DIALER TELEPHONE CIRCUIT

4 December 1993

Marlon Rose-Mighty
Computer/Electronic Engineering Technology
Mohawk College of Applied Arts and Technology
132 Stinson St.
Hamilton, Ontario

MASTER COPY. Explains All.

```
*INITIALIZATION ROUTINE
*SETUP PORTS AND DATA DIRECTION REGISTER
*set-up data direction for ports A,B and C
*1)set DD=0       2)set TO=1
*3) set CL=0      in that order          Set Tone out  Data Direction  Clock
*set all temporary locations and clear them
*       i.e TEMP, TMPX, COUNT
*define all locations in memory and clear them
PORTA        EQU     $00
PORTB        EQU     $01
PORTC        EQU     $02
DDRA         EQU     $04
DDRB         EQU     $05
DDRC         EQU     $06

        This is A HIDDEN EQUATE
              BUFFER = STORAGE LOC'N
                              50
50 BUFFER    ORG     $50
58 STORAGE   RMB     8       ;8 byte temp storage loc'n to HOLD # in PACKED BCD Form.
A8 COUNT     RMB     80      ;80 bytes for 10 loc'n of #'s
A9 KEY       RMB     1
AA TMPX      RMB     1       ;1 byte keycode storage
AB TEMP      RMB     1
AC DISPA     RMB     1
AD DISPX     RMB     1       ; 1 byte for Display
AE DLYX      RMB     1
AF DLYA      RMB     1
             ORG     $0100
*start of eprom in 705C8
*INIT        RSP             ;reset str ptr to $FF
*set up i/o ports

             LDA     #$00    ;set ports to O/P
             STA     PORTB   ;set 7segment off  Test all 7-segment on
             LDA     #$FF
             STA     DDRB    ; set up port B as o/p
             STA     DDRA      set up port A as o/p.
*DELAY FOR DEMO
             CLRX
INRLP        DECX
             BNE     INRLP
             DECA
             BNE     INRLP

             LDA     #$FF
             STA     PORTB   ; turn off Display

             LDA     #$10
             STA     PORTC   ; disable Tone o/p for ...
             LDA     #$00 #$16   ;portc set-up
             STA     DDRC    ;set-up data direction
```

p 1 of 10

```
*Main Routine
*
*

LOOP1
                JSR     BUFIN   ;to start reading the keypad
                BRA     LOOP1



*Subroutine to read a single digit from the keyboard
*
*NOTE: Data Direction=0 , Tone Out(TO)=1, clock(CL)=0
*
*Input A=?     X=n
*Output A=key X=n

ONEKEY          STX     TMPX            ;x-->TMPX
*simulate stack operation
CHNKEY          BRSET   3,PORTC,CHNKEY  ;WAITS FOR KEY RELEASE

CHKEY           BRCLR   3,PORTC,CHKEY   ;brnh if keypress PORTC
                JSR     DLY20

*wait for keypress

                CLR     KEY             ;
                BSET    1,PORTC         ;toggle bit
                JSR     DLY20           ;20uS delay
                BCLR    1,PORTC         ;clear clock bit
                JSR     DLY20           ;delay 20uS
                BSET    1,PORTC         ;set bit 1 high
                JSR     DLY20           ;
                BCLR    1,PORTC         ;
                CLRX                    ;CHECK POSITION OF THIS

t this point we have valid data

                CLR     KEY
RIN             JSR     DLY20
```

```
        LDA     PORTC
        AND     #$01        ;portc-->A
        LSL     KEY         ;mask out bottom bit
        ADD     KEY         ;shft lft all bits in key
        STA     KEY         ;Acc+key-->Acc
        BSET    1,PORTC     ;Acc-->key
        JSR     DLY20       ;
        BCLR    1,PORTC     ;
        INCX                ;
        CMPX    #$04        ;
        BNE     SERIN       ;
        LDX     TMPX        ;
        LDA     KEY         ;replace value
        JSR     DISPNUM     ;keycode --> A
        RTS                 ;display number
```

```
*Subroutine to read characters into keypad buffer
*
*Input   A=?     X=?
*Output  A=1100%(*)   1000%(#)   X=the number of characters in the buffer -1
*
*

BUFIN           CLRX
CLEARBUF        CLR     BUFFER,X        ;0'S buffer Keypad or I/O
                INCX
                CMPX    #$08
                BNE     CLEARBUF

                LDX     #$FF            ;-1 --> X
BUFSTART        JSR     ONEKEY          ;get a character
                                        ;check for STAR
                CMP     #%1100
                BEQ     RECALL
                CMP     #%1000          ;check for NUMBER
                BEQ     STORE
                INCX
*put digit in buffer
                STX     TMPX            ;x-->TMPX (count)
                ASRX                    ;divide X by 2
                BCS     LOWERNIB        ;branch if carry set
PPERNIB         LSLA                    ;logical shift left 4 bit
                LSLA                    ;positions
```

```
                LSLA
                LSLA
                STA
                        BUFFER,X

                                        ;store chrcter to buffer
LOWERNIB        BRA
                ADD     SKIPLOW         ;A-->buffer + X
SKIPLOW         STA     BUFFER,X
                LDX     BUFFER,X        ;buffer + X --> Acc
                STA     TMPX            ;Acc --> buffer
                BRA     PORTA           ;get back count
                BRA     BUFSTART        ;
```

