

**Direct Software LCD Drive with
ST621x and ST626x**

T.CASTAGNET / J. NICOLAI / N. MICHEL

INTRODUCTION

This note describes a technique for driving a Liquid Crystal Display (LCD) with a standard ST62 microcontroller (MCU), without any dedicated LCD driver. This technique offers a display capability for applications which require a small display at a low cost together with the versatile capabilities of the standard ST62xx MCU. Higher display requirements are easily handled by dedicated members of the ST62 MCU family, for example the ST6240.

The first part of this note describes the typical waveforms required to drive an LCD correctly with a multiplexing rate of 1 or 2 (duplex). The following parts present two solutions based on standard ST62 MCUs driving directly the LCD. The first is based on an ST6215 without using software interrupts and the second on an ST6265 where the LCD is controlled by timer interrupts.

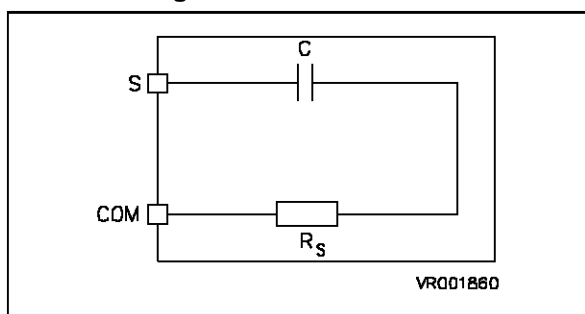
In both examples the program size, the CPU time occupation due to the LCD drive and the number of surrounding components are minimized. Consequently many additional tasks can be added to the MCU program.

LCD requirements

With a zero Root Mean Square (RMS) voltage applied to it, an LCD is practically transparent. The LCD contrast, which makes the segments turn dark or opaque and thus "on", is caused by the difference between the RMS LCD voltage applied and the LCD threshold voltage, specific to each LCD type.

The applied LCD voltage must alternate to give a zero DC value in order to ensure a long life time of the LCD. The higher the multiplexing rate is, the lower the contrast, also the period of the signal has to be short enough to avoid visible flickering of the LCD display.

The LCD voltage for each segment equals to the difference between the S and COM voltages (see Figure 1).

**Figure 1. Equivalent Electrical Schematic
of an LCD Segment**

- DC value should never be more than 100mV. Else time life can be shorten.
- Frequency range is 30 - 2000Hz typically. Less, it flickers; more, consumption grows.

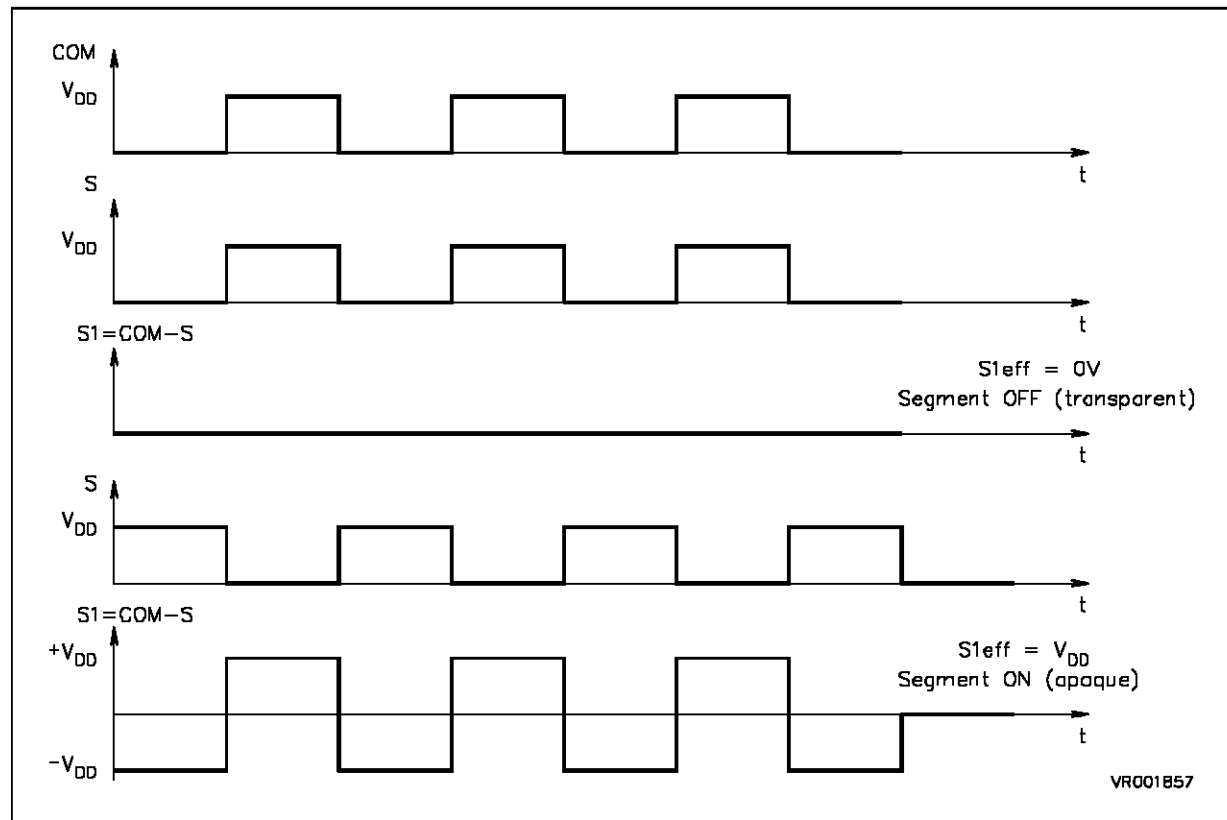
Direct LCD Drive

Direct LCD drive

Each LCD segment is connected to an I/O "Segment" and to one backplane common to all the segments. A display using S segments is driven with $S+1$ MCU output lines. The backplane is driven with a signal "COM" controlled between 0 and V_{DD} with a duty cycle of 50%.

When selecting a segment "ON", a signal with opposite polarity to "COM" is sent to the corresponding "Segment" pin. When the non-inverted signal "COM" is sent to the "Segment" pin, the segment is "OFF". Using an MCU the I/O operate in output mode either at the logic levels 0 or 1.

Figure 2. LCD Signals for Direct drive



Duplexed LCD drive

For duplexed drive, two backplanes are used instead of one. Each LCD pin is connected to two LCD segments, each one connected on the other side to one of the two backplanes. Thus, only $(S/2)+2$ MCU pins are necessary to drive an LCD with S segments.

Three different voltage levels have to be generated on the backplanes : 0, $V_{DD}/2$ and V_{DD} . The "Segment" voltage levels are 0 and V_{DD} only. The LCD segment is inactive if the RMS voltage is below the LCD threshold voltage and is active if the LCD RMS voltage is above the threshold voltage. Figure 4 shows typical Backplane, Segment and LCD waveforms.

The intermediate voltage $V_{DD}/2$ is only required for the "Backplane" voltages. The ST62 I/O pins selected as "Backplanes" are set by software to output mode for 0 or V_{DD} levels and to high impedance input mode for $V_{DD}/2$. This voltage $V_{DD}/2$ is defined by two equal valued resistors externally connected to the I/O pin.

By using an MCU with flexible I/O pin configuration such as the ST6215 or ST6265, duplexed LCD drive can be made with only 4 additional resistors.

