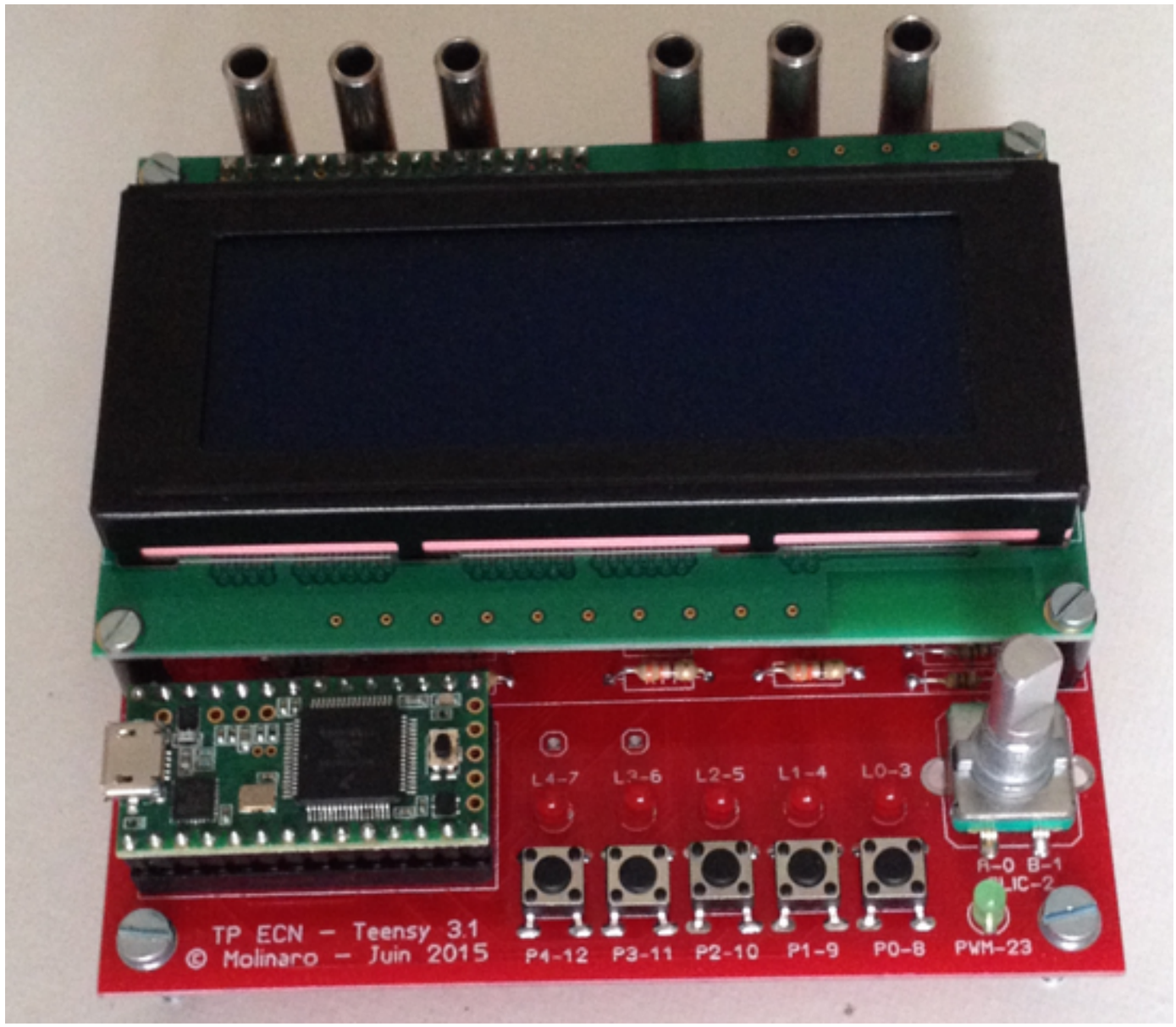
Temps Réel


L4-7
L3-6
L2-5
L1-4
L0-3
R-0 B-1
LIC-2
TP ECN - Teensy 3.1
© Molinaro - Juin 2015
P4-12
P3-11
P2-10
P1-9
P0-8
PWM-23

