SINGLE-CHIP DIGITAL SIGNAL PROCESSOR

HOW TO USE THIS DATA SHEET

- **Section 1** contains ordering information and the main features of the PCB5010 and PCB5011.
- **Section 2** describes the signals of the PCB5010 and PCB5011, with block diagrams and full descriptions of what functions can be performed by each block.
- **Section 3** describes how the blocks are controlled by the instructions given by the programmer. This section is used during programming, it assumes however, a full knowledge of section 2.0. Programming can be simplified by using the software tools available.
- **Section 4** describes all the electrical characteristics.
- **Section 5** gives details of the PCB5010 and PCB5011 packages.

CONTENTS

1.0 INTRODUCTION

The PCB5010 and PCB5011 are part of our SP 50 family of digital signal processors that contains devices for various applications. These CMOS devices have a common processor structure and are accompanied by a common set of development tools.

The processor structure is characterized by two separate data buses, a two operand hardware multiply/accumulate unit and a two operand ALU to improve throughput. Powerful parallel and serial interfaces enable communication with external devices. Large on-chip data memories, each with its own programmable address computation unit (ACU), offer the possibility to make systems with only a few components.

The PCB5010 and PCB5011 are the optimal solutions for implementing DSP functions in telecommunications, and can also be used to advantage in speech processing, high-speed control, image processing and many other fields.

- PCB 5010: Version with on-chip ROM (mask programmable)
- PCB 5011: ROMless bond-out version, for use with external program/data memory

ORDERING INFORMATION

order number	speed (MHz)	operating ambient temperature (°C)	package
PCB5010WP-8	8.2	0 to +70	68-pin PLCC
PCF5010WP-8*	8.2	−40 to +85	68-pin PLCC
PCB5011YC-8	8.2	0 to +70	144 PGA
PCF5011YC-8*	8.2	−40 to +85	144 PGA

* The PCF versions will be identical to the PCB versions except that they have an extended operating ambient temperature range. However, minor variations may occur in the AC/DC characteristics.

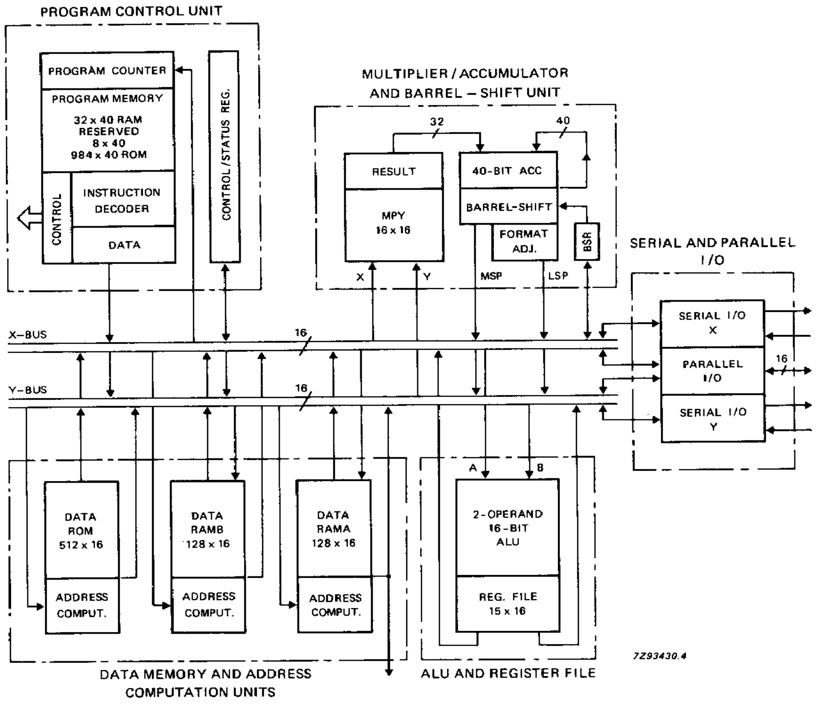


Fig. 1.0-1 Simplified block diagram of the PCB5010.

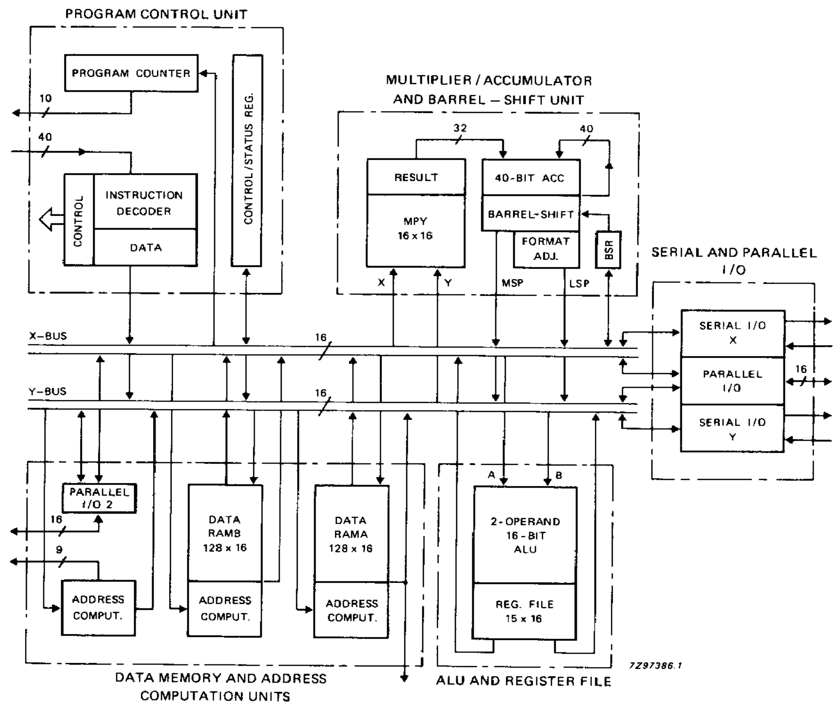


Fig. 1.0-2 Simplified block diagram of the PCB5011.

FEATURES

- Harvard architecture with two data buses of 16 bit width
- 4 instruction types:
  - multiply/accumulate operation + 2 data moves + 3 address calculations/memory read accesses
  - alu operation + 2 data moves + 3 address calculations/memory read accesses
  - load immediate data + 3 address calculations/memory read accesses
  - branch + 3 address calculations/memory read accesses

  Note: a high degree of parallel processing allows up to 6 basic operations to be performed simultaneously.
- Hardware two's complement 16 x 16 multiplier with 40-bit accumulator, full range barrel-shifter and format adjuster:
  - 45 different multiply/accumulate operations
  - multi-precision multiplication support
  - result bit-reversal possibility
  - 4 status flags
- 16-bit 2-operand ALU with:
  - 31 different operations
  - multi-precision operation support
  - 15 x 16-bit 3-port register-file
  - 5 status flags
- Program memory:
  PCB5010: 984 x 40-bit on-chip ROM (mask programmable)
  32 x 40-bit on-chip RAM (loaded via the X-bus)
  PCB5011: no on-chip program memory, but may be connected to a 1024 x 40-bit external memory
  (64K x 40-bit, by adding external logic circuitry)
- Data memory:
  PCB5010: 512 x 16 bit on-chip ROM (mask programmable)
  2 x (128 x 16) on-chip static RAM
  PCB5011: 512 x 16 bit external memory (read and write possible)
  2 x (128 x 16) on-chip static RAM
- 3 powerful programmable address computation units (ACU's) for the data ROM and both data RAMs and also for 16 pages of 4096 x 16 bit external data memory
  - each ACU has 8 different operations
  - 1 status flag for each ACU
- 5 level deep hardware stack (software extendable)
- 16-bit parallel I/O to access external data memory
  - 8.2 million words/s
  - WAIT facility so that "slow" peripherals can be connected
- 2 independent serial inputs and outputs (one pair for each data bus), with a maximum speed of 4.1 million bit/s under the control of external clocks
- 4 user input flags
- Maskable interrupt
- Possibility to repeat a single instruction
- Maximum clock rate 8.2 MHz
- Pipelined (P) and Non-pipelined (NP) modes under programmer control:
  P-mode: a new instruction can start every clock cycle
  NP-mode: a new instruction can start every two clock cycles
- Single 5 V power supply (±5%)
- All I/O are TTL compatible
- Operating ambient temperature range:
  PCB5010/11: 0 to +70 °C
  PCF5010/11: −40 to +85 °C

2.0 FUNCTIONAL DESCRIPTION

2.1 GENERAL DESCRIPTION

The detailed block diagram of the PCB5010 and PCB5011 are shown in Fig. 2.1-1 and Fig. 2.1-2. The signals of the processors are described briefly in Table 2.1-1.

Table 2.1-1 Signal description

SIGNAL	I/O	DESCRIPTION
V_{DD}	I	Supply voltage: 5 V ± 5%
V_{SS}	—	Ground
CLK	I	Clock
$\overline{RST}$	I	Reset
D15 . . . D0	I/O	16-bit wide parallel I/O port
A15 . . . A0	O	16-bit wide address for 64K-words in external data memory
$R/\overline{W}$	O	Read/write signal for control of external memory
$\overline{DS}$	O	Data strobe
$\overline{WAIT}$	I	Wait signal for synchronization of parallel I/O
DIX	I	Serial data input for the X-bus
$\overline{SIXEN}$	I	Serial input enable for the X-bus
$\overline{SIXRQ}$	O	Serial input request for the X-bus
CIX	I	Serial input clock for the X-bus
DIY	I	Serial data input for the Y-bus
$\overline{SIYEN}$	I	Serial input enable for the Y-bus
$\overline{SIYRQ}$	O	Serial input request for the Y-bus
CIY	I	Serial input clock for the Y-bus
DOX	O	Serial data output for the X-bus
$\overline{SOXEN}$	I	Serial output enable for the X-bus
$\overline{SOXRQ}$	O	Serial output request for the X-bus
COX	I	Serial output clock for the X-bus
DOY	O	Serial data output for the Y-bus
$\overline{SOYEN}$	I	Serial output enable for the Y-bus
$\overline{SOYRQ}$	O	Serial output request for the Y-bus
COY	I	Serial output clock for the Y-bus
$\overline{INT}$	I	Maskable interrupt
$\overline{IACK}$	O	Interrupt acknowledge
SYNC	O	Synchronization signal; indicates where execution of a new instruction starts
IFA	I	User flag
IFB	I	User flag
IFC	I	User flag
IFD	I	User flag

(continued on next page)

Table 2.1-1 (continued)

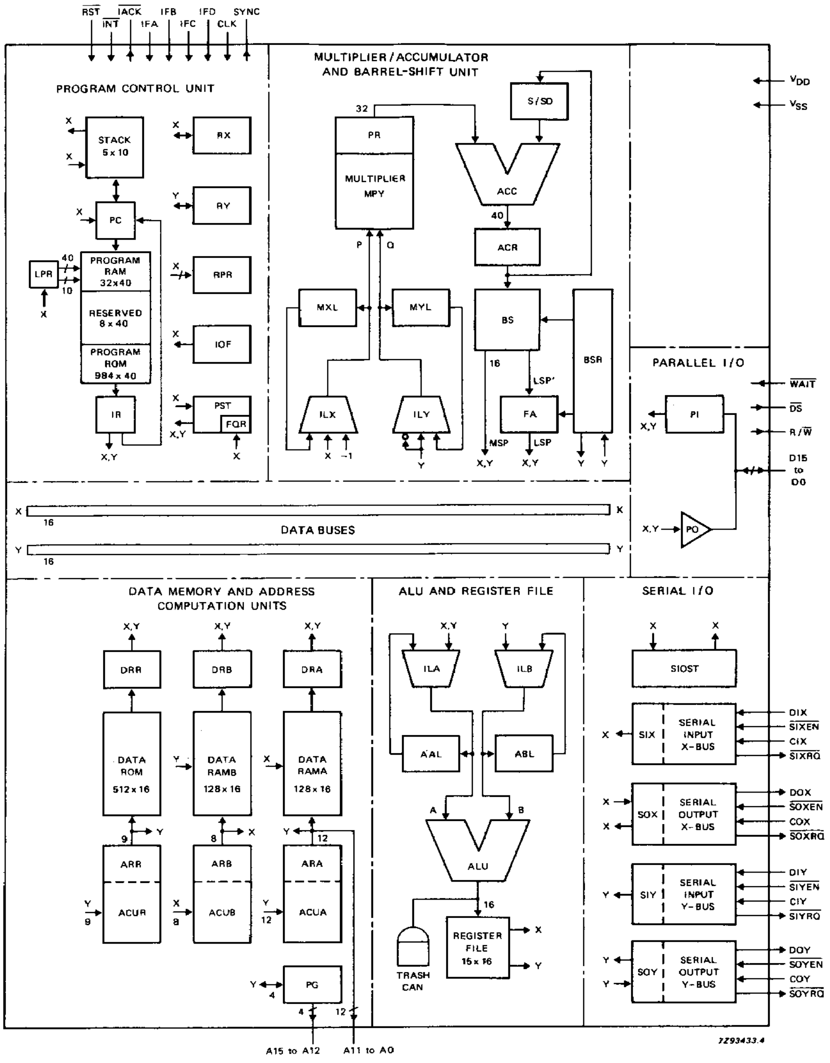
MNEMONIC	I/O	DESCRIPTION
PCB5011 only:		
PA9 . . . PA0	O	External program memory address
PD39 . . . PD0	I	External program word
ARR8 . . . ARR0	O	9-bit address for 512 words in external data memory
RD15 . . . RD0	I/O	Second 16-bit parallel I/O port
RR/$\overline{\text{W}}$	O	Read/write signal for second 16-bit parallel I/O port
$\overline{\text{RDS}}$	O	Data strobe for second 16-bit parallel I/O port

The main blocks of PCB5010/11 are:

- Program control unit with:
  - Program ROM (only for PCB5010)
  - IR (instruction register)
  - Sync pin
  - PC (program counter)
  - RPR (instruction repeat register)
  - Stack
  - PST (processor status register)
  - IOF (input/output status and user flag register)
  - User flag pins
  - $\overline{\text{INT}}$ pin and $\overline{\text{IACK}}$ pin
  - Bus-save registers RX and RY
  - $\overline{\text{RST}}$ pin
  - External program memory port (only PCB5011)
  - External program memory address port (only PCB5011)
- Data memory and address computation units with:
  - RAMA, ACUA (address computation unit A), DRA (data register A)
  - PG (page register)
  - RAMB, ACUB (address computation unit B), DRB (data register B)
  - ROM (only PCB5010), ACUR (address computation unit R), DRR (data register R)
  - External data word pins (only PCB5011)
  - External data memory address pins (only PCB5011)
- Multiplier/accumulator and barrel-shift unit with:
  - Input selectors ILX and ILY
  - Latches MXL and MYL
  - MPY (multiplier)
  - Accumulator with ACC (adder), ACR (multiplication/accumulation register) and S/SD (sign/scale-down block)
  - BS (barrel-shifter)
  - FA (format adjuster)
  - BSR (barrel-shift and format adjust control register)
- ALU and register file with:
  - Input selectors ILA and ILB
  - Latches AAL and ABL
  - ALU (arithmetic logic unit)
  - R1-R15 (register file)
  - Trash can
- Parallel I/O with:
  - PI (parallel data input latch)
  - PO (parallel data output buffer)
  - Parallel I/O data and control pins
- Serial I/O with:
  - SIX (serial input latch connected to X-bus)
  - SOX (serial output latch connected to X-bus)
  - SIY (serial input latch connected to Y-bus)
  - SOY (serial output latch connected to Y-bus)
  - SIOST (serial I/O control register)
  - Serial I/O data and control pins
- Data buses with:
  - 16 bits X-bus
  - 16 bits Y-bus

The functions of the main blocks are described in the following sections. The instruction set is described in the section 3.0.

