

OVERVIEW

This application note discusses the difference between absolute and relocatable programs and shows you how to write relocatable assembly programs for the Intel 8xC251Sx and 8x930Ax. If you need information about using RISM or the Intel USB evaluation board, refer to the following:

- *Keil Application Note 109: Using the Intel 8x930Ax and 8x930Hx with RISM251.*
- *Keil Application Note 110: Solving RISM, USB, and dScope Communication Problems.*

ABSOLUTE PROGRAMS

Absolute assembly programs are those programs that locate important code objects at fixed memory locations. Such objects include the reset and interrupt vectors as shown in the following example.

```
ORG    0FF:0000h
      LJMP RESET

ORG    0FF:0003h
      LJMP INT0

ORG    0FF:000Bh
      LJMP INT1

ORG    0FF:1000h
RESET: NOP
INT0:  NOP
INT1:  NOP

END
```

Initially, this appears to be a good way to locate objects at fixed memory locations. However, if you need to relocate the reset vector and interrupt vectors for a monitor program like RISM, you must modify the source code as follows:

```
ORG    00:4000h
      LJMP RESET

ORG    00:4003h
      LJMP INT0

ORG    00:400Bh
      LJMP INT1

ORG    00:5000h
RESET: NOP
INT0:  NOP
INT1:  NOP

END
```

You may use conditional assembly code to avoid modifying your program. However, this is still not the best solution.

RELOCATABLE PROGRAMS

A much easier way of relocating programs is to use the features of the linker to link your program and locate it at a target address. The following example shows how to write code that is easily relocated by the linker.

```

CSEG    AT 0000h        ; define an absolute CODE segment at offset 00h
        LJMP RESET

CSEG    AT 0003h        ; define an absolute CODE segment at offset 03h
        LJMP INTO

CSEG    AT 000Bh       ; define an absolute CODE segment at offset 0Bh
        LJMP INT1

MYSEG   SEGMENT CODE   ; define a relocatable CODE segment
        RSEG          MYSEG
RESET:  NOP
INT0:   NOP
INT1:   NOP
END

```

By default, the Keil L251 linker locates segments with the CODE classification at 0FF0000h. After linking the above example, the reset and interrupt vectors are located at 0FF0000h + 0000h.

USING THE LINKER TO RELOCATE PROGRAMS

The Keil L251 linker makes several assumptions about where certain segments are located. These assumptions are listed in the following table.

| Memory Class | Default Address Range | Maximum Length |
|--------------|-----------------------|----------------|
| BIT | 00:0020h-00:002Fh | 16 Bytes |
| CODE | FF:0000h-FF:FFFFh | 64 Kbytes |
| CONST | FF:0000h-FF:FFFFh | 64 Kbytes |
| DATA | 00:0000h-00:007Fh | 128 Bytes |
| EBIT | 00:0020h-00:007Fh | 96 Bytes |
| ECODE | FE:0000h-FF:FFFFh | 16 Mbytes |
| EDATA | 00:0000h-00:FFFFh | 64 Kbytes |
| HCONST | FE:0000h-FF:FFFFh | 16 Mbytes |
| HDATA | 00:0000h-FF:FFFFh | 16 Mbytes |
| IDATA | 00:0000h-00:00FFh | 256 Bytes |
| NCONST | 00:8000h-00:FFFFh | 64 Kbytes |
| PDATA | 01:??00h-01:??FFh | 256 Bytes |
| XDATA | 01:0000h-01:FFFFh | 64 Kbytes |

The default settings above are correct for most applications where you make code to go in an EPROM for your target hardware. However, in situations where you debug using a target monitor like RISM or MON251, you must relocate the reset vector, the interrupt vector, and your target program to the memory area reserved by the monitor.

The following table shows the memory area reserved for your program by the different monitors.

| Monitor | Program Location |
|---------------|-------------------|
| Intel RISM251 | 004000h - 00FFFFh |
| Keil MON251 | 004000h - 00FFFFh |

If you use the relocatable programming techniques described above, this can be done using only the L251 linker. You do not have to modify any of your source code.

Using the MS-DOS Command Line

If you link using the command-line tools, you may specify the following command to link and locate your program to 0x004000 (the program address space reserved by RISM).

```
L251 MYPROG.OBJ CL($0x004000,0x004000)
```

The first \$0x004000 tells the linker that the program counter starts at 0x004000. This is only used by the debugger to properly change the program counter when you load a program for debugging (by default, programs start at 0xFF0000).

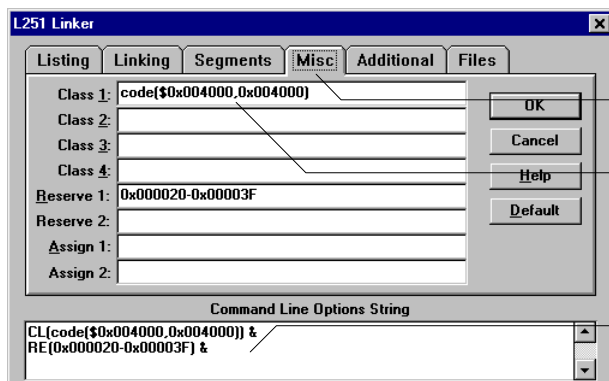
The second 0x004000 tells the linker to offset all absolute CODE segments by 0x004000. In other words, code generated by the following assembly code...

```
CSEG at 0
NOP
```

is located starting at 0x004000 rather than at 0xFF0000.

Using the µVision IDE

If you build and link your programs using the µVision Integrated Development Environment, specify the following information in the L251 Linker Options dialog box.



Class options are located under the Misc tab.

Change the class memory assignment here.

The resulting command-line options are displayed here.

As with the MS-DOS command line, `$0x004000` tells the linker that the program counter starts at `0x004000` and `0x004000` tells the linker to offset all absolute **CODE** segments by `0x004000`.

CONCLUSION

Using relocatable programming techniques makes changing program location easy when you use the linker to relocate your target programs.

Copyright © 1997 Keil Software, Inc. All rights reserved.

In the USA:
Keil Software, Inc.
16990 Dallas Parkway, Suite 120
Dallas, TX 75248-1903
USA

Sales: 800-348-8051
Phone: 972-735-8052
FAX: 972-735-8055

E-mail: sales.us@keil.com
support.us@keil.com

Internet: <http://www.keil.com/>

In Europe:
Keil Elektronik GmbH
Bretonischer Ring 15
D-85630 Grasbrunn b. Munchen
Germany

Phone: (49) (089) 45 60 40 - 0
FAX: (49) (089) 46 81 62

E-mail: sales.intl@keil.com
support.intl@keil.com