12. Detectar el ingreso de la pelota en la totalidad del área del aro. Para ello se utiliza 1 sensor **IR** en cada aro que envían una señal para incrementar el contador al ser bloqueados.

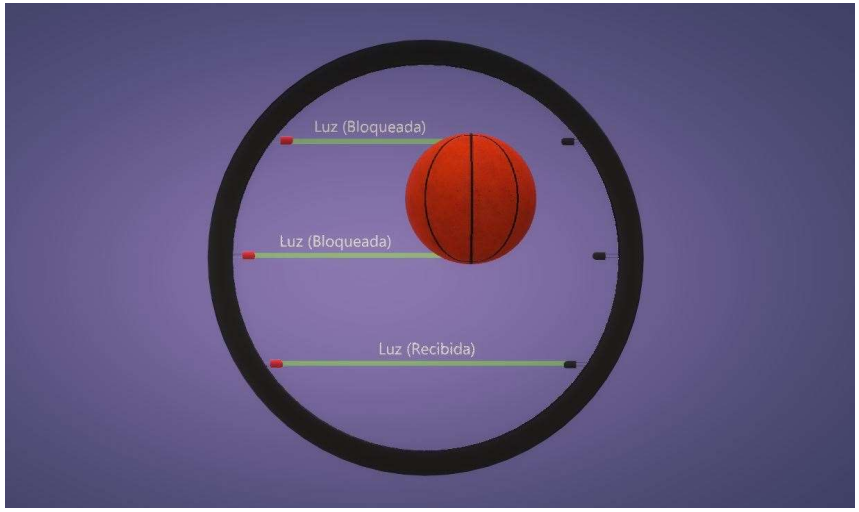


Figura 1: Sensores (En la imágen se ilustra la versión a tamaño real. En nuestra versión a escala un único sensor es suficiente).

3 Diagrama en bloques (hardware)

Se presenta a continuación un diagrama de bloques del conjunto de periféricos y el microcontrolador. Se especifican los puertos utilizados en "*Circuito Esquemático*".

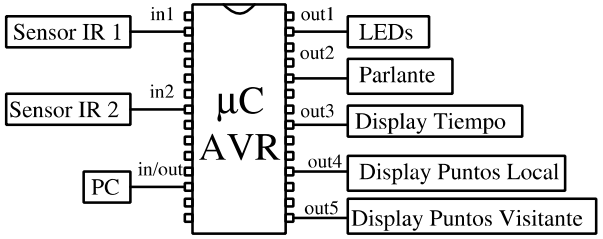


Figura 2: Diagrama en bloques del hardware.

4 Diagrama de flujo (firmware)

El programa que controla el sistema consiste principalmente en una rutina que cuenta el tiempo transcurrido mediante la utilización del timer dedicado del microcontrolador y actualiza los displays en orden regresivo en cada incremento de tiempo. Al interrupirse el haz de alguno de los sensores, se genera una interrupción que incrementa los puntos y guarda en una tabla en la memoria RAM el puntaje y el instante de anotación. Al llegar el contador a cero, se inicia la subrutina de transmisión por puerto serie y se cargan los puntos en la computadora.

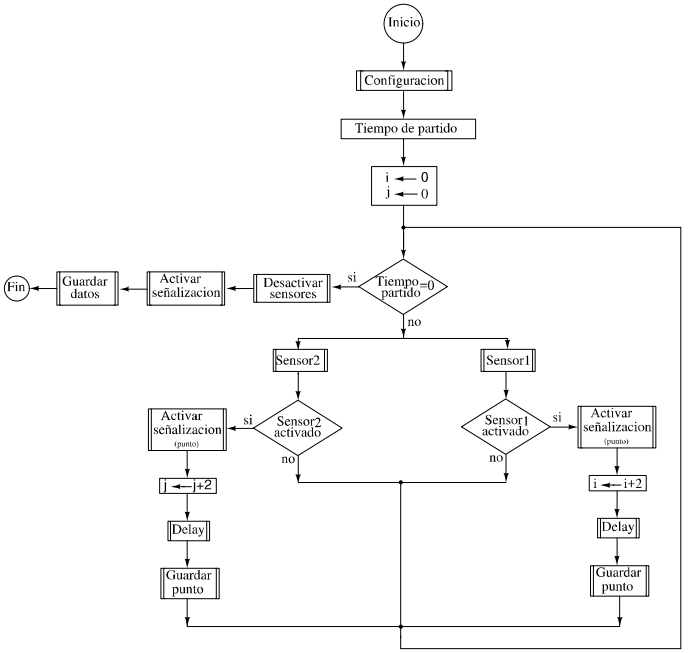


Figura 3: Diagrama de flujo.

5 Circuito Esquemático

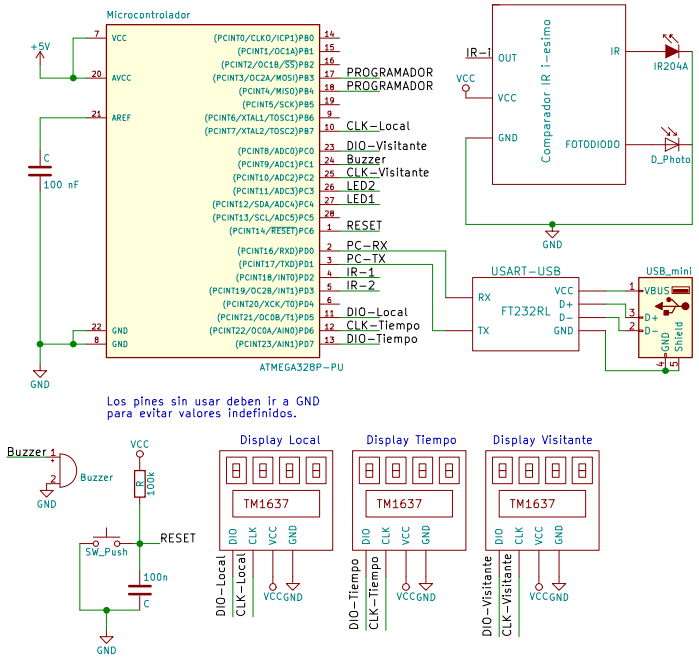


Figura 4: Esquemático.

6 Listado de componentes y costos

Se presenta a continuación una tabla con los costos de los componentes.

Componente	Cantidad	Precio (c/u) [$]
LEDs	1	100
Buzzer	1	10
FT232RL	1	80
Atmega 328p	1	75
Madera y Aros	1	300
Programador ISP	1	125
Sensor IR con comparador	2	50
Displays con cuatro 7 segmentos c/u	3	100
TOTAL	-	**1090**

7 Resultados

Se comentan a continuación una serie de aspectos a destacar.

Implementación de Displays

La solución para controlar 12 displays 7 segmentos, es decir, unos **120 segmentos** (contando puntos DP), consistió en la utilización de integrados TM1637. Los mismos cuentan con una interfaz *two wire* que permiten ser implementados como un dispositivo serie. Sin embargo, no se atienen a los protocolos I^2C ni *UART* y no permiten recibir un byte para selección de esclavo. Esto imposibilita la utilización de múltiples displays en el puerto serie dedicado del microcontrolador. Para solventar este problema, se programó una rutina que genera una señal de clock en un pin y transmite datos por otro utilizando una técnica apodada *Bit Banging*. Consecuentemente, se utilizaron 2 pines por display, llevando a un total de 6 pines para controlarlos todos. La hoja de datos del TM1637 especifica un modo de operación con direccionamiento fijo en el cual se envía un comando inicial para especificar dicho modo de operación, seguidamente se envía un segundo comando para especificar la dirección del display a escribir, luego se envía el byte dato y dicho proceso se reitera hasta escribir todos los displays. También puede enviarse un comando para ajustar el brillo. Para el envío de datos, deben enviarse señales de start (CLK en 1, flanco descendente de DIO), señales de stop (CLK en 1, flanco ascendente de DIO), señal de ACK y la transmisión de datos se realiza con el CLK en 0. Utilizando un delay de unos 50 μs para asegurar la correcta transición entre estados, se puede emular un puerto serie mediante *Bit Banging*.

Sensores IR

Para la implementación de los sensores, se optó por utilizar un integrado LM393 consistente en 2 comparadores de tensión y regulable a través de un preset. La regulación mediante el preset resulta importante dado que los sensores IR son sensibles a la iluminación ambiental. Su desempeño difiere en ambientes oscuros e iluminados. Este tipo de sensores genera un rebote al ser interrumpidos por lo cual es necesario implementar un algoritmo que evite detecciones erróneas.

Interfaz serie-USB

Para la conexión con la computadora, teniendo en cuenta que en las computadoras actuales predominan los puertos USB, se optó por utilizar un chip FT232RL de FTDI el cual genera una interfaz entre puerto serie (RX TX) y USB (D+ D- GND VCC). Dado que la complejidad de diseñar una interfaz del estilo excede la complejidad esperada del proyecto, esta resulta ser la opción más viable.

Timer

Para generar el tiempo de partido se eligió una base de 500 ms que se implementó con el *timer1*, el cuál corresponde a registros de 16 bits. Además se utilizó un *preescaler* de 8, `TCNT` de 10000 y se configuró en modo normal. En la figura 5 se muestra el registro `TCCR1B` con la configuración adecuada para la obtención del tiempo deseado.

Figura 5: Registro para la configuración del *timer*

8 Conclusiones

En el presente proyecto se desarrolló la implementación de un monitor de puntaje con sensado infrarrojo. Se cumplió el objetivo de implementarlo con una alimentación de bajo consumo. Específicamente, es posible alimentarlo con un puerto USB, cuya alimentación máxima es de 500 mA y un consumo por parte de los displays de unos 40 mA cada uno (los cuales no son alimentados por los puertos) por lo cual no requieren alimentación externa lo cual resulta realmente conveniente. Las principales complicaciones resultaron ser la programación del sistema que aplica *Bit Banging*, la correcta transmisión de los datos en RAM a la computadora a través del puerto serie y el desarrollo de un algoritmo que traduzca el tiempo transcurrido en el timer a bytes de 7 segmentos.

Por otra parte se debe destacar la versatilidad del sistema desarrollado, ya que puede ser utilizado en cualquier otra aplicación que requiera un monitoreo de interrupciones, conteo y almacenamiento de datos para un análisis estadístico posterior.

9 Apéndice: Código

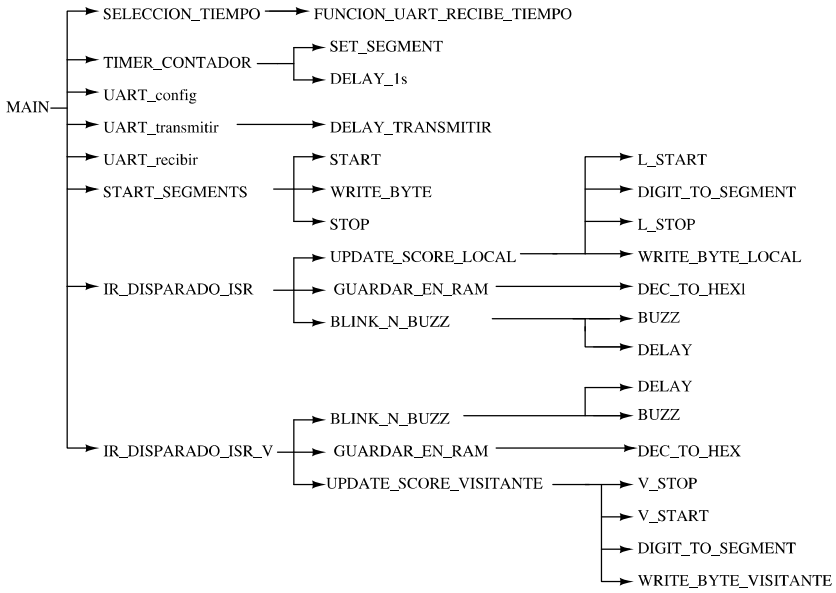


Figura 6: Estructura funcional.

En la figura 6 se muestra la estructura funcional del programa y luego la implementación.

config.inc

```
.INCLUDE "M328PDEF.INC"

.DEF BIPS_RESTANTES = r16        ;local
.DEF CONTADOR1 = r18             ;local
.DEF CONTADOR2 = r19             ;local
.DEF VISITANTE = r28             ;GLOBAL!! ;OJO: es el puntero Y,
                                 ;usar el puntero X para tabla RAM
.DEF LOCAL = r29                 ;GLOBAL!! ;OJO: es el puntero Z,
                                 ;usar el puntero X para tabla RAM
.DEF CANT_MINUTOS = r13          ;GLOBAL!!
.DEF SEGS_TRANSCURRIDOS = r0     ;GLOBAL!!
.DEF MINS_TRANSCURRIDOS = r1     ;GLOBAL!!

.EQU NRO_BIPS = 1
.EQU CANT_SEGUNDOS = 60

;----------vector interrupciones
.ORG 0
```



```
JMP CONFIG

.ORG 0x02
JMP IR_DISPARADO_ISR

.ORG 0x04
JMP IR_DISPARADO_ISR_V
;--------------------------------

.ORG 0x0100

CONFIG: ;config stack
        LDI r16, LOW(RAMEND)            ; inicializa el SP
        OUT SPL, r16
        LDI r16, HIGH(RAMEND)
        OUT SPH, r16

        ldi XL, 0x00         ;inicializacion del puntero X al
        ldi XH, 0x03         ;primer elemento de la tabla
        CLI                  ;interrupciones deshabilitadas

        ;config INT0 e INT1
        LDI r20,  0b00000011
        OUT EIMSK, r20       ;habilita interrupcion de hardware
                             ;externo 0, descendente por def
        LDI r20, 0b00001111 ;(1<<ISC00)|(1<<ISC01)
        STS EICRA, r20       ;IR por flanco ascendente

        ;config puertos
        SBI DDRC,1           ; buzzer
        SBI DDRC,2           ; tira led 1
        SBI DDRC,3           ; tira led 2
        CBI DDRD,2           ; sensor local INT0
        CBI DDRD,3           ; sensor visitante INT1

        ;Inicializo tantos y tiempo
        CLR LOCAL
        CLR VISITANTE
        LDI R17, 60          ; empieza en 60 para ser regresivo.
        MOV R0, R17
        CLR R1               ; empieza en 0 para ser regresivo.
        CLR CANT_MINUTOS     ; clr para standby
        RCALL UPDATE_SCORE_LOCAL
        RCALL UPDATE_SCORE_VISITANTE

        clr r31              ;   tiempo transcurrido
        clr r30              ;   ultimo instante de interrupcion

        RJMP MAIN
```

main.asm

```
.INCLUDE "config.inc" ; micro, stack, puertos, defs, equs, etc
.INCLUDE "ISR_ir_disparado_l.asm"
.INCLUDE "ISR_ir_disparado_v.asm"
.INCLUDE "Funciones_UART.asm"
```

```
.INCLUDE "M328Pdef.inc"
.INCLUDE "set_segment.asm"
.INCLUDE "TIMER.asm"
.INCLUDE "start_segments.asm"

MAIN:
    rcall START_SEGMENTS
    rcall USART_config
    rcall SELECCION_TIEMPO
    SEI
    rcall TIMER_CONTADOR
    cli                     ;se deshabilita la interrupcion
    rcall USART_transmitir

REINICIAR_PARTIDO:
    cli
    lds r17, UCSR0A         ;verifica si llego dato NUEVO
    sbrs r17, RXC0
    rjmp REINICIAR_PARTIDO

    lds r16, UDR0           ;carga lo que recibio y
                    ;esta en el buffer
    ldi r18, 'r'
    cp r18, r16
    breq SI_REINICIAR       ;si llego 'r' incrementa visitante
    rjmp REINICIAR_PARTIDO

SI_REINICIAR:
    jmp CONFIG
```

ISR_ir_disparado_l.asm

```
.INCLUDE "blink_n_buzz.asm"
.INCLUDE "update_score_LOCAL.asm"
.INCLUDE "update_score_VISITANTE.asm"
.INCLUDE "guardar_ram_v2.asm"

IR_DISPARADO_ISR:
    push r16
    IN r16, SREG        ;se guarda el SREG
    PUSH r16            ;pushea el valor de SREG
    push r17

    mov r17, r31        ;antirebote, ver r31 en TIMER_delay_1s
    SUB r17, r30
    cpi r17, 1          ;compara si pasaron 1 segundos
    brlt FIN_ISR        ;si no pasaron 1 segundos no incrementa

    RCALL BLINK_N_BUZZ
    INC LOCAL           ;doble como en basket
    INC LOCAL
    RCALL UPDATE_SCORE_LOCAL  ; updatea local
    RCALL GUARDAR_EN_RAM

    mov r30, r31        ;guarda ultimo instante de anotacion para
   antirebote
```



```

FIN_ISR:
    pop r17
    POP r16
    OUT SREG, r16
    pop r16
    RETI
```