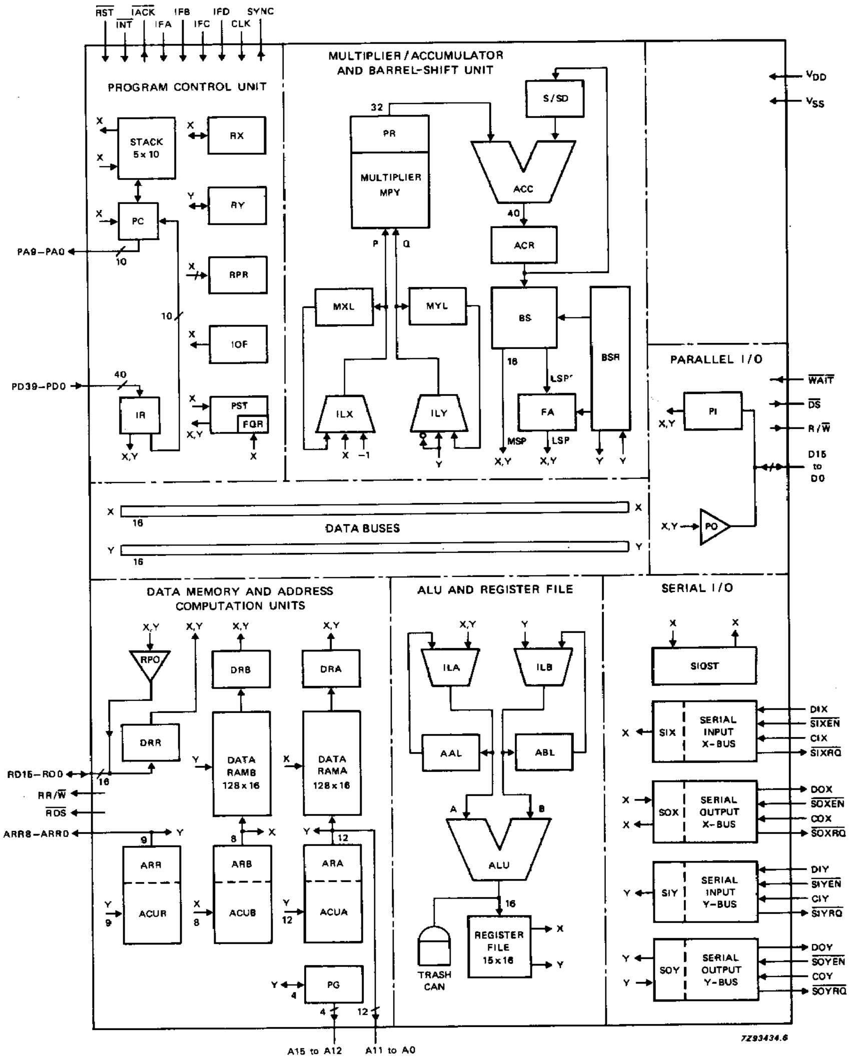
DEVELOPMENT DATA



Note 1: $\overset{X,Y}{\uparrow}$ means connected to the X-bus/Y-bus.

Note 2: all data paths are 16 bits wide unless indicated otherwise.

Fig. 2.1-1 Detailed block diagram of the PCB5010.



Note 1: $\overset{X,Y}{\uparrow}$ means connected to the X-bus/Y-bus.
Note 2: all data paths are 16 bits wide unless indicated otherwise.

Fig. 2.1-2 Detailed block diagram of the PCB5011.

2.2 PROGRAM CONTROL UNIT

2.2.1 Program memory

The PCB5011 has no on-chip program memory but may be connected to an external program memory. To access the external program memory, there are 40 program data pins (PD39-PD0) and 10 program address pins (PA9-PA0). The contents of the on-chip 10 -bit program counter are available via these address pins.

The PCB5010 has 1K x 40-bit on-chip program memory:

– 984 x 40-bit mask programmable ROM (address 0-983)
– 32 x 40-bit static RAM (address 992-1023)
– 8 x 40-bit reserved for test purposes (address 984-991)

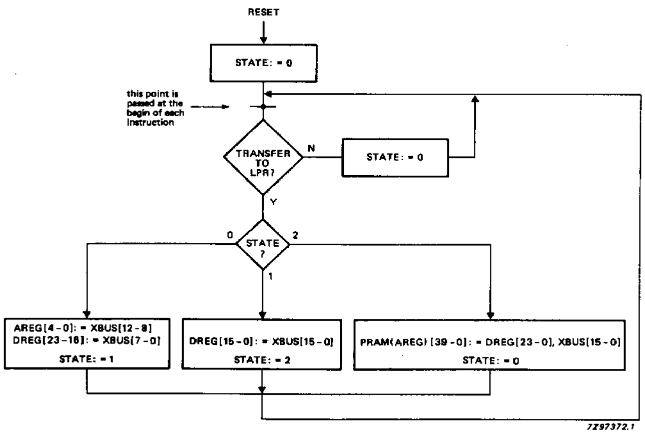
The memory is addressed by the 10-bit on-chip program counter. The static RAM of the program memory can be loaded via the X-bus by MOVE or LOAD IMMEDIATE operations, following the procedure described in the section 2.2.2 "Load program RAM circuitry".

2.2.2 Load program RAM circuitry (LPR)

LPR enables the programmer to load the 32 x 40-bit program RAM. To load each 40-bit instruction, three 16-bit words must be transferred to LPR via the X-bus, using MOVE or LOAD IMMEDIATE operations. LPR contains a 5-bit address register (AREG) in which the load address (range 0-31) is loaded and it contains a 24-bit data register (DREG) in which the instruction word is assembled. The loading procedure is shown in Fig. 2.2.2-1.

2.2.3 Program counter (PC) and mode circuitry (P and NP-mode)

The 10-bit program counter in the PCB5010 and PCB5011 is automatically incremented every instruction cycle during sequential program flow.



AREG = address register
DREG = data register
PRAM(i) = position i in the program RAM
[m – n] = bit m to bit n

Fig. 2.2.2-1 Program RAM loading procedure.

There are however, certain situations when the program counter is not incremented but updated differently:
- reset (see section 2.2.10)
- interrupt (see section 2.2.9)
- instruction repetition (see section 2.2.6)
- branch operations (see section 2.2.7)
- loading PC via the X-bus by means of a MOVE or LOAD IMMEDIATE operation (the 10 least significant bits of the X-bus data are loaded)

The processors work in two different modes:
1. Pipelined mode (P): instruction cycle = one clock cycle
2. Non-pipelined mode (NP): instruction cycle = two clock cycles

When the processor works in the P-mode, the result of a basic operation (see section 3.1 on instruction set) is not always available after the first instruction cycle but sometimes one clock cycle later. Since the processor uses pipelining, a new operation can start before the result of the previous operation is available. When the processor operates in NP-mode, the result of a basic operation is always available at the end of an instruction cycle. On reset, the processor is placed in the P-mode. Mode switching is possible under program control by setting and resetting the FQR bit in the PST register using a MOVE or LOAD IMMEDIATE operation. The instruction after the one that caused the change in the FQR bit is executed in the new mode. Return from an interrupt or a subroutine should be performed in the mode in which it was entered.

2.2.4 Instruction register (IR)

In every instruction cycle (i.e. every clock cycle when working in the P-mode and every two clock cycles when working in the NP-mode) an instruction word is fetched from the program memory. The program memory access and storing of the result in the instruction register (IR) takes one clock cycle. During the next clock cycle, the new contents of the IR is decoded and the processor controlled accordingly.

2.2.5 Stack

When an interrupt or subroutine call occurs, the value of PC (in the P-mode) or the value of PC+1 (in the NP-mode) is placed on the stack. The stack is a 5 x 10-bit LIFO register file that allows automatic nesting up to five levels of subroutines and/or interrupts. The top of the stack containing the most recent PC value can be accessed via the data buses. This enables the programmer to extend the stack in the data memory.

2.2.6 Instruction repeat circuitry (RPR)

RPR is a register that can be loaded via the X-bus with a number N by a MOVE or LOAD IMMEDIATE operation. The following instruction is then executed N times as long as $2 \leqslant N \leqslant 255$. The execution count is undefined when $N < 2$ or $N > 255$. The repetition of instructions by using the RPR is forbidden for the following operations:

- Load PC
- Load RPR
- Change of FQR bit in the PST
- Branch operations

2.2.7 Branch circuitry

The PCB5010 and PCB5011 make it possible to depart from the sequential program flow under software control. There are 4 branch types, and each branch can depend on any one of 50 different conditions.

The 4 branch types are:

- go to
- subroutine call
- return from subroutine
- return from interrupt.

The 50 branch conditions are the true and false status of the following flags or combination of flags:
- ALU flags: Z, N, C, $\overline{\text{C.OR.Z}}$, V, VL, {N.XOR.V}.OR.Z, $\overline{\text{N.XOR.V}}$
- Accumulator flags: SGNM, OVFL
- Barrel-shifter flags: OOR, OORL
- ALU and barrel-shifter flags: OORL.OR.VL
- ACU flags: ACA, ACB, ACR
- User flags: IFA, IFB, IFC, IFD, IFA.AND.IFB.AND.IFC.AND.IFD
- Serial I/O flags: SIXACK, SIYACK, SOXACK, SOYACK

2.2.8 The PST and IOF registers (PST, IOF, IFA – IFD pins)

PST and IOF are 16-bit registers that contain all the flags. Furthermore, PST also contains a bit (EI) that indicates whether the interrupt is enabled or not (enabled = 1; disabled = 0), a bit (FQR) indicating which mode (P = 0 or NP = 1) the processor is working in, and two bits (PIO1 and PIO2) determining the input criteria for the parallel input. The definition of each bit of PST and IOF registers is given in Table 2.2.8-1.

Table 2.2.8-1 PST and IOF registers; bit definition

bit	PST register		IOF register	
00	OVFL	(accumulator flag)	SIXACK	(serial I/O flag)
01	OORL	(barrel-shifter flag)	SOXACK	(serial I/O flag)
02	VL	(ALU flag)	SIYACK	(serial I/O flag)
03	V	(ALU flag)	SOYACK	(serial I/O flag)
04	C	(ALU flag)	IFA	(user flag)
05	Z	(ALU flag)	IFB	(user flag)
06	N	(ALU flag)	IFC	(user flag)
07	OOR	(barrel-shifter flag)	IFD	(user flag)
08	SGNM	(accumulator flag)	reserved	
09	ACA	(ACU flag)	reserved	
10	ACB	(ACU flag)	reserved	
11	ACR	(ACU flag)	reserved	
12	PIO2	(parallel I/O flag)	reserved	
13	PIO1	(parallel I/O flag)	reserved	
14	EI	(interrupt enable/disable)	reserved	
15	FQR	(operation mode)	reserved	

The flags in PST and IOF reflect the status of the functional units to which they belong and they are updated during each relevant instruction. The flags IFA, IFB, IFC and IFD reflect the signal level on their respective input pins. "Lock" type flags (OVFL, OORL and VL) can only be changed from 0 to 1 by the functional units to which they belong. The programmer can overrule the functional units updating the flags in the PST register: the PST register can be overwritten using a MOVE or LOAD IMMEDIATE operation. These operations are also used for loading the EI, FQR, PIO1 and PIO2 bits. Furthermore, the programmer can load the FQR bit using a MOVE or LOAD IMMEDIATE operation without changing the other bits in the PST register.

The PST and IOF register can be read using a MOVE operation. The flags can also be used as conditions in BRANCH operations (see section 2.2.7 on BRANCH circuitry). "Lock" type flags will automatically be reset to 0 when they are tested in a BRANCH operation.

2.2.9 Interrupt circuitry ($\overline{\text{INT}}$ and $\overline{\text{IACK}}$ pins, RX and RY)

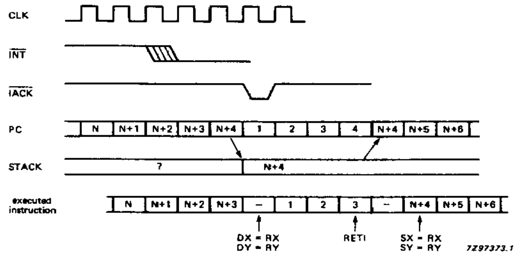
The processor has an interrupt facility that the user can access via the $\overline{\text{INT}}$ and $\overline{\text{IACK}}$ pins. When an interrupt is accepted by the processor the program counter is loaded with address 1. The interrupt procedure is described below and the logic timing diagrams are given in Figures 2.2.9-1 and 2.2.9-2.

An interrupt is initiated by a LOW on the $\overline{\text{INT}}$ pin. The first positive-going edge of CLK after a HIGH-to-LOW transition on $\overline{\text{INT}}$, the interrupt is clocked in by the processor. Two or three clock pulses later (see timing diagrams), the processor decides to accept the interrupt or postpone it. An interrupt is postponed:

- while the PC is being loaded (as part of a BRANCH, MOVE or LOAD IMMEDIATE operation)
- while the RPR register is loaded
- during instruction repetition
- when the stack is the source or destination to the X-bus
- when the interrupt is disabled by software (EI bit in PST register is 0).
- P-mode only: when RX or RY store data without having read it (MOVE or RETI BRANCH operations) after storing.

As soon as the above situations are completed, the postponed interrupt is accepted and thereafter handled in the same way as an interrupt that was accepted directly.

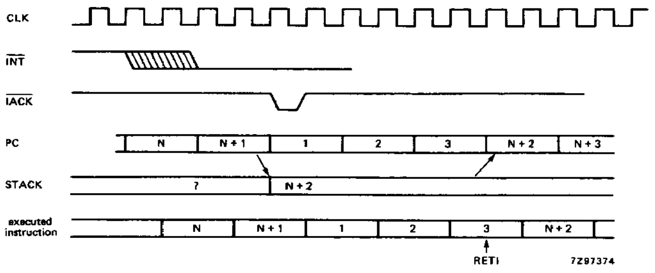
Accepting the interrupt means pushing the value of the PC (in P-mode) or PC+1 (NP-mode) on to the stack and loading the PC with address 1. When the PC contains address 1, the $\overline{\text{IACK}}$ pin goes LOW for one clock cycle to indicate to the outside world that the interrupt has been acknowledged.



Note 1: The interrupt is postponed when one of the following conditions is true during instruction N + 3:
- loading of PC (as part of BRANCH or MOVE operation)
- loading of RPR
- stack source or destination to X-bus
- repetition mechanism on
- interrupt disabled by EI = 0

Note 2: Instruction N + 4 is executed only after the interrupt routine; before the interrupt routine only the bus sources are stored in RX and RY

Fig. 2.2.9-1 Interrupt logic timing in P-mode.



Note: The interrupt is postponed when one of the following conditions is true during instruction N + 1:
- loading of PC (as part of BRANCH or MOVE operation)
- loading of RPR
- stack source or destination to X-bus
- repetition mechanism on
- interrupt disabled by EI = 0

Fig. 2.2.9-2 Interrupt logic timing in NP-mode.

In the P-mode, the instruction in the pipeline that should have been executed is not executed while the PC contains address 1 even though the expected X and Y-bus sources put their data on the buses. This data is stored in the bus-save register, RX and RY, that are automatically assigned as the destinations for the buses.

An interrupt routine is completed under program control using a RETI conditional BRANCH operation (see instruction set section 3.0). When the condition is true, PC is loaded with the address that was pushed onto the stack, and then the instruction at that address is executed. In the P-mode, however, RX and RY are used instead of the indicated X and Y-bus sources. Nested interrupts are permitted, but in the P-mode they are latched so long as the programmer has not stored the data in RX and RY elsewhere (i.e. so long as RX and RY have not been used as sources for the X and Y-buses). After returning from a nested interrupt in the P-mode, the old contents of RX and RY must be restored by the programmer.

To assure the correct functioning of the interrupt procedure, the following rules must be obeyed:

- an interrupt must be performed and exited in the same processor mode (P or NP)
- changing mode: the interrupt should be disabled before the FQR bit is altered

Note: Any READ/WRITE to the RX/RY bus-save registers, independent of the processor mode, will set/reset two internal interrupt disable flags. In P-mode the two internal interrupt disable flags are affected in the following ways:

- set by an interrupt
- individually set by writing to the RX/RY bus-save registers
- reset after an interrupt has been carried-out
- individually reset by reading from the RX/RY bus-save registers.

A new interrupt can only be generated after the $\overline{INT}$ signal has been HIGH for at least one positive going edge on CLK.

RX and RY can be used as general purpose registers when the interrupt is not used.

2.2.10 Reset circuitry (RST pin)

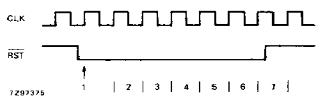
The processor is reset to the initial state when RST is LOW over at least 7 positive-going edges of CLK. A shorter reset may lead to an undefined situtation.

The initial state is characterized by:

– PC	: all zeros
– PST	: 0E00
– IOF	: SIXACK=SIYACK =0 and SOXACK = SOYACK = 1
– RX, RY	: all zeros
– STACK	: all zeros (5 x 10)
– RPR	: instruction repeat mechanism off
– LPR	: STATE = 0
– MXL, MYL	: all zeros
– RAMA(0)	: all zeros
– RAMB(0)	: all zeros
– PR	: all zeros
– ACR	: all zeros
– BSR	: all zeros
– ARA, AA, SA	: all zeros
– ARB, AB, SB	: all zeros
– ARR, AR, SR	: all zeros
– MA, MB, MR	: all zeros
– PG	: all zeros
– SIOST	: all zeros
– SOX, SOY	: all zeros

– $\overline{SIXRQ} = \overline{SIYRQ} = 0$
– $\overline{SOXRQ} = \overline{SOYRQ} = 1$

The logic timing of the reset is shown in Fig. 2.2.10-1.



1 – reset detection
2 – reset of all registers and latches mentioned above (excluding ACR), and "top of stack", serial I/O request signals, and PR
3 – reset of ACR and 2nd stack position
4 – reset of 3rd stack position
5 – reset of 4th stack position
6 – reset of 5th stack position
7 – reset of 6th stack position (note: this stack position must not be used, otherwise it may lead to an undefined state).

Fig. 2.2.10-1
The complete reset action.

2.2.11 Synchronization circuitry (SYNC pin)

The processor indicates where execution of a new instruction starts with a HIGH at the SYNC pin for half a clock cycle. This occurs once every two clock cycles in the NP-mode, and once every clock cycle in the P-mode. However, in the P-mode it will not occur when the program counter is loaded with a new address during the execution of a BRANCH operation, an interrupt or a reset.

