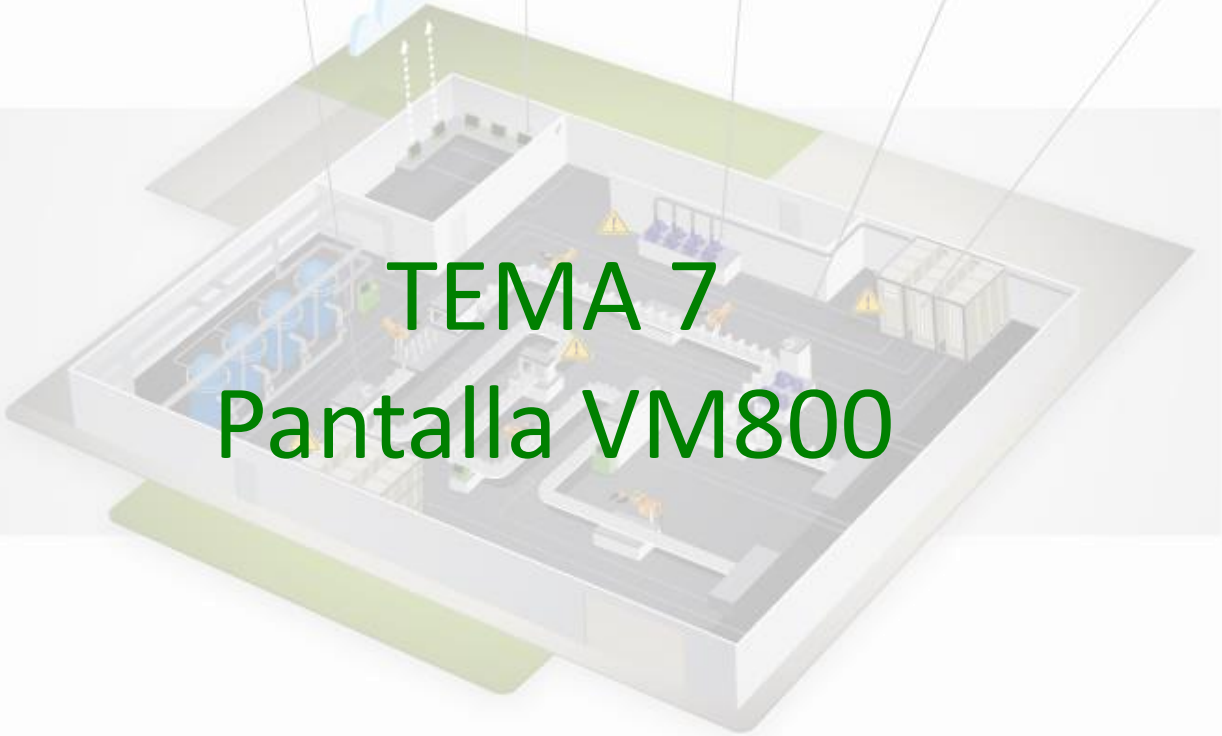
TEMA 7
Pantalla VM800



Objetivos

- Presentar la pantalla VM800 como solución OEM
- Manejo de la librería de funciones FT800.h

INDICE

- Introducción
- Procesador FT800 de FTDI
- Librería ft800.c
- Sistema VM800
- Ejemplo de programación

Introducción

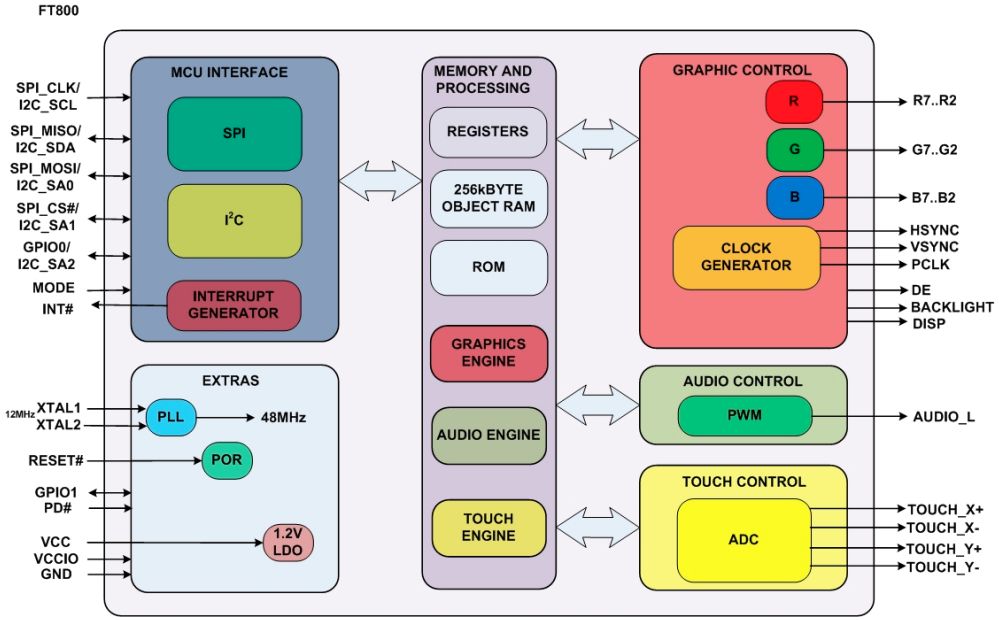
- Necesidad de desarrollo rápido de aplicaciones
- Cuello de botella: interfaz de usuario
- Sistemas OEM (original equipment manufacturer):
  - fabricación por terceros de partes del sistema
  - Soporte software necesario
- FTDI: *Future Technology Devices International*
  - Especializado en soluciones intermedias (OEM)
  - FT232: *estándar* como puente usb-rs232