ISR_ir_disparado_v.asm

```
.INCLUDE "blink_n_buzz_v.asm"

IR_DISPARADO_ISR_V:
    push r16
    IN r16, SREG        ;se guarda el SREG
    PUSH r16            ;pushea el valor de SREG
    push r18

    mov r18, r31        ;antirebote, ver r31 en TIMER_delay_1s
    SUB r18, r25
    cpi r18, 1          ;compara si pasaron 1 segundos
    brlt FIN_ISR_V      ;si no pasaron 1 segundos no incrementa

    RCALL BLINK_N_BUZZ_V
    INC VISITANTE
    INC VISITANTE
    RCALL UPDATE_SCORE_VISITANTE    ; updatea visitante
    RCALL GUARDAR_EN_RAM

    mov r25, r31        ;guarda ultimo instante de anotacion para
   antirebote

FIN_ISR_V:
    pop r18
    POP r16
    OUT SREG, r16
    pop r16
    RETI
```

TIMER_delay_500ms.asm

```
DELAY_500ms:push R20
            in R20, SREG            ;se guarda el SREG
            push R20                ;se pushea el valor de SREG
            ldi R20, high(10000)    ;se carga la parte alta y baja
            sts TCNT1H, R20         ;de TCNT1, por ser de 16 bits
            ldi R20, low(10000)
            sts TCNT1L,R20
            ldi R20, 0x00           ;no se usa, se carga con ceros
            sts TCCR1A, R20
            ldi R20, 0x2            ;preescaler de 8
            sts TCCR1B, R20
            sbi TIFR1, TOV1         ;se limpia el flag TVO1 CON UN
   1
ACA:        in R20, TIFR1
            sbrs R20, TOV1          ;skip si TOV1 llega a overflow
```

```
            rjmp ACA
            sbi TIFR1, TOV1          ;(1<<TOV1)
                                     ;se limpia el flag de overflow
            ldi R20, 0x00
            sts TCCR1B, R20          ;paro el timer
            sts TCCR1A, R20

            push r17
            push r16
            rcall USART_recibir      ;verifica si llego l o v
            inc r31                  ;para el rebote, ver ISR
            pop r16
            pop r17

            pop R20
            out SREG, R20
            pop R20
            ret
```

TIMER.asm

```
.include "TIMER_delay_1s.asm"

TIMER_CONTADOR: push R16
                push R17
                push R0
                push R1
                in R16, SREG
                push R16

                clr R0                   ;registro para segundos
                clr R1                   ;registro para minutos
                ldi R16, CANT_SEGUNDOS
                mov R17, CANT_MINUTOS

INC_SEGUNDO:    rcall DELAY_500ms
                rcall DELAY_500ms            ;espera de 1s
                rcall SET_SEGMENT        ;actualizo display
                inc R0                   ;se incrementa el contador
                                         ;de segundos
                cp R0, R16               ;compara, para ver si
                                         ;llego a 60s
                brne INC_SEGUNDO         ;si no son iguales, vuelve
                rjmp INC_MINUTO          ;se llego al minuto


INC_MINUTO:     clr R0                   ;se limpia el registro
                                         ;de segundos
                inc R1                   ;se incrementa el contador
                                         ;de minutos

                inc r17                  ;incremento para trabajar
                                         ;con tiempo regresivo
                cp R17, R1               ;se compara con el
                                         ;tiempo elegido
                push r16                 ;guarda el flag Z para
```



```
                in r16, SREG             ;saltar en brne
                push r16
                dec r17
                pop r16
                out SREG, r16
                pop r16

                brne INC_SEGUNDO         ;vuelve a incrementar
                inc R17                  ;los segundos
                rcall SET_SEGMENT
                dec r17

                push r16                 ;fin de tabla
                ldi r16, 0x04            ;caracter EOT fin
                st X, R16                ;de transmision
                pop r16

                pop R16
                out SREG,r16
                pop R1
                pop R0
                pop R17
                pop R16
                cli

                ret                      ;termino el juego
```

blink_n_buzz.asm

```
.INCLUDE "delay_blink.asm"

;los puertos ya fueron configurados en "config.inc"

BLINK_N_BUZZ:
    PUSH R16
    IN R16, SREG
    PUSH R16

    LDI R16, NRO_BIPS

KEEP_BLINKING:
    SBI PORTC,2              ;se setea el pni2 del puerto C
                             ;(tiras de led)
    CBI PORTC,3              ;se clearea el pin3 del puerto C
                             ;(tiras de led)
    RCALL BUZZ               ;se activa el buzz (doble sonido)
    RCALL DELAY              ;espera
    CBI PORTC,2              ;apaga los leds
    SBI PORTC,3              ;apaga los leds
    RCALL BUZZ
    RCALL DELAY
    DEC BIPS_RESTANTES
    BRNE KEEP_BLINKING
    CBI PORTC,1      ;si termina de blinkear, los apaga.
    CBI PORTC,2
    CBI PORTC,3
```

```
28
29      POP R16
30      IN R16, SREG
31      POP R16
32      RET
33
34  BUZZ:
35      SBIS PORTC,1
36      RJMP BUZZ_GOES_ON
37      RJMP BUZZ_GOES_OFF
38
39  BUZZ_GOES_ON:
40      SBI PORTC,1
41      RET
42
43  BUZZ_GOES_OFF:
44      CBI PORTC,1
45      RET
```

blink_n_buzz_v.asm

```
1   .INCLUDE "delay_blink_v.asm"
2
3   ;los puertos ya fueron configurados en "config.inc"
4
5   BLINK_N_BUZZ_V:
6       PUSH R16
7       IN R16, SREG
8       PUSH R16
9
10      LDI R16, NRO_BIPS
11
12  KEEP_BLINKING_V:
13      SBI PORTC,2             ;se setea el pni2 del puerto C
14                              ;(tiras de led)
15      CBI PORTC,3             ;se clearea el pin3 del puerto C
16                              ;(tiras de led)
17      RCALL BUZZ_V                ;se activa el buzz (doble sonido)
18      RCALL DELAY             ;espera
19      CBI PORTC,2             ;apaga los leds
20      SBI PORTC,3             ;apaga los leds
21      RCALL BUZZ_V
22      RCALL DELAY
23      DEC BIPS_RESTANTES
24      BRNE KEEP_BLINKING_V
25      CBI PORTC,1     ;si termina de blinkear, los apaga.
26      CBI PORTC,2
27      CBI PORTC,3
28
29      POP R16
30      IN R16, SREG
31      POP R16
32      RET
33
34  BUZZ_V:
35      SBIS PORTC,1
```



```
36      RJMP BUZZ_GOES_ON
37      RJMP BUZZ_GOES_OFF
38
39  BUZZ_GOES_ON_V:
40      SBI PORTC,1
41      RET
42
43  BUZZ_GOES_OFF_V:
44      CBI PORTC,1
45      RET
```

delay_blink.asm

```
1   DELAY:  push CONTADOR1
2           push CONTADOR2
3           push r16
4           in r16, SREG
5           push r16
6
7           ldi  CONTADOR1, 130
8           ldi  CONTADOR2, 222
9
10  LOOP1:  dec  CONTADOR2
11          brne LOOP1
12          dec  CONTADOR1
13          brne LOOP1
14          nop
15
16          pop r16
17          out SREG, r16
18          pop r16
19          pop CONTADOR2
20          pop CONTADOR1
21          ret
```

delay_blink_v.asm

```
1   DELAY_V:    push CONTADOR1
2               push CONTADOR2
3               push r16
4               in r16, SREG
5               push r16
6
7               ldi  CONTADOR1, 130
8               ldi  CONTADOR2, 222
9
10  LOOP1_V:    dec  CONTADOR2
11              brne LOOP1_V
12              dec  CONTADOR1
13              brne LOOP1_V
14              nop
15
16              pop r16
17              out SREG, r16
18              pop r16
19              pop CONTADOR2
```

```
            pop CONTADOR1
            ret
```

set_segment.asm

```
.INCLUDE "write_byte_TIMER.s"
.INCLUDE "digit_to_segment.asm"
.INCLUDE "SELECCION_DE_TIEMPO.asm"

SET_SEGMENT:            push r1
                        push r0
                        push r16
                        push r17
                        push r18
                        IN R18, SREG
                        push R18

                        ;Hago regresivo
                        ldi r16,CANT_SEGUNDOS
                        sub r16,r0

                        ;Paso digitos a segmentos
                        clr r17         ;primer digito segs
                        clr r18         ;segundo digito segs

DIVIDIR_DIGITOS_SEG:    cpi r16,0
                        breq SEGS       ;si termino de convertir
                                        ;se transmite al display
                        dec r16
                        inc r17         ;incremento 1 seg
                                        ;(primer digito)
                        cpi r17,10      ;comparo si pasaron 10s
                        brne DIVIDIR_DIGITOS_SEG
                        inc r18         ;si pasaron 10s,
                                        ;incremento 10s,
                                ;2do digito
                        clr r17         ;paso primer digito a 0
                        cpi r18,6       ;comparo si pasaron 60s
                        brne DIVIDIR_DIGITOS_SEG
                        clr r18         ;si llego a 60s
                                        ;paso el 2do digito a 0
                        rjmp DIVIDIR_DIGITOS_SEG

SEGS:                   mov r6, r17
                        mov r7, r18

                        clr r17         ;primer digito mins
                        clr r18         ;segundo digito mins

                        ;Hago regresivo
                        mov r16,CANT_MINUTOS ;desde donde empiezo
                                             ;a contar
                        sub r16,r1           ;limite-transcurridos
                        clr r1               ;compara contra 0
                        cp r0,r1             ;si hay 00 segs -->
                        brne DIVIDIR_DIGITOS_MIN
```



```
                        inc r16              ;bajo en uno -->
                                             ;los minutos

DIVIDIR_DIGITOS_MIN:    cpi r16,0
                        breq MINS       ;si termino de convertir,
                                        ;se transmite al display
                        dec r16
                        inc r17              ;incremento 1min,
                                             ;3er digito
                        cpi r17,10           ;comparo con 10 min
                        brne DIVIDIR_DIGITOS_MIN
                        inc r18              ;si pasaron 10 min,
                                             ;incremento 10 min
                                             ;(4to digito)
                        clr r17              ;paso 3er digito a 0
                        rjmp DIVIDIR_DIGITOS_MIN

MINS:                   mov r8, r17
                        mov r9, r18

;---------------- Configuro TM1637

SET_COMMAND11:          ldi r19, 0b01000100 ;COMANDO: Normal,
                                            ;escribir datos
                                            ;fixed ad
                        rcall START
                        rcall WRITE_BYTE
                        rcall STOP

;---------------- 7 Segmentos #1 (0X:XX)

SET_COMMAND21:          ldi r19, 0b11001000 ;Seteo direccion
                                            ;del display
                        rcall START
                        rcall WRITE_BYTE


SET_DATA1:              mov r19, r9
                        rcall DIGIT_TO_SEGMENT  ;Pasa digito a
                                                ;segmento en r19
                        rcall WRITE_BYTE        ;Mando dato
                        rcall STOP

;---------------- 7 Segmentos #2 (X0:XX)

SET_COMMAND22:          ldi r19, 0b11001001 ;Seteo direccion
                                            ;del display
                        rcall START
                        rcall WRITE_BYTE

SET_DATA2:              mov r19, r8
                        rcall DIGIT_TO_SEGMENT  ;Pasa digito a
                                                ;segmento en r19
                        ori r19, 0x80           ;agrega los 2 pts
                        rcall WRITE_BYTE        ;Mando dato
                        rcall STOP
```

```
;---------------- 7 Segmentos #3 (XX:0X)

SET_COMMAND23:          ldi r19, 0b11001010 ;Seteo direccion
                                            ;del display
                        rcall START
                        rcall WRITE_BYTE

SET_DATA3:              mov r19, r7
                        rcall DIGIT_TO_SEGMENT  ;Pasa digito a
                                                ;segmento en r19
                        rcall WRITE_BYTE        ;Mando dato
                        rcall STOP

;---------------- 7 Segmentos #4 (XX:X0)

SET_COMMAND24:          ldi r19, 0b11001011 ;Seteo direccion
                                            ;del display
                        rcall START
                        rcall WRITE_BYTE

SET_DATA4:              mov r19, r6
                        rcall DIGIT_TO_SEGMENT  ;Pasa digito a
                                                ;segmento en r19
                        rcall WRITE_BYTE        ;Mando dato
                        rcall STOP

;---------------- Configuro brillo

SET_COMMAND3:           ;seteo el brillo
                        ldi r19, 0b10001010
                        rcall START
                        rcall WRITE_BYTE
                        rcall STOP
                        pop R18
                        out SREG,r18
                        pop r18
                        pop r17
                        pop r16
                        pop r0
                        pop r1
                        ret
```

start_segments.asm

```
;.include "write_byte_TIMER.asm"

START_SEGMENTS: push r1
                push r0
                push r16
                push r17
                push r18
                in R18, SREG
                push R18

                ldi r16, 0b01000000 ;PALITO DEL MEDIO
```



```
;---------------- Configuro TM1637

SSET_COMMAND11: ldi r19, 0b01000100 ;COMANDO: Normal,
                                    ;escribir datos, fixed ad.
                rcall START
                rcall WRITE_BYTE
                rcall STOP

;---------------- 7 Segmentos #1 (0X:XX)

SSET_COMMAND21: ldi r19, 0b11001000 ; Seteo direccion del display
                rcall START
                rcall WRITE_BYTE

SSET_DATA1:     mov r19, r16
                rcall WRITE_BYTE        ; Mando dato
                rcall STOP

;---------------- 7 Segmentos #2 (X0:XX)

SSET_COMMAND22: ldi r19, 0b11001001 ; Seteo direccion del display
                rcall START
                rcall WRITE_BYTE

SSET_DATA2:     mov r19, r16
                ori r19, 0x80           ;agrega los 2 puntos
                rcall WRITE_BYTE        ; Mando dato
                rcall STOP

;---------------- 7 Segmentos #3 (XX:0X)

SSET_COMMAND23: ldi r19, 0b11001010 ; Seteo direccion del display
                rcall START
                rcall WRITE_BYTE

SSET_DATA3:     mov r19, r16
                rcall WRITE_BYTE        ; Mando dato
                rcall STOP

;---------------- 7 Segmentos #4 (XX:X0)

SSET_COMMAND24: ldi r19, 0b11001011 ; Seteo direccion del display
                rcall START
                rcall WRITE_BYTE

SSET_DATA4:     mov r19, r16
                rcall WRITE_BYTE        ; Mando dato
                rcall STOP

;---------------- Configuro brillo

SSET_COMMAND3:  ;seteo el brillo
                ldi r19, 0b10001010
                rcall START
                rcall WRITE_BYTE
```

```
                rcall STOP
                pop R18
                out SREG,r18
                pop r18
                pop r17
                pop r16
                pop r0
                pop r1
                ret
```

decimal_to_hex.asm

```
;r21 es el dato

DEC_TO_HEX:
    push r16
    in r16, SREG
    push r16
    push r23
    push r24
    push r22

    clr r16
    clr r22
    clr r23
    clr r24

N1: cpi r21, 0          ;primer digito
    breq GUARDAR_CONV

    dec r21
    inc r22
    cpi r22, 10
    brne N1

N2: inc r23             ;segundo digito
    clr r22
    cpi r23, 10
    brne N1

N3: inc r24
    clr r23
    cpi r24, 9
    brne N1

GUARDAR_CONV:
    mov r16, r22
    rcall HEX_TO_ASCII
    sts 0x0250, r16
    mov r16, r23
    rcall HEX_TO_ASCII
    sts 0x0251, r16
    mov r16, r24
    rcall HEX_TO_ASCII
    sts 0x0252, r16

