

8-bit telecom microcontrollers**PCD33xxA Family****CONTENTS**

1	INTRODUCTION
2	FEATURES
3	GENERAL DESCRIPTION
4	BLOCK DIAGRAM
5	PINNING INFORMATION
5.1	Pinning
5.2	Pin description
6	FUNCTIONAL DESCRIPTION
6.1	Central processing unit
6.2	Program memory
6.3	Data memory
6.3.1	Working registers
6.3.2	Program Counter stack
6.4	Program Counter
6.5	Program Status Word
6.6	Interrupts
6.6.1	External interrupt
6.6.2	Derivative interrupt
6.6.3	Timer/event counter interrupt
6.7	Timer/event counter 1
6.7.1	Test 1/count input (T1)
6.8	Parallel ports
6.9	Timing
6.10	Reduced power modes
6.10.1	Idle mode
6.10.2	Stop mode
6.11	Oscillator
6.12	Reset
6.12.1	Passive external reset
6.12.2	Active external reset
6.12.3	Internal reset
6.12.4	Reset state
6.13	Derivative logic
7	INSTRUCTION SET
7.1	Instruction map
8	DEFINITIONS
9	LIFE SUPPORT APPLICATIONS

8-bit telecom microcontrollers

PCD33xxA Family

1 INTRODUCTION

This data sheet describes the shared properties of the PCD33xxA Family of microcontrollers and its quickly growing number of derivative microcontrollers. For a particular microcontroller, this data sheet should be read in conjunction with the individual data sheet of the specific device.

2 FEATURES

- 8-bit CPU, ROM, RAM, I/O all in one package
- Up to 8 kbytes ROM
- Up to 256 bytes RAM
- Over 100 instructions (based on MAB8048) all of 1 or 2 cycles
- 8 or more quasi-bidirectional I/O port lines
- 8-bit programmable timer/event counter 1
- 3 single-level vectored interrupts: external, timer/event counter, derivative
- Two test inputs, one of which also serves as the external interrupt input
- Power-on-reset, Stop and Idle modes
- Supply voltage range: 1.8 to 6 V
- Clock frequency: 1 to 16 MHz
- Operating temperature: -25 to +70 °C
- Manufactured in silicon gate CMOS process.

3 GENERAL DESCRIPTION

The PCD33xxA Family of microcontrollers provide up to 8 kbytes of program memory and up to 256 bytes of RAM. All devices include flexible I/O ports, an 8-bit programmable timer/event counter and a choice of single-level vectored interrupts. The instruction set is based on that of the well-known MAB8048. Being similar to the MAB8400 family of NMOS controllers, some devices can serve as CMOS replacements, especially where the lower power consumption and higher speed provide advantages.

A range of One Time Programmable (OTP) devices with external program memory and 'Piggy-backs', as well as emulation probes and prototyping systems are available.

8-bit telecom microcontrollers

PCD33xxA Family

4 BLOCK DIAGRAM

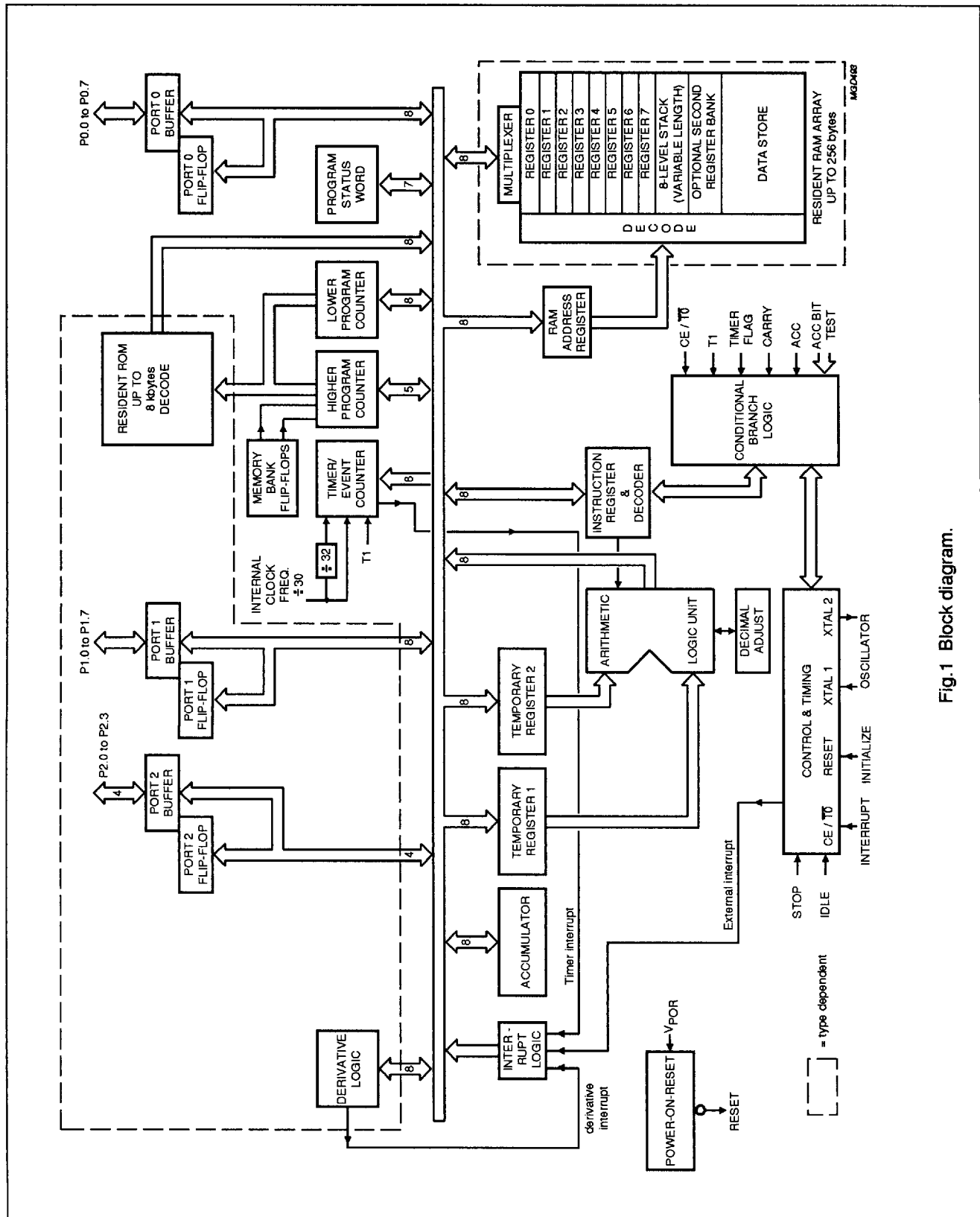


Fig.1 Block diagram.

8-bit telecom microcontrollers

PCD33xxA Family

5 PINNING INFORMATION**5.1 Pinning**

For individual pinning configurations consult the data sheet of the specific device.

5.2 Pin description

Table 1 describes the common functions of the devices. For full details of pin descriptions consult the data sheet of the specific device.

Table 1 Common functions

SYMBOL	TYPE	DESCRIPTION
V _{SS}	P	ground
V _{DD}	P	positive supply voltage
XTAL1	I	crystal oscillator/external clock input
XTAL2	O	crystal oscillator output
RESET	I	reset input
CE/ $\overline{T0}$	I	Chip enable/Test 0
T1	I	Test 1/count input of 8-bit timer/event counter 1
P0.0 to P0.7	I/O	Port 0: quasi-bidirectional I/O lines
P1.0 to P1.7	I/O	Port 1: quasi-bidirectional I/O lines
P2.0 to P2.3	I/O	Port 2: quasi-bidirectional I/O lines

8-bit telecom microcontrollers

PCD33xxA Family

6 FUNCTIONAL DESCRIPTION**6.1 Central processing unit**

The PCD33xxA Family provides an adequate instruction set with arithmetic, logic, branching, input/output and control facilities. Special highlights are the instructions for BCD arithmetic, nibble handling, conditional branches, loop control (DJNZ) and table look-up (MOVP).

Code and execution efficiency is achieved by using a maximum of two bytes and two execution cycles per instruction (see Chapter 7).

6.2 Program memory

The program memory consists of up to 8 kbytes of read-only memory (ROM). Each location is directly addressable by the Program Counter. The program memory is mask-programmed at the factory. Figure 2 illustrates the program memory map.

Four program memory locations are of special importance:

- Location 0: first instruction to be executed after the processor is reset
- Location 3: first instruction of an external interrupt ($\overline{\text{CE/T0}}$) routine
- Location 5: first instruction of a derivative interrupt routine
- Location 7: first instruction of a timer/event counter interrupt routine.

Only 11 bits of the 13-bit Program Counter function as a counter. The two most significant bits can only be preset. The program memory is therefore, structured into banks of 2 kbytes. Transfer of control to other memory banks is performed by unconditional branches (JMP) or subroutine calls (CALL) when another memory bank has been pre-selected (by SEL MB instruction).

Each program memory bank is further divided into 8 pages of 256 bytes. Indirect (JMPP) and conditional branches cannot cross page boundaries.

6.3 Data memory

Data memory consists of up to 256 bytes of random access memory (RAM). All locations are indirectly addressable using RAM pointer registers. Up to 16 register locations are directly addressable. Data memory also includes an 8-level Program Counter stack addressed by a 3-bit Stack Pointer. All RAM locations make efficient program loop counters if used with

the decrement register and test instruction (DJNZ). Figure 3 illustrates the data memory map.

6.3.1 WORKING REGISTERS

Locations 0 to 7 are working registers. They are accessible by efficient one byte/one cycle instructions, thus making these locations suitable for frequently accessed intermediate results.

As an alternative to locations 0 to 7, locations 24 to 31 may be used as working registers. Register Bank selection is made by SEL RB0/RB1 instructions. Register Bank 1 may be used as an extension of Register Bank 0, as an alternative register bank for interrupt service or as general purpose data memory.

The first two locations of each bank (R0, R1, R0' and R1') serve as RAM pointers that indirectly address all RAM locations.

6.3.2 PROGRAM COUNTER STACK

Locations 8 to 23 may be used as an 8-level Program Counter stack reserving 2 locations per level, or as general purpose RAM. The stack (see Fig.5) saves return addresses and status during interrupt or subroutine servicing. Nesting of subroutines and/or interrupts is permitted up to 8-levels deep.

