LCD Triplex Drive with COP820CJ

National Semiconductor
Application Note 953
Klaus Jaensch and
Siegfried Rueth
September 1994

INTRODUCTION

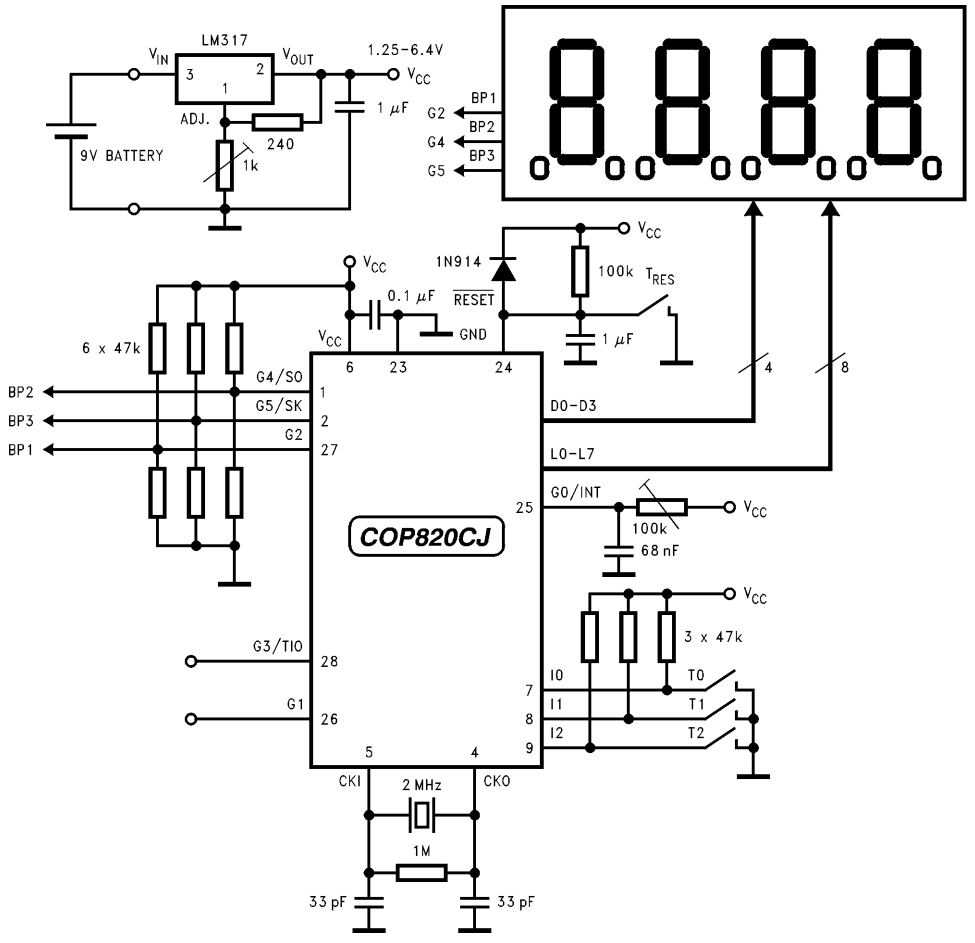
There are many applications which use a microcontroller in combination with a Liquid Crystal Display. The normal method to control a LCD panel is to connect it to a special LCD driver device, which receives the display data from a microcontroller. A cheaper solution is to drive the LCD directly from the microcontroller. With the flexibility of a COP8 microcontroller the multiplexed LCD direct drive is possible. This application note shows a way how to drive a three way multiplexed LCD with up to 36 segments using a 28-pin COP800 device.

ABOUT MULTIPLEXED LCD'S

There is a wide variety of LCD's, ranging from static devices to multiplexed versions with multiplex rates of up to 1:256.

The multiplex rate of a LCD is determined by the number of its backplanes (segment-common planes). The number of segments controlled by one line (with one segment pin) is equal to the number of backplanes on the LCD. So, a three way multiplexed LCD has three backplanes and three segments are controlled with one segment pin. For example in a three way multiplexed LCD with three segment inputs (SA, SB, SC) one can drive a 7-segment digit plus two special segments.

These are $3 \times 3 = 7 + 2 = 9$ segments. The special segments can have an application specific image. (''+'', ''−'', ''.'', ''mA'', . . . etc).



TL/DD/12076–1

FIGURE 1. Schematic for LCD Triplex Driver

RRD-B30M75/Printed in U. S. A.

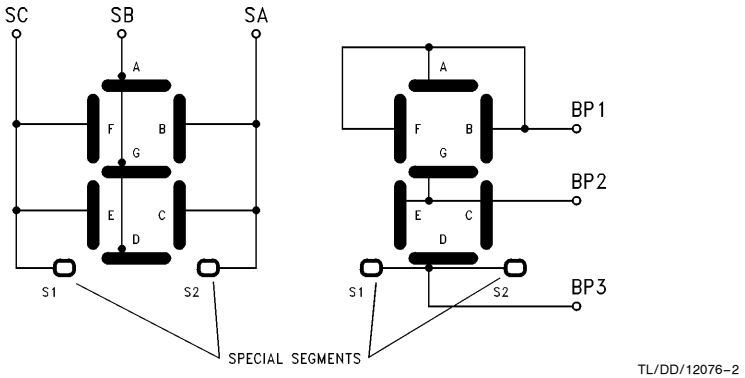


FIGURE 2. Example: Backplane-Segment Arrangement

A typical configuration of a triplex LCD is a four digit display with 8 special segments (thus having a total of 36 segments). Fifteen outputs of the COP8 are needed; **4 × 3 segment** pins and **3 backplane** pins.

Common to all LCD's is that the voltage across backplane(s) and segment(s) has to be an AC-voltage. This is to avoid electrochemical degradation of the liquid crystal layer. A segment being ''off'' or ''on'' depends on the **r.m.s.** voltage across a segment.

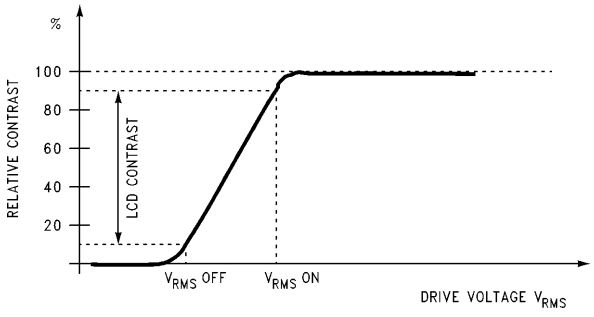
The maximum attainable ratio of ''on'' to ''off'' r.m.s. voltage (discrimination) is determined by the multiplex ratio. It is given by:

$(V_{ON}/V_{OFF})\max = SQR((SQR(N) + 1)/(SQR(N) - 1))$

N is the multiplex ratio.

The maximum discrimination of a 3 way multiplexed LCD is 1.93, however, it is also possible to order a customized display with a smaller ratio. With the approach used in this application note, it may not be possible to acheive the optimum contrast acheived with a standard 3 way muxed driver. As a result of decreased discrimination (1.93 to 1.73) the user may have to live with a tighter viewing angle and a tighter temperature range.

