National Semiconductor

August 1992

PC87911
Keyboard Controller with Real-Time Clock

General Description

The PC87911 integrates a keyboard controller and a Real-Time Clock (RTC) for use in PC based applications.

The keyboard controller block contains the system timing, control logic, RAM program memory, RAM data memory and 18 programmable I/O lines necessary to implement dedicated control functions. The device is designed to be an efficient controller. It has extensive bit handling capability as well as facilities for both binary and BCD arithmetic. Efficient use of program memory is derived from an instruction set comprised predominantly of single bytes. The remaining instructions are two bytes in length. The device contains 2 kbytes of program RAM in order to allow easy altering of program code. The keyboard controller is software compatible with the 8042AH microcontroller.

The RTC is a low-power block that provides a time-of-day clock and 100-year calendar with alarm features and battery operation. Other features include three maskable interrupt sources and 242 bytes of general purpose RAM. Valid RAM and time are maintained through the use of an external battery source. It is compatible with the DS1287 and MC146818 RTC.

Features

Keyboard Controller

- 8042AH software compatible
- 8-bit Microcomputer
- 2 kbytes program RAM and 256 bytes data RAM
- On-board configuration register for program downloading at power up
- Dedicated open drain outputs for keyboard controller application
- Asynchronous access to 2 data registers and 1 status register during normal operation
- Supports both interrupt and polling
- Supports DMA handshake
- 93 instructions
- 18 programmable I/O pins
- 4 dedicated open-drain outputs
- 8-bit Timer/Counter
- Binary and BCD arithmetic
- Expandable I/O
- 16 MHz clock

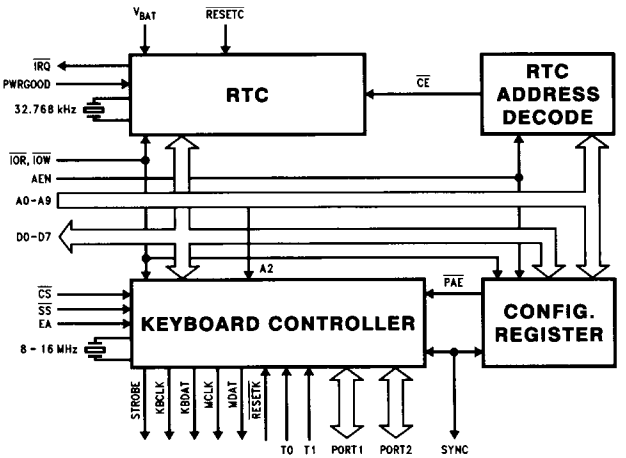
Real-Time Clock

- DS1287 and MC146818 compatible
- 242 bytes battery backed-up CMOS RAM in two banks
- Calendar in days, day of the week, months and years with automatic leap-year adjustment
- Time of day in seconds, minutes and hours:
  - — 12 or 24 hour format
  - — Optional daylight savings adjustment
- BCD or binary format for time keeping
- Three individually maskable interrupt event flags:
  - — Periodic rates from 122 μs to 500 ms
  - — Time-of-day alarm once per second to once per day
- Separate battery pin
- Double buffer time registers

General

- Low power CMOS technology
- Static design
- 68-pin Plastic Leaded Chip Carrier (PLCC) package

Block Diagram



TL/C/11470–1

TRI-STATE® is a registered trademark of National Semiconductor Corporation.

 TL/C/11470

RRD-B20M82/Printed in U. S. A.

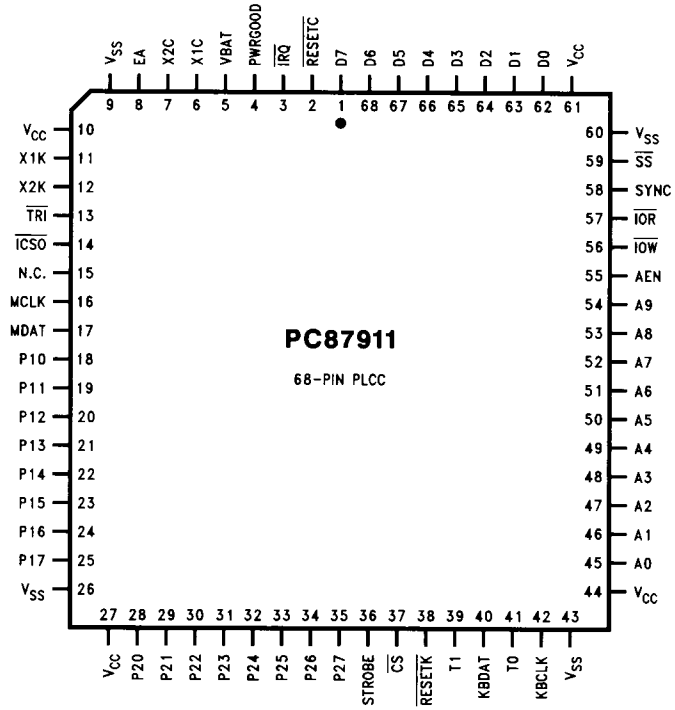
Table of Contents

List of Illustrations

List of Tables

1.0 Pin Description

CONNECTION DIAGRAM



TL/C/11470-2

1.0 Pin Description

Symbol	Pin	Type	Function
A0–A9	45–54	I	**A**ddress lines used in conjunction with the AEN input to address registers in the keyboard controller, the Real-Time Clock, and the configuration register.
AEN	55	I	**A**ddress **EN**able input used to validate accesses to the PC87911.
$\overline{CS}$	37	I	**C**hip **S**elect which enables the host to access the keyboard controller through D0-D7, including access to program memory.
D0–D7	62–68,1	I/O	Bi-directional **D**ata bus used to interface the PC87911 to the peripheral data bus of a host.
EA	8	I	**E**xternal **A**ccess input which will force the keyboard controller to fetch and execute instructions from external program memory. The program address will be sent out from P10–P17 and P20–P22 and the instruction will be fetched through D0–D7.
$\overline{ICSO}$	14	O	Internal **C**hip **S**elect **O**utput. Indicates either the RTC registers or the Configuration Register is being selected. It is active low.
$\overline{IOR}$	57	I	**I/O R**ead signal which enables a read operation to the PC87911 through D0–D7.
$\overline{IOW}$	56	I	**I/O W**rite signal which enables a write operation to the PC87911 through D0–D7.
$\overline{IRQ}$	3	O	Real-Time Clock **I**nterrupt **ReQ**uest output.
KBCLK	42	O	**K**ey**B**oard **CL**oc**K** output.
KBDAT	40	O	**K**ey**B**oard **DAT**a output.
MCLK	16	O	**M**ouse **CL**oc**K** output.
MDAT	17	O	**M**ouse **DAT**a output.
N.C.	15		**N**o **C**onnection.

1.0 Pin Description (Continued)

Symbol	Pin	Type	Function
P10–P17	18–25	I/O	Quasi-bidirectional **P**ort for general purpose input and output. The lower 8 bits of the program counter will be sent to P10–P17 when the chip is in Single Step mode or External Access mode.
P20–P27	28–35	I/O	Quasi-bidirectional **P**ort for general purpose input and output. The upper 3 bits of the program counter will be sent to P20–P22 when the chip is in Single Step mode or External Access mode. P20–P23 also serve as a 4-bit I/O expander bus when an 8243 compatible I/O expander is used.
PWRGOOD	4	I	An input to the PC87911 indicating that the **PoWeR** supply is **GOOD**. This input should be held low until the power supply is stable.
$\overline{\text{RESETC}}$	2	I	**RESET** input for the Real-Time Clock. Resets various flags and status bits in the Real-Time Clock.
$\overline{\text{RESETK}}$	38	I	**RESET** input for the **K**eyboard controller which initializes the processor and sets the program counter to zero.
$\overline{\text{SS}}$	59	I	**S**ingle **S**tep input used in conjunction with the SYNC output to single step the keyboard controller through instructions. This input should be tied to a logic 1 for normal operation.
STROBE	36	O	Address/data output **STROBE** to an 8243 compatible I/O expander.
SYNC	58	I/O	Clock output **SYNC**hronized to the keyboard controller instruction cycles. Can be used as an external strobe or with $\overline{\text{SS}}$ to single step the keyboard controller. SYNC is an input when PWRGOOD is low and is sampled by the rising edge of PWRGOOD. If it is sampled as low, the I/O address of the configuration register will be 6Eh. If it is sampled high, the address will be 7Fh. There is an internal pull up resistor on this pin.
T0	41	I	This input can be directly tested by the conditional jump instructions JT0 and JNT0 of the keyboard controller.
T1	39	I	This input can be directly tested by the conditional jump instructions JT1 and JNTI. T1 can also be used as the external input to the 8-bit counter/timer of the keyboard controller.
$\overline{\text{TRI}}$	13	I	TRI-STATE® input. All outputs (except X2K, X2C) are put into TRI-STATE when this pin is tied low.
VBAT	5	P	Real-Time Clock **BAT**tery pin.
V_{CC}	10, 27, 44, 61	P	Power supply pins.
V_{SS}	9, 26, 43, 60	P	Ground pins.
X1C	6	I	Input for the internal Real-Time Clock crystal oscillator amplifier.
X2C	7	O	Output for the internal Real-Time Clock crystal oscillator amplifier.
X1K	11	I	Input for the internal keyboard controller crystal oscillator amplifier.
X2K	12	O	Output for the internal keyboard controller crystal oscillator amplifier.

2.0 Functional Description

The PC87911 is designed to replace the real-time clock and the keyboard controller in a PC/AT compatible computer. *Figure 2-1* shows the comparison of a conventional design using standard keyboard controller and RTC chips vs a design Using the PC87911. The PC87911 not only combines two functional blocks together, it integrates some discrete logic such as open-collector drivers, CMOS clock oscillator, and the RTC power switching circuitry into a single chip.

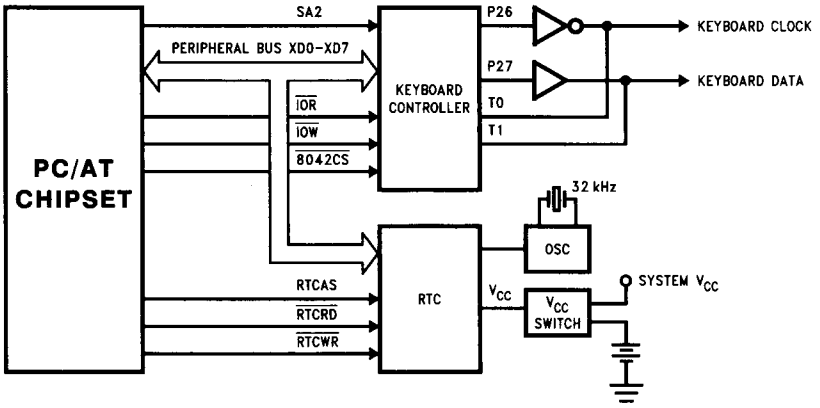
The following sections provide descriptions for each functional block of the PC87911.

2.1 CONFIGURATION REGISTER

There is one 8-bit configuration register in the PC87911. It controls the operating mode of the keyboard controller, selects different banks of CMOS RAM in the RTC, and selects the RTC test mode. *Figure 2-2* shows the bit definition of the configuration register.

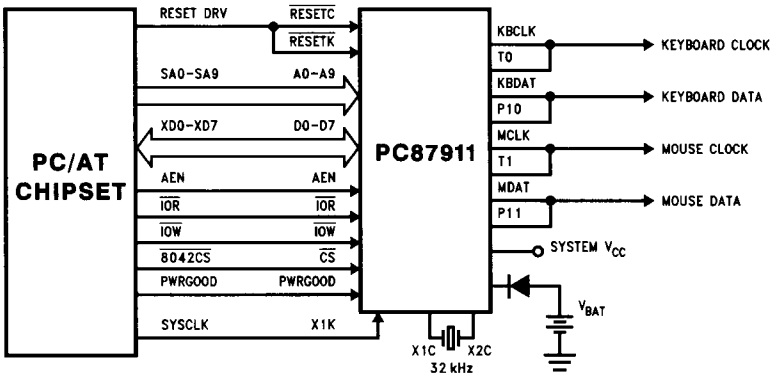
REV1	REV0	rsv	rsv	rsv	RAMSEL	CLKTEST	$\overline{PAE}$
bit 7	6	5	4	3	2	1	0

FIGURE 2-2. Configuration Register



TL/C/11470-3

FIGURE 2-1a. Conventional Keyboard Controller and RTC Circuit in PC/AT



TL/C/11470-4

FIGURE 2-1b. Keyboard Controller and RTC Replaced by PC78911

2.0 Functional Description (Continued)

Bit 0 **$\overline{PAE}$**. Program Access Enable of the keyboard controller

= 1: the keyboard controller is in normal mode

= 0: the keyboard controller is in Program Access Mode. See Program Memory Section in the Keyboard Controller for details.

Bit 1 **CLKTEST**. RTC clock test mode select

= 1: SYNC pin outputs the 32 kHz RTC clock

= 0: SYNC pin outputs the keyboard controller's SYNC signal

Bit 2 **RAMSEL.** RTC CMOS RAM bank select

= 1: Select the upper 128 bytes of CMOS RAM

= 0: Select the lower 128 bytes of CMOS RAM

Bits 3-5 **Reserved**. users should not change the power-up default values.

Bits 6,7 **REV**. Revision of the silicon (Read only)

REV1, REV0 = 00 PC87910

REV1, REV0 = 01 PC87911

REV1, REV0 = 10 Future Revisions

REV1, REV0 = 11 Future Revisions

The configuration register will be initialized to 40h when PWRGOOD is low.

The configuration register is I/O mapped to locations 6Eh or 7Fh of the system I/O space. These two locations are selected by a power strapping option on the SYNC pin as shown below.

Address of the configuration register:

= 7Fh SYNC pin is open or connected to V_{CC} through a 2k resistor

= 6Eh SYNC pin is connected to GND through a 2k resistor

The configuration register is a read/write register (except bits 6, 7). A normal I/O read will read its content. However, in order to protect it from being over-written accidentally, a double write scheme is used for writing to it. Two consecutive I/O write operations must be done. The data of the first operation is irrelevant, the data of the second operation will be written into the register.

2.2 PC87911 SYSTEM INTERFACE

The PC87911's keyboard controller, RTC, and the configuration register are interfaced to the system through a common system interface. The interface consists of the address bus A0–A9, AEN, data bus D0–D7, and control signals $\overline{IOR}$, $\overline{IOW}$, and $\overline{CS}$.

Address bus A0–A9 connects to the system address SA0–SA9 of the system. The PC87911 uses the system address and the AEN signal to decode the access to the configuration register. The RTC also makes use of the address to decode the index and data registers. Therefore, the RTC doesn't require the RTCAS, $\overline{RTCRD}$ and $\overline{RTCWR}$ signals from the system.

Data bus D0–D7 connects to the peripheral bus XD0–XD7 of the system.

The $\overline{CS}$ pin is the keyboard controller chip-select. It must be connected to the keyboard controller chip-select signal of the system. The PC87911 doesn't decode the keyboard controller chip-select from A0–A9.

The $\overline{IOR}$ and $\overline{IOW}$ inputs connect to the $\overline{IOR}$ and $\overline{IOW}$ lines of the system. All read and write operations to the PC87911 are I/O operations.

2.3 KEYBOARD CONTROLLER

Figure 2-3 shows the functional block diagram of the keyboard controller in the PC87911. The keyboard controller is a general purpose 8-bit microcontroller. It is software compatible to the industry standard keyboard controller 8042AH. All object codes written for 8042AH can run on the PC87911. The controller consists of 256 bytes of data memory and 2 kbytes of program memory. Unlike the 8042AH, the program memory of the PC87911 keyboard controller is RAM based. Control programs such as the keyboard BIOS have to be downloaded into the program memory at power-up. The PC87911 has a configuration register and a program access logic designed for this purpose.

2.3.1 Program Memory

The keyboard controller of the PC87911 has a 2k x 8 RAM based program memory. An 11-bit program counter allows direct access to every location of the program memory. *Figure 2-4* shows the memory map. There are three special locations associated with hardware functions.

1. 000h After the keyboard controller is reset, the program counter is initialized to 000h.
2. 003h When the input buffer of the host interface (DBBIN) is full and the IBF interrupt is enabled, the CPU will do an interrupt call to this location.
3. 007h When the timer is overflow and the timer interrupt is enabled, the CPU will do an interrupt call to this location.

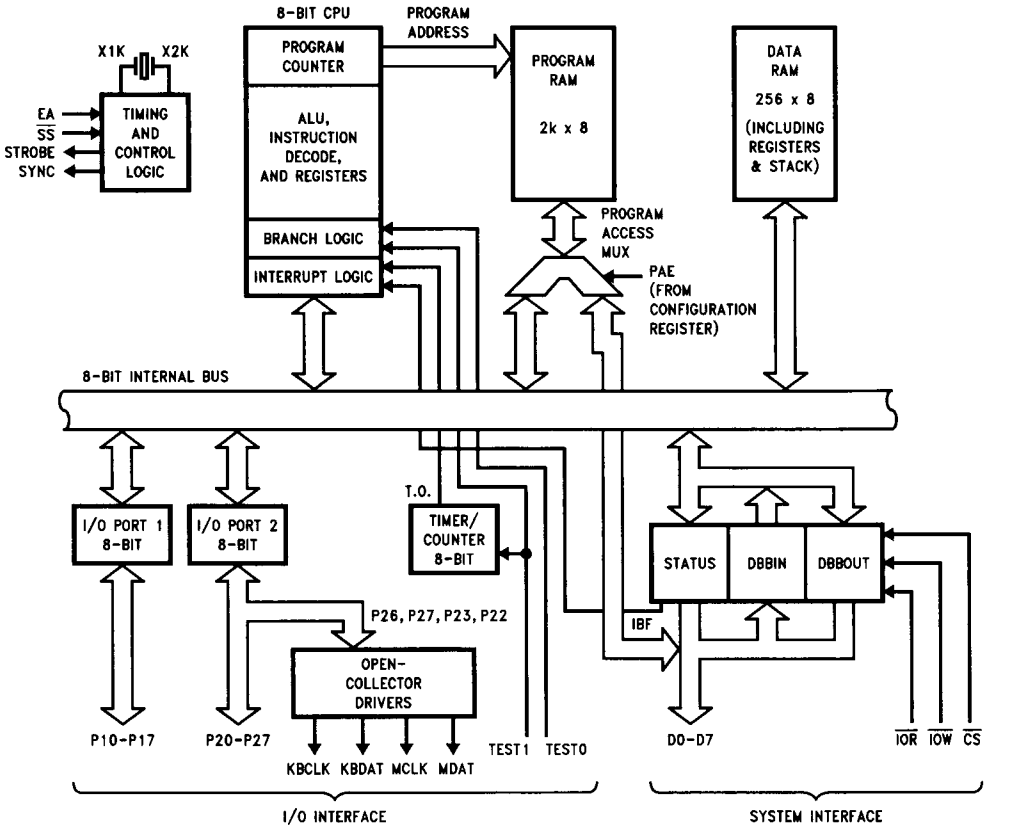


FIGURE 2-3. Functional Block Diagram of PC87911 Keyboard Controller

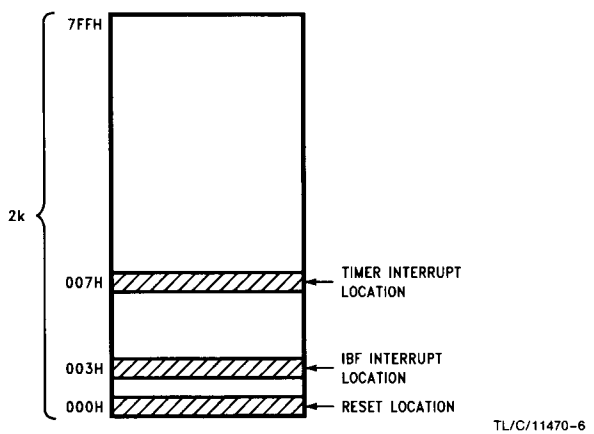


FIGURE 2-4. PC87911 Keyboard Controller Program Memory Map

2.0 Functional Description (Continued)

2.3.2 Program Access Mode

Because the program memory is RAM based, the PC87911 provides a mode to the users to gain access to the program memory through the system interface. Therefore, the users can download the program into the program memory at power-up, examine the contents of the program memory, or even swap-in another section of program code while the system is running. The $\overline{PAE}$ (Program Access Enable) bit in the PC87911 configuration register controls the operation mode of the keyboard controller.

