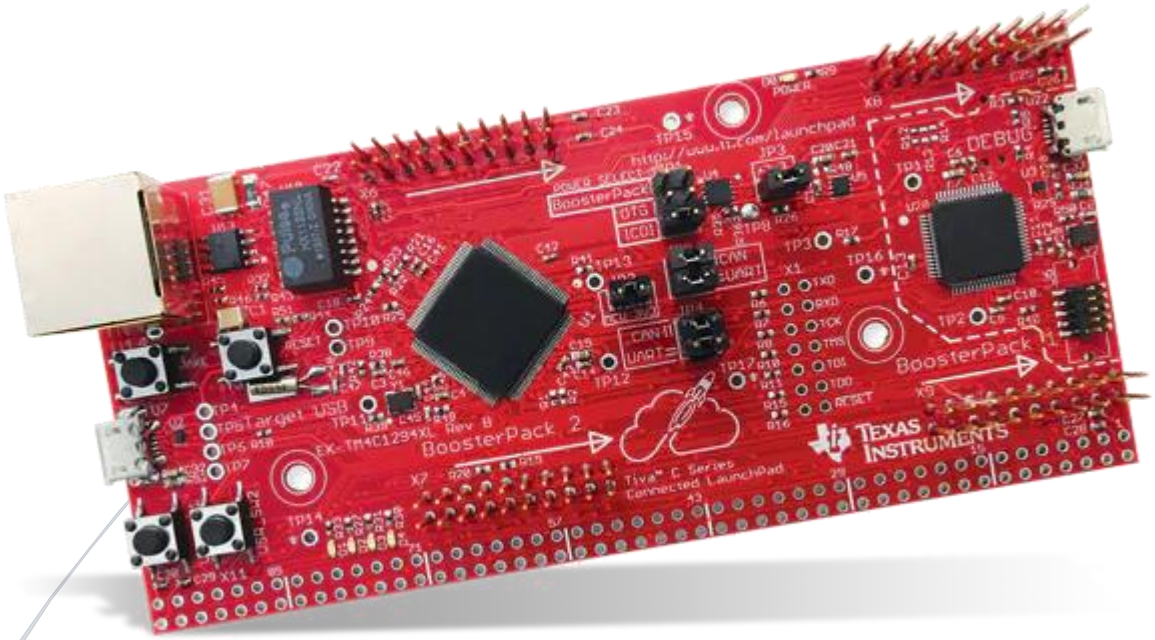
4ºGIERM

Práctica 3 SEPA:

Máquina de vending simple



Daniel Peinado Ramírez, Francisco Javier Román Escorza
SISTEMAS ELECTRÓNICOS PARA LA AUTOMATIZACIÓN

1. Primer Ejercicio: Diseño de una interfaz muy simple.

Partiendo del ejemplo suministrado, realizar un programa que interactúe con los botones y los leds de la placa. Para ello, se desea que, cuando se pulse un botón de la placa, aparezca en la pantalla la frase "HAS PULSADO B1" (o B2, según el caso) mientras esté pulsado el botón. Igualmente, se dispondrán en la placa 4 botones, que se denominarán L1, L2, L3, L4, de manera que al pulsarlos se encenderán los leds de la placa, mientras se mantengan pulsados. Los colores del fondo, letras, y botones, se dejan a elección del alumnado, pero se sugiere una cosa como esta. Es conveniente realizar el programa como una máquina de estados, en la que las entradas serán la pulsación de botones en la placa o en la pantalla, permaneciendo en reposo si no hay que hacer nada, y refrescando todo el sistema cada 50ms.

En el primer ejercicio nos encargaremos de realizar una interfaz en la que al pulsar los botones B1 y B2 de la placa, aparezca representada en la pantalla dicha acción con una cadena de texto escrita en el recuadro blanco, así como el encendido de los leds al pulsar su botón correspondiente.

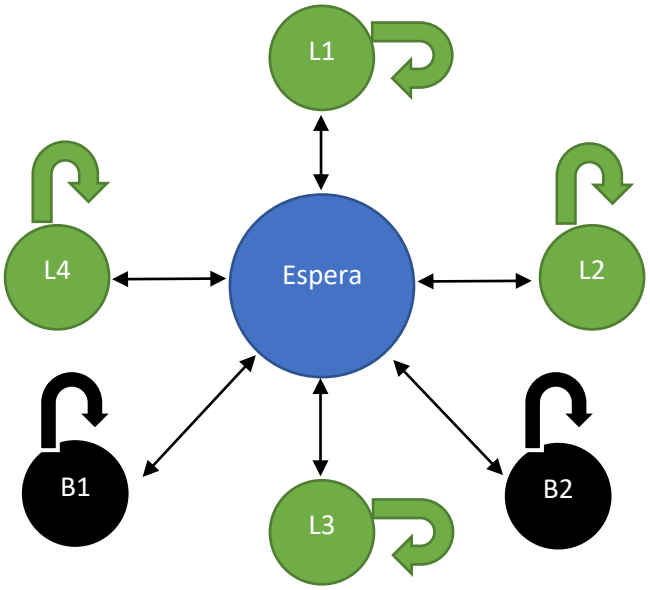
En primer lugar, necesitamos definir los pines que vamos a utilizar: los leds, el timer y los botones junto con la configuración de la pantalla FT800 en la ranura del Booster Pack 1. Para ir representando gráficamente elementos en la pantalla, recurrimos a las funciones de las librerías FT800. Dentro de ellas, podemos destacar algunas de las más importantes: Lee_pantalla para comenzar a utilizarla, ComFGcolor para ir alternando entre distintos colores, ComRect para dibujar rectángulos tanto rellenos como no, ComTXT para imprimir cadenas de texto, o Dibuja() para dibujar en la pantalla todas las órdenes que se hayan dado anteriormente referentes a la misma, entre otras.

La máquina de estados para el ejercicio se plantearía de la siguiente forma: Tras la ejecución del programa, comenzamos en el estado de Espera, y con una tasa de refresco de 50 ms comprobaremos si se han realizado cambios de estado.