2.3 DATA MEMORIES AND ACU'S

2.3.1 Data memories

The processor contains 3 on-chip data memories:

- RAMA: 128 x 16 bits, static
- RAMB: 128 x 16 bits, static
- ROM : 512 x 16 bits (only on PCB5010, external for PCB5011)

It is also possible to connect up to 64K of external data memory via the parallel I/O.

Memory outputs are connected to the data registers (DRA, DRB and DRR) and the parallel data input register PI. DRA, DRB and DRR are updated every instruction cycle, but in the P-mode, DRA and DRB are not updated when in that instruction cycle, data is moved (from one of the buses) into RAMA or RAMB. Updating PI is described in the section 2.6 on the parallel I/O. Data written into the data registers and PI can be transferred via the X or Y-bus during a subsequent instruction.

RAMA can be written-to via the X-bus, RAMB can be written-to via the Y-bus and, external RAM can be written-to via either bus.

With the PCB5011, it is not only possible to read from an external ROM, but in place of the read, it is also possible to write-to an external 512 x 16-bit memory because the port is bidirectional, and therefore can be used as a second parallel I/O port.

2.3.2 ACUs and PG register

The 3 address computation units, ACUA, ACUB and ACUR, function identically but, their address widths differ: 12, 8 and 9 bits respectively. ACUs calculate the addresses for the on-chip RAMs (only the 7 least significant address bits are used for this addressing) and data ROM. ACUA not only generates the address for RAMA but also the part of the address for external data memory that defines the position within a page of 4096 x 16 words. It is possible to have 16 pages of external data memory. Pages are selected using the address pins A12 to A15 that reflect the contents of the page register PG that is loaded using a MOVE or LOAD IMMEDIATE operation.

Fig. 2.3.2-1 illustrates the memory addressing.

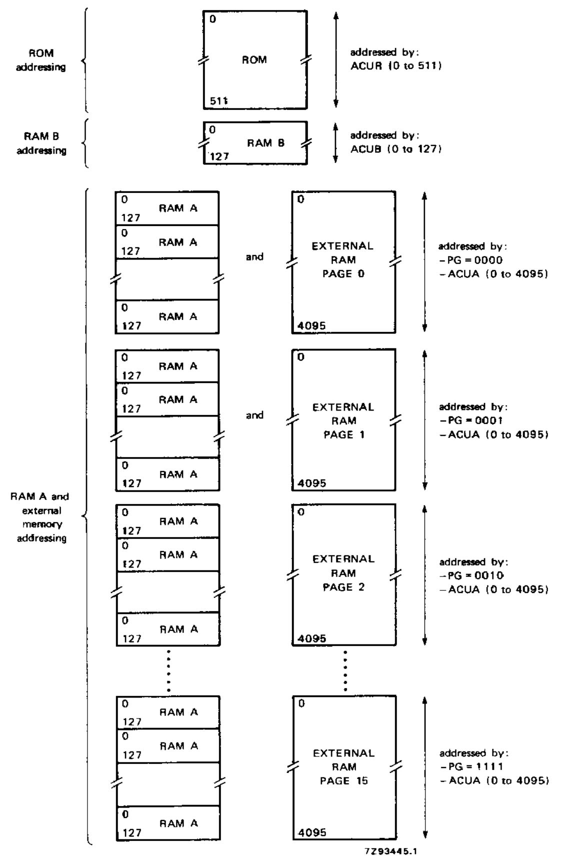


Fig. 2.3.2-1 Memory addressing.

Note to Fig. 2.3.2-1

RAMA and the external RAM are always simultaneously accessed by ACUA. A read-access results in loading of DRA and (under certain conditions determined by the PI01 and PI02 bits) simultaneous loading of PI. A next instruction specifies which of the two words (or both) are transferred to other places within the DSP via the data buses. A write-access results in an update of RAMA and/or the external RAM, depending on the specified destinations.

An ACU contains:
- an address register AR (which addresses the ROM respectively RAM's)
- a base address register A
- an offset register S
- an address masking register M
- a dedicated arithmetic unit

Only the AR registers (ARR, ARB and ARA) are shown in Figs 2.1-1 and 2.1-2. The A, S and M registers of RAMA, RAMB and the ROM (AA/BA/RA, AS/BS/RS and AM/BM/RM) are not shown.
The A, S, M and AR registers can be loaded directly (for initialization) via one of the data buses (X-bus for ACUB registers and Y-bus for ACUA and ACUR registers) using a MOVE or LOAD IMMEDIATE operation or they can be modified during address computation operations. Direct loading and address computation cannot take place simultaneously (see the section 3.0 on the instruction set).

Direct loading offers the following options:
- Load AR with value on the bus
- Load AR and A with value on the bus
- Load AR and S with value on the bus
- Load A with value on the bus
- Load S with value on the bus
- Load M with value on the bus
- Load AR, A and S with value on the bus
- Load AR with (value on the bus) !M

The notation !M means that, depending on the contents of the M register, all bits are not necessarily updated as the expression indicates:
- the bits whose corresponding bits in the M register are 1 are updated as specified in the expression
- the bits whose corresponding bits in the M register are 0 will retain their value (for AR) or will receive the previous value of their AR bit (for A and S).

The notation br(. . .) means that the bits are reversed in order.

Three flags in the PST register indicate the status of the 3 ACU's:
- ACA – ACUA flag is 1 when AR contained 0000 0000 0000 (binary) during the last clock cycle;
  ACUA flag is 0 when the AR was not 0000 0000 0000 (binary) during the last clock cycle
- ACB – ACUB flag is 1 when AR contained 0000 0000 (binary) during the last clock cycle;
  ACUB flag is 0 when the AR was not 0000 0000 (binary) during the last clock cycle
- ACR – ACUR flag is 1 when AR contained 0 0000 0000 (binary) during the last clock cycle;
  ACUR flag is 0 when the AR was not 0 0000 0000 (binary) during the last clock cycle.

Address computation offers the following options:

● AR:=AR	A:=A	S:=S	M:=M
● AR:=(A+1)!M	A:=(A+1)!M	S:=S	M:=M
● AR:=(A−1)!M	A:=(A−1)!M	S:=S	M:=M
● AR:=(A+S)!M	A:=(A+S)!M	S:=S	M:=M
● AR:=(S+1)!M	A:=A	S:=(S+1)!M	M:=M
● AR:=(A)!M	A:=A	S:=S	M:=M
● AR:=(S)!M	A:=A	S:=S	M:=M
● AR:=br(A+S)	A:=A+S	S:=S	M:=M

2.4 MULTIPLIER, ACCUMULATOR, BARREL-SHIFTER AND FORMAT-ADJUSTER UNIT

2.4.1 Multiplier

The multiplier performs a multiplication of two signed 16 bits operands P and Q presented in 2's complement notation. The result is presented by 32 bits in 2's complement notation and stored in the product result latch PR.

One of the following values can be chosen as P-operand:
- The value present on the X-bus
- The previous value which was automatically latched in the MXL latch
- The number −1

One of the following values can be chosen as Q-operand:
- The value present on the Y-bus
- The previous value which was automatically latched in the MYL latch
- The negated Y-bus value

  Note: When the Y-bus contains the highest negative value, -2^{15} (1000 0000 0000 0000 in binary) then the operand will be the highest positive value plus one, $+2^{15}$ (0111 1111 1111 1111 + 1 in binary). This number is stored in MYL which has a width of 17 bits for this particular situation.

The contents of MXL, MYL and PR are not changed when no MULTIPLY operation or a multiply HOLD operation is executed.

2.4.2 Accumulator

The accumulator unit consists of a 40 bit adder ACC, a 40 bit multiplication/accumulation register ACR and a sign and scale down block S/SD. The adder adds the results of the multiplication stored in PR to a second operand (provided by the S/SD block) which can be chosen from the following set:

- + or – the contents of ACR
- + or – the contents of ACR divided by 2^{15} (which allows multiprecision multiplication and addition)
- the number 0

The result of the addition is stored in ACR and is fed simultaneously to the barrel-shifter. The contents of the ACR register are not changed when no MULTIPLY operation is performed or a multiply HOLD operation is executed.

The 40-bit width of the accumulator allows the programmer to accumulate a number of multiplier results (at least 511) without the risk of overflow. Two flags in the PST register indicate the status of the accumulator:

- OVFL – overflow lock flag; this flag is set when overflow occurs in the adder (result outside the range -2^{39} to $+2^{39}-1$). For reset conditions see description of PST register (section 2.2.8).
- SGNM – sign flag; indicates the sign of the result of the addition (is identical to bit ACR(39)).

2.4.3 Barrel-shifter

From the 40-bit ACR contents, the barrel-shifter extracts 32 contiguous bits. The programmer determines which group of 32 bits is extracted by a value placed in the BSR register (bits BSR3 to BSR0). Sixteen different sets are possible; see Table 2.4.3-1.

Table 2.4.3-1 16 possible sets of 32 contiguous bits

BSR contents BSR3 - BSR0	32-bit word, E31 - E0, extracted by the barrel-shifter
0000	ACR30, ... ,ACR0,0
0001	ACR31, ... ,ACR0
0010	ACR32, ... ,ACR1
0011	ACR33, ... ,ACR2
0100	ACR34, ... ,ACR3
0101	ACR35, ... ,ACR4
0110	ACR36, ... ,ACR5
0111	ACR37, ... ,ACR6
1000	ACR38, ... ,ACR7
1001	ACR39, ... ,ACR8
1010	ACR39,ACR39, ... ,ACR9
1011	ACR39,ACR39,ACR39, ... ,ACR10
1100	ACR39,ACR39,ACR39,ACR39, ... ,ACR11
1101	ACR39,ACR39,ACR39,ACR39,ACR39, ... ,ACR12
1110	ACR39,ACR39,ACR39,ACR39,ACR39,ACR39, ... ,ACR13
1111	ACR39,ACR39,ACR39,ACR39,ACR39,ACR39,ACR39,. . . . ,ACR14

Two flags in the PST register indicate the status of the barrel-shifter:

- OOR – Out of range flag. It is set when the sign bit of the extracted word E31-E0 has no significance. This occurs when one or more bits of ACR to the left of the extracted word differs from E31.
- OORL – Out of range lock flag. The conditions for setting are identical to those of OOR. The conditions for resetting are given in the section describing the PST register.

2.4.4 Format adjuster

The output E31-E0 of the barrel-shifter is split into a 16-bit most significant part (MSP) and a 16-bit least significant part (LSP). MSP can be connected directly to the X and/or Y bus. LSP passes through a format adjuster (FA) before it reaches the X or Y bus. The output of FA is called LSP. Under software control, three FA options can be selected by placing a value in the BSR register (bits BSR5 to BSR4). The options are shown in Table 2.4.4-1.

Table 2.4.4-1 The three FA output options (LSP)

BSR contents BSR5 - BSR4	output LSP of format adjuster
00	E15-E0 (no change)
01	E0-E15 (bits reversed in order, used to speed-up certain serial outputs)
10	0, E15-E1 (bits shifted right over 1 position, left adjusted with zero; used for multi-precision multiplications)
11	reserved/undefined

2.4.5 Barrel-shifter register (BSR)

BSR is a 6-bit register. Its contents control the barrel-shifter (bit 0-3) and the format adjuster (bit 4-5) as explained in the previous sections. BSR can be loaded by means of a MOVE or LOAD IMMEDIATE operation. Loading has to be done at least one instruction before LSP or MSP is read to the X or Y-bus.

2.5 ALU AND REGISTER FILE

2.5.1 ALU

The PCB5010/11 has an ALU totally independent from the multiplier/accumulator unit. It is a 16-bit, 2-operand unit capable of executing 31 distinct operations. There are arithmetic, logic and some special purpose operations. The arithmetic operations defined as "extended" (mnemonic starts with X) are included to facilitate multi-precision operations. Extended operands are represented by 16-bit multiples.

An ALU operation produces a result R that may be stored in the register file or may be ignored (dumped in the trash can).

Several flags in the PST register give the status of the ALU. The flags are:

Z – Zero flag
N – Negative flag
C – Carry flag
V – Overflow flag
VL – Overflow lock flag (same as V but locked; see PST register description, section 2.2.8).

One of the following values can be chosen as A-operand:
- the value on the X-bus
- the value on the Y-bus
- the previous value which was automatically latched in the AAL-latch (this is not the case with the "byte swap" instruction).

One of the following values can be chosen as B-operand:
- the value on the Y-bus
- the previous value which was automatically latched in the ABL-latch

Dyadic operations require an A and B-operand, while monadic operations require only an A-operand. Some operations do not require an operand at all.

The ALU operations and their result R and flag settings are summed up in the following three tables:
A: Arithmetic operations
B: Logic operations
C: Other operations

The following notation is used:
- ZERO(R) – 0 (when not all 16 bits of R are 0)
  – 1 (when all 16 bits of R are 0)
- CARRY(F) – 0 (when a function F does not lead to a carry)
  – 1 (when a function F leads to a carry)
  Note: For this calculation, the operands are unsigned 16-bit numbers from 0 to 65535. The carry is 1 when the result of the addition is greater than 0. In all other cases the carry is 0.
- BORROW(F) – 0 (when a function F does not lead to a borrow)
  – 1 (when a function F leads to a borrow)
  Note: For this calculation, the operands are unsigned 16-bit numbers from 0 to 65535. The borrow is 1 when the result of the subtraction is below 0. In all other cases the borrow is 0.
- OVERFLOW(F) – 0 (when a function F does not lead to an overflow)
  – 1 (when a function F leads to an overflow)
  Note: For this calculation, the operands are signed 16-bit numbers between -2^{15} and $2^{15}-1$. The overflow is 1 when the result of the calculation is outside this range. In all other cases the overflow is 0.
- R(i) – Bit i of the 16-bit word R.

A: Arithmetic operations (Table 2.5.1-1)

For the calculation of result R, the operands A and B are considered to be binary numbers in 2's complement notation (between -2^{15} and $+2^{15}-1$). R is also a binary number in 2's complement notation. However, with the DIV operation, the operands and the result are unsigned numbers (between 0 and 65535).

B: Logic operations (Table 2.5.1-2)

For the calculation of result R, the operands A and B are considered to be 16-bit binary words. R is also a 16-bit binary word.

C: Other operations (Table 2.5.1-3)

