CONTENT

1.0 FEATURES
- 1.1 DIFFERENCE BETWEEN MX93011 and MX93011A

2.0 FUNCTION BLOCK DIAGRAM

3.0 PIN CONFIGURATION
- 3.1 PIN DESCRIPTIONS
- 3.2 PIN TYPE SUMMARY
- 3.3 MULTIPLEX PINS

4.0 FUNCTION DESCRITION
- 4.1 LOOP
- 4.2 MODULAR ADDRESSING
- 4.3 AUXILIARY REGISTERS
- 4.4 STACK
- 4.5 HOLD
- 4.6 MEMORY MAPS
- 4.7 CLOCK/TIMER/POWER DOWN
- 4.8 ADDRESSING MODES
- 4.9 INTERRUPT

5.0 REGISTERS SUMMARY

6.0 REGISTERS DESCRIPTION

7.0 INSTRUCTION SET SUMMARY

8.0 INSTRUCTION SET DESCRIPTION

9.0 DC CHARACTERISICS

10.0 AC TIMING AND CHARACTERISICS

11.0 ORDER INFORMATION

12.0 PACKAGE INFORMATION

1.0 FEATURES

- 16-bit, 46.5ns instruction cycle, up to 21MIPS DSP controller for DAM (Digital Answering Machine) application.
- 32-bit ALU and 16-bit auxiliary ALU (ARAU) work in parallel.
- 8 auxiliary registers for indirect addressing work with ARAU.
- 32-level hardware stack and nestable interrupt support.
- 32-bit barrel shifter.
- 8-instruction looped up to 128 times capability.
- 64k words program ROM space, 32k words may be internal.
- External ROM option may replace internal 32K for fast prototyping.
- 64k words SRAM space, 2048 words internal.
- 32 internal IO address.
- 1 independent interrupt pin, 1 NMI pin.
- 8 input pins.
- 8 bi-direction I/O pins.
- 19 output pins.
- Hold or slow system clock for power management.
- 1/1024 sec or1 ms system tick timer for system timing.
- One Codec interface.
- Built-in DRAM Controller;1G addressing space, with 1/4/8/16 data bit interface support.
- 0.6u Single 5V supply, 100 pins PQFP

1.1 DIFFERENCE BETWEEN MX93011 AND MX93011A

1. external memory wait state:

I/o register(8)

MX93011

7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1
	MMSIZE	SRAMWAIT	MMWAIT		ROMWAIT		

MX93011A

10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	1	1
MMSIZE		SRAMWAIT			MMWAIT			ROMWAIT		

2.stack register:

MX93011:16x16

MX93011A:32x16

stack pointer register:

MX93011

3	2	1	0
0	0	0	0

MX93011A

4	3	2	1	0
0	0	0	0	0

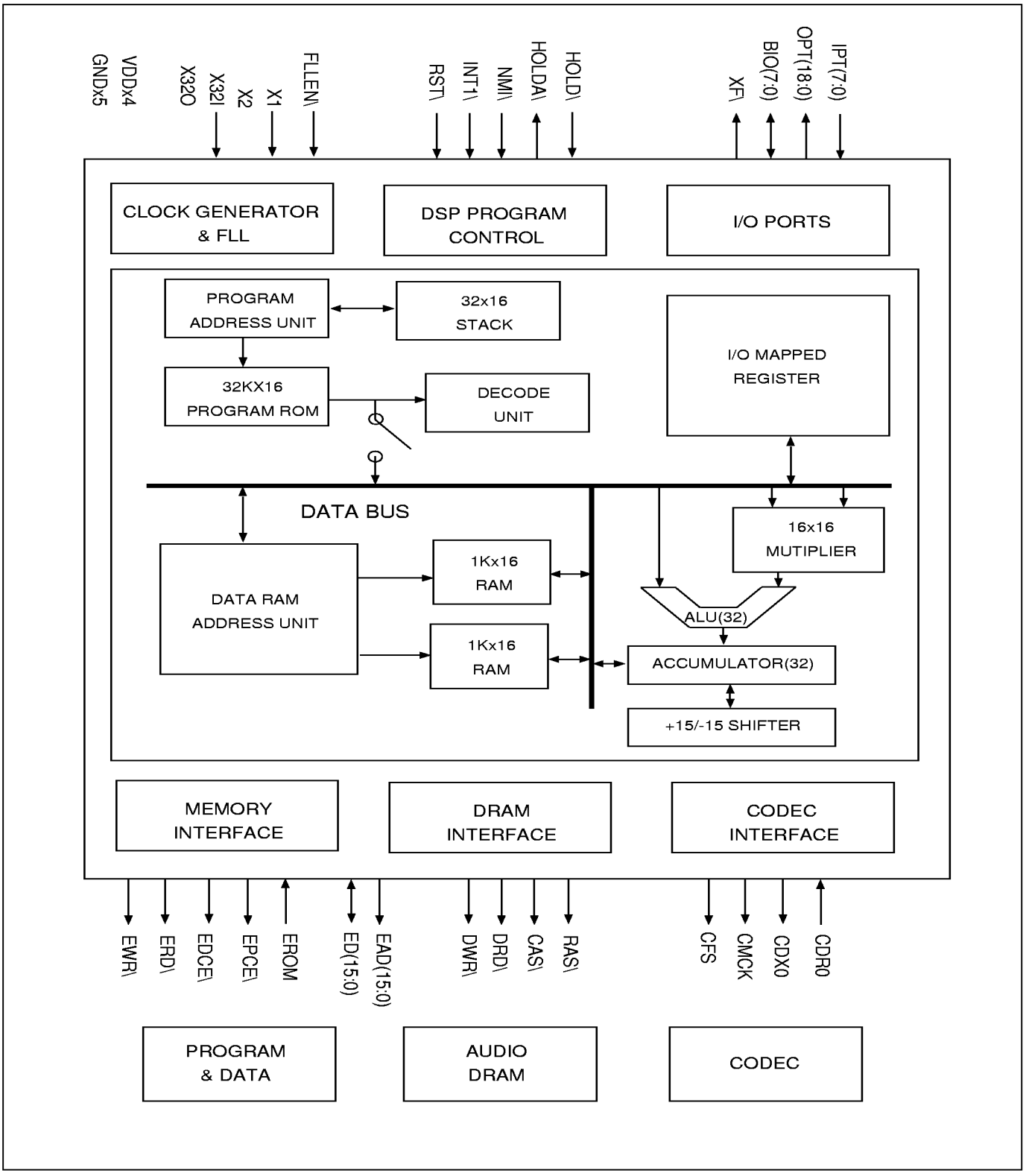
3.internal ROM:

MX93011	18Kx16
MX93011A	32Kx16

4.SINGLE LOW X' TAL MODE:

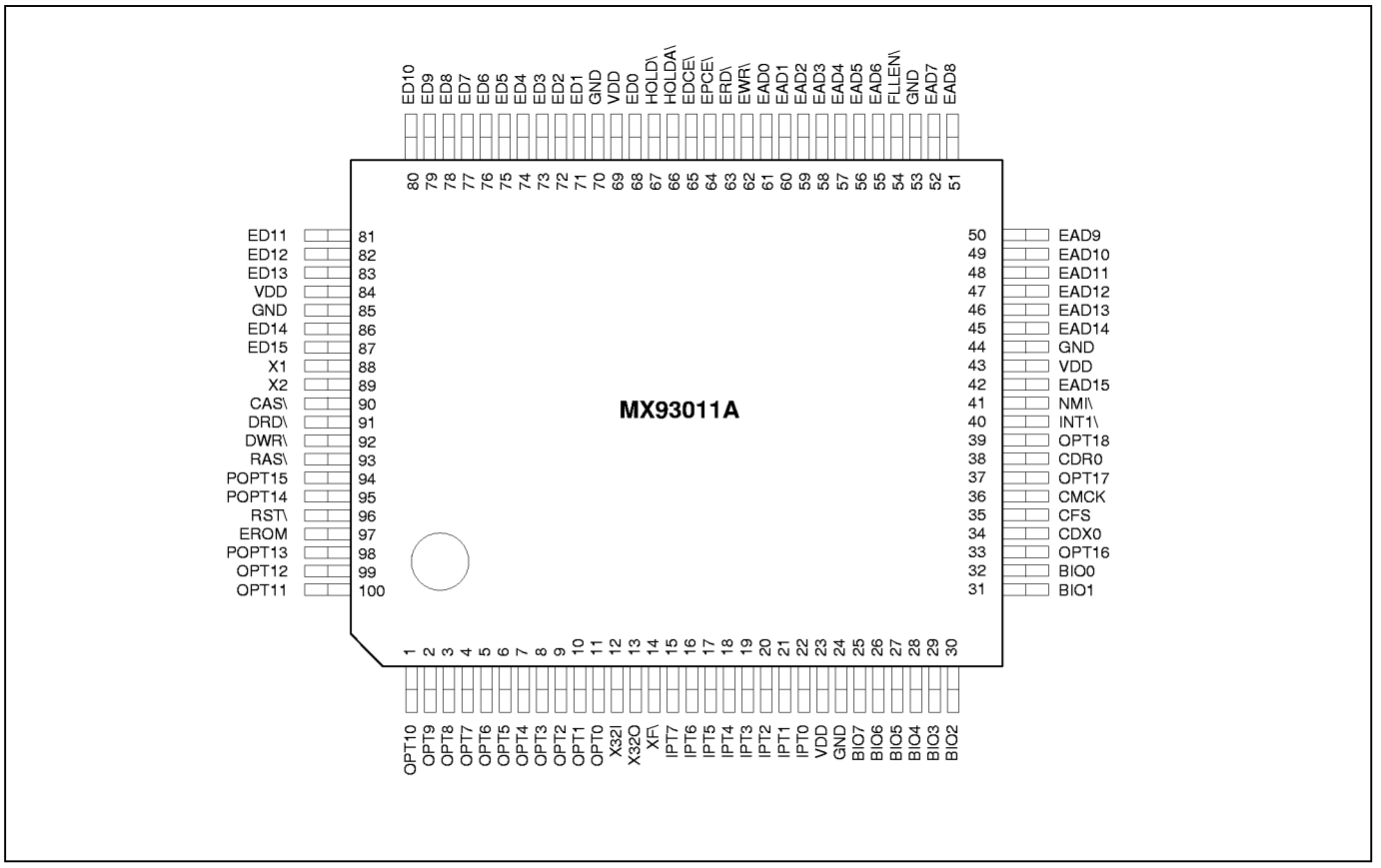
In MX93011A, high X' TAL is no longer needed. High clock(32.256 MHz) required for DSP running with can be generated from FLL (Frequence Locked Loop) by enabling FLLEN\ pin. X1 and X2 of high X'TAL should be connected to VDD and GND respectively in this mode.

2.0 FUNCTION BLOCK DIAGRAM



3.0 PIN CONFIGURATION

100 PQFP



3.1 PIN DESCRIPTIONS
POWER/CLOCK/CONTROL PINS:

SYMBOL	PIN TYPE	PIN NUMBER	DESCRIPTION
VDD		23, 43, 69, 84	5V power source
GND		24, 44, 53, 70, 85	Ground
X1/VDD		88	32.256MHZ Crystal input/CONNECT to VDD in single low X'tal mode
X2/GND		89	32.256MHZ Crystal output/CONNECT to VDD in single low X'tal mode
RST\	IS	96	Power-on Reset .
XF\	OA	14	External flag if UPMODX=1. This pin can be directly written by one DSP instruction. Default inactive (5V output).
HOLD\	IS	67	Hold DSP clock down and release bus
HOLDA\	OA/Z	66	Ack to HOLD\ signal
EROM	IS	97	Disable internal ROM; use external ROM only.
NMI\	IS	41	Non maskable interrupt pin.
INT1\	IS	40	Interrupt pin
X32O		13	32.768KHZ Crystal output.
X32I		12	32.768KHZ Crystal input.
FLLEN\	IS	54	1: Dual X'tal Mode. 0. Single low X'tal Mode.

MEMORY INTERFACE PINS :

SYMBOL	PIN TYPE	PIN NUMBER	DESCRIPTION
EAD0-EAD15	OA/Z	61-55, 52-45, 42	DSP IO/RAM/ROM external address bus. EAD0-EAD14 are for DRAM address.
ED0-ED15	IT/OA/ZR	68, 71-83, 86-87	DSP IO/RAM/ROM/DRAM external data bus. With Soft latch feed back current is 250uA.
EDCE\	OA/Z	65	External data chip enable.
EPCE\	OA/Z	64	External program chip enable.
ERD\	OA/Z	63	SRAM/ROM/IO external read.
EWR\	OA/Z	62	SRAM/ROM/IO external write.
CAS\	OA	90	DRAM column address select.
RAS\	OA	93	DRAM row address select.
DRD\	OA	91	DRAM read.
DWR\	OA	92	DRAM write.

CODEC INTERFACE PINS:

SYMBOL	PIN TYPE	PIN NUMBER	DESCRIPTION
CFS	OA	35	Codec frame sync, 8 KHz. (9.6KHz) Output low in power down mode.
CMCK	OA	36	Codec master clock, 1.536 MHz. Output low in power down mode.
CDX0	OA	34	Codec data transmit
CDR0	IS	38	Codec data receive

OPT : Output port

SYMBOL	PIN TYPE	PIN NUMBER	DESCRIPTION
OPT0-OPT15	OB	11-1,100-98 95,94	Output to pin, all output values are registered and may be read back when read by 'IN' instruction.
OPT16-OPT18	IT/OA/ZR	33,37,39	Output to pin, when UPMODX=1

BIO : Bi-direction I/O

SYMBOL	PIN TYPE	PIN NUMBER	DESCRIPTION
BIO7-BIO0	IT/OA	25-32	Input/output port when UPMODX=1. Direction is controlled by BIO15-BIO8, (see BIOR).

