## THE MAIN MODULE

**Data:**

**Prompt** for command entry is ASCII character '>'
**Command-Offset** into table of command characters and Jump-Table

**Process:**

Save-Values
Set-Up-Jump-Table
Set-Up-Interrupt
Get Start-Address
Repeat
  Display Prompt
  Get-Command
  Do-Command
Indefinitely

That completes our debugger program. At the moment it is rather fragmented, but that is a consequence of modular construction. At this point we can optimise the code if we wish by looking for short cuts. For example, you may find that you have had to move a lot of values around to make sure that they are in the right registers for a subroutine, so you might make savings by redefining register usage. This is not really advisable unless memory space is very restricted. We have defined the same data areas in a number of different places, as they are required. There are two ways in which you might handle data areas in the complete program: you can retain the data with the module that uses it, which is theoretically the best option; or you can define all the data together at the start of the program, which has real advantages if you ever want to use a disassembler (or even a debugger) on the program.

The debugger should be loaded into any spare memory not occupied or used by the program to be debugged. It is entered by making a jump to the DEBUG entry point, so it is necessary to know this address before you start.

In the later part of this 6809 machine code series, we have tried to show the best way in which programs are developed, illustrated with a variety of techniques. Therefore, the design of our debugger program is not necessarily the most efficient way to do this particular job. If you have followed everything, however, then you should have a fairly comprehensive understanding of Assembly language programming in general, and 6809 Assembly in particular.

### Set-Up-Jump-Table

```
JTABLE  RMB   16          Space for 8 two-byte addresses
SETUPJ  LEAY  JTABLE,PCR  Base address of table in Y
        LEAX  CMDB,PCR    Start address of CMDB subroutine
        STX   ,Y++        Store it in table
        LEAX  CMDU,PCR    Start address of CMDU subroutine
        STX   ,Y++        Store it in table
        LEAX  CMDD,PCR    Start address of CMDD subroutine
        STX   ,Y++        Store it in table
        LEAX  CMDS,PCR    Start address of CMDS subroutine
        STX   ,Y++        Store it in table
        LEAX  CMDG,PCR    Start address of CMDG subroutine
        STX   ,Y++        Store it in table
        LEAX  CMDR,PCR    Start address of CMDR subroutine
        STX   ,Y++        Store it in table
        LEAX  CMDM,PCR    Start address of CMDM subroutine
        STX   ,Y++        Store it in table
        LEAX  CMDQ,PCR    Start address of CMDQ subroutine
        STX   ,Y++        Store it in table
```

This is the actual jump to the subroutine. We assume that X contains the address of JTABLE and B the offset

```
DOCMD   JMP   [B,X]
```

### Save-Values

```
SAVED   RMB   5           Five bytes to be saved
SAVEIT  LEAX  SAVED,PCR   Get address to save in
        TFR   S,D         Move S to D
        ADDD  #2          Add two to take care of the return address
        STD   ,X++        Save it
        LDY   $FFFA       Get Interrupt vector address
        LDA   ,Y+         Get first byte to be saved
        STA   ,X+         Save it
        LDD   ,Y          Get other two bytes
        STD   ,X          Save them
        RTS
```

### Command Q

```
CMDQ    LEAX  SAVED,PCR   Address of Saved
        LDY   $FFFA       SWI-Vector
        LDA   2,X         First of three bytes
        STA   ,Y+         Restored
        LDD   3,X         Other two bytes
        STD   ,Y          Restored
        LDS   ,X          Saved Stack-Pointer
        JMP   [$FFFE]     Indirect jump via reset vector
```

### Main Module

```
PROMPT  FCB   '>
STACKP  RMB   2           Stack-Pointer for Display-Registers
DEBUG   BSR   SAVEIT      Save-Values
        BSR   SETUPJ      Set-Up-Jump-Table
        BSR   INIT        Set-Up-Interrupt and Get Start-Address
ENTRY   STS   STACKP,PCR  Save Stack-Pointer
        LEAX  JTABLE,PCR
REPT02  LDA   PROMPT,PCR  Get prompt and
        BSR   OUTCH       display it
        BSR   GETCOM      Get Command
        LSLB              Double offset for 16-bit table
        BSR   DOCMD       Obey Command
        BRA   REPT02      Next Command
```