When the $\overline{PAE}$ bit is 1, the keyboard controller is in the normal operating mode. When the $\overline{PAE}$ bit is 0, the keyboard controller is in the Program Access Mode. Operation of the chip in the Program Access Mode is as follows:

1. When $\overline{PAE}$ is written a "0", the keyboard controller will finish the current instruction (if it is running), stop fetching new instructions, and will start running the NOP instruction. The CPU status flags and all I/O ports will stay unchanged.
2. At this time, the upper 8 bits of the program counter and the program memory location pointed to by the program counter are accessible through the system interface (D0–D7, $\overline{IOR}$, $\overline{IOW}$, $\overline{CS}$).

   A command write to DBBIN ($\overline{CS}$ = 0, $\overline{WR}$ = 0, $\overline{RD}$ = 1, A2 = 1) will put the data on the data bus into the upper 8 bits of the program counter. The lower 3 bits of the program counter will be reset to 0.
3. A read to the Status Register ($\overline{CS}$ = 0, $\overline{WR}$ = 1, $\overline{RD}$ = 0, A2 = 1) will read the upper 8 bits of the current program counter.
4. A data write to DBBIN ($\overline{CS}$ = 0, $\overline{WR}$ = 0, $\overline{RD}$ = 1, A2 = 0) will write data into the program RAM pointed to by the PC. The PC will be incremented by 1 after each write operation. It will go back to 0 after counting to 2047.
5. A read to DBBOUT ($\overline{CS}$ = 0, $\overline{WR}$ = 1, $\overline{RD}$ = 0, A2 = 0) will read the contents of the program RAM pointed to by the PC. The PC will be incremented by 1 after each read operation. It will go back to 0 after counting to 2047.
6. When the $\overline{PAE}$ bit is programmed back to a "1", it terminates the program access function. The keyboard controller will start executing instructions from the location pointed to by the CURRENT PC. Therefore programmers should set the PC properly before leaving the Program Access Mode.

Whenever the PWRGOOD signal of the PC87911 is low, the $\overline{PAE}$ bit in the configuration register will be initialized to 0, i.e., the keyboard controller will be in the Program Access Mode.

Accessing or modifying the program code while the chip is running requires some precautions. First, the user has to guarantee program integrity. Second, the timer, if started, will remain counting while the keyboard controller is in the PA mode. This feature helps the user to keep track of real-time processes, however, the timer interrupt must be disabled to avoid an interrupt hit while in the PA mode. Third, if DMA is enabled, P27 (it becomes $\overline{DACK}$ as described later) must be held high in the PA mode. This is because $\overline{CS}$ and A2 will be overridden while $\overline{DACK}$ is low.

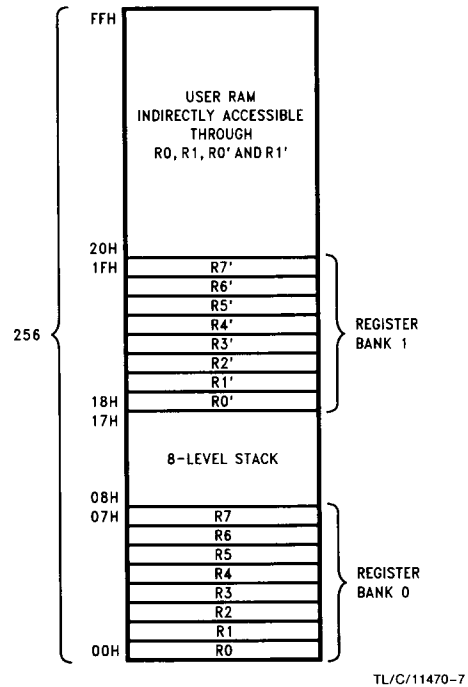
TABLE 2-1. Summary of Program Access Mode

$\overline{IOR}$	$\overline{IOW}$	A2	$\overline{CS}$	Operation
0	1	0	0	Read Program Memory
0	0	0	0	Write Program Memory
0	1	1	0	Read Upper 8 Bits of Program Counter
1	0	1	0	Write Upper 8 Bits of Program Counter, Reset Lower 3 Bits
X	X	X	1	No Operation

$\overline{PAE}$ bit in configuration register = 0.

2.3.3 Data Memory and Registers

The keyboard controller has 256 bytes of data memory. This includes 2 banks of registers, 8 registers each, and an 8-level stack. *Figure 2-5* shows the data RAM organization.



TL/C/11470–7

FIGURE 2-5. PC87911 Keyboard Controller Data Memory Map

RAM locations 0h–7h are used as register bank0. They are designated as R0–R7 correspondingly. Register bank1 (R0'–R7') are located in 18h-1Fh of the address map. Bank0 is the default register bank when the chip comes out from reset. Bank switching is accomplished by using "register bank select" instructions (SEL RB0, SEL RB1). Locations 8h-17h are reserved for the stack. Each stack entry consists of 2 bytes. *Figure 2-6* shows the organization of the stack. The stack pointer in the PSW register points to the top of the stack. A "0" in the stack pointer corresponds to RAM locations 8H and 9H.

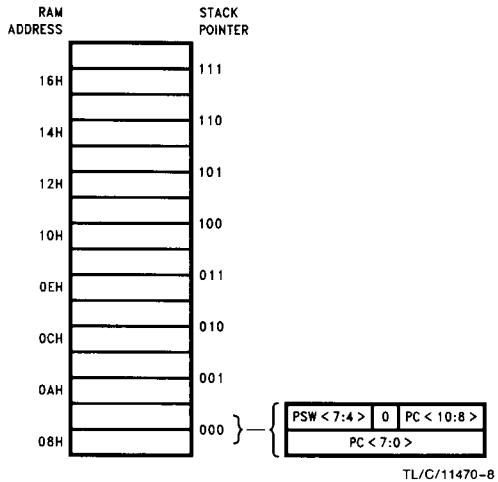


FIGURE 2-6. Keyboard Controller Stack Organization

2.3.4 Program Status Word

There is an 8-bit register in the keyboard controller that holds the program execution status. The bit definition of the PSW register is shown in *Figure 2-7*.

	CY	AC	F0	BS	rsv	SP <2:0>
bit	7	6	5	4	3	2-0

FIGURE 2-7. Program Status Word Register

Bit 7 **CY**. Carry flag of the accumulator.

Bit 6 **AC**. Auxiliary Carry flag of the accumulator, i.e., carry from bit 3 to 4 of the ALU.

Bit 5 **F0**. Flag 0. A general purpose software flag.

Bit 4 **BS**. The current active register bank, 0 = bank 0, 1 = bank 1.

Bit 3 **Reserved**. User should not change its power-up value.

Bits 2-0 **Stack Pointer**. 3-bit Stack Pointer for 8-level stack.

When the CPU performs a subroutine call or interrupt call, the PC and the upper 4 bits of the PSW are pushed into the stack. Upon return, the PSW can be restored in option. If a RETR instruction is executed, PSW will be restored. PSW will not be restored if an RET instruction is executed.

2.3.5 System Intertace

Figure 2-3 illustrates that the system interface of the keyboard controller consists of three 8-bit registers: DBBOUT, DBBIN, and STATUS.

The DBBOUT register is used to transfer data from the keyboard controller to the host system. It is written by the keyboard controller using the OUT DBB,A instruction. A data read operation by the host system reads its content.

The DBBIN register is used to transfer data from the host system to the keyboard controller. It is written by the host system. It is read by the keyboard controller using an IN A, DBB instruction.

The STATUS register holds status information of the system interface. *Figure 2-8* shows the bit definition. It is read-only by the system.

	ST7	ST6	ST5	ST4	F1	F0	IBF	OBF
bit	7	6	5	4	3	2	1	0

FIGURE 2-8. Status Register

Bit 0 **OBF**. Output Buffer Full. A "1" indicates that data is written into the DBBOUT register. It will be cleared by a system read operation.

Bit 1 **IBF**. Input Buffer Full. When a write operation is done by the host system, it will be set to 1. Upon executing an IN A,DBB instruction, it will be cleared.

Bit 2 **F0**. A general purpose flag that can be cleared or toggled by the keyboard controller software.

Bit 3 **F1**. Command/Data Flag. This flag holds the state of A2 while a write operation is done by the host system. Usually it is used to distinguish commands/data from the host system, e.g., a write with A2 = 1 (hence F1 = 1) can be defined as a command, or vice versa.

Bits 4-7 **ST4–ST7**. General purpose flags. They can be written by a MOV STS,A instruction.

Table 2-2 shows that address A2 determines which registers to be accessed.

TABLE 2-2. Summary of System Interface Operations

$\overline{IOR}$	$\overline{IOW}$	A2	$\overline{CS}$	Operation
0	1	0	0	Read DBBOUT
1	0	0	0	Write DBBIN, F1 Clear
0	1	1	0	Read STATUS
1	0	1	0	Write DBBIN, F1 Set
X	X	X	1	No Operation

2.0 Functional Description (Continued)

The system interface of the PC87911 also makes use of P24–P27 to enhance the handshaking with the host system. P24 and P25 can be used to reflect the states of the OBF and IBF flags in the STATUS register. This function is enabled by the "EN FLAGS" instruction. After executing this instruction, P24 becomes the OBF pin and P25 becomes the $\overline{IBF}$ pin. Data written to P24 and P25 serve as a mask to the output. When a "1" is written, P24 will output the state of the OBF flag and P25 will output the complement state of the IBF flag. When a "0" is written, the output will be forced to zero. These OBF and $\overline{IBF}$ pins can be used as the interrupt requests to the host system for data transfer handshaking.

The "EN DMA" instruction turns P26 and P27 into DRQ and $\overline{DACK}$ pins. After executing this instruction, P26 and P27 can be used for DMA transfer between the host system and the PC87911. P26 will go active when a "1" is written to it. It can be used as the DRQ to the host system. When P27 which is connected to the $\overline{DACK}$ of the system is pulled active, data can be read or written through D0–D7, hence completing the DMA cycle. P26 is reset to zero when P27 is pulled active or the "EN DMA" instruction is executed. The later case allows the PC87911 to abort a DMA request.

2.3.6 I/O Interface

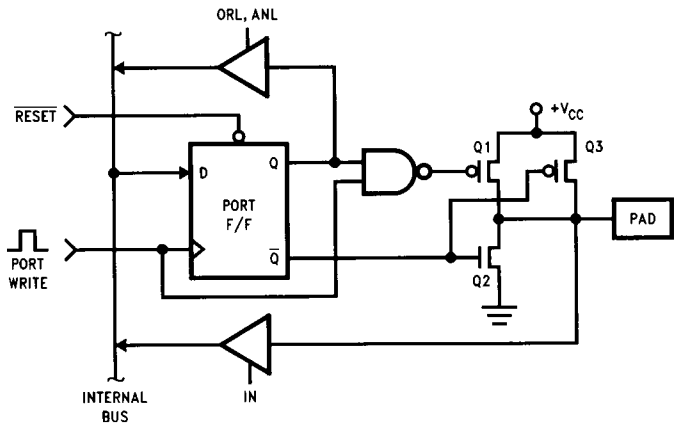
The keyboard controller of the PC87911 provides 16 general purpose I/O lines, 4 open-collector output lines and 2 input lines as shown in the Functional Block Diagram in *Figure 2-3*.

2.3.6.1 General Purpose I/O

The 16 general purpose I/O lines are P10–P17 and P20–P27, mapped to Port 1 and Port 2 respectively. These I/O lines are quasi-bidirectional because the output buffer cannot be turned off even if the I/O line is intended for input. *Figure 2-9* illustrates the structure of the I/O line. Q1 and Q2 are the normal output transistors. When the I/O line is intended for output, a "0" is written into the flip-flop, which will latch, and Q2 will turn on. When a "1" is written, Q1 will turn on briefly because it is gated by a short port-write pulse also. Q1 will charge up the pad to near V_{CC} and then turn off. An active pull-up transistor Q3 will turn on also, but it provides a large pull-up resistance because it is a weak transistor. At this stage, the I/O line can be used as input: a low impedance low voltage can override the pin and the input buffer will read a low level input. If the pin is driven high or undriven, a high level will be read. Therefore, in order to use a port pin as input, a logic "1" must first be written to it. When the keyboard controller is reset, all port lines will be initialized to logic "1".

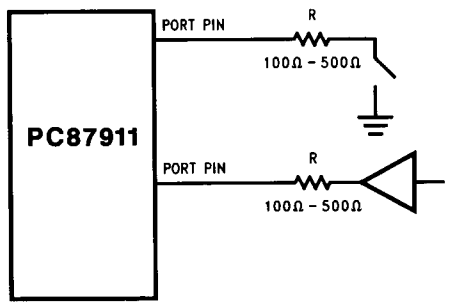
Because Q1 will turn on momentarily when a write (or ANL, ORL operation) to the port is done, a series resistor to the port lines used as input is recommended to limit the surge current *(Figure 2-10)*.

An active pull-up technique is used instead of a pull-up resistor in the PC87911 design. Q3 will turn on only when Q2 is off, hence eliminating a steady current flow when the port line is programmed a low output.



TL/C/11470–9

FIGURE 2-9. Active Pull-Up I/O Port Structure



R: current limiting resistor

A small-value series current limiting resistor is recommended when PORT PINS are used as inputs.

TL/C/11470–10

FIGURE 2-10. Current Limiting Resistor

2.3.6.2 Open-Collector Outputs

In order to reduce the glue logic used in a PC/AT compatible environment, there are four dedicated open-collector outputs: KBCLK, KBDAT, MCLK and MDAT. KBCLK is the complement of P26. KBDAT is the complement of P27. MCLK is the complement of P23. MDAT is the complement of P22. These four drivers can drive 16 mA, making them suitable for driving keyboard and mouse cables. TEST0 and TEST1 must be externally connected to KBCLK and KBDAT in the PC/AT compatible applications.

2.3.6.3 Test Inputs

TEST0 and TEST1 are two dedicated input pins. There are conditional jump instructions checking directly the level of these two pins. TEST1 also serves as the event counter input.

2.3.6.4 Connection to the I/O Expander

The number of I/O pins of the keyboard controller can be expanded by adding one or more 8243 I/O Expander chips. Each I/O expander offers 16 I/O port pins designated as Port 4, 5, 6, and 7. The keyboard controller makes use of the P20-P24 and the Strobe pin to interface with the 8243.

2.3.7 Timer/Counter

The keyboard controller is equipped with an 8-bit counter which can be used as a timer or an event counter. *Figure 2-11* shows the 2 different clock sources for the counter. Selection of the clock source is done by software.

2.3.7.1 Timer Operation

The counter can be set to the timer operation mode by connecting its clock source to the internal timing generator. The clock frequency to the timer is equal to the oscillator frequency divided by 480 (or SYNC frequency divided by 32).

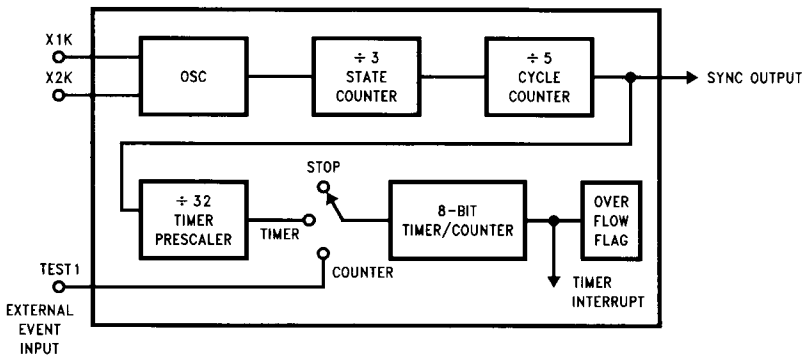
The initial value of the timer is programmable. After the timer is started (by STRT T instruction), it will count up continuously until it is stopped or the keyboard controller is reset. The Timer Overflow Flag will be set when the count value overflows from FFh back to 00h. The Timer Overflow Flag can be tested by a conditional jump instruction (JTF). This instruction will also reset the flag. When the timer interrupt is enabled, an interrupt will occur when the timer overflows. This will be discussed more in the interrupt section.

2.3.7.2 Event Counter Operation

When the clock input of the counter is switched to the external input (TEST1), it essentially becomes an event counter. The falling edge of the signal on the TEST1 pin causes the counter to increment. The external input frequency should be equal to or less than the SYNC frequency. Timer Overflow Flag and Timer interrupt operate in the same way as they do in the timer mode.

2.3.8 Interrupt

The keyboard controller of the PC87911 provides 2 different internal interrupts. They are the Input Buffer Full (IBF) interrupt and Timer Overflow interrupt. These two interrupts can be independently enabled or disabled by software. Both of them are disabled when the chip comes out from reset.



