NUMERO : 303 DATA : 14/06/2012

A P P U N T I

STUDENTE : Diana

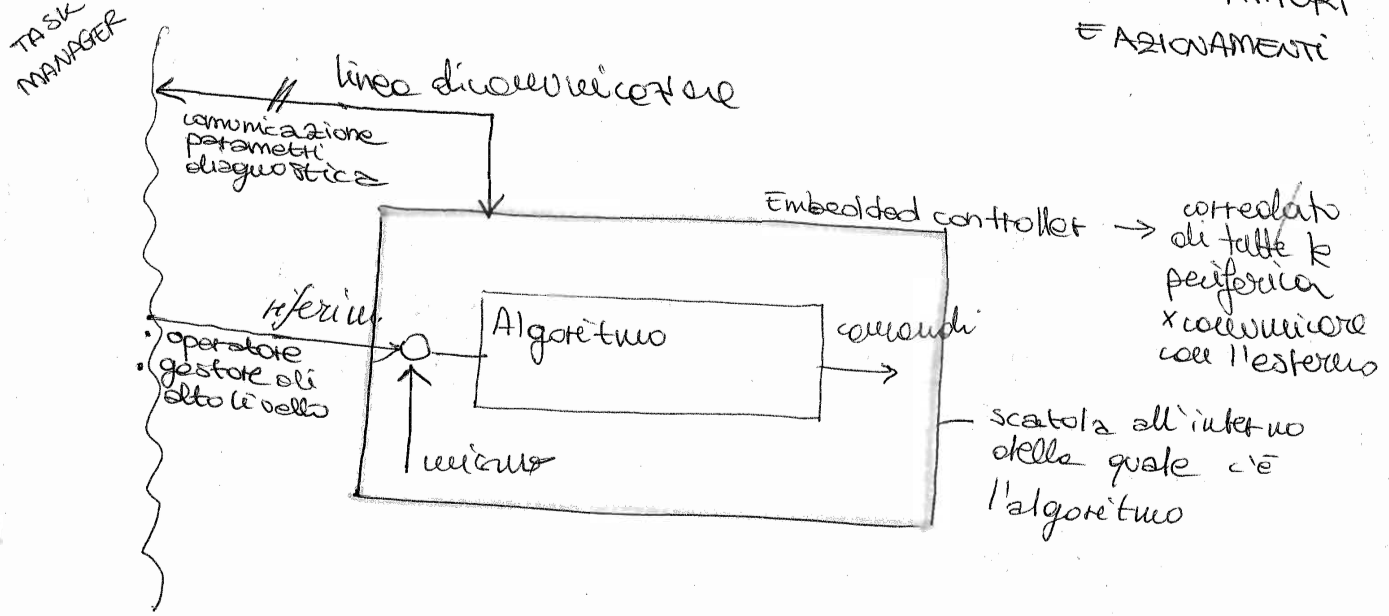
MATERIA : Controllo Digitale di Convertitori ed Azionamenti

Prof. Pellegrino

INTRODUZIONE

5/3/12 —

CONTROLLO DIGITALE DI CONVERTITORI E AZIONAMENTI



L'algoritmo di controllo digitale sta dentro la schede embedded controllet

È come se si dividesse all'interno delle scatole ~~scatole~~ quello di controllo

Si hanno

- comandi, sono del tipo
  - PWM → oggetto di modulatore switching (è la periferica di uscita)

misure possono essere

- analogiche (i, v, ω)
- digitali (sensore ad effetto Hall brushless trapezio, encoder ottico ecc....)

Riferimenti

- sono digitali e vengono o da
  - operatore (es. potenziometro

Il problema da risolvere è realizzare

CONTROLLO DI GITTATA DI PROCESSI A TEMPO CONTINUO

Preso un azionamento che lavora nel mondo analogico
ci si deve interfacciare con il mondo a tempo discreto

Per far comunicare le 2 cose serve un ESECUZIONE IN REAL TIME (es: Pspice con lo è)

↓

Se devo controllare un motore
lo devo controllare in tempo
reale campionando le misure
ogni ms ed eseguire i comandi in
tempo reale

↓ un TIMER che scandisce il tempo e il processore
deve eseguire i calcoli necessari nel tempo messi a disposizione

Gli attori che entrano in scena sono:

1) Hardware
   - processore (DSP, µController)
   - Periferiche (misure analogiche e digitali, modulatore PWM ecc)
   - comunicazioni (scelta di solo tipo a seconda dell'ambiente e di chi è il spedizione)

2) software

• TMS 3 20F2808

È un DSP fixed point ~~della~~ della TEXAS - INSTR.

↓

lavora su numeri interi (no floating con virgola)

Va bene un fixed point per gli azionamenti con rispetto economico ma fare operazioni con # interi richiede una

- ottimizzazione fondo scala } il programmatore deve
- taratura } sapere fare bene

Nascono complessità legate al fatto che un fixed point

nb: il d-space è un floating point → ma è semplice con matlab

3) HARDWARE ACCESSORI

- layout di potenza (filtri e compatibilità elettromagn.)
- misure (sensori)
- interfacce [(sensore-processore) → AD]

3) Diagnostica guasti

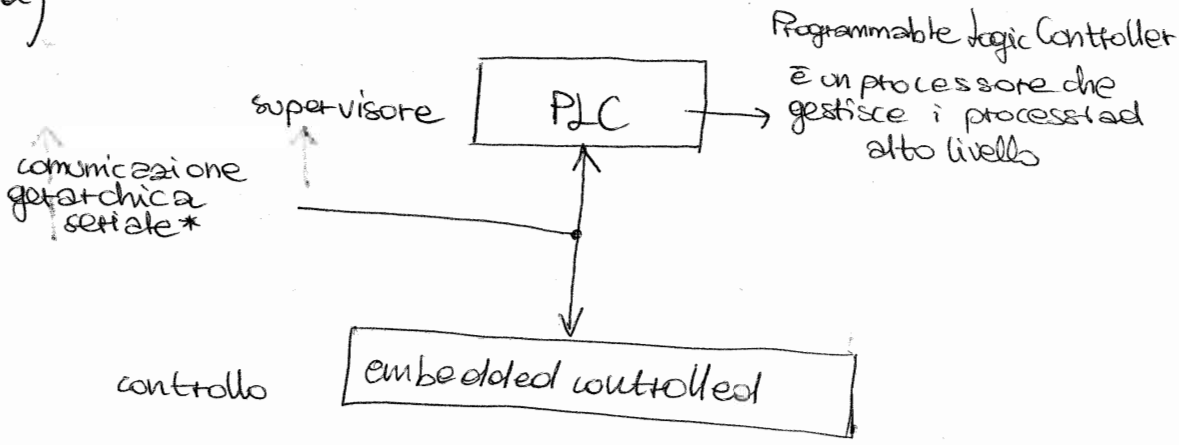
È possibile dedicare una parte di risorse (memoria e comunicazione) per registrare ciò che è accaduto, salvando ad esempio le info legate agli ultimi istanti pre-guasto

È possibile memorizzare e/o trasmettere stati e variabili.

4) Struttura gerarchica di facile implementazione

Può esserci una struttura gerarchica di tipo

a)



* con protocollo di comunicazione standard → PLC e embedded possono non parlare la stessa lingua

b) Multi-tasking

Si ha una gerarchia dei tasking di controllo
→ Gestione dei ≠ compiti nello stesso real time

Per definire il Multi-tasking bisogna definire il concetto di TASK

D'altra parte
È necessario che il load

$$t = \frac{C}{T} < 1$$

altrimenti sono in ritardo e non posso lavorare in real time

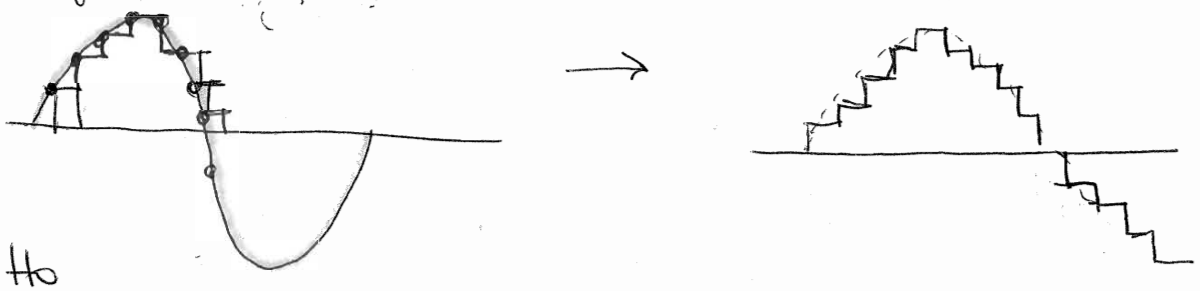
MISURE

Rappresento delle grandezze fisiche in modo digitale

$$A \longrightarrow D$$

la conversione AD porta con sé un errore

Es. Se si vuole campionare la forma d'onda → la si campiona ad intervalli regolari ... e si ottiene la forma d'onda



Ho

- errore di campionamento → il segnale continuo diventa a gradini (S & H) che introduce
  - ritardo
  - distorsione del segnale

→ si risolve realizzando un campionamento fitto (elevata ft di campionamento - tempo di campionam. piccolo)

- errore quantizzazione

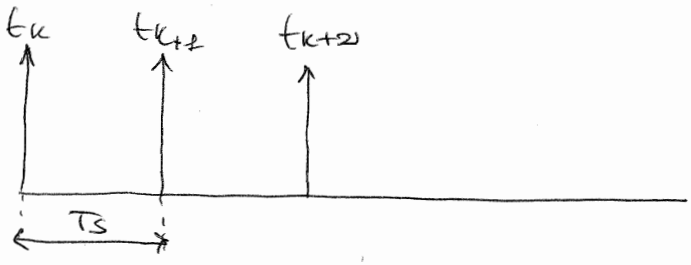
lunghezza numeri

Devo avere abbastanza bit per rappresentare i numeri
- errore di troncamento che dipende dal # bit
- es 16 bit, 32 bit

→ si risolve con un elevato numero di bit!

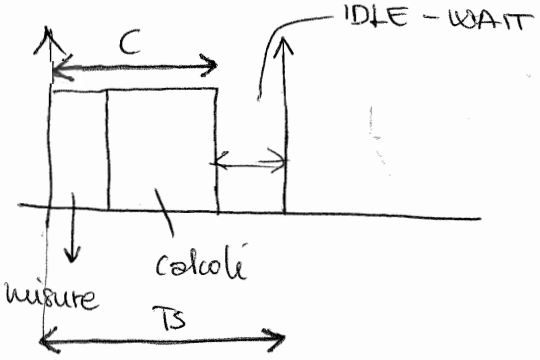
TASK → Un processore può dividere i compiti in tempi ≠

Il singolo task viene eseguito



Il timer produce gli impulsi interrupt ad istanti equispaziati.