The 3-bit Stack Pointer always points to the next free stack level. Following device reset, the Stack Pointer points to level 0 (locations 8 and 9). On each subroutine call (CALL) or interrupt, the contents of the Program Counter and bits 4, 6 and 7 of the Program Status Word are transferred to the level indicated by the Stack Pointer. The Stack Pointer increments and points to the next free level. Overflow from level 7 to level 0 occurs after nesting eight levels deep. Further subroutine calls and/or interrupts must not occur at this stage since this would result in loss of program content; overriding level 0 content.

Return from interrupt must be performed by the RETR instruction, which decrements the Stack Pointer and restores the Program Counter and Program Status Word, valid before the interrupt occurred. Return from subroutine should be performed by the RET instruction. In contrast to RETR, RET does not restore the Program Status Word.

As a general rule, the use of RETR in conjunction with a subroutine call is not recommended. The use of RETR must also be avoided with subroutines called from interrupt routines because it prematurely terminates the interrupt state (see Section 6.6).

8-bit telecom microcontrollers

PCD33xxA Family

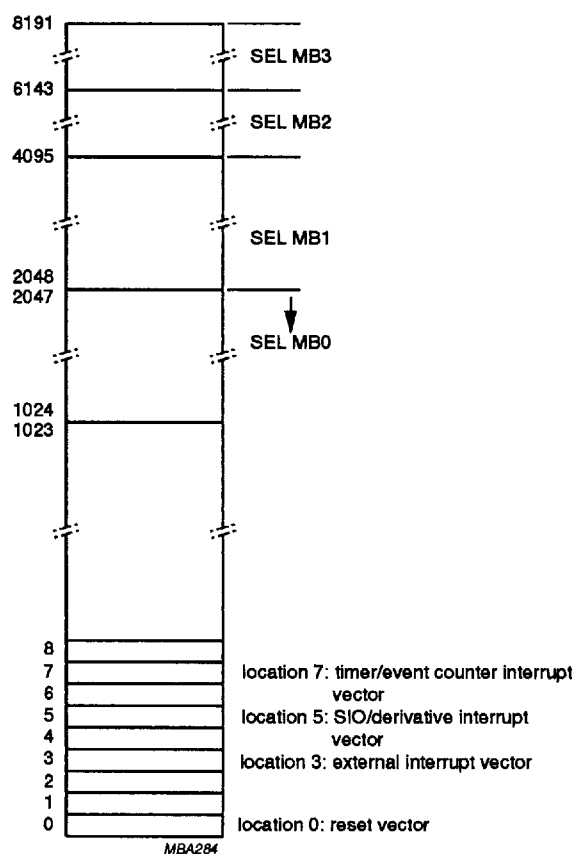


Fig.2 Program memory map.

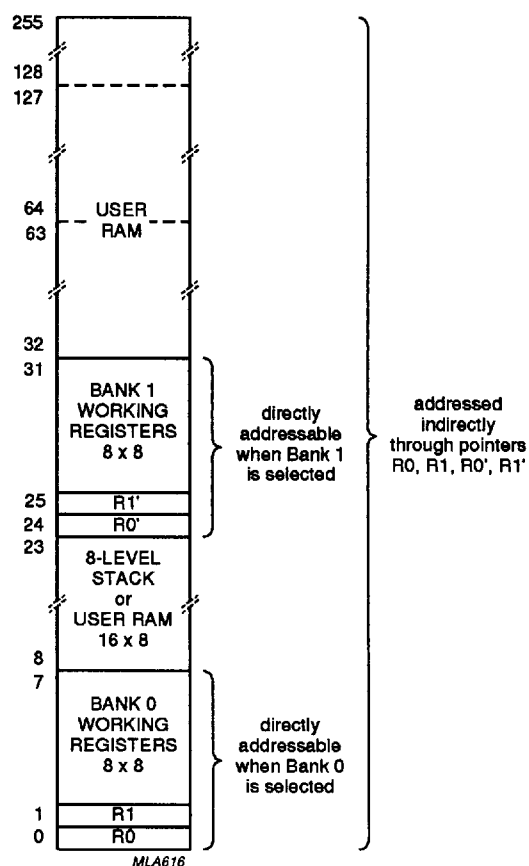


Fig.3 Data memory map.

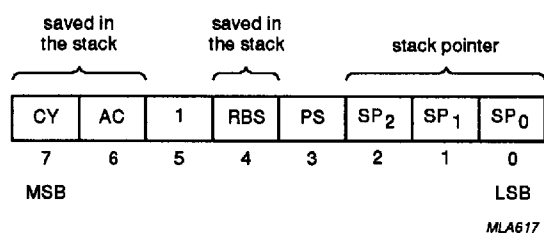


Fig.4 Program Status Word.

8-bit telecom microcontrollers

PCD33xxA Family

6.4 Program Counter

The 13-bit Program Counter is able to address up to 8 kbytes of ROM (see Fig.6). 11 bits (PC0 to PC10) are auto-incrementing. The two most significant bits (PC11 and PC12) must be changed under program control by SEL MB followed by a JMP or CALL instruction.

6.5 Program Status Word

The Program Status Word (PSW) is an 8-bit register in the CPU which stores information about the current status of the microcontroller (see Fig.4).

The PSW bits are:

- Bits 0 to 2: Stack Pointer bits (SP0, SP1, SP2)
- Bit 3: timer Prescaler Select (PS); 0 = modulo-32, 1 = modulo-1 (no prescaling)
- Bit 4: working Register Bank Select (RBS); 0 = register bank 0, 1 = register bank 1

- Bit 5: not used (fixed at 1)
- Bit 6: Auxiliary Carry (AC); half-carry bit generated by an ADD instruction and used by the decimal adjust instruction DA A
- Bit 7: Carry (CY); the carry flag indicates that the previous operation resulted in an overflow of the Accumulator.

All bits can be read using the MOV A, PSW instruction. Bits 0, 1 and 2 are affected by CALL, RET, RETR and interrupts. Bit 3 can be controlled by MOV PSW, A and bit 4 by SEL RB instructions. Bit 6 is set and cleared as a side-effect of ADD and ADDC instructions. Bit 7 is affected by ADD, ADDC, DA, RLC, RRC, CLR C and CPL C instructions.

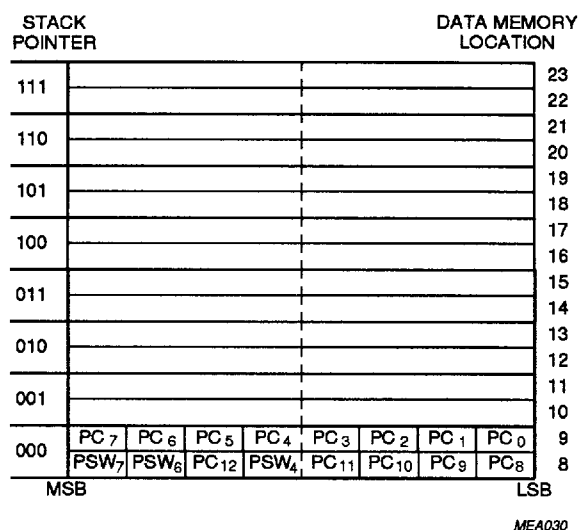


Fig.5 Program Counter Stack.

8-bit telecom microcontrollers

PCD33xxA Family

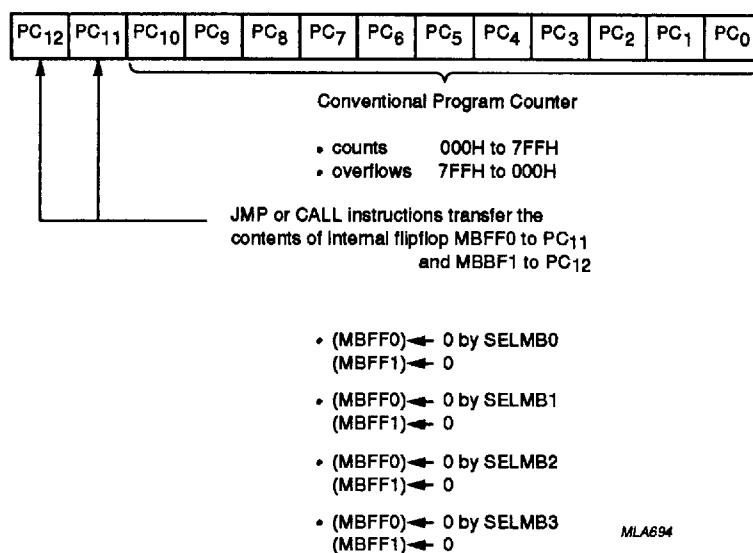


Fig.6 Program Counter.

6.6 Interrupts

External, derivative and timer/event counter interrupts are handled by the PCD33xxA Family. The interrupt mechanism is single level, i.e. an executing interrupt routine cannot be pre-empted unless by reset. Further interrupt requests are latched. If several interrupt requests are detected simultaneously, they are honoured according to their priority:

- External interrupt (highest priority)
- Derivative interrupt
- Timer/event counter interrupt (lowest priority).

An interrupt request is only sensed if the corresponding enable flag is set (see Fig.7). When the request is honoured, the contents of the Program Counter and bits 4, 6 and 7 of the Program Status Word are saved on the Program Counter stack. The Program Counter is loaded with the appropriate interrupt vector, thereby indicating the beginning of the interrupt routine. Since the Accumulator is not automatically saved, it must be saved and restored by user software. The interrupt routine must be terminated by the RETR (return and restore) instruction. At least one instruction of the main program will then be executed before another interrupt routine is entered.

To avoid erroneous real-time programs, a few words of caution:

- While the interrupt is in progress, the two most significant bits of the Program Counter are frozen at zero. Thus, interrupt routines and subroutines called from interrupt routines must reside entirely in Bank 0.
- The SEL MB instruction must not be used in interrupt routines and in subroutines called from interrupt routines. Otherwise, the changed contents of MBFF0 and MBFF1 (see Fig.6) may lead to erroneous JMP and CALL destinations after return from interrupt.
- Subroutines and nested subroutines called from the interrupt routine must all end with RET since RETR clears the Interrupt In Progress flag (IIP), as a side-effect (see Figs 7 and 8). Further pending interrupts would then interfere with the interrupt routine in progress.

8-bit telecom microcontrollers

PCD33xxA Family

6.6.1 EXTERNAL INTERRUPT

A LOW-to-HIGH transition on the $\overline{\text{CE/T0}}$ pin is latched in the digital filter/latch if the HIGH state exceeds 7 clock periods after a LOW state of more than 4 clock periods. If the external interrupt is enabled the External Interrupt Flag (EIF) is also asserted, thus constituting a valid external interrupt request. As soon as the IIP is clear, indicating that no interrupt routine is in progress, the external interrupt is invoked by a forced CALL to location 3. The EIF is simultaneously cleared (see Figs 7 and 8). The interrupt routine may acknowledge the interrupt via port lines. Execution of a DIS I (disable external interrupt) instruction cancels a stored interrupt request by clearing both the digital filter/latch and the EIF.