Especificaciones de la VM800

- Pantalla gráfica color
  - 320x240 pixels
  - 256k colores
- Coprocesador gráfico
- Táctil resistiva con procesamiento
- Sistema de audio
  - Sintetizador de audio basado en MIDI
  - Amplificador y altavoz de 8 Ω
- Interfaz SPI con el *host*

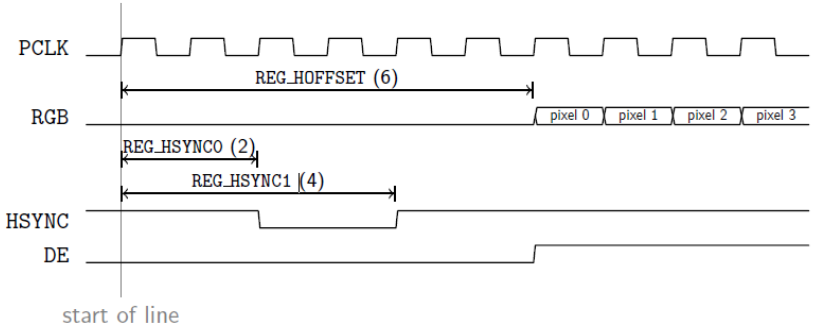
Procesador FT800

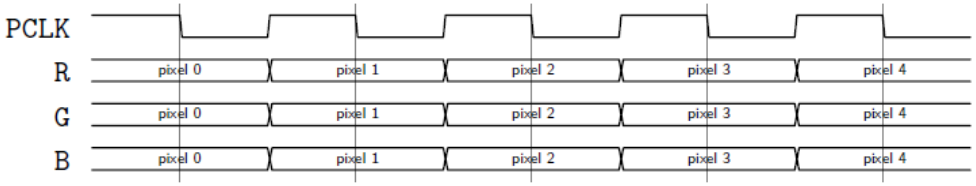
- Embedded Video Engine (EVE)
- Funcionamiento a 48MHz
- Interfaz SPI hasta 30MHz
- Interfaz I2C hasta 3,4MHz



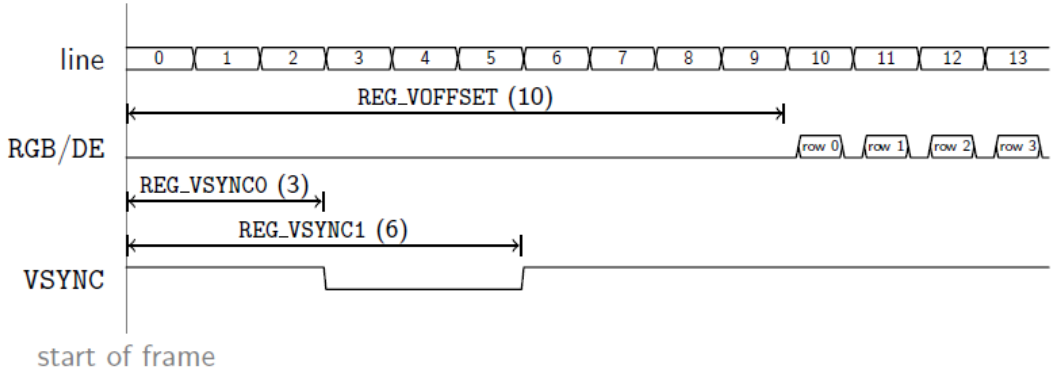
Dibujado de la pantalla

- Cada línea, pixel a pixel tras pulso de sincronismo y cabecera:





- Redibujado, línea a línea tras sincronismo vertical:



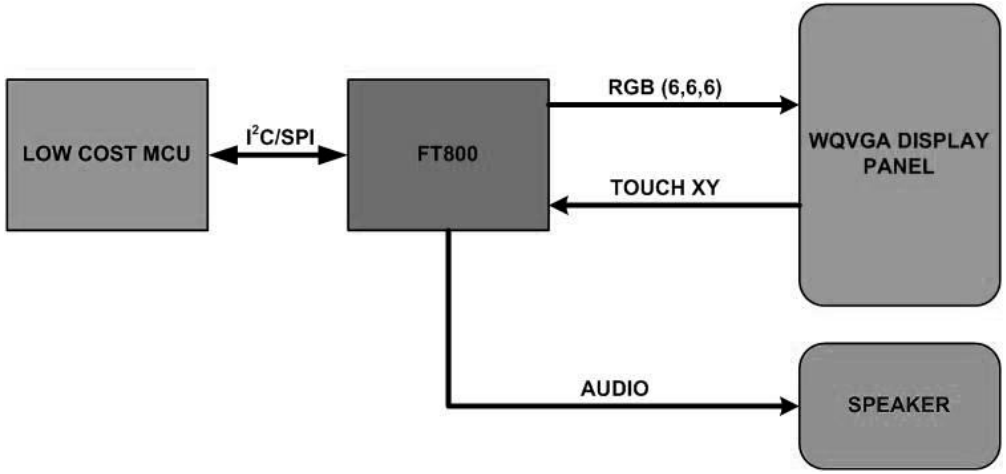
Mapa de memoria

- 256k RAM gráfica
- 275k ROM fonts y bitmap
- 4k de buffer de comandos
- 380 bytes para registros
  - 98 Reg. de 32 bits