But de cette partie

Objectif :

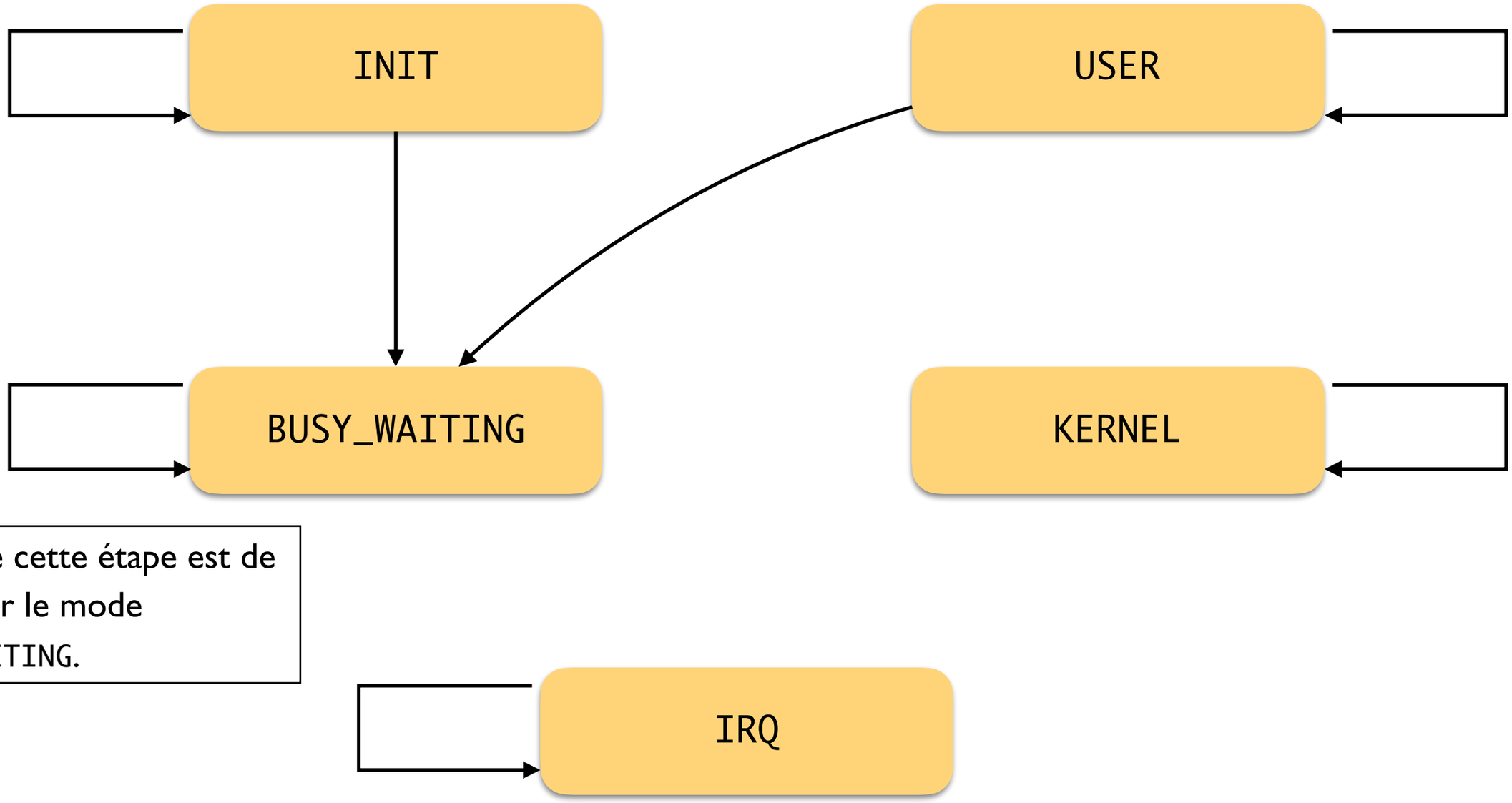
- *suppression du mode BUSY_WAITING de façon que les tâches invoquent une attente passive lors de l'accès à l'afficheur LCD.*