DEVELOPMENT DATA

Table 2.5.1-1 Arithmetic operations

function	menmonic	result (R)	flags				condition
			Z	N	C	V	
addition	ADD	A+B	ZERO(R)	R(15)	CARRY(A+B)	OVERFLOW(A+B)	no overflow
		A+B–2^{16}	ZERO(R)	R(15)	CARRY(A+B)	OVERFLOW(A+B)	positive overflow
		A+B+2^{16}	ZERO(R)	R(15)	CARRY(A+B)	OVERFLOW(A+B)	negative overflow
extended addition	XADD	A+B+C	ZERO(R).AND.Z	R(15)	CARRY(A+B+C)	OVERFLOW(A+B+C)	no overflow
		A+B+C–2(16)	ZERO(R).AND.Z	R(15)	CARRY(A+B+C)	OVERFLOW(A+B+C)	positive overflow
		A+B+C+2(16)	ZERO(R).AND.Z	R(15)	CARRY(A+B+C)	OVERFLOW(A+B+C)	negative overflow
subtraction	SUB	A–B	ZERO(R)	R(15)	BORROW(A–B)	OVERFLOW(A–B)	no overflow
		A–B–2^{16}	ZERO(R)	R(15)	BORROW(A–B)	OVERFLOW(A–B)	positive overflow
		A–B+2^{16}	ZERO(R)	R(15)	BORROW(A–B)	OVERFLOW(A–B)	negative overflow
extended subtraction	XSUB	A–B–C	ZERO(R).AND.Z	R(15)	BORROW(A–B–C)	OVERFLOW(A–B–C)	no overflow
		A–B–C–2^{16}	ZERO(R).AND.Z	R(15)	BORROW(A–B–C)	OVERFLOW(A–B–C)	positive overflow
		A–B–C+2^{16}	ZERO(R).AND.Z	R(15)	BORROW(A–B–C)	OVERFLOW(A–B–C)	negative overflow
conditional subtraction	CSUB	A	ZERO(R)	R(15)	0	0	N=0
		A–B	ZERO(R)	R(15)	BORROW(A–B)	OVERFLOW(A–B)	no overflow } N=1
		A–B–2^{16}	ZERO(R)	R(15)	BORROW(A–B)	OVERFLOW(A–B)	positive overflow } N=1
		A–B+2^{16}	ZERO(R)	R(15)	BORROW(A–B)	OVERFLOW(A–B)	negative overflow } N=1
negate	NEG	0–A	ZERO(R)	R(15)	BORROW(0–A)	OVERFLOW(0–A)	$A \neq -2^{15}$
		A	ZERO(R)	R(15)	BORROW(0–A)	OVERFLOW(0–A)	$A = -2^{15}$
extended negate	XNEG	0–A–C	ZERO(R).AND.Z	R(15)	BORROW(0–A–C)	OVERFLOW(0–A–C)	$A \neq -2^{15}$ .OR.C = 1
		A	ZERO(R).AND.Z	R(15)	BORROW(0–A–C)	OVERFLOW(0–A–C)	$A = -2^{15}$ .AND.C = 0
conditional negate	CNEG	A	ZERO(R)	R(15)	0	0	N=0
		0–A	ZERO(R)	R(15)	BORROW(0–A)	OVERFLOW(0–A)	$A \neq -2^{15}$ .AND.N = 1
		A	ZERO(R)	R(15)	BORROW(0–A)	OVERFLOW(0–A)	$A = -2^{15}$ .AND.N = 1
decrement	DEC	A–1	ZERO(R)	R(15)	BORROW(A–1)	OVERFLOW(A–1)	$A \neq -2^{15}$
		$2^{15}-1$	ZERO(R)	R(15)	BORROW(A–1)	OVERFLOW(A–1)	$A = -2^{15}$
extended decrement	XDEC	A–C	ZERO(R).AND.Z	R(15)	BORROW(A–C)	OVERFLOW(A–C)	$A \neq -2^{15}$ .OR.C = 0
		$2^{15}-1$	ZERO(R).AND.Z	R(15)	BORROW(A–C)	OVERFLOW(A–C)	$A = -2^{15}$ .AND.C = 1
increment	INC	A+1	ZERO(R)	R(15)	CARRY(A+1)	OVERFLOW(A+1)	$A \neq 2^{15}-1$
		-2^{15}	ZERO(R)	R(15)	CARRY(A+1)	OVERFLOW(A+1)	$A = 2^{15}-1$
extended increment	XINC	A+C	ZERO(R).AND.Z	R(15)	CARRY(A+C)	OVERFLOW(A+C)	$A \neq 2^{15}-1$.OR.C = 0
		-2^{15}	ZERO(R).AND.Z	R(15)	CARRY(A+C)	OVERFLOW(A+C)	$A = 2^{15}-1$.AND.C = 1
arithmetic shift left	ASL	2*A	ZERO(R)	R(15)	CARRY(A+A)	OVERFLOW(A+A)	$-2^{14} < A < 2^{14}$
		2*A–2^{16}	ZERO(R)	R(15)	CARRY(A+A)	OVERFLOW(A+A)	$A \geqslant 2^{14}$
		2*A+2^{16}	ZERO(R)	R(15)	CARRY(A+A)	OVERFLOW(A+A)	$A < -2^{14}$
extended arithmetic shift left	XASL	2*A+C	ZERO(R).AND.Z	R(15)	CARRY(A+A)	OVERFLOW(A+A)	$-2^{14} < A < 2^{14}$
		2*A+C–2^{16}	ZERO(R).AND.Z	R(15)	CARRY(A+A)	OVERFLOW(A+A)	$A \geqslant 2^{14}$
		2*A+C+2^{16}	ZERO(R).AND.Z	R(15)	CARRY(A+A)	OVERFLOW(A+A)	$A < -2^{14}$
arithmetic shift right	ASR	A/2-fraction	ZERO(R)	R(15)	1(0)	0	
extended arithmetic shift right	XASR	A/2-fraction	ZERO(R).AND.Z	N	A(0)	0	MSB of A = C
		A/2-fraction–2^{15}	ZERO(R).AND.Z	N	A(0)	0	MSB of A = 0.AND.C = 1
		A/2-fraction+2^{15}	ZERO(R).AND.Z	N	A(0)	0	MSB of A = 1.AND.C = 0

Table 2.5.1-1 (continued)

function	mnemonic	result (R)	flags				condition
			Z	N	C	V	
add MSB of B to A	ADDM	A+B(15) A+B(15)-2^{16}	ZERO(R) ZERO(R)	R(15) R(15)	CARRY(A+B(15)) CARRY(A+B(15))	OVERFLOW(A+B(15)) OVERFLOW(A+B(15))	A+B(15) $< 2^{15}$ A+B(15) $= 2^{15}$
unsigned division	DIV	2*(A–B) 2*A 2*(A–B) 2*A	ZERO(R) ZERO(R) ZERO(R) ZERO(R)	0 0 1 1	1 1 0 0	0 0 0 0	$0 \leqslant A-B \leqslant 2^{15}-1$ $2^{15} \leqslant A-B \leqslant 2^{16}-1$ $-2^{16} \leqslant A-B \leqslant -2^{15}-1$ $-2^{15} \leqslant A-B \leqslant -1$
sign extension	XSGN	NN . . . . N	Z.AND.NOT.N	N	N	V	

Table 2.5.1-2 Logic operations

function	mnemonic	result (R)	flags				condition
			Z	N	C	V	
complement	COM	.NOT.A(i)		R(15)	0	0	for i=0 . . . 15
logic AND	AND	A(i).AND.B(i)	ZERO(R)	R(15)	0	0	for i=0 . . . 15
logic OR	OR	A(i).OR.B(i)	ZERO(R)	R(15)	0	0	for i=0 . . . 15
exclusive OR	EXOR	A(i).EXOR.B(i)	ZERO(R)	R(15)	0	0	for i=0 . . . 15

byte swap	SWAP	A(i+8)
		A(i–8)
			ZERO(R)
			ZERO(R)
				R(15)
				R(15)
					0
					0
						0
						0
							for i=0 . . . 7
							for i=8 . . . 15

logic shift left	LSL	A(i–1)
		0
			ZERO(R)
			ZERO(R)
				A(14)
				A(14)
					A(15)
					A(15)
						0
						0
							for i=1 . . . 15
							for i=0

logic rotate left	LROL	A(i–1)
		C
			ZERO(R)
			ZERO(R)
				A(14)
				A(14)
					A(15)
					A(15)
						0
						0
							for i=1 . . . 15
							for i=0

logic shift right	LSR	A(i+1)
		0
			ZERO(R)
			ZERO(R)
				0
				0
					A(0)
					A(0)
						0
						0
							for i=0 . . . 14
							for i=15

logic rotate right	LROR	A(i+1)
		C
			ZERO(R)
			ZERO(R)
				0
				0
					A(0)
					A(0)
						0
						0
							for i=0 . . . 14
							for i=15

Table 2.5.1-3 Other operations

function	mnemonic	result (R)	flags				condition
			Z	N	C	V	
pass and flag update	PASS	A	ZERO(A)	A(15)	0	0	
generate 0	NULL	0	1	0	0	0	
no operation	–	–	Z	N	C	V	

2.5.2 Register file

The output of the ALU is connected to the register file. This register file contains fifteen 16-bit registers. The programmer can choose to which register the ALU result is written and also has the option of discarding the result by writing it to the trash can.

The contents of any register may be read to either or both of the buses. Moreover, the register file is implemented as a 3-port memory, so that in the same instruction cycle three registers can be accessed: two accesses to read the present contents of register(s) and one to write in a new value.

The register file can be filled directly from the buses by a MOVE or LOAD IMMEDIATE operation. This does not affect the ALU flags but the AAL-latch will be updated. ALU operations cannot be specified simultaneously with the aforementioned MOVE operation.

2.6 PARALLEL I/O

Via the D15 – D0 pins, the PCB5010/11 permits parallel communication between the X or Y-bus and the outside world. An input or output can take place during each instruction cycle. Output occurs when data is transferred via the X or Y-bus with destination PO. The output is direct, there is no latching. Inputs are loaded into the parallel input latch PI, whose contents can be transferred via the X or Y-bus during a subsequent instruction. The programmer can select one of four input criteria by writing specified values into PST bits, PIO1 and PIO2; see Table 2.6-1.

The following control signals are associated with the parallel I/O (see timing in section 4):
- R/$\overline{W}$: indicates a read or write action (output)
- $\overline{DS}$: data strobe (output)
- $\overline{WAIT}$: signals for synchronization of the parallel I/O (input). This signal delays the internal clock, so that "slow" peripheral devices may be connected. Note: The $\overline{WAIT}$ signal can also be used with the second parallel I/O port in the PCB5011 (see section 2.3.1).

A block diagram of the parallel I/O circuitry is shown in Fig. 2.6-1.

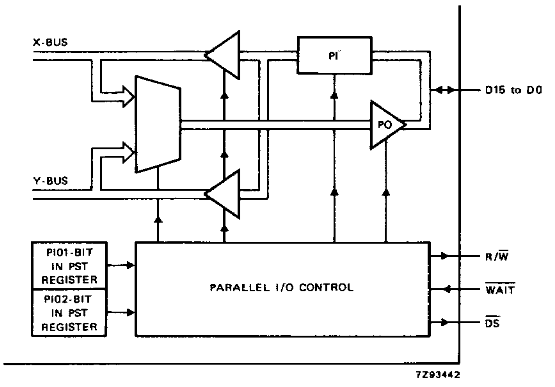


Fig. 2.6-1 Parallel I/O circuitry.

Table 2.6-1 Four selectable input criteria

PIO1	PIO2	DESCRIPTION
0	0	data is latched into PI every instruction cycle
0	1	data is latched into PI only when: – ACUA is not executing a "no operation" – PI latch is source to X or Y-bus
1	0	data is latched into PI only when: – PI latch is source to X or Y-bus
1	1	no data is latched into PI

Note to Table 2.6-1

A read is only performed when there is no write (MOVE or LOAD IMMEDIATE operation to PO).

2.7 SERIAL I/O

2.7.1 Serial data inputs/outputs

The PCB5010/11 has 2 independent serial data inputs DIX and DIY and 2 independent serial data outputs DOX and DOY (destinations and sources for X and Y-buses respectively). The maximum data transfer rate is 4.1 million bits/s. Data transfer is controlled by external clock signals CIX, COX, CIY and COY.

The following handshake signals, described in more detail in the following sections, are associated with the serial I/O:

- input/output enable $\overline{SIXEN}$, $\overline{SOXEN}$, $\overline{SIYEN}$ and $\overline{SOYEN}$ (input signals)
- input/output request $\overline{SIXRQ}$, $\overline{SOXRQ}$, $\overline{SIYRQ}$ and $\overline{SOYRQ}$ (output signals)

The following flags, whose functions are also described in the following sections, are associated with the serial I/O:

- serial I/O acknowledge flags SIXACK, SOXACK, SIYACK and SOYACK.

The programmer may control the length of the words to be transferred (between 1 and 16 bits) by writing the applicable values in the SIOST register fields, see Table 2.7-1.

Table 2.7-1 Word-length control

bit		word	word description
MSB	LSB		
03 to 00		SOLX	length of the serial output word, X-bus source
07 to 04		SILX	length of the serial output word, X-bus destination
11 to 08		SILY	length of the serial input word, Y-bus source
15 to 12		SOLY	length of the serial input word, Y-bus destination

Notes to Table 2.7-1

1. The binary number in each field specifies the length, the code 0000 indicates a length of 16 bits.
2. To avoid data transfers with incorrect word-length, the contents of the SIOST register should not be altered while the serial input/outputs are enabled.

2.7.2 Serial data input procedure

The X and Y serial data inputs are identical, therefore, only the serial input-X is described in the following paragraphs. Figures 2.7.2-1 and 2.7.2-2 are used to simplify the explanation of the serial data input procedure.

The serial data input procedure is controlled by the serial input control (SIC) and the serial input synchronization interface (SISI). The SIC controls the handshake signals $\overline{SIXEN}$ and $\overline{SIXRQ}$, the operation of the shift register SIXS, the counter CNTIX and the SIX latch. The SISI indicates, according to the CPU read actions and SIX write actions, the actual SIX latch status to both the CPU and SIC. SISI synchronizes the CPU events to the SIC and vice versa. Table 2.7.2-1 describes the serial data input signals.

The serial data input procedure consists of:

- the serial data input control procedure (Fig. 2.7.2-2)
- the serial data input synchronization procedure (Fig. 2.7.3-1).

Table 2.7.2-1 Serial data input signals

signal	description
SIXACK	SIX latch status indicator to CPU: 0; SIX data may be invalid 1; SIX data valid
SIXACKS	SIX latch status indicator to SIC: 0; SIX empty 1; SIX full
READ–SIX	SIX latch read indication from CPU
WRITE–SIX	SIX latch write indication from SIC
$\overline{SIXEN}$	SIC enable signal from an external device: 0; enables SIC 1; disables SIC
$\overline{SIXRQ}$	SIC ready indicator to external device: 0; SIC ready to shift 1; SIC not ready to shift
DIX	serial data input from an external device to the SIXS register
CIX	serial input clock signal from an external device

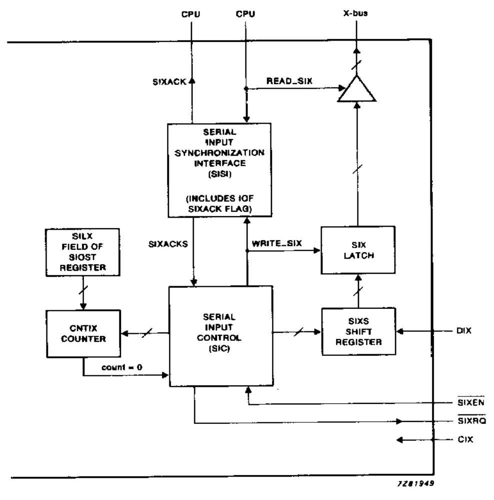

CPU
CPU
X-bus
SIXACK
READ_SIX
SERIAL INPUT SYNCHRONIZATION INTERFACE (SISI)
(INCLUDES IOF SIXACK FLAG)
SILX FIELD OF SIOST REGISTER
SIXACKS
WRITE_SIX
SIX LATCH
CNTIX COUNTER
SERIAL INPUT CONTROL (SIC)
SIXS SHIFT REGISTER
DIX
count = 0
SIXEN
SIXRQ
CIX
7Z81949

Fig. 2.7.2-1 Block diagram; serial data input.

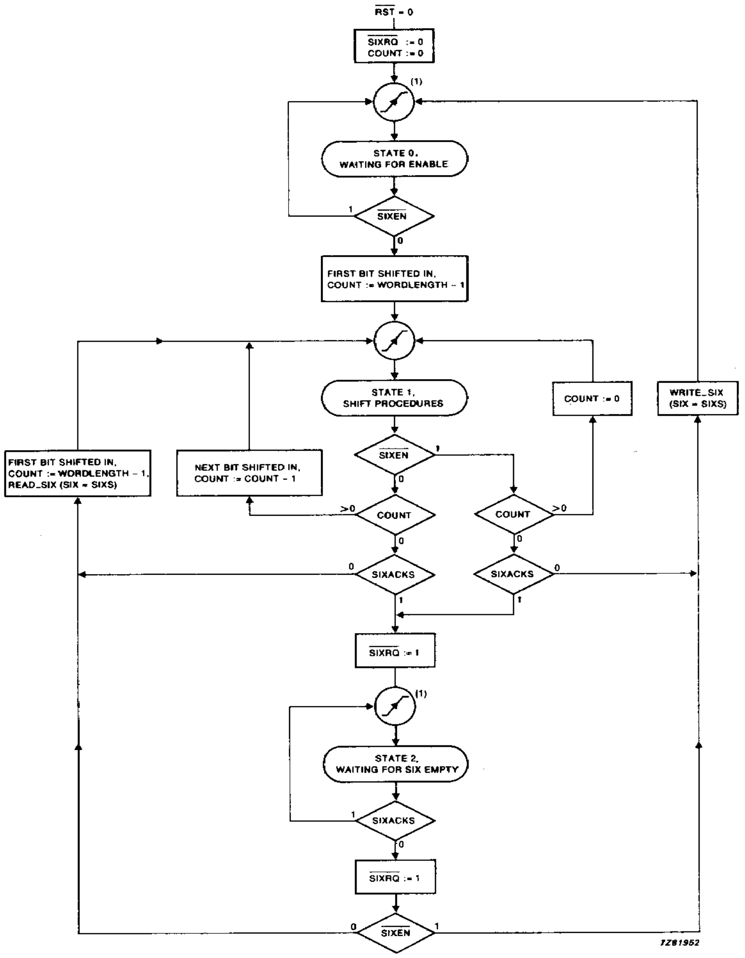


Fig. 2.7.2-2 Serial data input control procedure flow chart.

Note to Fig. 2.7.2-2

1. This point is passed at the next positive-going edge of CIX. At this point the assignments to SIX, SIXS and COUNT are executed. The assignments to $\overline{\text{SIXRQ}}$ are executed at the negative-going edge of SIX.

2.7.3 Serial data input control procedure

On RESET, SIC enters state 0 (awaiting enable), $\overline{\text{SIXRQ}}$ is reset to logic 0 (indicating that the serial input is ready to shift data) and the counter is reset to zero.

- state 0: SIC remains in state 0 until the serial input is enabled. Once enabled, the first bit (LSB) is shifted into SIXS, the counter is loaded with wordlength - 1 and SIC enters state 1 (shift procedure).
- state1: condition 1;
  - If the serial data input is enabled and the counter is not zero
  - then more bits are shifted-in, the counter decrements and SIC re-enters state 1.

  condition 2;
  - If the serial data input is enabled, the counter is zero and SIXACKS indicates an empty SIX latch
  - then the contents of SIXS is transmitted to SIX, a WRITE–SIX signal is transmitted to SISI, the counter is reloaded with word-length - 1, the first bit (LSB) of the next data word is shifted-in and SIC re-enters state 1.

  condition 3;
  - If the counter is zero and SIXACKS indicates that the SIX latch is full
  - then copy cannot performed, $\overline{\text{SIXRQ}}$ is set to logic 1 (indicating to an external device not to transmit new data) and SIC enters state 2.

  condition 4;
  - If the serial data input is disabled, the counter = 0 and SIXACKS indicates an empty SIX latch
  - then the contents of SIXS are transferred to SIX, a WRITE–SIX signal is transmitted to SISI and SIC enters state 0.

  condition 5;
  - If the serial data input is disabled and the counter is not zero
  - then shifting is aborted, the counter is reset to 0 and SIC enters state 1.

