Application Note

An Intelligent Phone-Line Auto-Switcher

AN000601-Z8X0400

ZiLOG Worldwide Headquarters • 910 E. Hamilton Avenue • Campbell, CA 95008
Telephone: 408.558.8500 • Fax: 408.558.8300 • www.ZiLOG.com

This publication is subject to replacement by a later edition. To determine whether a later edition exists, or to request copies of publications, contact:

ZiLOG Worldwide Headquarters
910 E. Hamilton Avenue
Campbell, CA 95008
Telephone: 408.558.8500
Fax: 408.558.8300
www.ZiLOG.com

Information Integrity

The information contained within this document has been verified according to the general principles of electrical and mechanical engineering. Any applicable source code illustrated in the document was either written by an authorized ZiLOG employee or licensed consultant. Permission to use these codes in any form, besides the intended application, must be approved through a license agreement between both parties. ZiLOG will not be responsible for any code(s) used beyond the intended application. Contact the local ZiLOG Sales Office to obtain necessary license agreements.

Document Disclaimer

Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZiLOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZiLOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. Except with the express written approval ZiLOG, use of information, devices, or technology as critical components of life support systems is not authorized. No licenses or other rights are conveyed, implicitly or otherwise, by this document under any intellectual property rights.

Table of Contents

List of Figures

List of Tables

Introduction

ZiLOG's Z8 CCP™ Emulator is a user-friendly and powerful 8-bit microcontroller. With the easy-to-use Z8 assembler and a low cost CCP™ Emulator, engineers can quickly develop many applications. For the intelligent Phone-Line Auto Switcher design, a Z86E30 is used as the main MCU. The source code emulates and debugs on a CCP emulator. The Z86E30 controls a serial E2PROM, a telephone line interface (DAA), a ringer generator, a DTMF decoder, an in-use line detector, and other circuit controls, making it ideal for use as an intelligent Phone-Line Auto-Switcher.

The Application

The Phone-Line Auto Switcher design incorporates many features similar to other AutoSwitchers in the market (see Reference 2 at the end of this application note). It features one CO line (Telephone Line In) and 3 internal line outputs (Phone 1 to Phone 3). The internal lines can be connected to either an answering machine, fax machine, modem, or regular telephone. When there is an incoming call, the Phone-Line Auto-Switcher answers, waits for the caller to enter digits, and then switches to one of the three internal outputs. The digits are generated by Dual Tone Multi Frequency (DTMF) signals and transmitted over the telephone line. If the digits match the numbers stored in the E2PROM, the Phone-Line Auto-Switcher switches the outside line to the proper internal device. If no digit is entered, it times out and switches to the default device.

Users can change the configuration setting remotely, using DTMF signals to enter the password and the configuration numbers. Software routines in this design can be reused for many other applications, such as PDA, SOHO, PBX and CTI.

Figure 1 on the following page illustrates the main control circuit. The Z86E30 Ports 20 to 23 are used to control E2PROM 93C46. Port 0.0 to Port 0.7 (8 I/O pins) are set to the output direction to control 74LS374, D-type Flip Flop, and to the input direction to read 4-bit data from the DTMF decoder (UM9204). Port 3.1 is used to detect Pulse Out, which determines the connection status between the phone line and the internal device. Port 3.2 is used for in-use detection, which determines whether or not the phone line is being used. After U3 (UM9204) detects a valid DTMF signal, it sets pin 12 to high, indicating that the DTMF signal received is valid. This data valid (DV) shares the same Z86E30 pin with ringer detection in Port 3.3—a good practical design method, because the MCU has limited I/O pins. Port 3.6 enables U6 and Port 3.7 enables U3.

Figure 1. Auto-Switcher Main Control Circuit

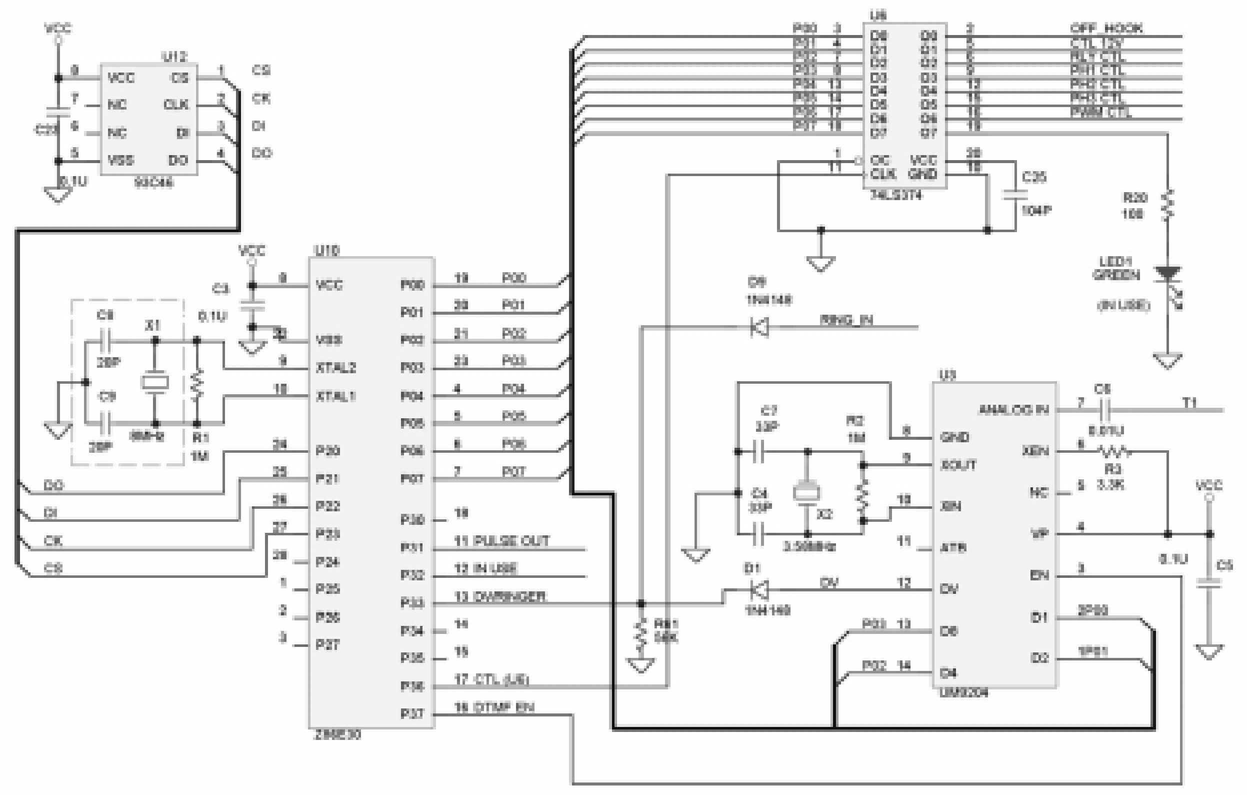


Figure 2 illustrates the ringer generation circuit. U9, LM324, converts the CTL 12V (digital, on/off) signal to a sine wave signal. U8 and associated circuits are used to generate high voltage (about +115V and –115V). However, other ringer generation circuits can be used for this application. The design note from Linear Technology (see Reference 1) demonstrates an example regarding telephone ring-tone generation.