IPT : Input port

SYMBOL	PIN TYPE	PIN NUMBER	DESCRIPTION
IPT4-IPT7	IS	18-15	Input port.
IPT0-IPT3	ISH	22-19	Input port with internal pull high resister(R=30k ohm)

NOTE:
IT TTL level input

IS CMOS level schmidt trigger input (hysteresis:2V~3V)

ISH CMOS level Schmidt trigger input with internal pull high resistor(~30k ohm)

OA 8mA drive level output

OB 16mA drive level output

Z high impedance state

ZR high impedance state with soft latch

3.2 PIN TYPE SUMMARY :

INPUT : CMOS level schmidt trigger INPUT:
IPT7~IPT4,CDR0,INT1\.NMI\,FLLEN\,HOLD\,RST\,EROM
CMOS level schmidt trigger INPUT with internal pull high resister:
IPT3~IPT0

OUTPUT: 8mA drive level output:
XF\,CDX0,CFS, CMCK,RAS\,CAS\,DRD\,DWR\
8mA drive level output/ high impedance state
EAD15~EAD0,HOLDA\,EPCE\,EDCE\,ERD\,EWR\
16mA drive level output :
OPT15~OPT0

BI-DIRECTION:
TTL level input/8mA OUTPUT /high impedance state
BIO7~BIO0
TTL level input/8mA OUTPUT/high impedance state/soft latch
ED15~ED0 , OPT18~OPT16

3.3 MULTIPLEX PINS

PIN NUMBER	PIN NAME	UPMODX=1 (non_up mode)	PIN NAME	UPMODX=0(up mode)
25~32	BIO(7:0)	Input/output port	HDB(7:0)	Host data bus
39	OPT18	Output port	HILO	High low data select
37	OPT17	Output port	HRD\	Host read
33	OPT16	Output port	HWR\	Host write
14	XF\	External flag	ACK\	Acknowledge to host

NOTE UPMODX:up mode select bit in CONTROL register,"0" is its power on reset value.

PIN NUMBER	PIN NAME	FLLEN\=1(Dual x'tal)	PIN NAME	FLLEN\=0(single x'tal)
88	x1	32.256MHz crystal input	VDD	Power VDD
89	x2	32.256MHz crystal output	GND	Power ground

NOTE FLLEN\:pin 54.

4.0 FUNCTIONAL DESCRIPTION

4.1 LOOP

Repeat or loop instruction is important in DSP operation. The MX93011A supports this function by implementing many instructions which are implictly repeated with the number stored in the RCR register. Loop up to 8 instructions with specified number of times (can be variable) is also implemented with hardware. Furthermore, flexible usage format is supported which makes the instruction more useful.

4.2 MODULAR ADDRESSING

Modular addressing is by modular operation at the output of ARAU. To use modular addressing user must first store non-zero number m which is stored to the MODR register. With this in effect, memory space beginning from $k \cdot 2^n$ to $k \cdot 2^n + m$, where k is an integer greater than or equal to zero and 2^n is a power-of-two integer greater than m, will form a circular memory space. Whenever boundary location, 0 or m, is addressed, the next AR content will be set/reset to m/0, independent of the instruction specification. Set MODR to 0 will deactivate modular addressing. For example, if MODR is set to 23, circular memory spaces will start from $32 \cdot k$ to $32 \cdot k + 23$. Any instruction can be indirectly addressed to 55, assuming that using AR1, with increasing operation, will make the next AR1 content to be reset to 32. Likewise, if AR1 content is in decreasing operation and the content of AR1 is set to 0, then the next value of AR1 will be reset to 23. If normal addressing mode is desired, simply output a 0 into the MODR registers. This instruction can help construct data RAM into circular buffer or delay line, thereby eliminating the need of physical data movement in the buffer or delay. However, the pointer need to be kept in the data RAM for easy access to the head/tail of this buffer/delay line.

4.3 AUXILIARY REGISTERS

Eight 16 bits auxiliary registers are allocated together with a 16-bit adder/subtractor. The results of adder/ subtractor always go through a modulator to get modular addressing before being stored to the auxiliary registers.

The process provides an independent processor to do address calculation and update in parallel with main data path which performs the instruction execution. Of course, AR registers can also be used as temporary registers and as another unsigned adder/subtractor. AR register modification of $\pm$(0,1,2,AR0) on the fly is also included.

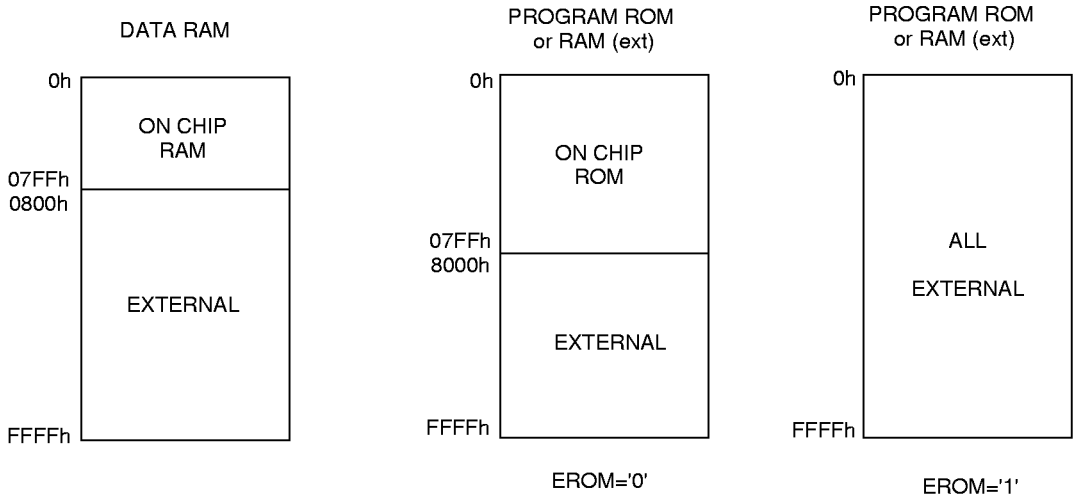
4.4 STACK

Hardware contains 32 deep dedicated stack memories, which support deep hierarchy code. Stack manipulation is transparent to firmware.

4.5 HOLD

Hardware hold is supported through pins HOLD\ and HOLDA\. When HOLD\ is activated, the MX93011A will enter hold state after the present instruction cycle is completed(instructions inside Loop and inherent repeat instruction cycles is considered one instruction cycle). At hold state, the MX93011A will release address and data bus to high impedence, stop executing instruction and output HOLDA\. After HOLD\ is invalid the MX93011A will bring HOLDA\ to high and resume normal operation.

4.6 MEMORY MAPS:



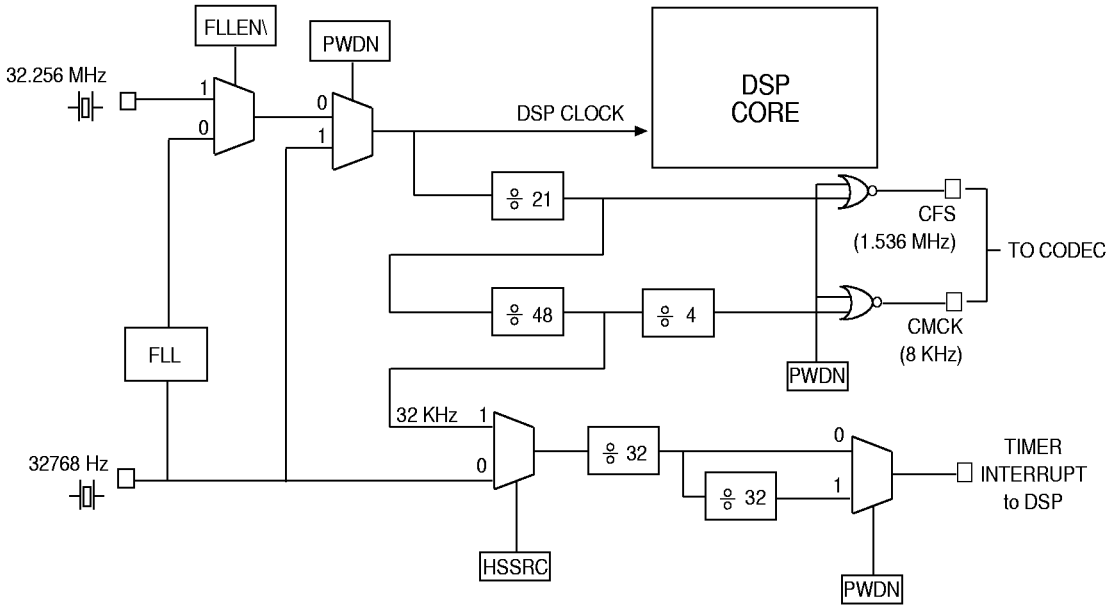
4.7 CLOCK/TIMER/POWERDOWN

High frequency clock(32.256MHz) required for DSP running with can be generated from X'TAL oscillator directly or derived from FLL (FREQUENCY LOCKED LOOP) by enabling FLLEN\ pin. One DSP instruction cycle needs one and half high clock cycle, so the DSP instruction cycle time is 46.5 nano seconds.

When PWDN bit in CONTROL register(I/O register 07) is set, high X'TAL and FLL will be disabled, the DSP running clock will switch to be low clock (32768 Hz) to reduce operating power. When this PWDN bit is reset, DSP will keep on slow speed running for 62.5 mili second, then switch back to normal speed running. LSRUNS bit in CONTROL register will reflect the status of the DSP running speed.

Timer interrupt request is generated every one milli second or 1/1024 second depending on HSSRC bit being set or reset. In power down mode, interrupt occurs every 1/32 second. HSSRC bit must be reset prior to high X'TAL shut down.

In single low X'TAL mode, clock from FLL output is not prescise enough to be used as timer base. Choosing low clock directly from low X'tal output is better.(HSSRC="0")



4.8 ADDRESSING MODES

IMMEDIATE CONSTANT

Immediate constant is coded directly in opcode.

DIRECT MEMORY ADDRESSING

DPR and IOPR are used to completely specify addressing spaces. 4 bits in DPR combined with 7-bits coded in opcode, make direct memory address. (direct memory addressing only for internal 2K WORDS RAM)

INDIRECT ADDRESSING

The memory address may be pointed by ARs. ARs also has post-addressing execution which provides powerful increment(s)/decrement(s) and modular indexing.

It takes only 7 bits to code all these into one opcode to enable program size compact. See AR, ARAU and MODR for more details.

MISCELLANEOUS ADDRESSING MODE

CALL--Call subroutine at the second word of call instruction.
CALA--ACCH indirect call, ACCH=called address
BACC-- ACCH indirect branch, ACCH=branch address
TRAP-- Always call to hex 000C address

4.9 INTERRUPT : OPERATIONS

The Interrupt source, vectoring address and priority are as follows:

NAME	VECTORED ADDRESS	DESCRIPTIONS
RST\	0000	Power-on reset (top priority)
NMI\	0002	NMI\non.maskable interrupt, edge-triggered (high to low)
SS	0004	Single-Step, Single step interrupt is for debugging purpose. If set, the MX93011A will be interrupted after every instruction cycle (instructions inside LOOP and inherent repeat instruction cycles is considered as one instruction cycle). User can put debugging service as the interrupt service routine.
INT1\	0006	INT1\ pin interrupt, edge trigged
CODEC	0008	Triggered when Codec registers get/send 16 bit data (see Timing diagram)
STMR	000A	Triggered by every 1/1024 second or 1milli second depend on the value of HSSRC in normal running, but triggered by 1/32 second in power down mode.
TRAP	000C	Triggered when executes TRAP

Interrupt Process: (Execute by hardware)

1. Release related ISR pending flag
2. Push SSR onto stack
3. Push return-address onto stack
4. Disable global interrupt (same to excuting DINT instruction)
5. If it is in software hold state (see WSTR register and power management), reset SWHOLD $\rightarrow$ 0, and come out of software hold state.

Issues of RETI instruction: (Execute by hardware)

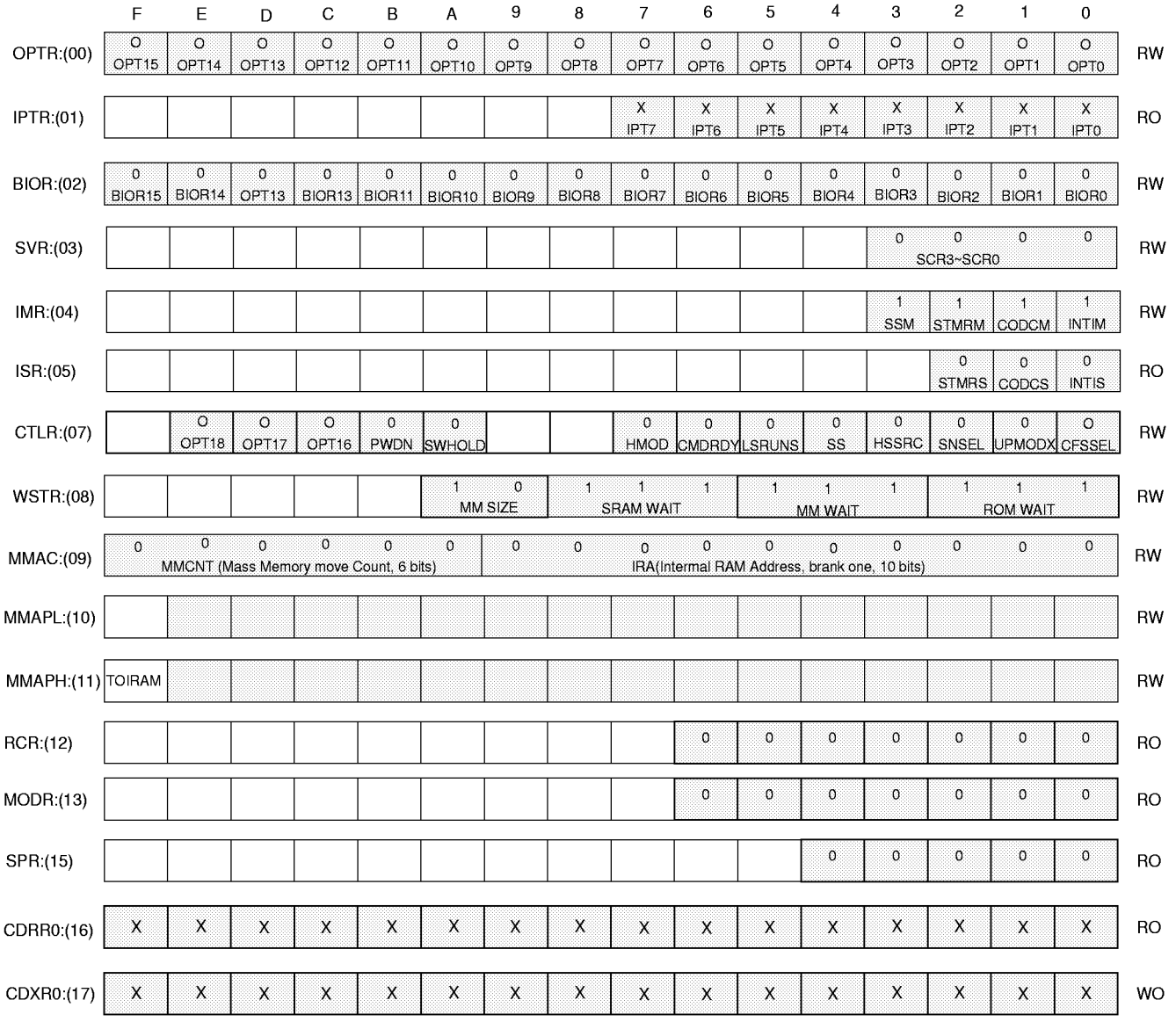
1. POP return address to PC
2. POP SSR

Note that ACC normally need to be saved. All other registers should also be carefully maintained when doing an in-and-out interrupt.

