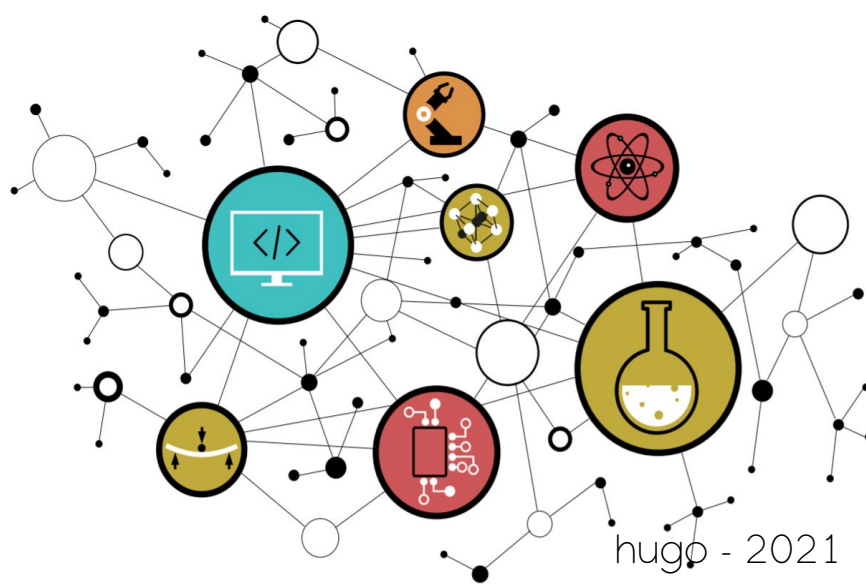


# TUTORIEL

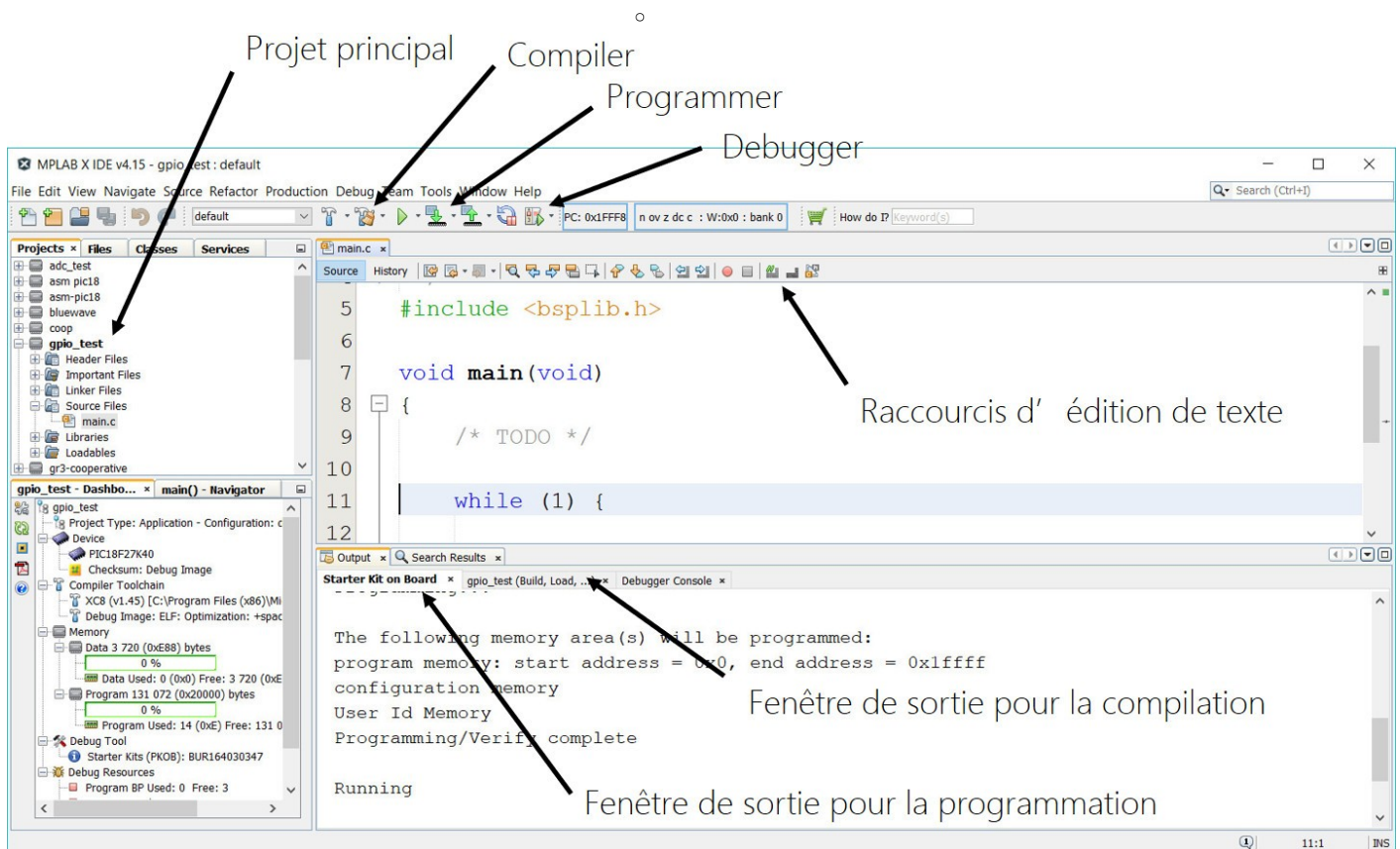
## CRÉATION DE PROJET SOUS IDE MPLABX

---



### ENVIRONNEMENT GRAPHIQUE MPLABX

Environnement de développement



Projet principal

Compiler

Programmer

Debugger

Raccourcis d' édition de texte

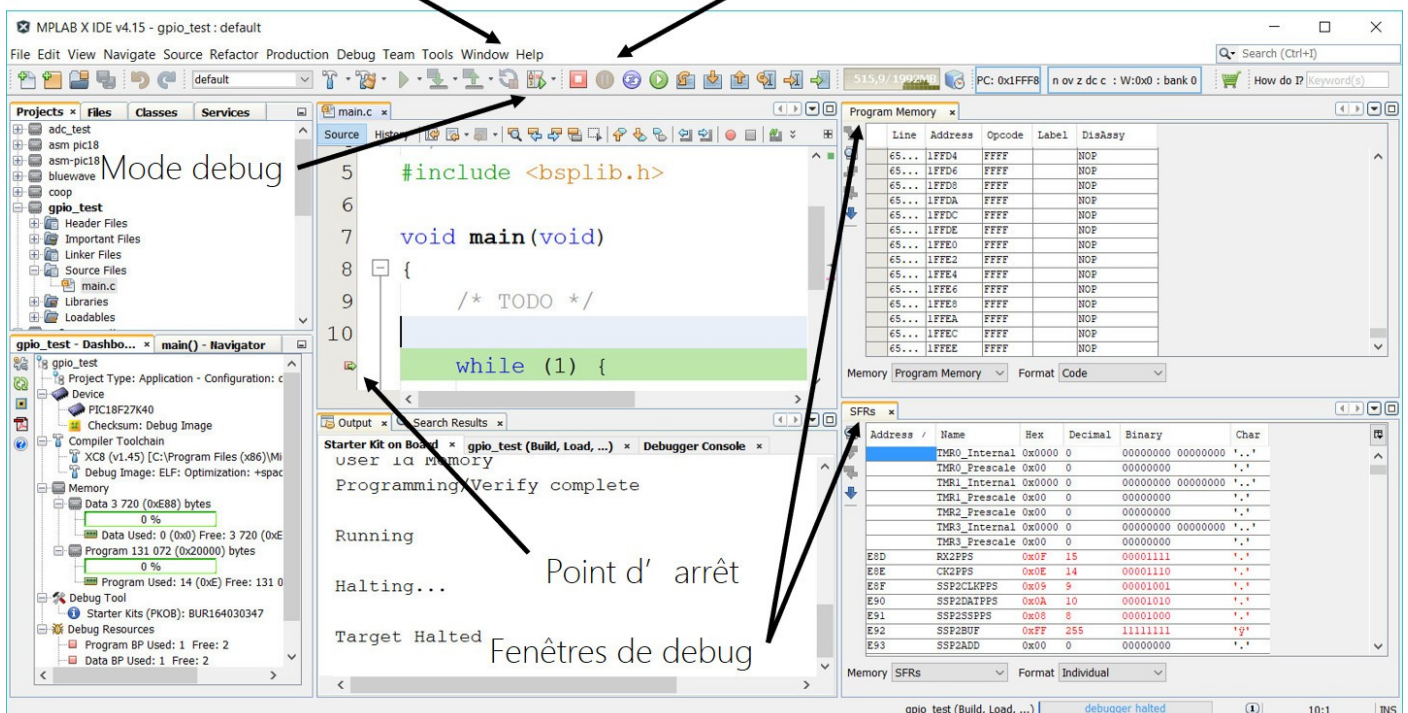
Fenêtre de sortie pour la compilation

Fenêtre de sortie pour la programmation

Environnement de debug

Gestion du fenêtrage

Panneau de contrôle de la sonde JTAG



Mode debug

Point d' arrêt

Fenêtres de debug

Target Halted

Line	Address	Opcode	Label	DisAssy
65...	1FFD4	FFFF	NOP	
65...	1FFD6	FFFF	NOP	
65...	1FFD8	FFFF	NOP	
65...	1FFDA	FFFF	NOP	
65...	1FFDC	FFFF	NOP	
65...	1FFDE	FFFF	NOP	
65...	1FFE0	FFFF	NOP	