TL/C/11470–11

FIGURE 2-11. Timing Generation and Timer Circuit

2.0 Functional Description (Continued)

2.3.8.1 Timer Interrupt

If the timer interrupt is enabled, upon the timer overflow, an interrupt occurs. It will cause the program to perform a subroutine call to program address 007h. The interrupt will be cleared by this subroutine call. The current Program Counter and the upper 4 bits of the PSW will be pushed into the stack before the call occurs. At the end of the timer interrupt service routine, an RETR instruction will restore the PSW and the PC. Because the timer interrupt has a lower priority than the IBF interrupt, simultaneous IBF and timer interrupts will cause the timer interrupt to be pending. It will be served as soon as the program returns from the IBF interrupt service routine.

2.3.8.2 Input Buffer Full (IBF) Interrupt

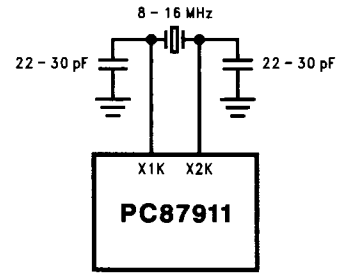
If the IBF interrupt is enabled, when there is a host write operation to the keyboard controller ($\overline{CS}$ = 0, $\overline{IOW}$ = 0), an interrupt occurs. The processor will save the current Program Counter and the upper 4 bits of the PSW into the stack and perform a subroutine call to the program address 003h. Upon entering the interrupt service routine, further interrupts are held off. The interrupt is re-enabled upon the execution of the RETR instruction.

2.3.9 Oscillator and Instruction Timing

Because the RTC requires a battery back-up oscillator, the keyboard controller runs on a separate oscillator. The oscillator pins of the keyboard controller are X1K and X2K. *Figure 2-12* shows the connections for internal and external clock configuration.

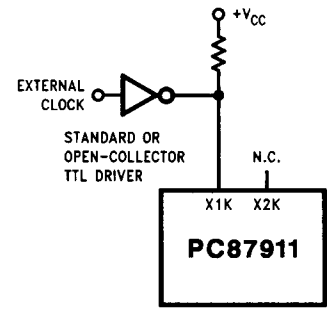
The oscillator clock is divided by 3 to generate the state timing, then is further divided down by 5 to generate the instruction timing *(Figure 2-13)*. Hence each instruction cycle consists of 5 states and 15 clock cycles. A SYNC signal is generated at state 3 to 4 of every instruction cycle. This SYNC signal is available at the SYNC pin of the PC87911. It can be used to synchronize the external circuit and single step circuit as discussed later. Because the timing generation circuit is synchronized by the keyboard controller reset signal $\overline{RESETK}$, SYNC output will stay high when $\overline{RESETK}$ is low.

Most of the keyboard controller instructions require only one instruction cycle. The others require two cycles. Refer to the instruction set for details.



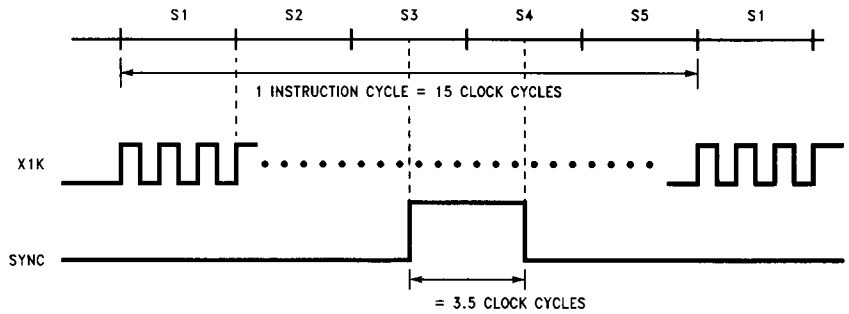
TL/C/11470-12

FIGURE 2-12a. Internal Clock Connection



TL/C/11470-13

FIGURE 2-12b. External Clock Connection



TL/C/11470-14

FIGURE 2-13. Instruction Cycle Timing and SYNC Output

2.0 Functional Description (Continued)

2.3.10 Program Development Functions

The keyboard controller of the PC87911 provides two functions to facilitate program development; Single-Step and External Access Mode.

2.3.10.1 Single-Step

By adding some external logic, a user can single step the execution of a program. *Figure 2-14* shows an example of a single-step circuit. One instruction will be executed when the button is pushed once. At the same time, the program address of the next instruction is available on P17–P10 (lower 8-bit of the PC) and P22–P20 (higher 3-bit of the PC). The waveform diagram in *Figure 2-15* helps explain the operation of the single-step circuit.

When the single-step $\overline{SS}$ input of the chip is sampled low by the rising edge of SYNC, the processor will stop. The SYNC output will stay high and the PC will be presented on the PORT lines. The internal circuit continuously samples $\overline{SS}$ until it is brought high.

When the button is pushed, flip-flop A acts as a debounce circuit and sends a clean clock pulse to flip-flop B. The $\overline{SS}$ pin (Q output of the flip-flop B) becomes high, making the processor return to normal operation. The SYNC output will then return to low as the instruction cycle proceeds. It clears flip-flop B and makes $\overline{SS}$ drop again. At the next instruction cycle, when the processor samples the $\overline{SS}$ being low, it stops again.

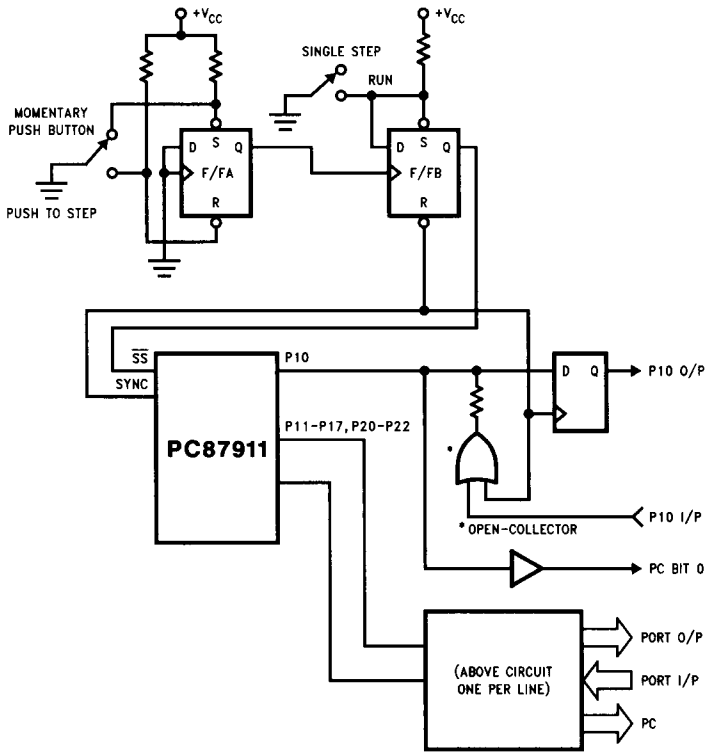


FIGURE 2-14. Example of Single-Step Circuit

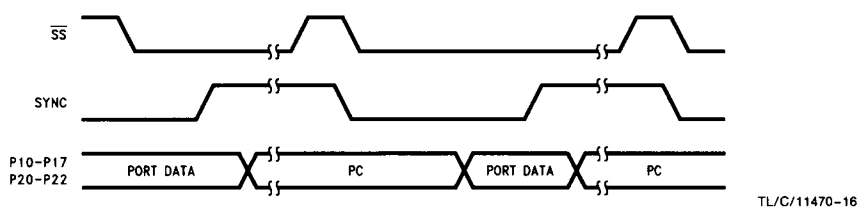


FIGURE 2-15. Single-Step Operation

2.0 Functional Description (Continued)

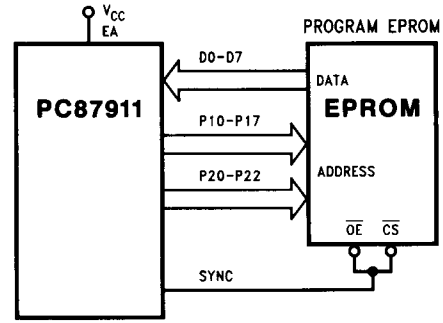
If it is necessary to maintain the normal I/O function of the port pins, a latch and an open-collector OR gate can be used as illustrated in *Figure 2-14*. The latch latches the port O/P data at the rising edge of the SYNC signal. The open-collector OR allows the input line to drive the port pin only when SYNC is low.

2.3.10.2 External Access Mode

In a prototype environment, when the program downloading hardware or software is not available, the External Access Mode would be a convenient way for program development. In the External Access Mode, the keyboard controller will fetch program codes externally through the system data bus D0–D7. The program address will be available on PORT lines P10–P17 and P20–P22. This mode is enabled by pulling the EA pin high. The EA pin is sampled when $\overline{\text{RESETK}}$ transitions from low to high. This mode cannot be changed while $\overline{\text{RESETK}}$ is high.

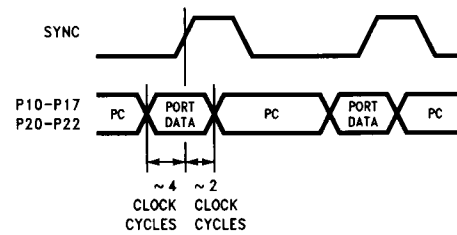
Figure 2-16 shows the connection of the PC87911 to an external program EPROM.

Figure 2-17 illustrates the timing waveforms of the PORT lines in the EA Mode. At the falling edge of SYNC. PC0–7 is available on P10–17, and PC8–10 is available on P20–22. At the rising edge of SYNC, the PORT lines will output port data as normal. A latch can be used to separate the port data from the program address. Although the system data bus is used for sending program data, it is still possible to access the system interface registers. The only limitation is these accesses must be restricted in the time during which SYNC is high.



TL/C/11470–17

FIGURE 2-16. Connection of an External Program EPROM in EA Mode



TL/C/11470–18

FIGURE 2-17. Timing Waveform of the PORT Lines in EA Mode

TABLE 2-3. RTC Memory Map

Index	Function	BCD Format	Binary Format	Comments
(RAMSEL = 0)				
00	Seconds	00–59	00–3b	R/W
01	Seconds Alarm	00–59	00–3b	R/W
02	Minutes	00–59	00–3b	R/W
03	Minutes Alarm	00–59	00–3b	R/W
04	Hours	12 hr. = 01–12 (AM)	01–0c (AM)	R/W
		12 hr. = 81–92 (PM)	81–8c (PM)	R/W
		24 hr. = 00–23	00–17	R/W
05	Hours Alarm	12 hr. = 01–12 (AM)	01–0c (AM)	R/W
		12 hr. = 81–92 (PM)	81–8c (PM)	R/W
		24 hr. = 00–23	00–17	R/W
06	Day of Week	01–07	01–07 (Sunday = 1)	R/W
07	Date of Month	01–31	01–1f	R/W
08	Month	01–12	01–0c	R/W
09	Year	00–99	00–63	R/W
0A	Control Register A			Bit 7 is Rd Only
0B	Control Register B			R/W
0C	Control Register C			All Bits Rd Only
0D	Control Register D			All Bits Rd Only
0E–3F	Battery-Backed RAM (114 Bytes)			R/W
(RAMSEL = 1)				
00–3F	Battery-Backed RAM (128 Bytes)			R/W

2.0 Functional Description (Continued)

2.4 REAL TIME CLOCK

The RTC in the PC87911 is a low-power clock that provides a time-of-day clock and 100-year calendar with alarm features and battery operation. Other features include three maskable interrupt sources and 242 bytes of general purpose RAM. Valid RAM and time can be maintained through the use of an external battery source. It is software compatible with the DS1287 and MC146818.

2.4.1 Memory Map

The RTC contains 14 time/control registers and 242 bytes of general purpose battery backed static RAM. Refer to Table 2-3.

2.4.2 Bus Interface

The RTC function is mapped to I/O locations 70H (index) and 71H (data). This decode is done internal to the PC87911.

2.4.3 Time Generation

The Time Generation function divides the 32.768 kHz from the X1C–X2C pins down to a one hertz signal. The divider chain is controlled by bits 6–4 of Control Register A. Bits 3–0 of Control Register A select one of fifteen taps from the divider chain to be used as a Periodic Interrupt. See Control Register A in the Control Register and Interrupt section for divider configurations and rate selections.

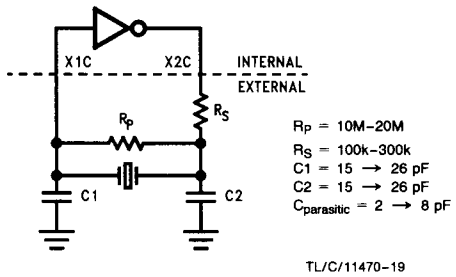
During divider reset (bits 6–4 of Control Register A = 11x), the divider chain is reset to 0. An update will occur 500 ms after the divider chain is activated into normal operational mode (bits 6–4 of Control Register A = 010). The periodic flag will also become active 1/2 period of the programmed value when the divider chain is activated.

Figure 2-18 represents the internal and external circuitry that comprise the oscillator. The oscillator input may be driven from an external source. If this is desired, the input should be driven rail to rail and be approximately a 50% duty cycle. The oscillator output should be open in this case. The external capacitor values should be chosen to provide the manufacturer's specified load capacitance for the crystal when combined with the parasitic capacitance of the trace, socket, and package, which can vary from 2 pF to 8 pF. The rule of thumb in choosing these capacitors is:

$C_L = (C1 * C2)/(C1 + C2) + C_{parasitic}$

$C2 > C1$

C1 can be trimmed to obtain the 32768.0 Hz.



TL/C/11470–19

FIGURE 2-18. Oscillator Internal and External Circuitry

The start-up time of this oscillator may vary from two to seven seconds and is due to the high "Q"of the crystal. The parameters below describe the crystal to be used:

Parallel Resonant, tuning fork (N cut) or XY Bar

$Q \geq 35k$

Load Capacitance (C_L)	9 pF to 13 pF
Accuracy	User Choice
Temperature Coefficient	User Choice

2.4.4 Time Keeping

The time is kept in BCD or binary format. The format is determined by bit 2 of Control Register B (DM). Either 12 or 24 hour representation for the hours can be maintained as determined by bit 1 of Control Register B (24/12). Note that when changing the format the time registers must be re-initialized to the corresponding data format.

Daylight savings and leap year exceptions are handled by the time keeping function. When bit 0 of Control Register A (DSE) is a "1", time advances from 1:59:59 AM to 3:00:00 on the first Sunday in April. On the last Sunday of October time changes from 1:59:59 to 1:00:00 when daylight savings is enabled. On leap year February is extended to 29 days.

The time is updated once per second. If a read of the timing registers coincides with an update, data read may not be valid. Also, writes to time registers during an update will have undefined results.

The RTC in the PC87911 provides a user copy of the time registers to avoid accessing invalid data. When a "1" is written to the SET bit (bit 7 of Control Register B), the user copy of time is frozen while the time registers continue to keep updating time. This bit can be set without concern of an update occurring. This is one of the four methods for reading and writing time to ensure that the correct time is written or read. These are:

Method 1—Set the SET bit.

Set the SET bit to "1". This will take a snapshot of the internal time registers and load it into the user copy. When the SET bit goes from 1 → 0 the user copy will update the internal registers if user copy registers have been updated.

Method 2—Access after detection of an Update Ended Interrupt.

This signifies that an update has just completed and there are 999 ms remaining until another one will occur.

Method 3—Poll update-in-progress (bit 7 in Control Register A). The update will occur 244 μs after the update-in-progress bit goes high. Therefore if a 0 is read, there will be a minimum of 244 μs in which the time is guaranteed to remain stable.

Method 4—Use of a Periodic interrupt to determine if an update cycle is in progress.

The periodic interrupt is first set to a desired period. The program can then use the periodic interrupt to signify that there is (Period of Periodic Interrupt/2 + 244 μs) remaining until another update will occur.

2.0 Functional Description (Continued)

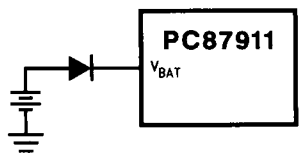
The Alarm condition is also generated by the Time Keeping function. After each update, the seconds, minutes, and hours are compared with the seconds alarm, minutes alarm, and hours alarm. If equal, the Alarm lag is set in Control Register C. This will cause an interrupt condition (IRQZ = "0") if the Alarm Interrupt Enable bit is set in Control Register B. If both bit 7 and bit 6 of an alarm byte (seconds alarm, minutes alarm, hours alarm) is a "1" then that byte is a "don't care".

2.4.5 RAM

The RAM data is accessed at locations 0E–3F when RAMSEL (bit 2) of the configuration registers is "0", and locations 00–3F when RAMSEL is "1". Battery backed power enables the RAM to retain information during system power-down.

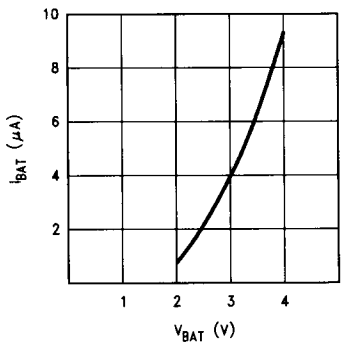
2.4.6 Power Management/System Bus LockOut/Power-up Detection/Oscillator

The Power Management function provides power to the RTC in the PC87911. During system operation, power from the system is used. When system voltage falls below battery voltage, the Power Management function switches the RTC cell to battery power. For proper operation, a 500 mV differential is needed between V_{CC} and VBAT. *Figure 2-19* represents a typical battery configuration and *Figure 2-20* represents typical battery current during battery backed mode.



TL/C/11470–20

FIGURE 2-19. Typical Battery Configuration



TL/C/11470–21

FIGURE 2-20. Typical Battery Current during Battery Backed Mode

As the RTC switches to battery power all inputs are locked out so that the internal registers can not be modified. This lockout condition continues for 62 ms (min) to 125 ms (max) after the RTC switches from battery power to system power. The 62 ms–125 ms lockout during power transition from battery to system power depends on the following conditions:

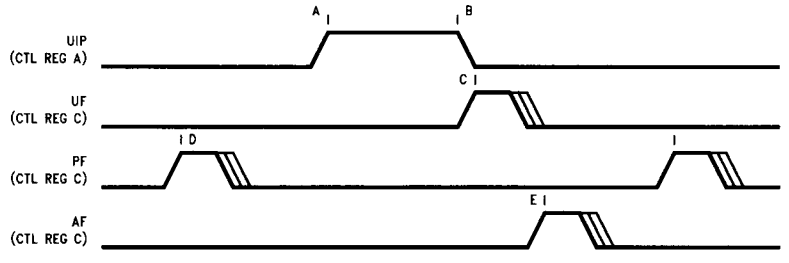
1. If the Divider Chain Control (bits 6–4 in Control Register A) is in any mode but Normal Operation ("010"), all inputs will be enabled immediately upon detection of system voltage above that of battery voltage.
2. When Battery voltage is below 1 volt and $\overline{\text{RESETC}}$ is a "0", all inputs are enabled immediately upon detection of system voltage above that of battery voltage. This will also initialize registers 00 through 0D to be "00".
3. If the VRT bit (bit 7 in Control Register D) is a "0", all inputs are enabled immediately upon detection of system voltage above that of battery voltage.