5.0 REGISTERS SUMMARY

NAME	BIT	CTLR	IO ADDRESS	RELATED INSTRUCTIONS	DESCRIPTIONS
optr	16	o	0	IN/OUT	output register
iptr	8	o	1	IN	input port register
bior	16	o	2	IN/OUT	bidirectional io register
svr	4		3	IN/OUT/SFR/SFL	shifter count (scr) and sign
imr	4		4	IN/OUT	interrupt mask register
isr	3		5	IN	interrupt status register
ctlr	15		7	IN/OUT	control register
wstr	8		8	IN/OUT	wait state register
rcr	7		12	IN/RPT/LUP	repeat counter
modr	7		13	IN/MOD	modulo register
spr	4		15	PSH/POP/IN/PSHH/POPH PSHL,POPL	stack pointer register
cdrr0	16	o	16	IN	codec 0 receive buffer
cdxr0	16	o	17	OUT	codec 0 transmit buffer
acch	16		–	(many instr.)	upper word of DSP accumulator
accl	16		–	SAL/ADL/SBL	lower word of DSP accumulator
ar0-7	16x8		–	LAR/MAR/SAR	for indirect memory access basically; also used in macro instructions
accx	32		–	SBL, ADL, SFL SFR,multiply	acch+accl=accx
ssr	16			SSS/OUT/BS/BZ INTM : EINT/DINT TB : BIT OVM : ROVM/SOVM ARP : MAR	status register
pc	16		–	CALL, CALA, TRAP, BS, BZ BACC, RET, RETI, interrupt, hardware reset	program counter

5.1 TABLE OF IO MAPPED REGISTERS AND ITS POWER ON VALUES



6.0 REGISTER DESCRIPTION

IO REGISTERS

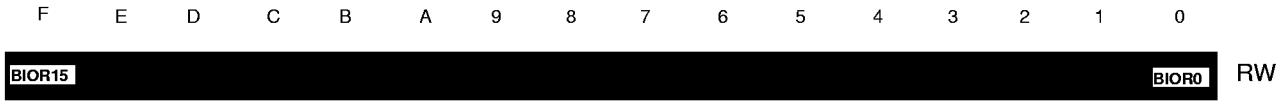
6.1 OPTR : Output Register (mapped to IO register 00)

Output Register (OPTR:15 ~ OPTR:0)
16 bit, connect to OPT15~OPT0 pins.
positive Logic, '1' will output 5 Volt on output pin. ('0' for 0 V)

6.2 IPT : Input Port Register (mapped IO address 01)

Input Port (IPT:7~IPT:0)
Positive Logic, 5 Volt input will read '1' (0V for '0')
IPT:7~IPT:0 connect IPT7~IPT0 pins.

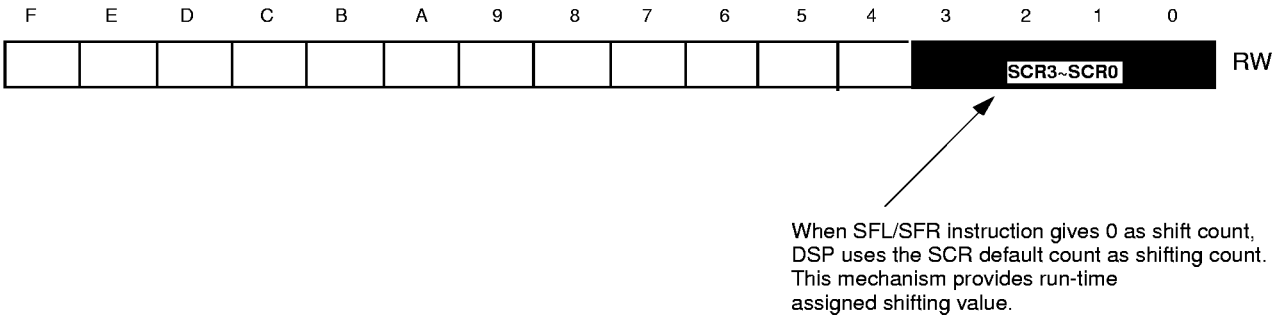
6.3 BIOR/CMDR : BI-DIRECTION IO REGISTER (mapped to IO register 02)



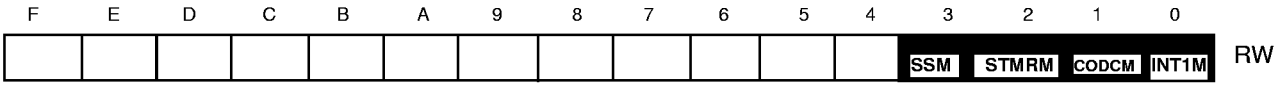
When UP MODX=1, used for bidirectional io register.
Programable bidirectional IO.
BIOR15~BIOR8 control I/O direction of BIOR7~BIOR0, respectively (bit 8 control bit 0)
BIOR7~BIOR0 connect to BIO7~BIO0 pins, respectively.

6.4 SVR : Shift Variable Register (mapped to IO register 03)

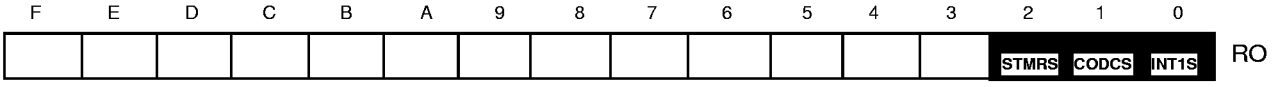
SVR includes Shift-Count Register (SCR)



6.5 IMR : Interrupt Mask Register (mapped to IO register 04)



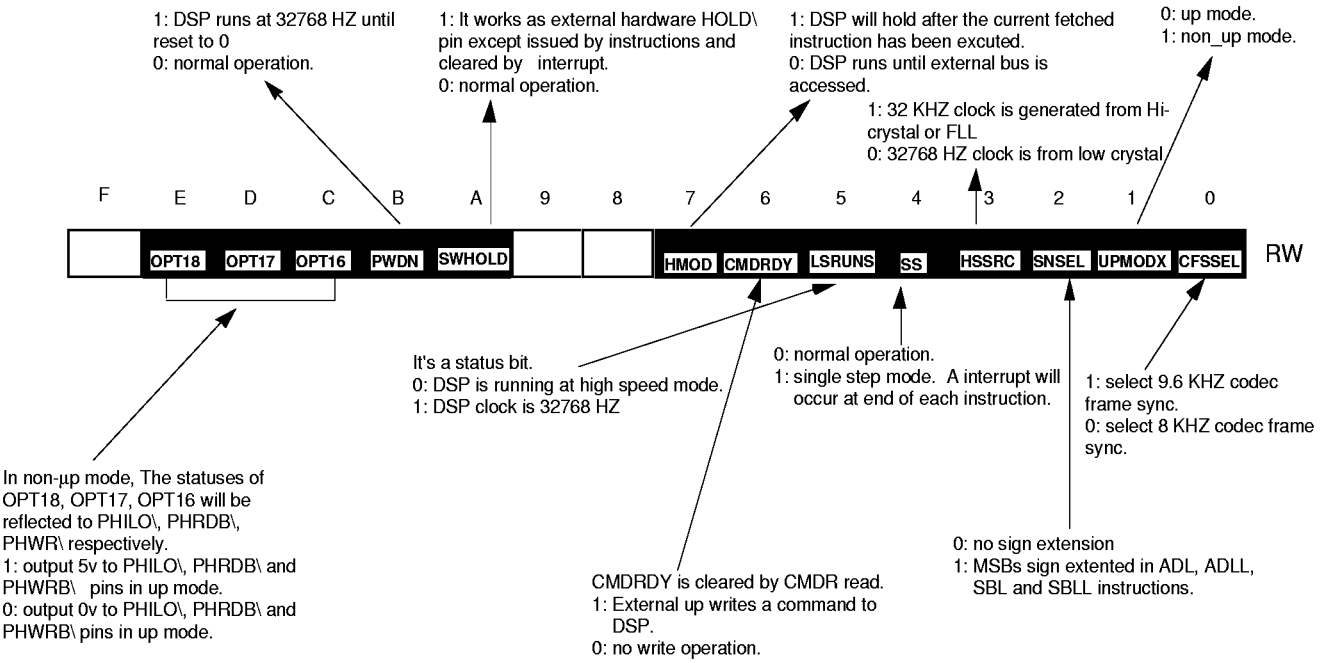
6.6 ISR : Interrupt Status Register (mapped to IO register 05)



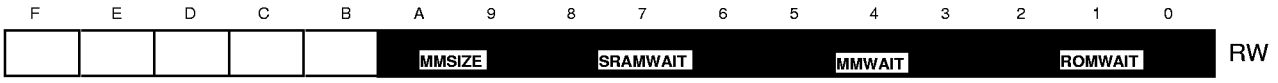
INT1M - INT1 \ Interrupt Mask 1
STMRM - System tick Timer interrupt Mask
CODCM - Codec Interrupt Mask
SSM - Single Step Interrupt Mask

Note 1: Codec Tx/Rx use this same mask.
This is because the 2 events are synchronized and always happen at the same time. Programmers should take care of these 2 events (if necessary) in this interrupt.
Note 2: ISR:2~0 will reflect interrupt pending status on IMR:2~0.
Note that Single-Step (CTLR:SS, register 07) is directly controlled by the program; no status exists.
Note 3: Read ISR will read and clear all pending flags.

6.7 CTLR : CONTROL REGISTER (MAPPED TO IO REGISTER 07)



6.8 WSTR: WAIT STATE, AND DRAM SIZE REGISTER (MAPPED TO IO REGISTER 08)



MASS MEMORY SIZE :

Select DRAM configuration
00 -- x1
01 -- x4
10 -- x8
11 -- x16

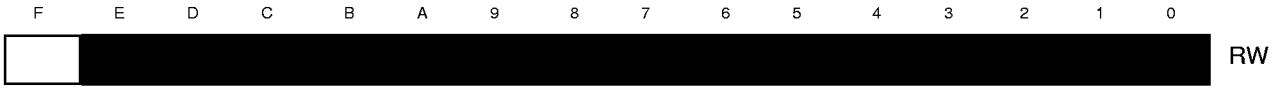
WAIT STATE :

Choose apporiate WAIT_STATE number to meet the following requirement.
1.RAM/ROM(SRAM WAIT,ROM WAIT)
TAA or TCS < 31 ns * (1.5 + WAIT_STATE) -20ns
2.DRAM(MM WAIT)
TRAC < 15.5ns * (6 + WAIT_STATE) -20ns.....(1)
TCAC < 15.5ns * (2+WAIT_STATE)-20ns .....(2)
Choose the larger WAIT_STATE in (1) and (2) as MM WAIT

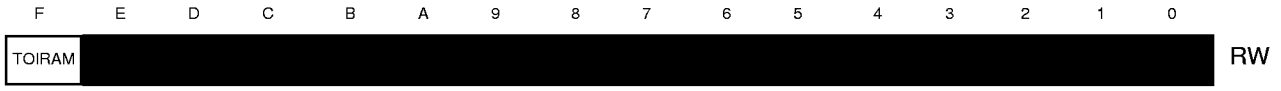
Note: TAA is the address access time.
TCS is the chip select access time.
TRAC is the access time from RAS\.
TCAC is the access time from CAS\.

6.9 MMACR : MASS MEMORY ACCESS CONTROL REGISTER (MAPPED TO IO REGISTER 9)

6.10 MMAPL : Mass Memory Access Pointer Low register (mapped to IO regsiter 10)



6.11 MMAPH : Mass Memory Access Pointer High register (mapped to IO register 11)



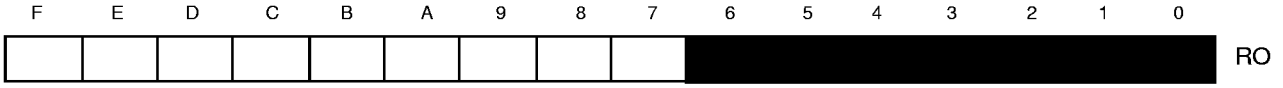
Writing (OUT) a non zero value into MMCNT (REG 9) will start DATA movement between external DRAM and internal data RAM and hold DSP operation till MMCNT=0. The starting address of data RAM and DRAM are pointed by IRA (REG 9) and MMAPH+MMAPL (REG 10 and 11). Data movement will stop when DRAM address reach MMAPH+MMAPL+MMCNT. Total data words being moved depend on what the DRAM configuration is.
Only data in RAM bank 1 can be moved around. The direction of movement is decided by TOIRAM(REG 11).

