Interface C - encallador

1 MODEL DE MEMÒRIA

En els programes d'alt nivell s'ha de definir la forma en que l'executable s'instal·larà en la memòria de l'ordinador. Això és el que s'anomena model de memòria i el programador l'ha de definir. Les paraules clau en C per a definir el model de memòria són: TINY, SMALL, MEDIUM, LARGE i HUGE. Els segments que utilitzen cada un dels models de memòria es mostren a la següent figura.

Model de memòria	Segments de codi	Segments de dades
TINY/SMALL	1	1
MEDIUM	Més d'un	1
COMPACT	1	Més d'un
LARGE/HUGE	Més d'un	Més d'un

Figura 5. Models de memòria en C.

Aquests models defineixen el nombre de segments de codi i de dades que utilitzarà el programa i per tant, els mòduls escrits en encallador hauran d'estar d'acord amb aquests models. Des del punt de vista de l'encallador els models TINY i SMALL són equivalents, així com també els models LARGE i HUGE. La directiva de l'encallador que permet definir el model de memòria és:

. model <model de memòria>

.model small

També s'ha de saber que el compilador de C afegeix el caràcter "_" a totes les referències externes d'un símbol definit en el mòdul del programa en C. És a dir, que qualsevol variable definida com per exemple *num*, la seva traducció pel compilador serà *_num*.

L'encallador TASM permet igualar etiquetes de forma automàtica sense haver de posar el caràcter "_" davant de les variables escrivint la lletra C al definir el model de memòria.

```
.model small, C
```

2 DIRECTIVES DE SEGMENT

L'encallador reconeix les següents directives per a obrir segment:

.DATA pel segment de dades.
.STACK pel segment de pila.
.CODE pel segment de codi.

A més a més, aquestes directives tanquen el segment anterior i per tant no és necessari la directiva ENDS per a tancar cada segment.

Per a llegir l'adreça d'inici d'un segment definit d'aquesta manera, s'haurà d'utilitzar una instrucció de moviment i utilitzar els símbols @DATA i @CODE.

```
mov   ax, @DATA
mov   ds, ax
```

3 INSERCIÓ DE CODI ENCALLADOR EN UN PROGRAMA EN C

És la forma més senzilla d'incloure instruccions en un mòdul de C. Per fer-ho, només s'ha de posar al davant de la instrucció la paraula clau **asm**. Cal tenir el programa TASM carregat en el directori de treball ja que el compilador l'invoca quan es troba una sentència d'aquest tipus.

```
#include <stdio.h>

void main(void)
{
        int compt1=10, compt2=22, res=0;

        asm   mov ax, compt1;
        asm   mov bx, compt2;
        asm   add ax,bx;
        asm   mov res,ax;

        printf("La suma dels comptadors és: %d", res);
}
```

No és recomanable utilitzar la programació en codi inserit quan cal resoldre tot un procediment d'una certa extensió en encallador. És preferible utilitzar aquest mètode quan es necessitin incloure algunes instruccions aïllades en el programa.

4 FUNCIONS EN ENCALLADOR UTILITZANT VARIABLES GLOBALS

Les variables globals són molt fàcil d'utilitzar en la programació mixta d'alt nivell i baix nivell, ja que són accessibles des de qualsevol mòdul amb independència d'on s'han definit.

En C, les variables definides fora de les funcions són globals i, en encallador totes les variables són globals, per tant es poden definir variables en un o altre mòdul i utilitzar-les indistintament sempre i quan s'especifiquin les seves propietats públiques o externes per a fer-les accessibles als altres mòduls.

En C no és necessari definir les variables com a públiques per a que siguin accessibles per altres mòduls. En canvi, si ha d'utilitzar una variable o funció definida en un altre mòdul, aquesta s'ha de definir com a externa amb la paraula clau EXTERN.

```
extern void interrupt far rutina_nova(void);
```

El distintiu **far** és per a models que utilitzen més d'un segment de codi de manera que aquesta funció, que en aquest cas correspon a una rutina d'interrupció, pugui ésser utilitzada des de qualsevol segment.

