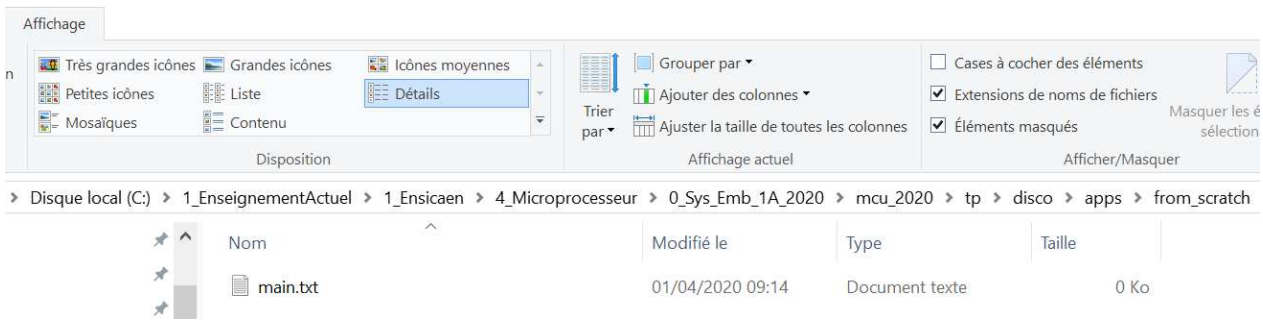


## From Scratch

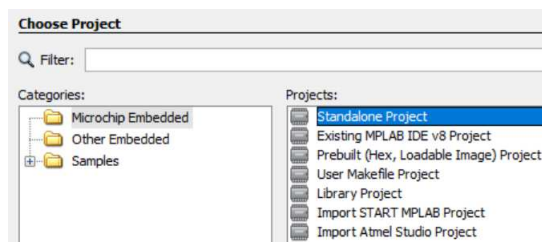
- Créer un nouveau fichier texte dans le dossier **disco/apps/fromscratch** nommé **main.txt**



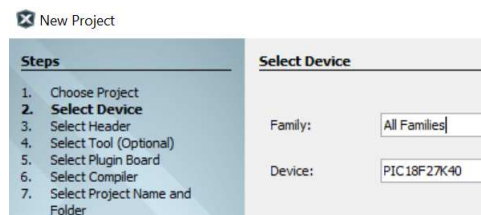
- Modifier l'extension en **.c**



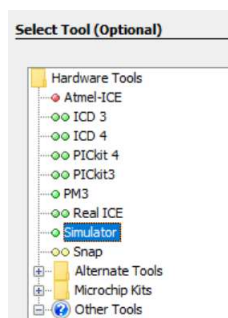
- Ouvrir MPLABx et créer un nouveau projet à partir de zéro (Standalone Project)



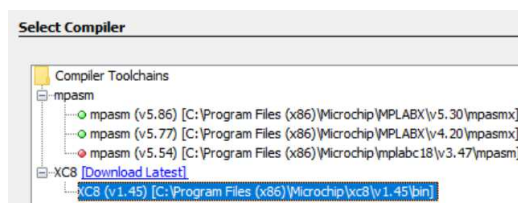
- Choisir le bon composant cible



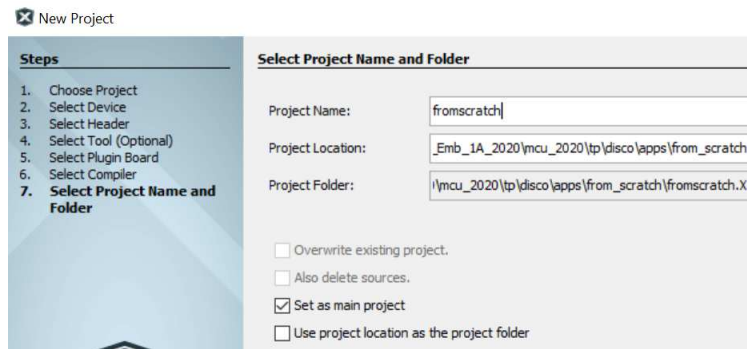
- Sélectionner le simulateur comme outil de **debug** (pour travailler sans maquette)