Start Address	End Address	Size	NAME	Description
00 0000h	03 FFFFh	256 kB	RAM_G	Main graphics RAM
0C 0000h	0C 0003h	4 B	ROM_CHIPID	FT800 chip identification and revision information: Byte [0:1] Chip ID: "0800" Byte [2:3] Version ID: "0100"
0B B23Ch	0F FFFBh	275 kB	ROM_FONT	Font table and bitmap
0F FFFCh	0F FFFFh	4 B	ROM_FONT_ADDR	Font table pointer address
10 0000h	10 1FFFh	8 kB	RAM_DL	Display List RAM
10 2000h	10 23FFh	1 kB	RAM_PAL	Palette RAM
10 2400h	10 257Fh	380 B	REG_*	Registers
10 8000 h	10 8FFFh	4 kB	RAM_CMD	Command Buffer

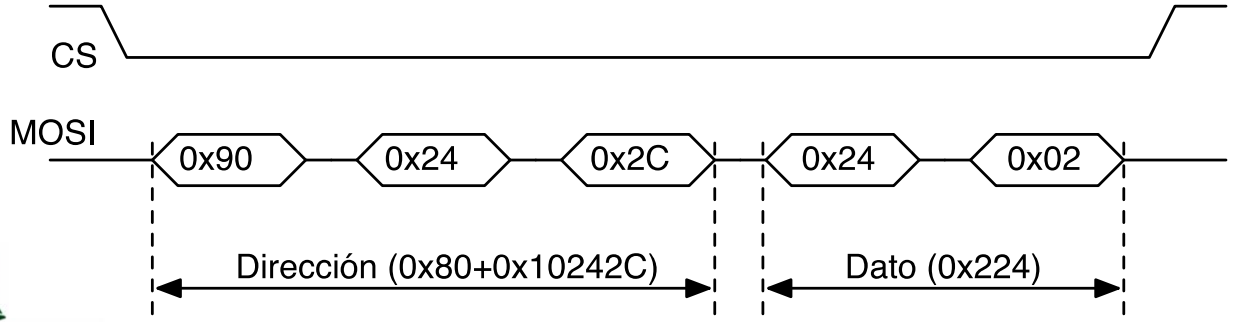
Modo de funcionamiento

- Mandar por SPI los comandos o datos necesarios
- Leer de la memoria los datos de vuelta
- No hay necesidad de *refrescar* la información
- El FT800 funciona en paralelo con nuestro MCU



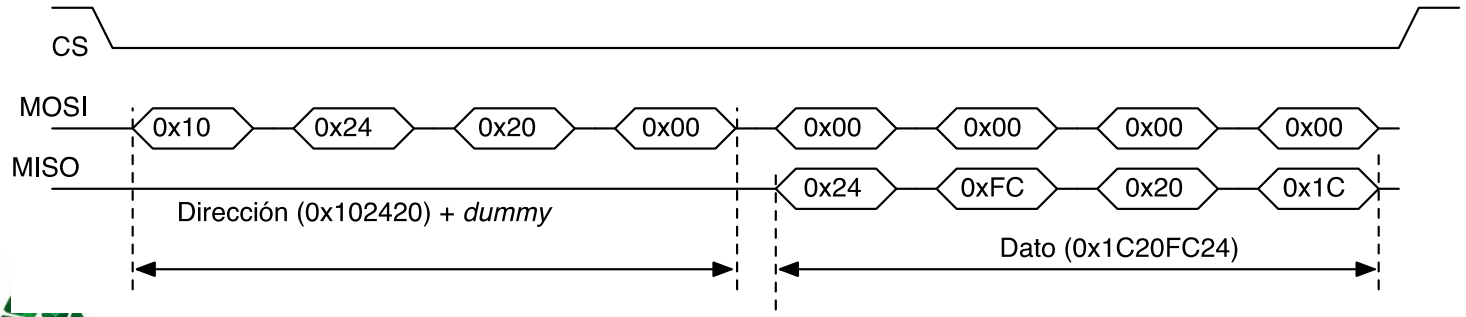
Escritura en memoria

- Dirección: 3 bytes (0x000000-0x108FFF)
- Protocolo:
  - CS low
  - Mandar dirección registro y marca de escritura
    - Sumar 0x80 al primer byte mandado
  - Mandar dato (8, 16 ó 32 bits)
  - CS high
- Ejemplo: escribir 0x224 en el registro 0x10242C



Lectura de memoria

- Dirección: 3 bytes (0x000000-0x108FFF)
- Protocolo:
  - CS low
  - Mandar dirección registro y marca de lectura
    - Forzar los dos primeros bits a 0.
  - Mandar un 0 ‘dummy’
  - Mandar 1, 2 ó 4 0 dummys para leer
  - CS high
- Ejemplo: leer el registro 0x102420 (32 bits)



Operaciones posibles

- Escritura o lectura de un registro
  - Memoria 0x102400-0x10257F
  - Configuración y manejo a bajo nivel
- Escritura de un *comando*
  - Operación a alto nivel
    - Comienzo de una línea, círculo…
  - Comandos del coprocesador
    - Botones, sliders, relojes, teclas...