*NOTE: THERE IS NO END HERE THIS SIMPLY LOOPS

```
*SUBROUTINE TO RECALL A NUMBER FROM MEMORY
*
*INPUT   ?
*OUTPUT  ?
*A=*START  X=the number of characters in the keyboard buffer -1
RECALL          ASRX                    ;Divide X by 2
                LDA     BUFFER,X        ;(m)-->Acc  Contents of
*                                       ;Indexed offset
                JSR     CONVERT         ;ld A with last kybrd #
                ASLA                    ;* by 8....shift 3 places
                ASLA
                ASLA
                ADD     #STORAGE        ;Addr of 1st Loc'n block

*TEMP is to point to the beginning, X simply offsets into RAM

                STA     TEMP
                STA     PORTA           ;Debug to monitor Address information
*               BSET    2,PORTC         ;set PC2=high
                                        ;thus, DD is high to tell ETC to use I/O as Input
                BSET    0,DDRC          ;set bit 0 DDRC to output
                                        ;to ETC I/O
                CLRX
OUTLOOP         STX     COUNT
                ASRX                    ;X divide by 2
                BCS     LOWRNIB         ;
UPPRNIB         TXA
                ADD     TEMP
                TAX
                LDA     ,X
```

USE CARRY TO DETECT IF NUMBER SHOULD BE PUT IN LOWER OR UPPER PART OF BYTE BY DETECTING ODD OR EVEN VALUES of X

critical error here cause info was not being sent or recieved.

```
          LSRA
          LSRA
          LSRA                    ;bring data down to
          LSRA                    ;lower nibble
LOWRNIB   BRA
          TXA       LOWSKIP
          ADD                     ;
          TAX       TEMP          ;
          LDA                     ;
LOWSKIP   AND       ,X            ;
          BEQ       #$0F          ;mask out uppr 4 bits
          JSR       DONELOOP      ;result=0 then branch
          LDX       OUTDATA       ;cause keycode is 0
          INCX      COUNT
DONELOOP  BRA       OUTLOOP
          BCLR      2,PORTC       ;set DD low again
          BCLR      0,DDRC        ;set bit 0, DDR to input

          RTS
```

X → A; contents of TEMP is added to A; A → X; using X as an offset into RAM to get next piece of data. "to DONELOOP"; "USED TO DETECT THE END." BNE (replacing BRA); CMPX #15; "count → X"; "which is, check for 15 digits max." "Restore # characters in buffer count." "set up... as ...l."

p5 of 10

```
*subroutine to store a number
*copy keyboard buffer --> memory
*NOTE: 8 bytes implies 16 numbers can be stored in keycode format

STORE     JSR       ONEKEY        * get key from keyboard
          JSR       CONVERT       *convert to BCD

          ASLA                    *A * 8d
          ASLA                    *find boundary of table
          ASLA

          ADD       #STORAGE      *add offset
          STA       TEMP          *Acc --> TEMP

          STA       PORTA         *DEBUG ADDRESS

          CLRX
COPYBUF   LDA       BUFFER,X      *ld 1st val of key buffer
          INCX
          STX       TMPX
          LDX       TEMP
          STA       ,X
          INCX
          STX       TEMP
          LDX       TMPX
          CPX       #$08

          BNE       COPYBUF
```

"by"; "to STORAGE"; "i.e. where the number is to be put in RAM"; "STORAGE (58) + A → A, THE ACTUAL ADDRESS TO STORE AT."; "FOR TESTING PURPOSES"; "into A"; "COUNT X → TMPX preserve count."; "TEMP → X Point to STORAGE LOCATION"; "STORE A AT STORAGE LOCATION"; "increment count"; "X → Temp preserve count (stack emulation)"; "Restore old count"; "TEST FOR END OF BUFFER"; "If NOT END get the next value from buffer."

