Laboratorio de Microprocesadores - 86.07

Registrador de firmas digital

Profesor:			Ing. Guillermo Campiglio								
Cuatrimestre/Año:			1er/2017								
Turno de las clases prácticas											
Jefe de trabajos prácticos:											
Docente guía:											
Autores			Seguimiento del proyecto								
Nombre	Apellido	Padrón									
Maxim	Dorogov	95631									
Franco	Accifonte	93799									

Observaciones:

.....
.....
.....
.....
.....
.....
.....
.....
.....

Fecha de aprobación				Firma J.T.P

Coloquio	
Nota final	
Firma profesor	

Índice

1. Introducción

En el siguiente informe se pretende explicar el proceso de diseño de un programa para un registrador de firma digital a través del microcontrolador Atmega 328P en lenguaje Assembly. Para ello se utilizó un display TFT de 2.4 pulgadas provisto del controlador ILI9325D con una pantalla táctil resistiva. Los datos registrados son transferidos a través de una interfaz serie y procesados por Matlab. Dada la complejidad en el desarrollo del circuito impreso para el display, se optó por adquirir un Shield compatible con ambos controladores, ILI9325D y ADC.

2. Desarrollo

Para poder interactuar con el display, es necesario inicializarlo. La rutina de inicialización consta de 51 instrucciones que se envían al controlador del display mediante un bus de 8 bits. Estas instrucciones son palabras de 16 bits, por lo que deben ser enviadas en dos partes.

Una vez inicializado el display, es posible enviarle imágenes para mostrarlas en la pantalla. En esta aplicación enviamos la información píxel a píxel, estos datos se graban en la memoria ram del controlador ILI9325D para mantener en pantalla todos los píxeles a medida que se traza la firma en tiempo real hasta que el dispositivo sea reiniciado para leer una nueva firma. A continuación se muestra el diagrama en bloques del proyecto seguido del diagrama de flujo del programa implementado para el sistema de adquisición de la firma:

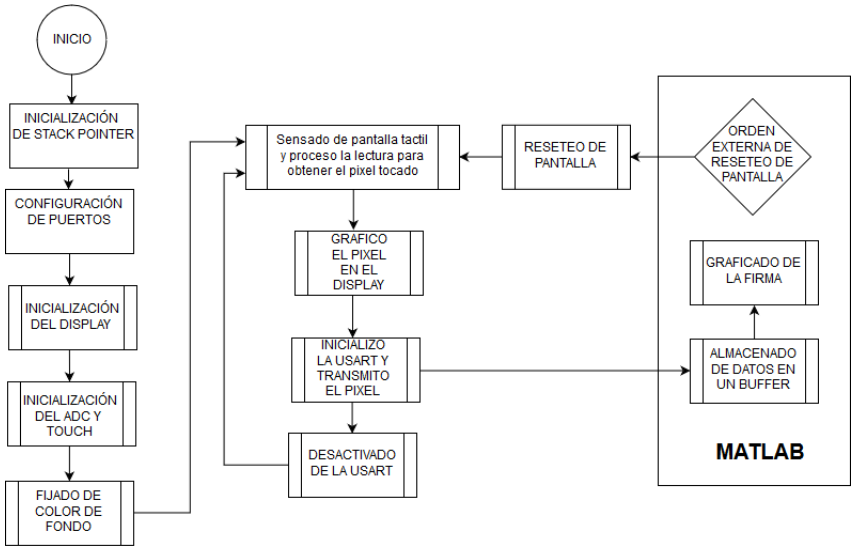


Figura 1: Diagrama de flujo del software desarrollado.

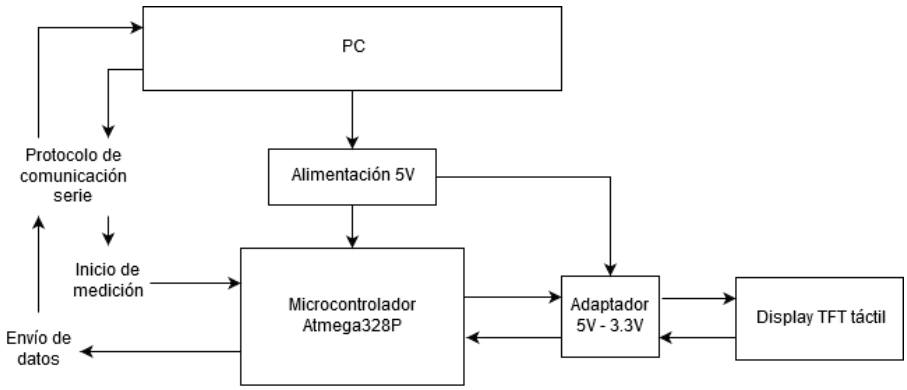


Figura 2: Diagrama en bloques del proyecto

Como se explicará mas adelante, se hace uso de prácticamente la totalidad de los puertos del micro-

controlador, con los cual los pines correspondientes a la USART son compartidos por el bus de datos del controlador de la pantalla. Debido a esto se tiene que activar la USART al momento de transmitir y desactivarla una vez finalizada la transmisión.

2.1. Comunicación con controlador `ILI9325D`

El dispositivo usado para controlar la parte gráfica es el ILI9325D. Es un integrado desarrollado para pantallas graficas de hasta 232 mil colores y una resolución de 240x320 pixeles, en cuanto a la comunicación cuenta con 2 modos, SPI y bus paralelo configurable en 8, 9, 16 o 18 bits. Posee además cuatro entradas de control:

- RS : Register select
- WR: Write enable
- CS: Chip select
- RST: Reset

Para nuestra aplicación se descartó el uso de SPI debido a la velocidad de actualización requerida para registrar una firma en tiempo real, por lo cual se opto por un bus de comunicación paralelo de 8bits de datos. Si bien los comandos que el display recibe son palabras de 16bits se programó el sistema de forma tal que sea posible mandar el comando empaquetado en dos bytes. En la figura 3 se muestra un diagrama de bloques simplificado del controlador.

Para comenzar la comunicación con este periférico es necesario inicializarlo y ponerlo en un modo de espera de comandos, para ello se implemento la rutina `INIT_LCD` en la cual se inicializa el modulo y se colocan todas las entradas a valores seguros. La rutina consiste en un envió sucesivo de valores al BUS de 8 bits del display los cuales fueron grabados en la memoria de programa del microprocesador y accedidos a través de un puntero

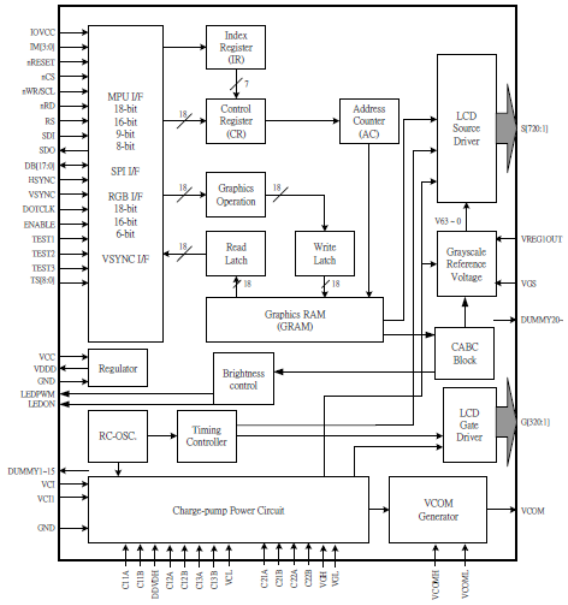


Figura 3: Diagrama en bloques simplificado del ILI9325D

Una vez inicializado el modulo es necesario configurar el área de pantalla sobre la cual se va a trabajar, para ello se implemento la macro `SET_XY_AREA`. Esta macro es llamada por todas las demás macros que utiliza el microprocesador para controlar la pantalla. Luego se llena la pantalla del color deseado con `FILL_SCREEN` y se procede a enviar píxel por píxel las coordenadas X e Y recibidas del sensor táctil a la macro `DRAW_PIXEL`.Esta ultima se encarga de grabar en la memoria ram del controlador las coordenadas a visualizar y envía los comandos necesarios para mostrar las coordenadas anteriores junto con la ultima en pantalla. Se prefirió el uso de macros en lugar de rutinas debido a la alta velocidad de respuesta requerida para la aplicación y además para no atentar contra la portabilidad del sistema. Todas las macros implementadas relacionadas con el manejo de la pantalla se encuentran en el archivo `tft_macros.mac`.

2.2. Comunicación con ADC

Para sensar las coordenadas del pixel presionado, es necesario comunicarse con otra parte del display táctil: el conversor analógico-digital `ADS7843`. Este sistema tiene su propio protocolo de comunicación, con otro juego de ordenes. Las entradas de control disponibles son las siguientes:

- TCLK : Clock interno del ADC
- TCS: Chip select del ADC
- TDIN: Bit de entrada de datos al ADC
- TDOUT: Bit de salida de datos del ADC

El sensor táctil consiste básicamente de un divisor resistivo doble, como se ve en la figura 4. Cuando se le pide al controlador sensar en la coordenada X, este conecta el terminal X+ a una tensión positiva, el terminal X- a una tensión negativa, y el terminal Y+ a la entrada del conversor.

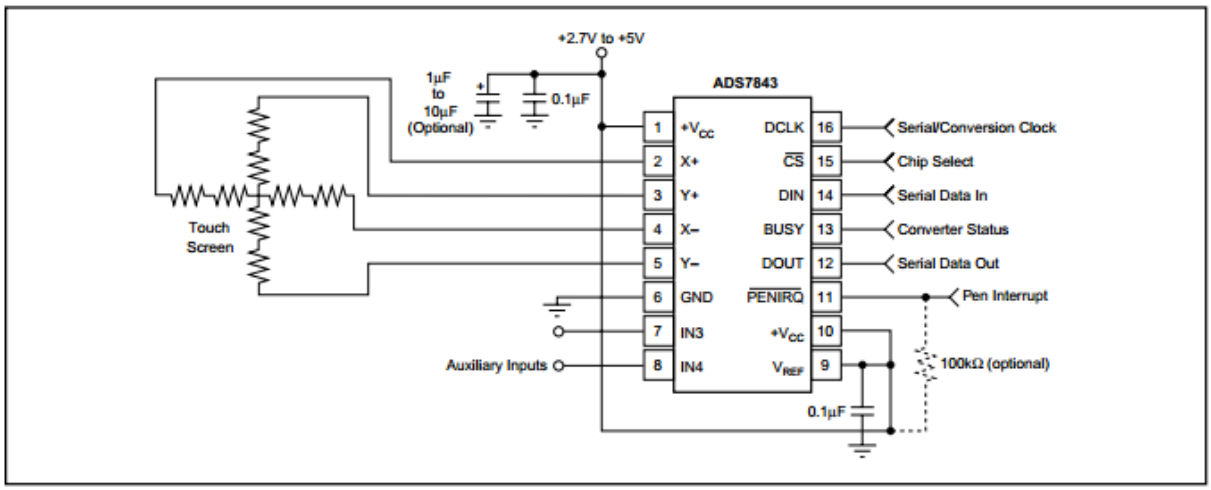


Figura 4: Operación básica del ADS7843

Para especificar cual coordenada se leerá, se envía por el pin DIN mediante comunicación serie la palabra `0xD0` para leer la coordenada X, o la palabra `0x90` para leer la coordenada Y.

Una vez finalizado el sensado el controlador devuelve, nuevamente por comunicación serie, una palabra de 12 bits variando entre `0x000` y `0xFFF`, la cual es separada en parte alta y parte baja y almacenada en dos registros. Este registro es luego convertido en la coordenada correspondiente del pixel seleccionado.

Se muestra en la imagen 5 el tren de pulsos que se envía para iniciar la adquisición, y el tren de pulsos que se recibe tras el sensado.