Figure 3. Basic LCD Segment Connection in duplexed mode

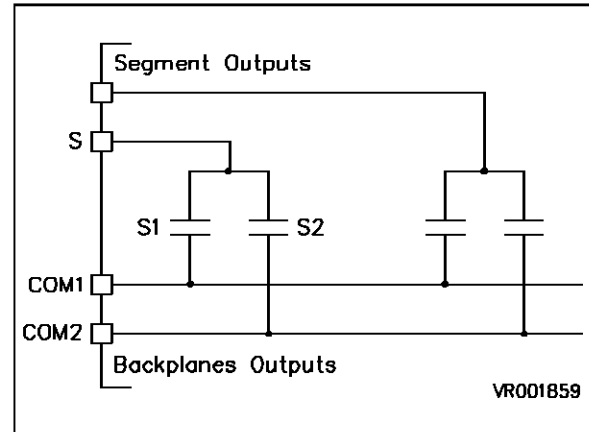
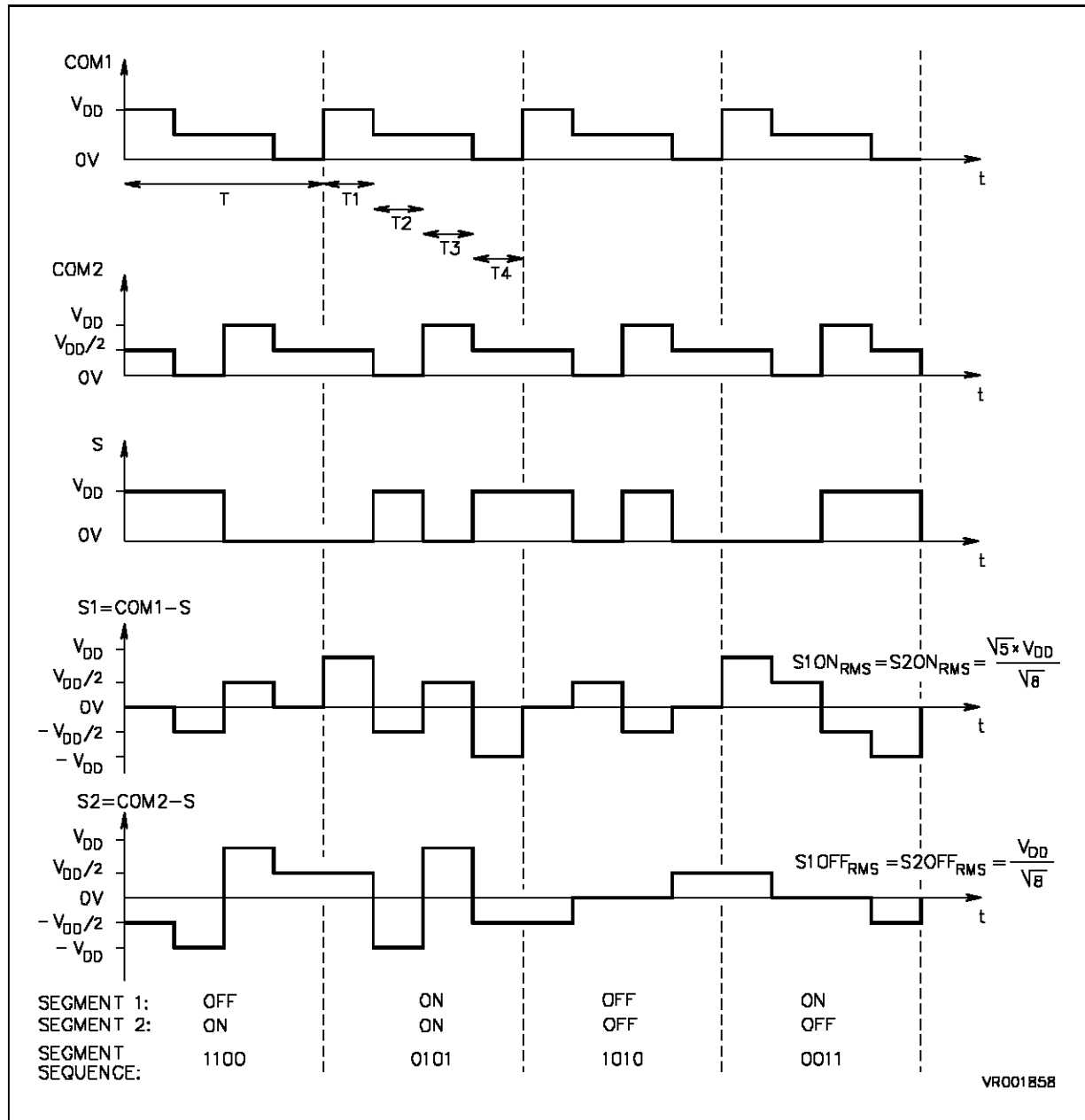


Figure 4. LCD Signals for Duplexed Mode (Used in ST6215 Example)



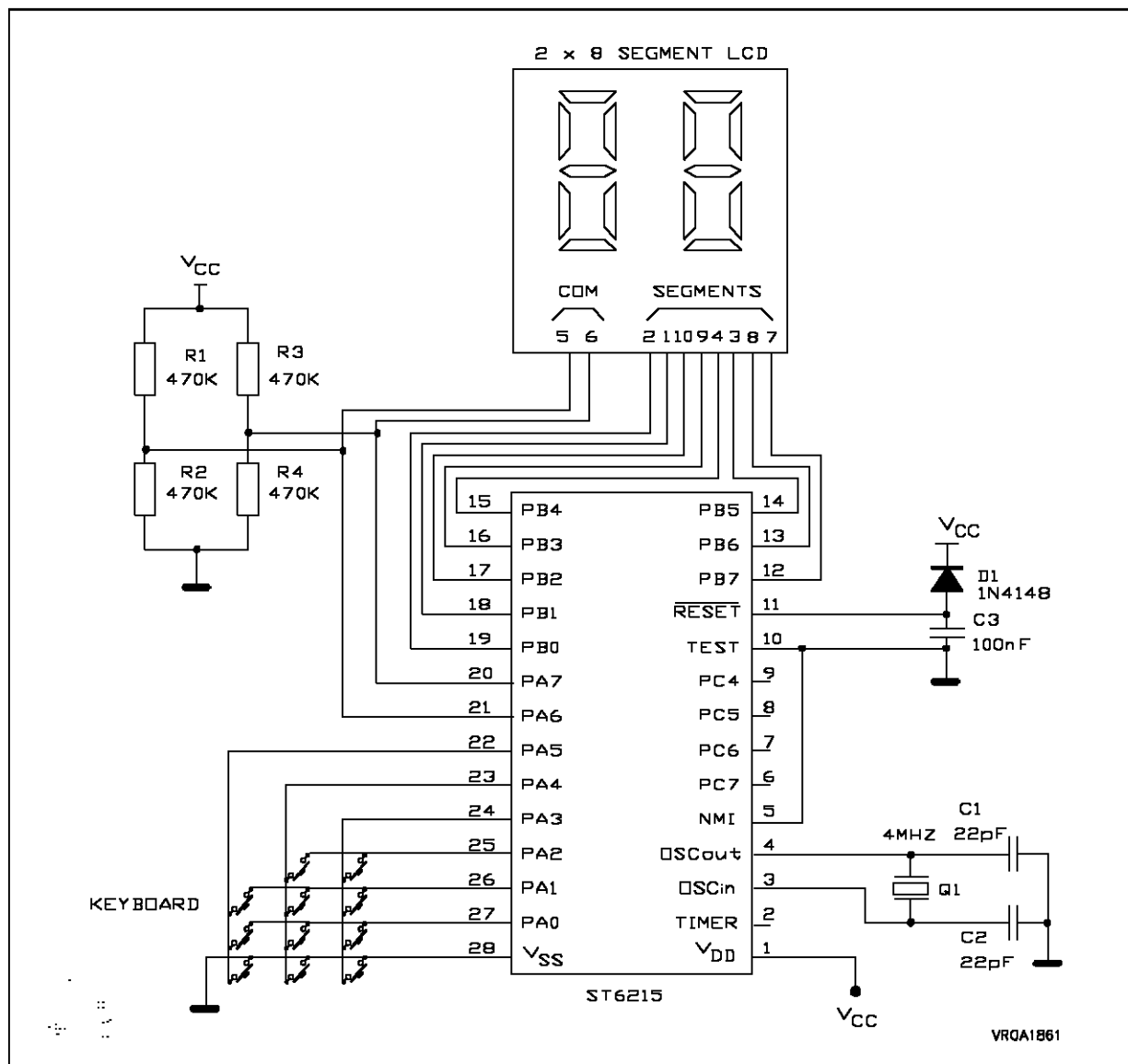
EXAMPLE OF DUPLEXED LCD DRIVE WITH ST6215

The following example describes the drive in duplexed mode of an LCD with an ST6215. The software is made in such way that no interrupt is generated and an OFF state for the LCD is possible. The only components necessary are those to drive the ST62 (oscillator, reset,...) and four resistors to generate the backplane intermediate voltages.