Note: Depending on the status of SIXACKS, condition 5 will result in aborted words being transferred from SIXS to SIX.

- state 2: When SIXACKS = logic 0 (indicating an empty SIX latch), then $\overline{\text{SIXRQ}}$ is set to logic 0, transmissions from SIXS to SIX are performed with or without starting (depending on the status of $\overline{\text{SIXEN}}$) the next shift transfer. SIC enters state 0 ($\overline{\text{SIXEN}}$ = logic 1) or state 1 ($\overline{\text{SIXEN}}$ = logic 0).

Note: If less than 16 bits have been received, the word is placed in the most significant part of the SIX latch and the remaining bits are set to logic 0.

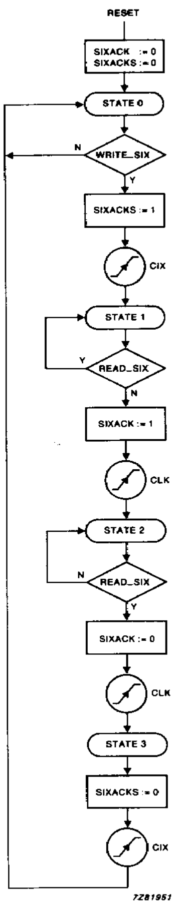


Fig. 2.7.3-1 Serial data input synchronization procedure flow chart.

2.7.4 Serial data input synchronization procedure

On RESET, SISI enters state 0. SIXACK and SIXACKS are reset to logic 0, indicating to the CPU and SIC that the SIX latch is empty.

- state 0: SISI awaiting a WRITE–SIX indication from SIC (indicates that a new data word has been transferred into the SIX latch). Then SIXACKS is set to logic 1 on the next positive-going edge of CIX, which indicates to the SIC that the SIX latch is full. SISI then enters state 1.

- state 1: If no READ–SIX occurs, SIXACK is set to logic 1 on the following positive-going edge of CLK and SISI enters state 2.
- state 2: If SIXACK = logic 1 (indicating to the CPU that valid data is available in the SIX latch), SISI awaits a read operation from the CPU. Then SIXACK is set to logic 0 on the next positive-going edge of CLK. SISI enters state 3.
- state 3: On the next positive-going edge of CIX, SIXACKS is reset to logic 0 and SISI re-enters state 0.

Note: If the CPU reads data from the SIX latch while SISI is not in state 2, invalid or old data maybe transferred onto the X-bus. This can be avoid by only allowing the CPU to perform read operations, from the SIX latch, when SIXACK = logic 1.

2.7.5 Serial data output

The X and Y serial data outputs are identical, therefore, only the serial output-X described in the following paragraphs. Figures 2.7.5-1 and 2.7.5-2 are used to simplify the explanation of the serial output procedure.

The serial output procedure is controlled by the serial output control (SOC) and the serial output synchronization interface (SOSI). The SOC controls the handshake signals $\overline{\text{SOXEN}}$ en $\overline{\text{SOXRQ}}$, the operation of the shift register SOXS and the counter CNTOX. The SOSI indicates, according to the CPU write actions and SOC read actions, the actual SOX latch status to both the CPU and SOC. SOSI synchronizes the CPU events to the SOC events and vice versa.

Table 2.7.5-1 describes the serial data output signals.

The serial data output procedure consists of:

– the serial data output control procedure (Fig. 2.7.5-2)
– the serial data output synchronization procedure (Fig. 2.7.6-1).

Table 2.7.5-1

signal	description
SOXACK	SOX latch status indicator to CPU: 0; SOX latch full 1; SOX latch empty
SOXACKS	SOX latch status indicator to SOC: 0; SOX data valid 1; SOX data may be invalid
WRITE–SOX	SOX latch write indication from CPU
READ–SOX	SOX latch read indication from SOC
$\overline{\text{SOXEN}}$	SOC enable signal from an external device: 0; enables SOC 1; disables SOC
$\overline{\text{SOXRQ}}$	SOC ready indicator to an external device: 0; SOC ready to shift 1; SOC not ready to shift
DOX	serial data output from an external device to the SOXS register
COX	serial output clock signal from an external device

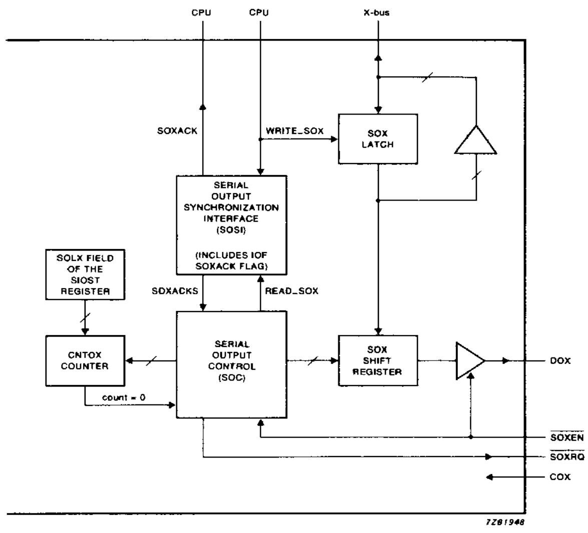

CPU
CPU
X-bus
SOXACK
WRITE_SOX
SOX LATCH
SERIAL OUTPUT SYNCHRONIZATION INTERFACE (SOSI) (INCLUDES IOF SOXACK FLAG)
SOLX FIELD OF THE SIOST REGISTER
SOXACKS
READ_SOX
CNTOX COUNTER
SERIAL OUTPUT CONTROL (SOC)
SOX SHIFT REGISTER
DOX
count = 0
SOXEN
SOXRQ
COX
7Z81948

Fig. 2.7.5-1 Block diagram; serial data output.

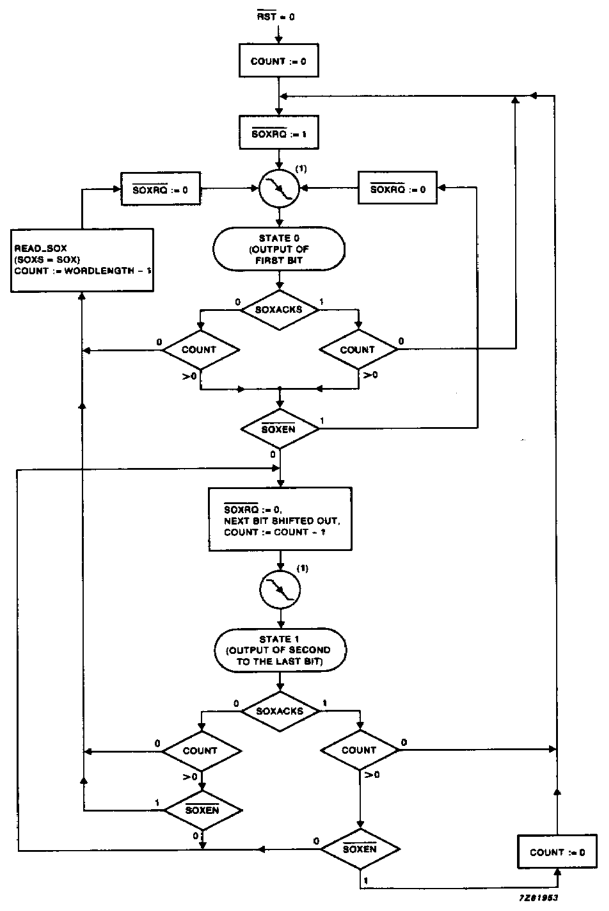


Fig. 2.7.5-2 Serial data output control procedure.

Note to Fig. 2.7.5-2

1. This point is passed at the next negative-going edge of COX. At this point the assignments to SOXS, COUNT and $\overline{SOXRQ}$ are executed.

2.7.6 Serial data output control procedure

On RESET, SOC enters state 0 (output first bit), $\overline{\text{SOXRQ}}$ is set to logic 1 (indicating that the serial output is not ready to shift out data) and the counter is reset to zero.

- state 0: condition 1;
  - If the SOX latch contains valid data (SOXACKS = logic 0) and the counter is zero
  - then the counter is reloaded with word-length - 1, SOX is copied into SOXS, the first bit (LSB) is available at the output if $\overline{\text{SOXEN}}$ = logic 0, a READ–SOX is transmitted to SOSI, $\overline{\text{SOXRQ}}$ is reset to logic 0 and SOC re-enters state 0.

  condition 2;
  - If the counter is not zero and the serial output is enabled
  - then the next bit is shifted out, the counter is decremented, $\overline{\text{SOXRQ}}$ remains at logic 0 and SOC enters state 1.

  condition 3;
  - If the counter is not zero and the serial output is disabled
  - then $\overline{\text{SOXRQ}}$ remains at logic 0 and SOC re-enters state 0.

  condition 4;
  - If the SOX latch does not contain valid data SOXACKS = logic 1) when the counter is zero
  - then $\overline{\text{SOXRQ}}$ is set to logic 1 and SOC re-enters state 0.
- state 1: condition 1;
  - If the SOX latch contains valid data (SOXACKS = logic 0) and the counter is zero or the serial data output is disabled
  - then the counter is reloaded with word-length - 1, SOX is copied into SOXS, the first data bit (LSB) is available at the output if $\overline{\text{SOXEN}}$ = logic 0, a READ–SOX is transmitted to SOSI, the SOXRQ remains at logic 0 and SOC enters state 0.

  condition 2;
  - If the counter is not zero and the serial data output is enabled
  - then the next bit is shifted out, the counter is decremented, $\overline{\text{SOXRQ}}$ remains at logic 0 and SOC remains in state 1.

  condition 3;
  - If the serial data output disabled while the counter is not zero and no new data is available from SOX
  - then the output procedure is aborted, the counter is reset to zero, $\overline{\text{SOXRQ}}$ is set to logic 1 and SOC enters state 0. When the serial output is re-enabled with new data available in SOX, then the serial output procedure will continue with the new data word and not with the aborted data word.

  condition 4;
  - If the SOX latch does not contain valid data (SOXACKS = logic 1) and the counter is zero
  - then $\overline{\text{SOXRQ}}$ is set to logic 1 and SOC re-enters state 0.

Note: If 1 bit words (counter loaded with zero) are transfer-red, the the counter is permanently zero and $\overline{\text{SOXEN}}$ is not examined by SOC. Only SOXACKS is checked to determine the status of the SOX latch. When a new word (bit) is available from SOX latch, then the word is loaded into SOXS (independent of the previous actions). The word is available from SOXS until the next word arrives but $\overline{\text{SOXRQ}}$ is only logic 0 for the first cycle. As soon as the word is transmitted from SOX to SOXS it should be read by the external device with $\overline{\text{SOXEN}}$ = 0 because during the next cycle $\overline{\text{SOXRQ}}$ will be set to logic 1 and this may disable the receiver.

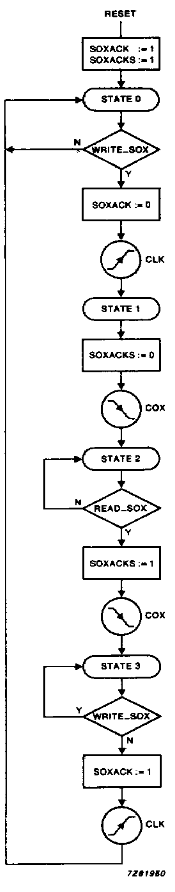


Fig. 2.7.6-1 Serial data output synchronization procedure flow chart.

2.7.7 Serial data output synchronization procedure

On RESET, SOSI enters state 0. SOXACK and SOXACKS are set to logic 1, indicating to the CPU and SOC that the SOX latch is empty.

- state 0: SOXACK = logic 1, indicating to the CPU that data may be written into SOX. SOSI waits for the CPU to perform a write operation. On the next positive-going edge of CLK, SOXACK is reset to logic 0 (indicating to the CPU that SOX is full) and SOSI enters state 1.
- state 1: On the next negative-going edge of COX, SOXACKS is reset to logic 0 (indicating to the SOC that SOX contains new data) and SOSI enters state 2.
- state 2: SOSI awaits a READ–SOX indication from SOC (indicating that a new word has been transmitted from SOX to SOXS). On the next negative-going edge of COX, SOXACKS is set to logic 1 (indicating that data has been transmitted) and SOSI enters state 3.
- state 3: If no WRITE–SOX operation occurs, then on the next positive-going edge of CLK, SOXACK is set to logic 1 and SOSI re-enters state 0.

Note: If the CPU writes data into the SOX latch while SOSI is not in state 0, data in the SOX latch maybe lost or corrupted data may be transferred into SOXS. This can be avoided by only allowing the CPU to perform write operations to the SOX latch, when SOXACK is logic 1.

APPENDIX A

PCB5010/PCB5011 INSTRUCTION SUMMARY

Type 0 instructions: ALU/MOVE/ADDRESS COMPUTATION

TY	ALU			AOPS			SX		SY		DX		DY		RFILE		ACUA		ACUR		ACUB	
	mnemonic	m/d	hex code	mnemonic	m/d	hex code	mnemonic	hex code	mnemonic	hex code	mnemonic	hex code	mnemonic	hex code	mnemonic	hex code	mnemonic	hex code	mnemonic	hex code	mnemonic	hex code
	ADD	d	14	SX,SY	d	0	–	00	–	00	–	0	–	0	–	0	**DY=A**		**DY=E**		**DX=E**	
0	XADD	d	15	AAL,SY	d	1	ROM	01	RAMB	01	FQR	1	BSR	1	R1	1	AR	0	AR	0	AR	0
	SUB	d	16	SX,ABL	d	2	–	02	–	02	SIOST	2	–	2	R2	2	AAR	1	AAR	1	AAR	1
	XSUB	d	17	AAL,ABL	d	3	TOS	03	RAMA	03	RPO	3	RAMB	3	R3	3	SAR	2	SAR	2	SAR	2
	CSUB	d	18				RX	04	RY	04	RX	4	–	4	R4	4	A	3	A	3	A	3
	NEG	m	0C	AAL	m	0	PI	05	ROM	05	PC	5	RY	5	R5	5	S	4	S	4	S	4
	XNEG	m	0D	(*/SWAP)			IOF	06	SIY	06	SOX	6	–	6	R6	6	M	5	M	5	M	5
	CNEG	m	0E	SX	m	1	SOX	07	–	07	(LPR)	7	–	7	R7	7	ASAR	6	ASAR	6	ASAR	6
	DEC	m	0A	SY	m	2	SIX	08	PG	08	–	8	–	8	R8	8	AR1M	7	AR1M	7	AR1M	7
	XDEC	m	0B	–	–	3	SIOST	09	BSR	09	RPR	9	PG	9	R9	9						
	INC	m	08				–	0A	ACU(ROM)	0A	PO	A	ACU(RAMA)	A	R10	A	**DY≠A**		**DY≠E**		**DX≠E**	
	XINC	m	09				RAMB	0B	ACU(RAMA)	0B	RAMA	B	PO	B	R11	B	–	0	–	0	–	0
	ASL	m	07				PST	0C	PST	0C	PST	C	RPO	C	R12	C	INCA	1	INCA	1	INCA	1
	XASL	m	06				RAMA	0D	PI	0D	TOS	D	SOY	D	R13	D	DECA	2	DECA	2	DECA	2
	ASR	m	03				ACU(RAMB)	0E	SOY	0E	ACU(RAMB)	E	ACU(ROM)	E	R14	E	STEP	3	STEP	3	STEP	3
	XASR	m	00				LSP	0F	LSP	0F	RFILE	F	RFILE	F	R15	F	INCS	4	INCS	4	INCS	4
	ADDM	d	1A				MSP	10	MSP	10							A	5	A	5	A	5
	DIV	d	19				R1	11	R1	11							S	6	S	6	S	6
	XSGN	–	1E				R2	12	R2	12							REV	7	REV	7	REV	7
	COM	m	10				R3	13	R3	13												
	AND	d	11				R4	14	R4	14												
	OR	d	12				R5	15	R5	15												
	EXOR	d	13				R6	16	R6	16												
	SWAP	m	1C				R7	17	R7	17												
	LSL	m	05				R8	18	R8	18												
	LROL	m	04				R9	19	R9	19												
	LSR	m	01				R10	1A	R10	1A												
	LROR	m	02				R11	1B	R11	1B												
	PASS	m	0F				R12	1C	R12	1C												
	NULL	–	1D				R13	1D	R13	1D												
	–	–	1B				R14	1E	R14	1E												
	–	–	1F				R15	1F	R15	1F												

* = exclusive

Type 1 instructions: MPY/MOVE/ADDRESS COMPUTATION

TY	MPY		MOPS		SX		SY		DX		DY		RFILE		ACUA		ACUR		ACUB	
	mnemonic	hex code	mnemonic	hex code	mnemonic	hex code	mnemonic	hex code	mnemonic	hex code	mnemonic	hex code	mnemonic	hex code	mnemonic	hex code	mnemonic	hex code	mnemonic	hex code
	–	0	SX,SY	0																
1	0	1	SX,–SY	1																
	+ACR	2	MXL,SY	2																
	–ACR	3	MXL,–SY	3																
	+ACRS	4	SX,MYL	4	as above		as above		as above		as above		as above		as above		as above		as above	
	–ACRS	5	MXL,MYL	5																
			–1,–SY	6																
			–1,SY	7																
			–1,MYL	8																