Registros internos

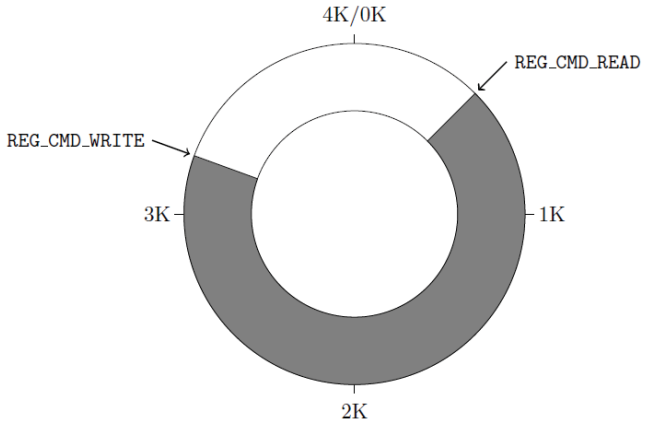
- Configuración del sistema
  - Motor gráfico
    - Tamaño, frecuencia de actualización, rotación…
  - Pantalla táctil
    - Calibración, valores leídos xy, modo de funcionamiento…
  - Módulo de sonido
    - Sonido sintetizado, volumen, audio play-stop…
  - Coprocesador gráfico
    - Leer o escribir comandos para el coprocesador
  - Miscelánea
    - Frecuencia del micro, estado de los gpio's, pwm para la retroiluminación…

Coprocesador gráfico

- Múltiples opciones de *alto nivel*
  - Texto
  - Relojes, indicadores circulares
  - Gradientes de color
  - Botones efecto 3D
  - Barras de progreso
  - Sliders
  - Mandos giratorios
  - Selectores
  - Números enteros

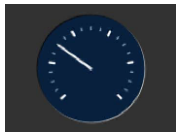
Manejo del coprocesador

- Cola circular de 4K
  - Escribo en la posición REG_CMD_WRITE
  - Él ejecuta por REG_CMD_READ
  - Datos de 4 bytes o múltiplos
- Escribir lista de **comandos**
  - P.ej: color de fondo, botón…
  - Terminar indicando que ejecute
- Esperar hasta que se haya ejecutado la anterior lista

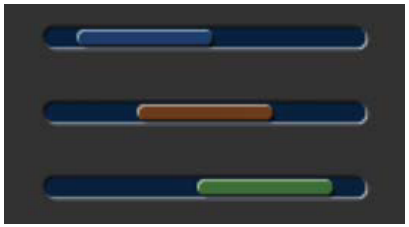


Comandos del coprocesador

- CMD_BUTTON: botón
- CMD_CLOCK: reloj analógico
- CMD_GAUGE: marcador de aguja
- CMD_KEYS: fila de teclas
- CMD_PROGRESS: barra de progreso
- CMD_SCROLLBAR: desplazamiento
- CMD_SLIDER: Slider
- CMD_DIAL: Control rotatorio
- CMD_TOGGLE: selector






Comandos del coprocesador

- Múltiples parámetros cada comando
- Modificadores (color, ancho de línea, 3D…)
- Ejemplo: CMD_BUTTON:
  - X: posición x, arriba a la izqda
  - Y: valor de Y, arriba a la izqda
  - W: ancho en píxeles
  - H: altura en píxeles
  - Font: fuente a usar
  - Options: plano, 3d,
  - Char * texto: cadena de texto
  - Completar hasta múltiplo de 4

VALOR	POS.
0xFFFFFF0D	0
150	4
100	6
100	8
50	10
26	12
0	14
'H'	15
'o'	16
'l'	17
'a'	18
0	19
0	20

Programación del FT800

- Muchos comandos principales (44)
- Muchos registros (98)
- Necesidad de librerías de *alto nivel*
  - HAL: hardware abstraction layer
  - Funciones básicas (acceso a spi)
  - Escritura y lectura en registros
  - Agrupación de funciones comunes
  - Manejo de los comandos como funciones