65...	1FFE2	FFFF	NOP	
65...	1FFE4	FFFF	NOP	
65...	1FFE6	FFFF	NOP	
65...	1FFE8	FFFF	NOP	
65...	1FFEA	FFFF	NOP	
65...	1FFEC	FFFF	NOP	
65...	1FFEE	FFFF	NOP	

Address	Name	Hex	Decimal	Binary	Char
THR0_Internal	0x0000	0	00000000	00000000	..
THR0_Prescale	0x0000	0	00000000	00000000	..
THR1_Internal	0x0000	0	00000000	00000000	..
THR1_Prescale	0x0000	0	00000000	00000000	..
THR2_Internal	0x0000	0	00000000	00000000	..
THR2_Prescale	0x0000	0	00000000	00000000	..
THR3_Internal	0x0000	0	00000000	00000000	..
THR3_Prescale	0x0000	0	00000000	00000000	..
E8D	RX2PFS	0x0F	15	00001111	..
E8E	CH2PFS	0x0E	14	00001110	..
E8F	SSP2CLKPFS	0x09	9	00001001	..
E90	SSP2DA1PFS	0x0A	10	00001010	..
E91	SSP2SPPFS	0x08	8	00001000	..
E92	SSP2BUF	0xFF	255	11111111	..
E93	SSP2ADD	0x00	0	00000000	..