Type 2 instructions: BRANCH/ADDRESS COMPUTATION

TY	NAP	BR		CONDITION		CONDITION			ACUA		ACUR		ACUB	
		mnemonic	hex code	mnemonic	hex code	mnemonic	hex code		mnemonic	hex code	mnemonic	hex code	mnemonic	hex code
2	address of next instruction if condition matches	GOTO	0	–	00	–	01							
		CALL	1	AN	02	NOT AN	03							
		RET	2	X0	04	NOT X0	05							
		RETI	4	–	06	–	07		as above		as above		as above	
				GE	08	LT	09							
				LE	0A	GT	0B							
				IFX	0C	NOT IFX	0D							
				HI	0E	LS	0F							
				OFL	10	NOT OFL	11							
				XOL	12	NOT XOL	13							
				AOL	14	NOT AOL	15							
				VL	16	NOT VL	17							
				Z	18	NOT Z	19							
				N	1A	NOT N	1B							
				C	1C	NOT C	1D							
				V	1E	NOT V	1F							
				SIX	20	NOT SIX	21							
				SOX	22	NOT SOX	23							
				SIY	24	NOT SIY	25							
				SOY	26	NOT SOY	27							
				ACU(RAMA)	28	NOT ACU(RAMA)	29							
				ACU(RAMB)	2A	NOT ACU(RAMB)	2B							
				ACU(ROM)	2C	NOT ACU(ROM)	2D							
				–	2E	–	2F							
				IFA	30	NOT IFA	31							
				IFB	32	NOT IFB	33							
				IFC	34	NOT IFC	35							
				IFD	36	NOT IFD	37							

Type 3 instructions: LOAD IMMEDIATE/ADDRESS COMPUTATION

TY		DATA	DX		DY		RFILE		ACUA		ACUR		ACUB	
			mnemonic	hex code	mnemonic	hex code	mnemonic	hex code	mnemonic	hex code	mnemonic	hex code	mnemonic	hex code
3		16-bit data transferred to both X and Y buses	as above		as above		as above		as above		as above		as above	

PCB5010/PCB5011 INSTRUCTION SUMMARY

—This Page Intentionally Left Blank—

2.8 DATA BUSES X AND Y

The data buses have a width of 16-bits and are used for transferring data between the functional units connected to them. Bus transfers are always part of MOVE and LOAD IMMEDIATE operations and can be part of MPY and ALU operations. The possible sources and destinations of the bus are given in Table 2.8-1.

Table 2.8-1 Possible destinations and sources for the X and Y-bus

source on X	destination on X	source on Y	destination on Y
STACK	STACK		
RX	RX	RY	RY
	PC		
	RPR		
IOF			
IR(DATA field)		IR(DATA field)	
PST	PST	PST	
	FQR		
DRA DRB DRR		DRA DRB DRR	
	RAMA		RAMB
ARB		ARA ARR	
	ACUB		ACUA ACUR
		PG	PG
	ILX		ILY
LSP MSP		LSP MSP	
		BSR	BSR
RFILE	RFILE	RFILE	RFILE
	ILA		ILA ILB
PI	PO	PI	PO
	RPO		RPO
SIOST	SIOST		
SIX SOX	SOX	SIY SOY	SOY
	LPR		

Notes to Table 2.8-1

1. FQR is loaded by putting the required value (0 or 1) in bit 15 of the 16-bit word that is sent.
   The other bits are 'don't cares'.
2. When the source or destination has a width of less than 16 bits, only the least significant bits on the bus (same width) contain information. For a source, the MSBs are set to zero.
3. ILX and ILY select the operands for the multiplier; ILA and ILB select the operands for the ALU.
4. Destination RPO is only valid for PCB5011.
5. Specifying the same destination for the X and Y-bus results in no data transfer.
6. Destination LPR is only valid for PCB5010.

3.0 INSTRUCTION SET

The behaviour of the processor is controlled by the instructions stored in on-chip ROM (PCB5010) or external program memory (PCB5011). Each instruction leads to the execution of one or more (up to 6) basic operations. The basic operations, the instruction format and the instruction fields are described in the following sections.

3.1 BASIC OPERATIONS

The execution of a basic operation can take either 1 or 2 clock cycles. This means that:

- for 2 cycle basic operations pipelining is used in the P-mode
- a 1 cycle operation is extended by a second "no action" cycle when the processor is operating in the NP-mode.

Basic operation sequencing is shown in Fig. 3.1-1 for the P-mode, and Fig. 3.1-2 for the NP-mode. There are 6 different basic operations which are described below.

- **ALU operation** (1 clock cycle)

  One of 31 different ALU operations is executed. The operations need 2, 1 or 0 operands. The operation's result is stored in the register file or thrown away (into the trash can). In addition, a number of flags are updated.

- **MOVE operation** (1 clock cycle)

  Data from an X-bus (Y-bus) source is transferred via the X-bus (Y-bus) to an X-bus (Y-bus) destination. In the case that both buses have the same destination no transfer takes place.

- **ACU (address computation and memory access) operation** (2 clock cycles)

  Clock cycle 1:
  An address is calculated by the ACU and written into an address register (ARA, ARB, or ARR).

  Clock cycle 2:
  RAMA, RAMB, ROM or external memory is accessed at the location given by the address register (ARA, ARB, or ARR). This access is normally a read which updates one of the data registers (DRA, DRB, DRR, or PI). If in the P-mode, however, the relevant RAM (RAMA, RAMB or external RAM) is assigned as a destination during a MOVE executed in parallel, the access will be a write. The result in this case is that the RAM is updated.

- **MPY (multiply/accumulate) operation** (2 clock cycles)

  Clock cycle 1:
  Two operands are multiplied together and the result stored in the PR latch.

  Clock cycle 2:
  The content of PR is added to the output of S/SD and stored in the ACR.

  Note 1: The contents of the ACR are accessed via the barrel-shifter and format adjuster allowing transfer onto the X or Y-bus during the clock cycle following ACR loading. The buses are selected by specifying LSP and/or MSP as the source for the X or Y-bus.

  Note 2: The explanation of the MPY operation is a simplified presentation of what really happens. In fact, clock cycle 1 is not just used for multiplication, but part of the accumulation as well: i.e. accumulation of the lower part of the multiplier result and the contents of the accumulator. Therefore, in the P-mode, an MPY with +ACRS or −ACRS in the MPY field (see section 3.3 on description of MPY field) that directly follows another MPY operation leads to an undefined situation where the higher part of the first multiplication/accumulation result is already required at the end of clock cycle 1, but is only ready after clock cycle 2. If it is necessary to have an MPY operation with +ACRS or −ACRS directly after a previous MPY operation (for multiprecision multiplication), then there has to be a no operation cycle (or at least no MPY operation) between them.

- **BRANCH operation** (1 or 2 clock cycles)

  Clock cycle 1:
  The branch condition is checked, the result of which can be true or false. When the result is false, the BRANCH operation is terminated. When the result is true, the program counter is loaded with the branch address.

  Clock cylce 2: (only when condition is true)
  The new instruction is fetched to the instruction register IR.

  Note: During this fetch no new operation is started.

- **LOAD IMMEDIATE operation** (1 clock cycle)

  The 16-bit data word specified in the instruction is put on the X-bus and the Y-bus and transferred to one or two specified destinations. In the case that two identical destinations are specified no transfer takes place.

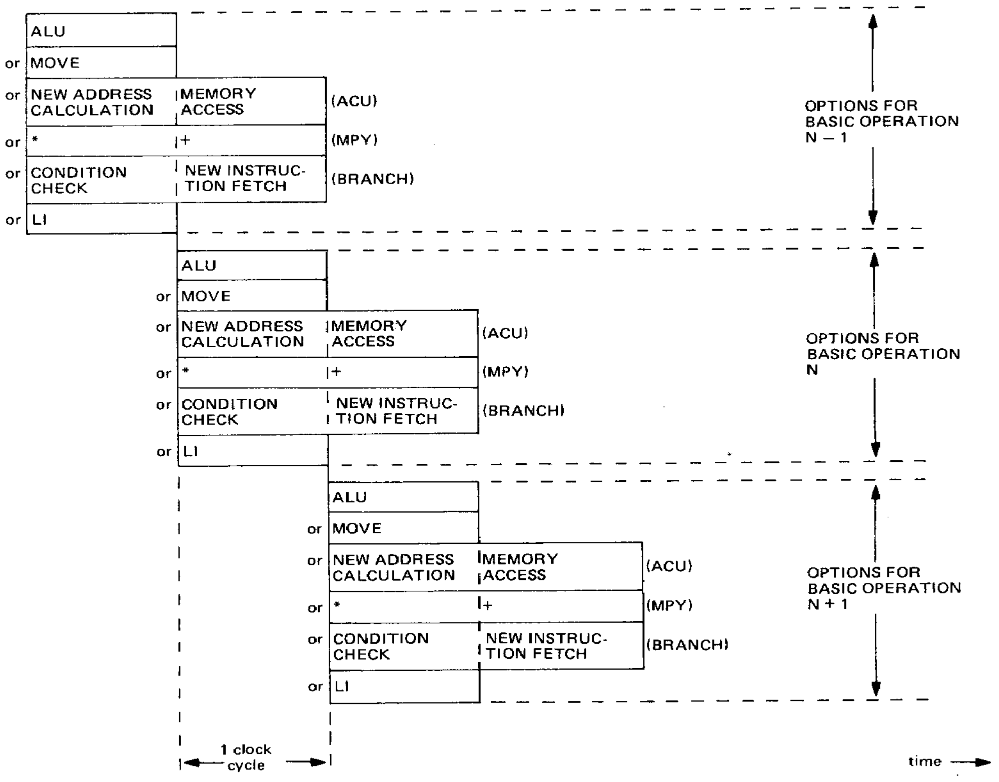

ALU
or MOVE
or NEW ADDRESS CALCULATION
MEMORY ACCESS
(ACU)
or *
+
(MPY)
or CONDITION CHECK
NEW INSTRUCTION FETCH
(BRANCH)
or LI
OPTIONS FOR BASIC OPERATION N − 1
OPTIONS FOR BASIC OPERATION N
OPTIONS FOR BASIC OPERATION N + 1
1 clock cycle
time

Fig. 3.1-1 Basic operation sequencing in P-mode.

Notes to Fig. 3.1-1

1. Each instruction may contain several simultaneous basic operations (see section 3.2 on instruction formats).
2. "New instruction fetch" (BRANCH) takes place only when the condition is true.
3. During "New instruction fetch" (BRANCH) no new operation is started.
4. Memory access is normally a data register (DRA, DRB, DRR, or PI) read. This read doesn't take place when there is a simultaneous MOVE to a RAM (during a RAM access).
5. An MPY operation with +ACRS or −ACRS in the MPY field directly following another MPY operation leads to an undefined situation (see description of MPY operation).
6. After instruction N which specifies a MOVE or LOAD IMMEDIATE operation to PC first instruction N + 1 is executed.

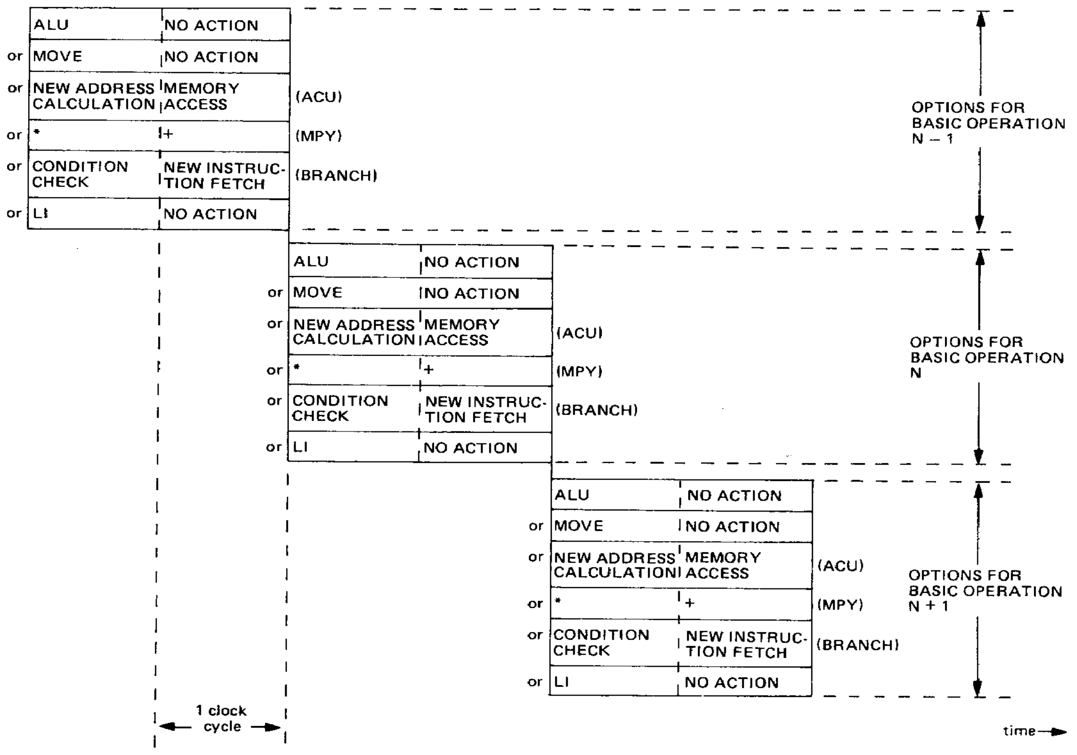


Fig. 3.1-2 Basic operation sequencing in NP-mode.

Notes to Fig. 3.1-2

1. Each instruction may contain several simultaneous basic operations (see section 3.2 on instruction format).
2. "New instruction fetch" (BRANCH) takes place only when the condition is true. If the condition is false, there is no action during the second clock cycle.
3. Memory access is a data register read.

3.2 INSTRUCTION FORMAT

All instruction words are 40-bits wide.

4 types of instructions are defined, each with their own set of basic operations:

Type 0: ALU operation + 2 MOVE operations + 3 ACU operations
Type 1: MULTIPLY operation + 2 MOVE operations + 3 ACU operations
Type 2: BRANCH operation + 3 ACU operations
Type 3: LOAD IMMEDIATE operation + 3 ACU operations.

Each basic operation is specified by the contents of one or more instruction fields (see Fig. 3.2-1).

The field at the far left of each instruction indicates the instruction type (see Table 3.2-1). The other fields are defined in Tables 3.3-1 to 3.3-13.

Table 3.2-1 Instruction types

ALU	=Type of ALU operation	dedicated for ALU operations
AOPS	=ALU operands	dedicated for ALU operations
SX	=Source on X-bus	These fields are for ALU and/or MOVE operations,
SY	=Source on Y-bus	or for MPY and/or MOVE operations
DX	=Destination on X-bus	for MOVE or LI operations
DY	=Destination on Y-bus	for MOVE or LI operations
RFILE	=Destination in Register file	for ALU, MOVE or LI operations
ACUA	=Type of ACUA operation	dedicated for ACU operations
ACUB	=Type of ACUB operation	dedicated for ACU operations
ACUR	=Type of ACUR operation	dedicated for ACU operations
MPY	=Type of accumulator operation	dedicated for MPY operations
MOPS	=Multiply operands	dedicated for MPY operations
NAP	=Address of next instruction when condition is true	dedicated for BRANCH operations
BR	=Type of branch operation	dedicated for BRANCH operations
COND	=Branch condition	dedicated for BRANCH operations
DATA	= 16 bits data word that is transmitted on X and Y-bus	dedicated for LI operations

3.3 INSTRUCTION FIELDS

For each instruction field, the valid codes and their function are specified. For most fields, not only the function is specified but also a mnemonic. The PCB5010/11 assembly language, using the mnemonics in this data sheet, is described in a separate document.

DEVELOPMENT DATA

Instruction type 0:

39	38	37 36 35 34 33	32 31	30 29 28 27 26	25 24 23 22 21	20 19 18 17	16 15 14 13	12 11 10 9	8 7 6	5 4 3	2 1 0
0	0	ALU	AOPS	SX	SY	DX	DY	RFILE	ACUA	ACUR	ACUB

Instruction type 1:

39	38	37 36 35	34 33 32 31	30 29 28 27 26	25 24 23 22 21	20 19 18 17	16 15 14 13	12 11 10 9	8 7 6	5 4 3	2 1 0
0	1	MPY	MOPS	SX	SY	DX	DY	RFILE	ACUA	ACUR	ACUB

Instruction type 2:

39	38	37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22	21 20 19	18 17 16 15 14 13	12 11 10 9	8 7 6	5 4 3	2 1 0
1	0	NAP*	BR	CONDITION	0 0 0 0	ACUA	ACUR	ACUB

Instruction type 3:

39	38	37	36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21	20 19 18 17	16 15 14 13	12 11 10 9	8 7 6	5 4 3	2 1 0
1	1	0	DATA	DX	DY	RFILE	ACUA	ACUR	ACUB

* Only bits 22 to 31 are used.