In this application you get a **VrmsOFF** voltage of 0.408*Vop and a **VrmsON** voltage of 0.707*Vop. Vop is the operating voltage of the LCD. Typical Vop values range from 3V–5V. With the optoelectrical curve of the LCD you can evaluate the maximum contrast of the LCD by calculating the difference between the relative ''OFF'' contrast and the relative ''ON'' contrast.



In this example:

VrmsON = 0.707*Vop

VrmsOFF = 0.408*Vop

FIGURE 3. Example Curve: Contrast vs r.m.s. Drive Voltage

The backplane signals are generated with the voltage steps **0V, Vop/2** and **Vop** at the backplanes; also see *Figure 4*.

Two resistors are necessary for each backplane to establish all these levels.

The backplane connection scheme is shown in *Figure 1*.

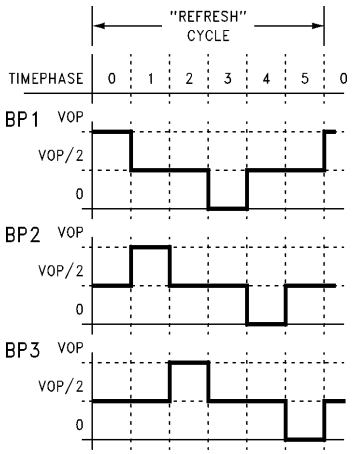
The Vop/2 level is generated by switching the appropriate COP's port pin to Hi-Z.

The following timing considerations show a simple way how to establish a discrimination ratio of 1,732.

TIMING CONSIDERATIONS

A Refresh cycle is subdivided in 6 timephases. *Figure 4* shows the timing for the backplanes during the equal distant timephases 0 . . . 5.

Backplane Control



TL/DD/12076–4

Note: After timephase 5 is over the backplane control timing starts with timephase 0 again.

FIGURE 4. Backplane Timing

While the backplane control timing continuously repeats after 6 timephases, the segment control depends on the combination of segments just being activated.

TABLE I. Possible Segment ON/OFF Variations

Tiphtab Address	Segment A	Segment B	Segment C
0	off	off	off
1	on	off	off
2	off	on	off
3	on	on	off
4	off	off	on
5	on	off	on
6	off	on	on
7	on	on	on

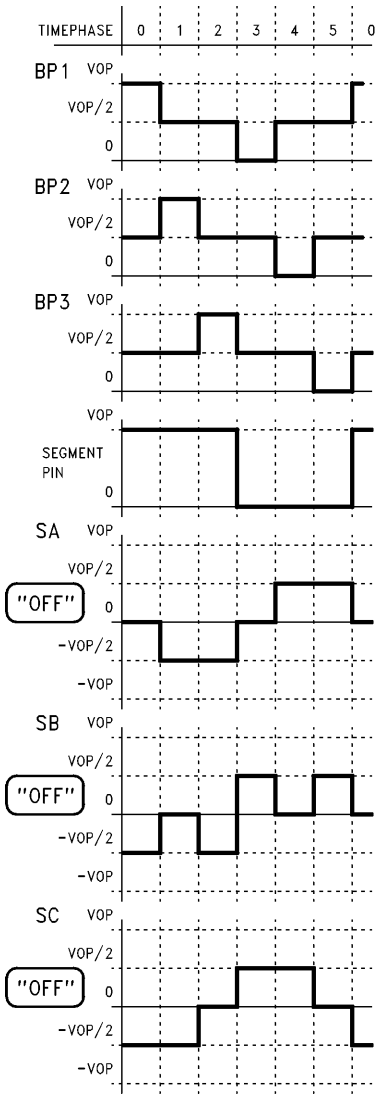
Figure 5 through *Figure 12* below show all possible combinations of controlling a **"Segment Triple"** with help of the 3 backplane connections and one segment pin. The segment switching has to be done according to the ON/OFF combination required (see also Table I).

Each figure shows in the first 3 graphs the constant backplane timing.

The 4th graph from the top shows the segment control timing necessary to switch the 3 segments (SA/SB/SC), activated from one pin, in the eight possible ways.

The 3 lower graphs show the resulting r.m.s. voltages across the 3 segments (SA, SB, SC).

