SUPSI

Dipartimento Tecnologie Innovative

Laboratorio di Sistemi Meccatronici

ISEA

Prof. Roberto Bucher

Laboratorio didattico

di Meccatronica

Utilizzo del laboratorio di Meccatronica

– Linux con Preempt RT –

Manno, 03.07.2017

Table of Contents

List of Figures

List of Tables

Appendices

List of Abbreviations and Symbols

Introduzione

Il laboratorio didattico di meccatronica è stato completamente ristrutturato sostituendo il sistema basato su Linux RTAI preesistente con un nuovo sistema basato su Linux con RT-PREEMPT.

Struttura del laboratorio

1 Sistema operativo

I PC del laboratorio sono stati riinstallati con una versione Debia Stretch (testing) con un kernel basato sulle patch di rt-preempt. Questo dovrebbe permettere un mantenimento più smplice di tutta la struttura attuale, con possibilità di modifiche e update da parte di tutti.

2 Preparazione dell' istallazione

Per fare l'istallazione occorrono:

- Una chiavetta USB con l'ultima versione “testing” scaricabile dal sito ufficiale di Debian
- Una serie di files supplementari per l'istallazione delle parti di sviluppo, tutti scaricabili da XXXXX
  - Makefile per istallare le parti supplementari
  - Matlab.tgz (Con matlab e addons)
  - Un template Simulink (rtpreempt.sltx) per la compilazione dei nuovi eseguibili
  - Un file con i driver “peaks” per il dongle CAN
  - due cartelle (jmodelica e pycontrol) con i files per l'istallazione del sistema basato su python

3 Istallazione

Dopo aver istallato tutto il sistema operativo partendo dall'immagine scaricata da Debian, occorre istallare tutti i files supplementari. Questo lavoro può essere fatto in forma più semplice utilizzando il Makefile disponibile sulla chiavetta.
Il Makefile va leggermente modificato con il passare del tempo per essere sempre attuale con alcuni files, in particolare il kernel da utilizzare.

Da shell:

- Determinare la versione istallata del kernel tramite il comando “uname -r”
- Cercare il corrispondente kernel con estensione “rt”
  apt-cache search linux-image

La parte del Makefile sotto “kernel” va modificata nel modo seguente:

- Modificare i nomi dei files con la versione corretta attuale del kernel (solo prima e seconda riga degli apt-get install!)

A questo punto è sufficiente entrare come amministratore (root) tramite il comand “su” e lanciare:

- make first

Ad un certo punto il sistema esegue un reboot

4 Seconda parte dell'istallazione

A questo punto è possibile istallare la seconda parte di files, da shell, tramite:

- sudo su (+pw)
  make second

Per terminare tutta l'istallazione ci sono ancora due passi

- Andare nella cartella /usr/local/pycontrol e lanciare come utente normale “make user”
- Andare nella cartella Documents e creare (se necessario) la cartella “MATLAB”: copiare in questa cartella il file “rtpreempt.sltx” presente sulla chiavetta.

Utilizzo del generatore di codice di Matlab

1 Operazioni di base

Utilizzare usualmente Matlab per le simulazioni. Per la generazione di codice occorre utilizzare il template “rtpreempt”.

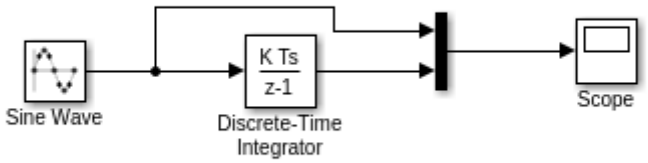
1. Lanciare Simulink
2. Dalla “Simulink Start Page” scegliere il template “ERT Linux” da “My Templates”: in questo modo si hanno le modifiche per il generatore già configurate correttamente per generare il codice per il target richiesto.

3. A questo punto si possono utilizzare blocchi di Simulink e propri per il design del sistema per la generazione di codice. È possibile utilizzare gli Scope classici di Simulink.
4. “Ctrl-B” genera il codice necessario per il target Linux RT-PREEMPT

A questo punto è sufficiente mandare in esecuzione il codice da una shell, quindi dallo schema di Simulink si apre il menu “Code” “External Mode Control Panel” e ci si connette (bottone “connect”) con il target in esecuzione per visualizzare i dati in tempo reale sullo scope di Simulink.

