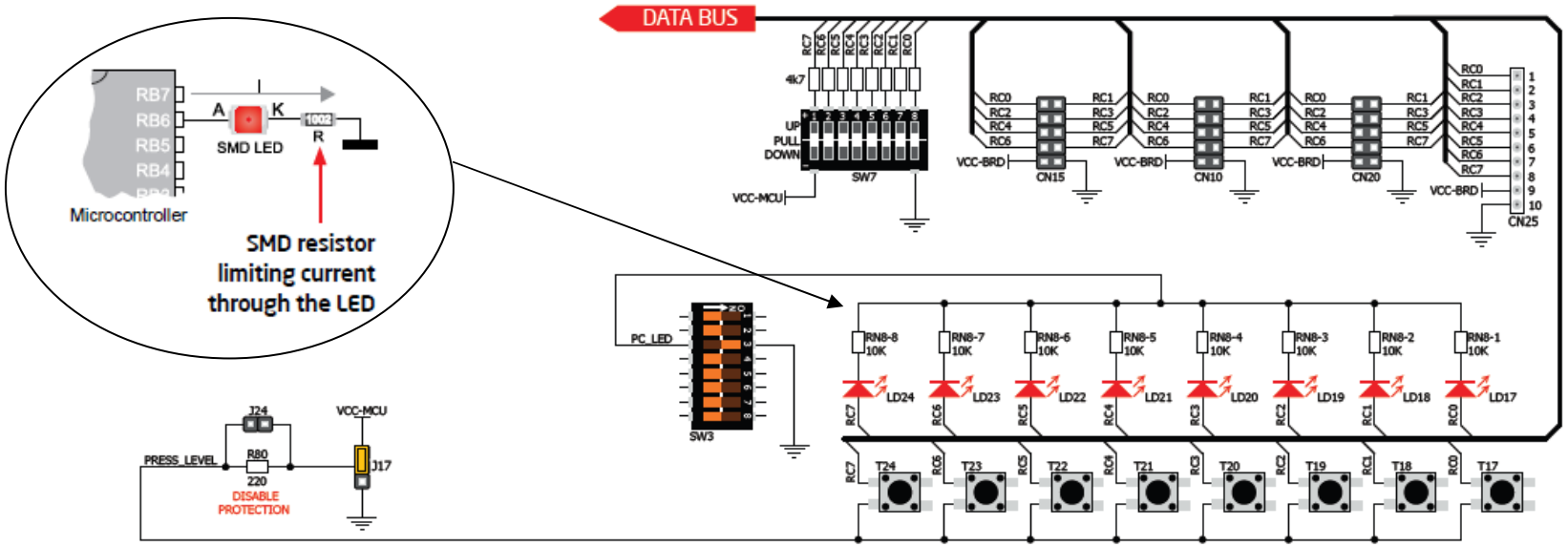
Les microcontrôleurs

Les DELs et boutons poussoirs



DEL : Diode Electro-Luminescentes (LED en anglais)

Chaque port (de A à E), est connecté à un réseau de résistances pull-up/pull-down (4,7 kΩ - switch SW7), à des connecteurs pour cartes filles, à des DELs (validation) et boutons poussoirs. Pour allumer une LED, il faudra la valider grâce au switch SW3 et appliquer un niveau logique haut « 1 » sur la broche correspondante du PIC. Pour utiliser les boutons poussoirs, il faudra positionner le cavalier J17 sur VCC (niveau haut) ou la masse (niveau bas).



Les microcontrôleurs

L'afficheur 7 segments – 4 digits

Chaque afficheur 7-segments est constitué de 7+1 DELs qui permettent d'afficher les chiffres de 0 à 9, quelques lettres de l'alphabet et le séparateur décimal (DP).

RA0 à RA3 : sélection de l'afficheur (*portA*)

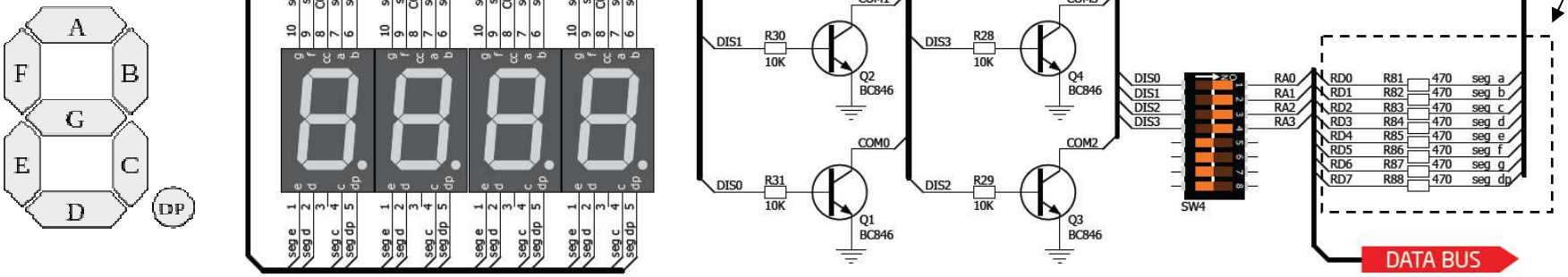
RD0 à RD6 : activation des 7 segments (*portD*)

RD 7 : activation du point DP (*portD*)

Segment b

Segment a

	Programmation								
Affichage	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Héxadecimal
0	0	0	1	1	1	1	1	1	0x3F
1	0	0	0	0	0	1	1	0	0x06
2	0	1	0	1	1	0	1	1	0x5B
3	0	1	0	0	1	1	1	1	0x4F
4	0	1	1	0	0	1	1	0	0x66
5	0	1	1	0	1	1	0	1	0x6D
6	0	1	1	1	1	1	0	1	0x7D
7	0	0	0	0	0	1	1	1	0x07
8	0	1	1	1	1	1	1	1	0x7F
9	0	1	1	0	1	1	1	1	0x6F
A	0	1	1	1	0	1	1	1	0x77
B	0	1	1	1	1	1	0	0	0x7C
C	0	0	1	1	1	0	0	1	0x39
D	0	1	0	1	1	1	1	0	0x5E
E	0	1	1	1	1	0	0	1	0x79
F	0	1	1	1	0	0	0	1	0x71



Les microcontrôleurs

L'afficheur LCD 2×16 (2 lignes, 16 colonnes, soit 32 caractères)

- Connexion sur le port RB (RB0 à RB5)
- Rétroéclairage possible grâce au switch SW4 :

  rétroéclairage maximal : SW4-6 on

  rétroéclairage réglable : SW4-5 on (connexion au port RC2 du MCU)