The ST6215 has 20 I/O pins, thus it is able to drive up to 36 LCD segments and 2 backplanes. One digit is defined with 8 segments connected to 2 backplanes. Each digit can display 11 values, from 0 to 9 and no display.

Each value to be displayed is associated to a certain LCD waveform. One LCD waveform period is separated in 4 steps corresponding to the 3 I/O configurations (1-0-input). A look-up table stores the bytes which relate the I/O configuration to the value to display.

Figure 5. ST6215 Based Example



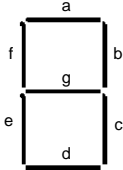
Direct LCD Drive

The following tables show an example of linking one LCD digit of the display with the relevant MCU port sequences. The second digit follows the same scheme. These examples are for the sample LCD display, please adapt these tables to your particular LCD.

The digit segments are connected to PB0-PB3 lines in this example. Schematic is shown in Figure 5. Each digit configuration defines a segment status. Each couple of segments defines a timing sequence to be output by MCU "Segment" lines (see Figure 4). These sequences are coded inside the ST6215.

Table 1: Example of drive of a digit with ST6215

Digit Segments to Display



Segment Connections

Segment Line	PB3	PB2	PB1	PB0
COM1 (PA6)	d	e	f	a
COM2 (PA7)	OFF	c	g	b

Digit Required	Segments Status					MCU Output Sequences
	COM1	ON	ON	ON	ON	
[]	COM2	OFF	ON	OFF	ON	0h 5h Ah Fh
	Timing Sequences					
	T1	0	0	0	0	
	T2	0	1	0	1	
	T3	1	0	1	0	
T4	1	1	1	1		

Digit Required	Segments Status					MCU Output Sequences
	COM1	OFF	OFF	OFF	OFF	
Blank	COM2	OFF	OFF	OFF	OFF	Fh 0h Fh 0h
	Timing Sequences					
	T1	1	1	1	1	
	T2	0	0	0	0	
	T3	1	1	1	1	
T4	0	0	0	0		

Digit Required	Segments Status					MCU Output Sequences
	COM1	OFF	OFF	OFF	OFF	
[]	COM2	OFF	ON	OFF	ON	Fh 5h Ah 0h
	Timing Sequences					
	T1	1	1	1	1	
	T2	1	0	1	0	
	T3	0	1	0	1	
T4	0	0	0	0		

Digit Required	Segments Status					MCU Output Sequences
	COM1	OFF	OFF	ON	OFF	
[]	COM2	OFF	ON	ON	ON	Dh 7h 8h 2h
	Timing Sequences					
	T1	1	1	0	1	
	T2	0	1	1	1	
	T3	1	0	0	0	
T4	0	0	1	0		

Digit Required	Segments Status					MCU Output Sequences
	COM1	ON	ON	OFF	ON	
[]	COM2	OFF	OFF	ON	ON	2h 3h Ch Dh
	Timing Sequences					
	T1	0	0	1	0	
	T2	0	0	1	1	
	T3	1	1	0	0	
T4	1	1	0	1		

Digit Required	Segments Status					MCU Output Sequences
	COM1	ON	OFF	ON	ON	
[]	COM2	OFF	ON	ON	OFF	4h 6h 9h Bh
	Timing Sequences					
	T1	0	1	0	0	
	T2	0	1	1	0	
	T3	1	0	0	1	
T4	1	0	1	1		

Digit Required	Segments Status					MCU Output Sequences
	COM1	ON	OFF	OFF	ON	
[]	COM2	OFF	ON	ON	ON	6h 7h 8h 9h
	Timing Sequences					
	T1	0	1	1	0	
	T2	0	1	1	1	
	T3	1	0	0	0	
T4	1	0	0	1		

Digit Required	Segments Status					MCU Output Sequences
	COM1	ON	ON	ON	ON	
[]	COM2	OFF	ON	ON	OFF	0h 6h 9h Fh
	Timing Sequences					
	T1	0	0	0	0	
	T2	0	1	1	0	
	T3	1	0	0	1	
T4	1	1	1	1		

Digit Required	Segments Status					MCU Output Sequences	
	COM1	OFF	OFF	OFF	ON		
	COM2	OFF	ON	OFF	ON		
	Timing Sequences					MCU Output Sequences	
	T1	1	1	1	0		Eh
	T2	0	1	0	1		5h
	T3	1	0	1	0		Ah
	T4	0	0	0	1		1h

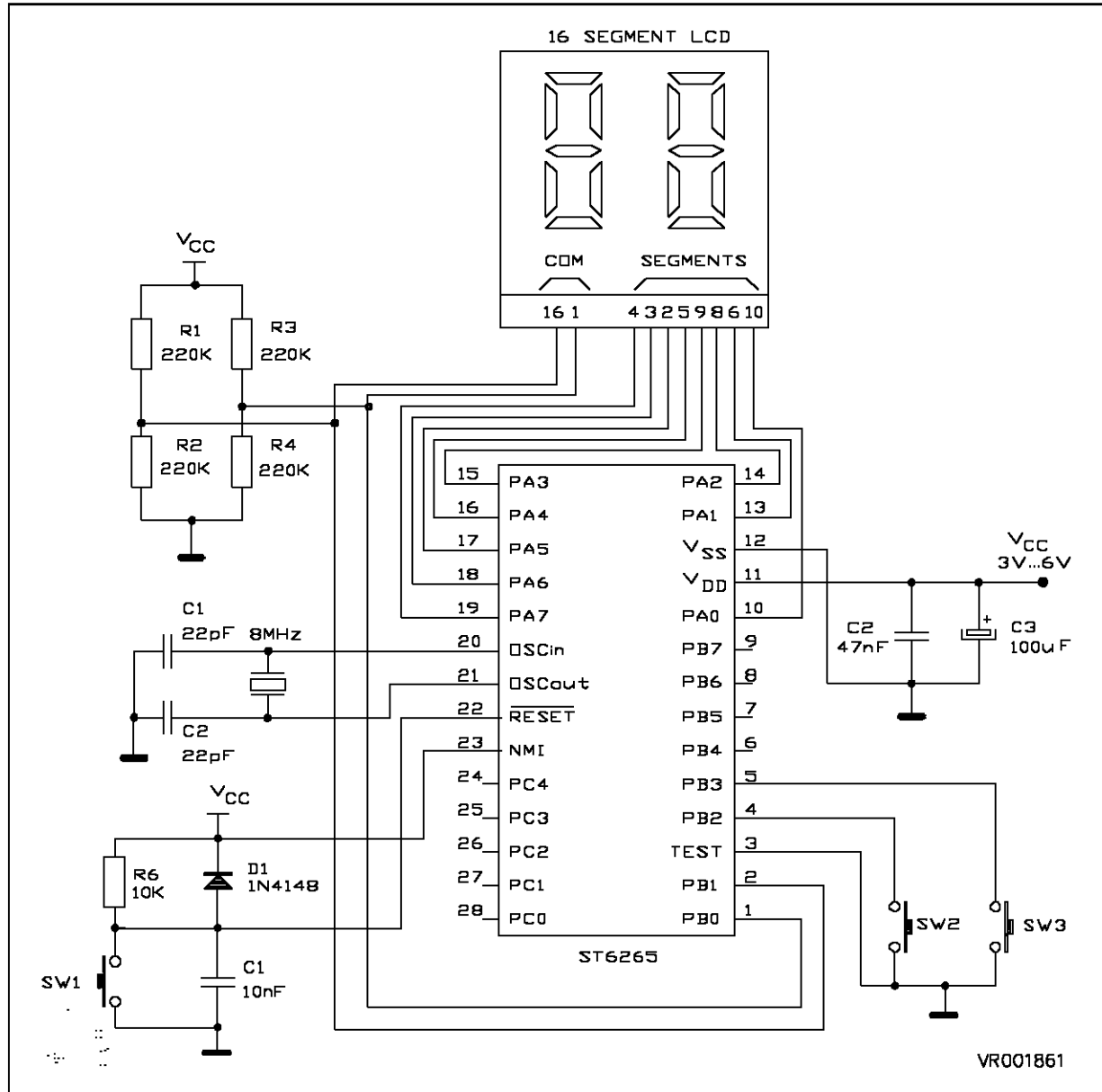
Digit Required	Segments Status					MCU Output Sequences	
	COM1	ON	OFF	ON	ON		
	COM2	OFF	ON	ON	ON		
	Timing Sequences					MCU Output Sequences	
	T1	0	1	0	0		4h
	T2	0	1	1	1		7h
	T3	1	0	0	0		8h
	T4	1	0	1	1		Bh

Digit Required	Segments Status					MCU Output Sequences	
	COM1	ON	ON	ON	ON		
	COM2	OFF	ON	ON	ON		
	Timing Sequences					MCU Output Sequences	
	T1	0	0	0	0		0h
	T2	0	1	1	1		7h
	T3	1	0	0	0		8h
	T4	1	1	1	1		Fh

EXAMPLE OF DUPLEXED LCD DRIVE WITH ST6265

In this example the LCD drive tasks are controlled by interrupts from an on-chip timer. This software block can be easily included in a central task such as motor control, temperature measurement or heating. Figure 6 shows the circuit of the application. Only 4 resistors are added to drive the LCD in the duplexed mode.