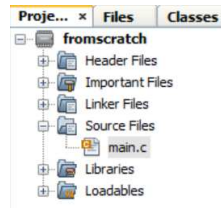
- Choisir la chaîne de compilation XC8 (compilateur pour MCU 8 bits)



➤ Nommer le projet **fromscratch** et le placer dans le dossier **disco/apps/fromscratch**




➤ Ajouter le fichier **main.c** au projet

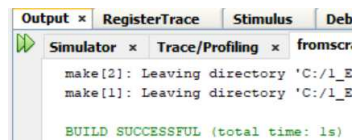


➤ Compléter le fichier

```
#include <xc.h>

main()
{
    while(1)
    {
    }
}
```

➤ Compiler  et vérifier que c'est un succès dans la fenêtre **output**



➤ Menu déroulant **window > target memory views > configuration bits** pour choisir l'horloge et 2 ou 3 autres configurations

Address	Name	Value	Field	Option	Category	Setting
300000	CONFIG1L	8C	FEXTOSC	OFF	External Oscillator mode Selection bits	Oscillator not enabled
			RSTOSC	HFINTOSC_64MHZ	Power-up default value for COSC bits	HFINTOSC with HFFRQ = 64 MHz and CDIV = 1:1
300001	CONFIG1H	FF	CLKOUTEN	OFF	Clock Out Enable bit	CLKOUT function is disabled
			CSWEN	ON	Clock Switch Enable bit	Writing to NOSC and NDIV is allowed
			FCMEN	ON	Fail-Safe Clock Monitor Enable bit	Fail-Safe Clock Monitor enabled
300002	CONFIG2L	FF	MCLR	EXTMCLR	Master Clear Enable bit	If LVP = 0, MCLR pin is MCLR; If LVP = 1, RE3
			PWRTE	OFF	Power-up Timer Enable bit	Power up timer disabled
			LPBOR	OFF	Low-power BOR enable bit	ULPBOR disabled
			BOREN	SBORDIS	Brown-out Reset Enable bits	Brown-out Reset enabled, SBOR bit is ignor
300003	CONFIG2H	FF	BORV	VBOR_2P45	Brown Out Reset Voltage selection bits	Brown-out Reset Voltage (VBOR) set to 2.45V
			ZCD	OFF	ZCD Disable bit	ZCD disabled. ZCD can be enabled by setting t
			PPS1WAY	ON	PPSLOCK bit One-Way Set Enable bit	PPSLOCK bit can be cleared and set only once;
			STVREN	ON	Stack Full/Underflow Reset Enable bit	Stack full/underflow will cause Reset
			DEBUG	OFF	Debugger Enable bit	Background debugger disabled
			XINST	OFF	Extended Instruction Set Enable bit	Extended Instruction Set and Indexed Addressi
300004	CONFIG3L	9F	WDTCPS	WDTCPS_31	WDT Period Select bits	Divider ratio 1:65536; software control of WD
			WDTE	OFF	WDT operating mode	WDT Disabled
300005	CONFIG3H	FF	WDICWS	WDICWS_7	WDT Window Select bits	window always open (100%); software control;

➤ Appuyer sur **Generate Source Code to Output**

```

// FIC18F27K40 Configuration Bit Settings

// 'C' source line config statements

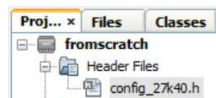
// CONFIG1L
#pragma config FEXTOSC = OFF // External Oscillator mode Selection bits (Oscillator not enabled)
#pragma config RSTOSC = HFINTOSC_64MHZ // Power-up default value for COSC bits (HFINTOSC with HFFRQ = 64 MHz and CDIV = 1:1)

// CONFIG1H
#pragma config CLKOUTEN = OFF // Clock Out Enable bit (CLKOUT function is disabled)
#pragma config CSWEN = ON // Clock Switch Enable bit (Writing to NOSC and NDIV is allowed)
#pragma config FCMEN = ON // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor enabled)

// CONFIG2L
#pragma config MCLRE = EXTMCLR // Master Clear Enable bit (If LVP = 0, MCLR pin is MCLR; If LVP = 1, RE3 pin function is MCLR )
#pragma config PWRTE = OFF // Power-up Timer Enable bit (Power up timer disabled)
#pragma config LPBOREN = OFF // Low-power BOR enable bit (ULPBOR disabled)
#pragma config BOREN = SBORDIS // Brown-out Reset Enable bits (Brown-out Reset enabled , SBOREN bit is ignored)

