TEXAS INSTRUMENTS

TMS34010
Assembly Language Tools
Reference Card

Phone Numbers

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

Assembler Directives

.align	**.line** *line number [,address]*
.bes *size in bits*	**.list**
.block *beginning line number*	**.long** *value1[,...,valuen]*
.bss *symbol,size in bits [,word alignment flag]*	**.member** *name,value[,type ,storage class,size,tag,dims]*
.byte *value1[,...,valuen]*	**.mlib** *["]filename["]*
.copy *["]filename["]*	**.mlist**
.data	**.mnolist**
.def *symbol1[,...,symboln]*	**.nolist**
.double *floating-point value*	**.option** *{B|D|F|L|M|T|X}*
.else	**.page**
.end	**.sect** *"section name"*
.endblock *ending line number*	*symbol* **.set** *value*
.endfunc *ending line number*	**.space** *size in bits*
.endif	**.stag** *name,size*
.eos	**.string** *"string1"[,...,"stringn"]*
.etag *name,size*	**.sym** *name,value[,type ,storage class,size,tag,dims]*
.even	
.field *value[,size in bits]*	**.text**
.file *"filename"*	**.ref** *symbol1[,...,symboln]*
.float *floating-point value*	**.title** *"string"*
.func *beginning line number*	*symbol* **.usect** *"section name" ,size in bits[,word alignment flag]*
.global *symbol1[,...,symboln]*	
.if *expression*	
.include *["]filename["]*	**.utag** *name,size*
.int *value1[,...,valuen]*	**.width** *page width*
.length *page length*	**.word** *value1[,...,valuen]*

Sample MEMORY and SECTIONS Linker Directives

```
MEMORY
{
    display : o =          0h, l = 01FFFF0h
    code    : o = 0D0000000h, l = 03FFFF0h
    space   : o = 0FFE00000h, l = 01FFFF0h
}

SECTIONS
{
    .text  0D0000000h : { }
    powers ALIGN(32)  : { }  > code
    .data             : { }  > code
    newvars           : { }  > space
    .bss              : { }  > space
}
```

Invoking the Assembler

gspa *input file [object file [listing file]][-options]*

Options:
-b makes blanks significant.
-c makes case insignificant.
-h allows alternate hex format.
-i *pathname*
specifies a directory where the assembler can find files named by the .copy, .include, or .mlib directives.
-l (lowercase "L") produces a listing file.
-q suppresses the banner and all progress information.
-s puts **all** defined symbols in the object file's symbol table.
-x produces a cross-reference listing of symbols.

Invoking the Linker

gsplnk *[-options] file1 ... filen*

Options:
-a produces an absolute, executable module.
-ar produces a relocatable, executable object module.
-c uses ROM autoinitialization model (for C code).
-cr uses RAM autoinitialization model (for C code).
-e *global symbol*
defines the primary entry point for the output module.
-f *16-bit fill value*
sets the default fill value for holes within output sections.
-h makes all global symbols static.
-i *pathname*
specifies a directory where the linker can find object libraries named with -l.
-l *libname*
names an object library file as linker input.
-m *map file name*
produces an output map listing.
-o *output file name*
names the executable output module (the default filename is *a.out*).
-q requests a quiet run (suppress the banner).
-r retains relocation entries in the output module.
-s strips symbol table information and line number entries from the output module.
-u *symbol*
places an unresolved external *symbol* into the output module's symbol table.

Invoking the Archiver

gspar *[-]command[option] libname [file1 ... filen]*

Commands:
-a adds the specified files to the library.
-d deletes the specified members from the library.
-r replaces the specified members in the library.
-t prints a table of contents of the library.
-x extracts the specified files.

Options:
e tells the archiver not to use the default extension .obj for member names.
q suppresses the banner and status messages.
s prints a list of the symbols that are defined in the library. (Valid only with the -a, -r, and -d commands.)
v describes the creation of a new library from an old library.