For some devices the external interrupt is shared between the $\overline{\text{CE/T0}}$ pin and additional wake-up interrupts from the derivative logic. Software polling may be necessary to determine the origin of request. Since the interrupt flags of the derivative logic are not cleared by DIS I, the external interrupt routine must include instructions that will remove the cause of the external interrupt. For more details about shared external interrupts consult the data sheet of the specific device.

6.6.1.1 Chip Enable/Test 0 Input ($\overline{\text{CE/T0}}$)

The $\overline{\text{CE/T0}}$ input has two purposes:

- External interrupt input (see Section 6.6.1)
- Test 0 input.

When used as a Test 0 input (external interrupt disabled) the conditional branch instruction JT0 will cause a jump if $\overline{\text{CE/T0}} = 1$. The conditional branch instruction JNT0 will also cause a jump if $\overline{\text{CE/T0}} = 0$. If $\overline{\text{CE/T0}}$ is not used, it must be tied to V_{DD} or V_{SS} .

6.6.2 DERIVATIVE INTERRUPT

The derivative interrupt is shared between all the interrupt sources in the derivative logic. Software polling may be necessary to determine the origin of a request.

An interrupt condition in the derivative logic will pull the PIN line LOW. If the derivative interrupt is enabled and no interrupt routine is in progress, the derivative interrupt routine will be invoked by a forced CALL to program memory location 5. The derivative interrupt routine must include instructions that will remove the cause of the derivative interrupt and thus reset PIN to its inactive HIGH state. For derivative interrupts, consult the data sheet of the specific device.

6.6.3 TIMER/EVENT COUNTER INTERRUPT

If the timer/event counter interrupt is enabled, a timer/event counter 1 overflow sets the Timer Interrupt Flag (TIF). As soon as IIP is clear, meaning that no interrupt routine is in progress, the timer/event counter interrupt routine is invoked by a forced CALL to program memory location 7. The TIF is simultaneously cleared (see Figs 7 and 8). Execution of a DIS TCNTI (disable timer/event counter interrupt) instruction cancels a stored interrupt request by clearing TIF.

The timer/event counter interrupt may also be used to simulate a second external interrupt. After an enable timer/event counter interrupt (EN TCNTI), the counter mode is enabled by a STRT CNT instruction which loads FFH (the state preceding overflow) into the counter. A positive edge on the T1 pin will overflow the counter and set TIF.

8-bit telecom microcontrollers

PCD33xxA Family

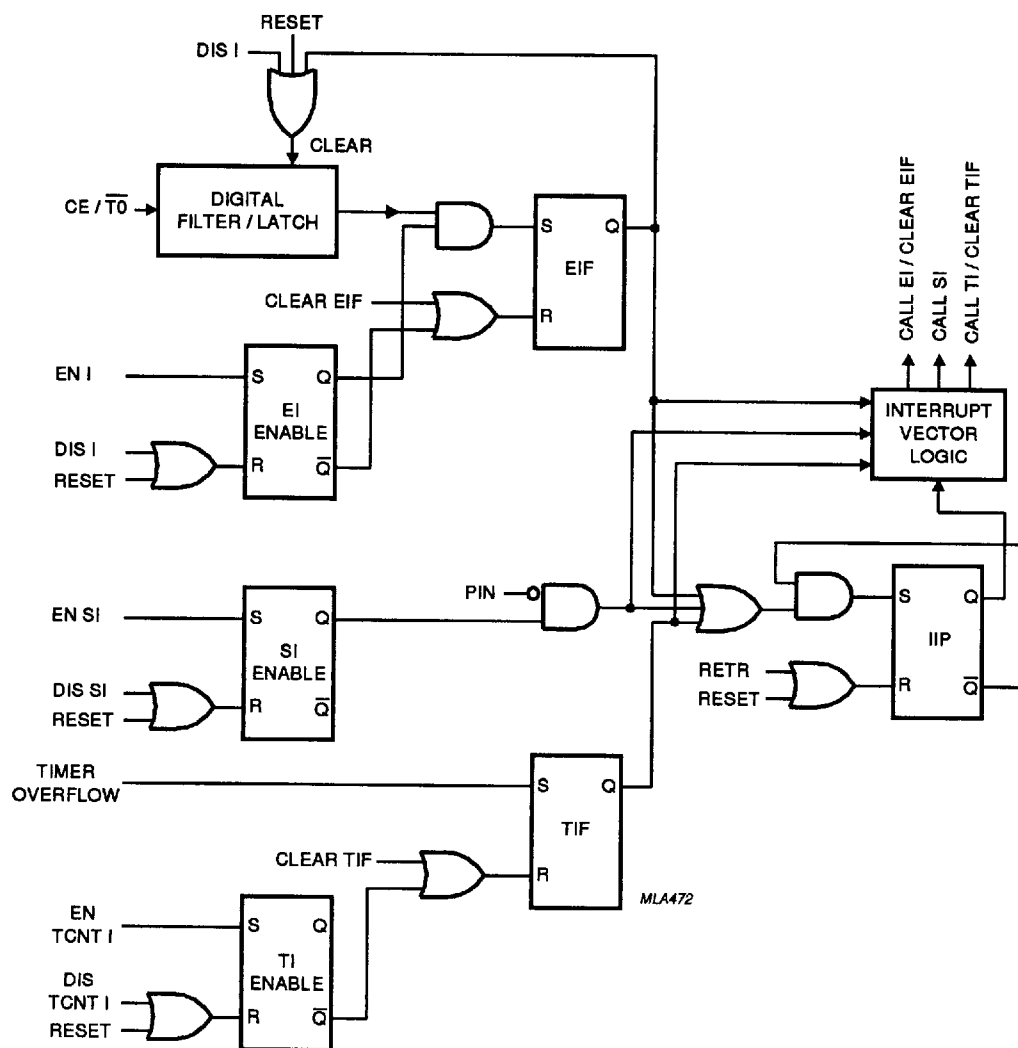


Fig.7 Simplified interrupt logic schematic (the R input overrules the S input for all flags).

8-bit telecom microcontrollers

PCD33xxA Family

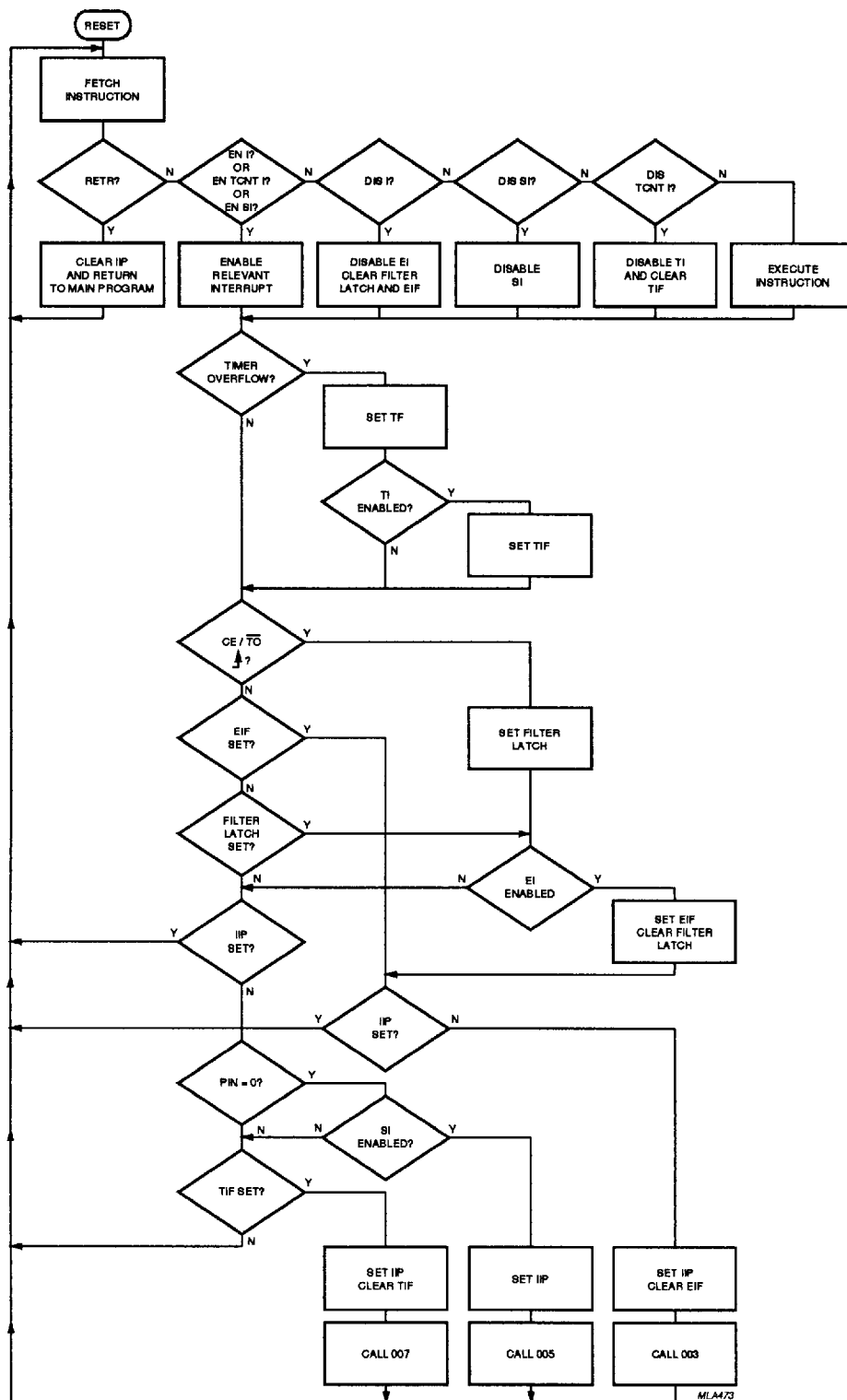


Fig.8 Flow chart illustrating CPU control in the presence of interrupts.

8-bit telecom microcontrollers

PCD33xxA Family

6.7 Timer/event counter 1

An internal 8-bit up counter is provided. The counter can be preset and read by the MOV T, A and MOV A, T instructions.