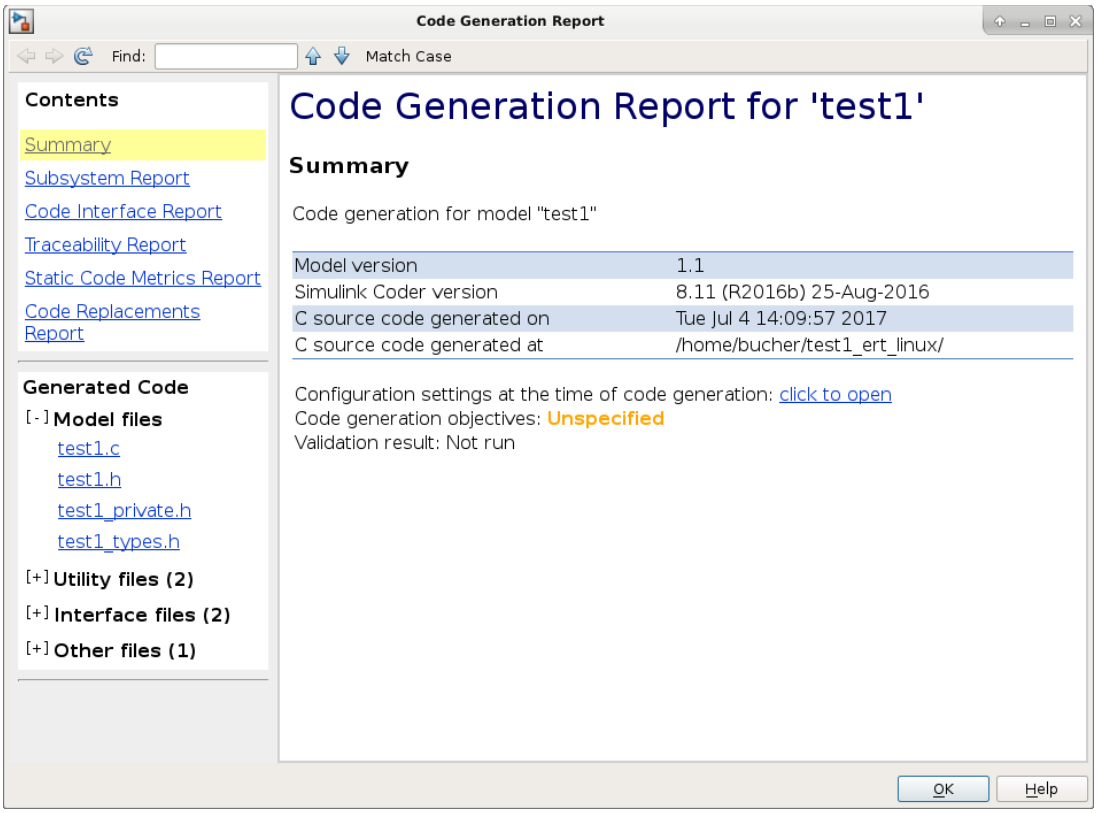
2 Esempio

La prossima figura mostra un semplice esempio di schema simulink con un segnale sinusoidale in input, un integratore discreto e uno scope. Si apre Simulink e si sceglie il template “ERT Linux”. Viene salvato come “test1.slx”. Nello schema sostituire il tempo finale (10") con “inf”



Il tempo di sampling è fissato a 1ms.

A questo punto è possibile generare il codice (già predisposto per RT Linux) con un semplice “Ctrl-B”. Se la compilazione ha successo compare una finestra con il report della compilazione, altrimenti viene indicato un errore.



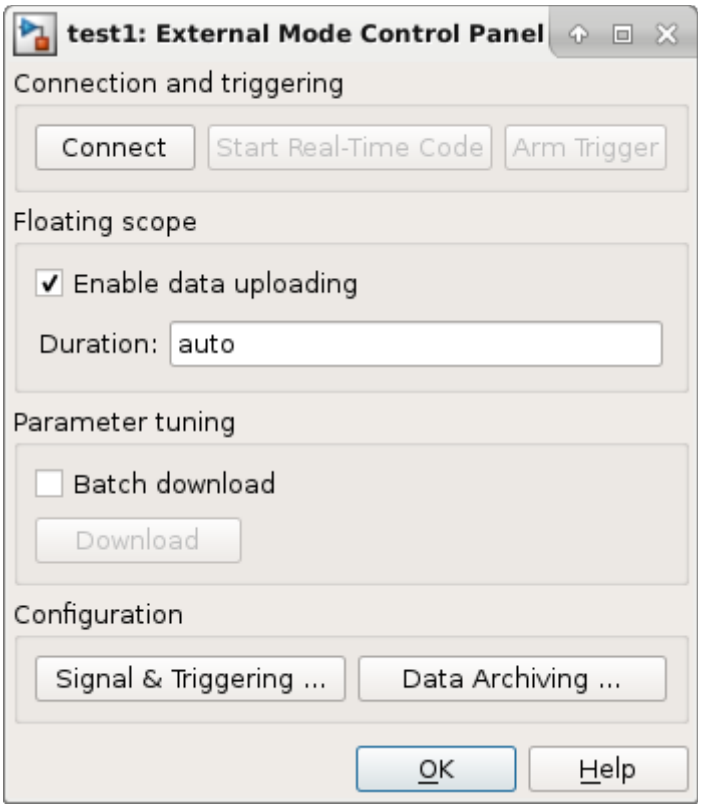
A questo punto occorre aprire una shell, posizionarsi nella cartella dove si è generato il codice e lanciarlo con il comando

