Compilation du firmware interrogateurs sans fil

20 septembre 2013, J.-M Friedt

1 Versions de lecteurs

Le logiciel embarqué est désormais compatible avec 3 types d'interrogateurs afin de mutualiser la partie algorithmique du code (`DDS.c`) et de ne faire appel qu'à quelques fonctions spécfiques au matériel de chaque plateforme matérielle :

1. les lecteurs classique et Schrader exploitent le microcontrôleur ADuC7026 et le DDS AD9954. Une variante est la PLL du lecteur classique qui a été modifiée en novembre 2012 mais n'impose qu'une option additionnelle,
2. le lecteur gilloux basé sur un DDS AD9958 et un microcontrôleur STM32 Cortex M3,
3. le lecteur low cost basé sur un STM32 Cortex M3 et un radiomodem Semtech XE1203F

2 Variables d'environnement et outils de compilation

Il est peu judicieux de déplacer des versions obsolètes de `libstm32` avec les divers codes sources de logiciels embarqués. Le choix a été fait, en vue de permettre l'utilisation de fonctions gourmandes en ressources que sont l'allocation dynamique de mémoire (`malloc()`) et le calcul en virgule flottante (`double`), de se lier avec la dernière version de `libstm32` disponible sur `summon-arm-toolchain`. Pour ce faire, les variables d'environnement suivantes sont nécessaires

```
export CORTEX_BASE_DIR=${HOME}/sat/
export STM32_INC=${CORTEX_BASE_DIR}/include
export STM32_LIB=${CORTEX_BASE_DIR}/lib
export PATH=${HOME}/sat/bin:$PATH
```

3 `Makefile`

D'un point de vue compilateur, le choix de la chaîne de compilation et des options pour le processeur associé sont définis dans `Makefile`. L'utilisateur n'a, *a priori*, jamais à toucher aux configurations de compilation incluses (`makefile.*`). Trois options sont disponibles dans `Makefile` pour les trois architectures citées ci-dessus :

1. `include makefile.aduc` pour les lecteurs classique et Schrader,
2. `include makefile.cortex_AD9958` pour le lecteur gilloux,
3. `include makefile.cortex_XE1203` pour le lecteur low cost,
4. `include makefile.pc` est une option additionnelle qui permet de tester sur PC (ie sans matériel embarqué dédié) les algorithmes de commande. L'intérêt de cette approche est de pouvoir synthétiser des signaux respectant certaines statistiques de bruit et de valider par des tests automatiques le fonctionnement de la partie algorithmique du firmware (`DDS.c`). Ces signaux sont synthétisés dans `dummy_pc.c`, dans la fonction interroge qui renvoie un puissance fonction de la fréquence sondée (faisant appel aux registres FTW0, cette méthode de développement n'est pas fonctionnelle pour le moment pour le low cost et n'a été testée que pour le lecteur classique). Le binaire résultant, `interrogateur`, est en général exécuté dans la console en mode texte, l'affichage dans un `xterm` induisant un saut de ligne entre les trames.

Une seule de ces options ne peut être active lors d'une compilation donnée, les autres options doivent être commentées en précédent la ligne d'un symbole #.

Par ailleurs, deux variables de compilation sont définies (=1) ou non (=0) dans le `Makefile` pour activer l'affichage sur écran LCD (uniquement testé sur lecteur Schrader) ou pour activer la communication sur bus MOSBUS (uniquement testé sur interrogateurs classique et Schrader pour une plateforme autour de l'ADuC7026 – le portage au Cortex M3 n'a pas été validée dans tous les cas et notamment si le lecteur est serveur). Ces options sont par défaut `MODBUS = 0` et `LCD = 0`.

4 Configurations

La configuration logicielle doit refléter les besoins du matériel. Nous déclinons de nouveau les options spécifiques à chaque matériel dans les options proposées dans le fichier `DDSinc.h` qui dépend des architectures :

1. pour les lecteurs classique et Schrader, le cœeur du DDS est cadencé à 200 MHz (`#undef f400`), la PLL est sélectionnée en fonction de la date de fabrication du lecteur (`#undef ADF4111` pour pré-Novembre 2012, `#define ADF4111` sinon) et le DDS est un AD9954 (donc `#undef AD9958`),
2. pour le lecteur gilloux, le cœeur du DDS est cadencé à 360 MHz (`#define f400`), il n'y a pas de PLL donc l'option `ADF4111` est sans importance, et le DDS est un AD9958 (donc `#define AD9958`),
3. pour le low cost, malgré l'absence de DDS nous considérons qu'il est peu judicieux d'activer l'option `AD9958` et dans le doute afficherons un message d'erreur si cette configuration est active. Par ailleurs, la fréquence est supposée définie comme si une horloge cadençant le DDS à 200 MHz existait, donc il faut `#undef f400`.

Le fichier `DDSinc.h` définit les options de compilation tandis que les variables susceptibles d'être modifiées par l'utilisateur (dans la fonction `communication()` de `DDS.c`) sont initialisées avec des valeurs par défaut dans `DDSvar.c`. On y trouvera donc les bornes de fréquences, nombre de résonances et nombre de points de mesure, configuration des fréquences par intervalles de tailles égales ou manuels ...

On préfèrera désactiver la définition d'une constante en la commentant plutôt que par l'utilisation de `#undef` puisque certaines options de compilation sont passées depuis la ligne de commande au compilateur par l'option `-D` et risquent d'être désactivées si un `#undef` est utilisé.

5 Simulateur sur PC

Le simulateur d'interrogateur sur PC est un outil pour déverminer les problemes d'affichage (dépassement de capacité) et d'immunité au bruit. Le fichier `dummy_pc.c` contient les fonctions qui émulent les réponses des composants matériels des interrogateurs. En tant que tel, un double-résonateur est simulé (sous forme de réponse somme de deux gaussiennes) dans la fontion `interroge()` qui renvoie une information de puissance en fonction de la fréquence émise par le DDS. Cet outil semble particulièrement approprié pour simuler une mesure bruitée, pour le moment avec une contribution de bruit blanc sur l'amplitude mais qui pourrait être enrichie de diverses statistiques de bruit si besoin est.

Le simulateur doit être exécuté depuis une console en mode texte sous GNU/Linux puisque la gestion du clavier ne semble pas appropriée pour l'environnement graphique X11.