Figure 2. Ringer Generation Circuit

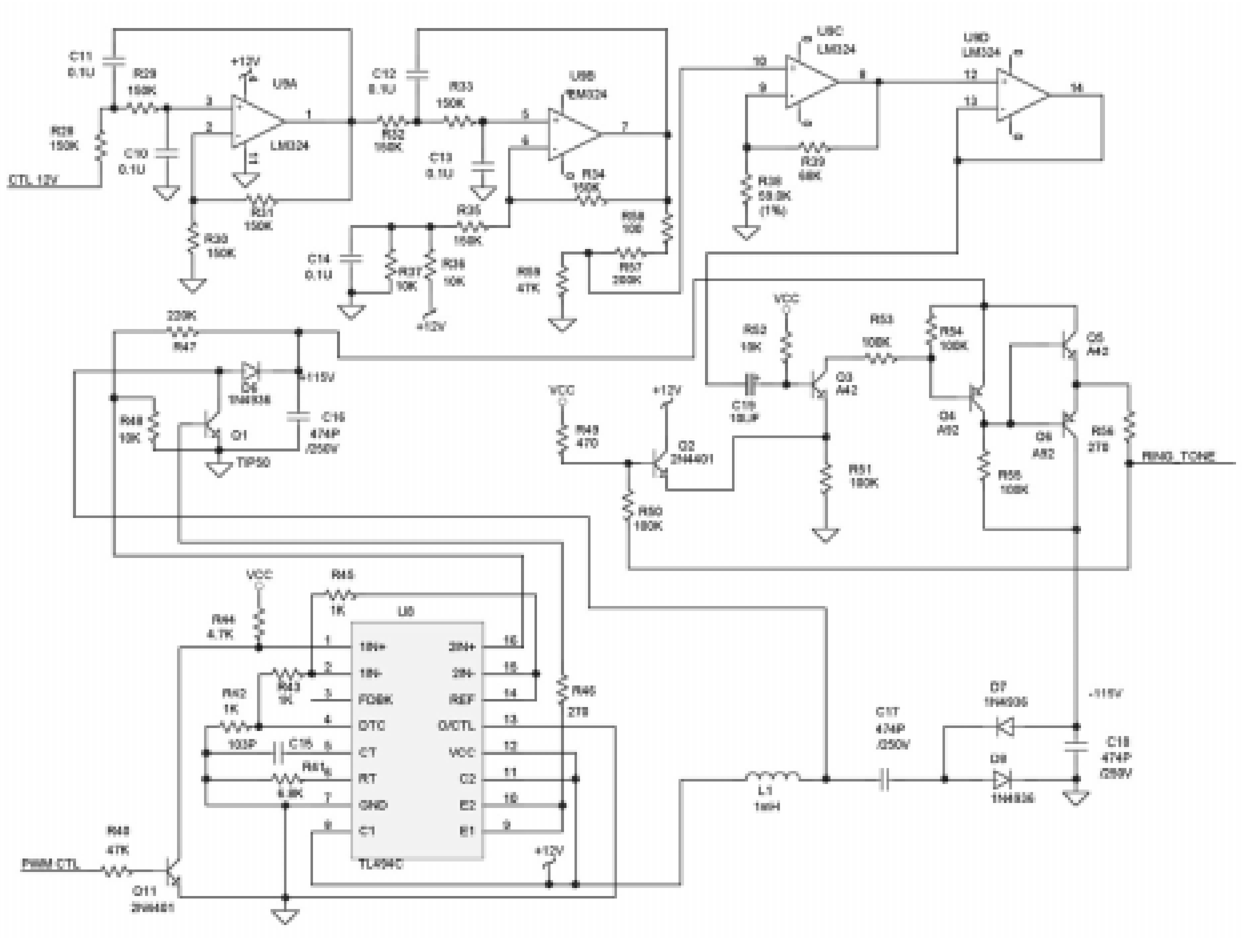


Figure 3 illustrates the line interface circuit.

Note: The U5 (Xecom XE0055S/T) is used to interface the telephone line. This highly modularized–and FCC approved–device is selected for this application. (See Reference 3.)

J1 is connected to an outside telephone company. J2, J3, and J4 are connected to internal telecommunication devices. RLY1, RLY2, and U7 are relay switches used to connect to the proper internal device controlled by RLY_CTL, PH1 CTL, PH2 CTL, and PH3 CTL from the Z86E30. Excepting for the power supply, all circuits in this hardware design are included in Figure 1, Figure 2, and Figure 3. By adding a simple power supply circuit and one (+15 DCV, 400 mA) AC adapter, the Phone-Line Auto-Switcher works very well.

Figure 3. Line Interface Circuit

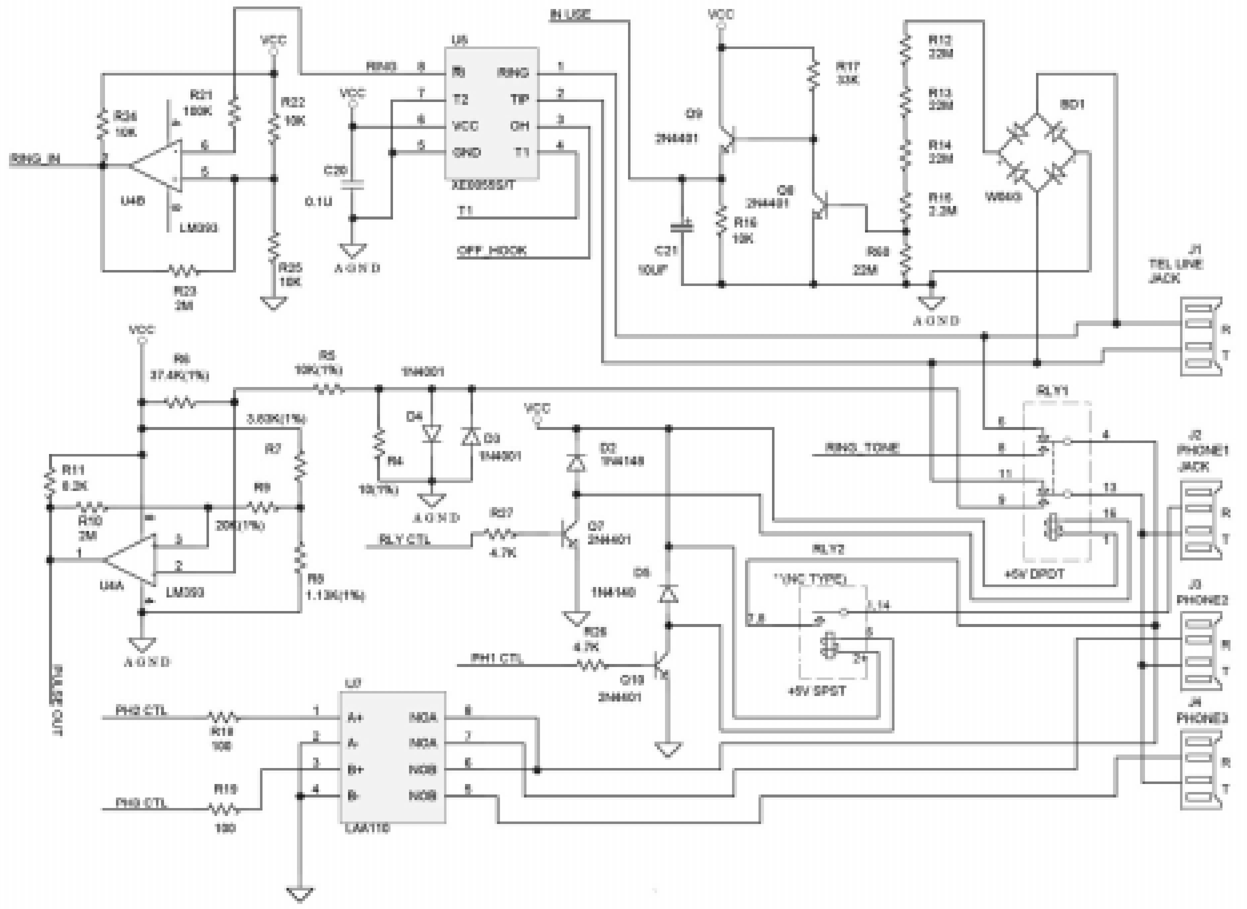


Figure 4 is a device block diagram indicating the system-level design of this application.