In una esecuzione real time il Task real time il task è fatto così



Affinché venga rispettata l'esecuzione in real time è necessario che

C → tempo reale di computation < Tempo ciclo

Ovvero

$$\downarrow = \frac{C}{Ts} < 1 \quad \text{load}$$

nb: in un modulatore Ts ≡ Tpwm ho un solo task
≡ Tcampionamento
≡ tempo ciclo

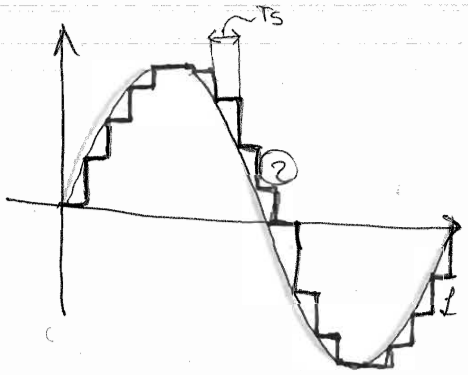
Per gli azionamenti 16 bit sono + che sufficienti, a meno di applicazioni particolari (posizionamento Nmetrico)

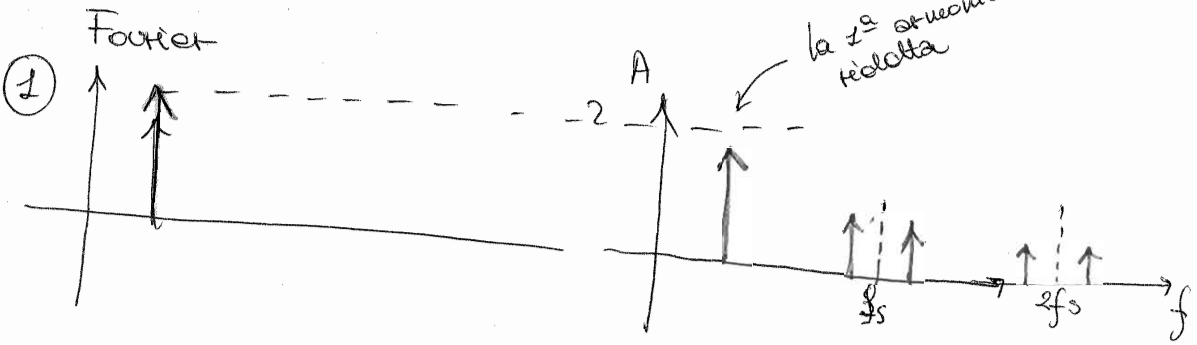
nb: ↑ è il # bit + è lento a convertire

Se segnale una volta campionato viene ritardato e presenta ritardo dovuto alla scalinatura!

La presenza dei gradini a istanti spaziati T_S introduce delle armoniche legate alla f. di campionamento e multiple

$$f_S = \frac{1}{T_S} \quad + \text{multiple}$$





Deve campionare con $T_S = \frac{1}{10} T_{segnale}$ → il teorema di Shannon non è sufficiente

Per risolvere il problema di ritardo e perdite di info legate al campionamento è necessario utilizzare una elevata f di campionamento, un breve periodo.

DIVERSI TIPI ESECUZIONE REAL-TIME

Può essere

- Soft → es: DVD player: non è critico se succede x qualche motivo che si degradi la prestazione: transitoriamente degrada la qualità dell'immagine
- firm → tollera errori saltuari
- **Hard** → non tollera errori ! → Se accade il controllo diventa inutile e/o pericoloso

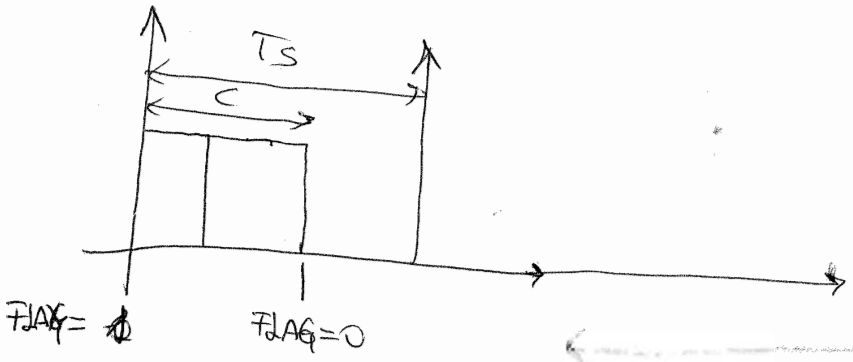
↓ noi siamo qui

L'algoritmo ha al suo interno if!

Bisogna fare la stima del tempo di calcolo, stima del

WCET (WC EXECUTION TIME)

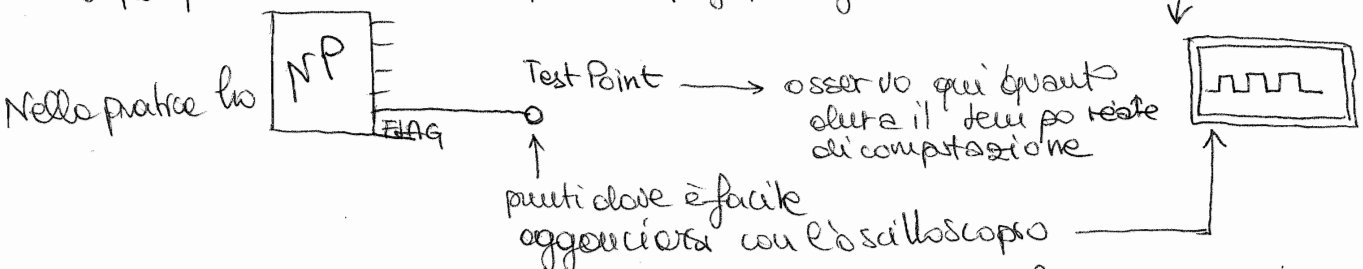
Sapendo che il processore riesce ad eseguire l'"x" controllo non si esegue un calcolo rigoroso (impossibile e poco pratico) ma si può valutare il tempo di esecuzione di algoritmi noti



Si inserisce nel codice un flag che va a
0 → quando i comandi sono pronti
1 → quando inizia

è un DIGITAL I/O

Si può pertanto valutare il tempo che impiega per eseguire i calcoli



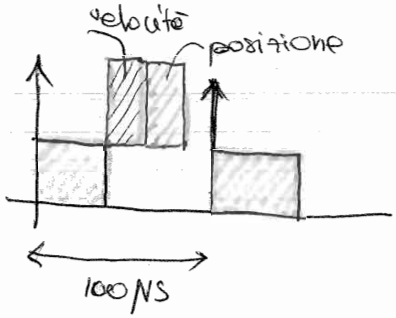
Si parte da un controllo semplice dio dio arricchendolo fin che non si sta + dentro al Ts

Esempio: motore DC controllato in posizione

- controllo di corrente → 100 NS → minor
- controllo di velocità → 200 NS
- controllo di posizione → 400 NS → major

Ci sto nei tempi? → lo verifico e ritento!

Se Task 1 ≡ Task 2



se in minor cycle non ci stanno tutti e 3 si va nell'altra modalità vista prima

Gli eventi + interni di un controllo in cascata sono i + veloci

Il major cycle in questo caso è 400 NS

Tutte le finestre di 400 µS sono uguali

Per definire l'algoritmo devo definirli tutti e 2 major e minor!

PROCESSORI DI UTILIZZO INDUSTRIALE

4 tipi di processori sono

- MCU micro controller unit
- MPU micro processore UNIT
- DSP digital signal processor ← utilizziamo questo x gli azionamenti
- PLC programmable logic controller
- PC industriale

MCU e DSP hanno le proprietà comuni di avere un

- processore di architettura semplificata specializzato per eseguire velocemente somme e moltiplicazioni e somme di mol (è di capacità limitata ma molto veloce) e lo fa in singoli tempi ciclo
- e hanno periferiche single chip execution nel chip ci sono le periferiche che servono

Per controllare un DC/DC converter serve un DSP con
- 2 canali - periferiche di ingresso (i, v)
- 1 modulatore PWM periferica di uscita

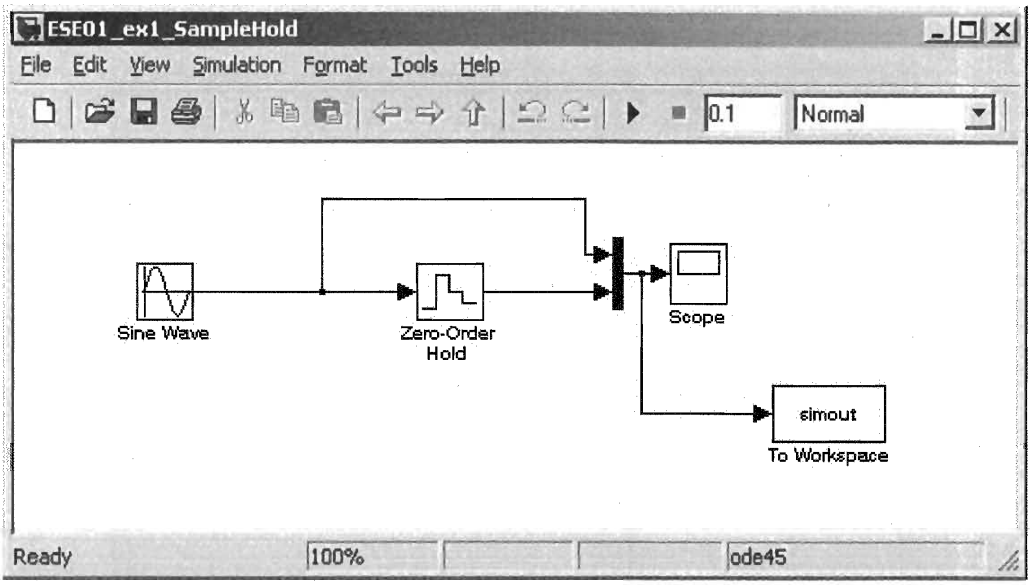
Per controllare un motore ci servono
- + canali AD (i, v, ω, ϑ
- interfaccia encoder (standardizzate)
- modulatore PWM almeno a 3 canali

A seconda della complessità del processo servono processori +

01NKXNC – Controllo digitale di azionamenti e convertitori
ESE1 – 7 marzo 2012

Ex 1 – Campionamento di un segnale analogico

Sample and hold di ordine zero di un segnale sinusoidale



Parametri:

Sine wave:

f frequenza in Hz

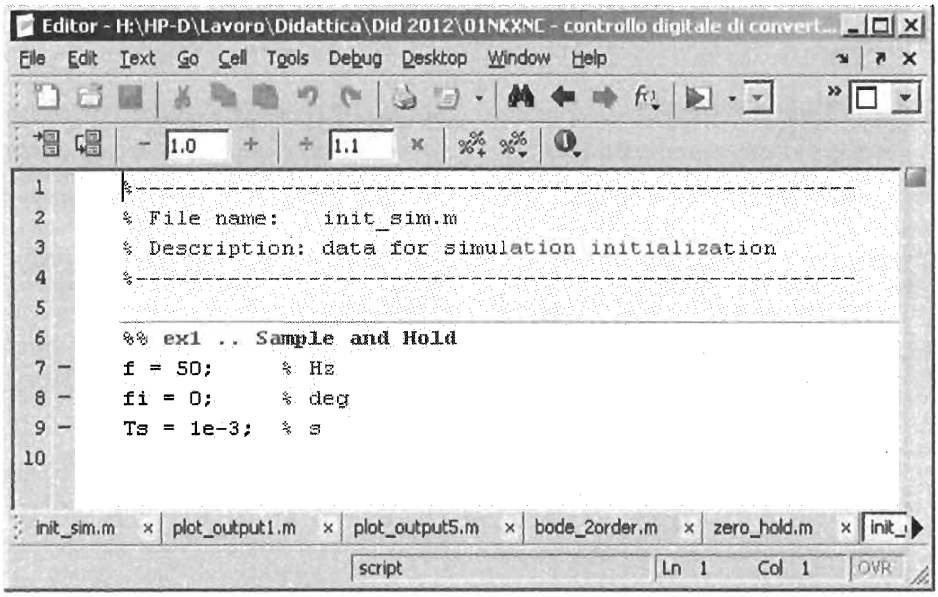
fi fase in gradi, rispetto all'istante t = 0