- En el caso de pulsar un botón táctil, encenderemos el led correspondiente a dicho botón (D4 al L1, D3 al L2, D2 al L3 y D1 al L4, para que se correspondan con la orientación de la pantalla). La comprobación para saber si los botones están pulsados se realiza únicamente con sentencias *if* y nunca se debe utilizar *else if* debido a que de esta forma comprueba siempre todos los botones de la pantalla y por ello los dibuja correctamente.
- En el caso de pulsar un botón de la placa, no hay ningún cambio en los botones de la pantalla de seguir los botones de la pantalla, escribiremos con la función ComTXT el texto correspondiente dentro del recuadro según se haya pulsado B1 o B2.

En el caso de mantener pulsado cualquiera de los botones, nos mantendremos infinitamente en ese estado hasta que lo soltemos. De esta manera, vamos alternando entre el estado de reposo y cualquier otro sucesivamente.

Diagrama de estados para el primer ejercicio:



- ***Código:***

```
#include <stdint.h>
#include <stdbool.h>
#include "driverlib2.h"
//Librería para el uso de la pantalla
#include "FT800_TIVA.h"

//Definimos cada uno de los colores en que vamos a utilizar, en formato RGB
#define rojo 163,47,41
#define amarillo 255,255,153
#define gris 128,128,128
#define verde 118,196,49
#define verde2 153,255,51
#define negro 0,0,0
#define blanco 255,255,255
//Hacemos una división del ancho y del alto de la pantalla, de forma que multiplicando
//dichas divisiones por constantes, podamos establecer coordenadas dentro de la "cuadrícula" de la pantalla
#define divHor HSIZE/26
#define divVer VSIZE/12

//Definición de cada una de las instancias por las que pasará la máquina de estados, en función
//de que estemos en estado de espera, pulsando un botón de la pantalla o de la placa
#define espera 0
#define l1 1
#define l2 2
#define l3 3
#define l4 4
#define b1 5
#define b2 6
```

```
//Definición para los estados de comprobación de la pulsación de los botones
#define B2_OFF GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0)
#define B2_ON !(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0))
#define B1_OFF GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_1)
#define B1_ON !(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_1))

//Definición de los estados encendido y apagado de los LEDs
#define L4_ON GPIOPinWrite(GPIO_PORTN_BASE,GPIO_PIN_1,GPIO_PIN_1)
#define L4_OFF GPIOPinWrite(GPIO_PORTN_BASE,GPIO_PIN_1,0)
#define L3_ON GPIOPinWrite(GPIO_PORTN_BASE,GPIO_PIN_0,GPIO_PIN_0)
#define L3_OFF GPIOPinWrite(GPIO_PORTN_BASE,GPIO_PIN_0,0)

#define L2_ON GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_4,GPIO_PIN_4)
#define L2_OFF GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_4,0)
#define L1_ON GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_0,GPIO_PIN_0)
#define L1_OFF GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_0,0)

//Definición de los 4 botones de la pantalla, con sus coordenadas en función
de las divisiones horizontal y vertical
#define dibL1 Boton((4)*divHor, VSIZE-4*divVer, 3*divHor, 2*divVer,27, "L1")
#define dibL2 Boton((5+4)*divHor, VSIZE-4*divVer, 3*divHor, 2*divVer,27,
"L2")
#define dibL3 Boton((10+4)*divHor, VSIZE-4*divVer, 3*divHor, 2*divVer,27,
"L3")
#define dibL4 Boton((15+4)*divHor, VSIZE-4*divVer, 3*divHor, 2*divVer,27,
"L4")

//DEFINICION PARA DIBUJO Definición para
#define alturaCuadro 20
#define largoCuadro 50

// ================================================================
// Function Declarations
// ================================================================
#define dword long
#define byte char

#define PosMin 750
#define PosMax 1000

#define XpMax 286
#define XpMin 224
#define YpMax 186
#define YpMin 54

unsigned int Yp=120, Xp=245;
// ================================================================
// Variable Declarations
// ================================================================

char chipid = 0;                        // Holds value of Chip ID read from
the FT800

unsigned long cmdBufferRd = 0x00000000;         // Store the value read from
the REG_CMD_READ register
unsigned long cmdBufferWr = 0x00000000;         // Store the value read from
the REG_CMD_WRITE register
unsigned int t=0;
```

```
//
#########################################################################
#################################
// User Application - Initialization of MCU / FT800 / Display
//
#########################################################################
#################################

unsigned long POSX, POSY, BufferXY;
unsigned long POSYANT=0;
unsigned int CMD_Offset = 0;
unsigned long REG_TT[6];
const int32_t REG_CAL[6]={21696,-78,-614558,498,-17021,15755638};
const int32_t REG_CAL5[6]={32146, -1428, -331110, -40, -18930, 18321010};


#define NUM_SSI_DATA  3

int RELOJ, Flag_ints, i;

int modo = espera; //Control de máquina de estados

//Función del SLEEP fake, utilizar en caso de que dé problemas al usar el
debugger
//#define SLEEP SysCtlSleep()
#define SLEEP SysCtlSleepFake()
void SysCtlSleepFake(void)
{
 while(!Flag_ints);
 Flag_ints=0;
}

int contador; //Variable que utlizamos en la rutina de interrupción del timer
para contar las veces que se activa
//Interrupción del Timer0
void IntTimer0(void)
{
    // Clear the timer interrupt
    TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    Flag_ints=1;
    contador ++;
}
int main(void)
{

    //Habilitar los periféricos implicados en el eje: GPIOF, GPIOJ, GPION
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOJ);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPION);

    //Habilitar los periféricos implicados en el eje en el Sleep: GPIOF,
GPIOJ, GPION
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_GPIOF);
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_GPIOJ);
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_GPION);

    //Definir tipo de pines, los botones como entradas y los leds como salidas
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_0 |GPIO_PIN_4); //F0 y
F4: salidas
```

```
    GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_0 |GPIO_PIN_1); //N0 y
N1: salidas
    GPIOPinTypeGPIOInput(GPIO_PORTJ_BASE, GPIO_PIN_0|GPIO_PIN_1);    //J0 y
J1: entradas