### CRÉATION DE PROJET SOUS MPLABX

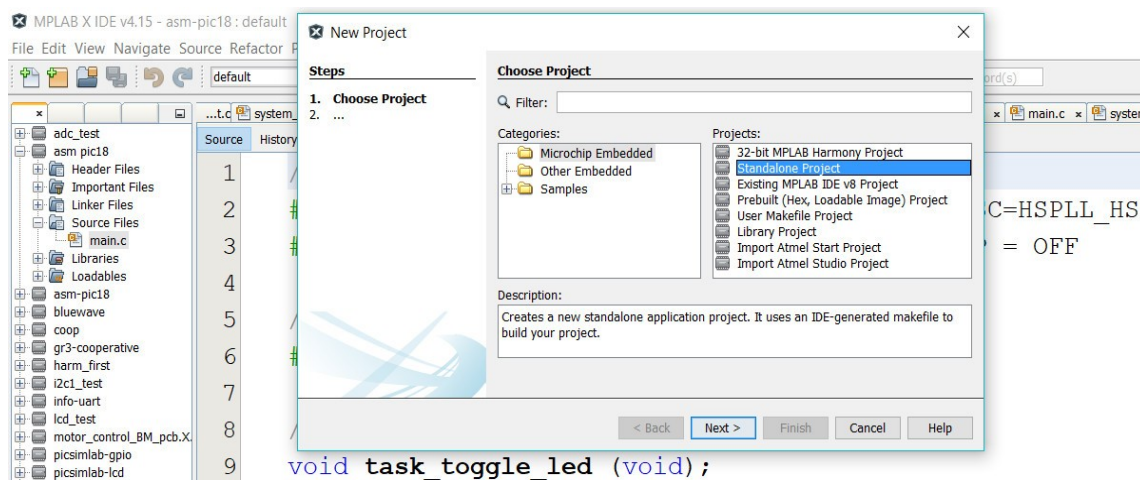


Il est à noter que la totalité des développements pourrait être réalisée sans environnement de développement ou IDE (Integrated Development Environment). Nous utiliserions alors la console système (UNIX ou MS-DOS) pour invoquer et paramétrer la chaîne de compilation XC8 (toolchain 8bits Microchip). De même, l'édition des fichiers sources se ferait depuis un éditeur de texte (vi, vim, nano, Notepad++, etc), l'automatisation du processus de compilation par édition d'un Makefile, etc.

Néanmoins, dans bien des cas, l'utilisation d'un IDE peut s'avérer pratique. Un IDE est dédié à centraliser et lier dans un même outil tous les services nécessaires à l'ingénieur développeur (gestion et paramétrage des chaînes de compilation, génération de Makefile, éditeur de texte intégré, interface et gestion des outils de debug, interface graphique développeur, etc).

#### Sélection du type de projet

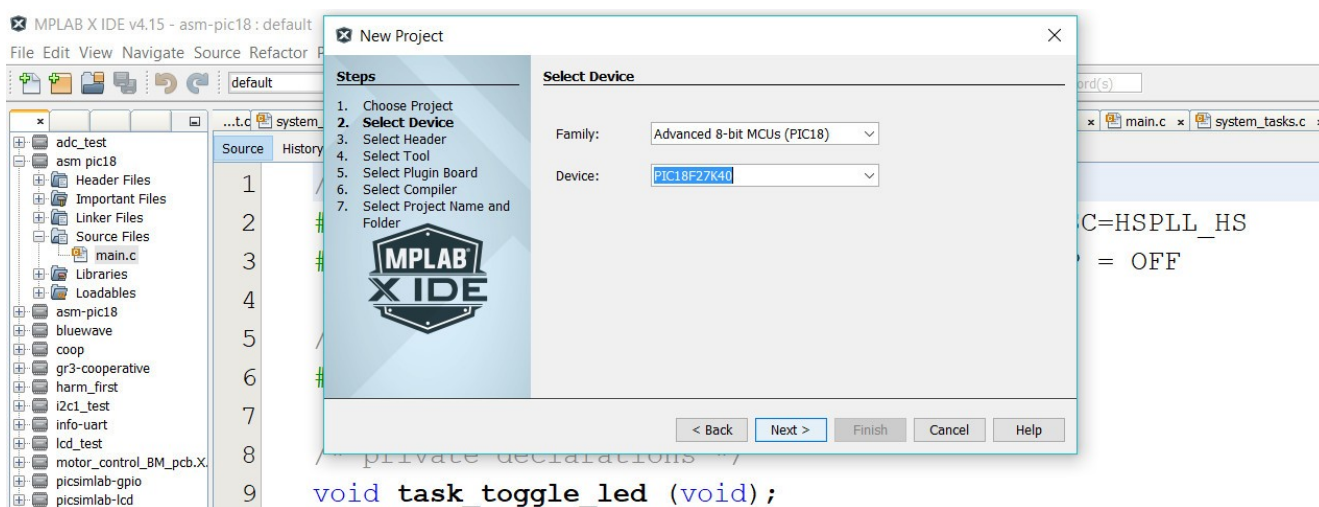
- Télécharger l'archive *mcu.zip* sur la plateforme *moodle* dans *download*
- Ouvrir MPLABX → *File* → *New Project...*
- Sélectionner le type de projet souhaité : *Standalone* (développement d'application), *Prebuilt* (charger un firmware précompilé), *Library* (développement bibliothèque statique), etc
- *Microchip Embedded* → *Standalone Project* → *Next*