Zero-Order Hold:

Ts tempo di campionamento in s

File di inizializzazione e Callback:

init_ex1.m



Va eseguito prima del run simulation. Per questo si usa

01NKXNC – Controllo digitale di azionamenti e convertitori
ESE1 – 7 marzo 2012

ESERCIZIO

Mantenendo costante la frequenza di ingresso, verificare l'effetto del tempo di campionamento sulla ampiezza fondamentale del segnale campionato

Verificare in particolare la condizione limite del teorema di Shannon (Ts = 0.5 * 1/f)

Ripetere le stesse prove con una fase (fi) diversa da zero e valutare quando la fase ha effetto sulla fondamentale e quando no

$f = 50\,Hz \longrightarrow f_{sw} = 10 f_{segnale} = 500\,Hz$

$T_{\&} f = 20\,ms \qquad Tf = \frac{1}{500} = 2 \cdot e^{-3}$

Possiamo fare

$f_{sw} = 5 f_{segnale} = 250\,Hz \quad Tf = 4\,ms$

$f_{sw} = 2 f_{segnale} = 100\,Hz \quad Tf = 2\,ms \longrightarrow$ Shannon

cresce la Tf diminuisce la frequenza

↓

Non si campiona nulla

Solo modificando φ la fase si riesce a campionare ancora. Xkè Xkè campiono sugli zeri

$100\,\mu s \longrightarrow 10\,ms$

↓

errore nullo

$0{,}001 \cdot 10^{-3}$

$100\,ns = 0{,}1\,ms$

Riporta in grafico $\frac{T}{Ts}$ e le fondamentale

A Shannon in uscita non vedo nulla con fase zero, in uscita non c'è segnale → xkè campiono sugli zeri

Se metto $\varphi = 90 \rightarrow$ si campiono sui picchi.

Quando ho a disposizione 2 campioni al periodo posso campionare tutto o il contrario di tutto (→ mi sincronizzo con zero) e non becco nulla! pass

Analizzare come viene trattato simout

Es 2/

Controllo in anello chiuso di un sistema RT

arancio → eccitazione motore in corrente continua
blu → PI
giallo → convertitore ideale
Sono a tempo continuo

Negli schemi a blocchi ho solo tensioni (i, W, sono tradotti in tensione)

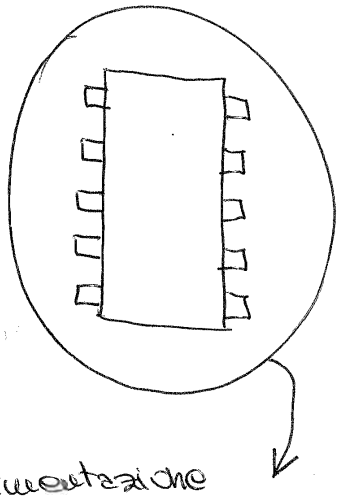
Nel PI
per aprirlo look under mask

- ho l'antiwind-up

Nell'es 3 ho un PI discreto

le cose principali sono

1) PIN OUT - quanti piu ha?

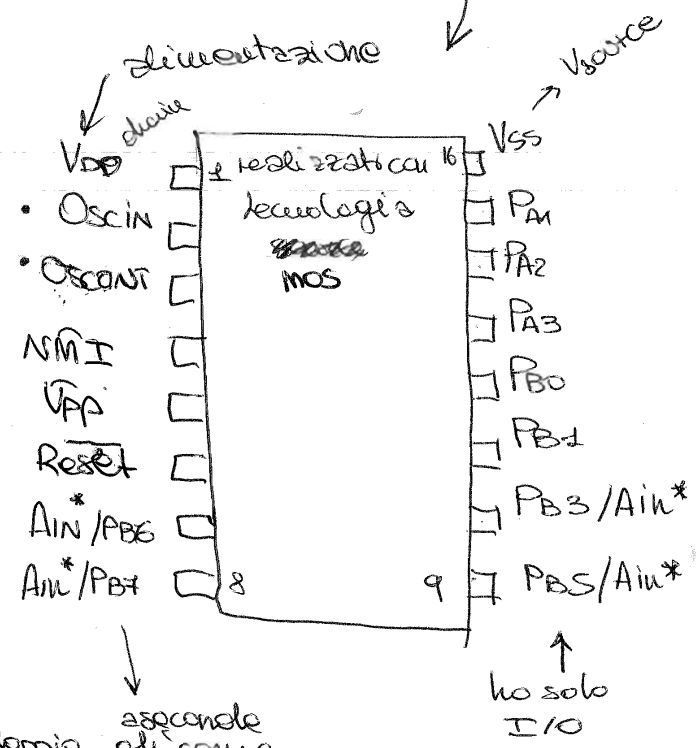


Es: l'ST6 è un MC a 8bit

Noi consideriamo ST6200C

Ha 16 pin

Ve più out ci dice la dimensione del processo ci dice quanti sono
- input
- output



VDD → tensione di alimentazione

OSC: voglio usare il clock interno x temporizzare qualcosa che sta fuori (OSC OUT) o per sincronizzarlo dall'esterno (OSC IN)

NMI è non mascherabile (non mascherabile interrupt) → quando attivo da 1; ha priorità 1

VPP → è un morsetto che serve x programmare il CIP (5V)

Quando VPP = 0 si entra in modalità programmando

negato → $\overline{\text{RESET}}$ → sia I/O top priority non mascherabile interrupt

P. le porte (insieme di bit) digitali I/O sono di tipo

A - B - C

Se è PB7 è il 7° bit della porta B!

Ho 2 memorie

- program
- data

PC → program counter è un registro
stack altri registri

JP BOS non esce dallo scatolo, non si può utilizzare memoria addizionale esterna (scheda) → ne ? che lo fanno

l'oscillatore interno produce un clock ad 8MHz : lo posso sfruttare x sincronizzare
- qualcosa all'esterno
- essere

oppure si può bloccare il clock interno OSC IN per sincronizzarlo dall'esterno

Yes

Registro → parola che corrisponde ad un indirizzo di memoria

- è una delle tante parole ad 8 bit
- può contenere • dati
  • istruzioni
  • indirizzi
  • parole di settaggio → es il settaggio registro del clock posso attraverso lo scrittore opportuno del # di registri.

la mappa di memoria STb
Si ha

e ogni # esadecimale è espresso come

F F F
↓ ↓ ↓
xxxx xxxx xxxx *

Se la dim. della memoria è * gli indirizzi devono essere parole a posto > di 8 bit (12 bit).

Della dimensione della memoria so quanti bit servono per indirizzare la memoria → me ne servono 12
e so quanti bit servono sul bus

la DATA Memory è + piccola → contiene tutte le varie celle nelle ≠ caselle. Tutto ciò che viene scritto e riscritto durante l'esecuzione
• 256 indirizzi di 8 bit

000h
8 bit
OFFh

ho parole di memoria di 8 bit che va da 000h a OFFh

Se l'indirizzo max è FFh = 255 le celle di memoria sono 256 (posti)
gli indirizzi per andare e leggere sono parole di 8 bit

Posso avere 80 + bus a 12 fili per accedere a tutti i 4096 indirizzi

Bus hanno funzionalità

- Program (dedicato alla program memory
  - data
  - address
- Data Memori
  - data
  - address

Ho un comparatore che dice quando il segnale digitale è sopra il livello: ci dice il # digitale y

decresce per ogni Tclock di ΔV

A start vale 0, poi conta 1, 2, 3. Ci si ferma sul y che è sul bit rappresentativo del segnale analogico.

y viene scritto nel data register dell'AD

Dopo di che continua il conteggio fino a che non arriva all'eq. dei 5 V che è 255!

→ in 8 bit voglio rappresentare un # da 0 a 5 Volt
Conto fino a 255 grazie a un clock in corrispondenza del quale si ha

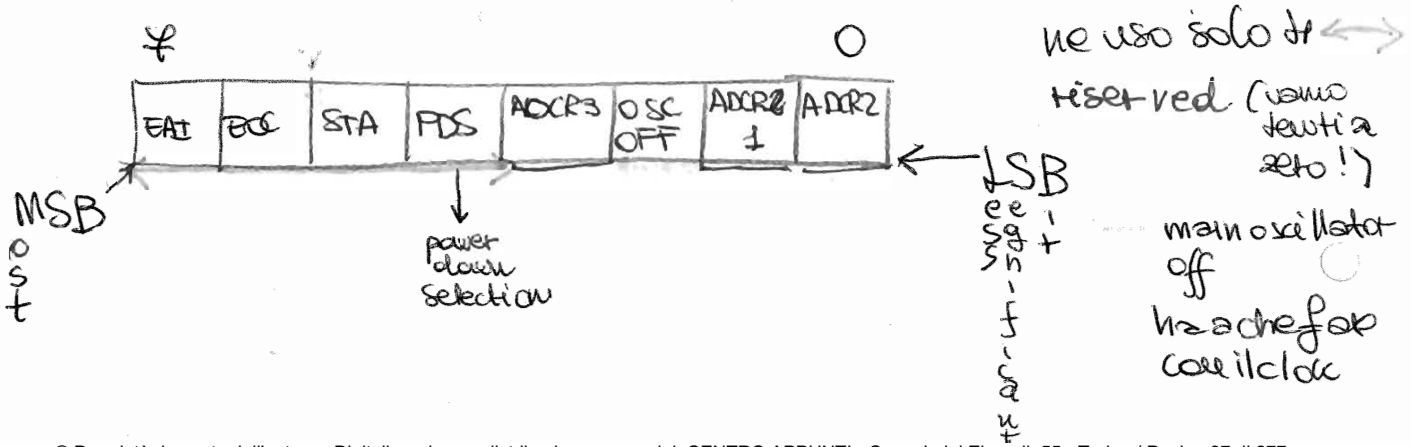
$$\Delta V = \frac{5V}{256}$$

→ con un comparatore ottengo un # digitale y

Quando il conteggio arriva a 255 ho End of conversion EOC

*Il control register è un registro
Se osserviamo il registro ADControl register

A pag 51/100 data sheet ho la descrizione del registro. Il registro ha 8 bit ma ne uso solo 4 ↔ reserved (vanno tenuti a zero!)