Invoking the Object Format Converter

gsprom *[-option] [file1 [file2 [file3]]]*

Filename Order:
1) Input filename
2) Output filename (TI-tagged format) or high-byte output filename (Tektronix or Intel format)
3) Low-byte output file (Tektronix or Intel format)

Options:
-i produces Intel hex object output.
-t produces TI-tagged object output.
-x produces Tektronix-hex object output (default).

Register File B

Reg	Function	Description	Reg	Function	Description
B0	SADDR	Source address	B7	DYDX	Delta Y/delta X
B1	SPTCH	Source pitch	B8	COLOR0	Color 0
B2	DADDR	Destination address	B9	COLOR1	Color 1
B3	DPTCH	Destination pitch	B10 B11 B12 B13 B14	TEMP TEMP TEMP TEMP TEMP	Used as temporary storage for PIXBLTs and FILLs
B4	OFFSET	Offset			
B5	WSTART	Window start			
B6	WEND	Window end	SP	SP	Stack pointer

I/O Registers

Address	Register	Description
0C00001F0h	REFCNT	DRAM refresh count
0C00001E0h	DPYADR	Display address
0C00001D0h	VCOUNT	Vertical count
0C00001C0h	HCOUNT	Horizontal count
0C00001B0h	DPYTAP	Display tap point
0C00001A0h 0C0000170h	Reserved	
0C0000160h	PMASK	Plane mask
0C0000150h	PSIZE	Pixel size
0C0000140h	CONVDP	Conversion (destination pitch)
0C0000130h	CONVSP	Conversion (source pitch)
0C0000120h	INTPEND	Interrupt pending
0C0000110h	INTENB	Interrupt enable
0C0000100h	HSTCTLH	Host control high (8 MSBs)
0C00000F0h	HSTCTLL	Host control low (8 LSBs)
0C00000D0h	HSTADRH	Host address high (16 MSBs)
0C00000E0h	HSTADRL	Host address low (16 LSBs)
0C00000C0h	HSTDATA	Host data
0C00000B0h	CONTROL	I/O control
0C00000A0h	DPYINT	Display interrupt
0C0000090h	DPYSTRT	Display start
0C0000080h	DPYCTL	Display control
0C0000070h	VTOTAL	Vertical total
0C0000060h	VSBLNK	Vertical start blank
0C0000050h	VEBLNK	Vertical end blank
0C0000040h	VESYNC	Vertical end sync
0C0000030h	HTOTAL	Horizontal total
0C0000020h	HSBLNK	Horizontal start blank
0C0000010h	HEBLNK	Horizontal end blank
0C0000000h	HESYNC	Horizontal end sync

Vector Address Map

Trap#	Address	Desc	Trap #	Address	Desc
0	0FFFFFFE0h	RESET	16	0FFFFFDE0h	
1	0FFFFFFC0h	INT1	17	0FFFFFDC0h	
2	0FFFFFFA0h	INT2	18	0FFFFFDA0h	
3	0FFFFFF80h		19	0FFFFFD80h	
4	0FFFFFF60h		20	0FFFFFD60h	
5	0FFFFFF40h		21	0FFFFFD40h	
6	0FFFFFF20h		22	0FFFFFD20h	
7	0FFFFFF00h		23	0FFFFFD00h	
8	0FFFFFEE0h	NMI	24	0FFFFFCE0h	
9	0FFFFFEC0h	HI	25	0FFFFFCC0h	
10	0FFFFFEA0h	DI	26	0FFFFFCA0h	
11	0FFFFFE80h	WV	27	0FFFFFC80h	
12	0FFFFFE60h		28	0FFFFFC60h	
13	0FFFFFE40h		29	0FFFFFC40h	
14	0FFFFFE20h		30	0FFFFFC20h	ILLOP
15	0FFFFFE00h		31	0FFFFFC00h	

