

https://dspace.org/

A software engineering approach to a digitally synthesised modem using a Texas Instruments TMS320 signal processor

Andrews, Donald Graham

2014-09-16
http://hdl.handle.net/10539/15550

Downloaded from a DSpace repository. https://dspace.org/

Table F.1 TMS 32010 Pin Assignment (Continued)

SIGNAL	PIN	I/O	DESCRIPTION
			PROGRAM MEMORY ADDRESS BUS AND PORT ADDRESS BUS
A11	27	OUT	Program memory A11 (MSB) through A0 (LSB) and port addresses PA2 (MSB) through PA0 (LSB). Addresses A11 through A0 are always active and never go to high impedance. During execution of the IN and OUT instructions, pins A2 through A0 carry the port addresses PA2 through PA0.
A10	28	OUT	
A9	29	OUT	
A8	34	OUT	
A7	35	OUT	
A6	36	OUT	
A5	37	OUT	
A4	38	OUT	
A3	39	OUT	
A2/PA2	40	OUT	
A1/PA1	1	OUT	
A0/PA0	2	OUT	

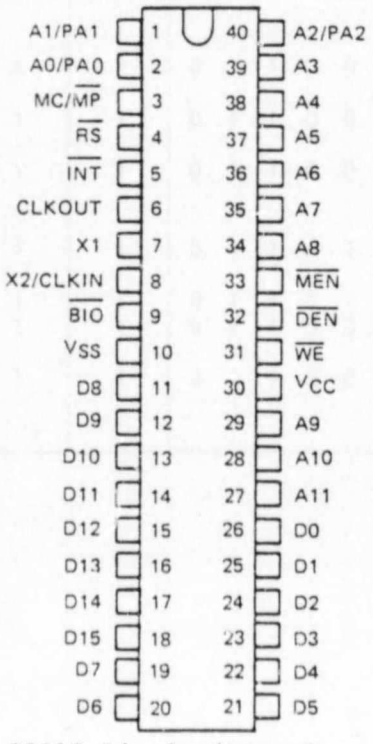


Fig F.2 TMS 32010 Pin Assignment

Table F.2 Instruction Set Summary

ACCUMULATOR INSTRUCTIONS																		
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER														
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABS	Absolute value of accumulator	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	0	0
ADD	Add to accumulator with shift	1	1	0	0	0	0	←	S	→		I	←			D	→		
ADDH	Add to high-order accumulator bits	1	1	0	1	1	0	0	0	0	0	I	←			D	→		
ADDS	Add to accumulator with no sign extension	1	1	0	1	1	0	0	0	0	1	I	←			D	→		
AND	AND with accumulator	1	1	0	1	1	1	1	0	0	1	I	←			D	→		
LAC	Load accumulator with shift	1	1	0	0	1	0	←	S	→		I	←			D	→		
LACK	Load accumulator immediate	1	1	0	1	1	1	1	1	1	0	←				K	→		
OR	OR with accumulator	1	1	0	1	1	1	1	0	1	0	I	←			D	→		
SACH	Store high-order accumulator bits with shift	1	1	0	1	0	1	1	←	X	→	I	←			D	→		
SACL	Store low-order accumulator bits	1	1	0	1	0	1	0	0	0	0	I	←			D	→		
SUB	Subtract from accumulator with shift	1	1	0	0	0	1	←	S	→		I	←			D	→		
SUBC	Conditional subtract (for divide)	1	1	0	1	1	0	0	1	0	0	I	←			D	→		
SUBH	Subtract from high-order accumulator bits	1	1	0	1	1	0	0	0	1	0	I	←			D	→		
SUBS	Subtract from accumulator with no sign extension	1	1	0	1	1	0	0	0	1	1	I	←			D	→		
XOR	Exclusive OR with accumulator	1	1	0	1	1	1	1	0	0	0	I	←			D	→		
ZAC	Zero accumulator	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	0	1
ZALH	Zero accumulator and load high-order bits	1	1	0	1	1	0	0	1	0	1	I	←			D	→		
ZALS	Zero accumulator and load low-order bits with no sign extension	1	1	0	1	1	0	0	1	1	0	I	←			D	→		

Table F.2 Instruction Set Summary (Continued)