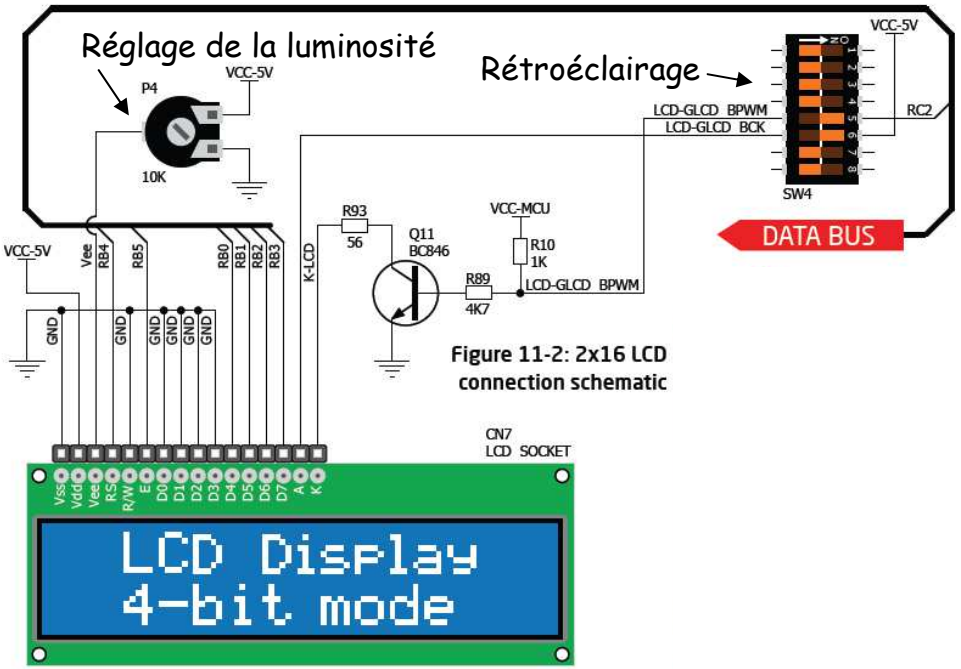
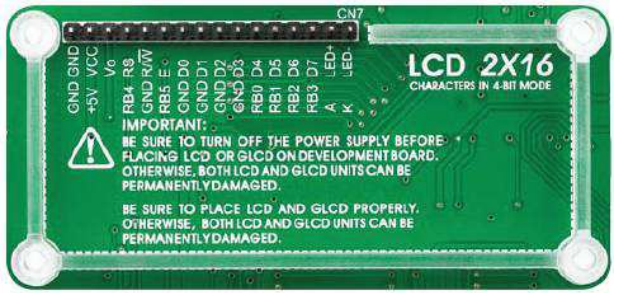


Figure 11-2: 2x16 LCD connection schematic



GND and VCC - Display power supply lines
Vo - LCD contrast level from potentiometer P4
RS - Register Select Signal line
E - Display Enable line
R/W - Determines whether display is in Read or Write mode. It's always connected to GND, leaving the display in Write mode all the time.
D0-D3 - Display is supported in 4-bit data mode, so lower half of the data byte interface is connected to GND.
D4-D7 - Upper half of the data byte
LED+ - Connection with the back-light LED anode
LED- - Connection with the back-light LED cathode

Les microcontrôleurs

L'afficheur LCD 2×16

Liste des caractères

(modèle GDM1602A)

Information complémentaire

datasheet du composant

Upper 4 Bits / Lower 4 Bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	CG RAM (1)			0	@	P	`	p				ー	タ	ミ	α	p
xxxx0001	(2)		!	1	A	Q	a	q			。	ア	チ	ム	ä	q
xxxx0010	(3)		"	2	B	R	b	r			「	イ	ツ	メ	ß	θ
xxxx0011	(4)		#	3	C	S	c	s			」	ウ	テ	モ	ε	∞
xxxx0100	(5)		$	4	D	T	d	t			、	エ	ト	ヤ	μ	Ω
xxxx0101	(6)		%	5	E	U	e	u			・	オ	ナ	ユ	σ	ü
xxxx0110	(7)		&	6	F	V	f	v			ヲ	カ	ニ	ヨ	ρ	Σ
xxxx0111	(8)		'	7	G	W	g	w			ァ	キ	ヌ	ラ	g	π
xxxx1000	(1)		(	8	H	X	h	x			ィ	ク	ネ	リ	√	x̄
xxxx1001	(2)		)	9	I	Y	i	y			ゥ	ケ	ノ	ル	-¹	y
xxxx1010	(3)		*	:	J	Z	j	z			ェ	コ	ハ	レ	j	千
xxxx1011	(4)		+	;	K	[	k	{			ォ	サ	ヒ	ロ	ˣ	万
xxxx1100	(5)		,	<	L	¥	l	|			ャ	シ	フ	ワ	¢	円
xxxx1101	(6)		−	=	M	]	m	}			ュ	ス	ヘ	ン	£	÷
xxxx1110	(7)		.	>	N	^	n	→			ョ	セ	ホ	゛	ñ	
xxxx1111	(8)		/	?	O	_	o	←			ッ	ソ	マ	°	ö	■

Les microcontrôleurs

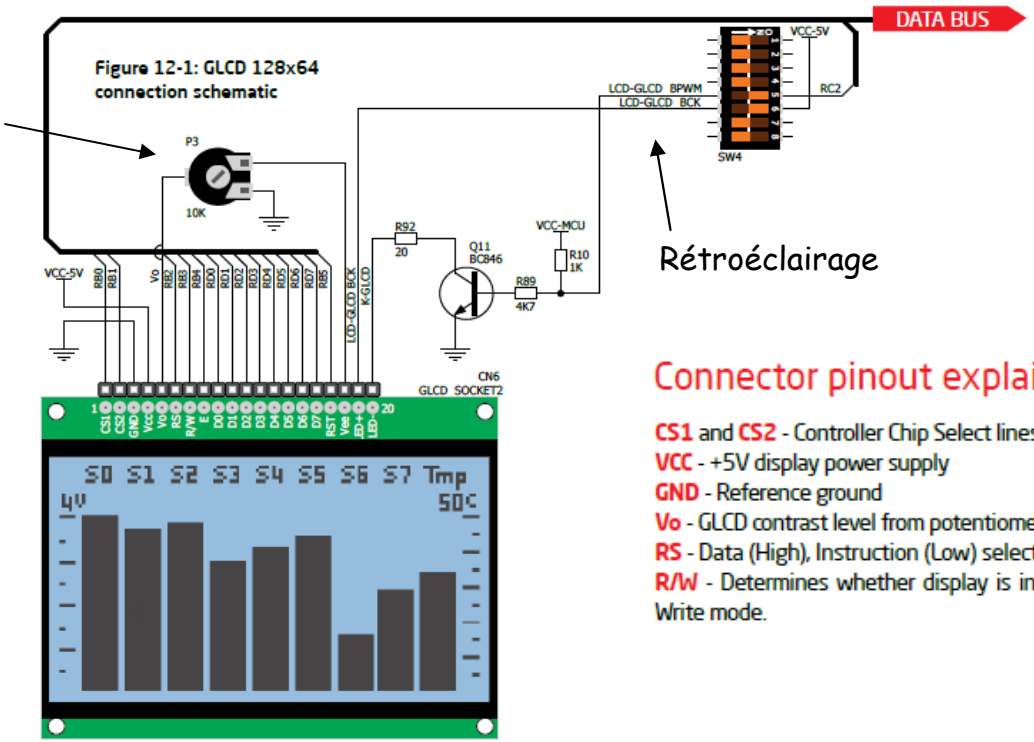
L'afficheur GLCD 128×64 (GLCD = Graphical Liquid Crystal Display)

Ecran monochromatique permettant d'afficher du texte, des images, une IHM etc

Les lignes de commande sont routées vers le port B, les lignes DATA vers le port C du µC

L'écran LCD 2×16 utilisant les mêmes ports, les 2 afficheurs ne peuvent être utilisés en même temps.

Réglage de la luminosité (P3)



Rétroéclairage

Connector pinout explained

CS1 and **CS2** - Controller Chip Select lines
VCC - +5V display power supply
GND - Reference ground
Vo - GLCD contrast level from potentiometer P3
RS - Data (High), Instruction (Low) selection line
R/W - Determines whether display is in Read or Write mode.
E - Display Enable line
D0-D7 - Data lines
RST - Display reset line
Vee - Reference voltage for GLCD contrast potentiometer P3
LED+ - Connection with the back-light LED anode
LED- - Connection with the back-light LED cathode

Les microcontrôleurs

Ecran tactile associé à l'afficheur GLCD 128×64

Cet écran tactile se place au dessus de l'afficheur GLCD. Il est constitué de 2 couches résistives superposées qui se mettent en contact lorsqu'une pression est exercée.