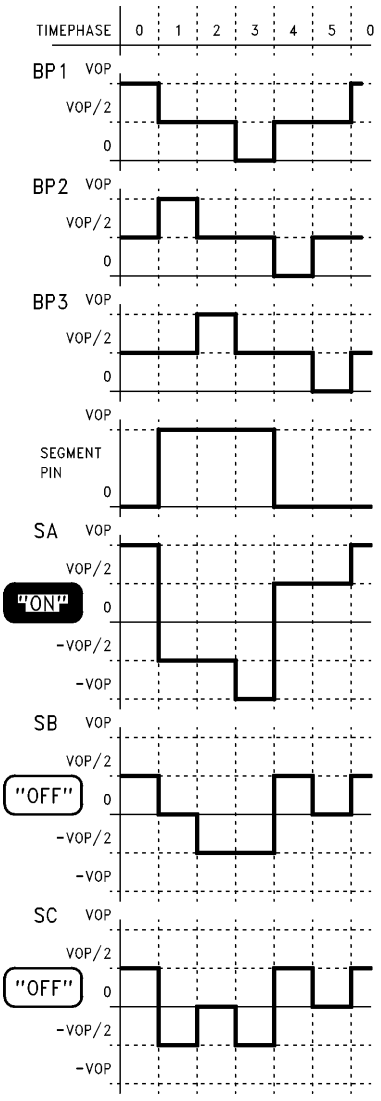
Segment/Backplane Control-Timing



TL/DD/12076–5

tiphtab address = 0

FIGURE 5

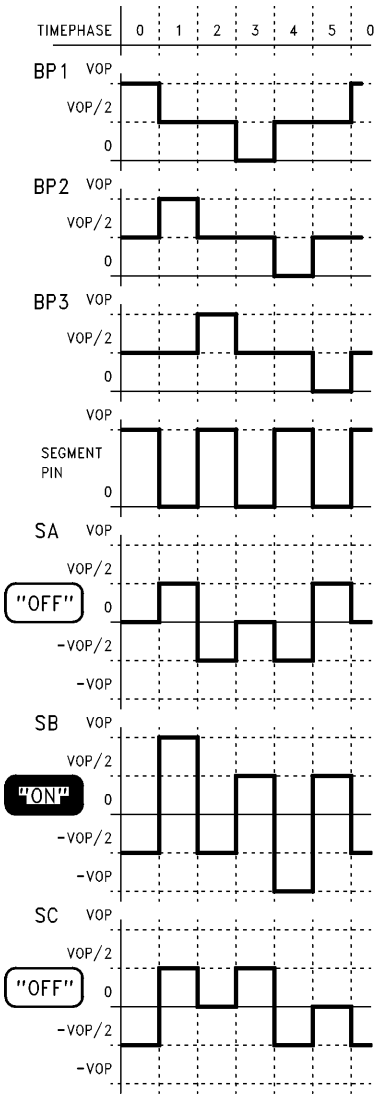


TL/DD/12076–6

tiphtab address = 1

FIGURE 6

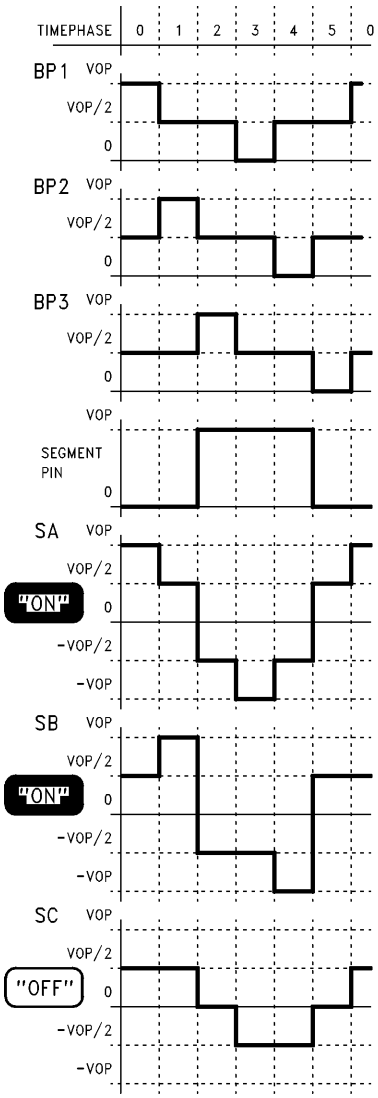
Segment/Backplane Control-Timing



TL/DD/12076–7

tiphtab address = 2

FIGURE 7

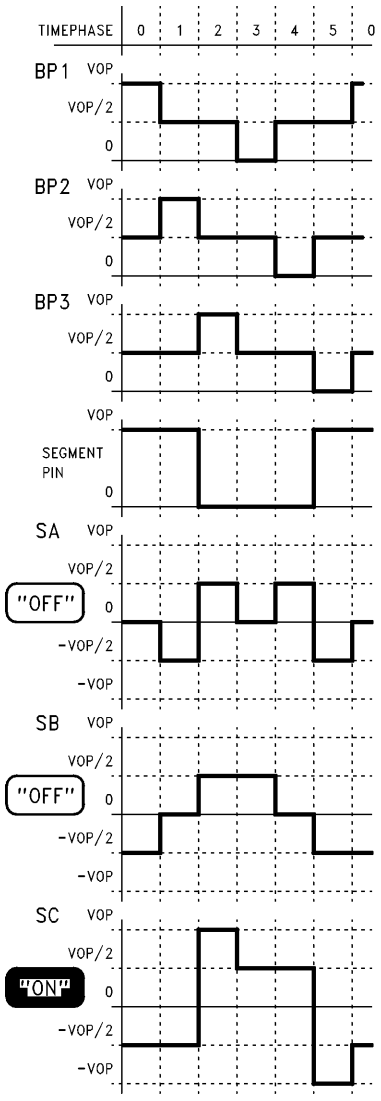


TL/DD/12076–8

tiphtab address = 3

FIGURE 8

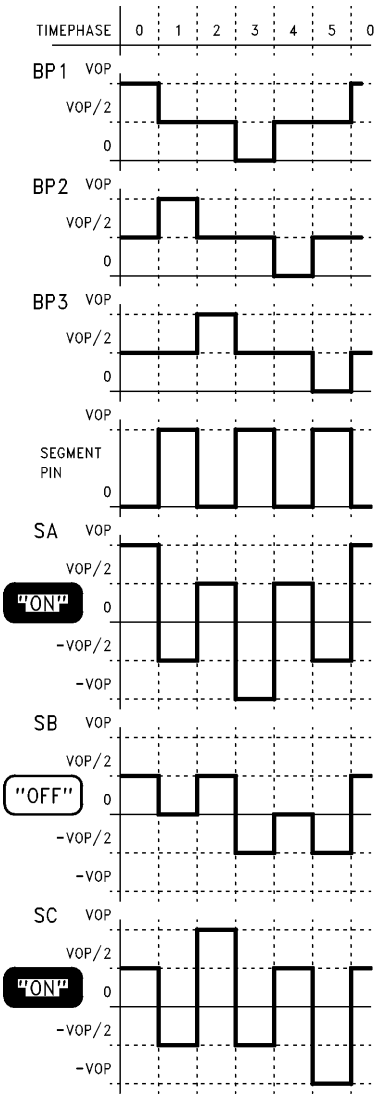
Segment/Backplane Control-Timing



TL/DD/12076–9

tiphtab address = 4

FIGURE 9

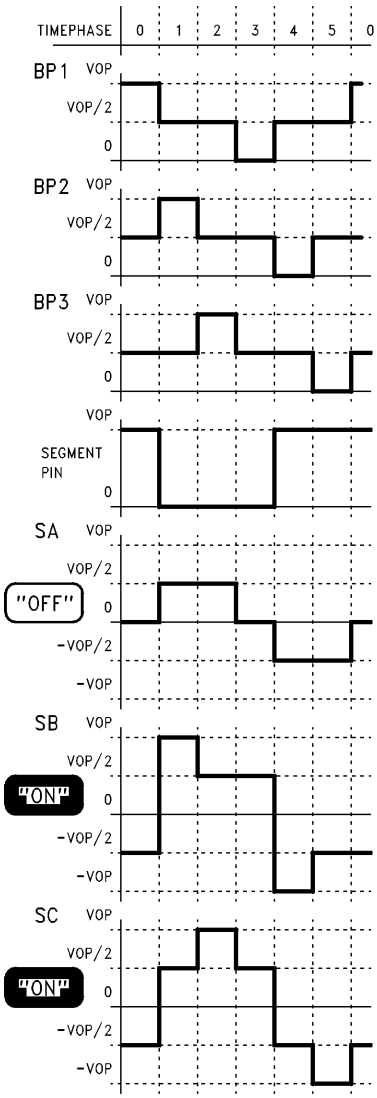


TL/DD/12076–10

tiphtab address = 5

FIGURE 10

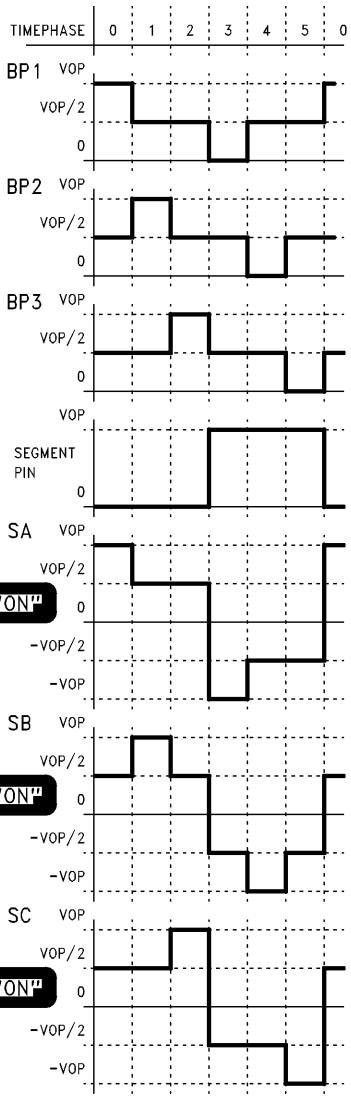
Segment/Backplane Control-Timing



TL/DD/12076–11

tiphtab address = 6

FIGURE 11



TL/DD/12076–12

tiphtab address = 7

FIGURE 12

REFRESH FREQUENCY

One period with six timephases is called a **refresh cycle** (also see *Figure 4*).

The refresh cycle should be in a frequency range of 30 . . . 60 Hz. A frequency below 30 Hz will cause a flickering display. On the other hand, current consumption increases with the LCD's frequency. So it is also recommended to choose a frequency below 60 Hz.