GPIOPadConfigSet(GPIO_PORTJ_BASE,GPIO_PIN_0|GPIO_PIN_1,GPIO_STRENGTH_2MA,GP
IO_PIN_TYPE_STD_WPU); //Pullup en J0 y J1

    GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, 0);
    GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, 0);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, 0);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0);



    RELOJ=SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ    |    SYSCTL_OSC_MAIN    |
SYSCTL_USE_PLL | SYSCTL_CFG_VCO_480), 120000000);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);       //Habilita T0
    TimerClockSourceSet(TIMER0_BASE, TIMER_CLOCK_SYSTEM);   //T0 a 120MHz
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);      //T0 periodico y
conjunto (32b)
    TimerLoadSet(TIMER0_BASE, TIMER_A, RELOJ/20 -1);
    TimerIntRegister(TIMER0_BASE,TIMER_A,IntTimer0);
    IntEnable(INT_TIMER0A); //Habilitar las interrupciones globales de los
timers
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);      // Habilitar las
interrupciones de timeout
    IntMasterEnable();  //Habilitacion global de interrupciones
    TimerEnable(TIMER0_BASE, TIMER_A);  //Habilitar Timer0
    HAL_Init_SPI(1, RELOJ);  //Boosterpack a usar, Velocidad del MC
    Inicia_pantalla();       //Arranque de la pantalla

    // Note: Keep SPI below 11MHz here

    //
=========================================================================
    // Delay before we begin to display anything
    //
=========================================================================

    SysCtlDelay(RELOJ/3);

    //
=========================================================================
=====================================
    // PANTALLA INICIAL
    //
=========================================================================
=====================================
    //Primera orden necesaria para el uso de la pantalla
    Lee_pantalla();
    //Pantalla en blanco
    Nueva_pantalla(blanco);
    //Cambio de color a rojo
    ComColor(rojo);
    //Dibujar el
    ComRect(divHor, divVer, HSIZE-divHor, VSIZE-divVer, true);
    ComLineWidth(10);
```

```
    //Pintamos en blanco el texto y el reciadro blanco inferior
    ComColor(blanco);
    ComTXT(HSIZE/2,1.8*divVer, 29, OPT_CENTERX,"BOTONES:");
    ComRect(4*divHor, 5.5*divVer, HSIZE-4*divHor, 3.5*divVer, true);

    //Impresión de cada uno de los botones en verde, equiespaciados y
respetando las proporciones
    //Lo repetimos 4 veces, cambiando el número en la cadena
    ComFgcolor(verde);
    int j;
    char textBot[]="L0";
    for(j=0;  j<4; j++)
    {
        textBot[1]++;
        Boton((j*5+4)*divHor,    VSIZE-4*divVer,    3*divHor,    2*divVer,27,
textBot);
    }

    Dibuja();
    //Espera_pant();

    //Calibración predeterminada de la pantalla
    int i;
#ifdef VM800B35
    for(i=0;i<6;i++)      Esc_Reg(REG_TOUCH_TRANSFORM_A+4*i, REG_CAL[i]);
#endif
#ifdef VM800B50
    for(i=0;i<6;i++)    Esc_Reg(REG_TOUCH_TRANSFORM_A+4*i, REG_CAL5[i]);
#endif

    while(1){
        //Ciclo de ejecución cada 50 ms
        SLEEP;

        //Definimos como cadena vacía el hueco sobre el que va a ir escrito
el texto
        char texto[]="               ";
        //Creamos una nueva pantalla, de color blanco
        Nueva_pantalla(blanco);
        //Pintamos un rectángulo rojo
        ComColor(rojo);
        ComRect(divHor, divVer, HSIZE-divHor, VSIZE-divVer, true);
        ComLineWidth(10);
        //Cambio de color a blanco, e imprimimos la pantalla botones centrada
en el eje X
        ComColor(blanco);
        ComTXT(HSIZE/2,1.8*divVer, 29, OPT_CENTERX,"BOTONES:");
        ComRect(4*divHor, 5.5*divVer, HSIZE-4*divHor, 3.5*divVer, true);
        ComFgcolor(verde);
        //Máquina de estados:
        //Dentro del modo espera, cuando no se pulsa ni la pantalla ni los
botones:
        if(modo==espera)
        {                       //Tenemos los leds apagados, y si se pulsa un
determinado botón se pasa el estado correspondiente
            if (B1_ON)  modo=b1;
            if (B2_ON) modo=b2;
            if (dibL1) modo=l1;
            if (dibL2) modo=l2;
```

```
            if (dibL3) modo=l3;
            if (dibL4) modo=l4;
            L1_OFF;
            L2_OFF;
            L3_OFF;
            L4_OFF;
        }
        else if (modo==l1)  //Si pulsamos el botón L1, pasamos al estado l1
        {                   //Redibujamos el resto de botones y encendemos
el primer led. Al dejar de pulsar, volvemos al estado de espera
            dibL2;
            dibL3;
            dibL4;
            L1_ON;
            if(!dibL1) modo=espera;
        }
        else if (modo==l2)  //Si pulsamos el botón L2, pasamos al estado l2
        {                   //Redibujamos el resto de botones y encendemos
el segundo led. Al dejar de pulsar, volvemos al estado de espera
            dibL1;
            dibL3;
            dibL4;
            L2_ON;
            if(!dibL2) modo=espera;
        }
        else if (modo==l3)  //Si pulsamos el botón L3, pasamos al estado l3
        {                   //Redibujamos el resto de botones y encendemos
el tercer led. Al dejar de pulsar, volvemos al estado de espera
            dibL1;
            dibL2;
            dibL4;
            L3_ON;
            if(!dibL3) modo=espera;
        }
        else if (modo==l4)  //Si pulsamos el botón L4, pasamos al estado l4
        {                   //Redibujamos el resto de botones y encendemos
el cuarto led. Al dejar de pulsar, volvemos al estado de espera
            dibL1;
            dibL2;
            dibL3;
            L4_ON;
            if(!dibL4) modo=espera;
        }
        else if (modo==b1)  //Si pulsamos el botón B1 de la placa, pasamos
al estado b1
        {                 //Escribimos el texto centrado dentro del recuadro
blanco y redibujamos los botones. Al dejar de pulsar volvemos al reposo.
            ComColor(negro);
            ComTXT(HSIZE/2, 4.5*divVer,29, OPT_CENTER, "HAS PULSADO B1");
            ComColor(blanco);
            dibL1;
            dibL2;
            dibL3;
            dibL4;
            if (B1_OFF) modo=espera;

        }
        else if (modo==b2)  //Si pulsamos el botón B1 de la placa, pasamos
al estado b1
```