Condition Codes for JRcc and JAcc Instructions

Mnemonic Code		Result of Compare	Status Bits	Code
Unconditional Compares				
UC	–	Unconditional	don't care	0000
Unsigned Compares				
LO (C)	–	Dst lower than Src	C	0001
LS	YLE	Dst lower than or same as Src	C + Z	0010
HI	YGT	Dst higher than src	$\overline{C} \cdot \overline{Z}$	0011
HS (NC)	–	Dst higher than or same as Src	$\overline{C}$	1001
EQ (Z)	–	Dst = Src	Z	1010
NE (NZ)	–	Dst ≠ Src	$\overline{Z}$	1011
Signed Compares				
LT	XLE	Dst < Src	$(N \cdot \overline{V}) + (\overline{N} \cdot V)$	0100
LE	–	Dst ≤ Src	$(N \cdot \overline{V}) + (\overline{N} \cdot V) + Z$	0110
GT	–	Dst > Src	$(N \cdot V \cdot \overline{Z}) + (\overline{N} \cdot \overline{V} \cdot \overline{Z})$	0111
GE	XGT	Dst ≥ Src	$(N \cdot V) + (\overline{N} \cdot \overline{V})$	0101
EQ (Z)	–	Dst = Src	Z	1010
NE (NZ)	–	Dst ≠ Src	$\overline{Z}$	1011
Compare to Zero				
Z	YZ	Result = zero	Z	0101
NZ	YNZ	Result is nonzero	$\overline{Z}$	1011
P	–	Result is positive	$\overline{N} \cdot \overline{Z}$	0001
N	XZ	Result is negative	N	1110
NN	XNZ	Result is nonnegative	$\overline{N}$	1111
General Arithmetic				
Z	YZ	Result is zero	Z	1010
NZ	YNZ	Result is nonzero	$\overline{Z}$	1011
C	YN	Carry set on result	C	1000
NC	YNC	No carry on result	$\overline{C}$	1001
B (C)	–	Borrow set on result	C	1000
NB (NC)	–	No borrow on result	$\overline{C}$	1001
V†	XN	Overflow on result	V	1100
NV†	XNN	No overflow on result	$\overline{V}$	1101

Note: A mnemonic code in parentheses is an alternate code for the preceding code.

† Also used for window clipping

\+ Logical OR

· Logical AND

– Logical NOT

Environment Variables

The environment variable for the assembler is **A_DIR**. The environment variable for the linker is **C_DIR**.

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

TMS34010 Instruction Set

Syntax	Operation
ABS *Rd*	|Rd| → Rd
ADD *Rs, Rd*	Rs + Rd → Rd
ADDC *Rs, Rd*	Rs + Rd + C → Rd
ADDI *IW, Rd, [W]*	16-bit immediate value + Rd → Rd
ADDI *IL, Rd, [L]*	32-bit immediate value + Rd → Rd
ADDK *K, Rd*	K + Rd → Rd
ADDXY *Rs, Rd*	Rs **X** + Rd**X** → Rd**X** Rs**Y** + Rd**Y** → Rd**Y**
AND *Rs, Rd*	Rs AND Rd → Rd
ANDI *IL, Rd*	IL AND Rd → Rd
ANDN *Rs, Rd*	(NOT Rs) AND Rd → Rd
ANDNI *IL, Rd*	(NOT IL) AND Rd → Rd
BTST *K, Rd*	Set status on value of: bit K in Rd
BTST *Rs, Rd*	Set status on: value of a bit in Rd (Rs specifies a bit number)
CALL *Rs*	PC' → TOS Rs → PC SP -32 → SP
CALLA *Address*	PC' → TOS Address → PC
CALLR *Address*	PC' → TOS PC' + (displacement × 16) → PC
CLR *Rd*	Rd XOR Rd → Rd
CLRC	0 → C
CMP *Rs, Rd*	Set status bits on result of: Rd - Rs
CMPI *IW, Rd, [W]*	Set status bits on the result of: Rd - 16-bit immediate value
CMPI *IL, Rd, [L]*	Set status bits on the result of: Rd - 32-bit immediate value
CMPXY *Rs, Rd*	Set status bits on the results of: Rd**X** - Rs**X** Rd**Y** - Rs**Y**
CPW *Rs, Rd*	point code → Rd
CVXYL *Rs, Rd*	XY address in Rs → linear address in Rd
DEC *Rd*	Rd - 1 → Rd
DINT	0 → IE
DIVS *Rs, Rd*	Rd even: Rd:Rd+1 / Rs → Rd Remainder → Rd+1 Rd odd: Rd / Rs → Rd
DIVU *Rs, Rd*	Rd even: Rd:Rd+1 / Rs → Rd Remainder → Rd+1 Rd odd: Rd / Rs → Rd