When the counter is to be used in the timer mode, a STRT T (start timer) instruction must be executed. Depending on the PS bit in the Program Status Word, the counter will increment every machine cycle ($PS = 1$, $\frac{1}{30} \times f_{xtal}$) or every 32 machine cycles ($PS = 0$, $\frac{1}{960} \times f_{xtal}$). STRT T clears the prescaler (see Fig.9) which is not otherwise accessible.

To count external events a STRT CNT (start event counter) instruction must be executed. A LOW-to-HIGH transition on pin T1 is counted if the HIGH state exceeds 4 clock periods after a LOW state of more than 4 clock periods. The maximum count rate is one increment per machine cycle ($1/30 \times f_{\text{XTAL}}$).

The timer mode and the event counter mode are both inhibited after reset or by executing a STOP TCNT (stop timer/event counter) instruction (see Fig.9).

In both the timer and in event counter modes, overflow has two effects:

- If the timer/event counter interrupt is enabled TIF is asserted thereby generating a timer/event counter interrupt request (see Section 6.6).
- The Timer Flag (TF) is set. TF can be tested by conditional branch instructions JTF (jump if TF = 1) or JNTF (jump if TF = 0). The JTF and JNTF instruction, as a side-effect, reset TF. The only other way to clear TF is to reset the microcontroller.

6.7.1 TEST 1/COUNT INPUT (T1)

The T1 input has two purposes:

- Count input of 8-bit timer/event counter 1 (see Section 6.7)
- Test 1 input.

When used as a Test 1 input the conditional branch instruction JT1 will cause a jump if $T1 = 1$. The conditional branch instruction JNT1 will also cause a jump if $T1 = 0$. If $T1$ is not used, it must be tied to V_{DD} or V_{SS} .

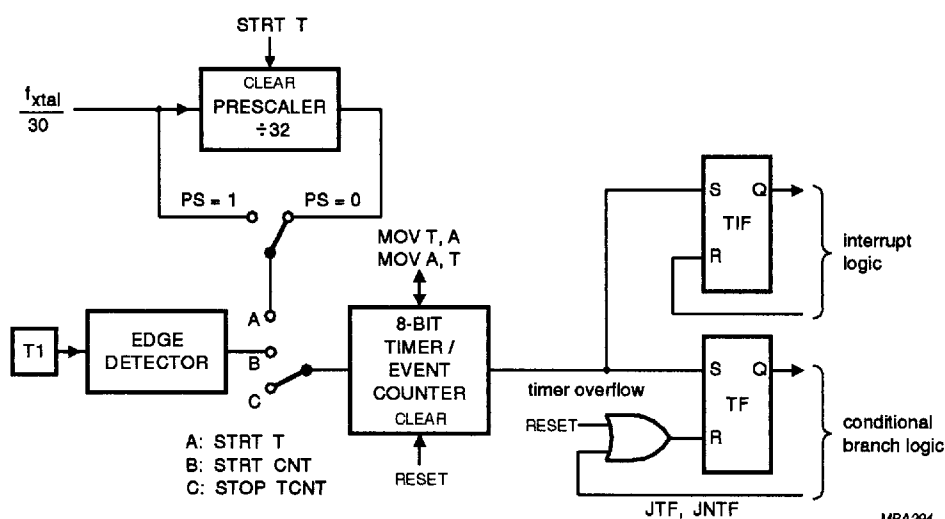


Fig.9 Timer/event counter 1.

8-bit telecom microcontrollers

PCD33xxA Family

6.8 Parallel ports

Three standard quasi-bidirectional I/O ports are defined:

- Port 0: parallel port of 8 lines (P0.0 to P0.7)
- Port 1: parallel port of 8 lines (P1.0 to P1.7)
- Port 2: parallel port of 4 lines (P2.0 to P2.3).

Several members of the PCD33xxA Family provide all 20 port lines. The eight Port 0 lines (P0.0 to P0.7) are available as a minimum. In addition to the standard ports, many PCD33xxA microcontrollers offer a variety of derivative ports. Please consult the data sheet of the specific device.

In general, all parallel ports can be used as either inputs or outputs. Output data written to a port is latched and remains unchanged until rewritten. If the port is used as an input, the external data is not latched and must remain stable until it is accessed by the CPU.

The standard port configuration is illustrated in Fig.11. When a logic 0 is written to the master/slave flip-flop, TR2 and TR3 are both in the OFF condition. TR1 turns ON and drives the output to V_{SS} .

When a logic 1 is written to the master/slave flip-flop, TR1 turns OFF. TR2 and TR3 both turn ON driving the output rapidly to V_{DD} . TR2 remains in the ON condition for the duration of the write pulse only. The constant current source is responsible for keeping the output line high. Sufficient source current is available for a TTL load HIGH level; the line can, however, be overridden by an external device. This is used when the port line serves as an input, but it may also be useful for wired-OR applications. In the latter case, unnecessary current through external devices is avoided since repeated logic 1 write operations will not activate TR2. The booster transistor TR2 is only asserted during a LOW-to-HIGH transition of the master/slave flip-flop. If the port line is to be used as an input, a logic 1 should first be stored in the master/slave flip-flop to turn TR1 OFF.

Access to Ports 0, 1 and 2 is provided by the parallel input/output instructions IN, OUTL, ANL and ORL. IN inputs port data to the Accumulator. OUTL outputs Accumulator data to the port. ANL and ORL are used for data manipulation in the port flip-flop. In contrast to Ports 0, 1 and 2, derivative ports are accessed by the derivative input/output instructions MOV, ANL and ORL. ANL and ORL are used for data manipulation in the port flip-flop. MOV is used for all data transfers between port and Accumulator. The source data for the Accumulator can be loaded from either the port line or the port flip-flop. Two derivative addresses are therefore provided per port (see Table 2).

All standard and derivative port accesses are performed by two-cycle instructions. Their instruction timing is shown in Fig.10. For input, data on port lines is sensed during timeslots 3 and 4 of machine cycle 2 (see Sections 6.9 and 6.11). For output, the data change occurs in timeslot 7. For OUTL, data changes during machine cycle 1. For ANL, ORL and MOV Dx, A, data changes during machine cycle 2.

Table 2 Derivative port address pair

ADDRESS	TYPE	ACCESS
8-bit line address	R	derivative port line
8-bit flip-flop address	R/W	derivative port flip-flop

Three port output configurations are available:

1. Standard Port; quasi-bidirectional I/O with switched pull-up current source of 100 μ A (typ.) and p-channel booster transistor TR2. TR2 is only active for 1 clock cycle during LOW-to-HIGH transitions (see Fig.11).
2. Open-drain; quasi-bidirectional I/O with only an n-channel open-drain output. Application as an output requires connection of an external pull-up resistor (see Fig.12). If unused, an Option 2 output should be tied to V_{SS} . This keeps the input path from floating, thereby avoiding undesirable current flow through input stages.
3. Push-pull; drive capability of the output will be 5 mA (typ.) at $V_{DD} = 3$ V in both polarities. Since short circuit currents would flow during input, push-pull lines must only be used as outputs (see Fig.13).

Besides port output configurations, the port flip-flop state, after reset, is specified for each individual port line. Usually the 'set option' will be selected, which avoids short-circuits for ports intended as inputs. However, there may be cases in which the port should output a logic zero after reset. The user may then specify the 'reset option' for certain port lines.

8-bit telecom microcontrollers

PCD33xxA Family

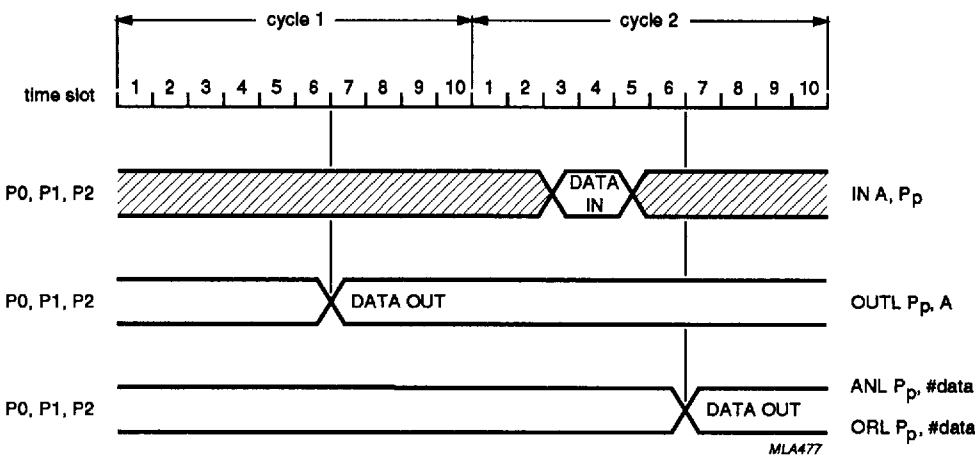


Fig.10 Input /output timing of standard and derivative ports.

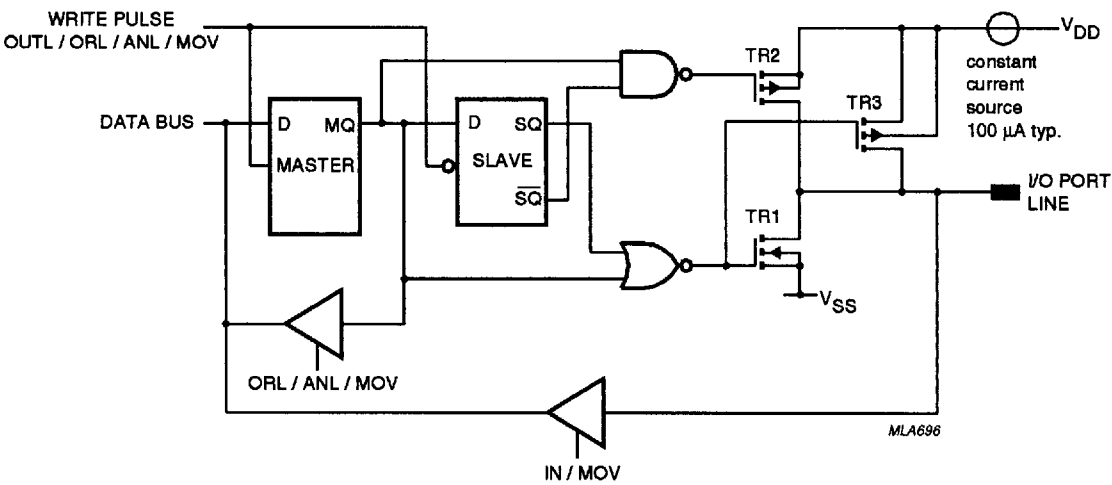


Fig.11 Standard output with switched current source.