```

- Copier tout le code généré (ctrl A, ctrl C)
- Créer un nouveau fichier de type **xc8\_header.h** et le nommer **config\_27k40.h** (ce fichier doit être dans le même dossier que **main.c**)



- Coller dedans ce qui a été copié juste avant

*Début*

*fin*

```

#ifndef XC_HEADER_config_H
#define XC_HEADER_config_H

// CONFIG1L
#pragma config FEXTOSC = OFF // External Oscillator mode Selection bits (Oscillator not enabled)
#pragma config RSTOSC = HFINTOSC_64MHZ // Power-up default value for COSC bits (HFINTOSC with HFFRQ = 64 MHz and CDIV = 1:1)

// CONFIG1H
#pragma config CLKOUTEN = OFF // Clock Out Enable bit (CLKOUT function is disabled)
#pragma config CSWEN = ON // Clock Switch Enable bit (Writing to NOSC and NDIV is allowed)
#pragma config FCMEN = ON // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor enabled)

// CONFIG2L
#pragma config MCLRE = EXTMCLR // Master Clear Enable bit (If LVP = 0, MCLR pin is MCLR; If LVP = 1, RE3 pin function is MCLR )
#pragma config PWRTE = OFF // Power-up Timer Enable bit (Power up timer disabled)
#pragma config LPBOREN = OFF // Low-power BOR enable bit (ULPBOR disabled)
#pragma config BOREN = SBORDIS // Brown-out Reset Enable bits (Brown-out Reset enabled , SBOREN bit is ignored)

// CONFIG2H
#pragma config BORV = VBOR_2P45 // Brown-out Reset Voltage (BORV = 0:VBOR = 2.45V; BORV = 1:VBOR = 2.7V; BORV = 2:VBOR = 2.85V; BORV = 3:VBOR = 3.0V)
#pragma config ZCD = OFF // Zero Current Detect (ZCD = 0:ZCD is disabled; ZCD = 1:ZCD is enabled)
#pragma config PPS1WAY = ON // Primary Programming Slew Rate Limiting (PPS1WAY = 0:PPS is enabled for all I/O; PPS1WAY = 1:PPS is disabled for all I/O)
#pragma config STVREN = ON // Stack Overflow Reset Enable bit (STVREN = 0:Stack Overflow Reset disabled; STVREN = 1:Stack Overflow Reset enabled)
#pragma config DEBUG = OFF // Background Debugger Enable bit (DEBUG = 0:Background Debugger disabled; DEBUG = 1:Background Debugger enabled)
#pragma config XINST = OFF // Extended Instruction Set Enable bit (XINST = 0:Extended Instruction Set disabled; XINST = 1:Extended Instruction Set enabled)

// CONFIG5L
#pragma config CP = OFF // Code Protection bit (CP = 0:Code protection disabled; CP = 1:Code protection enabled)
#pragma config CPD = OFF // Code Protection Default Selection bit (CPD = 0:No code protection on any page; CPD = 1:All pages but the boot page are protected)

// CONFIG5H
#pragma config EBTR0 = OFF // External Block TRAP Enable bit (EBTR0 = 0:Block TRAP disabled; EBTR0 = 1:Block TRAP enabled)
#pragma config EBTR1 = OFF // External Block TRAP Enable bit (EBTR1 = 0:Block TRAP disabled; EBTR1 = 1:Block TRAP enabled)
#pragma config EBTR2 = OFF // External Block TRAP Enable bit (EBTR2 = 0:Block TRAP disabled; EBTR2 = 1:Block TRAP enabled)
#pragma config EBTR3 = OFF // External Block TRAP Enable bit (EBTR3 = 0:Block TRAP disabled; EBTR3 = 1:Block TRAP enabled)
#pragma config EBTR4 = OFF // External Block TRAP Enable bit (EBTR4 = 0:Block TRAP disabled; EBTR4 = 1:Block TRAP enabled)
#pragma config EBTR5 = OFF // External Block TRAP Enable bit (EBTR5 = 0:Block TRAP disabled; EBTR5 = 1:Block TRAP enabled)
#pragma config EBTR6 = OFF // External Block TRAP Enable bit (EBTR6 = 0:Block TRAP disabled; EBTR6 = 1:Block TRAP enabled)
#pragma config EBTR7 = OFF // External Block TRAP Enable bit (EBTR7 = 0:Block TRAP disabled; EBTR7 = 1:Block TRAP enabled)

// CONFIG6H
#pragma config EBTRB = OFF // External Block TRAP Enable bit (EBTRB = 0:Block TRAP disabled; EBTRB = 1:Block TRAP enabled)

// #pragma config statements should precede project enums
// Use project enums instead of #define
#include <xc.h>
#endif /* XC_HEADER_TEMPLATE_H */