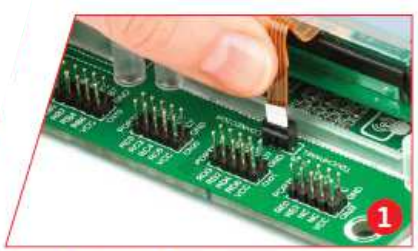


Figure 13-1: Put Touch panel flat cable in the connector

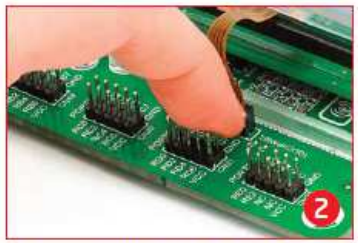


Figure 13-2: Use a tip of your finger to push it inside

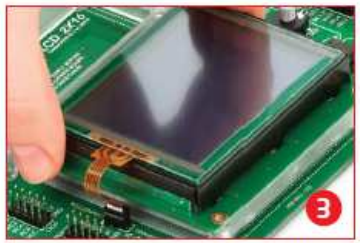


Figure 13-3: Now place GLCD with Touch panel into GLCD socket

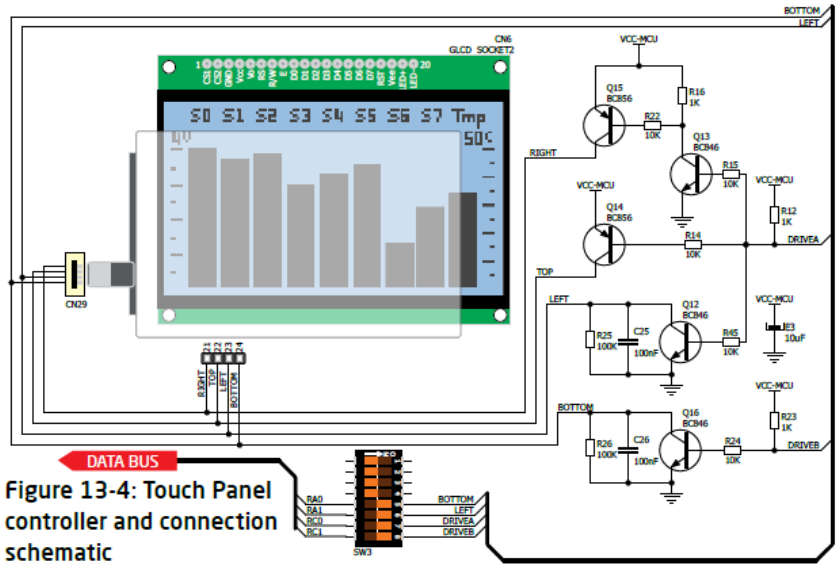


Figure 13-4: Touch Panel controller and connection schematic

Enabling Touch panel

Touch panel is enabled using **SW3.5**, **SW3.6**, **SW3.7** and **SW3.8** switches. They connect **READ-X** and **READ-Y** lines of the touch panel with **RA0** and **RA1** analog inputs, and **DRIVEA** and **DRIVEB** with **RC0** and **RC1** digital outputs on microcontroller sockets. Make sure to disconnect other peripherals, LEDs and additional pull-up or pull-down resistors from the interface lines in order not to interfere with signal/data integrity.

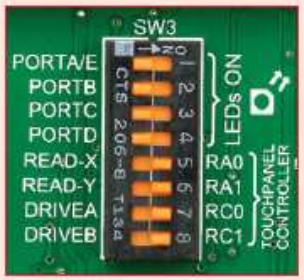


Figure 13-5: Turn on switches 5 through 8 on SW3 to enable Touch panel controller

SW3

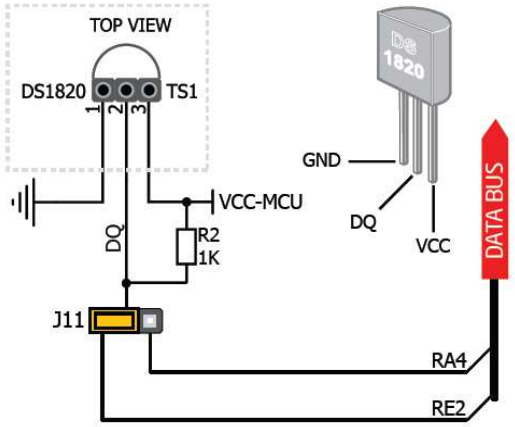
Les microcontrôleurs

Le capteur de T° DS1820

Capteur numérique de T° sur interface 1-fil.

Fonctionne en mode esclave

T° de -55°C à +128°C, Précision 0.5°C



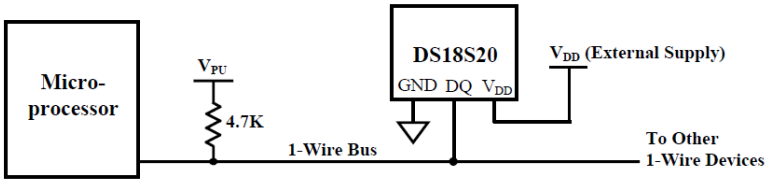
Connexion sur les broches RA4 ou RE2

(jumper J11)

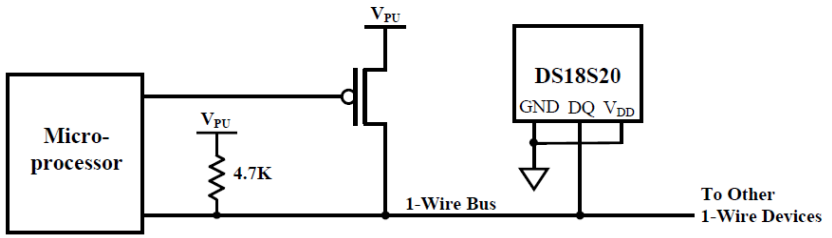
TO-92	SYMBOL	DESCRIPTION
1	GND	**Ground.**
2	DQ	**Data Input/Output pin.** Open-drain 1-wire interface pin. Also provides power to the device when used in parasite power mode (see "Parasite Power" section.)
3	V_{DD}	**Optional V_{DD} pin.** V_{DD} must be grounded for operation in parasite power mode.

2 méthodes pour alimenter ce circuit :

- soit par une alimentation externe



- soit par une alimentation parasite



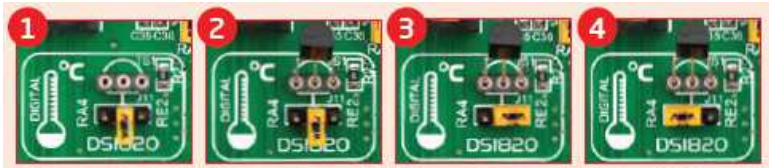


Figure 15-1: DS1820 not connected

Figure 15-2: DS1820 placed in socket

Figure 15-3: DS1820 connected to RE2 pin

Figure 15-4: DS1820 connected to RA4 pin

Les microcontrôleurs

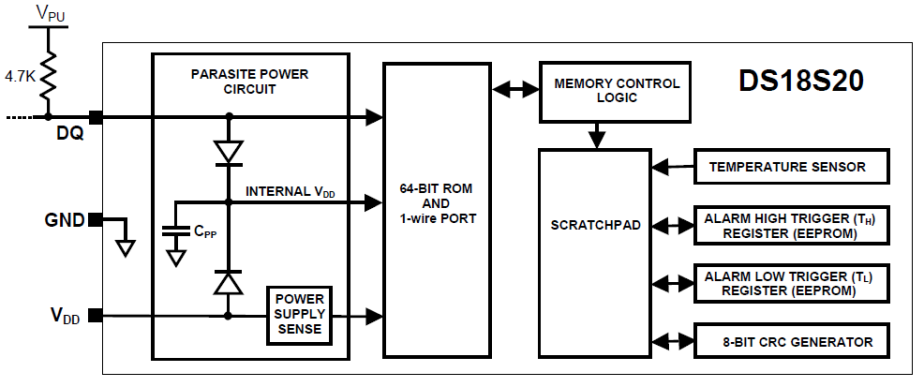
Le capteur de T° DS1820

Le capteur de T° DS1820 est un capteur numérique qui communique, via l'interface 1-fil (one-wire) avec le µprocesseur. Il intègre une mémoire ROM 64 bits et une mémoire RAM.