Librería FT800.h

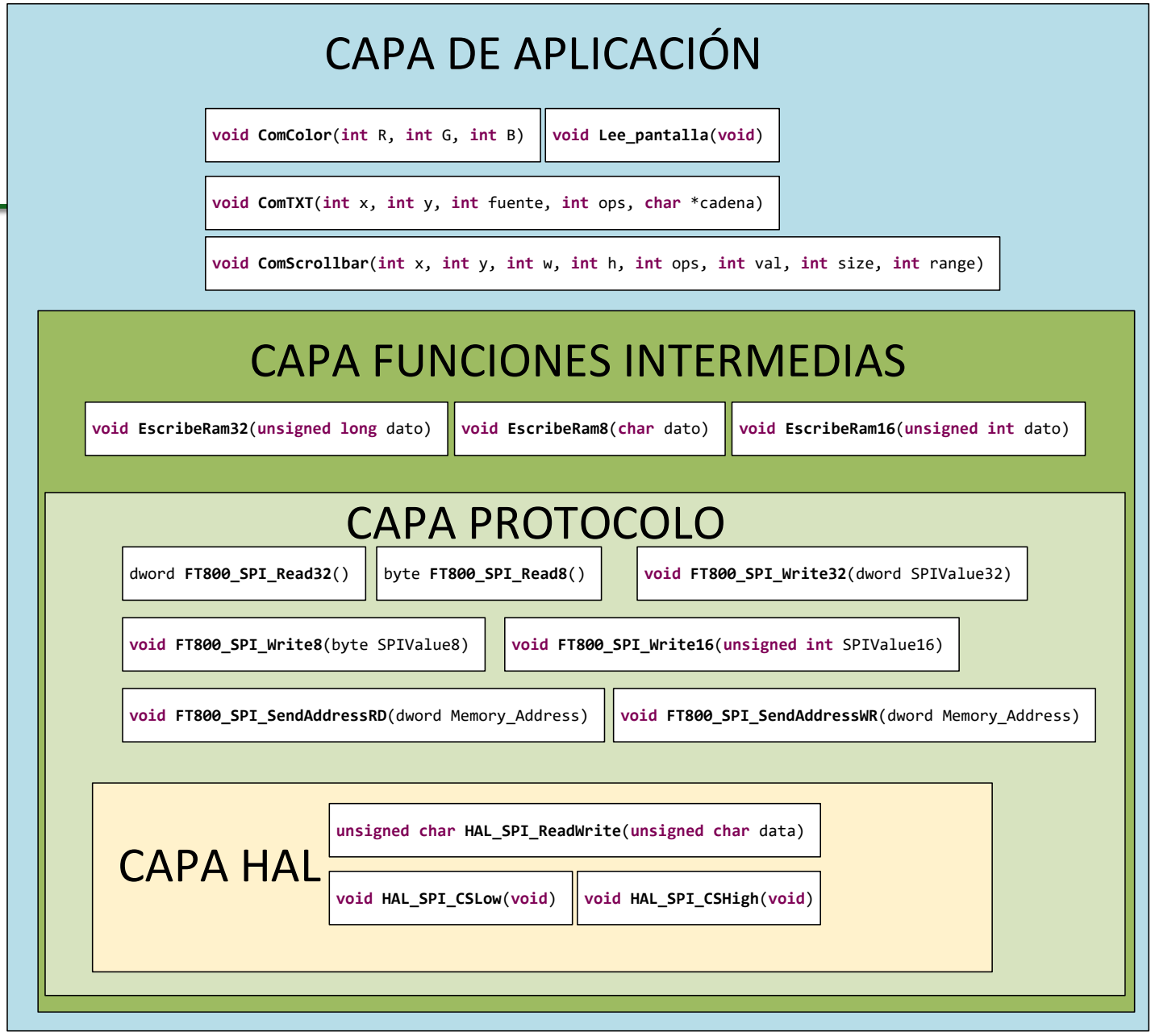
- Extraída de un ejemplo suministrado por ftdi
- En proceso de re-elaboración
  - Necesidad de depuración…
- Definición de los comandos más habituales
- Funciones de más alto nivel (inicialización…)
- Conveniente tener la FT800 programmer guide a mano

Funciones de la librería

```
/* ******* Hardware Abstraction Layer *******
 * Funciones de abstracción del hardware, especificas del micro a usar
 * Accesos a los pines de E/S, al SPI, y a la configuración
 ****************************************/
void HAL_Init_SPI(uint8_t Port, uint32_t RELOJ);
void HAL_Configure_MCU(void);
unsigned char HAL_SPI_ReadWrite(unsigned char);
void HAL_SPI_CSLow(void);
void HAL_SPI_CSHigh(void);
void HAL_SPI_PDlow(void);
void HAL_SPI_PDhigh(void);

/**********Funciones de bajo nivel:
 * mandar direccion para leer o escribir
 ****************************************/
void FT800_SPI_SendAddressWR(dword);
void FT800_SPI_SendAddressRD(dword);

/*******Siguiente nivel de abstraccion: Leer o escribir un dato
 * de 8, 16 o 32 bits, de cualquier posición interna de la pantalla
 ****************************************************/
char FT800_SPI_Read8(void);
long FT800_SPI_Read32(void);
void FT800_SPI_Write32(dword);
void FT800_SPI_Write16(unsigned int);
void FT800_SPI_Write8(byte);

void FT800_SPI_HostCommand(byte);
void FT800_SPI_HostCommandDummyRead(void);
unsigned int FT800_IncCMDOffset(unsigned int, byte);
void EscribeRam32(unsigned long dato);
void EscribeRam16(unsigned int dato);
void EscribeRam8(char dato);
void EscribeRamTxt(char* dato);
void Ejecuta_Lista(void);
void Dibuja(void);
unsigned long Lee_Reg(unsigned long dir);
void Esc_Reg(unsigned long dir, unsigned long valor);
```

```
/********Funciones de la capa superior, a usar:
* Inicialización de la pantalla, mandar comandos, y algunos comandos.
* NO TODOS LOS COMANDOS ESTÁN DESARROLLADOS.
************************************************/
void Inicia_pantalla(void);
void Comando(unsigned long COMM);
void ComEsperaFin(void);
void ComTXT(int x, int y, int fuente, int ops, char *cadena);
void ComNum(int x, int y, int fuente, int ops, unsigned long Num);
void ComTeclas(int x, int y, int w, int h, int fuente, unsigned int ops, char *Keys);
void ComVertex2ff(int x,int y);
void ComColor(int R, int G, int B);
void PadFIFO(void);
void Delay(void);
void Nueva_pantalla(int R, int G, int B);
void ComLineWidth(int width);
void ComPointSize(int size);
void Com_Punto(uint16_t x, uint16_t y, uint16_t R);void Lee_pantalla(void);
void ComScrollbar(int x, int y, int w, int h, int ops, int val, int size, int range);
void ComFgcolor(int R, int G, int B);
void ComBgcolor(int R, int G, int B);
void ComButton(int x, int y, int w, int h, int font, int ops, char *cadena);
char Boton(int x, int y, int w, int h, int font, char *cadena);void Espera_pant(void);
void ComRect(int x1, int y1, int x2, int y2, char relleno);
void ComCirculo(int x, int y, int r);
void ComLine(int x1, int y1, int x2, int y2, int ancho);
void Espera_pant(void);
void Calibra_touch(void);
void ComGradient(int x0,int y0, long color0, int x1, int y1, int color1);
void TocaNota( int instr, int nota);
void FinNota(void);
void VolNota(unsigned char volumen);
void Fadeout(void);
void Fadein(void);
```

