Documentazione Progetto emphiOS

Marco ing. Dau

04/05/2017

marco.ing.dau@gmail.com

Indice

Parte I

Premessa

Capitolo 1

Target

1.1 A chi è Rivolto

Il presente lavoro (codice sorgente e documentazione) è rivolto principalmente a programmatori in linguaggio C di microcontrollori ad 8bit.

1.2 Tipologia di applicazioni

Il firmware a cui la presente documentazione si riferisce è stato sviluppato pensando ad applicazioni PLC, sia per la parte funzionale che per quella interfaccia; le applicazioni PLC sono quelle applicazioni in cui devono essere gestite le logiche che legano gli ingressi alle uscite; gli ingressi e le uscite possono essere sia digitali che analogici.

1.3 Avvertenze

Nella presente documentazione ci sono ancora molte inesattezze (dovuti in parte anche alla difficoltosa sincronizzazione con il codice) e nel firmware aspetti da completare ma il codice sviluppato con le sue regole e i suoi modelli implementati, e qui descritti, già permette uno sviluppo modulare di una applicazione per microcontrollori 8-bit ed era questo l'obiettivo base di questo lavoro. Alcuni concetti o termini di informatica specie nell'ambito della programmazione ad oggetti nel tentativo di portare questa tecnica allo sviluppo di applicazioni per microcontrollori 8-bit usando il linguaggio C sono stati, forse, un po' abusati, quindi se qualcuno ritene che siano stati usati impropriamente è pregato di farmelo notare (la mia mail è "md120308 at gmail dot com"), ringrazio in anticipo tutti coloro che mi faranno questa cortesia. Nel caso qualche lettore gradisca un maggiore dettaglio di alcune parti della presente documentazione sarò ben lieto di esaudirne le richieste, stessa cosa per il firmware. In ultimo desidero preci-

sare che tutte le critiche positive e negative sulla presente documentazione e in generale sull'intero lavoro (firmware e documentazione) saranno bene accette.

Capitolo 2

Introduzione

2.1 Caratteristiche

Il codice è stato sviluppato pensando alle seguenti caratteristiche

Ottimizzazione del codice: minimizzare l'utilizzo di memoria ROM e RAM

Modularità: sviluppare nuove funzionalità senza dover modificare il codice di altri moduli.

2.2 Modularità e suoi vantaggi

La seconda caratteristica ha numerosi vantaggi, ben noti a coloro che usano il paradigma della programmazione orientata agli oggetti:

- Facilita il debug e quindi in generale anche la manutenzione del codice
- Facilita l'introduzione di nuove funzionalità minimizzando problemi di interferenza con altre parti del codice
- Permette un più veloce e "pulito" riutilizzo del codice in altri progetti anche con differenti piattaforme HW e/o SW
- Permette di identificare chiaramente le parti del codice indipendenti dalla piattaforma su cui far girare il fw (notare che la piattaforma può essere HW o SW)
- Possibilità di sviluppare progetti coinvolgendo più persone o gruppi e in ultima istanza creare in rete una repository di moduli e configurazioni fw/hw

2.3 Schemi e moduli ricorrenti in applicazioni PLC

In applicazioni PLC (ma non solo!) ci sono schemi e moduli spesso ricorrenti, di seguito i principali che sono stati implementati nel codice sorgente a cui la presente documentazione si riferisce:

- Schemi
  - produttore-consumatore: controllo del verificarsi di particolari situazioni o condizioni, chiamate anche eventi (Produzione evento), e successiva esecuzione di azioni associate a tali eventi (Consumazione evento)
  - timers di sistema: quasi sempre i moduli hanno bisogno di eseguire delle procedure a particolari intervalli di tempo; ad esempio gestione dell'anti-rimbalzo nella lettura dei tasti oppure la gestione del lampeggio nei moduli interfaccia utente
  - funzioni di init: quasi sempre i moduli hanno bisogno di una inizializzazione che deve essere eseguita alla partenza del microcontrollore
- Moduli
  - gestione interfaccia utente (componenti base: lista di voci, modifica valori)
  - gestione parametri
  - multilingua

2.4 Descrizione piattaforme usate

l codice sorgente a cui è associata la presente documentazione è stato implementato su due diverse piattaforme una HW ed una SW

- piattaforma HW: EasyPicV7 di MikroElektronica
- piattaforma SW: Windows7 Microsoft

Si vuole far notare che ci sono dei vantaggi nel poter eseguire il fw su piattaforme SW come Windows, Linux oppure OS X

- allestire debug automatici del fw (naturalmente sono esclusi i driver, questi possono essere testati solo sulle piattaforme per cui sono scritti)
- rendere il progetto FW (ad esclusione dei driver) indipendente dal progetto HW a cui è associato