La température est codée sur 2 octets mais les 8 bits de poids forts indiquent tous le signe : 0 si T°≥ 0, 1 si T°< 0

Si la T° est négative, les 8 bits de poids faibles sont codés en complément à 2.

Pour connaître la T° en °C, il suffit de convertir le nombre binaire en décimal et de diviser par 2.



	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}
	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MS BYTE	S	S	S	S	S	S	S	S

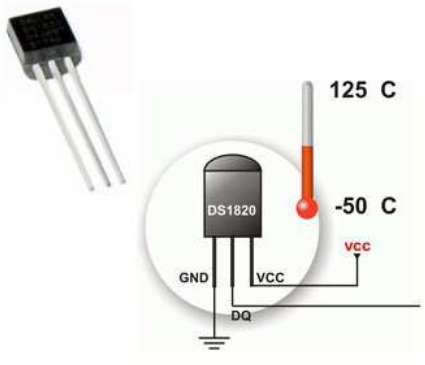
S = SIGN

TEMPERATURE (°C)	DIGITAL OUTPUT (BINARY)	DIGITAL OUTPUT (HEX)
+85.0*	0000 0000 1010 1010	00AAh
+25.0	0000 0000 0011 0010	0032h
+0.5	0000 0000 0000 0001	0001h
0	0000 0000 0000 0000	0000h
-0.5	1111 1111 1111 1111	FFFFh
-25.0	1111 1111 1100 1110	FFCEh
-55.0	1111 1111 1001 0010	FF92h

**The power-on reset value of the temperature register is +85°C.*

Les microcontrôleurs

Le capteur de T° DS1820



Mémoire ROM 64 bits

8-BIT CRC		48-BIT SERIAL NUMBER		8-BIT FAMILY CODE (10h)	
MSB	LSB	MSB	LSB	MSB	LSB

- Les 8 premiers bits correspondent au code famille du capteur (ici 10_h)
- Les 48 bits suivants correspondent au numéro de série du capteur (pour son identification)
- Les 8 bits restants permettent de détecter et de corriger les éventuelles erreurs sur l'interface 1-wire. CRC = contrôle de redondance cyclique, ou en anglais Cyclic Redundancy Check)

La mémoire RAM

La mémoire est constituée de ram statique SRAM pour la partie bloc-notes (Scratchpad) et de mémoire non volatile EEPROM pour les registres de seuils d'alarme TH et TL. A noter que si la fonctionnalité d'alarme n'est pas utilisée, ces registres peuvent servir de mémoire à usage général.

Registre de T° (2 octets)

	SCRATCHPAD (Power-up State)		EEPROM
byte 0	Temperature LSB (AAh	(85°C)	
byte 1	Temperature MSB (00h)		
byte 2	T_H Register or User Byte 1*	↔	T_H Register or User Byte 1
byte 3	T_L Register or User Byte 2*	↔	T_L Register or User Byte 2
byte 4	Reserved (FFh)		
byte 5	Reserved (FFh)		
byte 6	COUNT REMAIN (0Ch)		
byte 7	COUNT PER °C (10h)		
byte 8	CRC*		

*Power-up state depends on value(s) stored in EEPROM

Température seuil si utilisation d'une alarme

Les microcontrôleurs

Le capteur de T° DS1820 – Les commandes ROM

READ ROM (33_h) - Cette commande ne peut être utilisée que s'il n'y a qu'un seul esclave sur le bus. Celui ci répond alors ces 64 bits de code.

MATCH ROM (55_h) - Cette commande suivi de 64 bits de code, va permettre au maître de sélectionner un esclave particulier.

SKIP ROM (CC_h) - Commande d'appel général, pour adresser tous les esclaves. Cette fonction est utile pour adresser un esclave qui est seul sur le bus, sans avoir à envoyer les 64 bits de son code.

SEARCH ROM ($F0_h$) - Cette commande va permettre de rechercher bit à bit les codes de tous les esclaves raccordés au bus 1 Wire.

CONDITIONAL SEARCH (EC_h) - Cette commande fonctionne comme la commande SEARCH ROM, à la différence que seul les circuits ayant une condition bien spécifiée participent à la recherche. Par exemple les circuits de mesure de la température qui ont le Flag d'alarme actif ou les port E/S qui ont leur sortie à "1".

Les microcontrôleurs

Le capteur de T° DS1820 – Les codes commandes

Après avoir envoyé une commande ROM pour adresser un DS18S20 esclave, le maître doit envoyer un code de commande :

Début de conversion (44_h) - Cette commande lance la conversion de température. Le résultat est rangé dans les 2 octets LSB et MSB .

ECRITURE en RAM ($4E_h$) - Seuls les octets 2 et 3 de la zone RAM peuvent être écrits. Il s'agit des octets: Alarme seuil haut et Alarme seuil bas.

LECTURE de la RAM (BE_h) - Les 9 octets de la RAM sont envoyés vers le maître. L'esclave commence par le bit 0 du premier octet et transmet ainsi les 9 octets de sa RAM. Le maître peut interrompre à tout moment la lecture en faisant un Reset.

COPIE RAM en EEPROM (48_h) - Copie des octets 2 et 3 de la zone RAM dans la zone EEPROM pour sauvegarde en cas de coupure d'alimentation.

RECOPIE EEPROM en RAM ($B8_h$) - Cette commande récupère en EEPROM les octets Alarme seuil haut et Alarme seuil bas pour les placer en RAM dans les octets 2 et 3.

Les microcontrôleurs

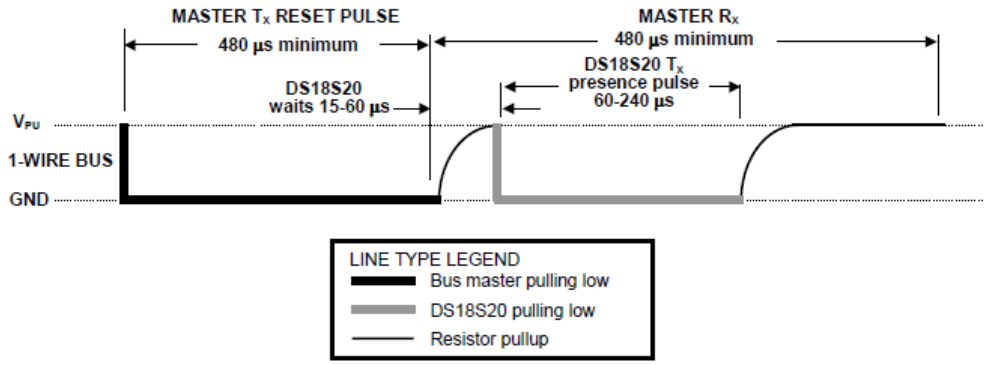
Le capteur de T° DS1820 – Les codes commandes

ALIMENTATION PARASITE ($B4_h$) - L'esclave répond à cette commande par un bit à "1" s'il fonctionne avec une alimentation extérieure, c'est-à-dire sur 3 fils, et par le bit à "0" s'il est en mode d'alimentation parasite, sur 2 fils.

Procédure d'initialisation

Toute communication avec un capteur de T° DS1820 commence par une séquence d'initialisation qui se déroule en 2 étapes :

- Le master envoie un pulse de **reset**, la ligne qui est à l'état haut au repos (présence d'une résistance de pullup) passe à l'état bas pour une durée minimale de 480 µs
- Lorsque la ligne revient à l'état haut, le capteur attend entre 15 et 60 µs avant de transmettre à son tour un pulse de **présence** d'une durée comprise entre 60 et 240 µs



Les microcontrôleurs

DS18S20 OPERATION EXAMPLE 3

In this example there is only one DS18S20 on the bus and it is using parasite power. The bus master initiates a temperature conversion then reads the DS18S20 scratchpad and calculates a higher resolution result using the data from the temperature, COUNT REMAIN and COUNT PER °C registers.