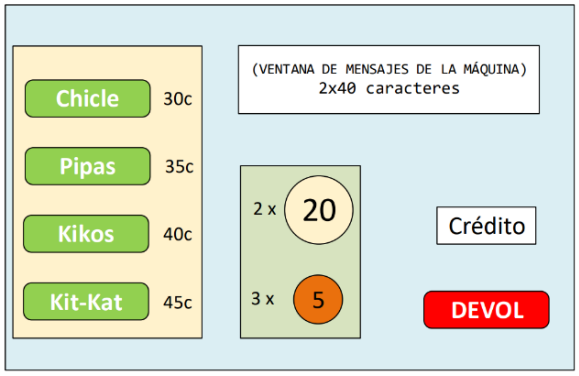
```
        {                   //Escribimos el texto centrado dentro del recuadro
blanco y redibujamos los botones. Al dejar de pulsar volvemos al reposo.

            ComColor(negro);
            ComTXT(HSIZE/2, 4.5*divVer,29, OPT_CENTER, "HAS PULSADO B2");
            ComColor(blanco);
            dibL1;
            dibL2;
            dibL3;
            dibL4;
            if (B2_OFF) modo=espera;
        }


        Dibuja();
    }
}
```

2. Segundo Ejercicio: Máquina de vending simple.

Se desea diseñar una máquina de vending sencilla, que disponga de 4 productos (a elegir por el alumno). El precio de los productos será de 30, 35, 40 y 45 céntimos. La máquina aceptará solamente monedas de 5 y 20 céntimos. El proceso de la máquina será el siguiente:



- ***Cada vez que se pulse el botón 1 será como si se introdujese una moneda de 5 céntimos.***
- ***Cada vez que se pulse el botón 2 será como si se introdujese una moneda de 20 céntimos.***
- ***Cada vez que se introduzca una moneda, se mostrará en la pantalla el mensaje "Moneda de xxcts introducida", durante 1s, y la suma total de dinero acumulado. A su vez, se reflejará en el cuadro central el número de monedas introducidas de cada tipo, y en el cuadro de 'Crédito' el dinero total introducido.***
- ***Cuando se pulse el botón de selección del producto deseado, en caso de tener crédito suficiente, se dispensará el producto: o Se mostrará en la pantalla "Dispensando producto", o Se encenderá uno de los 4 leds de la placa, dependiendo del producto seleccionado, o Se activará un servo a la posición extrema durante 2s, o Se volverá el servo a la posición inicial, o Se mostrará en la pantalla el mensaje "Producto dispensado" durante otros 2s, y se apagará el led.***
- ***Una vez dispensado el producto, se calculará la vuelta (usando el mínimo número de monedas) y se devolverá el cambio. Para ello: o Se mostrará en la pantalla un mensaje "Devolución de XX céntimos", o Se mostrará igualmente en la pantalla un mensaje "XX monedas de 20c, YY monedas de 5c". Se reflejará igualmente en el cuadro central cuántas monedas de cada tipo se devolverán. o El estado se mantendrá 5s y se volverá al principio.***
- ***Si se selecciona un producto y no hay crédito bastante, se mostrará un mensaje de "No hay crédito suficiente" durante 2s y se volverá a la situación anterior.***

- ***Si se pulsa el botón de devolución del dinero, se hará el mismo proceso que en el caso de devolución de la vuelta.***

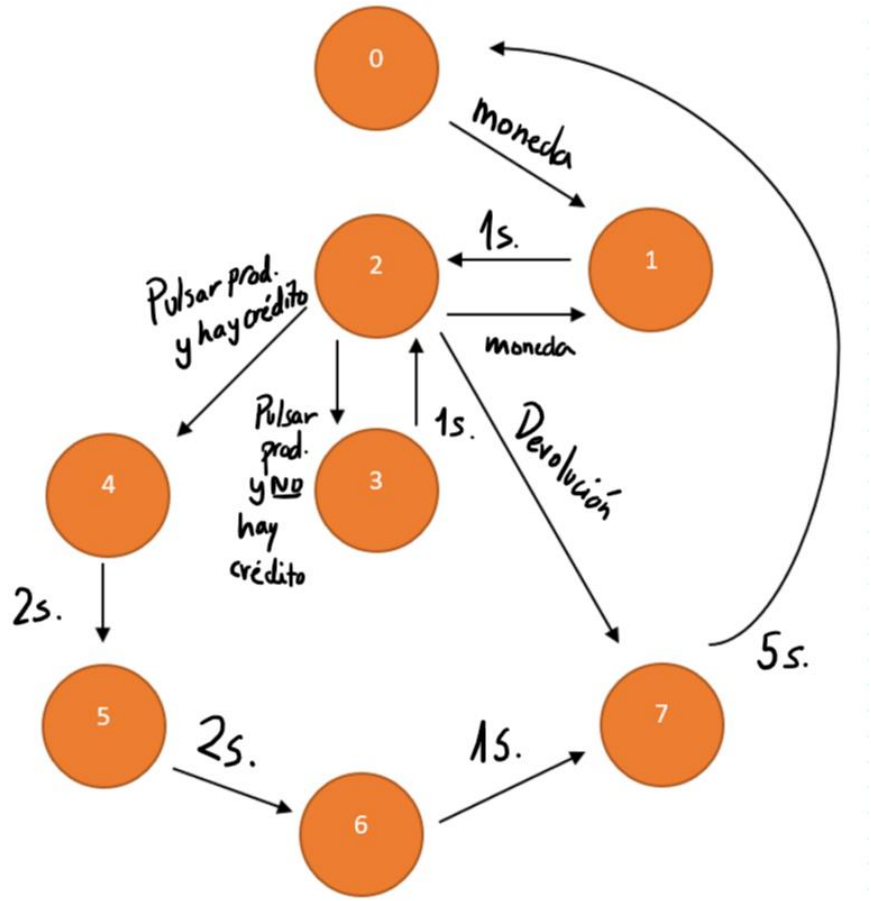
En el segundo ejercicio realizaremos una máquina dispensadora de productos. En este caso, usaremos los leds para indicar cuál de los 4 productos se ha elegido encendiendo su led correspondiente asociado, los botones B1 y B2 para introducir monedas de 20 o de 5 céntimos respectivamente, y el servomotor, con la misma configuración que en la segunda práctica, en el PIN S4, para simular el funcionamiento del dispensado. Por ello, recuperamos también la función *giro* con la que obteníamos el porcentaje de giro del rango operativo del servomotor. Utilizaremos la pantalla para interactuar con los botones de devolución así como para elegir el producto que se desee.