When power is applied to the RTC in the PC87911, the Oscillator is operational with the following exceptions:

1. The VRT bit (bit 7 in Control Register D) is a "0". The oscillator is disabled after initial power-up. This reduces power consumption during the time between when the battery is initially installed and when the RTC is initialized for timekeeping.
2. The Divider Chain Control (bits 6–4 in Control Register A) is in Oscillator Disabled modes ("000", and "001"). This provides a means for the user to "shut-down" the oscillator and reduce the power consumption of the RTC cell. The RAM remains functional when the oscillator is disabled.

2.4.7 Interrupt Handling

The Periodic, Alarm, and Update ended Interrupts are generated (IRQZ is driven low) when the respective enable bits in Control Register B are set and an interrupt condition occurs. A read from Register C clears the active interrupt. If a second interrupt condition occurs (other than that which caused the interrupt) during a read from Register C, IRQZ will remain active (low). Thus, it is recommended that when multiple interrupts are enabled, the interrupt service routine continues to read (and service interrupts) until bit 7 of Control Register C (IRQ Flag) returns to a "1". Note that if an interrupt is not serviced before a second condition of the same interrupt occurs, the second interrupt event will be lost.



TL/C/11470-22

A–B, C Update in Progress Bit High before Update Occurs = 244 μs
D–C Periodic interrupt to update = Period (periodic int)/2 + 244 μs
C–E Update to Alarm Interrupt = 30.5 μs

UIP Update In Progress Status Bit
UF Update Ended Flag (Update Ended Interrupt if Enabled)
PF Periodic Flag (Periodic Interrupt if Enabled)
AF Alarm Flag (Alarm Interrupt if Enabled)

Flags (and IRQ) are reset at the conclusion of Control Register C read or by $\overline{\text{RESETC}}$ low.

FIGURE 2-21. Interrupt/Status Timing

2.4.8 Control Registers

The Control Registers/Interrupt Generation contains four registers which can be accessed at any time during non-battery backed operation. Control Registers at address A, B, C, and D are described below.

Control Register A

	UIP	DV2	DV1	DV0	RS3	RS2	RS1	RS0
bit	7	6	5	4	3	2	1	0

Bits 0–3 RS[3:0]—Periodic Interrupt Rate Select
(Read/Write)

These bits control the rate of the periodic interrupt. Reset has no effect on these bits.

RS[3:0]				Periodic Interrupt Rate	
0	0	0	0	None	
0	0	0	1	3.90625	ms
0	0	1	0	7.8125	ms
0	0	1	1	122.070	μs
0	1	0	0	244.141	μs
0	1	0	1	488.281	μs
0	1	1	0	976.562	μs
0	1	1	1	1.953125	ms
1	0	0	0	3.90625	ms
1	0	0	1	7.8125	ms
1	0	1	0	15.625	ms
1	0	1	1	31.25	ms
1	1	0	0	62.5	ms
1	1	0	1	125	ms
1	1	1	0	250	ms
1	1	1	1	500	ms

Bits 4–7 DV[2:0]—Divider Chain Control
(Read/Write)

These bits control the configuration of the divider chain in the Timing Generation function.

Reset has no effect on these bits.

DV[2:0]			Divider Chain Configuration
0	0	0	Oscillator Disabled
0	0	1	Oscillator Disabled
0	1	0	Normal Operation
0	1	1	TEST
1	0	0	TEST
1	0	1	TEST
1	1	0	Divider Chain RESET
1	1	1	Divider Chain RESET

Bit 7 UIP—Update in Progress
(Read Only)

= 1: Signifies that the timing registers will be updated within 244 μs.

= 0: Signifies that an update will not occur before 244 μs. This bit will read 0 when bit 7 of Control Register B (SET) is a 1.

Reset has no effect on this bit.

Control Register B

	SET	PIE	AIE	UIE	0*	DM	24/12	DSE
bit	7	6	5	4	3	2	1	0

Bit 0 DSE—Daylight Savings Enable
(Read/Write)

= 1: Enables daylight savings. These two conditions are as follows:

Daylight Savings Spring: Time advances from 1:59:59 to 3:00:00 on the first Sunday in April.

Daylight Savings Fall: Time advances from 1:59:59 to 1:00:00 on the last Sunday in October.

= 0: Disables the daylight savings feature.

Reset has no effect on this bite.

2.0 Functional Description (Continued)

Bit 1 **24/12—24 or 12 Hour Mode**
(Read/Write)
= 1: Enables 24 hour format.
= 0: Enables 12 hour format.
Reset has no effect on this bit.

Bit 2 **DM—Data Mode**
(Read/Write)
= 1: Enables BINARY format.
= 0: Enables BCD format.
Reset has no effect on this bit.

Bit 3 **0***
This bit is defined as "Square Wave Enable" by the MC146818 and is not supported by the RTC Cell. This bit will always be read as a "0".

Bit 4 **UIE—Update Ended Interrupt Enable**
(Read/Write)
= 1: Enables generation of the update ended interrupt. This interrupt is generated at the time an update occurs.
= 0: Disables generation of the update ended interrupt. Reset forces this bit to a "0".

Bit 5 **AIE—Alarm Interrupt Enable**
(Read/Write)
= 1: Enables generation of the alarm interrupt. The alarm interrupt will be generated immediately after a time update in which the Seconds, Minutes, and Hours time compares with their respective alarm counterparts.
= 0: Disables generation of the alarm interrupt. Reset forces this bit to a "0".

Bit 6 **PIE—Periodic Interrupt Enable**
(Read/Write)
= 1: Enables generation of the periodic interrupt. Bits 3–0 of Control Register A determine the rate of the Periodic interrupt.
= 0: Disables generation of the Periodic interrupt.
Reset forces this bit to a "0".

Bit 7 **SET—Set mode**
(Read/Write)
= 1: The user copy of time is "frozen" allowing the time registers to be accessed without regard for an occurrence of an update.
= 0: The timing updates occur normally.
Reset has no effect on this bit.

Control Register C

	IRQF	PF	AF	UF	0	0	0	0
bit	7	6	5	4	3	2	1	0

Bits 0–3 Reserved "0"

Bit 4 **UF—Update Ended Interrupt Flag**
(Read Only)
= 1: When the time registers are updated. This bit is reset to a "0" at the conclusion of a read from this register.
Reset forces this bit to a "0".

Bit 5 **AF—Alarm Interrupt Flag**
(Read Only)
= 1: When an alarm condition is detected. This bit is reset to a "0" at the conclusion of a read from this register.
Reset forces this bit to a "0".

Bit 6 **PF—Periodic Interrupt Flag**
(Read Only)
= 1: When a transition occurs on the selected tap of the divider chain. This bit is reset to a "0" at the conclusion of a read from this register.
Reset forces this bit to a "0".

Bit 7 **IRQF—Interrupt Request Flag**
(Read Only)
= 1: When PIE and PF + AIE and AF + UIE and UF = 1. The IRQF bit mirrors the value on the IRQZ output. When IRQZ is active (low), IRQF is a "1".

Control Register D

	VRT	0	0	0	0	0	0	0
bit	7	6	5	4	3	2	1	0

Bits 0–6 Reserved "0"

Bit 7 **VRT—Valid RAM and Time**
(Read Only)
When a "1", this bit indicates that the contents of the RTC are valid. When a "0", this bit indicates that the battery source is low and that the RTC data and RAM data are questionable. This bit is set to a "1" at the conclusion of a read from this register.

3.0 Programming of the Keyboard Controller

The PC87911 keyboard controller instruction set is 100% opcode compatible with the 8042 microcontroller. There are 93 instructions in total. Most of them are single-byte instructions. The vest are 2-byte instructions. Instruction execution times range from 1 instruction cycle (15 clock cycles) to 2 instruction cycles. Refer to *Figure 2-13.*

3.1 ADDRESSING MODES

The PC87911 keyboard controller instruction set provides 4 basic addressing modes plus variations. Immediate addressing can be found in most of the ALU instructions and data move instructions. The 8-bit data operand immediately follows the instruction opcode. Example: 'MOV A, #data', it moves an immediate data into the accumulator.

Register addressing mode allows easy access to some of the data memory area. Data memory locations 0h–7h are designated as registers R0 to R7 respectively. Many ALU and data move instructions can make direct reference to these locations by referring to R0 to R7. Example: 'ADD A, R1', adds the accumulator with the data in data memory location 01h. In addition, another 8 data memory locations 18h–1Fh can be used as an alternate set of registers. By setting bit 4 (register bank select bit) of the program status word to 1, any references making to R0 through R7 will be mapped to data memory locations 18h through 1Fh. The power-up state of this bit is 0, i.e. R0–R7 are mapped to 0h–7h. Two special instructions: 'SEL RB0' and 'SEL RB1' are provided for register bank switching.

Indirect addressing mode provides access to all 256 bytes of data memory. The address of the operand is pointed to by either R0 or R1. Example: 'MOV A, @R0', moves the data with address in R0 to the accumulator.

3.2 JUMP INSTRUCTIONS

Page addressing mode is used in most of the jump instructions. All conditional jump instructions contain an 8-bit page address (addr). If the condition is true (a jump is going to occur), the lower 8-bit of the program counter will be replaced by the page address (addr) of the instruction. As a result, one should keep in mind that first, addr is not a relative address, second, if the instruction opcode is located at the end of a page, it will jump to the following page because the program counter has rolled to next page when addr is fetched.

Unconditional jump instruction and call instruction, however, use absolute addressing. The 11-bit address (aaddr) allows jumps or calls to any locations in the program memory. An indirect jump is provided for vector-table jump based on the contents of the accumulator.

3.3 INPUT/OUTPUT INSTRUCTIONS AND PORT PINS

The PC87911 keyboard controller has I/O instructions to transfer data between the accumulator and the port pins (P10–P17, P20–P27), the expander, and the system interface. P10–P17 is designated as port 1 while P20–P27 is designated as port 2. Port 4 to port 7 are available only when an I/O expander (8243) is connected to the PC87911. These expander ports are 4-bit wide. Instructions dealing with port 1 and 2 are different from the ones dealing with port 4 to port 7. Refer to the Instruction Set section for details of individual I/O instructions. Each port pin has a flip-flop holding the output state of the pin. However, because all port pins are quasi-bidirectional (refer to 2.3.6.1), the flip-flop output of the port pin may be different from the actual state of the pin. The input instruction reads directly the state of the port pins, hence the data reflects the actual state of the pins. On the other hand, the AND and OR instructions operate on the outputs of the port flip-flops. These instructions should be used to set/reset individual port pins.

The 'OUT DBB, A' and 'IN A, DBB' instructions provide communication between the PC87911 and the host system. The 'OUT DBB, A' instruction sends accumulator content to the DBBOUT register and set the OBF flag in the STATUS register. A system read from the DBBOUT register clears the flag. The 'IN A, DBB' instruction reads the DIBBIN register content into the accumulator and resets the IBF flag. A system write to the DBBIN register sets the IBF flag. The status of these flags can be polled by the host system by reading the STATUS register. The handshaking can be interrupt driven also. The 'EN FLAGS' instruction brings out the states of the flags from port pins P24 and P25. They can be used to interrupt the host system. On the other hand, if it is decided to use DMA transfers to communicate with the PC87911 for minimal CPU intervention, one can use the 'EN PMA' instruction. It converts P26 and P27 to DRQ and $\overline{\text{DACK}}$ pins. Refer to 2.3.5 for more details.

3.4 PC87911 KEYBOARD CONTROLLER INSTRUCTION SET

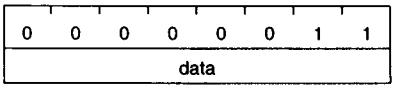
TABLE 3-1. Symbols Used in the Keyboard Controller Instructions

Symbol	Description
aaddr	11-bit Program Address
addr	8-bit Program Address (i.e. Page Address)
Bb	Bit Position Designator (b = 0–7) b2 b1 b0 Bb 0 0 0 B0 0 0 1 B1 0 1 0 B2 0 1 1 B3 1 0 0 B4 1 0 1 B5 1 1 0 B6 1 1 1 B7
data	8-bit Data
Pp	Port Designator (p = 1, 2 or 4–7) p0 p1 Pp(1,2) Pp(4–7) 0 0 X P4 0 1 P1 P5 1 0 P2 P6 1 1 X P7
Rr	Register Designator (r = 0,1 or 0–7) r Rr(0,1) r2 r1 r0 Rr(0–7) 0 R0 0 0 0 R0 1 R1 0 0 1 R1 0 1 0 R2 0 1 1 R3 1 0 0 R4 1 0 1 R5 1 1 0 R6 1 1 1 R7
#data	Immediate Data
@Rr	Indirect Addressing Referring to Rr

ADD A, #data

Add immediate data to the accumulator.

Instruction Format:



Description:

Add the immediate data to the accumulator. Result is stored back to the accumulator.

Cycles: 2
Bytes: 2
Flags: C, AC

Example:

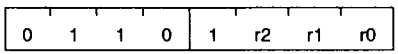
Check if the accumulator content is greater than F0h (unsigned)

```
ADD  A,#0Fh   ;add A with 0Fh, C is set if A
              ;> F0h
JC   A_greater_than_F0
```

ADD A, Rr

Add contents of register to the accumulator.

Instruction Format:



Description:

Add the contents of a register to the accumulator. Result is stored back to the accumulator. Any register from R0 to R7 can be specified.

Cycles: 1
Bytes: 1
Flags: C, AC

Example:

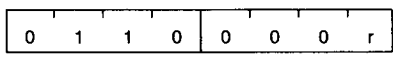
Add R0 with A

```
ADD  A,R0   ;add R0 with A
```

ADD A, @Rr

Add indirectly the contents of data memory to the accumulator.

Instruction Format:



Description:

Add the contents of a data memory to the accumulator. The address of the memory location is specified by R0 or R1. Result is stored back to the accumulator.

Cycles: 1
Bytes: 1
Flags: C, AC

Example:

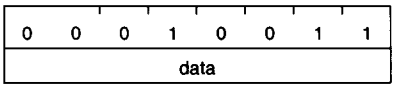
Add data in location 7Fh with A

```
LOC  EQU 7Fh       ;define data
                   ;memory location
     MOV R0, #LOC  ;initialize R0
     ADD A, @R0    ;add data of
                   ;location 7Fh with A
```

ADDC A, #data

Add immediate data with carry to the accumulator.

Instruction Format:



Description:

Add the immediate data with the carry bit to the accumulator. Result is stored back to the accumulator.

Cycles: 2
Bytes: 2
Flags: C, AC

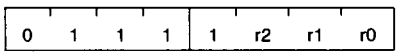
Example:

```
MOV  A,#0FFh   ;Move FFh to A, A=FFh
ADD  A,#1h     ;Add A with 1h, A=0, C=1
ADDC A,#10h    ;Add A with 10h and C, A=11h
```

ADDC A, Rr

Add the contents of a register with carry to the accumulator.

Instruction Format:



Description:

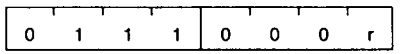
Add the contents of a register with the carry bit to the accumulator. Result is stored back to the accumulator. Any register from R0 to R7 can be specified.

Cycles: 1
Bytes: 1
Flags: C, AC

ADDC A, @Rr

Add indirectly the contents of data memory with carry to the accumulator.

Instruction Format:



Description:

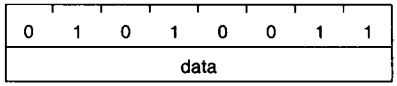
Add the contents of a data memory with the carry bit to the accumulator. The address of the memory location is specified by R0 or R1. Result is stored back to the accumulator.

Cycles: 1
Bytes: 1
Flags: C, AC

ANL A, #data

Logical AND immediate data to the accumulator.

Instruction Format:



Description:

Logical AND the immediate data to the accumulator. Result is stored back to the accumulator.

Cycles: 2
Bytes: 2
Flags: none

Example:

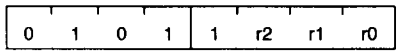
Mask off the upper nibble of the accumulator.

```
MOV  A,#55h  ;A=55h
ANL  A,#0Fh  ;AND with mask 0Fh, A=05h
```

ANL A, Rr

Logical AND contents of a register to the accumulator.

Instruction Format:



Description:

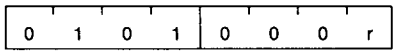
Logical AND the contents of a register to the accumulator. Result is stored back to the accumulator. Any register from R0 to R7 can be specified.

Cycles: 1
Bytes: 1
Flags: none

ANL A, @Rr

Logical AND indirectly the contents of data memory to the accumulator.

Instruction Format:



Description:

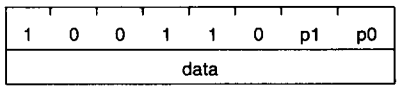
Logic AND the contents of a data memory location to the accumulator. The address of the memory location is specified by R0 and R1. Result is stored back to the accumulator.

Cycles: 1
Bytes: 1
Flags: none

ANL Pp, #data

Logical AND immediate data with port (1–2).

Instruction Format:



Description:

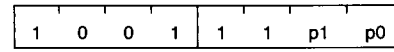
Logical AND the contents of the port register (NOTE: not the states of the port pins) with the immediate data. The result is stored back to the port register. The port of operation is specified by p1 and p0. Refer to Table 3-1 for port mapping.

Cycles: 2
Bytes: 2
Flags: none

ANLD Pp, A

Logical AND contents of the accumulator with port (4–7).

Instruction Format:



Description:

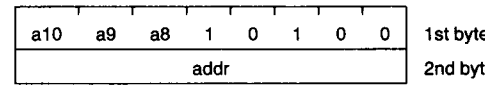
Logical AND the contents of the port register of the expander (port 4–7) with the lower nibble of the accumulator. The result is stored back to the port register. The port of operation is specified by p1 and p0. Refer to Table 3-1 for port mapping.

Cycles: 2
Bytes: 1
Flags: none

CALL aaddr

Call subroutine.