Fig. 3.2-1 The four instruction types and their fields.

Table 3.3-1 ALU instruction field

mnemonic	code binary	code hex	type	function
Arithmetic operations:				
ADD	10100	14	dyadic	addition
XADD	10101	15	dyadic	extended addition
SUB	10110	16	dyadic	subtraction
XSUB	10111	17	dyadic	extended subtraction
CSUB	11000	18	dyadic	conditional subtraction
NEG	01100	0C	monadic	negate
XNEG	01101	0D	monadic	extended negate
CNEG	01110	0E	monadic	conditional negate
DEC	01010	0A	monadic	decrement
XDEC	01011	0B	monadic	extended decrement
INC	01000	08	monadic	increment
XINC	01001	09	monadic	extended increment
ASL	00111	07	monadic	arithmetic shift left
XASL	00110	06	monadic	extended arithmetic shift left
ASR	00011	03	monadic	arithmetic shift right
XASR	00000	00	monadic	extended arithmetic shift right
ADDM	11010	1A	dyadic	add MSB of B to A
DIV	11001	19	dyadic	unsigned division step
XSGN	11110	1E	no operand	extended N flag
Logic operations:				
COM	10000	10	monadic	logic complement
AND	10001	11	dyadic	logic AND
OR	10010	12	dyadic	logic OR
EXOR	10011	13	dyadic	logic exclusive OR
SWAP	11100	1C	monadic	byte swap
LSL	00101	05	monadic	logic shift left
LROL	00100	04	monadic	logic rotate left
LSR	00001	01	monadic	logic shift right
LROR	00010	02	monadic	logic rotate right
Other operations:				
PASS	01111	0F	monadic	pass with flag update
NULL	11101	1D	no operand	generate 0
–	11111	1F	–	reserved
–	11011	1B	–	reserved

Table 3.3-2 AOPS instruction field (note 1)

mnemonic	code binary	code hex	ALU-type	A-operand	B-operand
*, *	00	0	dyadic	source on X-bus	source on Y-bus
AAL, *	01	1	dyadic	AAL	source on Y-bus
*, ABL	10	2	dyadic	source on X-bus	ABL
AAL, ABL	11	3	dyadic	AAL	ABL
AAL (note 2)	00	0	monadic	AAL	–
*	01	1	monadic	source on X-bus	–
*	10	2	monadic	source on Y-bus	–
–	11	3	–	reserved	–

Notes to Table 3.3-2

1. The AOPS field has no meaning when a no operand operation is specified in the ALU field and as there is no operation, AAL and ABL retain their values.
2. The operation is undefined when this operand is selected with a SWAP operation.

* The mnemonic of the particular source is used (see SX and SY fields).

Table 3.3-3 SX instruction field

mnemonic	code binary	code hex	full name of source
–	00000	00	no source
ROM	00001	01	data register DRR
–	00010	02	reserved
TOS	00011	03	top of stack (notes 1 and 2)
RX	00100	04	bussave register X
PI	00101	05	parallel data input register
IOF	00110	06	input/output status and user flag register
SOX	00111	07	serial output register connected to X-bus
SIX	01000	08	serial input register connected to X-bus
SIOST	01001	09	serial I/O control register
–	01010	0A	reserved
RAMB	01011	0B	data register DRB
PST	01100	0C	processor status register
RAMA	01101	0D	data register DRA
ACU(RAMB)	01110	0E	address register ARB
LSP	01111	0F	least significant 16 bits of multiply/shift/adjust result
MSP	10000	10	most significant 16 bits of multiply/shift/adjust result
R1	10001	11	register 1
R2	10010	12	register 2
R3	10011	13	register 3
R4	10100	14	register 4
R5	10101	15	register 5
R6	10110	16	register 6
R7	10111	17	register 7
R8	11000	18	register 8
R9	11001	19	register 9
R10	11010	1A	register 10
R11	11011	1B	register 11
R12	11100	1C	register 12
R13	11101	1D	register 13
R14	11110	1E	register 14
R15	11111	1F	register 15

Notes to Table 3.3-3

1. When TOS is used as a source, the stack is popped one level.
2. It is forbidden to simultaneously use TOS as a source and destination. The interrupt must be disabled before this operation is performed.

Table 3.3-4 SY instruction field

mnemonic	code binary	code hex	full name of source
–	00000	00	no source
RAMB	00001	01	data register DRB
–	00010	02	reserved
RAMA	00011	03	data register DRA
RY	00100	04	bussave register Y
ROM	00101	05	data register DRR
SIY	00110	06	serial input register connected to Y-bus
–	00111	07	reserved
PG	01000	08	page register
BSR	01001	09	barrel-shifter/format adjuster control register
ACU(ROM)	01010	0A	address register ARR
ACU(RAMA)	01011	0B	address register ARA
PST	01100	0C	processor status register
PI	01101	0D	parallel data input register
SOY	01110	0E	serial output register connected to Y-bus
LSP	01111	0F	least significant 16 bits of multiply/shift/adjust result
MSP	10000	10	most significant 16 bits of multiply/shift/adjust result
R1	10001	11	register 1
R2	10010	12	register 2
R3	10011	13	register 3
R4	10100	14	register 4
R5	10101	15	register 5
R6	10110	16	register 6
R7	10111	17	register 7
R8	11000	18	register 8
R9	11001	19	register 9
R10	11010	1A	register 10
R11	11011	1B	register 11
R12	11100	1C	register 12
R13	11101	1D	register 13
R14	11110	1E	register 14
R15	11111	1F	register 15

Table 3.3-5 DX instruction field

mnemonic	code binary	hex	full name of destination
–	0000	0	no destination
FQR	0001	1	FQR bit in PST; the required value for FQR must be present in bit 15 on the bus; the other bits are don't cares
SIOST	0010	2	serial I/O control register
RPO	0011	3	second parallel data output buffer (PCB5011 only)
RX	0100	4	bussave register X
PC	0101	5	program counter
SOX	0110	6	serial output register connected to X-bus
LPR	0111	7	Load program RAM circuitry (PCB5010 only)
–	1000	8	reserved
RPR	1001	9	instruction repeat register
PO	1010	A	parallel data output buffer
RAMA	1011	B	RAMA
PST	1100	C	processor status register
TOS	1101	D	top of stack (notes 1 and 2)
ACU(RAMB,*)	1110	E	ACUB
**	1111	F	register file

Notes to Table 3.3-5

1. When TOS is used as a destination, the stack is pushed one level.
2. It is forbidden to simultaneously use TOS as a source and destination.

Table 3.3-6 DY instruction field

mnemonic	code binary	hex	full name of destination
–	0000	0	no destination
BSR	0001	1	barrel-shifter/format adjuster control register
–	0010	2	reserved
RAMB	0011	3	RAMB
–	0100	4	reserved
RY	0101	5	bussave register Y
–	0110	6	reserved
–	0111	7	reserved
–	1000	8	reserved
PG	1001	9	page register
ACU(RAMA,*)	1010	A	ACUA
PO	1011	B	parallel data output buffer
RPO	1100	C	second parallel data output buffer (PCB5011 only)
SOY	1101	D	serial output register connected to Y-bus
ACU(ROM,*)	1110	E	ACUR
**	1111	F	register file

* fill in a mnemonic of the ACU-initialization field.
** fill in a mnemonic of the RFILE field.

Table 3.3-7 RFILE instruction field

mnemonic	code binary	code hex	full name of destination
–	0000	0	no destination
R1	0001	1	register 1
R2	0010	2	register 2
R3	0011	3	register 3
R4	0100	4	register 4
R5	0101	5	register 5
R6	0110	6	register 6
R7	0111	7	register 7
R8	1000	8	register 8
R9	1001	9	register 9
R10	1010	A	register 10
R11	1011	B	register 11
R12	1100	C	register 12
R13	1101	D	register 13
R14	1110	E	register 14
R15	1111	F	register 15

Table 3.3-8 ACU initialization (notes 1 and 2)

mnemonic	code		New values for:			
	binary	hex	AR	A	S	M
AR	000	0	source	A	S	M
AAR	001	1	source	source	S	M
SAR	010	2	source	A	source	M
A	011	3	AR	source	S	M
S	100	4	AR	A	source	M
M	101	5	AR	A	S	source
ASAR	110	6	source	source	source	M
AR1M	111	7	(source)!M	A	S	M

Table 3.3-9 ACU computation (notes 1 and 2)

mnemonic	code		New values for:			
	binary	hex	AR	A	S	M
	000	0	AR	A	S	M
INCA	001	1	(A+1)!M	(A+1)!M	S	M
DECA	010	2	(A–1)!M	(A–1)!M	S	M
STEP	011	3	(A+S)!M	(A+S)!M	S	M
INCS	100	4	(S+1)!M	A	(S+1)!M	M
A	101	5	(A)!M	A	S	M
S	110	6	(S)!M	A	S	M
REV	111	7	br(A+S)	A+S	S	M

Notes to Tables 3.3-8 and 3.3-9

1. The ACU fields for RAMA, RAMB and ROM are identical. The content of the field has a different meaning if it is for ACU initialization or address computation.
2. See section 2.3.2 on ACUs and PG register for explanation of !M and br(....).

Table 3.3-10 MPY instruction field

mnemonic	code binary	code hex	new ACR-value
–	000	0	ACR (HOLD)
0	001	1	P*Q
+ACR	010	2	P*Q + ACR
–ACR	011	3	P*Q – ACR
+ACRS	100	4	P*Q + ACR x 2^{-15}
–ACRS	101	5	P*Q – ACR x 2^{-15}
	other		reserved

Table 3.3-11 MOPS instruction field (note 1)

mnemonic	code binary	code hex	P-input of multiplier	Q-input of multiplier
*, *	0000	0	source on X-bus	source on Y-bus
, –	0001	1	source on X-bus	–(source on Y-bus)
MXL,*	0010	2	MXL	source on Y-bus
MXL, –*	0011	3	MXL	–(source on Y-bus)
*, MYL	0100	4	source on X-bus	MYL
MXL, MYL	0101	5	MXL	MYL
–1, –*	0110	6	–1	–(source on Y-bus)
–1, *	0111	7	–1	source on Y-bus
–1, MYL	1000	8	–1	MYL
	other		reserved	reserved

Note to Table 3.3-11

1. When the MPY-field contains 000, then independent of the MOPS-code the following is true.

mnemonic	code	P-input of multiplier	Q-input of multiplier
–	XXXX	MXL	MYL

* Fill in the mnemonic of the particular source (see SX and SY fields).

NAP

The NAP field contains the address of the next instruction to be executed if a branch condition is true.

Table 3.3-12 BR instruction field

mnemonic	code binary	code hex	function
GOTO	000	0	goto
CALL	001	1	subroutine call
RET	010	2	return from subroutine
RETI	100	4	return $\overline{\text{INT}}$ interrupt
	other		reserved

Table 3.3-13 COND instruction field (notes 1 and 2)

mnemonic	code binary	code hex(X=0)	condition	mnemonic explanation
–	00000X	00	always true	
AN	00001X	02	SGNM = 1	accumulator negative
XO	00010X	04	OOR = 1	
–	00011X	06	reserved	
GE or NOT LT	00100X	08	$\overline{\text{N.EXOR.V}}$ = 1	greater or equal to/not less than
NOT GT or LE	00101X	0A	(N.EXOR.V).OR.Z = 1	not greater than/less than or equal to
IFX	00110X	0C	IFA.AND.IFB.AND.IFC.AND.IFD = 1	
HI or NOT LS	00111X	0E	$\overline{\text{C.OR.Z}}$ = 1	higher/not less than or equal to
OFL	01000X	10	OORL.OR.VL = 1	system overflow
XOL	01001X	12	OORL = 1	extractor overflow
AOL	01010X	14	OVFL = 1	accumulator overflow
VL	01011X	16	VL = 1	
Z or EQ	01100X	18	Z = 1	
N	01101X	1A	N = 1	
C	01110X	1C	C = 1	
V	01111X	1E	V = 1	
SIX	10000X	20	SIXACK = 1	
SOX	10001X	22	SOXACK = 1	
SIY	10010X	24	SIYACK = 1	
SOY	10011X	26	SOYACK = 1	
ACU(RAMA)	10100X	28	ACA = 1	
ACU(RAMB)	10101X	2A	ACB = 1	
ACU(ROM)	10110X	2C	ACR = 1	
–	10111X	2E	reserved	
IFA	11000X	30	IFA = 1	
IFB	11001X	32	IFB = 1	
IFC	11010X	34	IFC = 1	
IFD	11011X	36	IFD = 1	
–	11100X	38	reserved	
–	11101X	3A	reserved	
–	11110X	3C	reserved	
–	11111X	3E	reserved	

X = 0: branch takes place when condition = true
X = 1: branch takes place when condition = false

Notes to Table 3.3-13

1. The mnemonic names GE, LT, GT, LE, HI, and LS refer to situations where the flag setting is a result of a subtraction (SUB) of operands A and B. For example, GE means that A is greater than or equal to B.
2. OOR and OORL are set to logic 1 one clock cycle after the ACR register has been filled with an out-of-range value. OOR is set to logic 0 one clock cycle after the ACR register has been filled with a value with the specified range.

DATA

Data is a 16-bit data word which is transmitted on the X and Y-bus.

4.0 ELECTRICAL SPECIFICATION

4.1 ABSOLUTE MAXIMUM RATINGS (NOTE 1)

parameter	conditions	symbol	min.	max.	unit
Supply voltage		V_{DD}	–0.5	+ 6.5	V
Voltage at any input		V_I	–0.5	V_{DD} + 0.5	V
Input/output current		$I_{I/O}$	–	5	mA
Total power dissipation	note 2				
	PLCC 68	P_{tot}	–	1	W
	PGA 144	P_{tot}	–	1	W
Operating ambient temperature range	PCB5010/11	T_{amb}	0	+ 70	°C
	PCF5010/11	T_{amb}	–40	+ 85	°C
Storage temperature range		T_{stg}	–55	+ 150	°C

Note to the Absolute Maximum Ratings

1. Stress above those listed under 'Absolute Maximum Ratings' may cause permanent damage to this device. This is a stress rating only and functional operating of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute rating conditions for extended periods may affect device reliability.
2. This value is based on the maximum allowable die temperature and thermal resistance of the packages.

DEVELOPMENT DATA

4.2 DC CHARACTERISTICS

V_{DD} = 5 V ± 5%; V_{SS} = 0 V; T_{amb} = 0 to + 70 °C; all voltages with respect to V_{SS}; unless otherwise specified

parameter	conditions	symbol	min.	typ.	max.	unit
Supply voltage		V_{DD}	4.75	–	5.25	V
Supply current	notes 1 and 2					
PCB5010		I_{DD}	–	100	130	mA
PCB5011		I_{DD}	–	100	120	mA
HIGH level input voltage	except CLK	V_{IH}	2.0	–	V_{DD} + 0.5	V
HIGH level input voltage	CLK only	V_{IH1}	2.4	–	V_{DD} + 0.5	V
LOW level input voltage		V_{IL}	–0.5	–	0.8	V
LOW level input current	V_I = 0.4 V	$-I_{IL}$	–	–	100	µA
HIGH level input current	V_I = 2.0 V	$-I_{IH}$	–	–	100	µA
HIGH level output voltage	$-I_{OH}$ = 100 µA	V_{OH}	2.4	–	–	V
LOW level output voltage	I_{OL} = 2.0 mA	V_{OL}	–	–	0.4	V

Notes to the DC characteristics

1. $I_{DD(typ.)}$ is based on V_{DD} = 5 V, T_{amb} = 22 °C and under operating conditions (excluding the reset condition).
2. $I_{DD(max.)}$ is based on V_{DD} = $V_{DD(max.)}$, $\overline{RST}$ = 0, f = 8.2 MHz
   and all other pins disconnected.