```



```
fin_conv:
    pop r22
    pop r24
    pop r23
    pop r16
    out SREG, r16
    pop r16
    ret

HEX_TO_ASCII:
    cpi r16, 0
    breq DEC_ES_0
    cpi r16, 1
    breq DEC_ES_1
    cpi r16, 2
    breq DEC_ES_2
    cpi r16, 3
    breq DEC_ES_3
    cpi r16, 4
    breq DEC_ES_4
    cpi r16, 5
    breq DEC_ES_5
    cpi r16, 6
    breq DEC_ES_6
    cpi r16, 7
    breq DEC_ES_7
    cpi r16, 8
    breq DEC_ES_8
    cpi r16, 9
    breq DEC_ES_9

DEC_ES_0:
    ldi r16, 0x30
    ret
DEC_ES_1:
    ldi r16, 0x31
    ret
DEC_ES_2:
    ldi r16, 0x32
    ret
DEC_ES_3:
    ldi r16, 0x33
    ret
DEC_ES_4:
    ldi r16, 0x34
    ret
DEC_ES_5:
    ldi r16, 0x35
    ret
DEC_ES_6:
    ldi r16, 0x36
    ret
DEC_ES_7:
    ldi r16, 0x37
    ret
DEC_ES_8:
```

```
    ldi r16, 0x38
    ret
DEC_ES_9:
    ldi r16, 0x39
    ret
```

write_byte_TIMER.asm

```
;R19 ES EL BYTE
;PD6 CLK
;PD7 DATA


WRITE_BYTE: push r16
            push r17
            push r18

            ldi r16, 8      ; Contador para transmitir 1 byte
            sbi DDRD,7      ; Configuro PORTB2 como salida
            sbi DDRD,6      ; Configuro PORTB3 como salida

CLK_LOW:    ;Seteo CLK en LOW
            cbi PORTD, 6    ; CLW en LOW
            rcall BIT_DELAY

SETEAR_BIT: ;Configuro bit del puerto dato
            mov r17, r19    ;mueve el dato a otro registro
            andi r17, 0x01  ;enmascara para ver el bit menos
                            ;significativo para enviarlo
            lsr r19         ;shiftea el byte a derecha
                            ;para la proxima iteracion
            cpi r17, 0
            brne BIT1

BIT0:       cbi PORTD, 7    ; pongo 0 en PB2
            rcall BIT_DELAY
            rjmp CLK_HIGH

BIT1:       sbi PORTD, 7    ; pongo 1 en PB2.
            rcall BIT_DELAY

CLK_HIGH:   ;Seteo CLK en HIGH
            sbi PORTD, 6 ; CLK en HIGH
            rcall BIT_DELAY


BYTE_END:   dec r16
            cpi r16, 0
            brne CLK_LOW ;vuelvo para transmitir
                         ;otro bit si no transmitio 8 bits.

ACK:        ;Acknowledge para ver si termine de transmitir

            ;Seteo CLK en LOW
            cbi PORTD, 6 ; CLK en LOW
            rcall BIT_DELAY
```



```

            ;Seteo CLK en HIGH
            sbi PORTD, 6 ; CLK en HIGH
            rcall BIT_DELAY

            sbis PORTD, 7;
            cbi PORTD, 7 ; CLK en LOW si no lo estaba
            rcall BIT_DELAY

            ;Seteo CLK en LOW
            cbi PORTD, 6 ; CLK en LOW
            rcall BIT_DELAY

FIN:        pop r18
            pop r17
            pop r16
            ret

BIT_DELAY:  ;delay de 50 us
            push r20
            ldi  r20, 16
            L1: dec  r20
            brne L1
            pop r20
            ret

START:      ;comienza transmision segun especifica datasheet
            cbi PORTD, 7 ; Data a LOW para iniciar.
            rcall BIT_DELAY
            ret

STOP:       ;termina transmision segun especifica datasheet
            cbi PORTD, 7 ; Data a LOW.
            rcall BIT_DELAY
            sbi PORTD, 6 ; CLK a HIGH.
            rcall BIT_DELAY
            sbi PORTD, 7 ; Data a HIGH para terminar.
            rcall BIT_DELAY
            ret
```

write_byte_LOCAL.asm

```
;R19 ES EL BYTE
;PB7 CLK
;PD5 DATA

WRITE_BYTE_LOCAL:   push r16
                    push r17
                    push r18

                    ldi r16, 8  ; Contador para transmitir 1 byte
                    sbi DDRD,5  ; Configuro PORTB2 como salida
                    sbi DDRB,7  ; Configuro PORTB3 como salida

L_CLK_LOW:  ;Seteo CLK en LOW
            cbi PORTB, 7 ; CLW en LOW
```

```
            rcall L_BIT_DELAY

L_SETEAR_BIT:   ;Configuro bit del puerto dato
                mov r17, r19    ;mueve el dato a otro registro
                andi r17, 0x01  ;enmascara para ver el bit menos
                                ;significativo para enviarlo
                lsr r19         ;shiftea el byte a derecha para
                                ;la proxima iteracion
                cpi r17, 0
                brne L_BIT1

L_BIT0:         cbi PORTD, 5    ; pongo 0 en PB2
                rcall L_BIT_DELAY
                rjmp L_CLK_HIGH

L_BIT1:         sbi PORTD, 5    ; pongo 1 en PB2.
                rcall L_BIT_DELAY

L_CLK_HIGH:     ;Seteo CLK en HIGH
                sbi PORTB, 7    ; CLK en HIGH
                rcall L_BIT_DELAY


L_BYTE_END:     dec r16
                cpi r16, 0
                brne L_CLK_LOW  ;vuelvo para transmitir otro bit
                                ;si no transmitio 8 bits.

L_ACK:          ;Acknowledge para ver si termine de transmitir

                ;Seteo CLK en LOW
                cbi PORTB, 7    ; CLK en LOW
                rcall L_BIT_DELAY

                ;Seteo CLK en HIGH
                sbi PORTB, 7    ; CLK en HIGH
                rcall L_BIT_DELAY

                sbis PORTD, 5;
                cbi PORTD, 5    ; CLK en LOW si no lo estaba
                rcall L_BIT_DELAY

                ;Seteo CLK en LOW
                cbi PORTB, 7    ; CLK en LOW
                rcall L_BIT_DELAY

L_FIN:          pop r18
                pop r17
                pop r16
                ret


L_BIT_DELAY:    ;delay de 50 us
                push r20
                ldi  r20, 16
                L_L1: dec  r20
                brne L_L1
                pop r20
                ret

L_START:        ;comienza transmision segun especifica datasheet
                cbi PORTD, 5 ; Data a LOW para iniciar.
                rcall L_BIT_DELAY
                ret

L_STOP:         ;termina transmision segun especifica datasheet
                cbi PORTD, 5 ; Data a LOW.
                rcall BIT_DELAY
                sbi PORTB, 7 ; CLK a HIGH.
                rcall L_BIT_DELAY
                sbi PORTD, 5 ; Data a HIGH para terminar.
                rcall L_BIT_DELAY
                ret
```

write_byte_VISITANTE.asm

```
;R19 ES EL BYTE
;PC2 CLK
;PC0 DATA

WRITE_BYTE_VISITANTE:   push r16
                        push r17
                        push r18

                        ldi r16, 8  ;Contador para transmitir
                                    ;1 byte
                        sbi DDRC,0  ; Configuro PORTB2 como salida
                        sbi DDRC,2  ; Configuro PORTB3 como salida

V_CLK_LOW:      ;Seteo CLK en LOW
                cbi PORTC, 2 ; CLW en LOW
                rcall V_BIT_DELAY

V_SETEAR_BIT:   ;Configuro bit del puerto dato
                mov r17, r19    ;mueve el dato a otro registro
                andi r17, 0x01  ;enmascara para ver el bit menos
                                ;significativo para enviarlo
                lsr r19         ;shiftea el byte a derecha para
                                ;la proxima iteracion
                cpi r17, 0
                brne V_BIT1

V_BIT0:         cbi PORTC, 0     ; pongo 0 en PB2
                rcall V_BIT_DELAY
                rjmp V_CLK_HIGH

V_BIT1:         sbi PORTC, 0     ; pongo 1 en PB2.
                rcall V_BIT_DELAY

V_CLK_HIGH:     ;Seteo CLK en HIGH
                sbi PORTC, 2    ; CLK en HIGH
                rcall V_BIT_DELAY
```

```
V_BYTE_END:     dec r16
                cpi r16, 0
                brne V_CLK_LOW  ;vuelvo para transmitir
                                ;otro bit si no transmite 8 bits.

V_ACK:          ;Acknowledge para ver si termine de transmitir

                ;Seteo CLK en LOW
                cbi PORTC, 2 ; CLK en LOW
                rcall V_BIT_DELAY

                ;Seteo CLK en HIGH
                sbi PORTC, 2 ; CLK en HIGH
                rcall V_BIT_DELAY

                sbis PORTC, 0;
                cbi PORTC, 0 ; CLK en LOW si no lo estaba
                rcall V_BIT_DELAY

                ;Seteo CLK en LOW
                cbi PORTC, 2 ; CLK en LOW
                rcall V_BIT_DELAY

V_FIN:          pop r18
                pop r17
                pop r16
                ret

V_BIT_DELAY:    ;delay de 50 us
                push r20
                ldi  r20, 16
                V_L1: dec  r20
                brne V_L1
                pop r20
                ret

V_START:        ;comienza transmision segun especifica datasheet
                cbi PORTC, 0 ; Data a LOW para iniciar.
                rcall V_BIT_DELAY
                ret

V_STOP:         ;termina transmision segun especifica datasheet
                cbi PORTC, 0 ; Data a LOW.
                rcall BIT_DELAY
                sbi PORTC, 2 ; CLK a HIGH.
                rcall V_BIT_DELAY
                sbi PORTC, 0 ; Data a HIGH para terminar.
                rcall V_BIT_DELAY
                ret
```

update_score_LOCAL.asm

```
.include "write_byte_LOCAL.s"

```



```
UPDATE_SCORE_LOCAL: push r16
                    push r17
                    push r18
                    push r19
                    push r20
                    in r16, SREG
                    push r16

                    clr r16
                    clr r17
                    clr r18
                    clr r20
                    mov r16, r29

DIVIDIR_DIGITOS:    cpi r16,0
                    breq PTS_SET_COMMAND11  ;si termino de
                                            ;convertir, transmite
                                            ;al display
                    dec r16
                    inc r17             ;incremento 1s, 1er digito
                    cpi r17,10          ;comparo si pasaron 10 pts
                    brne DIVIDIR_DIGITOS
                    inc r18     ;si pasaron 10 pts,
                                ;incremento 10 pts, 2do digito
                    clr r17     ;paso primer digito a 0
                    cpi r18,10  ;comparo si pasaron 100 pts
                    brne DIVIDIR_DIGITOS
                    inc r20         ;si pasaron 100 pts,
                                    ;incremento 100 pts, 3er
   digito
                    clr r18             ;paso segundo digito a 0
                    rjmp DIVIDIR_DIGITOS

;---------------- Configuro TM1637

PTS_SET_COMMAND11:  ldi r19, 0b01000100     ;COMANDO: Normal,
                                            ;escribir datos,
                                            ;fixed ad.
                    rcall L_START
                    rcall WRITE_BYTE_LOCAL
                    rcall L_STOP

;---------------- 7 Segmentos #1 (0X:XX)

PTS_SET_COMMAND21:  ldi r19, 0b11001000 ; Seteo direccion
                                        ;del display
                    rcall L_START
                    rcall WRITE_BYTE_LOCAL


PTS_SET_DATA1:      clr r19                 ;lo quiero apagado
                    rcall DIGIT_TO_SEGMENT  ;Pasa digito a
                                            ;segmento en r19
                    rcall WRITE_BYTE_LOCAL  ;Mando dato
                    rcall L_STOP

```

```
;---------------- 7 Segmentos #2 (X0:XX)

PTS_SET_COMMAND22:  ldi r19, 0b11001001 ;Seteo direccion
                                        ;del display
                    rcall L_START
                    rcall WRITE_BYTE_LOCAL

PTS_SET_DATA2:      mov r19, r20
                    rcall DIGIT_TO_SEGMENT  ;Pasa digito a
   segmento en r19
                    rcall WRITE_BYTE_LOCAL  ;Mando dato
                    rcall L_STOP

;---------------- 7 Segmentos #3 (XX:0X)

PTS_SET_COMMAND23:  ldi r19, 0b11001010     ;Seteo direccion
                                            ;del display
                    rcall L_START
                    rcall WRITE_BYTE_LOCAL

PTS_SET_DATA3:      mov r19, r18
                    rcall DIGIT_TO_SEGMENT  ;Pasa digito a
                                            ;segmento en r19
                    rcall WRITE_BYTE_LOCAL  ;Mando dato
                    rcall L_STOP

;---------------- 7 Segmentos #4 (XX:X0)

PTS_SET_COMMAND24:  ldi r19, 0b11001011     ;Seteo direccion
                                            ;del display
                    rcall L_START
                    rcall WRITE_BYTE_LOCAL

PTS_SET_DATA4:      mov r19, r17
                    rcall DIGIT_TO_SEGMENT  ;Pasa digito a
                                            ;segmento en r19
                    rcall WRITE_BYTE_LOCAL  ;Mando dato
                    rcall L_STOP

;---------------- Configuro brillo

PTS_SET_COMMAND3:   ;seteo el brillo
                    ldi r19, 0b10001010
                    rcall L_START
                    rcall WRITE_BYTE_LOCAL
                    rcall L_STOP

                    pop r16
                    out SREG, r16
                    pop r20
                    pop r19
                    pop r18
                    pop r17
                    pop r16
                    ret
```


update_score_VISITANTE.asm

```
.include "write_byte_VISITANTE.asm"

UPDATE_SCORE_VISITANTE: push r16
                        push r17
                        push r18
                        push r19
                        push r20
                        in r16, SREG
                        push r16

                        clr r16
                        clr r17
                        clr r18
                        clr r20
                        mov r16, r28

V_DIVIDIR_DIGITOS:      cpi r16,0
                        breq V_SET_COMMAND11
                        ;si termino de convertir, empiezo a
                        ;transmitir al display
                        dec r16
                        inc r17         ;incremento 1 seg
                                        ;(primer digito)
                        cpi r17,10      ;comparo si pasaron 10 pts
                        brne V_DIVIDIR_DIGITOS
                        inc r18         ;si pasaron 10 pts,
                                       ;incremento 10 pts, 2do dig
                        clr r17         ;paso primer digito a 0
                        cpi r18,10      ;comparo si pasaron 100
   pts
                        brne V_DIVIDIR_DIGITOS
                        inc r20         ;si pasaron 100 pts,
                                        ;incremento 100 pts
                                        ;(tercer digito)
                        clr r18         ;paso primer digito a 0
                        rjmp V_DIVIDIR_DIGITOS

;---------------- Configuro TM1637

V_SET_COMMAND11:        ldi r19, 0b01000100 ;COMANDO: Normal,
                                            ;escribir datos,
                                            ;fixed ad.
                        rcall V_START
                        rcall WRITE_BYTE_VISITANTE
                        rcall V_STOP

;---------------- 7 Segmentos #1 (0X:XX)

V_SET_COMMAND21:        ldi r19, 0b11001000     ; Seteo direccion
   del display
                        rcall V_START
                        rcall WRITE_BYTE_VISITANTE


V_SET_DATA1:            clr r19             ;lo quiero apagado
```

```
                        rcall DIGIT_TO_SEGMENT  ;Pasa digito a
                                                ;segmento en r19
                        rcall WRITE_BYTE_VISITANTE  ;Mando dato
                        rcall V_STOP

;---------------- 7 Segmentos #2 (X0:XX)

V_SET_COMMAND22:        ldi r19, 0b11001001 ;Seteo direccion
                                            ;del display
                        rcall V_START
                        rcall WRITE_BYTE_VISITANTE

V_SET_DATA2:            mov r19, r20
                        rcall DIGIT_TO_SEGMENT  ;Pasa digito a
                                                ;segmento en r19
                        rcall WRITE_BYTE_VISITANTE  ;Mando dato
                        rcall V_STOP

;---------------- 7 Segmentos #3 (XX:0X)

V_SET_COMMAND23:        ldi r19, 0b11001010     ;Seteo direccion
                                                  ;del display
                        rcall V_START
                        rcall WRITE_BYTE_VISITANTE

V_SET_DATA3:            mov r19, r18
                        rcall DIGIT_TO_SEGMENT  ;Pasa digito a
    segmento en r19
                        rcall WRITE_BYTE_VISITANTE  ;Mando dato
                        rcall V_STOP

;---------------- 7 Segmentos #4 (XX:X0)

V_SET_COMMAND24:        ldi r19, 0b11001011 ;Seteo direccion
                                            ;del display
                        rcall V_START
                        rcall WRITE_BYTE_VISITANTE

V_SET_DATA4:            mov r19, r17
                        rcall DIGIT_TO_SEGMENT      ;Pasa digito a
                                                      ;segmento en
    r19
                        rcall WRITE_BYTE_VISITANTE  ;Mando dato
                        rcall V_STOP

;---------------- Configuro brillo

V_SET_COMMAND3:         ;seteo el brillo
                        ldi r19, 0b10001010
                        rcall V_START
                        rcall WRITE_BYTE_VISITANTE
                        rcall V_STOP

                        pop r16
                        out SREG, r16
                        pop r20
```



```
                        pop r19
                        pop r18
                        pop r17
                        pop r16
                        ret
```

guardar_ram.asm

```
.include "decimal_to_hex.asm"

GUARDAR_EN_RAM:
                push R28
                push R29
                push R0
                push R1
                push R17
                in R17, SREG
                push R17

                LDI R17, 10     ;salto de linea
                st X+, R17
                ldi R17, 13     ;retorno de carro
                st X+, R17

                mov r21, r29
                rcall DEC_TO_HEX
                lds r29, 0x0252 ;guarda tercer digito local
                st X+, R29
                lds r29, 0x0251 ;guarda segundo digito local
                st X+, R29
                lds r29, 0x0250 ;guarda primer digito local
                st X+, R29

                ldi R17, 44     ;coma
                st X+, R17

                mov r21, r28
                rcall DEC_TO_HEX
                lds r28, 0x0252 ;guarda tercer digito visitante
                st X+, R28
                lds r28, 0x0251 ;guarda segundo digito visitante
                st X+, R28
                lds r28, 0x0250 ;guarda primer digito visitante
                st X+, R28

                ldi R17, 44     ;coma
                st X+, R17

                mov r21, r1
                rcall DEC_TO_HEX
                lds r17, 0x0251 ;guarda segundo digito minutos
                st X+, R17
                lds r17, 0x0250 ;guarda primer digito minutos
                st X+, R17

                ldi R17, 44     ;coma
```

```
                st X+, R17

                mov r21, r0
                rcall DEC_TO_HEX
                lds r17, 0x0251 ;guarda segundo digito segundos
                st X+, R17
                lds r17, 0x0250 ;guarda primer digito segundos
                st X+, R17

                pop R17
                out SREG, R17
                pop R17
                pop R1
                pop R0
                pop R29
                pop R28

                ret
```

restart_match.asm

```
REINICIAR_PARTIDO:
    push r19
    push r18
    push r17
    push r16
    in r16,SREG
    push r16

    lds r17, UCSR0A     ;verifica si llego dato NUEVO
    sbrs r17, RXC0
    rjmp SALIR

    lds r16, UDR0       ;carga lo que recibio y esta en el buffer

    ldi r18, 13         ;retorno de carro
    cp r18, r16
    breq CONFIG


NO_REINICIAR:
    pop r16
    out SREG,r16
    pop r16
    pop r17
    pop r18
    pop r19
    ret
```