Figure 6. ST6265 Based Example



The software for the LCD display operates in interrupt mode, driving I/O pins by the CPU at instants defined by the Timer1 interrupts. When the LCD drive tasks are finished, the main application program can continue.

The major tasks of the LCD display routine are:

- generation of the alternate signals which control the backplanes (PB0, PB1)
- drive of the LCD segments through PortA
- conversion of the hexadecimal data to decimal data
- program Timer1 for the next drive sequence

The backplane pattern sequences are different between ST6215 and ST6265 examples. The ST6215 software is based on non-symmetrical signals (see Figure 4). The backplane patterns are symmetrical in the ST6265 software. Both are equivalent on the LCD viewpoint.

The ROM code size used is 300 bytes for the program and 256 bytes for the tables. With an CPU frequency of 8MHz, the display task duration is 240 μ s and the full task duration including decimal conversion is 510 μ s. With an LCD period signal of 14ms, the CPU duty cycle of occupation is 2.5% for the LCD drive task. The program can be adjusted to other applications by modifying the timer duration (FASTIM) and the segment drive byte table. The LCD drive phase may also be synchronized to the mains zero crossing or a software loop duration, saving the timer for other tasks.

Summary

The examples presented in this note show that a simple LCD can be driven directly by standard ST621x/2x/6x/9x microcontrollers. The ST62's flexible I/O configuration and the large voltage range of operation of the ST62 family MCU allow an LCD driver to be achieved with very few surrounding components, small CPU time occupation and reduced ROM program size.

Such an approach is a very cost effective solution for simple LCD displays operating with a multiplexing rate of 1 or 2 and up to 36 segments. For LCDs requiring more segment drive capability and/or higher multiplexing rates ST624x and ST628x provide highly integrated and easy to implement solutions.

Such LCD drive features can easily be included in a larger application including keyboard interface, sensor display or motor control.

We thank the company Akotronic for the ST6215 application example they have developed for this note.

Annex 1: Software of the ST6215 based application

Annex 2: Flowchart and software of the ST6265 based application

ANNEX 1: Software of the ST6215 Based Application

```
*****
;DEMONSTRATION SOFTWARE FOR MANAGEMENT OF A BI-PLEXED LIQUID CRISTAL
;DISPLAY ( LCD ) WITH ST621X OR ST622X SGS THOMSON MICROCONTROLLERS
*****
;*
;          This program has been developped by AKOTRONIC comp.
;*
;          PARC DE LA MOTHE 03400 YZEURE FRANCE
*****
;*
;          PROGRAM FOR LIQUID CRISTAL DISPLAY DRIVE
*****
;*
;          REGISTER DECLARATION
*****

.ROMSIZE 4 .
VERS "ST6225"
; SWD, 4MHZ
x .DEF 80h!m ; INDEX REGISTER
Y .DEF 81h!m ; INDEX REGISTER
V .DEF 82h ; SHORT DIRECT REGISTER
W .DEF 83h ; SHORT DIRECT REGISTER
A .DEF 0FFh!m ; ACCUMULATOR
DRA .DEF 0C0h ; PORT A DATA REGISTER
DRB .DEF 0C1h ; PORT B DATA REGISTER
DRC .DEF 0C2h ; PORT C DATA REGISTER
DDRA .DEF 0C4h ; PORT A DIRECTION REGISTER
DDRB .DEF 0C5h ; PORT B DIRECTION REGISTER
DDRC .DEF 0C6h ; PORT C DIRECTION REGISTER
IOR .DEF 0C8h ; INTERRUPT OPTION REGISTER
DWR .DEF 0C9h ; DATA ROM WINDOW REGISTER
ORA .DEF 0CCh ; PORT A OPTION REGISTER
ORB .DEF 0CDh ; PORT B OPTION REGISTER
ORC .DEF 0CEh ; PORT C OPTION REGISTER
ADR .DEF 0D0h ; A/D DATA REGISTER
ADCR .DEF 0D1h ; A/D CONTROL REGISTER
PSC .DEF 0D2h ; TIMER PRESCALER REGISTER
TCR .DEF 0D3h ; TIMER COUNTER REGISTER
TSCR .DEF 0D4h ; TIMER STATUS CONTROL REGISTER
WDR .DEF 0D8h ; WATCHDOG REGISTER

;*****
;*
;          DATA DECLARATION
;*****
TOUCHU .DEF 084h ;Low Significant DIGIT button ( L.S. DIGIT )
TOUCHD .DEF 085h ;More significant DIGIT button ( M.S. DIGIT )
TOUCH .DEF 086h ;pushed button
COPYA .DEF 087h ;COPY of PORT A
COPYB .DEF 088h ;COPY of PORT B
COPYC .DEF 089h ;COPY of PORT C
TABD .DEF 08Ah ;data/ROM window address to display M.S. DIGIT
TABU .DEF 08Bh ;data/ROM window address to display L.S. DIGIT
LOOP .DEF 08Ch ;LOOP
RELACHE .DEF 08Dh ;latch counter
TOUCHP .DEF 08Eh ;previous validated button
FLAGS .DEF 08Fh ;FLAGS : 0/ push on/off
;*****
```

ANNEX 1: Software of the ST6215 Based Application (Continued)

```

;*****
;*          TABLE 1 of the Low Significant DIGIT ( L.S. DIGIT )
;*****
        .ORG 0F00H
        .BYTE 00FH,09FH,06FH,0FFH,0FFH,09FH,06FH,00FH
        .BYTE 04FH,0CFH,03FH,0BFH,05FH,0DFH,02FH,0AFH
        .BYTE 0BFH,0DFH,02FH,04FH,01FH,05FH,0AFH,0EFH
        .BYTE 00FH,05FH,0AFH,0FFH,07FH,09FH,06FH,08FH
        .BYTE 00FH,0DFH,02FH,0FFH,01FH,0DFH,02FH,0EFH
        .BYTE 00H,00H,00H,00H,00H,00H,00H,00H
        .BYTE 00H,00H,00H,00H,00H,00H,00H,00H
        .BYTE 00H,00H,00H,00H,00H,00H,00H,00H
;*****
;*****
;*          TABLE 2 of the More Significant DIGIT ( M.S. DIGIT )
;*****
        .ORG 0F40H
        .BYTE 0FFH,0F0H,0FFH,0F0H,0FFH,0F5H,0FAH,0F0H
        .BYTE 0F2H,0F3H,0FCH,0FDH,0F6H,0F7H,0F8H,0F9H
        .BYTE 0FDH,0F7H,0F8H,0F2H,0F4H,0F6H,0F9H,0FBH
        .BYTE 0F0H,0F6H,0F9H,0FFH,0FEH,0F5H,0FAH,0F1H
        .BYTE 0F0H,0F7H,0F8H,0FFH,0F4H,0F7H,0F8H,0FBH
        .BYTE 00H,00H,00H,00H,00H,00H,00H,00H
        .BYTE 00H,00H,00H,00H,00H,00H,00H,00H
        .BYTE 00H,00H,00H,00H,00H,00H,00H,00H
;*****
;*****
;*          INTERRUPT VECTORS
;*****
        .ORG 0FF0h
IT_ADC      NOP
            RETI

IT_TIMER    JP  T_IT_TIMER
IT_PORTBC   JP  T_ITPBC
IT_PORTA    JP  T_ITPA

            NOP
            NOP
            NOP
            NOP

NMI         NOP
            RETI

RES         JP  DEBUT
;*****