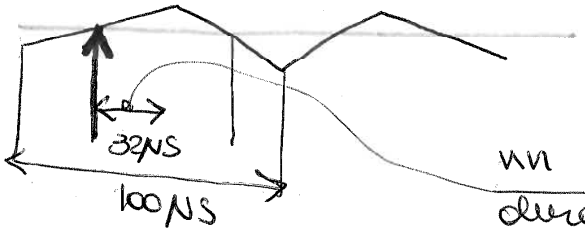
la conversione

Dura 256 x Tclock con

Tclock = 125 NS → 8 MHz = $\frac{1}{125NS}$

Tconv = 256 · 125NS = 32NS → non è veloce ma è molto semplice non ho S&H

Se un convertitore a 10 kHz non potrei sfruttarlo, si considera un buco. io vorrei campionare → ma lui si impiega 32NS



10kHz

va bene, sta cambiando il segnale durante questo intervallo di tempo

mi servono 32NS per il NS e i TS deve essere almeno 1ms (?)

Hardware è dedicato, nel mentre può fare altre cose!

la data memory si può 11/3/19

- leggere
- scrivere → durante l'esecuz. del programma

Contiene

- variabili
- settaggi → es. • AD/control register
  - registri dedicati alle porte (comunicazioni con l'esterno) I/O con l'esterno

↓

hanno vari registri

data register delle porte → è scritto quel che la porta vede o che vuole comunicare all'esterno c'è il dato

registri di settaggio

V porta e V pin della porta

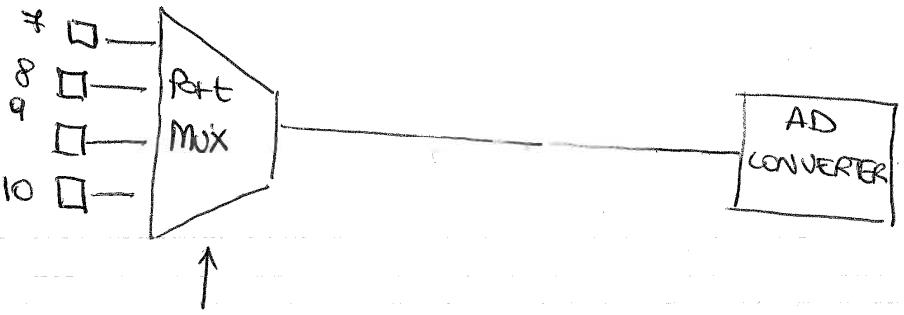
indica se è un Io su 0 cosa fisica di un settore? → metto 0 o 1 in 1 bit del director

Posso acquisire più grandezze purché rispetti i tempi.

Es misura di T o umidità sono fenomeni lenti, allora non è un probl non ho problema di dinamica del controllo.
& configurare e riconfigurare la porta in maniera tale da far ~~configurare~~ ricevere il segnale.

MULTIPLEXER

Y possibili ingressi analogici sono



Tramite i bit di settaggio posso collegare l'ingresso 7 all'AD

Settare i registri vuole dire configurare i NC e quindi le periferiche

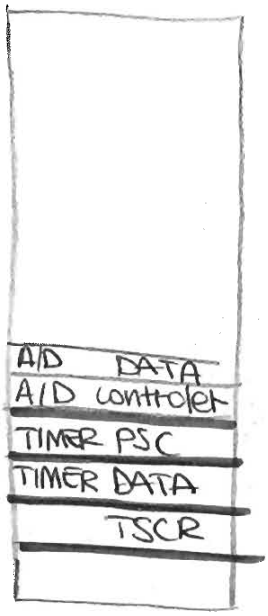
nb: non c'è controllo sull'incompatibilità: se dico 7.e 8 insieme ho due i/o vengono cortocircuitati

Ho dispositivo può

- ricevere In digitali
- produrre O digitali
- accettare In analogici $\longrightarrow \Delta T = 32\,NS \longrightarrow$ + comprovare + segnali di cui servono
- segnale basta nelle sue variazioni (in 32 NS deve exhost)
- deve avere tempo a sufficienza tra i campioni

Se devo campionare ogni 1 ms allora il tempo dedicato alla misura è trascurabile (32 NS)

Il timer è una periferica del NC (è un hardware dedicato)



Ha

- un registro di dato → mi dice il # a cui è arrivato il conteggio
- 2 registri di settaggio
  - PSC - pre scaler

    decidi quanto lungo deve essere il tempo di aggiornamento del contatore

    PSC = 1 → uso Tclock interno

    posso voluto rallentare il clock

    N x Tcloc in modo da far timer e contatori + lenti

Se metto PSC = 1 → $T_{TIMER} = T_{clock} = 125\,ns$

PSC = 16 $T_{TIMER} = 16 \times T_{clock} = 2\,\mu s$

Con $T_{TIMER} = 125\,ns$ $T_{PWM}' = 256 \cdot 125\,ns = 32\,\mu s$

$T_{TIMER} = 2\,\mu s$ $T_{PWM} = 16 \cdot T_{PWM}' = 512\,\mu s$

Scelgo T_{PWM} a seconda della velocità dei segnali da mandare in uscita. È sempre da trovare il compromesso tra

- velocità
- precisione

pag. 115 ÷ 116

- TSCR è il timer status/control register → è un registro di stato e o controllo.
  - status → ho dei bit che mi dicono info accessorie (se il processo è partito, se il conteggio è finito)

I timer non vengono sfruttati solo per la PWM ma servono x scandire altri processi ad esempio aspetta tot cicli quando arriva il dato

MPU : micro processore

È + evoluto ma non ha le periferiche di I/O

È all'interno di una Mother board, e a bordo schede ci sono periferiche (I analogici, I dig, ethernet..... O ....) esterne

DSP: Digital Signal Processor

specializzato per moltiplica e accumulazione su singolo ciclo

È un Controllore specializzato nella elaborazione di segnale (dove intendo segnali analogici)

Applicazioni Audio / Video / telefonia

Data la specializzazione di utilizzo si ha un CORE SPECIALIZZATO nella esecuzione di multiply and accumulate in un singolo ciclo (MAC)

L'hardware del processore è

- ipersemplice
- specializzato per fare sommatori MAC

L'operazione MAC mult + accum

$$Ris_{k+1} = Risk + A * B$$ l'uscita è l'accumulato

È l'operazione + complessa che devo sapere fare.
Se voglio fare solo somma metto $A_i = 1$ or $B=1$, se voglio fare solo la moltiplicazione metto $Risk = 0$!

Serve fare velocemente queste operazioni perché so ho un segnale (e I da elaborare

Da fdt è una sommatoria pesata di campioni vecchi.

Se il filtro è di ordine 3 $n=3$ se il filtro è di ordine 12 $n=12$ → mi servono 12 dati vecchi

---

DSP

Il PLC

Siemes
Allen bradley
OMRON
Shneider

19/6/12

È una logica programmabile

Serve come

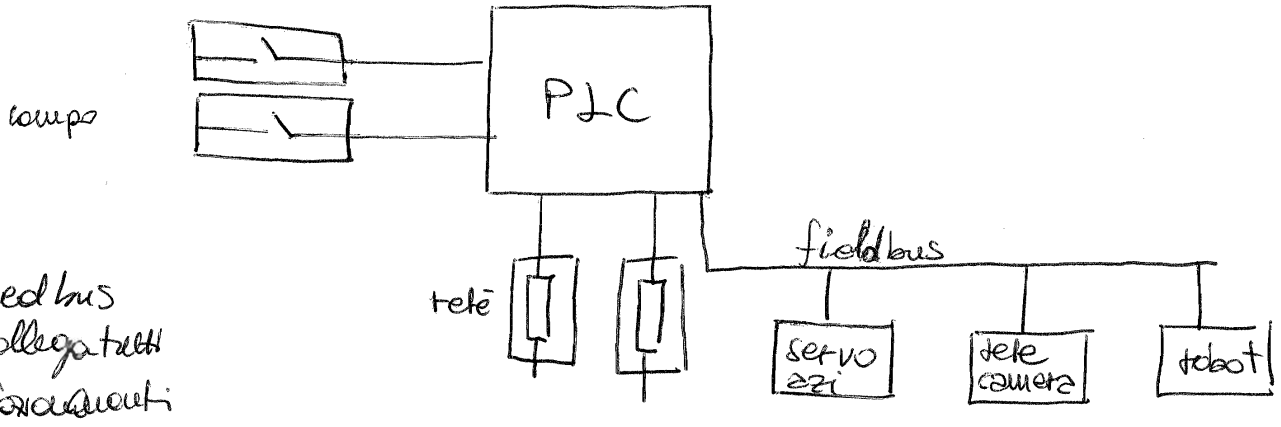
- gestore di una parte di un processo /o di un processo

e nasce come

- gestore di logiche a relè x gli impianti elettrici → strumento con logica semplice (apre e chiude sulla base di
  - sensori
  - ~~interr~~ operatori manuali

e si evolve in un

- computer che gestisce diversi e logica che comunica con il campo ad alto livello → non è un livello di automazione ma di direttamente
  - comandi
  - I/O diretti dal plc (con rete)
  - bus di campo → I/O su una unica linea seriale →



feedbus
collega tutti gli azionamenti gestiti dal PLC

Il field bus è una linea seriale di comunicazione
- standardizzato
- compatibile con l'ambiente industriale

↓
tutti sono collegabili // e possono ricevere o dare in ↓ memoria

Gli standard + tipici sono
- CAN → automotive
- CAN OPEN
- PROFIBUS → siemens

} sono tipici field bus e i nomi sono i nomi dei protocolli di comunicazione

Programmazione del PLC

Si può usare
- × LADDER → è un linguaggio schematico derivato dai sistemi elettrici (proprio xkè il PLC è semplice di tradizione). è anche detto linguaggio a contatti (*)
- × TESTO STRUTTURATO → è uno linguaggio di programmazione pse tipo PASCAL
  → serve × logiche complesse

(*) ⊣⊢ relè normalmente OFF

⊣/⊢ " " ON

Esempio listato LADDER

```
*
1
2   comp X[2]
3   ⊣⊢⊣/⊢──(Y[4])
4   └──TM[1].RST
```

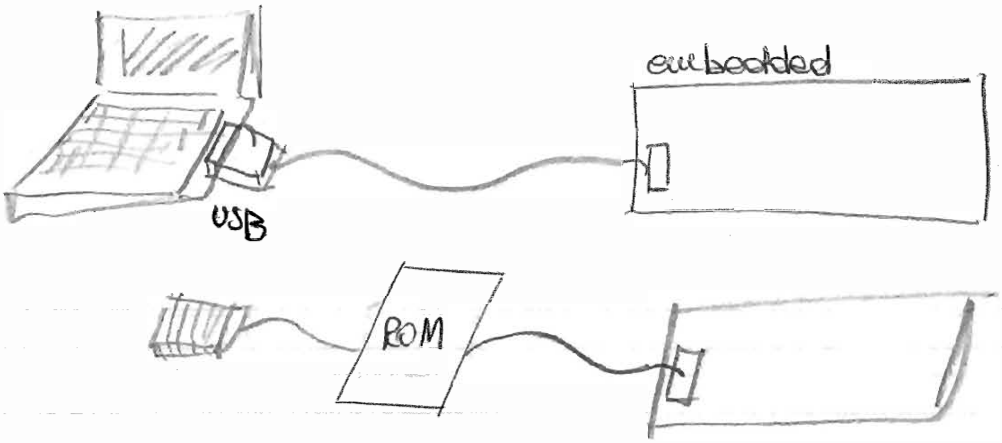
* istruzione inserite in sequenza e letta ↓
* ogni istruzione è un rung
* ogni rung va letto da sx a dx

es
Y[4]=COMP ✱ & X[2] (X = 2)
↓ ↑not
AND

PROGRAMMAZIONE HOST – TARGET

Come comunicano i sistemi embedded?