In order to periodically update the μC's port pins (involved in backplane or segment control) at the beginning of a new timephase, the COP8 needs a timebase of typ. 4 ms which is realized with an external RC-circuit at the G0/INT pin.

The G0 pin is programmable as input (Schmitt Trigger). The conditions for the external interrupt could be set for a low to high transition on the G0 pin setting the IPND-flag (external interrupt pending flag) upon an occurrence of such a transition. The external capacitor can be discharged, with the G0 pin configured as Push/Pull output and programmed to ''0''. When, switching G0 as input the Cap. will be charged through the resistor, until the threshold voltage of the Schmitt-Trigger input is reached. This triggers the external interrupt. The first thing the interrupt service routine has to do is to discharge the capacitor and switch G0 as input to restart the procedure.

This timing method has the advantage, that the timer of the device is free for other tasks (for example to do an A/D conversion).

The time interval between two interrupts depends on the RC circuit and the threshold of the G0 Schmitt Trigger V_{TH}.

The refresh frequency is independent of the clock frequency provided to the COPs device.

The variations of ''threshold'' levels relative to V_{CC} (over process) are as follows:

$$(V_{TH}/V_{CC})\ \text{min} = 0.376$$

$$(V_{TH}/V_{CC})\ \text{max} = 0.572$$

at $V_{CC} = 5V$

Charge Time:

$$T = -(\ln(1-V_{TH}/V_{CC})*RC)$$

To prevent a flickering display one should aim at a minimum refresh frequency of $f_{refr} = 30$ Hz. This means an interrupt frequency of $f_{int} = 6 \times 30$ Hz $= 180$ Hz. So, the maximum charge up time T_{max} must not exceed 5.5 ms ($T_{min} = 2.78$ ms).

With the formula:

$$RC_{max} = T_{max}/(-\ln(1-(V_{TH}/V_{CC})max)) = 5.5\ \text{ms} \times 0.849$$

$$RC_{max} = 6.48\ \text{ms}$$

$$(RC_{min} = 5.98\ \text{ms})$$

The maximum RC time-constant is calculated. The minimum RC time constant can be calculated similarly.

A capacitor in the nF-range should be used (e.g. 68 nF), because a bigger one needs too much time to discharge. To discharge a 68 nF Cap., the G0 pin of the device has to be low for about 40 μs.

On the other hand the capacitor should be large enough to reduce noise susceptibility.

When the RC combination is chosen, one can calculate the maximum refresh frequency by using the minimum values of the RC constant and the minimum threshold voltage:

$$T_{min} = RC_{min}*(-\ln(1-(V_{TH}/V_{CC})min = RC_{min}*0.472$$

and

$$f_{refr,max} = f_{int,max}/6 = 1/(T_{min}*6)$$

In the above example one timephase would be minimum 2.82 ms long. This means that about 250 instructions could be executed during this time.

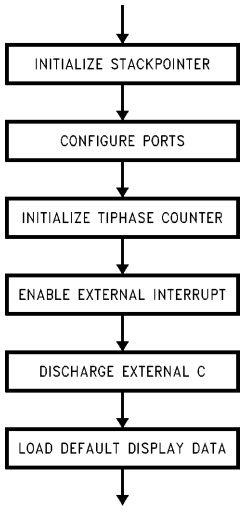
SOFTWARE

The software for the triplex LCD drive-demo is composed of three parts:

1. The initialization routine is executed only once after resetting the device, as part of the general initialization routine of the main program. The function of this routine is to **configure the ports, set the timephase counter (tiphase) to zero, discharge the external capacitor and enable the external interrupt.**

The initialization routine needs 37 bytes ROM.

Figure 13 shows the flowchart of this routine.



TL/DD/12076–13

FIGURE 13. Flowchart for Initialization Routine

2. The update routine calculates the port-data for each timephase according to the BCD codes in the RAM locations **'digit1' . . . 'digit4'** and the **special segments**. This routine is only called if the display image changes.

The routine converts the BCD code to a list **1st**, which is used by the refresh routine. *Figure 14* gives an overview and illustrates the data flow in this routine.

In *Figure 15* the data flow chart is filled with example data according to the display image in *Figure 16*.

First the routine creates the **seg1st** (4 bytes long), which contains the ''on/off'' configuration of each segment of the display. The display has 36 segments but the 4 bytes have only 32 bits, so the four special segments **S1** are stored in the **specbuf** location. The **bcdsegtab** table (in ROM) contains the LOOK-UP data for all possible Hex numbers from **0 to F**.

The routine takes three bits at the beginning of each time-phase from the **seg1st**.

These 3 bits address the 8 bytes of the **tiphtab** table in ROM. Each byte of this table contains the **time curve** for a segment pin (only 6 bits out of 8 are used).Using this information, the program creates the lists **for port D and port L (pod1st, pol1st)**. Every byte of this list contains the **timing representatives** for the pins D0−D3 and L0−L7, to allow an easy handling of the refresh routine.

The external interrupt has to be disabled while the **copy** routine is working, because the mixed data of two different display images would result in improper data on the display. *Figure 17* shows the flowchart of the **update** routine. The Flowchart of the **convert** subroutine is shown in *Figure 18*.

MEMORY REQUIREMENTS

ROM: 152 bytes incl. look up tables

RAM: 43 bytes (*Figure 15* illustrates the RAM locations)