```

ANNEX 1: Software of the ST6215 Based Application (Continued)

```

;*****
;*          INITIALIZATION SUBROUTINE
;*****
.ORG 880h
DEBUT  RETI          ; END OF RESET INTERRUPT
      LDI DDRA,0C7H   ; A0 to A2 PUSH PULL OUTPUT = 0
      LDI ORA,0C7H   ; A3 to A5 pull up input ; A6 & A7 output = 0
      LDI DRA,00H    ; A3 to A5 for keyboard ; A6 -> BP1, A7 -> BP2
      LDI COPYA,00H
      LDI DDRB,0FFH  ; B0 A B7 push pull output = 0
      LDI ORB,0FFH  ; port B controls LC Display
      LDI DRB,00H   ;
      LDI COPYB,00H ;
      LDI DDRC,00H  ;
      LDI ORC,00H   ; C4 C5 C6 C7 unused inputs
      LDI DRC,00H   ;
      LDI COPYC,00H ;
      LDI DWR,3CH   ; origin of the table
      LDI FLAGS,00h ; reset flags
;*****
;*          END OF INITIALIZATION SUBROUTINE
;*****
;*****
;*          MAIN LCD DRIVE SUBROUTINE
;*****
;*
;* task : generate alternative signals to control LCD backplanes BP1/BP2
;*       drive the segments of the LCD
;*       calculate duration of each duration phase
;*****
SOMMEIL RES 0,FLAGS
      LDI ORA,0CFh   ; A3 becomes interrupt input
      LDI IOR,10h   ; valid interrupt
      STOP          ; wait at ON/OFF button activation
      LDI ORA,0C7h  ;
      CLR IOR       ; inihabit INTERRUPTS
      CALL CLAVIER  ; keyboard test
      JRR 0,FLAGS,SOMMEIL ; IF no push on ON/OFF button , THEN stand by
I1BOUCLE RES 0,FLAGS ; ELSE INITIALIZATION of MAIN LOOP
      LDI TOUCHU,00h ; ON/OFF FLAG is reset, UNITEE A 0
      LDI TOUCHD,00h ; reset M.S. DIGIT
      LDI IOR,10h   ; valid interrupts
;*****
BOUCLE1 LDI TCR,18   ; initialization of timer
      LDI TSCR,7Fh  ; program it at 1,5 ms
      LD A,TOUCHU   ; determine data/rom window address
      SLA A         ;
      SLA A         ; multiply TOUCHU by 4
      LD TABU,A     ; initialize data/rom address
      LD A,TOUCHD   ; of TABLES 1 & 2
      SLA A         ;
      SLA A         ;
      LD TABD,A     ;
      CALL DATALCD  ; determine segments driver byte for PHASE 1
      LDI ORA,47h   ; BP1 = Vdd ; BP2 = Vdd/2 therefore
      LDI DDRA,47h  ; A6 becomes push pull output at VDD
      LDI DRA,0C0h  ; A7 becomes high impedance input
      LD DRB,A      ; load segments driver byte on port
      CALL CLAVIER  ; test of keyboard
      WAIT
;*****

```

ANNEX 1: Software of the ST6215 Based Application (Continued)

```

BOUCLE2  LDI  TCR,18          ; timer initialization
          LDI  TSCR,7Fh       ; program it at 1.5 ms
          INC  TABU           ; determine data/rom window address
          INC  TABD           ; of tables 1 & 2 for phase 2
          CALL DATALCD        ; determine segments driver byte for PHASE 2
          LDI  ORA,07h        ; BP1 = Vdd/2 ; BP2 = Vss therefore
          LDI  DRA,40h
          LDI  DDRA,87h       ; A6 becomes high impedance input
          LDI  ORA,87h       ; & A7 becomes push pull output at Vss
          LD   DRB,A          ; load segments driver byte on port
          WAIT
;*****
BOUCLE3  LDI  TCR,18          ; timer initialization
          LDI  TSCR,7Fh       ; program it at 1.5 ms
          INC  TABU           ; determine data/rom window address
          INC  TABD           ; of tables 1 & 2 for phase 2
          CALL DATALCD        ; determine segments driver byte for PHASE 3
          LDI  DRA,0C0h       ; BP1 = Vdd/2 ; BP2 = Vdd therefore
          ; A6 remains high impedance input
          ; A7 becomes push pull output at Vdd
          LD   DRB,A          ; load segments driver byte on port
          WAIT
;*****
BOUCLE4  LDI  TCR,18          ; timer initialization
          LDI  TSCR,7Fh       ; program it at 1.5 ms
          INC  TABU           ; determine data/rom window address
          INC  TABD           ; of tables 1 & 2 for phase 2
          CALL DATALCD        ; determine segments driver byte for PHASE 4
          LDI  ORA,07h        ; BP1 = Vss ; BP2 =A Vdd/2 therefore
          LDI  DRA,80h
          LDI  DDRA,47h       ; A6 becomes output at Vss
          LDI  ORA,47h       ; A7 becomes high impedance input
          LD   DRB,A          ; load segments driver byte on port
          WAIT
;*****
FINBOUCLE JRR  0,FLAGS,BOUCLE1 ;IF ON/OFF button remains pushed,
          ; THEN circuit is in stand by & display is
          ; ELSE continue digits display
          LDI  DRA,00h
          LDI  DDRA,0C7h
          LDI  ORA,0C7h       ; BP1 & BP2 on output to Vss
          LDI  DRB,00h
PREPSOMM CALL CLAVIER        ; test of keyboard
          LD   A,TOUCH
          CPI  A,0Ah          ; wait falling edge of ON/OFF button
          JRZ PREPSOMM       ; before stop display mode
          JP   SOMMEIL
;*****
;*                               END OF MAIN PROGRAM
;*****