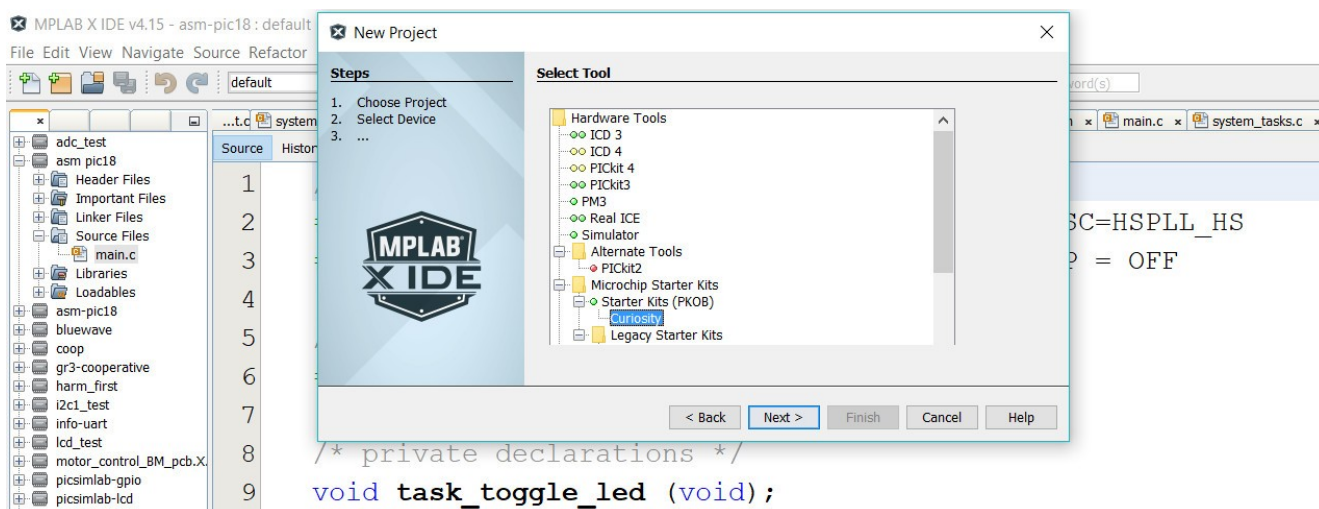
### Sélection du processeur cible

- L'étape qui suit est à adapter en fonction du MCU ciblé par les développements en cours (*PIC18F27K40 en TP 1A Systèmes Embarqués*). Attention, le nom du MCU doit être impérativement le bon, sous peine de voir apparaître des erreurs au chargement sur la cible ou à la compilation. Néanmoins, toutes les étapes présentées dans ce tutoriel sont modifiables par la suite après création du projet.
- *Family* → *Advanced 8-bits MCUs (PIC18)*
- *Device* → *PIC18F27K40* → *Next*

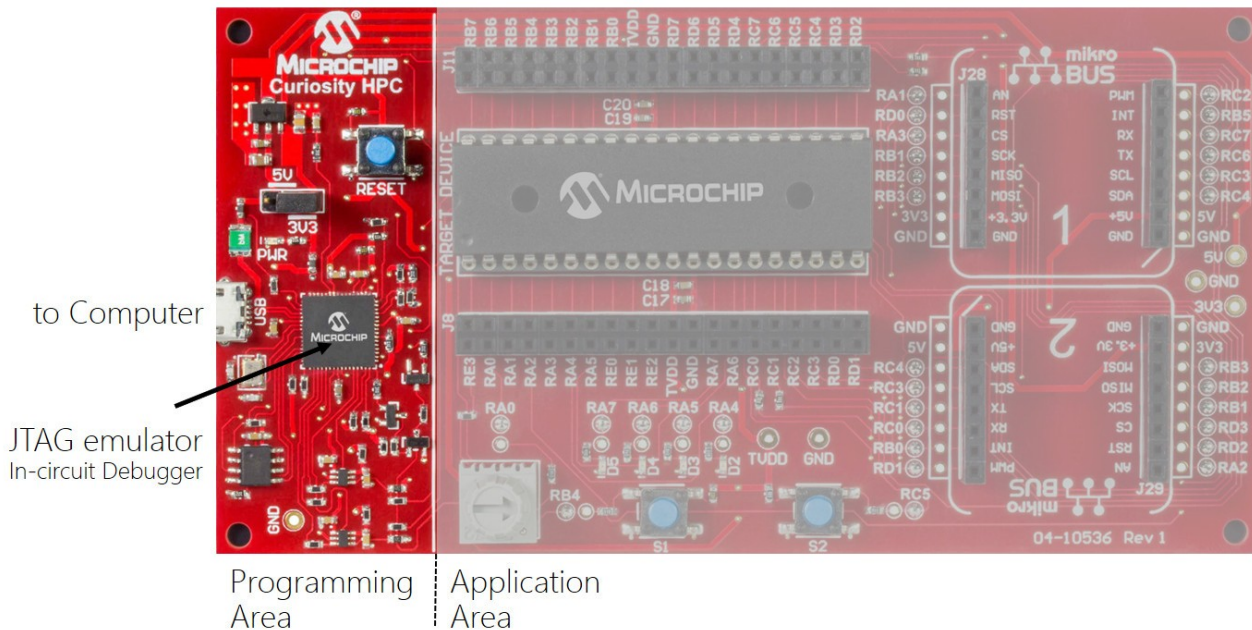


### Sélection de la sonde de programmation ou du starter kit

- Sélectionner l'outil de chargement du fichier exécutable de sortie (ELF, HEX, COFF, setc) vers le processeur cible. Nous parlons de sonde JTAG (Join Test Action Group, norme IEEE 1149.1), ou sonde de programmation, ou sonde d'émulation, etc.
- *Starter Kits (PKOB)* → *Curiosity* → *Next* (si carte physique en possession) ou *Simulator* → *Next*