TOIRAM=1, DRAM --> INTERNAL RAM

TOIRAM=0, DRAM<-- INTERNAL RAM

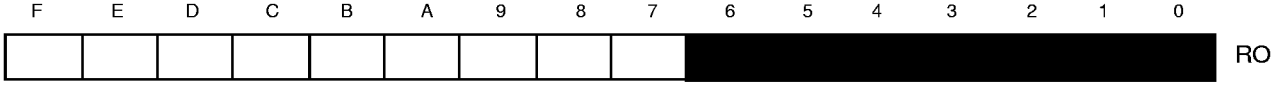
Total 30 bits of MMAPH+MMAPL can address up to 1 Giga DRAM space.

6.12 RCR : Repeat Counter Register (mapped to IO register 12)



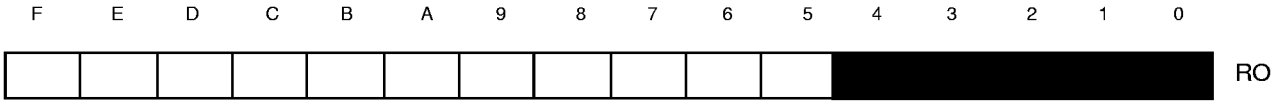
- RCR provides repeat count on, LUP
- RCR must be prepared before macro instructions are being executed. (RPT instruction)
- Repeat time is RCR+1

6.13 MODR : MODULAR REGISTER (MAPPED TO IO REGISTER 13)



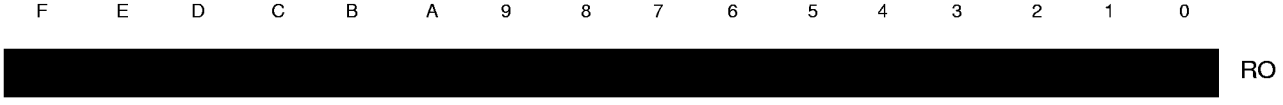
As MODR=M → 0, 1, 2, .., M-1, M modulo mechanism will be enforced (note: bounded by M --- not M-1)
Modular addressing is always performed at the output of ARAU.
As MODR=0, modulo addressing is disabled.
MOD type instructions are used to load MODR; use IN instruction to read MODR.

6.14 SPR : Stack Pointer Register (mapped to IO register 15)

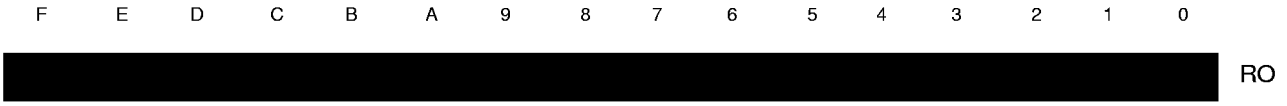


- 32-level stack provides nestable interrupt and controller level nested call capabilities.
- SPR is pointing to 'next-available' word, and initialized to 0.
- As SPR is over address 31, it wraps around to 0. As SPR=0, POP will also wrap SPR to 31.
- SPR can only be read by IN instructions; no write capability.

6.15 CDRR 0: Codec Data Receive Register (Mapped to IO register 16)

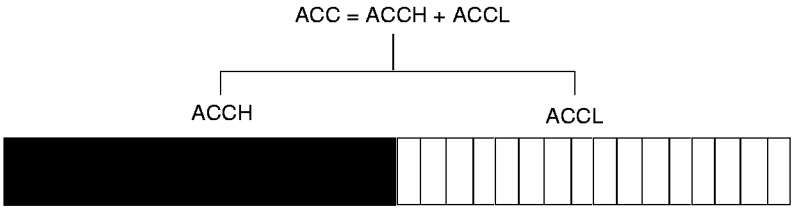


6.16 CDXR 0: Codec Data Transmit Register (Mapped to IO register 17)



1. Two Codec events from the above registers are always synchronized, so there is only one Codec interrupt for them.
2. These codecs are in 16-bit mechanism; however, 8-bit Codec is also applicable.
   In 8-bit case, to tx, the data byte to be transmitted must be in bit15~bit8. The received data byte is at bit15~bit8 as read from receive register.
3. MSB (Most-Significant-Bit) is shifted first.
4. Codec registers has shadow registers as buffer.

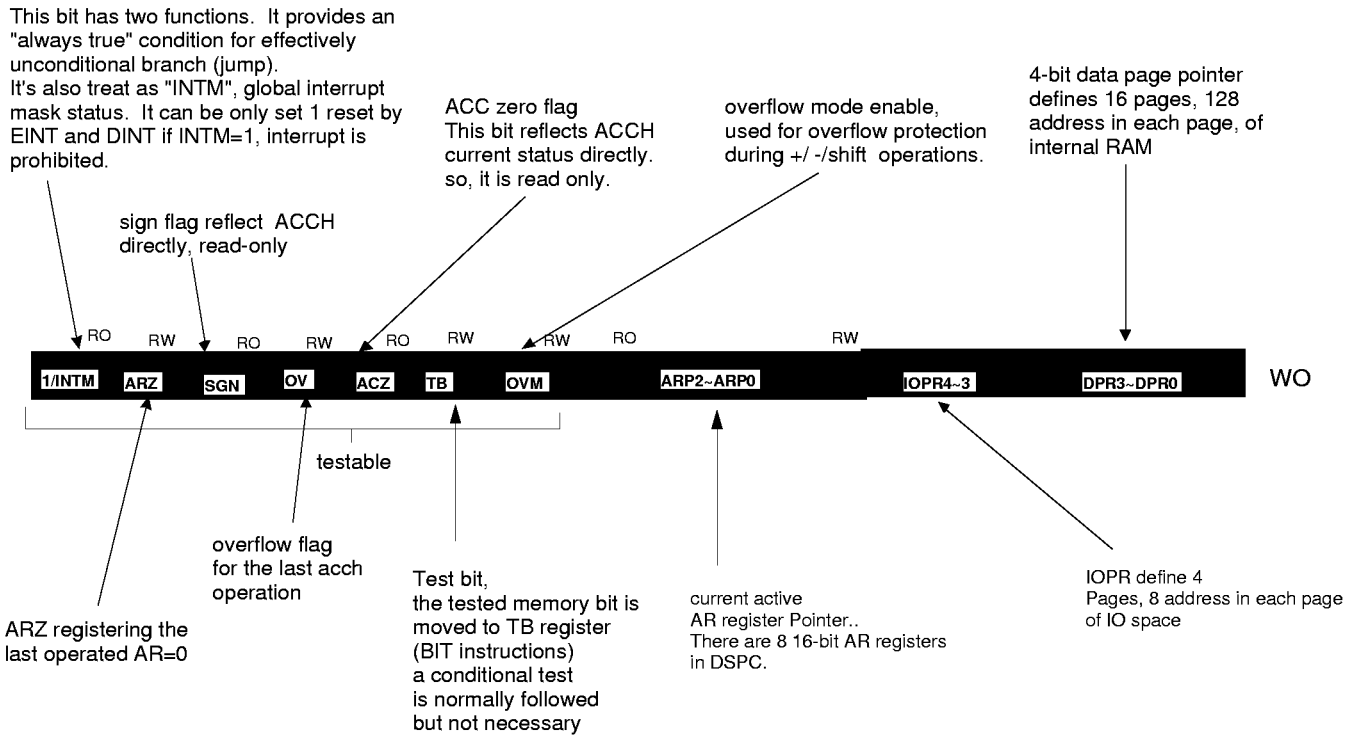
6.17 ACC:ACCUMULATOR



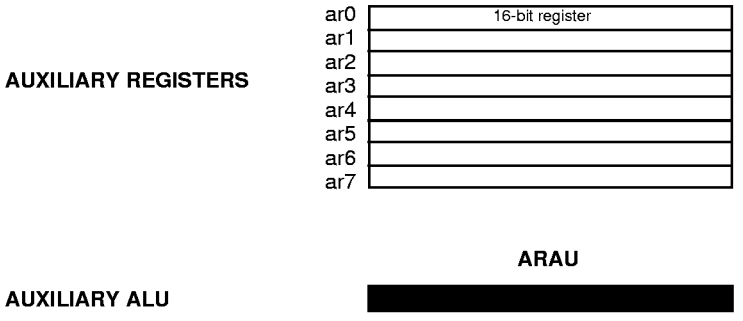
- acch+accl=acc
- Logic ALU operation is 16 bits and executed on acch. (ACCL is not affected)
- ADL/SBL is 32-bit operation. (SVR:SNSEL determine sign-extended)
- Lac will put ACCL to 0

6.18 SSR : STATUS REGISTER

ssr includes 8 testable status/register bits (ssr:15~8), and 3 arp bits, 2 IOP bits, 4 DP bits
SGN and ACZ reflect status of ACCH. (can not be saved)



6.19 AR (AUXILIARY REGISTER) AND ARAU (AUXILIARY ALU)



- 16x8 AR registers provide powerful indirect memory access.
- Modulo addressing (modulo memory indexing) provides easy implementation of ring buffer or delay line. See modulo register (MODR) for more details.
- ARAU provides +/ - (0, 1, 2, AR0) post execution after each addressing of ARs.
- ARAU works in parallel with main ALU.
- ARs may be also used as scratch registers.

6.20 PC AND PROGRAM FLOW CONTROL

Program flow is affected by:

1. BS/BZ (branch-if-set/branch-if-zero) conditional branch.
2. BACC - branch indirectly by ACCH.
3. CALA - call indirectly by ACCH, return address is pushed.
4. CALL - call subroutine, see 'Addressing Modes, Misc. Addressing mode'
5. TRAP - Trapped to call fixed hex 000C address.
6. Power-on reset and interrupt see 'interrupt Operations'

7.0 INSTRUCTION SET SUMMARY

ABBREVIATIONS

a	:	AR pointer
ar	:	AR
acc	:	accumulator
c	:	short constant
d	:	data memory address
dp	:	data page pointer
i	:	Addressing mode select bit
k	:	odd/even address select
l	:	loop counter
L	:	constant for shift left
mr	:	modulo register
o	:	io page pointer
pa	:	port address
pc	:	program counter
R	:	constant for shift right
rc	:	repeat counter
s	:	shift right sign extention select bit
sp	:	stack pointer
ss	:	status register
sv	:	shift register
v	:	AR arithemetic operation selection
x	:	don't care
y	:	Next AR arithmetic register selection

Mnemonic and Description			Words & cycles	16-bit opcode MSB			LSB
abs	;	absolute value of accumulator	1,1	1001	1000	0xxx	xxxx
adh	;	add to high acc	1.1	0000	0000	iddd	dddd
adhk	;	add to high acc. short immediate	1,1	0000	0001	0ccc	cccc
adhl	;	add to high acc. immediate	2,2	1000	0000	0xxx	xxxx
adl	;	add to low acc	1,1	0000	0010	iddd	dddd
adlk	;	add to low acc. short immediate	1,1	0000	0011	0xxx	xxxx
adll	;	add to low acc. immediate	2,2	1000	0001	0xxx	xxxx
and	;	and with high acc	1,1	0000	1010	iddd	dddd
andk	;	and short immediate with high acc	1,1	0000	1011	0ccc	cccc
andl	;	and immediate with high acc	2,2	1000	0101	0xxx	xxxx
bacc	;	branch to address specified by acc	1,2	1111	1010	0xxx	xxxx
bit	;	test bit	1,1	0110	bbbb	iddd	dddd

		Mnemonic and Description	Words & cycles	16-bit opcode MSB			LSB
bs	;	branch immediate if bit set	2,3	1101	1bbb	0xxx	xxxx
bz	;	branch immediate if bit reset	2,3	1101	0bbb	0xxx	xxxx
cala	;	call subroutine indirect specified by acc		1,2	1100	0000	0xxx
xxxx							
call	;	call subroutine	2,3	1111	1100	0000	0000
dint	;	disable interrupt	1,1	1111	0000	0xxx	xxxx
eint	;	enable interrupt	1,1	1111	0001	0xxx	xxxx
in	;	input data from port	1,1	1010	ppp0	iddd	dddd
lac	;	load acc	1,1	0000	1110	iddd	dddd
lack	;	load acc. short immediate	1,1	0000	1111	0ccc	cccc
lacl	;	load acc. immediate	2,2	1000	0111	0xxx	xxxx
lar	;	load auxiliary register	1,1	0111	aaa0	iddd	dddd
lark	;	load auxiliary register short immediate	1,1	0111	aaa1	0ccc	cccc
larl	;	load auxiliary register immediate	2,2	1000	1000	0aaa	0000
ldp	;	load data page register	1,1	0001	0100	iddd	dddd
ldpk	;	load short immediate to data page register	1,1	0001	0101	0xxx	cccc
		lip	;	load io page register			1,1
0001	0010		iddd	dddd			
lipk	;	load io page register with short immediate	1,1	0001	0011	0xxo	oxxx
	lup ;		loop instruction	1,1	0101	lll0	iddd
dddd							
lupk	;	load rc with 7-bit constant and enable loop operation	1,1	0101	lll1	0ccc	cccc
mar	;	modify auxiliary register	1,1	1111	0110	1ddd	dddd
mod	;	load modulo register	1,1	0001	0110	iddd	dddd
modk	;	load modulo register short immediate	1,1	0001	0111	0ccc	cccc
nop	;	no operation	1,1	1111	1111	1111	1111
or	;	or with high acc	1,1	0000	1000	iddd	dddd
ork	;	or short immediate with high acc	1,1	0000	1001	0ccc	cccc
orl	;	or immediate with high acc	2,2	1000	0100	0xxx	xxxx
out	;	output data to port	1,1	0100	ppp0	iddd	dddd
outk	;	output short immediate to port	1,1	0100	ppp1	0ccc	cccc
outl	;	output immediate to port	2,2	1000	1111	0ppp	0000
pop	;	pop top of stack to data memory	1,1	1011	0101	iddd	dddd
poph	;	pop top of stack to high accumulator	1,1	1001	0100	0xxx	xxxx
popl	;	pop top of stack to low accumulator	1,1	1001	1011	0xxx	xxxx
psh	;	push data memory value onto stack	1,1	1100	1010	iddd	dddd
pshh	;	push high accumulator onto stack	1,1	1100	1000	1vvv	vvvv
pshl	;	push low accumulator onto stack	1,1	1100	1001	1vvv	vvvv
ret	;	return from subroutine	1,2	1111	1000	0xxx	xxxx
reti	;	return from interrupt	1,2	1111	1001	0xxx	xxxx
rpt	;	load repeat counter	1,1	0001	0000	iddd	dddd
rptk	;	load rc with 7-bit constant	1,1	0001	0001	0ccc	cccc
rxf	;	reset external flag	1,1	1111	0010	0xxx	xxxx