- Varias capas:



El sistema VM800

- Pantalla+ sensor resistivo + amplificador+ alimentación + conector
- Marco para montar en caja

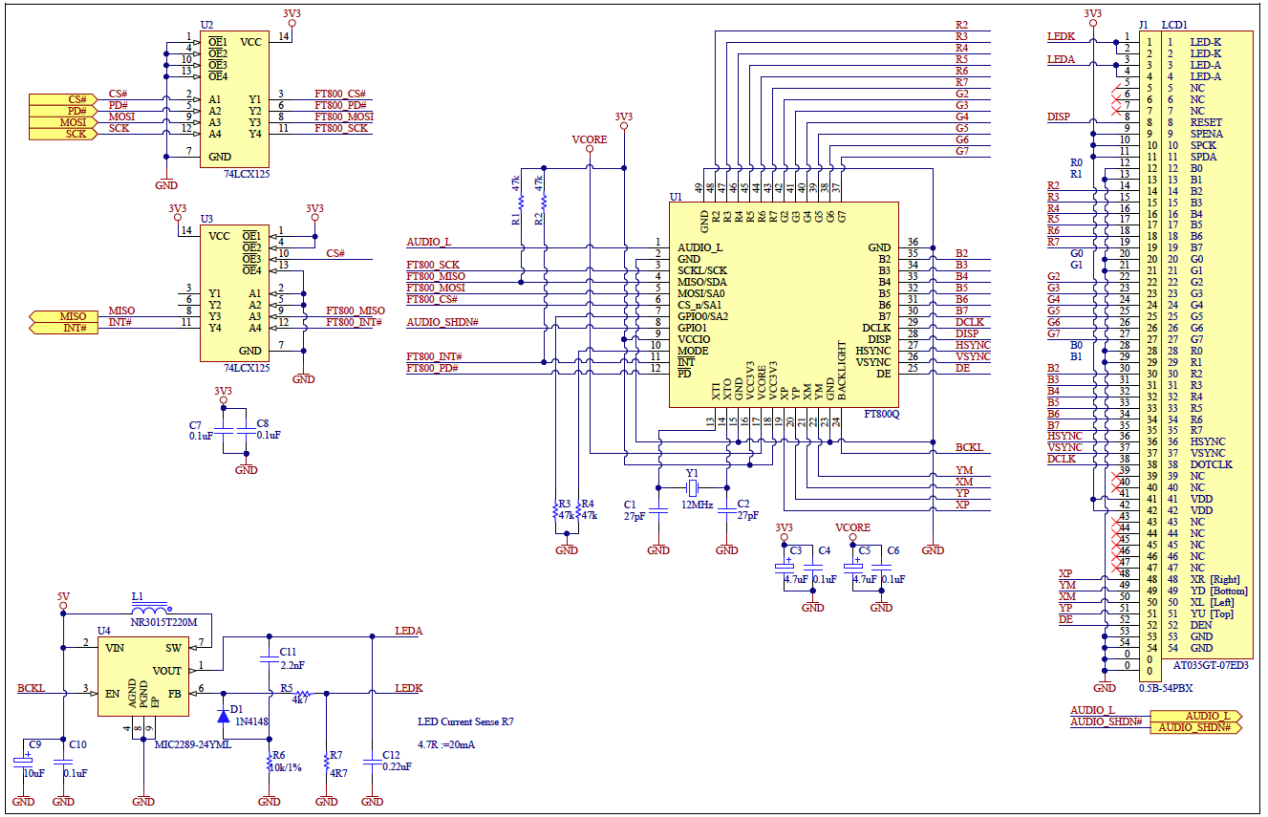
Pin-out del conector

- Señales del SPI (SCLK, MISO, MOSI, CS)
- Línea de interrupción (INT#)
- Power Down (PD#).
- Alimentación de dispositivos conectados, a 3.3V y a 5V

Pin No.	Name	Type	Description
1	SCLK	I	SPI Clock input, 3.3V (5V tolerant)
2	MOSI	I	Master Out Slave in, 3.3V (5V tolerant)
3	MISO	O	Master In Slave out, 3.3V
4	CS#	I	Chip select , active low, 3.3V (5V tolerant)
5	INT#	O	Interrupt output active low, 3.3V
6	PD#	I	Power down control input, active low , 3.3V (5V tolerant)
7	5V	P	5V power supply
8	3.3V	P	3.3V power supply
9	GND	P	Ground
10	GND	P	Ground