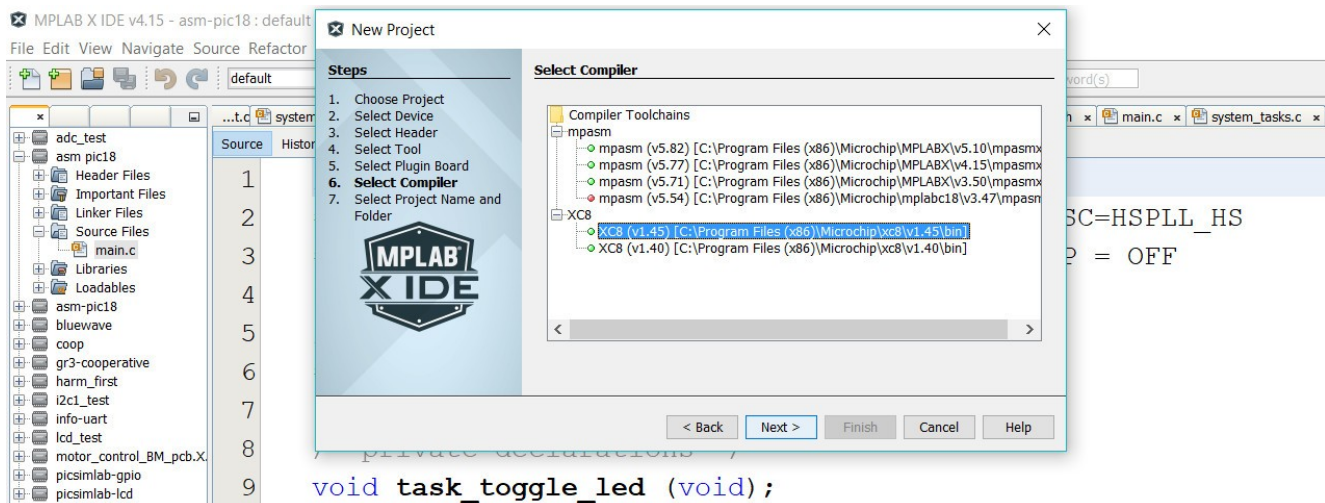


- Depuis sa normalisation en 1990, le JTAG est devenu le plus grand standard dans l'industrie pour la programmation et le test de processeur. Observons la sonde de programmation JTAG présente sur la carte de démarrage ou starter kit Curiosity HPC de Microchip



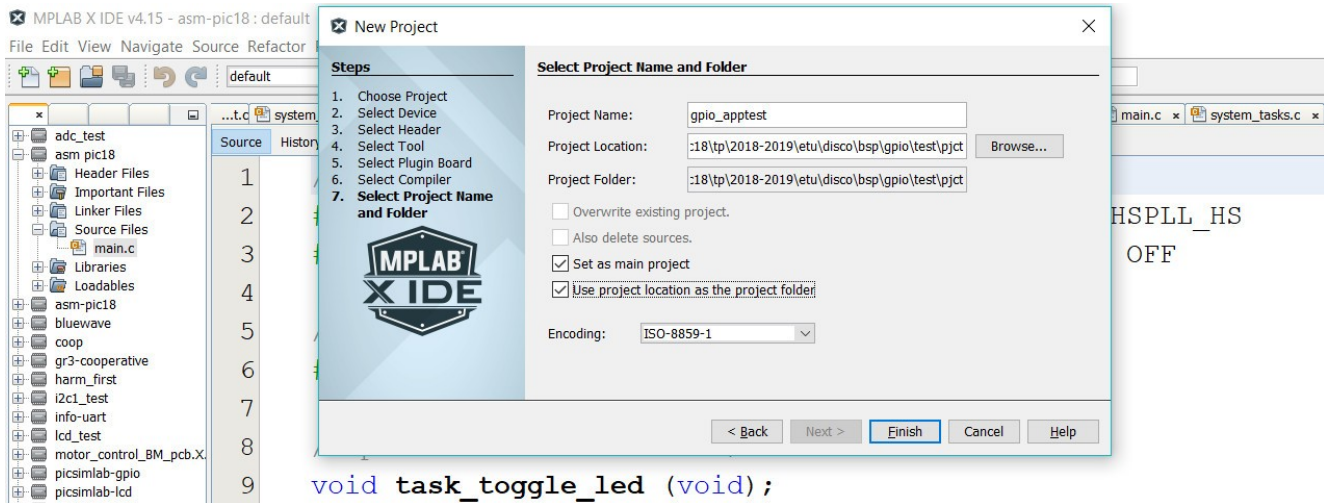
### Sélection de la chaîne de compilation

- Sélectionner la chaîne de compilation. Sur un même ordinateur de développement, un certain nombre de toolchains peuvent être installées. De même, pour une même chaîne de compilation dédiée à une famille de CPU, plusieurs versions peuvent également être installées. L'exemple ci-dessous montre que l'IDE MPLABX a reconnu plusieurs toolchain C pour PIC18. Dans le cas présent, 2 versions de XC8, la chaîne de compilation développée par Microchip pour leur famille de CPU PIC18.
- XC8 (v1.45) → Next*



### Sélection de l'emplacement du projet

- En première année, l'arborescence du système de fichiers du projet vous est donnée et a été spécifiée par l'architecte du projet. Construire une telle arborescence demande déjà une certaine maturité en ingénierie afin de la penser parcimonieuse, simple, claire, modulable, etc. Ce travail vous sera demandé durant les projets de 2<sup>ième</sup> et 3<sup>ième</sup> année en spécialisation. Chaque sous projet possède une arborescence locale commune (cf. ci-dessous). Toujours créer vos projets dans le répertoire *pjct* correspondant au travail demandé. De même, toujours donner à vos projets un nom en lien avec les développements en cours. Par exemple, gpio, adc\_pjct, uart\_test et donc éviter des nommages ne permettant pas de connaître le contenu du projet, par exemple test, projet, tp1, exo3, etc :
  - src* (fichiers sources .c)
  - include* (fichiers d'en-tête .h)
  - pjct* ou *test/pjct* (emplacement des fichiers internes de MPLABX, Makefile, fichiers de compilation et de production .obj, .elf, etc)
- Project Name* → *<choisir\_un\_nom\_court\_sans\_accent\_sans espace\_ayant\_un\_sens>*
- Project Location* → *<emplacement\_sur\_votre\_ordinateur>/mcu/tp/disco/bsp/gpio/test/pjct* (à adapter d'un projet à un autre)
- [x] Use project location as the project folder* → *Finish*

