TEXAS INSTRUMENTS

TMS34010 C Compiler Reference Card

Phone Numbers

TI Customer Response Center (CRC)
Hotline: (800) 232-3200
Graphics Hotline: (713) 274-2340

Invoking the Preprocessor

gspcpp *[input file] [options]*

gspcpp is the command that invokes the preprocessor.

input file is a C source file; it usually has an extension of **.c**.

options:

- **-c** copies comments to the output file.
- **-d***name*[**=def**] defines *name* as if it appeared in a #define statement.
- **-i***dir* adds *dir* to the list of directories that are searched for #include files.
- **-p** inhibits generation of line number and file information.
- **-q** suppresses the banner and status information.

The preprocessor creates an output file with an extension of *.cpp*.

Invoking the Parser

gspcc *[input file] [options]*

gspcc is the command that invokes the parser.

input file is a modified source file created by the preprocessor; it has an extension of **.cpp**.

options:

- **-z** retains the input file.
- **-q** suppresses the banner and status information.

The parser creates an output file with an extension of *.if*.

Invoking the Code Generator

gspcg *[input file] [options]*

gspcg is the command that invokes the code generator.

input file is an intermediate file created by the parser; it has an extension of **.if**.

options:

- **-a** indicates that the program contains aliasing.
- **-o** places symbolic debuging directives in the output file.
- **-q** suppresses the banner and status information.
- **-r** writes a register-status table to the output file.
- **-s** uses the small-code model.
- **-x** checks for stack overflow at run time.
- **-z** retains the input file.

The code generator creates an assembly-language output file with an extension of *.asm*.

Invoking the Batch Files

gspc *[input file]* **or** **gspq** *[input file]*

gspc invokes the `gspc.bat` batch file to invoke the compiler and assembler. This batch file prints banners and status messages and retains the intermediate *.asm* file.

gspq invokes the `gspq.bat` batch file. to invoke the compiler and assembler. This batch file is a quiet version that does not print any messages; note that this batch file **deletes** the intermediate *.asm* file.

input file is a C source file. Don't specify an extension; the batch files assume an extension of *.c*.

Linking a C Program

gsplnk -c *filenames -o name.out* **-l rts.lib -l flib.lib**
or
gsplnk -cr *filenames -o name.out* **-l rts.lib -l flib.lib**

-c/-cr are options that tell the linker to use special conventions necessary in the C environment.

filenames are object files created by compiling and assembling a C program.

-o name.out names the output file. If you don't use -o, the linker creates an output file with the default name of `a.out`.

rts.lib is the archive library that contains runtime-support functions; the -l option tells the linker that a file is an object library.

flib.lib is the archive library that contains floating-point functions; the -l option tells the linker that a file is an object library.

Additional options that you may want to use when linking C code include:

- **-m** creates a map file.
- **-r** retains relocation entries in the output file.
- **-i** names directories that contain object libraries.

Environment Variables

	Set	Reset
DOS	set C_DIR=*path1; ... ;pathn*	set C_DIR=
VMS	assign "*path1; ... ;pathn* " C_DIR	deassign C_DIR
UNIX	setenv C_DIR "*path1; ... ;pathn* "	setenv C_DIR " "
MPW	set C_DIR "*path1; ... ;pathn* " export C_DIR	unset C_DIR

TMS34010 Data Types and sizes

Type	Size
char	8 bits, signed ASCII
unsigned char	8 bits, ASCII
short	16 bits
unsigned short	16 bits
int	32 bits
unsigned int	32 bits
long	32 bits
unsigned long	32 bits
pointers	32 bits
float	32 bits Range: $\pm 5.88 \times 10^{-39}$ thru $\pm 1.70 \times 10^{38}$
double	64 bits Range: $\pm 1.11 \times 10^{-308}$ thru $\pm 8.99 \times 10^{308}$
enum	1–32 bits

C Operators

Highest Priority				
Operators				**Associativity**
`()`	`[]`	`->`	`.`	left to right

`!`
`-`
(type)
	`~`
	`*`
		`++`
		`&`
			`--`
			`sizeof`
				right to left

`*`	`/`	`%`		left to right
`+`	`-`			left to right
`>>`	`<<`			left to right
`<`	`<=`	`>`	`>=`	left to right
`==`	`!=`			left to right
`&`				left to right
`^`				left to right
`|`				left to right
`&&`				left to right
`||`				left to right
`?:`				right to left