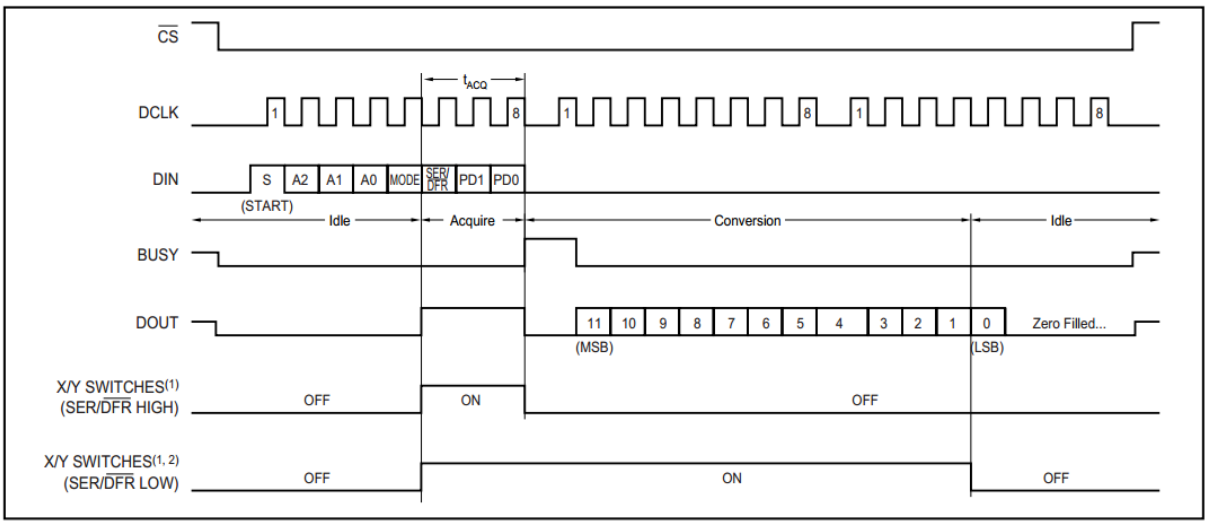


Figura 5: Comunicación con ADC

2.3. Conversión de datos

A causa de imperfecciones en el film táctil, cuando se toca con el lápiz cerca de los extremos el conversor no devuelve `0x000` o `0xFFF`, sino que devuelve valores cercanos. Es necesario tener en cuenta estos valores al hacer la conversión, porque de lo contrario se obtendrá un error importante cuando se grafique el pixel en la pantalla.

Para realizar la conversión, se implementaron las siguientes ecuaciones:

$$Y_{pix} = \frac{Y_{size}}{Y_{top} - Y_{bottom}}(Y_{ADC} - Y_{top})$$

$$X_{pix} = \frac{X_{size}}{X_{right} - X_{left}}(X_{ADC} - X_{left})$$

Donde X_{right}, X_{left}, Y_{top} e Y_{bottom} son constantes de calibración, y Y_{ADC} y X_{ADC} es la lectura entregada por el ADC.

Para implementar esto, se halló el valor de las constantes en base 16 y se programó un esquema de producto por punto fijo.

2.4. Envío de datos por puerto serie y recepción por Matlab

Mientras la firma es graficada en la pantalla táctil los datos son transferidos a la computadora para ser procesados y almacenados en una base datos, el programa de adquisición se desarrolló en Matlab. Su función es la recibir, validar y almacenar los datos provenientes del microcontrolador, graficarlos y a su vez darle la orden al microcontrolador para comenzar o finalizar la adquisición de la firma. La adquisición se hace a través del protocolo serie, vía USB.

Se configuró la USART del microcontrolador para transmitir datos de 8 bits de largo, con un bit de stop y un bit de inicio a una velocidad de 38400 baudios. Para ello se implementó la rutina la inicialización `INIT_USART`, y `DISABLE_USART` para interrumpir la comunicación. Para transmitir el dato se implementó la macro `DATA_TX` disponible en el archivo `IO.mac`.

Dado que el modulo utiliza la totalidad de pines del microcontrolador se tuvo que implementar un algoritmo de transmisión que no genere incompatibilidades entre los comandos enviados por el puerto hacia el software de adquisición y lo que puede interpretar el display. Para ello se activa la USART únicamente al momento de transmitir y luego se desactiva antes de pasar al estado de sensado. Dado que la memoria ram de un atmega328P es insuficiente para almacenar los píxeles de una firma se optó por realizar la trasmisión pixel a pixel en tiempo real en paralelo al sensado. A continuación se ilustra una foto con un dibujo en la pantalla táctil y la imagen generada por el software de adquisición.

Figura 6: Foto de un dibujo realizado sobre la pantalla táctil

Figura 7: Imagen recibida por el software de adquisición

2.5. Hardware utilizado

El modulo de display a utilizar acepta niveles lógicos de hasta 3.3 V, lo cual es un problema al momento de poner en practica el diseño. Para ello se hizo uso de un modulo que adapta las tensiones de salida de los puertos de la placa de desarrollo Arduino al nivel soportado por el display. El dispositivo adapta las tensiones de salida de 5 V provenientes de los puertos de Atmega 328p a logica de 3.3 V y realiza lo mismo con los datos provenientes del display hacia el microcontrolador mediante cuatro integrados 74HC541PW.

Se decidió usar una plataforma de desarrollo Arduino dada la compatibilidad de diversos módulos o "Shields"disponibles en el mercado. A continuación se muestran las imágenes correspondientes a la conexión realizada entre el microcontrolador, el display y un diagrama esquemático de todo el pryecto.

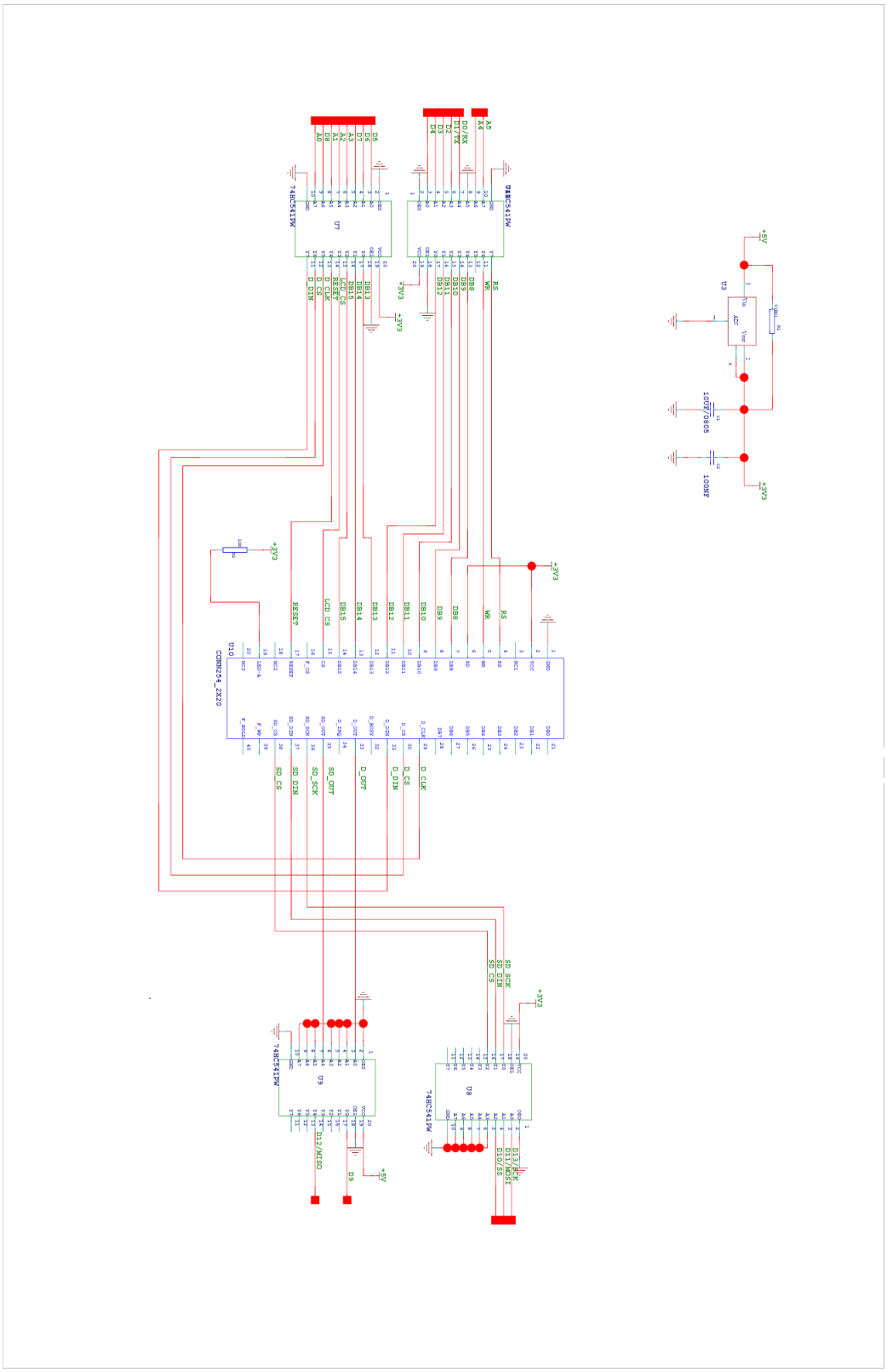


Figura 8: Diagrama esquemático del adaptador de niveles lógicos

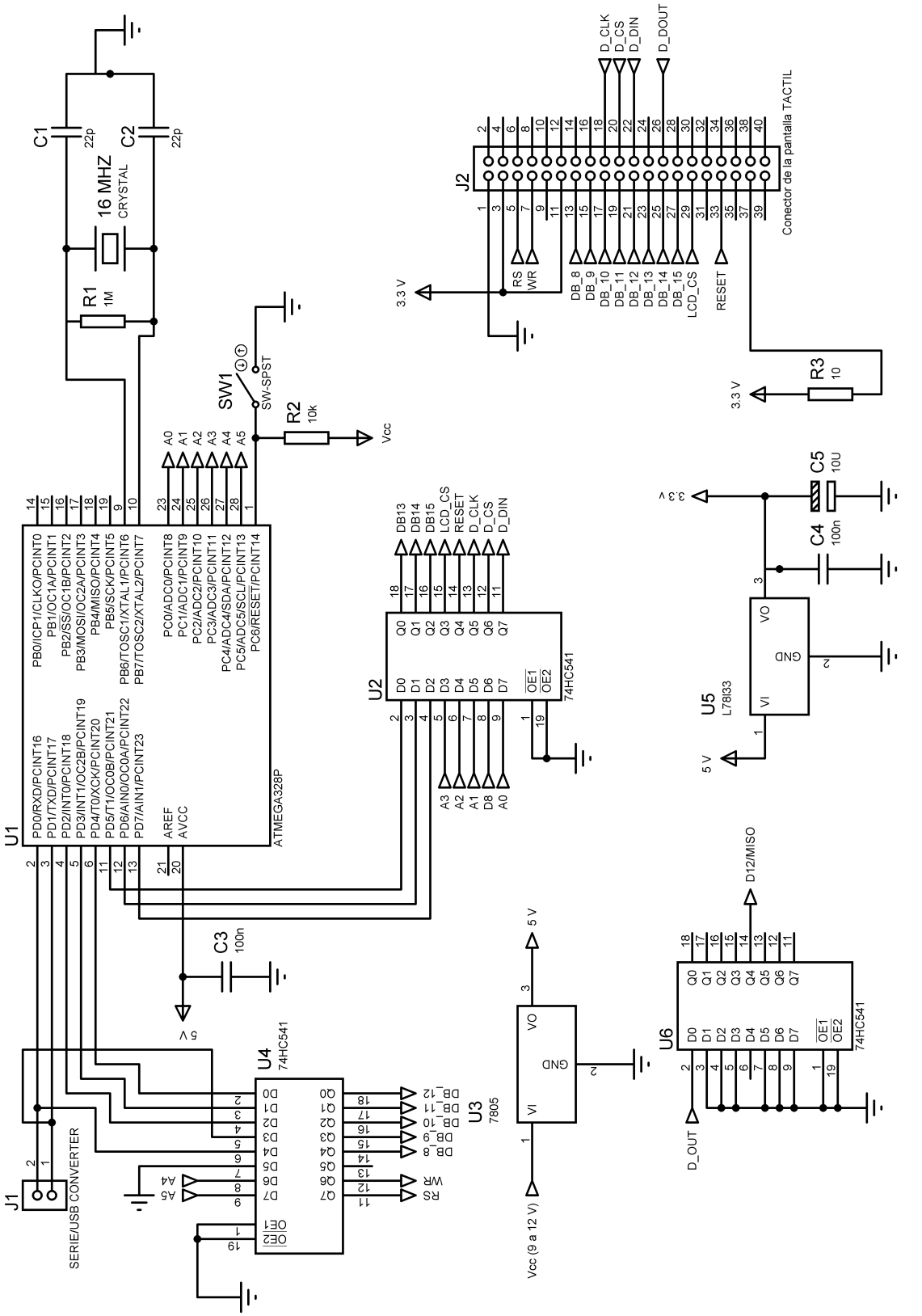


Figura 9: Diagrama esquemático del proyecto

El display táctil se conecta a la bornera referenciada en la figura 9, la cual se interconecta con los cuatro integrados 74HC541PW, las distintas salidas de los mismos van a los pines del microcontrolador referenciados desde A0-A5 y D0-D13 .