8-bit telecom microcontrollers

PCD33xxA Family

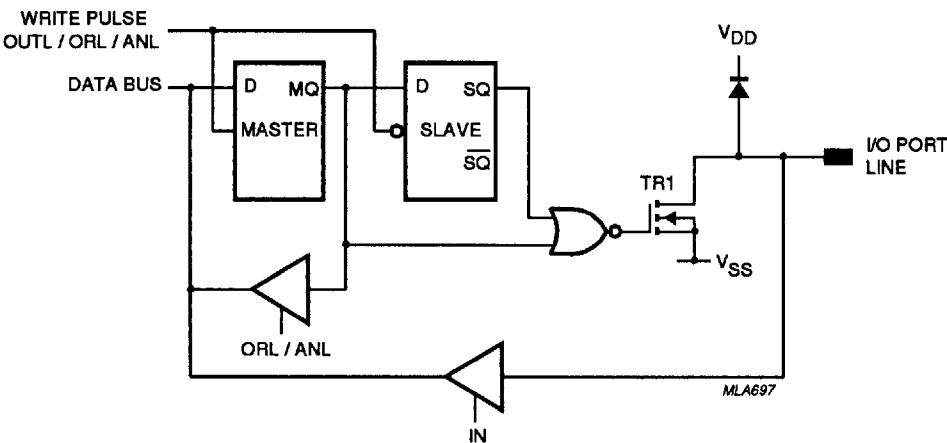


Fig.12 Open-drain output.

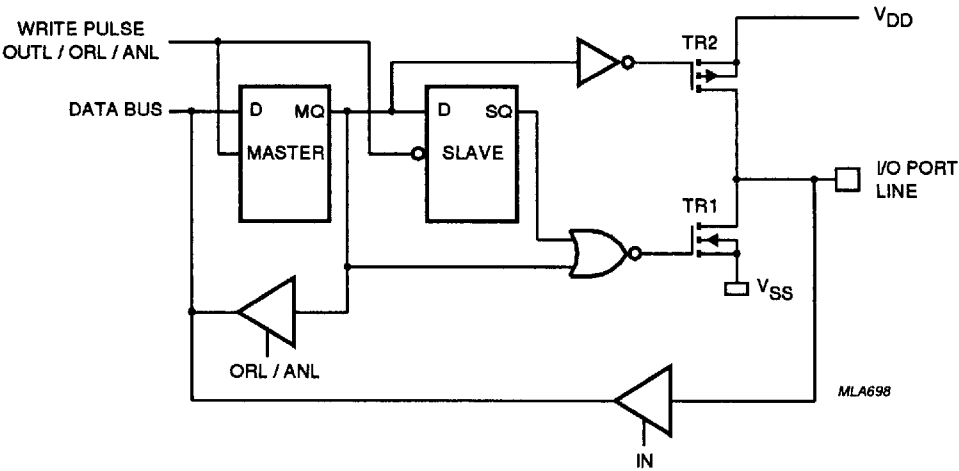


Fig.13 Push-pull output.

8-bit telecom microcontrollers

PCD33xxA Family

6.9 Timing

Every machine cycle consists of 10 time slots which are again subdivided into 3 clock periods each (see Fig.14).

Permitted clock frequencies range from 1 MHz to a maximum, which is a function of the supply voltage. At $V_{DD} \geq 4.5$ V, a 16 MHz maximum clock frequency is guaranteed.

The clock signal may be internally generated by an on-chip oscillator. Alternatively, an external clock may be applied to pin XTAL1. In this configuration, a short circuit with an internal pull-up transistor on XTAL1 may occur while the oscillator is inhibited (see Section 6.11). Care should be taken to avoid excessive current flow.

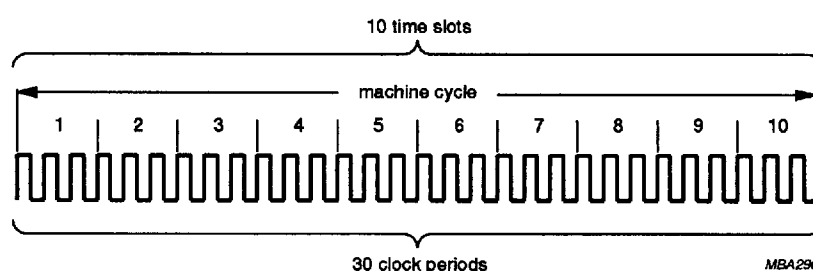


Fig.14 Timing of a machine cycle.

6.10 Reduced power modes

6.10.1 IDLE MODE

The Idle mode is very useful in low-power applications. When all computational tasks are completed, the device can be put into standby instead of into a busy waiting loop. Nevertheless, the device is on the alert and ready to respond rapidly to any interrupt.

The microcontroller enters the Idle mode via the IDLE instruction. In the Idle mode, all activity is halted except for the oscillator, the timer/event counter 1 and the serial I/O interface (if available).

The microcontroller leaves the Idle mode when an enabled interrupt occurs. The interrupt routine is executed before operation resumes with the instruction following the IDLE opcode.

For timer/event counter interrupts and derivative interrupts, termination of the Idle mode is straightforward. However, care must be taken when the Idle mode is left by the external interrupt since $\overline{CE/\overline{T0}}$ is triggered on the rising edge. If $\overline{CE/\overline{T0}}$ was HIGH prior to entering the Idle mode, it must be taken LOW before the positive edge can be generated. Figure 15 specifies the exact timing for leaving the Idle mode via the external interrupt $\overline{CE/\overline{T0}}$.

If no interrupt is enabled, the Idle mode can only be terminated by an active signal on the RESET pin. A normal reset sequence is executed (see Fig.15).

6.10.2 STOP MODE

The Stop mode allows very low-power applications. When all computational tasks are completed, the device can be almost completely shut off by stopping its oscillator. In contrast to the Idle mode, the device is not ready to respond rapidly to any interrupt.

When the microcontroller enters the Stop mode via the STOP instruction; the oscillator is switched off. All internal states and I/O levels are maintained.

The microcontroller leaves the Stop mode by a HIGH level on $\overline{CE/\overline{T0}}$ or a reset. In the latter case, a normal reset sequence is executed (see Fig.16).

In contrast to the Idle mode and the external interrupt mechanism, the microcontroller responds to a HIGH level on $\overline{CE/\overline{T0}}$ rather than to a positive edge. If $\overline{CE/\overline{T0}}$ is HIGH when the STOP instruction is executed, the Stop mode will not be entered.

A positive edge on $\overline{CE/\overline{T0}}$ continues program execution after a 1866 clock cycle delay, which ensures proper oscillator start-up. If the external interrupt is enabled, the device executes the instruction following the STOP opcode before diverting to the interrupt routine. If the external interrupt is disabled, program execution continues with the instructions following the STOP opcode (see Fig.16).

8-bit telecom microcontrollers

PCD33xxA Family

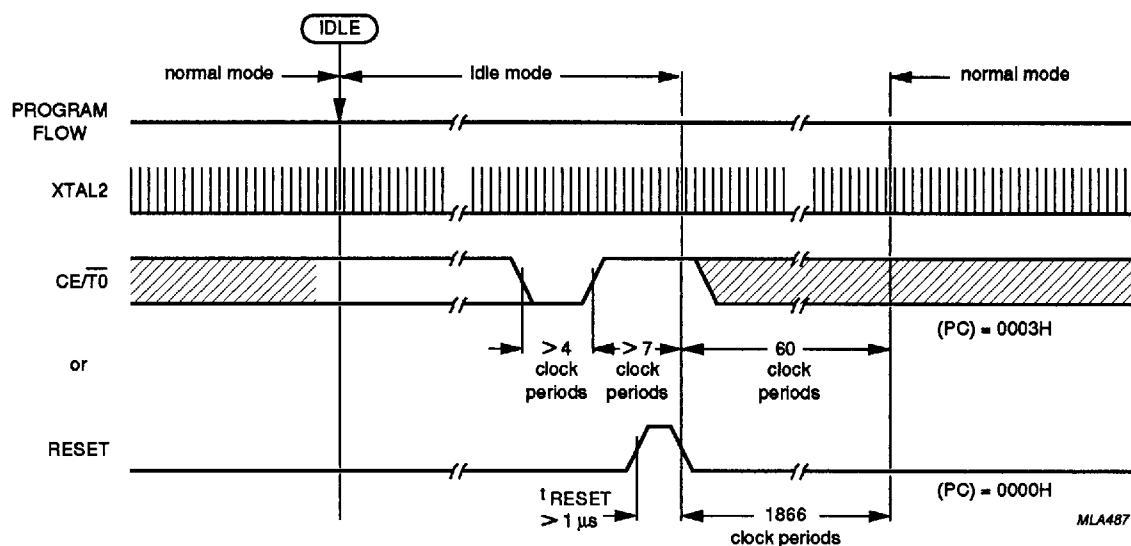


Fig.15 Entering and leaving the Idle mode.

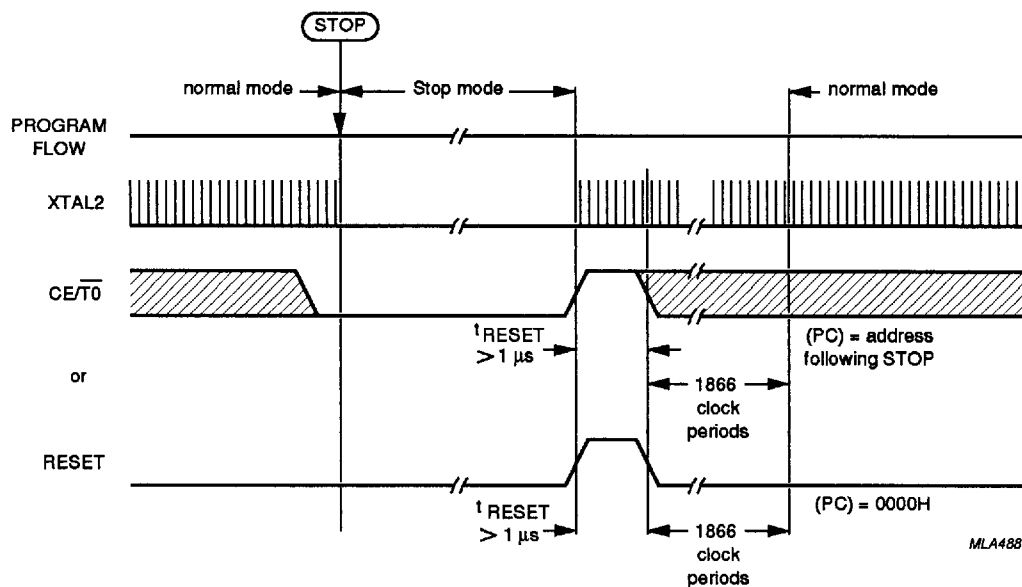


Fig.16 Entering and leaving the Stop mode.