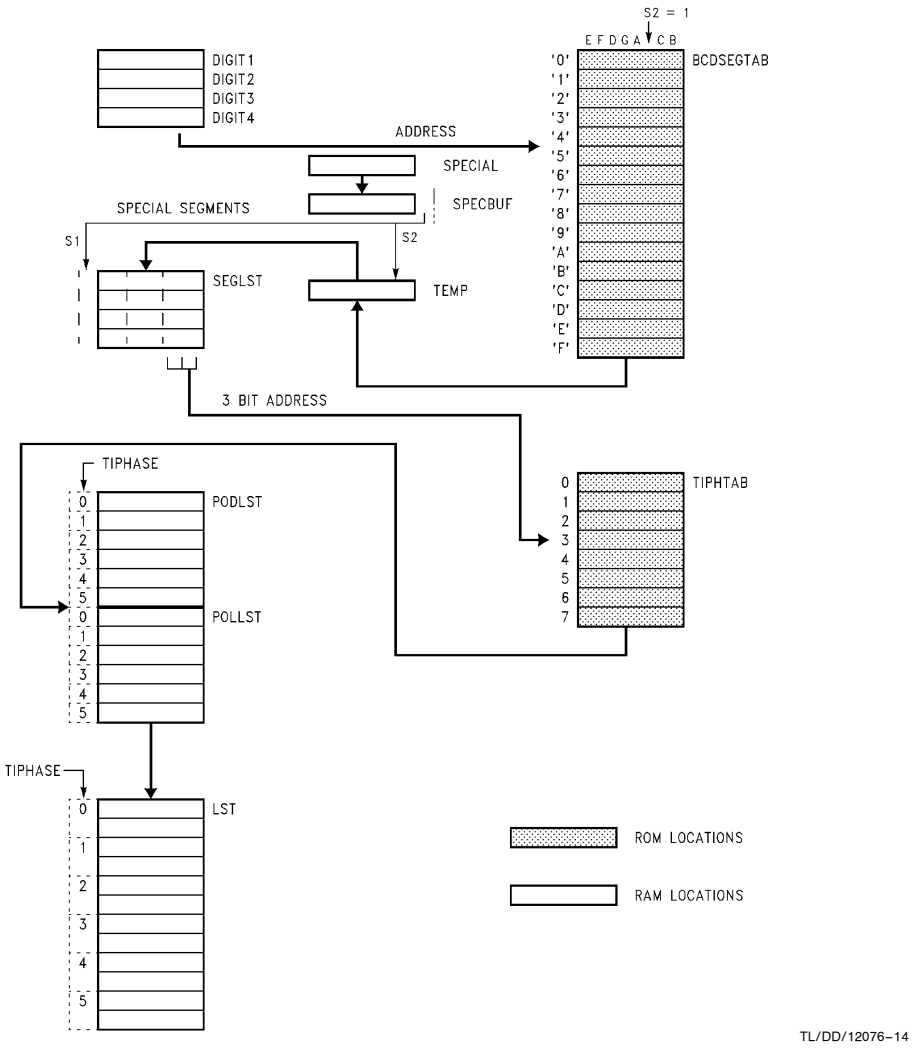


FIGURE 14. Data Flow Chart for Update Routine

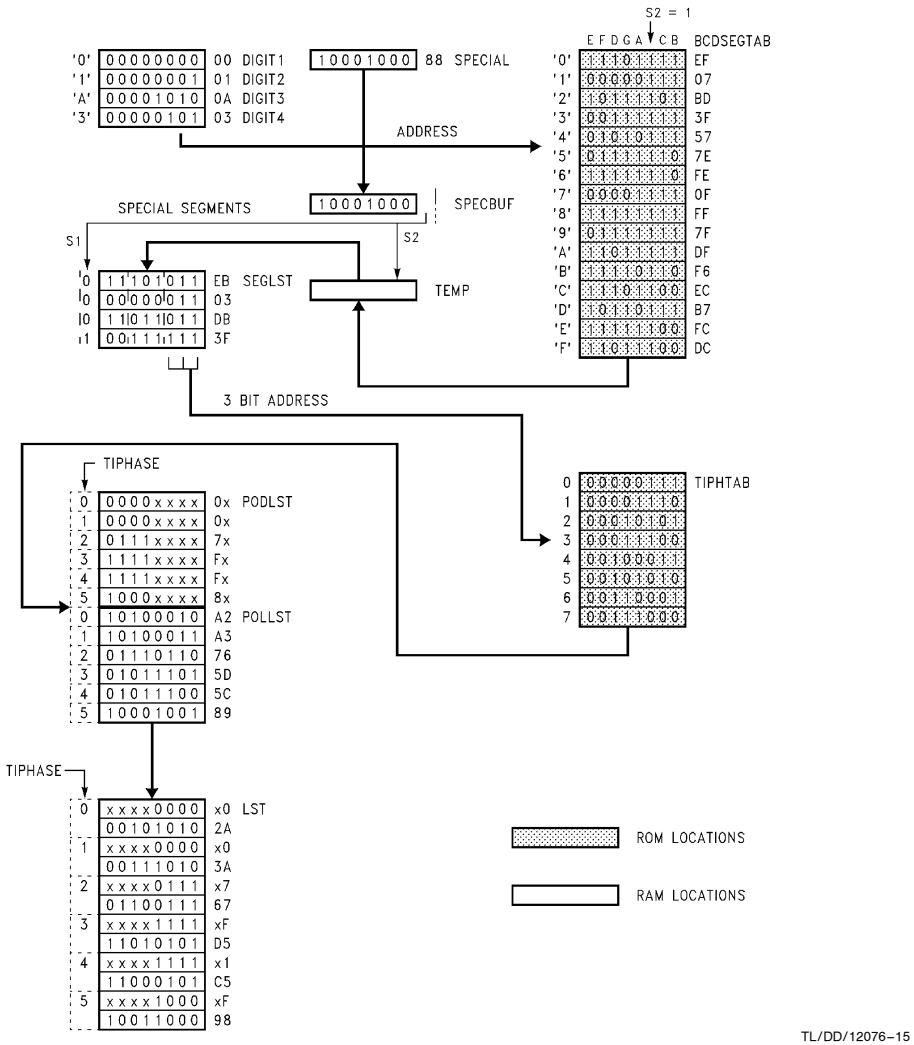
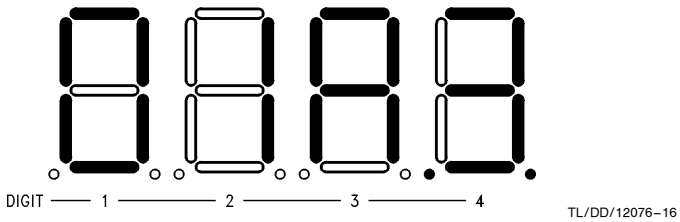


FIGURE 15. Data Flow Chart for Update Routine



TL/DD/12076–16

FIGURE 16. Display Example

3. The refresh routine is the interrupt service routine of the external interrupt and is invoked at the beginning of a new timephase. First the routine discharges the external capacitor and switches the G0/INT pin back to the input mode, to initialize the next timephase. The backplane ports G2, G4 and G5 and the segment pin ports D and L are updated by this routine according to the actual timephase. For the backplanes the data are loaded from the **bptab** table in ROM.

Table II shows how the **bptab** values are gathered. *Figure 20* shows the flowchart for the refresh routine.

TIME REQUIREMENTS

The routine runs max. 150 cycles.

For a non flickering display, the refresh frequency must be 30 Hz minimum. One refresh cycle has six timephases and is max. 33 ms long. So each timephase is 5.5 ms long. With an oscillator (CKI) frequency of 2 MHz, one instruction cycle takes 1/(2 MHz/10) = 5 μs to execute. During one timephase the controller can execute:

5.5 ms/5 μs = 1100 cycles. So the refresh routine needs 134/1100 = 0.122 = 12.2% of the whole processing time (in this case).

With a refresh frequency of 50 Hz the routine needs about 20.1% of the whole processing time.

The refresh routine needs about **103** ROM bytes.

TABLE II. Phase Values

Tiphase	G5	G4	G2	Portg Data	Hex	Portg Config.	Hex
0	0/0	0/0	1/1	XX00X1XX	04	XX00X1XX	04
1	0/0	1/1	0/0	XX01X0XX	10	XX01X0XX	10
2	1/1	0/0	0/0	XX10X0XX	20	XX10X0XX	20
3	0/0	0/0	0/1	XX00X0XX	00	XX00X1XX	04
4	0/0	0/1	0/0	XX00X0XX	00	XX01X0XX	10
5	0/1	0/0	0/0	XX00X0XX	00	XX10X0XX	20

data/configuration register of portg

0/0 : Hi-Z input
0/1 : output low
1/1 : output high

SUMMARY OF IMPORTANT DATA

LCD type:	3 way multiplexed
Amount of segments:	36
$V_{OP} = (V_{CC})$ (range):	2.5V to 6V
Oscillator frequency:	2 MHz (typ.)
Instruction cycle time:	5 μs
ROM requirements:	
init routine:	37 bytes
update routine:	152 bytes
refresh routine:	103 bytes
total:	292 bytes
RAM requirements:	
permanent use:	25 bytes
temporary use:	18 bytes
stack:	6 bytes
total:	49 bytes
	(also see *Figure 19*)
Timer:	not used
External interrupt:	with RC circuit used as time-base generator
Ports D, L:	used for LCD control
Port G:	3 G-pins are still free for other purposes +
Port I:	can be used as key-inp.