Instruction Format:



Description:

The contents of both the program counter and bit 4 to 7 of the program status word are saved on the stack. The stack pointer is then incremented by one. Address aaddr is loaded into the program counter. This instruction allows subroutine call to any locations within the 2K program memory.

Cycles: 2
Bytes: 2
Flags: none

Example:

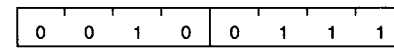
A basic subroutine call.

```
     CALL SUBR      ;main program calls
                    ;a subroutine SUBR
     (next inst.)   ;next instruction executed
     .              ;after the completion
     .              ;of the subroutine
SUBR:.              ;beginning of
     .              ;the subroutine
     .              ;body of the subroutine
     RET            ;end of subroutine, return
                    ;to the main program
```

CLR A

Clear the contents of the accumulator.

Instruction Format:



Description:

The contents of the accumulator are cleared to zero.

Cycles: 1
Bytes: 1
Flags: none

CLR C

Clear the carry flag to zero.

Instruction Format:

1	0	0	1	0	1	1	1

Description:

The carry flag is cleared to zero.

Cycles: 1
Bytes: 1
Flags: C

CLR F0

Clear the flag 0 to zero.

Instruction Format:

1	0	0	0	0	1	0	1

Description:

The flag 0 is cleared to zero.

Cycles: 1
Bytes: 1
Flags: F0

CLR F1

Clear the flag 1 to zero.

Instruction Format:

1	0	1	0	0	1	0	1

Description:

The flag 1 is cleared to zero.

Cycles: 1
Bytes: 1
Flags: F1

CPL A

Complement the contents of the accumulator.

Instruction Format:

0	0	1	1	0	1	1	1

Description:

Complement (1's complement) the contents of the accumulator. Result is stored back to the accumulator.

Cycles: 1
Bytes: 1
Flags: none

CPL C

Complement the carry flag.

Instruction Format:

1	0	1	0	0	1	1	1

Description:

The carry flag is complemented.

Cycles: 1
Bytes: 1
Flags: C

CPL F0

Complement the Flag 0.

Instruction Format:

1	0	0	1	0	1	0	1

Description:

The Flag 0 is complemented.

Cycles: 1
Bytes: 1
Flags: F0

CPL F1

Complement the Flag 1.

Instruction Format:

1	0	1	1	0	1	0	1

Description:

The Flag 1 is complemented.

Cycles: 1
Bytes: 1
Flags: F1

DA A

Decimal adjust the contents of the accumulator.

Instruction Format:

0	1	0	1	0	1	1	1

Description:

If BCD data representation is used in the programming, contents of the accumulator may require adjustment after arithmetic operations. The DA A instruction adjusts the 8-bit binary contents of the accumulator to form two 4-bit BCD digits. If the lower nibble of the accumulator is greater than 9 or if the auxiliary carry bit is set, the accumulator will be incremented by 6. The upper nibble will then be checked. If it is greater than 9 or if the carry bit is set, the upper nibble will be incremented by 6, and the carry bit will be set.

Cycles: 1
Bytes: 1
Flags: C

Example:

Each BCD addition must be followed by a DA A instruction to perform the decimal adjust. The following example shows the effect on the accumulator.

```
MOV  A,#6Fh   ;A=6Fh
DA   A        ;A=75h (BCD 75d)
ADD  A,#04h   ;A=79h
DA   A        ;no change, A=79h (BCD 79d)
ADD  A,#08h   ;A=81h, AC=1
DA   A        ;A=87h (BCD 87d)
ADD  A,#90h   ;A=17h, C=1
DA   A        ;A=77h, C=1 (BCD 177d)
```

DEC A

Decrement the contents of the accumulator by one.

Instruction Format:

0	0	0	0	0	1	1	1

Description:

Decrement the contents of the accumulator by one.

Cycles: 1
Bytes: 1
Flags: none

DEC Rr

Decrement register by one.

Instruction Format:

1	1	0	0	1	r2	r1	r0

Description:

Decrement the contents of a register by one. Any registers from R0 to R7 can be specified.

Cycles: 1
Bytes: 1
Flags: none

DIS I

Disable the IBF interrupt.

Instruction Format:

0	0	0	1	0	1	0	1

Description:

Disable the Input Buffer Full interrupt. This instruction inhibits IBF interrupt sequence. However, IBF interrupt request will be pending.

Cycles: 1
Bytes: 1
Flags: none

DIS TCNTI

Disable interrupt flag for timer/counter overflow.

Instruction Format:

0	0	1	1	0	1	0	1

Description:

Disable the timer/counter interrupt. This instruction clears any pending timer/counter interrupt. It will not affect the operation of the timer/counter. If the timer/counter is operating and overflows, the timer flag will be set, but no interrupt will occur.

Cycles: 1
Bytes: 1
Flags: none

DJNZ Rr, addr

Decrement register and branch if it is not zero.

Instruction Format:

1	1	1	0	1	r2	r1	r0	1st byte
data								2nd byte

Description:

This instruction decrements the contents of a register by one, then tests if it is zero. If it is zero, the next sequential instruction is executed. If the register is non-zero, the program execution jumps to the specified address addr. Any registers from R0 through R7 can be specified.

Because addr is a page address, program can only jump within the same page as where addr is located. For example, if the opcode is located at the last byte of page 2 and addr is located at the beginning of page 3, program will jump within page 3. This rule applies to all 2-byte conditional jump instructions.

Cycles: 2
Bytes: 2
Flags: none

Example:

A simple delay loop:

```
    MOV R1,#Delay   ;set the delay value
LP: NOP             ;or any instructions to
                    ;make a delay
    DJNZ R1,LP      ;loop until R1 = 0
```

EN DMA

Enable DRQ, $\overline{\text{DACK}}$ lines (P26, P27).

Instruction Format:

1	1	1	0	0	1	0	1

Description:

Enable the DMA handshaking function. P26 becomes the DRQ line and P27 becomes the $\overline{\text{DACK}}$ line. Writing a "1" to P26 after enabling DMA will set DRQ to 1. When the external circuit acknowledges the DMA request by pulling down $\overline{\text{DACK}}$ and the $\overline{\text{IOR}}$ or $\overline{\text{IOW}}$, DRQ will be reset to zero. Executing EN DMA instruction also resets DRQ to zero.

Cycles: 1
Bytes: 1
Flags: none

EN FLAGS

Enable OBF, $\overline{\text{IBF}}$ lines (P24, P25).

Instruction Format:

1	1	1	1	0	1	0	1

Description:

Enable the DBB (Data Bus Buffer) handshaking flags functions. P24 reflects the state of the OBF flag and P25 reflects the complement state of the IBF flag. Data written to P24 and P25 after EN FLAGS instruction serve as a mask to that output pin, i.e., a "1" written will enable the pin to reflect the state of the flag and a '0' written will force the pin to zero.

Cycles: 1
Bytes: 1
Flags: none

EN I

Enable the IBF interrupt.

Instruction Format:

0	0	0	0	0	1	0	1

Description:

Enable the Input Buffer Full interrupt. Any host write operation to the DBBIN will initiate the interrupt sequence. The processor will perform a subroutine call to the program address 003h.

Cycles: 1
Bytes: 1
Flags: none

EN TCNTI

Enable the interrupt flag for timer/counter overflow.

Instruction Format:

0	0	1	0	0	1	0	1

Description:

Enable the timer/counter interrupt. An overflow of the timer/counter will initiate the interrupt sequence. The processor will perform a subroutine call to the program address 007h.

Cycles: 1
Bytes: 1
Flags: none

IN A, DBB

Read DBBIN to the accumulator, clear IBF flag.

Instruction Format:

0	0	1	0	0	0	1	0

Description:

Read the contents of the DBBIN register into the accumulator, clear the IBF flag.

Cycles: 1
Bytes: 1
Flags: IBF

IN A, Pp

Input data from port (1-2) into the accumulator.

Instruction Format:

0	0	0	0	1	0	p1	p0

Description:

Read the states of the port pins into the accumulator. The read in value is independent of the contents of the port register. The port of operation is specified by p1 and p0. Refer to Table 3-1 for port mapping.

Cycles: 2
Bytes: 1
Flags: none

INC A

Increment the accumulator by one.

Instruction Format:

0	0	0	1	0	1	1	1

Description:

Increment the contents of the accumulator by one.

Cycles: 1
Bytes: 1
Flags: none

INC Rr

Increment the register by one.

Instruction Format:

0	0	0	1	1	r2	r1	r0

Description:

Increment the contents of a register by one. Any registers from R0 to R7 can be specified.

Cycles: 1
Bytes: 1
Flags: none

INC @Rr

Increment indirectly data memory location by one.

Instruction Format:

0	0	0	1	0	0	0	r

Description:

Increment the contents of a data memory location by one. The address of the location is stored in either R0 or R1.

Cycles: 1
Bytes: 1
Flags: none

JBb addr

Jump if accumulator bit is set.

Instruction Format:

b2	b1	b0	1	0	0	1	0	1st byte
addr								2nd byte

Description:

Program execution will jump to addr if the accumulator bit "b" is set, otherwise it will execute the next sequential instruction. b2, b1 and b0 specify the accumulator bit to be tested. Refer to Table 3-1 for the bit representation. The program can only jump within the same page where addr is located.

Cycles: 2
Bytes: 2
Flags: none

Example:

Command decoding by using JBb instruction.

```
      MOV A,R0        ;R0 holds a command,
                      ;if bit-n is set,
                      ;process_n must be
                      ;carried out
      JB7 process_7   ;end of process_7
                      ;will jump back to ck_6:
ck_6: JB6 process_6   ;end of process_6
                      ;will jump back to ck_5:
ck_5: .
      .
ck_0  JB0 process_0   ;all 8 bits of
                      ;the command have
                      ;been checked
```

JC addr

Jump if carry flag is set.

Instruction Format:

1	1	1	1	0	1	1	0	1st byte
addr								2nd byte

Description:

Program execution will jump to addr if the carry flag is set, otherwise it will execute the next sequential instruction. The program can only jump within the same page where addr is located.

Cycles: 2
Bytes: 2
Flags: none

JF0 addr

Jump if flag F0 is set.

Instruction Format:

1	0	1	1	0	1	1	0	1st byte
addr								2nd byte

Description:

Program execution will jump to addr if the flag F0 is set, otherwise it will execute the next sequential instruction. The program can only jump within the same page where addr is located.

Cycles: 2
Bytes: 2
Flags: none

JF1 addr

Jump if flag F1 is set.

Instruction Format:

0	1	1	1	0	1	1	0	1st byte
addr								2nd byte

Description:

Program execution will jump to addr if the flag F1 is set, otherwise it will execute the next sequential instruction. The program can only jump within the same page where addr is located.

Cycles: 2
Bytes: 2
Flags: none

JMP aaddr

Unconditional jump.

Instruction Format:

a10	a9	a8	0	0	1	0	0	1st byte
addr								2nd byte

Description:

Unconditionally transfer program execution to any location within the 2K program memory.

Cycles: 2
Bytes: 2
Flags: none

JMPP @A

Jump indirect to address pointed to by the accumulator in current page.

Instruction Format:

1	0	1	1	0	0	1	1

Description:

The lower 8 bits of the program counter is replaced by the contents of the program memory pointed to by the accumulator. Because only the lower 8 bits of the program counter are modified, the program can only jump within the current page.

Cycles: 2
Bytes: 1
Flags: none

Example:

```
ptr: DB   0f0h    ;this location holds a
                  ;jmp pointer to 0f0h
     MOV  A,#ptr  ;setup the address of
                  ;the pointer to A
     JMPP @A      ;program will jump to 0f0h
                  ;of the current page
```

JNC addr

Jump if carry flag is zero.

Instruction Format:

1	1	1	0	0	1	1	0	1st byte
addr								2nd byte

Description:

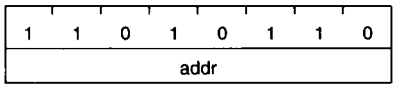
Program execution will jump to addr if the carry flag is zero, otherwise it will execute the next sequential instruction. The program can only jump within the same page where addr is located

Cycles: 2
Bytes: 2
Flags: none

JNIBF addr

Jump if input buffer (DBBIN) is empty.

Instruction Format:



Description:

Program execution will jump to addr if the IBF flag is zero, otherwise it will execute the next sequential instruction. The program can only jump within the same page where addr is located.

Cycles: 2
Bytes: 2
Flags: none

Example:

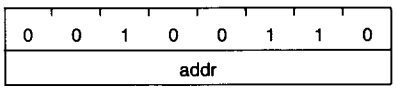
A wait-for-input loop.

```
wait: JNIBF wait    ;wait until input
                    ;buffer is full
      IN    A,DBB   ;read input buffer
```

JNT0 addr

Jump if Test 0 pin is low.

Instruction Format:



Description:

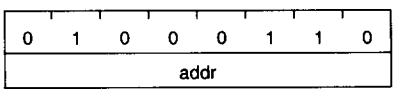
Program execution will jump to addr if the Test 0 pin is at low level, otherwise it will execute the next sequential instruction. The state of the Test 0 pin is sampled at phase 4 of the first instruction cycle. The program can only jump within the same page where addr is located.

Cycles: 2
Bytes: 2
Flags: none

JNT1 addr

Jump if Test 1 pin is low.

Instruction Format:



Description:

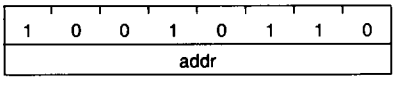
Program execution will jump to addr if the Test 1 pin is at low level, otherwise it will execute the next sequential instruction. The state of the Test 1 pin is sampled at phase 4 of the first instruction cycle. The program can only jump within the same page where addr is located.

Cycles: 2
Bytes: 2
Flags: none

JNZ addr

Jump if accumulator is non-zero.

Instruction Format:



Description:

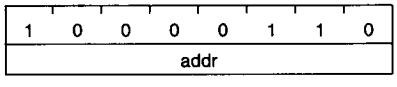
Program execution will jump to addr if the contents of the accumulator is non-zero, otherwise it will execute the next sequential instruction. The program can only jump within the same page where addr is located.

Cycles: 2
Bytes: 2
Flags: none

JOBF addr

Jump if output buffer (DBBOUT) is full.

Instruction Format:



Description:

Program execution will jump to addr if the OBF flag is set, otherwise it will execute the next sequential instruction. The program can only jump within the same page where addr is located.

Cycles: 2
Bytes: 2
Flags: none

Example:

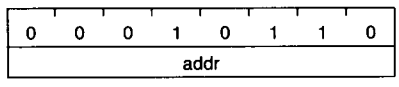
Send an array of data to the host.

```
      MOV  R0, top   ;top is the beginning
                     ;address of the data
                     ;array to be sent
                     ;to the host
      MOV  R1, cnt   ;cnt is the byte count
again: MOV A, @R0    ;get data
wait: JOBF wait      ;wait until output
                     ;buffer is empty
      OUT  DBB,A     ;output data to host
      INC  R0        ;increment data pointer
      DJNZ R1,again  ;check for end of
                     ;operation
```

JTF addr

Jump to address if the timer flag is set.

Instruction Format:



Description:

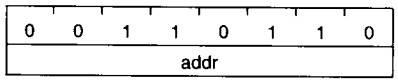
Program execution will jump to addr if the timer flag is set, otherwise it will execute the next sequential instruction. The timer flag is set when the timer/counter overflows. This instruction will reset the timer flag. The program can only jump within the same page where addr is located.

Cycles: 2
Bytes: 2
Flags: none

JT0 addr

Jump to address if Test 0 pin is high.

Instruction Format:



Description:

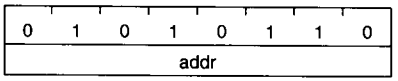
Program execution will jump to addr if the Test 0 pin is at high level, otherwise it will execute the next sequential instruction. The state of the Test 0 pin is sampled at phase 4 of the first instruction cycle. The program can only jump within the same page where addr is located.

Cycles: 2
Bytes: 2
Flags: none

JT1 addr

Jump to address if Test 1 pin is high.

Instruction Format:



Description:

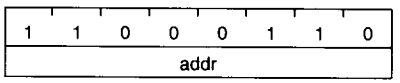
Program execution will jump to addr if the Test 1 pin is at high level, otherwise it will execute the next sequential instruction. The state of the Test 1 pin is sampled at phase 4 of the first instruction cycle. The program can only jump within the same page where addr is located.

Cycles: 2
Bytes: 2
Flags: none

JZ addr

Jump to address if accumulator is zero.

Instruction Format:



Description:

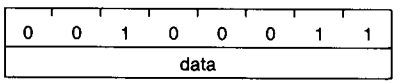
Program execution will jump to addr if the contents of the accumulator is zero, otherwise it will execute the next sequential instruction. The program can only jump within the same page where addr is located.

Cycles: 2
Bytes: 2
Flags: none

MOV A, #data

Move immediate data to the accumulator.

Instruction Format:



Description:

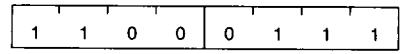
Move the immediate data to the accumulator.

Cycles: 2
Bytes: 2
Flags: none

MOV A, PSW

Move the contents of the Program Status Word to the accumulator.

Instruction Format:



Description:

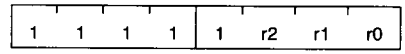
Move the contents of the Program Status Word to the accumulator.

Cycles: 1
Bytes: 1
Flags: none

MOV A, Rr

Move contents of register to the accumulator.

Instruction Format:



Description:

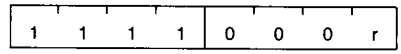
Move the contents of a register to the accumulator. Any register from R0 to R7 can be specified.

Cycles: 1
Bytes: 1
Flags: none

MOV A, @Rr

Move indirectly the contents of data memory to the accumulator.

Instruction Format:



Description:

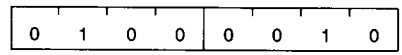
Move the contents of a data memory location to the accumulator. The address of the memory location is specified by R0 or R1.

Cycles: 1
Bytes: 1
Flags: none

MOV A, T

Move contents of the timer/counter into the accumulator.

Instruction Format:



Description:

Move the contents of the timer/counter into the accumulator. This instruction doesn't affect counting of the timer/counter.

Cycles: 1
Bytes: 1
Flags: none

MOV PSW, A

Move contents of the accumulator into the program status word.

Instruction Format:

1	1	0	1	0	1	1	1

Description:

Move contents of the accumulator into the program status word: **Note:** the stack pointer and register bank select will be altered.

Cycles: 1
Bytes: 1
Flags: C, AC, F0

MOV Rr, A

Move contents of the accumulator to a register.