Funciones_UART.asm

```
;Funciones UART para configurar, emitir tabla almacenada en
   posicion 0x0300, recibir un dato al r16 (ej)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
UART_config:    push r18
```



```
                push r17
                push r16
                in r16,SREG
                push r16

                ldi r17, 0
                ldi r16, 12         ;1100 (setea TXEN0
                                    ;(transmitter enable)
                sts UBRR0H, r17     ;UCSZ02 (tamanio de caracter))
                sts UBRR0L, r16
                ldi r18, 0x02       ;setea el doble de velocidad
                                    ;de baud rate
                sts UCSR0A, r18
                ldi r16, (1<<RXEN0)|(1<<TXEN0)  ;se activa
                                                   ;transmision y
                                                ;recepcion
                sts UCSR0B, r16
                ldi r16, (1<<USBS0)|(3<<UCSZ00) ;formato de 8bits
                                                ;y dos de stop
                sts UCSR0C, r16

                pop r16
                out SREG,r16
                pop r16
                pop r17
                pop r18
                ret


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
UART_transmitir: push r17
                 push r16
                 in r16,SREG
                 push r16

                 ldi XH, 0x0300     ;posiciono la tabla al
                 ldi XL, 0x00       ;comienzo para transmitir
                                    ;hasta encontrar el caracter
                                    ;de parada
SIGUE:
                ld r16,X+
                lds r17, UCSR0A
                sbrs r17, UDRE0
                rjmp UART_transmitir
                sts UDR0,r16            ;r20 para solo transmision
                call DELAY_TRANSMITIR
                cpi r16,0x04            ;simbolo fin de tabla RAM
                brne SIGUE

                pop r16
                out SREG,r16
                pop r16
                pop r17
                ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
UART_recibir:   push r19
                push r18
                push r17
                push r16
                in r16,SREG
                push r16

                lds r17, UCSR0A     ;verifica si llego dato NUEVO
                sbrs r17, RXC0
                rjmp SALIR

                lds r16, UDR0       ;carga lo que recibio y
                                    ;esta en el buffer
                ldi r18, 'v'
                cp r18, r16
                breq INC_VISITANTE  ;si llego 'v' incrementa
                                    ;visitante
                ldi r18, 'l'
                cp r18, r16
                breq INC_LOCAL      ;si llego 'l' incrementa
                                    ;local
                ldi r18, 'c'
                cp r18, r16
                breq DEC_VISITANTE  ;si llego 'c' decrementa
                                         ;visitante
                ldi r18, 'k'
                cp r18, r16
                breq DEC_LOCAL      ;si llego 'k' decrementa
                                    ;local
                rjmp SALIR

INC_VISITANTE:  inc r28
                rcall GUARDAR_EN_RAM
                rcall UPDATE_SCORE_VISITANTE
                rjmp SALIR

INC_LOCAL:      inc r29
                rcall GUARDAR_EN_RAM
                rcall UPDATE_SCORE_LOCAL
                rjmp SALIR

DEC_VISITANTE:  cpi r28, 0
                breq SALIR
                dec r28
                rcall GUARDAR_EN_RAM
                rcall UPDATE_SCORE_VISITANTE
                rjmp SALIR

DEC_LOCAL:      cpi r29, 0
                breq SALIR
                dec r29
                rcall GUARDAR_EN_RAM
                rcall UPDATE_SCORE_LOCAL
                rjmp SALIR


```



```
SALIR:
                pop r16
                out SREG,r16
                pop r16
                pop r17
                pop r18
                pop r19
                ret

DELAY_TRANSMITIR:
                push r18
                push r19
                push r16
                in r16, SREG
                push r16

                ldi  r18, 66
L21:            dec  r18
                brne L21

                pop r16
                out SREG, r16
                pop r16
                pop r19
                pop r18
                 ret
```

SELECCION_DE_TIEMPO.asm

```
SELECCION_TIEMPO:
    push r19
    push r18
    push r17
    push r16
    in r16,SREG
    push r16

    rcall FUNCION_UART_RECIBE_TIEMPO     ;Recibe el valor del
   tiempo y define el .EQU CANT_MINUTOS

    pop r16
    out SREG,r16
    pop r16
    pop r17
    pop r18
    pop r19
    ret


FUNCION_UART_RECIBE_TIEMPO:

NO_OPCION:      lds r17, UCSR0A
                sbrs r17, RXC0
                rjmp FUNCION_UART_RECIBE_TIEMPO
                ;Espera hasta que ingresen un valor 0,1,2 o 3
                ;(o basura)
```

```
                lds r17,UDR0

                ;indentifica si es 0,1,2 o 3 [1min, 5min, 10min o
                ;15min] ... etc.

                cpi r17,0x30      ;Valor de 0
                breq T_ES_0
                cpi r17,0x31      ;Valor de 1
                breq T_ES_1
                cpi r17,0x32      ;Valor de 2
                breq T_ES_2
                cpi r17,0x33      ;Valor de 3
                breq T_ES_3
                cpi r17,0x34      ;Valor de 4
                breq T_ES_4
                cpi r17,0x35      ;Valor de 5
                breq T_ES_5
                cpi r17,0x36      ;Valor de 6
                breq T_ES_6
                cpi r17,0x37      ;Valor de 7
                breq T_ES_7
                cpi r17,0x38      ;Valor de 8
                breq T_ES_8
                rjmp NO_OPCION
                ;salta al no ingresar ninguna
                ;de las cuatro opciones anteriores
EXIT:
                ret

T_Es_0: ldi r16, 1-1
        mov r13, r16
        rjmp EXIT

T_ES_1: ldi r16, 5-1
        mov r13, r16
        rjmp EXIT

T_ES_2: ldi r16, 10-1
        mov r13, r16
        rjmp EXIT

T_ES_3: ldi r16, 15-1
        mov r13, r16
        rjmp EXIT

T_ES_4: ldi r16, 20-1
        mov r13, r16
        rjmp EXIT

T_ES_5: ldi r16, 25-1
        mov r13, r16
        rjmp EXIT

T_ES_6: ldi r16, 30-1
        mov r13, r16
        rjmp EXIT
```



```

T_ES_7: ldi r16, 35-1
        mov r13, r16
        rjmp EXIT

T_ES_8: ldi r16, 40-1
        mov r13, r16
        rjmp EXIT
```

digit_to_segment.asm

```
//
//       A
//      ---
//  F |   | B
//      -G-
//  E |   | C
//      ---
//       D

/*0b00111111,    // 0
  0b00000110,    // 1
  0b01011011,    // 2
  0b01001111,    // 3
  0b01100110,    // 4
  0b01101101,    // 5
  0b01111101,    // 6
  0b00000111,    // 7
  0b01111111,    // 8
  0b01101111,    // 9
  0b01110111,    // A
  0b01111100,    // b
  0b00111001,    // C
  0b01011110,    // d
  0b01111001,    // E
  0b01110001,    // F
  0b10000000     // DP DOUBLE POINT   */


DIGIT_TO_SEGMENT:   cpi r19,9
                    breq ES_9

                    cpi r19,8
                    breq ES_8

                    cpi r19,7
                    breq ES_7

                    cpi r19,6
                    breq ES_6

                    cpi r19,5
                    breq ES_5

                    cpi r19,4
                    breq ES_4

                    cpi r19,3
                    breq ES_3

                    cpi r19,2
                    breq ES_2

                    cpi r19,1
                    breq ES_1

```



```
                    cpi r19,0
                    breq ES_0

ES_9:               ldi r19, 0b01101111
                    ret

ES_8:               ldi r19, 0b01111111
                    ret

ES_7:               ldi r19, 0b00000111
                    ret

ES_6:               ldi r19, 0b01111101
                    ret

ES_5:               ldi r19, 0b01101101
                    ret

ES_4:               ldi r19, 0b01100110
                    ret

ES_3:               ldi r19, 0b01001111
                    ret

ES_2:               ldi r19, 0b01011011
                    ret

ES_1:               ldi r19, 0b00000110
                    ret

ES_0:               ldi r19, 0b00111111
                    ret
```

10 Circuito esquemático de la placa de desarrollo

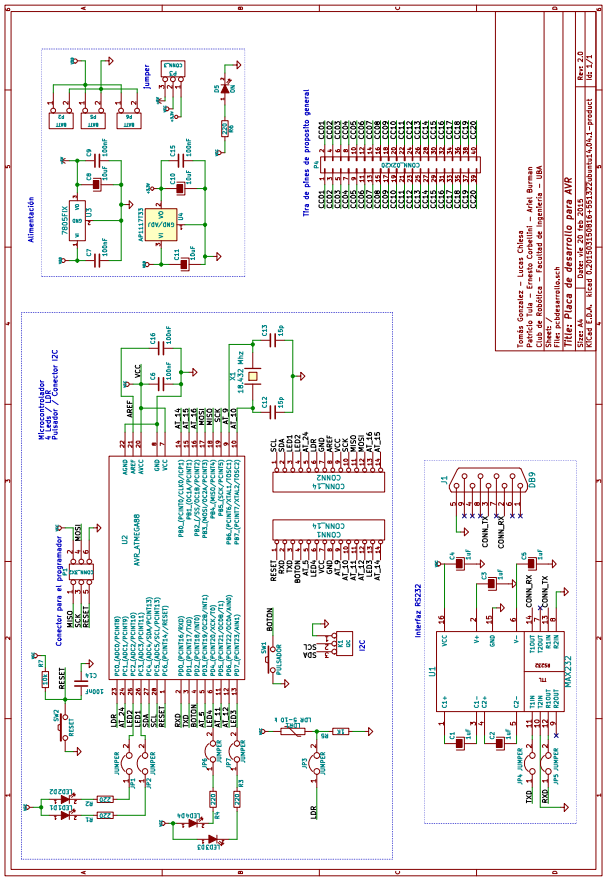


Figura 7: Esquemático.

11 Datos relevantes Atmega 328p

Atmel

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

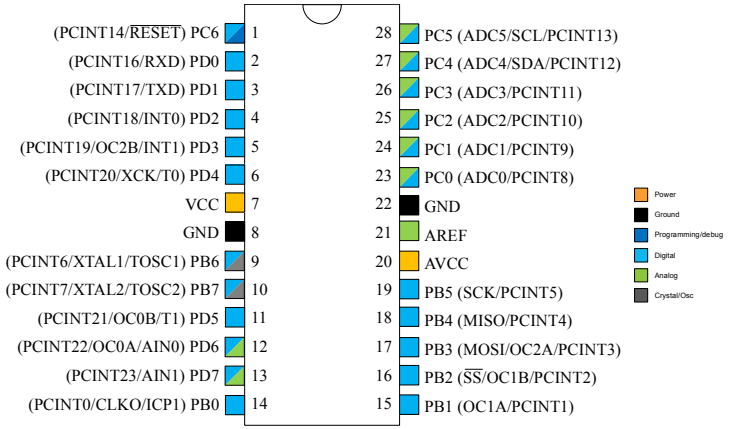
Atmel-42735B-ATmega328/P_Datasheet_Complete-11/2016

- 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)

5. Pin Configurations

5.1. Pin-out

Figure 5-1. 28-pin PDIP



16. Interrupts

This section describes the specifics of the interrupt handling of the device. For a general explanation of the AVR interrupt handling, refer to the description of *Reset and Interrupt Handling*.

- Each Interrupt Vector occupies two instruction words .
- Reset Vector is affected by the BOOTRST fuse, and the Interrupt Vector start address is affected by the IVSEL bit in MCUCR

16.1. Interrupt Vectors in ATmega328/P

Table 16-1. Reset and Interrupt Vectors in ATmega328/P

Vector No	Program Address(2)	Source	Interrupts definition
1	0x0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 0
4	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x000A	PCINT2	Pin Change Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER2_COMPA	Timer/Counter2 Compare Match A
9	0x0010	TIMER2_COMPB	Timer/Coutner2 Compare Match B
10	0x0012	TIMER2_OVF	Timer/Counter2 Overflow
11	0x0014	TIMER1_CAPT	Timer/Counter1 Capture Event
12	0x0016	TIMER1_COMPA	Timer/Counter1 Compare Match A
13	0x0018	TIMER1_COMPB	Timer/Coutner1 Compare Match B
14	0x001A	TIMER1_OVF	Timer/Counter1 Overflow
15	0x001C	TIMER0_COMPA	Timer/Counter0 Compare Match A
16	0x001E	TIMER0_COMPB	Timer/Coutner0 Compare Match B
17	0x0020	TIMER0_OVF	Timer/Counter0 Overflow
18	0x0022	SPI STC	SPI Serial Transfer Complete
19	0x0024	USART_RX	USART Rx Complete
20	0x0026	USART_UDRE	USART Data Register Empty
21	0x0028	USART_TX	USART Tx Complete
22	0x002A	ADC	ADC Conversion Complete
23	0x002C	EE READY	EEPROM Ready
24	0x002E	ANALOG COMP	Analog Comparator

Vector No	Program Address(2)	Source	Interrupts definition
25	0x0030	TWI	2-wire Serial Interface (I^2C)
26	0x0032	SPM READY	Store Program Memory Ready

Note:

1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see "Boot Loader Support – Read-While-Write Self- Programming"
2. When the IVSEL bit in MCUCR is set, Interrupt Vectors will be moved to the start of the Boot Flash Section. The address of each Interrupt Vector will then be the address in this table added to the start address of the Boot Flash Section.

The table below shows reset and Interrupt Vectors placement for the various combinations of BOOTRST and IVSEL settings. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa.

Table 16-2. Reset and Interrupt Vectors Placement

BOOTRST(1)	IVSEL	Reset Address	Interrupt Vectors Start Address
1	0	0x000	0x002
1	1	0x000	Boot Reset Address + 0x0002
0	0	Boot Reset Address	0x002
0	1	Boot Reset Address	Boot Reset Address + 0x0002

Note: 1. For the BOOTRST Fuse "1" means unprogrammed while "0" means programmed.