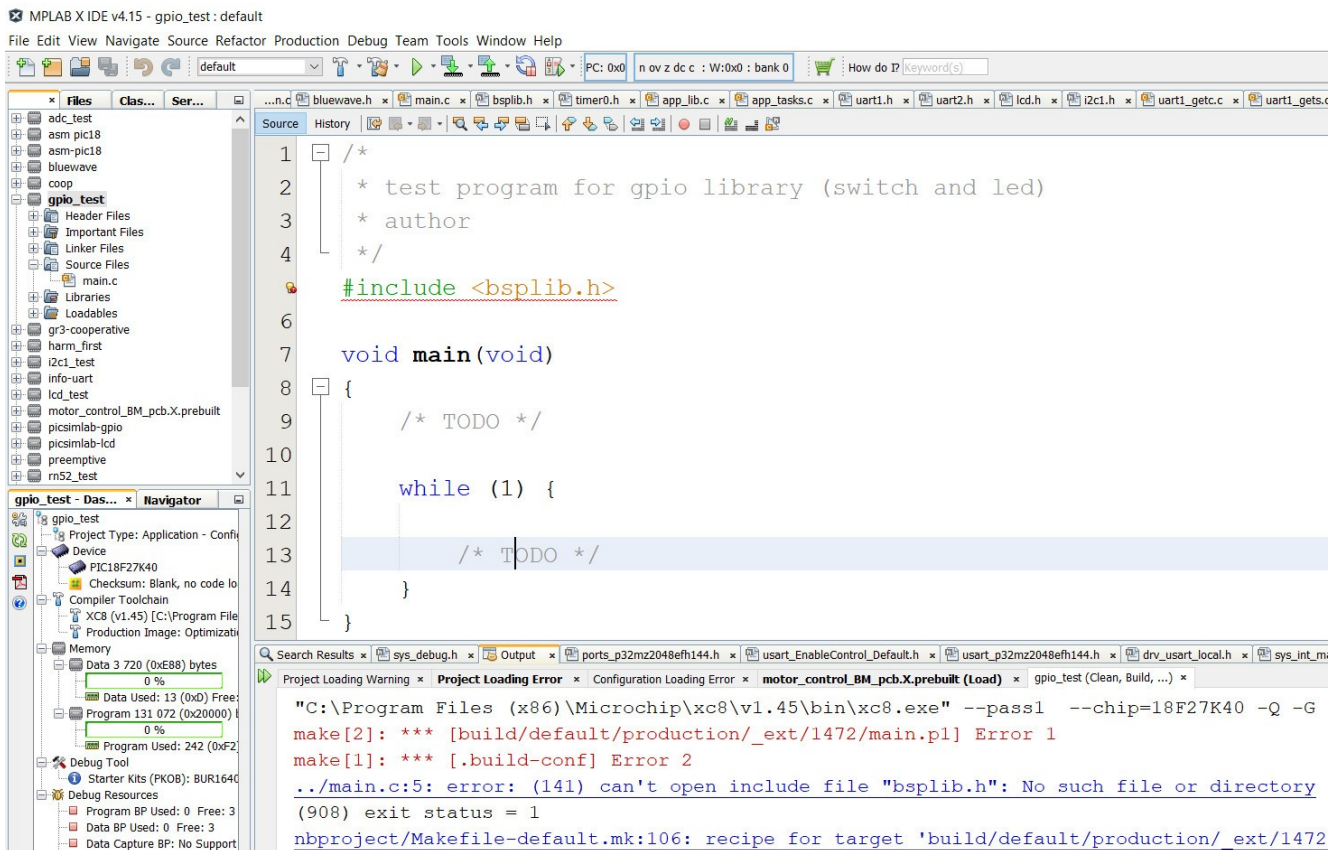


- La création du projet est maintenant terminée. Reste à inclure les sources, configurer les outils de compilation et s'assurer de la bonne compilation du projet complet. Les développements pourront alors commencer. Vous pouvez d'ailleurs constater sur chaque capture d'écran précédente que de nombreux projets avaient déjà été créés sur mon ordinateur personnel de travail.



### Ajouter les fichiers sources

- Clic droit sur le nom du projet (fenêtre à gauche) → Set as Main Project
- Clic droit sur Source Files → Add Existing Items... → <your\_parth>/test/main.c → Select
- Compiler le projet. Clic droit sur le nom du projet → Build ou cliquer le marteau et le balai

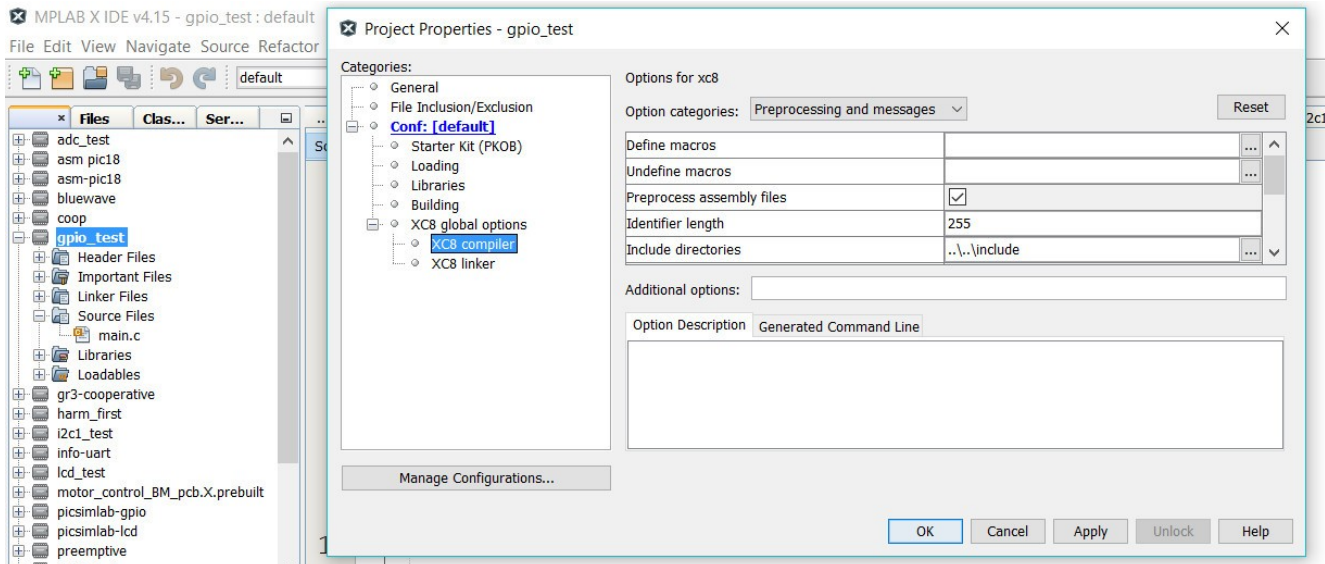


- Première erreur de compilation. Pour être plus précis, il s'agit du préprocesseur du C qui ne trouve pas le fichier *bsplib.h*. En effet, ce fichier a été développé spécifiquement pour l'application de TP sur la carte Curiosity HPC. La chaîne de compilation ne fera jamais d'hypothèse sur l'emplacement potentiel d'un fichier d'en-tête (header ou .h) applicatif. Les chemins doivent être explicitement précisés à la chaîne de compilation (toolchain) !