`=`
`/=`
`|=`
	`+=`
	`% =`
	`<<=`
		`-=`
		`&=`
		`>>=`
			`*=`
			`^=`
				right to left

				left to right
Lowest Priority				

Runtime-Support Functions

Header File: `assert.h`

```
void  assert(expression)
   int  expression;
```

Header File: `ctype.h`

```
int  isalnum(c)
   char  c;
int  isalpha(c)
   char  c;
int  isascii(c)
   char  c;
int  iscntrl(c)
   char  c;
int  isdigit(c)
   char  c;
int  isgraph(c)
   char  c;
int  islower(c)
   char  c;
int  isprint(c)
   char  c;
int  ispunct(c)
   char  c;
int  isspace(c)
   char  c;
int  isupper(c)
   char  c;
int  isxdigit(c)
   char  c;
char  toascii(c)
   char  c;
char  tolower(c)
   char  c;
char  toupper(c)
   char c;
```

Header File: math.h

```
double  acos(x)
   double  x;
```

```
double  asin(x)
   double  x;
```

```
double  atan(x)
   double  x;
```

```
double  atan2(y, x)
   double  y, x;
```

```
double  ceil(x)
   double  x;
```

```
double  cos(x)
   double  x;
```

```
double  cosh(x)
   double  x;
```

```
double  exp(x)
   double  x;
```

```
double  fabs(x)
   double  x;
```

```
double  floor(x)
   double  x;
```

```
double  fmod(x, y)
   double  x, y;
```

```
double  frexp(value, exp)
   double value;
   int    *exp;
```

```
double  ldexp(x, exp)
   double x;
   int    exp;
```

```
double  log(x)
   double  x;
```

```
double  log10(x)
   double  x;
```

```
double  modf(value, iptr)
   double value;
   int    *iptr;
```

```
double  pow(x, y)
   double  x, y;
```

```
double  sin(x)
   double  x;
```

```
double  sinh(x)
   double  x;
```

```
double  sqrt(x)
   double  x;
```

```
double  tan(x)
   double  x;
```

```
double  tanh(x)
   double  x;
```

Header File: stdarg.h

```
type va_arg(ap, type)
   va_list  ap
```

```
void  va_end(ap)
   va_list  ap
```

```
void  va_start(ap, parm)
   va_list  ap
```

Runtime-Support Functions

Header File: stdlib.h

```
int  abs(j)
   int  j;
```

```
void  abort()
```

```
void  atexit(fun)
   void  (*fun)();
```

```
int  atof(nptr)
   char  *nptr;
```

```
int  atoi(nptr)
   char  *nptr;
```

```
long int atol(nptr)
   char  *nptr;
```

```
void  *bsearch(key, base, nmemb,
                size, compar)
   void   *key, *base;
   size_t nmemb, size;
   int    (*compar)();
```

```
void  *calloc(nmemb, size)
   size_t  nmemb, size;
```

```
void  exit(status)
   int  status;
```

```
void  free(ptr)
   void  *ptr;
```

```
int  labs(j)
   int  j;
```

```
int  ltoa(n, buffer)
   long  n;
   char  *buffer;
```

```
void  *malloc(size)
   size_t  size;
```

```
void  *minit()
```

```
char  *movmem(src,dest,count)
   char  *src, *dest
   int   count;
```

```
void  qsort(base, nmemb, size, compar)
   void   *base;
   size_t nmemb, size;
   int    (*compar)();
```

```
int  rand()
```

```
void  *realloc(ptr, size)
   void   *ptr;
   size_t size;
```

```
int  srand(seed)
   unsigned int  seed;
```

```
double  strtod(nptr, endptr)
   char  *nptr, **endptr;
```

```
long int  strtol(nptr, endptr, base)
   char *nptr, **endptr;
   int  base;
```

```
unsigned long int  strtoul(nptr, endptr,
                           base)
   char *nptr, **endptr;
   int  base;
```

Header File: string.h

```
void  *memchr(s, c, n)
   void   *s;
   int    c;
   size_t n;
```

```
int  memcmp(s1, s2, n)
   void   *s1, *s2;
   size_t n;
```

```
void  *memcpy(s1, s2, n)
   void   *s1, *s2;
   size_t n;
```

```
void  *memmove(s1, s2, n)
   void   *s1, *s2;
   size_t n;
```

