SDCC Compiler User Guide

13th July 2001

Contents

1 INTRODUCTIOT0(OODUCTIOT0(OODUCTIOT0(OT0(O(1.2T0(OODUOpen)-250(Source)]TJET0 -3.985 3

the default directory where include, library and documention files are stored is no in /usr/local/share

char type parameters to vararg functions are casted to int unless explicitly casted, e.g.:
```
char a=3;
printf ("%d %c\n", a, (char)a);
```
will push a as an int and as a char resp.

option –regextend has been removed

option –noreparms has been removed

<*pending: more incompatibilities?*>

1.5 System Requirements

What do you need before you start installation of SDCC? A computer, and a desire to compute. The preferred method of installation is to compile SDCC from source using GNU gcc and make. For Windows some pre-compiled binary distributions are available for your convenience. You should have some experience with command line tools and compiler use.

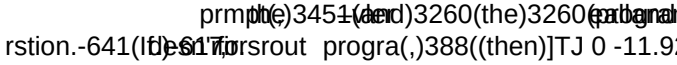
1.6 Other Resources

The SDCC home page at `http://sdcc.sourceforge.net/` is a great place to find distribution sets. You can also find links to the user mailing lists that offer help or discuss SDCC with other SDCC users. Web links to other SDCC related sites can also be found here. This document can be found in the DOC directory of the source

2.2.2 Windows Install Using Cygwin

1. Download and install the cygwin package from the redhat site `http://sources.redhat.com/cygwin/`. Currently, this involved downloading a small install program which then automates downloading and installing selected parts of the package (a large 80M byte sized dowload for the whole thing).
2. Bring up a Unix/Bash command line terminal from the Cygwin menu.
3. Follow the instructions in the preceding Linux/Unix installation section.

2.3 Testing out the SDCC Compiler



and header files to /usr/local/share/sdcc/lib and /usr/local/share/sdcc/include.

2.5 Additional Information for Windows Users

<pending: is this up to date?>

2.8.5 sdcdb - Source Level Debugger

Alternatively, *foomain.c* can be separately compiled as well:

```
sdcc -c foomain.c
sdcc foomain.rel foo1.rel foo2.rel
```

The file containing the *main()* function MUST be the FIRST file specified in the command 6ine, since the 6inkage editor processes file in the order they are presented to it.

3.1.3 Projects witi.9111d(edctialed)-24Librariesel

–code-loc<Value> The start location of the code segment, default value 0. Note when this option is used the interrupt vector table is also relocated to the given address. The value entered can be in Hexadecimal or Decimal format, e.g.: –code-loc 0x8000 or –code-loc 32768.

–stack-loc

the lower 1K of the internal RAM, which is mapped to 0x400000. Note

-E Run only the C preprocessor. Preprocess all the C source files specified

–int-long-reent

The basic blocks at this stage ordered in the depth first number, so they may not be in sequence of execution.

–dumpgcse Will create a dump of iCode's, after global subexpression elimination, into a file named <*source filename*>.*dumpgcse.*

–dumpdeadcode Will create a dump of iCode's, after deadcode elimination, into a filenamed4.916 -11.955 Td[(nam

```
ram */
unsigned char _data  *ucdp ; /* pointer to data in internal
ram */
unsigned char _code  *uccp ; /* pointer to data in R/O code
space */
unsigned char _idata *uccp;  /* pointer to upper 128 bytes
of ram */
```

All unqualified pointers are treated as 3-byte (4-byte for the ds390) *generic* pointers. These type of pointers can also to be expliceaiP declared.

```
unsigned char _generic *ucgp;
```

The highest order byte of the *generic*

a number less than 100 (which implies a limit of utmost 100 inline assembler labels *p-21]TJ -330.43 -11.955 Td[(fun*

are all developed in ANSI-C to facilitate porting to other MCUs, although some model specific assembler optimations are used. The following files contain the described rou-

Note: the dead stores created by this copy propagation will be eliminated by dead-code elimination.

4.1.4 Loop Optimizations

The more expensive multiplication is changed to a less expensive addition.

4.1.5 Loop Reversing

This optimization is done to reduce the overhead of checking loop boundaries for every iteration. Some simple loops can be reversed and implemented using a “decrement and jump if not zero” instrub1instru1(w490.4s0s263(zeies)-2(zeuce)-2w49folloe)-2wd zecr.955optimiw49onezede955m

4.1.7 'switch' Statements

SDCC changes switch statements to jump tables when the following conditions are true.