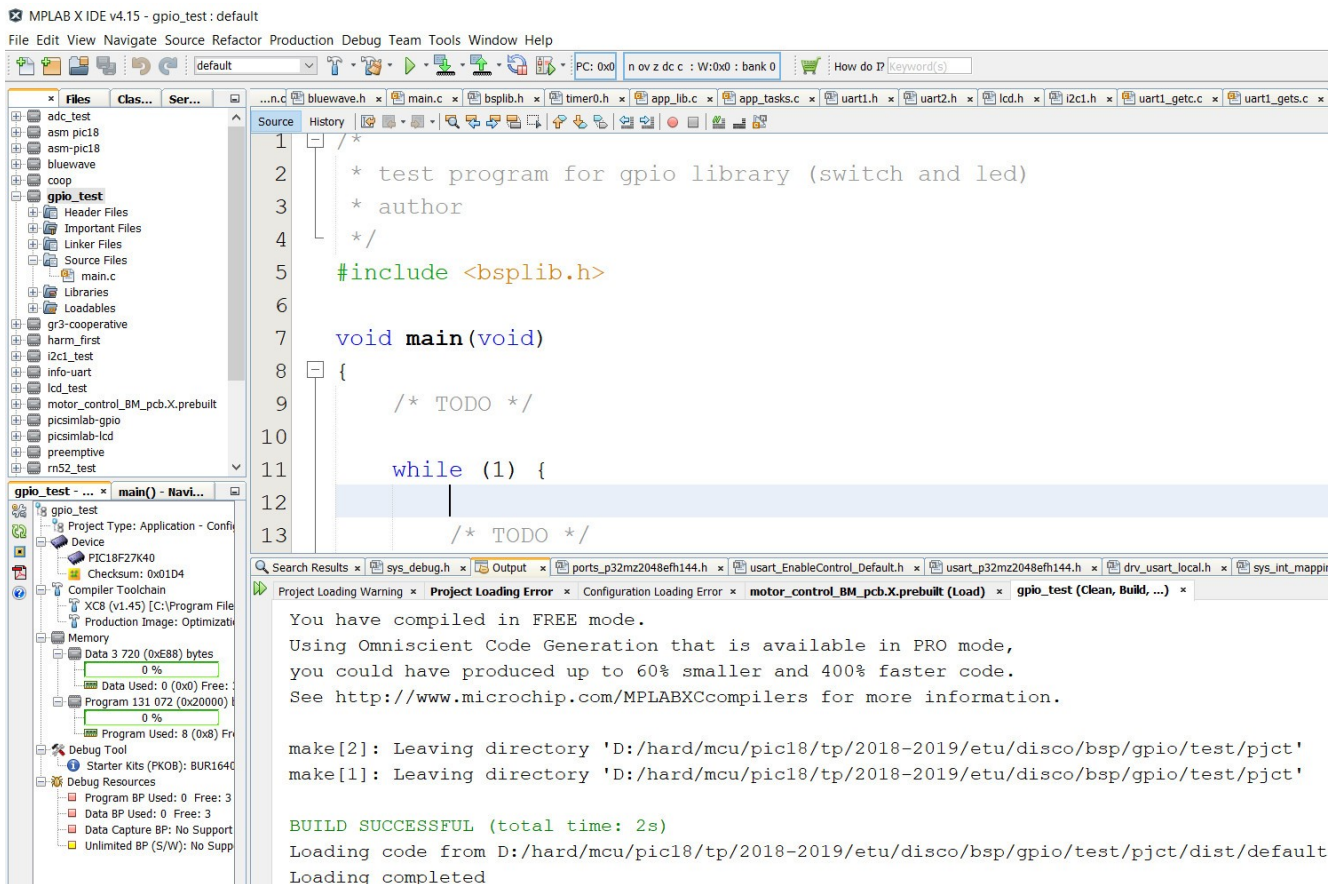
### Configurer les outils de compilation

- Clic droit sur le nom du projet → Properties
- Cliquer sur XC8 compiler
- Option Categories → Preprocessing and message
- Vérifier l'emplacement du fichier. Recherche Windows de *bsplib.h* dans le répertoire *disco*

- *Include directories* → ... → *Browse...* → *<your\_path>/bsp/* → *Apply*
- Vous constaterez que l'IDE définit par défaut des chemins relatifs à l'emplacement courant du projet sur la machine. Cela améliore la portabilité des projets de machine en machine. *Attention, le chemin illustré ci-dessous ne sera peut-être pas le même que celui sur votre machine. Tout dépend de l'emplacement où aura été créé le projet !*