Si prende l'ospite Host (pc esterno) e lo si collega e target che è il sistema embedded → è un esecutore che deve essere programmato



Per ottenerlo si programma una ROM

Il codice del PC viene scritto in C → tradotto in assembly e viene compilato in codice macchina (sequenze di 1 e 0) che deve essere scaricato sul sistema embedded

Suite di sviluppo → sistema di sviluppo

È installato sull' Host PC

Es: Code composer studio è il sistema di sviluppo della Texas Instrument

Produce alla fine compilato il codice eseguibile macchina che deve essere memorizzato nella program memory del target

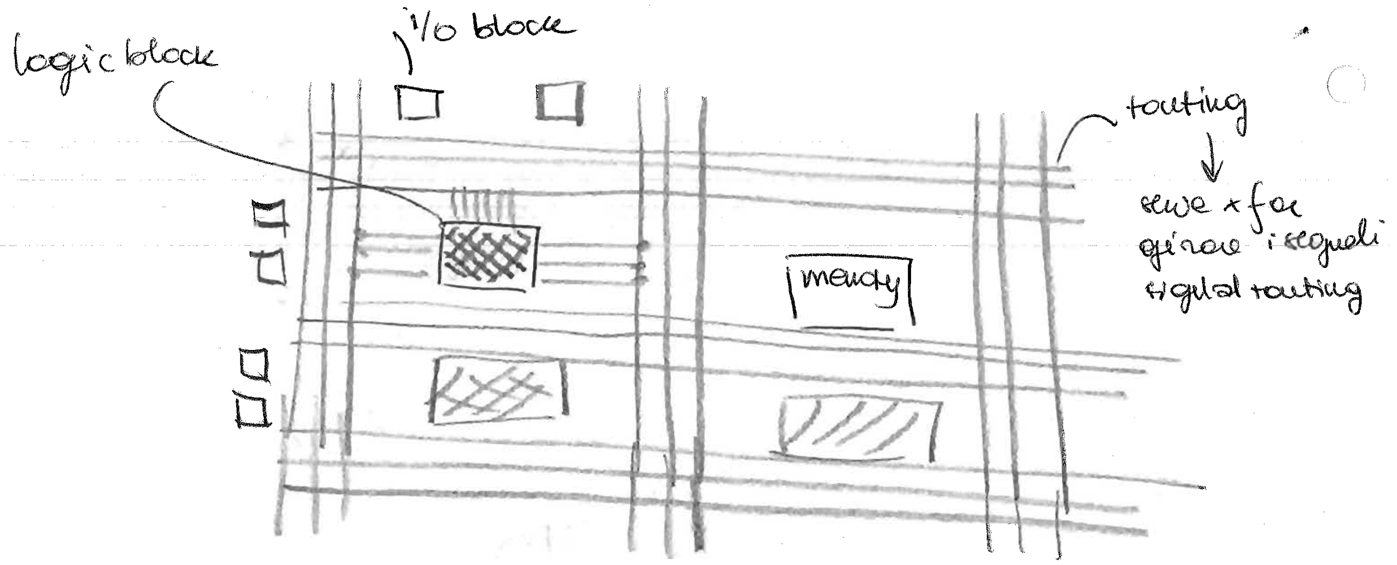
- o su una rom che viene programmata e poi montata (non modificabile)
- o a bordo scheda o dentro il MC c'è una EEPROM-FLASH (una ROM programmabile) sono memorie non volatili in cui

una volta stabilito che il NC sta comunicando tramite Jtag non è + padrone di se stesso ma esegue quanto stabilito dall'Host

FPGA (field Programmable Gate Array

È una
Logica programmabile a livello Hardware

La Gate Array ha una struttura a metà a matrice composta da transistor di tipo MOS



All'interno dei blocchi logici (composto di flip flop transistor ecc. ~~quali fare~~ è completamente configurabile

È il max della flessibilità, si possono
- implementare i DSP dentro l'FPGA → con x caratteristiche
- realizzare strutture
  - supercomplesse
  - superdelicate

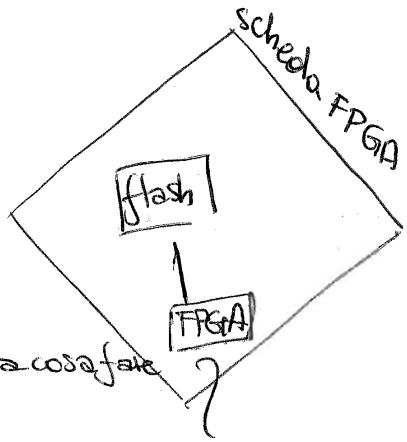
} permette di costruirsi un ~~host~~ ASIC (Application Specific Integrate Circuit) ↓ fa - un real time

Con l'FPGA puoi fare un Asic di volta in volta (NC, PLC .. tutto quello che voglio). Sono hardware ~~gr~~ costosi e ~~di~~ complessi

la memoria dell'FPGA è volatile

Al Power On serve qualcosa che ricarichi all'FPGA come deve essere programmato
Può essere:

- FLASH
- Flash interna
  - direttamente da flash
    (è già programmata all'accensione iniziale)
  - tramite stream



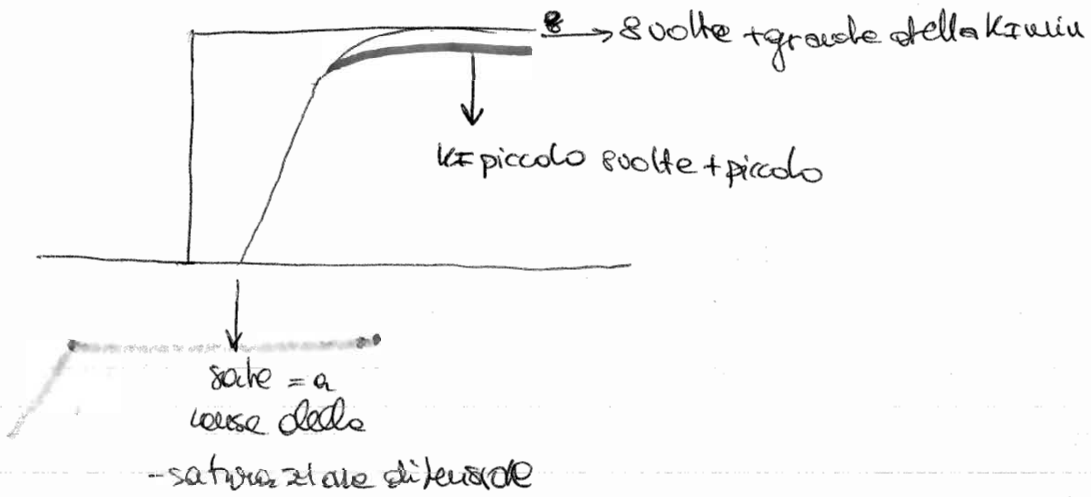
Cmq serve una fase in cui bisogna ricaricare alla scheda cosa fare
Questo perché la gate array è un pezzo di transistor che non deve dialogare con altri componenti

l'errore a regime si riduce con Kp

Con $K_p = 3500$ passo a situazioni vicine a $K_{I\,min}$

→ 8 volte + grande γ^{-1} → a metà tra $\frac{1}{2}$ cubo e γ^{-1}

8 volte + piccolo γ^{-1} → a sx del cubo



tempi di simulazione

Sono in una interfaccia analogica ma numericamente (calcolatore digitale). I tempi di settaggio di simulink devono essere appropriati

simulation
configuration

Max step t_{MS} → va tutto bene

↓

tempi del sistema è 100 ms allora t_{MS} è piccolo rispetto al sistema fisico

Se gli metto l'automatico stabilisce che per fare convergere la simulazione bisognerà tempi + lenti

Occhio che è un polare

Provando a tirare + la banda nascono problemi di stabilità

Ritardo di camp. e di esecuzione

La corrente verde e rossa sono + la campionata dell'altra a passo regolare 1ms. Anche l'analogico (

Tra tensione e corrente ho 1ms di differenza grazie al $\frac{1}{2}$ mesr a della

Va sempre un margine

→ tra valor max stazionario

→ quello applicabile transitoriamente

bias → dare un rif. polarizzato intorno a 0,28A → stagnazione + svuo.

possibile →

R=55 Ω

La risp di piccolo segnale senza bias non mi darebbe sensionale + (è unipolare) →

Il DC/AC diventa bipolare x collegamento

- → verificare che il step size non influenzi il risultato → i risultati non siano eccessivi

Ts = 0.2ms → risposte simili alle analogico

Ts = 1 ms → risposte con amplificazione

Tempo di simulazione

Uso SNS Algoritmo fixed step

→ realizzo una triangola con 40 campioni

Se provo a mettere "auto" sbaglio Time step

→ la triangola è a casaccio

Con i settaggi corretti del time step il risultato è lo stesso dell'altro dato. Il valore medio = con la Vref (non lo stesso dato)

Verificare l'effetto

nel caso a

$T_S = T_{PWM} = 200\,ms$

$T_S = T_{PW} = 1\,ms$

Verificare cosa cambia nell'uso master s_{fz} = auto

Sim power system → provando ad usare la modellistica circuitale di simulink

Per impostare il tempo di simulazione

simulation

configuration

Come si fa a simulare un controllo di DC/DC converter

→ lo corrente si assesta ad un valore

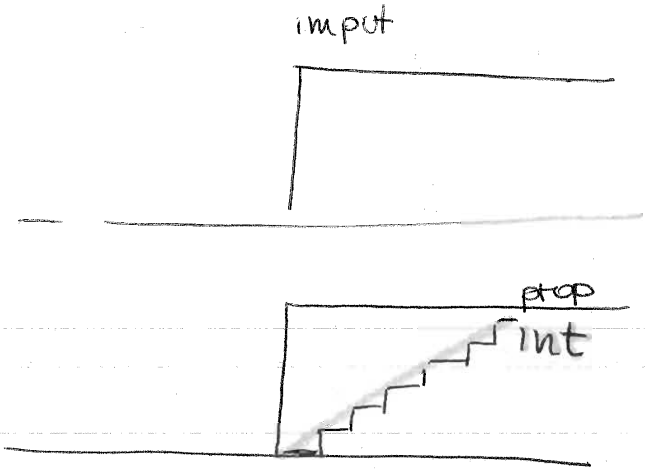
$$x = \frac{1V}{\cancel{k}\,\Omega}$$

Nel sistema fisico non lo devo mettere → ⚠ a quando uso il real time workshop

Model Update

È quello giusto da fare se devono S function soprattutto una folt

Se devo scrivere la folt analogica di un PI



Se devo fare un controllore real time devo ricordarsi le uscite

Nella cartella ho una S function ≠, PI model update da use

- model output → x la proporzionale
- integrale model update

Reagisce come un PI analogico comprende

Anche l'integrale si deve spostare + tardi

Notare che dentro PI è incompleto → manca IC wind-up

Viene ripetuto istante x istante il model output.

