*C COMPILERS • REAL-TIME OS • SIMULATORS • EDUCATION • EVALUATION BOARDS*
16990 Dallas Parkway • Suite 120 • Dallas, Texas • 75248 • 800-348-8051

# Porting 8051 C Programs to the 251 *Application Note 117*

November 12, 1997, Munich, Germany

by Reinhard Keil, Keil Elektronik GmbH rk@keil.com ++49 89 456040-13

The source code for this application note is located in the file named 117.zip on this CD.

This document shows how to port a program which was written originally for the Keil C51 compiler to the Keil C251 compiler version 2. If you simply re-compile an existing C51 program with C251 Version 2, the program often grows in code size. The reason is that most C51 programs are using memory types to place variables in C251 memory spaces. This is illustrated using a short program:

```
 1          /*
 2           *   This is a old existing 8051 program
 3           */
 4          char data  dval;
 5          int  idata ival;
 6          int  xdata xval;
 7          char pdata pval;
 8          char code  cval = 1;
 9          int  xdata * idata xp;
10
11          void main (void)  {
12   1        dval = cval;
13   1        xp   = &xval;
14   1        pval = 5;
15   1        for (ival = 0; ival < 10; ival++)  {
16   2          *xp = 0;
17   2        }
18   1      }
```

**EXAMPLE C SOURCE CODE**

If you compile with Keil C51 Version 5, the total code size is 57 Bytes. If you re-compile this program with the Keil C251 Version 2 compiler, you end up with a code size of 74 Bytes. This is bigger since you are using the new 251 instruction set in source mode. You might expect these instructions should generate better and therefore smaller assembler code.

The reason for this behavior is that Keil C251 Version 2 still supports all addressing modes of the 8051 CPU and therefore maps the memory spaces idata, pdata, code and xdata to less effective 251 instructions. If you are replacing these memory spaces by the new C251 near memory space, so can get the optimum performance out of your 251 CPU. One way to do that is shown in the following example:

We are replacing the C51 memory types in our existing C51 source module by defines. These defines are then replaced in the header file CONV51.H into optimum C251 memory types. In a typical 8051 application this modification can be done with global text replacements on all files.

```
 1
 2          #include "conv51.h"  // convert 51 mspaces to 251 mspaces
 3
 4          /*
 5           *   This is the modified 8051 program
 6           */
 7          char DRAM   dval;
 8          int  IRAM   ival;
 9          int  XRAM   xval;
10          char PRAM   pval;
11          char CROM   cval = 1;
12          int  XRAM * IRAM xp;
13
14          void main (void)  {
15   1        dval = cval;
16   1        xp   = &xval;
17   1        pval = 5;
18   1        for (ival = 0; ival < 10; ival++)  {
19   2          *xp = 0;
20   2        }
21   1      }
```

## MODIFIED EXAMPLE FOR C251 RE-COMPILING

The content of the CONV51.H header file is shown below:

```
/* This is the header file CONV51.H  */
#ifdef __C251__     // __C251__ is a define of C251
#define DRAM  data  // data can be mapped directly to the 251 data space
#define IRAM  near  // idata uses MOV Ri, use 251 near instead
#define XRAM  near  // xdata uses MOVX or MOV @DR56, use 251 near instead
#define PRAM  near  // pdata uses MOVX Ri, use 251 near instead
#define CROM  const // code uses MOVC A, use 251 const instead
#else
#define DRAM  data  // if you compile with C51 the memory spaces are
#define IRAM  idata // mapped to the previous definitions
#define XRAM  xdata
#define PRAM  pdata
#define CROM  code
#endif
```

After these modifications, the C251 compiler shows much better results than the C51 compiler. The code size of this small application has been reduced to 49 Bytes. In typical applications you can reduce the code size by 20 to 30% depending on the usage of the C51 memory types. For example, declaring loop variables as *char* are more efficient with an 8051 but using an *int* is better with the 251.

## C251 ASSEMBLER CODE

```
;       FUNCTION main (BEGIN)
                                                ; SOURCE LINE # 14
                                                ; SOURCE LINE # 15
000000 7EB30000    R  MOV      R11,cval         ; A=R11
000004 F500        R  MOV      dval,A           ; A=R11
                                                ; SOURCE LINE # 16
000006 7E340000    R  MOV      WR6,#WORD0 xval
00000A 7A370000    R  MOV      xp,WR6
                                                ; SOURCE LINE # 17
00000E 7405           MOV      A,#05H           ; A=R11
000010 7AB30000    R  MOV      pval,R11         ; A=R11
                                                ; SOURCE LINE # 18
000014 6D33           XRL      WR6,WR6
000016 7A370000    R  MOV      ival,WR6
              ?C0004:
                                                ; SOURCE LINE # 19
00001A 6D33           XRL      WR6,WR6
00001C 7A370000    R  MOV      xval,WR6
                                                ; SOURCE LINE # 20
000020 7E370000    R  MOV      WR6,ival
000024 0B34           INC      WR6,#01H
000026 7A370000    R  MOV      ival,WR6
00002A BE34000A       CMP      WR6,#0AH
00002E 78EA           JNE      ?C0004
                                                ; SOURCE LINE # 21
000030 22             RET
;       FUNCTION main (END)
```

---

*Keil Software Inc.*
*16990 Dallas Parkway • Suite 120 • Dallas, Texas • 75248 • 800-348-8051*