Travail à faire :

- reprendre le programme précédent, avec des tâches qui ne se terminent pas, l'une d'entre elles affichant régulièrement une information (par exemple un compteur) sur l'afficheur LCD. Dans l'étape précédente, l'accès de la tâche à l'afficheur LCD appelait de manière indirecte `busyWaitingDuringMS`, la led *Teensy* (activité du processeur) s'éclairait fortement durant l'accès. À l'issue de cette étape, après suppression du mode `BUSY_WAITING` et adaptation du code, la led *Teensy* reste très faiblement éclairée durant l'exécution du programme.

Graphe des modes logiciels

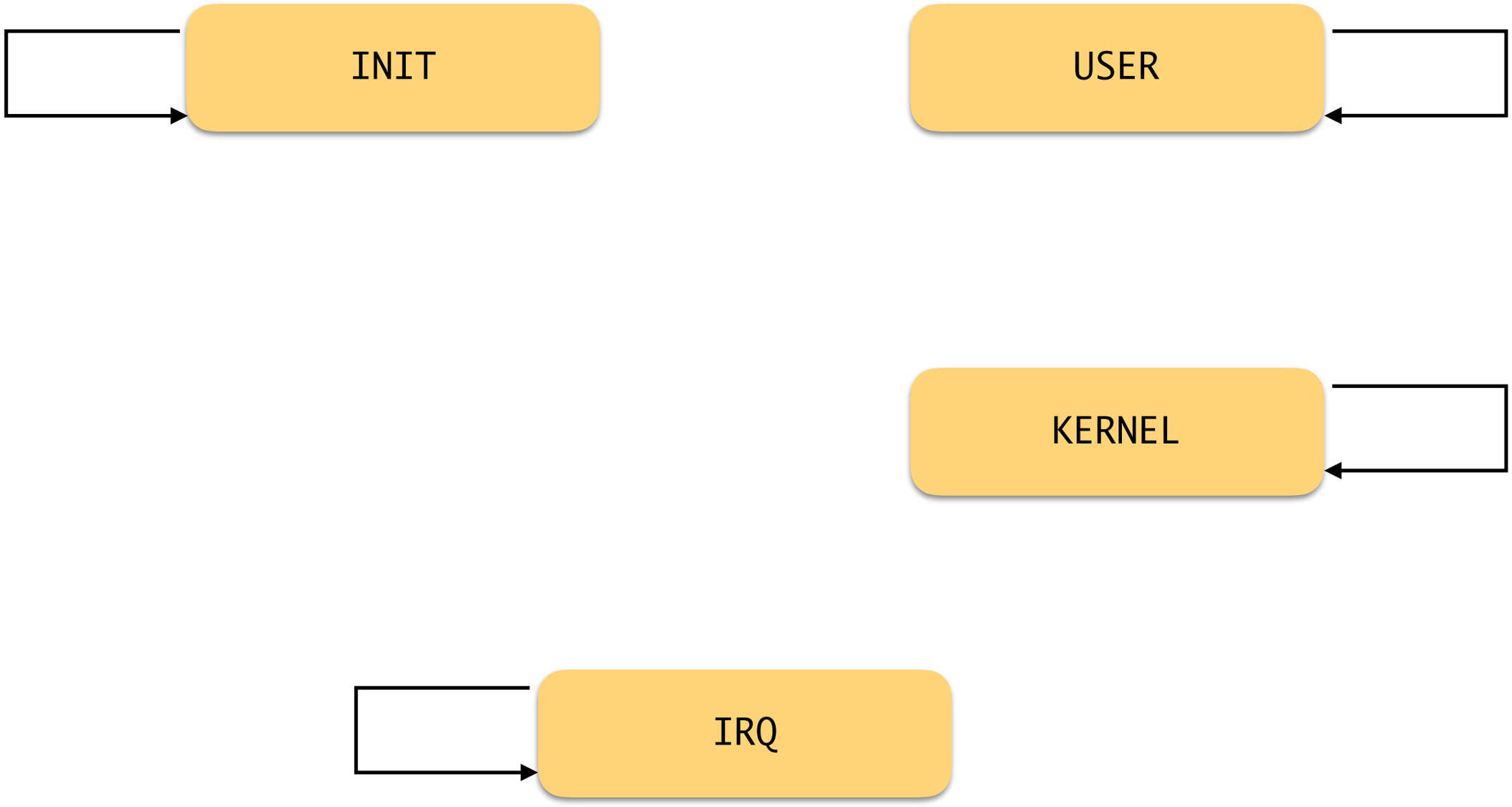
Présenté à l'étape 10, modifié dans cette étape