- Compiler le projet. *Clic droit sur le nom du projet → Build*

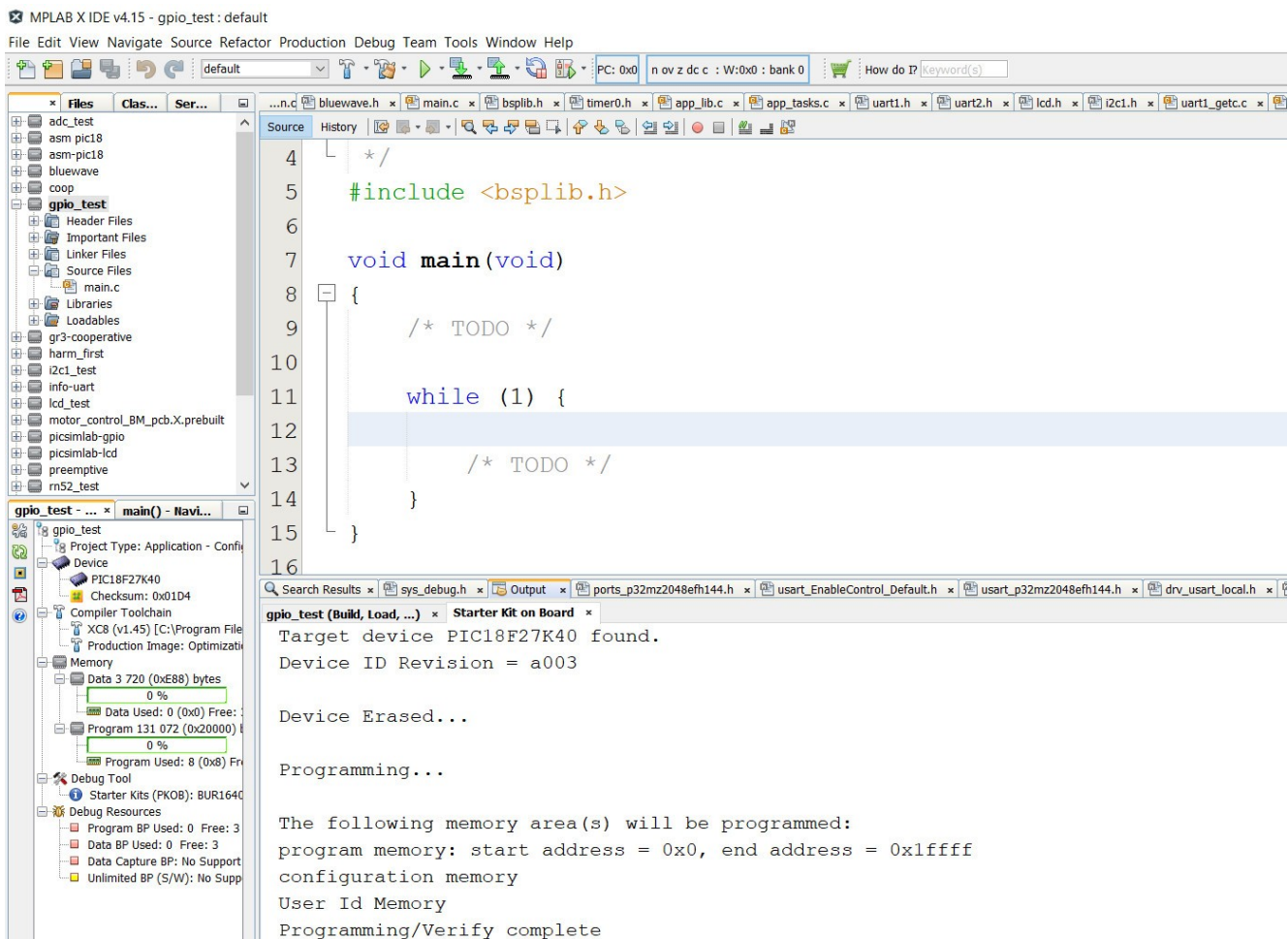


- *Le projet compile. Reste maintenant à charger le firmware de sortie sur la cible !*



### Charger et exécuter le programme sur la cible

- Une fois la compilation du projet validée, il reste à charger le firmware en mémoire programme flash interne du MCU (ou sur simulateur logique). Nous disons souvent "flasher" le processeur. Le transfert se fera par la sonde JTAG précédemment présentée. Au chargement, vous constaterez que la fenêtre de sortie de l'IDE commutera de la fenêtre "build, load,..." (compilation) à "Starter Kit on Board" (programmation et exécution). L'IDE est alors en train d'échanger avec la sonde JTAG.
- Clic droit sur le nom du projet → Run* ou icône ci-dessous "Fichier avec flèche verte vers le bas en direction de la puce du microcontrôleur"



- L'affichage **Programming/Verify complete** spécifie le bon chargement du firmware dans le MCU cible et le début de son exécution. Il reste maintenant à vérifier le bon fonctionnement de votre application !*

### Test sur carte physique Curiosity HPC

- Ouvrir les propriétés du projet : *Fenêtre Projects > clic droit sur le nom du projet > Properties*
- Sélectionner la carte Curiosity HPC : *Hardware Tool > Microchip Kits > Curiosity > Apply > OK*
- Compiler et exécuter le programme : *Fenêtre Project > clic droit sur le nom du projet > Run*

### Test sur simulateur MPLABX IDE (sans carte de développement)

- Ouvrir les propriétés du projet : *Fenêtre Projects > clic droit sur le nom du projet > Properties*
- Sélectionner le simulateur : *Hardware Tool > Simulator > Apply > OK*
- Compiler et exécuter le programme : *Fenêtre Projects > clic droit sur le nom du projet > Debug*
- Arrêter la simulation en cliquant sur le bouton carré rouge STOP (ou [Shift] + [F5] )

### Test sur simulateur de carte PICSimLab

- Sous MPLABX, compiler le programme : *Fenêtre Projects > clic droit sur le nom du projet > Clean and Build*
- Sous MPLABX, observer après compilation le dossier où a été sauvé le fichier .hex de sortie:

BUILD SUCCESSFUL

Loading code from `/<your_computer_path>/dist/default/production/<your_project_name>.hex`

- Ouvrir le simulateur de carte PICSIMLab : *File > Load Hex > charger le fichier .hex précédemment généré.*
- Ne pas hésiter à utiliser le module oscilloscope pour analyser la sortie : *Modules > Oscilloscope*