Figure 4. Device Block Diagram

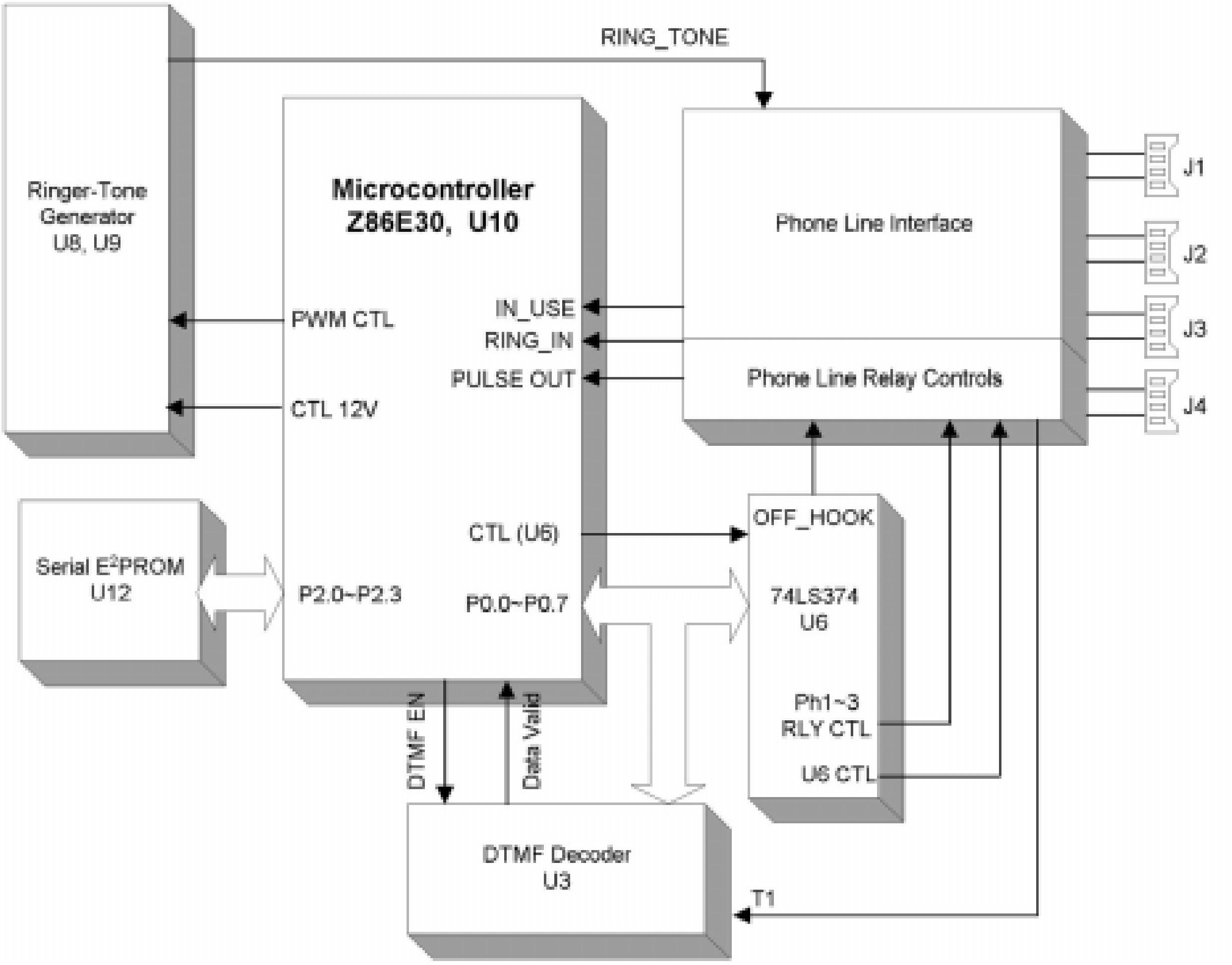


Table 1 lists a Bill of Materials (BOM) for parts reference.

Table 1. Bill Of Materials For A Phone-Line-Auto-Switcher

Item	Qty.	Reference	Part
1	1	BD1	W04G
2	2	C21,C19	10UF
3	9	C10,C11,C12,C13,C14 C3,C5,C23,C20	0.1U
4	1	C15	103P
5	3	C16,C17,C18	474P
6	2	C7,C4	33P
7	1	C6	0.01U
8	2	C8,C9	20P
9	1	C25	104P
10	4	D2,D5, D1,D9	1N4148
11	2	D3,D4	1N4001
12	3	D6,D7,D8	1N4936
13	1	J1	TEL LINE (RJ11)
14	1	J2	PHONE1 (RJ11)
15	1	J3	PHONE2 (RJ11)
16	1	J4	PHONE3 (RJ11)
17	1	LED1	GREEN
18	1	L1	1mH
19	6	Q7,Q8,Q9,Q10,Q11,Q2	2N4401
20	1	Q1	TIP50
21	2	Q3,Q5	A42
22	2	Q4,Q6	A92
23	1	RLY1	+5V DPDT
37	3	R27,R26,R44	4.7K
38	8	R28,R29,R30,R31,R32 R33,R34,R35	150K
39	4	R36,R37,R48,R52	10K
40	1	R38	59.0K
41	1	R39	68K
42	2	R59,R40	47K
43	1	R41	6.8K
44	3	R42,R43,R45	1K
45	2	R56,R46	270
46	1	R47	220K
47	1	R49	470
48	6	R50,R51,R53,R54,R55, R21	100K
49	1	R57	200K
50	4	R58,R20,R18,R19	100
51	2	R1,R2	1M
52	1	R3	3.3K
53	1	R61	56K
54	1	U4	LM393
55	1	U5	XE0055S/T
56	1	U7	LAA110
57	1	U8	TL494C
58	1	U9	LM324
59	1	U3	UM9204

Table 1. Bill Of Materials For A Phone-Line-Auto-Switcher

Item	Qty.	Reference	Part	Item	Qty.	Reference	Part
24	1	RLY2	+5V SPST	60	1	U6	74LS374
25	1	R4	10(1%)	61	1	U10	Z86E30
26	1	R5	10K(1%)	62	1	U12	93C46
27	1	R6	37.4K(1%)	63	1	X1	8MHz
28	1	R7	3.83K(1%)	64	1	X2	3.58MHz
29	1	R8	1.13K(1%)				
30	1	R9	20K(1%)				
31	2	R23,R10	2M				
32	1	R11	8.2K				
33	4	R12,R13,R14,R60	22M				
34	1	R15	2.2M				
35	4	R16,R22,R24,R25	10K				
36	1	R17	33K				

The assembly file, PLAS.S, listed elsewhere in this application note, is the complete Z8 source code. One easy way to compile and link the Z8 source file is to build a DOS batch file named Z8MAKE.BAT. Simply type Z8MAKE PLAS to compile and link the source code.

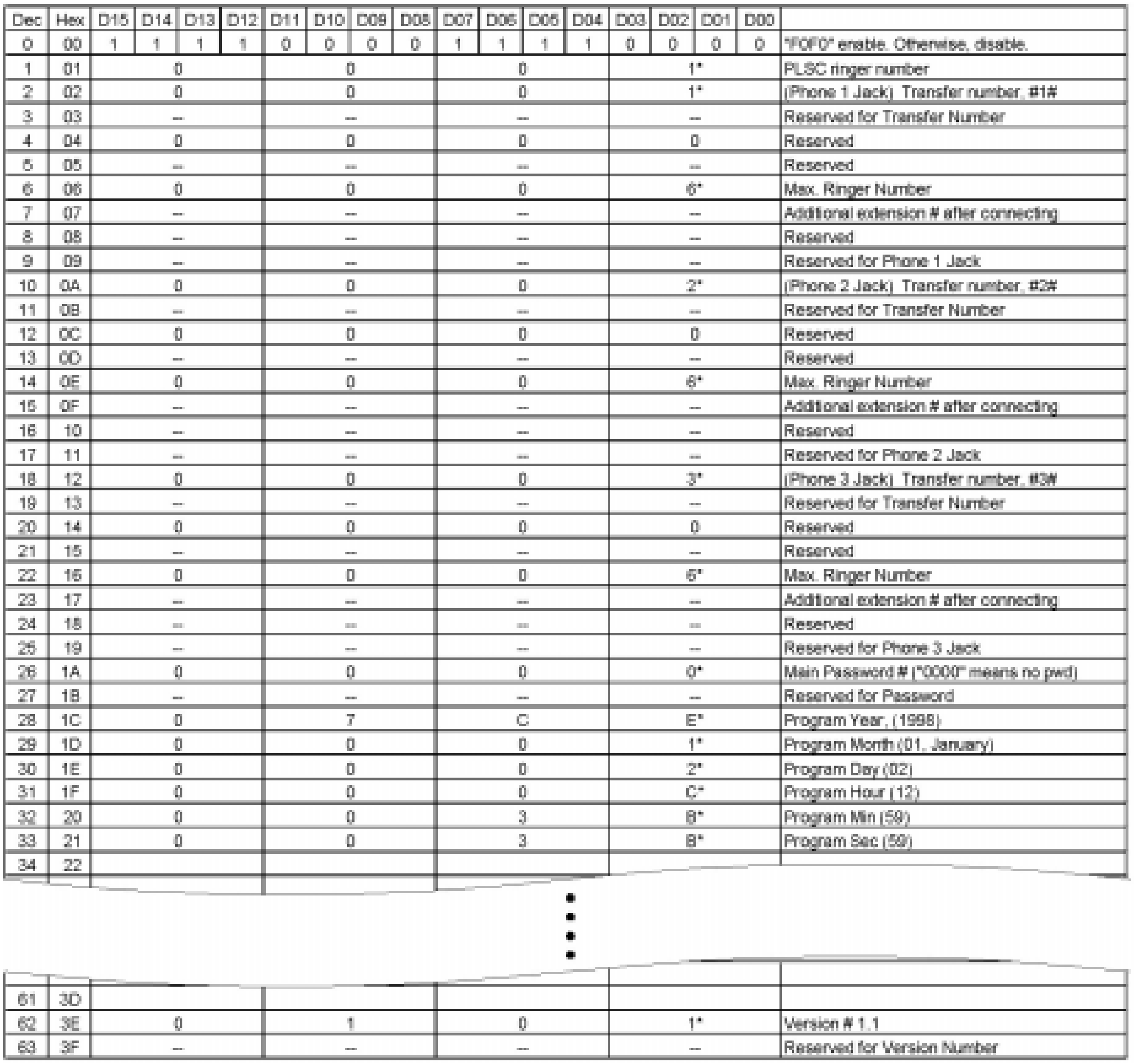
Following is the suggested batch file. The obj and hex files are generated after executing it. Either the obj or hex file (in this case PLAS.OBJ or PLAS.HEX) can be loaded into the Z8 CCP emulator for debugging and emulating functions.

```
Z8MAKE.BAT:
asms8 -s asms8 -o %1.o -rl -i %1.s
mlink -i %1.o -o %1.lnk
mload -i %1.lnk -o %1.hex
hexobj %1.hex %1.obj i
```