Runtime-Support Functions

Header File: string.h (continued)

```
void  *memset(s, c, n)
   void   *s;
   int    c;
   size_t n;
```

```
char  *strcat(s1, s2)
   char  *s1, *s2;
```

```
char  *strchr(s, c)
   char *s;
   int  c;
```

```
int  strcmp(s1, s2)
   char  *s1, *s2;
```

```
int  *strcoll(s1, s2)
   char  *s1, *s2;
```

```
char  *strcpy(s1, s2)
   char  *s1, *s2;
```

```
size_t  strcspn(s1, s2)
   char  *s1, *s1;
```

```
char  *strerror(errnum)
   int  errnum;
```

```
size_t  strlen(s)
   char  *s;
```

```
char  *strncat(s1, s2, n)
   char   *s1, *s2;
   size_t n;
```

```
char  *strncmp(s1, s2, n)
   char   *s1, *s2;
   size_t n;
```

```
char  *strncpy(s1, s2, n)
   char   *s1, *s2;
   size_t n;
```

```
char  *strpbrk(s1, s2)
   char  *s1, *s2;
```

```
char  *strrchr(s ,c)
   char *s;
   int  c;
```

```
size_t  *strspn(s1, s2)
   char  *s1, *s2;
```

```
char  *strstr(s1, s2)
   char  *s1, *s2;
```

```
char  *strtok(s1, s2)
   char  *s1, *s2;
```

Header File: time.h

```
char  *asctime(timeptr)
   struct tm *timeptr;
```

```
clock_t  clock()
```

```
char  *ctime(timeptr)
   struct tm *timeptr;
```

```
double  difftime(time1,time0)
   time_t time1, time0;
```

```
struct tm  *gmtime(timer)
   time_t *timer;
```

```
struct tm  *localtime(timer)
   time_t *timer;
```

```
time_t  mktime(timeptr)
   struct tm *timeptr;
```

```
size_t  strftime(s,maxsize,timeptr)
   char   *s, *format;
   size_t maxsize;
   struct tm *timeptr
```

```
time_t  time(timer)
   time_t *timer;
```