8-bit telecom microcontrollers

PCD33xxA Family

6.11 Oscillator

The on-chip oscillator basically consists of an inverter stage which includes a feedback resistor and load capacitors (see Fig.17). In most applications, a quartz crystal will be connected between XTAL1 and XTAL2. Alternatively, a ceramic resonator or an inductor may be used as a timing element.

When the supply voltage drops below the power-on reference level, the oscillator is inhibited. The internal oscillator can also be inhibited by the STOP instruction under software control (see Section 6.10.2).

The transconductance (g_m) of the inverter stage can be mask-programmed, thereby optimizing the oscillator for a specific frequency and resonator.

Three standard transconductance options, referred to as LOW, MEDIUM and HIGH, can be specified by the user depending on the device chosen.

With $C_1 = C_2 = 10$ pF on-chip, external capacitors are not required for quartz oscillators. However, for adequate frequency stability, PXE resonators need external capacitors in the order of the static resonator capacitance C_0 , such as external $C_1 = C_2 = 30$ to 100 pF.

Oscillator start-up time depends mainly on the external timing element. The start-up time of a quartz crystal is several milliseconds because of the narrow crystal bandwidth. For proper oscillator start-up, the transconductance (g_m) of the inverter stage must fulfil relationship (1) and (2); shown below.

$$g_m > 4.2 \left[R_X \omega^2 (C_L + C_0 + C_F)^2 + \frac{1}{R_P} \right] \quad (1)$$

$$g_m < \frac{C_1 \times C_2}{\left[R_X (C_0 + C_F)^2 + \frac{1}{\omega^2 R_P} \right]} \quad (2)$$

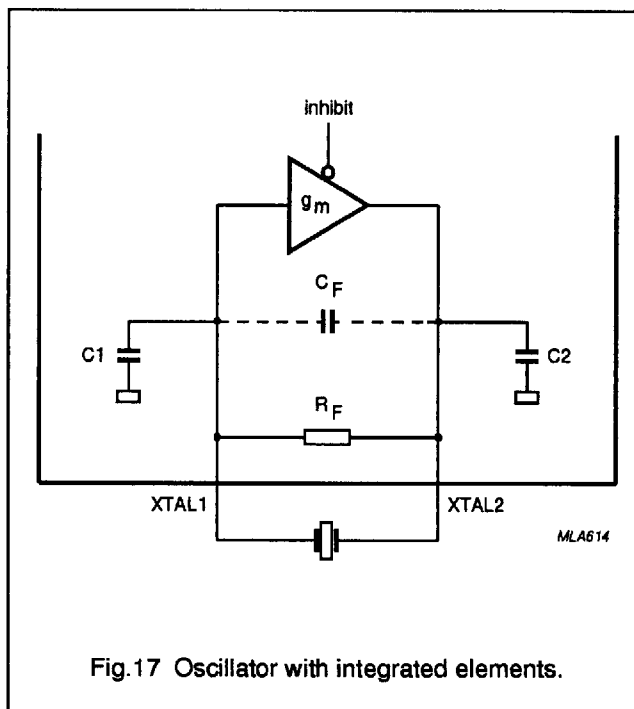


Fig.17 Oscillator with integrated elements.

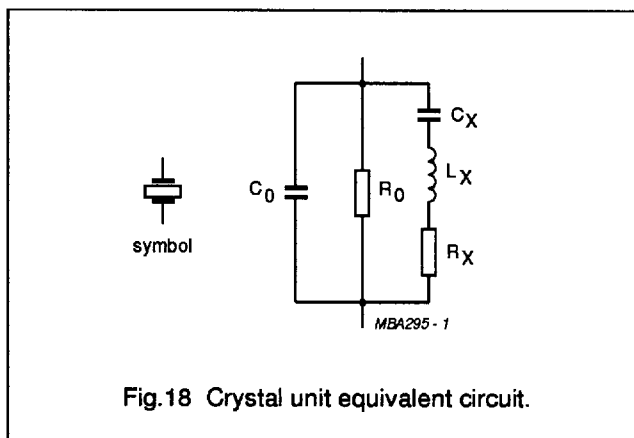


Fig.18 Crystal unit equivalent circuit.

Table 3 Notation to relationship (see Figs 17 and 18)

SYMBOL	DEFINITION
R_X	resonator series resistance
C_0	static resonator capacitance
R_0	resonator loss resistance
R_P	$R_0 // R_F$
R_F	feedback resistor
C_L	$C_1 \times C_2 / (C_1 + C_2)$ (load capacitance)
C_F	parasitic feedback capacitance (typically 2 pF on-chip, external value depends on printed-circuit board wiring)
ω	$2\pi f_{osc}$

8-bit telecom microcontrollers

PCD33xxA Family

6.12 Reset

To ensure proper start-up, the microcontroller must be initialized to a defined starting condition. The device executes the first instruction 1866 clock cycles after the falling edge of the internal reset.

6.12.1 PASSIVE EXTERNAL RESET

A passive reset is generated by the RC circuit illustrated in Fig.19. While V_{DD} rises, the discharged C_{reset} keeps the RESET pin near the V_{DD} level. When V_{DD} crosses the power-on reference level (V_{POR}) the Power-on-reset circuit generates a reset pulse of approximately 50 μs . This pulse is without effect since it feeds into the reset signal forced by the pulse on the RESET pin.

The f_{xtal} dependent minimum V_{DD} must be reached before the voltage on RESET drops below $V_{IH} = 0.7V_{DD}$.

This translates into a lower bound for $C_{reset}R_{reset}$ equal to twice the rise time of V_{DD} (for linearly rising V_{DD}) or eight times the time constant of V_{DD} (for exponentially rising V_{DD}). The internal diode rapidly discharges C_{reset} when V_{DD} falls off, ensuring reliable reset even after short interruptions of supply voltage. To avoid overload of the internal diode, an external diode should be added in parallel if $C_{reset} > 2.2 \mu F$.

6.12.2 ACTIVE EXTERNAL RESET

An active reset can be generated by driving the RESET pin HIGH from an external logic device. Such an active reset pulse should not fall off before V_{DD} has reached its f_{xtal} dependent minimum operating value.

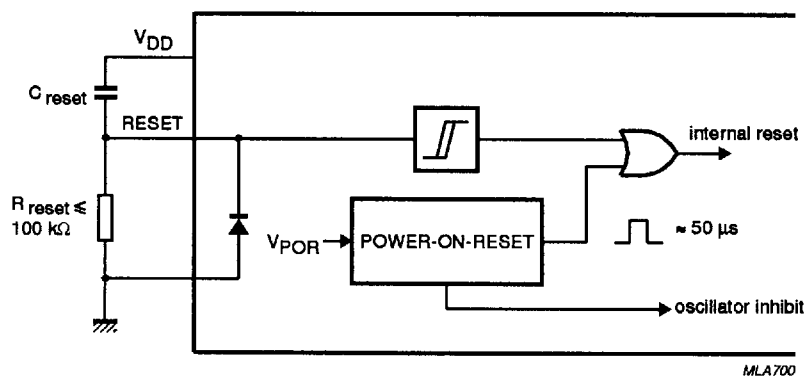


Fig.19 Passive external Power-on-reset.

8-bit telecom microcontrollers

PCD33xxA Family

6.12.3 INTERNAL RESET

In systems where V_{DD} reaches its f_{xtal} dependent minimum operating value before the clock f_{xtal} is applied, reset can be performed without external components. This condition is generally fulfilled with quartz and PXE resonators since oscillator start-up takes several milliseconds. Besides, rapid power-up is usually available in battery-powered systems.

If the internal Power-on-reset is used the RESET pin should be connected to V_{SS} . When V_{DD} increases above the power-on reference level V_{POR} , the Power-on-reset circuit generates a reset pulse of approximately 50 μs .

This pulse guarantees proper initialization under the conditions defined above.

The power-on reference level V_{POR} is a mask option. The user can select a reference voltage between 1.2 V and 3.6 V in discrete steps of 100 mV.

The chosen V_{POR} should have sufficient margin regarding the minimum intended V_{DD} .

A mask option without an internal Power-on-reset circuit is also available. It is recommended if the user does not intend to use the internal Power-on-reset circuit. In this case, the supply current requirements in Stop mode (see Section 6.10.2) will reduce to the level of leakage currents, i.e. virtually zero at ambient temperature.

6.12.4 RESET STATE

After a reset, the device state is characterized as follows:

- Program Counter 00H
- Memory bank 00H
- Register Bank 00H - Stack Pointer 00H (location pair 8 and 9)
- All interrupts disabled
- Timer/event counter 1 stopped and cleared
- Timer prescaler modulo-32 ($PS = 0$)
- Timer flag cleared
- All port flip-flops set to logic 1 or logic 0 depending on the port configuration
- Idle and Stop modes cancelled.

8-bit telecom microcontrollers

PCD33xxA Family

6.13 Derivative logic

Derivative logic is provided with many members of the PCD33xxA Family. The detailed description of the derivative circuitry is given in the data sheet of the specific device. In this section, the shared principles of derivative logic are briefly reviewed.

Derivative registers are accessed over the internal bus. The derivative registers are write-only, read-only or read/write (see Fig.20). They are addressed through the derivative Address Register when the derivative input/output instructions (MOV A, Dx; MOV Dx, A; ANL Dx, A and ORL Dx, A) are executed.

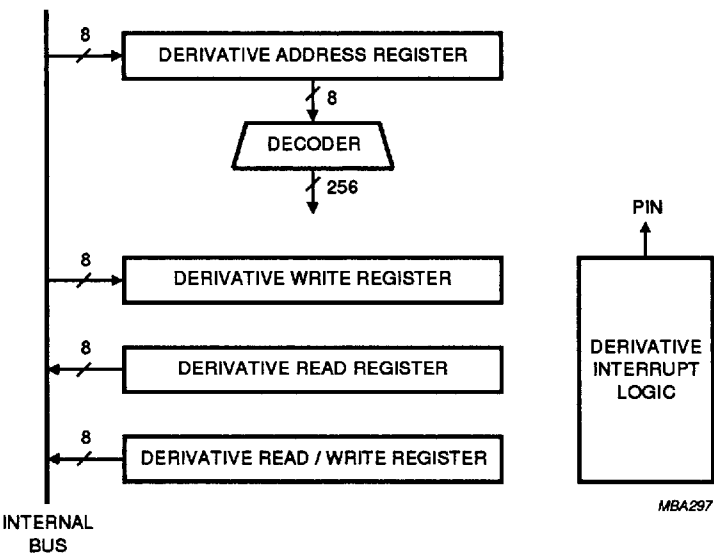


Fig.20 Block diagram of derivative logic.