All configuration settings in this design are stored in the 93C46 E2PROM. In the initialization of the main loop, the program calls the routine Chk_93C46 to check whether or not the E2PROM was previously set to the configuration information. If not, the program formats the E2PROM and restores the default settings. The utility routines, Read93C46 and Write93C46, allow word data to be stored or retrieved from the 93C46.

Figure 5 shows the data structure used in this design. Similar data structures have been applied to many designs, such as saving network configuration, I/O address, IRQ number in the Ethernet card (NIC), and saving security (encryption) data in the Dongle as a hardware key for high-end CAD tools

Figure 5. 93C46 Map Data Structure



Dec	Hex	D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00	
0	00	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	"F0F0" enable. Otherwise, disable.
1	01	0				0				0				1*				PLSC ringer number
2	02	0				0				0				1*				(Phone 1 Jack) Transfer number, #1#
3	03	--				--				--				--				Reserved for Transfer Number
4	04	0				0				0				0				Reserved
5	05	--				--				--				--				Reserved
6	06	0				0				0				6*				Max. Ringer Number
7	07	--				--				--				--				Additional extension # after connecting
8	08	--				--				--				--				Reserved
9	09	--				--				--				--				Reserved for Phone 1 Jack
10	0A	0				0				0				2*				(Phone 2 Jack) Transfer number, #2#
11	0B	--				--				--				--				Reserved for Transfer Number
12	0C	0				0				0				0				Reserved
13	0D	--				--				--				--				Reserved
14	0E	0				0				0				6*				Max. Ringer Number
15	0F	--				--				--				--				Additional extension # after connecting
16	10	--				--				--				--				Reserved
17	11	--				--				--				--				Reserved for Phone 2 Jack
18	12	0				0				0				3*				(Phone 3 Jack) Transfer number, #3#
19	13	--				--				--				--				Reserved for Transfer Number
20	14	0				0				0				0				Reserved
21	15	--				--				--				--				Reserved
22	16	0				0				0				6*				Max. Ringer Number
23	17	--				--				--				--				Additional extension # after connecting
24	18	--				--				--				--				Reserved
25	19	--				--				--				--				Reserved for Phone 3 Jack
26	1A	0				0				0				0*				Main Password # ("0000" means no pwd)
27	1B	--				--				--				--				Reserved for Password
28	1C	0				7				C				E*				Program Year, (1998)
29	1D	0				0				0				1*				Program Month (01, January)
30	1E	0				0				0				2*				Program Day (02)
31	1F	0				0				0				C*				Program Hour (12)
32	20	0				0				3				B*				Program Min (59)
33	21	0				0				3				B*				Program Sec (59)
34	22																	
													• • • •					
61	3D																	
62	3E	0				1				0				1*				Version # 1.1
63	3F	--				--				--				--				Reserved for Version Number

Other timer utility routines are also very useful. This design uses an 8-MHz crystal as its timing reference. During initialization, the registers of prescaler 0 and timer 0 are assigned to the values of 4 and 250 respectively. Therefore, timer 0 interrupt request, IRQ4 (T0), is generated every 1 ms. By counting (decreasing) the 1-ms ticker and 5-ms ticker when T0 interrupt occurs, the timer utility routines can be created. The following is an example of a 50-ms timer. The program waits for 50 ms until time-out.

```
WTime_50ms:
               ld      tick5ms,#10
wt_ms:
               cp      tick5ms,#0
               jr      ne,wt_ms
               ret
```