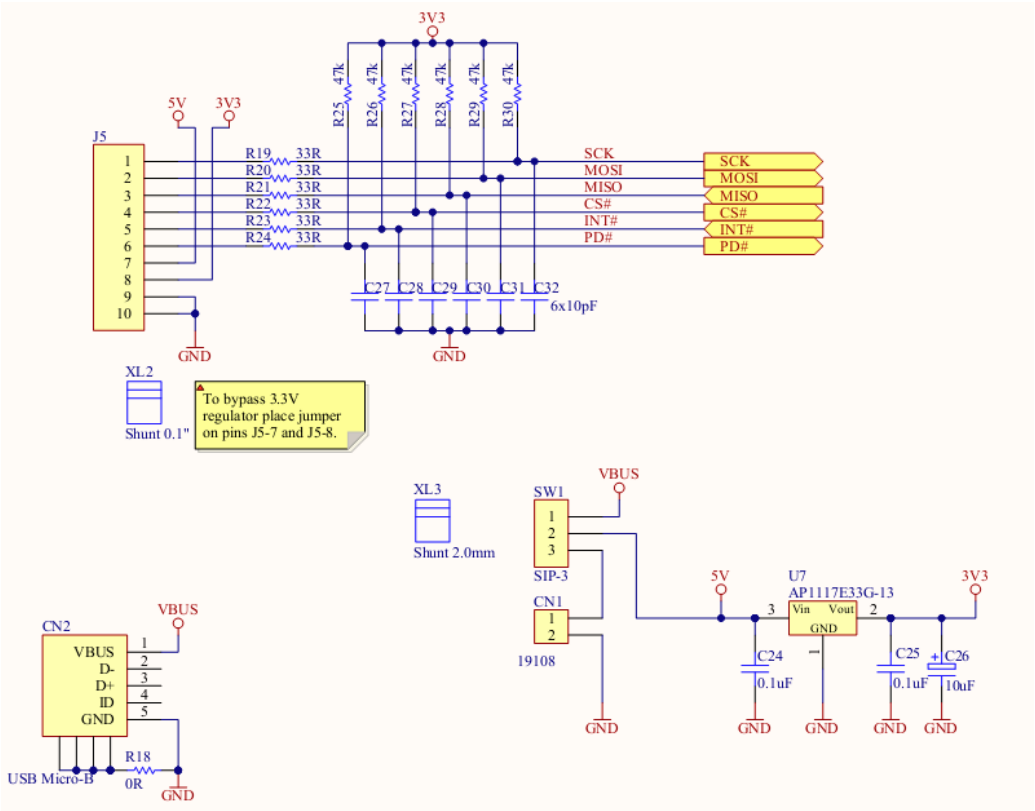
Esquemático del circuito (I)

- Conexión con la pantalla:



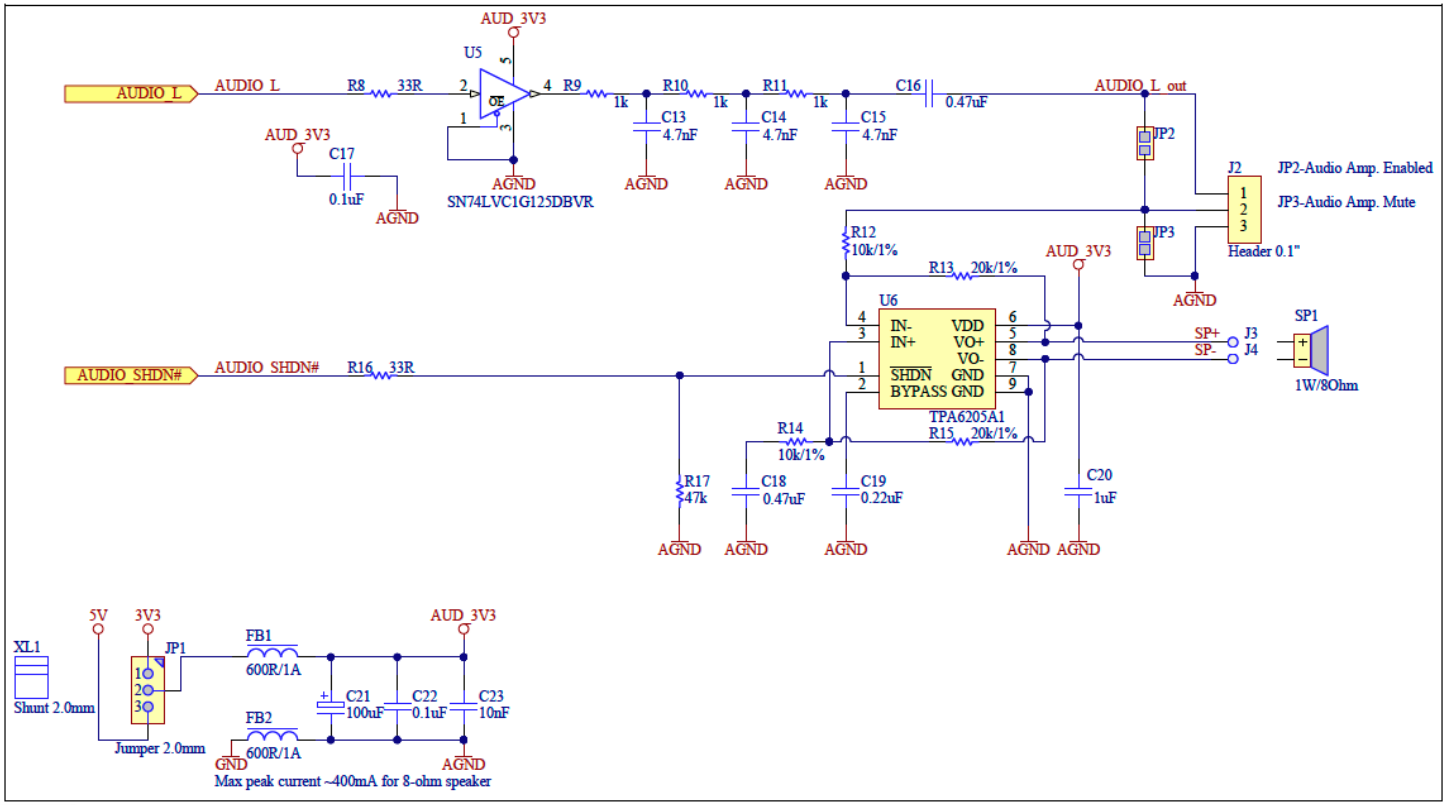
Esquemático del circuito (II)