2.5.1. Listado de componentes

A continuación se listan los módulos de desarrollo adquiridos junto con el costo de cada uno. A su vez se mencionan algunos de los componentes principales que las integran.

Dispositivo	Contenido	Precio
Display touch TFT 2.4”	ILI3925D ADS7843	$240
Shield adaptador	74HC541PW	$220
Placa de desarrollo	Atmega328P Atmega 16U2	$150
Total		$610

Tabla 1: Resumen de componentes

3. Conclusiones

Se logro implementar con éxito el modulo de registro y el software de adquisición para procesamiento de las firmas. Algunas de las mejoras para implementar a futuro: optimizar la transmisión serie para reducir el delay generado entre que se realiza la firma y su visualización en pantalla, transmitir los datos en forma inalámbrica por medio de bluetooth para mayor comodidad y portabilidad del dispositivo, migrar a una plataforma de 32 bits para manejar pantallas mas grandes con mayor resolución con el fin de desarrollar una tableta gráfica con prestaciones comerciales.

4. Apéndices

A. Codigo

A.1. Main.asm

```
.include "m328def.inc"
.include "IO.mac"
.include "tft_colors.inc"
.include "tft_macros.mac"


.equ RAM_START = 0x0100
.EQU B_RST = 2
.EQU B_CS = 3
.EQU B_WR = 4
.EQU B_RS = 5

.EQU NUM_COM_VALUES = 50
.EQU NUM_DATA_VALUES = 50

.EQU tft_max_x = 319
.EQU tft_max_y = 239

.DEF com1 = r20
.DEF data = r22
.DEF counter = r21
.DEF c_high = r18
.DEF c_low = r17

.EQU PIXEL_COLOR = VGA_WHITE
.DEF pixel_x_high = r3
.DEF pixel_x_low = r2
.DEF pixel_y = r1
.DEF offset = r20
;-----------------------------
;-----PARTE DE LA LECTURA-----
;-----------------------------
.def orden = r16
.def resp_high = r17
.def resp_low = r18
.def contador = r19
.def promedio = r20
.def ac_resphx = r21
.def ac_resplx = r22
.def ac_resphy = r23
.def ac_resply = r24
.def temp_memh = r25
.def temp_meml = r26
;Direcciones de memoria RAM
.equ xh_min = RAM_START
.equ xl_min = xh_min+1
.equ xh_max = xh_min+2
.equ xl_max = xh_min+3
.equ yh_min = xh_min+4
.equ yl_min = xh_min+5
.equ yh_max = xh_min+6
.equ yl_max = xh_min+7
```

```
.equ anterior_xh = xh_min+8
.equ anterior_hl = xh_min+9
.equ anterior_y = xh_min+10
;Varianzas aceptables de las mediciones
.equ  var_x = 7
.equ var_y = 16
;Puertos
  ;Bits
.equ TB_CLK = 1 ;A1->PC4
.equ TB_CS = 2 ;D10->PB2
.equ TB_DIN = 0 ;A0->PC5
.equ TB_DOUT= 1 ;D9->PB1
;.equ T_IRQ = 34 ;no mapeado
;.equ T_BUSY= 32 ;no esta mapeado segun el esquematico del shield
  ;DDRs
.equ TD_CLK = DDRC ;A1->PC4
.equ TD_CS = DDRB ;D10->PB2
.equ TD_DIN = DDRC ;A0->PC5
.equ TD_DOUT= DDRB ;D9->PB1
;.equ T_IRQ = 34 ;no mapeado
;.equ T_BUSY= 32 ;no mapeado
  ;Ports
.equ TP_CLK = PORTC ;A1->PC4
.equ TP_CS = PORTB ;D10->PB2
.equ TP_DIN = PORTC ;A0->PC5
.equ TP_DOUT= PORTB ;D9->PB1
;.equ T_IRQ = 34 ;no mapeado
;.equ T_BUSY= 32 ;no mapeado
  ;Pin del DOUT
.equ TPIN_DOUT = PINB


;-----------------------------
;-----PARTE DEL PRODUCTO------
;-----------------------------
.def Tconst_conv   = r16
.def Tresp_high_x  = r12
.def Tresp_low_x = r13
.def Tresp_high_y = r14
.def Tresp_low_y = r15
.def Taux = r21
.def Taux_h = r22

.equ  TCAL_X = 0x00378F66; = 0000 0000 0011 0111 1000 1111 0110 0110
.equ  TCAL_Y = 0x03C34155; = 0000 0011 1100 0011 0100 0001 0101 0101
  .equ  Ttouch_x_left = (TCAL_X>>14) & 0x3FF; = 0000 1101 1110 = 0x0DE = 222
  .equ  Ttouch_y_top = (TCAL_Y>>14) & 0x3FFF; = 1111 0000 1101 = 0xF0D = 3853
.equ TCONV_X = 0x11 ;0x10
.equ TCONV_Y = 0x17 ;0x17
;---------------------------------------------
;-----Def y equs de la transmicion serie------
;---------------------------------------------

.equ Fosc = 16000000 ; clock
.equ Baud = 38400 ; baud
.equ UBRR = (Fosc/(Baud*16))-1

;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
.cseg

MAIN:
;%%%%%%%%%%%%%%%% STACK POINTER INIT %%%%%%%%%%%%%%%%%%%%

ldi r21,LOW(RAMEND)
OUT SPL,r21
ldi r21,HIGH(RAMEND)
OUT SPH,r21


;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

RCALL PORTS_CONFIG
RCALL INIT_LCD
RCALL INIT_TOUCH


FILL_SCRN VGA_BLACK;Lleno la pantalla de color negro

ACA:
rcall READ_CONVERT ;

ldi offset,240 ;cargo el valor del offset
sub offset,r10
mov r10,offset

DRAW_PIXEL r11,r12,r10
;divido la coordenada X en 2 y guardo el resutado de la division en r12
lsr r11
ror r12
lsr r10
;---------------------------------------
RCALL INIT_USART
rcall DELAY_5MS
;
DATA_TX r12
rcall DELAY_5MS
DATA_TX r10
rcall DELAY_5MS
rcall DISABLE_USART

RJMP ACA


;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;%%%%%%%%%%%%%%%%%%%%%%    RUTINAS    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

PORTS_CONFIG:

LOADIO DDRB, 0xff ;defino el puerto B como salida
LOADIO DDRD, 0xff ; defino el puerto D como salida. Va a ser el puerto de escritura al display.
LOADIO DDRC,0xff ; defino el PORTC como salida.Van a ser los bits de control del display.
;%%%%%%%%%%%%%%%%%%%%INICIALIZO LOS PUERTOS A VALoROES SEGUROS%%%%%%%%%%%%
LOADIO PORTC,0x38 ; inicializo el puerto en un valor seguro
LOADIO PORTB,0x00

PORTS_CONFIG_EXIT:ret
```

```
INIT_USART:
;%%%%%%%%%%%%%%%% UART INIT %%%%%%%%%%%%%%%%%%%
; inicializo la UART para transmitir datos de 8 bits
clr r16
ldi r16,(1<<TXEN0)
sts UCSR0B,r16

ldi r16, (3<<UCSZ00) ; set Format Frame 8bit 1bit de stop.
sts UCSR0C, r16

; set baud rate
ldi r16,HIGH(UBRR) ;set baud rate (38400bds / 16MHz)
sts UBRR0H,r16
ldi r16,low(UBRR)
sts UBRR0L,r16
INIT_USART_EXIT: RET

;-----------------------------------------------------------------

DISABLE_USART:

ldi r16,(0<<TXEN0)
sts UCSR0B,r16

DISABLE_USART_EXIT:ret

;-----------------------------------------------------------------

INIT_LCD:

SBI PORTC,B_RST
RCALL DELAY_5MS
CBI PORTC,B_RST
RCALL DELAY_15MS
SBI PORTC,B_RST
RCALL DELAY_15MS
CBI PORTC,B_CS

ldi Z1,low(INIT_COM_DATA_VALUES_ILI9325C<<1)
ldi zh,HIGH(INIT_COM_DATA_VALUES_ILI9325C<<1)

ldi counter,16
LOOP_INIT_16:
RCALL WRITE_COM_DATA
dec counter
brne LOOP_INIT_16

RCALL DELAY_200MS
RCALL WRITE_COM_DATA
RCALL WRITE_COM_DATA
RCALL DELAY_50MS
RCALL WRITE_COM_DATA
RCALL DELAY_50MS
RCALL WRITE_COM_DATA
RCALL WRITE_COM_DATA
RCALL WRITE_COM_DATA
RCALL DELAY_50MS
```

```
ldi counter,28
LOOP_INIT_28:

RCALL WRITE_COM_DATA
dec counter
brne LOOP_INIT_28

SBI PORTC,B_CS

END_INIT_LCD:RET

WRITE_COM_DATA:

RCALL WRITE_COM
RCALL WRITE_DATA

END_WRITE_COM_DATA: RET

WRITE_COM:

LPM com1,Z+ ;cargo comando a com1 y Z apunta a dat_high
cbi PORTC,B_RS
LOADIO PORTD,0
cbi PORTC,B_WR
sbi PORTC,B_WR
OUT PORTD, com1
cbi PORTC,B_WR
SBI PORTC,B_WR

END_WRITE_COM: RET

WRITE_DATA:

LPM data,Z+ ;cargo data_high en data y Z apunta a dat_low
sbi PORTC,B_RS
OUT PORTD,data
cbi PORTC,B_WR
SBI PORTC,B_WR
LPM data,Z+ ;cargo data_low en data y Z apunta a com1
OUT PORTD,data
cbi PORTC,B_WR
SBI PORTC,B_WR

END_WRITE_DATA: RET

DELAY_5MS:

      ldi  R17, $86
WGLOOP0:  ldi  R18, $C6
WGLOOP1:  dec  R18
          brne WGLOOP1
          dec  R17
          brne WGLOOP0
; -----------------------------
; delaying 2 cycles:
          nop
          nop
END_DELAY_5MS: RET
```

```
DELAY_15MS:
         ; delaying 40239780 cycles:
          ldi  R17, $F5
WGLOOP15: ldi  R18, $E7
WGLOOP13: ldi  R19, $EC
WGLOOP14: dec  R19
          brne WGLOOP14
          dec  R18
          brne WGLOOP13
          dec  R17
          brne WGLOOP15
; ----------------------------
; delaying 360 cycles:
          ldi  R17, $78
WGLOOP19: dec  R17
          brne WGLOOP19

END_DELAY_15MS: RET

DELAY_50MS:
      ldi  R17, $5F
WGLOOP6:  ldi  R18, $17
WGLOOP7:  ldi  R19, $79
WGLOOP8:  dec  R19
          brne WGLOOP8
          dec  R18
          brne WGLOOP7
          dec  R17
          brne WGLOOP6
; ----------------------------
; delaying 3 cycles:
          ldi  R17, $01