The most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Address   Labels   Code                 Comments
0x0000             jmp    RESET         ; Reset
0x0002             jmp    INT0          ; IRQ0
0x0004             jmp    INT1          ; IRQ1
0x0006             jmp    PCINT0        ; PCINT0
0x0008             jmp    PCINT1        ; PCINT1
0x000A             jmp    PCINT2        ; PCINT2
0x000C             jmp    WDT           ; Watchdog Timeout
0x000E             jmp    TIM2_COMPA    ; Timer2 CompareA
0x0010             jmp    TIM2_COMPB    ; Timer2 CompareB
0x0012             jmp    TIM2_OVF      ; Timer2 Overflow
0x0014             jmp    TIM1_CAPT     ; Timer1 Capture
0x0016             jmp    TIM1_COMPA    ; Timer1 CompareA
0x0018             jmp    TIM1_COMPB    ; Timer1 CompareB
0x001A             jmp    TIM1_OVF      ; Timer1 Overflow
0x001C             jmp    TIM0_COMPA    ; Timer0 CompareA
0x001E             jmp    TIM0_COMPB    ; Timer0 CompareB
0x0020             jmp    TIM0_OVF      ; Timer0 Overflow
0x0022             jmp    SPI_STC       ; SPI Transfer Complete
0x0024             jmp    USART_RXC     ; USART RX Complete
0x0026             jmp    USART_UDRE    ; USART UDR Empty
0x0028             jmp    USART_TXC     ; USART TX Complete
0x002A             jmp    ADC           ; ADC Conversion Complete
0x002C             jmp    EE_RDY        ; EEPROM Ready
0x002E             jmp    ANA_COMP      ; Analog Comparator
0x0030             jmp    TWI           ; 2-wire Serial
0x0032             jmp    SPM_RDY       ; SPM Ready
;
0x0034    RESET:   ldi    r16,high(RAMEND)  ; Main program start
0x0035             out    SPH,r16           ; Set Stack Pointer to top of RAM
0x0036             ldi    r16,low(RAMEND)
0x0037             out    SPL,r16
0x0038             sei                      ; Enable interrupts
```

17.2.1. External Interrupt Control Register A

The External Interrupt Control Register A contains control bits for interrupt sense control.

Name: EICRA
Offset: 0x69
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					ISC11	ISC10	ISC01	ISC00
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:2 – ISC1n: Interrupt Sense Control 1 [n = 1:0]
The External Interrupt 1 is activated by the external pin INT1 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT1 pin that activate the interrupt are defined in the table below. The value on the INT1 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Value	Description
00	The low level of INT1 generates an interrupt request.
01	Any logical change on INT1 generates an interrupt request.
10	The falling edge of INT1 generates an interrupt request.
11	The rising edge of INT1 generates an interrupt request.

Bits 1:0 – ISC0n: Interrupt Sense Control 0 [n = 1:0]
The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in table below. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Value	Description
00	The low level of INT0 generates an interrupt request.
01	Any logical change on INT0 generates an interrupt request.
10	The falling edge of INT0 generates an interrupt request.
11	The rising edge of INT0 generates an interrupt request.

24.12.2. USART Control and Status Register 0 A

Name: UCSR0A
Offset: 0xC0
Reset: 0x20
Property: -

Bit	7	6	5	4	3	2	1	0
	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Access	R	R/W	R	R	R	R	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bit 7 – RXC0: USART Receive Complete
This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXC0 bit will become zero. The RXC0 Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE0 bit).

Bit 6 – TXC0: USART Transmit Complete
This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR0). The TXC0 Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC0 Flag can generate a Transmit Complete interrupt (see description of the TXCIE0 bit).

Bit 5 – UDRE0: USART Data Register Empty
The UDRE0 Flag indicates if the transmit buffer (UDR0) is ready to receive new data. If UDRE0 is one, the buffer is empty, and therefore ready to be written. The UDRE0 Flag can generate a Data Register Empty interrupt (see description of the UDRIE0 bit). UDRE0 is set after a reset to indicate that the Transmitter is ready.

Bit 4 – FE0: Frame Error
This bit is set if the next character in the receive buffer had a Frame Error when received. I.e., when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDR0) is read. The FEn bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSR0A.

This bit is reserved in Master SPI Mode (MSPIM).

Bit 3 – DOR0: Data OverRun
This bit is set if a Data OverRun condition is detected. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the Receive Shift Register, and a new start bit is detected. This bit is valid until the receive buffer (UDR0) is read. Always set this bit to zero when writing to UCSR0A.

This bit is reserved in Master SPI Mode (MSPIM).

Bit 2 – UPE0: USART Parity Error
This bit is set if the next character in the receive buffer had a Parity Error when received and the Parity Checking was enabled at that point (UPM01 = 1). This bit is valid until the receive buffer (UDR0) is read. Always set this bit to zero when writing to UCSR0A.

This bit is reserved in Master SPI Mode (MSPIM).

Bit 1 – U2X0: Double the USART Transmission Speed
This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

This bit is reserved in Master SPI Mode (MSPIM).

Bit 0 – MPCM0: Multi-processor Communication Mode
This bit enables the Multi-processor Communication mode. When the MPCMn bit is written to one, all the incoming frames received by the USART Receiver that do not contain address information will be ignored. The Transmitter is unaffected by the MPCM0 setting. Refer to Multi-Processor Communication Mode for details.

This bit is reserved in Master SPI Mode (MSPIM).

32. Electrical Characteristics

32.1. Absolute Maximum Ratings

Table 32-1. Absolute Maximum Ratings

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-0.5V to V_{CC}+0.5V
Voltage on $\overline{\text{RESET}}$ with respect to Ground	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0mA
DC Current V_{CC} and GND Pins	200.0mA

Note: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

32.2. Common DC Characteristics

Table 32-2. Common DC characteristics T_A = -40°C to 105°C, V_{CC} = 1.8V to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V_{IL}	Input Low Voltage, except XTAL1 and $\overline{\text{RESET}}$ pin	V_{CC} = 1.8V - 2.4V	-0.5		$0.2V_{CC}$(1)	V
		V_{CC} = 2.4V - 5.5V	-0.5		$0.3V_{CC}$(1)	
V_{IH}	Input High Voltage, except XTAL1 and $\overline{\text{RESET}}$ pins	V_{CC} = 1.8V - 2.4V	$0.7V_{CC}$(2)		V_{CC} + 0.5	V
		V_{CC} = 2.4V - 5.5V	$0.6V_{CC}$(2)		V_{CC} + 0.5	
V_{IL1}	Input Low Voltage, XTAL1 pin	V_{CC} = 1.8V - 5.5V	-0.5		$0.1V_{CC}$(1)	V
V_{IH1}	Input High Voltage, XTAL1 pin	V_{CC} = 1.8V - 2.4V	$0.8V_{CC}$(2)		V_{CC} + 0.5	V
		V_{CC} = 2.4V - 5.5V	$0.7V_{CC}$(2)		V_{CC} + 0.5	
V_{IL2}	Input Low Voltage, $\overline{\text{RESET}}$ pin	V_{CC} = 1.8V - 5.5V	-0.5		$0.1V_{CC}$(1)	V
V_{IH2}	Input High Voltage, $\overline{\text{RESET}}$ pin	V_{CC} = 1.8V - 5.5V	$0.9V_{CC}$(2)		V_{CC} + 0.5	V
V_{IL3}	Input Low Voltage, $\overline{\text{RESET}}$ pin as I/O	V_{CC} = 1.8V - 2.4V	-0.5		$0.2V_{CC}$(1)	V
		V_{CC} = 2.4V - 5.5V	-0.5		$0.3V_{CC}$(1)	

35. Register Summary

Offset	Name	Bit Pos.								
0x23	PINB	7:0	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
0x24	DDRB	7:0	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
0x25	PORTB	7:0	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
0x26	PINC	7:0		PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
0x27	DDRC	7:0		DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
0x28	PORTC	7:0		PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
0x29	PIND	7:0	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
0x2A	DDRD	7:0	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
0x2B	PORTD	7:0	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
0x2C ... 0x34	Reserved									
0x35	TIFR0	7:0						OCFB	OCFA	TOV
0x36	TIFR1	7:0			ICF			OCFB	OCFA	TOV
0x37	TIFR2	7:0						OCFB	OCFA	TOV
0x38 ... 0x3A	Reserved									
0x3B	PCIFR	7:0						PCIF2	PCIF1	PCIF0
0x3C	EIFR	7:0							INTF1	INTF0
0x3D	EIMSK	7:0							INT1	INT0
0x3E	GPIOR0	7:0	GPIOR0[7:0]							
0x3F	EECR	7:0			EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE
0x40	EEDR	7:0	EEDR[7:0]							
0x41	EEARL	7:0	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0
0x42	EEARH	7:0							EEAR9	EEAR8
0x43	GTCCR	7:0	TSM						PSRASY	PSRSYNC
0x44	TCCR0A	7:0	COM0A1	COM0A0	COM0B1	COM0B0			WGM01	WGM00
0x45	TCCR0B	7:0	FOC0A	FOC0B			WGM02	CS0[2:0]		
0x46	TCNT0	7:0	TCNT0[7:0]							
0x47	OCR0A	7:0	OCR0A[7:0]							
0x48	OCR0B	7:0	OCR0B[7:0]							
0x49	Reserved									
0x4A	GPIOR1	7:0	GPIOR1[7:0]							
0x4B	GPIOR2	7:0	GPIOR2[7:0]							
0x4C	SPCR0	7:0	SPIE0	SPE0	DORD0	MSTR0	CPOL0	CPHA0	SPR01	SPR00
0x4D	SPSR0	7:0	SPIF0	WCOL0						SPI2X0
0x4E	SPDR0	7:0	SPID[7:0]							
0x4F	Reserved									
0x50	ACSR	7:0	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0
0x51	DWDR	7:0	DWDR[7:0]							
0x52	Reserved									
0x53	SMCR	7:0					SM2	SM1	SM0	SE
0x54	MCUSR	7:0					WDRF	BORF	EXTRF	PORF

Offset	Name	Bit Pos.								
0x55	MCUCR	7:0		BODS	BODSE	PUD			IVSEL	IVCE
0x56	Reserved									
0x57	SPMCSR	7:0	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN
0x58 ... 0x5C	Reserved									
0x5D	SPL	7:0	(SP[7:0]) SPL							
0x5E	SPH	7:0						(SP[10:8]) SPH		
0x5F	SREG	7:0	I	T	H	S	V	N	Z	C
0x60	WDTCSR	7:0	WDIF	WDIE	WDP[3]	WDCE	WDE	WDP[2:0]		
0x61	CLKPR	7:0	CLKPCE				CLKPS3	CLKPS2	CLKPS1	CLKPS0
0x62 ... 0x63	Reserved									
0x64	PRR	7:0	PRTWI0	PRTIM2	PRTIM0		PRTIM1	PRSPI0	PRUSART0	PRADC
0x65	Reserved									
0x66	OSCCAL	7:0	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0
0x67	Reserved									
0x68	PCICR	7:0						PCIE2	PCIE1	PCIE0
0x69	EICRA	7:0					ISC11	ISC10	ISC01	ISC00
0x6A	Reserved									
0x6B	PCMSK0	7:0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0
0x6C	PCMSK1	7:0		PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8
0x6D	PCMSK2	7:0	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16
0x6E	TIMSK0	7:0						OCIEB	OCIEA	TOIE
0x6F	TIMSK1	7:0			ICIE			OCIEB	OCIEA	TOIE
0x70	TIMSK2	7:0						OCIEB	OCIEA	TOIE
0x71 ... 0x77	Reserved									
0x78	ADCL	7:0	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
0x79	ADCH	7:0							ADC9	ADC8
0x7A	ADCSRA	7:0	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
0x7B	ADCSRB	7:0		ACME				ADTS2	ADTS1	ADTS0
0x7C	ADMUX	7:0	REFS1	REFS0	ADLAR		MUX3	MUX2	MUX1	MUX0
0x7D	Reserved									
0x7E	DIDR0	7:0	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D
0x7F	DIDR1	7:0							AIN1D	AIN0D
0x80	TCCR1A	7:0	COM1	COM1	COM1	COM1			WGM11	WGM10
0x81	TCCR1B	7:0	ICNC1	ICES1		WGM13	WGM12	CS12	CS11	CS10
0x82	TCCR1C	7:0	FOC1A	FOC1B						
0x83	Reserved									
0x84	TCNT1L	7:0	TCNT1L[7:0]							
0x85	TCNT1H	7:0	TCNT1H[7:0]							
0x86	ICR1L	7:0	ICR1L[7:0]							
0x87	ICR1H	7:0	ICR1H[7:0]							
0x88	OCR1AL	7:0	OCR1AL[7:0]							

Offset	Name	Bit Pos.								
0x89	OCR1AH	7:0	OCR1AH[7:0]							
0x8A	OCR1BL	7:0	OCR1BL[7:0]							
0x8B	OCR1BH	7:0	OCR1BH[7:0]							
0x8C ... 0xAF	Reserved									