Le but de cette étape est de supprimer le mode `BUSY_WAITING`.

Graphe des modes logiciels

À partir de cette étape.



Comment supprimer le mode BUSY_WAITING

Éditez le fichier logical-modes.h et supprimer la déclaration de la classe BUSY_WAITING_mode_class (lignes 87 à 111) :

```
//-------------------------------------------------------------------------------*
//   B U S Y    W A I T I N G    M O D E                                          *
//-------------------------------------------------------------------------------*

#ifdef CHECK_ROUTINE_CALLS
  class BUSY_WAITING_mode_class {
    private : BUSY_WAITING_mode_class (void) ;
    private : BUSY_WAITING_mode_class & operator = (const BUSY_WAITING_mode_class &) ;

    public : BUSY_WAITING_mode_class (const BUSY_WAITING_mode_class &) ;
    public : BUSY_WAITING_mode_class (const INIT_mode_class &) ;
    public : BUSY_WAITING_mode_class (const USER_mode_class &) ;
  } ;
#endif

//-------------------------------------------------------------------------------*

#ifdef CHECK_ROUTINE_CALLS
  #define BUSY_WAITING_MODE const BUSY_WAITING_mode_class MODE
  #define BUSY_WAITING_MODE_ const BUSY_WAITING_mode_class MODE,
#else
  #define BUSY_WAITING_MODE void
  #define BUSY_WAITING_MODE_
#endif
```

Évidemment, le code ne compile plus ! Nous allons voir dans la suite comment y remédier.

busyWaitingDuringMS **et** waitDuringMS

Nous avons deux routines d'attente de délai :

- busyWaitingDuringMS qui effectue une attente active ;
- waitDuringMS qui effectue une attente passive.

Les contraintes :

- lors de l'initialisation de l'afficheur LCD (INIT_MODE), nous avons besoin d'une attente de délai ; or, l'exécutif n'est pas encore opérationnel à ce moment là ; on appellera donc busyWaitingDuringMS durant l'initialisation ;
- lors de l'accès à l'afficheur LCD par une tâche (donc USER_MODE), nous avons aussi besoin d'une attente de délai ; comme l'exécutif est opérationnel, on appellera donc waitDuringMS.

Nous voulons modifier le code de la façon suivante :

- busyWaitingDuringMS pourra être appelé en INIT_MODE, et ne pourra pas être appelé en USER_MODE ;
- waitDuringMS pourra être appelé en USER_MODE, et ne pourra pas être appelé en INIT_MODE.

L'intérêt des modes est qu'il permet à la compilation de vérifier le bon usage de ces fonctions.

Modification de busy-waiting.h et busy-waiting.c

Modifier la déclaration de busyWaitingDuringMS dans busy-waiting.h :

```
void busyWaitingDuringMS (INIT_MODE_ const uint32_t inDurationInMS) ;
```

Modifier en conséquence busy-waiting.c.

Compilation de `lcd.c`

Si vous recompilez le projet, les seules erreurs qui apparaissent sont dans `lcd.c`.