Per altre part, en el mòdul en encallador s'ha de definir el procediment com a públic i les variables que s'utilitzin del mòdul en C, declarar-les com a externes. Per això s'utilitzen les directives PUBLIC i EXTRN respectivament.

```
.data
extrn   compt1:word
extrn   compt2:word
extrn  segons:word, minuts:word, hores:word

.code
public rutina_nova
```

5 OBTENCIÓ D'UN MÒDUL EXECUTABLE

Quan es treballa amb alt i baix nivel s'han de seguir una sèrie de pasos per a obtenir el resultat desitjat. Aquestes fases o pasos són:

„ Obtenir els mòduls font tant de l'encallador com del C.

„ A continuació s'ha d'obtenir el mòdul objecte del programa encallador mitjançant el programa TASM. En aquest cas s'ha d'utilitzar l'opció **/mx** per que l'encallador distingueixi les etiquetes en majúscules i en minúscules. Si s'inclou l'opció **/zi**, es genera el fitxer de símbols que es podran utilitzar en el depurador.

TASM <nom de l'arxiu>,,,**/mx/zi**;

„ Per últim, i ja dintre del turbo C, es genera un PROJECT en el que s'han d'afegir els items necessaris. Aquests items són: el mòdul font en C i el mòdul objecte en encallador.

A continuació es mostra un exemple on apareixen els mòduls font en C i en encallador.

```
;=========================================
;      Rutina per a rellotge C/encallador
;=========================================
        .model medium, C

.data
extrn   cont1:word
extrn   cont2:word
extrn segundos:word, minutos:word, horas:word

.code
        public rutina_nova

rutina_nova     proc far
        push ds
        mov ax,@data
        mov ds,ax
        .
; Aquí s´'ha d'escriure el programa
        .
        pop ds
        iret
rutina_nova     endp

        end
```

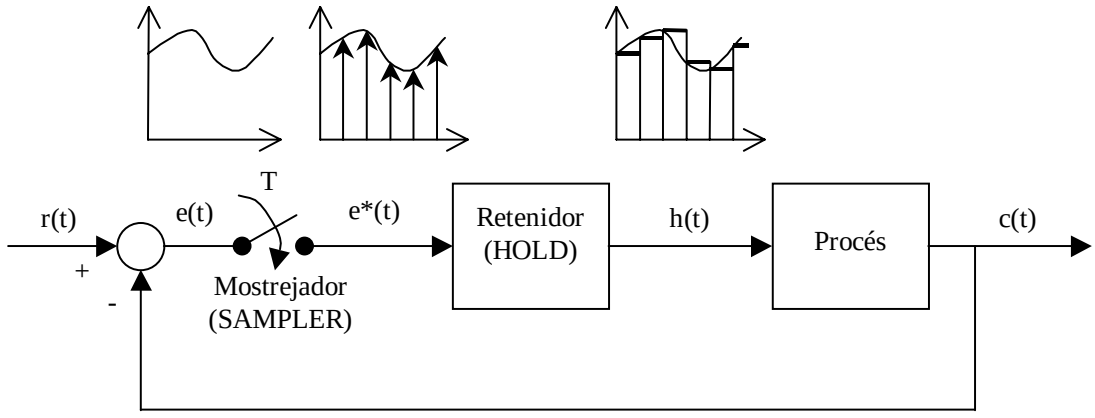
```
/*Rellotge utilitzant la interrupció IRQ0 del 8259 mestre */

#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <graphics.h>
#define         REG_MASCARES     0x21
#define         INTR                    0x1C

void interrupt(*rutina_ant)(void);
extern void interrupt far rutina_nova(void);
int compt1=0, compt2=0; int hores, minuts, segons;

void main(void)
{
         int mascara;
        printf("Introdueixi l'hora inicial\n\n");
        printf("HORES:");
        scanf("%d",&hores);
        printf("\nMINUTS:");
        scanf("%d",&minuts);
        printf("\nSEGONS:");
        scanf("%d",&segons);

        rutina_ant=getvect(INTR);
        mascara=inportb(REG_MASCARES);
        mascara=mascara|0x01;
        outportb(REG_MASCARAS, mascara);
        setvect(INTR, rutina_nova);
        mascara=inportb(REG_MASCARAS);
        mascara=mascara&0xFE;
        outportb(REG_MASCARAS, mascara);

        textmode(2);
        textmode(3);                                    /* Color 80 columnes */
        window(1,1,12,2);                               /* Defineix finestra */
        textattr(WHITE|BLUE*16);                        /* Text blau fons blanc */
        clrscr();
        for(;;){
                gotoxy(1,1);                            /* Posicionar cursor */
cprintf("%d : %d : %d    \r",hores,minuts,segons);
        }
}
```

