Apple Turnover

by Anthony Rose

This month: Tephelone Dialler

Well, Microbition 84 (incorporating Cape Computers 84) has come and gone. I made the mistake of attempting to let the Apple demonstrate a synthesizer that the crowds decided they would like to demonstrate for themselves. No problem. Simply reset the computer. I had thought of that possibility, so the system automatically rebooted and reran the demo program after a reset. However, the crowds this year were more computer literate than previously and soon discovered that if they kept hitting control-reset, the system would eventually cease rebooting, so that they could get on with their monofingered masterpieces without interruption.

This was not good enough for several young cretins, who decided that the Apple would be much more useful demonstrating their games programs. I observed one. His technique was simple. Stand around looking interested, while checking that the machine's owner is not present. Then adopt the reset-till-dead approach, or, find the on/off switch. Now insert your own disk, reboot, and voila, puckman!

Having witnessed the above, I have started planning next year's exhibit. The reset and control keys will each have a thin metal membrane, suitably camouflaged, and wired to a 30kV generator, such as a car ignition coil, which will be activated when both keys are depressed.

The program itself will not be disk-based, but will reside in ROM. The disk drive, however, will be prominently displayed and will contain a dummy disk. A powerful magnet will reside just behind the drive door, ready to degauss any disk brought within centimetres of it. Also inside the drive will be a solenoid-controlled razor blade, mounted on the read/write head. With suitable driving software, the dummy disk will be recognised and left unscathed, but all others will emerge through a small chute beneath the drive. Finally, the machine will occasionally produce laser-gun sounds, guaranteed to attract the disk-toting types like flies. The screen will simply bear the message 'Try Me'.

Tephelone Dialler

Here is a simple circuit with attendant software which allows your computer to automatically dial the tephelone. You are probably wondering what a tephelone is. Well, as you know, it is illegal to connect things to your telephone line without GPO approval. Now our GPO would make talking into a telephone illegal if they thought they could get away with it, so diallers are right out. Hence the system presented here is not meant for use on the telephone line. Rather, it was developed for A.R.T.'s internal communication system, the Tephelone, which coincidentally happens to be 100% compatible with the GPO's Telephone system.

The hardware

The hardware is simple, and is constructed on a prototyping card. If you use a 6v or lower relay, it can be mounted instead on a DIP header and plugged into the game paddle socket and driven by a transistor controlled by one of the annunciator outputs. I have put it on a prototyping card because this will house other tephelone line interfaces as well, such as a ring detector, which allows the computer to detect the tephelone ring voltages, and then anwser the call. This and other circuits will be presented in future issues.

The protocol

The tephelone system works as follows: With the tephelone on the hook (down), there is no load on the line, other than a capacitor, which the GPO uses on its rival telephone system to determine the number of phones you have connected. I could get in trouble for saying that the way to connect extra tephelones without anyone knowing is to disconnect the capacitor in them, but as I wish no trouble I will not.

When the tephelone is lifted off the hook, a 300 ohm or less resistance is placed across the line, which then drops from 50v to less than 11v, depending on the resistance. When a digit is dialled, the relay which connects the resistor across the line opens and closes, each time pulsing the line. Each pulse lasts 100 ms +/- 10%, with the line shorted for 30-37% of the total period. These pulses are repeated to make up the digit concerned, four for '4', for instance. Following this there is an 800 ms interdigit pause, before the next digit is dialled. During the duration of the call, the tephelone applies a 900 ohm impedance across the line. On completion of the call, the tephelone is replaced on the hook, the relay opens, disconnecting the resistor and allowing the line to rise to 50v again. On a tephelone, the 900 ohm impedance is a transformer which couples the voice signals onto the line.

Thus the tephelone interface requires relays for both 300 ohm and 900 ohm resistors. However, as the dialler is only connected while pulsing the line, the 900 ohm setup is not required. If a 300 ohm resistor is connected in series with the relay then line specifications will be observed while still allowing the tephelone access to the line, albeit muted. In practice, any resistance value up to at least 1k is sufficient to capture the line and dial.

The software

This is written in Forth, the main reason being that it is desirable to be able to create a dictionary of frequently used tephelone numbers, and Forth provides the perfect way for storing them without having to write a program to do so. For instance, supposing the word DIAL took a number and dialled it, then I could define

```
: MM-BBS 4 5 7 7 5 0 DIAL ;
```

and whenever I typed MM-BBS the system would dial MM Computers' Bulletin Board. Later I could execute the command SAVE, and the Forth system, including the numbers I had defined, would be saved to disk. Had I done the same in BASIC, I would have had to write the equivalent of a database, with routines to enter, recall, load and save numbers.