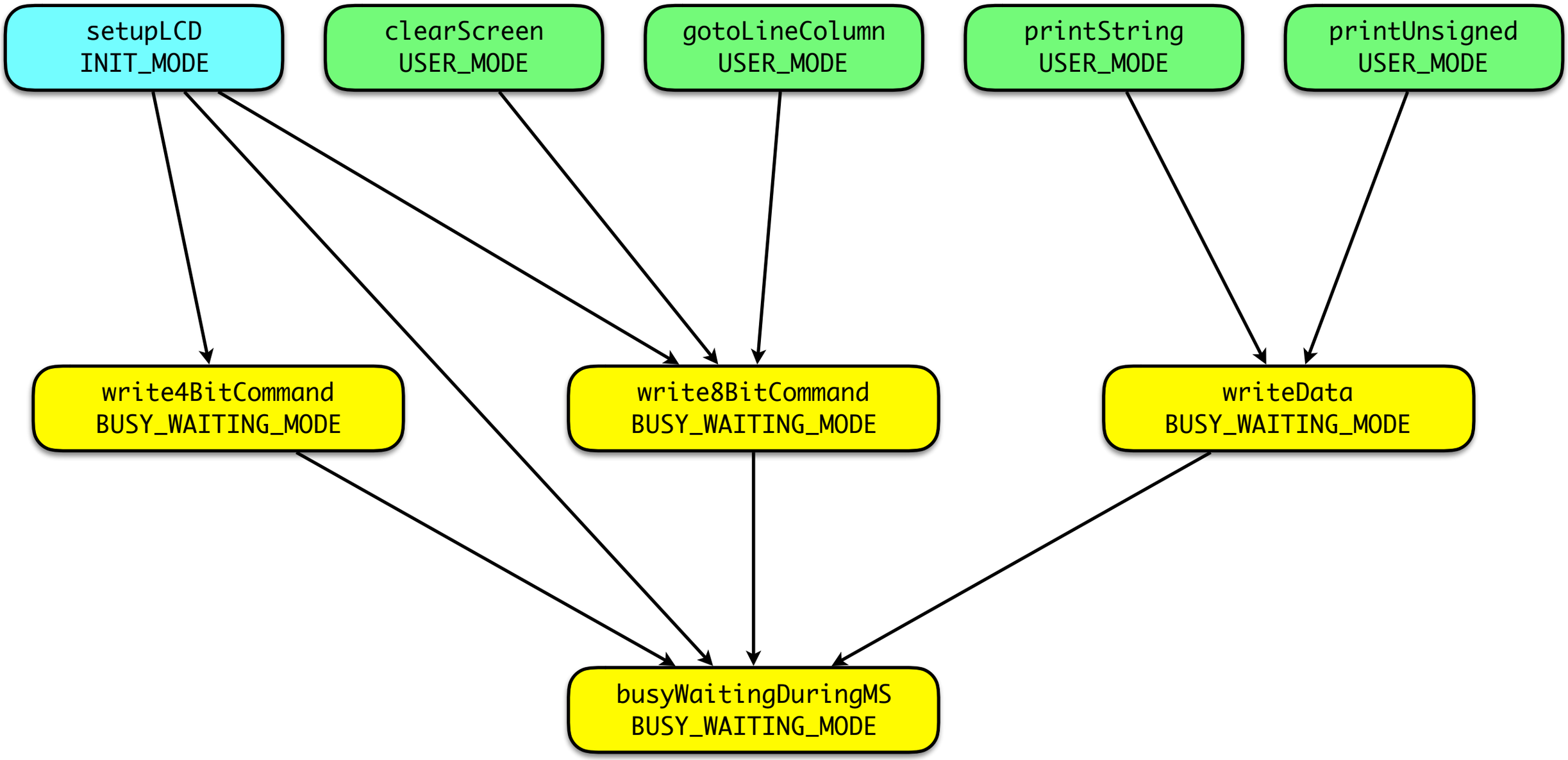
Astuce : utilisez `1-verbose-build.py` pour compiler un seul fichier à la fois et avoir des messages d'erreurs plus clairs.

Les messages d'erreur concernent `BUSY_WAITING_MODE` qui n'existe plus.

Pour comprendre les modifications à apporter, nous allons établir un graphe partiel des appels de fonctions dans `lcd.c`.