	Mnemonic and Description	Words & cycles	16-bit opcode MSB			LSB
sah ;	store high acc.	1,1	1011	0000	iddd	dddd
sal ;	store low acc	1,1	1011	0001	iddd	dddd
sar ;	store auxiliary register	1,1	1011	1aaa	iddd	dddd
sbh ;	subtract from high acc	1,1	0000	0100	iddd	dddd
sbhk ;	subtract short immediate from high acc	1,1	0000	0101	0ccc	cccc
sbhl ;	subtract immediate from high acc	2,2	1000	0010	0xxx	xxxx
sbl ;	subtract from low acc	2, 2	0000	0110	iddd	dddd
sblk ;	subtract short immediate from low acc	1,1	0000	0111	0ccc	cccc
sbll ;	subtract immediate from low acc	1,1	1000	0011	0xxx	xxxx
sdp ;	store datapage register	1,1	1011	0100	iddd	dddd
sfl ;	shift acc left	1,1	1001	1101	0000	LLLL
sfr/sfrs ;	shift acc right	1,1	1001	1110	000s	RRRR
sip ;	store iopage register	1,1	1011	0010	iddd	dddd
sss ;	store ss register	1,1	1011	0011	iddd	dddd
sxf ;	set external flag	1,1	1111	0011	0xxx	xxxx
trap ;	software interrupt	1,2	1100	0010	0xxx	xxxx
xor ;	xor with high acc	1,1	0000	1100	iddd	dddd
xork ;	xor short immediate with high acc	1,1	0000	1101	0ccc	cccc
xorl ;	xor immediate with high acc	2,2	1000	0110	0xxx	xxxx

8.0 INSTRUCTION SET DESCRIPTION

abs **absolute value of accumulator.**

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	0	1	1	0	0	0	0							

SYNTAX: ABS

EXECUTION: (pc) + 1 → pc
|acc(31:16)| → (acc (31:16))

WORDS: 1

CYCLES: 1

adh **add to high acc.**

direct:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0	0	data memory address						

indirect:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0	1	see note 1						

SYNTAX: adh dma7
adh *(,nar)

EXECUTION: (pc) + 1 → pc
(acc(31:16))+(dma) → (acc (31:16))

WORDS: 1

CYCLES: 1(DI) 2(DE)

Adhk **add to high acc. Short immediate.**

BIT:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	1	0	7-bit constant						

SYNTAX: adhk cnst 7

EXECUTION: (pc) + 1 → pc
(acc(31:16)) + (7-bit constant) → (acc (31:16))

WORDS: 1

CYCLES: 1

adhl **add to high acc. Immediate.**

BIT:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0							
16-bit constant															

SYNTAX: adhl cnst16

EXECUTION: (pc) + 2 → pc
(acc(31:16))+(16-bit constant) → (acc (31:16))

WORDS: 2

CYCLES: 2

adl **add to low acc.**

Direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	0	data memory address						

Indirect:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	1	see note 1						

SYNTAX: adl dma7
adl *(,nar)

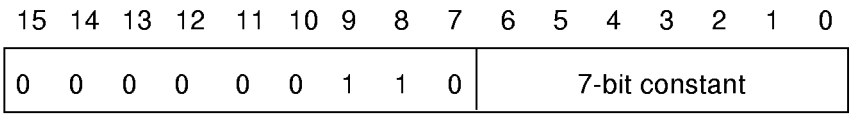
EXECUTION: (pc) + 1 → pc
(acc)+(dma with optional MSBs sign extension) → (acc)

WORDS: 1

CYCLES: 1(DI) 2(DE)

NOTE: Option is controlled by CTRL: SNSEL bit

adlk **add to low acc. Short immediate.**



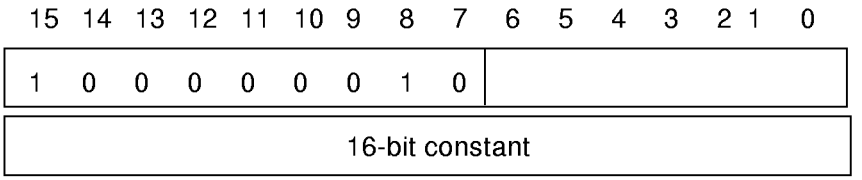
SYNTAX: adlk cnst7

EXECUTION: (pc) + 1 → pc
(acc)+(7-bit constant) → (acc)

WORDS: 1

CYCLES: 1

adll **add to low acc. Immediate.**



SYNTAX: adll cnst16

EXECUTION: (pc) + 2 → pc
(acc)+(16-bit constant with optional MSBs sign extension*) → (acc)

WORDS: 2

CYCLES: 2

Note:option is controlled by CTRL :SENSE bit

and **and with high acc.**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	0	data memory address						

indirect:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	1	see note 1						

SYNTAX:
and dma7
and *(,nar)

EXECUTION:
(pc) + 1→ pc
(acc(31:16)) .and. (dma) → (acc(31:16))

WORDS:
1

CYCLES:
1(DI) 2(DE)

andk **and short immediate with high acc.**

BIT:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	1	0	7-bit constant						

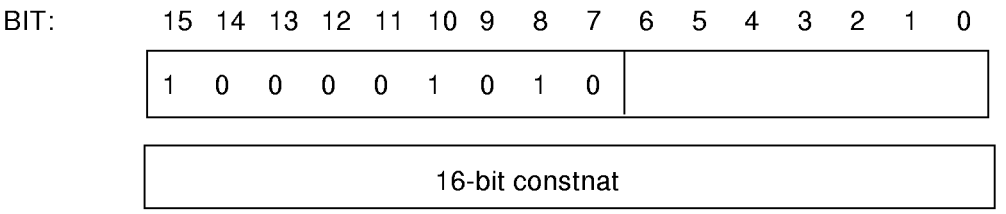
SYNTAX:
andk cnst7

EXECUTION:
(pc) + 1 → pc
(acc(23:16) .and. (7-bit constant) → (acc(23:16))
0 → acc(31:24)

WORDS:
1

CYCLES:
1

andl **and immediate with high acc.**



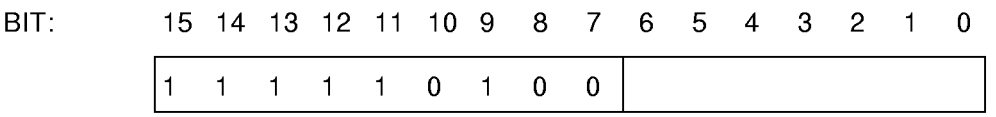
SYNTAX: andl cnst16

EXECUTION: (pc) + 2 → pc
(acc(31:16)) .and. (16-bit constant) → (acc(31:16))

WORDS: 2

CYCLES: 2

bacc **branch to address specified by acc.**



SYNTAX: bacc

EXECUTION: (acc (31:16)) → pc

WORDS: 1

CYCLES: 2

bit **test bit.**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	bbbb				0	data memory address						

indirect

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	bbbb				1	see note 1						

SYNTAX:
bit dma7, bbbb
bit *,bbbb (,nar)

EXECUTION:
(pc) + 1 → pc
(dma) → ss(tb)

WORDS:
1

CYCLES:
1(DI) 2(DE)

bs **branch immediate if bit set.**

BIT:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	1	bbb			0							
program memory address															

SYNTAX:
bbb, pma16

EXECUTION:
if ss(#1bbb)=1
then (pma) → pc
else (pc)+2 → pc

WORDS:
2

CYCLES:
3

bz **branch immediate if bit reset.**

BIT:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	bbb			0							
program memory address															

SYNTAX: bz bbb, pma16

EXECUTION: if ss(#1bbb)=0
then (pma) → pc
else (pc)+2 → pc

WORDS: 2

CYCLES: 3

cala **call subroutine indirect.**

BIT:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	0							

SYNTAX: cala

EXECUTION: (pc)+1 → (sp)
(acc(31:16))→ pc

WORDS: 1

CYCLES: 2

call **subroutine .**

BIT:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
16-bit constant															

SYNTAX: call pma16

EXECUTION: (pc)+1 → (sp)
(16-bit constant)→ pc

WORDS: 2

CYCLES: 3

dint **disable interrupt.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0							

SYNTAX: dint

EXECUTION: (pc) + 1 → pc
1 → (INTM) status bit

WORDS: 1

CYCLES: 1

eint **enable interrupt.**

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	0	0	1	0							

SYNTAX: eint

EXECUTION: (pc) + 1 → pc
0 → (INTM) status bit

WORDS: 1

CYCLES: 1

in **input data from port.**

direct:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	1	0	port address			0	0	data memory address						

indirect:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	1	0	port address			0	1	see note 1						

SYNTAX: in dma7,port
in *,port(,nar)

EXECUTION: (pc) + 1 → pc
port address → a2-a0
(IOPR(4:3)) → a4-a3
0 → a15-a6
(IOR) → dma

WORDS: 1

CYCLES: 1; note:only for internal memory

lac **load acc.**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	0	0	data memory address						

indirect:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	0	1	see note 1						

SYNTAX: lac dma7
lac *(,nar)

EXECUTION: (pc) + 1 → pc
(dma) → acc(31:16)
0 → acc(15:0)

WORDS: 1

CYCLES: 1(DI) 2(DE)

lack **load acc. short immediate.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	0	7-bit sonstant						

SYNTAX: lack cnst7

EXECUTION: (pc) + 1 → pc
(7-bit constant) → acc(23:16)
0 → acc(31:24)
0 → acc(15:0)

WORDS: 1

CYCLES: 1

lacl **load acc. Immediate**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	1	1	1	0							
16-bit constant															

SYNTAX: lacl cnst16

EXECUTION: (pc) + 2 → pc
(16-bit constant)→ acc(31:16)
0 → acc(15:0)

WORDS: 2

CYCLES: 2

lar **load auxiliary register.**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	arp			0	0	data memory address						

indirect:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	arp			0	1	see note 1						

SYNTAX:
lar dma7, arp
lar *,arp(,nar)

EXECUTION:
(pc) + 1 → pc
(dma)→ (ar)

WORDS: 1

CYCLES: 1(DI) 2(DE) ; no manipulation on ars

lark **load auxiliary register short immediate.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	arp			1	0	7-bit constant						

SYNTAX:
lark cnst7, arp

EXECUTION:
(pc) + 1 → pc
(7-bit constant) → (ar(6:0))
0 → ar(15:7)

WORDS: 1

CYCLES: 1

larl **load auxiliary register immediate.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0	0	arp			0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16-bit constant															

SYNTAX: larl cnst16, arp

EXECUTION: (pc) + 2 → pc
(16-bit constant)→ ar (15:0)

WORDS: 2

CYCLES: 2

ldp **load data-page register.**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	0	0	0	data memory address						

indirect:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	0	0	1	see note 1						

SYNTAX: ldp dma7
ldp *(,nar)

EXECUTION: (pc) + 1 → pc
(dma(3:0))→ (dp(3:0))

WORDS: 1

CYCLES: 1(DI) 2(DE)

ldpk **load short immediate to data page register.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	0	1	0	x	x	x	4-bit constant			

SYNTAX: ldpk cnst4

EXECUTION: (pc) + 1 → pc
(4-bit constant)→ (dp(3:0))

WORDS: 1

CYCLES: 1

lip **load io page register**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	1	0	0	data memory address						

indirect:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	1	0	1	see note 1						

SYNTAX: lip dma7
lip *(,nar)

EXECUTION: (pc) + 1 →pc
(dma)→ (iop(1:0))

WORDS: 1

CYCLES: 1(DI) 2(DE)

lipk **load io page register with short immediate.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	1	1	0	x	x	s1	s0	x	x	x

SYNTAX: lipk cnst2

EXECUTION: (pc) + 1→ pc
s1→ iop(1), s0 → iop(0)

WORDS: 1

CYCLES: 1

lup **loop instruction.**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	loop number			0	0	data memory address						

indirect:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	loop number			0	1	see note 1						

SYNTAX: lup dma, lic
lup *,lic(,nar)

EXECUTION: (pc) + 1 → pc
(dma)→ (rc)
(loop number) → (loop counter)

WORDS: 1

CYCLES: 1(DI) 2(DE); the next (loop number+1) words will be repeat (rc+1) times

lupk **load rc with 7-bit constant and enable loop operation.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	loop number			1	0	7-bit constant						

SYNTAX: lupk cnst7, lic

EXECUTION: (pc) + 1 → pc
(7-bit constant) → (rc)
(loop number) → (loop counter)

WORDS: 1

CYCLES: 1

mar **modify auxiliary register.**

indirect:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	1	0	1	see note 1						

SYNTAX: MAR *(,nar)

EXECUTION: (pc) + 1 → pc
modifies arp, ar(arp) as specified by the indirect addressing field

WORDS: 1

CYCLES: 1

mod **load modulo register.**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	1	0	0	data memory address						

indirect

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	1	0	1	see note 1						

SYNTAX: mod dma7
mod *(,nar)

EXECUTION: (pc) + 1 → pc
(dma(6:0))→ mr(6:0)