TEXAS INSTRUMENTS

TMS34010
Math/Graphics Function Library
Reference Card

```
double acos(x)
    double x;

void add_text_space(n)
    int n;

double asin(x)
    double x;

double atan(x)
    double x;

double atan2(u,v)
    double u,v;

void bit_expand(srcbits, srcpitch, w, h, xleft, ytop)
    short srcbits[];
    long  srcpitch;
    int   w, h, xleft, ytop;

void bound_fill(x, y, buffer, size, b_color)
    int  x, y, size;
    char buffer[];
    unsigned long b_color;

void bound_patnfill(x, y, buffer, size, b_color)
    int  x, y, size;
    char buffer[];
    unsigned long b_color;

double ceil(x)
    double x;

int char_high()

int char_wide_max()

void clear_screen(pixval)
    long pixval;

int close_vuport(index)
    int index;

void color_blend(pxlval, y1, y2, red1, grn1, blu1,
                 red2, grn2, blu2)
    int pxlval, y1, y2;
    int red1, grn1, blu1;
    int red2, grn2, blu2;

typedef long FIX
void copy_matrix(matrixin, matrixout)
    FIX  matrixin[16];
    FIX  matrixout[16];

void copy_vertex(n, vertexin, vertexout)
    typedef long FIX;
    int n;
    FIX vertexin[], vertexout[];

int copy_vuport(index1, index2)
    int index1, index2;

double cos(x)
    double x;

double cosh(x)
    double x;

double cotan(x)
    double x;

int cpw(x, y)
    int x, y;
```

```
void delay(n)
    int n;

int draw_char(x, y, c)
    int  x, y;
    char c;

void draw_line(x1, y1, x2, y2)
    int x1, y1, x2, y2;

void draw_oval(w, h, xleft, ytop)
    int w, h, xleft, ytop;

void draw_ovalarc(w, h, xleft, ytop, theta, arc)
    int w, h, xleft, ytop;
    int theta, arc;

void draw_piearc(w, h, xleft, ytop, theta, arc)
    int w, h, xleft, ytop;
    int theta, arc;

void draw_point(x, y)
    int x, y;

void draw_polyline(n, linelist, ptlist)
    int   n;
    short linelist[], ptlist[];

void draw_rect(w, h, xleft, ytop)
    int w, h, xleft, ytop;

int draw_string(x, y, s)
    int  x, y;
    char *s;

double exp(x)
    double x;

double fabs(x)
    double x;

int fill_convex(n, edgelist, ptlist)
    int   n;
    short edgelist[], ptlist[];

void fill_oval(w, h, xleft, ytop)
    int w, h, xleft, ytop;

void fill_piearc(w, h, xleft, ytop, theta, arc)
    int w, h, xleft, ytop;
    int theta, arc;

void fill_polygon(n, linelist, ptlist)
    int   n;
    short linelist[], ptlist[];

void fill_rect(w, h, xleft, ytop)
    int w, h, xleft, ytop;

float *fix_to_float(n, in_array, out_array)
    typedef long FIX;
    int   n;
    FIX   in_array[];
    float out_array[];

long *fix_to_long(n, in_array, out_array)
    typedef long FIX;
    int  n;
    FIX  in_array[];
    long out_array[];

short *fix_to_short(n, in_array, out_array)
    typedef long FIX;
    int   n;
    FIX   in_array[];
    short out_array[];

.global FIX2FL
.global FL2FIX
.global FL_ADD
.global FL_COS
.global FL_MULT
```

```
.global FL_SIN

FIX *float_to_fix(n, in_array, out_array)
    typedef long FIX;
    int   n;
    float in_array[];
    FIX   out_array[];

double floor(x)
    double x;

double fmod(x, y)
    double x, y;

void frame_oval(w, h, xleft, ytop, dx, dy)
    int w, h, xleft, ytop;
    int dx, dy;

void frame_rect(w, h, xleft, ytop, dx, dy)
    int w, h, xleft, ytop;
    int dx, dy;

double frexp(value, exp)
    double value;
    int   *exp;

void getall_palet(palet_array, reg_mask, y)
    short palet_array[16];
    int   reg_mask, y;

int get_ascent()

int get_descent()

int get_first_ch()

int get_font_max()

int get_last_ch()

int get_leading()

int get_patn_max()

int get_pixel(x, y)
    int  x, y;

long get_pmask()

long get_ppop()

int get_psize()

void get_rect(w, h, xleft, ytop, darray, dpitch)
    int   w, h, xleft, ytop;
    short darray[]
    long  dpitch;

int get_transp()

int get_vuport_max()

int get_width(s)
    char *s;

void init_grafix()

void init_matrix(matrix)
    typedef long FIX;
    FIX matrix[16];

void init_palet()

void init_screen()

void init_text()

int init_video(monitor_val)
    int monitor_val;

void init_vuport()

int install_font(index, fontname)
    int  index;
    FONT *fontname;

int install_patn(index, pattern)
    int   index;
    short pattern[16];
```

```
double ldexp(value, exp)
    double value;
    int    exp;
```

```
char *lib_id()
```

```
int lmo(n)
    long n;
```

```
double log(x)
    double x;
```

```
double log10(x)
    double x;
```

```
FIX *long_to_fix(n, in_array, out_array)
    typedef long FIX;
    int  n;
    long in_array[];
    FIX  out_array[];
```

```
double modf(value, exp)
    double value;
    int    *exp;
```

```
void move_pixel(xs, ys, xd, yd)
    int xs, ys, xd, yd;
```

```
void move_rect(w, h, xs, ys, xd, yd)
    int w, h;
    int xs, ys, xd, yd;