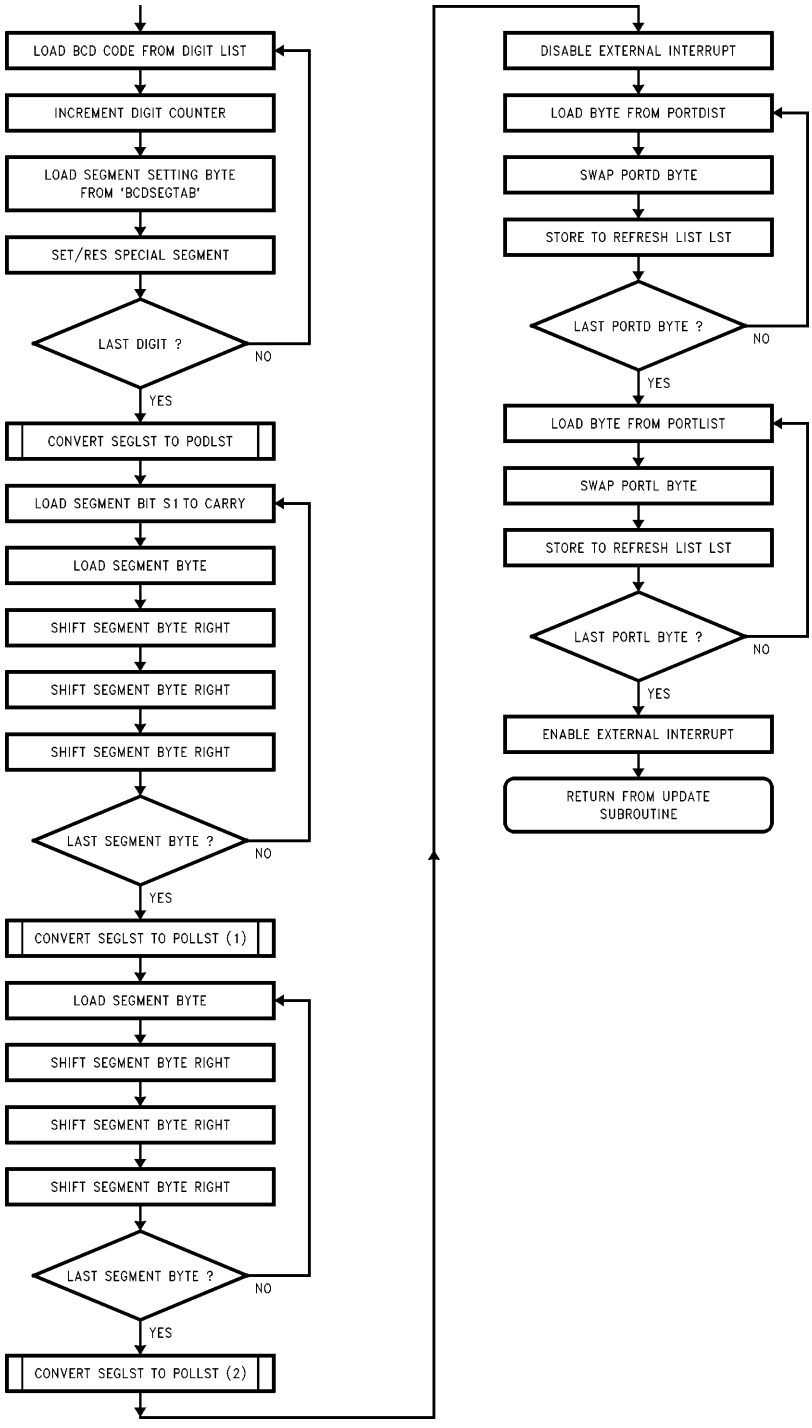
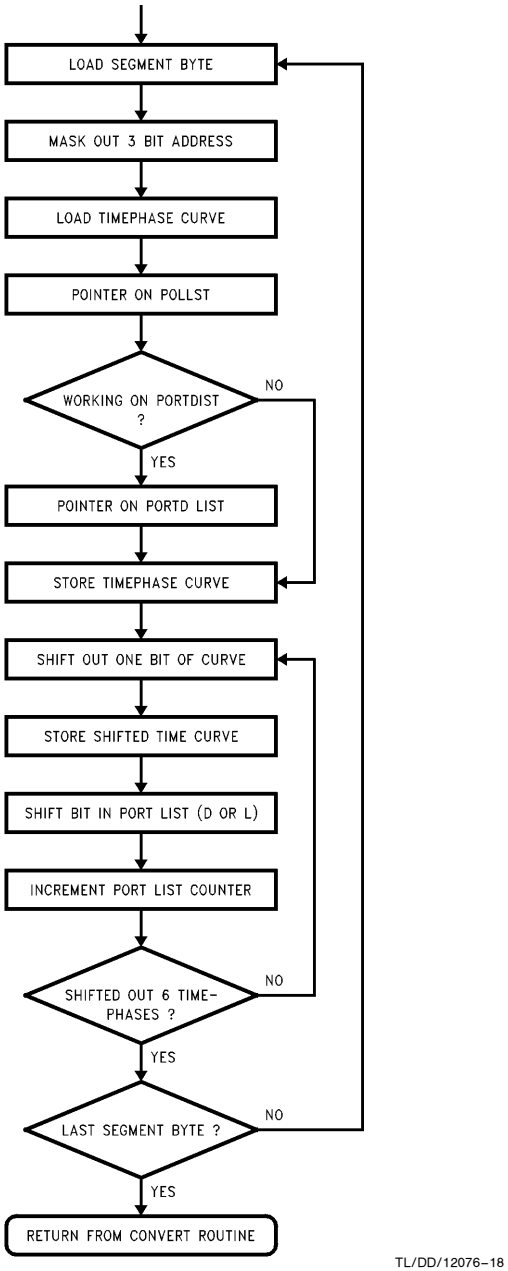