Sistemes digitals (programació en C)

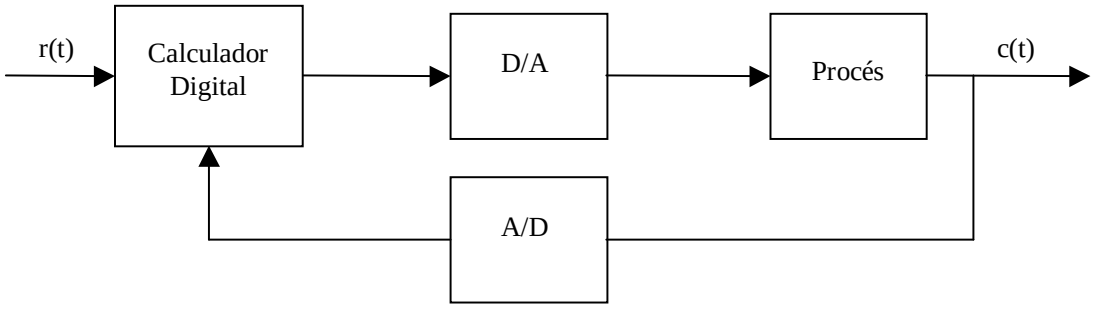
1 INTRODUCCIÓ

Durant molts anys tot l'anàlisi i el processament de senyals en forma elèctrica es realitzava utilitzant equips electrònics analògics. Originalment, el processament analògic era més adequat en termes de cost, tamany i velocitat. Si més no, la situació actual és exactament la contrària donada l'existència de dispositius VLSI per a funcions de processament específiques, i a la gran quantitat de programari desenvolupat per al seu us en computadors personals. Tot això, ha propiciat un desplazament de les tècniques analògiques de control en benefici dels computadors digitals.

En la següent figura es mostra un sistema de control digital típic on es veu que el senyals que intervenen nosón només analògics sino que també hi ha de discrets (digitals).



(a) SISTEMA MOSTREJAT



(b) SISTEMA DE COMANDAMENT DIGITAL

Figura 6. Sistemes discrets.

Evidentment, el calculador digital pot estar implementat per diversos dispositius, com poden ésser: un microprocessador de propòsit general, un de propòsit específic, un microcontrolador, un microprocessador DSP, etc.

Ens centrarem aquí en els sistemas digitals en que el calculador digital és un microprocessador depropòsit general. És a dir, un computador personal que juntament amb una targeta d'adquisició constitueixen els requeriments bàsics per al control per ordinador.

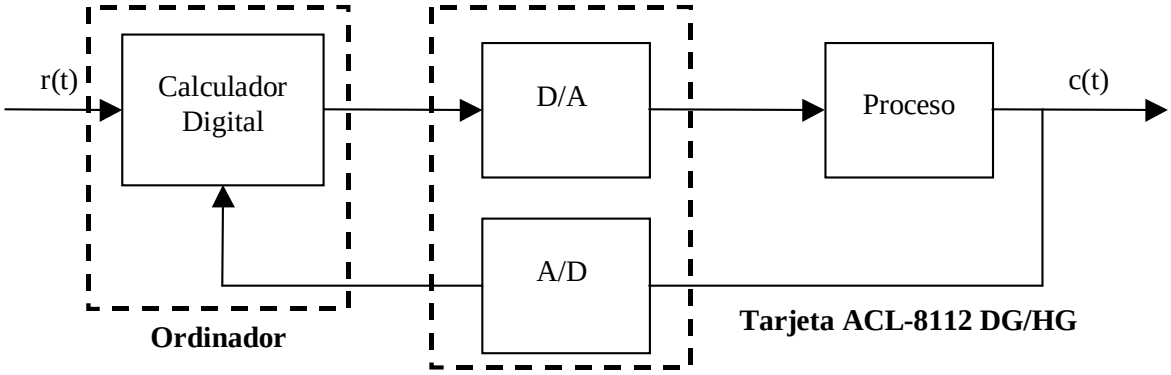


Figura 7. Sistema ordinador + targeta ACL-8112 DG/HG.