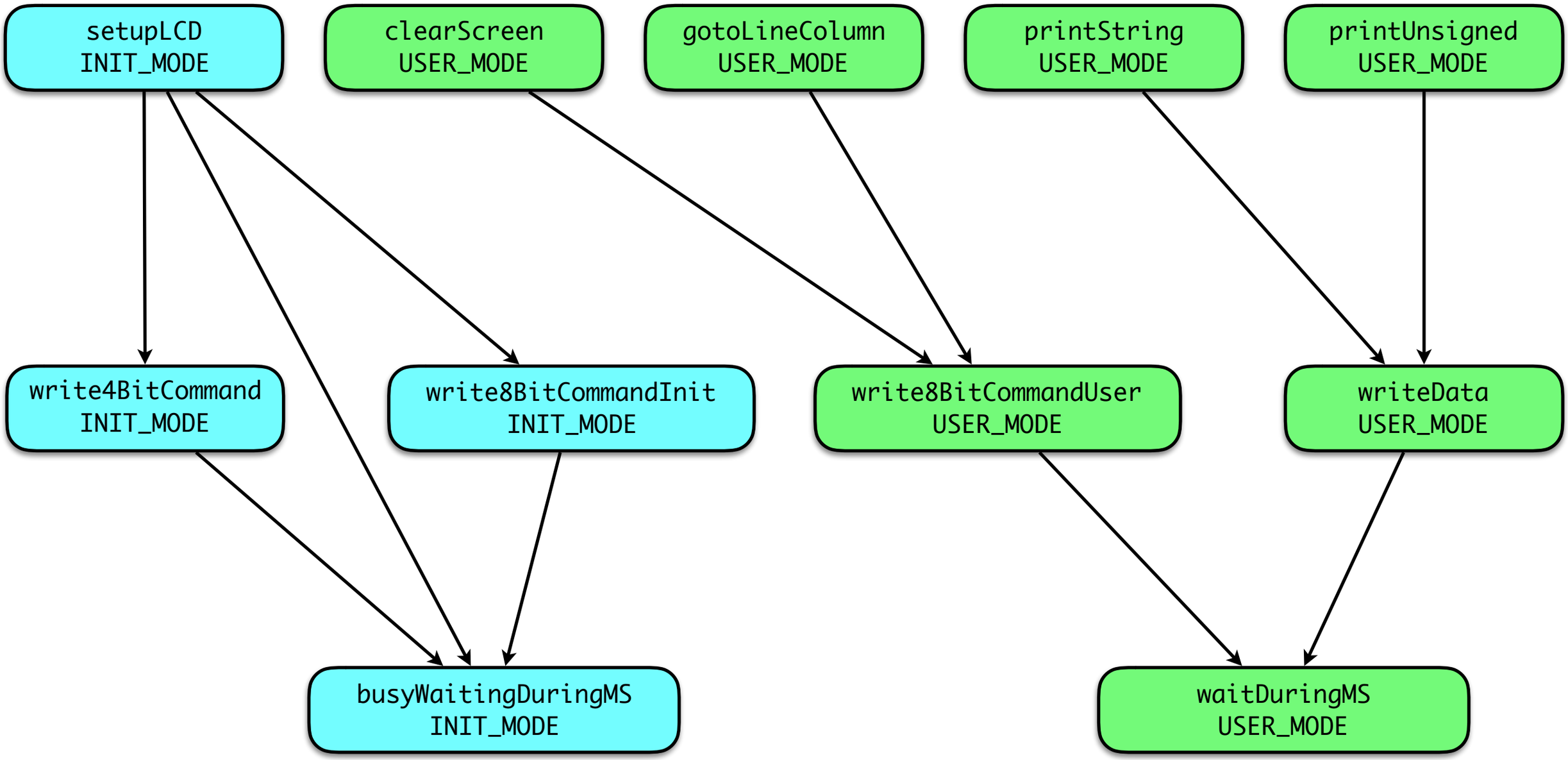
Graphe d'appel partiel des fonctions de `lcd.c`

