1. IMPORTARE I PROGETTI

Per importarli selezionare da menù di CODE COMPOSER Project -> Import CCS Project -> Select search-directory

1. INFO IMPORTANTI PROGETTI

I progetti sono completamente funzionanti.

* I progetti dual core richiedono solo di rimappare gli include a seconda del vostro path di installazione delle librerie
* Il progetto Sd\_card richiede la stessa procedura ed inoltre richiede una libreria aggiuntiva ff14b che potete trovare al link <http://elm-chan.org/fsw/ff/00index_e.html>

Di comune accordo è stato deciso che la libreria si deve trovare al path C:\ti\ff14b.

L’include nel progetto deve essere il seguente C:\ti\ff14b\source

* Il progetto sd\_card è stato fortemente modificato rispetto all’originale, chi è interessato alle modifiche può dare un’occhiata alla guida presente nella repository.

1. STATO DEI LAVORI

* Can, ADC ed EPWM correttamente funzionanti su cpu2 (già testati individualmente e in coppia)
* Sd\_card (Samsung) scrive tutti i dati presenti in memoria condivisa in circa 45 ms.
* Timestamp implementato grazie ad un timer presente in cpu2 che condivide una variabile alla cpu1.
* MANCA LoRa (da aggiungere dopo i test).
* MANCA Testare Schermo con Can
* MANCA integrare launch control

1. CONSIDERAZIONI
   * RICORDARSI SEMPRE CHE FUNZIONA SOLO IN FLASH
   * Le prestazioni della sd\_card sono estremamente legate alla qualità della scheda stessa, perciò provare sempre almeno due schede (di qualità diversa) prima di fare valutazioni.
   * Il codice che implementa la scrittura in sd non è sicuramente dei migliori, ma è l’unico presente nelle librerie. Per migliorare le prestazioni si deve scrivere i dati in blocchi più grandi possibile.
   * Vi è un limite di dati che si possono passare al comando writeSD() circa 250 byte (vedere implementazione della funzione). Sarebbe sempre meglio controllare a banco che la sd non si blocchi in caso di scrittura di troppi dati.
   * La logica del vecchio codice non è stata modificata se non per gli ADC che ora vengono letti tutti in blocco all’inizio del timer principale della cpu2.
   * Purtroppo la mole del lavoro mi ha portato a fare tante prove quindi ci possono essere delle variabili di debug “abbandonate” nel codice, se le trovate segnalatele con un commento o ditemelo
2. STRUTTURA DEL DUAL CORE

CPU 1

- Sd\_Card

- LoRa (telemetria)

- Nextion (schermo)

SHARED RAM

- Shared struct sh

- timestamp

CPU 2

- ADC

- EPWM

- Main Logic

- Timers

1. ASPETTI TECNICI e CORREZIONE FILE DI LIBRERIA

Durante il lavoro di scrittura codice in dual core sono state apportate le seguenti modifiche (spero di ricordarmele tutte, se ne avete qualcuna in mente vi prego di segnarla qui o parlarmene):

* Il progetto è stato compilato con la versione 20.2.1 del compilatore; in caso di problemi usare questa versione prima di modificare altri parametri.
* Al seguente path C:\ti\c2000\C2000Ware\_3\_04\_00\_00\device\_support\f2837xd\common\cmd

Il file 2837xD\_FLASH\_lnk\_cpu1 ha una zona di codice SECTION modificata nella seguente maniera:

/\* Allocate program areas: \*/

.cinit : > FLASHB PAGE = 0, ALIGN(8)

.text : > FLASHG PAGE = 0, ALIGN(8)

codestart : > BEGIN PAGE = 0, ALIGN(8)

/\* Allocate uninitalized data sections: \*/

.stack : > RAMLS1 PAGE = 0

.switch : > FLASHB PAGE = 0, ALIGN(8)

.reset : > RESET, PAGE = 0, TYPE = DSECT /\* not used, \*/

Il file 2837xD\_FLASH\_lnk\_cpu2 ha una zona di codice SECTION modificata nella seguente maniera:

/\* Allocate program areas: \*/

.cinit : > FLASHB PAGE = 0, ALIGN(8)

.text : > FLASHG PAGE = 0, ALIGN(8)

codestart : > BEGIN PAGE = 0, ALIGN(8)

Il motivo della modifica del .text è che il programma non sarebbe entrato in memoria. Il motivo della modifica del .stack è che serve dello spazio aggiuntivo per far funzionare il sprintf().

COME SEMPRE SONO DISPONIBILE PER QUALSIASI CHIARIMENTO.

TEST SUL CODICE SONO ASSOLUTAMENTE BEN ACCETTI E NECESSARI, SE LI FATE FATEMI SAPERE L’ESITO ED EVENTUALI PROBLEMI.