La targeta d'adquisició ACL-8112 DG/HG permet tres tipus de transferència de dades:

- Entrada/sortida per programa.
- Entrada/sortida per interrupció.
- Entrada/sortida per DMA.

La transferència per DMA és convenient fer-la mitjançant les libreries de la pròpia targeta degut a la complicació de programar el controlador de DMA. Els altres mètodes es poden poden programar en C amb facilitat.

La transferència per programa és molt simple. Aquest tipus de transferència s'utilitza quan la precisió en la freqüència de mostreig no és important, si més no, és el mètode més senzill d'implementar ja que no requereix la programació de cap circuit extern a la targeta d'adquisició.

Per últim, la transferència per interrupció requereix de la programació total o parcial de dos circuits. Un d'ells es troba a la pròpia targeta (timer 8253/4) mentres que l'altre es troba en l'ordinador (controlador d'interrupcions 8259). Amb el primer es programa la freqüència de mostreig per a l'adquisició. El segon s'ha de programar parcialment per a desemmascarar la interrupció utilitzada i per a

generar finals d'interrupcions necessaris pel correcte funcionament de les interrupcions. A continuació es mostra un exemple d'adquisició per interrupció.

```
/*********************************************************/
/*              TRANSFERÈNCIA PER INTERRUPCIÓ                */
/*********************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <dos.h>

#define BASE                    0x220                   /* Registres de la targeta
*/
#define COMPTADOR0      BASE+0                  /* ACL-8112 DG/HG */
#define COMPTADOR1      BASE+1
#define COMPTADOR2      BASE+2
#define PAL_CONTR       BASE+3
#define ADCLB           BASE+4
#define CH1DACLB        BASE+4
#define ADCHB           BASE+5
#define CH1DACHB        BASE+5
#define CH2DACLB        BASE+6
#define CH2DACHB        BASE+7
#define CINTREQ         BASE+8
#define CRANGOADC       BASE+9
#define CCANMUX         BASE+10
#define COMPT_MODE      BASE+11
#define SOFT_TRIG       BASE+12
#define REG_MASCARES  0x21                      /* OCW1 del 8259 */
#define REG_COMAND2   0x20                      /* OCW2 del 8259 */
#define EOI_N_E         0x20                    /* EOI no específic */
#define canal0_se       0x10                    /* Canal 0 */
#define unip_10v        4                       /* Tensió unipolar de 10 V
*/
#define INTR            0xF                     /* Interrupció IRQ7 del
8259 */

void interrupt(*pint)();

/* Declaració de variables globals */

void interrupt adquisicio(void)
{
        /* Declaració de variables locals */

        /* Lectura d'una mostra */

        /* Tractament de la mostra adquirida */
```

```
        outportb(CINTREQ, 0);                   /* Esborrar flag interrupció */
        outportb(REG_COMAND2, EOI_N_E);         /* EOI no específic */

}

void main(void)
{
        /* Declaració de variables locals */

        /* Entrada de dades */

        /* Seleccionar canal, rang de tensions, etc. */

        pint=getvect(INTR);                     /* Salva vector antic */
        setvect(INTR, adquisicio);              /* Instal·la nou vector IRQ7 */

                /* Programar els comptadors 1 i 2 per a la freqüència de mostreig
desitjada */

        /* Desemmascarar IRQ7 */

        outportb(REG_COMAND2, 0x20);            /* EOI no específic */
        outportb(COMPT_MODE, 0x6);              /* Mode de trigger del timer */
                                                /* transferència interrupció */
        outportb(CINTREQ, 0);                   /* Esborrar flag interrupció */

        /* Possible visualització de la informació */

        /* Control de final de programa */

        outportb(COMPT_MODE, 0);                /* Desactivació de trigger */
        setvect(INTR, pint);                    /* Reposar vector antic */
}
```