minimum number of clock cycles — that the instruction takes; so the actual time taken will also depend on the clock rate. A common clock rate for 6809 systems is one MHz — one million cycles per second. Thus, each clock cycle takes one millionth of a second. The straightforward instruction:

```
LDA DATITM
```

using a 16-bit address takes five cycles, and so executes in five millionths of a second. The instruction:

```
PSHS PC, B, CC
```

takes five cycles plus one cycle for each byte that is pushed onto the stack; in this case a total of nine cycles (remember that the PC is two bytes).

If a system does not include a real-time clock then the only way to measure elapsed time is by means of a software delay routine. This executes a sequence of instructions whose individual times have been chosen so that the sum gives the required interval. Such intervals are usually measured in milliseconds (thousandths of a second), so there is no need to be too exact — the odd millionth of a second will not matter. Assuming a clock rate of one MHz, the Software Delay routine we give here will produce delays in the range 1 to 255 milliseconds: the exact number of milliseconds (ms) being passed as a parameter in A. The notation (A) means the contents of accumulator A.

The calculation to find the constant COUNT can be expressed as follows:

Instruction	Number Of Clock Cycles	Number Of Times Executed	Time Taken (Clock Cycles)
PSHS B,CC	7	1	7
LDB #COUNT	2	(A)	(A) * 2
DECB	2	(A) * COUNT	(A) * COUNT * 2
BNE LOOP2	3	(A) * COUNT	(A) * COUNT * 3
DECA	2	(A)	(A) * 2
BNE LOOP1	3	(A)	(A) * 3
PULS PC,B,CC	9	1	9

This gives a total of (A) * (7 + 5 * COUNT) + 16 clock cycles. To make the calculation easier, we will ignore the 16. At a clock rate of one MHz, there are 1000 clock cycles in a millisecond so the total time should be (A) * 1000 clock cycles.

$$(A) * (7 + 5 * COUNT) = (A) * 1000$$
$$(7 + 5 * COUNT) = 1000$$
$$5 * COUNT = 973$$
$$COUNT = 195 \text{ (to the nearest integer)}$$

It is quite feasible to make more accurate delays, and to use the 16-bit registers for a greater range, but the principle of decrementing a register a fixed number of times remains the same.

Terminal Emulation Routine

```
ESCAPE  EQU   27
SPACE   EQU   32            (Space is ASCII 32)
OUTCH   EQU                 Enter operating system
                            address here
        ORG   $1000
CTABLE  RMB   32            Table of control characters
ETABLE  RMB   128           Table of Escape characters
EFLAG   FCB   0             Flag to indicate whether last
                            character was an Escape
DISPCH  PSHS  X             Save X
        TST   EFLAG, PCR    Check if last character was an
                            Escape
        BEQ   DISP1         If not an Escape, then go to
                            DISP1
        LEAX  ETABLE, PCR   Else get address of ETABLE in X
        LDA   A,X           Get replacement character using
                            the original character in A as the
                            offset
        CLR   EFLAG,PCR     Reset EFLAG
        BRA   FINISH
DISP1   CMPA  SPACE         Check if control character
        BGE   FINISH        If not control character, then
                            goto FINISH
        CMPA  ESCAPE        Else check if Escape
        BEQ   ESCCH         If it is Escape, then goto ESCCH
        LEAX  CTABLE,PCR    Get address of CTABLE in X
        LDA   A,X           Get replacement character using
                            the original character in A as
                            offset
        BRA   FINISH
ESCCH   INC   EFLAG,PCR     Set EFLAG to indicate character
                            was Escape
FINISH  PULS  X             Restore X
        JMP   OUTCH         Display character in A
        END                 Note that the RTS at the end of
                            OUTCH will return control from
                            here to the calling program
```

Software Delay Routine

Subroutine to delay (A) milliseconds

```
COUNT   EQU   195           See calculation
        ORG   $1000
DELAY   PSHS  B,CC          Save the other two registers
                            affected
LOOP1   LDB   #COUNT        Count for 1 ms
LOOP2   DECB                Keep decrementing
        BNE   LOOP2         Until B reaches zero
        DECA                Decrement A after each ms
        BNE   LOOP1         Until A reaches zero
        PULS  PC,B,CC       Return
```