PLAS.S Programming Listing

The following PLAS.S programming listing contains all subroutines for accessing EEPROM, telephone line interfaces, relays, and DTMF decoder, among others.

```
PORT ASSIGNMENT

port0           .equ    0              ; port0
port2           .equ    2              ; port2
port3           .equ    3              ; port3
;
; REGISTER ASSIGNMENT
RGINT           .equ    #%70       ; Assign Interrupt registers in %70
RGST            .equ    #%6F       ; Assign Stk Register(lower) in %6f
;
;
; REGISTER ASSIGNMENTS---- from %10 to %3f and %60 to %7f are for stack
;
u3v             .equ    %10             ; u3 input value
u6v             .equ    %11             ; u6 input value
flgs            .equ    %12             ; misc flags **
; D0 = first time to check # bit, 0= 1st time, 1= not 1st time
; D1 = program mode, 0=not prog mode, 1=prog mode.
; D2 = max ring num to ph bit, 0=pick up before max ring, 1=max ring
; D3 = Second tick is equal to 0, 1=ture, 0=faulse
; D4
; D5 = ph1 flag, 1=now is for ph1
; D6 = ph2 flag, 1=now is for ph2
; D7 = ph3 flag, 1=now is for ph3
ring_count      .equ    %13             ; ringer count number
ring_on_num     .equ    %14             ; ringing on number
EPadd           .equ    %15             ; EPROM address
EPdata0         .equ    %16             ; LS byte
EPdata1         .equ    %17             ; MS byte
phtemp          .equ    %18
tempa           .equ    %19
tempb           .equ    %1a
tempadd         .equ    %1b
tempnum         .equ    %1c
ringnum         .equ    %1d
cnt             .equ    %1e
tcnt            .equ    %1f
;
; Phone number received buffers -- from %30 to %3a total 11 digits
;
ph1             .equ    %30
flg1            .equ    %3b
; D0 = illegal password flag, 0=legal, 1=illegal38
; D1 = quick pwd or pass pwd flag, 0=quick pwd, 1=pass pwd
pwd0            .equ    %3c             ; general pwd lower byte
pwd1            .equ    %3d             ; general pwd second byte
pwd2            .equ    %3e             ; general pwd third byte
pwd3            .equ    %3f             ; general pwd higher byte
;
; INTERRUPT REGISTER ASSIGNMENT AREA  --- from %40 to %4f
;
seconds_tick    .equ    %40             ; seconds tick **
tick5ms         .equ    %41             ; t0 interrupt tick timer **
t0_tick         .equ    %42             ; timer0 timer tick **
tickms          .equ    %43             ; t0 ticker for 1ms
t0_t5ms         .equ    %44             ; timer0 timer
;
buf0            .equ    %4c
```

```
buf1            .equ    %4d
buf2            .equ    %4e
buf3            .equ    %4f

;
; ------------------------------------------
;           Program Start
; ------------------------------------------
            jp      Start
;
; ------------------------------------------
;           Program Macros
; ------------------------------------------
wait            .MACRO
                nop
                nop
                nop
                nop
                nop
                .ENDM

;====================================================================
;                                                                   |
;                   Entry Point of main Z8 Program                  |
;                                                                   |
;====================================================================
Start:
        di                              ; Interrupts disabled
        ld      p01m,#00000100B         ; Port 0 output
                                        ; %F8h, Port 0 Mode Register
        ld      pre0,#00010001B         ; presc0,(000100)#4, on 8 MHz
                                        ; Prescaler 0 Register
        ld      t0,#250                 ; 1ms t0 interrupt
                                        ; Counter/Timer 0 Register
        ld      tmr,#00001111B          ; enable/load T0 and T1
                                        ; Timer Mode Register
        ld      p2m,#11110001B          ; P20,4,5,6,in,P21,2,3, out
                                        ; Port 2 Mode Register
        ld      p3m, #1                 ; P3M, port 2 push-pull
                                        ; Port 3 Mode Register
        ld      imr,#00010000B          ; irq4,5 enabled (E30 #00110000)
                                        ; Interrupt Mask Register
        clr     irq                     ; Erase pending interrupts
                                        ; Interrupt Request Register
        ld      spl,RGST                ; User stack 6f

        srp     #%50                    ;
        ld      r15,#%5e                ; point to top of regs
ram_clr:
        clr     @r15                    ; clr the reg
        djnz    r15,ram_clr             ; do it all
        ei                              ; enable interrupts
reg_str1:
        clr     port3
        clr     port2
        clr     port0
        clr     u6v                     ; u6 is D-Type FF control IC
        call    u6_out
        call    WTime_50ms
;
; ------------------------------------------
;               Main Program
; ------------------------------------------
main:
        ld      u6v,#00001000B                  ; open RLY2
        call    u6_out
        call    Chk_93C46                       ; check cfg & format
main1:
        tm      p3,#00000100B                   ; check in-use
        jr      nz,main1
        call    RingInHandler
        jr      main1
```

```
;
; -----------------------------------------
;  Check 93C46 configuration settings,
;  and if necessary, format 93C46
; -----------------------------------------
Chk_93C46:
        ld      EPadd,#0        ; starting address 0
        call    Read93C46
        cp      EPdata0,#%F0
        jp      ne,Format_93C46
        cp      EPdata1,#%F0
        jp      eq,Chk_out
Format_93C46:
        ld      EPdata0,#%F0    ; initialized EPROM
        ld      EPdata1,#%F0
        ld      EPadd,#0
        call    Write93C46
        ld      EPdata0,#%01    ; default card ringer number
        ld      EPdata1,#%00
        ld      EPadd,#1
        call    Write93C46
        ld      phtemp,#1       ; starting ph1
EP_phlp:
        ld      EPdata0,phtemp
        ld      EPdata1,#0
        inc     EPadd
        call    Write93C46
        ld      tempa,#3
EP_try1:
        ld      EPdata0,#0
        ld      EPdata1,#0
        inc     EPadd
        call    Write93C46
        dec     tempa
        cp      tempa,#0
        jr      ne,EP_try1
        ld      EPdata0,#6
        ld      EPdata1,#0
        inc     EPadd
        call    Write93C46
        ld      tempa,#3
EP_try2:
        ld      EPdata0,#0
        ld      EPdata1,#0
        inc     EPadd
        call    Write93C46
        dec     tempa
        cp      tempa,#0
        jr      ne,EP_try2
        inc     phtemp
        cp      phtemp,#4
        jp      ne,EP_phlp

; ********** testing using password # 1234
;        ld      EPdata0,#%34
;        ld      EPdata1,#%12
; ********** testing**********************

        ld      EPdata0,#%00
        ld      EPdata1,#%00
        ld      EPadd,#26               ; set the default PWD-> "0000"
        call    Write93C46
        ld      EPdata0,#0
        ld      EPdata1,#0
        ld      EPadd,#27               ; reserved for General PWD
        call    Write93C46

; ************************** to record the program EEPROM date & time
        ld      EPdata0,#%CE            ; set year is 1998
        ld      EPdata1,#%07
        ld      EPadd,#28
        call    Write93C46
        ld      EPdata0,#%01            ; set mon is Jan. (01)
        ld      EPdata1,#%00
        ld      EPadd,#29
        call    Write93C46
        ld      EPdata0,#%02            ; set day is 02
        ld      EPdata1,#%00
        ld      EPadd,#30
        call    Write93C46
        ld      EPdata0,#%0C            ; set hour is 12
        ld      EPdata1,#%00
        ld      EPadd,#31
        call    Write93C46
        ld      EPdata0,#%3B            ; set min is 59
        ld      EPdata1,#%00
        ld      EPadd,#32
        call    Write93C46
        ld      EPdata0,#%3B            ; set sec is 59
        ld      EPdata1,#%00
        ld      EPadd,#33
        call    Write93C46

; **********************************************************************
        ld      EPdata0,#%01
        ld      EPdata1,#%01            ; the version# is 1.1
        ld      EPadd,#62
        call    Write93C46
        ld      EPdata0,#0
        ld      EPdata1,#0
        ld      EPadd,#63               ; reset "Force Re-format" to 0
        call    Write93C46
Chk_out:
        ret
;
Read93C46:
        or      p2,#00001010B           ; set CS and DI high (Start bit)
        call    One_cycle
        or      p2,#00000010B           ; Read first OP code
        call    One_cycle
        and     p2,#11111101B           ; Read second OP code
        call    One_cycle
        call    Send_addr
        call    Get_data
        and     p2,#11110111B           ; get CS low
        ret
;
Write93C46:
        or      p2,#00001010B           ; set CS and DI high (Start bit)
        call    One_cycle
        and     p2,#11111101B           ; Write Enable first OP code
        call    One_cycle