4.3 AC CHARACTERISTICS

V_{DD} = 5 V ± 5%; V_{SS} = 0 V , T_{amb} = 0 °C to 70 °C (PCB5010/5011); C_L 80 pF (capacitive load per output);
Timing measurements are taken at 2.0 V for logic 1 and 0.8 V for logic 0; unless otherwise specified.

no.	parameter	min.	max.	unit	note
0	CLK period	122	2000	ns	
1	CLK width HIGH	55	1000	ns	
2	CLK width LOW (time 2)	55	1000	ns	
3	CLK HIGH to SYNC HIGH	20	50	ns	
4	CLK HIGH to Ax valid	20	60	ns	
5	CLK HIGH to R/$\overline{\text{W}}$ LOW	20	85	ns	
6	CLK HIGH to R/$\overline{\text{W}}$ HIGH	20	85	ns	x = 0 . . . 15
7	CLK LOW to $\overline{\text{DS}}$ LOW	10	45	ns	
8	R/$\overline{\text{W}}$ LOW to $\overline{\text{DS}}$ LOW	0	—	ns	
9	CLK HIGH to $\overline{\text{DS}}$ HIGH	10	45	ns	
10	$\overline{\text{DS}}$ HIGH to R/$\overline{\text{W}}$ HIGH	0	—	ns	
11a	$\overline{\text{DS}}$ HIGH to Ax invalid	5	—	ns	x = 0 . . . 11
11b	$\overline{\text{DS}}$ HIGH to Ay invalid	0	—	ns	y = 12 . . . 15
12	$\overline{\text{DS}}$ width LOW	*	—	ns	
13	Dx access time	—	60	ns	
14	Dx hold time	0	—	ns	x = 0 . . . 15
15	Dx valid to $\overline{\text{DS}}$ HIGH	50	—	ns	x = 0 . . . 15
16	$\overline{\text{DS}}$ HIGH to Dx invalid	5	—	ns	x = 0 . . . 15
17	$\overline{\text{WAIT}}$ set-up time	15	—	ns	
18	$\overline{\text{WAIT}}$ hold time	30	—	ns	
19	$\overline{\text{INT}}$ set-up time	5	—	ns	
20	$\overline{\text{INT}}$ hold time	30	—	ns	
21	CLK HIGH to $\overline{\text{IACK}}$ LOW	—	60	ns	
22	CLK HIGH to $\overline{\text{IACK}}$ HIGH	—	60	ns	
23	IFA, IFB, IFC, IFD set-up time	5	—	ns	
24	IFA, IFB, IFC, IFD hold time	30	—	ns	
25	$\overline{\text{RST}}$ set-up time	10	—	ns	
26	$\overline{\text{RST}}$ hold time	30	—	ns	
27	CLK HIGH to PAx or ARRx valid	20	60	ns	x = 0 . . . 9 or 0 . 8
28	CLK HIGH to RR/$\overline{\text{W}}$ LOW	20	85	ns	
29	CLK HIGH to RR/$\overline{\text{W}}$ HIGH	20	85	ns	
30	CLK LOW to $\overline{\text{RDS}}$ LOW	10	45	ns	
31	RR/$\overline{\text{W}}$ LOW to $\overline{\text{RDS}}$ LOW	0	—	ns	
32	CLK HIGH to $\overline{\text{RDS}}$ HIGH	10	45	ns	
33	$\overline{\text{RDS}}$ HIGH to RR/$\overline{\text{W}}$ HIGH	0	—	ns	
34	$\overline{\text{RDS}}$ HIGH to ARRx invalid	5	—	ns	
35	$\overline{\text{RDS}}$ width LOW	*	—	ns	
36	RDx access time	—	60	ns	x = 0 . . . 15
37	RDx hold time	0	—	ns	x = 0 . . . 15
38	RDx valid to $\overline{\text{RDS}}$ HIGH	50	—	ns	x = 0 . . . 15
39	$\overline{\text{RDS}}$ HIGH to RDx invalid	5	—	ns	x = 0 . . . 15
40	COX, CIX, COY, CIY period	244	—	ns	
41	COX, CIY, COY, CIY width HIGH	110	—	ns	
42	COX, CIX, COY, CIY width LOW	110	—	ns	
43	COX(Y) LOW to $\overline{\text{SOX(Y)RQ}}$ LOW	—	40	ns	
44	COX(Y) LOW to $\overline{\text{SOX(Y)RQ}}$ HIGH	—	40	ns	
45	$\overline{\text{SOX(Y)EN}}$ set-up time	40	—	ns	
46	$\overline{\text{SOX(Y)EN}}$ hold time	10	—	ns	
47	COX(Y) LOW to next DOX(Y) valid	—	40	ns	
48	$\overline{\text{SOX(Y)EN}}$ LOW to DOX(Y) valid	—	40	ns	
49	$\overline{\text{SOX(Y)EN}}$ to DOX(Y) 3-state	—	40	ns	
53	CIX(Y) LOW to $\overline{\text{SIX(Y)RQ}}$ LOW	—	40	ns	
54	CIX(Y) LOW to $\overline{\text{SIX(Y)RQ}}$ HIGH	—	40	ns	
55	$\overline{\text{SIX(Y)EN}}$ set-up time	30	—	ns	
56	$\overline{\text{SIX(Y)EN}}$ hold time	10	—	ns	
57	DIX, DIY set-up time	10	—	ns	
58	DIX, DIY hold time	40	—	ns	
59	PDx access time	—	60	ns	x = 0 . . . 39
60	PDx hold time	20	—	ns	x = 0 . . . 39

* Value given is equal to (time 2) — 15 ns.

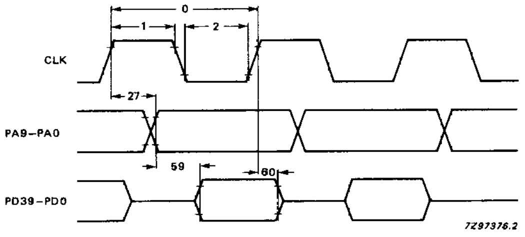


Fig. 4.3-1 Program memory access timing (for PCB5011 only).

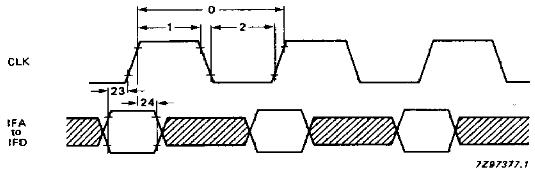


Fig. 4.3-2 User flag timing.

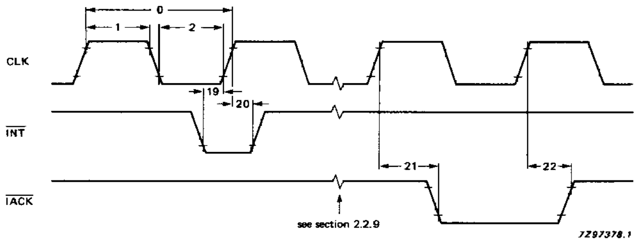


Fig. 4.3-3 Interrupt timing.

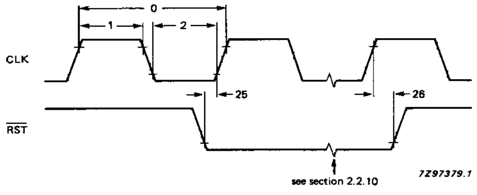


Fig. 4.3-4 Reset timing.

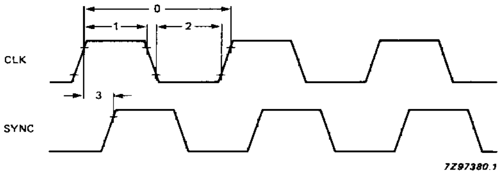
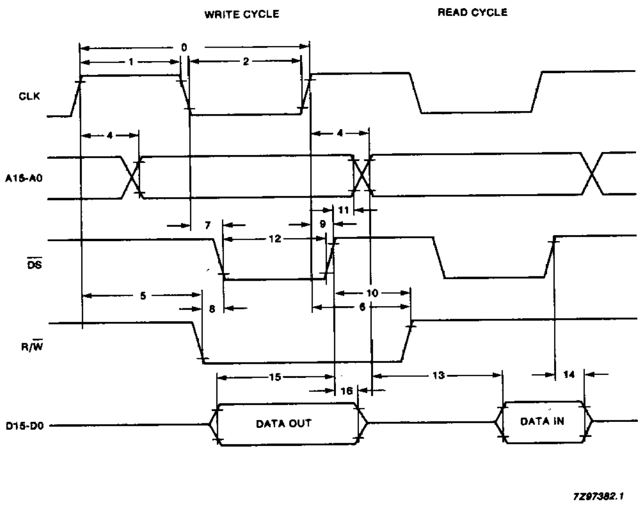


Fig. 4.3-5 Synchronization timing.



Note: (part of) the signals A15 – A0 can be used just as hardware controlling outputs if they are not needed for addressing external RAM.

Fig. 4.3-6 Parallel I/O timing; $\overline{\text{WAIT}}$ inactive.

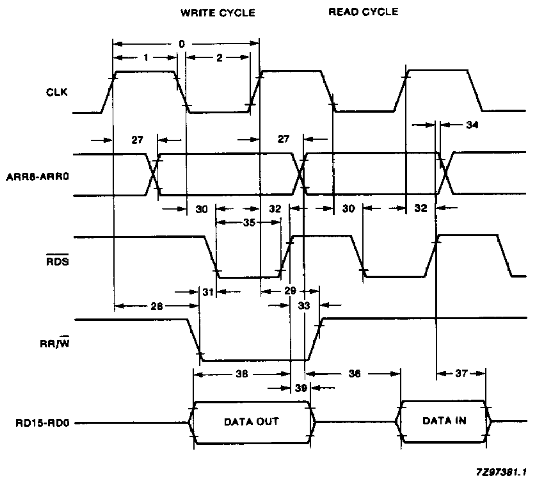
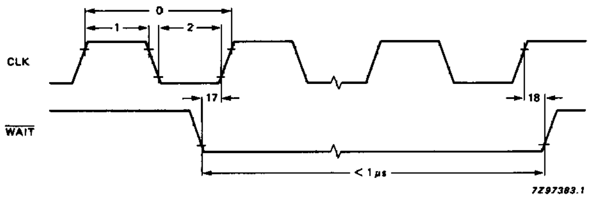


Fig. 4.3-7 Second parallel I/O timing; $\overline{\text{WAIT}}$ inactive (for PCB5011 only).



Note: As long as $\overline{\text{WAIT}}$ = 0 all clocked (by CLK) processor actions are suspended i.e. the internal status and all signals remain unchanged.

Fig. 4.3-8 $\overline{\text{WAIT}}$ signal timing.

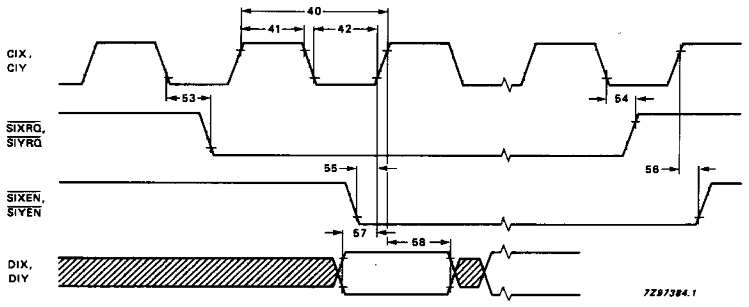


Fig. 4.3-9 Serial input timing.

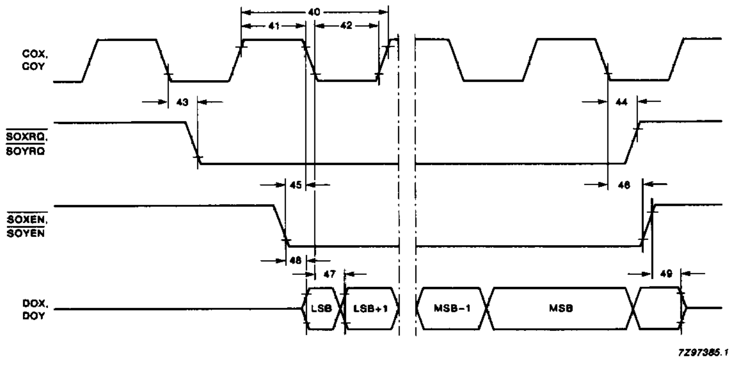


Fig. 4.3-10 Serial output timing.

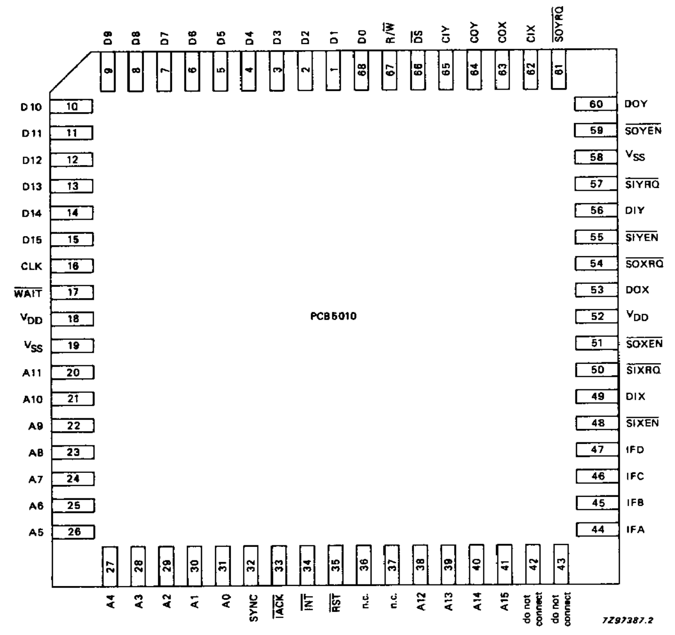


Fig. 5.1-2 Pinning for the PCB5010 (PLCC).

5.2 PCB5011 PACKAGE OUTLINE (dimensions in mm)

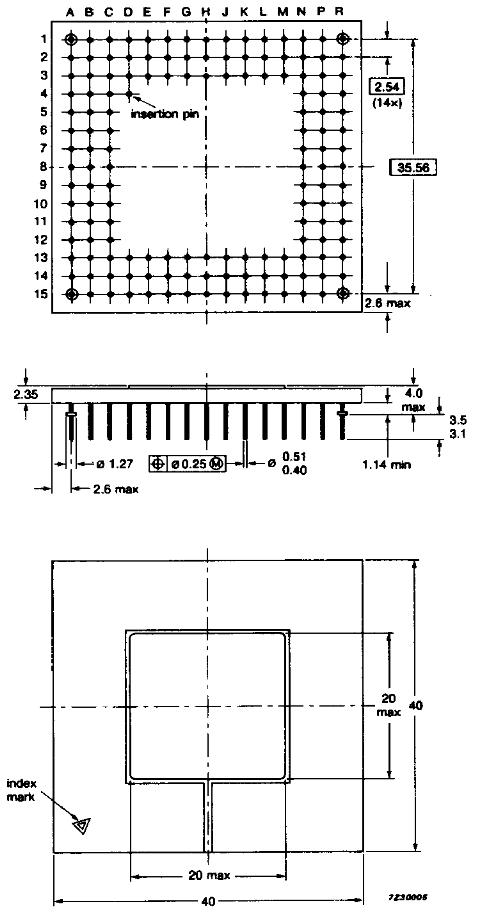


Fig. 5.2-1 144-pin grid array (PGA) (NO275) package for the PCB5011.

5.3 PIN ASSIGNMENT FOR Fig. 5.2-1

PCB5011 signal name	pin number
V_{DD}	N8, C8
V_{SS}	N9, A2
CLK	Q2
$\overline{RST}$	L14
D15	P3
D14	N4
D13	Q1
D12	P2
D11	N3
D10	M3
D9	P1
D8	N2
D7	L3
D6	M2
D5	N1
D4	M1
D3	L2
D2	L1
D1	K3
D0	K2
A15	A1
A14	B2
A13	C3
A12	C4
A11	N5
A10	Q3
A9	P5
A8	Q4
A7	N6
A6	P6
A5	Q5
A4	P7
A3	N7
A2	Q6
A1	Q7
A0	P8
$R/\overline{W}$	J2
$\overline{DS}$	K1
$\overline{WAIT}$	P4
DIX	B1

PCB5011 signal name	pin number
$\overline{SIXEN}$	C2
$\overline{SIXRQ}$	D2
CIX	D3
DIY	H3
$\overline{SIYEN}$	J3
$\overline{SIYRQ}$	H1
CIY	J1
DOX	C1
$\overline{SOXEN}$	E2
$\overline{SOXRQ}$	E3
COX	D1
DOY	G1
$\overline{SOYEN}$	F1
$\overline{SOYRQ}$	H2
COY	G3
$\overline{INT}$	M15
$\overline{IACK}$	M14
SYNC	N14
IFA	F3
IFB	F2
IFC	E1
IFD	G2
PA9	Q13
PA8	P12
PA7	N11
PA6	P13
PA5	Q14
PA4	N12
PA3	N13
PA2	P14
PA1	Q15
PA0	M13

(continued on next page)

DEVELOPMENT DATA

PCB5011 signal name	pin number
PD39	C5
PD38	B4
PD37	A3
PD36	A4
PD35	B5
PD34	A5
PD33	C6
PD32	B6
PD31	B7
PD30	A6
PD29	A7
PD28	C7
PD27	A8
PD26	B8
PD25	A9
PD24	A10
PD23	C9
PD22	B9
PD21	A11
PD20	B10
PD19	C10
PD18	A12
PD17	B11
PD16	A13
PD15	C11
PD14	B12
PD13	A14
PD12	B13
PD11	C12
PD10	G15
PD9	G13
PD8	K14
PD7	L15
PD6	J14
PD5	J13
PD4	K15
PD3	J15
PD2	H14
PD1	H15
PD0	H13
ARR8	Q8
ARR7	Q9
ARR6	Q10
ARR5	P9
ARR4	P10
ARR3	N10
ARR2	Q11
ARR1	P11
ARR0	Q12

PCB5011 signal name	pin number
RD15	F15
RD14	G14
RD13	F14
RD12	F13
RD11	E15
RD10	E14
RD9	D15
RD8	C15
RD7	D14
RD6	E13
RD5	C14
RD4	B15
RD3	D13
RD2	C13
RD1	B14
RD0	A15
RR/$\overline{\text{W}}$	L13
$\overline{\text{RDS}}$	N15
Do not connect	K13
Do not connect	P15
Do not connect	B3