TMS34010 Instruction Set

Syntax	Operation
DRAV *Rs, Rd*	COLOR1 pixel value → *Rd Rs**X** + Rd**X** → Rd**X** Rs**Y** + Rd**Y** → Rd**Y**
DSJ *Rd, Address* **DSJS** *Rd, Address*	Rd - 1 → Rd If Rd ≠ 0 (disp. × 16) + PC' → PC If Rd = 0 go to next instruction
DSJEQ *Rd, Address*	If Z = 1 Rd - 1 → Rd If Rd ≠ 0 (disp. × 16) + PC' → PC If Rd = 0 go to next instruction If Z = 0 go to next instruction
DSJNE *Rd, Address*	If Z = 0 Rd - 1 → Rd If Rd ≠ 0 (disp. × 16) + PC' → PC If Rd = 0 go to next instruction If Z = 1 go to next instruction
EINT	1 → IE
EMU	ST → Rd Conditionally enter emulator mode
EXGF *Rd [, F]*	Rd → FS0, FE0 **or** Rd → FS1, FE1 FS0, FE0 → Rd **or** FS1, FE1 → Rd
EXGPC *Rd*	Rd → PC PC' → Rd
FILL L	COLOR1 pixel values → pixel array (linear source address)
FILL XY	COLOR1 pixel values → pixel array (XY source address)
GETPC *Rd*	PC' → Rd
GETST *Rd*	ST → Rd
INC *Rd*	PC + 1 → Rd
JA*cc* *Address*	If *cc* = true Address → PC If *cc* = false go to next instruction
JR*cc* *Address*	If *cc* = true disp. + PC' → PC If *cc* = false go to next instruction
JUMP *Rs*	Rs → PC
LINE *[0 , 1]*	Perform the inner loop of Bresenham's line-drawing algorithm.
LMO *Rs, Rd*	31 - bit number of leftmost 1 in Rs → Rd
MMFM *Rs [, reg. list]*	If Register *n* is in the register list *Rs+ → R*n* (repeat for *n* = 0 to 15)
MMTM *Rd [, reg. list]*	If Register *n* is in the register list R*n* → -*Rd (repeat for *n* = 0 to 15)
MODS *Rs, Rd*	Rd mod Rs → Rd
MODU *Rs, Rd*	Rd mod Rs → Rd
MOVB	See MOVB summary
MOVE	See MOVE summary
MOVI *IW, Rd, [W]*	16-bit immediate operand → Rd
MOVI *IL, Rd, [L]*	32-bit immediate operand → Rd