WGLOOP9:  dec  R17
          brne WGLOOP9
; ----------------------------
; delaying 2 cycles:
          nop
          nop
END_DELAY_50MS: RET

DELAY_200MS:
   ldi  R17, $F1
WGLOOP10:  ldi  R18, $19
WGLOOP11:  ldi  R19, $B0
WGLOOP12:  dec  R19
          brne WGLOOP12
          dec  R18
          brne WGLOOP11
          dec  R17
          brne WGLOOP10
; ----------------------------
; delaying 2 cycles:
          nop
          nop
END_DELAY_200MS: RET

INIT_TOUCH:
```

```
sbi TD_CLK,TB_CLK
sbi TD_CS,TB_CS
sbi TD_DIN,TB_DIN
cbi TD_DOUT,TB_DOUT
sbi TP_CLK,TB_CLK
sbi TP_DIN,TB_DIN
sbi TP_CS,TB_CS
INIT_TOUCH_EXIT: ret


;--------------------------------------read_convert
READ_CONVERT:
ldi promedio,16 ;Contador para acumular 8 muestras
ldi ac_resphx,0 ;inicializo los acumuladores en cero
ldi ac_resplx,0
ldi ac_resphy,0
ldi ac_resply,0
ldi temp_memh,0xFF
; sts  xh_min,temp_memh ;inicializo el minimo con 0xFF y el maximo con 0x00
; sts  xl_min,temp_memh ;asi el primer dato que reciba se reemplaza
; sts  xh_max,ac_resphx
; sts  xl_max,ac_resphx
; sts  yh_min,temp_memh
; sts  yl_min,temp_memh
; sts  yh_max,ac_resphx
; sts  yl_max,ac_resphx

INICIO_PROMEDIO:
ldi orden,0xD0 ;Leo en X
rcall MAGIA_ADC
;rcall COMP_EXT_X
add ac_resplx,resp_low ;Acumulo para despues promediar
adc ac_resphx,resp_high
ldi orden,0x90 ;Leo en Y
rcall MAGIA_ADC
;rcall COMP_EXT_Y
add ac_resply,resp_low ;Acumulo para despues promediar
adc ac_resphy,resp_high
dec promedio
brne INICIO_PROMEDIO
;sbi TP_CS,TB_CS ;por algun motivo la rutina de C termina la comunicacion aca.

; lds temp_memh,xh_max
; sub ac_resphx,temp_memh ;cargo en un registro auxiliar el high y el low del maximo de x, y se lo r
; lds temp_memh,xl_max
; sbc ac_resplx,temp_memh ;
; lds temp_memh,xh_min ;repito para el valor minimo de x
; sub ac_resphx,temp_memh
; lds temp_memh,xl_min
; sbc ac_resplx,temp_memh ;

; lds temp_memh,yh_max ;Repito para Y
; sub ac_resphy,temp_memh
; lds temp_memh,yl_max
; sbc ac_resply,temp_memh
; lds temp_memh,yh_min
; sub ac_resphy,temp_memh
```

```
; lds temp_memh,yl_min
; sbc ac_resply,temp_memh

ldi promedio,4
 DIVIDO_POR_8:
lsr ac_resphx ;Shifteo el acumulador 3 veces a la derecha para dividir por 8
ror ac_resplx ;Esto es, promediar los 8 valores acumulados
lsr ac_resphy
ror ac_resply
dec promedio
brne DIVIDO_POR_8
mov Tresp_high_x,ac_resphx ;Muevo los promedios a los registros que toma como entrada la parte de la
mov Tresp_low_x,ac_resplx
mov Tresp_high_y,ac_resphy
mov Tresp_low_y,ac_resply
rcall  CONVERTIR_PIXEL
mov  r10,r2
mov  r11,r3
mov  r12,r4
ret

MAGIA_ADC: ;Rutina para recibir datos del ADC, tren de pulsos de ads784.pdf pagina 8, figura 5
;toma la orden en "orden=r16" y devuelve high y low en resp_high y resp_low
ldi contador,8
cbi TP_CS,TB_CS ;bajo CS para empezar a enviar datos
;----------------------Envio orden----------------------
cbi  TP_CLK,TB_CLK
ENVIO_ORDEN:
lsl orden ;shifteo a la izquierda la orden para que el mas significativo quede en el carry
brcs SETEO_DIN
cbi TP_DIN,TB_DIN ;si el carry es cero pongo DIN en low
rjmp DIN_SETEADO ;y salto a DIN_SETEADO
SETEO_DIN:
sbi TP_DIN,TB_DIN
DIN_SETEADO: ;DIN ya tiene el dato que corresponde,
cbi TP_CLK,TB_CLK ;ahora tengo que hacer titilar el clock para que el ADC levante el dato
sbi TP_CLK,TB_CLK
dec contador
brne  ENVIO_ORDEN
sbi TP_CLK,TB_CLK ;en este punto ya está cargada la orden en el ADC.
cbi TP_CLK,TB_CLK
;----------------------Recepcion de datos----------------
ldi contador,12
ldi resp_high,0x00
ldi resp_low,0x00
RECIBO_DATO:
; lsl resp_low ;shifteo a la izquierda el low de la respuesta para que el msb quede en carry
; rol resp_high ;shifteo a la izquierda, y pongo el carry en el lsb
sbi TP_CLK,TB_CLK
cbi  TP_CLK,TB_CLK
sbic TPIN_DOUT,TB_DOUT ;Mira el bit de DOUT, si esta en low, no hace nada, sino le suma uno a respue
inc resp_low
lsl resp_low ;shifteo a la izquierda el low de la respuesta para que el msb quede en carry
rol resp_high ;shifteo a la izquierda, y pongo el carry en el lsb
dec  contador
brne RECIBO_DATO
;En este punto recibi los 12 bits de lo que devuelve el ADC, pongo el CS en high y listo
sbi TP_CS,TB_CS ;seteo CS para terminar el envio de datos.
```

```
ldi contador,0xF0 ;Uso el contador como variable auxiliar.
and contador,resp_high ;si 0xF0 and resp_high no es cero, tengo un valor fuera de rango, lo trunco en
breq LEIDO_EN_RANGO
ldi resp_high,0x00
ldi resp_low,0x00
LEIDO_EN_RANGO:
ret




CONVERTIR_PIXEL:;-------------------------------------------------------------------------------
;Esta es la cuenta que tengo que hacer:
; VIEJOS X=(TP_X-Ttouch_x_left)*Tdisp_x_size/(Ttouch_x_right - Ttouch_x_left);
; Y=(TP_Y-Ttouch_y_top) *Tdisp_y_size/(Ttouch_y_bottom - Ttouch_y_top);
; Tdisp_x_size/(Ttouch_x_right - Ttouch_x_left)= (239)/(870-222)=0.368827160=0x0.5E
; Tdisp_y_size/(Ttouch_y_bottom - Ttouch_y_top)=(319)/(341-3853)=-0.090831435=0x0.17
;X=(TP_X - touch_y_top) *(disp_y_size)) /(touch_y_bottom - touch_y_top);
; disp_y_size/(touch_y_bottom-touch_y_top)=319/(341-3853)=-0.090831435=0x0.17
;Y=(TP_Y - touch_x_left)*(-disp_x_size))/(touch_x_right - touch_x_left) + disp_x_size;
; disp_x_size/(touch_x_right-touch_x_left)

;Factores de conversión:

;La defino como positiva e invierto el sentido en que se hace la resta:TP_Y-Ttouch_y_top -> Ttouch_y

;Esta es la funcion que voy a implementar para la coma fija:
; a b , 0
; 0 0 , c
;--------
;H(ac)|L(ac)+H(cb)|L(cb)|0 >> 16
;Basicamente es el producto entre ab0 y 00c shifteado 16 bits a la derecha.
; Eso es, descarto los dos registros menos significativos

;empiezo por x:
;segun la forma en que estan definidas las constantes, el eje x es el mas chico, que entra en un solo
;le resto el offset Ttouch_x_left
ldi Tconst_conv,TCONV_X
ldi Taux,low(Ttouch_x_left)
neg Taux
ldi Taux_h,0x00
com Taux_h
add Tresp_low_x,Taux
adc Tresp_high_x,Taux_h
;hago el producto
mul Tresp_low_x,Tconst_conv
mov r3,r1
mul Tresp_high_x,Tconst_conv
mov r2,r1
add r3,r0
brcc NO_CARRY_X
inc r2
NO_CARRY_X:
mov r2,r3


;Ahora con y:
```

```
ldi Tconst_conv,TCONV_Y
ldi Taux,low(Ttouch_y_top)
ldi Taux_h,high(Ttouch_y_top)
;tengo que ver si Tresp_low_y es 0x00 para ver si tengo que usar neg o com en Tresp_high_y
mov  r30,Tresp_low_y
cpi r30,0x00
breq RLOWY_NULL
;si estoy aca es porque Tresp_low_y no es cero
neg Tresp_low_y
com Tresp_high_y
rjmp RLOWY_NOT_NULL
RLOWY_NULL:
;si estoy aca es porque Tresp_low_y es cero
neg Tresp_high_y
RLOWY_NOT_NULL:
;resto el offset
add Tresp_low_y,Taux
adc Tresp_high_y,Taux_h
;hago el producto
mul Tresp_low_y,Tconst_conv
mov r4,r1
mul Tresp_high_y,Tconst_conv
mov r3,r1
add r4,r0
brcc NO_CARRY_Y
inc r3
NO_CARRY_Y:
ret

COMP_EXT_X:
lds temp_memh,xh_max
lds temp_meml,xl_max
cp resp_low,temp_meml
cpc resp_high,temp_memh
brcs COMP_X_NOT_MAX
sts xh_max,resp_high
sts xl_max,resp_low
COMP_X_NOT_MAX:
lds temp_memh,xh_min
lds temp_meml,xl_min
cp resp_low,temp_meml
cpc resp_high,temp_memh
brcc COMP_X_NOT_MIN
sts xh_min,resp_high
sts xl_min,resp_low
COMP_X_NOT_MIN:
ret

COMP_EXT_Y:
lds temp_memh,yh_max
lds temp_meml,yl_max
cp resp_low,temp_meml
cpc resp_high,temp_memh
brcs COMP_Y_NOT_MAX
sts yh_max,resp_high
sts yl_max,resp_low
COMP_Y_NOT_MAX:
lds temp_memh,yh_min
```

```
lds temp_meml,yl_min
cp resp_low,temp_meml
cpc resp_high,temp_memh
brcc COMP_Y_NOT_MIN
sts yh_min,resp_high
sts yl_min,resp_low
COMP_Y_NOT_MIN:
    ret

;%%%%%%%%%%%%%%%% TABLAS CON VALORES DE INICIALIZACION %%%%%%%%%%%%%%%%%%%%%%%%
;.ORG $200

;la secuencia de valores es : com , dat_h , dat_low

INIT_COM_DATA_VALUES_ILI9325C:
.db 0xE5,0x78,0xF0,0x01,0x01,0x00,0x02,0x07,0x00,0x03,0x10,0x30,0x04,0x00,0x00,0x08,0x02,0x07,0x09,0x
```

