Dialer Telephone Circuit

9 December 1993

Book 2 of 3

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

MC34010 Clocking

The Electronic Telephone Circuit clock speed requirement is slow enough (typically 20kHz) so that it is not necessary to divide down the processor system clock, but rather port C (PC1) output is toggled. This facilitates synchronizing the clock and data transfer, eliminating the need for hardware to generate the clock.

The data direction pin must be maintained at a logic"0" when the micro-controller interface section of the circuit is not in use. This is to permit normal operation of the keypad. When the micro-controller interface section is not in use, the supply voltage at the micro-controller (A+) may be disconnected to conserve power. Powering the circuit from the A+ terminal permits communication with a micro-controller and/or use of the transmit and receiver amplifiers, while the telephone is "on hook". This is required to facilitate automatic dialing.

In the automatic dialing mode, DD is a logic"1" and the four-bit code is serially entered in the sequence Bit3, Bit2, Bit1 and Bit0 into the four- bit shift register of the DTMF generator in the Electronic Telephone Circuit. Thus, only four clock cycles are required to transfer a number into the ETC. The tone generator is disabled in this mode with a logic "1" on the tone output pin. TO is then switched to a logic zero to enable the DTMF generator. The keypad decoder's "exclusive or" circuit generates the depressed push button output to indicate that only one button of the keypad is pressed. This indicates that valid data is available for the micro-controller.

The DP (depressed push button) output is also used to initiate a data transfer sequence to the micro-controller. The keypad decoder's "exclusive or" circuit also generates the mute/single tone output signals to indicate a row and/or column tone is being generated. A logic "0" on this terminal indicates that the tone generator is disabled. The single tone (MS) output terminal when a logic "1" indicates the DTMF generator is enabled and the speech network is muted or disabled.

```
*INITIALIZATION ROUTINE
*SETUP PORTS AND DATA DIRECTION REGISTER
*set-up data direction for ports A,B and C of uC
*set-up tone output, data direction and clock of ETC
*set-up all temporary locations
*define all locations in memory

PORTA
PORTB           EQU     $00
PORTC           EQU     $01
DDRA            EQU     $02
DDRB            EQU     $04
DDRC            EQU     $05
                EQU     $06


BUFFER          ORG     $50
*               RMB     8       *8 byte temp storage loc'n to hold #'s
                                *in packed BCD form
STORAGE
COUNT           RMB     80      *80 bytes for 10 loc'n of #'s
KEY             RMB     1       *reserve byte for count
TMPX            RMB     1       *reserve 1 byte for keycode storage
TEMP            RMB     1       *reserve temporary loc'n
DISPA           RMB     1
DISPX           RMB     1
DLYX            RMB     1
DLYA            RMB     1
                RMB     1

                ORG     $0100

                LDA     #$00    *set ports to O/P
                STA     PORTB   *7segment on
                LDA     #$FF
                STA     DDRB    *set-up portb as O/P
                STA     DDRA    *set-up porta as O/P

*DELAY FOR VEIWING START-UP TEST
                CLRX
INRLP           DECX
                BNE     INRLP
                DECA
                BNE     INRLP

                LDA     #$FF     *turn off display
                STA     PORTB    *

                LDA     #$10    *disable tone O/P
                STA     PORTC
                LDA     #$16    *portc set-up
                STA     DDRC    *set-up data direction
```

```
*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
*                                        *store X for use later
CHNKEY          BRSET   3,PORTC,CHNKEY   *WAITS FOR KEY RELEASE
*                                        *on BIT3 (DP)
CHKEY           BRCLR   3,PORTC,CHKEY    *brnh if keypress PORTC BIT3 (DP)
                JSR     DLY20           *debounce

*START CLOCKING AS PER MOTOROLA TIMING DIAGRAM SPEC's

                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                     *X is to be used for bit count

*at this point valid data is on I/O pin and it will be
*brought in a bit at a time on 'NGT' of clock cycle

                CLR     KEY             *keycode storage loc'n
SERIN           JSR     DLY20           *clock ETC for next bit
                LDA     PORTC            *portc-->A GET BIT...
                AND     #$01             *mask out bottom bit ie. I/O pin
                LSL     KEY              *shft lft all bits in key
                ADD     KEY              *(Acc)+(key)-->Acc
                STA     KEY              *Acc-->key GATHER KEYCODE
                BSET    1,PORTC          *clock ETC again for next bit
                JSR     DLY20            *
                BCLR    1,PORTC          *
                INCX                     *bit count
                CMPX    #$04             *check if we got all 4 bits of keycode
                BNE     SERIN            *not all 4 keycode bits get another
bit
                LDX     TMPX             *replace value TMPX --> X
                LDA     KEY              *keycode --> A
                JSR     DISPNUM          *display number
                JSR     DLY300
                RTS
```

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


*clear keypad buffer
BUFIN           CLRX
CLEARBUF        CLR     BUFFER,X        *O'S buffer Keypad or I/O
                INCX
                CMPX    #$08            *8 loc'n thus 16 digits in BCD form
                BNE     CLEARBUF

                LDX     #$FF            *-1 --> X
BUFSTART        JSR     ONEKEY          *get a character

* A=keycode     X=n character count

                CMP     #%1100          *check for STAR
                BEQ     RECALL          *
                CMP     #%1000          *check for NUMBER
                BEQ     STORE
                INCX                    *increment character count

*put digit in buffer if not * or # SYMBOL
                STX     TMPX            *x-->TMPX (count)
                ASRX                    *divide X by 2
*                                       *using carry to detect if number
                                        *should be put in lower or upper
*                                       *part of byte.
*                                       *since #'s are stored in packed BCD