        and     p2,#11111101B           ; Write Enable second OP code
        call    One_cycle
        ld      tempadd,#00110000B      ; "11XXXX" for EWEN
        call    Send_dt
        and     p2,#11110111B           ; get CS low

;*********** Start writing data **************************
        or      p2,#00001010B           ; set CS and DI high (Start bit)
        call    One_cycle
        and     p2,#11111101B           ; Write first OP code
        call    One_cycle
        or      p2,#00000010B           ; Write second OP code
        call    One_cycle
        call    Send_addr
        ld      tempadd,EPdata1
        ld      tempb,#8
        call    SendLp
        ld      tempadd,EPdata0
        ld      tempb,#8
        call    SendLp
        and     p2,#11110111B           ; get CS low
        wait
        or      p2,#00001000B           ; CS high
write_try1:
        tm      p2,#00000001B           ; check DO
        jr      z,write_try1
        and     p2,#11110111B           ; get CS low
        wait
;*********** End of writing data  **************************

        or      p2,#00001010B           ; set CS and DI high (Start bit)
        call    One_cycle
        and     p2,#11111101B           ; Write disable first OP code
        call    One_cycle
        and     p2,#11111101B           ; Write disable second OP code
        call    One_cycle
        ld      tempadd,#00000000B      ; "00XXXX" for EWDS
        call    Send_dt
        and     p2,#11110111B           ; get CS low
        ret
;
Get_data:
        ld      tempb,#8
get_datalp:
        call    One_cycle
        tm      p2,#00000001B           ; check Data out (MCU input pin)
        jr      z,get_0
        or      EPdata1,#00000001B      ; get data "1"
        jr      get_cont
get_0:
        and     EPdata1,#11111110B      ; get data "0"
get_cont:
        dec     tempb
        cp      tempb,#0
        jp      eq,get_lp0
        rl      EPdata1
        jr      get_datalp
get_lp0:
        ld      tempb,#8
get_lp1:
        call    One_cycle
        tm      p2,#00000001B           ; check Data out (MCU input pin)
        jr      z,get1_0
        or      EPdata0,#00000001B      ; get data "1"
        jr      get1_cont
get1_0:
        and     EPdata0,#11111110B      ; get data "0"
get1_cont:
        dec     tempb
        cp      tempb,#0
        jp      eq,get_lp_ret
        rl      EPdata0
        jr      get_lp1
get_lp_ret:
        ret
;
Send_addr:
        ld      tempadd,EPadd           ; addr temp has current addr #
Send_dt:
        ld      tempb,#6
        rl      tempadd
        rl      tempadd
SendLp:
        tm      tempadd,#10000000B      ; check the MSB
        jr      z,send_0
        or      p2,#00000010B           ; data "1"
        jr      send_cont
send_0:
        and     p2,#11111101B           ; data "0"
send_cont:
        call    One_cycle
        rl      tempadd
        dec     tempb
        cp      tempb,#0
        jp      ne,SendLp
        ret
;
One_cycle:
        or      p2,#00000100B           ; clock high
        wait
        and     p2,#11111011B           ; clock low
        wait
        ret
;
RingPreset:
        and     u6v,#00001000B          ; reset the ctl pins
        call    u6_out
        or      u6v,#00001000B          ; open ph1 relay
        call    u6_out
        ret
;
; ------------------------------------------
; check ringing number for toll saver
; ------------------------------------------
RingInHandler:
        tm      p3,#00001000B           ; check ringer input
        jp      z,done_r_chk            ; return if not ringing

        ld      tick5ms,#200            ; init timeout for 1 sec.
Valid_hi_lp1:
        tm      p3,#00001000B           ; check ringer input
        jp      z,done_r_chk            ; return if not ringing
        cp      tick5ms,#196
        jr      ne,Go_next1
        call    RingPreset
Go_next1:
        cp      tick5ms,#0
        jr      ne,Valid_hi_lp1
        ld      tick5ms,#100            ; second timeout for 1 sec.
Valid_hi_lp2:
        tm      p3,#00001000B           ; check ringer input
        jp      z,done_r_chk            ; return if not ringing
        cp      tick5ms,#0
        jr      ne,Valid_hi_lp2
Valid_hi_lp3:
        tm      p3,#00001000B           ; check ringer input
        jr      nz,Valid_hi_lp3
        ld      tick5ms,#200            ; init timeout for 1 sec.
Valid_lo_lp1:
        cp      tick5ms,#0
        jr      ne,Valid_lo_lp1
        ld      tick5ms,#200            ; second timeout for 1 sec.
Valid_lo_lp2:
        tm      p3,#00001000B           ; check ringer input
        jp      nz,done_r_chk           ; return if not ringing
        cp      tick5ms,#0
        jr      ne,Valid_lo_lp2

        clr     t0_tick
        ld      seconds_tick,#5         ; set timer to clear ring
        inc     ring_count
        ld      EPadd,#1                ; starting address 1
        call    Read93C46               ; get card ringer number
        cp      ring_count,EPdata0      ; wait for number of rings
        jr      ult,done_r_chk          ; don't answer yet (but don't
                                        ; clear ring_count)
        clr     ring_count              ; ANSWER THE PHONE NOW!
        jp      AnsTheCall
done_r_chk:
        cp      seconds_tick,#0         ; has 10 seconds passed?
        jr      ne,r_chk_2              ; skip clearing ring count
        clr     ring_count              ; if 10 secs hasn't passed
r_chk_2:
        ret
;
PreCheckPwd:
        ld      EPadd,#26               ; starting address 26
        call    Read93C46               ; get card ringer number
        cp      EPdata0,#%AA
        jp      ne,PchkPagain
        cp      EPdata1,#%AA
        jp      ne,PWD_mode
        jp      quick_pwd_end
PchkPagain:
        cp      EPdata0,#0
        jp      ne,PWD_mode
        cp      EPdata1,#0
        jp      ne,PWD_mode
        jp      quick_pwd_end
PWD_mode:
        ld      pwd0,EPdata0
        and     pwd0,#00001111B         ; get the lower 4 bits
        ld      pwd1,EPdata0
        and     pwd1,#11110000B         ; get the higher 4 bits
        swap    pwd1
        ld      pwd2,EPdata1
        and     pwd2,#00001111B         ; get the lower 4 bits
        ld      pwd3,EPdata1
        and     pwd3,#11110000B         ; get the higher 4 bits
        swap    pwd3

        clr     tempnum
        ld      seconds_tick,#2         ; set delay to 2s
Pwd_tone_lp:
        tm      p3,#00001000B           ; check DV output
        jr      nz,PwdGetDtmf
        cp      seconds_tick,#0         ; replaced by the above 2 lines
        jr      ne,Pwd_tone_lp
        jp      SettoGetout
PwdGetDtmf:
        ld      tempnum,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     tempnum,#00001111B      ; keep lower 4bits
        cp      tempnum,#00001011B      ; check '*' digit
        jp      ne,SettoGetout          ;

        clr     tempnum
        ld      seconds_tick,#5         ; set delay to 5s for 4-digit in
        ld      cnt,#4                  ; starting from pwd3
PwdGetd1_lp:
        tm      p3,#00001000B           ; check DV output
        jr      nz,Pchk_digit1
        cp      seconds_tick,#0
        jp      eq,SettoGetout
        jr      PwdGetd1_lp
Pchk_digit1:
        cp      cnt,#4
        jr      ne,Pchk_next1
        ld      buf3,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     buf3,#00001111B         ; keep lower 4bits
        jp      Pchk_lpagain
Pchk_next1:
        cp      cnt,#3
        jr      ne,Pchk_next2
        ld      buf2,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     buf2,#00001111B         ; keep lower 4bits
        jp      Pchk_lpagain
Pchk_next2:
        cp      cnt,#2
        jr      ne,Pchk_next3
        ld      buf1,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     buf1,#00001111B         ; keep lower 4bits
        jp      Pchk_lpagain
Pchk_next3:
        cp      cnt,#1
        jr      ne,Pchk_start
        ld      buf0,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     buf0,#00001111B         ; keep lower 4bits
Pchk_lpagain:
        dec     cnt
        cp      cnt,#0
        jp      ne,PwdGetd1_lp
Pchk_start:
        cp      buf3,pwd3
        jp      ne,SettoGetout
        cp      buf2,pwd2
        jp      ne,SettoGetout
        cp      buf1,pwd1
        jp      ne,SettoGetout
        cp      buf0,pwd0
        jp      ne,SettoGetout
Pwd_passed:
        or      flg1,#00000010B         ; pass pwd flag
        and     flg1,#11111110B         ; legal pwd
        jr      PreCheckRet
quick_pwd_end:
        and     flg1,#11111100B         ; legal pwd and quick pwd
PreCheckRet:
        ret
SettoGetout:
        or      flg1,#00000001B         ; illegal pwd
        jr      PreCheckRet
;
; -----------------------------------------
; Answer the incoming call
; -----------------------------------------
AnsTheCall:
        or      u6v,#10000001B          ; in-use led on, off-hook
        call    u6_out
        and     flgs,#11111100B         ; reset 1st time check and
```

```
                          ; prog bit
        ld      p01m,#00000101B         ; port0(pin0 ~pin3 input)
        or      p3,#10000000B           ; enable DTMF decoder