```

- L'inclure dans la fonction **main**

```

#include <xc.h>
#include "config_27k40.h"

main()
{
    while(1)
    {
    }
}

```

- Debug Main Project

- window > target memory views > Memory Program : analyser le programme assembleur

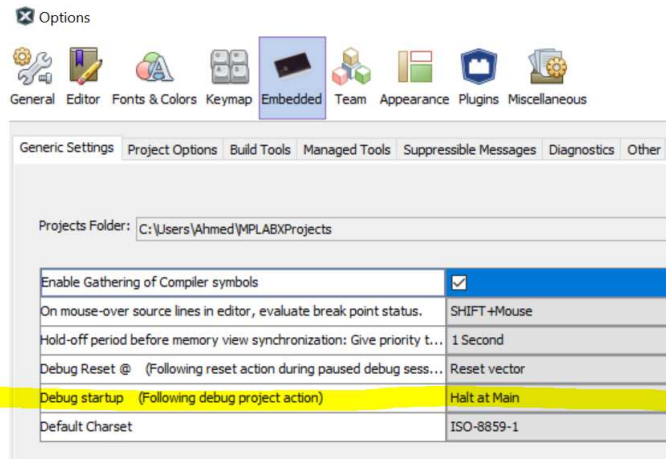
65 533	1FFF8	D7FF	main	BRA 0xFFFF8
65 534	1FFFA	0100		MOVLB 0x0
65 535	1FFFC	EFFC		GOTO 0x1FFF8
65 536	1FFFE	F0FF		NOP

Program Memory    Format Code

Compléter le programme pour allumer les LED D2, D3, D4 et D5 à tour de rôle

```
main()
{
    TRISA = 0x00; //PORTA en sortie (LED)
    LATA = 0x00; //LED éteintes
    while(1)
    {
        LATA = 0b00010000;
        LATA = 0b00100000;
        LATA = 0b01000000;
        LATA = 0b10000000;
    }
}
```

Menu déroulant (Tools >> Options) pour arrêter le **debugger** au début du **main** au démarrage