A.2. IO.mac

```
.MACRO LOADIO

ldi r16,@1
out @0,r16

.ENDMACRO

.MACRO DATA_TX

CHECK:
; Wait for empty transmit buffer
lds r16,UCSR0A

sbrs r16,UDRE0

rjmp CHECK

sts UDR0,@0 ; Put data (r16) into buffer, sends the data

.ENDMACRO
```

A.3. tft_colors.inc

```
.equ VGA_BLACK = 0x0000
.equ VGA_WHITE = 0xFFFF
.equ VGA_RED = 0xF800
.equ VGA_GREEN = 0x0400
.equ VGA_BLUE = 0x001F
.equ VGA_SILVER = 0xC618
.equ VGA_GRAY = 0x8410
.equ VGA_MAROON = 0x8000
.equ VGA_YELLOW = 0xFFE0
.equ VGA_OLIVE = 0x8400
.equ VGA_LIME = 0x07E0
.equ VGA_AQUA = 0x07FF
.equ VGA_TEAL = 0x0410
.equ VGA_NAVY = 0x0010
.equ VGA_FUCHSIA= 0xF81F
```

```
.equ VGA_PURPLE =   0x8010
```

A.4. tft_macros.mac

```
.MACRO SET_XY_AREA ;@0 x_init_h @1 x_init_l @2 y_init @3 x_final_h @4 x_final_l @5 y_final (y_max = 1

ldi r30,0x20
ldi r31,0
mov R29,@2 ;CARGO Y inicial
WR_COM r30
WR_DATA r31,r29
ldi r30,0x21
mov r31,@0 ;CARGO X_h inicial
mov R29,@1 ;CARGO X_l inicial
WR_COM r30
WR_DATA r31,r29
ldi r30,0x50
ldi r31,0
mov R29,@2 ;CARGO Y inicial
WR_COM r30
WR_DATA r31,r29
ldi r30,0x52
mov r31,@0 ;CARGO X_h inicial
mov R29,@1 ;CARGO X_l inicial
WR_COM r30
WR_DATA r31,r29
ldi r30,0x51
mov r31,@5 ;cargo y final
LDI R29,0
WR_COM r30
WR_DATA r29,r31
ldi r30,0x53
mov r31,@3 ;cargo x final
mov r29,@4
WR_COM r30
WR_DATA r31,r29
ldi r30,0x22
WR_COM r30

.ENDMACRO


.MACRO WR_COM ;recibe un registro de 8 bits en @0

cbi PORTC,B_RS
LOADIO PORTD,0
cbi PORTC,B_WR
sbi PORTC,B_WR
OUT PORTD,@0
cbi PORTC,B_WR
SBI PORTC,B_WR

.ENDMACRO

.MACRO WR_DATA ;recibe un registro en @0 data_high y en @1 data_low

sbi PORTC,B_RS
OUT PORTD,@0
```

```
cbi PORTC,B_WR
SBI PORTC,B_WR
OUT PORTD,@1
cbi PORTC,B_WR
SBI PORTC,B_WR

.ENDMACRO

.MACRO DRAW_PIXEL ;@0 x_pix_h @1 x_pix_low @2 y_pix
;copio los parametros recibidos a otro registro para mandarlos a otra macro
mov pixel_x_high,@0 ;x_pix_h
mov pixel_x_low,@1 ;x_pix_low
mov pixel_y,@2 ;y_pix

cbi PORTC,B_CS
SET_XY_AREA r3,r2,r1,r3,r2,r1

ldi r16,low(PIXEL_COLOR) ;cargo la parte baja del color con el que lleno el pixel
ldi r17,higH(PIXEL_COLOR) ;cargo la parte alta del color con el que lleno el pixel

WR_DATA r17,r16
sbi PORTC,B_CS

ldi r16,0 ;x_init_h,x_init_low,y_init
ldi r17,high(tft_max_x) ;x_final_h
ldi r18,low(tft_max_x) ;x_final_low
ldi r19,tft_max_y
SET_XY_AREA r16,r16,r16,r17,r18,r19

.ENDMACRO

.MACRO FILL_SCRN ;lleno la pantalla con un color, recibe como parametro un color en RGB565 (word)

ldi r16,0 ;x_init_h,x_init_low,y_init
ldi r19,high(tft_max_x) ;x_final_h
ldi r20,low(tft_max_x);x_final_l
ldi r21,tft_max_y ;y_final

cbi PORTC,B_CS

SET_XY_AREA  r16,r16,R16,r19,r20,r21;;@0 x_init_h @1 x_init_l @2 y_init @3 x_final_h @4x_final_l @5 y

sbi PORTC,B_RS
;para llenar la pantalla necesito repetir un loop 76800 veces, para ello armo 2 loops
;uno de 65535 y otro de 11265
;cargo 65535 en dos registros
ldi r29,$ff
ldi r28,$ff
;cargo 11265 en dos registros
ldi r25,$2c
ldi r24,$01

ldi c_high,HIGH(@0)
ldi c_low,LOW(@0)

START_FILL_1:

out PORTD,c_high
```

```
cbi PORTC,B_WR
sbi PORTC,B_WR
OUT PORTD, c_low
cbi PORTC,B_WR
SBI PORTC,B_WR
sbiw r29:r28,1
brne START_FILL_1
;hasta aca ejecute la rutina 65535 veces, ahora hago un loop de 11265 para completar los 76800

START_FILL_2:

out PORTD,c_high
cbi PORTC,B_WR
sbi PORTC,B_WR
OUT PORTD, c_low
cbi PORTC,B_WR
SBI PORTC,B_WR

sbiw r25:r24,1
brne START_FILL_2

sbi PORTC,B_CS
.ENDMACRO
```

B. Caracteristicas del microcontrolador

8-bit AVR Microcontrollers

ATmega328/P

DATASHEET COMPLETE

Introduction

The Atmel® picoPower® ATmega328/P is a low-power CMOS 8-bit microcontroller based on the AVR® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega328/P achieves throughputs close to 1MIPS per MHz. This empowers system designer to optimize the device for power consumption versus processing speed.

Feature

High Performance, Low Power Atmel®AVR® 8-Bit Microcontroller Family

- Advanced RISC Architecture
  - 131 Powerful Instructions
  - Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 32KBytes of In-System Self-Programmable Flash program Memory
  - 1KBytes EEPROM
  - 2KBytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data Retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Atmel® QTouch® Library Support
  - Capacitive Touch Buttons, Sliders and Wheels
  - QTouch and QMatrix® Acquisition
  - Up to 64 sense channels

- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    - Temperature Measurement
  - Two Master/Slave SPI Serial Interface
  - One Programmable Serial USART
  - One Byte-oriented 2-wire Serial Interface (Philips I^2C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - One On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V
- Temperature Range:
  - -40°C to 105°C
- Speed Grade:
  - 0 - 4MHz @ 1.8 - 5.5V
  - 0 - 10MHz @ 2.7 - 5.5V
  - 0 - 20MHz @ 4.5 - 5.5V
- Power Consumption at 1MHz, 1.8V, 25°C
  - Active Mode: 0.2mA
  - Power-down Mode: 0.1µA
  - Power-save Mode: 0.75µA (Including 32kHz RTC)

C. Hoja de datos ILI9325D

1. Introduction

ILI9325 is a 262,144-color one-chip SoC driver for a-TFT liquid crystal display with resolution of 240RGBx320 dots, comprising a 720-channel source driver, a 320-channel gate driver, 172,800 bytes RAM for graphic data of 240RGBx320 dots, and power supply circuit.

ILI9325 has four kinds of system interfaces which are i80-system MPU interface (8-/9-/16-/18-bit bus width), VSYNC interface (system interface + VSYNC, internal clock, DB[17:0]), serial data transfer interface (SPI) and RGB 6-/16-/18-bit interface (DOTCLK, VSYNC, HSYNC, ENABLE, DB[17:0]).

In RGB interface and VSYNC interface mode, the combined use of high-speed RAM write function and widow address function enables to display a moving picture at a position specified by a user and still pictures in other areas on the screen simultaneously, which makes it possible to transfer display the refresh data only to minimize data transfers and power consumption.

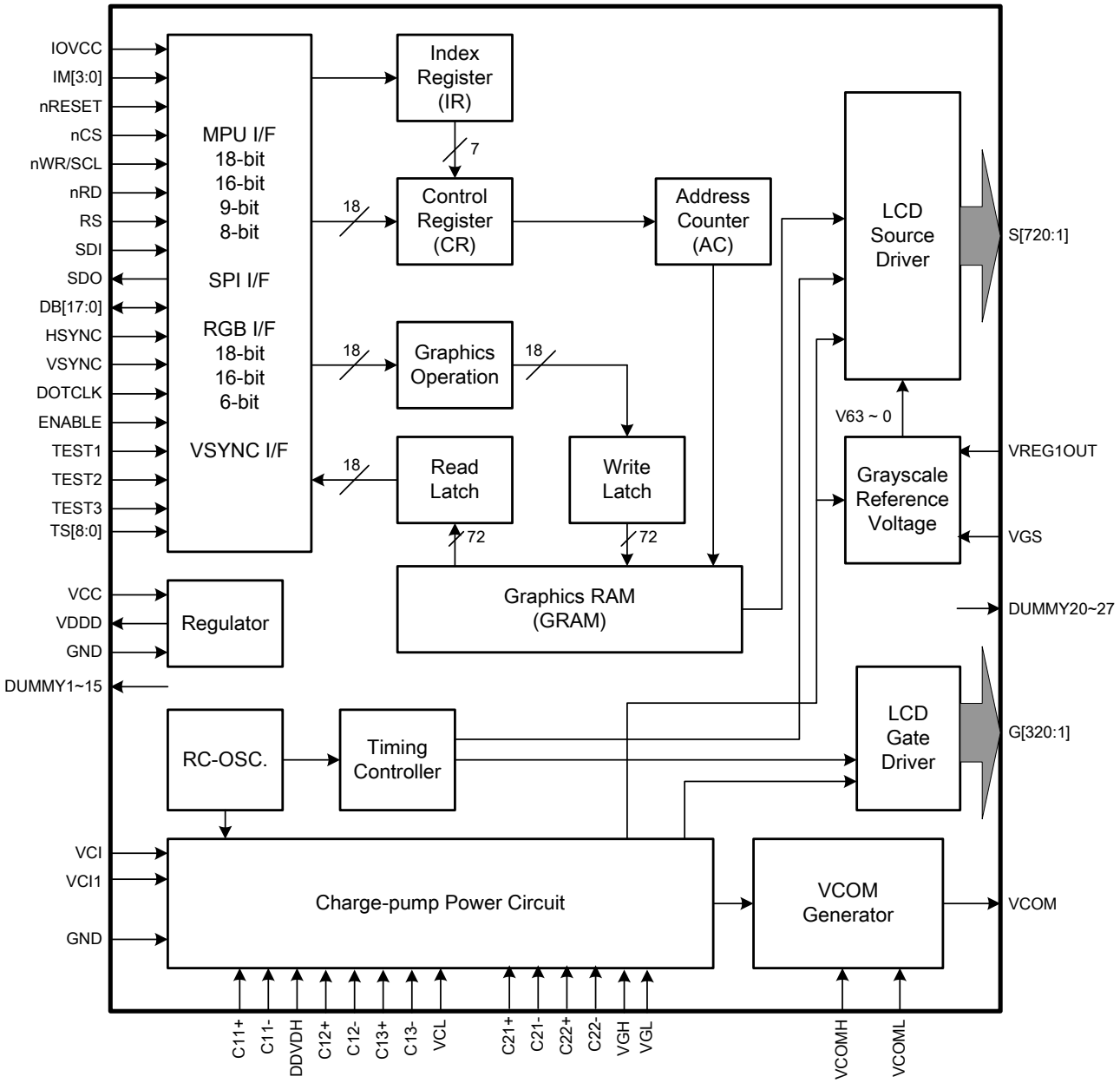
ILI9325 can operate with 1.65V I/O interface voltage, and an incorporated voltage follower circuit to generate voltage levels for driving an LCD. The ILI9325 also supports a function to display in 8 colors and a sleep mode, allowing for precise power control by software and these features make the ILI9325 an ideal LCD driver for medium or small size portable products such as digital cellular phones, smart phone, PDA and PMP where long battery life is a major concern.

2. Features

- Single chip solution for a liquid crystal QVGA TFT LCD display
- 240RGBx320-dot resolution capable with real 262,144 display color
- Support MVA (Multi-domain Vertical Alignment) wide view display
- Incorporate 720-channel source driver and 320-channel gate driver
- Internal 172,800 bytes graphic RAM
- High-speed RAM burst write function
- System interfaces
  - i80 system interface with 8-/ 9-/16-/18-bit bus width
  - Serial Peripheral Interface (SPI)
  - RGB interface with 6-/16-/18-bit bus width (VSYNC, HSYNC, DOTCLK, ENABLE, DB[17:0])
  - VSYNC interface (System interface + VSYNC)
- Internal oscillator and hardware reset
- Resizing function (×1/2, ×1/4)
- Reversible source/gate driver shift direction
- Window address function to specify a rectangular area for internal GRAM access
- Bit operation function for facilitating graphics data processing
  - Bit-unit write data mask function
  - Pixel-unit logical/conditional write function

- Abundant functions for color display control
  - γ-correction function enabling display in 262,144 colors
  - Line-unit vertical scrolling function
- Partial drive function, enabling partially driving an LCD panel at positions specified by user
- Incorporate step-up circuits for stepping up a liquid crystal drive voltage level up to 6 times (x6)
- Power saving functions
  - 8-color mode
  - standby mode
  - sleep mode
- Low -power consumption architecture
  - Low operating power supplies:
    - IOVcc = 1.65V ~ 3.3 V (interface I/O)
    - Vcc = 2.4V ~ 3.3 V (internal logic)
    - Vci = 2.5V ~ 3.3 V (analog)
- LCD Voltage drive:
  - Source/VCOM power supply voltage
    - DVDH - GND = 4.5V ~ 6.0
    - VCL – GND = -2.0V ~ -3.0V
    - VCI – VCL ≦ 6.0V
  - Gate driver output voltage
    - VGH - GND = 10V ~ 20V
    - VGL – GND = -5V ~ -15V
    - VGH – VGL ≦ 32V
  - VCOM driver output voltage
    - VCOMH = 3.0V ~ (DDVDH-0.5)V
    - VCOML = (VCL+0.5)V ~ 0V
    - VCOMH-VCOML ≦ 6.0V
- a-TFT LCD storage capacitor: Cst only

3. Block Diagram



4. Pin Descriptions

<table>
<tr><th>Pin Name</th><th>I/O</th><th>Type</th><th>Descriptions</th></tr>
<tr><td colspan="4">Input Interface</td></tr>
<tr><td>IM3,
IM2,
IM1,
IM0/ID</td><td>I</td><td>IOVcc</td><td>Select the MPU system interface mode
<table>
<tr><th>IM3</th><th>IM2</th><th>IM1</th><th>IM0</th><th>MPU-Interface Mode</th><th>DB Pin in use</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>Setting invalid</td><td></td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>Setting invalid</td><td></td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>i80-system 16-bit interface</td><td>DB[17:10], DB[8:1]</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>i80-system 8-bit interface</td><td>DB[17:10]</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>ID</td><td>Serial Peripheral Interface (SPI)</td><td>SDI, SDO</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>*</td><td>Setting invalid</td><td></td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>Setting invalid</td><td></td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>Setting invalid</td><td></td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>i80-system 18-bit interface</td><td>DB[17:0]</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>i80-system 9-bit interface</td><td>DB[17:9]</td></tr>
<tr><td>1</td><td>1</td><td>*</td><td>*</td><td>Setting invalid</td><td></td></tr>
</table>
When the serial peripheral interface is selected, IM0 pin is used for the device code ID setting.</td></tr>
<tr><td>nCS</td><td>I</td><td>MPU
IOVcc</td><td>A chip select signal.
Low: the ILI9325 is selected and accessible
High: the ILI9325 is not selected and not accessible
Fix to the DGND level when not in use.</td></tr>
<tr><td>RS</td><td>I</td><td>MPU
IOVcc</td><td>A register select signal.
Low: select an index or status register
High: select a control register
Fix to either IOVcc or DGND level when not in use.</td></tr>
<tr><td>nWR/SCL</td><td>I</td><td>MPU
IOVcc</td><td>A write strobe signal and enables an operation to write data when the signal is low.
Fix to either IOVcc or DGND level when not in use.