Volvemos a definir los pines necesarios: Los leds, los botones de la placa, el pin del servomotor (S4, como en la segunda práctica) y la pantalla configurada en la ranura del Booster Pack 1. Tras establecer divisiones HSIZE y VSIZE tal y como en el ejercicio anterior, disponemos una serie de rectángulos rellenos para simular la interfaz que se nos plantea en el enunciado. También nos encargamos de definir las cadenas de caracteres correspondientes a cada uno de los productos y de los mensajes que aparecerán por pantalla.

Dentro de la máquina de estados planteamos las siguientes posibilidades:

- **0: Inicio:** En el estado de inicio, se espera a que se pulse uno de los botones, B1 o B2, para introducir monedas de 5 o de 20 céntimos. Aumenta el contador correspondiente y mostramos dicha información por pantalla.
- **1: Moneda introducida:** Acumula la suma de crédito introducido, con contadores separados para cada tipo de moneda.
- **2: Contador de dinero:** Se produce una espera de 1 segundo y se preparan los mensajes correspondientes a la adición de crédito.
- **3: Crédito insuficiente:** Se produce cuando se elige un producto y no se ha introducido el crédito necesario para adquirirlo. En este punto se puede seguir introduciendo crédito hasta poder comprar alguno, o bien solicitar la devolución de todo el dinero introducido. Redirecciona al estado anterior.
- **4: Expulsión de producto 1:** Se corresponde a la compra de un producto. Se resta la cantidad de crédito al precio del producto seleccionado, se mueve el servomotor y se enciende el led correspondiente. Se produce también una espera de 2 segundos para pasar al siguiente estado.
- **5: Expulsión de producto 2:** Continuación del estado anterior, en el que devolvemos el servomotor a la posición inicial y se apagan todos los leds.
- **6: Calcular devolución:** Se realizan las operaciones necesarias para deducir el crédito que hay que devolver con el menor número de monedas posibles, es decir, priorizar la devolución de monedas de 20 céntimos antes que las de 5.
- **7: Devolución:** Devuelve la cantidad de crédito que no ha sido utilizado, ya sea porque se ha introducido más dinero del necesario para adquirir un producto o porque se haya solicitado la devolución. Realiza la espera de 5 segundos.

Diagrama de estados para el segundo ejercicio:



- ***Código:***

```
#include <stdio.h>
#include <stdint.h>
#include <stdbool.h>
#include "driverlib2.h"
#include "FT800_TIVA.h"

//Definición para los estados de comprobación de la pulsación de los botones
#define B2_OFF GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0)
#define B2_ON !(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0))
#define B1_OFF GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_1)
#define B1_ON !(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_1))

//Definición de los estados encendido y apagado de los LEDs
#define L3_ON GPIOPinWrite(GPIO_PORTN_BASE,GPIO_PIN_1,GPIO_PIN_1)
#define L3_OFF GPIOPinWrite(GPIO_PORTN_BASE,GPIO_PIN_1,0)
#define L2_ON GPIOPinWrite(GPIO_PORTN_BASE,GPIO_PIN_0,GPIO_PIN_0)
#define L2_OFF GPIOPinWrite(GPIO_PORTN_BASE,GPIO_PIN_0,0)

#define L1_ON GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_4,GPIO_PIN_4)
#define L1_OFF GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_4,0)
#define L0_ON GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_0,GPIO_PIN_0)
#define L0_OFF GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_0,0)

//Definimos cada uno de los colores en que vamos a utilizar, en formato RGB
```

```
#define rojo 163,47,41
#define amarillo 255,255,153
#define gris 128,128,128
#define verde 128,255,0
#define verde2 153,255,51
#define verdeClaro 191,255,179
#define negro 0,0,0
#define blanco 255,255,255
#define azulClaro 201,255,229
#define amarilloClaro 255,255,204
#define rojo2 255,0,0
#define naranja 230,115,0
//Hacemos una división del ancho y del alto de la pantalla, de forma que multiplicando
//dichas divisiones por constantes, podamos establecer coordenadas dentro de la "cuadrícula" de la pantalla
#define divHor HSIZE/20
#define divVer VSIZE/17

//Definición de cada una de las instancias por las que pasará la máquina de estados
#define inicio 0
#define contadorDinero 1
#define expulsionProducto1 2
#define expulsionProducto2 7
#define calculoVuelta 3
#define devolucionVuelta 4
#define creditoInsuficiente 5
#define monedaIntroducida 6
//Tenemos 4 productos, un número de producto mayor a 3 (de 0 a 3), no es considerado producto
#define NotAProduct 5
//Definición de los botones de los productos con sus respectivas coordenadas para que queden
//ajustados y equiespaciados a la izquierda de la pantalla
#define    botProd0    Boton(divHor,    3*divVer,    4*divHor,    2*divVer,27, productos[0])
#define    botProd1    Boton(divHor,    6*divVer,    4*divHor,    2*divVer,27, productos[1])
#define    botProd2    Boton(divHor,    9*divVer,    4*divHor,    2*divVer,27, productos[2])
#define    botProd3    Boton(divHor,    12*divVer,    4*divHor,    2*divVer,27, productos[3])
#define botDev   Boton(HSIZE-6*divHor, VSIZE-4*divVer, 5*divHor, 2*divVer, 27, "Devolver")
//#define bot20 ComColor(amarilloClaro);ComCirculo(10.5*divHor, 10*divVer, divHor);     ComColor(negro); ComTXT(10.5*divHor, 10*divVer,27,OPT_CENTER, "20c")
//#define    bot5    ComColor(naranja);ComCirculo(10.5*divHor,    13*divVer, 0.8*divHor);ComColor(negro);  ComTXT(10.5*divHor,  13*divVer,27,OPT_CENTER, "5c")
#define bot20 ComCirculo(10.5*divHor, 10*divVer, divHor)
#define bot5 ComCirculo(10.5*divHor, 13*divVer, 0.8*divHor)
//DEFINICION PARA DIBUJO
#define alturaCuadro 20
#define largoCuadro 50

// ========================================================================
// Function Declarations
```

```
// ========================================================================
#define dword long
#define byte char

#define PosMin 750
#define PosMax 1000

#define XpMax 286
#define XpMin 224
#define YpMax 186
#define YpMin 54

unsigned int Yp=120, Xp=245;
// ========================================================================
// Variable Declarations
// ========================================================================

char chipid = 0;                          // Holds value of Chip ID read from
the FT800

unsigned long cmdBufferRd = 0x00000000;         // Store the value read from
the REG_CMD_READ register
unsigned long cmdBufferWr = 0x00000000;         // Store the value read from
the REG_CMD_WRITE register
unsigned int t=0;
//
################################################################################
##################################
// User Application - Initialization of MCU / FT800 / Display
//
################################################################################
##################################

unsigned long POSX, POSY, BufferXY;
unsigned long POSYANT=0;
unsigned int CMD_Offset = 0;
unsigned long REG_TT[6];
const int32_t REG_CAL[6]={21696,-78,-614558,498,-17021,15755638};
const int32_t REG_CAL5[6]={32146, -1428, -331110, -40, -18930, 18321010};


#define NUM_SSI_DATA        3

int RELOJ, Flag_ints, PeriodoPWM;
//Control de maquina de estados
int modo =inicio;
//Vector con los precios de los productos
int precios[]={30, 35, 40, 45};
//Cadena con los nombres de los productos
char productos[4][10]={"Chicle", "Gominolas", "Pipas", "Phoskitos"};
//Cadena para mostrar por la ventana de credito
char credito[]="CREDITO";
//Contador de crédito
int contCredito=0;
//Variable para saber el producto seleccionado
int prod=NotAProduct;
//Cadena con el mensaje para el recuadro de la pantalla del Vending
char mensaje[2][35]={"", "TOTAL: 0c"};
//Contador monedas de 20c
```

```
int m20=0;
//Contador monedas de 5c
int m5=0;
char monedas[3];
char carPrecios[3];
//Función del SLEEP fake, utilizar en caso de que dé problemas al usar el
debugger
//#define SLEEP SysCtlSleep()
#define SLEEP SysCtlSleepFake()

void SysCtlSleepFake(void)
{
 while(!Flag_ints);
 Flag_ints=0;
}

int contador; //Variable que utlizamos en la rutina de interrupción del timer
para contar las veces que se activa
//Interrupción del Timer0
void IntTimer0(void)
{
    // Clear the timer interrupt
    TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    Flag_ints=1;
    contador ++;
}

//Función giro de la segunda práctica para convertir el ángulo de giro a un
%
void giro (int pos)
{//Valores máximo y mínimo que llegarán PWM, y serán utilizados en la función
giro
    int Max_Pos = 4700;//4200; //3750
    int Min_Pos = 1000; //1875
    int posicion=Min_Pos+((Max_Pos-Min_Pos)*pos)/100;
    PWMPulseWidthSet(PWM0_BASE, PWM_OUT_4, posicion);   //Inicialmente, 1ms