WORDS: 1

CYCLES: 1(DI) 2(DE)

modk **load modulo register short immediate.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	1	1	0	7-bit constant						

SYNTAX: modk cnst7

EXECUTION: (pc) + 1 → pc
(7-bit constant)→ mr(6:0)

WORDS: 1

CYCLES: 1

nop **no operation.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

SYNTAX: nop

EXECUTION: (pc) + 1 → pc

WORDS: 1

CYCLES: 1

or **or with high acc.**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	0	data memory address						

indirect:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	1	see note 1						

SYNTAX: or dma7
or *(,nar)

EXECUTION: (pc) + 1 → pc
(acc(31:16)).or. (dma) → (acc(31:16))

WORDS: 1

CYCLES: 1(DI) 2(DE)

ork **or short immediate with high acc.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	1	0	7-bit constant						

SYNTAX: ork cnst7

EXECUTION: (pc) + 1 → pc
(acc(23:16)).or. (7-bit constant)→ (acc(23:16)
(acc(31:24)) → acc(31:24)

WORDS: 1

CYCLES: 1

orl **or immediate with high acc.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	1	0	0	0							
16-bit constant															

SYNTAX: orl cnst16

EXECUTION: (pc) + 2 → pc
(acc(31:16)) .or. (16-bit constant)→ (acc(31:16)

WORDS: 2

CYCLES: 2

out **output data to port.**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	port address			0	0	data memory address						

indirect:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	port address			0	1	see note 1						

SYNTAX:
out dma7, port
out port *(,nar)

EXECUTION:
(pc) + 1 → pc
(pa) → address bus a1-a0
(IOPR)(4:3) → a4-a0
0 → a15-a5

WORDS: 1

CYCLES: 1;note: For internal memory only

outk **output short immediate to port.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	port address			1	0	7-bit constant						

SYNTAX: outk cnst7, port

EXECUTION:
(pc) + 1 → pc
(pa) → address bus a2-a0
(IOPR)(4:3) → a4-a3
0 → a15-a5
(7-bit constant)→ IOR (addressed by a4-a0)

WORDS: 1

CYCLES: 1; note: For internal memory only

outl **output immediate to port.**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	1	1	1	0	port address			0	0	0	0
16-bit constant															

SYNTAX: outl cnst16, port

EXECUTION:
(pc) + 1 → pc
(pa) → address bus a2-a0
(IOPR)(4:3) → a4-a3
0 → a15-a5
(16-bit constant) → IOR(addressed by a4-a0)

WORDS: 1

CYCLES: 1;note: for internal memory only

poph **pop top of stack to high accumulator.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	1	0	0	0							

SYNTAX: poph

EXECUTION:
(pc) + 1 → pc
(tos) → acc(31:16)

WORDS: 1

CYCLES: 1

popl **pop top of stack to low accumulator.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	0	1	1	0							

SYNTAX: popl

EXECUTION: (pc) + 1 → pc
(tos) → acc(15:0)

WORDS: 1

CYCLES: 1

pop **pop top of stack to data memory.**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	1	0	1	0	data memory address						

indirect:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	1	0	1	1	see note 1						

SYNTAX:
pop dma
pop *(,nar)

EXECUTION: (pc) + 1 → pc
(tos) → dma

WORDS: 1

CYCLES: 1(DI) 2(DE)

psh **push data memory value onto stack.**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	0	0	data memory address						

indirect:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	0	1	see note 1						

SYNTAX:
psh dma
psh *(,nar)

EXECUTION:
(pc) + 1 → pc
dma → (tos)

WORDS: 1

CYCLES: 1

pshh **push high accumulator onto stack.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	0	0	0	1	see note 1						

SYNTAX: pshh

EXECUTION:
(pc) + 1 → pc
acc(31:16) → (tos)

WORDS: 1

CYCLES: 1

pshl **push low accumulator onto stack.**

Bits:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	0	0	1	1	see note 1						

SYNTAX: pshl

EXECUTION: (pc) + 1 → pc
acc(15:0) → (tos)

WORDS: 1

CYCLES: 1

ret **return from subroutine.**

Bits:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	0	0							

SYNTAX: ret

EXECUTION: (sp) → pc
sp-1 → sp

WORDS: 1

CYCLES: 2

reti **return from interrupt.**

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	1	0	0	1	0							

SYNTAX: reti

EXECUTION:
(sp) → pc
(sp)-1 → sp
(sp) → ss
sp-1→ sp

WORDS: 1

CYCLES: 2

rpt **load repeat counter.**

direct:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	1	0	0	0	0	0	data memory address						

`

indirect:L	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	1	0	0	0	0	1	see note1						

SYNTAX:
rpt dma 7
rpt *(,nar)

EXECUTION:
(pc) + 1 → pc
(dma) → (rc)

WORDS: 1

CYCLES: 1(DI) 2(DE)

rptk **load rc with 7-bit constant.**

direct:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	1	0	0	0	1	0	7-bit constant						

SYNTAX: rptk cnst7

EXECUTION:
(pc) + 1 → pc
(7-bit constant) → (rc)

WORDS: 1

CYCLES: 1

rxf **reset external flag.**

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	0	1	0	0							

SYNTAX: rxf

EXECUTION: (pc) + 1 → pc
0 → (XF) pin and status bit.

WORDS: 1

CYCLES: 1

sah **store high acc.**

direct:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	1	1	0	0	0	0	0	data memory address						

indirect:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	1	1	0	0	0	0	1	see note 1						

SYNTAX:
sah dma7
sah *(,nar)

EXECUTION: (pc) + 1 → pc
(acc(31:16)) → (dma)

WORDS: 1

CYCLES: 1(DI) 2(DE)

sal **store low acc.**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	0	0	1	0	data memory address						

indirect:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	0	0	1	1	see note 1						

SYNTAX:
sal dma7
sal *(,nar)

EXECUTION:
(pc) + 1 → pc
(acc(15:0)) → (dma)

WORDS: 1

CYCLES: 1(DI) 2(DE)

sar **store auxiliary register.**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	ar			0	data memory address						

indirect:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	ar			1	see note 1						

SYNTAX:
sar dma7, arp
sar *, arp (,nar)

EXECUTION:
(pc)+1 → pc
(ar) → (dma)

WORDS: 1

CYCLES: 1(DI) 2(DE)

sbh **subtract from high acc.**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	0	data memory address						

indirect:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	1	see note 1						

SYNTAX:

sbh dma7
sbh *(,nar)

EXECUTION:

(pc) +1 → pc
(acc(31:16)) - (dma) → (acc(31:16))

WORDS: 1

CYCLES: 1(DI) 2(DE)

sbhk **subtract short immediate from high acc.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	1	0	7-bit constant						

SYNTAX:

sbhk cnst7

EXECUTION:

(pc)+1 → pc
(acc(31:16)) - (7-bit constant) → (acc(31:16))

WORDS: 1

CYCLES: 1

sbhl **subtract immediate from high acc.**

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	0	0	0	0	1	0	0							
	16-bit constant															

SYNTAX: sbhl cnst16

EXECUTION: (pc)+2 → pc
(acc(31:16)) - (16-bit constant) → (acc(31:16))

WORDS: 2

CYCLES: 2

sbl **subtract from low acc.**

direct:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	1	1	0	0	data memory address						
indirect:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	1	1	0	1	see note 1						

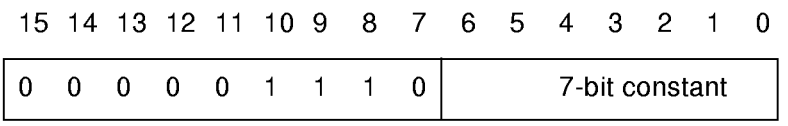
SYNTAX: sbl dma7
sbl *(,nar)

EXECUTION: (pc)+1 → pc
(acc) - (dma with optional MSBs sign extension*) → (acc)

WORDS: 1

CYCLES: 1(DI) 2(DE) ; note : Option is controlled by CTRL : SENSE bit

sblk **subtract short immediate from low acc.**



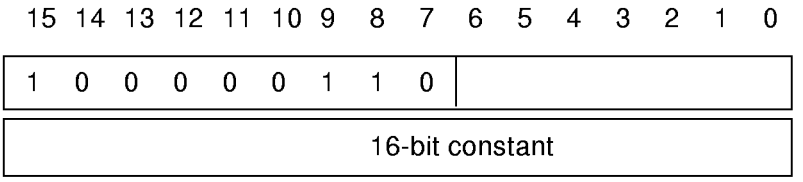
SYNTAX: sblk cnst7

EXECUTION: (pc)+1 → pc
(acc) - (7-bit constant) → (acc)

WORDS: 1

CYCLES: 1

sbll **subtract immediate from low acc.**



SYNTAX: sbll cnst16

EXECUTION: (pc)+2 → pc
(acc) - (16-bit constant with optional MSBs sign extension*) → (acc)

WORDS: 2

CYCLES: 2 ; note:Option is controlled by CTRL: SENSE bit

sdp **store data_page register.**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	1	0	0	0	data memory address						

indirect:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	1	0	0	1	see note 1						

SYNTAX:

```
sdp     dma7
sdp     *(,nar)
```

EXECUTION:

(pc)+1 → pc
(dp) → (dma)

WORDS: 1

CYCLES: 1(DI) 2(DE)

sfl **shift acc left.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	1	0	1	0	0	0	0	shift			

SYNTAX:

```
sfl     cnst4
```

EXECUTION:

(pc)+1 → pc
if (shift>< 0)
  then
    acc * (2** shift)→ acc
 else
   acc*(2**(sv(3:0))) → acc

WORDS: 1

CYCLES: 1 ; note

sfr/sfrs **shift acc right.**

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	0	1	1	1	1	0	0	0	0	s	shift			

SYNTAX:
sfr cnst4
sfrs cnst4

EXECUTION:
(pc)+1 → pc
if (shift>< 0)
then
acc * (2**(-shift))→ acc
else
acc*(2**(-sv(3:0)) → acc
* s=0 the msbs zero-filled
* s=1 the msbs sign-extended

WORDS: 1

CYCLES: 1

sip **store io_page register**

direct:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	1	1	0	0	1	0	0	data memory address						

indirect:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	1	1	0	0	1	0	1	see note 1						

SYNTAX:
sip dma7
sip *(,nar)

EXECUTION:
(pc)+1 → pc
IOPR(4:3)→ (dma (1:0))

WORDS: 1

CYCLES: 1(DI) 2(DE)

sss **store ss register.**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	0	1	1	0	data memory address						

indirect:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	0	1	1	1	see note 1						

SYNTAX:

```
sss     dma7
sss     *(,nar)
```

EXECUTION:

(pc)+1 → pc
(ss) → (dma)

WORDS: 1

CYCLES: 1(DI) 2(DE)

sxf **set external flag.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	1	1	1	0	0	1	1	0						

SYNTAX: sxf

EXECUTION:

(pc)+1 → pc
1 → (XF) pin and status bit.

WORDS: 1

CYCLES: 1

trap **software interrupt.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	1	0	0							

SYNTAX: trap

EXECUTION: (pc)+1 → sp
0c → pc

WORDS: 1

CYCLES: 2

xor **xor with high acc.**

direct:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	0	0	data memory address						

indirect:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	0	1	see note 1						

SYNTAX: xor dma7
xor *(,nar)

EXECUTION: (pc)+1 → pc
(acc(31:16)) .xor. (dma) → (acc(31:16)

WORDS: 1

CYCLES: 1(DI) 2(DE)

xork **xor short immediate with high acc.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	0	7-bit constant						

SYNTAX: xork cnst7

EXECUTION: (pc)+1 → pc
(acc(23:16)) .xor. (7-bit constant) → (acc(23:16))
(acc(31:24)) → acc(31:24)

WORDS: 1

CYCLES: 1

xorl **xor short immediate with high acc.**

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	1	1	0	0							
16-bit constant															

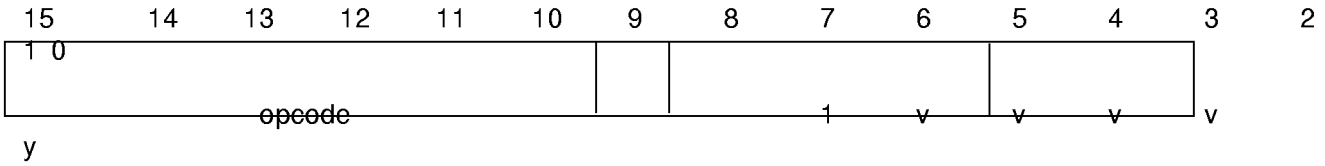
SYNTAX: xorl cnst16

EXECUTION: (pc)+2 → pc
(acc(31:16)) .xor. (16-bit constant) → (acc(31:16))

WORDS: 2

CYCLES: 2

* note 1 :



operation: case(vvvv)

0000: no manipulation of ars/arp
0001: y → arp
0010: ar(arp) - ar0 → ar(arp)
0011: ar(arp) - ar0 → ar(arp), y → arp
0100: ar(arp) + ar0 → ar(arp)
0101: ar(arp) + ar0 → ar(arp), y → arp
1000: ar(arp) +1 → ar(arp)
1001: ar(arp) +1 → ar(arp), y → arp
1010: ar(arp) - 1 → ar(arp)
1011: ar(arp) -1 → ar(arp), y → arp
1100: ar(arp) +2 → ar(arp)
1101: ar(arp) +2 → ar(arp), y → arp
1110: ar(arp) -2 → ar(arp)
1111: ar(arp) -2 → ar(arp), y → arp

operand:

+ 0
+ 0 ,y
- AR0
- AR0
+ AR0
+ AR0
+
+ , y
-
- , y
++
++ , y
--
-- , y

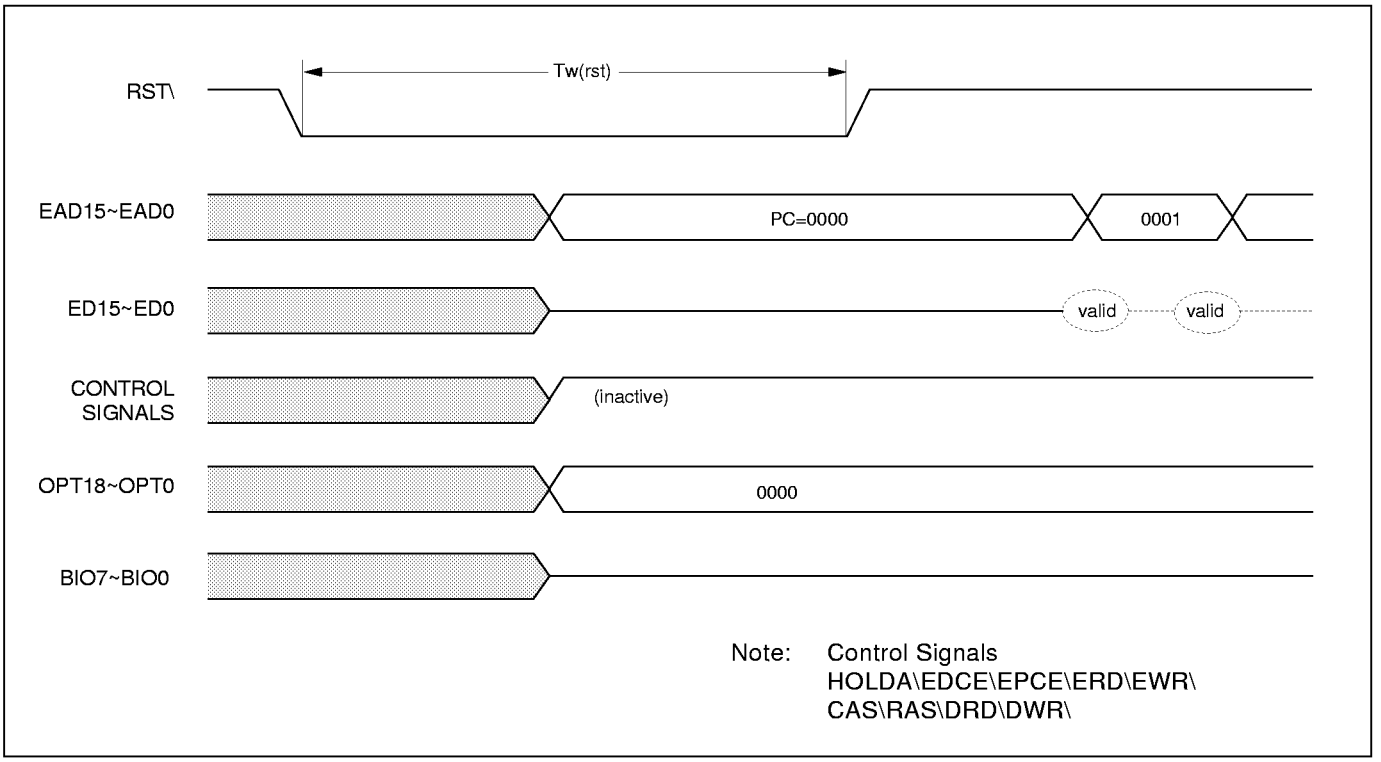
9.0 DC CHARACTERISTICS: TA = 0 to 70°C, VCC = 5V ± 10%

Storage temperature range : -55°C - 150°C

SYMBOL	PARAMETER	CONDITION	MIN	TYPE	MAX	UNIT
VCC	Supply voltage		4.5	5	5.5	V
GND	Ground			0		V
TTL LEVEL INPUT(IT)						
VIH	Input high voltage		2.0			V
VIL	Input low voltage				0.8	V
SCHMITT TRIGGER INPUT(IS)						
VIH	Input high voltage		0.7*VCC			V
VIL	Input low voltage				0.3*VCC	V
8mA OUTPUT(OA)						
VOH	Output high voltage	IOH=-8mA	2.4			V
VOL	Output low voltage	IOL= 8mA			0.4	V
16mA OUTPUT(OB)						
VOH	Output high voltage	IOH=-16mA	2.4			V
VOL	Output low voltage	IOL=16mA			0.4	V
SUPPLY CERRENT						
ICC	NORMAL			45	70	mA
ICC	HOLD MODE			10		mA
ICC	POWER DOWN			3	6	mA

10.AC TIMING AND CHARACTERISTICS:

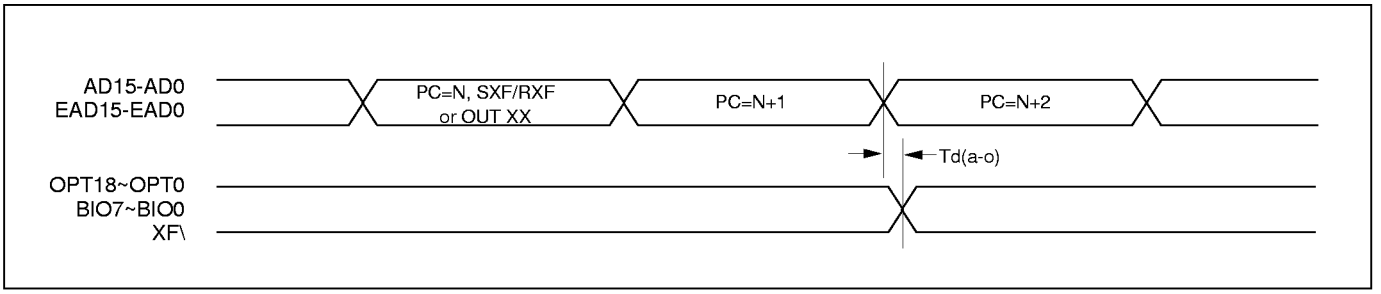
RESET TIMING



RESET TIMING

SYMBOL	PARAMETER	MIN	NOM	MAX	UNIT
Tw (rst)	Reset low pulse width	2*46.5ns			

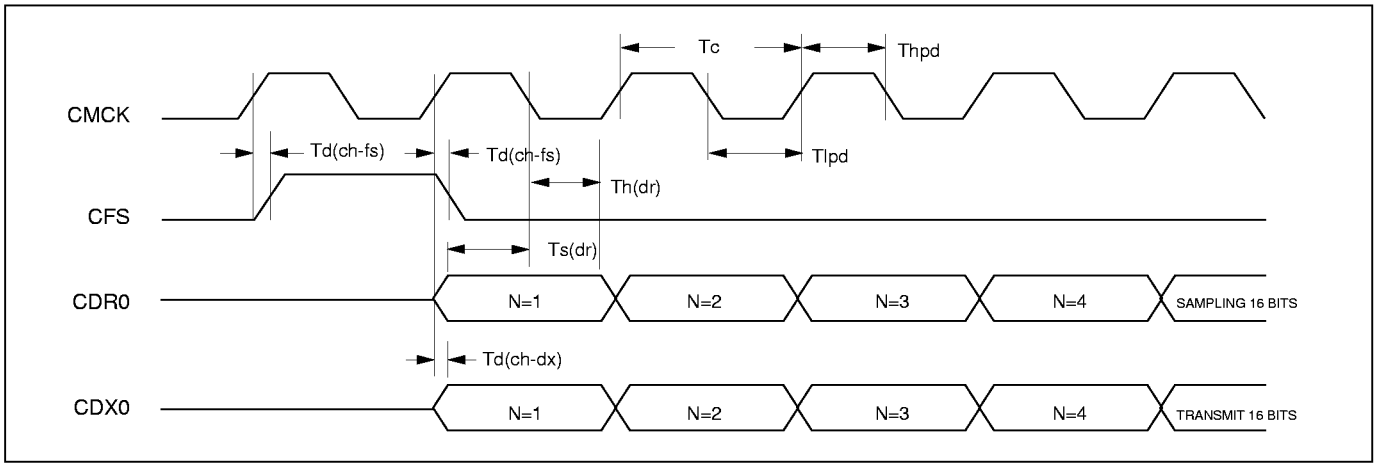
OUTPUT PORT AND EXTERNAL FLAG(XF\) TIMING



OUTPUT PORTS AND EXTERNAL FLAG (XF\) TIMING

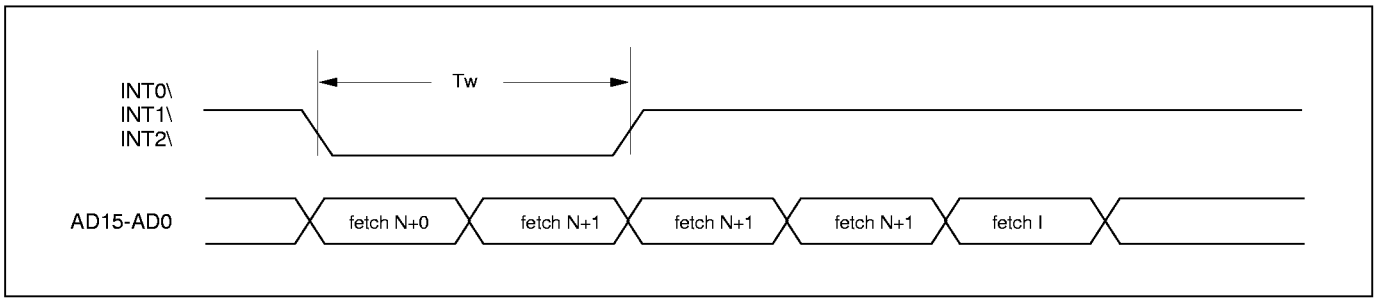
SYMBOL	PARAMETER	MIN	NOM	MAX	UNIT
Td (a-o)	Address to output ports delay time	0		10	ns

CODEC TRANSMIT AND RECEIVE TIMING



SYMBOL	PARAMETER	MIN	NOM	MAX	UNIT
Tc	CMCK cycle time		650		ns
Tlpd	CMCK low pulse duration	315		335	ns
Thpd	CMCK high pulse duration	315		335	ns
Td (ch-fs)	CMCK to CFS delay time			20	ns
Td (ch-dx)	CMCK rising edge to Dx valid			10	ns
Ts (dr)	DR set-up time before CMCK falling edge	10			ns
Th (dr)	DR hold time before CMCK falling edge	10			ns

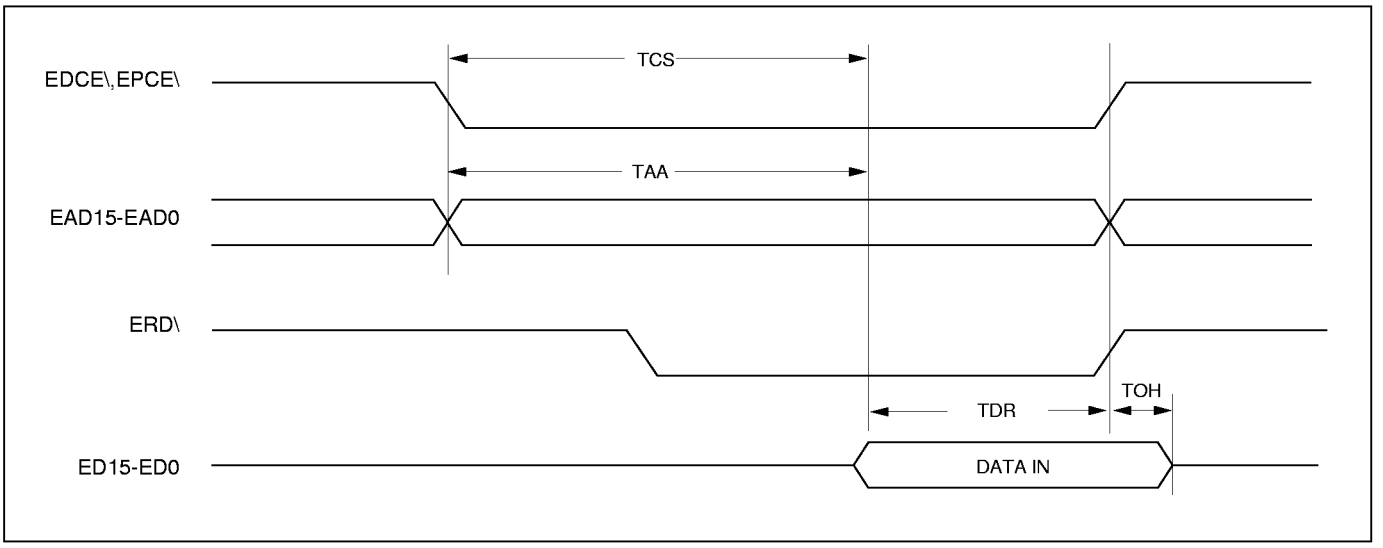
INTERRUPT TIMING



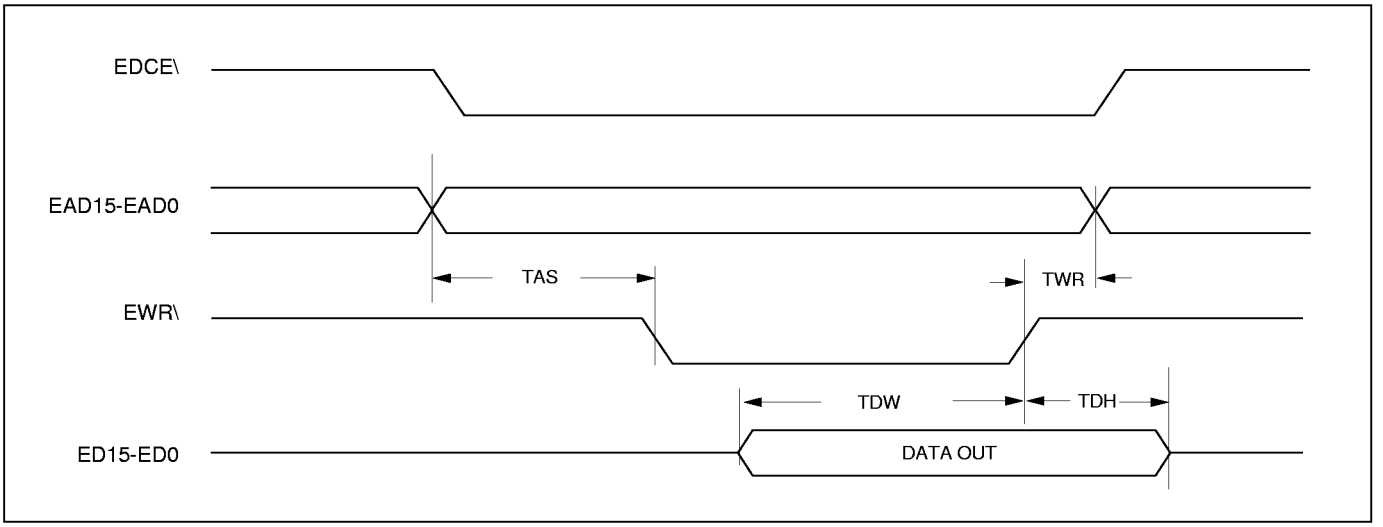
SYMBOL	PARAMETER	MIN	NOM	MAX	UNIT
Tw	INT\ low pulse duration	3Q*			ns