Hopefully it will also persuade more people to have a look at Forth. Incidentally, Robert Desbiens is running a series on Forth, so get in touch with him at 237581 as soon as possible if you are interested.

The program is easy to follow (if you understand Forth). The first two words, UP and DOWN, effectively lift and replace the tephelone receiver by turning the relay on and off. These words are correct for an interface board plugged into slot five. The value C0D0 can be changed to C0X0 for other slots, where X=slot number + eight, in hex. Next, the variable STOREVAR is created, and allotted hex 20 bytes. This variable temporarily stores the number to be dialled. The reason that the number is not pulled directly from the stack is that it would then be in reverse order, since the first digit is now deep in the stack. STORE pulls the digits off the stack and stores then in STOREVAR. A problem arose deciding how many digits to pull off the stack, as tephelone numbers range from four to sixteen digits. Initially I had to enter the number of digits in the number, after entering the number (e.g. 1 2 3 4 5 6 6), but this was quite clumsy, so the routine now empties the stack, thus accepting any length number.

The first line of STORE subtracts the stack pointer from the stack limit address, divides by two (a number is sixteen bits, or two bytes) and finally subtracts one. This is the number of items on the stack, and is then stored in the first location of STOREVAR, after which the digits are stored in dialling order descending from hex 16 bytes into STOREVAR.

DELAY is a fixed delay of 33 ms, used by the pulsing routine. If you are not using an Apple, or your machine runs at a different speed, then change the loop value in DELAY to obtain a 33 ms period.

This is simply done by

```
DECIMAL : WAIT 300 0 DO DELAY LOOP ;
```

This should take ten seconds +/- 1/2 second with DELAY correct.

DIALDIGIT waits 800 ms before pulsing the line N times, where N is the value of the digit to be dialled, and is on the stack. The highest-level word, DIAL, picks up the phone, waits two seconds, which, when added to DIALDIGITS' wait of just under one second, gives a three second delay, in accordance with GPO, sorry A.R.T, specifications, during which time the dial tone should appear. DIALDIGIT is then called for each digit in the number, with the number being dialled displayed. When the number has been dialled, the message 'Please pick up phone' is issued. You have two seconds in which to do so, as after this time the computer will disconnect the resistor, leaving your phone on-line with either a ringing or engaged tone, if you have lifted the receiver, or it will terminate the call if you have not.

As mentioned earlier, future articles will expand the system by incorporating auto-answer circuitry and other features. The design presented is a building-block for you to expand on, for instance making it more user-friendly by defining a new class of word DIALS so that you can type

```
4 5 7 7 5 0 DIALS MM-BBS
```

rather than the present

```
: MM-BBS 4 5 7 7 5 0 DIAL ;
```

(note the spaces between digits).

Also, if the name is unknown, you could be prompted to enter the number for it, and a new entry could then be created. Finally, to use the system on another (non-Apple) computer, simply change the addresses in UP and DOWN and the timing loop in DELAY. Happy Frothing...

```
( TEPHELONE DIALLING ROUTINES - PAGE 1 )

HEX

: UP   FF C0D0 C! ;
: DOWN 3F C0D0 C! ;

0 VARIABLE STOREVAR 20 ALLOT

: STORE  S0 @ SP@  - 1 - 2 /
         DUP STOREVAR ! 0 DO
         STOREVAR 16 I - + C! LOOP ;

: DELAY  ( WAIT 33  MS )
         128 0 DO LOOP ;

: DIALDIGIT  ( WAIT 800 MS )
         18 0 DO DELAY LOOP

            0 DO DOWN DELAY DELAY
                 UP  DELAY LOOP ;
-->
```

```
( TEPHELONE DIALLING ROUTINES - PAGE 2 )

: DIAL  ." DIALLING... "
        STORE UP

       ( WAIT TWO SECS BEFORE DIALLING )
       40 0 DO DELAY LOOP

       ( DIAL DIGITS )
        -1 STOREVAR @ 1 - DO
          STOREVAR 16 I - + C@ DUP
          DUP . BS  0= IF
            DROP 0A THEN
          DIALDIGIT -1 +LOOP CR BUZZ

       ." PLEASE PICK UP PHONE " CR
       40 0 DO DELAY LOOP DOWN ;

: MM-BBS  4 5 7 7 5 0 DIAL ;
: TRS-BBS 2 1 5 3 6 3 DIAL ;

DECIMAL
```