Table 4 Summary of configurations

FEATURE	CONFIGURATION	DESCRIPTION
ROM	any mix of instructions	program; size restricted by ROM size (see Tables 5 and 6)
Ports	configuration 1	standard output (see Fig.11)
	configuration 2	open-drain output (see Fig.12)
	configuration 3	push-pull output (see Fig.13)
	set	flip-flop at logic 1 after reset
	reset	flip-flop at logic 0 after reset
Power-on reference	V _{POR}	1.2 to 3.6 V in increments of 100 mV; with ±500 mV accuracy
Oscillator	g _{mL}	LOW transconductance
	g _{mM}	MEDIUM transconductance
	g _{mH}	HIGH transconductance

8-bit telecom microcontrollers

PCD33xxA Family

7 INSTRUCTION SET

The PCD33xxA instruction set consists of over 100 one and two-byte instructions. Program code efficiency is high because all RAM locations and all ROM locations on a 256-byte page require only a single-byte address. Table 6 lists the symbols that are used in Table 5 and the Instruction map is shown in Section 7.1.

Table 5 PCD33xxA Family instruction set

MNEMONIC	OPCODE (HEX)	BYTES/ CYCLES	DESCRIPTION	FUNCTION	NOTES
Accumulator					
ADD A, Rr ⁽¹⁾	6<8 + r>	1/1	Add register contents to A	$(A) \leftarrow (A) + (Rr)$	r = 0 to 7
ADD A, @Rr ⁽¹⁾	6r	1/1	Add RAM data, addressed by Rr, to A	$(A) \leftarrow (A) + ((Rr))$	r = 0, 1
ADD A, #data ⁽¹⁾	03 data	2/2	Add immediate data to A	$(A) \leftarrow (A) + \text{data}$	
ADDC A, Rr ⁽¹⁾	7<8 + r>	1/1	Add carry and register contents to A	$(A) \leftarrow (A) + (Rr) + (C)$	r = 0 to 7
ADDC A, @Rr ⁽¹⁾	7r	1/1	Add carry and RAM data, addressed by Rr, to A	$(A) \leftarrow (A) + ((Rr)) + (C)$	r = 0, 1
ADDC A, #data ⁽¹⁾	13 data	2/2	Add carry and immediate data to A	$(A) \leftarrow (A) + \text{data} + (C)$	
ANL A, Rr	5<8 + r>	1/1	AND Rr with A	$(A) \leftarrow (A) \text{ AND } (Rr)$	r = 0 to 7
ANL A, @Rr	5r	1/1	AND RAM data addressed by Rr, with A	$(A) \leftarrow (A) \text{ AND } ((Rr))$	r = 0, 1
ANL A, #data	53 data	2/2	AND immediate data with A	$(A) \leftarrow (A) \text{ AND } \text{data}$	
ORL A, Rr	4<8 + r>	1/1	OR Rr with A	$(A) \leftarrow (A) \text{ OR } (Rr)$	r = 0 to 7
ORL A, @Rr	4r	1/1	OR RAM data, addressed by Rr, with A	$(A) \leftarrow (A) \text{ OR } ((Rr))$	r = 0, 1
ORL A, #data	43 data	2/2	OR immediate data with A	$(A) \leftarrow (A) \text{ OR } \text{data}$	
XRL A, Rr	D<8 + r>	1/1	XOR Rr with A	$(A) \leftarrow (A) \text{ XOR } (Rr)$	r = 0 to 7
XRL A, @Rr	Dr	1/1	XOR RAM data, addressed by Rr, with A	$(A) \leftarrow (A) \text{ XOR } ((Rr))$	r = 0, 1
XRL A, #data	D3 data	2/2	XOR immediate data with A	$(A) \leftarrow (A) \text{ XOR } \text{data}$	
INC A	17	1/1	Increment A by 1	$(A) \leftarrow (A) + 1$	
DEC A	07	1/1	Decrement A by 1	$(A) \leftarrow (A) - 1$	
CLR A	27	1/1	Clear A to zero	$(A) \leftarrow 0$	
CPL A	37	1/1	One's complement A	$(A) \leftarrow \text{NOT}(A)$	
RL A	E7	1/1	Rotate A left	$(A_{n+1}) \leftarrow (A_n),$ $(A_0) \leftarrow (A_7)$	n = 0 to 6
RLC A ⁽²⁾	F7	1/1	Rotate A left through carry	$(A_{n+1}) \leftarrow (A_n),$ $(A_0) \leftarrow (C), (C) \leftarrow (A_7)$	n = 0 to 6
RR A	77	1/1	Rotate A right	$(A_n) \leftarrow (A_{n+1}),$ $(A_7) \leftarrow (A_0)$	n = 0 to 6
RRC A ⁽²⁾	67	1/1	Rotate A right through carry	$(A_n) \leftarrow (A_{n+1}),$ $(A_7) \leftarrow (C), (C) \leftarrow (A_0)$	n = 0 to 6
DA A ⁽²⁾	57	1/1	Decimal adjust A	$(A) \leftarrow (A) + 06H$ if AC = 1 or $(A_{0-3}) > 9$; $(A) \leftarrow (A) + 60H$ if $(A_{4-7}) > 9$	
SWAP A ⁽²⁾	47	1/1	Swap nibbles of A	$(A_{4-7}) \leftrightarrow (A_{0-3})$	

8-bit telecom microcontrollers

PCD33xxA Family

MNEMONIC	OPCODE (HEX)	BYTES/ CYCLES	DESCRIPTION	FUNCTION	NOTES
Data moves					
MOV A, Rr	F<8 + r>	1/1	Move register contents to A	(A)←(Rr)	r = 0 to 7
MOV A, @Rr	Fr	1/1	Move RAM data addressed by Rr, to A	(A)←((Rr))	r = 0, 1
MOV A, #data	23 data	2/2	Move immediate data to A	(A)←data	
MOV Rr, A	A<8 + r>	1/1	Move Accumulator contents to register	(Rr)←(A)	r = 0 to 7
MOV@Rr, A	Ar	1/1	Move Accumulator contents to RAM location addressed by Rr	((Rr))←(A)	r = 0, 1
MOV Rr, #data	B<8 + r> data	2/2	Move immediate data to Rr	(Rr)←data	r = 0 to 7
MOV @Rr, #data	Br data	2/2	Move immediate data to RAM location addressed by Rr	((R0))←data	r = 0, 1
XCH A, Rr	2<8 + r>	1/1	Exchange A contents with Rr	(A)↔(Rr)	r = 0 to 7
XCH A, @Rr	2r	1/1	Exchange Accumulator contents with RAM data addressed by Rr	(A)↔((Rr))	r = 0, 1
XCHD A, @Rr	3r	1/1	Exchange lower nibbles of A and RAM data addressed by Rr	(A ₀₋₃)↔((Rr ₀₋₃))	r = 0, 1
MOV A, PSW	C7	1/1	Move PSW contents to Accumulator	(A)←(PSW)	
MOV PSW, A ⁽³⁾	D7	1/1	Move Accumulator bit 3 to PSW ₃ (PS)	(PS)←(A ₃)	
MOV P A, @A	A3	1/2	Move indirectly addressed data in current page to A	(PC ₀₋₇)←(A), (A)←((PC))	
Carry flag					
CLR C ⁽²⁾	97	1/1	Clear carry bit	(C)←0	
CPL C ⁽²⁾	A7	1/1	Complement carry bit	(C)←NOT(C)	
Register					
INC Rr	1<8 + r>	1/1	Increment register by 1	(Rr)←(Rr) + 1	r = 0 to 7
INC @Rr	1r	1/1	Increment RAM data, addressed by Rr, by 1	((Rr))←((Rr)) + 1	r = 0, 1
DEC Rr	C<8 + r>	1/1	Decrement register by 1	(Rr)←(Rr) - 1	r = 0 to 7
DEC @Rr	Cr	1/1	Decrement RAM data addressed by Rr, by 1	((Rr))←((Rr)) - 1	r = 0, 1

8-bit telecom microcontrollers

PCD33xxA Family

MNEMONIC	OPCODE (HEX)	BYTES/ CYCLES	DESCRIPTION	FUNCTION	NOTES
Branch					
JMP addr	<2n>4 addr	2/2	Unconditional jump within a 2 kbyte bank	$(PC_{8-10}) \leftarrow n$ $(PC_{0-7}) \leftarrow \text{addr}$ $(PC_{11-12}) \leftarrow (MBFF0-1)$	$n = 0 \text{ to } 7$
JMPP @A	B3	1/2	Indirect jump within a page	$(PC_{0-7}) \leftarrow ((A))$	
DJZN Rr, addr	E<8 + r> addr	2/2	Decrement Rr by 1 and jump if not zero to addr	$(Rr) \leftarrow (Rr) - 1$; if (Rr) not zero, then $(PC_{0-7}) \leftarrow \text{addr}$	$r = 0 \text{ to } 7$
DJNZ @Rr, addr	Er	2/2	Decrement RAM data, addressed by Rr, by 1 and jump if not zero to addr	$((Rr)) \leftarrow ((Rr)) - 1$; if ((Rr)) not zero, then $(PC_{0-7}) \leftarrow \text{addr}$	$r = 0 \text{ to } 1$
JBb addr	<2b + 1> 2 addr	2/2	Jump to addr if Accumulator bit b = 1	If $(A_b) = 1$, then $(PC_{0-7}) \leftarrow \text{addr}$	$b = 0 \text{ to } 7$
JC addr	F6 addr	2/2	Jump to addr if C = 1	If $(C) = 1$, then $(PC_{0-7}) \leftarrow \text{addr}$	
JNC addr	E6 addr	2/2	Jump to addr if C = 0	If $(C) = 0$, then $(PC_{0-7}) \leftarrow \text{addr}$	
JZ addr	C6 addr	2/2	Jump to addr if A = 0	If $(A) = 0$, $(PC_{0-7}) \leftarrow \text{addr}$	
JNZ addr	96 addr	2/2	Jump to addr if A is NOT zero	If $(A) \neq 0$, then $(PC_{0-7}) \leftarrow \text{addr}$	
JT0 addr	36 addr	2/2	Jump to addr if T0 = 0	If $T0 = 0$, then $(PC_{0-7}) \leftarrow \text{addr}$	
JNT0 addr	26 addr	2/2	Jump to addr if T0 = 1	If $T0 = 1$, then $(PC_{0-7}) \leftarrow \text{addr}$	
JT1 addr	56 addr	2/2	Jump to addr if T1 = 1	If $T1 = 1$, then $(PC_{0-7}) \leftarrow \text{addr}$	
JNT1 addr	46 addr	2/2	Jump to addr if T1 = 0	If $T0 = 0$, then $(PC_{0-7}) \leftarrow \text{addr}$	
JTF addr ⁽⁴⁾	16 addr	2/2	Jump to addr if Timer Flag = 1	If $TF = 1$, then $(PC_{0-7}) \leftarrow \text{addr}$	
JNTF addr ⁽⁴⁾	06 addr	2/2	Jump to addr if Timer Flag = 0	If $T0 = 0$, then $(PC_{0-7}) \leftarrow \text{addr}$	