MASTER MODE	DATA (LSB FIRST)	COMMENTS
TX	Reset	Master issues reset pulse.
TR	Presence	DS18S20 responds with presence pulse.
TX	CCh	Master issues Skip ROM command.
TX	44h	Master issues Convert T command.
TX	DQ line held high by strong pullup	Master applies strong pullup to DQ for the duration of the conversion (t_{conv}).
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20 responds with presence pulse.
TX	CCh	Master issues Skip ROM command.
TX	BEh	Master issues Read Scratchpad command.
RX	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated. The master also calculates the TEMP_READ value and stores the contents of the COUNT REMAIN and COUNT PER °C registers.
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20 responds with presence pulse.
-	-	CPU calculates extended resolution temperature using the equation in the OPERATION - MEASURING TEMPERATURE section of this datasheet.

Les microcontrôleurs

Le capteur de T° DS1820 – Les codes commandes

ALIMENTATION PARASITE ($B4_h$) - L'esclave répond à cette commande par un bit à "1" s'il fonctionne avec une alimentation extérieure, c'est-à-dire sur 3 fils, et par le bit à "0" s'il est en mode d'alimentation parasite, sur 2 fils.

La librairie OneWire de l'IDE MikroC Pro for Pic

Library Routines

- Ow_Reset
- Ow_Read
- Ow_Write

Ow_Reset

Prototype	`unsigned short Ow_Reset(unsigned short *port, unsigned short pin);`
Returns	■ `0` if the device is present ■ `1` if the device is not present
Description	Issues OneWire reset signal for DS18x20. Parameters : ■ `port`: OneWire bus port ■ `pin`: OneWire bus pin
Requires	Devices compliant with the Dallas OneWire protocol.
Example	To reset the DS1820 that is connected to the RE2 pin: `Ow_Reset(&PORTE, 2);`

Les microcontrôleurs

Le capteur de T° DS1820 – Les codes commandes

La librairie OneWire de l'IDE MikroC Pro for Pic

Library Routines

- Ow_Reset
- Ow_Read
- Ow_Write

Ow_Read

Prototype	`unsigned short Ow_Read(unsigned short *port, unsigned short pin);`
Returns	Data read from an external device over the OneWire bus.
Description	Reads one byte of data via the OneWire bus. Parameters : • `port`: OneWire bus port • `pin`: OneWire bus pin
Requires	Devices compliant with the Dallas OneWire protocol.

Example	`// Read a byte from the One-Wire Bus`
	`unsigned short tmp;`
	`...`
	`tmp = Ow_Read(&PORTE, 2);`

Les microcontrôleurs

Le capteur de T° DS1820 – Les codes commandes

La librairie OneWire de l'IDE MikroC Pro for Pic

Library Routines

- Ow_Reset
- Ow_Read
- Ow_Write

Ow_Write

Prototype	`void Ow_Write(unsigned short *port, unsigned short pin, unsigned short par);`
Returns	Nothing.
Description	Writes one byte of data via the OneWire bus. Parameters : ■ `port`: OneWire bus port ■ `pin`: OneWire bus pin ■ `par`: data to be written
Requires	Devices compliant with the Dallas OneWire protocol.
Example	`// Send a byte to the One-Wire Bus` `Ow_Write(&PORTE, 2, 0xCC);`

Commande Skip ROM

Les microcontrôleurs

Le buzzer Piezo embarqué

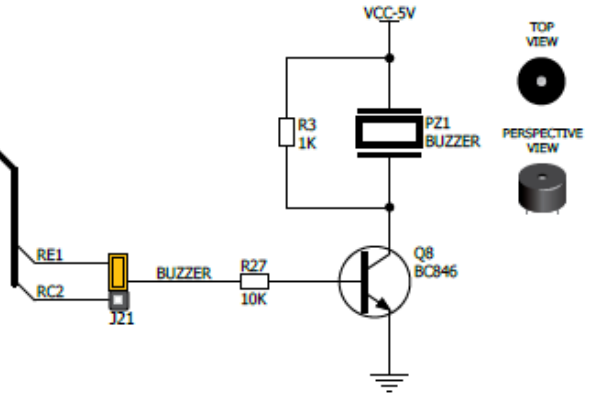
Fréquence de résonnance : 3,8 kHz

Utilisable entre 20 Hz et 20 kHz, plage idéale entre 2 et 4 kHz

Niveau acoustique : 85 dB

Figure 19-2: Use jumper J12 to connect Piezo buzzer on RE1 or RC2 pin

Figure 19-1: Piezo buzzer connected to RE1 microcontroller pin



Connexion sur les broches RE1 ou RC2 (jumper J21)

- Commande du son (Fréquence et volume) grâce à la modulation PWM
- Le volume est donné par le rapport cyclique de la modulation PWM

Les microcontrôleurs

Le CAN embarqué

Possibilité de tester 2 lignes du convertisseur analogique numérique directement sur la carte. Les potentiomètres (10 kΩ) P1 et P2 sont des ponts diviseurs de tension.

- Cavalier J15 : permet de connecter une des broches de RA0 à RA5
- Cavalier J16 : permet de connecter une des broches de RB0 à RB5)

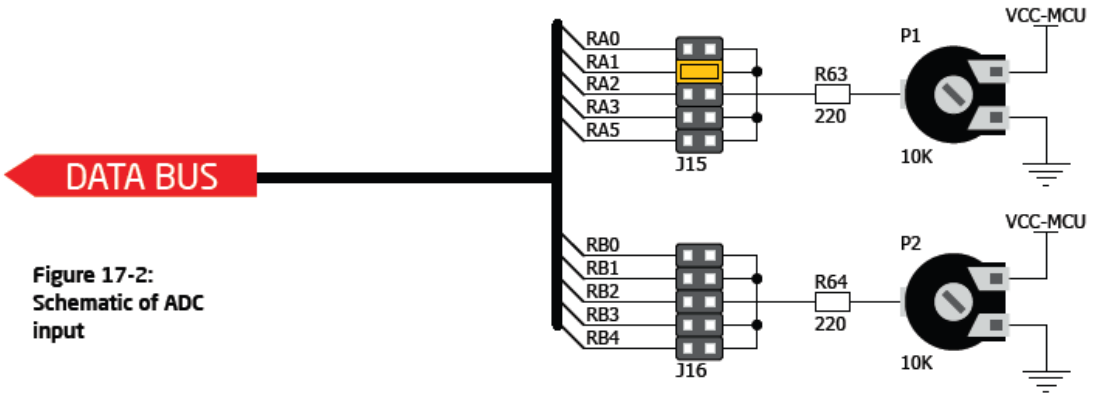
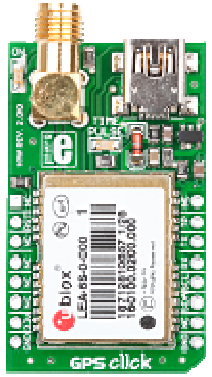