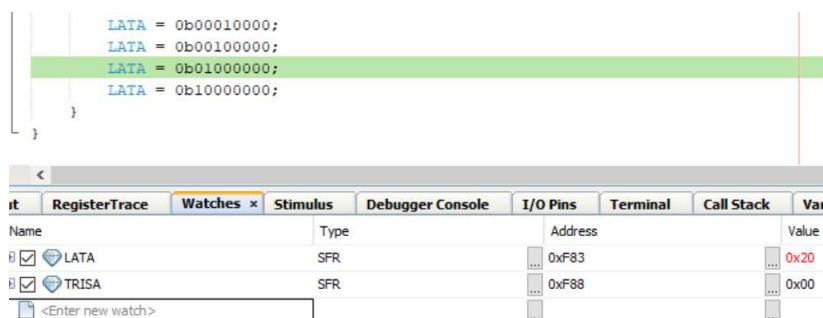
Maintenant, faire : **Debug Main Project**, le debugger doit s'arrêter au début du main.

```
4 main()
5 {
6     TRISA = 0x00; //PORTA en sortie (LED)
7     LATA = 0x00; //LED éteintes
8     while(1)
9     {
10        LATA = 0b00010000;
11        LATA = 0b00100000;
12        LATA = 0b01000000;
13        LATA = 0b10000000;
14    }
15 }
```

**Debug > New Watch** pour observer l'état des registres et variables. Ajouter les registres LATA et TRISA

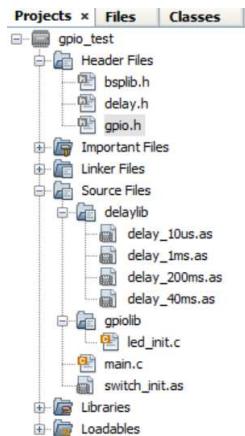


**Step Over (F8)** pour une exécution pas à pas. Observer l'évolution de LATA à chaque pas



## Programme gpio\_test, partie LED

- Créer un nouveau projet nommé **gpio\_test** dans le dossier **gpio/test/pjct**
- Associer les fichiers sources selon l'arborescence suivante :



- Commencer par compléter le fichier **gpio.h** (partie LED)

```
#define led2_on() // set output PORT bit : asm("...",...)

#define led3_on() ...

#define led4_on() ...

#define led5_on() ...

#define led2_off() // clear output PORT bit

#define led3_off() ...

#define led4_off() ...

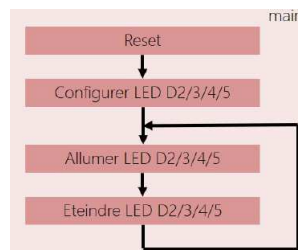
#define led5_off() ...
```

- Continuer avec **led\_init.c** (même travail pour toutes les LED)

```
void led2_init(void)
{
    /* TODO */

    //-> LATAbits...=...
    //-> TRISAbits...=...
}
```

- Programmer l'algorithme du polycopié de TP



- Tester le fonctionnement au simulateur
-

## Programme gpio\_test, partie poussoirs

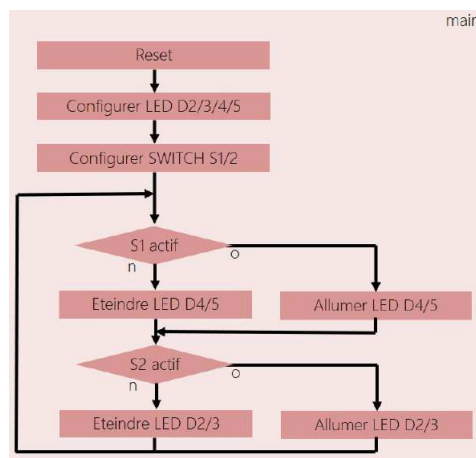
- Compléter le fichier **gpio.h** (partie switches)

```
#define switch1_read() // TRUE if PORT_bit_input = 1
#define switch2_read() ...
```

- Continuer avec **switch\_init.as** (même travail pour les 2 switches)

```
_switch1_init:
; TODO
//-> store BSR in bsr_tmp
//-> configure switch1 as input (TRISB...)
//-> select ANSELB bank
//-> configure switch1 as digital (ANSELB...)
//-> restore BSR register from bsr_tmp
return
```

- Programmer l'algorithme du TP



- Pour simuler les boutons poussoirs (entrées logiques sur les broches du PIC)

Fire	Pin	Action	Value	Units	Comments
	RC5	Toggle			
	RC5	Set High			Optional code
	RC5	Set Low			

Bouton de mise à feu      Broche      Action

- Step Over (F8)** pour une exécution pas à pas. Basculer entre la fenêtre précédente et la suivante pour observer l'effet des poussoirs sur les LED