}

int main(void)

{
    //Habilitar los periféricos implicados en el ejercio: leds, botones, la
salida del servomotor y el PWM
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOJ);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPION);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOG);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);

    //Habilitar los periféricos implicados en el Sleep
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_GPIOF);
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_GPIOJ);
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_GPION);
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_GPIOG);
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_PWM0);

    //Definir tipo de pines, los botones como entradas y los leds como salidas
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_0 |GPIO_PIN_4); //F0 y
F4: salidas
```

```
    GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_0 |GPIO_PIN_1); //N0 y
N1: salidas
    GPIOPinTypeGPIOInput(GPIO_PORTJ_BASE, GPIO_PIN_0|GPIO_PIN_1);    //J0 y
J1: entradas

GPIOPadConfigSet(GPIO_PORTJ_BASE,GPIO_PIN_0|GPIO_PIN_1,GPIO_STRENGTH_2MA,GP
IO_PIN_TYPE_STD_WPU); //Pullup en J0 y J1

    PWMClockSet(PWM0_BASE,PWM_SYSCLK_DIV_64);   // al PWM le llega un reloj
de 1.875MHz

    GPIOPinConfigure(GPIO_PG0_M0PWM4);          //Configurar el pin a PWM
    GPIOPinTypePWM(GPIO_PORTG_BASE, GPIO_PIN_0);

    //Configurar el PWM4, contador descendente y sin sincronización
(actualización automática)
    PWMGenConfigure(PWM0_BASE,      PWM_GEN_2,      PWM_GEN_MODE_DOWN      |
PWM_GEN_MODE_NO_SYNC);

    PeriodoPWM=37499; // 50Hz  a 1.875MHz
    PWMGenPeriodSet(PWM0_BASE, PWM_GEN_2, PeriodoPWM); //frec:50Hz
    giro(50); //Posicion inicial del servo
    PWMGenEnable(PWM0_BASE, PWM_GEN_2);     //Habilita el generador 0
    PWMOutputState(PWM0_BASE, PWM_OUT_4_BIT , true);    //Habilita la salida
1
    //Matriz con cada uno de los leds encendidos, tal y como en la primera
práctica
    int LED[4][4]={0,0,0,1,
                   0,0,1,0,
                   0,1,0,0,
                   1,0,0,0};
    //Comenzamos con los leds apagados y el servo al 50% (hacia abajo)
    L0_OFF;
    L1_OFF;
    L2_OFF;
    L3_OFF;
    giro(50);


    RELOJ=SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ    |    SYSCTL_OSC_MAIN    |
SYSCTL_USE_PLL | SYSCTL_CFG_VCO_480), 120000000);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);       //Habilita T0
    TimerClockSourceSet(TIMER0_BASE, TIMER_CLOCK_SYSTEM);   //T0 a 120MHz
    TimerConfigure(TIMER0_BASE,  TIMER_CFG_PERIODIC);      //T0 periodico y
conjunto (32b)
    TimerLoadSet(TIMER0_BASE, TIMER_A, RELOJ/20 -1);
    TimerIntRegister(TIMER0_BASE,TIMER_A,IntTimer0);
    IntEnable(INT_TIMER0A); //Habilitar las interrupciones globales de los
timers
    TimerIntEnable(TIMER0_BASE,  TIMER_TIMA_TIMEOUT);      // Habilitar las
interrupciones de timeout
    IntMasterEnable();  //Habilitacion global de interrupciones
    TimerEnable(TIMER0_BASE, TIMER_A);  //Habilitar Timer0
    HAL_Init_SPI(1, RELOJ);  //Boosterpack a usar, Velocidad del MC
    Inicia_pantalla();       //Arranque de la pantalla
    // Note: Keep SPI below 11MHz here
    //
================================================================
```

```
    // Delay before we begin to display anything
    //
=========================================================================
    SysCtlDelay(RELOJ/3);
    //Calibración predeterminada de la pantalla
    int i;
 #ifdef VM800B35
     for(i=0;i<6;i++)    Esc_Reg(REG_TOUCH_TRANSFORM_A+4*i, REG_CAL[i]);
 #endif
 #ifdef VM800B50
     for(i=0;i<6;i++)    Esc_Reg(REG_TOUCH_TRANSFORM_A+4*i, REG_CAL5[i]);
 #endif

    while(1){
        //Ciclo de ejecución cada 50 ms
        SLEEP;
        //Máquina de estados
        switch(modo){
        case inicio://En el estado inicial:
            sprintf(credito, "CREDITO");
            if (B1_ON)//Si pulsamos B1, introducimos una moneda de 20c,
aumentamos su contador y preparamos los mensajes correspondientes
            {
                modo=monedaIntroducida;
                m20++;
                sprintf(mensaje[0], "MONEDA INTRODUCIDA: 20c");
                contador=0;
                contCredito+=20;
                ComColor(verdeClaro);
            }else if (B2_ON) //Si pulsamos B2, introducimos una moneda de
5c, aumentamos su contador y preparamos los mensajes correspondientes
            {
                m5++;
                modo=monedaIntroducida;
                sprintf(mensaje[0],"MONEDA INTRODUCIDA:5c");
                contCredito+=5;
                contador=0;
                ComColor(verdeClaro);
            }
            break;
        case monedaIntroducida: //Esperamos 1 segundo y preparamos los
mensajes relacionados con la cantidad de monedas de 20c
            if (contador>=20) {
                modo=contadorDinero;
            }

            sprintf(credito, "%d", contCredito);
            sprintf(mensaje[1], "TOTAL: %d c  ",contCredito);
            break;
        case contadorDinero: //Indicamos en el recuadro que se ha introducido
una moneda de 20c al pulsar B1
            if (B1_ON) {modo=monedaIntroducida;
                        contador=0;
                        sprintf(mensaje[0], "MONEDA INTRODUCIDA: 20c");
                        m20++;}
                            //Indicamos en el recuadro que se ha introducido
una moneda de 20c al pulsar B1
            else if (B2_ON)  {modo=monedaIntroducida;
                              contador=0;
```

```
                            sprintf(mensaje[0],"MONEDA INTRODUCIDA:5c");
                            m5++;}
            else if (botDev) {
                //Si queremos devolver el crédito introducido, devolvemos el
mayor número de monedas de 20c posibles, e indicamos
                //cuantas monedas de 20c y de 5c hemos devuelto en total
                modo = devolucionVuelta;
                m20=contCredito/20;
                m5=(contCredito-20*m20)/5;
                contador=0;
                sprintf(mensaje[0], "DEVOLUCION DE %d c",contCredito);
                sprintf(mensaje[1], "%d x 20c, %d x 5c", m20,m5);
                sprintf(credito, "%d", 0);