SPI Mode:
Synchronizing clock signal in SPI mode.</td></tr>
<tr><td>nRD</td><td>I</td><td>MPU
IOVcc</td><td>A read strobe signal and enables an operation to read out data when the signal is low.
Fix to either IOVcc or DGND level when not in use.</td></tr>
<tr><td>nRESET</td><td>I</td><td>MPU
IOVcc</td><td>A reset pin.
Initializes the ILI9325 with a low input. Be sure to execute a power-on reset after supplying power.</td></tr>
<tr><td>SDI</td><td>I</td><td>MPU
IOVcc</td><td>SPI interface input pin.
The data is latched on the rising edge of the SCL signal.</td></tr>
<tr><td>SDO</td><td>O</td><td>MPU
IOVcc</td><td>SPI interface output pin.
The data is outputted on the falling edge of the SCL signal.

Let SDO as floating when not used.</td></tr>
<tr><td>DB[17:0]</td><td>I/O</td><td>MPU
IOVcc</td><td>An 18-bit parallel bi-directional data bus for MPU system interface mode
8-bit I/F: DB[17:10] is used.
9-bit I/F: DB[17:9] is used.
16-bit I/F: DB[17:10] and DB[8:1] is used.
18-bit I/F: DB[17:0] is used.

18-bit parallel bi-directional data bus for RGB interface operation
6-bit RGB I/F: DB[17:12] are used.</td></tr>
</table>

Pin Name	I/O	Type	Descriptions
			16-bit RGB I/F: DB[17:13] and DB[11:1] are used. 18-bit RGB I/F: DB[17:0] are used. Unused pins must be fixed to DGND level.
ENABLE	I	MPU IOVcc	Data ENEABLE signal for RGB interface operation. Low: Select (access enabled) High: Not select (access inhibited) The EPL bit inverts the polarity of the ENABLE signal. Fix to either IOVcc or DGND level when not in use.
DOTCLK	I	MPU IOVcc	Dot clock signal for RGB interface operation. DPL = "0": Input data on the rising edge of DOTCLK DPL = "1": Input data on the falling edge of DOTCLK Fix to the DGND level when not in use
VSYNC	I	MPU IOVcc	Frame synchronizing signal for RGB interface operation. VSPL = "0": Active low. VSPL = "1": Active high. Fix to the DGND level when not in use.
HSYNC	I	MPU IOVcc	Line synchronizing signal for RGB interface operation. HSPL = "0": Active low. HSPL = "1": Active high. Fix to the DGND level when not in use
FMARK	O	MPU IOVcc	Output a frame head pulse signal. The FMARK signal is used when writing RAM data in synchronization with frame. Leave the pin open when not in use.
LCD Driving signals			
S720~S1	O	LCD	Source output voltage signals applied to liquid crystal. To change the shift direction of signal outputs, use the SS bit. SS = "0", the data in the RAM address "h00000" is output from S1. SS = "1", the data in the RAM address "h00000" is output from S720. S1, S4, S7, … display red (R), S2, S5, S8, ... display green (G), and S3, S6, S9, ... display blue (B) (SS = 0).
G320~G1	O	LCD	Gate line output signals. VGH: the level selecting gate lines VGL: the level not selecting gate lines
VCOM	O	TFT common electrode	A supply voltage to the common electrode of TFT panel. VCOM is AC voltage alternating signal between the VCOMH and VCOML levels.
VCOMH	O	Stabilizing capacitor	The high level of VCOM AC voltage. Connect to a stabilizing capacitor.
VCOML	O	Stabilizing capacitor	The low level of VCOM AC voltage. Adjust the VCOML level with the VDV bits. Connect to a stabilizing capacitor.
VGS	I	GND or external resistor	Reference level for the grayscale voltage generating circuit. The VGS level can be changed by connecting to an external resistor.
Charge-pump and Regulator Circuit			
Vci	I	Power supply	A supply voltage to the analog circuit. Connect to an external power supply of 2.5 ~ 3.3V.
GND	I	Power supply	GND for the analog side: GND = 0V. In case of COG, connect to GND on the FPC to prevent noise.
Vci1	O	Stabilizing capacitor	An internal reference voltage for the step-up circuit1. The amplitude between Vci and DGND is determined by the VC[2:0] bits. Make sure to set the Vci1 voltage so that the DDVDH, VGH and VGL voltages are set within the respective specification.
DDVDH	O	Stabilizing	Power supply for the source driver and Vcom drive.

Pin Name	I/O	Type	Descriptions
		capacitor	
VGH	I	Stabilizing capacitor	Power supply for the gate driver.
VGL	I	Stabilizing capacitor	Power supply for the gate driver.
VCL	O	Stabilizing capacitor	VcomL driver power supply. VCL = 0.5 ~ –VCI . Place a stabilizing capacitor between GND
C11+, C11- C12+, C12-	I/O	Step-up capacitor	Capacitor connection pins for the step-up circuit 1.
C13+, C13- C21+, C21- C22+, C22-	I/O	Step-up capacitor	Capacitor connection pins for the step-up circuit 2.
VREG1OUT	I/O	Stabilizing capacitor	Output voltage generated from the reference voltage. The voltage level is set with the VRH bits. VREG1OUT is (1) a source driver grayscale reference voltage, (2) VcomH level reference voltage, and (3) Vcom amplitude reference voltage. Connect to a stabilizing capacitor. VREG1OUT = 3.0 ~ (DDVDH – 0.5)V.
Power Pads			
Vcc	I	Power supply	A supply voltage to the internal logic: Vcc = 2.4~3.3V
IOVcc	I	Power supply	A supply voltage to the interface pins: IM[3:0], nRESET, nCS, nWR, nRD, RS, DB[17:0], VSYNC, HSYNC, DOTCLK, ENABLE, SCL, SDI, SDO. IOVcc = 1.65 ~ 3.3V and Vcc ≧IOVcc. In case of COG, connect to Vcc on the FPC if IOVcc=Vcc, to prevent noise.
VDDD	O	Power	Digital circuit power pad. Connect these pins with the 1uF capacitor.
GND	I	Power supply	GND for the analog side: DGND = 0V.
Test Pads			
DUMMY1~ 15 DUMMY20 ~ 27	-	-	Dummy pad. Leave these pins as open.
IOGNDDUM	O	GND	GND pin.
TESTO1~16	O	Open	Test pins. Leave them open.
TEST1, 2, 3	I	IOGND	Test pins (internal pull low). Connect to GND or leave these pins as open.
TS0~8	I	OPEN	Test pins (internal pull low). Leave them open.

Liquid crystal power supply specifications Table 1

No.	Item		Description
1	TFT Source Driver		720 pins (240 x RGB)
2	TFT Gate Driver		320 pins
3	TFT Display's Capacitor Structure		Cst structure only (Common VCOM)
4	Liquid Crystal Drive Output	S1 ~ S720	V0 ~ V63 grayscales
		G1 ~ G320	VGH - VGL
		VCOM	VCOMH - VCOML: Amplitude = electronic volumes

5	Input Voltage	IOVcc	1.65 ~ 3.30V
		Vcc	2.40 ~ 3.30V
		Vci	2.50 ~ 3.30V
6	Liquid Crystal Drive Voltages	DDVDH	4.5V ~ 6.0V
		VGH	10V ~ 20V
		VGL	-5V ~ -15V
		VCL	-1.9V ~ -3.0V
		VGH - VGL	Max. 32V
		Vci - VCL	Max. 6.0V
7	Internal Step-up Circuits	DDVDH	Vci1 x2
		VGH	Vci1 x4, x5, x6
		VGL	Vci1 x-3, x-4, x-5
		VCL	Vci1 x-1

D. Hoja de datos ADS7843

ADS7843

SBAS090B – SEPTEMBER 2000 – REVISED MAY 2002

TOUCH SCREEN CONTROLLER

FEATURES

- 4-WIRE TOUCH SCREEN INTERFACE
- RATIOMETRIC CONVERSION
- SINGLE SUPPLY: 2.7V to 5V
- UP TO 125kHz CONVERSION RATE
- SERIAL INTERFACE
- PROGRAMMABLE 8- OR 12-BIT RESOLUTION
- 2 AUXILIARY ANALOG INPUTS
- FULL POWER-DOWN CONTROL

APPLICATIONS

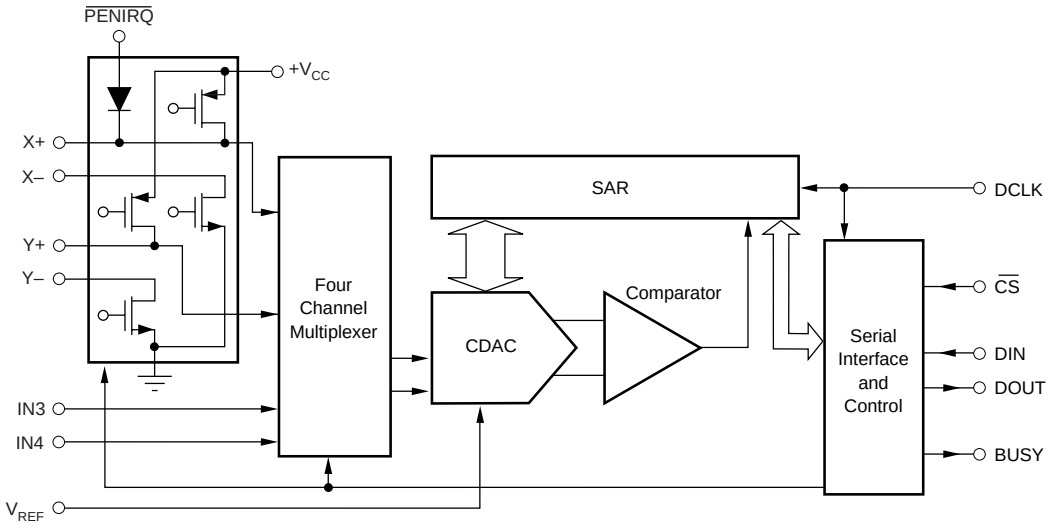
- PERSONAL DIGITAL ASSISTANTS
- PORTABLE INSTRUMENTS
- POINT-OF-SALES TERMINALS
- PAGERS
- TOUCH SCREEN MONITORS

DESCRIPTION

The ADS7843 is a 12-bit sampling Analog-to-Digital Converter (ADC) with a synchronous serial interface and low on-resistance switches for driving touch screens. Typical power dissipation is 750µW at a 125kHz throughput rate and a +2.7V supply. The reference voltage (V_{REF}) can be varied between 1V and $+V_{CC}$, providing a corresponding input voltage range of 0V to V_{REF}. The device includes a shutdown mode which reduces typical power dissipation to under 0.5µW. The ADS7843 is specified down to 2.7V operation.

Low power, high speed, and onboard switches make the ADS7843 ideal for battery-operated systems such as personal digital assistants with resistive touch screens and other portable equipment. The ADS7843 is available in an SSOP-16 package and is specified over the –40°C to +85°C temperature range.

US Patent No. 6246394



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

ABSOLUTE MAXIMUM RATINGS[1]

$+V_{CC}$ to GND	–0.3V to +6V
Analog Inputs to GND	–0.3V to $+V_{CC}$ + 0.3V
Digital Inputs to GND	–0.3V to $+V_{CC}$ + 0.3V
Power Dissipation	250mW
Maximum Junction Temperature	+150°C
Operating Temperature Range	–40°C to +85°C
Storage Temperature Range	–65°C to +150°C
Lead Temperature (soldering, 10s)	+300°C

NOTE: (1) Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum conditions for extended periods may affect device reliability.

ELECTROSTATIC DISCHARGE SENSITIVITY

This integrated circuit can be damaged by ESD. Texas Instruments recommends that all integrated circuits be handled with appropriate precautions. Failure to observe proper handling and installation procedures can cause damage.