```

```
void move_vuport(xleft, ytop)
    int xleft, ytop;
```

```
void new_screen(pixel, palet)
    long  pixel;
    short palet[16];
```

```
int open_vuport()
```

```
int patnfill_convex(n, edgelist, ptlist)
    int   n;
    short edgelist[], ptlist[];
```

```
void patnfill_oval(w, h, xleft, ytop)
    int w, h, xleft, ytop;
```

```
void patnfill_piearc(w, h, xleft, ytop, theta, arc)
    int w, h, xleft, ytop;
    int theta, arc;
```

```
void patnfill_polygon(n, linelist, ptlist)
    int   n;
    short linelist[], ptlist[];
```

```
void patnfill_rect(w, h, xleft, ytop)
    int w, h, xleft, ytop;
```

```
void patnframe_oval(w, h, xleft, ytop, dx, dy)
    int w, h, xleft, ytop;
    int dx, dy;
```

```
void patnframe_rect(w, h, xleft, ytop, dx, dy)
    int w, h, xleft, ytop;
    int dx, dy;
```

```
void patnpen_line(x1, y1, x2, y2)
    int  x1, y1, x2 y2;
```

```
void patnpen_ovalarc(w, h, xleft, ytop, theta, arc)
    int w, h, xleft, ytop;
    int theta, arc;
```

```
void patnpen_piearc(w, h, xleft, ytop, theta, arc)
    int w, h, xleft, ytop;
    int theta, arc;
```

```
void patnpen_point(x, y)
    int  x, y;
```

```
void patnpen_polyline(n, linelist, ptlist)
    int   n;
    short linelist[], ptlist[];
```

```
int peek(address)
    long address;
```

```
long peek_breg(breg)
    int breg;
```

```
void pen_line(x1, y1, x2, y2)
    int x1, y1, x2, y2;
```

```
void pen_ovalarc(w, h, xleft, ytop, theta, arc)
    int w, h, xleft, ytop;
    int theta, arc;
```

```
void pen_piearc(w, h, xleft, ytop, theta, arc)
    int w, h, xleft, ytop;
    int theta, arc;
```

```
void pen_point(x, y)
    int  x, y;
```

```
void pen_polyline(n, linelist, ptlist)
    int   n;
    short linelist[], ptlist[];
```

```
void perspec(n, vertlist, ptlist, xview, yview, zview)
    typedef long FIX;
    FIX   vertlist[];
    short ptlist[];
    int   n, xview, yview, zview;
```

```
void poke(address, value)
    long address;
    int  value;
```

```
void poke_breg(breg, value)
    long breg;
    int  value;
```

```
double pow(x, y)
    double  x, y;
```

```
void put_pixel(val, x, y)
    int val, x, y;
```

```
void put_rect(sarray, spitch, w, h, xleft, ytop)
    short sarray[];
    long  spitch;
    int   w, h, xleft, ytop;
```

```
long rep_pixel(val)
    int val;
```

```
int rmo(n)
    long n;
```

```
void rotate(matrix, angle)
    typedef long FIX;
    FIX matrix[16], angle[3];
```

```
void run_decode(xleft, ytop, image)
    int   xleft, ytop;
    short image[];
```

```
int run_encode(w, h, xleft, ytop, image, maxbytes)
    int   w, h, xleft, ytop, maxbytes;
    short image[];
```

```
void scale(matrix, factor)
    typedef long FIX;
    FIX matrix[16], factor[3];
```

```
void seed_fill(xseed, yseed, buffer, maxbytes)
    int  xseed, yseed, maxbytes;
    char buffer[];
```

```
void seed_patnfill(xseed, yseed, buffer, maxbytes)
    int  xseed, yseed, maxbytes;
    char buffer[];
```

```
int select_font(index)
    int index;
```

```
int select_patn(index)
    int index;
```

```
int select_vuport(index)
    int index;
```

```
void setall_palet(palet, reg_mask, n, y)
    short palet[16];
    int   reg_mask, n, y;
```

```
void set_cliprect(w, h, xleft, ytop)
    int w, h, xleft, ytop;
```

```
void set_color0(pixel_val)
    long pixel_val;
```

```
void set_color1(pixel_val)
    long pixel_val;
```

```
void set_origin(x0, y0)
    int x0, y0;
```

```
void set_palet(reg, red, grn, blu)
    int reg, red, grn, blu;
```

```
void set_pensize(w, h)
    int w, h;
```

```
void set_pmask(mask)
    long pmask;
```

```
void set_ppop(ppop_code)
    int ppop_code;
```

```
FIX *short_to_fix(n, in_array, out_array)
    typedef long FIX;
    int   n;
    short in_array[];
    FIX   out_array[];
```

```
double sin(x)
    double x;
```

```
double sinh(x)
    double x;
```

```
int size_vuport(w, h)
    int w, h;
```

```
double sqrt(x)
    double x;
```

```
long styled_line(x1, y1, x2, y2, style, mode)
    int  x1, y1, x2, y2, mode;
    long style;
```

```
double tan(x)
    double x;
```

```
double tanh(x)
    double x;
```

```
void transform(matrix, n, verts)
    typedef long FIX;
    FIX matrix[16], verts[];
    int n;
```

```
void translate(matrix, disp)
    typedef long FIX;
    FIX matrix[16], disp[3];
```

```
void transp_off()
```

```
void transp_on()
```

```
void vertex_to_point(n, verts, ptlist)
    int   n;
    FIX   verts[];
    short ptlist[];
```

```
void wait_scan(line)
    int line;
```

```
long xytoaddr(x, y)
    int x, y;
```

```
void zoom_rect(ws, hs, xs, ys, wd, hd, xd, yd, linebuf)
    int   ws, hs, xs, ys;
    int   wd, hd, xd, yd;
    short linebuf[];
```