sections. It is also a good subroutine to have available for use in other programs.

The subroutine needs two data items from the calling program — namely, the addresses of the two strings to be compared. Since the subroutine has to step through the strings byte by byte, it is best that these two values are passed to the index registers, X and Y, where they will be needed. The subroutine must also pass back two values, one to indicate whether or not a match has been found, and the other to show the address itself in the case of a match.

TRUE OR FALSE

It is possible to pass a Boolean parameter (true or false) using one of the condition code register flags, but this requires an exact knowledge of the effect of each instruction on the flags. In our program we will pass values back to the calling routine as either $00 (all zeros) if the match is found, or $FF (all ones) if it is not.

To make the subroutine more generally useful, we won't pass back the actual address for a found match, but will instead leave the X register pointing to the address where the required address can be found. This has the additional advantage that the X register, by stepping byte by byte through the string, should end up containing this information automatically, anyway.

One final point: our program contains one new 6809 instruction. TST (TeST) has no effect on any register, but simply sets the flags according to the current value of the named register.

```
TABLE   EQU   $10
STRNG   EQU   $12
ADDRS   EQU   $14
        ORG   $1000
        LDU   TABLE
LOOP1   LDA   ,U
        BEQ   NTFND1
        ADDA  #3
        TFR   U,X
        LEAU  A,U
        LDY   STRNG
        BSR   COMPAR
        TSTA
        BEQ   FOUND1
        BRA   LOOP1
FOUND1  LDD   ,X
        BRA   FINSH1
NTFND1  LDD   #0
FINSH1  STD   ADDRS
        SWI
COMPAR  LDB   ,X+
        CMPB  ,Y+
        BNE   NOTEQ
LOOP2   LDA   ,X+
        CMPA  ,Y+
        BNE   NOTEQ
        DECB
        BGT   LOOP2
        CLRA
        BRA   FINSH2
NOTEQ   LDA   #$FF
FINSH2  RTS
        END
```

Start of main program

Start of table in U

Get length byte

GOTO end of table if zero

Add one for length byte, two for address to length of string, giving length of table entry

Put start of string in table into X

Set U to point to next entry

Y points to start of search string

Compare the two strings

See if they are equal

If they are equal then GOTO FOUND1

ELSE get the next table entry

If found, X should be pointing to the address we want

Address will be zero if not found

Save what we came for

End of main program

Start of subroutine

Get length bytes and point X and Y to the first characters

If the strings are not the same length then GOTO NOTEQ

Get next character from table string

Compare it with next character from search string

Stop if they are not identical

Else take one from position pointer

Get next character

Make A zero to show that the strings are identical

Ones if not equal

Back to calling program

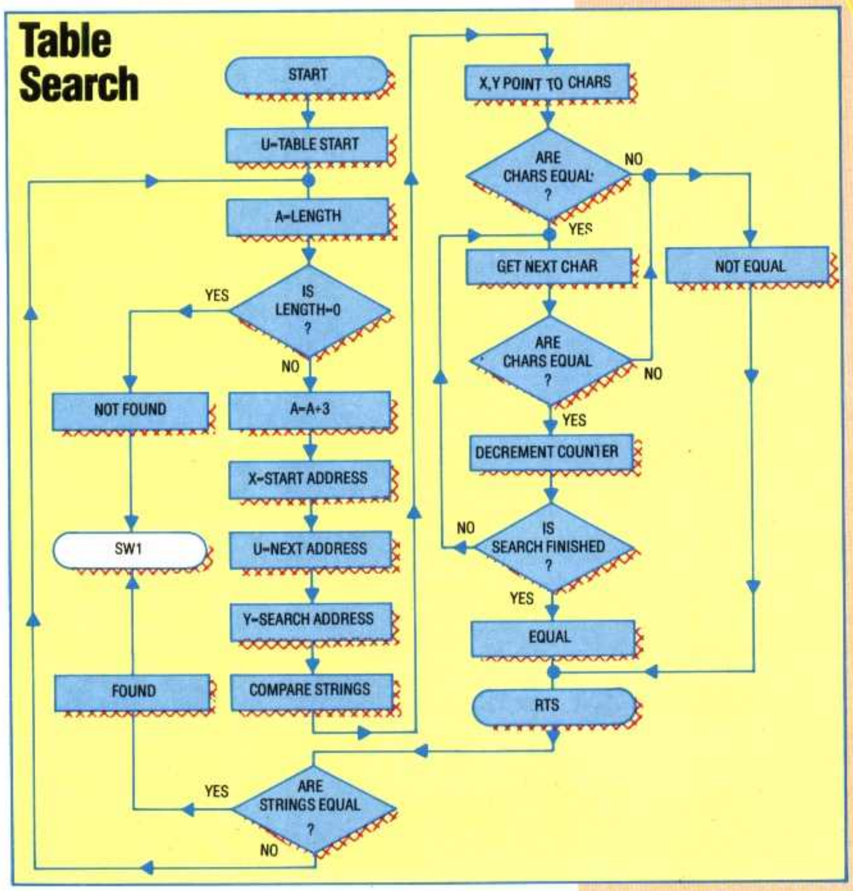
Table Search
START
U=TABLE START
A=LENGTH
IS LENGTH=0 ?
YES
NO
NOT FOUND
A=A+3
X=START ADDRESS
SW1
U=NEXT ADDRESS
Y=SEARCH ADDRESS
FOUND
COMPARE STRINGS
ARE STRINGS EQUAL ?
YES
NO
X,Y POINT TO CHARS
ARE CHARS EQUAL ?
NO
YES
GET NEXT CHAR
NOT EQUAL
ARE CHARS EQUAL ?
NO
YES
DECREMENT COUNTER
IS SEARCH FINISHED ?
NO
YES
EQUAL
RTS