```

ANNEX 1: Software of the ST6215 Based Application (Continued)

```

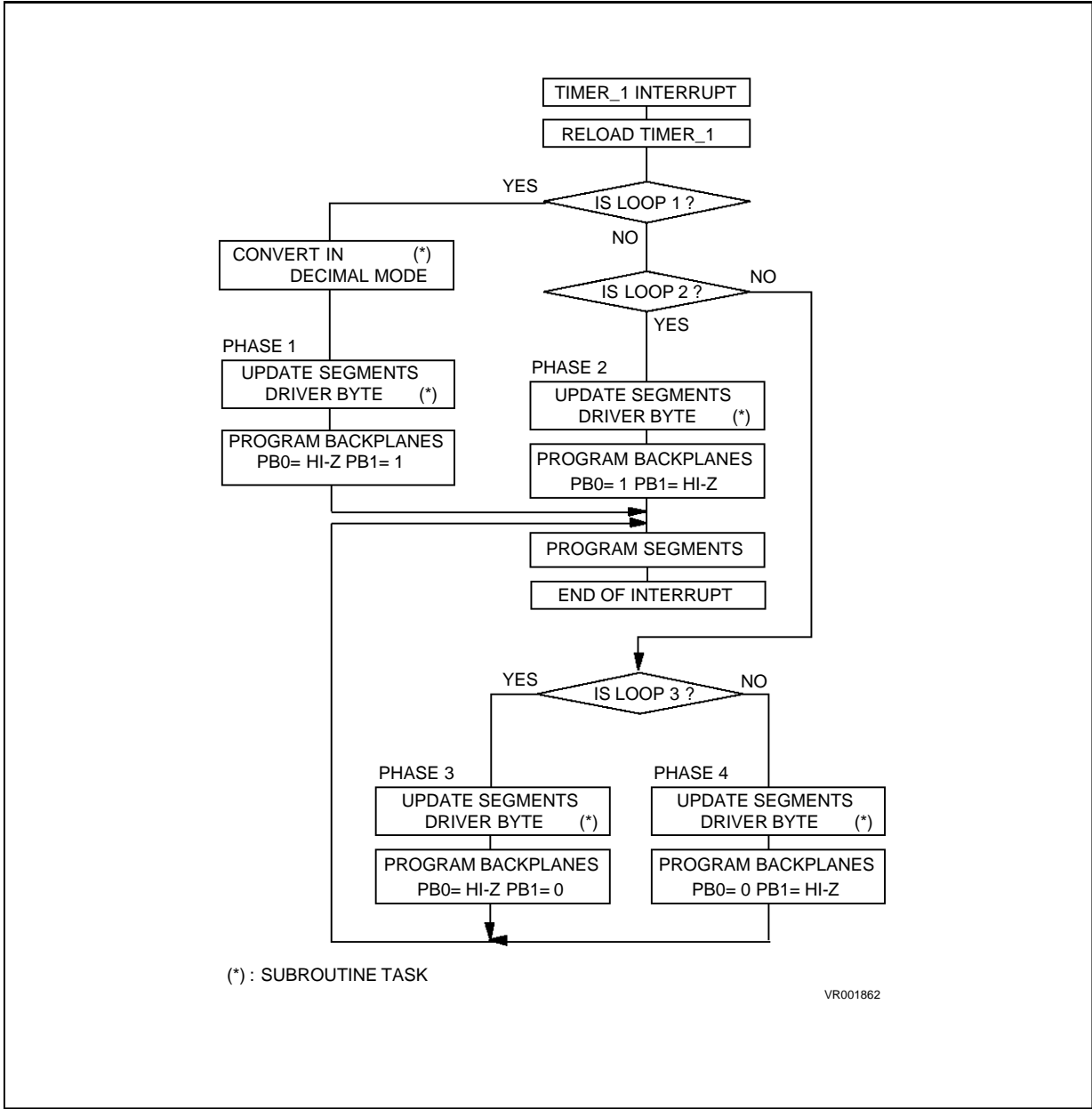
;*****
;*
;*          TABLE SUBROUTINE
;*
;* task : define LCD segments driver byte to load on port B
;*       TABU defines half driver byte for L.S.DIGIT
;*       TABD defines half driver byte for M.S.DIGIT
;*****
DATALCD  LDI  DWR,3Ch          ; move data/rom window to L.S.DIGIT TABLE 2
         LDI  A,40H
         ADD  A,TABU
         LD   X,A
         LD   A,(X)
         LD   Y,A              ; half segment driver byte is loaded
         LDI  DWR,3Dh         ; move data/rom window to M.S.DIGIT TABLE 1
         LDI  A,40h
         ADD  A,TABD
         LD   X,A
         LD   A,(X)
         AND  A,Y              ; LCD driver byte is loaded in accumulator
         RET
;*****
;*          END OF TABLE SUBROUTINE
;*****
;*****
;*          KEYBOARD SUBROUTINE
;*
;* task : controls display operation
;*       searchs TOUCHU (L.S.DIGIT) & TOUCHD (M.S.DIGIT) displayed data
;*
;*****
CLAVIER  LDI  LOOP,02h
CLAVIER1 LD   A,DRA
         ANDI A,38h
         CPI  A,38h           ; test 2 times if some buttons are pushed
         JRNZ CLAVIER2       ; IF yes THEN check them ( A3 -> A5 )
         DEC  LOOP
         JRNZ CLAVIER1
         LDI  TOUCH,0FFh
         JP   CLAVIER4       ; ELSE test if keyboard is changed
CLAVIER2 LD   Y,A
         LD   A,DRA
         ANDI A,38h
         CP   A,Y
         JRZ  TSTCOL         ; IF check is OK , THEN determine column
         LDI  TOUCH,0FFh
         JP   CLAVIER4       ; ELSE test if one button was pushed
TSTCOL   JRR  3,Y,COL1       ; if A3 = 0 then column #1
         JRR  4,Y,COL2       ; if A4 = 0 then column #2
COL3     LDI  TOUCH,3        ; else column #3 is selected so TOUCH <=3
         JP   TSTLIGN
COL2     LDI  TOUCH,2        ;column #2 so TOUCH <=2
         JP   TSTLIGN
COL1     LDI  TOUCH,1        ;column #1 so TOUCH <=1

```


ANNEX 1: Software of the ST6215 Based Application (Continued)

```
*****  
;*          PORT A INTERRUPT SUBROUTINE  
*****  
T_ITPA      NOP  
            RETI  
*****  
;*          END OF PORT A INTERRUPT SUBROUTINE  
*****  
;*          OTHER INTERRUPTS SUBROUTINE  
*****  
T_IT_TIMER  LDI TSCR,00h  
            RETI  
T_ITPBC     NOP  
            RETI  
*****
```

ANNEX 2: Flowchart of the ST6265 Based Application



ANNEX 2: Software of the ST6265 Based Application

```
*****
;*
;*          SGS THOMSON MICROELECTRONICS
;*
;*          CENTRAL APPLICATIONS LABORATORY
;*
;*          ROUSSET FRANCE
;*
*****
;*
;*          19 FEB 1993
;*
;*
;*          LCD005
;*
*****
;*
;*          LCD DRIVER SOFTWARE PROGRAM
;*          ST62E65
;*          DUPLEXED CONTROL OF 2 DIGITS WITH PORT A;
;*          PA0 to PA3 FOR DIGIT1;PA4 to PA7 FOR DIGIT2;
;*          BACPLANES ON PB0 & PB1
;*
;*          with 4 phase sequences generator
;*          with Hexadecimal/Decimal DATA conversion
;*
*****
;*
;*          ST6265/6 Registers Declaration
*****
x      .def 80h          ; Index register.
y      .def 81h          ; Index register.
v      .def 82h          ; Short direct register.
w      .def 83h          ; Short direct register.
a      .def 0ffh         ; Accumulator.
pa     .def 0c0h         ; Port a data register.
pb     .def 0c1h         ; Port b data register.
pc     .def 0c2h         ; Port c data register.
padir  .def 0c4h         ; Port a direction register.
pbdir  .def 0c5h         ; Port b direction register.
pcdir  .def 0c6h         ; Port c direction register.
paopt  .def 0cch         ; Port a option register.
pbopt  .def 0cdh         ; Port b option register.
pcopt  .def 0ceh         ; Port c option register.
ior     .def 0c8h         ; Interrupt Option Register.
drwr   .def 0c9h         ; Data rom window register.
adc     .def 0d0h         ; A/D result register.
adcc   .def 0d1h         ; A/D control register.
pscl   .def 0d2h         ; Timer 1 prescaler register.
tcr1   .def 0d3h         ; Timer 1 counter register.
tscl   .def 0d4h         ; Timer 1 status control register.
```

ANNEX 2: Software of the ST6265 Based Application (Continued)

```

mc      .def 0d5h          ; Timer 2 mode control register.
sc0     .def 0d6h          ; Timer 2 status/control register.
sc1     .def 0d7h          ; Timer 2 status/control register.
rc      .def 0d9h          ; Timer 2 reload/capture register.
cp      .def 0dah          ; Timer 2 compare register.
lr      .def 0dbh          ; Timer 2 load register.

wdt     .def 0d8h          ; Watchdog register.
oscr    .def 0dch          ; Oscillator control register.
lvi     .def 0ddh          ; Multiplex register / lvi flag register.
spirad  .def 0e0h          ; SPI register RAD.
spidiv  .def 0e1h          ; SPI register DIV.
spimod  .def 0e2h          ; SPI register MOD.
mbr     .def 0e8h          ; memory bank register.
eecr    .def 0eah          ; eeprom control register.
;*****
;*
;*****
RAM DATA DEFINITION
;*****
duty0   .def 087h,0ffh,0ffh ; user request reference
counter1 .def 089h,0ffh,0ffh ; duty0 variation rate control byte
DIG1    .def 0a6h,0ffh,0ffh ; data/rom @ of DIGIT1 segment driver
DIG2    .def 0a7h,0ffh,0ffh ; data/rom @ of DIGIT2 segment driver
LCDCTL  .def 0a8h,0ffh,0ffh ; LCD phase sequence control byte
pbbuf   .def 0aah,0ffh,0ffh ; buffer byte of data port b
BCD     .def 0a9h,0ffh,0ffh ; BinCodDec converted DATA
AUX1    .def 0a3h,0ffh,0ffh ; accumulator save byte
AUX2    .def 0a4h,0ffh,0ffh ; data/rom window register save byte
AUX3    .def 0a5h,0ffh,0ffh ; x register save byte
;*****
;*
;*****
EQUATES DEFINITION
;*****
FASTIM  .equ 036h          ; duration of each LCD drive phase (14 ms at 8 MHz)
;*****
;*****
;*****
MACROFUNCTIONS DEFINITION
;*****
.macro jumpnc jpadress,?lbl
        jrc lbl
        jp jpadress
lbl
.endm
;*****
.macro jumpz jpadress,?lbl
        jrnz lbl
        jp jpadress
lbl
.endm
;*****