"sudo ./test1"

È possibile anche lanciare il comando di esecuzione senza "sudo" caricando prima il dribe "nrt" con il comando "loadnrt"[1]

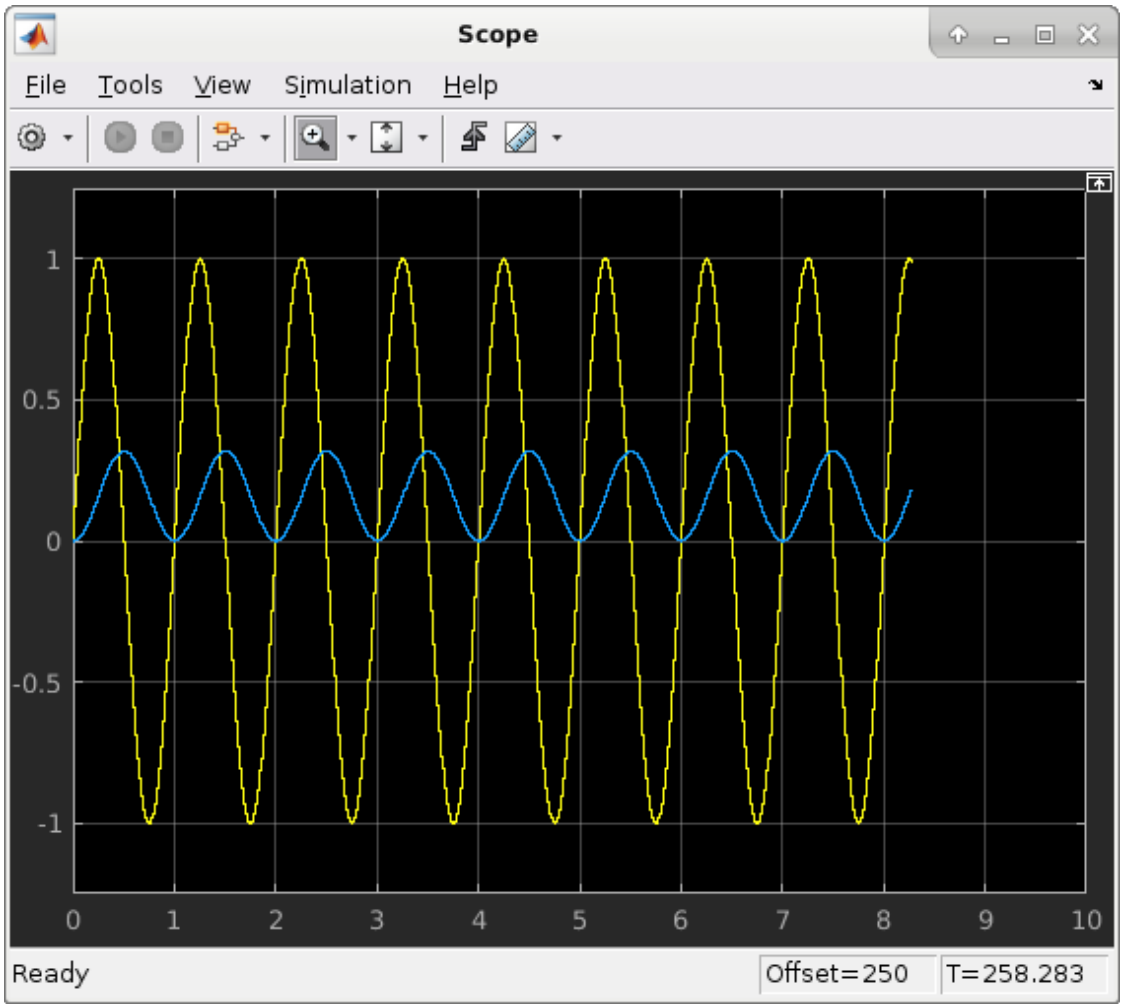
Nel menu dello schema di Simulink scegliere "Code" → "External mode Control Panel" e si apre una nuova finestra

[1] Questa opzione deve ancora essere testata sui vari PC e al momento non è disponibile sui CPCI



Scegliere il bottone “Connect” per connettere lo schema Simulink con il target RT in esecuzione.