8-bit telecom microcontrollers

PCD33xxA Family

MNEMONIC	OPCODE (HEX)	BYTES/ CYCLES	DESCRIPTION	FUNCTION	NOTES
Timer/event counter					
MOV A, T	42	1/1	Move timer/event counter contents to A	(A) ← (T)	
MOV T, A	62	1/1	Move A contents to timer/event counter	(T) ← (A)	
STRT CNT	45	1/1	Start event counter		
STRT T	55	1/1	Start timer		
STOP TCNT	65	1/1	Stop timer/event counter		
EN TCNTI	25	1/1	Enable timer/event counter interrupt		
DIS TCNTI	35	1/1	Disable timer/event counter interrupt		
Control					
EN I	05	1/1	Enable external (chip enable) interrupt		
DIS I	15	1/1	Disable external (chip enable) interrupt		
SEL RB0 ⁽⁵⁾	C5	1/1	Select Register Bank 0	(RBS) ← 0	
SEL RB1 ⁽⁵⁾	D5	1/1	Select Register Bank 1	(RBS) ← 1	
SEL MB0 ⁽⁹⁾	E5	1/1	Select program memory bank 0	(MBFF0) ← 0, (MBFF1) ← 0	
SEL MB1 ⁽⁹⁾	F5	1/1	Select program memory bank 1	(MBFF0) ← 1, (MBFF1) ← 0	
SEL MB2 ⁽⁹⁾	A5	1/1	Select program memory bank 2	(MBFF0) ← 0, (MBFF1) ← 1	
SEL MB3 ⁽⁹⁾	B5	1/1	Select program memory bank 3	(MBFF0) ← 1, (MBFF1) ← 1	
STOP	22	1/1	Enter Stop mode		
IDLE	01	1/1	Enter Idle mode		
NOP	00	1/1	No operation		
Subroutine					
CALL addr ⁽⁶⁾	<2n + 1> 4addr	2/2	Jump to subroutine	((SP)) ← (PC) (PSW _{4,6,7}), (SP) ← (SP) + 1, (PC ₈₋₁₀) ← n, (PC ₀₋₇) ← addr, (PC ₁₁₋₁₂) ← (MBFF0-1)	n = 0 to 7
RET ⁽⁶⁾	83	1/2	Return from subroutine	(SP) ← (SP) - 1, (PC) ← ((SP))	
RETR ⁽⁶⁾	93	1/2	Return from interrupt and restore bits 4, 6 and 7 of PSW	(SP) ← (SP) - 1, (PSW _{4,6,7}) + (PC) ← ((SP))	

8-bit telecom microcontrollers

PCD33xxA Family

MNEMONIC	OPCODE (HEX)	BYTES/ CYCLES	DESCRIPTION	FUNCTION	NOTES
Parallel input/output					
IN A, P0	08	1/2	Input Port 0 data to Accumulator	(A)←(P0)	
IN A, P1	09	1/2	Input Port 1 data to Accumulator	(A)←(P1)	
IN A, P2 ⁽⁷⁾	0A	1/2	Input Port 2 data to Accumulator	(A)←(P2)	
OUTL P0, A	38	1/2	Output A data to Port 0	(P0)←(A)	
OUTL P1, A	39	1/2	Output A data to Port 1	(P1)←(A)	
OUTL P2, A	3A	1/2	Output A data to Port 2	(P2)←(A)	
ANL P0, #data	98 data	2/2	AND Port 0 data with immediate data	(P0)←(P0) AND data	
ANL P1, #data	99 data	2/2	AND Port 1 data with immediate data	(P1)←(P1) AND data	
ANL P2, #data	9A data	2/2	AND Port 2 data with immediate data	(P2)←(P2) AND data	
ORL P0, #data	88 data	2/2	OR Port 0 data with immediate data	(P0)←(P0) OR data	
ORL P1, #data	89 data	2/2	OR Port 1 data with immediate data	(P1)←(P1) OR data	
ORL P2, #data	8A data	2/2	OR Port 2 data with immediate data	(P2)←(P2) OR data	
Derivative input/output					
MOV A, Dx ⁽⁸⁾	8C direct	2/2	Move derivative register contents to A	(A)←(Dx)	x = 0 to 255
MOV Dx, A ⁽⁸⁾	8D direct	2/2	Move A contents to derivative register	(Dx)←(A)	x = 0 to 255
ANL Dx, A ⁽⁸⁾	8E direct	2/2	AND derivative register with A	(Dx)←(Dx) AND (A)	x = 0 to 255
ORL Dx, A ⁽⁸⁾	8F direct	2/2	OR derivative register with A	(Dx)←(Dx) OR (A)	x = 0 to 255
EN SI	85	1/1	Enable derivative interrupt		
DIS SI	95	1/1	Disable derivative interrupt		

Notes to Table 5

1. PSW CY, AC affected.
2. PSW CY affected.
3. PSW PS affected.
4. Execution of a JTF or JNTF instruction resets the Timer Flag (TF).
5. PSW RBS affected.
6. PSW SP₀, SP₁ and SP₂, affected.
7. (A) = 0000, P2.3, P2.2, P2.1 and P2.0.
8. For more information on the derivative I/O instructions of a particular microcontroller, consult the specific microcontroller data sheet.
9. SEL MB instructions may not be used within interrupt routines.

8-bit telecom microcontrollers

PCD33xxA Family

Table 6 Definitions of symbols used in Table 5

SYMBOL	DESCRIPTION
A	Accumulator
AC	auxiliary (half) carry
addr	program memory address
Bb	bit designation (b = 0 to 7)
CE/T0	CE/T0 input
CY	carry bit
Dx	mnemonic derivative register
data	8-bit number or expression
MB0	program memory bank 0
MB1	program memory bank 1
MB2	program memory bank 2
MB3	program memory bank 3
MBFF0	memory bank flip-flop 0
MBFF1	memory bank flip-flop 1
PC	Program Counter
PS	timer prescaler select
PSW	Program Status Word
RB0	Register Bank 0
RB1	Register Bank 1
RBS	Register Bank Select
Rr	register designation (r = 0 to 7)
SPn	Stack Pointer (n = 0, 1 or 2)
T	Timer 1
T1	T1 input
TF	Timer Flag
x	derivative register address (x = 0 to 255)
(X)	contents of X
((X))	contents of location addressed by X
←	is replaced by
↔	is exchanged with
#	immediate data prefix
@	indirect address prefix
*	hexadecimal; 8...F selects R0...R7
&	hexadecimal; 0, 2, 4, 6, 8, A, C, E selects page 0...7 in JMP, i.e. $(PC_{8-10}) \leftarrow \&_{1-3}$
%	hexadecimal; 1, 3, 5, 7, 9, B, D, F selects page 0...7 in CALL, i.e. $(PC_{8-10}) \leftarrow \&_{1-3}$ selects bit b = 0...7 in JBB, i.e. $b = \&_{1-3}$

8-bit telecom microcontrollers

PCD33xxA Family

7.1 Instruction map

	first hexadecimal character of opcode				second hexadecimal character of opcode											
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	IDLE		ADD A, #data	JMP page 0	EN I	JNTF addr	DEC A	0	1	2					
1	INC @Rr		JB0 addr	ADDC A, #data	CALL page 0	DIS I	JTF addr	INC A	0	1	2	3	4	5	6	7
2	XCH A, @Rr		STOP	MOV A, #data	JMP page 1	EN TCNTI	JNT0 addr	CLR A	0	1	2	3	4	5	6	7
3	XCHD A, @Rr		JB1 addr		CALL page 1	DIS TCNTI	JT0 addr	CPL A	0	1	2					
4	ORL A, @Rr		MOV A, T	ORL A, #data	JMP page 2	STRT CNT	JNT1 addr	SWAP A	0	1	2	3	4	5	6	7
5	ANL A, @Rr		JB2 addr	ANL A, #data	CALL page 2	STRT T	JT1 addr	DA A	0	1	2	3	4	5	6	7
6	ADD A, @Rr		MOV T, A		JMP page 3	STOP TCNT		RRC A	0	1	2	3	4	5	6	7
7	ADDC A, @Rr		JB3 addr		CALL page 3			RR A	0	1	2	3	4	5	6	7
8				RET	JMP page 4	EN SI			0	1	2			MOV A, Dx	MOV Dx, A	ANL Dx, A
9			JB4 addr	RETR	CALL page 4	DIS SI	JNZ addr	CLR C	0	1	2					
A	MOV @Rr, A			MOVP A, @A	JMP page 5	SEL MB2		CPL C	0	1	2	3	4	5	6	7
B	MOV @Rr, #data		JB5 addr	JMPP @A	CALL page 5	SEL MB3			0	1	2	3	4	5	6	7
C	DEC @Rr				JMP page 6	SEL RB0	JZ addr	MOV A, PSW	0	1	2	3	4	5	6	7
D	XRL A, @Rr		JB6 addr	XRL A, #data	CALL page 6	SEL RB1		MOV PSW, A	0	1	2	3	4	5	6	7
E	DJNZ @Rr, addr				JMP page 7	SEL MB0	JNC addr	RL A	0	1	2	3	4	5	6	7
F	MOV A, @Rr		JB7 addr		CALL page 7	SEL MB1	JC addr	RLC A	0	1	2	3	4	5	6	7

MGD566