```

ANNEX 2: Software of the ST6265 Based Application (Continued)

```

;*****
;*                               INTERRUPT VECTORS
;*****

.org      0ff0h
it_tim1  jp  LCD_LP           ; timer1 interrupt synchronizes LCD drive
it_tim2  nop                 ; it.timer2
         reti
it_pc_spi nop                 ; it.port c & SPI
         reti
it_pa_pb nop                 ; it. port a & port b
         reti
         nop
         nop
         nop
         nop
nmi      nop
         reti
res      jp  START

;*****
;*                               MAIN PROGRAM EXAMPLE
;*****

.org      0880h
START    reti ; end of reset interrupt
         call INIT
MAIN     call DUTY0
         jp  MAIN

;*****
;*                               END OF MAIN PROGRAM
;*****

;*****
;*                               MAIN INITIALIZATION SUBROUTINE
;*****

INIT     ldi  wdt,00000111b ; watchdog initialization
         ldi  eecr,040h   ; EEPROM in stand by for power saving
         ldi  oscr,008h   ; CKOUT output disabled for power saving
         ldi  pbdir,00110000b ; b0 & b1 is common :Hi/Impedance Input
         ldi  pbopt,00110000b ; b2 &b3 is input for +/- push button
         ldi  pbbuf,00000011b ; pb buffer byte load
         ldi  pb,00000011b ; b6,b7 in input with pull up
         ldi  padir,0ffh   ; a0 to a7 in push pull output
         ldi  paopt,0ffh   ; pa = driver of LCD segments
         ldi  pa,00h      ; output is zero
         ldi  pcdir,00h    ;
         ldi  pcopt,00h   ; c0 -> c7 inputs with pull up
         ldi  pc,00h      ;
         ldi  drwr,3ch    ; data/rom window origin
         clr  LCDCTL      ; clear LCD phase sequence control
         ldi  ior , 010h  ; interrupt validation
         ldi  tcr1 ,009h  ; load timer1 for LCD phase generation
         ldi  tscr1,07fh  ; timer initialization
         clr  duty0
         ldi  wdt, 00000111b ; hello watchdog
         ret

;*****
;*                               END OF MAIN INITIALIZATION
;*****

```

ANNEX 2: Software of the ST6265 Based Application (Continued)

```

;*****
;*   MAIN TASK EXAMPLE : ACQUISITION OF USER SPEED REFERENCE DUTY0   *
;*   *                                                                *
;*   duty0 is a user reference that can vary from 0 to 255d         *
;*****
DUTY0   jrr    2,pb,slower    ; if PB2=0,then slower ; priority on slower
        jrr    3,pb,faster    ; if PB3=0,then faster ; else continue
        ret     ; if PB2 & PB3 are high; then continue

slower  ld     a,duty0
        cpi   a,000h
        jumpz retour
        ld   a,counter1
        addi a,01h
        ld   counter1,a
        jumpnc retour
        dec  duty0      ; increment duty cycle
        ret

faster  ld     a,duty0
        cpi   a,0ffh
        jrz   retour
        ld   a,counter1
        addi a,01h
        ld   counter1, a
        jrnc retour
        inc  duty0      ; decrement duty cycle

retour  ret

;*****
;*                               end subroutine get_dut0
;*****

;*****
;*   DUPLEXED LCD DRIVER INTERRUPT SUBROUTINE
;*   *
;*   task : generate alternative signals of LCD backplanes control
;*   drive the segments of LCD through port a
;*   calculate duration of each LCD phase
;*   *
;*****
LCD_LP  ldi    wdt,0ffh      ; hello watchdog
;
;   ld     AUX1, a          ; |
;   ld     a, x             ; | save context of main task (if needed)
;   ld     AUX3, a          ; |
;   ldi    tscr1,00h        ; timer stop (if needed)
;
;   ldi    tcr1 , FASTIM ; LCD phase duration calculation
;   ldi    tscr1, 07fh     ; timer initialization
;*****
;* LCD phase generation can be here synchronized by other clock system.
;*   for instance : mains voltage synchronization or external clock
;*****
        jrr   0,LCDCCTL,LOOP1; determine phase1 operation
        jrr   1,LCDCCTL,LOOP2; determine phase2 operation
        jrr   2,LCDCCTL,LOOP3; determine phase3 operation

```

ANNEX 2: Software of the ST6265 Based Application (Continued)

```
***** PHASE 4 *****
LOOP4  inc    DIG1      ; increment DIG1 & DIG2 for phase 4
        inc    DIG2      ;
        call   DATALCD   ; calculate phase4 segments driver byte
        ldi    pbbuf,00000011b ; pb buffer load
        ldi    pb ,00000011b ; pb1 in High Impedance (HI) & pb0 to 0
        ldi    pbdir,00110001b ;
        ldi    pbbuf,00000010b ; pb buffer load
        ldi    pb ,00000010b ;
        ld     pa,a      ; load segment driver byte on port a
        ld     a,AUX3    ;
        ld     x,a      ;
        ld     a,AUX2    ; return to main tasks with context ;
        ld     drwr,a    ; ( AUX2 <== drwr in main program ) ;
        ld     a,AUX1    ;
        clr    LCDCTL    ; end of loop4 & full LCD sequence
        reti

***** PHASE 3 *****
LOOP3  inc    DIG1      ; increment DIG1 & DIG2 for phase 3
        inc    DIG2      ;
        call   DATALCD   ; calculate phase3 segments driver byte
        ldi    pbopt,00110000b ; pb0 in HI & pb1 to 0
        ldi    pbdir,00110010b ;
        ldi    pbbuf,00000001b ; pb buffer load
        ldi    pb, 00000001b ;
        ld     pa ,a     ; load segment driver byte on port a
;      ld     a,AUX3    ;
;      ld     x,a      ;
;      ld     a,AUX2    ; return to main tasks with context
;      ld     drwr,a    ; ( AUX2 <== drwr in main program )
;      ld     a,AUX1    ;
        set    2,LCDCTL  ; end of loop3
        reti

***** PHASE 2 *****
LOOP2  inc    DIG1      ; increment DIG1 & DIG2 for phase 2
        inc    DIG2      ;
        call   DATALCD   ; calculate phase2 segments driver byte
        ldi    pbopt,00110000b ; pb1 in HI & pb0 to 1
        ldi    pbdir,00110001b ;
        ldi    pbopt,00110001b ;
        ld     pa,a      ; load segment driver byte on port a
;      ld     a,AUX3    ;
;      ld     x,a      ;
;      ld     a,AUX2    ; return to main tasks with context
;      ld     drwr,a    ; ( AUX2 <== drwr in main program )
;      ld     a,AUX1    ;
        set    1,LCDCTL  ; end of loop2
        reti
```

ANNEX 2: Software of the ST6265 Based Application (Continued)

```

;***** PHASE 1 *****
LOOP1  call  DECONV      ; do hexa-decimal data conversion
       call  DATALCD    ; calculate phase1 segment driver byte
                          ; driver data is stored in accumulator

       ldi  pbbuf,0000011b ; pb buffer load
       ldi  pb,00000011b  ; b1 to 1 & b0 in HI
       ldi  pbdir,00110010b ; b2 -> b7 unchanged
       ldi  pbopt,00110010b ;
       ld   pa,a          ;load segment driver byte on port a
;       ld   a,AUX3        ;
;       ld   x,a          ;
;       ld   a,AUX2        ; return to main tasks with context
;       ld   drwr,a        ;(AUX2 <= drwr in main program)
;       ld   a,AUX1        ;
       set  0,LCDCTL      ; end of loop1
       reti

;*****
;*          END OF LCD DRIVER SUBROUTINE
;*****
;*****
;*          DEC DATA/LCD SEGMENT CONVERSION SUBROUTINE
;*
;* task : calculate the segments driver byte of the LCD
;*        depends on the displayed data AND on the # of phase
;*        based on DIG1 & DIG2 calculation
;*        LCD segments driver full byte is in accumulator
;*****
DATALCD ldi  drwr,3ch     ; move data/rom window on DIGIT2 table
        ld   a,DIG2
        ld   x,a
        ld   a,(x)
        ld   y,a        ; load DIGIT2 driver half byte (MSB)
        ldi  drwr,3dh    ; move data/rom window on DIGIT1 table
        ld   a,DIG1
        ld   x,a
        ld   a,(x)
        and  a,y        ; load DIGIT1 driver half byte (LSB)
        ret

;*****
;*          END OF DATA/LCD CONVERSION SUBROUTINE
;*****