Figure 17-2: Schematic of ADC input

Les microcontrôleurs

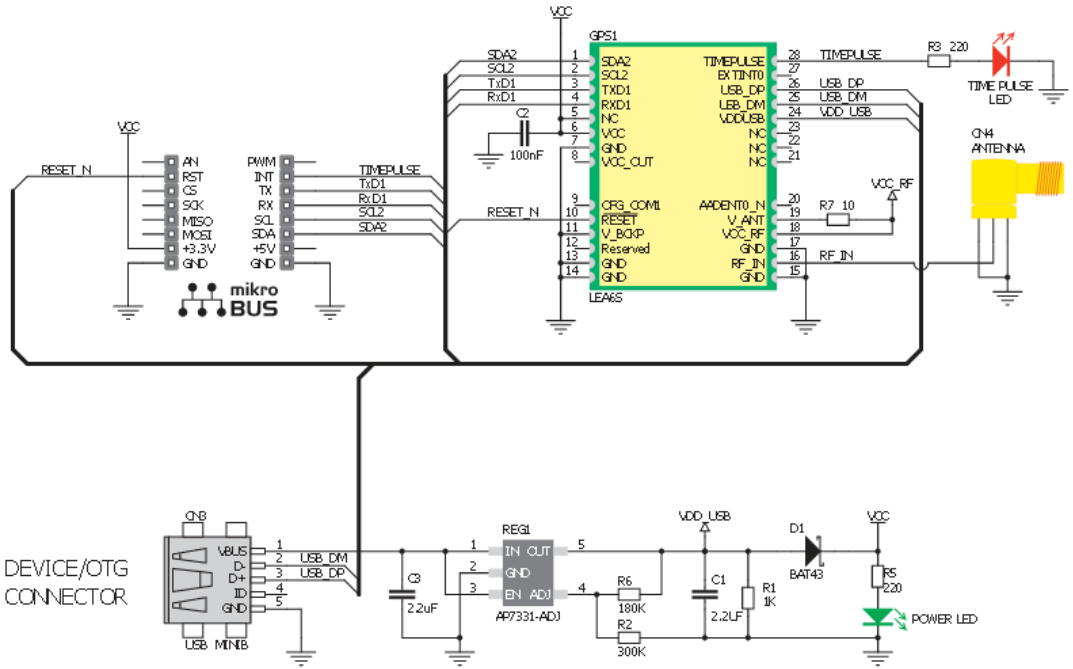


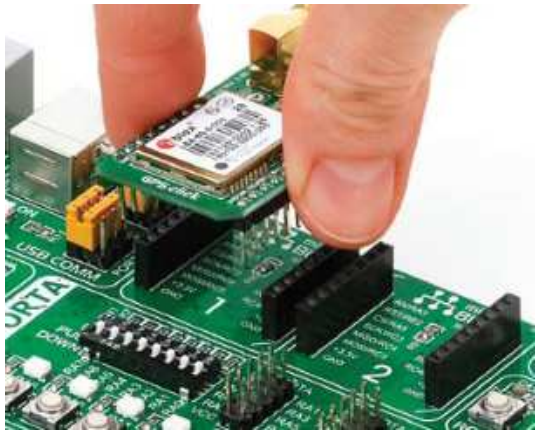
Le module GPS Click

Composant u-blox LEA-6S

Communication avec le MCU via l'interface UART ou I²c

Les data peuvent aussi être récupérées sur un PC via l'interface USB



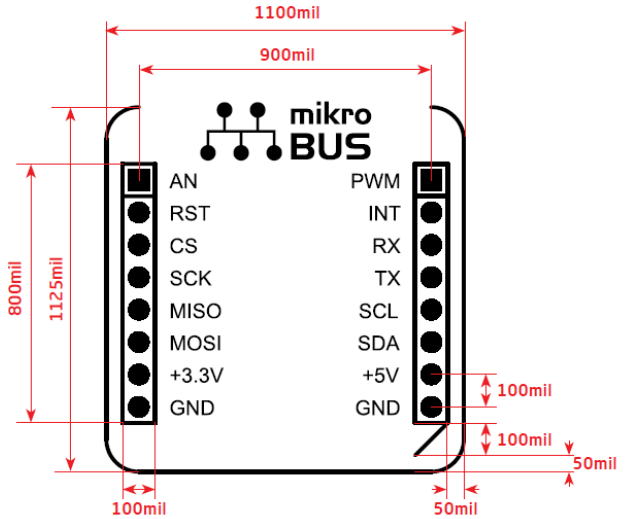


Les microcontrôleurs

Bus mikroBUS

Bus propriétaire MikroElectronika :

- Permet de connecter facilement différents modules sur la carte
- 2 connecteurs 1×8 femelles
- 3 types de ports communicants : SPI, UART et I²C
- Plusieurs broches pour les signaux PWM, entrée analogique, interruption, reset et chip select
- Alimentation 3.3 ou 5 V



AN	- Analog pin	**PWM**	- PWM output line
RST	- Reset pin	**INT**	- Hardware Interrupt line
CS	- SPI Chip Select line	**RX**	- UART Receive line
SCK	- SPI Clock line	**TX**	- UART Transmit line
MISO	- SPI Slave Output line	**SCL**	- I²C Clock line
MOSI	- SPI Slave Input line	**SDA**	- I²C Data line
+3.3V	- VCC-3.3V power line	**+5V**	- VCC-5V power line
GND	- Reference Ground	**GND**	- Reference Ground

Les microcontrôleurs

Langage C & microcontrôleur

De plus en plus, les programmes pour micro-contrôleur sont écrits en langage C. Ce langage permet de développer rapidement des programmes, des bibliothèques qui pourront être ensuite utilisées sur différentes machines.

Pourquoi un langage tel que le C ?

- Universel : il n'est pas dédié à une application !
- Moderne : structuré, déclaratif, récursif, avec structures de contrôle et de déclaration.
- Proche de la machine : manipulations de bits, pointeurs, possibilité d'incorporer de l'assembleur, etc.
- Portable : le même code peut être utilisé sur plusieurs machines (il faut toutefois faire attention à ne pas créer des fonctions spécifiques à une machine).
- Extensible : il est possible de créer des bibliothèques ou d'en incorporer.

Les microcontrôleurs

Langage C & microcontrôleur

Vous avez accès à la majorité des fonctionnalités du C de la norme ANSI C. Vous pouvez donc écrire quelque chose comme vitesse = 0x27 pour faire l'affectation d'une variable codée sur 8 bits, ou encore if(vitesse == limite) pour effectuer un test d'égalité entre deux variables entières codées sur un nombre de bits appropriés.

Plus précisément, vous avez accès :

- aux déclaration de variables, constantes, tableaux, pointeurs, structures, etc.
- aux opérateurs arithmétiques et logiques,
- aux opérateurs d'affectation, incrémentation et décrémentation,
- aux opérateurs de contrôle de flux (test et boucles),
- à la conversion de type,
- à l'intégration de routines en assembleur.

Par contre, les instructions gérant les entrées et sorties (écran, clavier, disque, etc.) n'ont pas toujours de sens dans ce type de contexte.

Les microcontrôleurs

Un premier programme

```
 * NOTES:
     - Turn ON the PORT LEDs at SW3.
 */

void main() {

  TRISA = 0;            // set direction to be output
  TRISB = 0;            // set direction to be output
  TRISC = 0;            // set direction to be output
  TRISD = 0;            // set direction to be output
  TRISE = 0;            // set direction to be output

  do {
    LATA = 0x00;        // Turn OFF LEDs on PORTA
    LATB = 0x00;        // Turn OFF LEDs on PORTB
    LATC = 0x00;        // Turn OFF LEDs on PORTC
    LATD = 0x00;        // Turn OFF LEDs on PORTD
    LATE = 0x00;        // Turn OFF LEDs on PORTE
    Delay_ms(1000);     // 1 second delay

    LATA = 0xFF;        // Turn ON LEDs on PORTA
    LATB = 0xFF;        // Turn ON LEDs on PORTB
    LATC = 0xFF;        // Turn ON LEDs on PORTC
    LATD = 0xFF;        // Turn ON LEDs on PORTD
    LATE = 0xFF;        // Turn ON LEDs on PORTE
    Delay_ms(1000);     // 1 second delay
  } while(1);           // Endless loop
}
```

Affectation des registres TRISx du MCU (sens des I/O)

Toutes les LEDs sont éteintes

Délais d'une seconde

Toutes les LEDs sont allumées

Les microcontrôleurs

Les opérateurs du langage C

Opérateurs *arithmétiques* et *relationnels* :