ESD damage can range from subtle performance degradation to complete device failure. Precision integrated circuits may be more susceptible to damage because very small parametric changes could cause the device not to meet its published specifications.

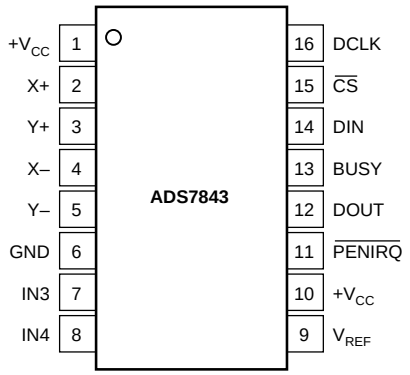
PACKAGE/ORDERING INFORMATION

PRODUCT	MAXIMUM INTEGRAL LINEARITY ERROR (LSB)	PACKAGE-LEAD	PACKAGE DESIGNATOR[1]	SPECIFIED TEMPERATURE RANGE	PACKAGE MARKING	ORDERING NUMBER	TRANSPORT MEDIA, QUANTITY
ADS7843E	±2	SSOP-16	DBQ	–40°C to +85°C	ADS7843E	ADS7843E	Rails, 100
"	"	"	"	"	ADS7843E	ADS7843E/2K5	Tape and Reel, 2500

NOTES: (1) For the most current specifications and package information, refer to our web site at www.ti.com.

PIN CONFIGURATION

Top View **SSOP**



PIN DESCRIPTION

PIN	NAME	DESCRIPTION
1	$+V_{CC}$	Power Supply, 2.7V to 5V.
2	X+	X+ Position Input. ADC input Channel 1.
3	Y+	Y+ Position Input. ADC input Channel 2.
4	X–	X– Position Input
5	Y–	Y– Position Input
6	GND	Ground
7	IN3	Auxiliary Input 1. ADC input Channel 3.
8	IN4	Auxiliary Input 2. ADC input Channel 4.
9	V_{REF}	Voltage Reference Input
10	$+V_{CC}$	Power Supply, 2.7V to 5V.
11	$\overline{PENIRQ}$	Pen Interrupt. Open anode output (requires 10kΩ to 100kΩ pull-up resistor externally).
12	DOUT	Serial Data Output. Data is shifted on the falling edge of DCLK. This output is high impedance when $\overline{CS}$ is HIGH.
13	BUSY	Busy Output. This output is high impedance when $\overline{CS}$ is HIGH.
14	DIN	Serial Data Input. If $\overline{CS}$ is LOW, data is latched on rising edge of DCLK.
15	$\overline{CS}$	Chip Select Input. Controls conversion timing and enables the serial input/output register.
16	DCLK	External Clock Input. This clock runs the SAR conversion process and synchronizes serial data I/O.

ELECTRICAL CHARACTERISTICS

At T_A = –40°C to +85°C, $+V_{CC}$ = +2.7V, V_{REF} = +2.5V, f_{SAMPLE} = 125kHz, f_{CLK} = 16 • f_{SAMPLE} = 2MHz, 12-bit mode, and digital inputs = GND or $+V_{CC}$, unless otherwise noted.

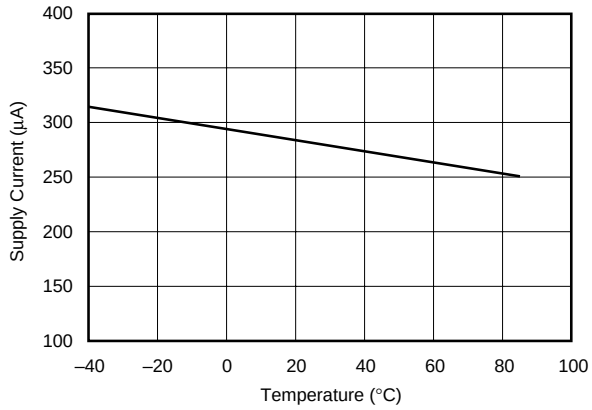
PARAMETER	CONDITIONS	ADS7843E MIN	ADS7843E TYP	ADS7843E MAX	UNITS
ANALOG INPUT					
Full-Scale Input Span	Positive Input – Negative Input	0		V_{REF}	V
Absolute Input Range	Positive Input	–0.2		$+V_{CC}$ +0.2	V
	Negative Input	–0.2		+0.2	V
Capacitance			25		pF
Leakage Current			0.1		µA
SYSTEM PERFORMANCE					
Resolution			12		Bits
No Missing Codes		11			Bits
Integral Linearity Error				±2	LSB(1)
Offset Error				±6	LSB
Offset Error Match			0.1	1.0	LSB
Gain Error				±4	LSB
Gain Error Match			0.1	1.0	LSB
Noise			30		µVrms
Power-Supply Rejection			70		dB
SAMPLING DYNAMICS					
Conversion Time				12	Clk Cycles
Acquisition Time		3			Clk Cycles
Throughput Rate				125	kHz
Multiplexer Settling Time			500		ns
Aperture Delay			30		ns
Aperture Jitter			100		ps
Channel-to-Channel Isolation	V_{IN} = 2.5Vp-p at 50kHz		100		dB
SWITCH DRIVERS					
On-Resistance					
Y+, X+			5		Ω
Y–, X–			6		Ω
REFERENCE INPUT					
Range		1.0		$+V_{CC}$	V
Resistance	$\overline{CS}$ = GND or $+V_{CC}$		5		GΩ
Input Current			13	40	µA
	f_{SAMPLE} = 12.5kHz		2.5		µA
	$\overline{CS}$ = $+V_{CC}$		0.001	3	µA
DIGITAL INPUT/OUTPUT					
Logic Family			CMOS		
Logic Levels, Except $\overline{PENIRQ}$					
V_{IH}	$\lvert I_{IH} \rvert \le +5\mu A$	$+V_{CC}$ • 0.7		$+V_{CC}$ +0.3	
V_{IL}	$\lvert I_{IL} \rvert \le +5\mu A$	–0.3		+0.8	V
V_{OH}	I_{OH} = –250µA	$+V_{CC}$ • 0.8			V
V_{OL}	I_{OL} = 250µA			0.4	V
$\overline{PENIRQ}$					
V_{OL}	T_A = 0°C to +85°C, 100kΩ Pull-Up			0.8	V
Data Format			Straight Binary		
POWER-SUPPLY REQUIREMENTS					
$+V_{CC}$	Specified Performance	2.7		3.6	V
Quiescent Current			280	650	µA
	f_{SAMPLE} = 12.5kHz		220		µA
	Shutdown Mode with DCLK = DIN = $+V_{CC}$			3	µA
Power Dissipation	$+V_{CC}$ = +2.7V			1.8	mW
TEMPERATURE RANGE					
Specified Performance		–40		+85	°C

NOTE: (1) LSB means Least Significant Bit. With V_{REF} equal to +2.5V, 1LSB is 610µV.

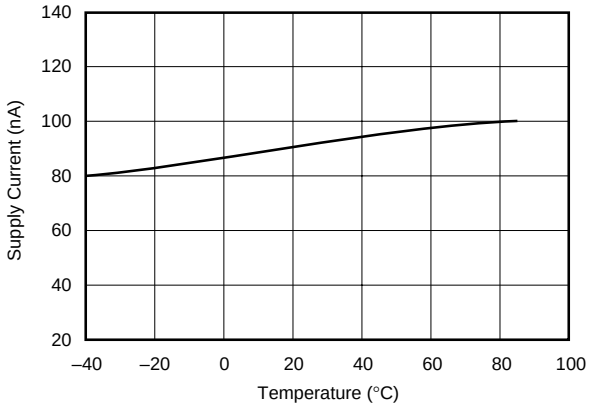
TYPICAL CHARACTERISTICS

At T_A = +25°C, $+V_{CC}$ = +2.7V, V_{REF} = +2.5V, f_{SAMPLE} = 125kHz, and f_{CLK} = 16 • f_{SAMPLE} = 2MHz, unless otherwise noted.

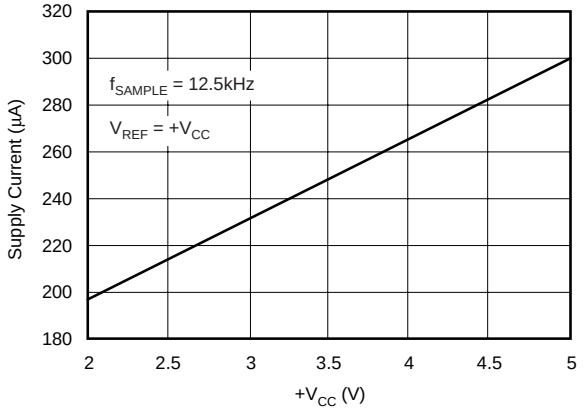
SUPPLY CURRENT vs TEMPERATURE



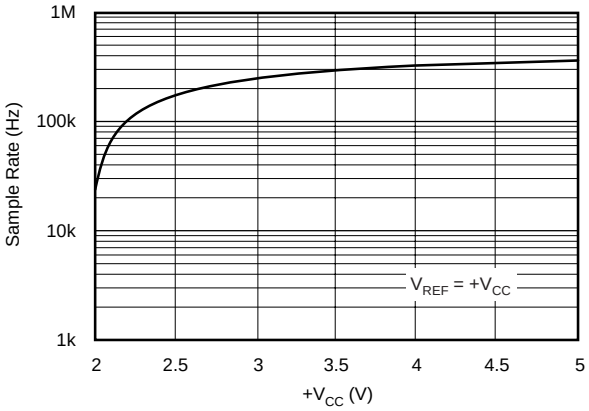
POWER-DOWN SUPPLY CURRENT vs TEMPERATURE



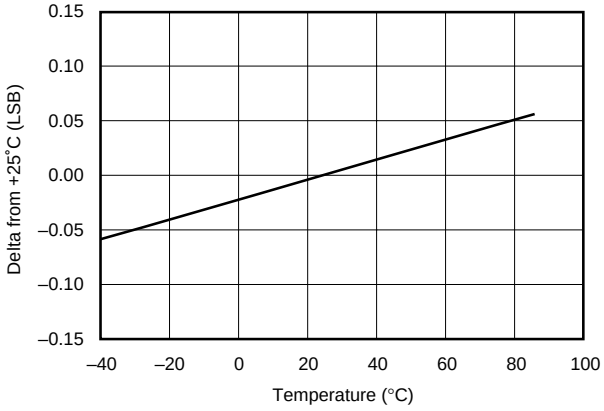
SUPPLY CURRENT vs $+V_{CC}$



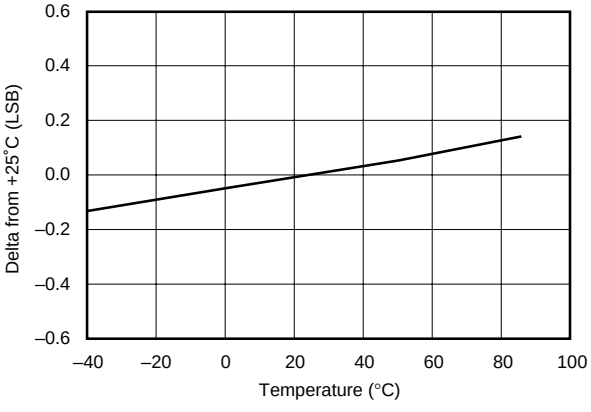
MAXIMUM SAMPLE RATE vs $+V_{CC}$



CHANGE IN GAIN vs TEMPERATURE



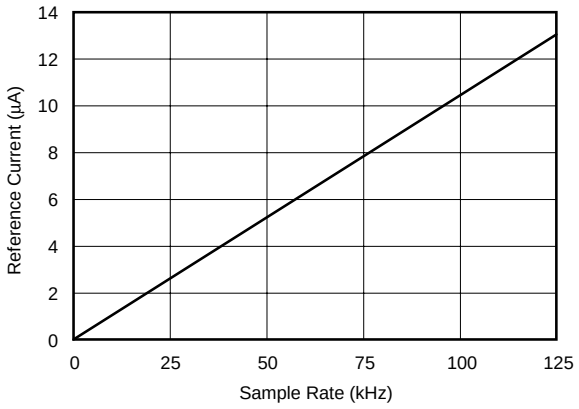
CHANGE IN OFFSET vs TEMPERATURE



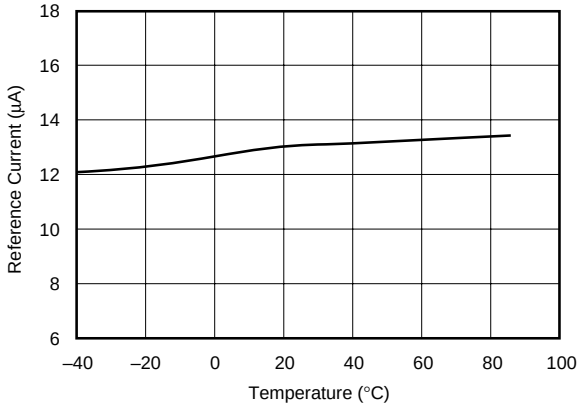
TYPICAL CHARACTERISTICS (Cont.)