Situation avant modification



Graphe d'appel partiel des fonctions de `lcd.c`

Situation après modification

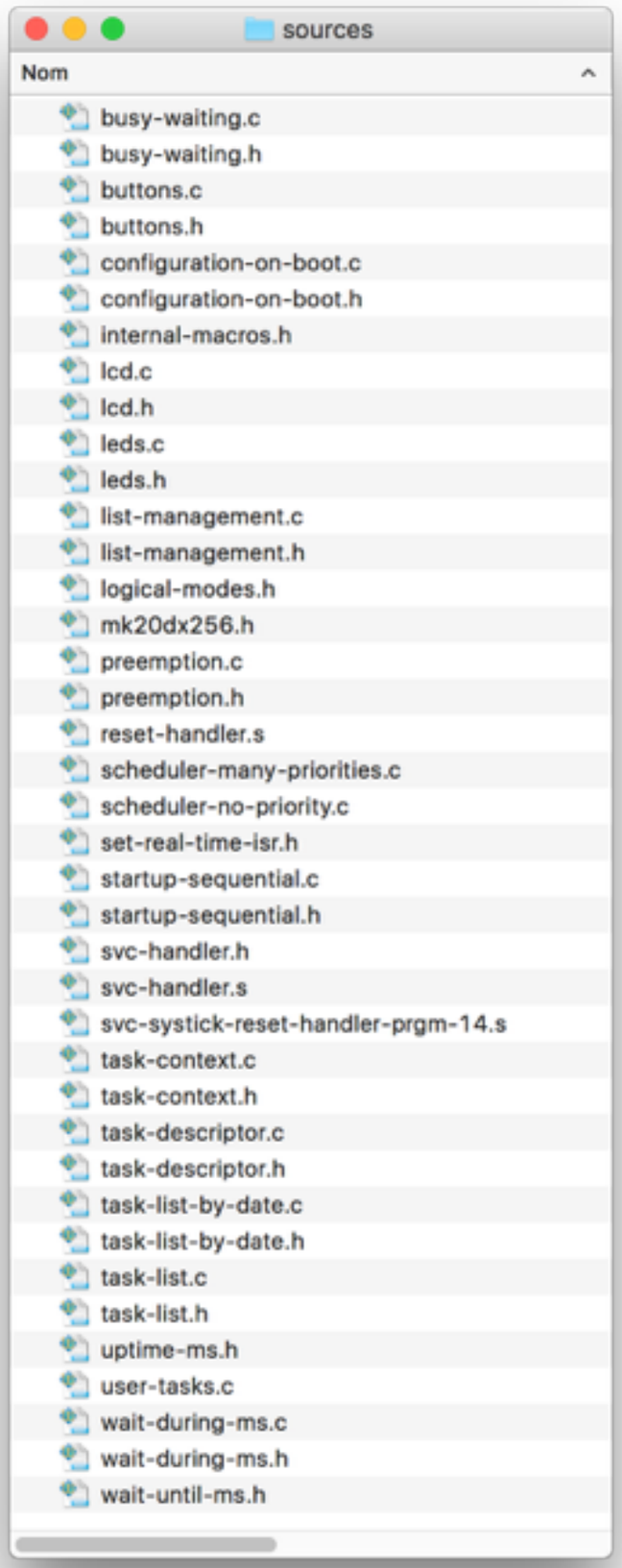


Le `BUSY_WAITING` mode a disparu, la fonction `write8BitCommand` doit être dupliquée et adaptée à chaque mode.

Travail à faire

- faire les modifications décrites dans les pages précédentes ;
- il n'y a pas de fichier source à ajouter ni à retirer par rapport à l'étape précédente.

Rappel : la led *Teensy* est contrôlée par l'exécutif, de façon à montrer la charge du processeur.



Résultat attendu

Maintenant, la led *Teensy* doit toujours s'éclairer faiblement, même durant l'usage de l'afficheur LCD par une tâche. Au démarrage, la led *Teensy* s'éclaire fortement, l'initialisation utilise le processeur à temps plein (à cause de l'appel de `busyWaitingDuringMS`).

Rappel : depuis l'ajout de la préemption sur interruption temps-réel, au plus une tâche a droit d'utiliser l'afficheur LCD.