Instruction Format:

1	0	1	0	1	r2	r1	r0

Description:

Move contents of the accumulator to a register. Any registers from R0 to R7 can be specified.

Cycles: 1
Bytes: 1
Flags: none

MOV Rr, #data

Move immediate data to register.

Instruction Format:

1	0	1	1	1	r2	r1	r0	1st byte
data								2nd byte

Description:

Move the immediate data to a register. Any registers from R0 through R7 can be specified.

Cycles: 2
Bytes: 2
Flags: none

MOV @Rr, A

Move contents of the accumulator indirectly into the data memory.

Instruction Format:

1	0	1	0	0	0	0	r

Description:

Move contents of the accumulator indirectly into the data memory. The address of the memory location is specified by R0 or R1.

Cycles: 1
Bytes: 1
Flags: none

MOV @Rr, #data

Move immediate data into data memory.

Instruction Format:

1	0	1	1	0	0	0	r	1st byte
data								2nd byte

Description:

Move an immediate data indirectly into the data memory. The address of the memory location is specified by R0 or R1.

Cycles: 2
Bytes: 2
Flags: none

MOV STS, A

Move contents of the accumulator to the STS register.

Instruction Format:

1	0	0	1	0	0	0	0

Description:

Move the upper nibble of the accumulator into the upper nibble of the status register. Lower nibble of the status register is unchanged.

Cycles: 1
Bytes: 1
Flags: none

MOV T, A

Move contents of the accumulator to the timer/counter.

Instruction Format:

0	1	1	0	0	0	1	0

Description:

Move contents of the accumulator into the timer/counter. This instruction doesn't affect counting of the timer/counter.

Cycles: 1
Bytes: 1
Flags: none

MOVD A, Pp

Move contents of port (4–7) into the accumulator.

Instruction Format:

0	0	0	0	1	1	p1	p0

Description:

Read the expander port (4–7) contents into accumulator bits 0–3. Accumulator bits (4–7) are reset to zero.

Cycles: 2
Bytes: 1
Flags: none

MOVD Pp, A

Move contents of the accumulator to port (4–7).

Instruction Format:

0	0	1	1	1	1	p1	p0

Description:

Move the lower nibble of the accumulator to the expander port 4–7.

Cycles: 2
Bytes: 1
Flags: none

MOVP A, @A

Move contents of program memory in the current page addressed by the contents of the accumulator into the accumulator.

Instruction Format:

1	0	1	0	0	0	1	1

Description:

Move a byte in the program memory into the accumulator. The page address is supplied by the accumulator, the upper 3 bits of the address come from the PC. Hence this instruction makes reference to the current page of the program memory. If the instruction is at the last byte of a page, the reference will be made to the next page.

Cycles: 2
Bytes: 1
Flags: none

Example:

Table lookup using MOVP A, @A

```
MOV  A,R0           ;assume R0 contains the
                    ;index of a table
ADD  A,#table_top   ;add the beginning
                    ;address of the table
MOVP A,@A           ;the data in the table
                    ;pointed to by the
                    ;index is retrieved
```

MOVP3 A, @A

Move contents of program memory in page 3 addressed by the contents of the accumulator into the accumulator.

Instruction Format:

1	1	1	0	0	0	1	1

Description:

Move a byte in the program memory into the accumulator. The page address is supplied by the accumulator, the upper 3 bits are fixed to 011b. Hence this instruction makes reference to page 3 of the program memory.

Cycles: 2
Bytes: 1
Flags: none

NOP

No operation.

Instruction Format:

0	0	0	0	0	0	0	0

Description:

No operation.

Cycles: 1
Bytes: 1
Flags: none

ORL A, #data

Logical OR immediate data to the accumulator.

Instruction Format:

0	1	0	0	0	0	1	1	1st byte
data								2nd byte

Description:

Logical OR the immediate data to the accumulator. Result is stored back to the accumulator.

Cycles: 2
Bytes: 2
Flags: none

ORL A, Rr

Logical OR contents of a register to the accumulator.

Instruction Format:

0	1	0	0	1	r2	r1	r0

Description:

Logical OR contents of a register to the accumulator. Result is stored back to the accumulator. Any registers from R0 to R7 can be specified.

Cycles: 1
Bytes: 1
Flags: none

ORL A, @Rr

Logical OR indirectly the contents of data memory to the accumulator.

Instruction Format:

0	1	0	0	0	0	0	r

Description:

Logical OR contents of a data memory location to the accumulator. The address of the memory location is specified by R0 or R1. Result is stored back to the accumulator.

Cycles: 1
Bytes: 1
Flags: none

ORL Pp, #data

Logical OR immediate data with port (1–2).

Instruction Format:

1	0	0	0	1	0	p1	p0	1st byte
data								2nd byte

Description:

Logical OR contents of the port register (NOTE: not the states of the port pins) with the immediate data. The result is stored back to the port register. The port of operation is specified by p1 and p0. Refer to Table 3-1 for port mapping.

Cycles: 2
Bytes: 2
Flags: none

ORLD Pp, A

Logical OR contents of the accumulator with port (4–7).

Instruction Format:

1	0	0	0	1	1	p1	p0

Description:

Logical OR contents of the port register of the expander port (4–7) with the lower nibble of the accumulator. The result is stored back to the port register. The port of operation is specified by p1 and p0. Refer to Table 3-1 for port mapping.

Cycles: 2
Bytes: 1
Flags: none

OUT DBB, A

Write contents of the accumulator to DBBOUT, set OBF flag.

Instruction Format:

0	0	0	0	0	0	1	0

Description:

Write contents of the accumulator to the DBBOUT register, set the OBF flag.

Cycles: 1
Bytes: 1
Flags: OBF

OUTL Pp, A

Output contents of the accumulator to port (1–2).

Instruction Format:

0	0	1	1	1	0	p1	p0

Description:

Output contents of the accumulator to the port register. The data is latched in the register.

Cycles: 2
Bytes: 1
Flags: none

RET

Return from subroutine without restoring program status word.

Instruction Format:

1	0	0	0	0	0	1	1

Description:

Restore program counter from the top of the stack. The stack pointer is decremented by 1. Then the contents of the stack pointed to by the stack pointer are loaded into the program counter. Bits 4–7 of the program status word are not affected. This instruction allows the program to return from a subroutine.

Cycles: 2
Bytes: 1
Flags: none

RETR

Return from subroutine and restore program status word.

Instruction Format:

1	0	0	1	0	0	1	1

Description:

Restore program counter and the program status word from the top of the stack. The stack pointer is decremented by 1. Then the contents of the stack pointed to by the stack pointer are loaded into the program counter and bits 4–7 of the program status word. This instruction allows the program to return from an interrupt service routine. This instruction should not be used to return from a subroutine within an interrupt routine. This is because this instruction indicates the end of an interrupt routine and re-enables interrupt.

Cycles: 2
Bytes: 1
Flags: none

RL A

Rotate the accumulator left by one bit without carry.

Instruction Format:

1	1	1	0	0	1	1	1

Description:

Contents of the accumulator are rotated left by one bit position. Bit 7 goes to bit 0 position.

Cycles: 1
Bytes: 1
Flags: none

RLC A

Rotate the accumulator left by one bit through carry.

Instruction Format:

1	1	1	1	0	1	1	1

Description:

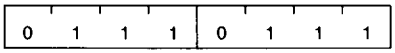
Contents of the accumulator are rotated left by one bit position. Bit 7 goes to carry and carry goes to bit 0 position.

Cycles: 1
Bytes: 1
Flags: C

RR A

Rotate the accumulator right by one bit without carry.

Instruction Format:



Description:

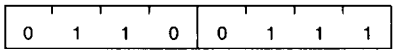
Contents of the accumulator are rotated right by one bit position. Bit 0 goes to bit 7 position.

Cycles: 1
Bytes: 1
Flags: none

RRC A

Rotate the accumulator right by one bit through carry.

Instruction Format:



Description:

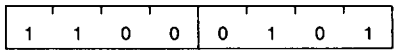
Contents of the accumulator are rotated right by one bit position. Bit 0 goes to carry and carry goes to bit 7 position.

Cycles: 1
Bytes: 1
Flags: C

SEL RB0

Select Register Bank 0 (R0–R7).

Instruction Format:



Description:

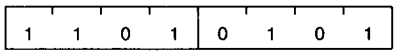
The bank select bit in the program status word (bit 4) is reset to zero. All instructions reference to register R0 to R7 address data memory locations 0h to 7h respectively.

Cycles: 1
Bytes: 1
Flags: none

SEL RB1

Select Register Bank 1 (R0'–R7').

Instruction Format:



Description:

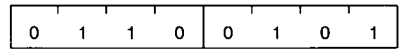
The bank select bit in the program status word (bit 4) is set to one. All instructions reference to register R0 to R7 address data memory locations 18h to 1Fh respectively.

Cycles: 1
Bytes: 1
Flags: none

STOP TCNT

Stop timer/counter.

Instruction Format:



Description:

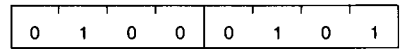
Stop counting of the timer/counter, regardless of the mode of operation.

Cycles: 1
Bytes: 1
Flags: none

STRT CNT

Start event counter.

Instruction Format:



Description:

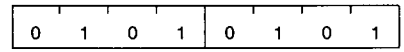
Start counting of the timer/counter in the event counter mode. Test 1 pin is used as the input of the event counter. The counter is incremented at the falling edge of the input signal.

Cycles: 1
Bytes: 1
Flags: none

STRT T

Start timer.

Instruction Format:



Description:

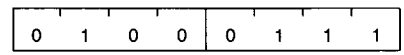
Start counting of the timer/counter in the timer mode. The counter is incremented by 1 at every 32 instruction cycles. The divide-by-32 prescaler is cleared by this instruction.

Cycles: 1
Bytes: 1
Flags: none

SWAP A

Swap the lower and upper nibbles of the accumulator.

Instruction Format:



Description:

Accumulator bits 0 through 3 are swapped with accumulator bits 4 through 7.

Cycles: 1
Bytes: 1
Flags: none

XCH A, Rr

Exchange the accumulator and the register's contents.

Instruction Format:

0	0	1	0	1	r2	r1	r0

Description:

Exchange the contents of the accumulator and a register. R0 to R7 can be specified.

Cycles: 1
Bytes: 1
Flags: none

XCH A, @Rr

Exchange indirectly contents of the accumulator and data memory.

Instruction Format:

0	0	1	0	0	0	0	r

Description:

Exchange contents of the accumulator and a data memory. The location of the data memory is pointed to by R0 or R1.

Cycles: 1
Bytes: 1
Flags: none

XCHD A, @Rr

Exchange indirectly lower nibbles of the accumulator and data memory.

Instruction Format:

0	0	1	1	0	0	0	r

Description:

Exchange only the lower nibble of the accumulator and a data memory. The location of the data memory is pointed to by R0 or R1. The upper nibble of the accumulator and the data memory remain unchanged.

Cycles: 1
Bytes: 1
Flags: none

XRL A, #data

Logical exclusive OR immediate data to the accumulator.

Instruction Format:

1	1	0	1	0	0	1	1	1st byte
data								2nd byte

Description:

Logical exclusive OR the immediate data to the accumulator. Result is stored back to the accumulator.

Cycles: 2
Bytes: 2
Flags: none

XRL A, Rr

Logical exclusive OR contents of register to the accumulator.

Instruction Format:

1	1	0	1	1	r2	r1	r0

Description:

Logical exclusive OR the contents of a register to the accumulator. Result is stored back to the accumulator. Any registers from R0 to R7 can be specified.

Cycles: 1
Bytes: 1
Flags: none

XRL A, @Rr

Logical exclusive OR indirectly contents of data memory to the accumulator.

Instruction Format:

1	1	0	1	0	0	0	r

Description:

Logical exclusive OR the contents of a data memory location to the accumulator. The address of the memory location is specified by R0 or R1. Result is stored back to the accumulator.

Cycles: 1
Bytes: 1
Flags: none

3.0 Programming of the Keyboard Controller (Continued)

3.5 PC87911 KEYBOARD CONTROLLER INSTRUCTION SET SUMMARY

TABLE 3-2. PC87911 Keyboard Controller Instruction Set Summary

Mnemonic	Function	Description	Cycles	Bytes	Flags			
					C	AC	F0	F1
ARITHMETIC AND LOGIC FUNCTIONS								
ADD A, #data	(A) ← (A) + Data	Add Immediate Data to the Accumulator.	2	2	•	•		
ADD A, Rr	(A) ← (A) + (Rr) for r = 0–7	Add Contents of Register to Accumulator	1	1	•	•		
ADD A, @Rr	(A) ← (A) + ((Rr)) for r = 0–1	Add Indirect the Contents of Data Memory to the Accumulator.	1	1	•	•		
ADDC A, #data	(A) ← (A) + (C) + Data	Add Immediate Data with Carry to the Accumulator.	2	2	•	•		
ADDC A, Rr	(A) ← (A) + (C) + (Rr) for r = 0–7	Add with Carry the Contents of Register to the Accumulator.	1	1	•	•		
ADDC A,@Rr	(A) ← (A) + (C) + ((Rr)) for r = 0–1	Add Indirect with Carry the Contents of Data Memory to the Accumulator.	1	1	•	•		
ANL A, #data	(A) ← (A) AND Data	Logic AND Immediate Data with Accumulator.	2	2				
ANL A, Rr	(A) ← (A) AND (Rr) for r = 0–7	Logic AND Contents of Register with Accumulator	1	1				
ANL A, @Rr	(A) ← (A) AND ((Rr)) for r = 0–7	Logic AND Indirect the Contents of Data Memory with Accumulator.	1	1				
CLR A	(A) ← 0	Clear the Contents of the Accumulator.	1	1				
CPL A	(A) ← NOT (A)	Complement the Contents of the Accumulator.	1	1				
DA A		Decimal Adjust the Contents of the Accumulator.	1	1	•			
DEC A	(A) ← (A) − 1	Decrement the Accumulator's Contents by One.	1	1				
INC A	(A) ← (A) + 1	Increment the Accumulator's Contents by One.	1	1				
ORL A, #data	(A) ← (A) OR Data	Logical OR Immediate Data with Accumulator.	2	2				
ORL A, Rr	(A) ← (A) OR (Rr) for r = 0–7	Logical OR Contents of Register with Accumulator.	1	1				
ORL A, @ Rr	(A) ← (A) OR ((Rr)) for r = 0–1	Logical OR Indirect the Contents of Data Memory with Accumulator.	1	1				
RL A	(An + 1) ← (An) for n = 0–6 (A0) ← (A7)	Rotate Accumulator Left by 1 Bit without Carry.	1	1				
RLC A	(An + 1) ← (An); n = 0–6 (A0) ← (C) (C) ← (A7)	Rotate Accumulator Left by 1 Bit through Carry.	1	1	•			
RR A	(An) ← (An + 1); n = 0–6 (A7) ← (A0)	Rotate Accumulator Right by 1 Bit without Carry.	1	1				
RRC A	(An) ← (An + 1); n = 0–6 (A7) ← (C) (C) ← (A0)	Rotate Accumulator Right by 1 Bit through Carry.	1	1	•			
SWAP A	(A4–A7) ⟷ (A0–A3)	Swap the 2, 4-Bit Nibbles in the Accumulator.	1	1				
XRL A, #data	(A) ← (A) XOR Data	Logical Exclusive OR Immediate Data with Accumulator.	2	2				
XRL A, Rr	(A) ← (A) XOR (Rr) for r = 0–7	Logical Exclusive OR Contents of Register with Accumulator.	1	1				
XRL A, @Rr	(A) ← (A) XOR ((Rr)) for r = 0–1	Logical Exclusive OR Indirect the Contents of Data Memory with Accumulator.	1	1				

3.0 Programming of the Keyboard Controller (Continued)

KEYBOARD CONTROLLER INSTRUCTION SET SUMMARY

TABLE 3-2. PC87911 Keyboard Controller Instruction Set Summary (Continued)

Mnemonic	Function	Description	Cycles	Bytes	Flags			
					C	AC	F0	F1
JUMP								
DJNZ Rr, addr	(Rr) ← (Rr) − 1; r = 0–7 (PC 0–7) ← addr if (Rr) ≠ 0 (PC) ← (PC) + 2 if (Rr) = 0	Decrement Register and Branch if it is not Zero.	2	2				
JBb addr	(PC 0–7) ← addr if Bb = 1 (PC) ← (PC) + if Bb = 0	Jump if Accumulator Bit is Set.	2	2				
JC addr	(PC 0–7) ← addr if C = 1 (PC) ← (PC) + 2 if C = 0	Jump if Carry Flag is Set.	2	2				
JF0 addr	(PC 0–7) ← addr if F0 = 1 (PC) ← (PC) + 2 if F0 = 0	Jump if Flag F0 is Set.	2	2				
JF1 addr	(PC 0–7) ← addr if F1 = 1 (PC) ← (PC) + 2 if F1 = 0	Jump if Flag F1 is Set.	2	2				
JMP aaddr	(PC 8–10) ← aaddr 8–10 (PC 0–7) ← aaddr 0–7	Unconditional Jump.	2	2				
JMPP @A	(PC 0–7) ← ((A))	Jump Indirect to Address Pointed to by the Accumulator in Current Page.	2	1				
JNC addr	(PC 0–7) ← addr if C = 0 (PC) ← (PC) + 2 if C = 1	Jump if Carry Flag is Zero.	2	2				
JNIBF addr	(PC 0–7) ← addr if IBF = 0 (PC) ← (PC) + 2 if IBF = 1	Jump if Input Buffer (DBBIN) is Empty.	2	2				
JNT0 addr	(PC 0–7) ← addr if T0 = 0 (PC) ← (PC) + 2 if T0 = 1	Jump if Test 0 is Low.	2	2				
JNT1 addr	(PC 0–7) ← addr if T1 = 0 (PC) ← (PC) + 2 if T1 = 1	Jump if Test 1 is Low.	2	2				
JNZ addr	(PC 0–7) ← addr if A ≠ 0 (PC) ← (PC) + 2 if A = 0	Jump if Accumulator is Non-Zero.	2	2				
JOBF addr	(PC 0–7) ← addr if OBF = 1 (PC) ← (PC) + 2 if OBF = 0	Jump if Output Buffer (DBBOUT) is Full.	2	2				
JTF addr	(PC 0–7) ← addr if TF = 1 (PC) ← (PC) + 2 if TF = 0	Jump if Timer Flag is Set to 1	2	2				
JT0 addr	(PC 0–7) ← addr if T0 = 1 (PC) ← (PC) + 2 if T0 = 0	Jump if Test 0 is High.	2	2				
JT1 addr	(PC 0–7) ← addr if T1 = 1 (PC) ← (PC) + 2 if T1 = 0	Jump if Test 1 is High.	2	2				
JZ addr	(PC 0–7) ← addr if A = 0 (PC) ← (PC) + 2 if A = 1	Jump if Accumulator is 0.	2	2				
CONTROL								
EN DMA		Enable DRQ, $\overline{\text{DACK}}$ lines (P26, P27).	1	1				
EN FLAGS		Enable OBF, $\overline{\text{IBF}}$ Lines (P24, P25).	1	1				
EN I		Enable the IBF Interrupt.	1	1				
DIS I		Disable the IBF Interrupt.	1	1				
NOP		No Operation.	1	1				
SEL RB0	(BS) ← 0	Select Register Bank 0 (R0–R7).	1	1				
SEL RB1	(BS) ← 1	Select Register Bank 1 (R0′–R7′).	1	1				