Side calculations: 0001 shift → 1000; 10000 = 16d; START → 16 + 58 = 10h + 58h = 68h; 59 … 60 → 1; 67 – addr of Lo; "16d be..." → 68 ← 2

```
        RTS
*NOTE: now we have stored the number for later recall
```

```
*subroutine to convert BINARY keycode to BCD????????
*INPUT  A=key    X=n
*OUTPUT A=BCD    X=n

CONVERT     STX     TMPX        *X-->TMPX
            STA     TEMP        *keep Acc
            AND     #$0F        *mask lower bits
            BNE     HINIB       *
**lets work on lower NIB

            LDA     TEMP        *temp ---> A
            LSRA                *shift upper nib to lower
            LSRA                *
            LSRA                *
            LSRA                *

HINIB       TAX                 *A-->X
            LDA     TABLE,X     *table + X --> Acc
                                ;Acc = BCD of keycode
            LDX     TMPX        *retrieve index register
            RTS
```

; using keycode as an offset into the table

```
TABLE
        FCB
        FCB     $0D,$0B,$0C,$0A,$00
        FCB     $05,$08,$02,$0F,$06
                $09,$03,$0E,$04,$07,$01
                *
```

p7 §10

1010 – A
1011 – B
1100 – C
1101 – D
1110 – E
1111 – F

```
 *subroutine to actually output one character to "ETC"
 *subroutine to write data to controler
 *to send DTMF out???
 *Note: attempt for 300mS per digit here !!!!!
 *     :  there are 2 keycodes per byte
*INPUT  A=keycode       X=n
*OUTPUT A=?             X=n

OUTDATA
        STX     TMPX    *save X register
        JSR     DISPNUM *display number being dialed
        LSLA            *logical shift left to put stuff KEYCODE.
        LSLA            *in upper four bits
        LSLA
        LSLA
        LDX     #$04    *set count
TBIT    BSET    1,PORTC *toggle clock high
        LSLA            *send bit3--->carry  (CARRY IS BEING USED TO DETECT DATA)
        BCS     ONEOUT  *if the carry is set then branch
        BCLR    0,PORTC *clear I/O data
        BRA     MOREBITS
```

```
EOUT
              BSET     0,PORTC *puts a one out on I/O port
REBITS
              JSR      DLY20   *delay 20uS
              BCLR     1,PORTC *toggle CL low "clocking"
              JSR      DLY20   *dly,keep data valid for at least
                                ;10uS to capture data and store.
                                ;as requierd
                                ;by the ETC
              DECX
              BNE      OUTBIT
              BCLR     4,PORTC *set TO -- to tone output
              JSR      DLY 300
*delay loop for 300mS
OUTLP         LDA      #192    *pg 2-19 delay loop (6*50mS)??
INNRLP        CLRX
              DECX
              BNE      INNRLP
              DECA
              BNE      OUTLP   RTS
              BSET     4,PORTC *set TO=high
              LDX      TMPX
              RTS
```

JSR DLY300

STX DLYX. STA DLYA.

DLY 300

LDX DLYX LDA DLYA.

Disable tone generation

TMPX → X restore X register

How do we know this ETC is accepting this info

★ MAGIC ★

Ans: Because it's been clocked IN

```
*Reference pg. 4-41 DELAY ROUTINE
DLY20         STX      DLYX
              LDX      #3      *3*7~*1uS/~=21uS
L20           DECX             *delay loop approx. 21uS
              BNE      L20     *3-2,2-1,1-0
              LDX      DLYX
              RTS
```

p8 of 10

p9 A10

```
*7 segment display subroutine
*INPUT  X=N      A=keycode
*OUTPUT X=n      A=keycode

DISPNUM         STA     DISPA           *stack emulate
                STX     DISPX           *
                TAX                     *Acc ---> X index reg
                LDA     DISPTBL,X       *use keycode as offset
*into the table to load                 *the
*correct value
*                                       *for LED code
                STA     PORTB           ; TURN ON LED's
                LDA     DISPA           ; RESTORE VALUES.
                LDX     DISPX           ; "     "
                RTS

NOTE:  converting KEYCODE-->DSPLY CODE
this table is in order of keycode

SPTBL           FCB     $FF,$FF,$FF,$FF,$A0
                FCB     $92,$80,$C4,$D6,$82
                FCB     $98,$D0,$C3,$99,$F8
                FCB     $F9

                ORG     $1FFE   **********
                FCB     $01
                FCB     $00     **********
```

A --> DISPA
X --> DISPX

DISPA --> A
DISPX --> X

RESET VECTOR
START OF CODE