```

ANNEX 2: Software of the ST6265 Based Application (Continued)

```

;*****
;*      HEXADECIMAL -> DECIMAL DATA CONVERSION SUBROUTINE
;*
;* task : convert an hexadecimal data in a Binary Coded Decimal data
;*        determine data/rom window address of segments driver byte
;*
;*
;*        hexadecimal data is a stable data varying from 0 to 256
;*        result is a percent decimal data varying from 0 to 99
;*        BCD Binary Coded Decimal data
;*
;*        for instance, data is duty0
;*
;*        DIG1 = data/rom WDW @ of DIGIT1 segments driver = 01wxyzAB
;*        DIGIT1 = wxyz ( from 0 to 9 ) and
;*        AB are defined by phase operation : AB = 00 for phasel ...
;*        AB = 11 for phase4
;*        DIG2 = data/rom WDXW @ of DIGIT2 segments driver
;*        similar writing than DIG1
;*
;*****
DECCONV  jrr    7,duty0,ET1    ; DATA <= duty0
         ldi    drwr,039h     ; b7 of DATA is 1 : 50 < DATA < 99
         jp     ET2
ET1      ldi    drwr,038h     ; b7 of DATA is 0 : 00 < DATA < 49
ET2      ld     a,duty0
         rlc   a
         rlc   a
         rlc   a            ; calculation of BinCodDec DATA
         rlc   a            ; address
         rlc   a
         rlc   a
         rlc   a
         set   6,a
         res   7,a
         ld   x,a
         ld   a,(x)
         ld   BCD,a        ; BCD <= Binary Coded Decimal DATA
         andi a,00fh       ; determine DIGIT1 value and the
         sla  a            ; data/rom address of DIGIT1 segments
         sla  a            ; driver
         addi a,040h
         ld   DIG1,a       ; DIG1 = data/rom @ of segment driver
         ld   a,BCD        ; determine DIGIT2 value and the
         andi a,0f0h       ; data/rom address of DIGIT2 segments
         rlc  a            ; driver
         rlc  a
         rlc  a
         sla  a
         sla  a
         addi a,040h
         ld   DIG2,a       ; DIG2 = data/rom @ of LCD driver
         ret
;*****
;*      END OF HEXA -> DE CIMAL DATA CONVERSION SUBROUTINE
;*****

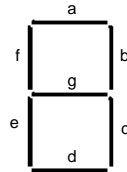
```

ANNEX 2: Software of the ST6265 Based Application (Continued)

```

;*****
;*   TABLE OF HEXA -> DECIMAL DATA CONVERSION ( 00 TO 49 BinCodDec )
;*****
.org       0e00h
.byte     00h,01h,02h,02h,03h,04h,05h,05h
.byte     06h,07h,08h,09h,09h,10h,11h,12h
.byte     12h,13h,14h,15h,16h,16h,17h,18h
.byte     19h,20h,20h,21h,22h,23h,23h,24h
.byte     25h,26h,27h,27h,28h,29h,30h,30h
.byte     31h,32h,33h,34h,34h,35h,36h,37h
.byte     37h,38h,39h,40h,41h,41h,42h,43h
.byte     44h,45h,45h,46h,47h,48h,48h,49h
;*****
;*   END OF HEXA -> DECIMAL DATA CONVERSION TABLE ( 00 to 49 )
;*****
;*****
;*   TABLE OF HEXA -> DECIMAL DATA CONVERSION ( 50 to 99 BinCodDec )
;*****
.org       0e40h
.byte     50h,51h,52h,52h,53h,54h,55h,55h
.byte     56h,57h,58h,59h,59h,60h,61h,62h
.byte     62h,63h,64h,65h,66h,66h,67h,68h
.byte     69h,70h,70h,71h,72h,73h,73h,74h
.byte     75h,76h,77h,77h,78h,79h,80h,80h
.byte     81h,82h,83h,84h,84h,85h,86h,87h
.byte     88h,88h,89h,90h,91h,91h,92h,93h
.byte     94h,95h,95h,96h,97h,98h,98h,99h
;*****
;*   END OF HEXA -> DECIMAL DATA CONVERSION TABLE ( 50 to 99 )
;*****
;*****
;*   SEGMENT CONTROL WITHOUT ANY POINT DISPLAY
;*****
;*PA0,4 --> SEG CB
;*PA1,5 --> SEG DH
;*PA2,6 --> SEG DH
;*PA3,7 --> SEG GA
;*
;*PA0,PA1,PA2,PA3--> DIGIT1
;*PA4,PA5,PA6,PA7--> DIGIT2
;*
;*Backplane #1 (pb1) biases C,D,E,G
;*Backplane #2 (pb0) biases A,B,F,H
;*
;*These tables are dedicated to one LCD type with two digits and LCD
;*control is described above ; when LCD is changing these tables have
;* to be modified

```



ANNEX 2: Software of the ST6265 Based Application (Continued)

```
*****  
;*          DIGIT2  TABLE  
*****  
    .org    0f00h  
    .byte   08fh,02fh,07fh,0dfh,0efh,0efh,01fh,01fh  
    .byte   01fh,06fh,0efh,09fh,04fh,06fh,0bfh,09fh  
    .byte   06fh,0afh,09fh,05fh,04fh,03fh,0bfh,0cfh  
    .byte   00fh,03fh,0ffh,0cfh,0efh,06fh,01fh,09fh  
    .byte   00fh,02fh,0ffh,0dfh,04fh,02fh,0bfh,0dfh  
    .byte   0ffh,0ffh,00fh,00fh,000h,000h,000h,000h  
    .byte   000h,000h,000h,000h,000h,000h,000h,000h  
    .byte   000h,000h,000h,000h,000h,000h,000h,000h  
*****  
;*          DIGIT1  TABLE  
*****  
    .org    0f40h  
    .byte   0f8h,0f2h,0f7h,0fdh,0feh,0feh,0f1h,0f1h  
    .byte   0f1h,0f6h,0feh,0f9h,0f4h,0f6h,0fbh,0f9h  
    .byte   0f6h,0fah,0f9h,0f5h,0f4h,0f3h,0fbh,0fch  
    .byte   0f0h,0f3h,0ffh,0fch,0feh,0f6h,0f1h,0f9h  
    .byte   0f0h,0f2h,0ffh,0fdh,0f4h,0f2h,0fbh,0fdh  
    .byte   0ffh,0ffh,0f0h,0f0h,000h,000h,000h,000h  
    .byte   000h,000h,000h,000h,000h,000h,000h,000h  
    .byte   000h,000h,000h,000h,000h,000h,000h,000h  
*****
```

THE SOFTWARE INCLUDED IN THIS NOTE IS FOR GUIDANCE ONLY. SGS-THOMSON SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM USE OF THE SOFTWARE.

Information furnished is believed to be accurate and reliable. However, SGS-THOMSON Microelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of SGS-THOMSON Microelectronics. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. SGS-THOMSON Microelectronics products are not authorized for use as critical components in life support devices or systems without the express written approval of SGS-THOMSON Microelectronics.

© 1994 SGS-THOMSON Microelectronics - All rights reserved.

Purchase of I²C Components by SGS-THOMSON Microelectronics conveys a license under the Philips I²C Patent. Rights to use these components in an I²C system is granted provided that the system conforms to the I²C Standard Specification as defined by Philips.

SGS-THOMSON Microelectronics Group of Companies

Australia - Brazil - France - Germany - Hong Kong - Italy - Japan - Korea - Malaysia - Malta - Morocco - The Netherlands
Singapore - Spain - Sweden - Switzerland - Taiwan - Thailand - United Kingdom - U.S.A.



LittleDiode supplies new, hard to find or obsolete electronic components and semiconductors all over the world.

With over two million different components listed you are sure to find the part you need.

Feel free to visit us today at our online store:

LittleDiode.com

Looking forward to providing you with the best possible service.