At T_A = +25°C, $+V_{CC}$ = +2.7V, V_{REF} = +2.5V, f_{SAMPLE} = 125kHz, and f_{CLK} = 16 • f_{SAMPLE} = 2MHz, unless otherwise noted.

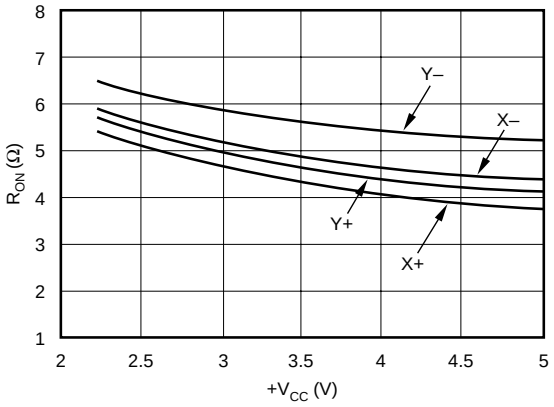
REFERENCE CURRENT vs SAMPLE RATE



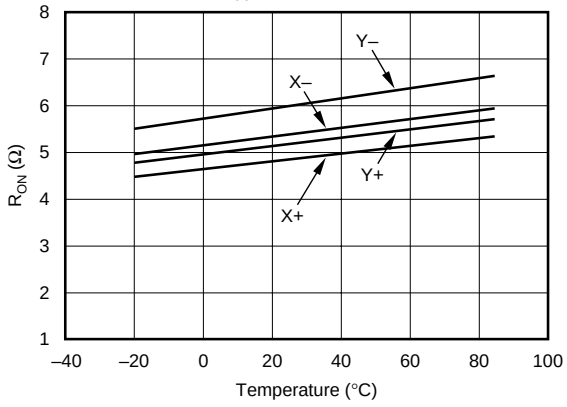
REFERENCE CURRENT vs TEMPERATURE



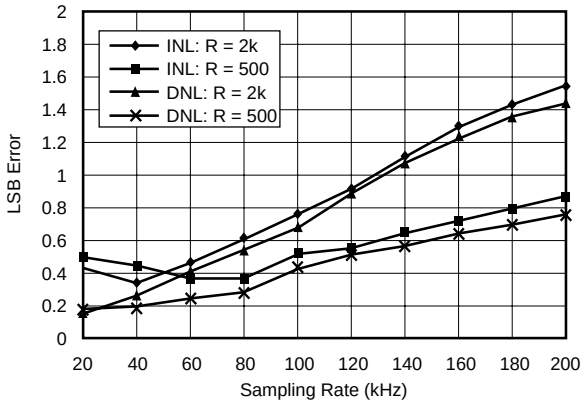
SWITCH-ON RESISTANCE vs $+V_{CC}$
(X+, Y+: $+V_{CC}$ to Pin; X–, Y–: Pin to GND)



SWITCH-ON RESISTANCE vs TEMPERATURE
(X+, Y+: $+V_{CC}$ to Pin; X–, Y–: Pin to GND)



MAXIMUM SAMPLING RATE vs R_{IN}



E. Hoja de datos del display

CONTENTS PAGE

1. Features & Mechanical Specifications

Item	Contents	Unit
	LCD	
LCD Type	TFT Transmissive Normal White	--
Viewing direction	12:00	--
Backlight	White LED x4 in Parallel	--
Interface	16bit parallel bus interface	--
Driver IC	ILI9325	--
Outline Dimension	42.72(W) × 60.26(H) × 3.7(T)	mm
Glass area (W×H×T)	41.1 ×57.1× 1.44	mm
Active area (W×H)	36.72×48.96	mm
Number of Dots	240(RGB) × 320	--
Pixel pitch (W×H)	0.153 × 0.153	mm
Operating Temperature	-20 ～ +70	℃
Storage temperature	-30 ～ +80	℃

2. Dimensional Outline

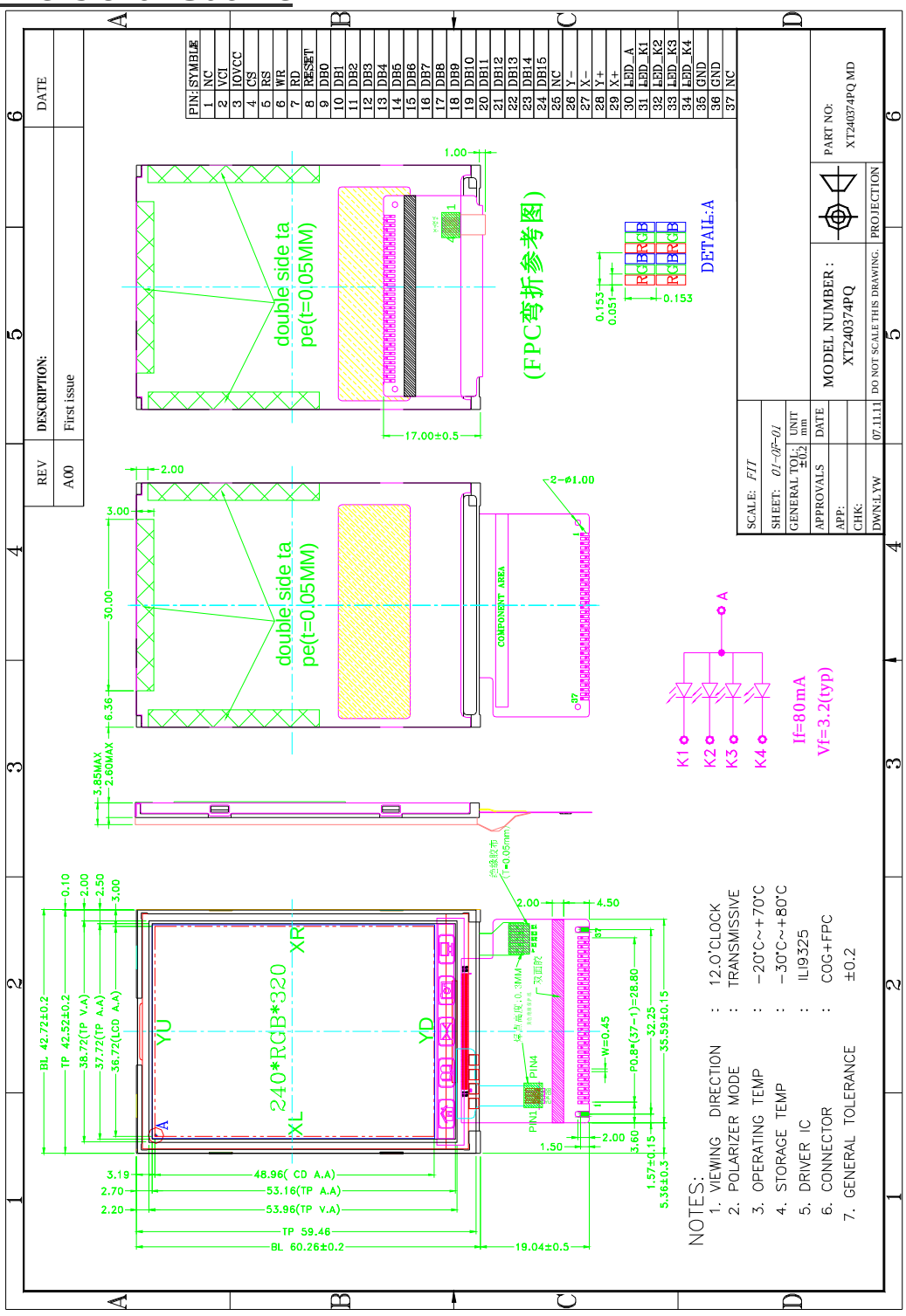


Figure 1. Dimensional outline

3. Block Diagram

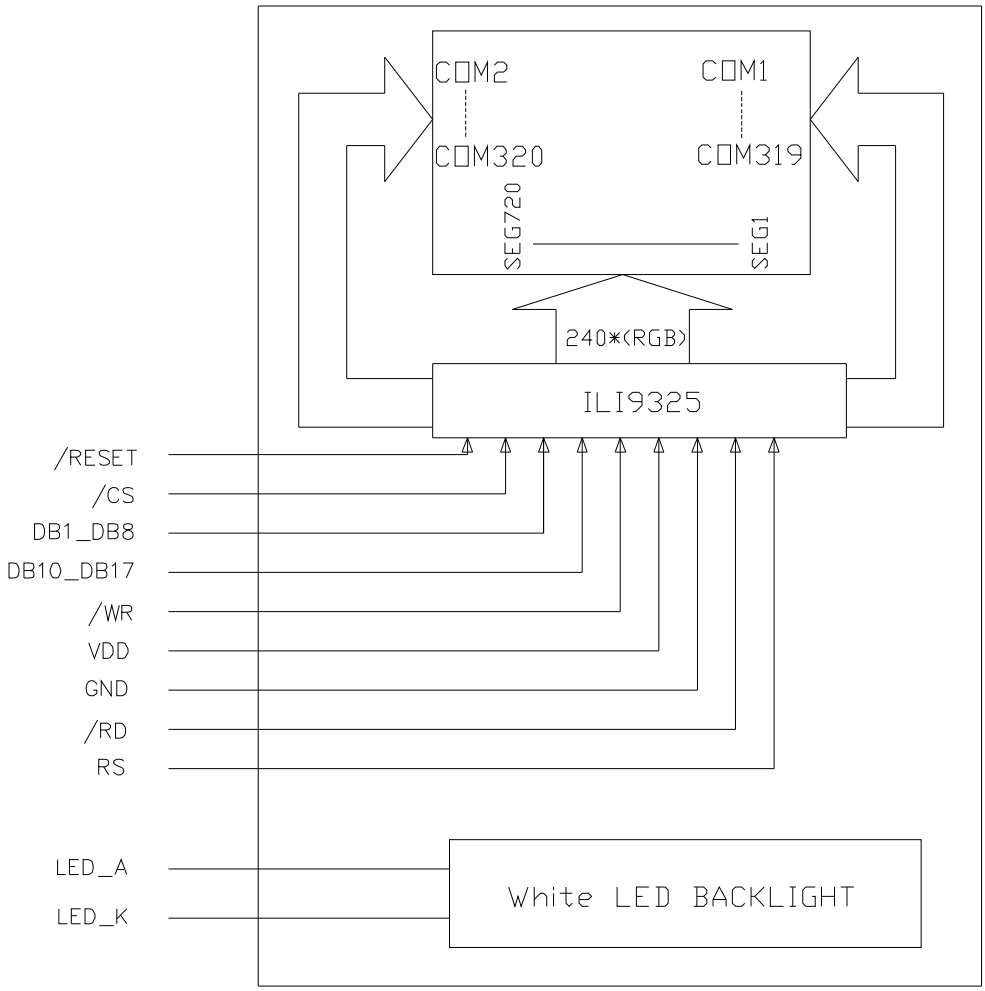


Figure 2. Block diagram

4. Pin Description

PIN No.	SYMBOL	Function
1	NC	NC
2	VDD	Power supply
3	VDD	Power supply
4	CS	Chip Select input pin. (Active Low)
5	RS	Data or command select pin. “H”: Date, “L”: Command.
6	WR	Write signal input pin. (Active Low)
7	RD	Read signal input pin. (Active Low)
8	RESET	Reset Signal pin (“Low” is enable)
9	DB0	Data bus
10	DB1	Data bus
11	DB2	Data bus
12	DB3	Data bus
13	DB4	Data bus
14	DB5	Data bus
15	DB6	Data bus
16	DB7	Data bus
17	DB8	Data bus
18	DB9	Data bus
19	DB10	Data bus
20	DB11	Data bus
21	DB12	Data bus
22	DB13	Data bus
23	DB14	Data bus
24	DB15	Data bus
25	NC	NC
26	Y-	Touch panel contrl signal pin
27	X-	Touch panel contrl signal pin
28	Y+	Touch panel contrl signal pin
29	X+	Touch panel contrl signal pin
30	LEDA	Backlight LED Anode.
31	LEDK1	Backlight LED Cathode.
32	LEDK2	Backlight LED Cathode.