        call    WTime_500ms
        clr     tempnum
        call    PreCheckPwd

        tm      flg1,#00000001B         ; check pwd legal or illegal
        jp      nz,Getout
        tm      flg1,#00000010B         ; check quick or pass pwd
        jr      z,Shorttick
        ld      seconds_tick,#4         ; set delay to 4s
        jr      ATickStart
Shorttick:
        ld      seconds_tick,#2         ; set delay to 2s
ATickStart:
        clr     tempnum
Chk_tone_lp:
        tm      p3,#00001000B           ; check DV output
        jr      nz,PreGetDtmf
Chk_tone_lp2:
        cp      seconds_tick,#0         ; replaced by the above 2 lines
        jr      ne,Chk_tone_lp
        jp      Turn_ph0                ; turn on the phone1 jack immi
    ; after 4s
PreGetDtmf:
GetDtmf:
        ld      tempnum,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     tempnum,#00001111B      ; keep lower 4bits
        cp      tempnum,#00001100B      ; check '#' digit
        jp      eq,Trans_mode
        cp      tempnum,#00001011B      ; check '*' digit
        jp      eq,PWD_mode             ; password mode
        cp      tempnum,#00001010B      ; check '0' digit
        jp      eq,Prog_mode
        cp      tempnum,#00000001B      ; check '1' digit
        jp      eq,Ph1_Direct
        cp      tempnum,#00000010B      ; check '2' digit
        jp      eq,Ph2_Direct
        cp      tempnum,#00000011B      ; check '3' digit
        jp      eq,Ph3_Direct
Getout:
        and     p3,#01111111B           ; disable DTMF decoder
        ld      p01m,#00000100B         ; port0(pin0 ~pin3 output)
        jp      Ans_rdy_out             ; exit ans loop for not thing
Ph1_Direct:
        ld      seconds_tick,#4         ; set delay to 4s
PGetd1_lp:
        cp      seconds_tick,#0
        jp      eq,Getout
        tm      p3,#00001000B           ; check DV output
        jr      z,PGetd1_lp
        ld      buf0,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     buf0,#00001111B         ; keep lower 4bits
        cp      buf0,#00000001B         ; check '1' digit
        jp      ne,Getout
        and     p3,#01111111B           ; disable DTMF decoder
        ld      p01m,#00000100B         ; port0(pin0 ~pin3 output)
        jp      Turn_ph1                ; turn on the phone1 jack
Ph2_Direct:
        ld      seconds_tick,#4         ; set delay to 4s
PGetd2_lp:
        cp      seconds_tick,#0
        jp      eq,Getout
        tm      p3,#00001000B           ; check DV output
        jr      z,PGetd2_lp
        ld      buf0,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     buf0,#00001111B         ; keep lower 4bits
        cp      buf0,#00000010B         ; check '2' digit
        jp      ne,Getout
        and     p3,#01111111B           ; disable DTMF decoder
        ld      p01m,#00000100B         ; port0(pin0 ~pin3 output)
        jp      Turn_ph2                ; turn on the phone2 jack
Ph3_Direct:
        ld      seconds_tick,#4         ; set delay to 4s
PGetd3_lp:
        cp      seconds_tick,#0
        jp      eq,Getout
        tm      p3,#00001000B           ; check DV output
        jr      z,PGetd3_lp
        ld      buf0,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     buf0,#00001111B         ; keep lower 4bits
        cp      buf0,#00000011B         ; check '3' digit
        jp      ne,Getout
        and     p3,#01111111B           ; disable DTMF decoder
        ld      p01m,#00000100B         ; port0(pin0 ~pin3 output)
        jp      Turn_ph3                ; turn on the phone3 jack
Trans_mode:
        ld      seconds_tick,#4         ; set delay to 4s
Getd1_lp:
        tm      p3,#00001000B           ; check DV output
        jr      nz,Tchk_digit1
        cp      seconds_tick,#0
        jp      eq,Getout
        jr      Getd1_lp
Tchk_digit1:
        ld      buf0,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     buf0,#00001111B         ; keep lower 4bits
Getd2_lp:
        tm      p3,#00001000B           ; check DV output
        jr      nz,Tchk_digit2
        cp      seconds_tick,#0
        jp      eq,Getout
        jr      Getd2_lp
Tchk_digit2:
        ld      tempnum,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     tempnum,#00001111B      ; keep lower 4bits
        cp      tempnum,#00001100B      ; check '#' digit
        jp      ne,Getout
        and     p3,#01111111B           ; disable DTMF decoder
        ld      p01m,#00000100B         ; port0(pin0 ~pin3 output)

        ld      EPadd,#2                ; starting address 2 (ph1)
        call    Read93C46               ; get card ringer number
        cp      buf0,EPdata0            ; wait for number of rings
        jp      eq,Turn_ph1             ; turn on the phone1 jack
        ld      EPadd,#10               ; starting address 10 (ph2)
        call    Read93C46               ; get card ringer number
        cp      buf0,EPdata0            ; wait for number of rings
        jp      eq,Turn_ph2             ; turn on the phone2 jack
        ld      EPadd,#18               ; starting address 18 (ph3)
        call    Read93C46               ; get card ringer number
        cp      buf0,EPdata0            ; wait for number of rings
        jp      eq,Turn_ph3             ; turn on the phone3 jack
        jp      Getout
;
Turn_ph0:
        and     p3,#01111111B           ; disable DTMF decoder
        ld      p01m,#00000100B         ; port0(pin0 ~pin3 output)
Turn_ph1:
        or      u6v,#10000101B          ; in-use led on, off-hook,
    ; main rly on
        call    u6_out
        and     u6v,#11110111B          ; switch to ph1
        call    u6_out
        or      flgs,#00100000B         ; set ph1
        call    ToRingPh
        jp      Ans_rdy_out
Turn_ph2:
        or      u6v,#10011101B          ; in-use, off-hook, main rly on,
                             ; ph2 on
        call    u6_out
        or      flgs,#01000000B         ; set ph2
        call    ToRingPh
        jp      Ans_rdy_out
Turn_ph3:
        or      u6v,#10101101B          ; in-use,off-hook,rly1 on,ph3 on
        call    u6_out
        or      flgs,#10000000B         ; set ph3
        call    ToRingPh
        jp      Ans_rdy_out
Prog_mode:
        clr     tempnum
        ld      seconds_tick,#5         ; set delay to 5s
Prog_start:
        tm      p3,#00001000B           ; check DV output
        jr      nz,Prog_digit1
        cp      seconds_tick,#0
        jp      eq,Getout
        jr      Prog_start
Prog_digit1:
        ld      tempnum,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     tempnum,#00001111B      ; keep lower 4bits
        cp      tempnum,#00001011B      ; check '*' digit
        jp      eq,PgPwdOK
        cp      tempnum,#00001100B      ; check '#' digit
        jp      ne,Getout
PgPhnum:
        ld      cnt,#3
PgGetd1_lp:
        tm      p3,#00001000B           ; check DV output
        jr      nz,Prog_digit2
        cp      seconds_tick,#0
        jp      eq,Getout
        jr      PgGetd1_lp
Prog_digit2:
        cp      cnt,#3
        jr      ne,Pg_next1
        ld      buf3,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     buf3,#00001111B         ; keep lower 4bits
        jp      Pg_lpagain
Pg_next1:
        cp      cnt,#2
        jr      ne,Pg_next2
        ld      buf2,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     buf2,#00001111B         ; keep lower 4bits
        jp      Pg_lpagain
Pg_next2:
        cp      cnt,#1
        jr      ne,Pg_TrChk
        ld      buf1,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     buf1,#00001111B         ; keep lower 4bits
Pg_lpagain:
        dec     cnt
        cp      cnt,#0
        jp      ne,PgGetd1_lp
Pg_TrChk:
        tm      p3,#00001000B           ; check DV output
        jr      nz,Prog_digit3
        cp      seconds_tick,#0
        jp      eq,Getout
        jr      Pg_TrChk
Prog_digit3:
        ld      tempnum,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     tempnum,#00001111B      ; keep lower 4bits
        cp      tempnum,#00001100B      ; check '#' digit
        jp      ne,Getout

; ** so far, buf3 is the ph#, buf2 is Transfer#, & buf1 is Max. ring#
        cp      buf3,#4                 ; ph# only is 1, 2, or 3
        jp      uge,Getout
        cp      buf2,#10                ; allow trans# from 1 to 9 only
        jp      uge,Getout
        cp      buf1,#10                ; from 1 ~ 9
        jp      uge,Getout
        cp      buf3,#1
        jp      ne,Pnxt_ph1
        ld      EPdata0,buf2
        ld      EPdata1,#0
        ld      EPadd,#2                ; phone 1 transfer# address
        call    Write93C46
        ld      EPdata0,buf1
        ld      EPdata1,#0
        ld      EPadd,#6                ; phone 1 max ringer# address
        call    Write93C46
        jp      PgmodeEnd
Pnxt_ph1:
        cp      buf3,#2
        jp      ne,Pnxt_ph2
        ld      EPdata0,buf2
        ld      EPdata1,#0
        ld      EPadd,#10               ; phone 2 transfer# address
        call    Write93C46
        ld      EPdata0,buf1
        ld      EPdata1,#0
        ld      EPadd,#14               ; phone 2 max ringer# address
        call    Write93C46
        jp      PgmodeEnd
Pnxt_ph2:
        ld      EPdata0,buf2
        ld      EPdata1,#0
        ld      EPadd,#18               ; phone 3 transfer# address
        call    Write93C46
        ld      EPdata0,buf1
        ld      EPdata1,#0
        ld      EPadd,#22               ; phone 3 max ringer# address
        call    Write93C46
        jp      PgmodeEnd
PgPwdOK:
        ld      cnt,#4                  ; starting from pwd3
PgPwdGetd1_lp:
        tm      p3,#00001000B           ; check DV output
        jr      nz,PgPchk_digit1
        cp      seconds_tick,#0
        jp      eq,Getout
        jr      PgPwdGetd1_lp
PgPchk_digit1:
        cp      cnt,#4
        jr      ne,PgPchk_next1
        ld      buf3,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     buf3,#00001111B         ; keep lower 4bits
        jp      PgPchk_lpagain
PgPchk_next1:
        cp      cnt,#3
        jr      ne,PgPchk_next2
        ld      buf2,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     buf2,#00001111B         ; keep lower 4bits
        jp      PgPchk_lpagain
PgPchk_next2:
        cp      cnt,#2
        jr      ne,PgPchk_next3
        ld      buf1,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     buf1,#00001111B         ; keep lower 4bits
        jp      PgPchk_lpagain
PgPchk_next3:
        cp      cnt,#1
        jr      ne,PgPchk_start
        ld      buf0,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     buf0,#00001111B         ; keep lower 4bits
PgPchk_lpagain:
        dec     cnt
        cp      cnt,#0
        jp      ne,PgPwdGetd1_lp
PgPchk_start:
        tm      p3,#00001000B           ; check DV output
        jr      nz,PgPchk_digit2
        cp      seconds_tick,#0
        jp      eq,Getout
        jr      PgPchk_start
PgPchk_digit2:
        ld      tempnum,p0
        call    WaitDig                 ; wait for this dtmf completed
        and     tempnum,#00001111B      ; keep lower 4bits
        cp      tempnum,#00001100B      ; check '#' digit
        jp      ne,Getout
PgPwd_write:
; ** each of 4-digit password can only be set from 0 ~ 9
        cp      buf3,#11                ; from 0 ~ 9
        jp      uge,Getout
        cp      buf2,#11                ; from 0 ~ 9
        jp      uge,Getout
        cp      buf1,#11                ; from 0 ~ 9
        jp      uge,Getout
        cp      buf0,#11                ; from 0 ~ 9
        jp      uge,Getout

        swap    buf3
        or      buf3,buf2
        ld      EPdata1,buf3
        swap    buf1
        or      buf1,buf0
        ld      EPdata0,buf1
        ld      EPadd,#26               ; general Password address
        call    Write93C46
PgmodeEnd:
        jp      Getout
Ans_rdy_out:
        and     u6v,#01111011B          ; turn off led
        call    u6_out                  ; main rly back to orig position
        call    WTime_100ms
        and     u6v,#11111110B          ; on hook
        call    u6_out
        call    Wait2secs
        clr     flgs
        ret
WaitDig:
        tm      p3,#00001000B           ; check DV output
        jr      nz,WaitDig
        ret
;
ToRingPh:
        tm      flgs,#00100000B         ; check ph1
        jr      z,tnext_ph2
        ld      EPadd,#6                ; starting address 6 (ph1)
        call    Read93C46               ; get card ringer number
        ld      ringnum,EPdata0         ; number of rings of ph1
        jp      ToRing_begin
tnext_ph2:
        tm      flgs,#01000000B         ; check ph2
        jr      z,tnext_ph3
        ld      EPadd,#14               ; starting address 14 (ph2)
        call    Read93C46               ; get card ringer number
        ld      ringnum,EPdata0         ; number of rings of ph1
        jp      ToRing_begin
```

```
tnext_ph3:
        ld      EPadd,#22               ; starting address 22 (ph3)
        call    Read93C46               ; get card ringer number
        ld      ringnum,EPdata0         ; number of rings of ph3
ToRing_begin:
        or      u6v,#01000000B          ; enable PWM ctl
        call    u6_out
        call    WTime_5ms
        ld      cnt,#40                 ; total ringing 40 times
trp_lp:
        or      u6v,#00000010B          ; ctl 12v pin
        call    u6_out
        call    WTime_25ms
        and     u6v,#11111101B          ; ctl 12v pin off
        call    u6_out
        call    WTime_25ms
; -----------------------------------------
; assume Pulse Out is +5V in the normal
; condition, and 0V when phone is picked up
; -----------------------------------------
        tm      p3,#00000010B           ; check pulse pin
        jp      z,Pickup_ok             ; some one pick up the phone
        dec     cnt
        cp      cnt,#0
        jp      ne,trp_lp
;        call    Wait4secs               ; 4 sec idle
CheckIdleRing4secs:
        ld      cnt,#80
CIR4_loop:
        call    WTime_50ms
        tm      p3,#00000010B           ; check pulse pin
        jp      z,Pickup_ok             ; some one pick up the phone
        dec     cnt
        cp      cnt,#0
        jp      ne,CIR4_loop
; -----------------------------------------
; assume Pulse Out is +5V in the normal
; condition, and 0V when phone is picked up
; ----------------------------------------
;        tm      p3,#00000010B           ; check pulse pin
;        jp      z,Pickup_ok             ; some one pick up the phone
        dec     ringnum
        cp      ringnum,#0
        jp      ne,ToRing_begin
        or      flgs,#00000100B         ; D2 is set for max ring number
        jr      Ringout
Pickup_ok:
        and     flgs,#11111011B         ; before 6 rings
Ringout:
        and     u6v,#10111111B          ; disable PWM ctl
        call    u6_out
        ret
;
RingChkPick:
        ld      tick5ms,#5
RingChkLoop:
        tm      p3,#00000010B           ; check pulse pin
        jp      z,Pickup_ok             ; some one pick up the phone
        cp      tick5ms,#0
        jr      ne,RingChkLoop
        ret
;
RingChk4sPick:
        ld      seconds_tick,#4         ; set delay to 4s
        clr     t0_tick                 ; reset 4ms timer tick
RingChk4sLoop:
        tm      p3,#00000010B           ; check pulse pin
        jp      z,Pickup_ok             ; some one pick up the phone
        cp      seconds_tick,#0         ; wait delay
        jr      ne,RingChk4sLoop
        ret
;
```

```
;-----------------------------------------
;         Z8 Processor IRQ routines
; -----------------------------------------
;
irq0:
irq1:
irq2:
irq3:
        iret
; -----------------------------------------
;     timer and tick register updating
; -----------------------------------------
t0_int:
        push    rp                      ; save user rp
;        srp     RGINT                    ; set interrupt bank
        dec     tickms                  ; 1ms ticker
        inc     t0_t5ms
        cp      t0_t5ms,#5              ; for 5 ms
        jr      ne,t00
        clr     t0_t5ms
        dec     tick5ms                 ; count 5ms timer
t0str:
        inc     t0_tick                 ;
        cp      t0_tick,#200            ; for 1 sec
        jr      ne,t00
        clr     t0_tick
        dec     seconds_tick            ;
        cp      seconds_tick,#0
        jr      ne,t00
        or      flgs,#00001000B         ; set flag to true

; --------- for testing
;        xor     u6v,#10000000b           ; LED flashing
;        call    u6_out
; ----------------------
;
```

```
t00:
t0_exit:
        pop     rp                      ; restore user rp
        iret
;
t1_int:
        iret

; ------------------------------------------
; general ms and seconds delay rountines
; ------------------------------------------
Wait4secs:
        ld      seconds_tick,#4         ; set delay to 4s
        jr      ws_starh
Wait2secs:
        ld      seconds_tick,#2         ; set delay to 2s
        jr      ws_starh
Waitsecs:
        ld      seconds_tick,#1         ;
ws_starh:
        clr     t0_tick                 ; reset 4ms timer tick
waitdelay:
        cp      seconds_tick,#0         ; wait delay
        jr      ne,waitdelay
        ret
;
; based on 5ms per tick time
;
WTime_5ms:
        ld      tick5ms,#1
        jr      wt_ms
WTime_25ms:
        ld      tick5ms,#5
        jr      wt_ms
WTime_50ms:
        ld      tick5ms,#10
        jr      wt_ms
WTime_100ms:
        ld      tick5ms,#20
        jr      wt_ms
WTime_200ms:
        ld      tick5ms,#40
        jr      wt_ms
WTime_500ms:
        ld      tick5ms,#100
wt_ms:
        cp      tick5ms,#0
        jr      ne,wt_ms
        ret
u6_out:
        di
        ld      p0,u6v
        wait
        or      p3,#01000000B                ; u6 clk goes high
        wait
        and     p3,#10111111B                ; u6 clk goes low
        ei
        ret
```

Conclusion

By successfully performing the above subroutines and accessing the EPROM using the Z86E30 and a CCP™ Emulator, any engineer can easily implement a smart phone-line auto-switcher for a variety of SOHO applications.

References

1. Linear Technology Design Note 134: “Telephone Ring-Tone Generation.”
   http://www.linear-tech.com
2. Hello Direct Telephone Catalog, “Line Switches” section.
   http://www.hello-direct.com
3. Xecom, Inc.
   http://www.xecom.com