TABLE 3-2. PC87911 Keyboard Controller Instruction Set Summary (Continued)

Mnemonic	Function	Description	Cycles	Bytes	Flags			
					C	AC	F0	F1
DATA MOVES								
MOV A, #data	(A) ← data	Move Immediate Data into the Accumulator.	2	2				
MOV A, PSW	(A) ← (PSW)	Move Contents of the Program Status Word into the Accumulator.	1	1				
MOV A, Rr	(A) ← (Rr); r = 0–7	Move the Contents of Register into the Accumulator	1	1				
MOV A, @Rr	(A) ← ((Rr)); r = 0–1	Move Indirect the Contents of Data Memory into the Accumulator.	1	1				
MOV PSW, A	(PSW) ← (A)	Move Contents of Accumulator into the Program Status Word.	1	1	•	•	•	
MOV Rr, A	(Rr) ← (A); r = 0–7	Move Accumulator Contents into the Register.	1	1				
MOV Rr, #data	(Rr) ← data; r = 0–7	Move Immediate Data into Register.	2	2				
MOV @Rr, A	((Rr)) ← (A); r = 0–1	Move Indirect Accumulator Contents into Data Memory.	1	1				
MOV @Rr, #data	((Rr)) ← data; r = 0–1	Move Immediate Data into Data Memory.	2	2				
MOVP A, @A	(PC 0–7) ← (A) (A) ← ((PC))	Move Contents of Program Memory in the Current Page Addressed by the Content of Accumulator into the Accumulator.	2	1				
MOVP3 A, @A	(PC 0–7) ← (A) (PC 8–10) ← 011 (A) ← ((PC))	Move Contents of Program Memory Location in Page 3 Address by the Content of Accumulator into the Accumulator.	2	1				
XCH A, Rr	(A) ⟷ (Rr); r = 0–7	Exchange the Accumulator and Register's Contents.	1	1				
XCH A, @Rr	(A) ⟷ ((Rr)); r = 0–1	Exchange Indirect Contents of Accumulator and Data Memory.	1	1				
XCHD A, @Rr	(A0–A3) ⟷ (((Rr)) 0–3); r = 0–1	Exchange Indirect 4-Bit Contents of Accumulator and Data Memory.	1	1				
TIMER COUNTER								
EN TCNTI		Enable Interrupt Flag for Timer/Counter Overflow.	1	1				
DIS TCNTI		Disable Interrupt Flag for Timer/Counter Overflow.	1	1				
MOV A, T	(A) ← (T)	Move Contents of Timer/Counter into Accumulator.	1	1				
MOV T, A	(T) ← (A)	Move Contents of Accumulator into Timer/Counter.	1	1				
STOP TCNT		Stop Timer/Counter.	1	1				
STRT CNT		Start Event Counter.	1	1				
STRT T		Start Timer.	1	1				

TABLE 3-2. PC87911 Keyboard Controller Instruction Set Summary (Continued)

Mnemonic	Function	Description	Cycles	Bytes	Flags			
					C	AC	F0	F1
INPUT/OUTPUT								
ANL Pp, #data	(Pp) ← (Pp) AND data; p = 1–2	Logical AND Immediate Data with Port (1 or 2).	2	2				
ANLD Pp, A	(Pp) ← (Pp) AND (A0–A3); p = 4–7	Logical AND Contents of Accumulator with Port (4–7).	2	1				
IN A, DBB	(A) ← (DBBIN)	Read DBBIN to Accumulator, Clear IBF flag.	1	1				
IN A, Pp	(A) ← (Pp); p = 1–2	Input Data from Port (1–2) into Accumulator.	2	1				
MOV STS, A	(STS4–STS7) ← (A4–A7)	Move Contents of Accumulator to STS Register.	1	1				
MOVD A, Pp	(A0–A3) ← (Pp); (A4–A7) ← 0; p = 4–7	Move Contents of Port (4–7) into Accumulator.	2	1				
MOVD Pp, A	(Pp) ← (A0–A3); p = 4–7	Move Contents of Accumulator to Port (4–7).	2	1				
ORL Pp, #data	(Pp) ← (Pp) OR data; p= 1–2	Logical OR Immediate Data with Port (1–2).	2	2				
ORLD Pp, A	(Pp) ← (Pp) OR (A0–A3); p = 4–7	Logical OR Contents of Accumulator with Port (4–7).	2	1				
OUT DBB, A	(DBBOUT) ← (A)	Write Contents of Accumulator to to DBBOUT, Set OBF Flag.	1	1				
OUTL Pp, A	(Pp) ← (A); p = 1–2	Output Contents of Accumulator to Port (1–2).	1	1				
REGISTERS								
DEC Rr	(Rr) ← (Rr) − 1; r = 0–7	Decrement Register by One.	1	1				
INC Rr	(Rr) ← (Rr) + 1; r = 0–7	Increment Register by One.	1	1				
INC @Rr	((Rr)) ← ((Rr)) + 1; r = 0–1	Increment Indirect Data Memory Location by One.	1	1				
SUBROUTINE								
CALL aaddr	((SP)) ← (PC) ((SP)) ← (PSW 4–7) (SP) ← (SP) + 1 (PC 8–10) ← aaddr 8–10 (PC 0–7) ← aaddr 0-7	Call Subroutine.	2	2				
RET	(SP) ← (SP) − 1 (PC) ← ((SP))	Return from Subroutine without Restoring Program Status Word.	2	1				
RETR	(SP) ← (SP) − 1 (PC) ← ((SP)) (PSW 4–7) ← ((SP))	Return from Subroutine and Restore Program Status Word.	2	1	•	•		
FLAGS								
CLR C	(C) ← 0	Clear Carry Flag to 0.	1	1	•			
CLR F0	(F0) ← 0	Clear Flag 0 to 0	1	1			•	
CLR F1	(F1) ← 0	Clear Flag 1 to 0.	1	1				•
CPL C	(C) ← NOT (C)	Complement Carry Flag.	1	1	•			
CPL F0	(F0) ← NOT (F0)	Complement Flag F0.	1	1			•	
CPL F1	(F1) ← NOT (F1)	Complement Flag F1.	1	1				•

3.0 Programming of the Keyboard Controller (Continued)

3.6 PC87911 KEYBOARD CONTROLLER INSTRUCTION OPCODE SUMMARY

TABLE 3-3. PC87911 Keyboard Controller Instruction Opcode Summary

Mnemonic	1st Byte								2nd Byte	Hex	
ARITHMETIC AND LOGIC FUNCTIONS											
ADD A, #data	0	0	0	0	0	0	1	1	data	03	data
ADD A, Rr	0	1	1	0	1	r2	r1	r0		68–6F	
ADD A, @Rr	0	1	1	0	0	0	0	r		60–61	
ADDC A, #data	0	0	0	1	0	0	1	1	data	13	data
ADDC A, Rr	0	1	1	1	1	r2	r1	r0		78–7F	
ADDC A, @Rr	0	1	1	1	0	0	0	r		70–71	
ANL A, #data	0	1	0	1	0	0	1	1	data	53	data
ANL A, Rr	0	1	0	1	1	r2	r1	r0		58–5F	
ANL A, @Rr	0	1	0	1	0	0	0	r		50–51	
CLR A	0	0	1	0	0	1	1	1		27	
CPL A	0	0	1	1	0	1	1	1		37	
DA A	0	1	0	1	0	1	1	1		57	
DEC A	0	0	0	0	0	1	1	1		07	
INC A	0	0	0	1	0	1	1	1		17	
ORL A, #data	0	1	0	0	0	0	1	1	data	43	data
ORL A, Rr	0	1	0	0	1	r2	r1	r0		48–4F	
ORL A, @Rr	0	1	0	0	0	0	0	r		40–41	
RL A	1	1	1	0	0	1	1	1		E7	
RLC A	1	1	1	1	0	1	1	1		F7	
RR A	0	1	1	1	0	1	1	1		77	
RRC A	0	1	1	0	0	1	1	1		67	
SWAP A	0	1	0	0	0	1	1	1		47	
XRL A, #data	1	1	0	1	0	0	1	1	data	D3	data
XRL A, Rr	1	1	0	1	1	r2	r1	r0		D8–DF	
XRL A, @Rr	1	1	0	1	0	0	0	r		D0–D1	
JUMP											
DJNZ Rr, addr	1	1	1	0	1	r2	r1	r0	addr	E8–EF	addr
JBb addr	b2	b1	b0	1	0	0	1	0	addr	12–F2	addr
JC addr	1	1	1	1	0	1	1	0	addr	F6	addr
JF0 addr	1	0	1	1	0	1	1	0	addr	B6	addr
JF1 addr	0	1	1	1	0	1	1	0	addr	76	addr
JMP aaddr	a10	a9	a8	0	0	1	0	0	addr	04–E4	addr
JMPP @A	1	0	1	1	0	0	1	1		B3	
JNC addr	1	1	1	0	0	1	1	0	addr	E6	addr
JNIBF addr	1	1	0	1	0	1	1	0	addr	D6	addr
JNT0 addr	0	0	1	0	0	1	1	0	addr	26	addr
JNT1 addr	0	1	0	0	0	1	1	0	addr	46	addr
JNZ addr	1	0	0	1	0	1	1	0	addr	96	addr
JOBF addr	1	0	0	0	0	1	1	0	addr	86	addr
JTF addr	0	0	0	1	0	1	1	0	addr	16	addr
JT0 addr	0	0	1	1	0	1	1	0	addr	36	addr
JT1 addr	0	1	0	1	0	1	1	0	addr	56	addr
JZ addr	1	1	0	0	0	1	1	0	addr	C6	addr
CONTROL											
EN DMA	1	1	1	0	0	1	0	1		E5	
EN FLAGS	1	1	1	1	0	1	0	1		F5	
EN I	0	0	0	0	0	1	0	1		05	
DIS I	0	0	0	1	0	1	0	1		15	
NOP	0	0	0	0	0	0	0	0		00	
SEL RB0	1	1	0	0	0	1	0	1		C5	
SEL RB1	1	1	0	1	0	1	0	1		D5	

3.0 Programming of the Keyboard Controller (Continued)

TABLE 3-3. PC87911 Keyboard Controller Instruction Opcode Summary (Continued)

Mnemonic	1st Byte								2nd Byte	Hex	
DATA MOVES											
MOV A, #data	0	0	1	0	0	0	1	1	data	23	data
MOV A, PSW	1	1	0	0	0	1	1	1		C7	
MOV A, Rr	1	1	1	1	1	r2	r1	r0		F8–FF	
MOV A, @Rr	1	1	1	1	0	0	0	r		F0–F1	
MOV PSW, A	1	1	0	1	0	1	1	1		D7	
MOVRr, A	1	0	1	0	1	r2	r1	r0		A8–AF	
MOV Rr, #data	1	0	1	1	1	r2	r1	r0	data	B8–BF	data
MOV @Rr, A	1	0	1	0	0	0	0	r		A0–A1	
MOV @Rr, #data	1	0	1	1	0	0	0	r	data	B0–B1	data
MOVP A, @A	1	0	1	0	0	0	1	1		A3	
MOVP3 A, @A	1	1	1	0	0	0	1	1		E3	
XCH A, Rr	0	0	1	0	1	r2	r1	r0		28–2F	
XCH A, @Rr	0	0	1	0	0	0	0	r		20–21	
XCHD A, @Rr	0	0	1	1	0	0	0	r		30–31	
TIMER COUNTER											
EN TCNTI	0	0	1	0	0	1	0	1		25	
DIS TCNTI	0	0	1	1	0	1	0	1		35	
MOV A, T	0	1	0	0	0	0	1	0		42	
MOV T, A	0	1	1	0	0	0	1	0		62	
STOP TCNT	0	1	1	0	0	1	0	1		65	
STRT CNT	0	1	0	0	0	1	0	1		45	
STRT T	0	1	0	1	0	1	0	1		55	
INPUT/OUTPUT											
ANL Pp, #data	1	0	0	1	1	0	p1	p0	data	98–9B	data
ANLD Pp, A	1	0	0	1	1	1	p1	p0		9C–9F	
IN A, DBB	0	0	1	0	0	0	1	0		22	
IN A, Pp	0	0	0	0	1	0	p1	p0		08–0B	
MOV STS, A	1	0	0	1	0	0	0	0		90	
MOVD A, Pp	0	0	0	0	1	1	p1	p0		0C–0F	
MOVD Pp, A	0	0	1	1	1	1	p1	p0		3C–3F	
ORLD Pp, A	1	0	0	0	1	1	p1	p0		8C–8F	
ORL Pp, #data	1	0	0	0	1	0	p1	p0	data	88–8B	data
OUT DBB, A	0	0	0	0	0	0	1	0		02	
OUTL Pp, A	0	0	1	1	1	0	p1	p0		38–3B	
REGISTERS											
DEC Rr	1	1	0	0	1	r2	r1	r0		C8–CF	
INC Rr	0	0	0	1	1	r2	r1	r0		18–1F	
INC @Rr	0	0	0	1	0	0	0	r		10–11	
SUBROUTINE											
CALL aaddr	a10	a9	a8	1	0	1	0	0	addr	14–F4	addr
RFT	1	0	0	0	0	0	1	1		83	
RETR	1	0	0	1	0	0	1	1		93	
FLAGS											
CLR C	1	0	0	1	0	1	1	1		97	
CLR F0	1	0	0	0	0	1	0	1		85	
CLR F1	1	0	1	0	0	1	0	1		A5	
CPL C	1	0	1	0	0	1	1	1		A7	
CPL F0	1	0	0	1	0	1	0	1		95	
CPL F1	1	0	1	1	0	1	0	1		B5	

4.0 Device Specifications

4.1 ABSOLUTE MAXIMUM RATINGS (Notes 1 and 2)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	−0.5V to +7.0V
DC Input Voltage (V_{IN}) or DC Input Diode Current	−0.5V to V_{CC} + 0.5V ±20 mA
DC Output Voltage (V_{OUT}) or DC Output Current, per Pin (I_{OUT})	−0.5V to V_{CC} + 0.5V ±20 mA
DC V_{CC} or V_{SS} Current, per Pin	±50 mA
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (P_D)	500 mW
Lead Temperature (Soldering, 10 sec.)	260°C
ESP Tolerance: C_{ZAP} = 120 pF, R_{ZAP} = 1500Ω	2.0 kV

4.2 OPERATING CONDITIONS

	Min	Max	Units
Supply Voltage (V_{CC})	4.5	5.5	V
DC Input or Output Voltage (V_{IN}, V_{OUT})	0.0	V_{CC}	V
Operating Temp. Range (T_A)	0	70	°C
Input Rise and Fall Times (t_r, t_f)		500	ns

4.3 DC ELECTRICAL CHARACTERISTICS V_{CC} = 5V ± 10% (Unless otherwise specified)

Symbol	Parameter	Conditions	Guaranteed Limits 0°C–70°C	Units
V_{IH}	Minimum High Level Input Voltage	All Inputs except X1C, X1K	2.0	V
V_{IHX}	Minimum High Level Input Voltage	X1C, X1K	3.5	V
V_{IL}	Maximum Low Level Input Voltage	All Inputs except X1C, X1K	0.8	V
V_{ILX}	Maximum Low Level Input Voltage	X1C, X1K	1.7	V
V_{OH}	Minimum High Level Output Voltage	V_{IN} = V_{IH} or V_{IL}		
	KBCLK, KBDAT, MCLK, MDAT (Note 3)	$\|I_{OUT}\|$ = 90.0 μA	2.4	V
	P10–P17, P20–P27	$\|I_{OUT}\|$ = 2.0 mA	2.4	V
	All Other Outputs	$\|I_{OUT}\|$ = 4.0 mA	2.4	V
V_{OL}	Maximum Low Level Output Voltage	V_{IN} = V_{IH} or V_{IL}		
	P10–P17, P20–P27	$\|I_{OUT}\|$ = 2.0 mA	0.4	V
	KBDAT, KBCLK, MCLK, MDAT	$\|I_{OUT}\|$ = 16.0 mA	0.4	V
	All Other Outputs	$\|I_{OUT}\|$ = 4.0 mA	0.4	V
I_{IN}	Maximum Input Current			
	Inputs Only: A0–A9, AEN, $\overline{CS}$, EA, $\overline{IOR}$, $\overline{IOW}$, PWRGOOD, $\overline{RESETC}$, $\overline{RESETK}$, T0, T1	V_{IN} = V_{CC} or V_{SS}	10	μA
	Inputs with Resistor:			
	$\overline{SS}$, TRI	V_{IN} = V_{CC}	10	μA
		V_{IN} = V_{IL}	300	μA
	I/Os Only: D0–D7	V_{IN} = V_{CC} or V_{SS}	10	μA
	I/Os with Resistors:			
	P10–P17, P20–P27, STROBE, SYNC	V_{IN} = V_{CC}	10	μA
		V_{IN} = V_{IL}	300	μA
I_{OZ}	Maximum TRI-STATE Output Leakage Current (Note 4)		10	μA
IBAT	V_{BAT} Quiescent Supply Current	V_{BAT} = 3.0V V_{IN} = V_{CC} = V_{SS} = 0V Real-Time Clock f = 32.768 kHz	8	μA
	Input Leakage	V_{CC} = 5.0V V_{BAT} = 3.0V	0.4	μA
$V_{BAT(min)}$	Minimum Battery Voltage		2.4	V
$V_{BAT(max)}$	Maximum Battery Voltage		V_{CC} − 0.5	V
I_{CC}	Maximum Operating Supply Current Total to 4 V_{CC} Pins (Note 5)	V_{IN} = V_{CC} or V_{SS} Keyboard Controller f = 12 MHz Real-Time Clock f = 32.768 kHz	40	mA

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified, all voltages are referenced to ground.

Note 3: KBCLK, KBDAT, MCLK and MDAT are open drain outputs with 10 kΩ minimum pull-up resistors.