NOTE:Q=15.5 ns

SRAM/ROM READ TIMING



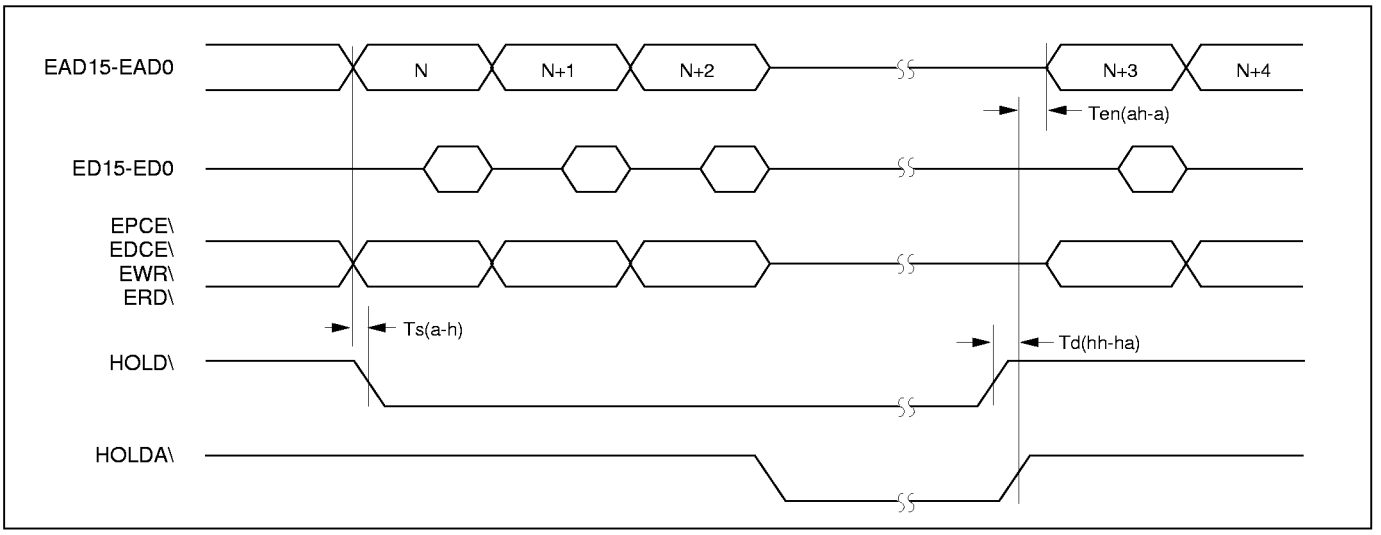
SRAM WRITE TIMING



SYMBOL	PARAMETER	MIN	NOM	MAX	UNIT
TCS	Chip select access time			26.5+WxT	ns
TAA	Address access time			26.5+WxT	ns
TDR	Data read setup time	12			ns
TOH	Data hold from end of read	0			ns
TAS	Address setup time	0		5	ns
TDW	Data to EWR\ low overlap			12	ns
TDH	Data hold from end of write	0			ns
TWR	Write recovery time	0			ns

*NOTE : T=31ns
W:wait state number

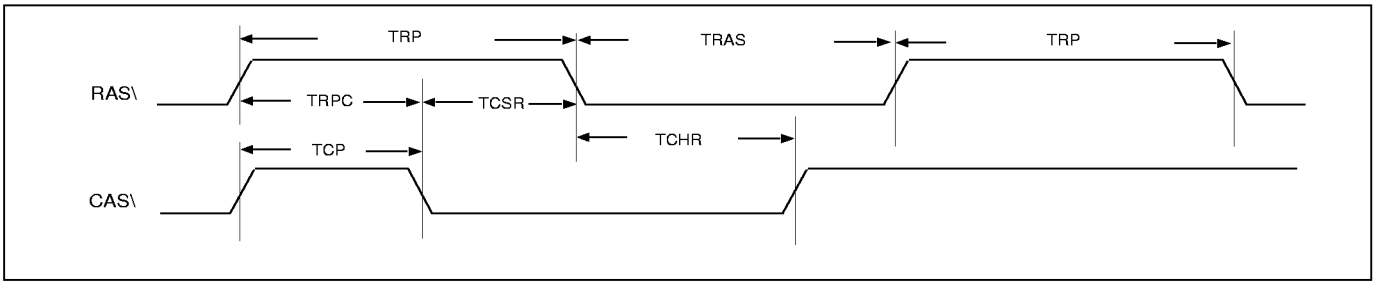
HOLD TIMING



SYMBO	PARAMETER	MIN	NOM	MAX	UNIT
Ts (a-h)	Address set-up time before HOLD\ low	5		3Q-10	ns
Td (hh-ha)	HOLD\ high to HOLDA\ high	0	1Q	1Q+10	ns
Ten (ah-a)	Address driven after HOLDA\ high	1Q-10	1Q	2Q	ns

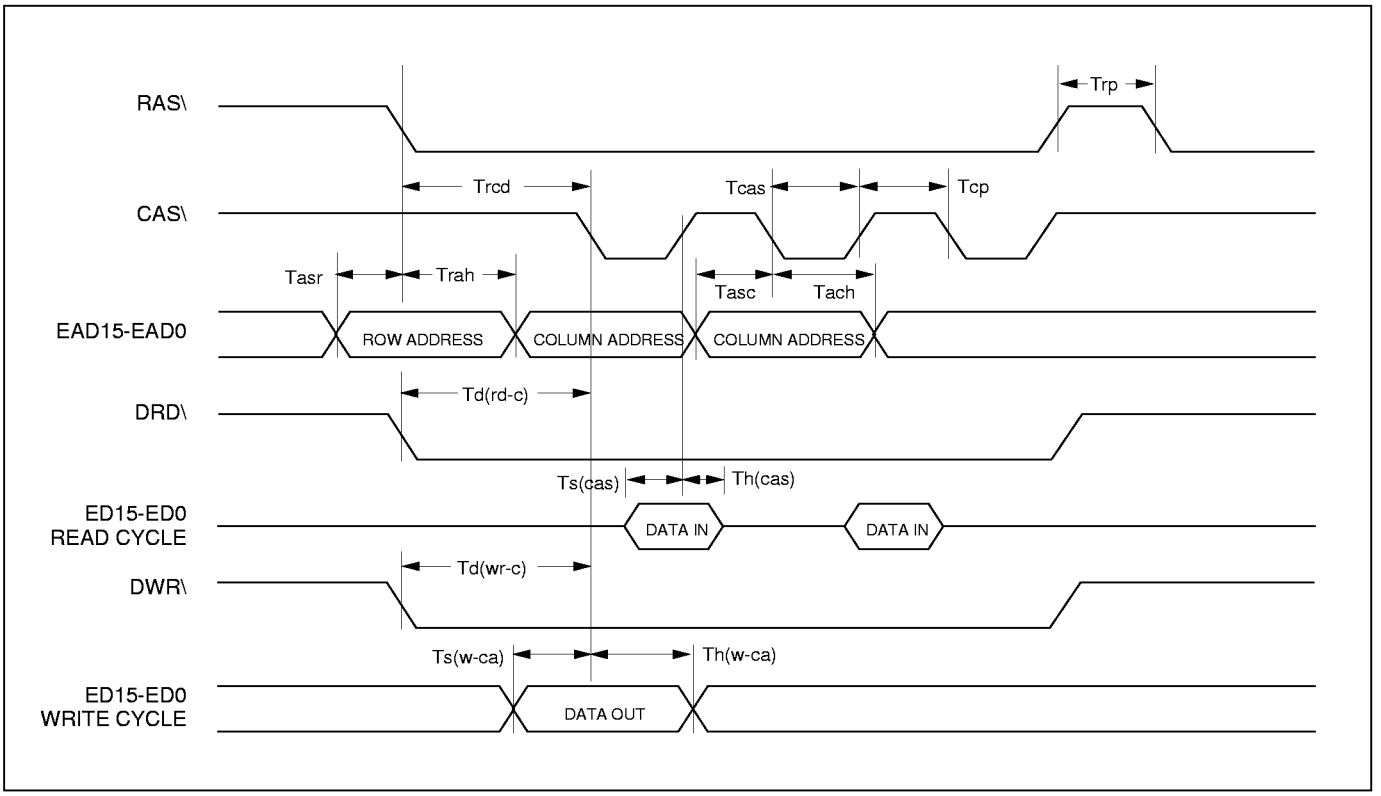
* NOTE : Q=15.5n

CAS\ BEFORE RAS\ REFRESH TIMING



SYMBO	PARAMETER	MIN	NOM	MAX	UNIT
TRP	RAS\ precharge time	77.5			ns
TRPC	RAS\ to CAS\ precharge time	62			ns
TCP	CAS\ precharge time		31		ns
TCSR	CAS\ set-up time (CBR cycle)		15.5		ns
TCHR	CAS\ hold time (CBR cycle)	62			ns
TRAS	RAS\ pulse width	108.5			ns

DRAM READ/WRITE TIMING

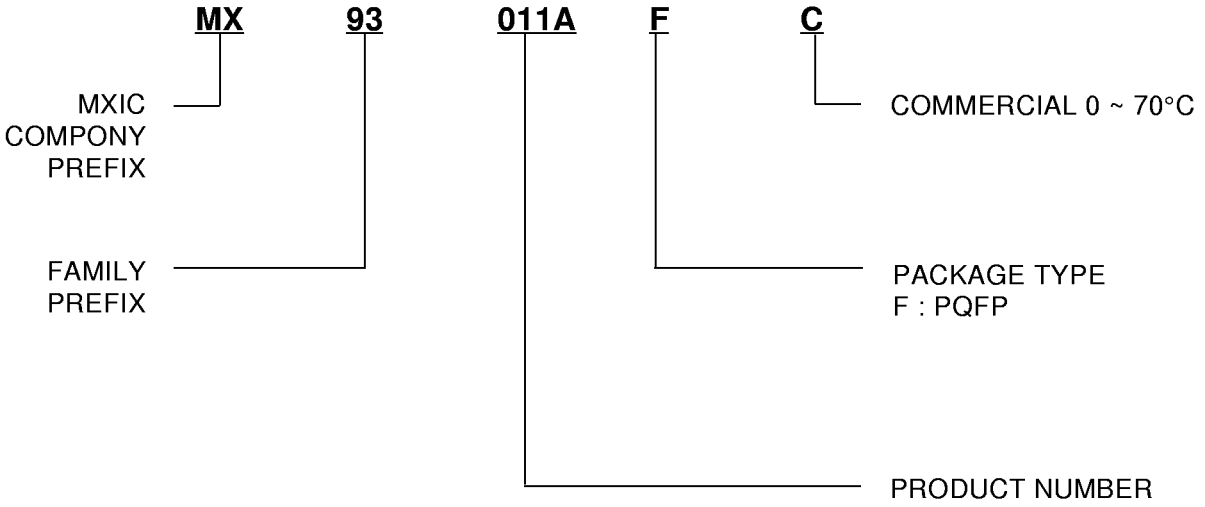


SYMBO	PARAMETER	MIN	NOM	MAX	UNIT
Trp	RAS\ precharge time	77.5			ns
Trcd	RAS\ to CAS\ delay time		62		ns
Tcas	CAS\ low pulse duration		31+W*Q		ns
Tcp	CAS\ precharge time		31		ns
Tasr	Row address set-up time	0			ns
Trah	Row address hold time	31			ns
Tasc	Column address setup time	0			ns
Tach	Column address hold time	31			ns
Td(rd-c)	DRD\ low to CAS\ low	0			ns
Td(wr-c)	DWR\ low to CAS\low	0			ns
Ts(cas)	Data set-up time before CAS\ high	20			ns
Th(cas)	Data hold time after CAS\high	0			ns
	Ts(w-ca)	Data set-up time before CAS\low			0
		ns			
Th(w-ca)	Data hold time after CAS\low	46.5			ns

*NOTE: W:Wait state number of DRAM
Q:15.5ns

12.0 ORDERING INFORMATION

PART NO.	PACKAGE
MX93011A	PQFP



13.0 PACKAGE INFORMATION

100-PIN PQFP

ITEM	MILLIMETERS	INCHES
A	24.80 ± .40	.976 ± .016
B	20.00 ± .13	.787 ± .005
C	14.00 ± .13	.551 ± .005
D	18.80 ± .40	.740 ± .016
E	12.35 [REF]	.486 [REF]
F	.83 [REF]	.033 [REF]
G	.58 [REF]	.023 [REF]
H	.30 [Typ.]	.012 [Typ.]
I	.65 [Typ.]	.026 [Typ.]
J	2.40 [Typ.]	.094 [Typ.]
K	1.20 [Typ.]	.047 [Typ.]
L	.15 [Typ.]	.006 [Typ.]
M	.10 max.	.004 max.
N	2.75 ± .15	.108 ± .006
O	.10 min.	.004 min.
P	3.30 max.	.130 max.

NOTE: Each lead centerline is located within .25mm[.01 inch] of its true position [TP] at a maximum material condition.