FIGURE 17. Flowchart for Update Routine



TL/DD/12076–18

FIGURE 18. Flowchart for Convert Subroutine

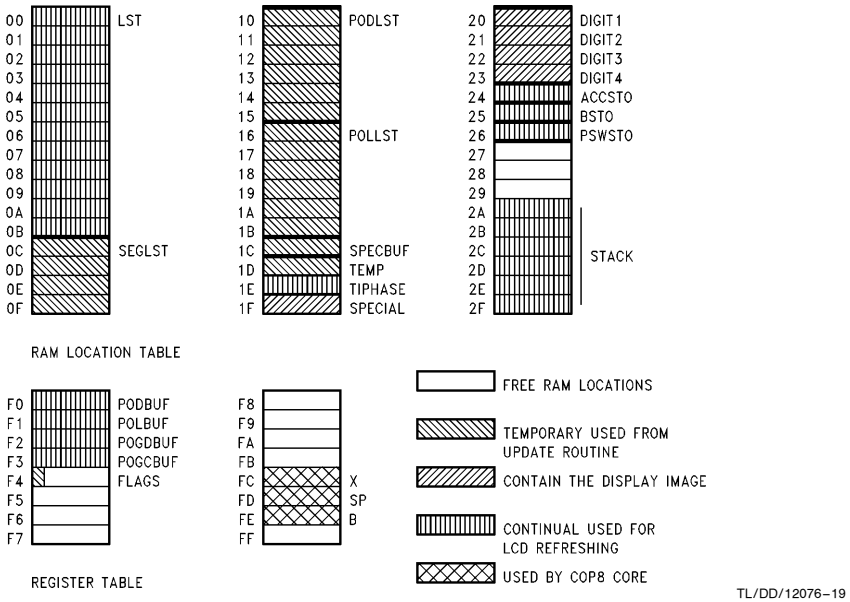
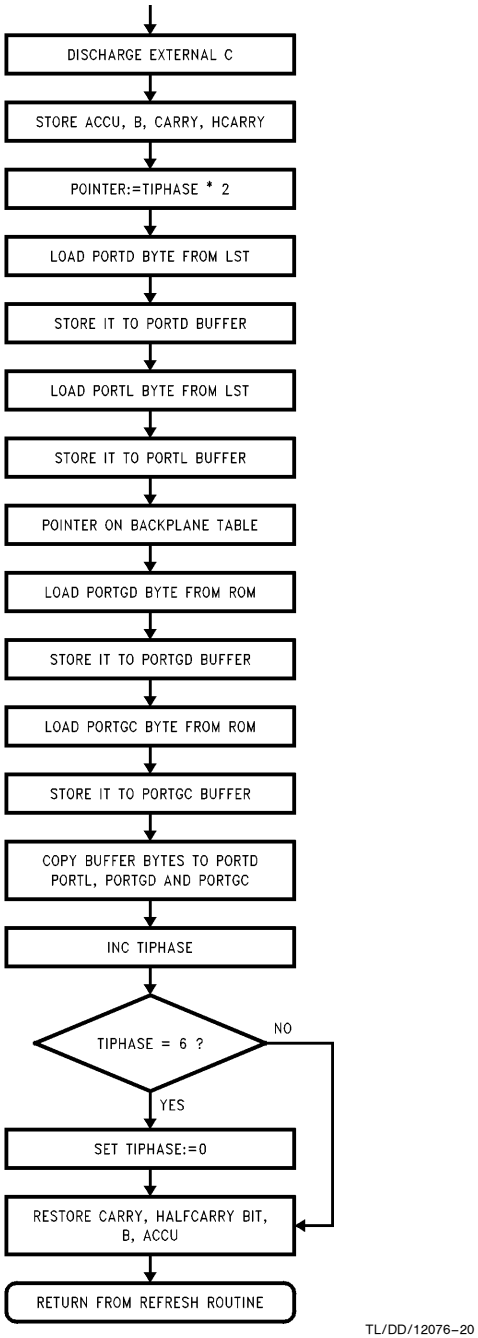


FIGURE 19. RAM Assignment



TL/DD/12076–20

FIGURE 20. Flowchart for Refresh-Routine

Listing

```
;  DEMO FOR COP820CJ:
;  3 WAY MULTIPLEXED LCD DRIVER DEMO
;  CONSTANT DISPLAY "01A3" and two special segments on

                .incld cop820cj.inc

;RAM assignments

                tiphase=01E
                special=01F             ;this byte must contain the
                                        ;on/off configuration of
                                        ;the extra segments
                                        ;('-','low bat',etc.)

                digit1=020              ;in these RAM locations the
                digit2=021              ;BCD code of the display
                digit3=022              ;digits are stored.
                digit4=023              ;

                accsto=024              ;accu buffer used during
                                        ;interrupt service routine
                bsto=025                ;b buffer
                pswsto=026              ;psw buffer

;register definition:

                podbuf=0f0              ;portd buffer
                polbuf=0f1              ;portl buffer
                pogdbuf=0f2             ;portgd buffer
                pogcbuf=0f3             ;portgc buffer
                flags=0f4               ;flag byte for podfla

;flag definition in flags byte

                podfla=07

;************** initialization routine ******************************

init:

                ld sp,#02f              ;initialize stackpointer

                ld portlc,#0ff          ;port l output
                ld portgc,#037          ;port g:G1,G2,G4,G5  are
                                        ;outputs
                ld portgd,#00           ;all outputs low, all
                                        ;inputs Hi-Z
                                        ;C at G0 is discharged
                ld tiphase,#00          ;begin with timephase 0
                ld psw,#002             ;ext. interrupt enable
```

TL/DD/12076–21

```
begin:          sbit #gie,psw           ;interrupts are welcome now
                rbit #00,portgc         ;now the external C can be
                                        ;charged
                ld b,#special
                ld [b+],#088            ;two special segments
                                        ;are 'ON'

                                        ;display:"01A3"
                ld [b+],#00             ;digit1
                ld [b+],#001            ;digit2
                ld [b+],#00A            ;digit3
                ld [b],#003             ;digit4

;************** main program ************************************
loop:
                jsr update
                jp loop

;************** update subroutine *******************************

;RAM definitions:

                specbuf=01C             ;buffer for 'special'
                temp=01D                ;temporary used

;pointer on tables:

                podlst=010              ;adress of list for port d
                pollst=016              ;adress of list for port l
                lst   =000              ;main list for display
                                        ;routine to refresh
                                        ;port d,l each timephase

                seglst=00C              ;this list contains the
                                        ;on/off configuration of
                                        ;the segments

                .=0200
                .local
update:
                ld a,special            ;load 'special' register
                x a,specbuf             ;to the buffer 'specbuf'
                ld x,#seglst            ;x points the segmentlist
                ld b,#digit1            ;b points digitlist

nxtdig:         ld a,[b+]               ;load BCD code of
                                        ;current digit
                add a,#L(bcdsegtab)     ;set pointer on look up
                                        ;table for segment setting
                laid                    ;load segment data of
                                        ;current digit
                x a,temp                ;store it to RAM
                ld a,specbuf            ;load special bit
                rrc a                   ;to carry
```

TL/DD/12076–22

```
                x a,specbuf             ;prepare for next
                                        ;special segment
                ifnc                    ;special bit not set ?
                rbit #2,temp            ;then reset it in the
                                        ;temp byte
                ld a,temp               ;store temp
                x a,[x+]                ;to the seglst list
                ifbne #04               ;if not last digit
                jp nxtdig               ;load data for next digit

                sbit #podfla,flags      ;set flag for working at
                                        ;port d list
                jsr convert             ;convert 3 bits from the
                                        ;segment bytes to the
                                        ;timephaselist for portd


;shift with carry

shwc:
                ld b,#seglst            ;b points seglst
nxtshwc:        ld a,specbuf            ;load special segment bit
                rrc a                   ;to carry
                x a,specbuf             ;prepare for next
                                        ;special segment
                ld a,[b]                ;shift the segmentbyte
                rrc a                   ;three positions right
                rrc  a                  ;and append the special
                                        ;segment bit
                rrc  a                  ;
                x a,[b+]                ;store shifted byte
                ifbne #00               ;end of segment list
                                        ;not reached ?
                jp nxtshwc              ;then shift the next
                                        ;segment byte

                rbit #podfla,flags      ;reset flag for working
                                        ;at port 1 list
                jsr convert             ;convert 3 bits of the
                                        ;segment bytes to the
                                        ;timephaselist for port 1
;shift (without carry)

shift:          ld b,#seglst            ;b points segmnet list
nxtshift:       ld a,[b]                ;load segment byte
                rrc a                   ;shift the segmentbyte
                rrc a                   ;three positions right
                rrc a                   ;
                x a,[b+]                ;store shifted byte
                ifbne #00               ;end of segment list
                                        ;not reached ?
                jp nxtshift             ;then shift the next
                                        ;segment byte
```