Note 4: The PC87911 has no TRI-STATE output only buffers. The maximum leakage current for TRI-STATE I/Os is the same as their Maximum Input Current. Refer to the I_{IN} specification.

Note 5: No DC loading.

4.0 Device Specifications (Continued)

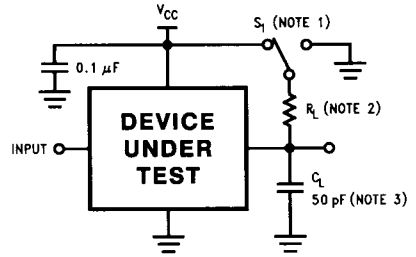
A.C. Test Conditions

Note 1: S_1 = V_{CC} for t_{PZL} and t_{PLZ} measurements
S_1 = GND for t_{PZH} and t_{PHZ} measurements
S_1 = Open for push-pull outputs

Note 2: R_L = 1.1k

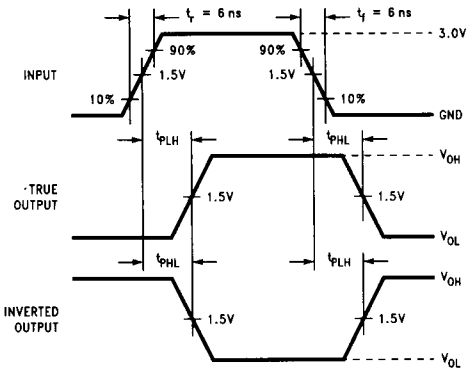
Note 3: C_L includes scope and jig capacitance

Test Circuit for Output Tests



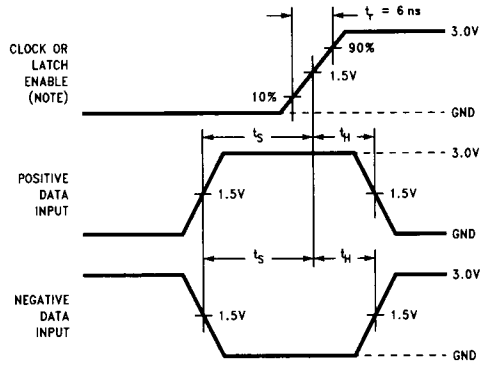
TL/C/11470-23

Propagation Delay Waveforms



TL/C/11470-24

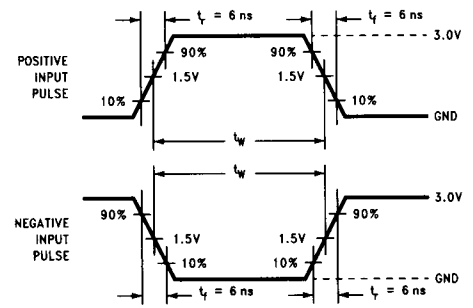
Setup and Hold Time Waveforms



TL/C/11470-25

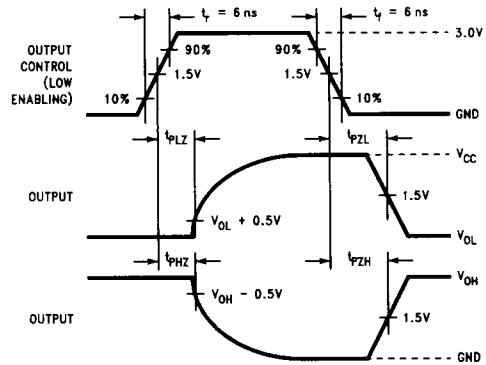
Note: Waveform for negative edge sensitive circuits will be inverted.

Input Pulse Width Waveforms Except for Clock Pins



TL/C/11470-26

TRI-STATE Output Enable and Disable Waveforms



TL/C/11470-27

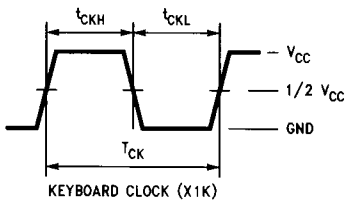
FIGURE 4-1. Switching Characteristic Measurement Waveforms

4.0 Device Specifications (Continued)

4.4 AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{DD} = +5V \pm 10\%$

Symbol	Parameter	Test Conditions	Min	Max	Unit
EXTERNAL CLOCK					
t_{CKH}	KBC Clock High Time		30		ns
t_{CKL}	KBC Clock Low Time		30		ns
T_{CK}	KBC Clock Cycle Time		63		ns
SYSTEM INTERFACE					
t_{AR}	$\overline{CS}$, AEN, Address to $\overline{IOR}$ Setup Time		10		ns
t_{RA}	$\overline{CS}$, AEN, Address to $\overline{IOR}$ Hold Time		10		ns
t_{RL}	$\overline{IOR}$ Low Time	Except Read from RAM	50		ns
t_{RLR}	$\overline{IOR}$ Low Time	Read from RAM	105		ns
t_{RH}	$\overline{IOR}$ High Time		25		ns
t_{RD}	Delay from $\overline{IOR}$ to Data	Except Read from RAM		50	ns
t_{RDR}	Delay from $\overline{IOR}$ to Data	Read from RAM		105	ns
t_{DHR}	Data Hold Time after $\overline{IOR}$		5		ns
t_{DZ}	Data Turn Off Time after $\overline{IOR}$			25	ns
t_{AW}	$\overline{CS}$, AEN, Address to $\overline{IOW}$ Setup Time		10		ns
t_{WA}	$\overline{CS}$, AEN, Address to $\overline{IOW}$ Hold Time		10		ns
t_{WL}	$\overline{IOW}$ Low Time		80		ns
t_{WH}	$\overline{IOW}$ High Time		25		ns
t_{DS}	Data Setup Time from $\overline{IOW}$		45		ns
t_{DHW}	Data Hold Time from $\overline{IOW}$			0	ns
t_{WR}	$\overline{IOR}$ Low after $\overline{IOW}$ High		80		ns
RTC					
t_{RDI}	$\overline{IOR}$ to $\overline{IRQ}$ TRI-STATE			36	ns
t_{RCI}	$\overline{RESETC}$ to $\overline{IRQ}$ TRI-STATE			25	ns
t_{RCL}	$\overline{RESETC}$ Low Time		100		ns
t_{VPG}	V_{CC} (4.5V) to PWRGOOD		0		ns
t_{PVG}	PWRGOOD to V_{CC} (4.5V)		0		ns

4.4.1 External Clock

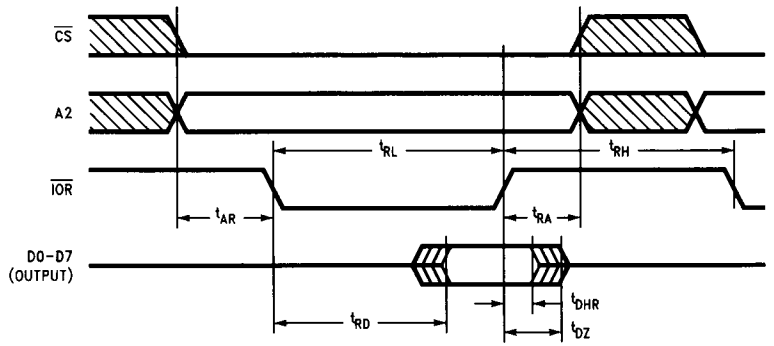


TL/C/11470–28

4.0 Device Specifications (Continued)

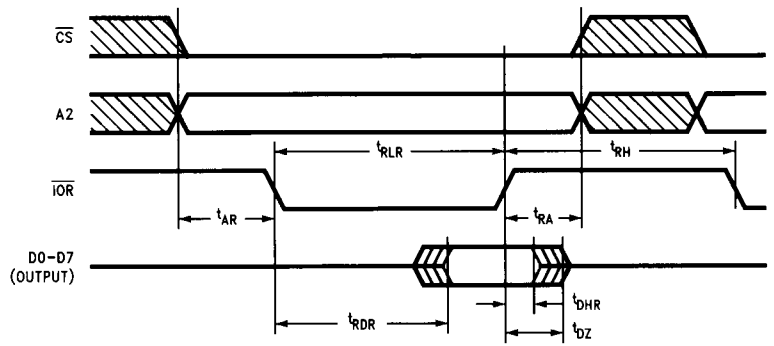
4.4.2 System Interface

4.4.2.1 Read PC, Status, DBBOUT



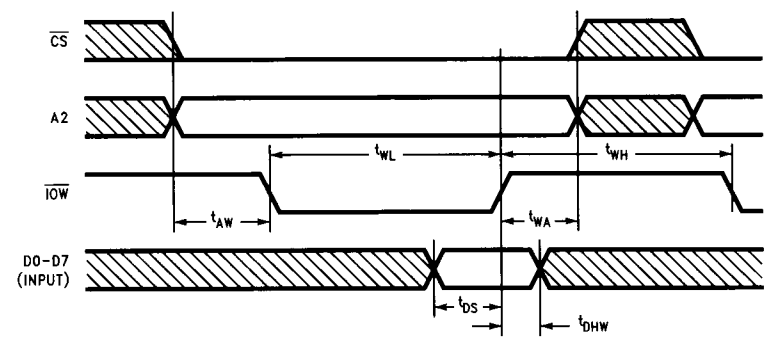
TL/C/11470-29

4.4.2.2 Read Program RAM



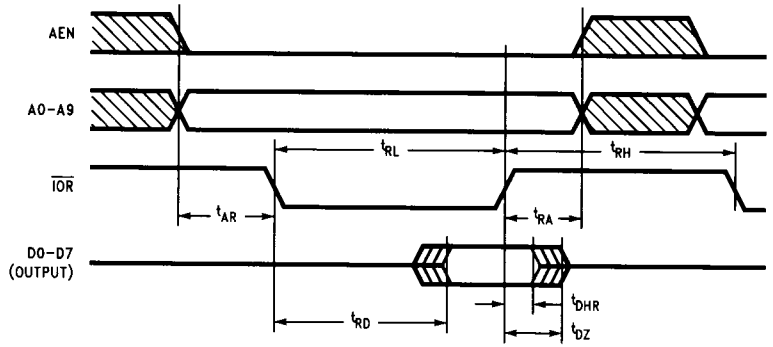
TL/C/11470-30

4.4.2.3 Write PC, Program RAM, DBBIN



TL/C/11470-31

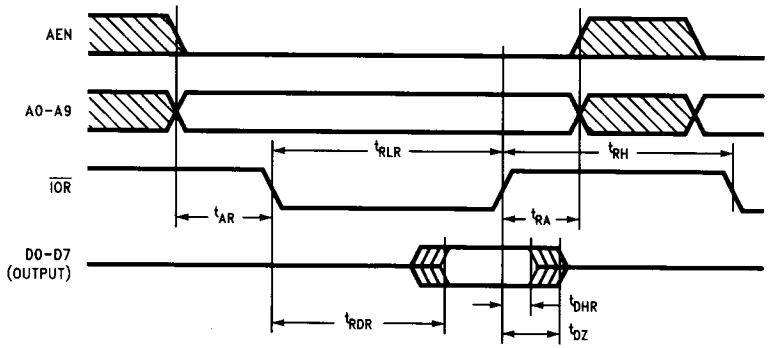
4.4.2.4 Read Configuration Register, Index, Data (RTC) (Note 1)



TL/C/11470–32

Note 1: Address locations 0h–0Fh, lower bank.

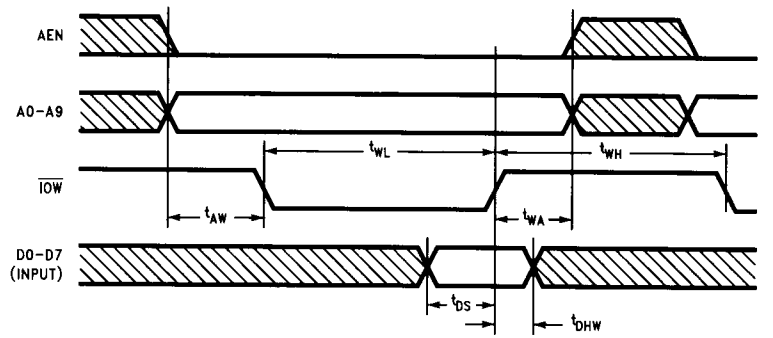
4.4.2.5 Read Data (RTC) (Note 2)



TL/C/11470–33

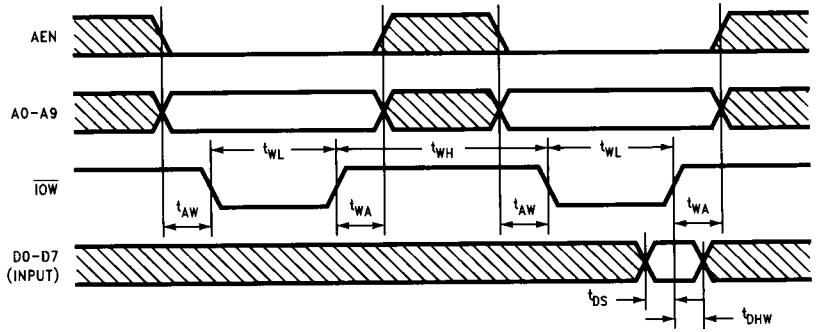
Note 2: Address locations 10h–3Fh, lower bank; 0h–3Fh, upper bank.

4.4.2.6 Write Index, Data (RTC)



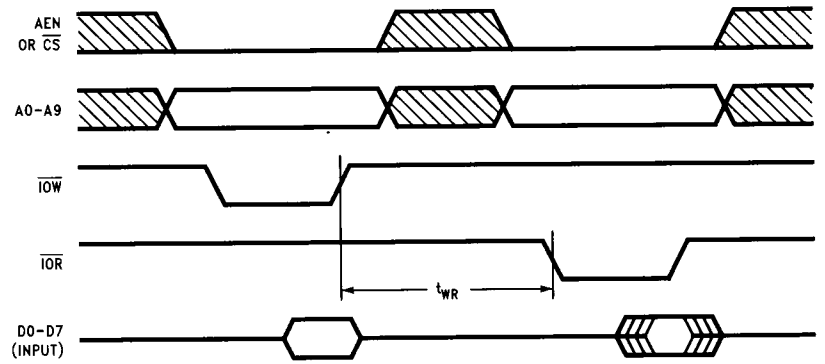
TL/C/11470–34

4.4.2.7 Write Configuration Register



TL/C/11470-35

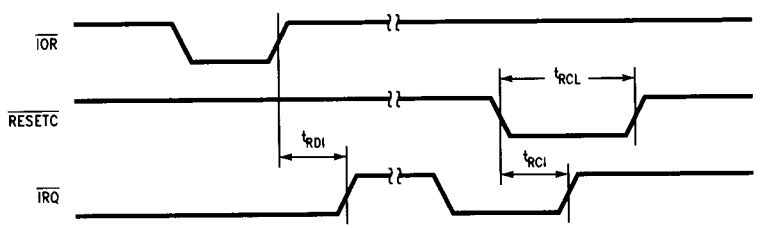
4.4.2.8 Read after Write Operation to All Registers and RAM



TL/C/11470-36

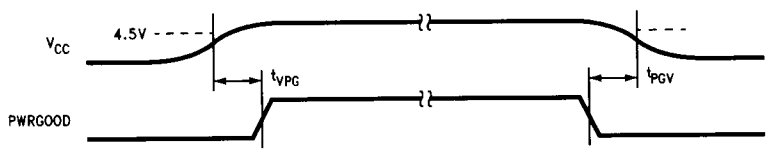
4.4.3 RTC

4.4.3.1 IRQ Release Delay



TL/C/11470-37

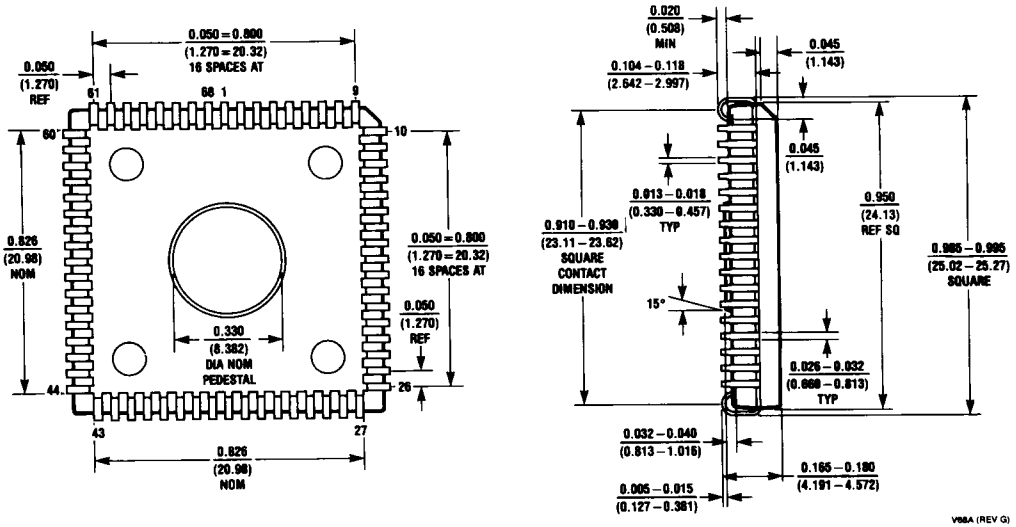
4.4.3.2 PWRGOOD Timing



TL/C/11470-38

Physical Dimensions inches (millimeters)

Lit. # 113100



68-Lead Plastic Chip Carrier (V)
NS Package Number V68A

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: 1(800) 272-9959
TWX: (910) 339-9240

National Semiconductor GmbH
Industriestrasse 10
D-8080 Furstenfeldbruck
West Germany
Tel: (0-81-41) 103-0
Telex: 527-649
Fax: (08141) 103554

National Semiconductor Japan Ltd.
Sanseido Bldg. 5F
4-15 Nishi Shinjuku
Shinjuku-Ku,
Tokyo 160, Japan
Tel: 33-299-7001
FAX: 33-299-7000

National Semiconductor Hong Kong Ltd.
Suite 513, 5th Floor
Chinachem Golden Plaza,
77 Mody Road, Tsimshatsui East,
Kowloon, Hong Kong
Tel: 3-7231290
Telex: 52996 NSSEA HX
Fax: 3-3112536

National Semicondutores Do Brasil Ltda.
Av. Brig. Faria Lima, 1383
6.0 Andor-Conj. 62
01451 Sao Paulo SP, Brasil
Tel:
Fax

National Semiconductor (Australia) PTY, Ltd.
1st Floor, 441 St. Kilda Rd.
Melbourne, 3004
Victory, Australia

031022 ✓ R