            }//Posibles casos de producto, del 0 al 3, 4 en total
            else if (botProd0)  prod=0;
            else if (botProd1)  prod=1;
            else if (botProd2)  prod=2;
            else if (botProd3)  prod=3;

            if (prod<4)
                //Si elegimos uno de los 4 productos disponibles y tenemos
crédito disponible, encendemos su led correspondiente,
                //y giramos el servomotor
            {   if(contCredito>=precios[prod]){
                    sprintf(mensaje[0],"DISPENSANDO PRODUCTO");
                    sprintf(mensaje[1],"");
                    modo=expulsionProducto1;
                    giro(0);
                    GPIOPinWrite(GPIO_PORTN_BASE,               GPIO_PIN_1,
GPIO_PIN_1*LED[prod][0]);
                    GPIOPinWrite(GPIO_PORTN_BASE,               GPIO_PIN_0,
GPIO_PIN_0*LED[prod][1]);
                    GPIOPinWrite(GPIO_PORTF_BASE,               GPIO_PIN_4,
GPIO_PIN_4*LED[prod][2]);
                    GPIOPinWrite(GPIO_PORTF_BASE,               GPIO_PIN_0,
GPIO_PIN_0*LED[prod][3]);
                    }
                else if (contCredito<precios[prod]){
                    ////Si elegimos uno de los 4 productos disponibles y no
tenemos crédito disponible, se indica en la pantalla
                    modo=creditoInsuficiente;
                    sprintf(mensaje[0],"CREDITO INSUFICIENTE");
                }
                contador=0;}
            //El crédito introducido es el número de monedas de 20c * 20 +
el número de monedas de 5c * 5
            contCredito=20*m20+5*m5;
            break;
        case creditoInsuficiente:
            //Si no tenemos crédito suficiente, reseteamos el contador y
esperamos a que se introduzca más o se pida devolución
            if (contador>=20)   {
                contador=0;
                modo=contadorDinero;
                sprintf(mensaje[0],"");
                prod=NotAProduct;
            }
            break;
        case expulsionProducto1:
```

```
            //Caso de elegir un producto, sin devolución de crédito
            if (contador >=40)
            {   modo=expulsionProducto2;
                sprintf(mensaje[0],"PRODUCTO DISPENSADO");
                contCredito-=precios[prod];
                giro(50);
                L0_OFF;
                L1_OFF;
                L2_OFF;
                L3_OFF;
                contador=0;
            }
            break;
        case expulsionProducto2:
            //Caso de elegir un producto, con devolución de crédito (con el
mínimo número de monedas posible)
            if (contador>=40){
                modo=calculoVuelta;
                m20=contCredito/20;
                m5=(contCredito-20*m20)/5;
                contador=0;
                sprintf(mensaje[0], "DEVOLUCION DE %d c",contCredito);
                sprintf(mensaje[1], "%d x 20c, %d x 5c", m20,m5);
                sprintf(credito, "%d", 0);
                contador=0;
            }
            break;
            //Tras devolver el crédito correspondiente, se resetea el número
de monedas y se vuelve al estado inicial
        case calculoVuelta:
            if(contador >=20)
            {modo = devolucionVuelta;}
            break;
        case devolucionVuelta:
            if (contador>=100) {
                modo =inicio;
                sprintf(mensaje[0],"");
                sprintf(mensaje[1],"TOTAL: 0c");
                m20=0;
                m5=0;
                contCredito=0;
                sprintf(credito,"CREDITO");
                prod=NotAProduct;
            }
            break;
            }

        //Dibujar la pantalla
        Lee_pantalla();
        Nueva_pantalla(azulClaro);
        //Fondo botones productos
        ComColor(amarilloClaro);
        ComRect(0.5*divHor, 2*divVer, 7*divHor, VSIZE-2*divVer, true);
        //Pantalla para mensajes
        ComColor(blanco);
        ComRect(8*divHor, 2*divVer, HSIZE-divHor, 5*divVer, true);
        //Pantalla con cantidad de crédito
        ComRect(HSIZE-5*divHor,   VSIZE-8*divVer,   HSIZE-divHor,   VSIZE-
6*divVer, true);
```

```
        //Fondo de la caja de las monedas
        ComColor(verdeClaro);
        ComRect(8*divHor, 8*divVer, 12*divHor, VSIZE-2*divVer, true);
        //BOTONES
        ComColor(verdeClaro);//Pintamos el texto de cada uno de los
productos, y el de devolución
        botProd0;
        botProd1;
        botProd2;
        botProd3;
        botDev;
        //Pintamos el circulo que representa a la moneda de 20c
        ComColor(amarilloClaro);
        ComCirculo(10.5*divHor, 10*divVer, divHor);
        //Pintamos el circulo que representa a la moneda de 5c
        ComColor(naranja);
        ComCirculo(10.5*divHor, 13*divVer, 0.8*divHor);
        //Pintamos "20c" y "5c" dentro de cada moneda
        ComColor(negro);
        ComTXT(10.5*divHor, 10*divVer,27,OPT_CENTER, "20c");
        ComTXT(10.5*divHor, 13*divVer,27,OPT_CENTER, "5c");
        //Cantidad de monedas de 20c, representada junto al dibujo de la
moneda
        ComColor(negro);
        sprintf(monedas, "%d x", m20);
        ComTXT(8.5*divHor, 10*divVer, 27, OPT_CENTER, monedas);
        //Cantidad de monedas de 5c, representada junto al dibujo de la
moneda
        sprintf(monedas, "%d x", m5);
        ComTXT(8.5*divHor, 13*divVer, 27, OPT_CENTER, monedas);
        //Sacar por pantalla lo anterior
        ComTXT(HSIZE-3*divHor, VSIZE-7*divVer,22, OPT_CENTER, credito);
        ComTXT(HSIZE-6.5*divHor, 3*divVer,22, OPT_CENTER, mensaje[0]);
        ComTXT(HSIZE-6.5*divHor, 4*divVer,22, OPT_CENTER, mensaje[1]);
        //Vector de los precios de los productos
        int k;
        for (k=0; k<4; k++)
            {
                sprintf(carPrecios, "%dc", precios[k]);
                ComTXT(6*divHor, (3*k+4)*divVer,22, OPT_CENTER, carPrecios);
            }
        ComColor(negro);
        //Marcos negros para cada uno de los recuadros de la pantalla
        ComRect(8*divHor-4, 2*divVer-4, HSIZE-divHor+4, 5*divVer+4, false);
        ComRect(8*divHor-4,   8*divVer-4,   12*divHor+4,   VSIZE-2*divVer+4,
false);
        ComRect(HSIZE-5*divHor-4, VSIZE-8*divVer-4, HSIZE-divHor+4, VSIZE-
6*divVer+4, false);
        ComRect(0.5*divHor-4,   2*divVer-4,   7*divHor+4,   VSIZE-2*divVer+4,
false);
        Dibuja();
    }
}
```