TL/DD/12076–23

```
                jsr convert             ;convert 3 bits of the
                                        ;segment bytes to the
                                        ;timephaselist for port 1

;copy portdata to the list on which the refresh routine will access

copy:
                rbit #eni,psw           ;disable interrupt to
                                        ;prevent fail display
                ld b,#podlst            ;b points podlst
                ld x,#lst               ;x points refresh list
nxtd:           ld a,[b+]               ;load portbyte
                swap a                  ;swap it
                x a,[x+]                ;store it to refresh list
                ld a,[x+]               ;increment x
                ifbne #06               ;if the end of the podlst
                                        ;is not reached
                jp nxtd                 ;then next timephase
                ld b,#pollst            ;b points pollst
                ld x,#lst               ;x points refresh list
nxtl:           ld a,[x+]               ;increment x
                ld a,[b+]               ;load portbyte
                swap a                  ;swap it
                x a,[x+]                ;store it to refresh list
                ifbne #0C               ;if the end of the pollst
                                        ;is not reached
                jp nxtl                 ;then next timephase
                sbit #eni,psw           ;refresh routine allowed
                                        ;again

                ret                     ;end of update routine

;subroutines for update routine:

convert:
                ld x,#seglst            ;x points segment list
nxtsgl:         ld a,[x+]               ;load segment byte
                and a,#007              ;mask out first three bits
                add a,#L(tiphtab)       ;pointer on timephase table
                laid                    ;load timephase curve for
                                        ;one segment pin
                ld b,#pollst            ;b points list for portd
                ifbit #podfla,flags     ;working at podlst ?
                ld b,#podlst            ;then b points on podlst

;shift timephase data according to 3 bits ( 8 combinations are
;possible with 3 segments)

tipsh:
                x a,temp                ;store timephase curve to
                                        ;temp buffer
nxtphsh:
                ld a,temp               ;load timephase curve again
                rrc a                   ;shift out one bit into
```

TL/DD/12076–24

```
                                        ;carry bit
                x a,temp                ;store shifted curve
                ld a,[b]                ;load portbyte
                rrc a                   ;shift in one bit from
                                        ;carry bit
                x a,[b+]                ;store shifted portbyte
                                        ;again
                ld a,#pollst            ;end of podlst ?
                ifeq a,b                ;
                jp eplst                ;then return
                ifbne #0C               ;else end of pollst
                jp nxtphsh              ;
eplst:
                ld a,#L(seglst+4)       ;if the end of the segment
                ifgt a,x                ;list is not reached
                jp nxtsg1               ;work at next segment byte
                ret


bcdsegtab:

;in this bytes are the on/off configuration of the segments
;for a digit are stored. there are only 7 bits of each byte
;the configuration of the 2 special segments is stored
;in the 'special' byte.

                .BYTE 0EF,007,0BD,03F   ;'0'...'3'
                .BYTE 057,07E,0FE,00F   ;'4'...'7'
                .BYTE 0FF,07F,0DF,0F6   ;'8'...'B'
                .BYTE 0EC,0B7,0FC,0DC   ;'C'...'F'
tiphtab:

;one pin controls 3 segments. there are 8 possible
;combinations. for each combination there is one byte.
;6 bits of one byte control the pin for each timephase.

                .BYTE 007,00E,015,01C,023,02A,031,038



;************** interrupt service routine ************************

                .=0ff
refresh:
                x a,accsto              ;store accu
                ld a,b                  ;store b
                x a,bsto                ;

                ld b,#portgd
                                        ;discharge C
                rbit #00,[b]
                ld a,[b+]               ;increment b (b=#portgc)
                sbit #00,[b]            ;by switching G0 to a
                                        ;low output
```

TL/DD/12076–25

```
rbit #00,[b]            ;C can be charged again
ld b,#psw
rbit #ipnd,[b]          ;reset ext. interrupt
                        ;pending flag

ld a,[b]                ;load psw
x a,pswsto              ;store psw

ld a,tiphase            ;accu:=tiphase*2
add a,tiphase           ;

x a,b                   ;store accu in b
ld a,[b+]               ;load portbyte from
                        ;refresh list('lst')
x a,podbuf              ;store it to port d buffer
ld a,[b+]               ;load portbyte
x a,polbuf              ;store it to port l buffer
ld a,b                  ;accu:=timephase*2+2
add a,#L(bptab)-2       ;accu points on
                        ;backplane table
x a,b                   ;store pointer
ld a,b                  ;
laid                    ;load port g data byte
x a,pogdbuf             ;store it to port g data
                        ;buffer
ld a,[b+]               ;increment b
ld a,b                  ;load pointer
laid                    ;load portg conf. byte
x a,pogcbuf             ;store it to buffer


ld b,#podbuf            ;b points buffer list
ld a,[b+]               ;
x a,portd               ;refresh port d
ld a,[b+]               ;
x a,portld              ;refresh port l

ld portgc,#00           ;all backplane wires on
                        ;Vop/2 level to prevent
                        ;spikes

ld a,[b+]               ;
x a,portgd              ;refresh port g data
ld a,[b+]               ;
x a,portgc              ;refresh port g config.

ld a,tiphase            ;update timephase counter
inc a                   ;
ifeq a,#06              ;tiphase = 0..5
ld a,#00                ;
x a,tiphase             ;
ld b,#pswsto
rc                      ;restore carry bit
ifbit #07,[b]           ;
```

TL/DD/12076–26

```
                sbit #07,psw
                ifbit #06,[b]           ;restore halfcarry bit
                sbit #06,psw            ;

                ld a,bsto               ;restore b
                x a,b                   ;
                ld a,accsto             ;restore accu


                reti                    ;return from lcd
                                        ;refresh routine
bptab:          .BYTE 004,004,010,010,020,020
                .BYTE 000,004,000,010,000,020

                .END
```

TL/DD/12076–27

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: 1(800) 272-9959
TWX: (910) 339-9240

National Semiconductor GmbH
Livry-Gargan-Str. 10
D-82256 Fürstenfeldbruck
Germany
Tel: (81-41) 35-0
Telex: 527649
Fax: (81-41) 35-1

National Semiconductor Japan Ltd.
Sumitomo Chemical
Engineering Center
Bldg. 7F
1-7-1, Nakase, Mihama-Ku
Chiba-City,
Ciba Prefecture 261
Tel: (043) 299-2300
Fax: (043) 299-2500

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductores Do Brazil Ltda.
Rue Deputado Lacorda Franco
120-3A
Sao Paulo-SP
Brazil 05418-000
Tel: (55-11) 212-5066
Telex: 391-1131931 NSBR BR
Fax: (55-11) 212-1181

National Semiconductor (Australia) Pty, Ltd.
Building 16
Business Park Drive
Monash Business Park
Nottinghill, Melbourne
Victoria 3168 Australia
Tel: (3) 558-9999
Fax: (3) 558-9998