Parte II

Descrizione del codice

Capitolo 3

Files e Nomenclatura

Tutti i nomi dei file eccetto il file main.c rispettano la seguente codifica:

[hw_|ui_][kr|ui|fn|rs2|ms]<Nome del modulo>[Def|Tbl|Glo|Edt].[c|h]

3.1 Prefissi

Notare che ci sono due livelli di prefissi la prima termina con un underscore "_"; di seguito il significato e l'utilizzo di questi prefissi

3.1.1 Prefissi di primo livello

hw_: evidenziare un file in cui il codice gestisce istruzioni dipendenti dalla piattaforma, sono i file che costituiscono i driver del sistema, tipicamente fanno parte dei moduli delle risorse (moduli rs2)

ui_: evidenziare il file header che contengono variabili globali di un modulo che si desidera siano presenti nella lista dei parametri che si possono modificare via interfaccia utente; i file con questo primo livello di prefisso sono sempre i file con postfisso "Edt" e quindi sono sempre file header

3.1.2 Prefissi di secondo livello

Questo livello di prefisso identifica anche il gruppo di moduli. Di seguito la loro descrizione:

kr: (sta per kernel) sono i moduli base di un progetto che non possono mai mancare; ci sono due moduli principali e ciascuno offre dei "servizi" agli altri moduli del firmware:

krProcess: è lo schedulatore dei processi; nel codice i processi sono rappresentati dalle righe della matrice dei processi e sono caratterizzati da una funzione che genera eventi (Produce un evento) e da una funzione che processa gli eventi (Consuma un evento)

krTimer: gestisce le funzioni evento associate alle variabili timer di cui necessitano i vari moduli del firmware

ui: (sta per user interface) sono i moduli che si occupano di gestire l'interfaccia utente;

fn: (sta per funzionale) sono i moduli funzionali dove è presente il codice che gestisce la logica tra gli ingressi e le uscite;

rs2: (sta per RiSorse, togliendo le vocali si ha rsrs da cui rs2) sono i moduli che forniscono agli altri moduli l'interfaccia con l'hardware, tipicamente al proprio interno contengo i driver del sistema

ms: (sta per miscellaneous), sono moduli generici che non apparetngono a nessuna delle precedenti categorie

3.2 Postfissi

Def: (è un postfisso e sta per Definition) sono quei file in cui sono "fisicamente" definite le variabili globali dei moduli; questi file sono sempre con estensione ".c" non hanno il nome dei moduli ma solo il prefisso di secondo livello.

Tbl: (è un postfisso e sta per Table) sono quei file in cui sono defnite Tabelle statiche necessarie per la gestione delle dipendenze indirette come ad esempio le funzioni timer dei moduli o gli eventi provenienti dai tasti, per maggiori dettagli si veda il paragrafo <2.4.1.1 Dipendenze Dirette e Indirette>;

Glo: (è un postfisso e sta per Global) è un file header in cui sono presenti strutture e variabili condivisi da più moduli, di solito quando venfgono condivise le strutture e/o le variabili c'è sempre un modulo di riferimento da cui il file prende anche il nome

Edt: (è un postfisso e sta per Editable) sono quei file in cui sono definite le variabili globali di un modulo con le loro proprietà di visualizzazione e di modifica che si vuole poter modificare via interfaccia utente.

Parte III

Possibili sviluppi

Capitolo 4

Progetti e sviluppi futuri

Possibili sviluppi futuri del presente lavoro vertono su tre linee, una rivolta all'ambiente di sviluppo l'altra rivolta alla riutilizzabilità e condivisione del codice e l'ultima relativa ad un progetto PLC openSource:

- permettere lo sviluppo rapido di appicazioni per microcontrollori 8-bit, allo stesso modo delle applicazioni Windows/OS X/Linux
  - creare interfaccia utente con tool grafici senza modificare manualmente il codice nello stile degli ambienti di sviluppo come Delphi, QtCreator / QtDesign, VisualStudio oppure degli ambienti di sviluppo PLC per la programmazione delle interfacce utente
- creare in rete una repository per la condivisione di moduli relativi alle più disparate applicazioni e necessità, e driver relativi alle più svariate configurazioni di "microcontrollore + piattaforma"; per quest'ultimo aspetto già si può trovare qualcosa in rete
- sviluppare applicazioni nello stile del linguaggio LADDER ma sfruttando anche moduli più complessi che permettano di uscire dal paradigma procedurale che quando l'applicazione è leggermente più articolata crea difficoltà di manutenzione e debug; si veda in rete il progetto OpenPLC (progetto open) e CodeFlow (progetto proprietario) Notare che i primi due punti sono tra loro strettamente legati.

Capitolo 5

Progetti simili o complementari

FlowCode
- FemtoOS
- OpenPLC
- Librerie per vari microcontrollori per la gestione dei registri
- EICASLAB