The case labels are in numerical sequence, the labels need not be in order, and the starting number need not be one or zero.

```
switch(i) {                          switch (i) {
case 4:...                                       case 1:
...
case 5:...                                       case 2:
...
case 3:...                                       case 3:
...
case 6:...                                       case 4:
...
}                                    }
```

Both the above switch statements will be implemented using a jump-table.

The number of case labels is at least three, since it takes two conditional statements to handle the boundary conditions.

```
rrc a
mov _i,a
```

Note that SDCC stores numbers in little-endian format (i.e. lowest order first).

4.1.9 Bit-rotation

```
000D E4                  64         clr  a
000E 13                  65         rrc  a
000F F5*02               66         mov  _foo_hob_1_1,a
```

Variations of this case however will *not* be recognized. It is a standard C expression, so I heartily recommend this be the only way to get the highest order bit, (it is portable). Of course it will be recognized even if it is embedded in other expressions, e.g.:

```
xyz = gint + ((gint >> 15) & 1);
```

will still be recognized.

4.1.11 Peep-hole Optimizer

The compiler uses a rule based, pattern matching and re-writing mechanism for peep-hole optimization. It is inspired by *copt* a peep-hole optimizer by Christopher W. Fraser (cwfraser@microsoft.com). A default set of rules are compiled into the compiler, additional rules may be added with the *–peep-file <filename>* option. The rule language is best illustrated with examples.

```
replace {
mov %1,a
mov a,%1
} by {
mov %1,a
}
```

mov a,%1

```
push ar1
```

with the restart option the rule will be applied again to the resulting code and then all the pop-push pairs will be eliminated to yield:

```
; nop
; nop
```

A conditional function can be attached to a rule. Attaching rules are somewhat more involved, let me illustrate this with an example.

```
replace {
ljmp %5
%2:
} by {
sjmp %5
%2:
} if labelInRange
```

The optimizer does a look-up of a function name table defined in function *callFuncByName* in the source file SDCCpeeph.c, with the name *labelInRange*. If it finds a corresponding entry the function is called. Note there can be no parameters specified for these functions, in this case the use of *%5* is crucial, since the function *labelInRange* expects to find the label in that particular variable (the hash table containing the vari-

Note the header file “serial.h” MUST be included in the file containing the ’main’ function.

ser.h - Alternate serial routine provided by Wolfgang Esslinger <wolfgang@WiredMinds.com> these routines are more compact and faster. Please see documentation in file SDCCDIR/sdcc51lib/ser.c

```
return c_funcˇ10,9ı;
}
```

4.7 Cyclomatic Complexity

If you detect that the stack is over writing you data, then the following can be done. –xstack will cause an external stack to be used for saving registers and (if –stack-auto

7.2 How the Debugger Works

When the –debug option is specified the compiler generates extra symbol information some of which are put into the the assembler source and some are put into the .cdb file, the linker updates the .cdb file with the address information for the symbols. pe0 -11dhe Deb the inform5iler t

7.5.7 step

Step program until it reaches a different source line.

7.5.8 next

Step program, proceeding through subroutine calls.

7.5.9 run

Start debugged program.

```
;; x               sdcdbsrc-delete             SDCDB Delete
all breakpoints if no arg
;;                                             given or delete
arg (C-u arg x)
;; m               sdcdbsrc-frame              SDCDB Dis-
play current frame if no arg,
;;                                              given or dis-
play frame arg
;;                                              buffer point

;; !                sdcdbsrc-goto-sdcdb         Goto the SD-
CDB output buffer
;; p               sdcdb-print-c-sexp          SDCDB print
command for data at
;;                                              buffer point

;; g               sdcdbsrc-goto-sdcdb         Goto the SD-
CDB output buffer
;; t               sdcdbsrc-mode               Toggles Sd-
cdbsrc mode (turns it off)
;;
;; C-c C-f         sdcdb-finish-from-src       SDCDB fin955 Td[(cdbsrc)-600(
```

holding varibles. IY is currently unusued. Return values are stored in HL. One bad side effect of using IX as the base pointer is that a functions stack frame is limited to

Index