Opérateur	Traduction	Exemple	Résultat
+	Addition	`x + y`	l'addition de x et y
-	Soustraction	`x - y`	la soustraction de x et y
*	Produit	`x * y`	la multiplication de x et y
/	Division	`x / y`	le quotient de x et y
%	Reste	`x % y`	Reste de la division euclidienne de x par y
+(unaire)	Signe positif	`+x`	la valeur de x
-(unaire)	Signe négatif	`-x`	la négation arithmétique de x
++(unaire)	Incrément	`x++` ou `++x`	x est incrementé ($x = x + 1$). L'opérateur préfixe `++x` (resp. suffixe `x++`) incrémente x avant (resp. après) de l'évaluer
--(unaire)	Decrément	`x--` ou `--x`	x est décrementé ($x = x - 1$). L'opérateur préfixe `--x` (resp. suffixe `x--`) décrémente x avant (resp. après) de l'évaluer

Opérateur	Traduction	Exemple	Résultat
<	inférieur	`x < y`	1 si x est inférieur à y
<=	inférieur ou égal	`x <= y`	1 si x est inférieur ou égal à y
>	supérieur	`x > y`	1 si x est supérieur à y
>=	supérieur ou égal	`x >= y`	1 si x est supérieur ou égal à y
==	égalité	`x == y`	1 si x est égal à y
!=	non inégalité	`x != y`	1 si x est différent de y

Opérateurs *logiques* et *de manipulation de bits* :

Opérateur	Traduction	Exemple	Résultat (pour chaque position de bit)
&	ET bit à bit	`x & y`	1 si les bits de x et y valent 1
|	OU bit à bit	`x | y`	1 si le bit de x et/ou de y vaut 1
^	XOR bit à bit	`x ^ y`	1 si le bit de x ou de y vaut 1
~	NON bit à bit	`~x`	1 si le bit de x est 0
<<	décalage à gauche	`x << y`	décale chaque bit de x de y positions vers la gauche
>>	sécalage à droite	`x >> y`	décale chaque bit de x de y positions vers la droite

Opérateur	Traduction	Exemple	Résultat
&&	ET logique	`x && y`	1 si x et y sont différents de 0
||	OU logique	`x || y`	1 si x et/ou y sont différents de 0
!	NON logique	`!x`	1 si x est égal à 0. Dans tous les autres cas, 0 est renvoyé.

Les microcontrôleurs

Opérateurs *d'accès à la mémoire, etc.*

Opérateur	Traduction	Exemple	Résultat
`&`	Adresse de	`&x`	l'adresse mémoire de x
`*`	Indirection	`*p`	l'objet (ou la fonction) pointée par p
`[ ]`	Elément de tableau	`t[i]`	L'équivalent de `*(x+i)`, l'élément d'indice i dans le tableau t
`.`	Membre d'une structure ou d'une union	`s.x`	le membre x dans la structure ou l'union s
`->`	Membre d'une structure ou d'une union	`p->x`	le membre x dans la structure ou l'union pointée par p

Opérateur	Traduction	Exemple	Résultat
`()`	Appel de fonction	`f(x,y)`	Exécute la fonction f avec les arguments x et y
`(type)`	cast	`(long)x`	la valeur de x avec le type spécifié
`sizeof`	taille en bits	`sizeof(x)`	nombre de bits occupé par x
`? :`	Evaluation conditionnelle	`x?:y:z`	si x est différent de 0, alors y sinon z
`,`	séquencement	`x,y`	Evalue x puis y

Les microcontrôleurs

Les formats bit et sbit

Le compilateur mikroC PRO for PIC permet d'accéder individuellement aux bits des différents registres 8 bits. Il supporte également les types *sbit* et *bit*.

Prenons le bit d'interruption générale (GIE) comme exemple. Ce bit est définit dans le fichier de définition de notre microcontrôleur (C:\Program Files\Mikroelektronika\mikroC PRO for PIC\defs\PIC18F45K22.c) :

```
const register unsigned short int GIE = 7;
sbit  GIE_bit at INTCON.B7;
```

Ainsi, pour accéder à ce bit individuel, vous pouvez passer par son alias :

```
// Set Global Interrupt Bit (GIE)
GIE_bit = 1;
```

Une autre manière d'accéder directement aux bits individuels d'un registre est d'utiliser les identificateurs B_0,…B_7 ou F_0,…, F_7 (avec F_7 = MSB) :

```
// predefined globals as bit designators
// Clear bit 0 in INTCON register
INTCON.B0 = 0;

// Set bit 0 in INTCON register
INTCON.F0 = 1;
```

Si on revient à notre bit GIE, on pourrait donc y accéder de cette manière :

```
// Clear Global Interrupt Bit (GIE)
INTCON.GIE = 0;
```

Les microcontrôleurs

Les formats sbit

Le compilateur mikroC PRO for PIC possède un type de donnée appelé sbit qui donne accès aux registres, SFR, variables etc. Vous pouvez par exemple déclarer une variable sbit de telle manière qu'elle pointe vers l'un des bits d'un registre :

```
extern sfr sbit Abit; // Abit is precisely defined in some external file, for example in the main program unit
```

Notez l'utilisation du mot clé « sfr » lors de la déclaration de Abit, cela vient du fait que PORTB est un registre de type SFR. Dans le programme principal, vous devez spécifier vers quel registre la variable Abit pointe, par exemple :

```
sbit Abit at PORTB.B0; // this is where Abit is fully defined
...
void main() {
...
}
```

Remarque : la déclaration d'une variable sbit n'est pas possible avec les identifieurs F_0,...F_7.

```
extern sbit AnotherBit; // AnotherBit is precisely defined in some external file, for example in the main program unit
```

Si vous souhaitez déclarer un bit qui n'est pas lié à un registre SFR, le mot clé « sfr » n'est plus indispensable :

```
char MyVar;
sbit AnotherBit at MyVar.B0; // this is where AnotherBit is fully defined
...
void main() {
...
}
```

Les microcontrôleurs

Mot clé « at »

Vous pouvez utiliser le mot clé « at » pour créer un alias sur une variable. Par exemple, vous pouvez créer une librairie sans utiliser les noms des registres et plus tard, dans le programme principal, définir ces registres :

```
extern char PORTAlias; // here in the library we can use its symbolic name
```

```
char PORTAlias at PORTB; // this is where PORTAlias is fully defined
...
void main() {
...
}
```

Le type bit

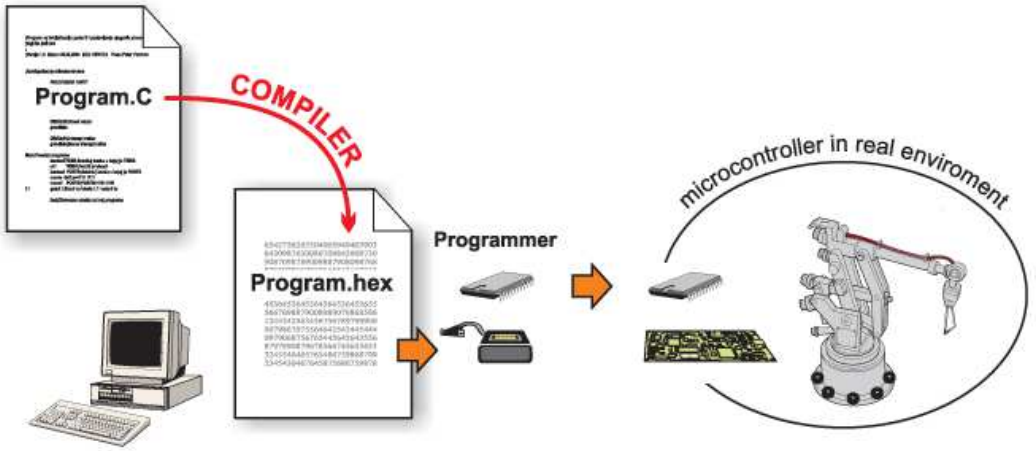
Il est possible d'utiliser le format de donnée *bit* pour déclarer une variable. Il ne peut cependant pas etre utilisé comme argument dans une liste ou comme valeur de retour d'une fonction. Il ne peut pas non plus être utilisé pour la définition d'un pointeur ni pour celle d'un tableau.