0xB0	TCCR2A	7:0	COM2A1	COM2A0	COM2B1	COM2B0			WGM21	WGM20
0xB1	TCCR2B	7:0	FOC2A	FOC2B			WGM22	CS2[2:0]		
0xB2	TCNT2	7:0	TCNT2[7:0]							
0xB3	OCR2A	7:0	OCR2A[7:0]							
0xB4	OCR2B	7:0	OCR2B[7:0]							
0xB5	Reserved									
0xB6	ASSR	7:0		EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB
0xB7	Reserved									
0xB8	TWBR	7:0	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0
0xB9	TWSR	7:0	TWS4	TWS3	TWS2	TWS1	TWS0		TWPS1	TWPS0
0xBA	TWAR	7:0	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
0xBB	TWDR	7:0	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0
0xBC	TWCR	7:0	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN		TWIE
0xBD	TWAMR	7:0	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	
0xBE ... 0xBF	Reserved									
0xC0	UCSR0A	7:0	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
0xC1	UCSR0B	7:0	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
0xC2	UCSR0C	7:0	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 / UDORD0	UCSZ00 / UCPHA0	UCPOL0
0xC3	Reserved									
0xC4	UBRR0L	7:0	UBRR0[7:0]							
0xC5	UBRR0H	7:0					UBRR0[3:0]			
0xC6	UDR0	7:0	TXB / RXB[7:0]							

35.1. Note

1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the Status Flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such Status Flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega328/P is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For

12 Hoja de datos TM1637

TITAN MICRO™ ELECTRONICS **LED Drive Control Special Circuit** **TM1637**

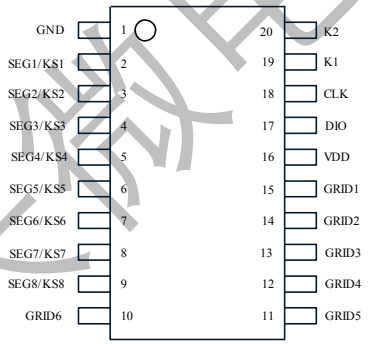
Features description

TM1637 is a kind of LED (light-emitting diode display) drive control special circuit with keyboard scan interface and it's internally integrated with MCU digital interface, data latch, LED high pressure drive and keyboard scan. This product is in DIP20/SOP20 package type with excellent performance and high quality, which is mainly applicable to the display drive of induction cooker, micro-wave oven and small household electrical appliance.

Function features

- Applied power CMOS technique
- The display mode (8 segments*6 bit) supports output by common anode LED.
- Keyboard scan (8×2bit), with enhanced identification circuit with anti-interference keys
- Luminance adjustment circuit (adjustable 8 duty ratio)
- Two-wire serial interface (CLK, DIO)
- Oscillating type: Built-in RC oscillator
- Built-in power-on reset circuit
- Built-in automatic blanking circuit
- Package type: DIP20/SOP20

Pin information



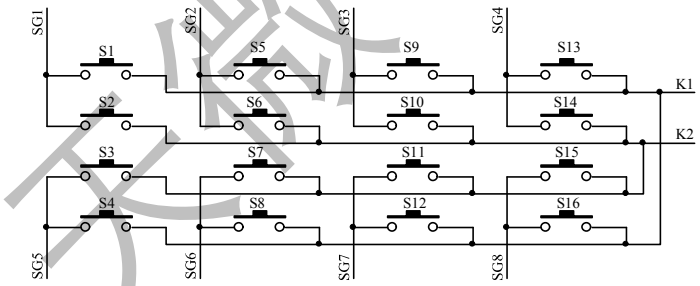
Pin functions

Symbols	Pin name	Pin No.	Description
DIO	Data input/output	17	It is used for serial data input and output. The input data has a low level fluctuation while high level transfer at SCLK. Once one bit is transferred, one ACK is generated at failing edge of the 8th clock inside the chip.
CLK	Clock input	18	It is used for data input and output at rising edge.
K1~K2	Data input by keyboard scan	19-20	Inputting the pin data here and it will be latched when the display cycle is over.
SG1~SG8	Output (segment)	2-9	Segment Output (also keyboard scan) and N-channel open drain output
GRID6~GRID1	Output (bit)	10-15	Bit output and P-channel open drain output
VDD	Logic Supply	16	Anode power connection
GND	logic ground	1	Grounding connection

Electrostatic discharge led by much static at dry weather or environment could damage the integrated circuit. TITAN MICRO ELECTRONICS suggests you to take every measure to protect integrated circuit. ESD damage or decreased performance by inappropriate operation or welding could lead to chip failure.

Read the key scan data

Key scan matrix of 8×2bit is shown as the following:



When a key is pressed, the key scan data is as follows: (Where low level is forward and high level is backward, 1110_1111 stands for 0xF7) .

	SG1	SG2	SG3	SG4	SG5	SG6	SG7	SG8
K1	1110_1111	0110_1111	1010_1111	0010_1111	1100_1111	0100_1111	1000_1111	0000_1111
K2	1111_0111	0111_0111	1011_0111	0011_0111	1101_0111	0101_0111	1001_0111	0001_0111

Note: Where there is no key pressed down, the key read data should be 1111_1111 with forward low level and backward high level. Since strong interference exists in the use of kitchen appliances, such as induction cooker, negative edge trigger mode was applied in TM1637 to avoid mistake trigger, which is also the called “key jumping”. TM 1637 doesn’t support combined key pressing.

Display register address

Stored data in the register is transferred to the TM1637 from outside elements by serial interface, with 6 bytes units of address from C0H to C5H in correspondence with the LED lights connected with SEG pin and GRID pin on the chip.LED data is displayed from low level to high level in respect of display address, and should be operated from low level to high level in respect of data bytes.

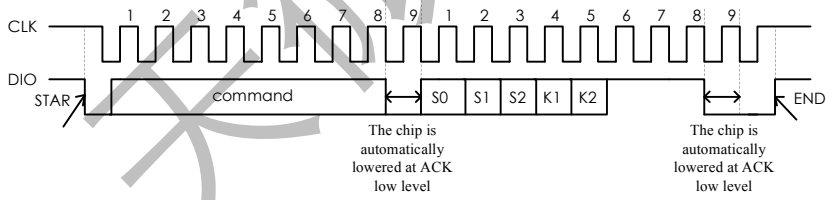
SEG1	SEG2	SEG3	SEG4	SEG5	SEG6	SEG7	SEG8	
xxHL (low four bits)				xxHU(high four bits)				
B0	B1	B2	B3	B4	B5	B6	B7	
C0HL				C0HU				**GRID1**
C1HL				C1HU				**GRID2**
C2HL				C2HU				**GRID3**
C3HL				C3HU				**GRID4**
C4HL				C4HU				**GRID5**
C5HL				C5HU				**GRID6**

Interface interpretation

Microprocessor data realize the communication with TM1637 by means of two–wire bus interface (Note: The communication method is not equal to 12C bus protocol totally because there is no slave address). When data is input, DIO signal should not change for high level CLK and DIO signal should change for low level CLK signal. When CLK is a high level and DIO changes from high to low level, data input starts. When CLK is a high level and DIO changes from low level to high level, data input ends.

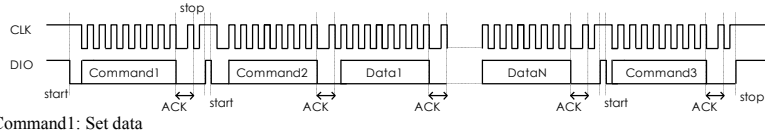
TM1637 data transfer carries with answering signal ACK. For a right data transfer, an answering signal ACK is generated inside the chip to lower the DIO pin at the failing edge of the 8th clock. DIO interface wire is released at the end of the 9th clock.

1、Command data transfer is as follows (Reading Key Data Timing)



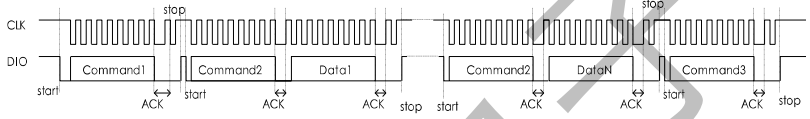
Command: command to read the keys; Key information coding consists of S0, S1, S2, K1 and K2. SGn coding consists of S0, S1, and S2. K1 and K2 are coding for K1 key and K2 key. The key should be read from low level to high level and the clock frequency should be less than 250K.

2、Write SRAM data in address auto increment 1 mode.



Command1: Set data
Command2: Set address
Data1~N: Transfer display data
Command3: Control display

3、Write SRAM data in a fixed address mode



Command1: Set data
Command2: Set data
Data1~N: Transfer display data
Command3: Control display

Data command

Command is used to set the display mode and the LED driver status.

The first byte input from DIO at CLK failing edge acts as a command. The highest B7 and B6 bytes after decoding are used to distinguish different commands.

B7	B6	Command
0	1	Data command setting
1	0	Display and control command setting
1	1	Address command setting

When STOP command is sent during command or data transfer, serial communication is initialized and command or data transferring becomes invalid (Command or data transferred before remain effective.).

1、Data command setting

This command is to set data write and data read. 01 and 11 are not permitted to set for B1 and B0 bits.

MSB LSB

B7	B6	B5	B4	B3	B2	B1	B0	Function	Description
0	1	Zero should be inserted for irrelevant items.				0	0	Data write and read mode setting	Write data to display register
0	1					1	0		Read key scan data
0	1				0			Address adding mode setting	Automatic address adding
0	1				1				Fix address
0	1			0				Test mode setting (for internal)	Normal mode
0	1			1					Test mode

2、Address command setting

MSB LSB

B7	B6	B5	B4	B3	B2	B1	B0	Display address
1	1	Zero should be inserted for irrelevant items.		0	0	0	0	C0H
1	1			0	0	0	1	C1H
1	1			0	0	1	0	C2H
1	1			0	0	1	1	C3H
1	1			0	1	0	0	C4H
1	1			0	1	0	1	C5H

The command is used to set the display register address. If the address is set as C6H or a higher one, the data will be ignored until effective address is set. Once electrified, the default address is C0H.

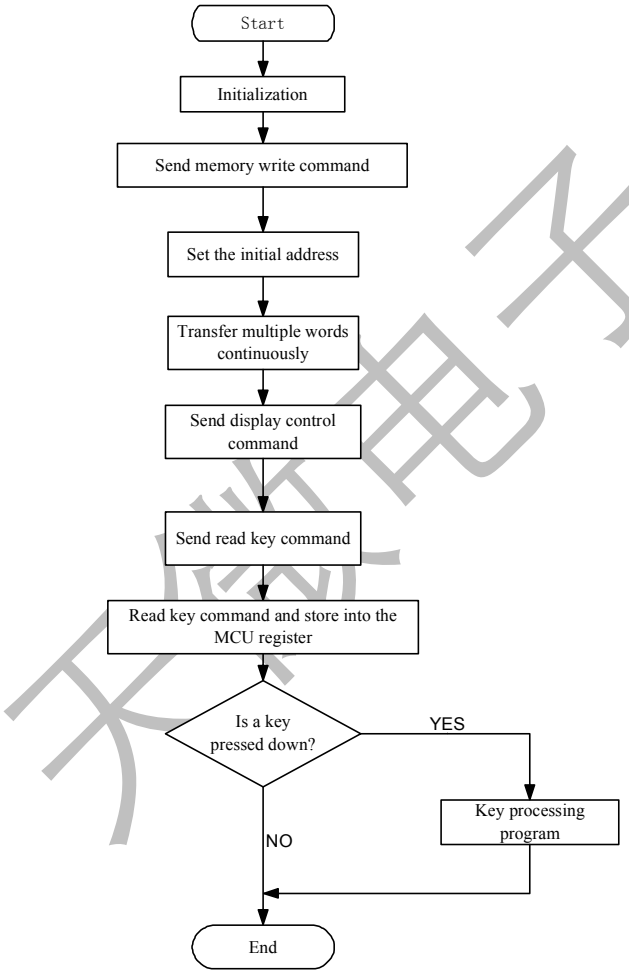
3、Display control

MSB LSB

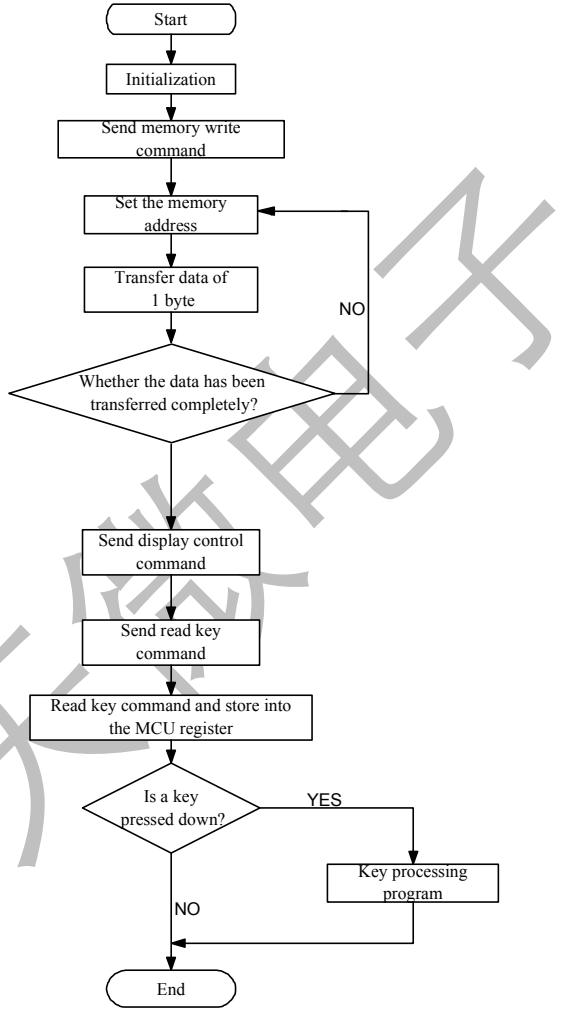
B7	B6	B5	B4	B3	B2	B1	B0	Function	Description
1	0	Zero should be inserted for irrelevant items.			0	0	0	Setting of extinction number	1/16 Pulse width is set as 1/16.
1	0				0	0	1		2/16 Pulse width is set as 2/16
1	0				0	1	0		4/16 Pulse width is set as 4/16
1	0				0	1	1		10/16 Pulse width is set as 10/16
1	0				1	0	0		11/16 Pulse width is set as 11/16
1	0				1	0	1		12/16 Pulse width is set as 12/16
1	0				1	1	0		13/16 Pulse width is set as 13/16
1	0				1	1	1		14/16 Pulse width is set as 14/16
1	0			0				Display switch setting	Display OFF
1	0			1					Display ON

Program flow chart

1、Flow chart of program using address auto increment 1 mode

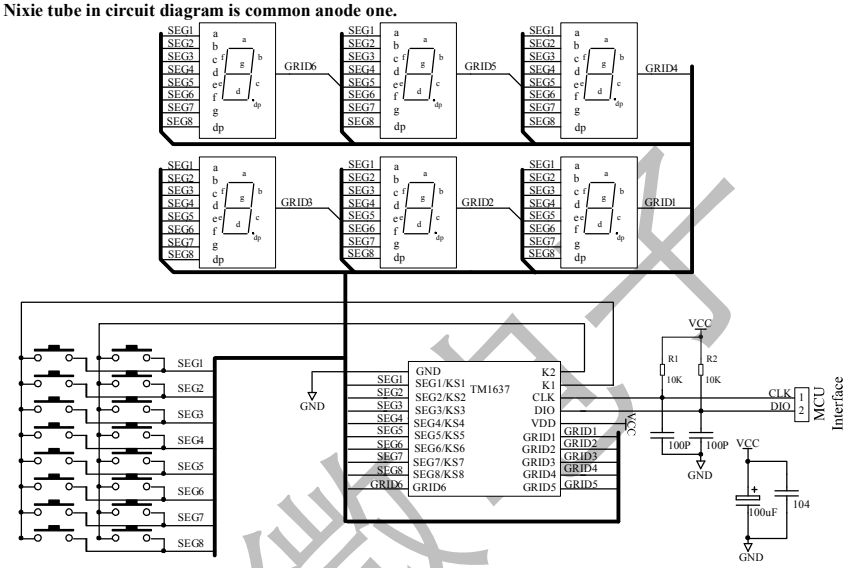


2、Flow chart of program design using fixed address



Hardware connection drawing

Nixie tube in circuit diagram is common anode one.



Note: 1. filtering capacitor between VDD and GND should be arranged on PCB plate as close to TM1637 chip as possible to strengthen filtering effect.

2. 100pF capacitor connected to the DIO, CLK communication port pull-up and pull-down can reduce interference to radio communications port.

3. Since blue-ray nixie tube break over step-down voltage is 3V, TM1637 power supply should be 5V.

Electrical parameter

1、 Limit parameter (Ta = 25℃, Vss = 0 V)

Parameters	Symbol	Range	Unit
Logic power supply voltage	VDD	-0.5 ～+7.0	V
Logic input voltage	VI1	-0.5 ～ VDD + 0.5	V
LED and SEG drive sink current	IO1	50	mA
LED and GRID drive source current	IO2	200	mA
Power loss	PD	400	mW
Work temperature	Topt	-40 ～ +85	℃
Storage temperature	Tstg	-65 ～+150	℃

2、 Normal working range (Ta = -40～+85℃, Vss = 0 V)

Parameters	Symbol	Minimum	Typical	Maximum	Unit	Test condition
Logic power supply voltage	VDD		5		V	-
High-level input voltage	VIH	0.7 VDD	-	VDD	V	-
Low-level input voltage	VIL	0	-	0.3 VDD	V	-

3、Electrical character (Ta = -40 - +85℃, VDD = 4.5 - 5.5 V, Vss = 0 V)

Parameters	Symbol	Minimum	Typical	Maximum	Unit	Test condition
GRID drive source current	Ioh1	80	120	180	mA	GRID1~GRID6, Vo = vdd-2V
	Ioh2	80	140	200	mA	GRID1~GRID6, Vo = vdd-3V
SEG drive sink current	IOL1	20	30	50	mA	SEG1~SEG8 Vo=0.3V
DOUT pin output low current	Idout	4	-	-	mA	Vo = 0.4V, dout
High-level output current tolerance	Itolsg	-	-	5	%	Vo = VDD – 3V, GRID1～GRID6
Output pull down resistor	RL		10		KΩ	K1~K2

Input current	II	-	-	±1	μA	VI = VDD / VSS
High-level input voltage	VIH	0.7 VDD	-		V	CLK，DIN
Low-level input voltage	VIL	-	-	0.3 VDD	V	CLK，DIN
Lagging voltage	VH	-	0.35	-	V	CLK，DIN
dynamic current loss	IDDdyn	-	-	5	mA	Non-loaded, display OFF

4、Switching character (Ta = -40 - +85℃, VDD = 4.5 - 5.5 V, Vss = 0 V)

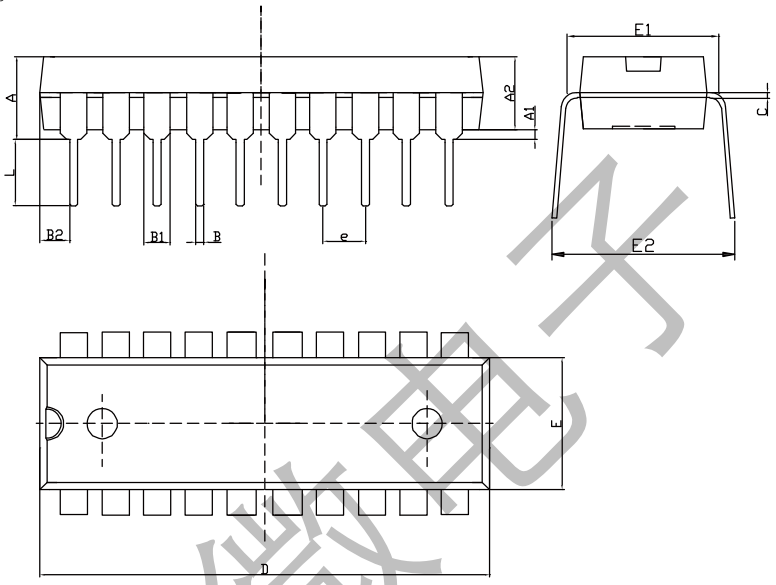
Parameters	Symbol	Minimum	Typical	Maximum	Unit	Test condition	
oscillation frequency	fosc	-	450	-	KHz		
Transmission delay time	tPLZ	-	-	300	ns	CLK → DIO	
	tPZL	-	-	100	ns	CL = 15pF, RL = 10K Ω	
Rise time	TTZH 1	-	-	2	μs	CL = 300p F	GRID1～GRID6
	TTZH 2	-	-	0.5	μs		SEG1～SEG8
Fall time	TTHZ	-	-	120	μs	CL = 300pF，Segn，Gridn	