                BCS     LOWERNIB        *branch if carry set

UPPERNIB        LSLA                    *logical shift left 4 bit
                LSLA                    *positions to shift keycode
                LSLA                   *to upper part of byte
                LSLA
                STA     BUFFER,X        *store number in buffer
*                                       *A-->buffer + X
                BRA     SKIPLOW
LOWERNIB        ADD     BUFFER,X        *buffer + X --> Acc
*                                       *concatenate data
                STA     BUFFER,X        *Acc --> buffer
*                                       *store Acc.A back in buffer
SKIPLOW         LDX     TMPX            *get back count TMPX-->X

                STA     PORTA           *debug cmd to check value stored

                BRA     BUFSTART        *

*NOTE: THERE IS NO END HERE THIS SIMPLY LOOPS
```

```
*SUBROUTINE TO RECALL A NUMBER FROM MEMORY
*
*INPUT    ?
*OUTPUT   ?
*A=star        X=the number of characters in the keyboard buffer -1

RECALL          ASRX                    *to establish position in bffer
*                                       *since stored in BCD format
                LDA     BUFFER,X        *contents of loc'n(buffer + X)-->Acc
*                                       *Indexed offset
                JSR     CONVERT         *ld A with last kybrd #
                ASLA                    *multiply by 8....shift 3 places
                ASLA
                ASLA
                ADD     #STORAGE        *storage + Acc.A to get
*                                       *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 info
                BSET    2,PORTC         *set DD on ETC high to tell
*                                       *ETC to use I/O pin as an I/P

                BSET    0,DDRC          *set bit 0 portc DDR on uC to output

                CLRX
OUTLOOP         STX     COUNT           *X --> count
                ASRX                    *X divide by 2 in order to
                BCS     LOWRNIB         *use carry to detect if number
*                                       *should be put in lower or upper
*                                       *part of byte by detecting odd or
*                                       *even values of X
UPPRNIB         TXA                     *x --> A
                ADD     TEMP            *(temp)+(Acc.A)-->Acc.A
                TAX                     *A-->X
                LDA     ,X              *use X as an offset into RAM
                LSRA                    *bring data down to
                LSRA                    *lower nibble
                LSRA
                LSRA
                BRA     LOWSKIP         *
LOWRNIB         TXA                     *to allow opperation on X(X->A)
                ADD     TEMP            *actual addr of character needed
                TAX                     *A->X
                LDA     ,X              *load the character needed X is offset
                AND     #$0F            *mask out uppr 4 bits
LOWSKIP         BEQ     DONELOOP        *result=0 then branch to doneloop
                                        *cause keycode is 0 which is used to
*                                       *detect the end.
```

```
                JSR     OUTDATA
                LDX     COUNT           *COUNT-->X
                INCX
                CMPX    #15             *max numbers possible
                BNE     OUTLOOP
DONELOOP        BCLR    2,PORTC         *set DD low again
                BCLR    0,DDRC          *CLR bit 0,set-up DDR to input on uC
                RTS


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

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

                ASLA                    *A * 8d
                ASLA                    *to find boundary of storage table
                ASLA                    *i.e where the number is to go in RAM


                ADD     #STORAGE        * STORAGE(58)+(Acc.A)-->Acc.A
                STA     TEMP            *Acc --> TEMP the actual addr to store
                                        at

                STA     PORTA           *DEBUG ADDRESS for testing purpose

                CLRX

COPYBUF         LDA     BUFFER,X        *load value of buffer into Acc.A
                INCX                    *count
                STX     TMPX            *X-->TEMPX ,preserve count
                LDX     TEMP            *TEMP-->X point to storage loc'n
                STA     ,X              *store Acc.A at storage loc'n
                INCX                    *increment count
                STX     TEMP            *X-->TEMP preserve count
                LDX     TMPX            *restore old count
                CPX     #$08            *test for end of buffer
                BNE     COPYBUF         *if not end. get next value from
                                         buffer
                RTS
```

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

```
*subroutine to convert BINARY keycode to BCD
*using keycode as an offset into the table
*INPUT  A=key   X=n
*OUTPUT A=BCD   X=n

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

*lets work on lower NIBBLE
                LDA     TEMP            *temp ---> A
                LSRA                    *shift upper nib to lower
                LSRA                    *
                LSRA                    *
                LSRA                    *

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

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

```
*subroutine to actually output one character to "ETC"
*subroutine to write data to controler
*to send DTMF out.
*Note: attempt for 300mS per digit approx.
*     :  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 keycode
                LSLA                    *in upper four bits
                LSLA
                LSLA
                LDX     #$04            *set count
OUTBIT          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

ONEOUT          BSET    0,PORTC         *puts a one out on I/O port ETC
*NOTE: no data is actually moving

MOREBITS        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                    *decrement count
                BNE     OUTBIT          *if counts not 0 get another bit
                BCLR    4,PORTC         *enable tone output on ETC
                JSR     DLY300
                BSET    4,PORTC         *set TO=high
                JSR     DLY300
                LDX     TMPX
                RTS

*delay SUBroutine for appproximatey 300mS

DLY300          STX     DLYX            *save X in DLYX
                STA     DLYA            *save Acc in RAM
                LDA     #192            *do outer loop 192 times
OUTLP           CLRX                    *X used as inner loop count
INNRLP          DECX                    *0-FF, FF-FE...1-0 256 LOOPS
                BNE     INNRLP          *6CYC*256*1uS/cyc = 1.536mS
                DECA                    *192-191, 191-190,...1-0
                BNE     OUTLP           *1545cyc*192*1uS/cyc= 296.640mS
                LDX     DLYX            *recover saved index value
                LDA     DLYA            *recover saved Acc value
                RTS                     *Return
```