Name	Type	Address	Value
LATA	SFR	0xF83	0x00
TRISA	SFR	0xF88	0x0F
PORTC	SFR	0xF8F	0x20

## Délai logiciel

### Coder la fonction `delay_10us.as`

```
PSECT bss0,class=BANK0,space=1

delay_cnt1: ds 1 ; 1 byte static memory allocation
bsr_tmp: ds 1

; linker memory static allocation directives for code
PSECT text, class=CODE, reloc=2, space=0

global _delay_10us

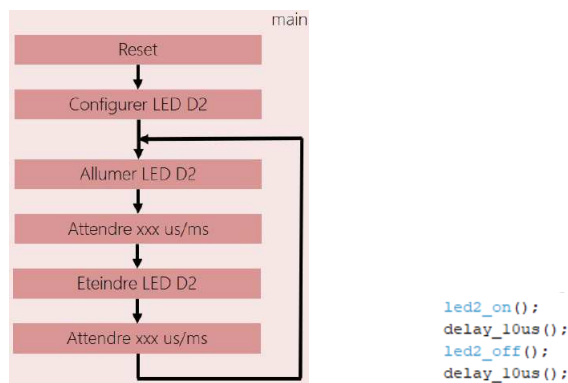
_delay_10us:
    ; TODO (->)

    ;-> save BANK nb in bsr_tmp var (BSR context)
    ;-> select BANK0 in data memory
    ;-> store value in delay_cnt1 var

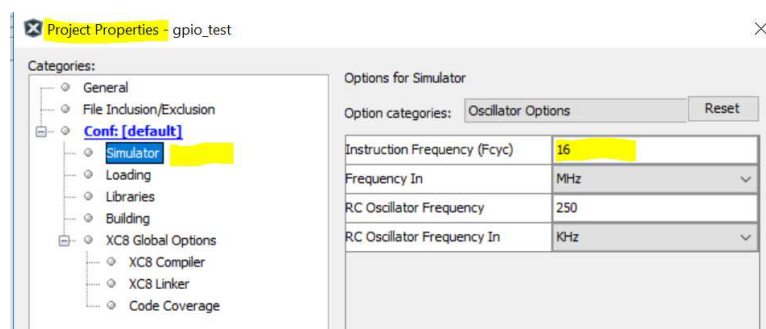
delay_10us_L1:
    ;-> decremente delay_cnt1, skip if zero
    ;-> branch to delay_10us_L1

    ;-> restore BSR context
    return
END
```

### Programmer l'algorithme du TP



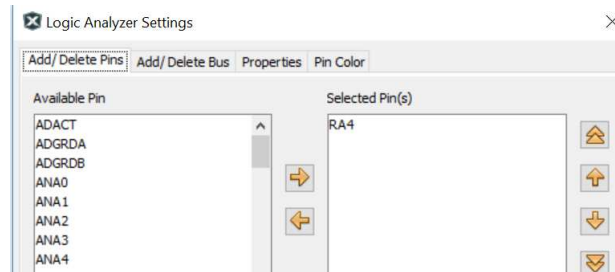
### Configurer le simulateur pour une fréquence de cycle machine = $64/4 = 16$ MHz



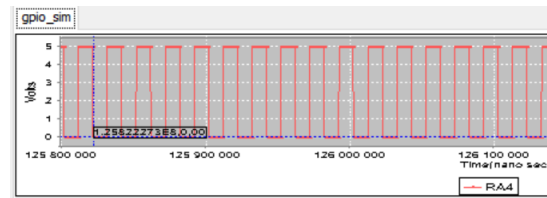
### Debug Project et faire run pour lancer l'exécution



### Window > Simulator > Logic analyzer : pour ouvrir l'analyseur logique. Aller dans Settings et sélectionner la broche RA4 (LED2)



🔧 Faire plusieurs fois **Continue** / **Pause** jusqu'à voir la courbe se tracer



🔧 Faire un zoom et mesurer la durée de la temporisation avec les curseurs