Maximum clock frequency	Fmax	-	-	500	KHz	占空比50% 50% duty ratio	
Input capacitance	CI	-	-	15	pF	-	

5、Timing character (Ta = -40 - +85℃, VDD = 4.5 - 5.5 V, Vss = 0 V)

Parameters	Symbol	Minimum	Typical	Maximum	Unit	Test condition
Clock pulse width	PWCLK	400	-	-	ns	-
Data setup time	tSETUP	100	-	-	ns	-
Data hold time	tHOLD	100	-	-	ns	-
Waiting time	tWAIT	1	-	-	μs	CLK↑→CLK↓

IC Package diagram

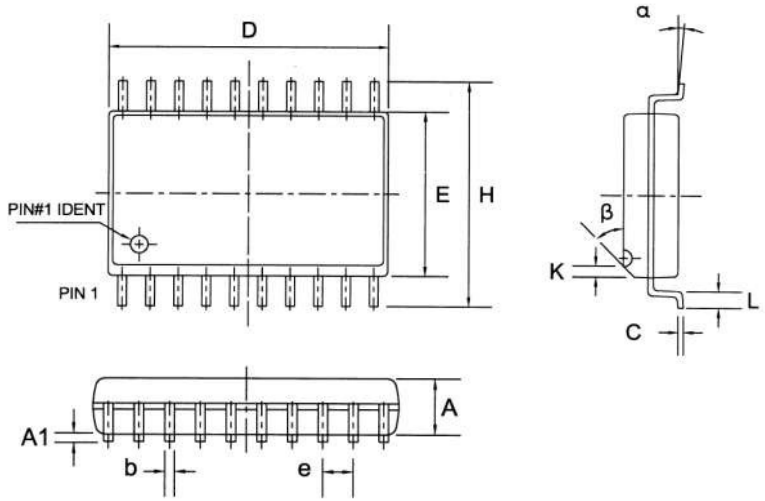
DIP20



Symbol	Unit: mm		
	Minimum	Typical value	Maximum value
A	3.71	4.00	4.31
A1	0.50	0.60	0.80
A2	3.20	3.40	3.60
B	0.33	0.45	0.53
B1		1.525(TYP)	
C	0.20	0.28	0.36
D	25.70	26.00	26.54
E	6.20	6.40	6.75
E1	7.32	7.78	8.25
e		2.54(TYP)	
L	3.00	3.30	3.60
E2	8.20	8.70	9.10
B2	0.87	1.02	1.17

TITAN MICRO ELECTRONICS™ **LED Drive Control Special Circuit** **TM1637**

SOP20



Symbol	Dimensions In Millmeters			Dimensions In Inches		
	Min	Nom	Max	Min	Nom	Max
A	2.15	2.35	2.55	0.085	0.093	0.100
A1	0.05	0.15	0.25	0.002	0.006	0.010
b	—	0.40	—	—	0.016	—
C	—	0.25	—	—	0.010	—
D	12.40	12.70	13.00	0.488	0.500	0.512
E	7.40	7.65	7.90	0.291	0.301	0.311
e	—	1.27	—	—	0.050	—
H	10.15	10.45	10.75	0.400	0.411	0.423
K	—	0.50	—	—	0.020	—
L	0.60	0.80	1.00	0.024	0.031	0.039
α	0°	—	8°	0°	—	8°
β	—	45°	—	—	45°	—

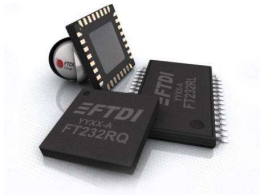
All specs and applications shown above subject to change without prior notice.

13 Datos relevantes FT232R

FTDI Chip **FT232R USB UART IC** Datasheet
Version 2.13
Document No.: FT_000053 Clearance No.: FTDI# 38

Future Technology Devices International Ltd.

FT232R USB UART IC



The FT232R is a USB to serial UART interface with the following advanced features:

- Single chip USB to asynchronous serial data transfer interface.
- Entire USB protocol handled on the chip. No USB specific firmware programming required.
- Fully integrated 1024 bit EEPROM storing device descriptors and CBUS I/O configuration.
- Fully integrated USB termination resistors.
- Fully integrated clock generation with no external crystal required plus optional clock output selection enabling a glue-less interface to external MCU or FPGA.
- Data transfer rates from 300 baud to 3 Mbaud (RS422, RS485, RS232) at TTL levels.
- 128 byte receive buffer and 256 byte transmit buffer utilising buffer smoothing technology to allow for high data throughput.
- FTDI's royalty-free Virtual Com Port (VCP) and Direct (D2XX) drivers eliminate the requirement for USB driver development in most cases.
- Unique USB FTDIChip-ID™ feature.
- Configurable CBUS I/O pins.
- Transmit and receive LED drive signals.
- UART interface support for 7 or 8 data bits, 1 or 2 stop bits and odd / even / mark / space / no parity
- FIFO receives and transmits buffers for high data throughput.
- Synchronous and asynchronous bit bang interface options with RD# and WR# strobes.
- Device supplied pre-programmed with unique USB serial number.
- Supports bus powered, self-powered and high-power bus powered USB configurations.
- Integrated +3.3V level converter for USB I/O.
- Integrated level converter on UART and CBUS for interfacing to between +1.8V and +5V logic.
- True 5V/3.3V/2.8V/1.8V CMOS drive output and TTL input.
- Configurable I/O pin output drive strength.
- Integrated power-on-reset circuit.
- Fully integrated AVCC supply filtering - no external filtering required.
- UART signal inversion option.
- +3.3V (using external oscillator) to +5.25V (internal oscillator) Single Supply Operation.
- Low operating and USB suspend current.
- Low USB bandwidth consumption.
- UHCI/OHCI/EHCI host controller compatible.
- USB 2.0 Full Speed compatible.
- -40°C to 85°C extended operating temperature range.
- Available in compact Pb-free 28 Pin SSOP and QFN-32 packages (both RoHS compliant).

1 Typical Applications

- USB to RS232/RS422/RS485 Converters
- Upgrading Legacy Peripherals to USB
- Cellular and Cordless Phone USB data transfer cables and interfaces
- Interfacing MCU/PLD/FPGA based designs to USB
- USB Audio and Low Bandwidth Video data transfer
- PDA to USB data transfer
- USB Smart Card Readers
- USB Instrumentation
- USB Industrial Control
- USB MP3 Player Interface
- USB FLASH Card Reader and Writers
- Set Top Box PC - USB interface
- USB Digital Camera Interface
- USB Hardware Modems
- USB Wireless Modems
- USB Bar Code Readers
- USB Software and Hardware Encryption Dongles

1.1 Driver Support

Royalty free VIRTUAL COM PORT (VCP) DRIVERS for...

- Windows 10 32,64-bit
- Windows 8/8.1 32,64-bit
- Windows 7 32,64-bit
- Windows Vista and Vista 64-bit
- Windows XP and XP 64-bit
- Windows 98, 98SE, ME, 2000, Server 2003, XP, Server 2008 and server 2012 R2
- Windows XP Embedded
- Windows CE 4.2, 5.0 and 6.0
- Mac OS 8/9, OS-X
- Linux 2.4 and greater

Royalty free D2XX *Direct* Drivers (USB Drivers + DLL S/W Interface)

- Windows 10 32,64-bit
- Windows 8/8.1 32,64-bit
- Windows 7 32,64-bit
- Windows Vista and Vista 64-bit
- Windows XP and XP 64-bit
- Windows 98, 98SE, ME, 2000, Server 2003, XP, Server 2008 and server 2012 R2
- Windows XP Embedded
- Windows CE 4.2, 5.0 and 6.0
- Linux 2.4 and greater
- Android(J2xx)

The drivers listed above are all available to download for free from FTDI website (www.ftdichip.com). Various 3rd party drivers are also available for other operating systems - see FTDI website (www.ftdichip.com) for details.

For driver installation, please refer to http://www.ftdichip.com/Documents/InstallGuides.htm

3 Device Pin Out and Signal Description

3.1 28-LD SSOP Package

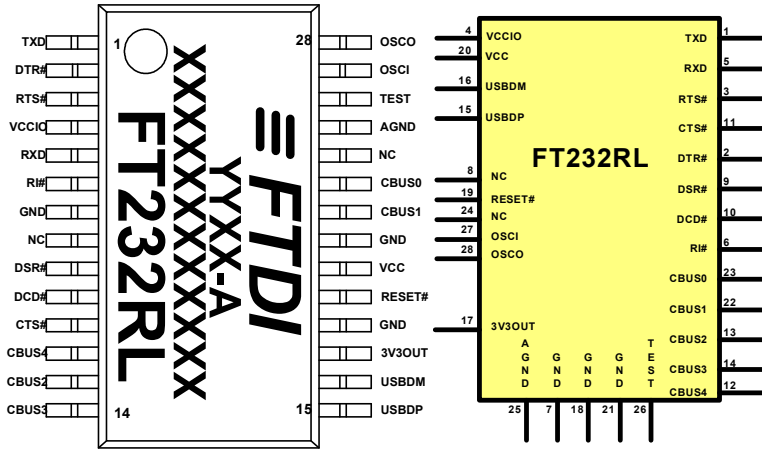


Figure 3.1 SSOP Package Pin Out and Schematic Symbol

3.2 SSOP Package Pin Out Description

Note: The convention used throughout this document for active low signals is the signal name followed by#

Pin No.	Name	Type	Description
15	USBDP	I/O	USB Data Signal Plus, incorporating internal series resistor and 1.5kΩ pull up resistor to 3.3V.
16	USBDM	I/O	USB Data Signal Minus, incorporating internal series resistor.

Table 3.1 USB Interface Group

Pin No.	Name	Type	Description
4	VCCIO	PWR	+1.8V to +5.25V supply to the UART Interface and CBUS group pins (1...3, 5, 6, 9...14, 22, 23). In USB bus powered designs connect this pin to 3V3OUT pin to drive out at +3.3V levels, or connect to VCC to drive out at 5V CMOS level. This pin can also be supplied with an external +1.8V to +2.8V supply in order to drive outputs at lower levels. It should be noted that in this case this supply should originate from the same source as the supply to VCC. This means that in bus powered designs a regulator which is supplied by the +5V on the USB bus should

5.2 DC Characteristics

DC Characteristics (Ambient Temperature = -40°C to +85°C)

Parameter	Description	Minimum	Typical	Maximum	Units	Conditions
VCC1	VCC Operating Supply Voltage	4.0	---	5.25	V	Using Internal Oscillator
VCC1	VCC Operating Supply Voltage	3.3	---	5.25	V	Using External Crystal
VCC2	VCCIO Operating Supply Voltage	1.8	---	5.25	V	
Icc1	Operating Supply Current	---	15	---	mA	Normal Operation
Icc2	Operating Supply Current	50	70	100	µA	USB Suspend
3V3	3.3v regulator output	3.0	3.3	3.6	V	

Table 5.2 Operating Voltage and Current

Parameter	Description	Minimum	Typical	Maximum	Units	Conditions
Voh	Output Voltage High	3.2	4.1	4.9	V	I source = 2mA
Vol	Output Voltage Low	0.3	0.4	0.6	V	I sink = 2mA
Vin	Input Switching Threshold	1.0	1.2	1.5	V	**
VHys	Input Switching Hysteresis	20	25	30	mV	**

Table 5.3 UART and CBUS I/O Pin Characteristics (VCCIO = +5.0V, Standard Drive Level)

Parameter	Description	Minimum	Typical	Maximum	Units	Conditions
Voh	Output Voltage High	2.2	2.7	3.2	V	I source = 1mA
Vol	Output Voltage Low	0.3	0.4	0.5	V	I sink = 2mA
Vin	Input Switching Threshold	1.0	1.2	1.5	V	**
VHys	Input Switching Hysteresis	20	25	30	mV	**

Table 5.4 UART and CBUS I/O Pin Characteristics (VCCIO = +3.3V, Standard Drive Level)

14 Hoja de datos LM393

Order this document by LM393/D

MOTOROLA

LM393, LM393A, LM293, LM2903, LM2903V

Low Offset Voltage Dual Comparators

SINGLE SUPPLY, LOW POWER DUAL COMPARATORS

SEMICONDUCTOR TECHNICAL DATA

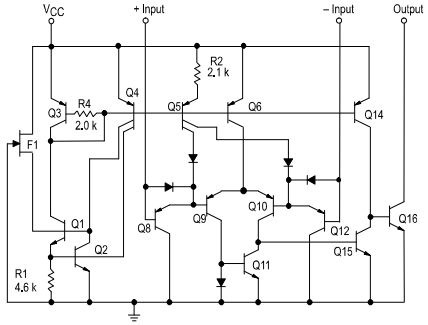
The LM393 series are dual independent precision voltage comparators capable of single or split supply operation. These devices are designed to permit a common mode range–to–ground level with single supply operation. Input offset voltage specifications as low as 2.0 mV make this device an excellent selection for many applications in consumer automotive, and industrial electronics.

- Wide Single–Supply Range: 2.0 Vdc to 36 Vdc
- Split–Supply Range: ±1.0 Vdc to ±18 Vdc
- Very Low Current Drain Independent of Supply Voltage: 0.4 mA
- Low Input Bias Current: 25 nA
- Low Input Offset Current: 5.0 nA
- Low Input Offset Voltage: 2.0 mV (max) LM393A
  5.0 mV (max) LM293/393
- Input Common Mode Range to Ground Level
- Differential Input Voltage Range Equal to Power Supply Voltage
- Output Voltage Compatible with DTL, ECL, TTL, MOS, and CMOS Logic Levels
- ESD Clamps on the Inputs Increase the Ruggedness of the Device without Affecting Performance

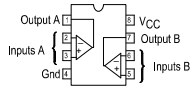
N SUFFIX
PLASTIC PACKAGE
CASE 626

D SUFFIX
PLASTIC PACKAGE
CASE 751
(SO–8)

Representative Schematic Diagram
(Diagram shown is for 1 comparator)



PIN CONNECTIONS



(Top View)

ORDERING INFORMATION

Device	Operating Temperature Range	Package
LM293D	T_A = –25° to +85°C	SO–8
LM393D	T_A = 0° to +70°C	SO–8
LM393AN,N		Plastic DIP
LM2903D	T_A = –40° to +105°C	SO–8
LM2903N		Plastic DIP
LM2903VD	T_A = –40° to +105°C	SO–8
LM2903VN		Plastic DIP

 Rev 1

LM393, LM393A, LM293, LM2903, LM2903V

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Power Supply Voltage	V_{CC}	+36 or ±18	Vdc
Input Differential Voltage Range	V_{IDR}	36	Vdc
Input Common Mode Voltage Range	V_{ICR}	–0.3 to +36	Vdc
Output Short Circuit–to–Ground Output Sink Current (Note 1)	I_{SC} I_{Sink}	Continuous 20	mA
Power Dissipation @ T_A = 25°C Derate above 25°C	P_D $1/R_{\theta JA}$	570 5.7	mW mW/°C
Operating Ambient Temperature Range LM293 LM393, 393A LM2903 LM2903V	T_A	 –25 to +85 0 to +70 –40 to +105 –40 to +125	°C
Maximum Operating Junction Temperature LM393, 393A, 2903, LM2903V LM293	$T_{J(max)}$	 125 150	°C
Storage Temperature Range	T_{stg}	–65 to +150	°C

ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 Vdc, $T_{low} \le T_A \le T_{high}$,* unless otherwise noted.)

Characteristic	Symbol	LM393A Min	LM393A Typ	LM393A Max	Unit
Input Offset Voltage (Note 2) T_A = 25°C $T_{low} \le T_A \le T_{high}$	V_{IO}	 – –	 ±1.0 –	 ±2.0 4.0	mV
Input Offset Current T_A = 25°C $T_{low} \le T_A \le T_{high}$	I_{IO}	 – –	 ±50 –	 ±50 ±150	nA
Input Bias Current (Note 3) T_A = 25°C $T_{low} \le T_A \le T_{high}$	I_{IB}	 – –	 25 –	 250 400	nA
Input Common Mode Voltage Range (Note 4) T_A = 25°C $T_{low} \le T_A \le T_{high}$	V_{ICR}	 0 0	 – –	 V_{CC} –1.5 V_{CC} –2.0	V
Voltage Gain $R_L \ge 15$ kΩ, V_{CC} = 15 Vdc, T_A = 25°C	A_{VOL}	50	200	–	V/mV
Large Signal Response Time V_{in} = TTL Logic Swing, V_{ref} = 1.4 Vdc V_{RL} = 5.0 Vdc, R_L = 5.1 kΩ, T_A = 25°C	–	–	300	–	ns
Response Time (Note 5) V_{RL} = 5.0 Vdc, R_L = 5.1 kΩ, T_A = 25°C	t_{TLH}	–	1.3	–	µs
Input Differential Voltage (Note 6) All $V_{in} \ge$ Gnd or V– Supply (if used)	V_{ID}	–	–	V_{CC}	V