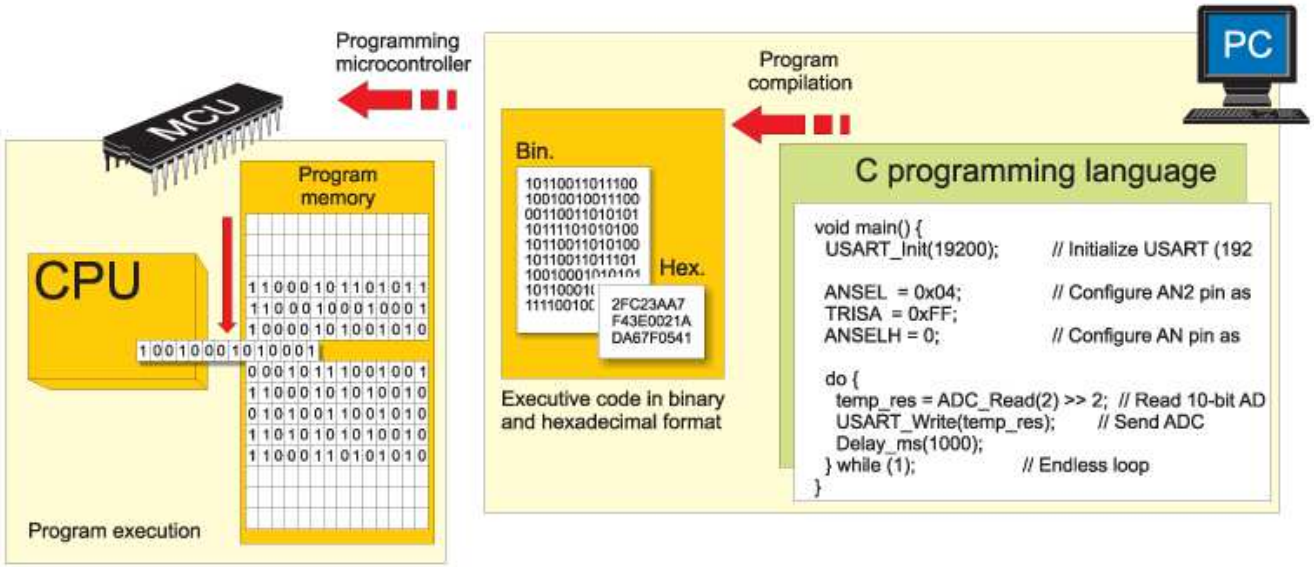
```
bit bf;     // bit variable
```

```
bit *ptr;     // invalid
```

```
bit arr[5];     // invalid
```

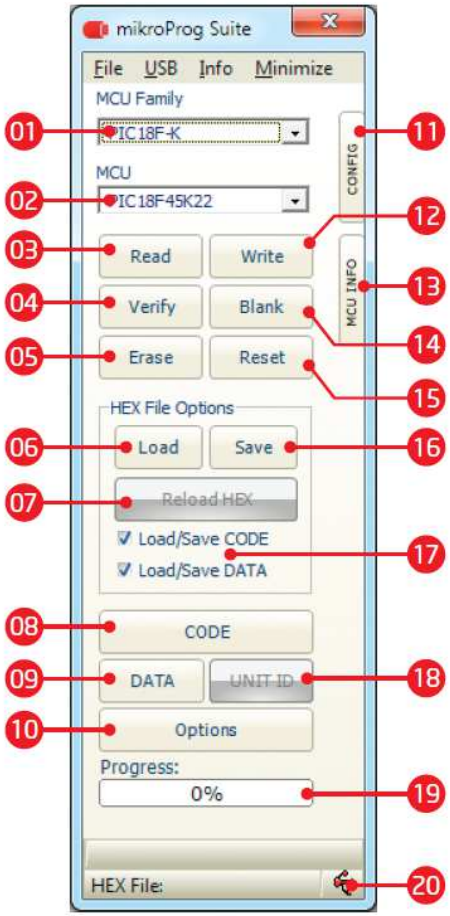
Les microcontrôleurs





Les microcontrôleurs

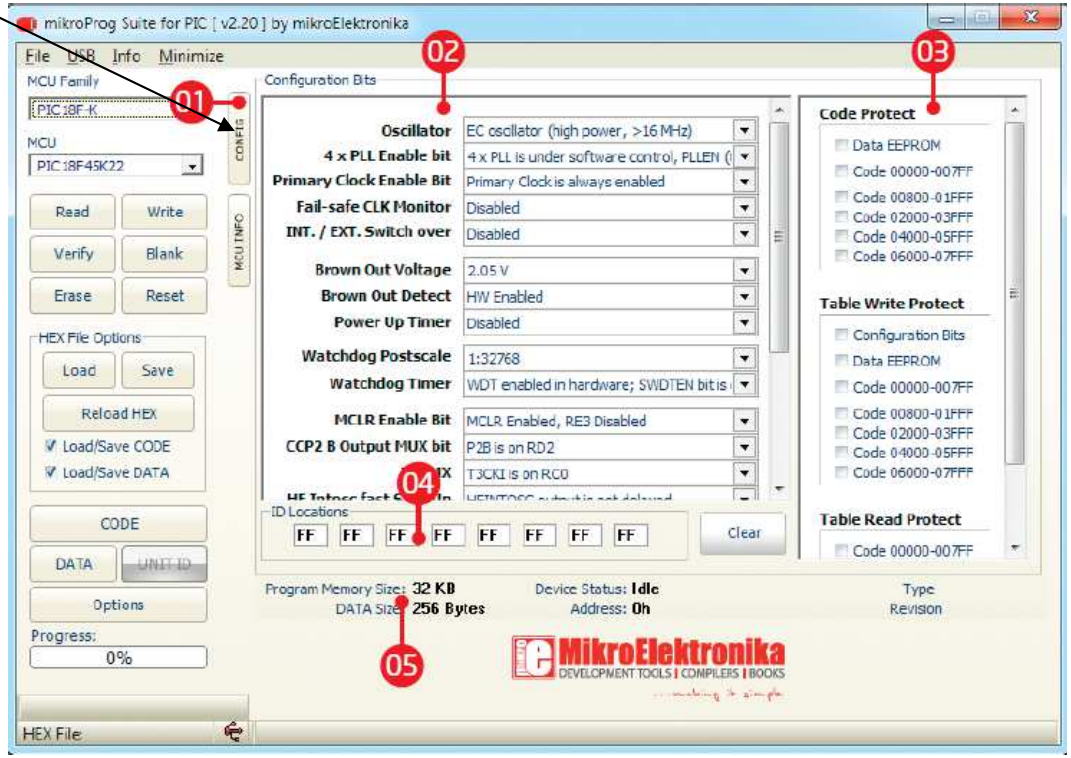
Le programmateur MikroProgSuite et la fenêtre principale



01 MCU family selection list
02 MCU type selection list
03 Read program from MCU
04 Verify the loaded program
05 Erase MCU memory contents
06 Browse for a .hex file on your PC
07 Reload previously loaded .hex file
08 Preview program which is in buffer and ready for uploading in MCU FLASH memory
09 Preview program which is in buffer and ready for uploading in MCU EEPROM memory
10 Various settings of visual, advanced and programming options.
11 Expand configuration bits menu
12 Upload .hex file in to MCU memory
13 Expand MCU info menu
14 Check whether the MCU is empty
15 Reset the microcontroller
16 Save buffer to a .HEX file
17 Load/Save CODE/DATA in buffer
18 Used for some MCU-s ID
19 Progress bar
20 Shows that programer is connected to USB port on a PC (red if connected)

Les microcontrôleurs

Le programmateur MikroProgSuite et la page CONFIG



01 CONFIG button opens config window
02 Configuration Bits section is used to set specific options for chosen MCU.
03 Protect parts of MCU memory from unauthorized reading and writing.
04 ID Location in MCU memory.
05 Basic information about selected MCU.

Les microcontrôleurs

Le programmateur MikroProgSuite

et la page MCU INFO