Edu-C – FAQ (foire aux questions)

Edu-C, EduC, educ ? Quelle est la logique?

Edu-C est le nom du produit. EduC aussi si l'esthétique typographique l'exige. EduC pour les activités. Les fichiers sont nommés EduC-NomDecrivantLeContenu,extension
EduC désigne aussi la librairie EduC.h, complétée actuellement par Oled.h. Il y aura inévitablement évolution, donc soit des librairies supplémentaires, soit des librairies parallèles à installer (EduC2.h). Arduino ne permet pas de recharger une librairie sous le même nom, ce qui se comprend.
Sur le site Didel, tous les dossiers (sauf exception compréhensible) ont un nom en minuscules, et tous les fichiers ont une première lettre majuscule avec la séparation des mots "en dos de chameau".
Le format des fichiers (extension) est soit pdf soit HTML s'il y a plusieurs références où que cela facilite le copier-coller des programmes.
Donc, tout ce qui concerne l'Edu-C est en principe sous www.didel.com/educ/
Comme on le voit en haut de la page, ce fichier est mis à jour
sous www.didel.com/educ/EduC-FAQ.pdf

Lettres accentuées

Arduino tolère les minuscules accentuées dans les programmes. Certains programmes les convertissent en séquence Iso. Sous Notepad++, les menus Edition, Affichage, Encodage, montrent les options qui existent et permettent des conversions.

Pourquoi DelMs() plutôt que le delay() d'Arduino?

Le compilateur reconnaît l'instruction delay(); et met en route les interruptions, ce qui ne sert à rien puisqu'il faut de toute façon attendre. Aussi, delay(); est une fonction, la première lettre devrait être une majuscule. Mais comme pour le digitalWrite, des programmeurs avec d'autres habitudes sont intervenus.
Avec la librairie EduC, vous pouvez écrire delay(). Il est converti en DelMs() par un #define. Mais delayMicrosecond() n'a pas été converti. Ecrivez le programme le plus simple et notez la taille. Ajoutez un delayMicrosecond(100). La taille du programme a passablement augmenté!

Pourquoi ne pas utiliser des digitalWrite?

Si je vois digitalWrite(6,0); dans un programme, je ne sais vraiment pas de quoi il s'agit.
Si je vois digitalWrite (LED, HIGH); je sais qu'il s'agit d'une Led, mais HIGH allume ou éteint? Cela dépend du câblage.
Si on a défini ON au lieu de HIGH pour dire quelle tension allume, c'est clair. Mais comment faire s'il y a plusieurs Leds, certaines allumées par un 1, d'autre par un 0?
Les digitalWrite ne doivent apparaître que dans un bloc de définitions, jamais dans le programme. On réfléchit une fois et c'est noté dans un fichier de définition initial. Si on modifie le câblage, on modifie à cet endroit. Ce serait absurde et risqué de devoir modifier à plusieurs endroits dans le programme.

Pourquoi Arduino n'utilise pas les #define?

Arduino croit que c'est trop difficile. C'est absurde d'associer un nom à une position mémoire, comme
int LedR=3; pour dire que LedR est un nombre qui a la valeur 3. Cela permet de faire des opérations sur LedR sans que le compilateur rouspète.
#define LedR 3 est vraiment difficile à comprendre. L'absence de ; à la fin peut troubler, mais si on la met dans les cas simples, on ajoute dans le cas pire une instruction neutre inutile.

Pourquoi Arduino ignore les "byte?

C'est plus simple de n'avoir qu'un type d'objet. Dans 16 bits on peut mettre 8 bits, et on vous dit tout de suite qu'il y a assez de mémoire, et que vous pouvez acheter l'Arduino machin qui en a encore plus.
L'AVR328 des Arduino usuels est un processeur 8 bits. Il doit prendre 2 positions mémoires pour avoir 16 bits, les instructions arithmétiques sont plus de 2 fois plus longues en 16 bits qu'en 8 bits.

Quand on a des applications temps réel, cela a de l'importance. Evidemment on vous dira que les nouveaux processeurs sont tous 16,32 ou 64 bits et travaillent à 100 MHz.
Si vous avez choisi Arduino, c'est pour avoir un outil facile à apprendre et à maîtriser. Il n'a pas besoin d'être en or (même si l'or est très bon marché). Le problème du débutant, c'est de comprendre. Pour comprendre le bois, on commence avec une scie à main pas avec une machine à commande numérique.

Pourquoi des fonctions avec et sans()

RougeOn; n'a pas de () parce que c'est une définition, une macro. Le C permet des macros compliquées, mais nous ne les utilisons que pour décrire le matériel, là ou Arduino utilise des digitalWrite.
Les fonctions qui n'ont pas de paramètre ont toujours une parenthèse vide ().
Attention de ne oublier les () dans une fonction, le compilateur ne dit rien et le programme plante!
Par contre, si vous mettez une parenthèse vide quand la fonction a besoin d'un paramètre, l'erreur est reconnue et signalée.Si vous écrivez LedGOn(); l'erreur est signalée.
On peut très bien définir #define LedGOn() bitClear (PORT,bg) et utiliser LedGOn(); Mais alors, LedGOn; plante sans être signalé, comme pour les fonctions. C'est un point incompréensible du C.

Quand faut-il un espace et un retour de ligne

`if (!PousD)` ou `if ( ! PousD )` ? Les deux sont acceptés, l'espace est ignoré. Il y a toujours un signe pour séparer les noms. Il faut choisir ce qui est le plus lisible. Le retour de ligne termine l'instruction. Si l'écran fait passer la suite de la ligne en dessous, ce n'est pas une fin de ligne pour le C. Le signe \ en fin de ligne permet de continuer l'instruction sur la ligne écran suivante.
On peut avoir des lignes vides, et il faut mettre des lignes vides pour séparer les blocs d'instructions; c'est un excellent commentaire !
Le signe \ en fin de ligne permet de continuer l'instruction sur la ligne écran suivante, ce qui est nécessaire dans des grands tableaux.

next