Output Sink Current $V_{in} \ge$ 1.0 Vdc, V_{in+} = 0 Vdc, $V_O \le$ 1.5 Vdc, T_A = 25°C	I_{Sink}	6.0	16	–	mA
Output Saturation Voltage $V_{in} \ge$ 1.0 Vdc, V_{in+} = 0 Vdc, $I_{Sink} \le$ 4.0 mA, T_A = 25°C $T_{low} \le T_A \le T_{high}$	V_{OL}	 – –	 150 –	 400 700	mV

*T_{low} = 0°C, T_{high} = +70°C for LM393/393A

NOTES: 1. The maximum output current may be as high as 20 mA, independent of the magnitude of V_{CC}, output short circuits to V_{CC} can cause excessive heating and eventual destruction.
2. At output switch point, $V_O \approx$ 1.4 Vdc, R_S = 0 Ω with V_{CC} from 5.0 Vdc to 30 Vdc, and over the full input common mode range (0 V to V_{CC} = –1.5 V).
3. Due to the PNP transistor inputs, bias current will flow out of the inputs. This current is essentially constant, independent of the output state, there fore, no loading changes will exist on the input lines.
4. Input common mode of either input should not be permitted to go more than 0.3 V negative of ground or minus supply. The upper limit of common mode range is V_{CC} –1.5 V.
5. Response time is specified with a 100 mV step and 5.0 mV of overdrive. With larger magnitudes of overdrive faster response times are obtainable.
6. The comparator will exhibit proper output state if one of the inputs becomes greater than V_{CC}, the other input must remain within the common mode range. The low input state must not be less than –0.3 V of ground or minus supply.

LM393, LM393A, LM293, LM2903, LM2903V

ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 Vdc, $T_{low} \le T_A \le T_{high}$,* unless otherwise noted.)

Characteristic	Symbol	LM393A Min	LM393A Typ	LM393A Max	Unit
Output Leakage Current V_{in-} = 0 V, $V_{in+} \ge$ 1.0 Vdc, V_O = 5.0 Vdc, T_A= 25°C V_{in-} = 0 V, $V_{in+} \ge$ 1.0 Vdc, V_O = 30 Vdc, $T_{low} \le T_A \le T_{high}$	I_{OL}	 – –	 0.1 –	 – 1.0	µA
Supply Current R_L = ∞ Both Comparators, T_A = 25°C R_L = ∞ Both Comparators, V_{CC} = 30 V	I_{CC}	 – –	 0.4 1.0	 1.0 2.5	mA

ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 Vdc, $T_{low} \le T_A \le T_{high}$, unless otherwise noted.)

Characteristic	Symbol	LM392, LM393 Min	LM392, LM393 Typ	LM392, LM393 Max	LM2903, LM2903V Min	LM2903, LM2903V Typ	LM2903, LM2903V Max	Unit
Input Offset Voltage (Note 2) T_A = 25°C $T_{low} \le T_A \le T_{high}$	V_{IO}	 – –	 ±1.0 –	 ±5.0 9.0	 – –	 ±2.0 9.0	 ±7.0 15	mV
Input Offset Current T_A = 25°C $T_{low} \le T_A \le T_{high}$	I_{IO}	 – –	 ±5.0 –	 ±50 ±150	 – –	 ±5.0 ±50	 ±50 ±200	nA
Input Bias Current (Note 3) T_A = 25°C $T_{low} \le T_A \le T_{high}$	I_{IB}	 – –	 25 –	 250 400	 – –	 25 200	 250 500	nA
Input Common Mode Voltage Range (Note 3) T_A = 25°C $T_{low} \le T_A \le T_{high}$	V_{ICR}	 0 0	 – –	 V_{CC} –1.5 V_{CC} –2.0	 0 0	 – –	 V_{CC} –1.5 V_{CC} –2.0	V
Voltage Gain $R_L \ge$ 15 kΩ, V_{CC} = 15 Vdc, T_A = 25°C	A_{VOL}	50	200	–	25	200	–	V/mV
Large Signal Response Time V_{in} = TTL Logic Swing, V_{ref} = 1.4 Vdc V_{RL} = 5.0 Vdc, R_L = 5.1 kΩ, T_A = 25°C	–	–	300	–	–	300	–	ns
Response Time (Note 5) V_{RL} = 5.0 Vdc, R_L = 5.1 kΩ, T_A = 25°C	t_{TLH}	–	1.3	–	–	1.5	–	µs
Input Differential Voltage (Note 6) All $V_{in} \ge$ Gnd or V– Supply (if used)	V_{ID}	–	–	V_{CC}	–	–	V_{CC}	V
Output Sink Current $V_{in} \ge$ 1.0 Vdc, V_{in+} = 0 Vdc, $V_O \le$ 1.5 Vdc T_A = 25°C	I_{Sink}	6.0	16	–	6.0	16	–	mA
Output Saturation Voltage $V_{in} \ge$ 1.0 Vdc, V_{in+} = 0, $I_{Sink} \le$ 4.0 mA, T_A = 25°C $T_{low} \le T_A \le T_{high}$	V_{OL}	 – –	 150 –	 400 700	 – –	 – 200	 400 700	mV
Output Leakage Current V_{in-} = 0 V, $V_{in+} \ge$ 1.0 Vdc, V_O = 5.0 Vdc, T_A = 25°C V_{in-} = 0 V, $V_{in+} \ge$ 1.0 Vdc, V_O = 30 Vdc, $T_{low} \le T_A \le T_{high}$	I_{OL}	 – –	 0.1 –	 – 1000	 – –	 0.1 –	 – 1000	nA
Supply Current R_L = ∞ Both Comparators, T_A = 25°C R_L = ∞ Both Comparators, V_{CC} = 30 V	I_{CC}	 – –	 0.4 –	 1.0 2.5	 – –	 0.4 –	 1.0 2.5	mA

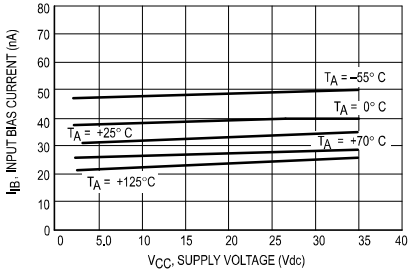
*T_{low} = 0°C, T_{high} = +70°C for LM393/393A
LM293 T_{low} = –25°C, T_{high} = +85°C
LM2903 T_{low} = –40°C, T_{high} = +105°C
LM2903V T_{low} = –40°C, T_{high} = +125°C

NOTES: 2. At output switch point, $V_O \approx$ 1.4 Vdc, R_S = 0 Ω with V_{CC} from 5.0 Vdc to 30 Vdc, and over the full input common mode range (0 V to V_{CC} = –1.5 V).
3. Due to the PNP transistor inputs, bias current will flow out of the inputs. This current is essentially constant, independent of the output state, there fore, no loading changes will exist on the input lines.
5. Response time is specified with a 100 mV step and 5.0 mV of overdrive. With larger magnitudes of overdrive faster response times are obtainable.
6. The comparator will exhibit proper output state if one of the inputs becomes greater than V_{CC}, the other input must remain within the common mode range. The low input state must not be less than –0.3 V of ground or minus supply.

LM393, LM393A, LM293, LM2903, LM2903V

LM293/393,A

Figure 1. Input Bias Current versus Power Supply Voltage



LM2903

Figure 2. Input Bias Current versus Power Supply Voltage

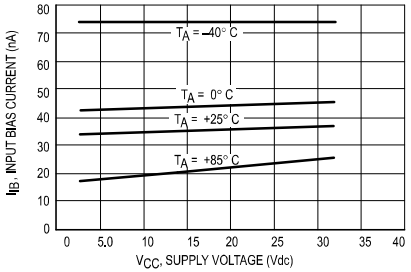


Figure 3. Output Saturation Voltage versus Output Sink Current

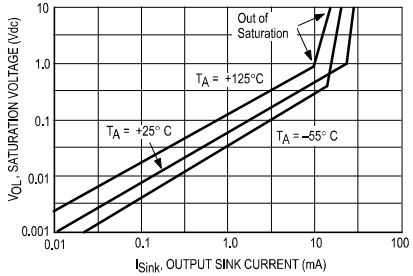


Figure 4. Output Saturation Voltage versus Output Sink Current

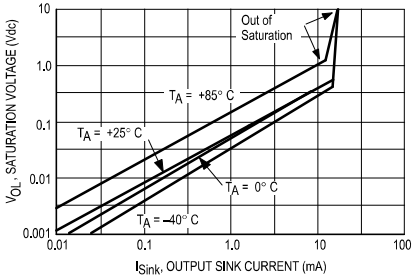


Figure 5. Power Supply Current versus Power Supply Voltage

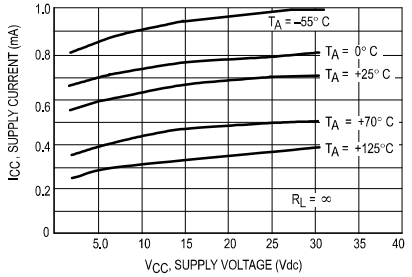
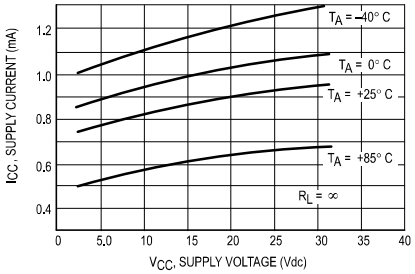


Figure 6. Power Supply Current versus Power Supply Voltage



LM393, LM393A, LM293, LM2903, LM2903V

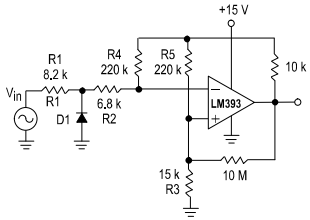
APPLICATIONS INFORMATION

These dual comparators feature high gain, wide bandwidth characteristics. This gives the device oscillation tendencies if the outputs are capacitively coupled to the inputs via stray capacitance. This oscillation manifests itself during output transitions (V_{OL} to V_{OH}). To alleviate this situation, input resistors < 10 kΩ should be used.

The addition of positive feedback (< 10 mV) is also recommended. It is good design practice to ground all unused pins.

Differential input voltages may be larger than supply voltage without damaging the comparator's inputs. Voltages more negative than –0.3 V should not be used.

Figure 7. Zero Crossing Detector (Single Supply)

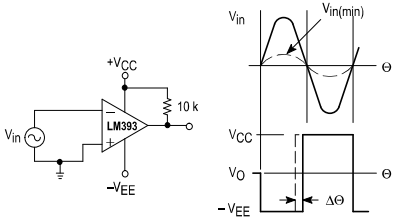


D1 prevents input from going negative by more than 0.6 V.

$R1 + R2 = R3$

$R3 \leq \frac{R5}{10}$ for small error in zero crossing.

Figure 8. Zero Crossing Detector (Split Supply)



$V_{in(min)} \approx 0.4$ V peak for 1% phase distortion (ΔΘ).

Figure 9. Free–Running Square–Wave Oscillator

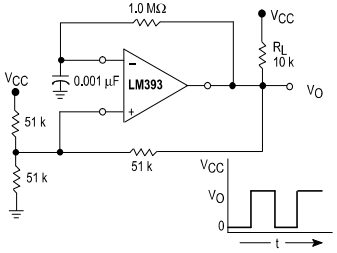
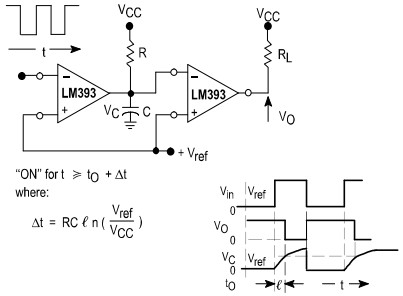


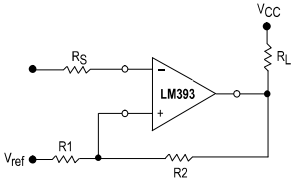
Figure 10. Time Delay Generator



"ON" for $t \geq t_O + \Delta t$ where:

$\Delta t = RC \ell n \left(\frac{V_{ref}}{V_{CC}} \right)$

Figure 11. Comparator with Hysteresis



$R_S = R1 \,||\, R2$

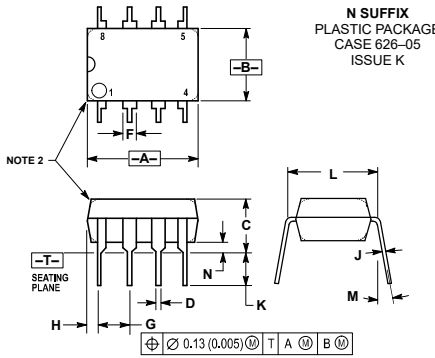
$V_{th1} = V_{ref} + \frac{(V_{CC} - V_{ref})\ R1}{R1 + R2 + R_L}$

$V_{th2} = V_{ref} - \frac{(V_{ref} - V_{O\ Low})\ R1}{R1 + R2}$

LM393, LM393A, LM293, LM2903, LM2903V

OUTLINE DIMENSIONS

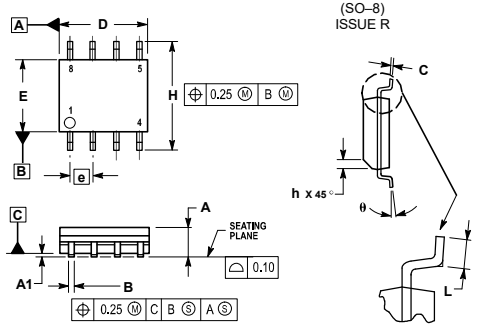
N SUFFIX
PLASTIC PACKAGE
CASE 626–05
ISSUE K



NOTES:
1. DIMENSION L TO CENTER OF LEAD WHEN FORMED PARALLEL.
2. PACKAGE CONTOUR OPTIONAL (ROUND OR SQUARE CORNERS).
3. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	9,40	10,16	0,370	0,400
B	6,10	6,60	0,240	0,260
C	3,94	4,45	0,155	0,175
D	0,38	0,51	0,015	0,020
F	1,02	1,78	0,040	0,070
G	2,54 BSC		0,100 BSC	
H	0,76	1,27	0,030	0,050
J	0,20	0,30	0,008	0,012
K	2,92	3,43	0,115	0,135
L	7,62 BSC		0,300 BSC	
M	—	10°	—	10°
N	0,76	1,01	0,030	0,040

D SUFFIX
PLASTIC PACKAGE
CASE 751–05
(SO–8)
ISSUE R



NOTES:
1. DIMENSIONING AND TOLERANCING PER ASME Y14.5M, 1994.
2. DIMENSIONS ARE IN MILLIMETERS.
3. DIMENSION D AND E DO NOT INCLUDE MOLD PROTRUSION.
4. MAXIMUM MOLD PROTRUSION 0.15 PER SIDE.
5. DIMENSION B DOES NOT INCLUDE MOLD PROTRUSION, ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.127 TOTAL IN EXCESS OF THE B DIMENSION AT MAXIMUM MATERIAL CONDITION.

	MILLIMETERS	
DIM	MIN	MAX
A	1,35	1,75
A1	0,10	0,25
B	0,35	0,49
C	0,18	0,25
D	4,80	5,00
E	3,80	4,00
e	1,27 BSC	
H	5,80	6,20
h	0,25	0,50
L	0,40	1,25
θ	0°	7°

How to reach us:
USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036. 1–800–441–2447 or 602–303–5454

MFAX: RMFAX0@email.sps.mot.com – TOUCHTONE 602–244–6609
INTERNET: http://Design–NET.com

JAPAN: Nippon Motorola Ltd.; Tatsumi–SPD–JLDC, 6F Seibu–Butsuryu–Center, 3–14–2 Tatsumi Koto–Ku, Tokyo 135, Japan. 03–81–3521–8315

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852–26629298

15 Datos relevantes I2C

Detection of START and STOP conditions by devices connected to the bus is easy if they incorporate the necessary interfacing hardware. However, microcontrollers with no such interface have to sample the SDA line at least twice per clock period to sense the transition.

3.1.5 Byte format

Every byte put on the SDA line must be eight bits long. The number of bytes that can be transmitted per transfer is unrestricted. Each byte must be followed by an Acknowledge bit. Data is transferred with the Most Significant Bit (MSB) first (see Figure 6). If a slave cannot receive or transmit another complete byte of data until it has performed some other function, for example servicing an internal interrupt, it can hold the clock line SCL LOW to force the master into a wait state. Data transfer then continues when the slave is ready for another byte of data and releases clock line SCL.

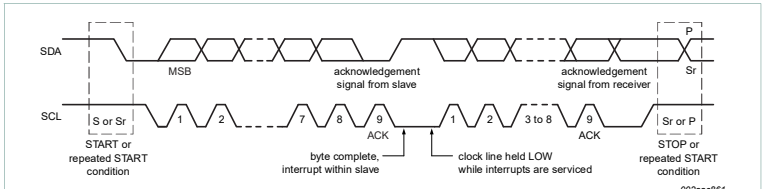


Fig 6. Data transfer on the I²C-bus

3.1.6 Acknowledge (ACK) and Not Acknowledge (NACK)

The acknowledge takes place after every byte. The acknowledge bit allows the receiver to signal the transmitter that the byte was successfully received and another byte may be sent. The master generates all clock pulses, including the acknowledge ninth clock pulse.

The Acknowledge signal is defined as follows: the transmitter releases the SDA line during the acknowledge clock pulse so the receiver can pull the SDA line LOW and it remains stable LOW during the HIGH period of this clock pulse (see Figure 4). Set-up and hold times (specified in Section 6) must also be taken into account.

When SDA remains HIGH during this ninth clock pulse, this is defined as the Not Acknowledge signal. The master can then generate either a STOP condition to abort the transfer, or a repeated START condition to start a new transfer. There are five conditions that lead to the generation of a NACK:

1. No receiver is present on the bus with the transmitted address so there is no device to respond with an acknowledge.
2. The receiver is unable to receive or transmit because it is performing some real-time function and is not ready to start communication with the master.
3. During the transfer, the receiver gets data or commands that it does not understand.
4. During the transfer, the receiver cannot receive any more data bytes.
5. A master-receiver must signal the end of the transfer to the slave transmitter.