→ Quando devo fare l'integratore mi servono gli stati. Mentre con le S function usiamo dentro le model output (dove scrivere tutto)

→ Verrà eseguita in un loop temporizzato.

→

Model terminate → è un function in C

~~dati~~

void → vuoto → non da un numero in uscita

S = sim Struct

quelle che manipolano le simstruct iniziano con SS

Il tempo di campionamento dominio l'ambiente ~~d~~ a cui è associato

The model Outp è l'algoritmo

chiamo l'output ad ogni step

Una volta scritto il listato va compilato scrivendo nelle command

windows ~~sl~~ simatlab ~~&~~ mex ____
timestwo.c

Ottengo mex. x → a seconda della versione
↓
deve rilanciarlo con mex..

↓ ~~ve~~ quando arriva l'interrupt lo servo

~~INTERRUPT~~ ISR → è l'interrupt della modulazione → il modulatore è una periferica

↓

~~Inter~~ Il modulatore può mandare interrupt da cui servono + dire quando fare i conti:
- vertice alto della triangolo
- " basso della "

~~OVERFL~~
underflow interrupt

È sincronizzato con la triangolo!

Il listato c che esegue l'algoritmo di controllo è lo stesso con il quale posso programmare un NC o un DSP

La 1ª esigenza è
- simulare il controllo prima e durante l'implementaz. Senza e proprie → 1° motivo per cui far S-function

Il codice
- simulato
- implementato sul DSP

deve essere proprio =!

~~S-function~~
~~Simulazione~~
per simulare un controllo
- se uso la S function in C non è utile

Cercare di fare un controllo di azionamento con gli schemi a blocchi è + complesso a vari livelli

- la simulazione deve coincidere con la implementazione
  - dSPace → implementati in ambiente simulink e compilati direttamente programmando un DSP che sta a bordo della scheda DSpace
  - dSPace è basato su RTW (real time workshop) lo schema simulink può essere tradotto in codice macchina.
    Altro è code-generation per real time controller
    Viene compilato in listato C e poi in un eseguibile

Per fare

$$K_I \int ERR\,dt = K_I\, T_S \cdot (ERR)_k \quad * \rightarrow \text{Termine addizionale}$$

ciò che deve essere aggiunto

l'integrale totale è la somma di *

Se sono all'istante $t = t_k$ (misure e riferimenti campionati a $t = t_k$)

Sto calcolando le uscite per t_{k+1}

$$ERR_{k+1} = u^*_k - u_k ;$$

$$(PROP)_{k+1} = k_p * (ERR)_{k+1} ;$$

$$(INT)_{k+1} = (INT)_k + T_S \cdot K_I \cdot (ERR)_{k+1}$$

$$OUT_{k+1} = (PROP)_{k+1} + (INT)_{k+1} \quad *$$

⚠

L'ordine di esecuzione è ciò che conta!

Se cambio l'ordine di esecuzione posso fare errori gravi

Se calcolo ERR alla fine farei un errore! → lavorare sull'errore vecchio

Calcolo l'errore ma lo tolto perché lo faglio. all'errore aggiornato

L'algoritmo funzionerebbe lo stesso ma rischio ottimizzato perdo del tempo: piuttosto che utilizzare l'informazione utile utilizzo l'informazione vecchia. È come se eseguissi approssimativamente in ritardo (z^{-1})

Con Simulink e con realtime workshop si riesce facilmente realizzato un PI corretto, ma in un sistema + complesso è + semplice scrivere * di A generico

↓

uno degli standard!

26/3/12

CONTROLLO MOTORE DC AD ECCITAZIONE SEPARATA

Il motore DC ad eccitaz. separata richiede

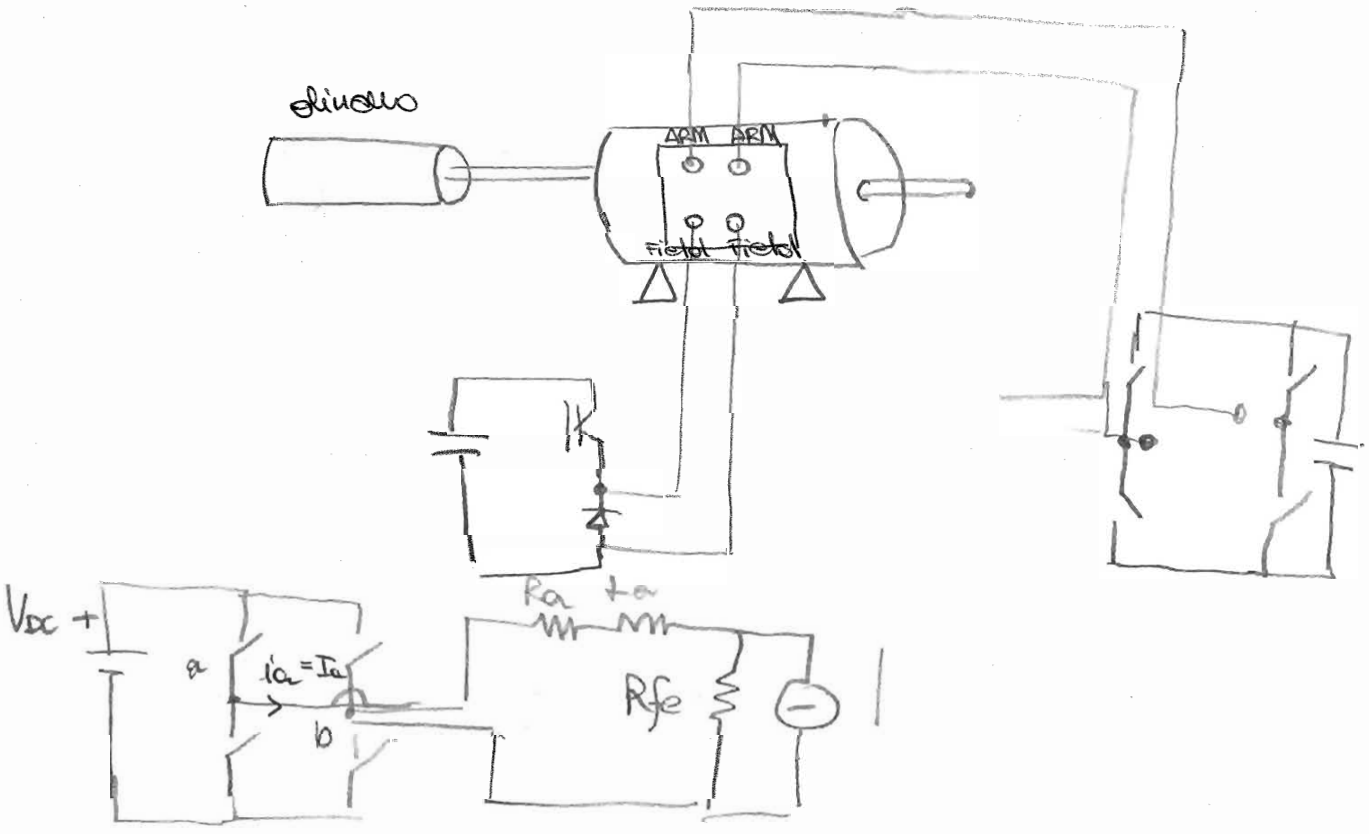
- ⊗ un convertitore DC/DC per l'eccitazione
- ⊗ un " " per l'armatura

la corrente di eccitazione richiede un

- convertitore a ad 1 quadrante → cambiando qui cambia il verso di rotazione

* L'armatura richiede un convertitore

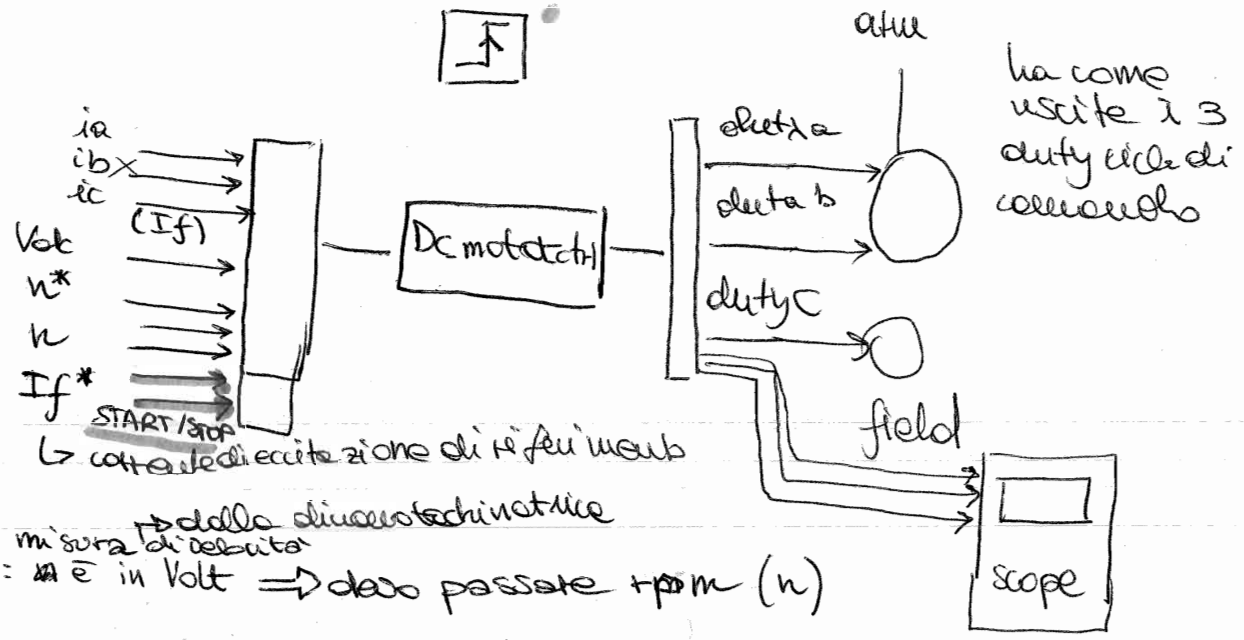
- a 4 quadranti → ponte ad H



le misure di corrente sono piuttosto precise.

