FREE TRANSFER

As the tutorial section of our 6809 Assembly language course draws to a close, we begin to take a more general look at the techniques of machine code programming. Our first topics are relocatable code, instruction lengths and timing routines.

A program that is written using *relocatable*, or *position-independent*, code can be placed at any position in memory and run without any changes having to be made. This is particularly important in multi-tasking or multi-user systems where several programs may be loaded into memory at the same time, and in order to ensure efficient use is made of memory space, the operating system must be able to load them at the most convenient place. Even in simpler, single-user systems it is usually important to be able to maintain subroutine libraries and to construct a program out of self-contained modules, in which case the position of a routine in memory may vary.

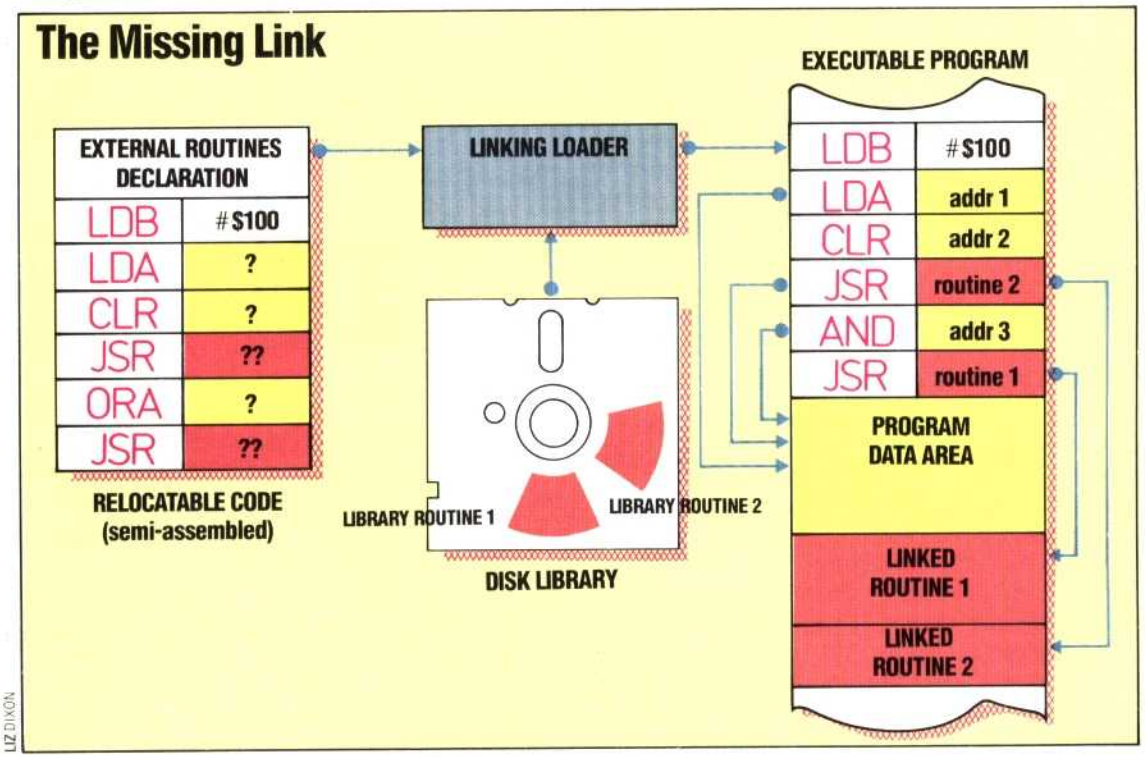
Most processors deal with this by using what is known as a *linking loader*. The assembler produces relocatable code, which leaves out all references to actual addresses in memory; it is the job of the linking loader to insert the addresses as it loads the program into memory. Since it is the loader itself that handles the addresses, it is straightforward to ensure that transfers of control between different modules are handled correctly. In this way, sections of code can be written in different languages that all compile or assemble to the same relocatable code; thus, for example, PASCAL programs can call FORTRAN library routines. This approach can also be used with the 6809, and indeed it is necessary if modular construction is used. The 6809 makes the process a lot easier by allowing fully relocatable code to be written directly, so there is no need for the extra stage of inserting addresses.

The key to writing relocatable code is to refer to all addresses by means of an *offset* from the program counter (PC). There are two ways in which a program can use an address: as data and as the destination for a transfer of control. Branch instructions (BRA, BSR, etc.) calculate their destinations as offsets from the PC and should be used for all transfers of control within the user program. The absolute transfer instructions (JMP and JSR) should be used only for destinations that will always be at the same place in memory, such as operating system routines.

The more difficult task is to make all the references to a data position independent, and the 6809 achieves this by allowing the PC to be used for indexing. The instruction:

```
LDA OFFSET,PC
```

will add the (signed) offset to the current value of



Linking Loader
In large systems, machine code programs are actually loaded into memory by the Linking Loader. This operating system utility takes the semi-assembled machine code (containing no absolute addresses) from the assembler, and determines the best ORG address for it from the current state of the system. It uses this address to replace the symbolic addresses that the assembler left in the program with absolute addresses, and then links to the program any library routines requested by the programmer; these routines are loaded from the library disk and attached to the program. Their absolute call addresses can then replace the symbolic addresses in the program. Finally, the Loader passes the complete program to the operating system for execution