- Conectores y regulador de tensión



Esquemático del circuito (III)

- Subsistema de audio

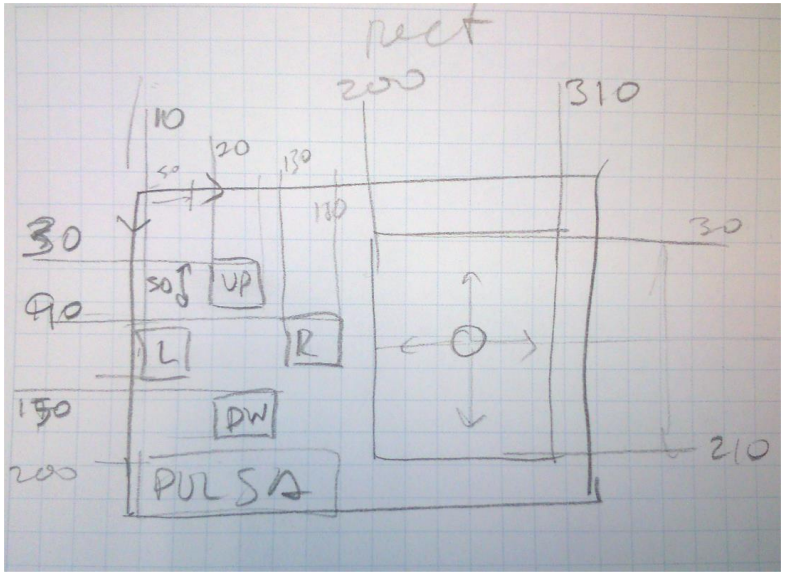


Apéndice: Pantalla VM800B50

- 5 pulgadas en lugar de 3.5 pulgadas
- Resolución de 480 x 272
- Cambian los parámetros de configuración
- Cambia el orden de los colores:
  - BGR en lugar de RGB
- FT800_tiva.c adaptada a los cambios
  - Una definición al comienzo de FT800_tiva.h distingue ambos casos: comentar la que no proceda
    - `#define VM800B35 //Pantalla de 3.5"`
    - `//#define VM800B50 //Pantalla de 5"`

Ejemplo de programación

- Cuatro botones, para mover un punto en la pantalla
- Primer paso: diseño de la interfaz:
- Bucle:
  - Comprobar pulsación
  - Cambiar coordenadas
  - Pintar punto



Configuración librería

- Tres #define en el fichero ft800.h:
  - Dos de ellos indican si la pantalla es de 3.5 o de 5 (en caso de 4.2, elegir 5)
  - El tercero, indica si se escalarán los valores:
    - Usar siempre la resolución 480x272
    - Mejora compatibilidad, pero cambia factor de escala

```
#define VM800B35   //Pantalla de 3.5"
//#define VM800B50 //Pantalla de 5"
#define ESCALADO   //Escalar a 480x272 (compatibilidad con 4.3" y 5")
```

Inicialización

- Función que configuran los periféricos del microcontrolador.
- Configuración de la pantalla

```
HAL_Init_SPI(1, RELOJ);  //Boosterpack a usar, Velocidad del MC
Inicia_pantalla();       //Arranque de la pantalla
```

```
void HAL_Init_SPI(uint8_t Port, uint32_t RELOJ_ACT){
//Inicializar el puerto SSI en función del BP usado, 1 ó 2
switch (Port){
case 1:
    SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI2);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPION);
    //  pin PN3 es /CS
    GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_3);
    CS_PORT=GPIO_PORTN_BASE;
    CS_PIN=GPIO_PIN_3;
    //  PD0 es SSI2 SSI2XDAT1
    GPIOPinConfigure(GPIO_PD0_SSI2XDAT1);
    GPIOPinTypeSSI(GPIO_PORTD_BASE, GPIO_PIN_0);
```

Dibujar pantalla inicial

- Se empieza con Nueva_pantalla(r,g,b)
- Acaba con la orden Dibuja(void)

```
Nueva_pantalla(16,16,16);
ComColor(21,160,6);
ComLineWidth(5);
ComRect(10, 10, HSIZE-10, VSIZE-10, true);
ComColor(65,202,42);
...

Dibuja();
```

Bucle principal

- Bucle while(1)
- Se lee la pantalla táctil (se almacena en POSX, POSY)
- Si está en un botón se pinta apretado. Si no, no
- Dibuja la pelota según sus nuevas coordenadas, borrando la anterior

```
while(1){
    Lee_pantalla();
    Nueva_pantalla(0x10,0x10,0x10);
    ComColor(100,100,255);
    ComFgcolor(200, 200, 10);

        if(Boton(10, 90, 50, 50, 28, "L"))
        {
            Xp--; if (Xp<=XpMin)  Xp=XpMin;
        }
    ...
    ComRect(200, 30, 310, 210, true);
    ComColor(0x30,0x50,0x10);
    ComLineWidth(3);
    ComRect(200, 30, 310, 210, false);
    ComCirculo(Xp, Yp, 20);
    Dibuja();
    }
```