AUXILIARY REGISTER AND DATA PAGE POINTER INSTRUCTIONS																		
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER														
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LAR	Load auxiliary register	1	1	0	0	1	1	1	0	0	R	I	←			D			→
LARK	Load auxiliary register immediate	1	1	0	1	1	1	0	0	0	R	←				K			→
LARP	Load auxiliary register pointer immediate	1	1	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	K
LDP	Load data memory page pointer	1	1	0	1	1	0	1	1	1	1	I	←			D			→
LDPK	Load data memory page pointer immediate	1	1	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	K
MAR	Modify auxiliary register and pointer	1	1	0	1	1	0	1	0	0	0	I	←			D			→
SAR	Store auxiliary register	1	1	0	0	1	1	0	0	0	R	I	←			D			→

BRANCH INSTRUCTIONS																		
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER														
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B	Branch unconditionally	2	2	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
BANZ	Branch on auxiliary register not zero	2	2	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
BGEZ	Branch if accumulator ≥ 0	2	2	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
BGZ	Branch if accumulator > 0	2	2	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
BIOZ	Branch on $\overline{BIO}$ = 0	2	2	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
BLEZ	Branch if accumulator ≤ 0	2	2	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
BLZ	Branch if accumulator < 0	2	2	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
BNZ	Branch if accumulator ≠ 0	2	2	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
BV	Branch on overflow	2	2	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
BZ	Branch if accumulator = 0	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
CALA	Call subroutine from accumulator	2	1	0	1	1	1	1	1	1	1	1	0	0	0	1	1	0	0
CALL	Call subroutine immediately	2	2	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
RET	Return from subroutine	2	1	0	1	1	1	1	1	1	1	1	0	0	0	1	1	0	1

Table F.2 Instruction Set Summary

T REGISTER, P REGISTER, AND MULTIPLY INSTRUCTIONS																	
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER													
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
APAC	Add P register to accumulator	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1
LT	Load T register	1	1	0	1	1	0	1	0	1	0	I	←			D			→
LTA	LTA combines LT and APAC into one instruction	1	1	0	1	1	0	1	1	0	0	I	←			D			→
LTD	LTD combines LT, APAC, and DMOV into one instruction	1	1	0	1	1	0	1	0	1	1	I	←			D			→
MPY	Multiply with T register; store product in P register	1	1	0	1	1	0	1	1	0	1	I	←			D			→
MPYK	Multiply T register with immediate operand; store product in P register	1	1	1	0	0	←								K				→
PAC	Load accumulator from P register	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0
SPAC	Subtract P register from accumulator	1	1	0	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0

CONTROL INSTRUCTIONS																	
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER													
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DINT	Disable interrupt	1	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1
EINT	Enable interrupt	1	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0
LST	Load status register	1	1	0	1	1	1	1	0	1	1	I	←			D			→
NOP	No operation	1	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
POP	Pop stack to accumulator	2	1	0	1	1	1	1	1	1	1	1	0	0	1	1	1	0	1
PUSH	Push stack from accumulator	2	1	0	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0
ROVM	Reset overflow mode	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	1	0
SOVM	Set overflow mode	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	1	1
SST	Store status register	1	1	0	1	1	1	1	1	0	0	I	←			D			→

I/O AND DATA MEMORY OPERATIONS																	
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER													
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMOV	Copy contents of data memory location into next location	1	1	0	1	1	0	1	0	0	1	I	←			D			→
IN	Input data from port	2	1	0	1	0	0	0		PA		I	←			D			→
OUT	Output data to port	2	1	0	1	0	0	1		PA		I	←			D			→
TBLR	Table read from program memory to data RAM	3	1	0	1	1	0	0	1	1	1	I	←			D			→
TBLW	Table write from data RAM to program memory	3	1	0	1	1	1	1	1	0	1	I	←			D			→

Author Andrews Donald Graham
Name of thesis A Software Engineering Approach To Digitally Synthesised Modem Using A Texas Instruments Tms320 Signal Processor. 1985

PUBLISHER:
University of the Witwatersrand, Johannesburg
©2013

LEGAL NOTICES:

Copyright Notice: All materials on the University of the Witwatersrand, Johannesburg Library website are protected by South African copyright law and may not be distributed, transmitted, displayed, or otherwise published in any format, without the prior written permission of the copyright owner.

Disclaimer and Terms of Use: Provided that you maintain all copyright and other notices contained therein, you may download material (one machine readable copy and one print copy per page) for your personal and/or educational non-commercial use only.

The University of the Witwatersrand, Johannesburg, is not responsible for any errors or omissions and excludes any and all liability for any errors in or omissions from the information on the Library website.