Devo costruire una sfunction DC MOTOR

DC_motor_ctrl



↳ controllo di eccitazione di riferimento

misura di velocità ⇒ dalla dinamo tachimetrica

nb: è in Volt ⇒ devo passare a rpm (n)

La sfunction è eseguita ciclicamente con lo PWM interrupt

L'obbiettivo finale

- Controllo di eccitazione
- Controllo di velocità in cascata al controllo di corrente

Struttura dell'algoritmo

- segnali start/stop
- operazioni preliminari verifica
- " " precarico

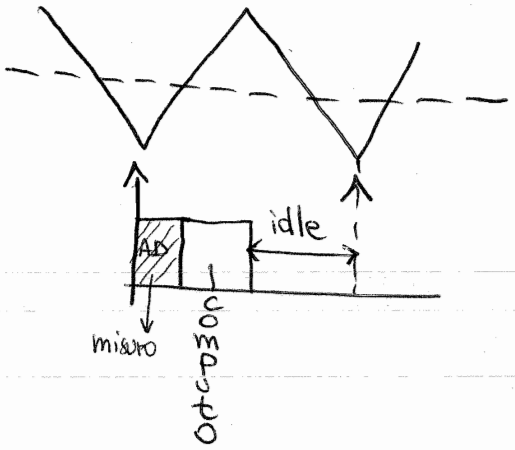
} operazioni preliminari

- TASK di controllo
  - corrente di eccitazioni
  - controllo della n* dalla ia*
  - controllo Ia*

Negli istanti zero

- ho nuove misure
  - ω
  - coppia
- eseguo i comandi

⇒ TASK SINCRONO PER LA PWM

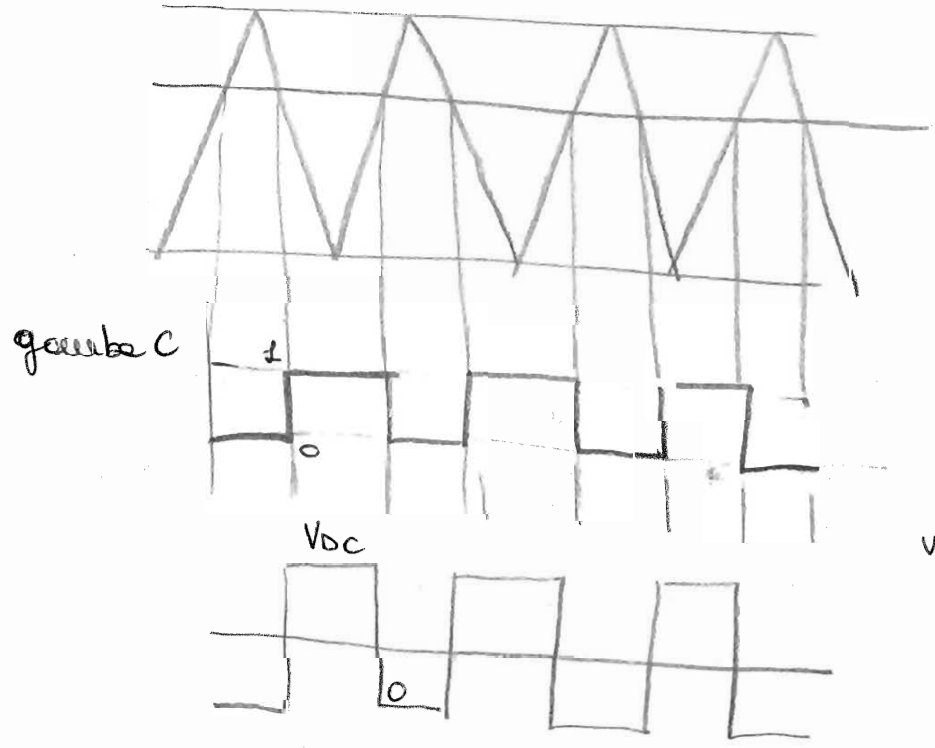


→ fare il doppio aggiornamento ha senso se faccio anche il doppio campionamento

↓

fare 2 campionamenti se mai x eliminare gli errori

Duty - ciclo



moltiplico per Vdc

Se conosciamo i parametri

$R_f = 550\ \Omega$

$L_f = 55\ H$

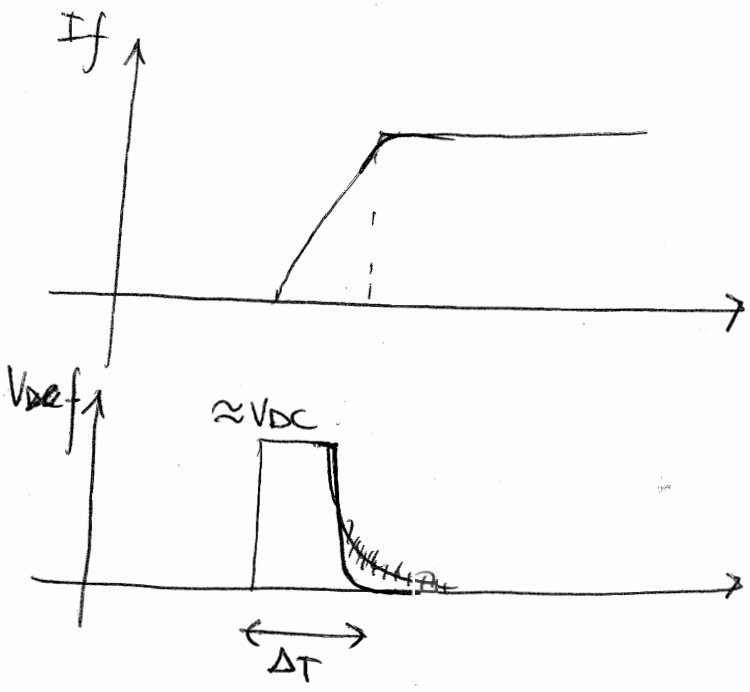
$V_{DC} \neq (V_{I,max}) \Rightarrow$

$\Rightarrow 308\ V$

Quando chiudo con una V_{DC} quanto tempo impiego per raggiungere 0,4 A?

Calcolo la costante di tempo

$\tau_f = \frac{L_f}{R_f} = 100\ ms$ → è valido in anello aperto



Quanto tempo ci metterebbe a salire se fosse una rampa costante

~~$\Delta t = 2t$~~

$V_{DC} = V_f \cdot \frac{\Delta i}{\Delta t}$

$\Delta t = L_f \cdot \frac{\Delta I}{V_{DC}}$

$= 55 \frac{0,4}{308}$

$= 71\ ms$

La banda non è significativa del tempo di risposta → conta quando non sono in dinamica saturata (risposte di piccolo segnale). Se sono ad in dinamica ~~non~~ saturate raggiungo subito ~~sempre~~ ciò

Si fa in modo che

$$\begin{cases} duty_b = 1 - duty_a = 1 - duty \\ duty_a = duty \end{cases}$$

$$\boxed{V_a = (2\,duty - 1)\,V_{dc}}$$

$$duty = \frac{V_a + V_{DC}}{2} = \frac{1}{2}\,\frac{V_{dc}}{V_{dc}} + \frac{1}{2}\,V_a^*/V_{dc}$$

↑ che voglio realizzare

Se voglio realizzare una V_a^* devo avere

$$duty_a = \frac{1}{2} + \frac{1}{2}\,\frac{V_a^*}{V_{dc}}$$

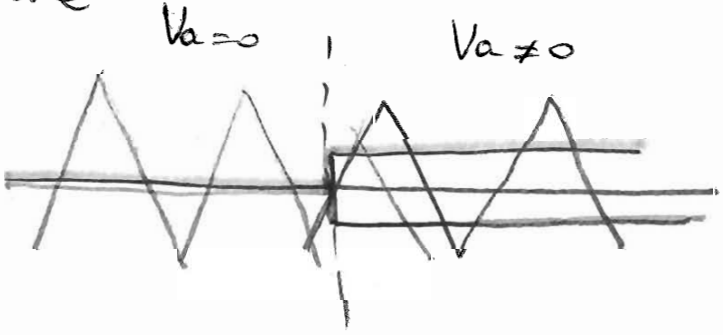
$$duty_b = \frac{1}{2} - \frac{1}{2}\,\frac{V_a^*}{V_{dc}}$$

V_a = V differenziale tra le 2 gambe del ponte

La tensione viene presa $\frac{1}{2} * 1$ e una polarizzazione di $\frac{1}{2}$ che coincide con lo stato zero per il ponte ad H.

La tensione è ad onda quadra di 1ª armonica ma è monofase non segue alcuna.

Se $D = 0{,}5$ il motore è alimentato da tensione nulla, se ho $D \neq 0{,}5$ entrambi le modulanti si spostano rispetto alla media e



28/3/24

CONTROLLO MOTORE DC (ecc separata) con cmex s-function

Controllicmo

- armatura controllata in tensione (open loop speed control → poi controllo
  - T
  - ω
- campo: closed loop current control

Si fa così x non complicare troppo il codice

Quando passo alimentare in tensione, allora posso controllare il resto.

$R_a = 2{,}6\ \Omega$

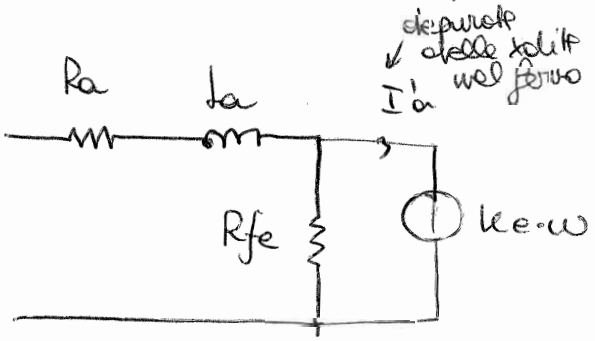
$V_a = 220\ V$

$I_a = 7{,}73\ A$

$V_f = 220\ V$
$I_f = 0{,}4\ A$ } campo

$K_{I,nom} = 0{,}5 \frac{V}{rad/s}$

Se alimento il motore a 100 V questo si muoverà a una velocità



dispersa dalle spire nel ferro

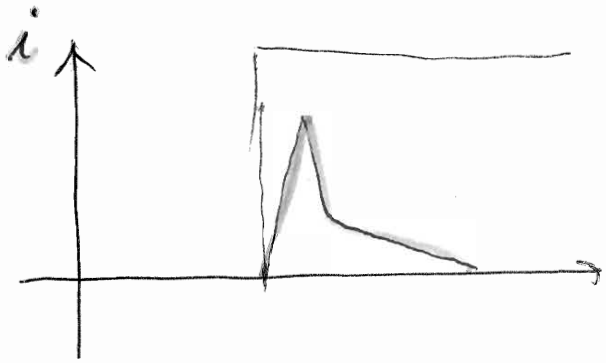
→ a vuoto passa solo la corrente che serve x Ω perdute

I_a' → produce la coppia elettromeccanica

A vuoto si mette a girare ad un ω che è proporzionale alla tensione di alimentazione

Controllare la V è un modo x controllare la velocità

• Corrente controllata → in quello perché la corrente fa cose poco piacevoli in risposta al gradino



Rispetto alle costanti di tempo

$$T_a = \frac{L_a}{R_a}$$

$$T_{mec} = \frac{R_a J}{k_e k_T}$$

→ Si usa questo controllo.

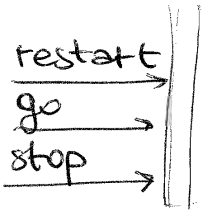
X controllare le correnti di avviamento si deve ridurre un gradino, si ha una corrente a gradino.