Aprendo ora lo “Scope” di Simulink si possono visualizzare I due segnali in tempo reale.



4

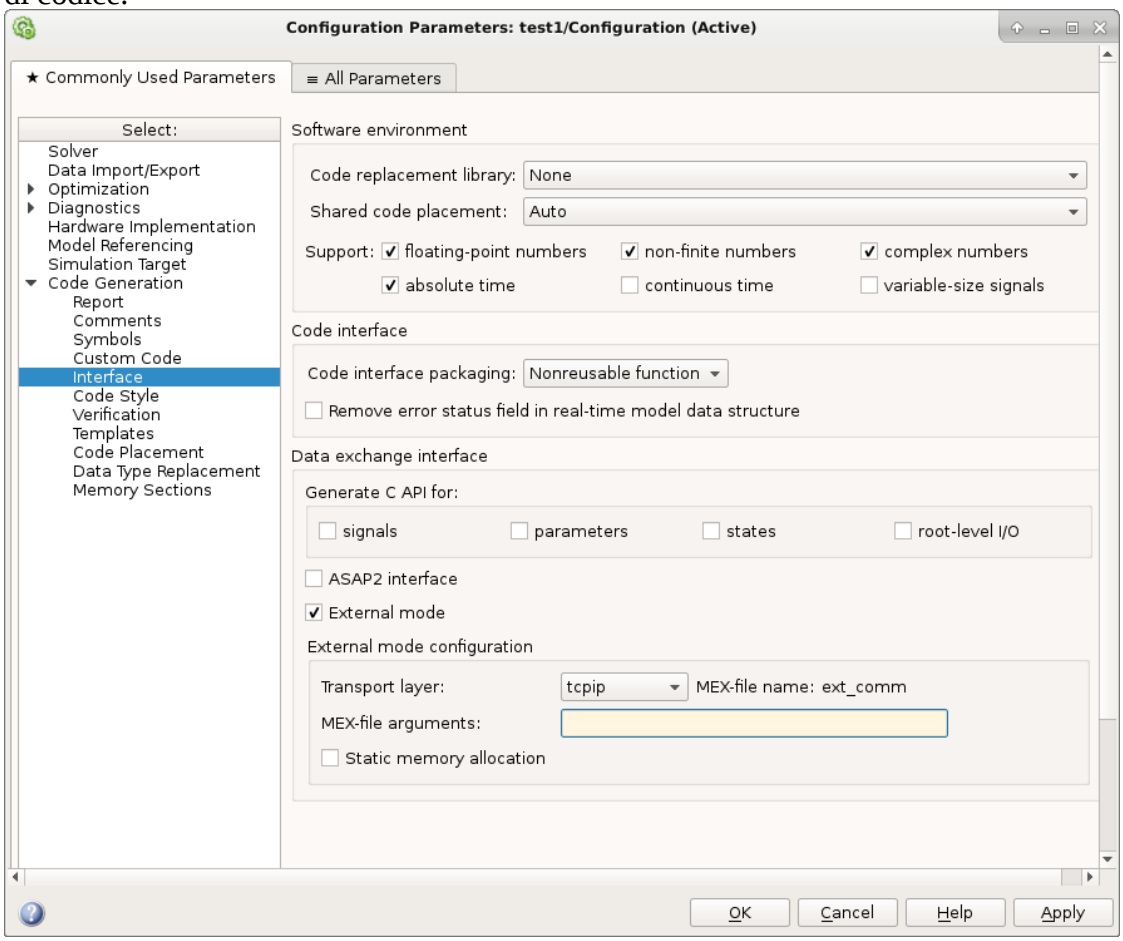
Generazione di codice per Compact PCI

1 Opzioni, generazione di codice e monitoraggio

Utilizzando il Compact PCI, subentra un problema supplementare dovuto al fatto che il sistema operativo sul CPCI è a 32 bit, a differenza di quello sul PC di sviluppo che è a 64 bit. Pertanto occorre effettuare una compilazione per un sistema a 32 bit.

Per generare correttamente il codice è sufficiente aprire le opzione della generazione di codice (Code → C/C++ Code → Code Generation Options…) e scegliere come target RT “ert_linux_CPCI.tlc” (Linux Embedded coder for compact PCI). Il Makefile genera a questo punto un eseguibile a 32 bit da scaricare poi sul target.

Per connettersi con gli scopes occorre aprire nuovamente le opzioni del generatore di codice.



Sotto “Interface” se necessario vistare “External mode” e nei “MEX-file arguments” inserire l' IP del CPCI target. Questo permette, una volta lanciato l'eseguibile RT sul CPCI, di connettersi dagli scope di Simulink locali.