TMS34010 Instruction Set

Syntax	Operation
MOVK *K, Rd*	K → Rd
MOVX *Rs, Rd*	Rs **X** → Rd**X**
MOVY *Rs, Rd*	Rs **Y** → Rd**Y**
MPYS *Rs, Rd*	Rd even: Rs × Rd → Rd:Rd+1 Rd odd: Rs × Rd → Rd
MPYU *Rs, Rd*	Rd even: Rs × Rd → Rd:Rd+1 Rd odd: Rs × Rd → Rd
NEG *Rd*	-Rd → Rd
NEGB *Rd*	-Rd - C → Rd
NOP	No operation
NOT *Rd*	NOT Rd → Rd
OR *Rs, Rd*	Rs OR Rd → Rd
ORI *IL, Rd*	IL OR Rd → Rd
PIXBLT	See PIXBLT summary
PIXT	See PIXT summary
POPST	*SP+ → ST
PUSHST	ST → -*SP
PUTST *Rs*	Rs → ST
RETI	*SP+ → ST *SP+ → PC
RETS *[N]*	*SP → PC (N defaults to 0) SP + 32 + 16N → SP
REV *Rd*	revision number → Rd
RL *K, Rd*	Rd rotated left by K → Rd
RL *Rs, Rd*	Rd rotated left by Rs → Rd
SETC	1 → C
SETF *FS, FE [, F]*	(FS, FE) → ST
SEXT *Rd [, F]*	field in Rd → sign-extended field in Rd
SLA *K, Rd*	left-shift Rd by K → Rd
SLA *Rs, Rd*	left-shift Rd by Rs → Rd
SLL *K, Rd*	left-shift Rd by K → Rd
SLL *Rs, Rd*	left-shift Rd by Rs → Rd
SRA *K, Rd*	right-shift Rd by K → Rd
SRA *Rs, Rd*	right-shift Rd by Rs → Rd
SRL *K, Rd*	right-shift Rd by K → Rd
SRL *Rs, Rd*	right-shift Rd by Rs → Rd
SUB *Rs, Rd*	Rd - Rs → Rd
SUBB *Rs, Rd*	Rd - Rs - C → Rd
SUBI *IW, Rd, [W]*	Rd - 16-bit immediate value → Rd
SUBI *IL, Rd, [L]*	Rd - 32-bit immediate value → Rd
SUBK *K, Rd*	Rd - K → Rd
SUBXY *Rs, Rd*	Rd **X** - Rs**X** → Rd**X** Rd**Y** - Rs**Y** → Rd**Y**
TRAP *N*	PC → -*SP ST → -*SP trap vector *N* → PC
XOR *Rs, Rd*	Rs XOR Rd → Rd
XORI *IL, Rd*	IL XOR Rd → Rd
ZEXT *Rd [, F]*	field in Rd → zero-extended field in Rd

Key:

Rs -Source register
Rd -Destination register
IW -16-bit (short) immediate value
IL -32-bit (long) immediate value
K -5-bit constant
PC'-Next instruction
RsX, RdX -X half (16 LSBs) of Rs or Rd
RsY, RdY -Y half (16 MSBs) of Rs or Rd
SAddress-32-bit source address
DAddress-32-bit destination address
Address -32-bit address (label)
F -Field select, defaults to 0
F=0 selects FS0 and FE0
F=1 selects FS1 and FE1

MOVE Instructions Summary

	Destination					
Source	*Rd*	**Rd*	**Rd+*	*-**Rd*	**Rd(offset)*	*@DAddress*
Rs	√	√	√	√	√	√
**Rs*	√	√				
**Rs+*	√		√			
*-**Rs*	√			√		
**Rs(offset)*	√		√		√	
@SAddress	√		√			√

A check mark (√) in a box indicates a valid combination of source and destination operands. For example,

MOVE *Rs,* **Rd(offset)*

is a valid form of the MOVE instruction.

MOVB Instructions Summary

	Destination			
Source	*Rd*	**Rd*	**Rd(offset)*	*@DAddress*
Rs		√	√	√
**Rs*	√	√		
**Rs(offset)*	√		√	
@SAddress	√			√

A check mark (√) in a box indicates a valid combination of source and destination operands. For example,

MOVB *Rs,* **Rd(offset)*

is a valid form of the MOVB instruction.

PIXT Instructions Summary

	Destination Pixel		
Source Pixel	*Rd*	**Rd*	**Rd.XY*
Rs		√	√
**Rs*	√	√	
**Rs.XY*	√		√

A check mark (√) in a box indicates a valid combination of source and destination operands. For example,

PIXT **Rs, Rd*

is a valid form of the PIXT instruction.

PIXBLT Instructions Summary

	Destination Array	
Source Array	**L**	**XY**
B	√	√
L	√	√
XY	√	√

B – Binary array address
L – Linear array address
XY – XY array address

A check mark (√) in a box indicates a valid combination of source and destination array types. For example,

PIXBLT B, XY

is a valid form of the PIXBLT instruction.