Regolando la pendenza rampa regolo la I_{max} transitoria. Bisogna conoscere

- macchina
- inerzia
- corrente massima

Il controllo di tensione si usa x applicazioni senza prestazioni particolari e solo in fase di test dell'algoritmo.

Se ho come obbiettivo il controllo in cascata di coppia e velocità in quello chiuso si usa come test preliminare



Altre uscite nello

w*_ manip
w*_m_fit
ia_out } mi consente di vedere
ibout } se le sto campionando bene
iC out
if*
ia* → è il ref di corrente dell'anello di controllo

SPAte → cartella di test → lo uso x la verifica dell'algoritmo
Stdte → è lo stato operativo : può assumere dei livelli ex sono
- error (tutto è fermo) ERR
- wait - up
- stop (alla sessione nello in uscita
- Go (porte) /START

```
#define  S. function_name  DC_motor_ctrl_Va
                                  ↑ deve definire
=
=
=

static void mdlOutputs (

}*
```

* scrivo l'algoritmo

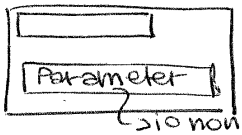
}

```
#define  N INPUT   1
         N OUTPUT  13

#define  ctrl_type   (mxGetPr(ssGetSFcn
                      Param(S,0))[0]
```

variabile o host

è il 1° parametro

```
#define  ACCEL
#  "     KPF      %field
         KIF
         KPA
         KIA
         KPw
         KIw

#define  NPARAMS  8
```



→ io non metto nomi etima k che mi mandano all'amask → va a prendere i parametri dati da fuori (maschera simulink) e li assegna a ke l variabile

Output Model
↓
Ricopio i parametri dentro delle variabili perché
- non riuscirei ad utilizzarli nelle function (non è una variabile global

```
mdlOutputs(
{
  ----
```

```
Accel = ACCEL * tpm*2*pi/30*Ts;
Ctrl_Reg_Param_f.Kp = KPF;
Ctrl_Reg_Param_f.KI = KIf*Ts;
```

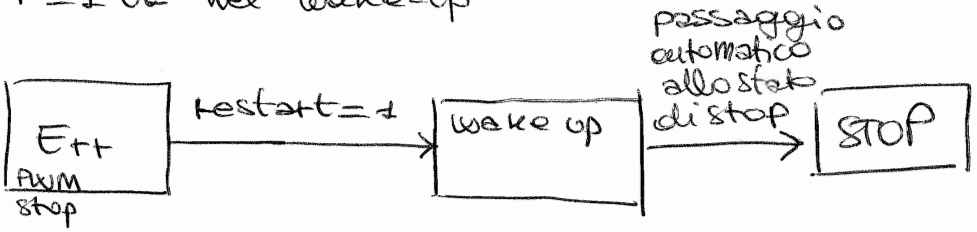
Devo definire accel in variabile h!
Io faccio x tutti.
I parametri possono avere bisogno di essere riscalati dalla maschera da KI ma direttamente KI*Ts!

— è una struttura che serve a contenere tutti i parametri del regolatore PI (Kp, KI, limite di saturazione)
È del data type x reg param!

Wake-up →

Quando inizio l'algoritmo si mette nello stato di error + uscire dal quale si deve schiacciare il tasto restart

restart = 1 va nel wake-up



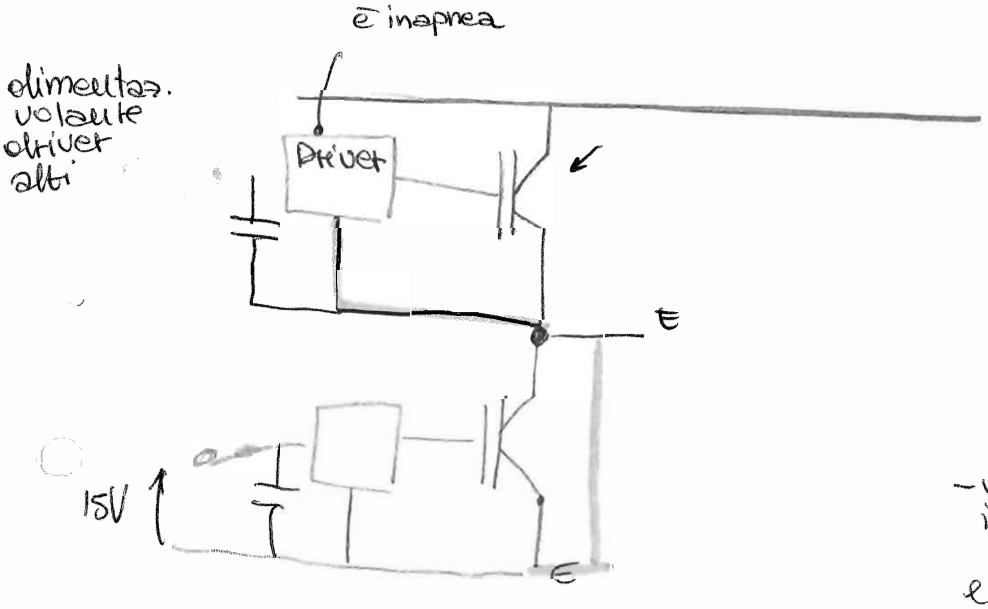
Nu Wake up fa operazioni automatiche

- fa il reset delle variabili (reset var. controll)
- offset delle correnti (misuro le correnti senza modulare (per un certo # di cicli PWM misuro le correnti, se la corrente è nulla leggo valori ≠ 0 → ho l'offset. Faccio 64 misure di accumulo (accumulo i 64 campioni della offset) e divido x 64 $\frac{\sum_{64} \text{offset}}{64} = \text{offset}$ corrente a!

Se l'offset è 0 misuro sempre zero. Lo si fa 64 volte x mediare il rumore

- Bootstrap

→ è un modo economico x alimentare gli IGBT lato alto



I driver lato basso sono facili da alimentare perché è stesso riferimento

Questo modulo per alimentarlo posto
- mette una alimentazione isolata SPSupply switching

e balla su e giù con la modulazione

```
/*-------------------------------------------------------------------------*/
/* File: DC_motor_ctrl_Va.c                                                */
/*                                                                         */
/* Description:                                                            */
/*                                                                         */
/*-------------------------------------------------------------------------*/
/* Motor data:                                                             */
/* - MotorData_DC.h                                                        */
/*-------------------------------------------------------------------------*/
/* Date:    01.03.2012                                                     */
/* Author:  GP                                                             */
/*-------------------------------------------------------------------------*/
/* Status:  tested on dspace                                               */
/* dspace reminders:                                                       */
/* 1. add State = ERROR in mdlInitializeConditions                         */
/* 2. add if (State != DRIVE_INIT) in iabcs calculation                    */
/*-------------------------------------------------------------------------*/

#define S_FUNCTION_NAME DC_motor_ctrl_Va
#define S_FUNCTION_LEVEL 2

#define U(element) (*uPtrs[element])

#include "simstruc.h"
#include <math.h>

#define DC      1

#include "include\User_data_types.h"
#include "include\Constants.h"
#include "include\Variables.h"
#include "include\User_Macros.h"

#ifdef DC
#include "include\MotorData_DC.h"
#define MOTOR   DC
#endif

#include "include\MotorControl.h"

#define NINPUTS     11
#define NOUTPUTS    13

#define CTRL_TYPE   (mxGetPr(ssGetSFcnParam(S,0))[0])   /*  control type - not USED */
#define ACCEL       (mxGetPr(ssGetSFcnParam(S,1))[0])   /*  speed ramp (rpm/s)      */
#define KPF             (mxGetPr(ssGetSFcnParam(S,2))[0])   /*  kp - field control
*/
#define KIF             (mxGetPr(ssGetSFcnParam(S,3))[0])   /*  ki - field control
*/
#define KPA             (mxGetPr(ssGetSFcnParam(S,4))[0])   /*  kp - armature control
*/
#define KIA             (mxGetPr(ssGetSFcnParam(S,5))[0])   /*  ki - armature control
*/
#define KPW             (mxGetPr(ssGetSFcnParam(S,6))[0])   /*  kp - speed control
*/
```

prende simstruct.h e lometto in questo punto di codice

incluso pezzi di codice definiti in altre parti, librerie

intestazione

includo i dati del motore

function che servono x il controllo

definisco # input output

```
    // copy parameters into variables
    accel = ACCEL * rpm2rad * Ts;    // NEVER ADJUST SCALE FACTORS IN DSP
IMPLEMENTATION!!!

    CrtRegParams_f.kp   = KPF;
    CrtRegParams_f.ki   = KIF*Ts;
    CrtRegParams_a.kp   = KPA;
    CrtRegParams_a.ki   = KIA*Ts;
    SpeedRegParams.kp   = KPW;
    SpeedRegParams.ki   = KIW*Ts;

    input.ch0   =U(0);
    input.ch1   =U(1);
    input.ch2   =U(2);

    if (State != WAKE_UP)
    {
        iabcs.a     =input.ch0-offset_in.ch0;
        iabcs.b     =input.ch1-offset_in.ch1;
        iabcs.c     =input.ch2-offset_in.ch2;
    }

    idc          = U(3);
    vdc          = U(4);
    omega_m      = rpm2rad * U(5);
    omega_ref_in = rpm2rad * U(6);

    // dSPACE CONTROL PANEL (Go - Err - Restart)
    if (State == ERROR)
        Restart     = U(8);     // control panel

    if (State == STOP)
        Go          = U(9);     // control panel
    else
        Go = 0;

    Err          = U(10);   // control panel

    //Filter for the speed
    omega_m_filt = Filter(omega_m,omega_m_filt,Kfilt_25);
    //Filter for the dc link voltage
    vdc_filt = Filter(vdc,vdc_filt,Kfilt_25);

    //Current protection
    CurrentProtection();

    // State machine (start-up buttons: Reset - Err - Go)
    switch(State)
{
    // Operating states:
    // ERROR: error state (default)
    // WAKE_UP: preliminary operations, after RESTART
    // STOP: everything ok, switch zero voltage
    // START: motor control
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 95 di 277

```
        break;


    case STOP:

        pwm_stop = 0.0;
        InitVarCtrl();

        //H-brodge modulates 50-50
        duty_abc.a=0.5;
        duty_abc.b=0.5;
        // Excitation chopper modulates ZERO
        duty_abc.c=0.0;

        if (Go > ONE_HALF)
            State = START;

        break;

    case START:

        /*-------------------------*/
        // FIELD CURRENT CONTROL
        // Ignore everything during excitation start-up
        if_ref = U(7);
        if (start_counter<START_FLUX)
        {
            omega_ref_ramp = 0.;
            va_ref = 0.;
            start_counter++;
        }

        if (vdc>0.25)
        vdc_inv = 1/vdc;
        else
            // avoid division by zero
            vdc_inv=0;

        //Error computation
        if_error = if_ref - iabcs.c;
        vc = CrtRegParams_f.kp * if_error;
        if (vc>vdc)
            vc=vdc;

        //Limit of the integral part
        int_lim = vdc - vc;

        //Integral part
        integral_f+=CrtRegParams_f.ki*if_error;

        //Saturation of the integral part
        if (integral_f>int_lim)
            integral_f=int_lim;

        //Output computation
```